

Supporting engineering design modeling by domain specific modeling language

Japheth Bunakiye. Richard.¹, Ogheneovo Edward. Erhieyovwe.²

¹Department of Mathematics/Computer Science, Niger Delta University, Yenagoa, Nigeria

²Department of Computer Science, University of Port Harcourt, Port Harcourt, Nigeria

Email address:

jbunakiye@yahoo.com(B. R. Japheth), edward_ogheneovo@yahoo.com(E. E. Ogheneovo)

To cite this article:

Japheth Bunakiye. Richard., Ogheneovo Edward. Erhieyovwe.. Supporting Engineering Design Modeling by Domain Specific Modeling Languag. *AmericanJournal of Software Engineering and Applications*. Vol. 2, No. 3, 2013, pp. 86-91. doi: 10.11648/j.ajsea.20130203.11

Abstract: Domain specific modeling methodology employed in this solution provides abstractions in the problem domain that expresses designs in terms of concepts in the application domain. Presented in this paper therefore is a metamodeling tool, an integrated platform which offers layered collections of reusable software primitives whose semantics are familiar only to engineering design mechanisms. It is intended to eliminate the complexities associated with the domain of computing technologies such as CAD systems where the focus is solely on engineering designs expertise in the software systems logic. This tool which was built on the DSL processor engine that compiles the DSL Builder files at the core will enable non design experts to be able to evolve designs specific to their domains of operations and reflecting their view points. At the Development Interface, the templates are created for every transformation added to our model that can be applicable in the physical design of objects in the engineering industry. It will in line remove hassles and complexities of expertise centric design platforms to produce artifacts that will help engineers manage very complex design concepts.

Keywords: Domain-Specific modeling, Primitives, Models, Platform Complexity, Domain Classes

1. Introduction

Computer graphics often referred to as graphics models are representations on a computer screen of physical entities [1]. Graphics [21], models usually are created interactively from graphics primitives (primitives are stored libraries of shapes from common interactive CAD Systems) and can be assembled to more complex forms known as graphics assemblies. A graphics model could also be a final product of the aggregation of graphics primitives, graphics assemblies and subassemblies. The entire process of creating a final graphics model is termed graphics design and the result is termed graphics model [9, 12]. This graphics models can then reflect human conceptions in different domains. In the domain of engineering, it invariably becomes the engineering design.

1.1. Engineering Designs

Design is the creation of a set of drawings for the production of an object or a system of objects. Engineering designs are examples of such objects created from design activities. They are usually graphics [12], models created interactively from graphics primitives with specification that

describe the function the designed piece is to achieve [8].

These models representsalient characteristics of target domains of technological advancement and can also be expanded into concepts that explain simplifications of systems at certain circumstances that can be built upon abstractions [11].

As much as Common interactive CAD systems (e.g. AutoCAD) are utilized by designers for graphics design modeling, they lack the power of expression to specify the engineering design in formal notations [13]. A language which could capture concepts of a specific application domain as formally specified and reflected in the engineering design and can link these concepts to appropriate abstraction levels within the problem domain [17, 11].

What is required to overcome the stated inability of CADs to express domain concepts effectively is domain specific representation. Domain specific representation is made possible through domain specific modeling (DSM). It requires the construction of a domain specific modeling language that can provide a means for expressing domain concepts in a model [17]. This leads to increase in productivity, and provides non-designers and domain

experts the resource to operate on very familiar notations without being burdened by expertise on design [18].

The focus of this paper therefore is utilizing this DSM technology as the background methodology to develop a metamodeling framework in order to tackle the associated complexities of design software. Interactive CADs design platforms usually requires specific expertise in design practice to be able to develop a design solution. But with a modelling tool that offers familiar notations, non-designers and stakeholders of target domains will find it easy for likely design and related activities [20].

The rest of this paper is organised as follows: Section 2 provides related work and review of engineering designs; section 3 discusses the methodology and materials; and section 4 describes the framework in more details. First, we show the modelling framework where concepts are captured from a possible engineering design to form the grammar metamodel. Second, we gave the interpretation and the benefits of the framework. Third, we show the relationships between model elements and the subsequent XML tree representations. Section 5 concludes the paper.

2. Related Work

Quite a number of research works have dealt with efforts geared towards using domain specific modeling in the creation of technologies that addresses platform complexities. Domain specific modeling further addresses the inability of general purpose computing systems such as interactive Computer-Aided Design systems to alleviate these complexities by expressing domain concepts effectively. Douglas C. Schmidt [13] gave a lot of fruitful notes for the development of a domain-specific layer as part of the Model-Driven Engineering (MDE) technologies that can expressing concepts as a necessary step in tackling platform complexities. In the scope of engineering designs, the work by Melgoza, E. L., Rosell, A [5], has a considerable relationship with our focus, and in addition Selic B, [3] gave much insight on determining ways to enable the extraction of concepts from a model as the core entity throughout development in the application of DSM. In this context Marcus Volter [20] did showcase some results in implementing the DSM approach where some useful guidelines were translated into the processes that made our framework feasible [18, 16, 11].

2.1. Solid Modeling

Engineering designs [1] are simply solid models. They are modeled by building them up from primitives found in the libraries of interactive CAD systems such as AutoCAD. Two basic data structures employed in solid modeling are constructive solid geometry (CSG) and boundary representation (B-rep) respectively [2]. CSG as shown in figure 1 uses primitives such as cylinder, sphere, and prism with combined Boolean operations to produce the solid [8].

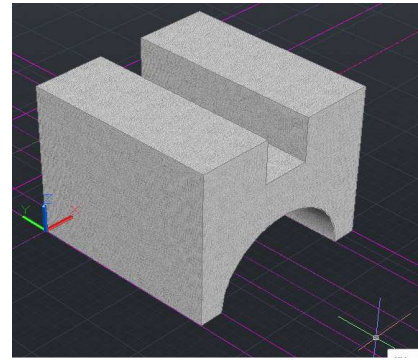


Figure 1: A CSG Solid Model

B-rep solid models as shown in figure 2 are usually constructed using surfaces, curves and points in a 3D space.

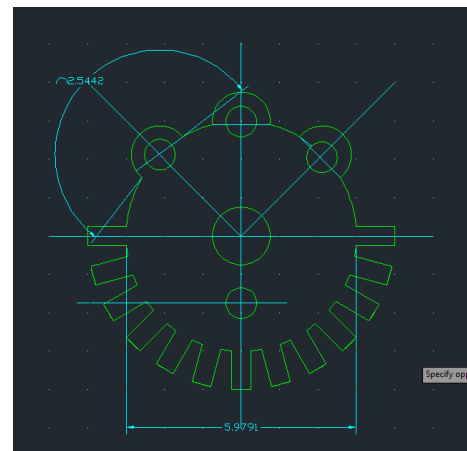


Figure 2: A B-rep Model of Curves and Surfaces

The users often times find it difficult to change the geometry of these solids especially because of the static nature of their dimensions [1]. But parametric modeling capabilities, recently associated with common interactive CAD systems such as AutoCAD however [10] addresses all of the shortcomings of static dimensions. With parametric modeling they can be used for detailed engineering of 3D models of physical components [5]. This possibility however is seen in conventional modeling where objects are clearly labeled [9].

The resultant effect is that quite a lot of modifications have to be made to reflect the intentions of the designer whenever one aspect of the model is changed. The expertise of the designer is greatly depended upon in all circumstances of the application of the rules of the software during design because the software itself lacks the ability to automatically keep track of the changes. The use of common interactive CADs such as AutoCAD for creating conventional engineering designs however, hampers non AutoCAD experts and other domain stakeholders from bringing up engineering designs that could possibly reflect the perspectives of their domains [12]. This means current CAD systems coupled with the third generation programming APIs inherent in them still lack sufficient linguistic power to handle domain and platform complexities. These systems

hasn't moved speedily with domain technologies i.e. software engineering methodologies that can be used to foster model interaction in order to create new objects that encapsulate and relate the details pertinent to the viewpoint of domain experts.

The believe is that such software development efforts will enable stakeholders to cope with platform complexities, it will also be cost effective, save time, and raise productivity levels [8].

Major efforts in tackling these problems are putting the model at the core of development and ignoring any detail not relevant to the application domain perspectives that these models represent. Concepts associated with the domain's technical content are specified and appropriate reductions are made by raising the abstractions and expressions of the characteristics of the models as they relate to engineering designs [2]. Critical in the process is identifying the problem domain (i.e., the problem space), the exact needs these raised levels of abstractions are to be met and the task of how these different domains can be integrated to form a whole modeling platform. A capable software engineering practice to addressing the mapping of these abstractions from the problem domain that expresses design in terms of the concepts to the application domain is the Model-Driven Engineering technology (MDE).

Model-Driven Engineering technology is powered by the Model-Driven Development software paradigm. There are two approaches to Model-Driven Development; the Model Driven Architecture (MDA) and Domain-Specific Modeling (DSM) [14]. While the Model-Driven Architecture (MDA) [20] as part of the object management group (OMG) standards is focused more on the concepts of the problem domain and not on the technical contents of the application domain [2, 14], the Domain-specific modelling approach focused on semantic mappings of the concepts in the application domain to the problem domain and then enabling the system to process the models via the semantic relationships to produce desired artefacts as well as automating transformations and code generation [10, 18].

MDE with the DSML approach is declarative, usually expresses what the program should accomplish without presenting the complexities of how to solve a problem in terms of sequences of actions to be taken. Policies are specified at a higher level of abstraction using models and are separated from the mechanisms used to enforce the policies. DSMLs help overcome the semantic gap between the design intent and the expression of such design intent, and help users from difficulties of how the policies are mapped onto the underlying mechanisms implementing them, thereby, allowing system reuse easily [5, 6].

3. Method and Materials

3.1. Method

The initiative is a modelling tool that could aid domain experts develop and produce designs directly related to their

view points. Applying the DSML approach, the complexities of CAD engineering designs were incorporated as design concepts and the platform complexities of constructing codes from programming languages were encapsulated as abstractions. These concepts were then mapped to the abstractions via semantic relationships [2, 5]. The application domain was taken to be pipelines design domain where design issues pertaining to pipelines can be tackled.

The problem domain was then considered i.e. the concise information about stakeholders needs in this language, the needs that when met, pipeline design issues can be solved by the pipelines design engineers and related experts. The information in this direction led the gathering of a component library framework in order to identify the concepts and associated rules targeted at this methodology to be created utilizing a Microsoft DSL tool (Visual Studio Visualization and Modeling SDK) [2].

In addition to the experts viewpoints we looked at the existing system descriptions, component devices and services, and standards to form the concepts. From the components frameworks relevant vocabulary was created in terms of physical products structures in pipelines design e.g. valves, joints, loops, angles etc., the attributes such as size of pipe, shape, direction of loops etc. were sorted, rules on how a valve may relate to angle etc., and taking note of several levels of abstractions possible only with pipelines design industry. We continued to adding the grammar to the vocabulary and on into the language construct.

In the language construct, the abstraction rules were followed, and the semantics specified. The abstract syntax representing the structure and appearance of our sentences is being generated resulting directly from language definitions and the concrete syntax in form of notations about the key words describing concepts mapped to the design symbols.

3.2. Materials

The solution was built employing the Microsoft Visualization and Modeling SDK in Visual Studio 2012. The System structure was a refinement of the semantic base of the .Net framework built on the DSL processor engine that compiles the DSL Builder files at the core. This possibility created a layer of reusable software, an integrated environment for seamless operations.

4. Solution Framework

The framework as shown in figure 3 is a modelling tool specific to the engineering field in the domain of engineering design. The domain model, which invariably is the engineering design, depicts the domain concepts and then these concepts are mapped to the abstractions via semantic relationships.

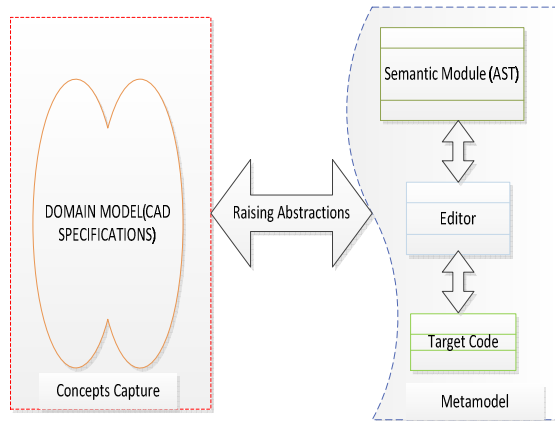


Figure 3: Modeling Framework

The result will be a metamodel, a language definition covering all of the application and problem domains into one development environment capable of processing these models via the editor to produce desired products and automatic code generation [2] [5].

4.1. Solution Interpretation

A primary purpose is the generation of efficient implementation code for the target engineering design domain by enabling domain experts to express their view points through the interface. At the heart of the solution framework is the definition of the model created to represent concepts in the application domain. The chosen example as shown in figure 4 is the pipeline design domain. It is the graphical editor component of the framework in which users can view the model definitions. It is an example of the expression of the models in the DSL core with concrete notations.

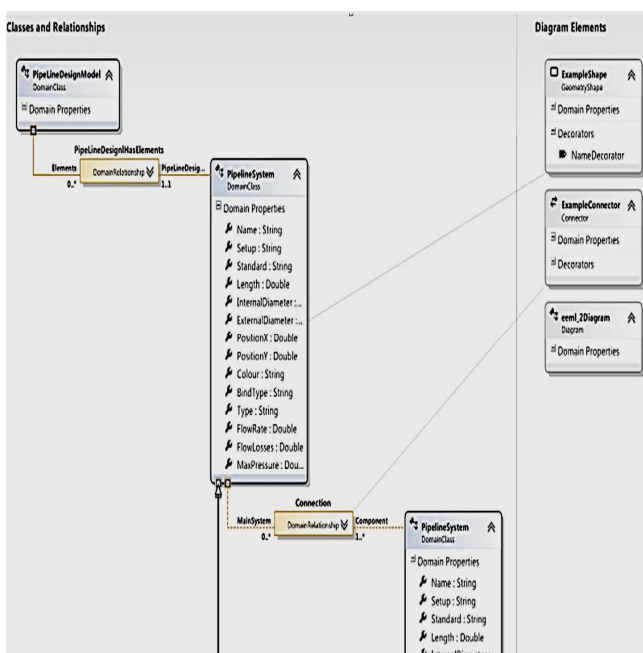


Figure 4: Graphical Component of Solution Framework

These notations represent the language concepts captured from peculiar domain terminologies embodied in the model and are expressed as the concrete syntax through abstraction rules. The model definition therefore signifies the specification of the actual abstract syntax of the language that renders processing feasible by enabling users to be able to read and write through this graphical interface with the familiar notations. In the internal structure, the grammar which was created out of the concepts generates the abstract syntax trees as readable XML as shown in figure 5. The essence is for the ability to generate code and other artifacts.

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema id="eeml_2Schema"
targetNamespace="http://schemas.microsoft.com/dsltools/eeml_2"
elementFormDefault="qualified" xmlns="http://schemas.microsoft.com/dsltools/eeml_2"
xmlns:core="http://schemas.microsoft.com/VisualStudio/2008/DslTools/Core"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:import id="CoreSchema"
namespace="http://schemas.microsoft.com/VisualStudio/2008/DslTools/Core" />
  <!-- PipelineDesignModel -->
  <xsd:element name="pipelineDesignModel" type="PipelineDesignModel"
substitutionGroup="core:modelElement" />
  <xsd:complexType name="PipelineDesignModel">
    <xsd:annotation>
      <xsd:documentation>The root in which all other elements are embedded. Appears
a diagram.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="core:ModelElement">
        <xsd:sequence minOccurs="0" maxOccurs="1">
          <!-- Relationship: PipelineDesignModelHasElements -->
          <xsd:element name="elements" minOccurs="0" maxOccurs="1">
            <xsd:annotation>
              <xsd:documentation>Instances of
PipelineDesignModelHasElements</xsd:documentation>
            </xsd:annotation>
            <xsd:complexType>
              <xsd:sequence>
                <xsd:choice minOccurs="0" maxOccurs="unbounded">
                  <xsd:element ref="PipelineDesignModelHasElements">

```

Figure 5: Readable XML nodes

The names in each of the XML nodes have the same name as it is in the domain class name. For example, *PipelineDesignModels* and *PipelineDesignSystem* properties such as Name, diameter, position etc. are serialized as attributes in the XML nodes.

The relationships between the core model elements are at the same time embedded and serialized as XML nodes inside the core of the framework. For example, in the XML, this is represented by the node named *PipelineDesignModelHasElements* and in the DSL Definition, this part can be found at the *DomainRelationship* class. The result of the relationship is a serialized content of the target code at the code generation level and then a reusable editor platform showcasing different parts of the model and diagram files.

4.2. Discussion of Results

The Code Generator drives the solution created with model to the possible layer of reusable software with interconnected interface of domain notations. These

notations which are familiar with users and representing the domain model, i.e. the engineering design are carefully scripted in the interfaces for ease of use. Having applied C# *SerializableAttribute* *Moniker* class which creates a strong name for the components, single assemblies were formed basically to link the logically related code into independent layers that also makes it easy to capture specific pieces of inputs into the core domain model as shown in figure 6.

The salient technical characteristics of the domain prevalent in the domain model clarify the identity of the concepts and associated rules. These are relational abstraction rules from which the relevant vocabulary is created in terms of the physical products structures and attributes in the model such as the Domain Classes, Connectors and Elements and the concrete syntax in form of notations about the key words describing concepts mapped to design symbols.

The Code Generator actually generates all the codes that will drive the solution created with the MS VMSDK modeling tool. The models drive the template files (.tt) which drive the resultant compile (.cs) files which are the actual compiler files working the resultant language model. To build the solution model, all templates were transformed to ensure that the model items are captured and updated in the code generator; and then build and debugged to see the user experience as shown in figure 4 above.



Figure6: Core Domain Model XML Schema

5. Conclusion and Future Work

This framework is applicable in the physical design of objects in the engineering industry. It will produce artifacts that will help engineers manage very complex design concepts. The core part of it is built on the DSL processor engine that compiles the DSL Builder files at the core. At the Development Interface, the templates are created for every change or line added to our model. The abstractions would offer benefits such as removing the difficulties in creating engineering designs currently experienced by non-professional CAD users. It will, in line, remove hassles and complexities of expertise-centric design platforms.

As work progresses with accompanying requirements elicitation in the pipelines design domain, we will extend the proposed approach for more concrete language specifications that can enable editing actions that directly modify the abstract syntax tree. In such a way, the consideration will be directed along framework extensions involving transformation of the DSL models and implementation results that justify stakeholder's perspectives. As much as UML profiles provide powerful mechanisms for modelling, in the future, use-cases will be avoided as much as possible to drive home our focus on both the concepts of the problem domain as well as the technical contents of the application domains as opposed to code-centric diagram definition standard to resolving deficiencies prevalent with the object management group (OMG). The aim would be to provide engineers with hands on modeling tool with which they can readily develop designs and related systems.

References

- [1] Autodesk Inc. (2013) AutoCAD Release 2013 Programmers Reference Manual.
- [2] Alessandro NADDEO (2010), Cad Active Models: An Innovative Method in Assembly Environment, Journal of Industrial Design and Engineering Graphics Volume 5 Issue No. 1
- [3] Selić, B. (2011), the Theory and Practice of Modelling Language Design (for Model-Based Software Engineering), MODELS 2011 Wellington New Zealand
- [4] Seifert, D., Dahlweid, M. and Santen, T. (2011), A FORMULA for Abstractions and Automated Analysis, Wellington New Zealand MODELS 2011
- [5] Melgoza, E. L., Rosell, A. (2012), An integrated parameterized tool for designing a customized tracheal stent Computer-Aided Design, Vol. 44, Issue 12, pp.1173-1181
- [6] Giachetti, G., Marin, B. and Pastor, O. (2009), Integration of domain-specific modelling languages and UML through UML profile extension mechanism, Int'l Journal of Computer Science and Applications, Vol. 6, No. 5, pp. 145-174.

- [7] Vangheluwe, H. (2010), Domain-Specific Modelling Language Engineering, Lisboa, Portugal
- [8] Deng, J., Hormann, K. and Kazhdan, M. (2012), Geometric Modeling and Processing, Computer Aided Geometric Design, Volume 29, Issue 7, October 2012, 421.
- [9] Leake, J. (2012), Engineering Design Graphics: Sketching, Modeling, and Visualization. John Wiley & Sons, Inc. USA.
- [10] Stavric, M. and Marina, O. (2011), Parametric Modeling for Advanced Architecture, International Journal of Applied Mathematics and Informatics, University Press 9 16.
- [11] John Charlery, Chris D. Smith, An approach to modeling domain-wide information, based on limited points' data – part I, American Journal of Software Engineering and Applications 2013, 2(2):pp32-39 www.sciencepublishinggroup.com/j/ajsea
- [12] Lockhart, S. and Johnson, C. (2012), Engineering Design Communications: Conveying Design through Graphics (2nd Edition), Prentice Hall, USA.
- [13] Douglas C. Schmidt (2006) Model-Driven Engineering Vanderbilt University, the IEEE Computer Society
- [14] Nicola CAPPETTI (2010), Parametric Model of Lumbar Vertebra, Journal of Industrial Design and Engineering Graphics Vol. 5, Issue No. 2.
- [15] Mernik, M, Heering, J., Sloane, A. M. (2005) When and how to Develop Domain-Specific Languages, ACM Computing Surveys Vol. 37, No. 4, pp. 316-34.
- [16] Baker, S. D. and Slaby, J. M. (2006), Domain-Specific Modeling Languages for Enterprise DRE System Journal of Computers, IEEE Computer Society 2006
- [17] Kelly, S. (2007), Domain-Specific Modeling Languages: Moving from Writing Code to Generating It, [DSM Forum, 2007] "DSM Tools."
- [18] Sanna Sivonen (2008), Domain-specific modelling language and code generator for developing repository-based Eclipse plug-ins, VTT Technical Research Centre of Finland
- [19] Gustavo C. M. Sousa Fábio M. Costa Goiânia-GOPeter J. Clarke Andrew A. Allen (2012), Model-Driven Development of DSML Execution Engines, Proceedings of ACM Conference, eduMRT '12, Innsbruck, Austria
- [20] Markus Völter (2008) Domain Specific Languages Implementation Techniques voelter@acm.org <http://se.radio.net>
- [21] Petru DUMITRACHE (2011), Parametric Modeling of Rops/Fops Protective Structures Geometry In Order To Study Of Their Behaviour Using Finite Element Method, Journal of Industrial Design and Engineering Graphics Vol. 6, Issue No.