

An Implementation Of Embedded Real Time System Framework In Service Oriented Architecture

Mahmood Aghajani

Department of Software Engineering
Faculty of Computer Science and Information Systems,
Universiti Teknologi Malaysia
Johor Bahru, Malaysia
email: ajstmahmood2@live.utm.my

Dayang N. A. Jawawi

Department of Software Engineering
Faculty of Computer Science and Information Systems,
Universiti Teknologi Malaysia
Johor Bahru, Malaysia
e-mail: dayang@utm.my

Abstract—Currently, the complexity of embedded software is increased, hence, more efficient design approaches are demanded. Although component based design is well-defined for developing Embedded Real Time (ERT) systems, the design and implementation of ERT component software is slow and complex. Distributed ERT systems can reduce the complexity of a component and increase its reliability and re-usability as well. Currently, Service Oriented Architecture (SOA) is an excellent technology for the implementation of distributed software. Some platforms are introduced to implement the components in SOA concept such as Service Component Architecture (SCA) and OSGI (Open Services Gateway Initiative). SCA provides a hierarchical component composition, distributed configurations and an interconnection with various means to design and combine services. However, SCA is unable to discover and reference services dynamically. In contrast, OSGI focuses on loading the service component. The services can be stopped, loaded and unloaded in frameworks supported by OSGI. Hence, this paper proposes an integration of SCA and OSGI to introduce a new framework for the implementation of distributed ERT systems.

Index Terms—Component Based Embedded Real Time System, Distributed, Open Services Gateway Initiative, Service Component Architecture.

I. INTRODUCTION

Currently, researchers focus on safety critical large scale ERT systems because of the enormous damage they can bring to human's life, environment and property associated with such events, like the Chernobyl nuclear power plant accident in 1986; the oil spills from the Exxon Valdez in 1989, the Erika in 1999, and the Prestige in 2002 [1].

A safety critical system provides multiple connections between the tasks of ERT systems which increase the dependency among them. In other words, the system becomes more complex.

Component Based Software Development (CBSD) manages the software complexity by using components which support the functionality needed by the system. It establishes significant benefits in ERT systems development such as re-usability which is provided by a library. The library component can be used to configure ERT software. Components provide evolutionary design which is used in complex ERT systems that require hardware and software upgrades [2]. It provides a very useful and promising design direction [3].

Hence, most of the existing component based ERT frameworks aim to achieve three goals: self-contained, platform-independent and real-time predictable, for resource constrained ERT systems in their reused framework. These three issues have a direct affect on the complexity of the ERT system.

However, applying component-based reuse to ERT systems poses a significant challenge due to the resource-constrained and real-time requirement properties of ERT systems, and the degree of diversity in AMR systems. Furthermore, there is no universally accepted agreement on what exactly constitutes a component [4]. Component developer cannot represent the integration and the operating condition of the components [5]. Recently, researchers introduced distributed software technology as a proper solution for the above problems [6].

Currently, distributed ERT systems are developed with a multitude of ecosystems in an ERT system. Service Oriented Embedded Systems (SOES) framework proposes a dynamic service composition to enable a flexible and dependable service oriented embedded systems. It reduces the complexity of evolving the software where services are developed dynamically [7]. However, there is no formal protocol for the interaction of components among several ERT applications.

SOA is an approach for developing the independent distributed computing which can implement the different components [8]. Currently, SOA is one of the most common architectures for increasing the adaptation component design in software. Reusing and adapting are provided based on loose coupling and location transparency concept which allows services to be autonomous and context independent [9] in SOA. It is one of the most promising architectural styles used on the web, or i distributed and dynamic systems. SOA creates the building distributed software architecture paradigm [10]. These SOA features can enable a distributed and excellent interaction among the components.

The integration of component based software and SOA can improve the freedom of the system and the implementation efficiency, reduce software development costs and improve the software quality [11]. However, SOA is an abstract concept. Hence, some software platforms are introduced to implement SOA concepts such as SCA and OSGI. These platforms support component composition strongly.

SCA can support a hierarchical component composition, distributed configurations and interconnection with various means. Thus, it is a proper platform to implement distributed ERT component composition.

On the other hand, Component Oriented Programming (COP) provides a useful graphical representation for component composition of Autonomous Mobile Robot (AMR) [12]. Consequently, the integration of the SCA and COP provides a distributed component base framework in AMR.

However, SCA is unable to modify the components dynamically. In contrast, OSGI can load and unload components dynamically. For that, a combination of the SCA and OSGI can support the distributed component technology dynamically. This paper provides a distributed ERT framework in SOA, integrating SCA and OSGI in the COP framework. Moreover, an AMR case study will present the features of this integration.

II. BACKGROUND

A. ERT FRAMEWORK BASED ON SOA ARCHITECTURE

A component based ERT framework is provided by using SCA, therefore, a component architecture design in SOA is considered for developing ERT component based framework according to [11]. Generally, the system architecture design of development program in SOA has a five layer structure; presentation layer (system-system, human-system interactions), web service layer (deals with various requests from users and forwards them to the business component layer); business component layer (it provides a variety of components for the system); XML processing layer (Responsible to save the heterogeneous data); data access layer (the final destination in which business object data is stored). Components are a loosely coupled collection in the business component layer, which is the core of the system. In addition, this layer includes business logic, calculations and validation of business objects [11].

Furthermore, an ERT system includes some criteria such as time and prediction of the temporal behaviour of unforeseen services. It is necessary to deal with these challenges to reduce the complexity of advanced ERT systems and support heterogeneous environments. Hence, a proper ERT framework must support dynamic service adaptation in a predictable manner [7].

B. COP

UML has a limitation in supporting hardware concepts such as concurrency, timeliness and properties of a functional ERT system. Moreover, a number of limitations are identified in modelling ERT environments by using UML such as temporal concerns, run-time constrained composition, constraints and type extension [7]. Hence, many ERT frameworks have been introduced to support ERT system constraints such as COP component based framework which provides a new component composition for supporting the above ERT features. COP Component composition is represented in [13]. It provides a graphical representation for the definition of the components and their composition.

There are three entities in COP model; components, ports and connectors. The computation and data are semantically well defined in components. A port is responsible to send and receive data among the related components. Connectors provide a relation between the ports. Details of the COP component composition are represented in [13]. Although COP component composition provides an static structure model for supporting the ERT framework in AMR, it is unable to support distributed ERT system.

Moreover, COP component consists of one or more structures and behaviours. It is also generalized with three different types of components, which are passive, active and event components. The component itself, besides having a subcomponent, is capable of refining. Passive and event components are associated with the system's behaviour.

C. SCA

The current motivation on distributed systems is the production of component configuration models that are aligned with business process models automatically. Currently, component compositions are supported by several systems such as SCA and Fractal. However, SCA can compose heterogeneous platforms that communicate with different protocols [14].

System architects apply the modularized and composed business functions in component assemblies which are provided by SCA. The final system deployment architecture is provided by SCA in an extended network of connected enterprises [10]. SCA is a combination of SOA and component-based software engineering (CBSE) which puts forward a set of specifications for building distributed applications [15].

SCA can support technology independence, hierarchical component composition, distributed configurations and interconnection with various means. Several platforms have been introduced which implement the SCA specification, such as Tuscany (tuscany.apache.org), and Newton (newton.codecauldron.org) [15]. SCA provides a component model for the implementation of the services in SOA. It supports components as interfaces (called services), require interfaces (called references) and expose properties. Reference and services are connected through wires.

SCA architecture has four main features: (1) Independence from programming languages, (2) Independence from interface definition languages, (3) Independence from communication protocols, (4) Independence from non-functional properties [15]. These features can be implemented with a broad range of solutions for SCA applications (e.g., SCA components programmed with EJB or OSGi).

SCA supports the re-usability of components which is described in [14]. The type descriptions of individual components are prepared by the developers, then proper individual components will be chosen by the final application.

These components can be built in a different platform, yet, in a single application, which represents a high level of distribution. Any type of application can be supported by SCA.

D. OSGI

OSGi is a proper framework for supporting a large number of service oriented applications such as embedded systems, soft real-time, consumer electronics, mobile and etc. Re-usability, replaceability and adaptation are high in OSGI [16]. The standard of creating applications using collaborative components was set by OSGI technology [17].

The functions of an OSGI service platform can modify the component composition on a variety of devices dynamically without restarting. Moreover, components of a component composition can be discovered by another collaboration based on OSGI technology. OSGI has been used in many ERT systems such as the winning Eclipse integrated development, automation and telecommunication and etc. Dependencies between components are defined by bundles in the OSGI nomenclature [18].

A Hybrid ERT framework was designed based on OSGI by [19]. This framework has two dimension; firstly it is focuses on supporting the real time requirement. At the same time, OSGI bundles wraps the real time components. It provides life cycle and component constrains management. Internal SOA can be used in other applications dynamically by bundles. It provides version control and life cycle management for realizing the non functional real time component.

XML parser service was provided to define and support the bundles which are responsible for defining the real time interfaces and constraints in XML. It enables a user to engage a powerful search for real time applications. Hence, components and their facilities can be shared by their existence in repositories.

III. EVALUATION

Table I represents a comparison between three technologies (COP, OSGI and SCA).

TABLE I
NON-LINEAR MODEL RESULTS

	SCA	OSGI	COP
Reuse and integration	+	+	-
Distributed apps	+	++	-
Heterogeneous apps	+	+	-
Runtime environment change	-	+	-
ERT timing	-	-	+

COP framework represents the ERT requirement of the presented framework such as Component type (active, passive, event), Period and Priority. In fact, COP meta model is responsible to provide ERT constrains such as period and priority. Although COP supports some ERT resource constrains such as service priority and period, but it is unable to provide distributed component composition based on SOA concepts. The disabilities of COP framework in supporting distribution are tabulated in Table I.

In contrast, SCA is unable to support ERT resource constraints in SCA model. Hence, we use the COP component features in SCA component compositions by adding the Period

and priority for each component. It is represented in the component layer of Fig. 1.

Although SCA provides a proper architecture which supports component reuseability according to Table I, the services can not be discovered and referenced dynamically by SCA. It uses the component composition to design and combine services. Since SCA can request other services through the references, it is still static [20]. While OSGI supports dynamic loading of service component through a set of service platform frameworks. Thus, we recommend the OSGI for the implementation of ERT web services.

IV. INTEGRATION OF THE SCA AND OSGI IN COP

There are two main issues in the implementation of the COP with SCA; integration of the user view of a COP component composition to a SCA based implementation, and the assembly management of the COP at execution time. This paper discusses about first issue. It contains mapping components, implementation of Inport/Outport as a set of services/references. Priority and period are provided by COP component composition. This mapping is shown in Fig. 1.

The component layer encapsulates a module based on different technologies, by using the SCA specifications that can provide service interface to other components. However, SCA is unable to communicate with other system directly, while OSGI is responsible for providing a proper communication platform. Hence, a web service layer is implemented by OSGI architecture to provide distributed services for Presentation Layer. In this paper, the OSGI host is recommended to manage the services which are created by SCA of COP framework. The SCA-OSGI-COP framework is shown in Fig. 1.

The OSGI layer manages the registration of the bundles that contain a couple of components. It can be used to design new applications.

When the Presentation layer requests the component, the related bundle is firstly chosen based on the COP component composition that needs a new component. Also, the features of new component parameter can be identified in component composition to help in finding the right component. If the component does not exist, the OSGI returns a "does not exist" message.

In [21], an architecture which can register the services and manage the distributed services was designed. Basic service, registry service and proxy service are three services which are introduced to increase the distribution in SOA.

V. META-MODEL OF COP-SCA

The quality of software development can be improved by defining a set of preconditions and postconditions that are associated with each operation during the development process. This definition is achieved by a proper design. So, we model the components and the interaction among them to identify the proper services for our component composition.

Implementing a distributed ERT framework for discovering more services motivates to create a new meta model based

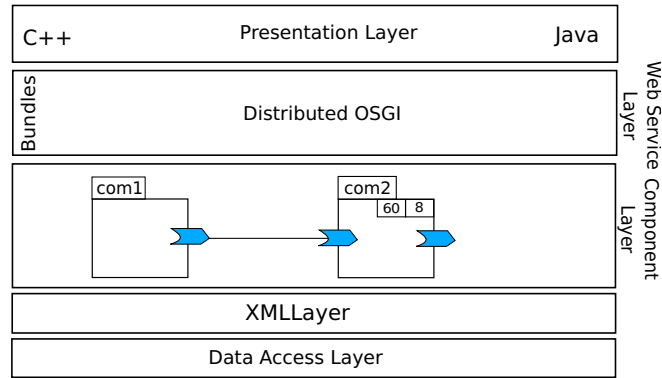


Fig. 1. ERT Framework in SOA

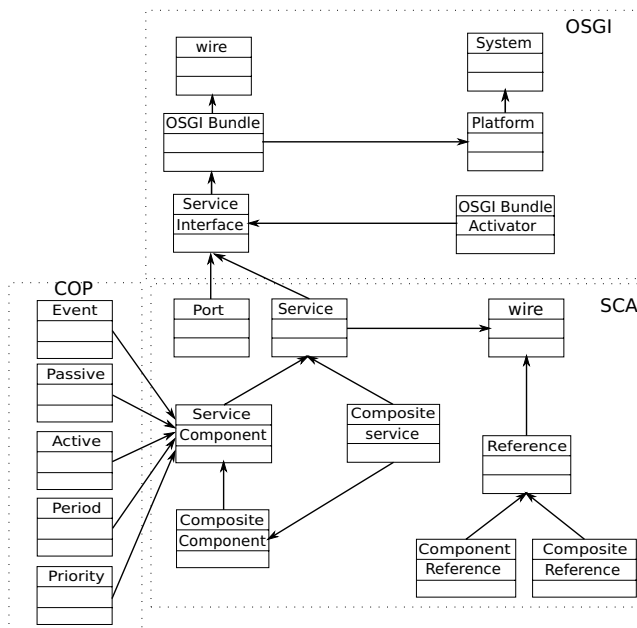


Fig. 2. Meta Model of COP-SCA-OSGI

on an SOA platform like OSGI and SCA. We use the meta-model, which is shown in Fig. 2, by adding the elements after analysing the COP component composition, SCA and OSGI.

Currently, there are some meta-models for the implementation of the SCA, OSGI and COP. Although SCA meta model has previously been introduced in [22], the latest proper SCA meta-model is introduced by [23]. In addition, [24] introduced a proper OSGI meta model because it provides four meta model definitions for the OSGI meta model which contains: Application, Implementation Oriented, Execution Platform, and Allocation meta-models. However, we focus on the Implementation Oriented Meta-model of OSGI because it provides 1) service provider and 2) bundle distribution concepts of OSGI. The COP Meta-model was only introduced in [25].

These three meta-model approaches are there to support the structure and the behaviour of the software framework. These three meta-models support component based framework, hence many elements are common among them such as components, ports, wires and etc. Moreover, they have some common features in supporting the behaviour, but they are not mentioned to stay within the boundaries of our scope.

ERT elements of our meta-model are adopted from the ones designed in [25], that contain: Events, Passive, Active, Priority and Period elements. Fig. 2 shows these ERT elements.

The meta-model of COP is shown in Fig. 2, done by [25]. The key elements of a component composition in SCA meta-model are designed by [23]. Syntax, process and assembly manners of service component such as Wire, which are defined in SCA meta-model, are considered in this study because of the many advantages described earlier.

An SCA component composition connects to another application by Contract, Interface and Protocol as in [23]. But in this research, these elements are replaced by OSGI bundle based on [24]. Our meta-model uses the four elements in [24] that contain: OSGI bundle, Platform, system and OSGI bundle activator.

The concept of interface is very important in component base framework because it is the only communication port between the components. It is divided into two types: (1) service type (represents the component which provides the service for other components), (2) Reference type (represents the components that require some services). Hence, OSGI can discover the appropriate services based on the reference type, or it can respond to the proper service by comparing the service request and the existing service type. A set of interfaces can be considered in a Platform. Also, a set of applicable interfaces is stored as an OSGI bundle to be used in other applications. The interface of component service is defined in the OSGI meta model.

VI. CASE STUDY- AMR

To show the benefits of integrating SCA and OSGI in COP framework of ERT systems, the deployment of the AMR

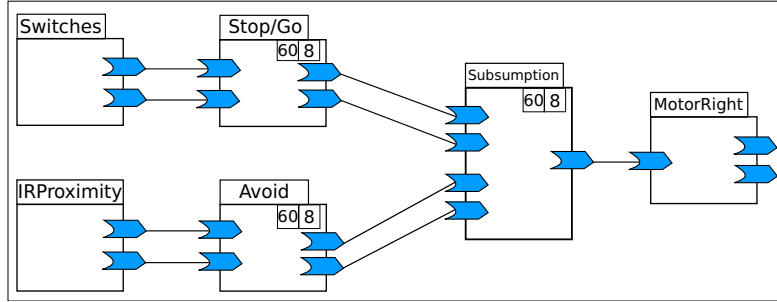


Fig. 3. Component Composition of AMR by using Component Layer of the SCA-COP

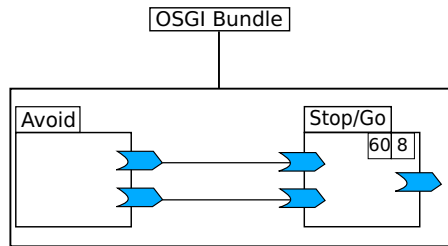


Fig. 4. An example of OSGI bundle

software analysis patterns is illustrated by using an AMR case study named IMR71848 robot.

The main requirement from ERT frameworks is to enable the AMR to navigate by controlling the speeds of the motors while avoiding obstacles existing around it. This subset of requirements involved the most critical components in the AMR system, which are: Sensors and Encoders to read the current speed of the robot based on Proportional-Integral (PI) control algorithm and behaviour based Control components. Fig. 5 shows the behaviour layer architecture of this study.

A. Component Layer

The components present in the component layer by using the SCA concept are shown in Fig. 3. They consist of three active components and three passive components. In this figure, components with period and priority fields are active components, while those without the mentioned fields are passive components. IRProximity and Switches components receive hardware status, then they forward the received data to the Avoid and Stop/GO components to provide a suitable decision to control the AMR's navigation process. Then the Subsumption component decides which service must be executed based on component priority and period. Finally, Motor control component controls the AMR.

Component composition of the component Layer is considered in a platform if all components are designed by a single designer. While if Components are provided by web services, they may be provided by several OSGI bundles. In the AMR case study, components are provided by [12]. Hence, the COP component composition is considered as a platform and it is

connected to an OSGI bundle.

B. Web Service Layer

OSGI can provide the AMR's case study with proper components by the following process:

OSGI which is enabled by the web service layer discovers some related AMR components based on the AMR's requirements. Components introduced by OSGI are connected to the component composition. In addition, a popular applicable component composition can be considered as a new bundle. For example, if several robots are using the same components such as switches and StopGo, a new independent OSGI bundle should be considered to provide these components for other robot applications. Fig. 4 shows an example of the independent OSGI bundle which can be requested from other ERT applications.

C. XML Layer

Components and their relation can be stored in the XML layer. For example, Avoid component is represented as follows:

```
<Component>
<Identification>001</Identification>
<Name>Avoid</Name>
<Period>60</Period> <priority>9</priority>
< Ports>
< ReferenceType Id=IP00 Minvalue=0 MaxValue=10>
< ReferenceType > < SevriceType Id=OP00 Minvalue=0
MaxValue=0 Value=1>
< SevriceType > < /Ports>
</Component>
```

In addition, interfaces of the component which are represented by the OSGI bundle are stored in the XML layer. A bundle should contain at least a Referencetype or serviceType, and as follows:

```
< bundle RequestNumber=12>
<Identification>101</Identification >
< Ports >
< ReferenceType value=001 > < SevriceType value=002 >
< /Ports >
```

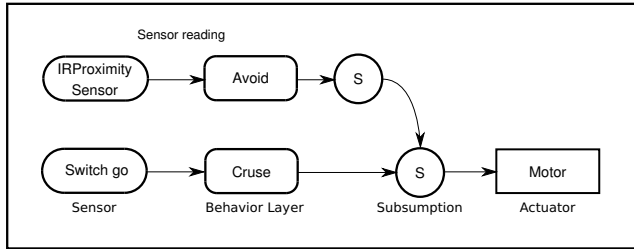


Fig. 5. An example of OSGI bundle

</bundle >

VII. CONCLUSION

In this paper, the current COP framework has been extended to a distributed ERT framework by integrating SCA and OSGI platforms into the COP framework. It is proposed to manage the complexity of integration and composition in an ERT system. To support the distributed COP framework, two platforms were combined in the COP framework based on SOA features. The first is SCA, which designs and combines services by using the component composition. Then OSGI, which manages the loading and unloading of the services dynamically.

Furthermore, the modified COP framework provides reusable components which can be useful in designing new software components. It is provided by the OSGI bundles. The key concept of our approach is providing the global platform with an ability to share a couple of components via web services. This indicates that the use of SOA technologies enable component engineering products to be created directly from other projects.

In addition, features of the distributed COP framework are presented in an AMR case study based on the architecture layers proposed for the SOA. For example, integrated SCA-COP component composition presented in the component layer.

As for the future work, some techniques can be investigated to evaluate the provided services based on the ERT constraints that exist in the COP framework. Moreover, a tool will be implemented to support the presentation and data layers of the extended COP framework. Thus, the interaction of the user and the COP framework will be provided by a COP tool.

VIII. ACKNOWLEDGEMENTS

Special thanks to the Universiti Teknologi Malaysia(UTM) grant under GUP vot 00H60 for financing and funding support and also to our Embedded Real-Time Software Engineering Laboratory (EReTSEL) members for their continuous support.

REFERENCES

[1] E. Coskun and M. Grabowski, "Software complexity and its impacts in embedded intelligent real-time systems," *Journal of Systems and Software*, vol. 78, pp. 128–145, 2005.

[2] D. Nyström, J. Hansson, and C. Norström, "Aspects and Components in Real-Time System Development : Towards Reconfigurable and Reusable Software," *Development*, no. February, pp. 1–16, 2004.

[3] E. De Freitas, M. Wehrmeister, and C. Pereira, "Using aspects and component concepts to improve reuse of software for embedded systems product lines," *Components*, 2008.

[4] G. Buchgeher and R. Weinreich, "Tool Support for Component-Based Software Architectures," *2009 16th Asia-Pacific Software Engineering Conference*, pp. 127–134, Dec. 2009.

[5] D. Lee and A. S. Chul, "A hierarchical fault tolerant architecture for component-based service robots," *Industrial Informatics (INDIN)*, pp. 487–492, 2010.

[6] J. Hill, H. Sutherland, P. Stodinger, T. Silveria, D. C. Schmidt, J. Slaby, and N. Visnevski, "OASIS: A Service-Oriented Architecture for Dynamic Instrumentation of Enterprise Distributed Real-Time and Embedded Systems," *2010 13th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*, pp. 10–17, 2010.

[7] S. Brennan, S. Fritsch, Y. Liu, A. Sterritt, J. Fox, E. Linehan, C. Driver, R. Meier, V. Cahill, W. Harrison, and Others, "A framework for flexible and dependable service-oriented embedded systems," *Architecting dependable systems VII*, pp. 123–145, 2010.

[8] J. Wiklander, J. Eliasson, A. Kruglyak, P. Lindgren, and J. Nordlander, "Enabling Component-Based Design for Embedded Real-Time Software," *Journal of Computers*, vol. 4, no. 12, pp. 1309–1321, Dec. 2009.

[9] N. Ali, R. Nellipaiappan, R. Chandran, and M. A. Babar, "Model driven support for the Service Oriented Architecture modeling language," *Proceedings of the 2nd International Workshop on Principles of Engineering Service-Oriented Systems - PESOS '10*, p. 8, 2010.

[10] K. Dahman, F. Charoy, and C. Godart, "Generation of Component Based Architecture from Business Processes: Model Driven Engineering for SOA," *2010 Eighth IEEE European Conference on Web Services*, pp. 155–162, Dec. 2010.

[11] D. Yue, "Based on SOA architecture and component software reuse architecture research," *2010 2nd IEEE International Conference on Information Management and Engineering*, pp. 517–520, 2010.

[12] D. N. A. Jawawi, R. Mamat, and S. Deris, "A Component-Oriented Programming for Embedded Mobile Robot Software," *Advanced Robotics*, vol. 4, no. 3, pp. 371–380, 2007.

[13] D. Norhayati, A. Jawawi, and R. Mamat, "A Component-Oriented Programming Framework for Developing Embedded Mobile Robot Software using PECOS Model," in *The Second Malaysian Software Engineering Conference (MySEC'06)*, 2006.

[14] P. Hnetyinka, L. Murphy, and J. Murphy, "Comparing the Service Component Architecture and Fractal Component Model," *The Computer Journal*, May 2010.

[15] L. Seinturier, P. Merle, D. Fournier, N. Dolet, V. Schiavoni, and J.-B. Stefani, "Reconfigurable SCA Applications with the FraSCaTi Platform," *2009 IEEE International Conference on Services Computing*, pp. 268–275, 2009.

[16] J. L. A. H. J. C. D. n. L. U. Cauca, and C. Popayán, "Evolvability Characterization in the Context of SOA," *Quality*, pp. 242–253, 2010.

[17] C. Hang, "Research and application of distributed OSGi for cloud computing," *Design*, 2010.

[18] J. M. Marquez, J. Jimenez, and I. Agudo, "Secure Real-Time Integration of Services in a OSGi Distributed Environment," *2008 Fourth International Conference on Networked Computing and Advanced Information Management*, pp. 631–635, Sep. 2008.

[19] N. Gui, V. De Florio, H. Sun, and C. Blondia, "A hybrid real-time component model for reconfigurable embedded systems," *Proceedings of the 2008 ACM symposium on Applied computing - SAC '08*, p. 1590, 2008.

[20] W. Li, Y. Zhang, and J. Jin, "Research of the Service Design Approach Based on SCA_OSGi," in *Services Science, Management and Engineering, 2009. SSME'09. IITA International Conference on*. IEEE, 2009, pp. 392–395.

[21] Y. Wang, M. Song, and J. Song, "An extended distributed OSGI architecture for implementation of SOA," *Architecture*, 2010.

[22] C. Parra, X. Blanc, A. Cleve, and L. Duchien, "Unifying design and runtime software adaptation using aspect models," *Science of Computer Programming*, vol. 76, no. 12, pp. 1247–1260, Dec. 2011.

[23] D. Du, J. Liu, and H. Cao, "A Rigorous Model of Contract-Based Service Component Architecture," *2008 International Conference on Computer Science and Software Engineering*, pp. 409–412, 2008.

- [24] J. Cano, N. M. Madrid, and R. Seepold, "OSGi services design process using model driven architecture," *2009 IEEE/ACS International Conference on Computer Systems and Applications*, no. Itea 04006, pp. 791–794, 2009.
- [25] S. Sabil and D. N. A. Jawawi, "Integration of PECOS into MARMOT for Embedded Real Time Software Component-Based Development," *2009 Fourth International Conference on Software Engineering Advances*, pp. 265–270, Sep. 2009.