A Service Component-Oriented Design and Development Methodology for Developing SOA-based Applications

Soumia Bendekkoum, Mahmoud Boufaida
LIRE Laboratory
Mentouri University of Constantine
Constantine, Algeria
{soumia_bendekkoum,mboufaida}@umc.edu.dz

Lionel Seinturier
LIFL Laboratory & INRIA Lille
University Lille 1, Villeneuve d'Ascq
Lille, France
Lionel.Seinturier@univ-lille1.fr

Abstract—**The Service-Oriented Architecture (SOA) is a promising technology based standard for easily developing distributed, interoperable and loosely coupled applications. The emergence of the Service Component Architecture (SCA) standard, which uses service components, has more and more facilitated the development and deployment of SOA independently from technologies and standards. However, SOA does not define a complete and clear methodology for developing service based systems. It does not address the issue of the way these services could be defined despite the fact that there exist successful specification tools (UML profiles, Service Component) for modeling service-based applications. This paper presents the Service Component-oriented Design and Development Methodology (SCDD-Methodology), which combines software engineering approach and service component models to specify and indentify adequate services. It discusses the key principles in its design: the adoption of service component model for the development of SOA-based applications for well defining the structure of the application behind the service's layer. The paper presents a case study of a commercial company producing machine tools (MTP).**

Keywords-SOA; SCA; UML Profiles Modeling approaches; Service Component.

I. INTRODUCTION

SOA is an architectural style for the reorganization and redeployment of the information system [1]. It encapsulates the functions of an application into a set of loosely coupled services. These services are defined with a contract, and published using an interface description, so that they can be invoked by remote clients.

SOA is not only a technology or a recipe. It is a way of thinking and structuring distributed information systems. It requires appropriate modeling tools and good methodologies for the design, development and management of distributed applications conforming to its principals of autonomy, reuse, interoperability and loose coupling of its different elements. Despite the success of SOA, it remains a partial solution [2], [3], because it describes how application's functionalities are structured into autonomic and distributed entities (services) as well as how they are published and used on the web. But it does not define what is behind the scene and how these services are structured [4]. The recently developed Service Component Architecture (SCA) is proposed to fulfill that shortcoming with a set of specifications, which supports a view of service as software component. This means that behind the service layer there exists a set of components named "Service components" implementing service behavior. Several platforms have already been developed that implement SCA specification, such as Tuscany [5], Newton [6] and FraSCAti [7] for java-based SCA applications.

SOA development and implementation methods are primarily different in their details, but they all have the same principal, which is according to Heubès [8] in the same spirit as the engineering business processes or information systems. Zimmermann et al. [9] and Papazoglou and Heuvel [10] reveal that there exist three methods for developing an SOA. The first one is the *Top-down* in which the business logic of existing processes is used to identify services. In contrast, *Bottom*-up approach starts with the analysis of applications to determine the existing functions of the information system, and from these artifacts, it is possible to identify the functions that are eligible to the level of service. Finally, the hybrid approach called also *Meet in the Middle* approach advocates to conduct in the same time a top-down and bottom up methods. Therefore, the main steps of an SOA development approach can be summarized as follows [11]: (1) business processes are represented, (2) analyzed, (3) improved and (4) built on the top of the existing legacy applications.

Unfortunately, although that there exist successful specification tools for realizing a service-oriented development project [12], these specification approaches do not rely on an effective development methodology [3], especially for complex service-oriented architecture projects, where we need enormously. Our studies on existing SOA development projects reveal that these approaches do not provide a comprehensive, clear and precise strategy to achieve a successful SOA development project. In consequence, the developers are usually facing a major difficulty, which is the definition of the concept of service. This problem causes to the companies an important question which is what a service is, and what level of granularity can it takes to properly define the relevant services to a business based integration project.

In this paper, we present the Service Component-oriented Design and Development Methodology (SCDD-Methodology), which combines software engineering approach and service component models to specify and construct as well as structured SOA-based applications.

This paper is organized as follows. Section 2 presents the SCA specification standard used in the SCDD methodology, and discusses its key principals for achieving a service oriented development project. Section 3 describes the service component design and development methodology. Section 4 presents a case study of a commercial company producing machine tools and describes some implementation aspects. Section 5 discusses related work. Finally, Section 6 concludes the paper and outlines some future works.

II. SCA MODELING APPROACH AND SOA CHALLENGES

The SCA is a set of specifications defining component model for building Service-based applications using SOA and component-based software engineering (CBSE) principals.

SCA entities are software components, which may provide interfaces (called services), require interfaces (called references) and expose properties. References and services are connected through wires (Figure 1.).

These models allow us to offer specification tools, which facilitate the modeling and implementation of architectures using services as well as a flexible layer for application interobperability. So that, these models have self-configuration capacities that permit to cope with the continuous changes of both the environment and client needs.

Service component models have important properties that motivate our choice of research. We summarize those which are related to our work, as follows [14]:

- Hierarchy: the SCA model is hierachical with components being implemented by primitive language entities or by subcomponent
- Autonomy: The service component architecture model is equipped with the autonomy feature, which is useful to define the functions of components that implement business services inside or outside of the grid services of the model architecture.
- Reconfiguration: service component models are equipped with an important property called self-reconfiguration or dynamic reconfiguration [15], which can be useful to ensure effortlessly the adaptability of SOA business services. For example, it is conceivable that any change in the definition of a requested service can be easily mapped in a reconfiguration of Fractal component model [16]. This is due to the interfaces (internal and external) and also the property of the plug and play between components of the hierarchical model.

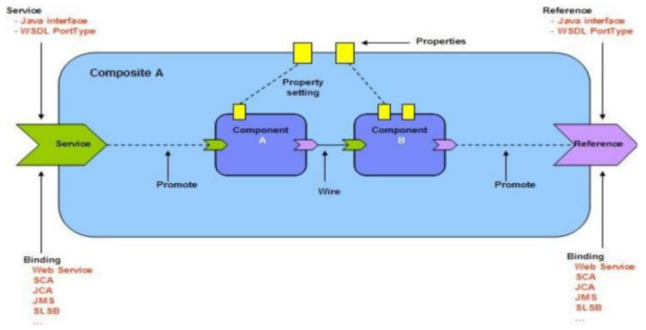

Figure 1. Exemple of SCA Component Architecutre [13].

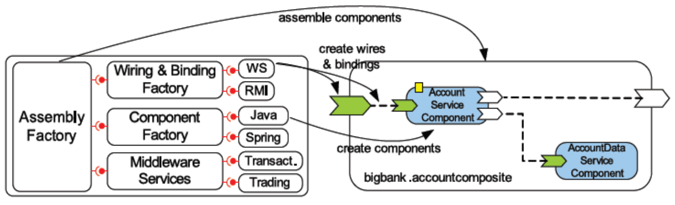

Figure 2. FraSCAti Platform Architecture [13].

SCA is based on the idea that business function is provided as a series of services, which are assembled together to create solutions that serve a particular business need. These composite applications can contain both new services created specifically for the application and also business function from existing systems and applications, reused as part of the composition. SCA provides a model both for the composition of services and for the creation of service components, including the reuse of existing application function within SCA compositions [17], and this motivates more our choice of using component model.

SCA aims to encompass a wide range of technologies for service components and for the access methods, which are used to connect them. For components, this includes not only different programming languages, but also frameworks and environments commonly used with those languages. For access methods, SCA compositions allow for the use of various communication and service access technologies that are in common use, including, for example [13], Web services, messaging systems and Remote Procedure Call (RPC).

We present, as an example, the SCA FraSCAti platform, which we are chosing for the deployment of the service component-based applications. The platform has four main components, as shown in Figure 2: *Component Factory*, *Wiring and binding factory*, *Middleware services,and Assembly factory* [13].

In the next section, we detail the various phases of the proposed approach, each of which is illustrated with an exemple.

III. PHASES OF THE SCDD METHODOLOGY

The objective of the service component-oriented design and development methodology is to achieve service integration and to facilitate service interoperability as well as service composition. In our approach we focus on the SCA concepts (autonomy and composition) for achieving our objective to create an SOA-based system, which responds more to the client requirements and to well structure the business logic of its functionalities. The SCDD methodology provides appropriate principals and guidelines to specify, construct and customize well defined services (fine grained functions) and business processes (coarse grained functions) orchestrated from fine grained services (service composition).

As shown in Figure 3, the SCDD-Methodology constitutes four main phases: (i) Modeling phase: Modeling Business Needs and Modeling Existing Applications, (ii) Service Components Identification, (iii) Service Components deployment, and finally, (iv) Service Identification and Publication phase. In the following section, we detail separately each phase:

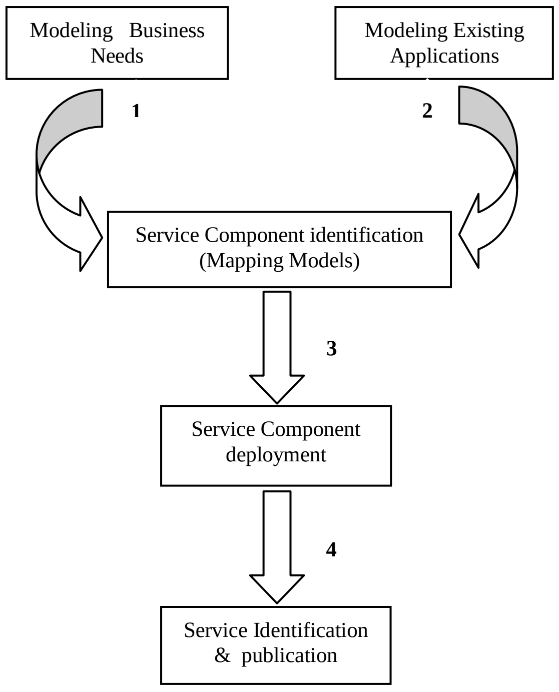
Modeling Business Needs
Modeling Existing Applications
1
2
Service Component identification (Mapping Models)
3
Service Component deployment
4
Service Identification & publication

Figure 3. Principal Phases of SCDD Methodology.

A. Modeling Phase:

The first phase of SCDD methodology consists of analyzing and modeling the requirements of the new system. This includes studying and modeling existing applications. This phase contains two steps, the first one is Business Needs Modeling for analyzing and modeling clients' requirements, and the second is Existing Applications Modeling for analyzing and modeling applications of the enterprise legacy system:

1) *Business Needs Modeling:* objectives are business goals which custumors want to achieve, or target behaviors which customers want to see in the system. It is important when setting objectives to make sure they are in line with overall business needs of the target information system, this is the principal objective of our approach: the allignement of the business services of the futur system to the client business needs. To do so, we propose, in this first step, to analyze the business goals of the future information system, and model these goals in an Objective Tree Model.

Figure 4 represents an UML metamodel defines the structure of an Objective Tree Model. This model represents a hierarchical structure specification of all needs of the future SOA-based system. Each node of the tree represents an objective and its sub-nodes represent sub-objectives realizing the overall objective. Constraints are functional conditions, which must be supported by an objective to achieve an other objective, these constraints are very important in the objective tree model, to identify to the designer and, in the future, to the developer any conditions might make echeiving the objective more diffecult or even in some cases impossible.

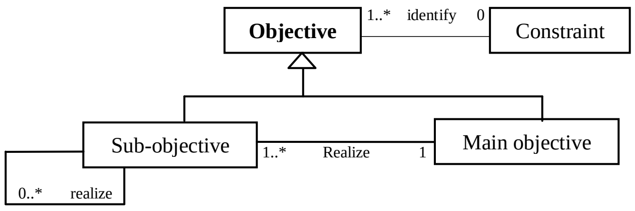

Figure 4. UML Meta-model of Objective Tree Model

The tree model obtained, in this step, is crucial to the alignment of the business process of the target SOA-based system to the client and environment business needs. This model will be used in the follwing phase for analyzing and defining the components that permit to implement the adequate functions that best respond to the company's needs.

2) *Existing Applications Modeling:* In SOA-based systems business processes are modelled in different granularities, these processes incorporate functions (services) supported by the existing enterprise legacy system, as well as by new functions not supported by the existing system, which must be developed. In this step, we identify adequat services in the right granularity. To do so, our approach proposes the use of Service Components Models, to specify existing legacy systems. These models represent an hierarchical abstraction of the sructure interne of all applications deployed within the existing system.

The specification of the legacy system is based on the granularity of applications, it is represented as a functional tree. The root of the tree represents the main application (function), the nodes represent the sub-functions that implement the main function, the sub-nodes represent the sub-functions that implement functions in higher level of granularity, and the leaves of the tree represent the primitive instructions that implement functions in higher hierarchical level on the tree. Figure 5, represents an UML metamodel defines the hierarchical structure of a *Component-based Function Tree Model*. In this metamodel, we presents the hierarchy of different functions constituting an application, and the data flow (output and input data) between functions.

B. Service Components Identification

The target SOA-based system must align its business processes with environement business goals, in order to improve its ability to respond rapidly to marcket forces.

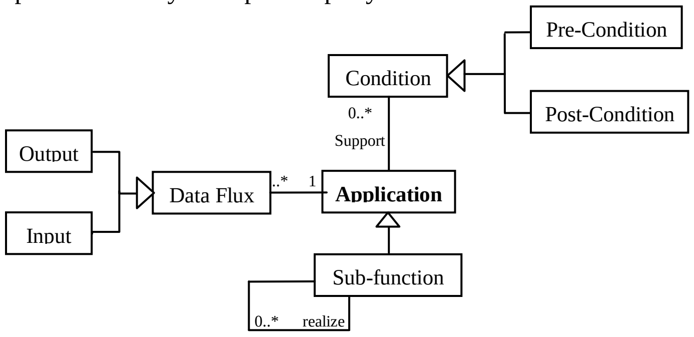

Figure 5. UML Meta-model of Component-based Function Tree Model

The first question in an SOA-based system devlopment project is how to identify business services, which responds to cunstomers' needs. This question incorporates, firstly, the problem of extraction of the relevant services from existing (legacy) system, secondly, the problem of definition of new services, which respond dynamically to the environement needs. SCDD approach for developing an SOA that we propose is a code-based approach, since the business rules that implement the core of the target enterprise system can be analyzed from the enterprise application models. In this phase, we identify the parts of condidat code that implement the business logic of future Web services. This phase consists of the *Extraction of service components*; here we extract the code for future services according to business needs of the enterprise, in order to aligning the code of existing legacy applications to the business needs discussed in the previous phases. We propose to determine which of those existing features can be exposed as Web services, to apply a method of *mapping models*. This method aims to associate the nodes of the Component-based Functional Tree model of an application components to the nodes of the model of a business process model of basic objectives.

The activities of the phase detection code that implements the web service based on the projection that we propose are summarized as follows:

- The first task is the selection of an activity of a basic business processes, which is afterwards associated to the component that contains functions performing this activity, by analyzing the final result specified for each function.
- The second task is to check if an instruction affects the final result returned by the activity of selected processes. Beginning with primitive instructions of the selected component, and rising to the nodes.
- If the selected component returns some variables required to the activity, it is necessary to select the node that is at a higher level in the model.
- else, if the variables returned are equivalent to the selected activity, the function is defined as the future public service that implements the activity of selected elementary process.
- These activities are applied to each node model targets, in order to best meet the business needs of the company.

C. *Service Components Deployment*

This step attempts encapsulate parts of candidate web services identified in the previous phase, in service components, and to defining the required interfaces (references) and interfaces provided (services) for the execution of each component or each service.

D. *Service Identification and Publication*

The final step is the technical wrapping of the functions defined and encapsulated in Service component in the preceding phases. This comprises the definition of Web Service Description Language interfaces (WSDL) that specify the Service Components interfaces. We summarize the different activities of this phase as follows:

- Identification of Service component interfaces provided for them published in the form of web services, and data requirements for their invocation.
- Definition the functionality of web services WSDL. Each Web service exposes an interface that defines the message types and patterns of trade. To do this we must first specify a well-defined interface for each service detected. Next, we turn to the definition of the functionality of web services WSDL.
- Publication services. Finally, the resulting web services must be registered in the UDDI, for use by other customers or other service-oriented systems, integrating them into a business process.

IV. Case Study and Some Implementation Aspects

In order to evaluate the SCDD approach, we take as a case study example the enterprise Production Machine Tools (PMT). PMT is a commercial enterprise, which produces and markets machine tools. For over 30 years, this enterprise uses to manage its service a monolithic inflexible information system implemented in *Basic* (Beginner's All-purpose Symbolic Instruction Code) programming language. Year by year enterprise's activities grow and the number of its clients increase, however the information system capacities remains incapable to follow this evolution while the majority of its activities are done using traditional methods through phone calls, Excel files, etc. To overcome this problem the company decides to promote its information system in order to support the new requirements as well as standards and technologies. As we know, the development of a new system costs and takes more time; so, the business directors and information system engineers decide to develop the new system on the top of the existing legacy system.

We present in this section the application of the different phases of SCDD methodology phases and some implementation details.

A. *Modeling phase*

In this phase, we analyze and model clients' business needs and existing applications in order to identify the structure of the future Service-based system. In our approach, objective and functional tree models help us to see all this laid out clearly. Our example reveals *a treatment of a client order* as a main objective.

Figure 6 shows just the beginning of an Objective Tree Model that illustrates the hierarchy of customer needs. The tree model starts with the overall objective

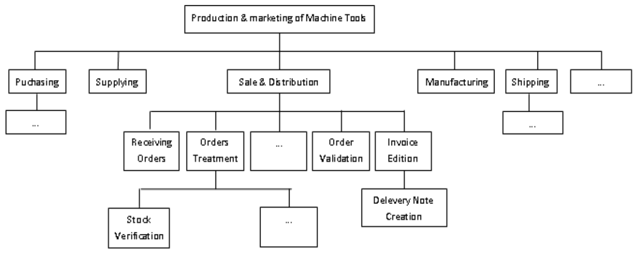

Figure 6. Goals hierarchical Tree model of the case study.

“Poduction and Marketing of Machine Tools”. The branches coming from this are sub-objectives, which are small abjectives that we need to achieve the main objective as: “*Purchasing/supplying, Manufacturing, Sale and Distribution, shipping, ... etc.*” which are necessary after to specify the business logic of the company. In reality, this tree would have many more branches and levels detailing all the sub-objectives necessary for realizing the main objective, as well as for covering all customer’s separate objectives.

B. *Service components creation*

After applying the first phase of SCDD methodology on the case study, and the extraction of the component implementing future services, we obtain finally a diagram of the hierarchy of Service Components Composing the target SOA-based application. In this section, we are describing how these components can be implemented and deployed in a distributed environment.

Figure 7 shows an example of a service component model resulting after the mapping models phase, where the *Order Client Service Component* represent a composite, which contains six sub-component.

We are choosing the JAVA programming environment ‘Eclipse’ for implementing the functionalities of the resulting component. We used the 3.5.1 version integrated already with the platform FraSCAti specification, where we affect for each component an interface; we give more detail on interface implementation in the next section.

C. *Interfaces definition and implementation*

In order to define relations between service components, different interfaces affected to each component must be developed. In this phase we create ADL descrition file in which we define the binding between different component using interfaces defined. For each component we specify the relationship with an other component it must be a relation where the component require a service from an other component as well as a relatio where the component profides an service to an other service.

V. Related Work

In this section, we present some related works, and compare the SCDD methodology with these approaches in terms of service identification strategies, SOA specification techniques as well as service deployment and programming facility.

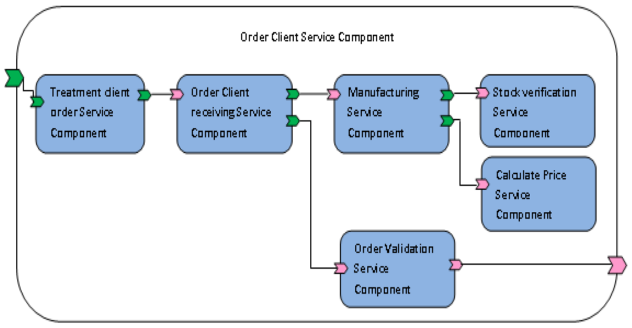
Order Client Service Component
Treatment client order Service Component
Order Client receiving Service Component
Manufacturing Service Component
Stock verification Service Component
Calculate Price Service Component
Order Validation Service Component

Figure 7. Order Client Service Component Schema.

Service identification strategies; Several approaches of SOA development are available that use different strategies to identify services, either Buttom-up ones: Chang and Kim [18], Top-down ones: Emig and al. [17], and Erl [19], or Hybrid ones: SOMA (Service Oriented Modeling and Architecture) [20], SOAD (Service-Oriented Analysis and Design) [18], CSOMA (Contextual Service Oriented Modelling and Analysis) [21], also the projects of Papazoglou and Heuvel [10]. Comparing these approaches with the SCDD approach that we present in this paper, we use in our methodology a model based strategy for identifying services from the point of view of both the producers and requesters of service, so our approach align more the business process of the target application with the client and environement needs.

Specification techniques; The existing approaches that we discussed in the second section propose to use existing modeling techniques and present a set of models or meta-models supports. For example, the approach of SOAD and SOMA that use the business oriented models (BPM: Business Process Modeling) and the object oriented models (OOAD: Object Oriented Analysis and Design) for the development of SOA, and the approach of Papazoglou and Heuvel [10], which uses the development based component (CBD: Component Based Development).

In the proposed approach, we use a business oriented service component development models in order to highlight the advantages its properties of composition and autonomy, to identify services in the right level of granularity, also reconfiguration property of component that permit to services to project easly different changes on components implementing them. The SCDD methodology view service as component entities, which can romotly eccessed independently as possible form the underlying implementation technologies.

Deployment and programming; The Service component-oriented methodology facilitates the deployment and realization of distributed service-oriented applications. The SCDD methodology, compared with existing design and development methodologies which achieve a complex architecture and not deployable in different platforms, uses service component-based models that define a well structured service-oriented application independent from technologies.

VI. Conclusion and Future Research

In this paper, we have presented the SCDD methodology considering the enterprise’s requirements as well as the existing legacy systems. SCA is a standard for distributed SOA-based systems. We motivated and described how the behavior that implements legacy system business can be aligned to enterprise needs, and modeled as service components to be used first to identify single relevant services in right granularity, and second to develop and integrate easily these services in business processes that implement the target SOA-based system.

We focused mainly in this approach on the problem of the definition of the concept *service*. Thus, we proposed in the first stage of our approach to use objective tree to model the hierarchical nature of the requirements, or business needs for designing the system. The service components model to implement and deploy the elements of the SOA considering its characteristics of reuse, autonomy and distribution. The use of this specification allows technology using service to benefit from distribution, autonomy and composition properties of service component.

Although SCDD is not the first approach that combines software engineering approach and service component models, its objective is very important, as it permits to treat the problem of SOA development starting from the modeling phase in order to describe how services constituting SOA based-application are structured. Generally, the Enterprise Service Bus (ESB) is used to implement an SOA-based system [22]. Our future research has to establish links between characteristics of SOA enterprise models based on service components and their realization in the ESB. In other words, although there exist works that permit to publicize service component as web service [23], they are still technical solutions. So, we must reveal how to realize an SOA based on service component in the ESB. This comprises the problem of monitoring and controlling, as well as the problem of adaptability and security requirements specified in enterprise models and realized in the SOA.

REFERENCES

[1] F. Tonic, M. Boulier, B. Paroissin, J. Clune, F. Bernnard, and M. Gardette, “SOA : votre nouvelle architecture,” le magazine du développement : Programmez !, N° 78, June 2006.

[2] J. Zhao, M. Tanniru, and L. Zhang, “Service Computing as the Foundation of Enterprise Agility: Overview of Recent Advances and Introduction of Special Issue,” Proc. Inf Syst. LNCS, Springer Press, March 2007, vol. 9, pp. 1-8, doi : 10.1007/s10796-007-9023-x.

[3] P.M. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, “Service Oriented Computing: a research roadmap,” International Journal of Cooperative Information systems, vol. 17, June 2008, pp. 223-255, doi: 10.1142/S0218843008001816.

[4] L. Seinturier, Ph. Merle, R. Rouvoy, D. Romero, V. Schiavoni and J. B. Stefani, “A component-based middleware platform for reconfigurable service-oriented architectures,” International Journal Software Practice & Experience, vol. 42, May 2012, pp. 559-583 , doi: 10.1002/spe.1077.

[5] tuscany.apache.org. visited: 08.04.2013.

[6] newton.codecauldron.org . visited: 14.04.2013.

[7] frascati.ow2.org. visited: 08.04.2013.

[8] Ch. Heubès, “Mise en œuvre d’une SOA : Les clés de succès,” Xebia France, Février 2008, http://blog.xebia.fr/2007/08/16/mise-en-oeuvre-dune-soa-les-cles-du-succes/, visited : 10.04.2013.

[9] O. Zimmermann, P. Krogdahl, and C. Gee, “Elements of Service-Oriented Analysis and Design,” ftp://ftp.software.ibm.com/software/webservices/SOADpaperdWv1.pdf. visited: 15.04.2013.

[10] M. Papazoglou and P. W., Heuvel, “Service-oriented design and development methodology,” International Journal of Web Engineering and Technology (IJWET 06), vol. 2, July 2006, pp. 412-442, doi: 10.1504/IJWET.2006.010423.

[11] O. Zimmermann, N. Schlimm, G. Waller, and M. Pestel, “Analysis and Design Techniques for Service-Oriented Development and Integration,” http://ozimmer.de/download/INF05-ServiceModelingv11.pdf. visited: 15.4.2013.

[12] A. Kenzi, “Ingénierie des Systèmes Orientés Services Adaptables: Une Approche Dirigée par les Modèles,’’ PhD thesis, Ecole Normale supèrieure d’Informatique et d’Analyse de Systèmes, Université Mohamed V, Rabat. Octobre 2010.

[13] L. Seinturier, Ph. Merle, D. Fournier and N. Dolet, “Reconfigurable SCA Applications with FraSCAti Platform,” Proc. IEEE International Conference on service Computing (SCC 09), IEEE Press, June 2009, pp. 268-275, doi: 10.1109/SCC.2009.27.

[14] F. Baude, D. Caromel, C. Dalmasso, M. Danelutto, V. Getov, L. Henrio, and C. Pérez, “GCM: A grid extension to Fractal for autonomous distributed components,” Annals of Telecommunications, vol. 64, September 2009, pp. 5-24.

[15] A. Solange, A. Ludovic, B. Tomàs, C. Antonio, M. Eric, and S. Emil, “Specifying Fractal and GCM Components With UML,” Proc. IEEE International Conference of the Chilean Computer Science Society (CCS 07), IEEE Press, 2007, pp. 53-62, doi: 10.1109/SCCC.2007.17.

[16] L. Seinturier, “Le modèle de composants Fractal”, http://www.lifl.fr/~seinturi/middleware /fractal.pdf, 2008.

[17] C. Emig and S. Abeck, “Development of SOA-Based Software System and Evolutionary Programing Approach,” Proc. The Advanced International Conference On Telecommunications and on International Conference on Internet and Web Applications and Services (AICT/ICIW 06), 2006, pp. 182-187.

[18] S. H. Chang and S. D. Kim, “A Service-Oriented Analysis and Desing Approach to Dveloping Adaptable Services,” Proc. IEEE International Conference on Services Computing (SCC 07), IEEE Press, July 2007, pp. 204-211, doi: 10.1109/SCC.2007.16.

[19] T. Erl, “ Service-Oriented Architecture (SOA): Concepts, Technology, and Design,” Prentice Hall, 2005, ISBN: 0131858580.

[20] A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Gariapathy, K. Holley, “SOMA: Service-Oriented Modeling and Architecture: How to identify, specify and realize services for your Service Oriented Architecture (SOA),” IBM Systems Journal, vol. 47, July 2008, pp. 377-396, doi: 10.1147/sj.473.0377.

[21] K. Boukadi, “Coopération inter-entreprises à la demande : Une approche flexible à base de services adaptables,” Thèse de doctorat de l’Ecole supérieure des Mines de Seint-Etienne, Novembre 2009.

[22] H. Christophe, “Exposer ses composants Fractal en Web service dans Petals ESB #2,” http://chamerling.org/2010/06/08/exposer-ses-composants-fractal-en-webservice-dans-petals-esb-2/, visited : 11.04.2013.

[23] Z. El Hafiane, M. Ghaoui, S. Harmach, A. Maalej, and N.M.Tourè, “Composants Fractal et Web Services,” http://deptinfo.unice.fr/twiki/pub/Minfo05/DepotDesRapports/Rapport-TER-9.pdf, visited: 15.04.2013.