Engineering Applications of Artificial Intelligence 24 (2011) 1355–1367

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Steps, techniques, and technologies for the development of intelligent applications based on Semantic Web Services: A case study in e-learning systems

Heitor Barros, Alan Silva, Evandro Costa, Ig Ibert Bittencourt *, Olavo Holanda, Leandro Sales

Instituto de Computação – Federal University of Alagoas (UFAL), Campus A. C. Simões, S/N, Maceió, Alagoas, Brazil

ARTICLE INFO

Available online 30 June 2011

Keywords:
Semantic Web Services
Intelligent Tutoring System
Grinv Middleware
Ontology

ABSTRACT

Semantic Web Services domain has gained special attention in academia and industry. It has been adopted as a promise to enable automation of all aspects of Web Services provision and uses, such as service creation, selection, discovery, composition, and invocation. However, the development of intelligent systems based on Semantic Web Services (SWS) is still a complex and time-consuming task, mainly with respect to the choice and integration of technologies. In this paper, we discuss some empirical issues associated with the development process for such systems and propose a systematic way for building intelligent applications based on SWS by providing the development process with steps, techniques and technologies. In addition, one experiment concerning the implementation of a real e-learning system using the proposed approach is described. The evaluation results from this experiment showed that our approach has been effective and relevant in terms of improvements in the development process of intelligent applications based on SWS.

1. Introduction

Semantic Web provides an environment where software agents can navigate through Web documents and perform sophisticated tasks. In this context, we can observe a great demand for Semantic Web-based applications (Berners-Lee et al., 2001) due to its robustness on providing rich data description mechanisms, such as ontologies (Gruber, 1993). In this manner, researchers and industry use Semantic web inside their web applications to express information accurately. As a consequence, software agents become able to process, share, reuse and understand the terms being described and take better decisions according to the information processed by Semantic Web tools, for example, Semantic search engines (Devedzic, 2004).

Semantic Web requires explicit declaration of knowledge by using ontologies to make information understandable to computer in the Web. In some applications, it is also necessary to provide services that machines and intelligent agents can understand, select, compose and invoke automatically. This is possible through the use of Semantic Web Services (McIlraith et al., 2001), the focus of this work.

Semantic Web Services promise the combination of Semantic Web and Web Service technology. It inherits characteristics from both approaches: semantic interoperability from Semantic Web, and dynamics of resources availability from the Web Services technology (Daconta et al., 2003).

Although Semantic Web Services have emerged as a good candidate for Intelligent Systems development, it adds another one level of complexity in terms of systems development process. It usually increases processing time and can become a very expensive process in terms of choosing and integrating all the roles involved in this process, such as tools, developers and protocols (Srinivasan, 2006). This occurs because the development process that is composed by a series of complex steps, such as: (i) Selection and development of services in order to meet the functional requirements of the system; (ii) Implementation of semantic annotations for these services to describe them, while this step requires the complete specification of the system domain using ontologies in order to describe the services; (iii) Selection of a service repository to store both services and their respective semantic annotations; (iv) Selection of techniques and mechanisms to perform automatic discovery and composition of services in accordance to the needs of the system. This step also includes development of proprietary solutions; (v) Integration of technologies chosen in the previous steps in order to provide an ecosystem with the processes of discovery, composition and invocation of services.

In this context, the SWS community has proposed a series of documents to support developers on building Semantic Web

* Corresponding author. Tel.: +55 82 32141401; fax: +55 82 32141400.
E-mail addresses: heitorbarros@ic.ufal.br (H. Barros), alanpedro@dsc.ufcg.edu.br (A. Silva), evandro@ic.ufal.br (E. Costa), ig.ibert@ic.ufal.br (I.I. Bittencourt), olavoholanda@ic.ufal.br (O. Holanda), leandro@ic.ufal.br (L. Sales).

0952-1976/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2011.05.007

Services based systems, but there is not any proposal to integrate all the main tools required by developers to facilitate their software development process. In this manner, Section 2 presents some of these proposals.

In order to reduce the main problems of integrating related tools for the development of Semantic Web Services based system, this work proposes a framework for managing and integrating the processes involved in the use of Semantic Web Services into intelligent applications. This work introduces the steps and techniques required during the development of Semantic Web Services systems. The evaluation methodology here adopted addressed an e-learning system as a real world example of application aiming at explaining the use of all steps and techniques applied in the process of developing Semantic Web Services systems suggested and discussed along this work. Based on results, we showed that our proposal has been effective and relevant once we presented the improvements reached on the development of intelligent applications.

2. Development problems and current solutions

This section discusses some of the main issues regarding the challenges faced by developers in the process of building Semantic Web Services based systems. For each problem, we discuss several solution proposals by describing their benefits and limitations.

The research on building SWS based systems indicates that problems related to maintenance and performance of its tools and services make it difficult to adopt this kind of application. These problems are discussed below:

- *Unguided development*: The authors of the work available in Brambilla et al. (2007) propose a Model-Driven Design (MDD) to develop SWS applications. They use WebML (Brambilla et al., 2007) to specify the Process Models of the system, and provide a methodology to generate Semantic Web Services according to these Process Models by using the WSMO (Roman et al., 2006) Language. However, this work does not present techniques for developing mechanisms to manipulate these services (repositories, discovery, composition and invocation). As discussed by the authors of the work available in Weise et al. (2008), the use of different discovery and composition algorithms may influence on system performance and on the results obtained in the process of discovery and composition. Thus, the choice of these mechanisms should also be considered in the development of the application, where a wrong choice of the mentioned set of algorithms may lead to worse development process or system performance.
- *Using Static Planning approaching is not always possible*: Despite allowing the execution of composed services with high level of complexity, the use of discovery through the Static Planning approach is not feasible for all systems. This is the case of the METEOR-S (Aggarwal et al., 2004) project. In high dynamicity systems, the processes constantly change, making the Static Planning approach unfeasible for this kind of system. This problem occurs in works such as the one available in Song et al. (2004), that presents a new proposal for dynamic discovery of services in pervasive environments. Due to the environment dynamics, it cannot be defined by Static Planning processes.
- *Performance on loading ontologies*: Ontologies are available on the Web, this means that, first, applications have to load the required ontologies and only after process them in order to use or compose Web Services. The process of loading ontologies in almost all cases becomes a bottleneck, cause negative impact of the system performance. A possible solution for this case is to store the ontologies in a local database. This approach may avoid the loading ontology problem, like it is accomplished by Broekstra et al. (2001). However, this solution requires a mechanism to handle ontologies synchronization between the original one and the copy stored in the local database. In this case, it adds a point of failure for the system and increase the consumption of system resource, although it avoids systems working with out-of-date ontologies.
- *Integration of discovery and invocation mechanisms*: The process of service discovery ensure that a given system works dynamically, where invocation and access of service repository should be performed automatically without requiring users or developers direct interactions with the system to link these repositories. In this case, the system should provide mechanism for integrating this process. In the existing approaches, such as the one provided by METEOR-S, these mechanisms do not allow users perform changes on the system features, once features are provided through SWS.
- *Fault tolerance*: Systems based on SWS may suffer problems related to errors in the execution of a service or in handling an ontology due to access problems on the network. This problem can be observed in the related work available in Erl (2005). This occurs because SWS are based on distributed services available on the Web. In this way, these systems must provide mechanisms for fault tolerance to avoid the system to stop performing its tasks. If a faulty service is detected, the system should use discovery mechanisms to find a service that is similar to the faulty service without requiring user intervention. Another problem related to fault tolerance is that faulty services may become zombie, remaining available for the system indefinitely, causing continuous and unexpected system errors. In this case, the system should not allow the use of the faulty service in the processes of discovery and invocation until the problem in the faulty service is fixed. This means that the system should not discard this service permanently, as it can be used in the future.

3. Our proposal

In this section we present our proposal to alleviate the existent problems on developing Semantic Web Services based systems, as we discussed in the previous section. Initially, a set of steps is provided in order to guide the system developer during the development process, presenting the characteristics of each step and the tools that must be analyzed in each of them. Thereafter, we present the Grinv, a Middleware for integrating the service discovery and invocation mechanisms. The Grinv allows developers to adapt the system to accommodate their needs and therefore avoiding some problems discussed in the previous section.

3.1. Steps on development of systems based on Semantic Web Services

Fig. 1 illustrates the process related to the development of systems based on SWS. The steps of this process are presented as following:

- *Domain specification*: the domain is formally specified through ontologies. These ontologies have the purpose to specify the concepts and relationship in the system.

  In this context, developers should use a methodology to develop these ontologies (Ontology Engineering). Community has presented some approaches to define methodologies for