Towards the Use of Formal Ontologies in Enterprise Architecture Framework Repositories

Aurona Gerber and Alta van der Merwe

Meraka Institute, Pretoria, South Africa

Abstract. An enterprise architecture (EA) framework is a conceptual tool that assists organizations and businesses with the understanding of their own structure and the way they work. Normally an enterprise architecture framework takes the form of a comprehensive set of cohesive models or enterprise architectures that describe the structure and the functions of an enterprise. Generically, an architecture model is the description of the set of components and the relationships between them. The central idea of all architectures is to represent, or model (in the abstract) an orderly arrangement of the components that make up the system under question and the relationships between these components. It is clear within this context that the models within an enterprise architecture framework are complex. However, recent advances in ontologies and ontology technologies may provide the means to assist architects with the management of this complexity.

In this position paper we want to argue for the integration of formal ontologies and ontology technologies as tools into enterprise architecture frameworks. Ontologies allow for the construction of complex conceptual models, but more signicant, ontologies can assist an architect by depicting all the consequences of her model, allowing for more precise and complete artifacts within enterprise architecture framework repositories, and because these models use standardized languages, they will promote integration and interoperability with and within these repositories.

1 Introduction

An enterprise architecture (EA) framework is a conceptual tool that assists organizations and businesses with the understanding of their own structure and the way they work. It provides a map of the enterprise and is a route planner for business and technology change. Normally an enterprise architecture framework takes the form of a comprehensive set of cohesive models or *enterprise architectures* that describe the structure and the functions of an enterprise [1, 2].

Probably the most widely adopted definition for *enterprise architecture (EA)* is the IEEE definition where EA is described as a widely adopted means for coping with organizations' ever-increasing complexity and for ensuring that organizations appropriately use and optimize their technical resources. EA is an integrated and holistic vision of a system's fundamental organization, embodied in its elements (people, processes, applications, and so on), their relationships to each other and to the environment, and the principles guiding its design and evolution [3].

Gerber A. and van der Merwe A. (2009).
Towards the Use of Formal Ontologies in Enterprise Architecture Framework Repositories.
In *Proceedings of the Joint Workshop on Advanced Technologies and Techniques for Enterprise Information Systems*, pages 114-124
DOI: 10.5220/0002220801140124

The term *Enterprise Architecture (EA)* originated from the thinking around both the terms 'business' and 'architecture'. EA describes the business process of IT by creating a relationship between the IT structure that is used in the organization and in each specific system [4].

Within this context, an *enterprise* is regarded as a company, business, organization, or other purposeful endeavour. An enterprise has the following characteristics [5]:

- An enterprise consists of people, information, and technologies.
- An enterprise performs business functions.
- An enterprise has a defined organizational structure that is commonly distributed in multiple locations.
- An enterprise responds to internal and external events.
- An enterprise has a purpose for its activities.
- An enterprise provides specific services and products to its customer.

The enterprise is thus an holistic term for 'business entity' in all its facets.

Generically, *architecture* is the description of the set of components and the relationships between them [6]. The central idea of all architectures is to represent, or model (in the abstract) an orderly arrangement of the components that make up the system under question and the relationships or interactions of these components [5]. A further definition of an architecture is that an architecture is defined by the recommended practice as the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution [3]. A system architecture is an essential mechanism required to conceptualise, analyse and design systems [7, 8] and there is consensus among researchers and system architects that the determination of the architecture of a system is crucial to the successful understanding and development thereof, especially when the system envisaged is intricate and multifaceted [9, 7].

In this paper, we want to emphasize the notion from the definitions above that an architecture is a model, and an EA is also a model or set of models in that it *is an integrated and holistic vision of a system's fundamental organization, embodied in its elements (people, processes, applications, and so on), their relationships to each other and to the environment, and the principles guiding its design and evolution*. An EA framework is a comprehensive set of EA models used by the architect.

But what is a model? Dijkstra introduced the concept of *models* into computer science in the early '70s [10]. Models were recommended to simplify unmastered complexity. He argued that the *programmer and his mind are an important part of the computing process* and that *modularised, goto-less programs lead to more efficiency in the use of the computer* [11]. Avison and Fitzgerald define a model as an abstraction and representation of part of the real world [12]. *Abstraction* is the process of stripping an idea or a system of its concrete or physical features for a simplified representation of a complex application. Models are used at various levels of system abstraction. A model provides a way of viewing the important aspects of a system at a specific level in such a way that higher levels depict the *essence* of the system and the lower levels show detail that does not compromise the essence. The conceptual level is a high-level overview description of the universe of discourse (UoD) or the domain of interest such as the overall information system, the business system, or even the society [12].

Furthermore, Lipitt defines a model as a symbolic representation of all the aspects, as well as their interrelationships, of a complex event or situation. The true value of any model lies in the fact that is an abstraction or representation of reality that is useful for analytical purposes [13]. According to Lippitt (1973, p.9) *modelling will help expedite problem-solving and change because it enables those involved to conceptualize the multiple factors through visualized thinking. The interrelationship between the cognitive process of thinking cannot be separated from perception: problem-solving involves cognition, and cognition includes perception. Visualization improves the capability to perceive and, therefore, assists the cognitive process.*

Since EA models within EA frameworks depict an integrated and holistic vision of a system's fundamental organization, embodied in its elements (people, processes, applications, and so on), their relationships to each other and to the environment, these models are complex and multi-faceted, and need to depict all elements on different levels of abstraction, as well as the relationships between elements. In general, tools do not exist that can assist an enterprise architect to manage the complexity of EA architectures.

In the past ten to fifteen years, advances in reasoning and modeling technologies have ensured that issues regarding the complexity of models can be addressed. It is here that ontologies play a crucial role. Roughly speaking, an ontology structures a conceptual model in ways that are appropriate for a specific application domain, and in doing so, provides a way to attach meaning to the terms and relations used in describing the domain. A more formal and widely used definition is that of Grüber who defines an ontology as a formal specification of a conceptualisation [14]. The importance of this technology is evidenced by the growing use of ontologies in a variety of application areas, and is in line with the view of ontologies as the emerging technology driving the Semantic Web initiative [15].

In this position paper we want to argue for the integration of formal ontologies and ontology technologies as tools into enterprise architecture frameworks. Ontologies allow for the construction of complex conceptual models, but more significant, ontologies can assist an architect by depicting all the consequences of her model. Formal ontologies also allow an architect to view and understand the implicit consequences of explicit statements and the reasoning technologies can help to ensure that a model is consistent.

In this section we have argued that *conceptual models* are an integral part of an EA. The rest of the paper is structured as follows: Section 2 will provide background information on enterprise architectures and ontologies (Sections 2.1, 2.2 and 2.3). Section 3 describes a proof-of-concept experiment where we modeled a process diagram using a formal ontology. Section 4 discusses the findings, as well as perceived advantages and disadvantages, and the paper concludes in Section 5.

2 Background

This section provides some background into enterprise architectures (Section 2.1), enterprise architecture frameworks (Section 2.2) and ontologies (Section 2.3).

2.1 Enterprise Architectures

As mentioned in the introduction, the IEEE definition describes enterprise architecture as an integrated and holistic vision of a systems fundamental organization, embodied in its elements (people, processes, applications, and so on), their relationships to each other and to the environment, and the principles guiding its design and evolution [3].

For completeness, we list the more prominent of the numerous definitions for the concept *enterprise architecture (EA)*:

- Chung and McLeod (2002) describe EA as a comprehensive model of an enterprise: a master plan, which acts as a planning, structuring, and integrating guideline and force for an organization. EA covers business structure and context, information technology dimension and organizational structure, and workflow dimension in achieving the organization's goals and strategies. It seeks to promote synergy between the various dimensions, aligned with achieving overall business purposes [16].
- According to Kaisler, Armour et. al (2005) an EA identifies the main components of the organization, its information systems, the ways in which these components work together in order to achieve defined business objectives, and the way in which the information systems support the business processes of the organization. The components include staff, business processes, technology, information, financial and other resources, etc. [6].
- Barnett, Presley et. al. (1994) defines an EA as a 'blueprint' or 'picture' which assists in the design of an enterprise [17].
- Rood (1994) argues that an EA shows the primary components of an enterprise and depicts how these components interact with or relate to each other. An EA is a conceptual framework that describes how an enterprise is constructed by defining its primary components and the relationships among these components [5].

Kaisler, Armour et. al. (2005) describes enterprise architecting as the set of processes, tools, and structures necessary to implement an enterprise-wide coherent and consistent IT architecture for supporting the enterprise's business operations. It takes a holistic view of the enterprise's IT resources rather than an application-by-application view [18]. According to Ernst, Lankes et. al. (2006) EA management is a continuous and iterative process controlling and improving the existing and planned IT support for an organization. The process not only considers the information technology (IT) of the enterprise, but also business processes, business goals, strategies, etc. are considered in order to build a holistic and integrated view on the enterprise. The goal is a common vision regarding the status quo of business and IT as well as of opportunities and problems arising from these fields, used as a basis for a continually aligned steering of IT and business [19]. These two definitions place more emphasis on the way in which enterprise architectures are used.

The primary focus of this paper is on the notion in the above definitions that an enterprise architecture is an *architecture of an enterprise*, and that an *architecture is a model*. Furthermore, an *EA framework* is a comprehensive set of EA models used by an enterprise architect.

2.2 Enterprise Architecture Frameworks

An enterprise architecture (EA) framework is a conceptual tool that assists organizations and businesses with the understanding of their own structure and the way they work and it often takes the form of a comprehensive set of cohesive models or *enterprise architectures* that describe the structure and the functions of an enterprise [1, 2].

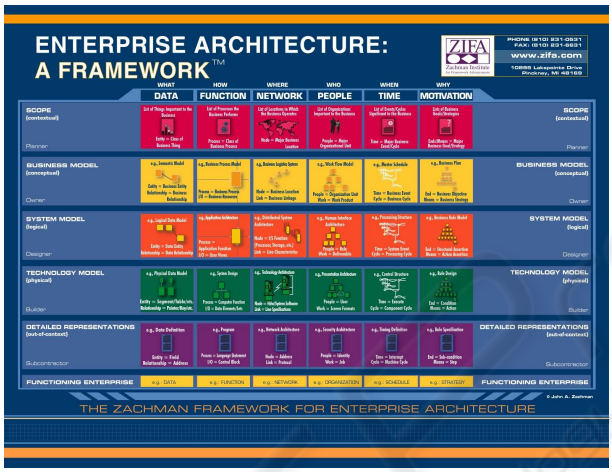

Fig. 1. The Zachman Enterprise Architecture Framework.

Within the field of Enterprise Architecture, several popular frameworks are used to architect enterprises [20], including the Department of Defense Architecture Framework (DoDAF), the Federal Enterprise Architecture Framework (FEAF), the Treasury Enterprise Architecture Framework (TEAF), the ANSI/IEEE 1471 Standard [21] and the Zachman framework [22, 23].

The focus of the Zachman framework for enterprise architecture originated with the idea of the classical architectural representation and production of a complex engineering product. When developing an information technology system, various parties are involved [24]. The Zachman framework proposes a logical structure for classifying and organizing the descriptive representations of an enterprise, in different dimensions, and each dimension can be perceived as a different perspective. The Zachman framework is depicted in Figure 1.

According to Bahill et. al [20], the Zachman framework is a classification schema consisting of six rows and six columns, used for organizing descriptive representations of an enterprise. The rows focus on the different stakeholder perspectives of an enterprise, while the columns focus on different areas of interest within those perspectives. According to Zachman [23], the Zachman Framework is an ontology ”a theory of the existence of a structured set of essential components of an object for which explicit expressions is necessary and perhaps even mandatory for creating, operating, and changing the object (the object being an Enterprise, a department, a value chain, a *sliver*,

a solution, a project, an airplane, a building, a product, a profession or whatever or whatever)".

The forte of the Zachman framework (Figure 1) is that it is a normalized schema [20]; it provides an even coverage of important topics and does not have redundancy built into it. Each cell in the schema can be thought of as having two dimensions: scope (width) and level of detail (depth). Each cell in the schema contains at least one 'primitive' model or artifact. The rows represent different perspectives, including the Scope, Business model, System model, Technology model, detailed representation and the Real system. In contrast, the columns (with no significance in the order, focus on the data, functions, networks, people, time and motivation (or also referred to as the What, Why, When, How, Where and Who. In building the enterprise model, a framework such as the Zachman framework guides the architect in constructing the different models applicable to each perspective being modelled.

Since EA models within EA frameworks depict an integrated and holistic vision of a system's fundamental organization, embodied in its elements (people, processes, applications, and so on), their relationships to each other and to the environment, these models are complex and multi-faceted, and need to depict all elements on different levels of abstraction, as well as the relationships between elements. In general, tools do not exist that can assist an enterprise architect to manage the complexity of EA architectures.

2.3 Ontologies

The concept of an ontology was inherited from philosophy and only recently became commonplace in computer systems technology descriptions where an ontology specifies a machine readable vocabulary. Ontologies on the Semantic Web and expert systems or AI (artificial intelligence) technologies of the 1980s are based on the same motivations but they emerged from different architectures which implies that the technologies are deployed or applied differently [25]. The term ontology has become widespread within ICT and refer to anything from a taxonomy, a domain vocabulary and a conceptual model, to a formal ontology. Lassila and McGuinness gave a spectrum of ontologies as depicted in Figure 2 [26]. Even Zachamn refers to his enterprise architecture framework as an ontology, but this is in the sense that it depicts a conceptual model of the architecture models necessary to depict an enterprise [23]. In this paper we use the term *ontology* when we mean a formal ontology based on one of the OWL standards which is DL-based.

A *formal ontology* specifies a *machine-readable vocabulary* in computer systems technology descriptions. Generally such an ontology is defined as a shared, formal, explicit specification of a conceptual model of a particular domain [27, 28]. A formal ontology typically describes a hierarchy of resource concepts within a domain and associates each concept's crucial properties with it and therefore ontologies are used to define and manage concepts, attributes and relationships in a precise manner [29].

The construction and maintenance of formal ontologies greatly depend on the availability of ontology languages equipped with a well-defined semantics and powerful reasoning tools. Fortunately there already exists a class of logics, called description logics or DLs, that provide for both, and are therefore ideal candidates for ontology languages

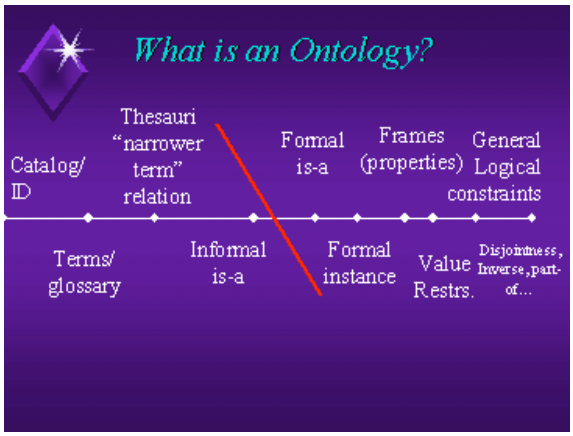

Fig. 2. Web ontologies may be viewed as a spectrum of detail in their specification [26].

[30]. That much was already clear fifteen years ago, but at that time, there was a fundamental mismatch between the expressive power and the efficiency of reasoning that DL systems provided, and the expressivity and the large knowledge bases that ontologists needed. Through the basic research in DLs of the last fifteen years, this gap between the needs of ontologists and the systems that DL researchers provide has finally become narrow enough to build stable bridges. In fact, the web ontology language OWL, which was accorded the status of a World Wide Web Consortium (W3C) recommendation in 2004, and is therefore the official Semantic Web ontology language, is based on an expressive DL [31].

Due to the advances in DL research mentioned above, there is growing interest in the use of ontologies and related semantic technologies in a wide variety of application domains. Arguably the most successful application area in this regard is the biomedical field [32, 33]. Some of the biggest breakthroughs in ontological reasoning can be traced back to the pioneering work of Horrocks [34] who developed algorithms specifically tailored for medical applications. And recent advances have made it possible to perform standard reasoning tasks on large-scale medical ontologies such as SNOMED CT—an ontology with more than 300 000 concepts and more than a million semantic relationships—in less than half an hour; a feat that would have provoked disbelief ten years ago [35].

At present the ultimate vision of the Semantic Web based on expressive ontologies as formulated by Berners-Lee et al. [36] remains largely a research initiative. However, the vision initiated significant interest with regards to the required technologies for the enabling of the Semantic Web [2, 3, 4, 5]. Notably, the W3C recommended a number of standards for languages of increasing expressivity as depicted by the Semantic Web layered architecture [36, 27]. One of these is OWL, the Web Ontology Language[10] used as a standard to express formal ontologies [37, 31].

One of the consequences of the standardisation of OWL by die W3C is the development of several tools and reasoners that support the development of formal ontologies based on the OWL standard. Notable ontology editors are Protégé 4 [38] and SWOOP

[39]. Reasoners provide computable and complete reasoning for OWL ontologies, and some are integrated into the ontology editors. Notable reasoners are Fact++ [40] and Pellet [41]. A summary of a substantial number of Semantic Web tools, including OWL ontology editors and reasoners can be found at http://esw.w3.org/topic/ SemanticWebTools, From the above it is clear that, even though these tools are still under development, the momentum generated will soon ensure that formal ontologies with their supporting technologies and tools enter mainstream modeling approaches.

The next section will argue for the incorporation of these tools and technologies into the modeling of the architectures necessary in an enterprise architecture framework.

3 The use of Formal Ontologies to Model Enterprise Architectures in EA Frameworks

In the previous sections we discussed enterprise architectures (EA) as models, and their relationships within enterprise architecture frameworks (EAF), as well as the recent technologies in computer science used for formal model building (ontologies). In this section we discuss a proof-of-concept experiment conducted to investigate the use of a state-of-the-art ontology editing tool namely Protég ;e 4 with its associated reasoners (Fact++ and Pellet 1.5) to model an existing process model developed as part of PhD thesis research work to extract generic process models [42].

Within the Zachman framework, one of the tools used to model column two (the *How* or *FUNCTION* column (refer to Figure 1 and row two and three (Business model (Conceptual) and System model (Logical))is the process model.. Curtis et. al [43] define a process model as an abstract description of an actual or proposed process that represents selected process elements that are considered purpose of the model. Jacobs [44] argues that there are three types of models, those used for enterprise architectures, those used as framework models for standardization and those for applying methods. The process model developed focus on models used to view the behaviour of processes for workflow application development, similar to those published in articles by Weske, Goesmann, Holten and Striemer [45] and Wu, Deng and Li [46].

The process model depicted in Figure 3 was compiled from a case study environment to derive process reference models was the higher education institution (HEI) application domain. The goal was to identify this high-level process reference model and also to do a more in-depth analysis of one of the high-level processes, in order to comment on the generic nature of sub-processes [42]. This is a typical process model that will be extracted to depict the high-level processes of an enterprise in the Zachman framework.

3.1 Approach

For the proof-of-concept experiment, we used the process model discussed in the previous section and depicted in Figure 3 and used Protgégé 4 to translate this process model into an OWL 2.0 ontology. We used the Beta build 106 of Protégé 4 on a MacBook.

Level of ontology engineer could be described as low intermediate if we have three levels: beginner, intermediate and advanced.

The steps within Protégé 4 could be summarised as follow:

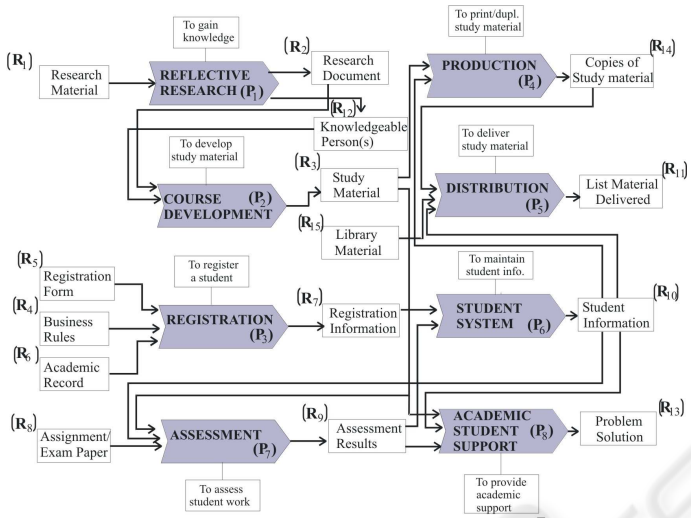

Fig. 3. The high level process model of Higher Educational Institutions.

1. We first identified the components *Process*, *ProcessGoal* and *ProcessInputOutput* as concepts.
2. We identified the object properties *hasGoal*, *hasInput* and *hasOutput*
3. We then defined a *Process* as having at least one (or none)[1] goals, inputs and outputs.
4. Next all the process goals and process inputs and outputs were entered, first as individuals and then as concepts[2].
5. In the last step of the first iteration, a new concept *NamedProcess* was created, and all the specific processes in the process model were entered. The class hierarchy is depicted in Figure 4.
6. During the modeling, both reasoners (Fact++ and Pellet 1.5) were used to debug the ontology and ensure that it is consistent. This ensured that a *NamedProcess* is also classified as a *Process* when it conformed to the definition as is evident in Figure 5.
7. During the second iteration we investigated the use of individuals versus concepts for specific processes, process goals and process inputs and outputs.
8. During the third iteration, we tightened the definition of a *Process* to have at least one *ProcessGoal* and only *ProcessGoals* as goals, and similarly for process inputs and outputs via the *hasGoal*, *hasInput* and *hasOutput* properties as depicted in Figure 6.

The next section discusses some experiences and findings.

[1] This is a DL or OWL / Protégé phenomenon in the modeling of existential quantification

[2] An individual in OWL is an assertion in the ABox or an instantiation of a concept e.g. `Susan` as an instance of the concept *Person*. A concept (or class) resides in the TBox and can be refined further in the model.

Fig. 4. The process model concept hierarchy.

3.2 Experience

The following are noted experiences during the construction of an formal ontology for a process model using Protégé 4.

- It was problematic to know how to model a process with its components and to make reusable modeling decisions. Is it necessary to define another level of abstraction, or should the process components be on the highest level of the concept hierarchy? Are the components depicted sufficient for all process models? In the end we were guided by the *KISS* principle for this experiment - *Keep It Simple and Straightforward*.
- The old problem of ontology engineering also cropped up - should we define all the 'things' as concepts, or are some individuals? During the first iteration we went as far as to classify all processes, process inputs and outputs, and process goals as individuals, but this constrained the refining of concepts and querying, so we decided to keep most processes and process inputs and outputs as concepts. Process goals could be modeled as individuals, and we use some goals as individuals to test the approach as is evident in Figure 7 where *hasGoal* use the *value* property to link to a *ProcessGoal* individual called *iPG_ToProvideAcademicSupport*.

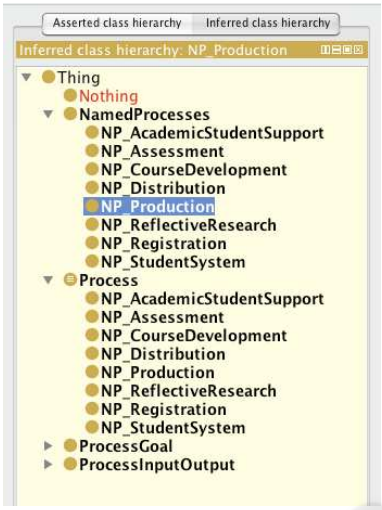

Fig. 5. The inferred process model concept hierarchy depicting all *NamedProcesses* also classified as *Processes* according to the definition of a process.

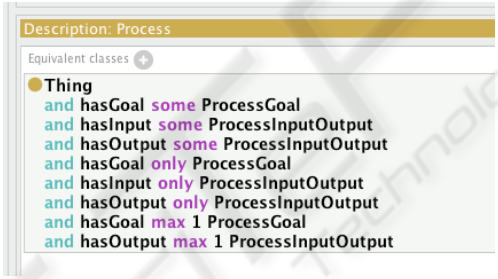

Fig. 6. The final definition of a *Process*.

- It is clear that the construction such a formal model would require several iterations and the input from domain experts both with regards to modeling and the processes. This is not different from the construction of the original process model, however, since ontology definitions are concise and both implicit and explicit consequences are depicted and often unexpected, more iterations would probably be required. This would however, result in a more concise model where the precise meaning of concepts are expressed.
- The ontology tool lost the flow that was depicted in the original diagram. This was experienced as a huge drawback since this is one of the key elements of a process diagram. The process flow knowledge are captured in the model as is evident in Figure 8, but it could not be displayed graphically and in a holistic view. In Figure 8 we view the *usage* of the *hasInput* object property and we can drive what inputs are used for which processes.

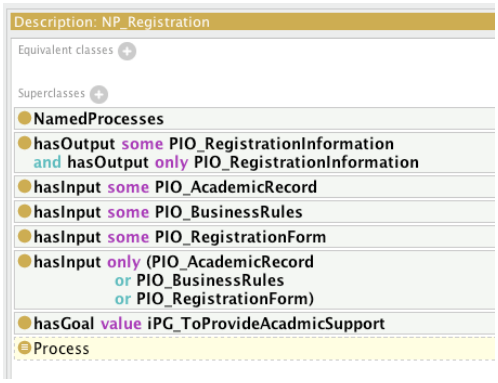

Fig. 7. Using individuals and the *value* property for *hasGoal*.

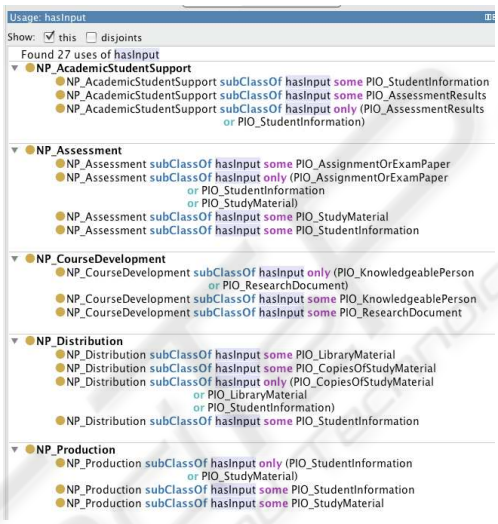

Fig. 8. The *usage* of the *hasInput* object property.

4 Findings

The experiment conducted for the proof-of-concepts for this position paper was very basic and it is clear that the use of formal ontologies as enterprise architecture models require extensive research. However, the following became evident:

– It was relatively easy to capture the knowledge in the chosen process model diagram in an ontology. This is probably because a high-level process model is not very precise. An example thereof is the process outputs *Knowledgeable person* and *Student information* that are clearly very different. It would be possible to define much more elegantly what is meant by these terms, but that would complicate the model substantially, and it is not knowledge that was in the original diagram. To

model both these as sub-concepts of a *ProcessInputOutput* concept was straightforward and captured the same knowledge as contained in the original diagram.
– Protégé 4 was easy to use and enabled us to easily create the formal ontology. The only drawback was the graphical rendering of the process flow information that was evident in the original diagram and which cold not be rendered satisfactorily.

4.1 Advantages

From the experiment, the following evident advantages could be listed, but the list is by no means complete:

– The use of this approach allows an architect to specify concise definitions of concepts. In our experiment we defined a *Process* specifically, and this definition could be used to ensure that all the specific processes are indeed processes when we used the reasoning support.
– The use of a precise definition also assisted with debugging. When a specific process was not classified as a *Process*, we knew there was something wrong in our process description or in the definition. this enables us to ensure that the model is complete and consistent.
– The reasoners depict all consequences of our model, not only the explicit statements we made, but also implicit consequences. In our experiment these are relatively trivial, but it was evident that implicit consequences will be very valuable once the models are complex.
– The Protégé 4 environment was easy to use if one knew what one wanted to do, but for that familiarity with the modeling language is a prerequisite.
– Ontology editors such as Protégé 4 assists architects to specify models in a standardised language (usually OWL) which will promote interoperability.

4.2 Disadvantages

From the experiment, the following disadvantages were evident:

– One of the first observations in this regard is that there are currently no firmly established methodologies for ontology engineering. It is generally recognised that this is a research topic that warrants urgent attention [47]. Within an enterprise architecture framework, this is even more important and will probably have to be tailored towards the specific architecture model required within the framework
– There are still only a limited number of tools available. These tools have matured substantially over the past few years, but their availability remains a disadvantage.
– Tied to the above is the limited functionality of ontology tools. The most evident was the ability to generate a graphical display of the ontology, and specifically in our experiment - the ability to view the process flow that was contained in the model but could not be easily extracted. Graphical displays are very important for conceptual modeling.
– It was also evident that, although a variety of tools exist for ontology construction and maintenance [48, 49, 38], these tools remain accessible mainly to those with specialised knowledge about the theory of ontologies.

– It was difficult to debug the model because the reasoner would only depict a concept as inconsistent and would not show a reason or explanation. One had to resolve errors using trial and error, and these errors were often due to unexpected consequences of statements made earlier.

5 Conclusions

From the proof-of-concept experiment it is clear that formal ontologies and the associated technologies can play a substantial role to enhance the models required for enterprise architecture frameworks, because the models are more explicit, precise and consequences can be exposed. These models are also based on standardised languages and will promote interoperability of models within an enterprise architecture framework.

With this position paper, we want to set the agenda for further research in this field. The proof-of-concept experiment yielded some promising insights, and we identified the following areas of interest for immediate research attention:

– It is necessary to do research into defining base models or vocabularies for all the architectures in an enterprise architecture framework. For example, it is necessary to precisely define a *process* precisely and ensure that all models using the process concept adhere to the definition.
– It is necessary to do research into the development of methodologies, techniques and tools that support the development of formal models within an enterprise architecture framework.
– It is necessary to investigate integration and interoperability between architecture models within an enterprise architecture framework.

References

1. Zachman, J.: The zachman framework for enterprise architecture, primer for enterprise engineering and manufacturing. Zachman International (2003)
2. Session, R.: A comparison of the top four enterprise-architecture methodologies, building distributed application. Web (2007)
3. IEEE: Recommended practie for architectural description of software-intensive systems. IEEE Report (2000)
4. Kim, J.W., Kim, Y.G., Kwon, J.H., Hong, S.H., Song, C.Y., Baik, D.K.: An enterprise architecture framework based on a - common information technology domain (eafit) for improving interoperability among heterogeneous information systems. In: Third ACIS Int'l Conference on Software Engineering Research, Management and Applications, Central Michigan University, Mount Pleasant, Michigan, USA. (2005)
5. Rood, M.: Enterprise architecture: definition, content and utility. In: Third Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, Morgantown, West Virginia. (1994)
6. Armour, F., Kaisler, S., Liu, S.: A big-picture look at enterprise architectures. IEEE: IT Professional January/February 1999, 1 Issue 1 (1999) 35 – 42
7. Fowler, M.: Patterns of Enterprise Application Architecture. Addison-Wesley, Boston, MA, USA (2003)

8. Garlan, D., Shaw, M.: An introduction to software architecture (1994) Last accessed 15/9/2006.
9. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice. Addison Wesley Professional (2003) Last accessed 12/8/2006.
10. Dijkstra, E.W.: The end of computing science? Communications of the ACM, 44 (2001) 92
11. Weiner, L.H.: The roots of structured programming. In: Papers of the SIGCSE/CSA technical symposium on Computer science education, New York, NY, USA, ACM Press (1978) 243–254
12. Avison, D., Fitzgerald, G.: Information Systems Development: Methodologies, Techniques and Tools. third edn. McGraw-Hill, UK (2003)
13. Lippitt, G.L.: Visualizing Change: Model Building and the Change Process. University Associates, Inc. (1973)
14. Grüber, T.: A translation approach to portable ontology specifications. Knowledge Acquisition, 5 (1993) 199–220
15. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American, 284 (2001) 34–43
16. Chung, M., McLeod, G.: Enterprise architecture, implementation, and infrastructure management. In: 35th Annual Hawaii International Conference on System Sciences, Hawaii. (2002)
17. Barnett, W., Presley, A., Johnson, M., Liles, D.H.: An architecture for the virtual enterprise. In: IEEE International Conference on Systems, Man, and Cybernetics, Piscataway, New Jersey. (1994)
18. Kaisler, S., Armour, F., Valivullah, M.: Enterprise architecting: Critical problems. In: 38th Hawaii International Conference on System Sciences, Hawaii, USA. (2005)
19. Ernst, A.M., Lankes, J., Schweda, C.M., Wittenburg, A.: Tool support for enterprise architecture management - strenghts and weaknesses. In: 10th IEEE International Enterprise Distributed Object Computing Conference, Hong Kong. (2006) 13 – 22
20. Bahill, T., Botta, R., Daniels, J.: The zachman framework populated with baseball models. Journal of enterprise architecture (2007)
21. Maier, M., Emery, D., Hilliard, R.: Ansi/ieee 1471 and systems engineering. Systems Engineering, 7(3) (2004) 257–270
22. Zachman, J.: A framework for information systems architecture,. IBM Systems Journal, 26(3) (1987)
23. : The zachman frameworkł: The official concise definition. Zifa website (2009) last accessed 26/2/2009.
24. Goethals, F.: An overview of enterprise architecture framework deliverables. Web (2003) last accessed 26-02-2009.
25. Palmer, S.B.: The semantic web: An introduction. W3C Web site (2001) Last accessed 16/9/2006.
26. Lassila, O., McGuinness, D.L.: The role of frame-based representation on the semantic web. Technical report, Knowledge Systems Laboratory Report KSL-01-02, Stanford University (2001)
27. Broekstra, J., Klein, M., Decker, S., Fensel, D., van Harmelen, F., Horrocks, I.: Enabling knowledge representation on the web by extending rdf schema. In: Proceedings of the 10th International World Wide Web Conference (WWW10), Hong Kong. Volume ACM 1-58113-348-0/01/0005. (2001) 467 last accessed 18/3/2006.
28. Decker, S., Melnik, S., van Harmelen, F., Fensel, D., Klein, M., Broekstra, J., Erdmann, M., Horrocks, I.: The semantic web: The roles of xml and rdf. IEEE Internet Computing, 4 (2000) 63–74
29. Bussler, C., Fensel, D., Maedche, A.: A conceptual architecture for semantic web enabled web services. ACM SIGMOD, SPECIAL ISSUE: Special section on semantic web and data management, 31, issue 4 (2002) 24 – 29

30. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
31. McGuinness, D.L., van Harmelen, F.: Owl web ontology language overview. W3C Web site (2004) Last accessed 13/9/2006.
32. Wolstencroft, K., Brass, A., Horrocks, I., Lord, P., Sattler, U., Stevens, R., Turi, D.: A little semantic web goes a long way in biology. In: Proceedings of the 2005 International Semantic Web Conference (ISWC 2005). LNAI, Springer (2005)
33. Hahn, U., Schulz, S.: Ontological foundations for biomedical sciences. Artificial Intelligence in Medicine, 39 (2007) 179–182
34. Horrocks, I.: Semantic web: the story so far. In: W4A '07: Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A), New York, NY, USA, ACM (2007) 120–125
35. Suntisrivaraporn, B., Baader, F., Schulz, S., Spackman, K.: Replacing SEP-Triplets in SNOMED CT using Tractable Description Logic Operators. In: Proceedings of AIME 2007. (2007)
36. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. The Scientific American, 5 (2001) Last accessed 20/9/2006.
37. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: Owl web ontology language reference. W3C Web site (2004) Last accessed 15/3/2005.
38. Protégé: The protégé Ontology Editor. http://protege.stanford.edu/ (2009)
39. : Swoop - semantic web ontology editor. Web (2009)
40. : Fact++ ontology reasoner. Web (2009)
41. : Pellet: The open source owl dl reasoner. Web (2009)
42. van der Merwe, A., Kotzé, P.: A systematic approach for the identification of process reference models. In: In Proceedings of the IASTED International Conference on Software Engineering SE 2009 , Innsbruck, Austria. (2009)
43. Curtis, B., Kellner, M., Over, J.: Process modeling. Communication of the ACM, 35(9) (1992) 75–90
44. Jacobs, D.: Towards a business process model warehouse framework. In: School of Computing, University of South Africa: Pretoria. (2008)
45. Weske, Goesmann, Holten, Striemer: A reference model for workflow application development processes. In: The International Joint Conference on Work Activities coordination and Collaboration. (1999) 1–10
46. Wu, Z., Deng, S., Li, Y.: Introducing eai and service components into process m anagment. In: The 2004 IEEE International Conference on Service Computing (SCC '04). (2004) 271 – 276
47. Gómez-Pérez, A., Fernández-López, M., Chorco, O.: Ontological Engineering. Springer (2004)
48. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Journal of Web Semantics, 5 (2007)
49. Kalyanpur, A., Parsia, B., Sirin, E., Cuenca-Grau, B., Hendle, J.: Swoop: A Web Ontology Editing Browser (2005)