International Journal of Advanced Computer Technology (IJACT)

ISSN:2319-7900

A Domain Model Framework for Engineering Designs Metamodeling

Piah, Z. Patrick[1] , Dept. Of Computer Science, Ken Saro-Wiwa Polytechnic, Bori, Nigeria. zigalopia@yahoo.com
Japheth R. Bunakiye[2] , Dept. of Mathematics/Computer Science, Niger Delta University, Wilberforce Island, Nigeria. jbunakiye@gmail.com
Igulu Kingsley Theophilus[3] ,Dept. Of Computer Science, Ken Saro-Wiwa Polytechnic, Bori, Nigeria. igulukt@gmail.com
Moses Onengiye Georgewill[4] ,Dept. Of Computer Science, Ken Saro-Wiwa Polytechnic, Bori, Nigeria., m-georgewill@yahoo.com

Abstract
The use of graphics models for engineering design and practice alone is quite outdated; much more emphasis is centered on enhancing modeling by transforming these models to products that satisfies varied design intents. A more appropriate view, is modeling engineering designs through metamodeling. Metamodeling controls platform complexities and is productive by forming metamodel of models at a higher level of abstraction. It provides the semantic rules for the modeling environment, which consists of the instances of concepts in the metamodel. A metamodel in its original form describes the rules and constraints of the domain metatypes and metarelationships that are instantiated for use in regular modeling effort. This paper therefore presents a metamodeling framework for modeling engineering designs.

Key Words: Engineering Design; Metamodel; Domain Abstractions; Semantic Rules; Model Commonalities

1. Introduction

Metamodeling forms a metamodel of a model at a higher level of abstraction. It provides the semantic rules for the modeling environment, which consists of the instances of concepts in the metamodel [4]. A metamodel in its original form describes the rules and constraints of the domain metatypes and metarelationships that are instantiated for use in regular modeling effort. Creating a metamodel of engineering design therefore applies to the process of generating such metamodels of making a model of an engineering design model. Models are outcomes from human conceptions, for example physical objects such as telephones, automobiles etc. are models designed to meet to meet specific engineering standards, they are a set of drawings for the production of an object or a system of objects aimed at bringing to bear some descriptions from a set of specifications that describe the function that the designed piece is to achieve [2]. Most engineering designs are created by human effort in a bit to completing a task more efficiently by bringing together technologies to meet human needs. One distinguishing factor in this method of problem solving is the open ended nature of engineering design problems, which means there can be more than one correct solution [3]. Though there are many processes of design relating to engineers as much as possible, the solution to a design problem requires some framework methodology or process. In this paper a framework to improve the solution to specific engineering design is proposed, where the model becomes a core metamodeling entity. The model also represents the concepts within which the metamodeling formalism is created to control the flow of processes without including extra or unnecessary properties captured in the design. The whole idea is processing the models to produce specific executable models [5]. The paradigm shift in engineering design as a solution to meeting technological needs is moving from mere use of graphics primitives to a much more emphasis on transforming these models to products that satisfies varied design intents. As much as there is a lot of software platforms suited for modeling, they also portend a lot of shortcomings; users are often limited by their knowledge of the software or by problems solvable by it [6]. These deficiencies tend to make engineering design standards as merely applicable to the creation of physical objects or, perhaps, software. A more appropriate view, however, is to branch out of these modeling platforms that repeatedly required significant designing or programming expertise by capturing the characteristics of the model as concepts for analysis, for creation and for the implementation of new ideas and inventions. To achieve a successful branch out from these apparent deficiencies, specifying these models in a metamodel becomes paramount, so that the boundaries of possible designs are identify for the elimination of impractical, or otherwise undesirable designs [8].

2. Related Work

Dae-Kyoo et al. [10] presented a metamodel for describing generic solution for problems that occur repeatedly. They applied the descriptions to design patterns with graphical notation and complementing

text. Their suggestion is that to encourage the use of design patterns, the development of pattern supporting tools is imperative. This complements ours in the areas of considerable work on formality and features for variability and commonalities. Matthew Emerson et al. [11] used metamodeling and metaprogrammable tools to aid control engineers build on computing and communication technology to design robust, adaptive and distributed control systems for operating plants with partially known nonlinear dynamics. The tools helped erase the problem of designing and integrating large scale systems. The interesting aspect of their work is the where networked embedded computing is increasingly taking over the role of "universal use of models on different levels of abstractions. J.-M. JÈzÈquel, H. Hussmann, and S. Cook (Eds.) [12] discussed a metamodel for the Unified Modeling Language Critically examining the fact that models, rather than code, now become the key artefacts of software development, they declared that consequently, this raises the level of requirements for modeling languages on which modeling practitioners should rely in their work. Which means, like ours any inconsistency in a metamodel may cause major problems in the subsequent applications. Complementing the submissions, Oscar L´opez, Miguel A. Laguna, and Francisco J. Garc´ıa [13] offered metamodeling for requirements reuse. Their discussion suggested that correct requirements determination is a critical factor in software development as it takes domain resources into consideration. They presented a metamodel to integrate some different types of semiformal diagrams into a requirements reuse approach, which is capable of controlling the diversity of notations and formats, so that any existence of different levels of requirements description will not make requirements reuse difficult.

3 Defining the Engineering Design Model

The engineering design model is simply the domain model that represents real world concepts in the engineering domain. The necessity for a domain model is founded on the fact that it is the exact conceptual entity that forms the metamodeling instances. Figure 1 is a reservoir engineering design model, it is one example of the numerous physical components that can be found in a typical storage facility for fluids [17].

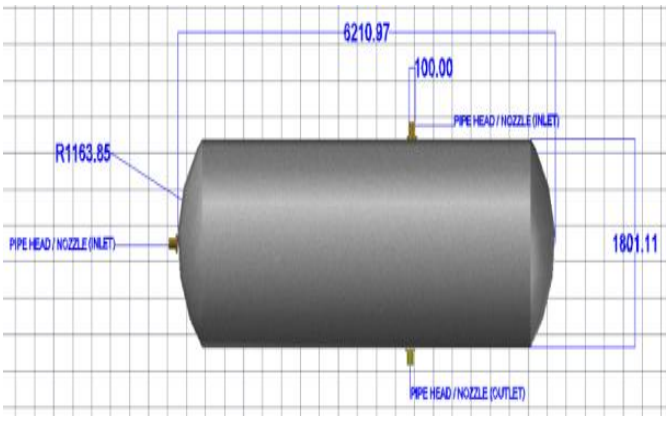

Figure 1: Reservoir Model

3.1 Metamodeling Metrics

Usually engineers sort for domain knowledge consisting of domain analysis outputs and application with a view to solving the design issues in the problem space. Figure 2 depicts such analysis products involving: domain definition, defining the scope of the engineering domain, describing components of the context model, and feature models for variabilities and commonalities [1].

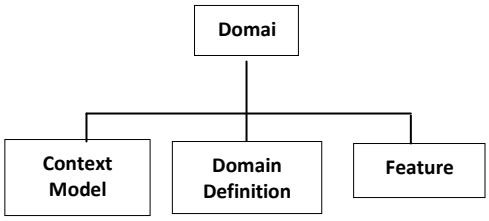

Figure 2: Domain Analysis Products

In pursuant to meeting responsibilities domain experts do prepare design documents from time to time to align with current codes and standards within the scope of the engineering scope under consideration. This criterion may be part of overall project design criteria or may be a separate document prepared solely for the engineering design [18]. In either case, it reiterates the design requirements delineated in the contract specification and should define the applicable codes and standards, environmental conditions, design parameters, and other pertinent design bases that will govern the project. Even calculations are prepared to support the establishment of flow rates, system pressures, temperatures, and wall thickness, and other design parameters. In this detail metamodeling applicability is a style that guides the design of a complement of engineering products or settings. Engineers wishing to create flexible and reusable suite of products would critically follow metamodeling metrics in the design of each object, which can describe choices for design aspects such as materials, colour schemes, shapes, patterns, textures, or layouts.

ISSN:2319-7900

4. Domain Model Framework

The domain model framework is the solutions outline that setts easier and faster design practice. In this example the domain refers to relevant themes solely within any engineering domain. The basic idea is to express the domain specific terms as envisioned by domain experts in a meaningful way, such that the specific problem of evolving corresponding designs can be solved. Essentially for proper conceptualization and functioning, the domain model framework as shown in figure 3 contain domain classes that denotes a type of object, attributes that describes named slots of specified types in a domain class holding separate values, associations representing relationships between two or more domain classes that describes links between their object instances. Associations can have roles, describing the multiplicity and participation of a class in the relationship, and additional rules that govern the model logic.

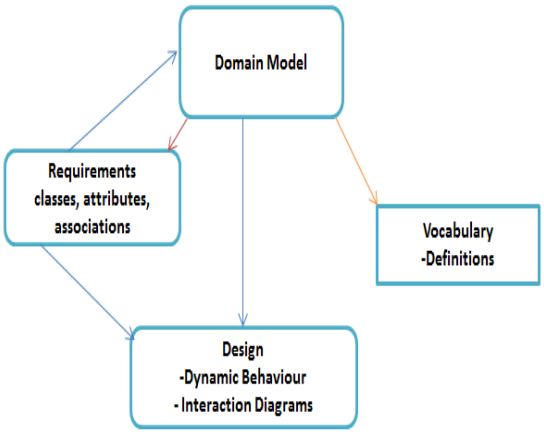

Figure 3: Domain Model Framework

The structure of the domain model entails conceptualization of the design issues and their relationships. The structure depicts a conceptual model, where all the issues are related to finding solutions to a specific problem. The specific problem in the problem domain simply examines the salient views and areas of interest, and then excludes everything not applicable in the course of solving the problem. It means explicitly describing the area of application that needs to be looked at to solving the particular problem [19]. Also evident in the domain model are the correct relationships between different concepts clearly chosen to be independent of design or implementation concerns. Such independence will remove any sort of confusion among stakeholders, especially those responsible for designing and implementing a solution; where the model provides the key artifact of the metamodel understanding and clarity [14]. Once the domain concepts have been modeled, they can then be mapped into physical design or implementation constructs that supports higher-level abstractions and code generation.

4.1 Complexity Control

Domain classes and relationships forms the basic defining components for a domain model. While the domain classes represent the concepts from the domain; each domain relationship represents the bindings of these concepts relative to the structural logic of the core of the metamodel definitions [13]. Requirements elicitation is seen to be the most crucial phase when defining a domain model, the domain modeling framework and the requirements are mutually dependent in such a way that, whereas requirements help building up and clarifying the model, the framework supports clarification of the requirements [15]. Therefore, a carefully crafted domain model definition is a great tool for enhancement and for controlling complexity of the system under development. It helps a great deal in resolving numerous uncertainties in both the requirements and the design intent. In numerous engineering domains, effective domain model definition stems from capturing industry level requirements in the form of vocabulary. These requirements now become valid domain entities for the associated behaviours and relationships that describe the entire problem space [16]. Domain model is a very significant aspect among the different pieces that must be created; since a domain model describes and constrains the scope of the problem space it can be effectively used to verify and validate the understanding of the problem domain among various stakeholders [9]. All domain specific metamodeling involves a domain model at its core because it defines the vocabulary represented by the metamodel, also involved are the properties, and the relationships between model attributes and values, which serves as helpful communication tools. The domain model at the core of a metmodel defines the elements that constitute a model (for example, the inner and outer diameters that make up a beam model in a typical bridge design), it also gives rules for how these elements may be connected together and provides the foundation for notation definitions, validation, and serialization properties in the metamodel [8].

4.2 Mappings and Abstraction Levels

The metamodeling vocabulary, which represents detailed technical characteristics comes from the application domain, it describes the needs that when met, engineering design metamodeling can be solved by domain engineers and users. These needs are in form of specific abstraction levels and can be met by mapping the appropriate concepts to the abstraction levels [7]. The vocabulary serves a useful purpose of

ISSN:2319-7900

mapping these abstractions to concepts in the form of products of the attributes of the engineering domain model, and represented as sequences of the Meta instances. Figure 4 is such representation of a Meta instance of a domain specific language (DSL) Script where a real life physical pipeline build engineering activity according to some design specification and layout is given. The metamodeling sequence provides the transformation mechanisms of what the DSL does and what is carried out in real life using the vocabulary mappings and abstraction levels. What happens in real life is a pipeline design to specifications according to customers or stakeholders needs, what metamodeling does is to be able to process stakeholders input specifications through a meta-control scheme and direction. The resultant effect of the internal working mechanism is an interpreter program running on the target platform that loads the script, and then acts on it. Which means having got a model, all the important semantic behavoiur is captured by the semantic model being populated.

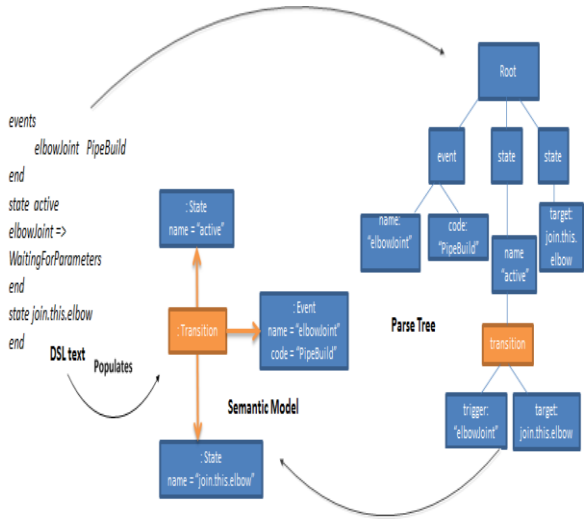

Figure 4: Representation of a DSL Script

5. System Requirements

Engineering development process involves the analysis phase, design, and implementation phases. The domain model framework is the solutions outline that enhances the engineering designs metamodeling. Based on the logical potentials of complexity control, and abstraction mappings flexibility of the domain model framework for engineering design metamodeling, a set of requirements are found to be necessary for a metamodel definition tailored towards modeling engineering designs [18].

5.1 Tackling Platform Complexities

In conventional modeling, for example, parameters define certain aspects of a design that can be used to build the model. Though conventional modeling embodies classes, methods, and function names that becomes available by object creation and method invocation to any program using the library, domain concepts cannot be expressed effectively during development. A metamodel for modeling engineering designs in any specific engineering domain can tackle the complexities of efficient expression of domain concepts in a metamodel for possible orientations. It involves domain-specific constructs and abstractions from the start that are adapted towards the particular application domain of engineering practice. With domain specific notations it can help shelve users from platform complexities and reduce the amount of programming expertise needed to solve specific problems.

5.2 Productivity

Instead of struggling with identified complexities that often evolve from the semantic gap between design intent and the expression of this intent in thousands of lines of codes whose huge syntax neither conveys domain semantics nor design intent, a specifically designed metamodel for engineering designs can help stakeholders focus on a new approach to engineering designs modeling. A new method that allows them get involved with familiar notations and have their design intents achieved. Metamodeling sees the model only as the key entity throughout development, it is an approach derived from model driven engineering (MDE) technologies comprising of Model Driven Architecture (MDA), and Domain-Specific Modeling (DSM). MDAs language specifications focus more on the universal modeling language (UML) diagram definition standards, whereas DSM language specifications focus more on requirements within a particular domain. UML is not an end user representation language, and as such couldn't possibly capture appropriately domain concepts specific to stakeholder viewpoints. However, the DSM approach to the complex problem of efficiently and effectively aiding engineering design is declarative, usually expresses what the program should accomplish by hiding from the user the complexities of how to solve the problem in terms of sequences of actions to be taken [5]. Policies are specified at a higher level of abstraction using models and are separated from the mechanisms used to enforce the policies thereby enhancing development time and productivity.

5.3 Framework Functionality

Metamodeling has grown significantly over time. Particularly existing technologies such as Model-Centric Software Development (MCSD), System Execution Modeling (SEM), MetaEdit+ (Graph,

ISSN:2319-7900

Object, Property, Port, Relationship, and Role meta-objects (GOPRR) tool for developing domain-specific modeling languages and code generators) and Generic Modeling Environment (GME) are widely available to the software engineering community [18]. As new technologies and methodologies continue to emerge, metamodeling will continue to enhance engineering designs modeling by offering the groundwork for extensions without tampering with the functionality of computing and underlying frameworks. However some existing systems in the engineering industry are multifaceted; one aspect may be powered with 3D CAD software to generate project reports, and design specifications. The other aspects may be web based and executes all financial and allied duties. A bottleneck is the inability of these tools to give the engineers the required interface to freely interact with the systems without being guided by strict design policies inherent in the software. For these reasons, stakeholders become so dependent on programming expertise that is required all the time to leverage the CAD systems and the available APIs for artefact orientations. A metamodel for modeling engineering designs needs to have three collaborative sub-systems i.e. a domain model that captures the metrics of the engineering field; the user interface model that can enable stakeholders to interact with the system and a solution model that integrates for artefact orientation and code generation. As far as collaborators and domain experts could see through to a design scenario, the system should be able to capture it and meet their needs [19].

6. Conclusion

The use of graphics models for engineering design and practice alone is quite outdated; much more emphasis is centered on enhancing modeling by transforming these models to products that satisfies varied design intents. As much as there is a lot of software platforms suited for modeling, they also portend a lot of shortcomings; users are often limited by their knowledge of the software or by problems solvable by it. These deficiencies tend to make engineering design standards as merely applicable to the creation of physical objects or, perhaps, software. A more appropriate view, however, is to branch out of these modeling platforms that repeatedly required significant designing or programming expertise to metamodeling. Metamodeling specification are formal, where all the models are instances of the metamodel, and the engineering design models based on the concepts and rules set in the metamodel.

References

[1] F. Rubén, Miguel A. and J. Requejo, Development of a Feature Modeling Tool using Microsoft DSL Tools. GIRO Technical Report 2009

[2] Braha D, Reich Y (2001) Topological structures for modeling engineering design processes. International conference on engineering design (ICED 01), Glasgow

[3] Randy H. Shih P AutoCAD 2013 Tutorial - First Level: 3D Fundamentals

[4] B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Y¨ucesan, Using domain specific language for modeling and simulation: Proceedings of the 2010 Winter Simulation Conference USA

[5] Steve Cook, Gareth Jones, and Stuart Kent, (2007) Domain-Specific Development with Visual Studio DSL Tools, Pearson Education, Inc, USA

[6] Trask, B., Paniscotti, D., Roman, A. and Bhanot, V. Using model-driven engineering to complement software product line engineering in developing software defined radio components and applications. In Proceedings of the ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA'06), 2006, 846 – 853.

[7]Wagner, S., Deissenboeck, F.: An Integrated Approach to Quality Modeling. Fifth International Workshop on Software Quality, In: Proc. of ICSE'07, 6 p. (2007

[8] K. Czarnecki and U. W. Eisenecker. *Generative Programming|Methods, Tools, and Applica-tions*. Addison-Wesley, 2000. ISBN 0-201-30977-7.

[9] S. Wartik and R. Prieto-Diaz. Criteria for Comparing Domain Analysis Approaches. *International Journal of Software Engineering and Knowledge Engineering*, 2(3):403{431, Sept. 1992.

[10] A Dae-Kyoo Kim, Robert France, Sudipto Ghosh, Eunjee Song. UML-Based Metamodeling Language to Specify Design Patterns Computer Science Department Colorado State University Fort Collins, CO 80523, USA {dkkim,france,ghosh,song}@cs.colostate.edu

[11] Matthew Emerson and Sandeep Neema and Janos Sztipanovits Metamodeling Languages and Metaprogrammable Tools Institute for Software Integrated Systems Vanderbilt University Nashville, TN 37203 E-mail: mjemerson@isis.vanderbilt.edu June 29, 2006 Published in the Handbook of Real-Time and Embedded Systems, Ed. Insup Lee, Joseph Leung, Sang H. Son, CRC Press, 2006

[12] J.-M. JÈzÈquel, H. Hussmann, S. Cook (Eds.): A Metamodel for the Unified Modeling Language UML 2002, LNCS 2460, pp. 2-17, 2002. Springer-Verlag Berlin Heidelberg 2002

[13] Oscar L´opez, Miguel A. Laguna, and Francisco J. Garc´ıa. Metamodeling for Requirements Reuse

ISSN:2319-7900

WER 2002 The fifth workshop on Requirements Engineering (WER 2002) Valencia, Spain,
[14] H. Dieter Rombach. Sofware specifications: A framework. SEI Curriculum Module. Technical Report SEI-CM-11-2.1, Software Engineering Institute, Carnegie Mellon University, January 1990.
[15] Mark Simos, Dick Creps, Carol Klingler, Larry Levine, and Dean Allemang. Organization domain modeling (ODM) guidebook - version 2.0. Technical Report STARS-VCA025/001/00, Lockheed Martin Tactical Defense Systems, 9255 Wellington Road Manassas, VA 22110-4121, June 1996.
[16] A. Sutcliffe and N. Maiden. The domain theory for requirements engineering. IEEE Transactions on Software Engineering, 24(3):174–196, March 1998.
[17] R. Geisler, M. Klar, and C. Pons. Dimensions and dichotomy in metamodeling. In Proceedings of 3th BCS-FACS Northern Formal Methods Workshop. Springer-Verlag, Sep. 1998.
[18] M. Jarke, J. Bubenko, C. Rolland, A. Sutcliffe, and J. Vassiliou. Theory underlying requirement engineering: An overview of NATURE at genesis. Proceedings of the IEEE International Symposium on Requirements Engineering, 1993.
[19] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson. Feature-Oriented Domain Analysis (FODA). Feasibility study. Technical Report CMU/SEI-90-TR21 (ESD- 90-TR-222), Software Engineering Institute, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213, November 1990.