1-1-2009

Reflecting on ontologies towards ontology-based agent-oriented software engineering

Ghassan Beydoun
University of Wollongong, beydoun@uow.edu.au

Brian Henderson-Sellers
University of Technology, Sydney

Jun Shen
University of Wollongong, jshen@uow.edu.au

G. Low
University of New South Wales

Reflecting on ontologies towards ontology-based agent-oriented software engineering

Abstract

"Ontology" in association with "software engineering" is becoming commonplace. This paper argues for the need to place ontologies at the centre of the software development lifecycle for multi agent systems to enhance reuse of software workproducts as well as to unify agent-based software engineering knowledge. The paper bridges the state-of-the-art of ontologies research from Knowledge Engineering (KE) within Artificial Intelligence and Metamodelling within Software Engineering (SE). It presents a sketch of an ontology-based Multi Agent System (MAS) methodology discussing key roles on ontologies and their impact of workproducts, illustrating these in a MAS software development project for an important application that utilizes dynamic web services composition.

Keywords

Reflecting, ontologies, towards, ontology, based, agent, oriented, software, engineering

Disciplines

Physical Sciences and Mathematics

Publication Details

Beydoun, G., Henderson-Sellers, B., Shen, J. & Low, G. (2009). Reflecting on ontologies towards ontology-based agent-oriented software engineering. In T. Meyer & K. Taylor (Eds.), Advances in Ontologies: 5th Australasian Ontology Workshop (pp. 23-33). Sydney, Australia: Australian Computer Society.

Reflecting on Ontologies Towards Ontology-based Agent-oriented Software Engineering

G. Beydoun[1], B. Henderson-Sellers[2], J. Shen[1], G. Low[3]

[1]{beydoun, jshen} @uow.edu.au, School of Information Systems and Technology, University of Wollongong, Wollongong

[2] brian@it.uts.edu.au, School of Software, University of Technology, Sydney

[3] g.low@unsw.edu.au, School of Information Systems, Technology and Management, University of New South Wales, Sydney

Abstract

"Ontology" in association with "software engineering" is becoming commonplace. This paper argues for the need to place ontologies at the centre of the software development lifecycle for multi agent systems to enhance reuse of software workproducts as well as to unify agent-based software engineering knowledge. The paper bridges the state-of-the-art of ontologies research from Knowledge Engineering (KE) within Artificial Intelligence and Metamodelling within Software Engineering (SE). It presents a sketch of an ontology-based Multi Agent System (MAS) methodology discussing key roles on ontologies and their impact of workproducts, illustrating these in a MAS software development project for an important application that utilizes dynamic web services composition.

Key words: *Software Development Lifecycle (SDLC), Ontologies, Agents, Multi Agent Systems (MAS), Services*

1 Introduction

This paper promotes ontology-based software development with a current focus on methodologies for building a multi agent[1] system (MAS). Substantial integration between ontologies and software engineering has been achieved e.g. in ODE of (Falbo *et al.*, 2005) and Onto (Leppänen, 2007). This paper is part of an ongoing effort to place ontologies at the centre of the software development lifecycle (SDLC) for MASs to enhance the reuse of MAS workproducts as well as to unify agent-based software engineering knowledge.

In a MAS composed of a heterogeneous collection of agents with distinct knowledge-bases and capabilities, coordination and cooperation between agents facilitate the achievement of global goals that cannot be otherwise achieved by a single agent working in isolation (Wooldridge, 2000). The unique characteristics of a MAS have rendered most standard systems development methodologies inapplicable, leading to the development of Agent Oriented Software Engineering (AOSE) methodologies. Several AOSE methodologies exist (Henderson-Sellers and Giorgini, 2005). Indeed any one of the extant methodologies has limited applicability (Tran and et al, 2005) e.g. to a specific domain or a specific type of software application. This limits adoption of AOSE. Furthermore, a review (Tran and Low, 2005) of sixteen prominent AOSE methodologies revealed that most ignore system extensibility, maintenance, interoperability and reusability issues. This imposes a second barrier to the adoption of AOSE. This paper outlines a path towards resolution of both of these barriers through the use of ontologies during the software development lifecycle. Given that the "fixed costs" associated with learning or configuring methodologies to suit the requirements of a given project are high, it is critical to address these concerns and protect the various facets of investments associated with using a MAS including: interoperability of systems, reuse of their components, reuse of human skills acquired and reuse of designs generated during development.

As a first step towards using ontologies as a central software engineering construct throughout the whole development lifecycle of a MAS, this paper reviews the state-of-the-art of ontology research in two key communities: the Artificial Intelligence (AI) community and the Information Systems (IS)/Software Engineering (SE) community. Much of our understanding of ontologies has been derived from the AI community; in contrast, the IS/SE community have focussed on the use of a systematic relationship and understanding of models and metamodels. To illustrate how ontologies can be central to MAS development, we use an example application that also highlights the power of agents. The example chosen is a MAS Peer to Peer system constructed to allow dynamic composition of web services in highly distributed and heterogeneous environments.

The rest of the paper is organised as follows: Section 2 provides a conceptual analysis bridging software engineering concepts and existing ontology research emanating largely from the knowledge engineering community. Section 3, using the grounded position on what an ontology can do to the SDLC, provides an argument placing ontologies at the heart of SDLC specifically tailored for Agent Oriented Software Engineering. Section 4 develops this into a sketch of an AOSE ontology-based methodology. Section 5 illustrates key concepts in an application. Section 6 concludes with a summary and discussion of future work.

2 Background: Bridging Ontologies in KE to Models and Metamodels in SE

In SE, terms such as model, metamodel and ontology are often used with disparate meanings across the literature even within the same sub-domain of SE. To pin down the appropriate usage of an ontology within the SDLC of a methodology, it is important to describe how an ontology may be linked to a model and/or a metamodel and, indeed, how models and metamodels are defined and

[1] Agents are highly autonomous, situated and interactive software components. They sense their environment and respond accordingly.

inter-related. This leads to the need to understand the relationship between ontologies and a metamodelling hierarchy such as that of the (OMG, 2005a) or (ISO/IEC, 2007). (Favre *et al.*, 2007) note the lack of a general, systematic technique to map between metamodels and ontologies which is the focus of this section.

In SE, additional characteristics for an ontology are required. It is widely agreed that it needs to be formal e.g. (Corcho *et al.*, 2006; OMG, 2005b; Guizzardi, 2005; Rilling *et al.*, 2007). However the meaning of 'formal' is not very well agreed. For example, (Corcho *et al.*, 2006) suggests it to mean 'understandable by a computer', OMG suggests it to mean underpinned by a metamodel and (Guizzardi, 2005) uses "formal" to mean "having form" rather than precise or mathematical. A second required characteristic is that it should represent *shared* knowledge e.g. (Gruber, 1993; Noy and McGuinness, 2001) and a third characteristic is that an ontology is represented by a vocabulary (Gruber 1993; Guarino 1998). This last notion is used to differentiate between an ontology linked to a representation in a specific vocabulary *but with a common conceptualization (*Guarino 1998). Following (Guarino, 1998), (Ruiz and Hilera, 2006; Guizzardi, 2005) identify four general kinds of ontologies: *high-level ontologies* (or upper level ontologies)[2], domain ontologies, task ontologies, and application ontologies. This is a scheme that will underpin our ontology-centric SDLC to be detailed in Section 4. This is in accordance with (Ruiz and Hilera, 2006) as shown in Figure 1 which also compares two classification schemes (of (Guarino, 1998 and Fensel, 2004)) and differentiates between domain-independent ontologies and domain-dependent ontologies (a discrimination also adopted in this paper).

To link ontologies to metamodels in current SE, two stacked architectures are commonly used. It is worth noting the OMG architecture based on strict metamodelling wherein the only inter-level relationship permitted is "instance of" (in Figure 2). This is not universally accepted within SE, for instance, the architecture used in ISO/IEC 24744 (ISO/IEC, 2007) (Figure 3) uses the powertype pattern (Gonzalez-Perez and Henderson-Sellers, 2006), which permits both instance-of and generalization relationships between levels. Indeed, as observed in several papers summarized in (Gonzalez-Perez and Henderson-Sellers, 2008) application of the four layer hierarchy used by the OMG to methodologies results in several contradictory situations – hence the creation of the newer architecture in Figure 3. For our purpose, we can say that a metamodel describes a domain that is representative of more than one instance in a less abstract domain and, importantly, each model/metamodel describes a domain of discourse, the *language used* for a metamodel domain and a model domain (although relative) is distinct.

We can now ask which 'metamodels' or 'models' (or both) are useful, both theoretically and pragmatically, to link our SE-defined "ontology" definition. (Atkinson *et al.*, 2006) suggest that ontologies and models may be different technologies since they appear to be derived from different subfields of computing and knowledge representation and there appear to be several projects, for example within the OMG and W3C, aimed at producing a bridge between the technologies. Their conclusion is that ontologies are a subset of models since ontologies fulfil the criteria for being models but have additional characteristics i.e. they are specializations in the object-oriented (OO) sense.

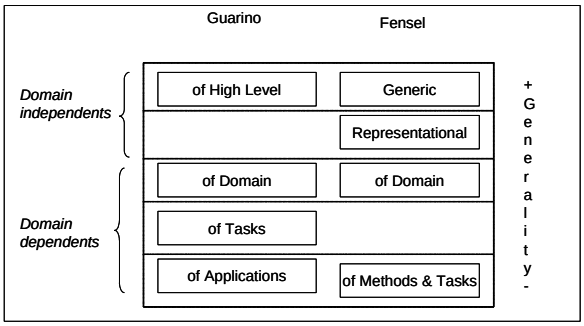

Figure 1 Ontologies by generality level (after (Ruiz and Hilera, 2006))

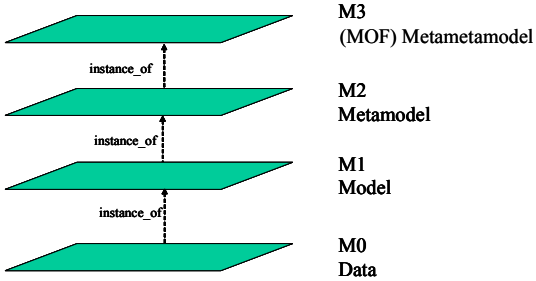

Figure 2 The 4 layer hierarchy of the OMG- based on (ANSI, 1989) (after (Henderson-Sellers and Unhelkar, 2000)) ©Pearson Education Limited

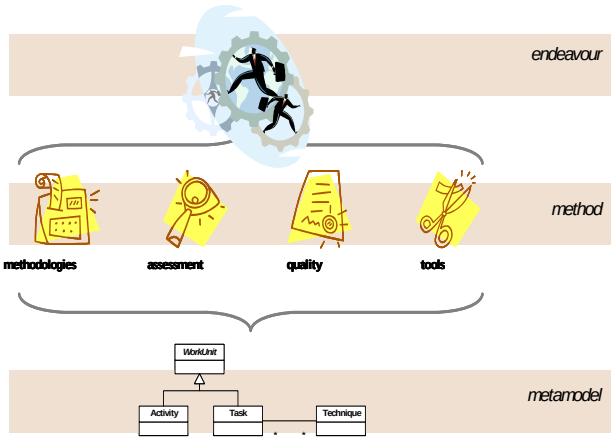

Figure 3 Three layer architecture of ISO/IEC 24744 International Standard (after (Henderson-Sellers, 2006))

While noting that much of ontology design originated in OO design, (Noy and McGuinness, 2001) suggest that OO stresses operational rather than the structural properties of classes, which are the focus of ontology design. This suggests an alignment with data models. On the other hand, the equivalencing of models with database-focussed models, as is done by (Ruiz and Hilera, 2006), unnecessarily restricts the meaning of model for such a comparison to be useful here. In contrast, (OMG, 2005b) takes a broader meaning to the term "conceptual model". It notes some missing concepts in the UML – in

[2] Uschold (2005) suggests that, while an upper-level ontology is important, it is less important *which* such ontology is used. In fact, we omit upper level ontology from our methodological sketch in Section 4.

particular, the treatment of disjoint classes, set intersection and set complement. They argue that ontology instances may also be required without the prior definition of a class (not permissible using UML).

Many other authors equate ontologies with models despite noting the difference in intent i.e. that an ontology is descriptive and a model typically (but not always) prescriptive e.g. (Wand and Weber, 2005; Ruiz and Hilera, 2006). For example, (Gruber, 1993) states that "Ontologies are also like conceptual schemata in database systems" which "provide a logical description of shared data"; and (Guarino, 1998) clearly indicates that he regards an ontology as belonging to the model domain and not the metamodel domain. (Ruiz and Hilera, 2006) suggest differences based on arguing that an ontology is descriptive whereas a metamodel is prescriptive, belonging to the solution domain.

In the context of agent modelling languages, (Guizzardi and Wagner, 2005a) propose a unified foundational ontology (UFO). The UFO is categorized as an upper level ontology (a.k.a. foundational ontology), and an application to business modelling is given in (Guizzardi and Wagner, 2005b). (Guizzardi, 2005) states that a foundational ontology is a *meta-ontology*. Since he, and others, effectively equates "ontology" with "model", then we must conclude that a meta-ontology can be effectively equated with metamodel, at least in the OMG sense. Indeed, in (Guizzardi and Wagner, 2005a) it is clearly stated that a foundational ontology can be represented as a MOF (Metaobject Facility) model, MOF being a language for defining modelling languages i.e. it is used as a metamodelling language. In other words, a foundational ontology is at the metamodel level in that it is equivalent to the UML or the ER definition. This means that we need to reassess Figure 1 because "domain independence" is also seen as a feature of a meta-ontology whilst, in contrast (see Figure 1) a generic model is widely recognized as *not* being at this meta level. In a section entitled *"combining metamodels and ontologies to achieve semantic interoperability"* – words suggesting that ontologies belong to the metalevel – (Karagiannis *et al.*, 2008) go on to describe "semantic mappings between metamodel elements and ontology concepts". Arguably this latter statement, at odds with the former, can be interpreted as ontology concepts being the classes in the ontology metamodel – as for instance documented in the OMG's Ontology Defintion Metamodel (ODM) (OMG, 2005b).

Contrasting several chapters from the same book (Calero *et al.*, 2006), we see that while the software maintenance ontology of (Anquetil *et al.*, 2006) and the software development environment ontology of de (Oliveira *et al.*, 2006) clearly discuss a domain ontology, the ontology for software measurement of (Bertoa *et al.*, 2006) and the ontology for software development methodologies and endeavours are all clearly defined in terms of a metamodel. Indeed, (OMG, 2005b) clearly differentiates between the OWL *metamodel* that allows users to define ontology models and the ontology that is "generally specified as a system of classes and properties (the structure) which is populated by instances (the extents)". Hence, the UoD is described by a set of ontologies where ontologies are used to enhance the target system and be complementary to UML modelling artefacts. In other words, ontologies belong to the M1 level (Figure 2) or Method Domain (Figure 3) since an ontology is a conceptual model (OMG, 2005b), sharing characteristics with more traditional data models. This OMG ODM approach suggests a multi-level ontology architecture (Figure 4). Here, the "M2" level is equivalent with (Guizzardi and Wagner, 2005a)'s foundational ontology, with the OMG's ODM and with the term "upper level ontology". The "M1" level includes not only domain-specific ontologies (such as that for, say, a banking domain) but also a domain-independent generic ontology (cf. Figure 1). Instances of elements of a domain-specific ontology (Figure 4) are discussed in (Noy and McGuinness, 2001) where it is argued that the depth in the ontology hierarchy at which this occurs is context dependent, making no attempt to align with the strict metamodelling architecture of Figure 2.

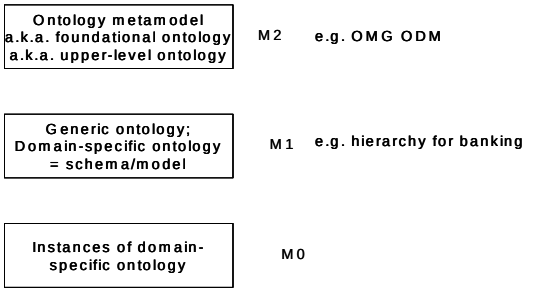

Figure 4. Three level ontology architecture suggested by OMG.

If an ontology refers to a universe of discourse and to conceptualization, as according to (Gruber, 1993), then the term "ontology" would appear to be equally applicable to either M1 or M2 (although not both simultaneously), in just the same way that the term "model" can be applied to a M1 UML visualization (e.g. a system design) or to a M2 visualization (e.g. the UML metamodel). This may explain the ambiguity regarding whether an ontology is an M1 or M2 thing. In some contrast to the notion of ontologies being focussed at their specification level i.e. the metamodel, most "ontologies" found by a web search and documented, for example in Protégé, are hierarchies of terms in a specific (often commercial) domain. For instance, we have located an ontology for newspaper publishing containing elements such as editor, journalist and printing press; an ontology for health care with concepts including doctor and patients. Such an ontological hierarchy bears a good correspondence to a UML model (M1) that might be constructed if one were building software as opposed to the ontological usage of knowledge.

3 Why Must Agent-Oriented SE be Ontology-Centric?

Many of the IS/SE focussed application areas are brought together in (Green and Rosemann, 2005), a volume on business systems analysis. However, there remain only a small number of existing MAS methodologies that 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, 2004) specify how a domain model that includes goal and role analyses is developed from an

initial ontology. Another example (Dileo et al., 2002) uses ontologies to mediate the transition between goal and task analyses. An ontology-based methodological framework that can be used to build new ontology-centric AOSE methodologies from scratch, or a repository of add-on methodological elements that can be added to an existing AOSE methodology to enhance it with new support for ontology-based AOSE, would be a significant innovation in the support for ontology-based AOSE.

In addition, while existing methodologies suffer from other deficiencies (Tran and Low, 2005), there is a growing realization that some form of consolidation is needed. To merge this existing body of agent-oriented software engineering knowledge into a more effective methodological approach, we consider two key issues: how easy it is for software developers to actually apply the outcome (usability) and how feasible is the merging approach (realisability). We identify the following three candidate approaches:

Approach 1: An ad-hoc approach consisting of merging existing methodologies one at a time, with an arbitrary methodology as a starting point, and without guidance on attaching methodologies, beyond avoiding repetition and inconsistent use of terms.

Approach 2: A metamodelling method engineering approach characterised by having a formal unifying formal language (a metamodel) to express various methodology fragments from different sources.

Approach 3: A feature-identification-guided approach to identify AOSE development steps and modelling concepts from existing AOSE methodologies to produce a unified methodological framework that in turn can be used to easily generate methodologies as required.

Approach 1 does not offer any guide on the scope of software development lifecycle concepts and can lead to one of two types of errors: assuming differences of concern when none exists, or falsely assuming similarity of concern because of the common use of terms. The first type of error may lead to repetition and to an unnecessarily large and cumbersome methodology, rendering it less accessible to developers. Tolerating errors of the first type, a successful unification effort would result in a *large* methodology with its bulk concerned with a collection of 'exceptional cases' without common structures. We find that this is exactly what happened with UML (in a slightly different domain but nevertheless providing a highly relevant parallel). The second type of error can create inconsistencies because of inconsistent interpretations of terms. Tolerating such errors, the resultant methodology would produce inconsistent models and lower its usability, as software developers subsequently struggle to deal with problems resulting from inconsistencies and would most likely lead to its abandonment (Bernon et al., 2004).

Approach 2 requires a formal language, a metamodel, whose units serve to generate methodology fragments with similar concerns, but with a different flavour according to the context of the development project. This approach has been the focus of (Beydoun et al., 2006b; Beydoun et al, 2009). In this approach, the development project decides the concern and the flavour of the methodology generated rather than subjective 'interpretations' skewed towards a forced merging between methodologies and their fragments (as in Approach 1). Such interpretations are avoided, preventing any inconsistencies. However, to avoid inconsistencies only a select subset of the rewritten components of methodologies can be integrated at any one time. For example, in every given object-oriented development project (Brinkkemper et al., 2001), a customised integration of selected components is required. For an emerging area of application such as MASs, development experience is limited and the criteria of selection are not yet easily discerned. Hence, the benefit in the applicability of this approach does not outweigh the added effort required for assembling selected method components. Consequently, to balance the work on Framework Agent Modelling Language (FAML) (e.g. (Beydoun et al., 2009)), in this paper Approach 3 is pursued as an alternative, and potentially complementary, approach to a method engineering approach (Henderson-Sellers, 2003) with the aim to explore cross fertilisation between the two approaches. For example, the ontology techniques developed for Approach 3 will be used to enhance the method engineering repository of Approach 2. Approach 3, guided by feature-identification, does not require the cumbersome re-writing of existing methodologies using a common formal language (metamodel) as in Approach 2. It is sufficient to validate and refine the set of candidate steps and modelling concepts and overlay these on top of existing methodologies. Hence, this approach requires much less effort and it is the most realizable as it does not require the collaboration of the creators of the existing methodologies. Crucially, this approach rids developers of the highly specialised and difficult task of the merging of methodology components on a per project basis. The approach instead relies on using explicit ontologies as a focal point during development to facilitate combining features from different AOSE methodologies. This will use ontologies as a means for semantic mappings to convert software work products to suit various development steps. This can substantially support integration of processes and products; and, for the finally implemented MAS, this can support its inter-operation with other systems.

Using off-the-shelf domain ontologies as a starting point of system development, will become the focus of our efforts on the applied use of ontologies in an AOSE methodology (not their actual creation). This will enable the transfer and adaptation existing techniques for ontologies (e.g. techniques for mapping and translating between multiple ontologies) to obtain a more economical approach to MAS development, addressing interoperability and work product reuse. Not only will an ontology-based AOSE methodology be complete and consistent and produce systems that can easily be evolved to new contexts but, in addition, it can have a highly developed maintenance phase to guide developers in reusing existing systems and components previously developed (using an ontological approach). This will foster wider deployment of agent-based systems by

industry by focussing on the commercial success of the technology.

At least three significant contributions to the state-of-the-art in AOSE are identified: firstly, designers will have a tested and verified framework to handle interoperability issues in an heterogeneous environment at design time by allowing a MAS to be formed from loosely coupled components connected through ontological mappings. Thus, they will be inherently flexible and their actual design and architecture will be reusable across applications and in different settings. Secondly, ontological commitments related to a MAS will be explicitly integrated with its actual design and development. In exploring the currently overlooked ontology-related interactions between the analysis and design phases of software development for MAS, iterative verification during the design and development of the system will become possible, increasing the likelihood of producing a correct system. Thirdly, all key concerns of AOSE practitioners will be combined into one methodological framework. The first two contributions are actually interrelated: The explicit and extensive support for *ontology-based* MAS development will address the interoperability concerns in heterogeneous environments.

4 From ontologies in SE to Ontology-based Agent Oriented SE

Inclusion of ontologies into a specific SE methodology for the development of MAS permits the long term reuse of software engineering knowledge and effort and can produce reusable MAS components and designs. (Beydoun et al., 2006a) argue that using ontologies in developing a MAS is complicated by having to simultaneously provide knowledge requirements to different Problem Solving Methods[3] that are still required to share results using a common terminology. This is even further complicated because individual PSMs may operate at different levels of abstraction of the domain, they may be complementary, and they may have varying degrees of prescription to the domain requiring various degrees of adjustment to suit the domain. A set of six requirements were proposed for developing a MAS using an ontology-based software engineering approach. In this section, we present the methodology sketch motivated by the original drive for using ontologies for *reuse* (as also discussed in (Beydoun et al, 2006a)). Specifically, we propose the unification of and reuse of AOSE knowledge (as outlined briefly in the previous section). As targeted by this methodology, the role of ontologies during the SDLC is detailed. Similar to KBS development, it is assumed that the choice of PSM may be made independently of domain analysis. Moreover, it is also assumed that a domain ontology describing domain concepts and their relationships is available. Such an ontology may be available from an existing repository e.g. (DARPA, 2000) or a domain analysis may be considered the first stage of developing the system. The purpose of such a domain analysis would only be to identify concepts and their relationships.

There is inter-play between the role of reuse and other roles of ontologies in a MAS. Various reuse roles cannot be smoothly accommodated (e.g. interoperability at run-time) without careful consideration of run-time temporal requirements. For example, an ontology's role in reasoning at run-time is based on fulfilling PSM knowledge requirements at design time. This requires scoping domain analysis for each individual agent at design time. The key to ontology-based design of a MAS is the appropriate allocation of a PSM to individual agents in order to match system requirements. Towards this, we note that goal analysis is the usual way to express requirements e.g. (Giunchiglia et al, 2003; Wooldridge et al, 2000) and we suggest associating PSMs (using PSM libraries) and system goals in the early stages of a MAS design. The ontologies provides a conceptualization and the basis upon which a machine accessible definition of PSMs may be created (similar to (Fensel, 1997)).

We envisage that the MAS development starts with a domain ontology, an application ontology and a collection of task ontologies used to identify goals and roles of the agents in the system. This in turn is used to index an appropriate set of problem solving capabilities from an appropriate existing library of capabilities. Individual ontologies corresponding to the requirements of each capability are then extracted from the initial common ontology in order to provide knowledge representation and allow reasoning by individual agents. Those ontologies will form the basis for an iterative process to develop a common communication ontology between all agents and verify the knowledge requirements of chosen capabilities. Individual localised ontologies may also require incremental refinement during the iterative process. Appropriate ontology mappings are needed between local ontologies and the communication ontology. To be complete, the methodology needs technical guidelines to develop the various ontology mappings, operators to extract localized agent ontologies from the domain ontology, operators for consistency checking between related ontologies and support for managing reuse tasks in the maintenance phase of the methodology.

The SDLC requires three related ontologies (shown in Figure 5): First is a domain ontology to describe the domain knowledge for the problem and the requirements for a solution to the problem. Domain ontologies may be unique to the problem itself or may be adapted from previous problems in similar domains. Second is problem-type ontology to describe types of problems to which PSMs have been developed to solve. The problem-type ontology is necessary for defining the PSM interface (capabilities and preconditions). In the construction of a PSM library, the problem-type ontology is necessary for indexing suitable PSMs. Third is a PSM ontology to describes knowledge required for the tasks, control structure, and PSM dependencies. An agent that seeks to dynamically select a PSM (or its coded implementation) to solve a problem needs to know this ontology.

[3] PSMs are high-level structures that describe a reasoning process employed to solve general problems (Rodríguez, 2003)

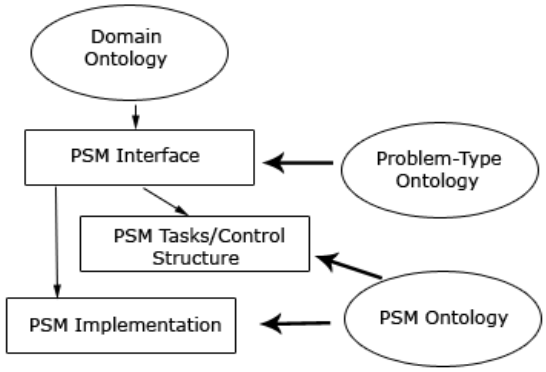

Figure 5 illustrates the role that the ontologies play in PSM implementations. We omit upper level ontologies, we domain ontologies, application ontologies (Problem-type), task ontologies (or PSM ontology)

The collection of all PSMs for local goals should also be verified for completeness against stated system goals. These goals should also be checked against cooperation potential. (A form of distributed goal interaction evaluation could be done using existing approaches e.g. (van Lamsweerde et al, 1998)). Most current methodologies view the decision of problem-solving mechanisms as a low level design step. In our current view, paralleling KBS development, ontology-based design and development requires elevating this to an early design phase and making it central to a later decision on the communication and interface requirement of each agent (rather than the other way around as in many other methodologies e.g. (Giunchiglia et al, 2003; Wooldridge et al, 2000)).

Chosen problem solving capabilities for different agents in a given MAS do not necessarily have the required degree of domain dependence. Hence, for a PSM chosen for some agents, the ontology required may need to be adapted. For this, the domain ontology and the problem-type ontology (application ontology) are again the most convenient reference point. Ontology mapping (between portions of these two ontologies and the local agent's knowledge) is required to ensure that all PSMs have their knowledge requirement available to their reasoning format (adaptors of (Fensel, 1997) may be useful here).

Agents need to communicate their results and instigate cooperation using a common language. For this purpose, we recommend a global communication ontology (as in (Esteva et al, 2002)), rather than many-to-many individual mappings between agents. Such a communication ontology is most conveniently based on the domain ontology available, and it depends on the individual ontology of each agent. In some cases, an ontology mapping may be required between PSM ontologies and the communication ontology. The same adaptation between the reasoning and domain ontology can be used to map the result of reasoning back to a common communication ontology. Our work so far is geared towards 'extendable closed' systems. In the case of 'open systems', introducing new agents may require *runtime* extension of the communication ontology or some local ontologies to allow cooperation with new agents. This is currently beyond our current scope. It is worth noting, that we never assume that local ontologies for agents are complete from the perspective of the agent. This is a considerable step in the right direction towards implementing completely 'open systems'.

Hierarchical ontologies are one way to have flexible domain ontology refinement for agents according to their PSMs, and to accommodate differences in strength of the PSM of agents. A common hierarchical domain ontology can be used as a starting point for verification during development and for multiple access at multiple abstraction levels depending on the individual knowledge requirement of each agent PSM.

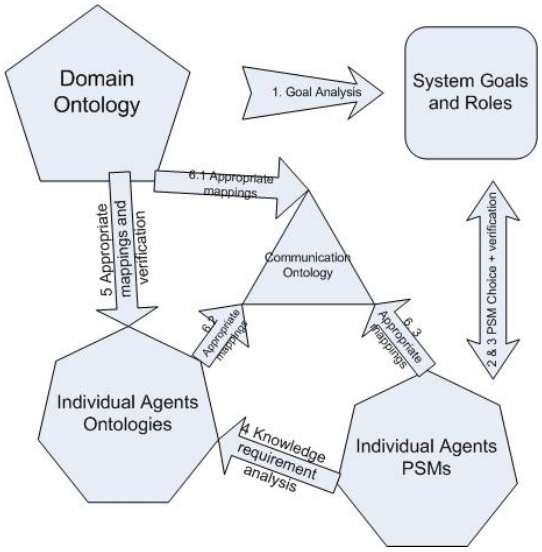

Figure 6. 1. Ontology-based MAS development: Domain Ontology produces Goal Analysis 2. Goal analysis produces a collection of PSMs (using a PSM bank) 3. Knowledge requirement analysis (4). can then be used to delineate local ontologies that can be verified against the domain ontology (step 5). Finally, in step 6 the communication ontology (language) can then be derived using appropriate mappings.

Figure 6 provides the methodological sketch accommodating the observations of this section. The MAS development process starts with a domain and an application ontology (domain-type ontology). These are used to identify goals and roles and to create appropriate interfaces to index an appropriate set of PSMs from a bank of PSMs (see Figure 5 in combination with Figure 6). Appropriate individual ontologies for each PSM are extracted from the initial task ontology. These individual ontologies are used for reasoning by individual problem solvers and may be used to represent results communicated by the individual problem solver. They are next verified against the knowledge requirement of chosen PSMs. The collection of the individualised task ontologies, in combination with the application and domain ontologies, is then used to develop a common communication ontology. Appropriate mappings may be required between individual local ontologies and the communication ontology, to facilitate communicating results between individual agents. Verification between problem solvers and the communication ontology is undertaken, which may result in further localized ontology mappings.

5 Case Study of Ontologies in a MAS application: MAS for Dynamic Web Services Compostion

To illustrate how ontologies can be central to MAS development, we use an example application that also highlights the power of cooperative agents. The application example is a MAS P2P system to allow dynamic composition of web services in highly distributed and heterogenous computing environment and is adapted from (Shen et al, 2007) to highlight how ontologies can be used[4] (using semantically driven composition of services as is often advocated e.g (Souza et al, 2009)). The system will provide, to both service requestors and service providers, Quality of Service (QoS) evaluation. The system will identify service providers' capability and performance so as to enhance the service composition for service clients over the real distributed service network. Due to the complexity of QoS metrics, well-defined QoS service description does not actually exist. With a P2P architecture the QoS is gauged by a service client through cooperative interactions with other peers that can potentially provide the service. The scope of using ontologies in this MAS development is available given that most of the current work focuses on the definition of QoS ontology, vocabulary or measurements and to a lesser extent on a uniform evaluation of qualities, however. Furthermore, a Problem Solving Method unit of analysis nicely corresponds to a service carried by an agent. In this application, the agents themselves will dynamically select PSM implementations that best suit the service or the QoS required. This selection will be made using a P2P searching mechanism to locate appropriate services from other peer agents. Cooperative communication between agents about their existing services, their past services requests and their performance will enable service requesters to locate the service with the most suitable QoS. An ontology-based approach described here will complement existing service repositories, which will provide PSM implementations that may be used in both the design and implementation phases (Figure 7).

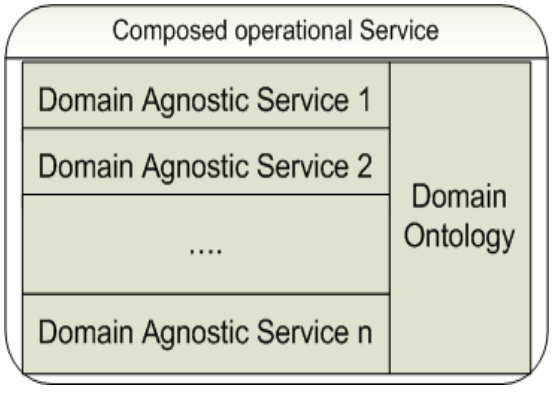

Figure 7. Ontologies can be used to give a dynamic interface to services to agents within a MAS.

When an agent receives a service request that it cannot fulfil, it seeks out a service from another agent or repository of services. This may happen as follows:

1. Identify the corresponding domain of the request
2. Use the domain knowledge to map to the service interface in order to index the PSM corresponding to the service requested.
3. Map its domain knowledge to the individual PSM tasks and perform the tasks to fulfil the service request.

For example, suppose that an agent is interested in engaging in a specific negotiation with another opponent agent. Assuming it is aware of the negotiation protocol, with limited domain knowledge and information about its opponent's preferences, it needs a method to model the opponent and a method to devise a strategy to act. By mapping its domain knowledge to the PSM library, it identifies and employs a suitable coded implementation for a model and strategy. As the negotiation commences, the agent feeds information to the PSM model interface, the model updates, the agent feeds the output of the model (along with the negotiation protocol) to the strategy interface, and follows the recommended course of action. The agent has no fixed automated negotiation approach but, rather, has the capacity to dynamically select the approaches that best suit its circumstance.

In this P2P service evaluation and exchange application, ontologies at various levels of abstractions and details have been developed. This offers a unique workbench to test the reuse of ontologies and places them at the centre of the SDLC. For instance, OWL-S is an ontology to describe Web services with rich semantics. It will allow individual software agents to discover, invoke, compose and monitor Web services with a high degree of automation under dynamic circumstances. The use of this ontology has also been delineated to easily identify problem solving methods of individual agents, bypassing problems identified in (Beydoun et al, 2006a). In fact, OWL-S (OWL-S Coalition, 2006) ontology consists of three main components: the services profile, the process model and the grounding. The services profile is for advertising and discovering Web services. The process model is used to describe detailed operations of services and define composite Web services. The grounding is used to map the abstract definition of services to concrete specifications of how to access the services.

The services profile component of the ontology (corresponding to Task/PSM ontology in Section 4) can be detailed and refined to allow detailed services' description and evaluation. Basically, the service profile does not mandate any representation of services; rather, using the OWL subclass it is possible to create specialised representations of services that can be used as service profiles. OWL-S provides one possible representation through the class "Profile". An OWL-S "Profile" describes services individually as a combination of three basic types of information: what organisation provides each service, what functions each service computes, and a host of features that specify characteristics of each service. In this way, the complementary descriptions about Web services

[4] As a reviewer noted, existing methodologies for creating PSMs are often inadequate. In this example, this problem is by-passed as services do exist and they are typically used to describe atomic tasks within a business process.

including the QoS can be extended in the services profile, so that we can improve the automation and reliability of Web services' composition in dynamic circumstance.

QoS is an important criterion for e-service selection in dynamic environment. In general, QoS refers to the capability of a network to provide better service to selected network traffic over various technologies. As for P2P-based network, the dynamic and unpredictable nature in e-service processes always affects the service's composition and performance significantly. In addition, the dynamic e-business vision calls for a seamless integration of business processes, applications, and e-services over the Web space and time. In other words, QoS properties such as reliability and availability for an e-service process are in high demand. Furthermore, changes and delay in traffic patterns, denial-of-service attacks and the effects of infrastructure failures, low performance in executions, and other quality issues over the Web are creating QoS complications in a P2P network. Quite often, unresolved QoS issues cause critical transactional applications to suffer from unacceptable performance degradation. Consequently, there is a need to distinguish e-services using a set of well-defined QoS criteria.

With the large number of e-services, consumers definitely would like to require a means to distinguish between 'good' and 'bad' service providers. In such a case, QoS is the means to select a 'better' e-service among various providers. From another aspect, the different collaborating e-services applications will compete for network resources in an unreasonable and uncontrollable manner if their interactions are not coordinated by any agreements or specification on QoS differentiation. Naturally, these factors will force service providers to understand and achieve QoS-aware services to meet the demands. Also, a better QoS specification for e-service will become more significant by being a unique selling point for a service provider. Fundamentally, the Web services QoS requirement refers to the quality, both functional and non-functional, aspects of an e-service. This includes performance, reliability, integrity, accessibility, availability, interoperability, and security (Mani and Nagarajan, 2002). The properties become even more complex when adding transactional features to e-services.

How to properly design and integrate QoS criteria in P2P-based e-service process is an important innovation for e-business development in decentralised network. It particularly lends itself to ontology based development, as services correspond to tasks that can be indexed using a task ontology. In a dynamic environment, higher level ontologies (application and domain ontologies) can be used by agents to locate appropriate providers of services and undertaking dynamic evaluation through appropriate communication between agents. (Greco et al., 2004) present an ontology-driven framework to build complex process models that can be reused in this application. More specifically, a web services modelling ontology is described in detail in (Roman et al., 2005) and a "Generic Negotiation Ontology (GNO)" in (Ermolayev and Keberle, 2006) as an upper level negotiation ontology for software agents. All these can be reused in this application.

6 Summary and Conclusions

This paper promotes ontology-based software development with a focus on methodologies for MAS development. The paper first provides a conceptual analysis bridging software engineering concepts (models, modelling, metamodels etc.) and existing ontology research emanating largely from the knowledge engineering community. This provides a grounded position on what an ontology can do to the SDLC and to launch a methodological sketch of an ontology-based multi agent system methodology. Key concepts and roles of an ontology in a SDLC are illustrated in an application, which is amenable to both the deployment of agents and ontologies. Whilst this is a preliminary illustration, it does clearly argue for enhanced reuse by using ontologies as a central software workproduct.

Much work remains to refine the concepts presented in this and to ensure that they are applicable to areas where the use of ontologies is less obvious than the domain discussed in this paper. Towards this, the first step is to develop required ontological techniques. These include ontology-based techniques for consistency checking across products and processes, and ontology-based techniques for testing completeness of products and processes within and across methodologies. Underlying complex issues need to be resolved, e.g. as how to reconcile requirements from multiple sources and multiple versions of ontologies. Another issue is how do candidate Problem Solving Methods get identified to be reused. Moreover, if new Problem Solving Methods are needed for the system and if creating these is too cumbersome, then this could certainly lead to the ontology-based approach to be abandoned (as one reviewer pointed out). It may well turn out that an ontology-based approach is most suited to areas of applications where the set of possible agent actions are well specified in advance e.g. in modelling service oriented systems.

Acknowledgement

This research is supported by the Australian Research Council. This is Contribution number 09/07 of the Centre for Object Technology Applications and Research.

References

Anquetil, N., de Oliveira, K.M. and Dias, M.G.B. (2006), Software maintenance ontology, in *Ontologies for Software Engineering and Software Technology*, Springer, 153-173

ANSI (1989), *Information Resource Dictionary System (IRDS)*, American National Standards Institute, New York

Atkinson, C., Gutheil, M. and Kiko, K. (2006), On the relationship of ontologies and models, in *Meta-Modelling and Ontologies. Proceedings of the 2nd Workshop on Meta-Modelling, WoMM 2006,* LNI Volume P-96, 47-60

Aussenac-Gilles, N. and Sörgel, D. (2005), Supervised text analysis for ontology and terminology engineering, *Applied Ontology*, 1(1), 35-46

Bernon, C., Cossentino, M., Gleizes, M., Turci, P. & Zambonelli, F. (2004), A Study of some Multi-Agent Meta-Models, in proceedings of AOSE04, New York

Bertoa, M.F., Vallecillo, A. and Garcia, F. (2006), An ontology for software measurement, in *Ontologies for Software Engineering and Software Technology*, Springer, 175-196

Beydoun, G., Tran, N., Low, G. and Henderson-Sellers, B. (2006a), Foundations of ontology-based MAS methodologies, *Procs. AOIS2005*, LNAI 3529, Springer-Verlag, 111-123.

Beydoun, G., Gonzalez-Perez, C., Henderson-Sellers, B. and Low, G.C. (2006b), Developing and evaluating a generic metamodel for MAS work products, *Software Engineering for Multi-Agent Systems IV: Research Issues and Practical Applications*, Springer-Verlag, 126-142

Beydoun G., Low G., Henderson-Sellers B., Mouratidis H., Gomez-Sanz J.J., Pavon J., Gonzalez-Perez C. (2009), FAML: A Generic Metamodel for MAS Development, *IEEE Transaction on Software Engineering*, 35 (6)

Brinkkemper, S., Saeki, M. & Harmsen, F. (2001), A Method Engineering Language for the Description of Systems Development Methods, in proceedings of CAiSE04, 473-476

Calero, C., Ruiz, F. and Piattini, M. (2006), *Ontologies for Software Engineering and Software Technology*, Springer

Corcho, O., Fernandez-Lopez, M. and Gomez-Perez, A. (2006), Ontological engineering: principles, methods, tools and languages, in *Ontologies for Software Engineering and Software Technology*, Springer, 1-48

DARPA (2000), Ontology Repository, http://www.daml.org/ontologies/, accessed on July 2009

Dileo, J., Jacobs, T. & Deloach, S. (2002), Integrating Ontologies into MAS Engineering, in 4th International Bi-Conference Workshop on Agent Oriented Information Systems, Italy,

Dillon, T.S., Chang, E. and Wongthongtham, P. (2008), Ontology-based software engineering - software engineering 2.0, in *Procs. 19th ASWEC2008*

Ermolayev, V. and Keberle, N. (2006), A Generic ontology of rational negotiation, in proceedings of *Information Systems Technology and its Applications, 5th International Conference ISTA*, Germany

Esteva, M., Cruz, D.d.l, and Sierra, C. (2002), ISLANDER: an electronic institutions editor, in *International Conference on Autonomous Agents & Multiagent Systems (AAMAS02)*, Italy, ACM

Falbo, R.A., Ruy, F.B. and Moro, R.D. (2005), Using ontologies to add semantics to a software engineering environment, *Procs. SEKE 2005*, KSI Graduate School, USA, 151-156

Favre, J.-M., Gašević, D., Lämmel, R. and Winter, A., 2007, 3rd International Workshop on Metamodels, Schemas, Grammars and Ontologies, 52-55

Fensel, D., 2004, *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*, second edition, Springer-Verlag, Berlin, Heidelberg.

Fensel, D., 1997, The tower-of-adaptor method for developing and reusing problem-solving methods, in *European Knowledge Acquisition Workshop*, Springer, Spain.

Girardi, R. & Serra, I. 2004. Using ontologies for the specification of domain-specific languages in multi-agent domain engineering, CAiSE04 Workshops (2), 295-308

Giunchiglia, F., J. Mylopoulos and Perini, A., 2003, The Tropos Software Development Methodology: Processes, Models and Diagrams, in proceedings of *Agent-Oriented Software Engineering III: Third International Workshop, AOSE 2002*, Springer. p. 162-173.

Gonzalez-Perez, C. and Henderson-Sellers, B., 2006a, A powertype-based metamodelling framework, *Software and Systems Modeling*, 5(1), 72-90

Gonzalez-Perez, C. and Henderson-Sellers, B., 2008, *Metamodelling for Software Engineering*, J. Wiley & Sons, Chichester, UK

Greco, G., Guzzo, A., Pontieri, L. and Saccà, D, 2004, An Ontology-Driven Process Modeling Framework, in *Database and Expert Systems Applications*, Springer

Green, P. and Rosemann, M., 2005a, *Business Systems Analysis with Ontologies*, IGI, Hershey, PA, USA

Gruber T. 1993, A translation approach to portable ontology specifications, *Knowledge Acquisition*, 5(2), 199-220

Guarino, N., 1998, Formal ontology and information systems, in *Formal Ontology in Information Systems. Procs. FOIS'98*, IOS Press, Amsterdam, 3-15

Guessoum, Z., Rejeb, L. & Durand, R., 2004, Using adaptive Multi-Agent Systems to Simulate Economic Models, in proceedings of AAMAS04, New York

Guizzardi, G. and Wagner, G., 2005a, Towards ontological foundations for agent modelling concepts using the Unified Foundational Ontology (UFO), in *Agent-Oriented Information Systems II* , Springer, Berlin

Guizzardi, G. and Wagner, G., 2005b, Some applications of a Unified Foundational Ontology in business modeling, in *Business Systems Analysis with Ontologies*, IGI, USA

Hamza, H.S., 2005, A Pattern-based approach for developing business object models with ontologies, *SEKE 2005*, Knowledge Systems Institute Graduate School, USA

Happel, H.-J., Seedorf, S., 2006, Applications of Ontologies in Software Engineering, Procs. 2nd International Workshop on Semantic Web Enabled Software Engineering, U.S.A.

Henderson-Sellers, B., 2003, Method engineering for OO systems development, *Comm. ACM*, 46, 73-78.

Henderson-Sellers, B. (2005), Creating a comprehensive agent-oriented methodology - using method engineering and OPEN, *Agent-Oriented Methodologies*, Idea Group.

Henderson-Sellers, B. and Giorgini, P. (Eds.) (2005), Agent-Oriented Methodologies, Hershey, USA, Idea Group

Henderson-Sellers, B. (2006), Method engineering: theory and practice, /Information Systems Technology and its Applications, 5th International Conference, Klagenfurt, Austria, Lecture Notes in Informatics (LNI), Bonn, 13-23

Henderson-Sellers, B. and Unhelkar, B. (2000), *OPEN Modeling with UML*, Addison-Wesley

Hesse, W. (2008a), Engineers discovering the "real world" – from model-driven to ontology-based software engineering, in *UNISCON 2008*, Springer-Verlag

Holten, R., Dreiling, A. and Becker, J. (2005), Ontology-driven method engineering for ISD, in *Business Systems Analysis with Ontologies*, IGI, USA, 174-217

ISO/IEC (2007), *24744: Software Engineering - Metamodel for Development Methodologies*, ISO/IEC, Geneva

Karagiannis, D., Fill, H.-G., Höfferer, P. and Nemetz, M. (2008), Metamodelling: some application areas in information systems, in *UNISCON 2008*, Springer-Verlag, 175-188

van Lamsweerde, A., Darimont, R. and Letier, E. (1998), Managing Conflict in Goal-Driven Requirements Engineering. *IEEE Transaction on Software Engineering*, 24(11)

Leppänen, M. (2007), An Ontological framework of method engineering: an overall structure, *Procs. EMMSAD'07*, Tapir Academic Press, Trondheim, Norway, 47-57

Mani, A. and Nagarajan, A. (2002), Understanding quality of service for Web services, http://www.ibm.com/developerworks/library/ws-quality.html. accessed June 2008

Mendes, O. and Abran, A. (2005), Issues in the development of an ontology for an emerging engineering discipline, *Procs. SEKE 2005*, KSI Graduate School, USA, 139-144

Noy, N.F. and McGuinness, D.L. (2001), *Ontology development 101: a guide to creating your first ontology,* Stanford Technical Reports KSL-01-05 and SMI-2001-0880.

Oliveira, K.M. de, Villela, K., Regina Rocha, A. and Horta Travassos, G. (2006), Use of ontologies in software development environments, in *Ontologies for Software Engineering and Software Technology*, Springer, 275-309.

OMG (2005a), Unified Modeling Language: Superstructure, Version 2.0, formal/05-07-04, 709pp

OMG (2005b), Ontology definition metamodel, ad/2005-08-01

Purao, S. and Storey, V.C. (2005), A multi-layered ontology for comparing relationship semantics in conceptual models of databases, *Applied Ontology,* 1, 117-139

Rilling, J., Zhang, Y., Meng, W.J., Witte, R., Haarslev, V. and Charland, P. (2007), A unified ontology-based process model for software maintenance and comprehension, in *Models in Software Engineering*, Springer, 56-65

Rodríguez, A., Palma, J. and Quintana, F. (2003), Experiences in Reusing Problem Solving Methods - An Application in Constraint Programming. *Proceedings of Knowledge-Based Intelligent Information and Engineering Systems, 7th International Conferece, KES*, 1299-1306

Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C. and Fensel, D. (2005), Web Service Modeling Ontology, *Applied Ontology*, 1(1), 77-106

Ruiz, F. and Hilera, J.R. (2006), Using ontologies in software engineering and technology, in *Ontologies for Software Engineering and Software Technology*, Springer, 49-102

Shen J., Yang Y., Yan J. (2007), A P2P based Service Flow System with Advanced Ontology-based Service Profiles, *International Journal of Advanced Engineering Informatics, Elsevier, 21 (2), pp.221-229*

Sousa, J.P.P., Carrapatoso, E., Fonesca, B., Pimentel, M.G.C., Bulcao-Neto, R.F. (2009), Composition of Context-Aware Mobile Services Using a Semantic Context Model, *IARIA International Journal on Advances in Software*, 2(2):1-13

Tran, Q. N. N., Low, G. C. (2005), Comparison of Methodologies, in *Agent-Oriented Methodologies*, Idea Group Publishing, 341-367

Tran, Q., Low, G., Beydoun, G. (2006), A methodological framework for ontology centric oriented software engineering, International Journal of Computer Systems Science & Engineering, 21(2), 117-132.

Uschold, M. (2005), An ontology research pipeline, *Applied Ontology*, 1, 13-16

Wand, Y. and Weber, R. (2005), Introduction: setting the scene, in *Business Systems Analysis with Ontologies*, IGI Group Publishing, Hershey, PA, USA, xii-xv

Wooldridge, M., N.R. Jennings and D. Kinny (2000), The Gaia Methodology for Agent-Oriented Analysis and Design, in *Autonomous Agents and Multi-Agent Systems*. The Netherlands: Kluwer Academic Publishers