2

Towards Semantic Interoperability in Information Technology: On the Advances in Automation

Gleison Baiôco, Anilton Salles Garcia and Giancarlo Guizzardi
Federal University of Espírito Santo
Brazil

1. Introduction

Automation has, over the years, assumed a key role in various segments of business in particular and, consequently, society in general. Derived from the use of technology, automation can reduce the effort spent on manual work and the realization of activities that are beyond human capabilities, such as speed, strength and precision. From traditional computing systems to modern advances in information technology (IT), automation has evolved significantly. At every moment a new technology creates different perspectives, enabling organizations to offer innovative, low cost or custom-made services. For example, advents such as artificial intelligence have enabled the design of intelligent systems capable of performing not only predetermined activities, but also ones involving knowledge acquisition. On the other hand, customer demand has also evolved, requiring higher quality, lower cost or ease of use. In this scenario, advances in automation can provide innovative automated services as well as supporting market competition in an effective and efficient way. Considering the growing dependence of automation on information technology, it is observed that advances in automation require advances in IT.

As an attempt to allow that IT delivers value to business and operates aligned with the achievement of organizational goals, IT management has evolved to include IT service management and governance, as can be observed by the widespread adoption of innovative best practices libraries such as ITIL (ITIL, 2007) and standards such as ISO/IEC 20000 (ISO/IEC, 2005). Nonetheless, as pointed out by Pavlou & Pras (2008), the challenges arising from the efforts of integration between business and IT remain topic of various studies. IT management, discipline responsible for establishing the methods and practices in order to support the IT operation, encompasses a set of interrelated processes to achieve this goal. Among them, configuration management plays a key role by providing accurate IT information to all those involved in management. As a consequence, semantic interoperability in the domain of configuration management has been considered to be one of the main research challenges in IT service and network management (Pras et al., 2007). Besides this, Moura et al. (2007) highlight the contributions that computer systems can play in terms of process automation, especially when they come to providing intelligent solutions, fomenting self-management. However, as they emphasize, as an emerging paradigm, this initiative is still a research challenge.

According to Pras et al. (2007), the use of ontologies has been indicated as state of the art for addressing semantic interoperability, since they express the meaning of domain concepts and relations in a clear and explicit way. Moreover, they can be implemented, thereby enabling process automation. In particular, ontologies allow the development of intelligent systems (Guizzardi, 2005). As a result, they foment such initiatives as self-management. Besides this, it is important to note that ontologies can promote the alignment between business and IT, since they maximize the comprehension regarding the domain conceptualization for humans and computer systems. However, although there are many works advocating their use, there is not one on IT service configuration management that can be considered as a de facto standard by the international community (Pras et al., 2007).

As discussed in Falbo (1998), the development of ontologies is a complex activity and, as a result, to build high quality ontologies it is necessary to adopt an engineering approach which implies the use of appropriate methods and tools. According to Guizzardi (2005, 2007), ontology engineering should include phases of conceptual modeling, design and implementation. In a conceptual modeling phase, an ontology should strive for expressiveness, clarity and truthfulness in representing the domain conceptualization. These characteristics are fundamental quality attributes of a conceptual model responsible for its effectiveness as a reference framework for semantic interoperability. The same conceptual model can give rise to different ontology implementations in different languages, such as OWL and RDF, in order to satisfy different computational requirements. Thus, each phase shall produce different artifacts with different objectives and, as a consequence, requires the use of languages which are appropriate to the development of artifacts that adequately meet their goals. As demonstrated by Guizzardi (2006), languages like OWL and RDF are focused on computer-oriented concerns and, for this reason, improper for the conceptual modeling phase. Philosophically well-founded languages are, conversely, committed to expressivity, conceptual clarity as well as domain appropriateness and so suitable for this phase.

Considering these factors, Baiôco et al. (2009) present a conceptual model of the IT service configuration management domain based on foundational ontology. Subsequently, Baiôco & Garcia (2010) present an implementation of this ontology, describing how a conceptual model can give rise to various implementation models in order to satisfy different computational requirements. The objective of this chapter is to provide further details about this IT service configuration management ontology, describing the main ontological distinctions provided by the use of a foundational ontology and how these distinctions are important to the design of models aligned with the universe of discourse, maximizing the expressiveness, clarity and truthfulness of the model and consequently the semantic interoperability between the involved entities. Moreover, this chapter demonstrates how to apply the entire adopted approach, including how to generate different implementations when compared with previous ones. This attests the employed approach, makes it more tangible and enables to validate the developed models as well as demonstrating their contributions in terms of activity automation.

It is important to note that the approach used in this work is not limited to the domain of IT service configuration management. In contrast, it has been successfully employed in many fields, such as oil and gas (Guizzardi et al., 2009) as well as medicine (Gonçalves et al., 2011). In fact, the development of a computer system involves the use of languages able to adequately represent the universe of discourse. According to Guizzardi (2005), an imprecise representation of state of affairs can lead to a false impression of interoperability, i.e.

although two or more systems seem to have a shared view of reality, the portions of reality that each of them aims to represent are not compatible. As an alternative, ontologies have been suggested as the best way to address semantic interoperability. Therefore, in particular, the ontological evaluation realized in this work contributes to the IT service configuration management domain, subsidizing solutions in order to address key research challenges in IT management. In general, this chapter contributes to promote the benefits of the employed approach towards semantic interoperability in IT in various areas of interest, maximizing the advances in automation. Such a contribution is motivated in considering that although recent research initiatives such as that of Guizzardi (2006) have elaborated on why domain ontologies must be represented with the support of a foundational theory and, even though there are many initiatives in which this approach has been successfully applied, it has not yet been broadly adopted. As reported by Jones et al. (1998), most existing methodologies do not emphasize this aspect or simply ignore it completely, mainly because it is a novel approach.

In this sense, this chapter is structured as follows: Section 2 briefly introduces the IT service configuration management domain. Section 3 discusses the approach to ontology development used in this work. Section 4 presents the conceptual model of the IT service configuration management domain. Section 5 shows an implementation model of the conceptual model presented in Section 4 and finally Section 6 relates some conclusions and future works.

2. IT service configuration management

The business of an organization requires quality IT services economically provided. According to ITIL, to be efficient and effective, organizations need to manage their IT infrastructure and services. Configuration management provides a logical model of an infrastructure or service by identifying, controlling, maintaining and verifying the versions of configuration items in existence. The logical model of IT service configuration management is a single common representation used by all parts of IT service management and also by other parties, such as human resources, finance, suppliers and customers. A configuration item, in turn, is an infrastructure component or an item that is or will be under the control of configuration management (ITIL, 2007; ISO/IEC, 2005). For innovative IT management approaches such as ITIL and ISO/IEC 20000, configuration items are viewed not only as individual resources but as a chain of related and interconnected resources compounding services. Thus, just as important as controlling each item is managing how they relate to each other. These relationships form the basis for activities such as impact assessment.

According to ITIL and ISO/IEC 20000, a configuration item and the related configuration information may contain different levels of detail. Examples include an overview of all services or a detailed view of each component of a service. Thus, a configuration item may differ in complexity, size and type, ranging from a service, including all hardware, software and associated documentation, to a single software module or hardware component. Configuration items may be grouped and managed together, e.g. a set of components may be grouped into a release. Furthermore, configuration items should be selected using established selection criteria, grouped, classified and identified in such a way that they are manageable throughout the service lifecycle.

As with any process, IT service configuration management is associated with goals that in its case include: (i) supporting, effectively and efficiently, all other IT service management processes by providing configuration information in a clear, precise and unambiguous way; (ii) supporting the business goals and control requirements; (iii) optimizing IT infrastructure settings, capabilities and resources; (iv) subsidizing the dynamism imposed on IT by promoting rapid responses to necessary changes and by minimizing the impact of changes in the operational environment. To achieve these objectives, configuration management should, in summary, define and control the IT components and maintain the configuration information accurately. Based on best practices libraries such as ITIL and standards such as ISO/IEC 20000 for IT service management, the activities of an IT service configuration management process may be summarized as: (i) planning, in order to plan and define the purpose, scope, objectives, policies and procedures as well as the organizational and technical context for configuration management; (ii) identification, aiming to select and identify the configuration structures for all the items (including their owner, interrelationships and configuration documentation), allocate identifiers and version numbers for them and finally label each item and enter it on the configuration management database (CMDB); (iii) control, in order to ensure that only authorized and identified items are accepted and recorded, from receipt to disposal, ensuring that no item is added, modified, replaced or removed without appropriate controlling documentation; (iv) status accounting and reporting, which reports all current and historical data concerned with each item throughout its life cycle; (v) verification and audit, which comprises a series of reviews and audits that verify the physical existence of items and check that they are correctly recorded in the CMDB.

As an attempt to promote efficiency and effectiveness, IT management has evolved to include IT service management and governance, which aims to ensure that IT delivers value to business and is aligned with the achievement of organizational goals. As emphasized by Sallé (2004), in this context, IT processes are fully integrated into business processes. Thus, one of the main aspects to be considered is the impact of IT on business processes and vice versa (Moura et al., 2008). As a consequence, IT management processes should be able to manage the entire chain, i.e. from IT to business. For this reason, the search for the effectiveness of such paradigms towards business-driven IT management has been the topic of several studies in network and service management (Pavlou & Pras, 2008). According to Moura et al. (2007), one of the main challenges is to achieve the integration between these two domains. Configuration management, in this case, should be able to respond in a clear, precise and unambiguous manner to the following question: what are the business processes and how are they related to IT services and components (ITIL, 2007)? Furthermore, as cited by ITIL, due to the scope and complexity of configuration management, keeping its information is a strenuous activity. In this sense, research initiatives consider automation to be a good potential alternative. In fact, the automation of management processes has been recognized as one of the success factors to achieve a business-driven IT management, especially when considering intelligent solutions promoting self-management (Moura et al., 2007). Besides its scope and complexity, configuration management is also closely related to all other management processes. In IT service management and governance, this close relationship includes the interaction among the main entities involved in this context, such as: (i) business, (ii) people, (iii) processes, (iv) tools and (v) technologies (ITIL, 2007). Thus, semantic interoperability among such entities has been characterized as one of the main research challenges, not only in terms of the

configuration management process, but in the whole chain of processes that comprise the discipline of network and service management (Pras et al., 2007).

According to Pras et al. (2007), the use of semantic models, in particular, the use of ontologies, has been regarded as the best way with respect to initiatives for addressing issues related to semantic interoperability problems in network and service management. According to these authors, ontologies make the meaning of the domain concepts such as IT management, as well as the relationships between them, explicit. Additionally, this meaning can be defined in a machine-readable format, making the knowledge shared between humans and computer systems, enabling process automation, as outlined by these authors. From this point of view, it is worth mentioning that ontologies are considered as potential tools for the construction of knowledge in intelligent systems (Guizzardi, 2005). Thus, they allow the design of intelligent and above all interoperable solutions, fomenting initiatives as self-management. Finally, it is important to note that ontologies can promote the alignment between business and IT when applied in the context of IT service management and governance since they maximize the expressiveness, clarity and truthfulness of the domain conceptualization for humans and computer systems. However, Pras et al. (2007) point out that despite the efforts of research initiatives, there are still many gaps to be addressed.

Several studies claim that the use of ontologies is a promising means of achieving interoperability among different management domains. However, an ontology-based model and formalization of IT service configuration management remains a research challenge. Regarding limitations, it should be mentioned that ontologies are still under development in the management domain. In fact, the technology is not yet mature and there is not an ontology that can be considered as a *de facto* standard by the international community (Pras et al., 2007). In general, the research initiatives have not employed a systematic approach in the development of ontologies. According to Falbo (1998), the absence of a systematic approach, with a lack of attention to appropriate methods, techniques and tools, makes the development of ontologies more of an art rather than an engineering activity. According to Guizzardi (2005, 2007), to meet the different uses and purposes intended for the ontologies, ontology engineering should include phases of conceptual modeling, design and implementation. Each phase should have its specific objectives and thus would require the use of appropriate languages in order to achieve these goals. However, in most cases, such research initiatives are engaged with the use of technologies and tools such as Protégé and OWL. Sometimes these technologies and tools are used in the conceptual modeling phase, which can result in various problems relating to semantic interoperability, as shown in Guizzardi (2006). At other times, however, they are employed in the implementation phase, ignoring previous phases such as conceptual modeling and design. As a result, such initiatives are obliged to rely on models of low expressivity. Moreover, in most cases, such initiatives propose the use of these technologies and tools for the formalization of network management data models, such as MIB, PIB and the CIM schema. It is noteworthy that data models are closely related to the underlying protocols used to transport the management information and the particular implementation in use. In contrast, information models work at a conceptual level and they are intended to be independent of any particular implementation or management protocol. Working at a higher level, information models usually provide more expressiveness (Pras et al., 2007). Following this approach, Lopez de Vergara et al. (2004) propose an integration of the concepts that currently belong to different

network management data models (e.g. MIB, PIB and the CIM schema) in a single model, formalized by ontology languages such as OWL. In an even more specific scenario, i.e. with no intention to unify the various models but only to formalize a particular model, Majewski et al. (2007) suggest the formalization of the CIM schema through ontology languages such as OWL. Similarly, while differentiating the type of data model, Santos (2007) presents an ontology-based network configuration management system. In his work, the proposed ontology was developed according to the MIB data model concepts. As MIB is limited in describing a single system, a view of the entire infrastructure, including the relationships between its components, is not supported by the model. In practice, this gap is often filled by functionalities provided by SNMP-based network management tools which, for example, support the visualization of network topologies (Brenner et al., 2006). Aside from the fact that, in general, the research initiatives are committed to the use of technologies and tools, it is also observed that they are characterized by specific purposes in relation to peculiar applications in information systems that restrict their conceptualizations. In Xu and Xiao (2006), an ontology-based configuration management model for IP network devices is presented, aiming at the use of ontology for the automation of this process. In Calvi (2007), the author presents a modeling of the IT service configuration management described by the ITIL library based on a foundational ontology. The concepts presented and modeled in his work cover a specific need regarding the demonstration of the use of ITIL processes for a context-aware service platform. Finally, there are approaches that seek to establish semantic interoperability among existing ontologies by means of ontological mapping techniques, as evidenced in Wong et al. (2005). However, it is not within the scope of such approaches to develop an ontology but rather to integrate existing ones.

Therefore, in considering the main challenges as well as the solutions which are considered to be state of the art and in analyzing the surveyed works, it is observed that there are gaps to be filled, as highlighted by Pras et al. (2007). In summary, factors such as the adoption of *ad hoc* approaches, the use of inappropriate references about the domain, the intention of specific purposes and, naturally, the integration of existing ontologies, all result in gaps. As a consequence, such factors do not promote the conception of an ontology able to serve as a reference framework for semantic interoperability concerning the configuration management domain in the context of IT service management and governance. This scenario demonstrates the necessity of a modeling which considers the gaps and, therefore, promotes solutions in line with those suggestions regarded as state of the art for the research challenges discussed earlier in this chapter. In particular, it demonstrates the necessity of an appropriate approach for the construction of ontologies as a subsidy for such modeling. In this sense, the next section of this chapter presents an approach for ontology development.

3. Ontology engineering

In philosophy, ontology is a mature discipline that has been systematically developed at least since Aristotle. As a function of the important role played by them as a conceptual tool, their application to computing has become increasingly well-known (Guizzardi et al., 2008). According to Smith and Welty (2001), historically there are three main areas responsible for creating the demand for the use of ontologies in computer science, namely: (i) database and information systems; (ii) software engineering (in particular, domain engineering); (iii) artificial intelligence. Additionally, Guizzardi (2005) includes the semantic web, due to the important role played by this area in the current popularization of the term.

According Guizzardi et al. (2008), an important point to be emphasized is the difference between the senses of the term ontology when used in computer science. In conceptual modeling, the term has been used as its definition in philosophy, i.e. as a philosophically well-founded domain-independent system of formal categories that can be used to articulate domain-specific models of reality. On the other hand, in most other areas of computer science, such as artificial intelligence and semantic web, the term ontology is generally used as: (i) an engineering artifact designed for a specific purpose without giving much importance to foundational issues; (ii) a representation of a particular domain (e.g. law, medicine) expressed in some language for knowledge representation (e.g. RDF, OWL).

From this point of view, the development of ontologies should consider the various uses and, consequently, the different purposes attributed to ontologies as well as any existing interrelationship in order to enable the construction of models that satisfactorily meet their respective goals. However, despite the growing use of ontologies and their importance in computing, the employed development approaches have generally not considered these factors, resulting in inadequate models for the intended purpose. In considering such distinctions Guizzardi (2005, 2007) elaborates and discusses a number of questions in order to elucidate such divergences and thus provide a structured way with respect to the use of ontologies. In addition, Guizzardi and Halpin (2008) describe that the interest in proposals for foundations in the construction of ontologies has been the topic of several studies and they report some innovative and high quality research contributions. It is based on such questions that are elaborated the further discussions contained in this section and thus the approach used for the construction of the ontological models proposed in this work.

As discussed in Falbo (1998), the development of ontologies is a complex activity and, hence, in order to build high quality ontologies, able to adequately meet their various uses and purposes, it is necessary to adopt an engineering approach. Thus, unlike the various *ad hoc* approaches, the construction of ontologies must use appropriate methods and tools. Falbo (2004) proposes a method for building ontologies called SABiO (Systematic Approach for Building Ontologies). This method proposes an life cycle by prescribing an iterative process that comprises the following activities: (i) purpose identification and requirements specification, which aims to clearly identify the ontology's purpose and its intended use by means of competence questions; (ii) ontology capture, viewing to capture relevant concepts existing within the universe of discourse as well as their relationships, properties and constraints, based on the competence questions; (iii) ontology formalization, which is responsible for explicitly representing the captured conceptualization by means of a formal language, such as the definition of formal axioms using first-order logic; (iv) integration with existing ontologies, in order to search for other ones with the purpose of reuse and integration; (v) ontology evaluation, which aims to identify inconsistencies as well as verifying truthfulness in line with the ontology's purpose and requirements; (vi) ontology documentation. Noticeably, the competence questions form an important concept within SABiO, i.e. the questions the ontology should be able to answer. They provide a mechanism for defining the scope and purpose of the ontology, guiding its capture, formalization and evaluation - regarding this last aspect, especially with respect to the completeness of the ontology.

The elements that constitute the relevant concepts of a given domain, understood as domain conceptualization, are used to articulate abstractions of certain states of affairs in reality, denominated as domain abstraction. As an example, consider the domain of product sales.

A conceptualization of this domain can be constructed by considering concepts such as, *inter alia*: (i) customer, (ii) provider, (iii) product, (iv) is produced by, (v) is sold to. By means of these concepts, it is possible to articulate domain abstractions of certain facts extant in reality, such as: (i) a product is produced by the provider and sold to the customer. It is important to highlight that conceptualizations and abstractions are abstract entities which only exist within the mind of a user or a community of users of a language. Therefore, in order to be documented, communicated and analyzed, they must be captured, i.e. represented in terms of some concrete artifact. This implies that a language is necessary for representing them in a concise, complete and unambiguous way (Guizzardi, 2005). Figure 1-a presents "Ullmann's triangle" (Ullmann, 1972), which illustrates the relation between a language, a conceptualization and the part of reality that this conceptualization abstracts. The relation "represents" concerns the definition of language semantics. In other words, this relation implies that the concepts are represented by the symbols of language. The relation "abstracts", in turn, denotes the abstraction of certain states of affairs within the reality that a given conceptualization articulates. The dotted line between language and reality highlights the fact that the relation between language and reality is always intermediated by a certain conceptualization. This relation is elaborated in Figure 1-b, which depicts the distinction between an abstraction and its representation, as well as their relationships with the conceptualization and representation language. The representation of a domain abstraction in terms of a representation language is called model specification (or simply model, specification or representation) and the language used for its creation is called modeling language (or specification language).

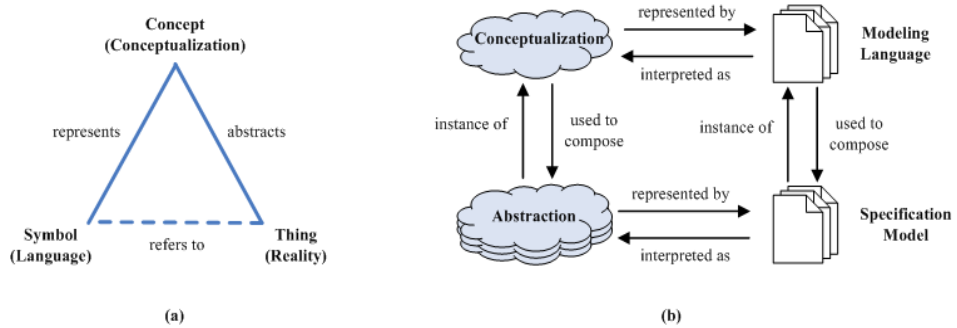

Fig. 1. Ullmann's triangle and relations between conceptualization, abstraction, modelling language and model, according to Guizzardi (2005).

Thus, in addition to the adoption of appropriate methods, able to systematically lead the development process, ontology engineering as an engineering process aims at the use of tools, which should be employed in accordance with the purpose of the product that is being designed. In terms of an ontology development process such tools include modeling languages or even ontology representation languages. According to Guizzardi (2005), one of the main success factors regarding the use of a modeling language is its ability to provide its users with a set of modeling primitives that can directly express the domain conceptualization. According to the author, a modeling language is used to represent a

conceptualization by compounding a model that represents an abstraction, which is an instance of this conceptualization. Therefore, in order for the model to faithfully represent an abstraction, the modeling primitives of the language used to produce the model must accurately represent the domain conceptualization used to articulate the abstraction represented by the model. According to Guizzardi (2005), if a conceptual modeling language is imprecise and coarse in the description of a given domain, then there can be specifications of the language which, although grammatically valid, do not represent admissible state of affairs. Figure 2-a illustrates this situation. The author also points out that a precise representation of a given conceptualization becomes even more critical when it is necessary to integrate different independently developed models (or systems based on these models). As an example, he mentions a situation in which it is necessary to have the interaction between two independently developed systems which commit to two different conceptualizations. Accordingly, in order for these two systems to function properly together, it is necessary to ensure that they ascribe compatible meanings to the real world entities of their shared subject domain. In particular, it is desirable to reinforce that they have compatible sets of admissible situations whose union (in the ideal case) equals the admissible states of affairs delimited by the conceptualization of their shared subject domain. The ability of entities (in this case, systems) to interoperate (operate together) while having compatible real-world semantics is known as semantic interoperability (Vermeer, 1997). Figure 2-b illustrates this scenario.

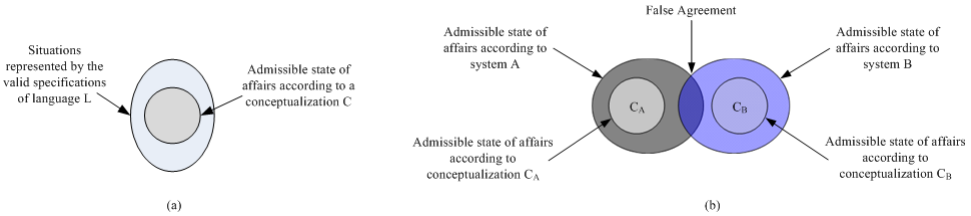

Fig. 2. Consequences of an imprecise and coarse modelling language (Guizzardi, 2005).

In Figure 2-b, C_A and C_B represent the conceptualizations of the domains of systems A and B, respectively. As illustrated in this figure, these conceptualizations are not compatible. However, because these systems are based on poor representations of these conceptualizations, their sets of considered possible situations overlap. As a result, systems A and B agree exactly on situations that are neither admitted by C_A nor by C_B. In summary, although these systems appear to have a shared view of reality, the portions of reality that each of them aims to represent are not compatible. Therefore, the more it is known about a given domain and the more precisely it is represented, the bigger the chance of obtaining interpretations that are consistent with the reality of that domain and, therefore, of achieving semantic interoperability between the entities involved in these interpretations. Thus, Guizzardi (2005) concludes that, on the one hand, a modeling language should be sufficiently expressive to adequately characterize the conceptualization of the domain and, on the other hand, the semantics of the produced specifications should be clear, allowing users to recognize what language constructs mean in terms of domain concepts. Moreover, the specification produced by means of the language should facilitate the user in understanding and reasoning about the represented state of affairs.

In view of the different purposes, Guizzardi (2005, 2007) highlights that ontology engineering, analogous to software engineering and information systems, must include phases of conceptual modeling, design and implementation. Each phase has its specific objectives and thus requires different types of methods and tools to meet its particular characteristics. As mentioned, during a conceptual modeling phase, an ontology must strive for expressivity, clarity and truthfulness in representing the domain conceptualization. Therefore, the conceptual modeling phase requires specialized languages so as to create ontologies that approximate as closely as possible to the ideal representation of the domain. The same conceptual model can give rise to different implementation models in different languages, such as OWL and RDF, in order to satisfy different non-functional requirements, such as decidability and completeness. The section delimited as *Level* in Figure 3 illustrates this approach based on relations between conceptualization, abstraction, modeling language and model, shown in Figure 1-b. According to Guizzardi (2006), semantic web languages such as OWL and RDF are focused on computation-oriented concerns and are therefore inadequate for the conceptual modeling phase. Philosophically well-founded languages, on the other hand, are engaged in expressivity, conceptual clarity and domain appropriateness and are therefore suitable for this phase. To support his assertion, Guizzardi (2006) presents several problems of semantic interoperability from the use of semantic web languages in the representation of the domain and demonstrates how philosophically well-founded languages are able to address these problems.

As shown in Guizzardi (2005), while domain conceptualizations and, consequently, domain ontologies are established by the consensus of a community of users with respect to a material domain, a conceptual modeling language (which can be used to express these domain ontologies) must be rooted in a domain independent system of real-world categories, philosophically and cognitively well-founded, i.e. a foundational ontology. Foundational ontologies aggregate contributions from areas such as descriptive metaphysics, philosophical logic, cognitive science and linguistics. The theories inherent to these areas are called (meta-) conceptualizations and describe knowledge about reality in a way which is independent of language and particular states of affairs. A foundational ontology, in turn, is the representation of these theories in a concrete artifact. Thus, foundational ontologies, in the philosophical sense, can be used to provide real-world semantics for modeling languages as well as to constrain the possible interpretations of their modeling primitives, increasing the clarity of interpretation and, consequently, reducing ambiguities (which are key success factors in achieving semantic interoperability). Accordingly, it is possible to build domain ontologies by means of conceptual modeling languages based on foundational ontologies. In this sense, the *Meta-level* section in Figure 3, in addition to the *Level* section, represents the approach proposed by Guizzardi (2005, 2007).

An example of a foundational ontology is UFO (Unified Foundational Ontology). UFO was initially proposed in Guizzardi and Wagner (2004) and its most recent version is presented by Guizzardi et al. (2008). It is organized in three incrementally layered compliance sets: (i) UFO-A, which is essentially the UFO's core, defining terms related to endurants (objects, their properties etc); (ii) UFO-B, which defines as an increment to UFO-A terms related to perdurants (events etc); (iii) UFO-C, which defines as an increment to UFO-B terms explicitly related to the spheres of social entities.

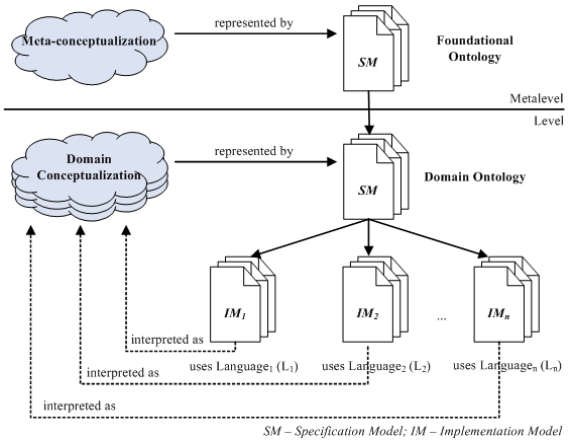

Fig. 3. Ontology engineering approach proposed by Guizzardi (2005, 2007).

As a conclusion, the ontology engineering approach described in this section considers the distinctions of the term ontology as well as their interrelationships, thereby establishing the phases, their respective objectives, as well as the methods and tools appropriate for the characteristic of each phase, allowing thus the construction of models capable of meeting the various purposes intended for them. Therefore, based on the study about IT service configuration management, as well as the study about ontology development, it is possible to construct an ontology of this domain as the objective of this chapter.

4. Conceptual model of the IT service configuration management domain

In considering the main research challenges, as well as the initiatives for solutions which are regarded state of the art, which properly lead the identification of gaps, as much as the appropriated approach to fulfil them, this section presents the conceptual models proposed in this work. On the basis of Figure 3, which shows the ontology development approach adopted in this work, the conceptual model proposed in this section concerns the domain ontology, whose conceptualization in discussion is the IT service configuration management. Regarding the foundational ontology, this work uses UFO, which represents the meta-conceptualization responsible for promoting the philosophical base of the work.

As discussed in Section 2, configuration management is responsible for maintaining information about configuration items and providing them to all the other management processes. In the context of IT service management and governance, configuration management must be able to answer questions such as: what are the business processes and how do they relate to the IT services and components? Based on this question and given that the main goal of this ontology is to describe a theory of the domain of IT service configuration management independent of specific applications, the defined competency questions reflect this intention. In this case, they lead to a mapping between IT and business concepts, as follows:

CQ1: How do the IT services and the business processes of an organization relate?

CQ2: How do the IT services and the IT components such as hardware and software relate?

To answer these questions, it is necessary to address others, such as: (i) what is an IT service? (ii) what is a business process? For this reason, Baiôco et al. (2009) propose an IT service configuration management ontology, which addresses such questions in order to provide a basis for the ontology presented in this chapter. As described through the SABiO method, if the domain of interest is too complex, a decomposing mechanism should be used in order to better distribute this complexity. In this case, a potentially interesting approach is to consider sub-ontologies. Therefore, to answer the competency questions, the following sub-ontologies were developed: (i) business process; (ii) IT service; (iii) IT component and lastly (iv) configuration item. These sub-ontologies complement each other in constituting the IT service configuration management ontology discussed in this work.

In terms of reusing existing ontologies, it is important to mention that besides the adopted literature, the conceptual modeling of this section also takes into consideration the discussions inherent to processes in general done in Falbo (1998) and Guizzardi et al. (2008), as well as the discussions inherent to IT services done in Calvi (2007) and Costa (2008). Still in line with the SABiO method, during the capture of the ontology the use of a graphic representation is essential to facilitate the communication between ontology engineers and domain experts. However, a graphical model is not enough to completely capture an ontology. This way, axioms should be provided to reinforce the semantics of the terms and establish the domain restrictions. Thus, the sub-ontologies developed in this work are connected by relations between their concepts and by formal axioms. Due to limitations of space, will be shown only those axioms also used in the next section. To distinguish the subject domain concepts and the UFO concepts, these last ones are presented in blank in the conceptual model that follows. Figure 4 presents part of the proposed ontology.

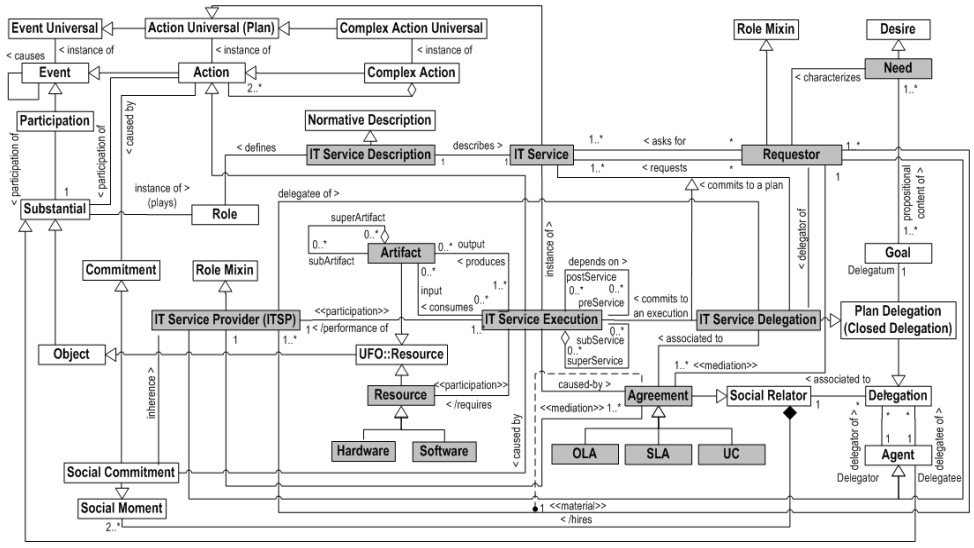

Fig. 4. Part of the IT service configuration management ontology.

According to the ITIL library, an IT service is a service which is provided by an IT organization to one or more clients. Therefore, in accordance with the UFO terminology, an IT service is characterized as a type of plan, i.e. an intentional event. As a result, the properties inherent to events are applied to IT services. As established in UFO, events are possible changes from one portion of reality to another, which means they can transform reality by altering the state of affairs from a pre-state to a post-state. Consequently they can produce, direct or indirectly, situations that satisfy the necessary conditions for other events to happen. On that account, events can cause other events, including then the service executions as represented in the model by the relation causes. An IT service execution (ITSE), in turn, denotes one or more particular actions that occur at specific time intervals, aiming to satisfy the propositional content of a commitment. On this note, an IT service execution is an action that instantiates a type of plan, in this case an IT service. This distinction, derived from the foundational ontology, provides greater adequacy to the domain, making it possible to distinguish services from their executions and allowing the comparison, for example, between achieved and planned results.

Because it denotes one or more actions, an IT service execution can be atomic or complex. As a complex execution, it is decomposed into other smaller service executions, termed subservices. In this way, a subservice is a service execution that is part of a bigger service execution, its super-service. As the properties inherent to events are applied to the IT services, the decomposition of service executions, as any other event decomposition, is characterized as a transitive, asymmetric and irreflexive relation. This inheritance of properties from the foundational ontology facilitates modeling decisions and minimizes the possibilities of incoherent descriptions of the domain.[1]

According to the ITIL library, an IT service execution aims to produce resources in order to satisfy the needs of its customers. On the conceptual model proposed in this section, the produced resources are said to be outputs. On the other hand, an IT service execution can consume resources, seen as raw material, to produce results. On the model, these resources are said to be inputs. From the point of view of UFO, an artifact of a service execution is a type of resource (UFO::Resource) which in turn is mapped to the notion of an object. As such, the subartifact and superartifact relations are then governed by the axioms defined for the (different types of) parthood relations between substantials, as described in Guizzardi (2005).

According to UFO, a resource (UFO::Resource) is a role that an object plays in an event. Thus, the artifacts of a service execution are roles played by objects in the scope of this service instance. This being said, it is important to highlight a contribution from UFO attributed to the model. As the notions of objects and roles are defined, it becomes possible to represent real situations of the domain, as with those where the same object plays the role of an output to a service execution and input to another, distinguishing only the type of participation performed by the object and keeping its identity throughout its existence. This is because, according to UFO, an object is a type of endurant which, in contrast to a

[1] UFO makes explicit distinctions often ignored by many languages. For example, while it is possible to consider that an event *x* is a part of an event *z* because *x* is a part of an event *y* that is a part of *z*, it is not the case that the musician's hand (and so a part thereof) is a part of the band within which the musician is a part. In the first case, there is transitivity, but in the latter this does not exist. In this sense, parthood relations denote distinctions that should be considered.

perdurant (e.g. an event), is an individual that keeps its identity throughout its existence. On the model, the participation of the output is represented through the relation "produces" while the participation of the input is represented through the relation "consumes".

As objects, the participation of artifacts in a service execution should correspond to the types of participation of objects in an action, defined in UFO. In fact, being produced or consumed does not express the exact notion on the participation of an object in a service execution. Namely, outputs can be created or modified. Inputs, on the other hand, can be used, modified into new products or else terminated. As such, the relation "produces", as well as the relation "consumes", designates distinct notions which should be considered. Therefore, in light of the UFO terminology and the subject domain, the relation "produces" denotes the participation of creation or change, while the relation "consumes" indicates the participation of usage, change or termination as defined in UFO. This demonstrates the foundation of the domain concepts and relations in terms of philosophically well-founded concepts and relations, making the representation of the universe of discourse even more clear, expressive and coherent with reality.

As described by the ITIL library, an IT service is based on the use of the information technology, which includes, among other things, IT components such as hardware and software. Therefore, an IT service execution, as any other activity, presumes the use of resources, in particular IT components, in order to achieve results. Essentially, under the UFO's perspective, the resources of a service execution are types of resources (UFO::Resource), i.e. objects participating in an action. Therefore, not only artifacts but also resources are roles played by objects within the scope of a service execution. Again, the distinction between objects and roles contributes to the representation of real situations in the universe of discourse, including those where the same object is produced by a service execution but is required by another, though remaining as the same individual. In the model, the participation of a resource is represented by the relation "requires". Once these resources are used as support tools in a service execution, the foundation associated with this relation leads to only one of the types of resource participation defined in UFO, i.e. the usage participation.

In considering the definitions of artifacts and resources, especially regarding the foundation provided by UFO, it is noted that the inputs and the resources are objects which can play the same type of participation in a service execution, i.e. the usage participation. However, taking a service execution as a transformation primitive, inputs indicate raw materials which are incorporated into the product. Resources, on the other hand, refer to components that support a service execution, but are not intended as products of this execution. Thus, the resources employed in a service execution cannot be considered as products within the scope of this execution. Therefore, by clearly representing the participation of inputs and resources in a service execution, this foundation promoted the identification with regard to the similar type of participation of these roles. As a consequence, it required the use of domain definitions able to characterize such roles, since fundamentally they are similar. Hence, as pointed out, this observation becomes evident, in this case, especially due to the foundation of the domain concepts in terms of UFO. This demonstrates the contribution of a foundational ontology, in general, and of UFO, in particular, in supporting the construction of appropriate models regarding domain and comprehension.

As described in Costa (2008), an IT service execution is an intended and orderly execution of one or more actions in order to satisfy the propositional content (goal) of a commitment agreed with an agent. This ordering, in some cases, is governed according to certain situations that culminate into the execution of the service. In this sense, the causality relation between events leads to a dependence relation between them. This happens because, by changing the state of affairs of the reality from one state (pre-state) to another (post-state), events can generate (directly or indirectly) a situation which satisfies a necessary condition for other events to occur. In summary, an event depends on another if the first is caused, directly or indirectly, by the second. Hence, service executions can depend on events (or on other service executions) in order to occur. It is relevant to mention that artifacts and resources can imply a dependency relation between service executions. In fact, as described in UFO, situations are complex entities that agglutinate other entities (including objects) and denote the pre- and post-state of an event. So, as a type of object, artifacts and resources are present in situations that symbolize the pre- and post-states of service executions. As an example, a service may require resources or consume inputs that are products of other services, characterizing thus a dependence relation between them because of their respective objects. Besides this, it is important to highlight that if an event depends on the other then there is a temporal relation between them that should be taken into account. Concerning dependence between service executions, this temporal relation is represented by the pre-service and post-service relations. As the causality relation between events, the dependence relation between them (including the IT service executions) is transitive, asymmetric and irreflexive. These properties are also valid for the relations pre- and post-service. Temporal relations between events make possible to define the temporal ordering in which the events (including the service executions) are submitted and, therefore, establish the order in which they occur, even when there is no dependence relation defined between them. This is because, according to UFO, events are framed into temporal intervals, from which originate the temporal relations. Thus, the model proposed in this section is based on a framework of concepts and relations defined in UFO which makes possible to specify the flow in which the events are associated and, consequently, the ordering associated with the IT service executions. In this way, this structure supports the modeling decisions at the same time as it contributes to the creation of a more expressive, clear and truthful model with regard to the universe of discourse.

An IT service is described by a normative description, termed IT service description. The description of an IT service, for instance, describes the roles played by each agent in a service execution. Agents, as well as objects, are substantial from the UFO point of view. However, agents differ from objects because of the fact that they can possess beliefs, desires and intentions. Intentions are characterized as desired states of affairs for which the agent commits itself to pursuing, i.e. an internal commitment. For this reason, intentions cause the agent to perform actions. In this sense, the participation of an agent in an action is characterized as an action contribution. Consequently, service executions are performed by agents. Indeed, the action contribution of an agent in a service execution is caused by a social commitment of the agent in performing this service execution (or part thereof, the subservice) with its consequent permissions and obligations. Therefore, the role modeling pattern described by UFO applies to the IT service domain. In fact, as advanced in Guizzardi (2006), as a domain independent knowledge representation language, foundational ontologies in general and UFO in particular aim to support the construction

of domain-dependent models by acting as a reference framework and thus guiding modeling decisions and allowing the creation of models that clearly and accurately represent, as much as possible, the real situations in a universe of discourse. This pattern has resolved various role modeling problems in the literature. Thus, the execution of a service instance is characterized by an agent playing a role in an IT service, in this case, acting as a service provider. It is worth noting that not all types of agents are responsible for the service executions. According to ITIL, an IT service is one provided to one or more customers by an IT organization. On the other hand, not all the instances of service providers are IT organizations, but possibly another type of agent. Therefore, the execution of an IT service occurs by means of an IT organization acting as a service provider. Nonetheless, given that IT organizations can play other roles, such as the role of a customer, it is not the case that all instances of IT organizations act as service providers in every situation, but only when they perform IT services. So, the relation between service provider and IT organization cannot be direct in any sense. On the contrary, it should be intermediated by a role (in this case, IT Service Provider - ITSP) that aggregates the criteria of identity of the species (in this case, IT Organization) and performs the mixed role (in this case, Service Provider). In this way, it is defined that not all service provider is characterized as an IT organization, which may be another type of agent. Furthermore, it is defined that there are IT organizations which in certain circumstances are service providers but not in others, possibly playing other roles. The responsibility of each played role is described by the IT service description.

The execution of an IT service occurs by means of a request, which is motivated according to the requestor's needs. Therefore, a customer is a type of requestor, i.e. an agent who requests an IT service. The process of requesting an IT service, as well as other concepts inherent to the domain of service-level management, is discussed and modeled in Costa (2008). In summary, the author describes that an IT service is appropriate for a certain need when the product from its execution satisfies the requestor's requirements, i.e. its needs. As such, an IT service can achieve a need if, and only if, the post-state of an occurrence of this service is a situation that satisfies the propositional content of the referred need, as formalized by the axiom A1.

$$\text{(A1) } \forall x,y \text{ (IT-Service}(x) \wedge \text{Need}(y) \wedge \text{can-achieve}(x,y) \leftrightarrow \exists a,b,c \text{ (ITSE}(a) \wedge \text{Situation}(b) \wedge \text{Proposition}(c) \wedge \text{instance-of}(a,x) \wedge \text{postState}(b,a) \wedge \text{satisfies}(b,c) \wedge \text{propositional-content-of}(c,y)))$$

An IT service is requested by means of a document that describes the need of the requestor, termed service-level requirement (SLR). Thus, given a SLR that describes a certain need which can be achieved by an IT service, then this SLR is used to request this service (COSTA, 2008). This definition is formalized by the axiom A2.

$$\text{(A2) } \forall x,y,z \text{ (IT-Service}(x) \wedge \text{Need}(y) \wedge \text{SLR}(z) \wedge \text{can-achieve}(x,y) \wedge \text{describes}(z,y) \rightarrow \text{used-to-ask-for}(z,x))$$

So, through a SLR, a requestor can search for an IT service that satisfies its needs. This allows the requestor to find out whether or not there are services that can achieve its needs and request, in case there is, the most adequate, according to its demand. This definition is formalized by the axiom A3.

$$(A3)\ \forall x,y,z,w\ (\text{IT-Service}(x) \wedge \text{Need}(y) \wedge \text{SLR}(z) \wedge \text{Requestor}(w) \wedge \text{can-achieve}(x,y) \wedge \text{describes}(z,y) \wedge \text{used-to-ask-for}(z,x) \wedge \text{characterizes}(y,w) \rightarrow \text{asks-for}(w,x))$$

Once the requester asks for services that can achieve its necessities and finds such services offered by the providers, this requestor is able to request such services. According to Costa (2008), when an IT service is requested, the requestor hires the provider responsible for this service, as formalized by the axiom A4.

$$(A4)\ \forall x,y,z\ (\text{Requestor}(x) \wedge \text{ITSP}(y) \wedge \text{IT-Service}(z) \wedge \text{asks-for}(x,z) \wedge \text{requests}(x,z) \wedge \text{provides}(y,z) \rightarrow \text{hires}(x,y))$$

According to the ITIL library, the provision of an IT service to a requestor is mediated by an agreement. Thus, once the requesting process has been established, there will be an agreement that will mediate all the IT service provision, establishing the claims and obligations related to both parts, the requester and the provider. This relation is formalized by the axiom A5.

$$(A5)\ \forall x,y\ (\text{Requestor}(x) \wedge \text{ITSP}(y) \wedge \text{hires}(x,y) \rightarrow\ \exists z\ (\text{Agreement}(z) \wedge \text{mediates}(z,x) \wedge \text{mediates}(z,y)))$$

With regard to UFO, an agreement is a type of social relation. A social relation is composed of social moments, called claims and commitments. A social commitment is a type of commitment and thus the motivating cause of an action performed by an agent. Thus, the agreement established between the requestor and the provider will cause the execution of the service requested by the requestor, for it is composed of social commitments inherent to the provider.

The requesting of a service is motivated by a need of a requestor and it is fulfilled through a service execution performed by a provider. In this context, there is a relation of dependence between the requestor and the provider. From the UFO point of view, a dependence relation between agents leads to a delegation relation. According to UFO, an agent *a* depends on an agent *b* regarding a goal *g* if *g* is a goal of agent *a*, but *a* cannot achieve *g* and agent *b* can achieve *g*. This matter may be the reason why agent *a* decides to delegate such goal achievement to agent *b*. A delegation is thus associated with a dependency, but it is more than that. As a material relation, it is founded on more that its connected elements. In this case, the connected elements are two agents, namely, the requestor (delegator) and the provider (delegatee), as well as a goal (delegatum), which is the delegation object that represents the needs of the requestor. The foundation of this material relation is the social relator (a pair of commitments and claims), i.e. the agreement established between the two agents involved in this delegation, entitled requestor and provider. In other words, when a requestor delegates a goal to a provider, besides the fact that the requestor depends on the provider with regard to the goal, the provider commits itself to achieving the goal on behalf of the requester. This commitment is established by means of an agreement. As described in Calvi (2007), an IT service delegation (ITSD) is a delegation committed to achieve a goal according to a specific plan, the IT service. Therefore, according to UFO, an IT service delegation is a closed delegation, since it describes a specific plan which the provider should adopt to achieve the goal delegated by the requestor. In summary, if there is an agreement between the requestor and the provider which causes the execution of a service, then there will be a service delegation associated with the agreement. In addition, this delegation will

be committed with the service and consequently with its execution. In this context, the requestor assumes the role of delegator while the provider assumes the role of delegatee. This relation is formalized by the axiom A6.

(A6) $\forall x,y,z$ (Requestor(x) $\wedge$ ITSP(y) $\wedge$ Agreement(z) $\wedge$ mediates(z,x) $\wedge$ mediates(z,y) $\rightarrow \exists w$ (ITSD(w) $\wedge$ associated-to(w,z) $\wedge$ delegator-of(x,w) $\wedge$ delegatee-of(y,w)))

As defined by UFO, when an agent is said to be able to achieve a certain goal it means that such agent can achieve this goal by itself or else delegate it to another agent that would be able to achieve it on its behalf. Thus, when a provider receives a delegation by the means of a service-level agreement, this provider analyses the delegated service and, if needed, delegates that service to other service providers, termed internal providers (e.g. an IT infrastructure department) and external providers (e.g. suppliers). In this manner, each subservice execution contributes to the service delegated by the customer, in this case, the super-service.

In view of this discussion, it is defined the relationship between business process and IT service inherent to the competency question CQ1. In summary, information technology is frequently used to support the business process activities through IT services. Given that a business process activity is an activity that is owned and performed by the business (commonly by a business unit) and an IT service is a service provided by an IT organization, the relation between business process and IT service occurs as a result of the dependence relation between the respective agents, which means business unit and IT organization. In other words, the fact of a certain business process activity occurrence being performed by the business denotes a social commitment of this agent in performing this occurrence. This social commitment contains a propositional content, i.e. a goal. Thereby this agent has a commitment to perform a certain action which satisfies this goal. In case this agent, i.e. the business, needs an IT service to achieve such a goal, this agent can delegate the goal (or part thereof) to the agent responsible for providing the IT service. In this context, the agent responsible for the business process activity assumes the role of requestor (more specifically, the role of customer) and the agent responsible for providing the service assumes the role of IT service provider. Therefore, the relation between business process and IT service is derived from the dependency relation between its respective agents, leading to a delegation relation. In this sense, it is worth noting the UFO contribution as it fundaments the entire service delegation process. Regarding the relationship between IT services and IT components, which is inherent to the competence question CQ2, the discussions during this section have defined that such components are seen as resources under the UFO point of view, assuming the usage participation. Moreover, in considering the definition of hardware and software presented by IEEE 610.10 (IEEE, 1994) as well as by ISO/IEC 2382-1 (ISO/IEC, 1993) it is possible to conclude that a hardware is a physical component that processes the instructions described by a software. In terms of UFO, a software is a type of normative description which describes a computer process, i.e. a type of event (an event universal). An instance of this process denotes an occurrence of such a process, termed computer process occurrence, i.e. a predetermined course of events whose execution includes the participation of a hardware. In this sense, the concept of hardware, as well as of software, is mapped to the notion of substantial. As such, if a service execution requires a software resource which is processed by a hardware resource, then this service execution also requires this hardware resource. In other words, if a service execution x requires a software resource y and this last

resource describes a computer process *z* whose occurrence *w* includes the participation of a hardware *q*, then the service execution *x* also requires the hardware resource *q*, as formalized by the axiom A7.

$$\text{(A7) } \forall x,y,z,w,q \text{ (ITSE(x)} \wedge \text{Software(y)} \wedge \text{Computer-Process(z)} \wedge \text{Computer-Process-Occurrence(w)} \wedge \text{Hardware(q)} \wedge \text{requires(x,y)} \wedge \text{describes(y,z)} \wedge \text{instance-of(w,z)} \wedge \text{participation-of(w,q)} \rightarrow \text{requires(x,q))}$$

In this sense, it is useful to note the UFO contribution, as it allows the distinction between the notions pertinent to software, its processes and its occurrences, as well as the proper participations of hardware components, granting more expressiveness, clarity and veracity to the model. Indeed, the use of appropriate tools such as UFO and methods such as SABiO promoted o development of important ontological distinctions, as discussed in this section. Considering the ontology engineering approach adopted in this chapter, the next section explores the benefits from the implementation of the conceptual model.

5. An implementation and application of the proposed ontology

In view of the contributions that ontologies provide towards automated solutions, including important features such as artificial intelligence and, above all, interoperability, this section aims to present a case study in order to: (i) perform a proof of concept of the models developed in the previous section and also (ii) demonstrate how the concepts modeled by using ontologies can be implemented and applied in a computational system, in order to enable automation, including important features such as cited in this paragraph. Taking into accounting Figure 3, which illustrates the ontology development approach adopted in this work, the ontology proposed in this section concerns to the implementation model, which represents an implementation of the conceptual model proposed in the previous section.

As discussed in Guizzardi (2005, 2007), each ontology engineering phase requires the use of appropriate languages in relation to the context within which the model is being designed. In the implementation phase, the choice of a language must be conducted by the end-application requirements. This refers to languages focused on computational requirements, such as decidability and efficient automated reasoning. In terms of the configuration management process, factors such as decidability, completeness and expressiveness are considered to be key requirements because of its role in relation to all other processes in service management. Thus, for the implementation of the conceptual models, this work used the OWL DL sublanguage, since it allows a greater degree of expressiveness, as compared with OWL Lite, while maintaining computational guarantees such as completeness and decidability, features not guaranteed by OWL Full (Bechhofer et al., 2004). In addition to OWL DL, the implementation of the models also used SWRL (Horrocks et al., 2003). The SWRL language allows the representation of the axioms defined in the conceptual models presented in Section 4 in an integrated way with the concepts and relations implemented by means of OWL. Finally, for the implementation of the models in OWL and the definition of the axioms in SWRL, this work used the Protégé tool (Protégé, 2011), an ontology editor that enables the integration of different languages such as OWL and SWRL inside the same implementation environment.

Once defined the development environment, the implementation models were developed according to the modular structure of the conceptual models, as follows: (i) UFO.owl; (ii) BusinessProcess.owl; (iii) ITService.owl; (iv) ITComponents.owl and finally (v) ConfigurationItem.owl. Due to the expressivity restrictions inherent in the implementation languages, the main issue concerning the mapping from conceptual models into implementation models is related to the treatment of the reduction in semantic precision. In order to maintain this reduction at an acceptable level, the most relevant losses that were found were related to the transformation of all ontologically well-founded concepts and relations into OWL classes and properties, respectively. Regardless of the application scenario, this mapping must consider the information contained in the notation used for the development of the conceptual models, such as cardinality, transitivity, domain and range. With respect to cardinality and transitivity, in OWL it is not possible to represent them simultaneously (Bechhofer et al., 2004). As a result, this work considers that the representation of cardinality restrictions is more relevant to the implementation models developed in this section. In addition, to represent the cardinality restrictions in both directions inverse relations were used. For instance, the relation "requests" is represented by the pair of relations "requests" and "is requested by". However, according to Rector and Welty (2001), the use of inverse relations significantly increases the complexity of automated reasoning. Thus, they should be used only when necessary. With respect to domain and range, an issue that should be considered is how to organize and represent many generic relations. For example, if a generic relation "describes" is created, it is not possible to restrict the domain and the range. In this case, the design choice was to use specific relations like "describes_Software_ComputerProcess", which is represented as a sub-relation of a generic relation "describes". Finally, with respect to SWRL restrictions, this language has neither negation operators nor existential quantifiers (Horrocks et al., 2003). In addition, the SWRL language might lead to undecidable implementation models. Nevertheless, this issue may be worked around by restricting the use of rules and manipulating only those that are DL-safe (Motik et al., 2005). As an attempt to make this tangible, consider an implementation of the axiom A7, which concerns the competence question QC2, discussed in Section 4. This implementation is represented by the rule R7a.

(R7a) IT_Service_Execution(?IT-SERVICE-EXECUTION) ∧ Software(?SOFTWARE) ∧ ComputerProcess(?COMPUTER-PROCESS) ∧ ComputerProcessOccurrence(?COMPUTER-PROCESS-OCCURRENCE) ∧ Hardware(?HARDWARE) ∧ requires_ITServiceExecution_Resource(?IT-SERVICE-EXECUTION,?SOFTWARE) ∧ describes_Software_ComputerProcess(?SOFTWARE,?COMPUTER-PROCESS) ∧ isInstanceOf_ComputerProcessOccurrence_ComputerProcess(?COMPUTER-PROCESS-OCCURRENCE,?COMPUTER-PROCESS) ∧ hasParticipationOf_ComputerProcessOccurrence_Hardware(?COMPUTER-PROCESS-OCCURRENCE,?HARDWARE) → requires_ITServiceExecution_Resource(?IT-SERVICE-EXECUTION,?HARDWARE)

The axiom A7 constitutes the set of axioms that establishes the relationship between the computational resources that are required by an IT service execution as a response to the competence question QC2. Thus, this axiom involves concepts such as IT service execution, hardware and software, as well as the interrelationship between these concepts, such as the

relations "requires" and "describes". As discussed earlier, concepts are implemented as classes, while relations are implemented as properties, according to OWL. According to described throughout this work, the same conceptual model can give rise to a variety of implementation models in order to meet different requirements, in accordance with the purpose of the application scenario. The rule R7a is intended to meet configuration management activities, especially the activities of identification and inference of managerial information. In Baiôco and Garcia (2010), the axiom A7 is implemented mainly in order to meet the latter activity. In this case, the axiom is implemented as formalized by the rule R7b.

(R7b) IT_Service_Execution(?IT-SERVICE-EXECUTION) ∧ Software(?SOFTWARE) ∧ ComputerProcess(?COMPUTER-PROCESS) ∧ ComputerProcessOccurrence(?COMPUTER-PROCESS-OCCURRENCE) ∧ Hardware(?HARDWARE) ∧ requires_ITServiceExecution_Resource(?IT-SERVICE-EXECUTION,?SOFTWARE) ∧ describes_Software_ComputerProcess(?SOFTWARE,?COMPUTER-PROCESS) ∧ isInstanceOf_ComputerProcessOccurrence_ComputerProcess(?COMPUTER-PROCESS-OCCURRENCE,?COMPUTER-PROCESS) ∧ hasParticipationOf_ComputerProcessOccurrence_Hardware(?COMPUTER-PROCESS-OCCURRENCE,?HARDWARE) → query:select(?IT-SERVICE-EXECUTION,?SOFTWARE, ?HARDWARE) ∧ query:orderByDescending(?IT-SERVICE-EXECUTION)

There are numerous contributions offered by ontology engineering for the construction of autonomous, intelligent and above all interoperable computational applications. Although done in a different area, Gonçalves et al. (2008) presents an application for the interpretation of electrocardiogram results where the use of an ontology model provides a graphical simulation of the heart behavior of an individual and the correlation of the heart behavior with the known pathologies. Regarding configuration management, this process identifies, controls, maintains and checks the versions of the existing configuration items and reports the information of the IT infrastructure to all those involved in the management. Thus, this section aims to demonstrate how implementation models can be applied in a computational environment in order to support management activities in an automated manner. In addition, the results will provide a proof of concept of the developed ontology.

The first part comprises the mapping between business processes activities and business units responsible for these activities. In addition, it includes the needs that characterize these business units. Figure 5-a shows, for example, that the business process activity BPAO_Sales is composed of the activity BPAO_Ordering, which is owned by the business unit BU_Sales, as shown in Figure 5-b. The business unit BU_Sales, in turn, has need inherent to this activity, as presented in Figure 5-c. It is worth mentioning that such information, as well as any assertion that appears highlighted in blue, concerns information previously inserted into the implementation models. On the other hand, assertions that appear highlighted in yellow are information automatically inferred by the implementation models, which denote knowledge acquisition. Such inferences are performed by the Pellet reasoner (Sirin et al., 2007).

As discussed in Section 4, IT services can achieve business needs by supporting its activities. Thus, Figure 5-d illustrates the IT services that can achieve the needs of the business. This information is inferred by executing the axiom A1, implemented in this section. Figure 5-d illustrates, for example, that the service IT_Service_Ordering can achieve the need

Need_Ordering. Figure 5-e, in turn, presents the requirements used for service requests. This inference is performed by executing the implemented axiom A2. This scenario motivates the requestor, in this case the business unit, to ask for services which can achieve its needs, as shown in Figure 5-f. This inference is performed by means of the implemented axiom A3. It should be mentioned that information of this nature is fundamental to processes such as service level management, which interacts with configuration management in requesting information in order to find services able to meet the needs of the requestors.

According to the axiom A4, once a service capable meeting the need is found, the requestor can then initiate the delegation process by contracting the service. Thus, Figure 5-g shows the provider hired by the business unit. As shown in Figure 5-g, the provider hired by the business unit BU_Sales is the IT_Department, because it is the provider responsible for providing the requested service. As discussed in Section 4, the hiring process is mediated by an agreement, as illustrated in Figure 5-h. This inference is performed by means of the implemented axiom A5. Figure 5-h illustrates, for example, that the agreement SLA_Ordering mediates the requestor BU_Sales and the provider IT_Department. This agreement, in turn, characterizes the delegation process, which has a delegator (in this case the requestor) and a delegatee (in this case the provider), as shown in Figure 5-i. This inference is performed by means of the implemented axiom A6. Figure 5-i illustrates, for example, that the delegation ITSD_Ordering is associated to the agreement SLA_Ordering and has as the delegator the BU_Sales and as the delegatee the IT_Department.

As described in Section 4, the hired provider receives the service delegation from the requestor and provisions the necessary resources for the service execution. In this sense, the received service execution is characterized as a complex action which is delegated to the support groups by means of subservices. In this context, the hired provider, in a manner similar to that of the business unit, plays the role of requestor and the support groups, in turn, play the role of service provider. Thus, Figure 5-j shows, for instance, that the execution ITSE_Ordering, which represents an instance of the service IT_Service_Ordering, is composed of the sub-executions ITSE_Ordering_Processing and ITSE_Ordering_Printing. The delegation performed by the provider to the support groups is mediated by agreements. Figure 5-k shows, for example, that the agreement OLA_Ordering_Processing mediates the delegation process of the sub-execution ITSE_Ordering_Processing between the IT_Department and the IT_Department_System. In this case, IT_Department plays the role of delegator while the support groups play the role of delegatee, as shown in Figure 5-l.

IT services are based on the use of information technology. Thus, Figure 5-m relates the software required by each service execution. Figure 5-m shows, for example, that the execution ITSE_Ordering_Processing requires the software Software_Ordering_Processing. As discussed in Section 4, software is processed by hardware. Thus, Figure 5-n presents the hardware associated with the processing of the concerned software. In addition, Section 4 states that if a service execution requires a software and this software is processed by a hardware, then this service execution also requires this hardware, as illustrated in Figure 5-o. This inference is performed by means of the implemented axiom A7. Figure 5-o illustrates, for example, that the execution ITSE_Ordering_Processing requires the hardware Hardware_Sales, since such hardware processes the software Software_Ordering_Processing (as shown in Figure 5-n) required by such an execution.

Thus, this case study concludes the mapping between business and IT. This mapping is fundamental to other management processes. As an illustration, the configuration management process of this case study is able to correlate and determine that a particular event of unavailability on the hardware Hardware_Sales affects the software Software_Ordering_Processing, which is used by the service execution ITSE_Ordering_Processing. This execution is part of the execution ITSE_Ordering which is instance of the service IT_Service_Ordering and, in turn, supports related activity of the business process activity BPAO_Sales. This correlation, provided by the implementation model, is the basis for activities such as: (i) event correlation in event management; (ii) workaround identification in incident management; (iii) root cause analysis in problem management and (iv) impact analysis in change management.

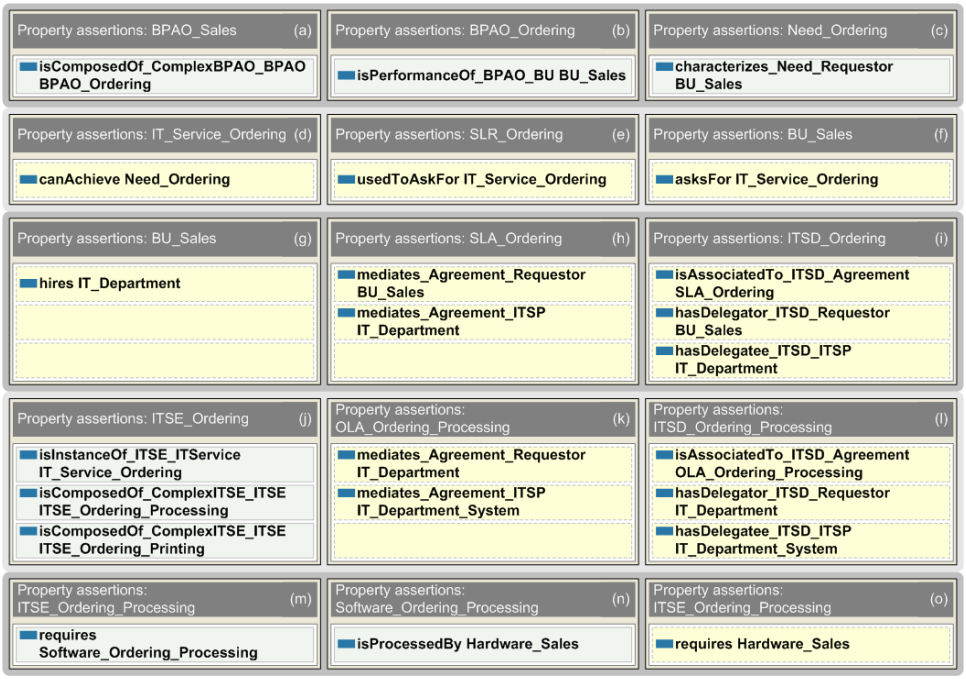

Fig. 5. Application of the implementation model.

To complete this case study, consider intelligent software agents playing the roles of requestor and provider and, consequently, negotiating the provision of services that meet the needs of the environment. Regarding the role of provider, such activities refer to the various management disciplines, as presented throughout this section. In particular, this scenario denotes implementation models subsidizing paradigms known as autonomous networks. In general, it denotes implementation models promoting automation in various areas of interest.

6. Conclusion

As discussed in this chapter, automation enables organizations to explore opportunities as well as supporting challenges in an effective and efficient way. The important role played by

IT as an instrument for automation has made automation increasingly dependent on IT, consequently rising the demands for advances, as observed by the growing challenges arising from the conception of systems continuously more complex, intelligent and, above all, interoperable. Moreover, the need for an efficient and effective IT management has grown substantially, as evidenced by widespread adoption of innovative best practices libraries and standards. For this reason, this work presented an ontology of IT service configuration management. The objective was not only to adopt the state of the art in order to address key research challenges in IT management, but also to foment novel approaches which can be applied in IT in various areas of interest.

The diverse uses attributed to the ontologies in computer science and the interrelation between their purposes promote the search for approaches capable of providing the construction of ontological models able to achieve the various objectives assigned to them. Based on innovative and high quality research initiatives, this chapter discussed about a systematic approach for building ontologies known as Ontology Engineering. In considering the various uses and purposes, as well as their interrelationships, these initiatives attempt to establish a structured means of development as an alternative to the various *ad hoc* approaches that characterize the current developments and imply in models unable to achieve their goals. In summary, this approach allowed the development of conceptual models which are application-independent artifacts and, as a result, it enabled their use as a reference ontology for the subsequent development phases, deriving implementation models in order to address the different purposes of end applications.

According to Guizzardi and Halpin (2008), the practice of conceptual modeling is permeated by philosophical questions. This demonstrates the need for an appropriate theoretical foundation for conceptual modeling languages so as to ensure that the quality requirements of domain and comprehensibility appropriateness can be fulfilled by the produced conceptual models. In this sense, they advance that philosophically well-founded ontologies play a key role in this initiative. They complement this line of reasoning by citing Guarino and Guizzardi (2006) and emphasizing that although typical conceptual modeling languages provide facilities for structuring domain elements, such as taxonomies and data value structures, the justification for the validity of many structuring choices, as much as the justification for the grammar of many natural language sentences, can only be made on ontological grounds, in this sense, on a philosophical basis. As a final consideration, Guizzardi and Halpin (2008) point out that philosophical foundations are vital components with respect to conceptual modeling, in general, and domain ontology engineering, in particular, as mature disciplines with sound principles and practices. Thus, in quoting the physicist and philosopher of science Mario Bunge, "every science presupposes some metaphysics", they conclude that a scientific field can either choose to develop and make explicit its philosophical foundations or to remain oblivious to its inevitable and often *ad hoc* ontological and epistemological commitments.

Accordingly, in addition to the appropriate methods and techniques, such as the SABiO method, this approach used a philosophically well-founded ontology, termed UFO. The SABiO method provided a systematic approach that led the development of ontology proposed in this chapter, describing an iterative process, closely related to evaluation. Thus, with emphasis on the concept of competence questions, the SABiO method provided a means for defining the scope and purpose of the ontology, leading its capture and

formalization, the reuse of existing ontologies, as well as its evaluation and documentation, enabling the ontology proposed in this chapter to adequately meet the requirements for which it was designed, as confirmed by proof of concept. The UFO ontology, in turn, was useful in building a conceptual model committed to maximizing the expressivity, clarity and truthfulness of the modeled domain. These characteristics are key quality attributes of a conceptual model, responsible for its effectiveness as a reference framework for the tasks of semantic interoperability and reuse. In fact, as a knowledge representation language philosophically well-founded, the UFO ontology aims to provide a sound basis for the representation of a conceptualization and therefore inhibit arbitrary descriptions of concepts and relationships of a universe of discourse. As discussed during the development of the conceptual models proposed in this chapter, the UFO ontology guided diverse modeling decisions, contributing to the derivation of new knowledge or the identification and elucidation of ambiguous and inconsistent representations of the domain, often represented in various literatures.

As discussed in this chapter, the more it is known about a universe of discourse and the more precisely it is represented, the bigger the chance of producing models that reflect, as much as possible, the appropriate conceptualization of the domain. In this sense, besides the use of appropriate languages, such as ontologies philosophically well-founded, as a common and shared specification, it is important that the specification of the domain considers appropriate literatures and, especially, that it considers its main concepts and relationships as well as application independence. Considering the universe of study of this chapter, these aspects associated with the use of appropriate methods and tools enabled the development of models able to maximize the alignment between IT and business for humans and computers. Moreover, these aspects allude to a point of view which should be mentioned. By maximizing the capacity of a model in acting as a common and shared source about a universe of discourse, conceptual models, representing norms and standards, can be potentially used as an addendum to such literatures. This is because, in general, these libraries are described in natural languages, which are susceptible to ambiguities and inconsistencies, as opposed to conceptual models, which are formally described.

Considering the importance of automation, as well as the contributions that an implementation provides in terms of ontology evaluation, it was developed an implementation model, derived from the conceptual model proposed in this chapter. In addition, by applying the entire approach discussed in this chapter, it is possible to attest it, as well as making it more tangible, promoting its benefits. It should be mentioned that the development of conceptual models followed by the development of implementation models became evident the distinction between ontology representation languages, as discussed throughout this chapter. This demonstrates that the approach adopted in this chapter shows itself appropriate by considering the various uses and purposes assigned to ontologies. In this way, despite the expressivity restrictions inherent in implementation languages, it was possible to perform a proof of concept of the ontology developed in this chapter as well as demonstrating how such models can be derived and implemented in computing environments with a view to the different computational requirements. In particular, it was possible to show how implementation models can support automation, including special characteristics such as knowledge acquisition and interoperability. From this point of view it is important to note that the concepts inherited from UFO are important in promoting paradigms such as artificial intelligence by making explicit, for computational agents,

concepts that express the daily lives of human agents, such as intentions, goals and actions. These factors are especially important in face of the increasing need for integration of complex as well as heterogeneous systems and also when considering autonomous systems.

Therefore, the contributions of this chapter are not restricted to the domain of configuration management. Instead, they promote semantic interoperability in IT in diverse areas of interest, maximizing the advances in automation. Additionally, this chapter makes possible other researches. In this sense, future works include: (i) the extension of the ontology for covering other business-driven IT management concepts, such as IT services metrics and business measures as well as their relationships, improving the alignment between these two domains; (ii) the extension of the ontology in order to cover other configuration management concepts, such as baseline, version and variant; (iii) the extension of the ontology to cover other management process, such as change management and release management; (iv) the application of conceptual models as an addendum to norms and standards; (v) the application of the implementation models, especially with techniques like artificial intelligence, promoting paradigms such as autonomous networks.

7. References

Baiôco, G., Costa, A.C.M., Calvi, C.Z. & Garcia, A.S. (2009). IT Service Management and Governance – Modeling an ITSM Configuration Process: a Foundational Ontology Approach, In *4th IFIP/IEEE International Workshop on Business-driven IT Management, 11th IFIP/IEEE International Symposium on Integrated Network Management*, 2009

Baiôco, G. & Garcia, A.S. (2010). Implementation and Application of a Well-Founded Configuration Management Ontology, In *5th IFIP/IEEE International Workshop on Business-driven IT Management (BDIM), 12th IFIP/IEEE Network Operations and Management Symposium (NOMS)*, Osaka, 2010

Bechhofer, S. et al. (2004). OWL Web Ontology Language Reference, In *W3C Recommendation*, Oct 2011, Available from http://www.w3.org/TR/owl-ref/

Brenner, M., Sailer, M., Schaaf, T. & Garschhammer, M. (2006). CMDB - Yet Another MIB? On Reusing Management Model Concepts in ITIL Configuration Management, In *17th IFIP/IEEE Distributed Systems Operations and Management (DSOM)*, 2006

Calvi, C. Z. (2007). IT Service Management and ITIL Configuration Process Modeling in a Context-Aware Service Platform (in Portuguese), Master Dissertation, UFES, 2007

Costa, A.C.M. (2008) ITIL Service Level Management Process Modeling: An Approach Using Foundational Ontologies and its Application in Infraware Platform (in Portuguese), Master Dissertation, UFES, 2008

Falbo, R.A. (1998). Knowledge Integration in a Software Development Environment (in Portuguese), Doctoral Thesis, COPPE/UFRJ, 1998

Falbo, R.A. (2004). Experiences in Using a Method for Building Domain Ontologies, In *16th Conference on Software Engineering and Knowledge Engineering (SEKE)*, Canada, 2004

Gonçalves, B. et al. (2009). An Ontology-based Application in Heart Electrophysiology: Representation, Reasoning and Visualization on the Web, In *ACM Symposium on Applied Computing*, 2009

Gonçalves, B.N., Guizzardi, G. & Pereira Filho, J.G. (2011). Using an ECG reference ontology for semantic interoperability of ECG data, In *Journal of Biomedical Informatics*, Special Issue on Ontologies for Clinical and Translational Research, Elsevier, 2011

Guarino, N., Guizzardi, G. (2006). In the Defense of Ontological Foundations for Conceptual Modeling, In *Scandinavian Journal of Information Systems*, ISSN 0905-0167, 2006

Guizzardi, G. & Wagner, G. (2004). A Unified Foundational Ontology and some Applications of it in Business Modeling, In *Open INTEROP Workshop on Enterprise Modelling and Ontologies for Interoperability, 16th CAiSE*, Latvia, 2004

Guizzardi, G. (2005). *Ontological Foundations for Structural Conceptual Models*, Ph.D. Thesis, University of Twente, ISBN 90-75176-81-3, The Netherlands, 2005

Guizzardi, G. (2006). The Role of Foundational Ontology for Conceptual Modeling and Domain Ontology Representation, In *Companion Paper for the Invited Keynote Speech, 7th International Baltic Conference on Databases and Information Systems*, 2006

Guizzardi, G. (2007). On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models, In *Frontiers in Artificial Intelligence and Applications, Databases and Information Systems IV*, IOS Press, ISBN 978-1-58603-640-8, Amsterdam, 2007

Guizzardi, G., Falbo, R.A. & Guizzardi, R.S.S. (2008). The Importance of Foundational Ontologies for Domain Engineering: The case of the Software Process Domain (in Portuguese), *IEEE Latin America Transactions*, Vol. 6, No. 3, July 2008

Guizzardi, G. & Halpin, T. (2008). Ontological foundations for conceptual modeling, In *Journal of Applied Ontology*, v.3, p.1-12, 2008

Guizzardi, G., Lopes, M., Baião, F. & Falbo, R. (2009). On the importance of Truly Ontological Distinctions for Ontology Representation Languages: An Industrial Case Study in the Domain of Oil and Gas, In *Lecture Notes in Business Information Processing*, 2009

Horrocks, I. et al. (2003). SWRL: A Semantic Web Rule Language Combining OWL and RuleML, In *DAML*, Oct 2011, From http://www.daml.org/2003/11/swrl/

IEEE (1994). *IEEE Standard Glossary of Computer Hardware Terminology,* IEEE Std 610.10-1994

ISO/IEC (1993). *IT – Vocabulary – Part 1: Fundamental terms*, ISO/IEC 2382-1:1993

ISO/IEC (2005). *Information technology – Service management*, ISO/IEC 20000, 2005

ITIL (2007). *ITIL Core Books*, Office of Government Commerce (OGC), TSO, UK, 2007

Jones, D.M., Bench-Capon, T.J.M. & Visser, P.R.S. (1998). Methodologies For Ontology Development, In *15th IFIP World Computer Congress*, Chapman-Hall, 1998

Lopez de Vergara, J.E., Villagra, V.A. & Berrocal, J. (2004). Applying the Web Ontology Language to management information definitions, *IEEE Communications Magazine*

Majewska, M., Kryza, B. & Kitowski, J. (2007). Translation of Common Information Model to Web Ontology Language, In *International Conference on Computational Science*, 2007

Moura, A., Sauve, J. & Bartolini, C. (2007). Research Challenges of Business Driven IT Management, In *2nd IEEE/IFIP Business-driven IT Management, 10th IFIP/IEEE IM*

Moura, A., Sauve, J. & Bartolini, C. (2008). Business-Driven IT Management - Upping the Ante of IT: Exploring the Linkage between IT and Business to Improve Both IT and Business Results, In *IEEE Communications Magazine*, vol. 46, issue 10, October 2008

Motik, B., Sattler, U. & Studer, R. (2005). Query Answering for OWL-DL with Rules, In *Journal of Web Semantics*, Vol. 3, No. 1, pp. 41-60, 2005

Pavlou, G. & Pras, A. (2008). Topics in Network and Service Management, In *IEEE Communications Magazine*, 2008

Pras, A., Schönwälder, J., Burgess, M., Festor, O., Pérez, G.M., Stadler, R. & Stiller, B. (2007). Key Research Challenges in Network Management, *IEEE Commun Magazine*, 2007

Protégé (2011). The Protégé Ontology Editor and Knowledge Aquisition System, October 2011, Available from http://protege.stanford.edu/

Rector, A. & Welty C. (2001). Simple part-whole relations in OWL Ontologies, In *W3C Recommendation*, Oct 2011, Available from http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole

Sallé, M. (2004). IT Service Management and IT Governance: Review, Comparative Analysis and their Impact on Utility Computing, In *HPL-2004-98*, 2004

Santos, B.C.L. (2007). O-bCNMS: An Ontology-based Network Configuration Management System (in Portuguese), Master Dissertation, UFES, 2007

Sirin E. et al. (2007). Pellet: a Practical OWL-DL Reasoner, In *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, Vol. 5, No. 2, pp 51-53, 2007

Smith, B. & Welty, C. (Eds.). (2001). Ontology: Towards a new synthesis, Chris Welty and Barry Smith, Formal Ontology in Information Systems, ACM Press, 2001

Ullmann, S. (1972). Semantics: An Introduction to the Science of Meaning, Basil Blackwell, Oxford, 1972

Vermeer, M.W.W. (1997). Semantic interoperability for legacy databases, PhD Thesis, University of Twente, The Netherlands, 1997

Wong, A.K.Y., Ray, P., Parameswaran, N. & Strassner, J. (2005). Ontology Mapping for the Interoperability Problem in Network Management, *IEEE Journal on Selected Areas in Communications (JSAC)*, Vol. 23, No. 10, October 2005

Xu, H. & Xiao, D. (2006). A Common Ontology-Based Intelligent Configuration Management Model for IP Network Devices, *In Proceedings of the First International Conference on Innovative Computing, Information and Control (ICICIC)*, 2006