Expert Systems with Applications 41 (2014) 1189–1205

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Intelligent business processes composition based on multi-agent systems

José A. García Coria, José A. Castellanos-Garzón *, Juan M. Corchado

Department of Computers and Automation, University of Salamanca, Plaza de la Merced s/n, 37008 Salamanca, Spain

ARTICLE INFO

Keywords:
Multi-agent system
Web service
Cloud computing
Software component reuse
Software engineering
Business process composition
Machine learning

ABSTRACT

This paper proposes a novel model for automatic construction of business processes called IPCASCI (*Intelligent business Processes Composition based on multi-Agent systems, Semantics and Cloud Integration*). The software development industry requires agile construction of new products able to adapt to the emerging needs of a changing market. In this context, we present a method of software component reuse as a model (or methodology), which facilitates the semi-automatic reuse of *web* services on a *cloud computing* environment, leading to business process composition. The proposal is based on *web* service technology, including: (i) Automatic discovery of *web* services; (ii) Semantics description of *web* services; (iii) Automatic composition of existing *web* services to generate new ones; (iv) Automatic invocation of *web* services. As a result of this proposal, we have presented its implementation (as a tool) on a real case study. The evaluation of the case study and its results are proof of the reliability of IPCASCI.

1. Introduction

The software development industry is constantly evolving and looking for new technologies, languages and tools that are increasingly powerful, efficient and safe. In this process, it is essential to build models, architectures and agile technologies able to introduce new tools as simply and economically as possible. Component reuse is actually one of the techniques that more clearly contribute to such a development by providing efficient mechanisms to create quality software (Lemley & O'Brien, 1997; Poulin, 1997, 2006; Rehesaar, 2011; Schmid, 2011; Shang, Mohan, Lang, & Vragov, 2012). Reuse increases software reliability (because it uses tested software components), development productivity and implies a clear cost reduction (Garcia et al., 2007; Rehesaar, 2011; Xu, Singh, & Deshpande, 2011). Since the recent increasing of the volume and complexity of software products, reuse has been a field taken into great consideration, being a fundamental stage in design and quality models as CMMI (Capability Maturity Model Integration) (Osiecki, Phillips, & Scibilia, 2011).

In this context, *web* service reuse appears as an interesting alternative with respect to code classical reuse. Indeed, a *web* service is a software component representing a service deployed on a *web* platform and supporting automatic interactions between machines on a computer network (Walsh, 2002). In this framework, arises SOA (Service-Oriented Architecture) as a new architecture leading to conventional software development (Alizadeh, Seyyedi, & Mohsenzadeh, 2012; Erl, 2009; Ralyté, Mirbel, & Deneckère, 2011). SOA introduces a new method to create distributed applications where their basic services can be discovered, published, and linked in order to achieve more complex services. Applications interact through the existing services from entry points on an interface rather than at the level of implementation (Papazoglou, 2008).

Software development can be analyzed from different perspectives, offering us a wide variety of alternatives to a methodological, functional and instrumental level, among others. One of the paradigms revolutionizing this industry in recent years has been *cloud computing* (Antonopoulos, Anjum, & Gillam, 2012; Cloud Computing., 2011; Duy, Sato, & Inoguchi, 2011; EuroCloud, 2011; Stage & Setzer, 2009). The *cloud* represents a novel concept of service and information distribution, providing many possibilities of scaling solutions, facilitating the use of relatively simple and economic terminals (interesting models of pay per use, and multi-platform accessibility, etc.). However, this model has certain drawbacks related to maintenance complexity and application development (Dölitzscher, Reich, & Sulistio, 2010; Jaeger, Lin, Grimes, & Simmons, 2009). Moreover, the number of experienced engineers in this field is relatively low and the development time is significantly high (Massonet et al., 2011; Schaaf, Koschel, Grivas, & Astrova, 2010; Sulistio, Reich, & Dölitzscher, 2009).

Starting from all the above, this research proposes a methodology (IPCASCI, *Intelligent business Processes Composition based on multi-Agent systems, Semantics and Cloud Integration*) that facilitates the business process construction on *cloud computing* environments

* Corresponding author. Tel.: +34 923294451.
E-mail addresses: jalberto@usal.es (J.A. García Coria), jantonio@usal.es (J.A. Castellanos-Garzón), corchado@usal.es (J.M. Corchado).

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.08.003

in an agile and efficient way from developed components. It deals with the development of a proposal that facilitates the creation of such processes in form of *web* services from other semi-automatic functional services. All this is carried out on the framework of a computer system guided by relatively inexpert programmers. The process of business process construction is guided by a multi-agent architecture based on virtual organizations (Dignum, Vazquez-Salceda, & Dignum, 2005; Escriva et al., 2006; Rodríguez, de Paz, Bajo, & Corchado, 2011), which is able to implement the intelligent behavior needed for process management by using *ontology* (Chandrasekaran & Josephson, 1999; Guarino, 1998; López, 1999; Maedche & Staab, 2001; Noy & McGuinness, 2001). The goal of this research is then to provide a model allowing the automatic construction of a business process from specifications in text format (with some constraints) of the process concerning us. The multi-agent system based on virtual organizations will facilitate the process composition by using standard BPEL (Business Process Execution Language). This standard allows the automatic composition of *web* services in an easy way by adding the advantage of a direct projection to a diagram BPMN (Business Process Management Notation) (Pedrinaci et al., 2008).

To reach the goals proposed in this research, the remainder of the present paper has been structured into the following sections: Section 2 outlines information and works related to this research. Section 3 deals with an overview of the different components coupled to our proposal to make the *web* service composition, that is, architecture IPCASCI. This section gives a general idea of the whole composition process from IPCASCI. Section 4 presents *cloud computing* and *web* service technology, which have been included in our proposal. On one hand, we describe the advantages of developing IPCASCI on a *cloud* environment and provide two *cloud* functionalities to support its performance. On the other hand, we introduce *web* services technology and explain the inclusion of semantics to the *web* services through *ontology* to later facilitate the process of discovery and composition from a multi-agent system. Section 5 describes the agent-based virtual organizations defined in the multi-agent system of IPCASCI to make the discovery of the *web* services fit the specifications of the user to compose them and obtain the solution *web* service. Section 6 develops a real case study to give an implementation of IPCASCI, which is evaluated through software metrics. Conclusions, Appendix A with information supporting our proposal and the references of this research have been given at the end of this paper.

2. Related work

As previously explained, we have introduced an architecture (IPCASCI) aimed at composing *web* services to obtain new ones in an automatic way by starting from requirements given by the user. This process is carried out in such a way that the service automatic construction be efficient and has an intelligent behavior. Intelligent behavior is intended as the ability of processing of the input requirements introduced by the user, that is, the system be able to: (i) analyze the input, (ii) find the *web* services allowing to hold the requirements, and (iii) carry out an automatic composition of *web* services for their corresponding business processes. Thus, as a final result, new *web* services implementing the provided requirements will be achieved in an automatic process of discovery and composition. In this sense, there exist several approximations of architectures based on semantic *web* services. In Su, Chen, Zhu, and Zeng (2012), a solution focused on *fuzzy logic* to discover semantic *web* services has been proposed. In Sycara, Paolucci, Ankolekar, and Srinivasan (2003), a solution based on agents and ontological language DAML-S has been proposed, and in García, Ruiz, and Ruiz-Cortés (2012), a proposal based on queries SPARQL and ontological language OWL-S has also been used. In general terms, most of the proposals on architectures of semantic *web* services are based on language OWL-S. The proposal given in this paper differs from the existing ones in the following:

- Design of a global platform, which is embedded in a *cloud* environment and whose structure has been provided for an agile and efficient running.
- The semantics information of the *web* services does not depend on the internal structure of the ontology, which allows reusing ontologies regardless of their formats.
- We have included a *multi-agent system* based on virtual organizations that facilitates requirement analysis of the user and the discovery process of *web* services.
- The solution *web* service has been built from a simple definition given through a set of specifications introduced by the client.

3. Architecture IPCASCI for business process composition

This section describes the different components integrated to our proposal IPCASCI, which allow carrying out the automatic composition of *web* services. So Fig. 1 shows a conceptual diagram of the components integrating the proposed architecture. This figure describes the following items:

1. *Cloud system:* Platform of cloud computing on which is supported our proposal. The platform provides an environment for running and storage.
2. *Web services:* The *web* services existing in the architecture that will be used by the process of *web* service composition.
3. *UDDI register:* Universal register system where used *web* services are registered. This allows us an open access to the *web* services of the architecture. This way, the *web* services implemented on the architecture may be reused regardless of our proposal.
4. *Multi-agent system based on virtual organization:* This system supports functionalities to make the discovery and the composition of *web* services:
    a. *Analysis system:* Analyze the semantic content introduced by the user in order to structure it in computable items.
    b. *Search system:* Discover the web services that hold the semantics as syntactic constraints given by the user.
    c. *Composition system:* Once determined the web services and their relationships holding the user semantic requirements, a service composition is applied on such web services to obtain the new web service.
5. *Ontologies:* Distinct ontologies modeling the semantic knowledge that can be included in the *web* services.
6. *BPEL file:* Store the composition of the *web* services that meet the requirements indicated by the user to obtain the desired solution.

As a general schema of working, IPCASCI accepts the requirements of the user through the assistant software to raise the processes of discovery and composition. The steps followed by IPCASCI have been given the following algorithm and represented in Fig. 2:

3.1. Algorithm for web service discovery and composition (AWSDC)

1. The user introduces the requirements of the *web* service to build by means of an assisted system. Such a system allows users to define a list of terms (in form of modules) representing the requirements of the new *web* service. The list of terms is bounded by the *web* service repository in the *cloud* system.