Journal of Physical Science and Innovation **Volume 5, No. 2, 2013**
ISSN: 2277-0119

DOMAIN SPECIFIC MODELING OF PARAMETERIZED OBJECTS

Japheth[1] B. R. and Ogheneove[2] E. E.
[1]*Department of Mathematics/Computer Science, Niger Delta University, Yenagoa, Nigeria*
[2]*Department of Computer Science, University of Port Harcourt, Port Harcourt, Nigeria*
E-mail: jbunakiye@yahoo.com, edward_ogheneovo@yahoo.com

ABSTRACT

Presented in this paper is a simplification platform for engineering design that redirects designers from the conventional form of vector geometry. Using programming features it will promote automatic interchange of parametric data used in the creation of graphic models and those used in the design analysis of the models in order to accomplish specific design tasks. The entire process of creating a final graphics model which is termed graphics design and which results in a graphics model will involve the aggregation of the graphics primitives, graphics assemblies and subassemblies to form the model. As a result an integrated domain of language constructs capable of enhancing the automatic data manipulation evolves. This integrated domain, which is actually an enhanced design modeling framework, will free the designer of being constrained by a limiting set of tools.

Keywords: CAD Systems, Graphics Models, Homogeneous Programming, Design Environment.

INTRODUCTION

Computer Aided Design systems [1], provide tools for building good designs that conforms to solid objects that depicts the creation of a set of drawings for the production of a model. Models are simplification of operations or systems that targets particular circumstances, and that is the more reason graphics models produced from CAD systems are used for modeling. Graphics models usually are created interactively from graphics primitives and can be assembled to more complex forms known as graphics assemblies [1, 4]. CAD systems do provide assemblies whose dimensions are functions of key parametric values produced by the user, as they are utilized by designers for graphics design modeling. But one key consideration is that common interactive CAD systems (e.g. AutoCAD) lack programmable parameters, which could enable the automatic interchange of data between related modules such as design analysis routines, graphical modeling parameters and schedules [3, 4].There is a sharp distinction, therefore, between building models, a concrete graphical editing activity, and programming, an abstract, textual, algorithm-construction activity [8]. The focus of this paper is the integration of both the graphics design and the graphics design analysis attributes in a conventional homogeneous programming platform. This can then enable usability with a viable CAD API that can solve a general workflow problem by providing a means for creating clearly refined graphics models that targets particular application areas for solutions [7, 8]. The parametric notations of this domain specific models can be captured as the necessary entities representing both the problem and application spaces in the domain, and such integration will facilitate automatic interchange of data and workflow parameters and can adequately remove the drawbacks encountered by designers in common CAD systems [5].

LITERATURE REVIEW

A research in this direction is how to combine design and programming activities in one platform. One identified approach was the extension of a visual logic programming language to incorporate the notions of solids and operations on solids [6, 9]. A model for parameterized solids in a visual design language was then developed as the engine to drive the parameterized assembly of objects [9]. This allows the designer/programmer to work at a higher level, giving declarative specifications of a design in order to obtain the design descriptions. Such methodology integrates problem solving, design synthesis, and prototype assembly in a single homogeneous programming/design environment [6, 10]. This was an effort to remove the sharp distinction between designing and the programming required to achieve parameterization. There are quite a number of challenges with present day CAD systems, some either provide poor facilities for engineering design, or provide visual design facilities without well integrated programming capabilities. The primary reason for including programming capabilities in design environments is to allow for the representation of domain specific modeling [2, 4].

Trends in Computer Aided Design

Almost all engineering design often involves much, complex diagrams with visual graphical content. One of the areas where computer-aided design is particularly useful is in the generation of these diagrams [4, 6]. However, they lack the expressive capability of textual programming languages. This suggests that although a graphical CAD system can be considered a visual design language, it is a low-level language in that it does not include higher-level concepts such as conditional execution, iteration and recursion that are found in high-level programming languages [7, 12]. Much of it model based led to several functional modeling for engineering systems, these includes functional parametric representations, visual techniques for analysis of shapes, and structural behaviours of solids. In examining research to addressing these limitations, Philip Cox and Trevor Smedley [9] proposed a declarative logical-based visual language for structured design; the advantage of this approach is the integration of concrete representations of design components with operations that specify how components are assembled from other components. The Language for Structured Design (LSD) [9, 12] contain specific features and is an extension of Lograph, a visual logic programming language, with the notion of solids and operations on them. The features of LSD include high level, visual, logic programming [9, 12], a language for design of structured objects that combines the design and programming activities in a homogeneous programming/design environment [7, 14], and at the back-end, a solid modeling kernel for maintaining low level description of solids and operations. In the context of this paper, which focuses on making the parameterized objects domain specific uses this approach as a benchmark for language development, specifying clearly that programming capabilities, however, are also necessary for addressing the shortcomings of engineering design systems. Computer-aided design covers a broad area of research focusing on supporting design processes that shift and adapt even as the underlying computational technology is evolving. Stavric *et al.,* [6] envision that the next generation of knowledge-based CAD systems will be characterized by four features: they will be based on cognitive accounts of design, and they will support collaborative design, conceptual design, and creative design. This discussion further expects CADs to be able to investigate content, organize, communicate and share knowledge, coupled with present day standards such as computational geometry, computer graphics, connectionist networks, evolutionary computing, numerical analysis, optimization,

simulation, etc. Previous generations of CAD systems and useful sources of early works and impact can be found in [13, 14].

PARAMETERIZED OBJECTS

A key theme in this paper is parametric modeling, which invariably is a basis for integration of differing possibilities of engineering design modeling in a homogenous domain specific platform. CAD systems [12] enable designers to create geometric models of products in the computer, to be reused and manipulated by the designers as needed. CAD systems were, and remain, highly technical software with extremely rich features and functions for detailed design work. They typically encode the semantics of shapes as so-called parametric features. A parametric feature-based modeler [14] is a CAD software package that uses either a Constructive Solid Geometry (CSG) or a Boundary Representation (B-Rep) modeler that allows a user to refer to features instead of the underlying geometry. That is, when a parametric modeler is used, geometry is easy to modify, features contain manufacturing as well as geometric information, dimensions are parametric, the model history is available to the operator, and there is an assembly environment [10, 11]. Features are the basic three-dimensional building blocks for creating parts. Features available in parametric modelers can copy tangible design and manufacturing features (e.g., rib, draft angle). There are two kinds of model features, sketched and placed. Sketched features (e.g., extrude, revolve) require that a 2D sketch be made before the feature can be created. Placed features (e.g., hole, chamfer, fillet, shell, face draft) can be created without sketch geometry [2, 4 and 11].

METHODOLOGY FOR MODELING PARAMETERIZED OBJECTS

This section details the concept and terminology of parametric modeling adopted to suit a specific domain of engineering designs. Parametric modeling [6] employs parametric dimensions and geometric constraints to define features, and to create relationships between these features in order to create intelligent models. These dimensions as shown in figure 1 are associatively linked to the model geometry that they describe. If a dimension changes, so does the associated geometry.

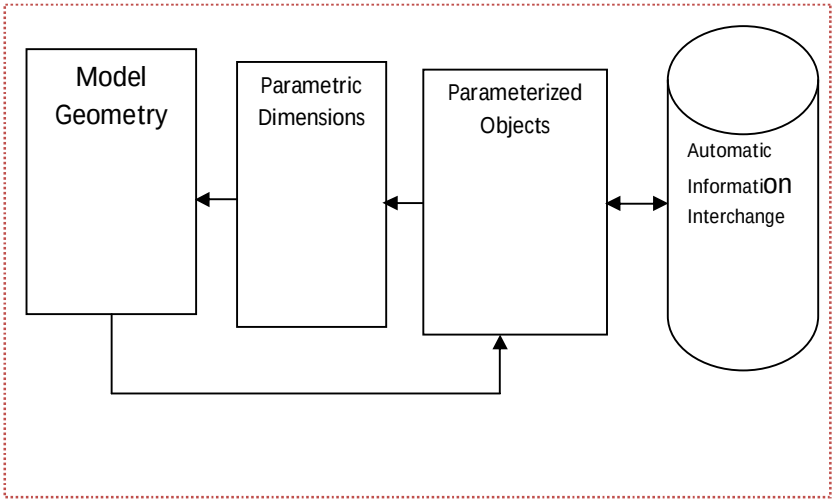

Figure 1: Flow and Interchange of Information through Parameterized Objects

Though this flexibility is often utilized by designers to satisfy customers' needs for product varieties, but in conventional modeling where an object is explicitly described, when one aspect of the model is changed, often several changes has to be made to satisfy a design intent or the implicit rules of the design [1]. What this means is that this CAD system couldn't automatically display the implicit rules that govern the design intent whenever a change has to be effected in the parametric constrains of the designed objects. So the application of domain specific modeling can create a domain specifically tailored designing that can be routed to automatically permit flow and interchange of information through the parameterized objects [15]. Having considered the presence of programming environment in an interactive CAD system, a resulting integrated domain of parameters set for the design purpose, coupled with our own specifications is now possible in the application of domain specific methodology. This possibility is shown in figure 2 as the XML grammar content that drives the integrated domain specific modeling peculiarity platform [16].

```
<?xmlversion="1.0"encoding="utf-8"?>
<xsd:schemaid="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:importid="CoreSchema"namespace="http://schemas.microsoft.com/VisualStudio/2008/DslTo
ols/Core" />
<!--DesignModel-->
<xsd:elementname="DesignModel"type="DesignModel"substitutionGroup="core:modelElement" />
<xsd:complexTypename="DesignModel">
<xsd:annotation>
<xsd:documentation>The root in which all other elements are embedded. Appears as a
diagram.</xsd:documentation>
</xsd:annotation>
<xsd:complexContent>
<xsd:extensionbase="core:ModelElement">
<xsd:sequenceminOccurs="0"maxOccurs="1">
<!-- Relationship: DesignModelHasElements-->
<xsd:elementname="elements"minOccurs="0"maxOccurs="1">
<xsd:annotation>
<xsd:documentation>Instances of DesignModelHasElements</xsd:documentation>
</xsd:annotation>
<xsd:complexType>
<xsd:sequence>
<xsd:choiceminOccurs="0"maxOccurs="unbounded">
<xsd:elementref="DesignModelHasElements">
<xsd:annotation>
```

Figure 2: XML Grammar Content

This integrated domain, which is actually an enhanced design platform, will propagate creative thinking and not the designer being constrained by a limiting set of tools. It is a key to innovative designs and design processes [15, 16].

THE FUNCTIONAL STRUCTURE

Two popular data structures used in the description of solid models are [7], Constructive Solid Geometry (CSG) and Boundary Representation (B-Rep). The CSG representation [8], stores

the model data in a tree structure in terms of the solid primitives and the Boolean operations that are used to combine them. Because CSG organize data solely in terms of primitives and Boolean operations, constructive models represent a solid as a combination of primitive solids as well as the result of applying Boolean operators to a set of instantiated and transformed CSG primitives. Therefore, in all of the (CSG) examples it is clear knowledge that the design of the solid is not unique [9]. On the other hand [7], boundary representation stores the boundaries of the solid (e.g. vertices, edges, faces) in the database along with information regarding how these entities are connected. Because B-Rep includes such topological information, a solid is represented as a closed space in 3D space. Either of these approaches requires users to go through a lot of rigor during design, some need to consult other experts for algorithm and programming constructs to be able to achieve the desired results. This present structure [18] as shown in figure 3 is a combination of both CSG and B-rep to be seen as one model entity. This will allow for the provision of an integrated API within a solid modeler (AutoCAD) that can allow for automatic exchange of data during design [15].

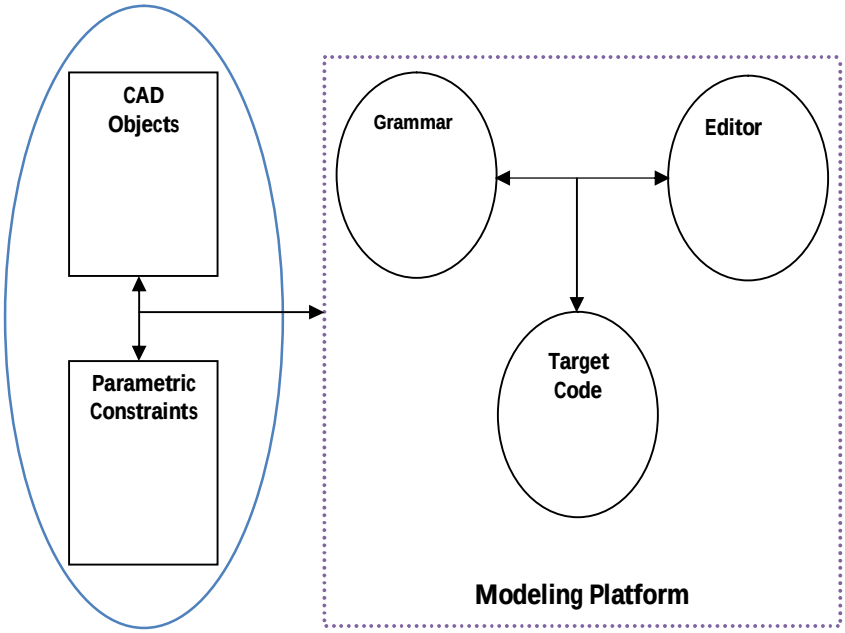

Figure 3: Integrated Modeling Platform

The design is purely based on parametric modeling techniques built into the language, and the model representing a particular domain of interest will be domain specific [17]. Shown in figure 4 is the syntax structure of the language construct designed to enable domain specific modeling of the CAD objects as the domain model with parametric constraints instead of depending on the static instructional logic of dump solids.

```
</xsd:annotation>
</xsd:element>
<xsd:elementref="System">
<xsd:annotation>
<xsd:documentation>[Target role-player]</xsd:documentation>
</xsd:annotation>
</xsd:element>
</xsd:choice>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
<!-- Id -->
<xsd:attributename="Id"type="xsd:string">
<xsd:annotation>
<xsd:documentation>Instance Guid of this element, needed because SerializeId is set to
true.</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<!--dslVersion-->
<xsd:attributename="dslVersion"type="xsd:string">
<xsd:annotation>
<xsd:documentation>Version of the model serialized in this file.</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType
```

Figure 4: Language Syntax

Therefore with a common API with domain specific definitions, a user can easily achieve a result through the automatic exchange of data as shown in figure 5.

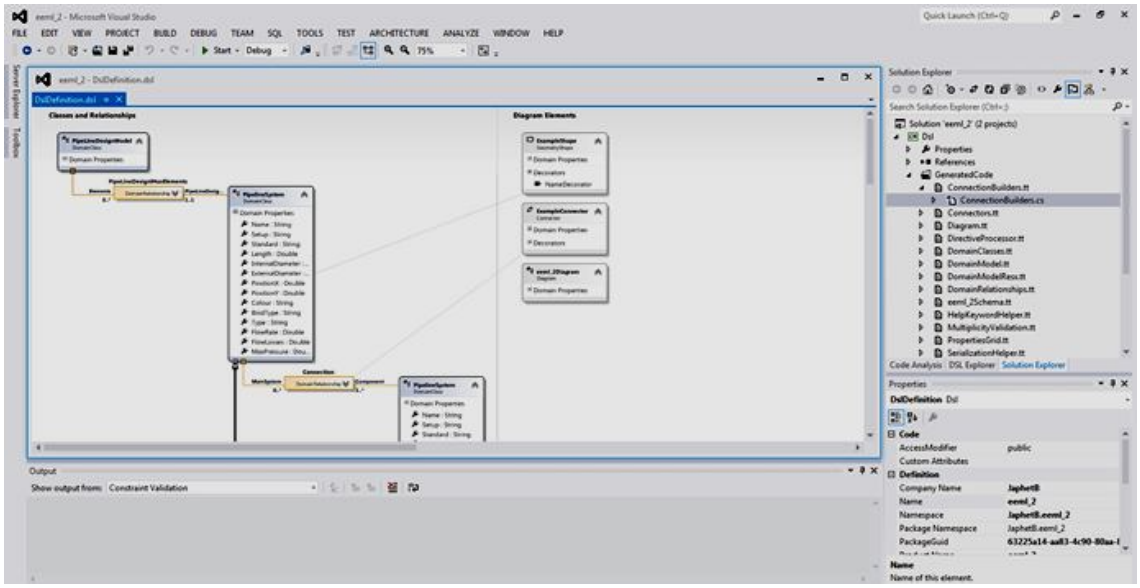

Figure 5: User Experience

CONCLUSION AND FUTURE WORK

This present work will overlook the strict syntax definitions in a host language and tends to go beyond the capabilities of parametric modeling to include more graphics design attributes. Re-engineer the graphics analysis modules with wider and most recent programmable parameters by employing relevant data structures, variables that correspond explicitly and implicitly with variables and data structures of the graphical models spatial forms. Having considered the presence of programming environments in interactive CADs such as AutoCAD coupled with user defined specifications and methods resulting into an integrated domain of parameters set for the design purpose, a domain specific modeling framework was created to permit automatic interchange of data. This integrated domain, which is actually an enhanced design modeling framework, will propagate creative thinking and not the designer being constrained by a limiting set of tools. In the future, a reusable layer of interconnected software can be created to enable the definition of common parameters in a domain specific language to ensure parameterization of both graphics modeling and the graphics design analysis processes so as to avoid manual intervention in the use of such graphics models.

REFERENCES

1. Autodesk Inc. (2013) *AutoCAD Release 2013 Programmers Reference Manual*.

2. Alessandro NADDEO, Cad Active Models: An Innovative Method in Assembly Environment, *Journal of Industrial Design and Engineering Graphics* Volume 5 Issue No. 1 – 2010.

3. Jiansong Deng, Kai Hormann, Misha Kazhdan, Geometric Modeling and Processing 2012 *Computer Aided Geometric Design, Volume 29, Issue 7, October 2012, 421*.

4. J. Leake (2012) Engineering Design Graphics: Sketching, Modeling, and Visualization. John Wiley & Sons, Inc. USA.

5. Kaskil, D.J, Buxton, W, Ferguson, D.R. (2005), Ten CAD Challenges. IEE Computer Graphics and Applications, 25(2): 81-92.

6. M. Stavric and O. Marina (2011), Parametric Modeling for Advanced Architecture, *International Journal of Applied Mathematics and Informatics, University Press* 9- 16.

7. Neil C. Katz, Skidmore, Owings & Merrill, LLP (2007), Parametric Modeling in AutoCAD, AECbytes Viewpoint Issue #32.

8. P.T. Daniel, (2009) Cohabitation of Programmable Parametric Graphics Modeling and Design Analysis, *PhD Theses*, DCIS, Birkbeck College, London.

9. P.T. Cox & T. Smedley, A Formal Model for Parameterized Solids in a Visual Design Language, *Journal of Visual Languages and Computing* 6(6), Academic Press, (2000) 687-710.

10. Sohy, C., Wang, Z. (2000) Parametric Coordinator for Engineering Design. *Journal of Computing in Civil Engineering*, 14(4): 233-240.

11. Nicola CAPPETTI, Parametric Model of Lumbar Vertebra, *Journal of Industrial Design and Engineering Graphics* Volume 5 Issue No. 2 – 2010.

12. S. Lockhart and C. Johnson (2012) Engineering Design Communications: Conveying Design through Graphics (2nd Edition). Prentice Hall, USA.

13. Carmen POPA, Mathematical Methods to Determine the Intersection Curves of The Cylinders, *Journal of Industrial Design and Engineering Graphics* Volume 5 Issue No. 1 – 2010.

14. Evila L. Melgoza, Lídia Serenó, Antoni Rosell, Joaquim Ciurana, An Integrated Parameterized Tool for Designing a Customized Tracheal Stent *Computer-Aided Design, Volume 44, Issue 12, 2012,1173-1181.*

15. Angel Roman & Bruce Trask (2011), Applying Model Driven Technologies in the Creation of Domain Specific Modelling Languages, *Proceedings of 14th International Conference on Model Driven Engineering Languages and Systems Wellington New Zealand*.

16. BranSelić (2011), The Theory and Practice of Modelling Language Design (for Model-Based Software Engineering), *Proceedings of 14th International Conference on Model Driven Engineering Languages and Systems Wellington New Zealand*.

17. Dirk Seifert, Markus Dahlweid and Thomas Santen (2011) A FORMULA for Abstractions and Automated Analysis, European Microsoft Innovation Center (EMIC), Microsoft Research, MODELS 2011.

18. Gustavo C.M. Sousa Fábio M. Costa Goiânia-GO Peter J. Clarke Andrew A. Allen (2012), Model-Driven Development of DSML Execution Engines, *Proceedings of ACM Conference*, eduMRT '12, Innsbruck, Austria.

Reference to this paper should be made as follows: Japheth B.R. and Ogheneove E.E. (2013), Domain Specific Modeling of Parameterized Objects. *J. of Physical Science and Innovation*, Vol. 5, No. 2, Pp. 103 – 110.