Future Generation Computer Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Aligning ontology-based development with service oriented systems

Jun Shen [a,*], Ghassan Beydoun [a], Graham Low [b], Lijuan Wang [a]

[a] School of Information Systems and Technology, University of Wollongong, NSW, Australia
[b] Australian School of Business, The University of New South Wales, Sydney, NSW, Australia

HIGHLIGHTS

- We proposed an ontology-based development approach for service oriented system.
- We proposed a three layer abstraction of ontology alignment.
- We proposed a semantic integration life cycle for semantic Web services.
- We applied our approach in the building QoS ontology for service and cloud systems.
- We developed multi-agent and peer-to-peer based service system to align ontologies.

ARTICLE INFO

Article history:
Received 20 February 2013
Received in revised form
4 July 2013
Accepted 2 August 2013
Available online xxxx

Keywords:
Software development life cycle
Ontologies
Agents
Multi-agent systems
Peer-to-peer systems
Service oriented systems

ABSTRACT

This paper argues for placing ontologies at the centre of the software development life cycle for distributed component-based systems and, in particular, for service-oriented systems. It presents an ontology-based development process which relies on three levels of abstraction using ontologies: architecture layer, application layer and domain layer. The paper discusses the key roles of ontologies with respect to the various abstraction layers and their corresponding impact on the concomitant workproducts. In addition, a peer-to-peer-based service selecting and composing tool is suggested as a way of supporting the process. The paper presents the architecture of the proposed tool and illustrates the whole process in the development of a mobile banking application based on dynamic Web services.

1. Introduction

A service oriented architecture (SOA) promotes loose coupling between components to enable faster and more flexible reconfiguration of business processes and provides a means of organising system resources in an open and flexible arrangement. From an enterprise management perspective, the successful delivery of SOA systems translates into responsive business processes that can adjust to varying customer service requirements [1,2]. Expected responsiveness and adjustment is based on leveraging the knowledge of relationships between various services and mixing and matching groups of services to satisfy new requirements. Service oriented computing (SOC) has become a convincing paradigm for enterprises in tackling traditional hurdles to improving the efficiency and effectiveness of their ubiquitous software applications [3]. The vice President of Gartner, J. Fenn, predicted that by 2013, SOA will have delivered transformational results to the role and capabilities of IT for businesses [4]. Furthermore in 2012, one third of IT budgets were spent gaining access to services developed by other vendors [5]. This increased attraction to SOC comes with the expectation that energies from software development and acquisition will be shifted to other business activities, and at the same time deliver better alignment with business requirements.

Service-oriented software engineering promises to deliver enormous tangible improvements to enable business processes. In practice it has been hard to realise, especially for complex application systems, when it is advantageous to use services from various providers. There is a need to find ways to collect available business services from the different providers and to specify how such a collection of services should be combined and integrated seamlessly [6]. This ability to easily integrate services can increase flexibility and agility, not only in systems development but also in business process management. However, existing service components do not provide a clear and comprehensive definition of the business process semantics. Therefore many existing services are often isolated and opaque to information system developers. Current software techniques and tools do not alleviate this and place

* Corresponding author. Tel.: +61 2 42213873; fax: +61 2 42214045.
E-mail addresses: jshen@uow.edu.au (J. Shen), beydoun@uow.edu.au (G. Beydoun), g.low@unsw.edu.au (G. Low), lw840@uowmail.edu.au (L. Wang).

0167-739X/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.future.2013.08.005

too much burden on developers attempting to reuse existing services. This hampers the realisation of the monetary benefits of the technology and the collective adoption of reuse by the required large number of players to ensure a critical mass of shared services [7].

The use of ontologies can pave the way for an intelligent software development environment where developers submit new business requirements and an automatic tool generates the service oriented software system. With semantic driven composition, services will be shared between teams of developers and across multiple organisations connected via the Internet. We are acutely aware that existing Web languages are not easily accessible as described in Berners-Lee [8]. To overcome this, we aim to provide ontological support to the requirements analysis phase to ensure that any newly created service can be appropriately indexed by a semantically rich layer of ontologies. There are a number of promises and arguments around semantic Web services [9–11], one big issue being the limited generality of the isolated efforts made by independent research groups, for example, as reported in [12]. Indeed, this paper heeds visionary comments in [13,14] that, in order to meet the requirements of pervasiveness and autonomy in services development, the maturity and awareness of semantic Web and ontology technologies (key components of Web 3.0) should not be overlooked. This research uses an innovative ontology- and agent-based technology to support the SOA environment by providing automatic identification and merging of services which will in turn enable process changes in the business requirements.

The holistic semantic Web driven SOA development approach will play a significant role in better exploitation of services at both the technical and business level. This paper is well timed coinciding with the rapid development of Web 2.0 technologies which enable a single enterprise to use the Web to offer value-added customer services via connection to various services or applications from other public or commercial organisations. The innovations in this paper will harness the Web as a support medium for systems development enabling automatic exchange of services between various development teams across multiple organisations. The significance of these innovations is more pronounced coinciding with challenges of dynamic composition of distributed services [15] created by the high demand for portable appliances such as personal digital assistants (PDAs) and next-generation mobile devices as well as for P2P applications e.g. [16,17]. It is challenging to apply a hierarchy of ontologies in developing such applications. For example, concerns remain about how to measure the quality of the ontologies and the alignments among them [18].

This paper proposes an intelligent and supportive software development environment where developers can focus on semantic enrichment of business requirements and proper alignment with business processes rather than the time consuming task of service identification and integration. This paper's contributions are: a holistic ontology based development approach for service oriented systems such Grid and Cloud platforms; a three layer abstraction of ontology alignment; and the introduction of a semantic integration life cycle for semantic Web services. Our proposed approaches were demonstrated by applying them in the building of quality of service (QoS) ontology for a service oriented system and developing a multi-agent, peer-to-peer (P2P) based service system to align ontologies before tools and illustrating the process in the development of a mobile banking application based on dynamic Web services.

The rest of the paper is organised as follows: Section 2 provides an argument for placing ontologies at the heart of Software development life cycle (SDLC) of component-based systems generally and service oriented development in particular. Section 3 articulates the requirements for an ontology-based service oriented development and the existing supporting ontologies from the literature. Section 4 discusses the architecture of the intelligent development environment which uses a MAS for peer-based Web service composition system. A case study highlighting the ontology-based development of bank loan approval service oriented system is presented in Section 5. Section 6 concludes with a summary and discussion of future work.

2. Ontology-centric service orientation

It is often a complex task for developers to locate the appropriate service components to customise and integrate into their system. To reduce this cost and to automate much of the service selection and composition effort, we advocate an ontology-based approach that uses a semantically enriched representation of services and business requirements in order to enhance interoperability of services. A domain ontology can facilitate reuse of services undertaken across different areas (or industries). For example, the services for certain accounting practices may vary but only slightly across application areas. Such practices, if well documented using a domain ontology, can provide reusable services that can be adapted using appropriate application information. It is fair to say that the development of any IS system can benefit from a domain ontology and/or an application ontology with this being most evident to developers during the analysis phase. Such ontologies may be available from existing repositories (e.g. [19]) or a domain analysis yielding an ontology may be considered the first stage of developing the system (e.g. as proposed in [20] or in [21]). Some industries such as banking and finance are inclined to provide their own ontologies to enable speedier IS development.

Unfortunately, only a small number of existing methodologies include ontologies in their workproducts and processes. This support is generally confined to the early phases of the development (the analysis phase). For example, Girardi and Serra specify how a domain model that includes goal and role analyses is developed from an initial domain ontology in their methodology [4]. Another example [22] uses ontologies to mediate the transition between goal and task analyses. A better inclusion of ontologies into a development methodology permits the long term reuse of software engineering knowledge and effort and can produce reusable components and designs [23]. But first some of the challenges discussed in [24,25] need to be addressed. Various software components have different knowledge requirements and they relate to the application domain from various layers of abstraction. Components may be complementary and they may have varying degrees of prescription to the domain requiring various degrees of adjustment to suit the domain. For example, a user interface will operate at a different level of abstraction to the application domain than say a component interfacing to a data mining agent or a database service. In other words, the degree of linkage between software components and analysis models depends on the nature of components themselves and how the components relate to the system as a whole. An ontology based approach for development therefore further complicates the analysis activities during development. Even if we assume a high degree of independence between various software components and the application domain, identifying and appropriately using ontologies in the development of components remains difficult. Elsewhere, e.g. in [21] the use of two ontologies, a domain ontology and an application ontology are advocated to guide the verification of requirement models. In this paper, we advocate for an additional ontology that describes the relations between the software components and the way they are structured in the system. For the development of a service oriented system, the use of this *architecture ontology* will in turn enable proper assumptions to be made with relation to the domain and application ontologies and how they can be related to service components. In doing this, we not only identify various roles of an

Please cite this article in press as: J. Shen, et al., Aligning ontology-based development with service oriented systems, Future Generation Computer Systems (2013), http://dx.doi.org/10.1016/j.future.2013.08.005