11
An Ontological Approach to Domain Modeling for MDA-oriented Processes

Dilip Patel, Michael Kasovski, Shushma Patel

There exists a gap between objects which exist in the real world and the elements which represent them in a software system. This paper addresses this gap with a domain modeling process able to construct, manage, and negotiate the appropriate domain concepts within a domain model. This process is based on an ontological approach and its constituting components. It was developed as a The Model Driven Architecture (MDA) provides a framework of models which can store the various aspects contained within a software system. Within the context of the MDA, the domain modeling process and models will be situated within the Computational Independent Model. This approach is illustrated by its application to a simple situational case study.

1. Introduction

Many organisations employ the use of software as a means to support their continued existence (Patel, 2002). Software can assist in accomplishing their day-to-day business activities. It can promote the communication and organisation of its employees. Customers can also benefit from the services provided by the system. Hence, the software system must be able to incorporate such issues in its development to provide a well-suited system for the organisation.

There exists a gap between the existing business concepts in the real world and the ones which exist within the software (Daniels, 2002). Business elements require an appropriate representation in the software system which will determine how concepts within the system should exist and describe the relationships between one another. Domain modeling is a method which is concerned with the capture and representation of business concepts in a software system (Agrawal, Karsai, and Ledeczi, 2003). It provides an articulation of the meaning behind concepts which are implemented in the system. The resulting domain model acts as a formalised context behind the nature of the elements in the developing system.

The Model Driven Architecture focuses on the use of models as the central artifact in the development of software systems (Miller and Mukerji, 2003). It provides a hierarchy of models with the purpose of separating the aspects of the developing system into each model. The system progresses in the development lifecycle as it undergoes a series of transformations between models.

An ontological approach within the context of the MDA can provide the necessary depth in developing a domain model. The domain model should not be hindered by computational dependencies in order to exhibit the meaning behind related concepts in the system. Hence, the ontological approach will contribute
to the development of a domain modeling situated as part of the Computational Independent Model (CIM) of the MDA.

This paper will provide a domain modeling method which is able to be incorporated into a MDA oriented process. We first discuss in section 2 the underlying concepts which support this research. Namely, there will be a discussion on the notion of ontology, the model driven framework, and the relationship between them. We exhibit previous works which have influenced the development of our process in section 3. In section 4, the domain modeling process is described. We apply the process to a case study in section 5 to express its practical implications. Section 6 examines research currently undertaken in this research area. We discuss future research opportunities which would stem from this work in section 7. Finally, section 8 outlines the conclusions of the paper.

2. Underlying Concepts

This section discusses the theoretical underpinning of the concepts behind the development of the domain modeling process. The topics discussed will be the notion of ontologies, the MDA, and the relationships between them.

2.1 Ontologies

The study of ontologies has undergone much research over the past few decades (Guarino and Welty, 2002). It has contributed to the advancement of many fields within computer science, including AI, knowledge management and software engineering. There are many recent developments which address the various activities within a software development process (Benjamin, Patki and Mayer, 2006; Kaiya and Saeki, 2005; Hoss and Carver, 2006). An ontology provides many of the qualities which are essential for developing a valuable domain model.

Gruber can be seen as providing much of the foundations concerning modern ontology development. According to Gruber (1993), “an ontology is an explicit specification of a conceptualization”. He believed that ontologies articulated a formalised account of what is said to exist within a given area of existence. This area of existence stems from its context, or purpose which dictates the representations constituting a simplified description of the real world. Relationships within an ontology form a bond between concepts which could then be interpreted as knowledge (Gruber, 1993). Constraints on ontological elements would ensure the semantic integrity of the model.

Guarino’s (1998) seminal paper builds upon the foundations of ontological development, debating some of Greuber’s views on the definition of an ontology. From his perspective, Guarino argues that it is the semantic meanings, or axioms which define the ontology, rather than its vocabulary. Since the axioms determine the validation of ontological elements, one must adhere to the axioms in order to be considered part of the ontology. According to Guarino, a vocabulary developed by concepts and relationships may envelop models and implementations which are outside the context of the ontology. The axioms provide a filter from which only the appropriate knowledge may be extracted.
Kaiya and Saeki (2005) apply Guarno’s ontological theory in an attempt to analyse and evaluate requirements. Their ontology was created from the semantics and relationships existing within a given set of requirements. Their research does support the relationship between requirement and domain concepts. However, their process generates the ontology form the requirement, whereas we argue that the ontology should serve as the basis for defining a requirements model.

Nguyen and Corbett (2004) approach the notion of ontologies from a mathematical perspective. They formally define the notion of concepts, relations, and properties in order to mathematically analyse system knowledge. However, their formalisms add robustness to the model at the cost of model comprehension. Our process will place a stronger emphasis on model comprehension, as a reference to the meaning behind elements in a system model.

An ontological-based approach can provide the necessary tools and information to develop a simple, yet meaningful domain model. A separation of concerns must exist between the domain and the system elements which will apply them. This alleviates complexity when understanding the meaning encapsulated within the domain concepts.

2.2 The Model Driven Architecture (MDA)

The MDA is a modelling framework developed by the OMG which address the development of software systems (Miller and Mukerji, 2003). Models in the MDA represent information about the system at varying levels of abstraction. Development processes which incorporate the MDA develop systems by the development and transformation of the system models until a complete software product is generated. The MDA was designed to encourage the reuse and interoperability of system models and their information to develop quality systems.

Each model within the MDA articulates the developing system according to a particular view (Miller and Mukerji, 2003). Each view consists of a level of abstraction, and a set of system concepts which are relevant to the model’s purpose at that stage. This information is articulated as a metamodel, whose purpose is to formally define the platform upon which the model rests upon. Elements within the system models are defined by the semantics and formalisms associated by their corresponding concepts defined in the metamodel.

The models within the MDA are the Computational Independent Model (CIM), the Platform Independent Model (PIM), and the Platform Specific Model (PSM). Each model is dependent on its relative abstract model, yet is independent if its implementing, or more specific model. The CIM contains a model which is independent of any design, or computational implementations. It contains elements which articulate the system’s requirements and environment. The PIM expresses the system independent of any implementing technology. The PSM expresses the designed system according to a specific implementing technology. As a system model transforms into a more specific mode, so too must the metamodel to accommodate the shift in level of abstraction. Eventually, the final
transformation generally transforms the PSM into a technology-specific format executable by the computer system.

This process develops a domain model which exists from a computational independent viewpoint. It serves as a reference of meaning behind the concepts existing in a CIM. This acknowledges the domain model’s role as metamodel for the CIM within a MDA oriented process. As the CIM translates into a PIM, so too must the domain model transform to maintain the relationship between element and concept throughout the MDA framework.

3. Themes and Rationale

This section presents previous works which have influenced the domain modeling process. The underlying themes and rationale behind the development of the method will be discussed.

The framework is designed to address the notion of representing the domain of a software-based system. A domain in this paper is a representation of the elements which exist within the organisation and environment (Neusibeh & Easterbrook, 2000). The domain model may be subject to reuse across multiple projects. Also, the organisation may evolve over time to adapt to changing needs. Therefore, a domain model must contain mechanisms which will allow it to adapt alongside the changing business environment. The domain is an important aspect to communicating the nature of the organisation in order to create the appropriate system to address it.

A domain model would encapsulate the concepts which exist within the real world and their meaning, which would be incorporated within the system models (Daniels, 2002). The domain model does not contain the information concerning how and where the domain will be used by the system. Within the context of the MDA, a domain model would then be represented as a metamodel due to its use in encapsulating the meaning of system concepts.

This separation creates a distinction between the nature of the organisation and the system which will address the organisation. This separation allows for easier management of the domain independent of the system which will incorporate it. However, since the system addresses the needs of the organisation, its elements are dependent on the representations of the real world elements within the domain model.

Reuse has been shown to be a factor of great importance to the success of developing a domain model (Agrawal, Karsai and Ledeczi, 2003). The systems which are developed for an organisation should incorporate similar concepts which represent it in the domain. The reuse of a domain model would allow for quicker software development by the immediate deployment of a previously made domain model. Also, reuse of a domain model could provide easier communications across different systems due to the standardised understanding of the business concepts generated from sharing a common model (Miller and Mukerji, 2003).

An organisation may change over time to adapt to its changing needs (Patel,
Since a domain model represents elements from the organisation, it must also be able to adapt in order to be consistent with the organisational changes. Therefore, the domain model should contain mechanisms to handle the evolution of models over time. Elements which were previously included in the domain model may require modification to incorporate subtle, yet previously unrecognisable properties. The model may also undergo an extension to encompass concepts whose importance was previously unknown to the organisation (Gause, 2005).

Domain models generally appear in the early stages of a development process, when there are multiple perspectives of the system and its environment (Lamsweerde, 2000). This would entail an elicitation process which would extract the necessary information from the real world into the model (Neusibeh and Easterbrook, 2000). Furthermore, in order to consolidate the different perspectives, the domain model must be able to be readily communicated and managed between project members. To ensure communication, a balance must be made between the simplicity of comprehending the model with the robustness of the information which it contains.

A domain model is most appropriate within the CIM of an MDA oriented process. The CIM was designed to hold the elements which express the requirements and environment of the system (Miller and Mukerji, 2003). Elements within a CIM are defined in terms of its real world representations within the domain model. Therefore, the domain model will act as the metamodel to the CIM. As the CIM transforms into a PIM, the domain model must also transform into a PIM-compliant metamodel. To accommodate this, elements within the model must adhere to MOF conventions and be UML-compliant.

4. Development method

This domain model will be situated as a metamodel within the CIM of an MDA-oriented process. It will contain the ontological aspects necessary to support the elements of a requirements CIM. The domain model in this work will serve as the context to and environment of the requirements as a representation of the real world organisation. It will be independent of its application as a PIM, hence it will not contain elements of data structures or operational behaviour. This process contains cyclical elements which pose to reexamine and improve the quality of the developing domain model. The domain model will be UML-compatible so that it can easily be transformable into a PIM metamodel for further development.

4.1 Elicitation and Documentation Phase

The purpose of this phase is to capture as much information from the organisation as possible for definition within the domain model. This phase is influenced by the requirements engineering activities in (Neusibeh & Easterbrook, 2000). In order to depict an accurate representation of the organization, the domain concepts should be elicited and modeled. This resembles Silva’s (2000) preliminary phase in its purpose to gather information about the situation. The domain concepts
should correspond to an element existing in the real world. However, the domain concepts should not contain any attributes or properties. At this stage, importance is stressed on modeling as many concepts as possible to allow for thorough examination later. Once the concepts are defined, relationships between concepts will then be identified. Each relationship is an identified binary link between two concepts. The relationship will also include a description of the relationship in order to better comprehend the connection between concepts. Since the domain model is independent of the system design, cardinality will not be defined on any of the relationships. Once all of the possible relationships are identified, the constraints will then be incorporated into the model. Constraints will determine any properties of the domain which represent similar limitations or rules which exist in the organisation. Constraints in this model will contain a description of the constraint and the domain elements to which it belongs. A constraint may be attributed to a single domain element, to multiple elements at once, or as a global constraint applicable across the domain.

4.2 Domain Audit Phase

This phase is concerned with the examination and improvement of the domain model. Within the first phase, emphasis was placed on eliciting as much information about the domain as can be found. In this phase however, the domain modeler who had elicited the elements will now examine them to determine their relevance to the domain and to remove apparent inconsistencies. The model has a greater chance of undergoing a more rigorous assessment due to the modeler's previous experience with the domain. Elements should be assessed in the same order to which they were elicited; domain concepts, then their relationships, and finally the domain constraints. This ensures a uniform strategy towards the domain assessment. When an element in the domain model is deemed to be unnecessary to the domain model, any directly associated elements must be reviewed to determine whether it still applies to the model. For instance, a constraint related to a removed concept should be examined to determine whether it still applies to other elements within the model or be removed along with the concept. In the case which a concept is found to not be relevant within a domain model, it should be removed from the model along with any associated relationships. This removes any unrelated elements from the domain model. Any constraints related to the removed concept should be examined to determine whether it still applies to another element within the model or be removed along with the concept.

However, in some occurrences removed elements may turn out to exhibit its relevance under a future examination. To minimize the effects of such errors, a mechanism will exist which will contain the removed errors. Each removed element would still possess its information as well as the related elements where the element was placed before the removal. If the situation arises where a removed element is found to have belonged to its original place, the replacement of the element back into the model would be easier to manage.
4.3 Domain Negotiation Phase

Different participants in this development process observe the domain through their own perspectives (Silva, 2002). This may lead to various interpretations of the same domain. A consensus must be made on the nature of the domain model among the participants to ensure a consistent understanding of the environment and its meaning. Once a domain model has undergone examination by the modeler, it must then undergo a collaborated inquiry among all of the participants. Any domain models which other participants have developed will aid in determining points of comparison and difference in interpretations. When negotiating the different interpretations of the domain, any removed elements should be placed separately with a description over its previous placement, similar to those removed in Phase 2. Once the domain is agreed upon, it is ready for application and reference by elements within the CIM.

As time passes, the real world organisation may undergo changes to adapt to the evolving environment. This may prompt a modification of the domain model to reflect this change. The model is sent back to a domain expert to determine the most appropriate alteration and its effect on the domain. Once the modification is made, it undergoes another mediation among the participants to consolidate the new interpretation of the domain for further use.
5. Case study
This work will present its approach on a case study examined in (Hoss and Carver, 2005). It provides a simple scenario which can clearly exhibit the domain modeling process and visualised representations of the developing domain model. The following figure will show the progress in the initial phase of the process.
Figure 2(a, b, c). Domain model at various stages during the Elicitation and Documentation Phase.
In figure 2a, the domain expert begins by identifying the many concepts on the model. Any concept which may be uniquely described, such as the open and closed elevator buttons are expressed separately on the domain model. Figure 2b exhibits the domain model after the initial attempt at relationship modeling. The relationships specified in this model are articulated in terms of phrases which express modeler’s understanding of the domain expressed in the case study. Finally, this phase concludes with the addition of constraints seen in figure 2c. A constraint was added on the door concept which limits the possible states of the door to being open or shut.

**Figure 3. Domain model after Domain Audit Phase**

Once the modeler becomes familiar with the domain, he or she may provide insight onto improvement of the initial domain model. One major change consists of the consolidation of the open and shut buttons into one concept encompassing the trigger(s) for elevator door movement. The open and shut buttons are therefore removed from the model, however are still stored for future reference. Also, the relationships between the buttons and their lights may have been better articulated as one illuminating the other rather than a simple switch. Once the modeler is satisfied with their domain model, it is brought up for discussion among the other participants. There, it undergoes the scrutiny of other people as well as other domain models which may have been made by others. Once completed, it is ready for application and reference within the CIM.
6. Related works

Model Integrated Computing (MIC) is an approach which employs domain concepts in a framework of models in order to develop software (Szpanovits and Karsai, 1997). From their perspective, domain models in MIC articulate a system’s design. They are based on the formalisms and language constraints expressed in the metamodel. The intricacy behind the language formalisms in the metamodel increases the amount of information contained in the metamodel at the cost of complexity.

Moon, Yeom, and Chae (2005) emphasize the importance commonality and variability within domain requirements. Their research presents a domain as the set of similar constructs among a group of products and the variants which may exist from product to product. The concepts within the domain do not accommodate for constraints, but rather are articulated a part of the domain requirements. This weakens the separation of concerns as the domain requirements become burdened with the additional complexity entailing the inclusion of constraints.

Garrido et al. (2007) approach domain ontologies and developing computation independent models from a groupware perspective. Their research stresses on the collaboration and communication aspects of knowledge sharing and requirements specification. However, they do not emphasize on the issues of ontology complexity and its impact on the development process.

7. Implications and Future work:

This paper introduced a process which addresses the notion of domain modeling for incorporation within an MDA-oriented process. It delves into the articulation of real world concepts at the early stages of development, where the system is independent of computational constraints. At this level of abstraction, due to the lack of computational rigour, a domain model may be evaluated according to the degree to which it can communicate its knowledge to other people. What also should be considered is the extent to which another person is able to comprehend about the domain and about the system from the domain model.

The process contains elements which promote the idea of collaboration with other participants and iterative domain development. A study could be constructed to determine the capabilities of such a process as a learning tool towards understanding the organisation via the domain model. This may lead to future applications of learning-based methods into MDA-oriented processes. Not only would the learning curve to MDA processes decrease, but it could aid in bringing different participants such as stakeholders to directly be able to contribute to the system development process.

The domain modeling process proposed in this paper contained a mechanism which would store previously removed domain elements for future reference. The paper outlined the necessary information to be stored, yet does not delve into the effectiveness of incorporating such a mechanism. A question which may be
This approach discusses the metamodeling aspects of developing a CIM appropriate for an MDA oriented process. However, this paper does not explain in detail the nature of the model which will employ the formalisms developed in this process. Future work may include developing a requirements model which would be able to capture the needs of the system, while articulating them in terms of relevant domain terminology. Such a framework would then be applied to MDA-oriented processes to examine the extent to which the models can integrate and the value it adds to model driven development.

8. Conclusions

This paper focused on developing a domain modeling method suitable for an MDA-oriented process. We identified a link between ontology development and model driven development using the common notion of domain modeling. The process promoted iterative and collaborative development of models representing the elements present in the real world. It promotes the idea of incorporating domain development into a CIM as a component of an MDA-oriented process. We also exhibited the process by applying it to a basic case study and observing the domain’s development. This paper supports the notion that ontology-based domain development can become a valuable asset in capturing and comprehending the meaning behind model driven systems development.

References


