Vol 8. No. 1 Issue 2 – May, 2015
African Journal of Computing & ICT

www.ajocict.net

Model Driven Design of a DSL for Transmission Pipelines: A Roadmap

J.R. Bunakiye
Dept. of Mathematics/Computer Science
Faculty of Science
Niger Delta University
Wilberforce Island, Nigeria
jbunakiye@gmail.com

E.E. Ogheneovo
Department of Computer Science
University of Port Harcourt
Port Harcourt, Nigeria
edward_ogheneovo@yahoo.com

ABSTRACT

This paper presents an overview of the application of domain specific modeling (DSM) approach in Model-Driven Engineering (MDE) technologies to describe a domain specific language (DSL) software development roadmap. An approach in which models of transmission pipelines in the domain of pipeline engineering systems are created and systematically transformed to actual implementations. The discussion centered on some of the major transformation paths that must be undertaken in order to realize the impressions of the DSL development. Clearly, the full realizations of the vision for creating a DSL is made possible by providing language instances and modeling primitives from domain concepts that were used to significantly reduce the gap between design intents and the expression of such intents in several lines of syntax representations.

Keywords: Domain Specific Language, Modeling, Pipeline Engineering, Design Intent, Domain Model, Meta model

African Journal of Computing & ICT Reference Format:
J.R. Bunakiye & E.E. Ogheneovo (2015): A Distributed Database Architecture For Location Independent Scheme In Mobile Networks. Afr J. of Comp & ICTs. Vol 8, No. 1, Issue 1. Pp 137-144.

1. INTRODUCTION

The focus is on the development of a domain specific language (DSL) for modeling oil and gas pipeline systems. Traditionally the aim is to describe a roadmap to developing a cohesive tool for the design of such artefacts through the use of disparate tools [7]. The approach to the development of the DSL is based upon Model Driven Engineering (MDE) technologies. The two main schools of MDE are Model Driven Architecture (MDA), and Domain-Specific Modelling (DSM). MDA language specifications restrict the user to diagram definition standards (e.g. UML), whereas DSM languages identify the problem and the goal to be reached during the process provided.

Primarily, the DSM approach to the complex problem of efficiently and effectively aiding the engineer in the design and implementation of pipeline configurations was adopted [9]. General purpose software design and development tools usually adhere to a protocol. For example, a particular API call sequence that is required to create executable commands. Generally, these rules are at the same level of abstractions within the mechanism that implements them; the problem is that it creates a substantial semantic gap between design intent and the expression of this intent in several lines of codes. These possibility has posed lots of problems in the pipeline engineering work place. The motivation is to solving this problem by coming up with a domain specific language formalism that separates the policies and the mechanisms that implements them. Defining a metamodel about the domain model is suited with focus on closing the semantic gap by mapping domain concepts to appropriate levels of abstractions so that design intents and viewpoints can be freely expressed through domain specific modeling (DSM).

Domain-Specific Modelling (DSM) is about defining a model in some language formality, which is metamodeling. Metamodeling facilitates the rapid, inexpensive development of domain-specific languages (DSLs) that hides code centred development to a language formalism that enables simple expression of intents through guided notations. In the metamodel, each of the syntactic and semantic DSL components is defined precisely and completely using AutoCAD objects as the Pipeline Context Model representing typical physical components of Oil and Gas transmission Pipeline Systems [2].

The main contribution is that it moves toward an infrastructure for DSL design that integrates formal specifications with practical pipeline engineering principles, which is in principle, a rule processing system based on Syntax-Directed Translation [10]. The remainder of this paper is organized as follows: In section 2 we discuss the foundation and the open problems. In section 3 we provide an overview about the domain terminology and models. In section 4 the tools set in model driven engineering for DSL development are described. Section 5 describes the language idea. Section 6 summarizes and concludes the paper.

2. FOUNDATION

It is a common agreement that general purpose languages are extremely useful for describing software and applications. Despite their usefulness, they are never called modeling languages; the reason being that metamodeling is often lacking and at the same time domain specific concepts cannot be adequately expressed. The research roadmap in this paper is therefore showcasing a carefully crafted DSL design path for modeling pipeline systems in the domain of transmission pipelines engineering. Joerg.Kienzle et al. [20] discussed a crisis management systems showcasing a case study for aspect-oriented modeling. The intent is to define a common case study for systems that help in identifying, assessing, and handling a crisis situation through the involvement of all parties in handling the crisis, by allocating and managing resources, and by providing access to relevant crisis-related information to authorized users.

The systems in the crisis management system complimented ours but the submission of Bernhard, S., and Fortiss, G. G., [19] From Solution to Problem Spaces: Formal Methods in the Context of Model-Based Development and Domain-Specific Languages clarifies more on the domain specific modeling paradigm for providing software solutions in the domain-oriented problem space. Markus Voelter [22], also asked a question in his work, Domain specific - a binary decision? In the subsequent arguments, the difference between general purpose and domain specific languages were made very clear. The clear indication is that domain specific design is simply one pattern that is continuous, declarative and productive, especially when it relates to a domain.

DSLs can be graphical, constraint-based, textual or descriptive, and can be executable [5]. Graphical languages use diagram techniques with named symbols that represent concepts and relationships. A typical graphical modeling language is *Behaviour Trees.* Textual modeling languages use standardized keywords accompanied by parameters to make computer-interpretable expressions. An example is *TVL* (A Text-based Variability Language) [16]. Constraints-based modeling languages do not specify a step or sequence of steps to execute, but rather the properties of a solution to be found. Typical examples include VHDL, and AutoCAD. Executable modeling languages often includes the idea of code generation: automating the creation of executable source code directly from the domain-specific language models. An example is SysML (Systems Modelling Language). The structure and behaviour of the domain specific modelling language in perspective, is an integrated functionality allowing the user the flexibility of working with familiar notations, and yet able to effectively express the constraints and limitations of the proposed network [18].

3. TRANSMISSION PIPELINES ENGINEERING DOMAIN

This section contains the description of the domain concepts as a prerequisite step to the definition of the DSL metamodel. The description of concepts entails the characteristics, functions, and design criteria of a suit of AutoCAD objects as the pipeline physical components [3]. This is to ensure that all the objects are collectively seen to be the model for the DSL, and also to make sure the vocabulary necessary for the subsequent language specification are captured, which will invariably form the instance of the language creation.

3.1 Pipeline Systems Model

The design of a pipeline system requires the knowledge of physical components, physical attributes, and materials factors. Physical components, which are represented as AutoCAD objects include pipe, tee, joint and valve [2]. Tee, joint and valve when joined to a pipe results into a pipeline system (see fig. 1). Physical attributes are those parameters that govern the size, layout, and dimensional limits or proportions of the pipeline [15]. Material factors relate to pipeline design and highlights parameters that must be considered in completing a modeling process.

Figure 1: Pipes Fitted with Valves and Gauges in a Pipeline *(Source: Shell Nig. Ltd)*

3.2 Valves

A valve is a device that regulates, directs or controls the flow of a fluid (gases, liquids) by opening, closing, or partially obstructing various passageways. In an open valve, fluid flows in a direction from higher pressure to lower pressure [1]. The valve model shown in figure 2 comprises of various components like: bolts and nuts, hollow pipes, flanges, a hollow sphere, torus, cylinder, and so on. The circular body that will contain the channels and vents through which fluid are controlled was designed first through the usage of two spheres of required sizes, with the smaller one subtracted from the bigger one to form the hollowness of the body. Next, a cone frustum made hollow through subtraction was attached to the base of the sphere (with the bigger end of the frustum facing the base of the sphere).

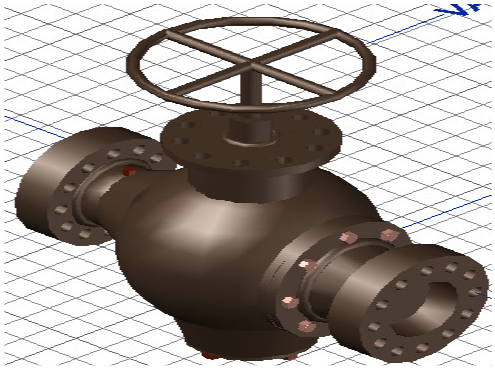

Figure 2: Valve

A cylinder, with a plate-like height, that is almost proportionate to the diameter of the smaller end of the frustum was designed and moved through the centre point and centralized at the base of the smaller end of the frustum (using 2D wireframe from virtual style instead of realistic). The right and left side of the main body of the hollow sphere were also attached with the bigger end of the cone frustum. Flanges are also attached to the ends of the smaller sides of the just attached frustum, using move tool and picking it from centre points to centre points, using all the necessary views like top, left, right, and southwest isometric views.

A sizable torus was then attached to the interface between the flanges and the smaller end of the frustum [2]. To design the control wheel of the valve, the combination of torus and cylinder was employed, a torus of a desired size was drawn and two cylinder with a diameter slightly smaller than that of the tube radius of the torus was drawn and laid diagonally in - to - in inside the drawn torus, using top, front and isometric views (to form the handle of the wheel). Another sizable cylinder is placed vertically from the centre of the just designed wheel to link the control wheel with a bigger cylinder placed at the top of the hollow sphere.

A cylinder was used to design a plate, and another smaller cylinder was arrayed round this plate and was subtracted from the plate to form a perforated plate, which was then placed at the interface between the smaller and bigger cylinder between the wheels, picking as always from centre point to centre point as base points. Finally, materials are added to appropriate components to produce a nice finishing.

1) 3.3 Tee

A tee is a short piece of pipe with a lateral outlet, it is a common pipe fitting, used to either combine or split a fluid flow. It is a type of pipe fitting which is T-shaped having two outlets, at 90° to the connection to the main line. A tee is used for connecting pipes of different diameters or for changing the direction of pipe runs. They are extensively used in pipeline networks to transport two-phase fluid mixtures [17].

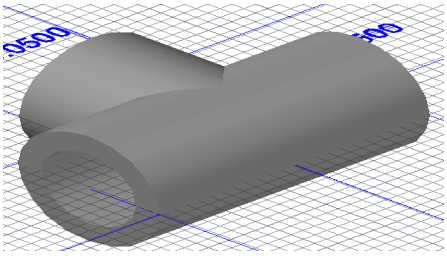

Figure 3: Tee Fitting

2) Shown in figure 3 is the example tee model from AutoCAD. To obtain the model, a hollow pipe is sliced and placed at a cross-road to each other to form a shape like the letter "T"- from where the name was derived. The pipes arranged in the afore-mentioned manner are then union to form a single entity, before they are then finished by the addition of materials [2].

3.4 Compression Joints

A model of compression joints are shown in figure 4. These joints are used to join plain end pipe without special end preparations. Advantages include the ability to absorb a limited amount of thermal expansion and angular misalignment and the ability to join dissimilar piping materials, even if their outside diameters are slightly different [2].

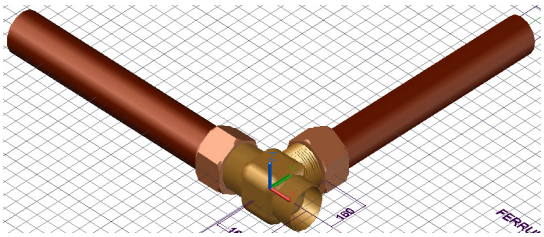

Figure 4: Compression Joint

3.5 Pipe Cross Section

The pipe cross section is the major component in the pipeline, linked and connected by other components. To obtain our pipeline context model of the pipe, the nominal diameter or outside diameter and the inner diameter of the pipe dimensional standards were specified. Also specified are the pipe directions (from point to point), the pipe length, and slope [3]. The specifications were made particularly to get a simple uniform pipe sizing in the pipeline. Getting the model from AutoCAD, with the dimensions as shown in figure 5, the sweep method was used; by using polyline from the draw tool bar, to draw the required pipe length and shape. thereafter, two circles that represents the inner and outer diameter of the pipe was drawn near one end of the pipe, then, the inner and outer circles were each swept across the drawn line to finally get the pipe, the inner pipe was subtracted from the outer one to get the hollowed required pipe and appropriate library materials applied to the pipe to give it a realistic appearance [2].

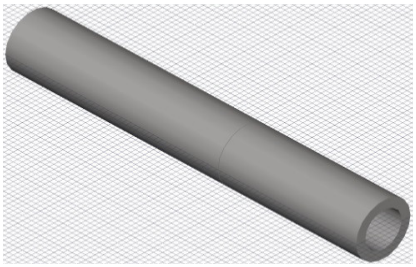

Figure 5: Pipe Cross Section

4 The Model Driven Tool Set in the DSL

This section contains an extensive case study that illustrates the various ways of integrating language modules; these include what's traditionally considered "programming" and what's traditionally considered "modeling". The case study is from the embedded systems domain. Let's start with some background.

4.1 Model Interactions

Clearly, with existing modeling approaches (e.g. modelling with UML, AutoCAD and Programming Languages), typical pipeline objects are customarily explicitly described [2]. The custom is that, when one aspect of the model is changed, often several changes have to be made to satisfy design intent or the implicit rules of the design. This is because the software [13] does not keep track of the rules and the user must decide where and when they are broken. For example, in figures 1 to 5, when defining their dimensions in a pipeline design, several changes have to be made until the exact points of intersection are met, which means until the rules governing the solid behaviours for that design intent are met no design intent can be achieved [14]. The challenge with such expressions is that model interaction in the way of domain concepts that can produce other complete models with noticeable domain properties is limited. Even with integral third generation programming APIs such modeling approach, still lack sufficient linguistic power to handle domain complexities and hasn't moved speedily with domain technologies [18]. Domain technologies in this context refers to model driven methodologies used to foster model interaction in order to create new objects that encapsulates and relates 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 levels of productivity [8]. Major efforts in model driven design are putting the model at the core of development and ignoring any detail not relevant to the application domain perspectives that these models represent [6]. 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 the designs. 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 (the application domain and the problem domain) can be integrated to form a whole modeling DSL platform.

4.2 Representation and Functionality

The design intent of stakeholders and domain experts are characterized as the view points of the input parameters [6] in the language logic. There are competing design requirements among stakeholders; each one has their own set of constraints, objectives and responsibilities. Whereas stakeholders' objectives describe the bit of problem(s) addressed by the typical DSL tool, the responsibilities describe associated design intents. Accommodating these determining factors in the language design gives thorough domain representation and functionality with an optimal solution, so that stakeholders who are non-programmer pipeline engineers can be shelved from the complexities associated with conventional modeling to express their design intents easily [12]. The representation need to start with the domain model specifications, clearly identifying the vocabulary and key concepts of the problem domain, and also identifying the relationships and attributes of all the entities within the scope of the problem domain. Closely tied to the DSM manifesto, which says "Raise the Abstractions and Hide the Complexities" the language functionality has to move the representation further to an application model for design intents declaration and editing, and for artefact orientations [11].

5. The Language System and Case Study

This section presents the diagrammatical description of the research roadmap. It also contains a case study that illustrates a way of expressing design intents in the DSL editor.

5.1 Research Roadmap

Adopting the Domain Specific Modeling (DSM) paradigm for language specifications (see figure 6), the focus is more on requirements within the oil and gas pipeline engineering domain [21]. The structure and behaviour of the prototype system should capture stakeholders design intents that depict various pipeline design scenarios prevalent in the domain under consideration.

www.ajocict.net

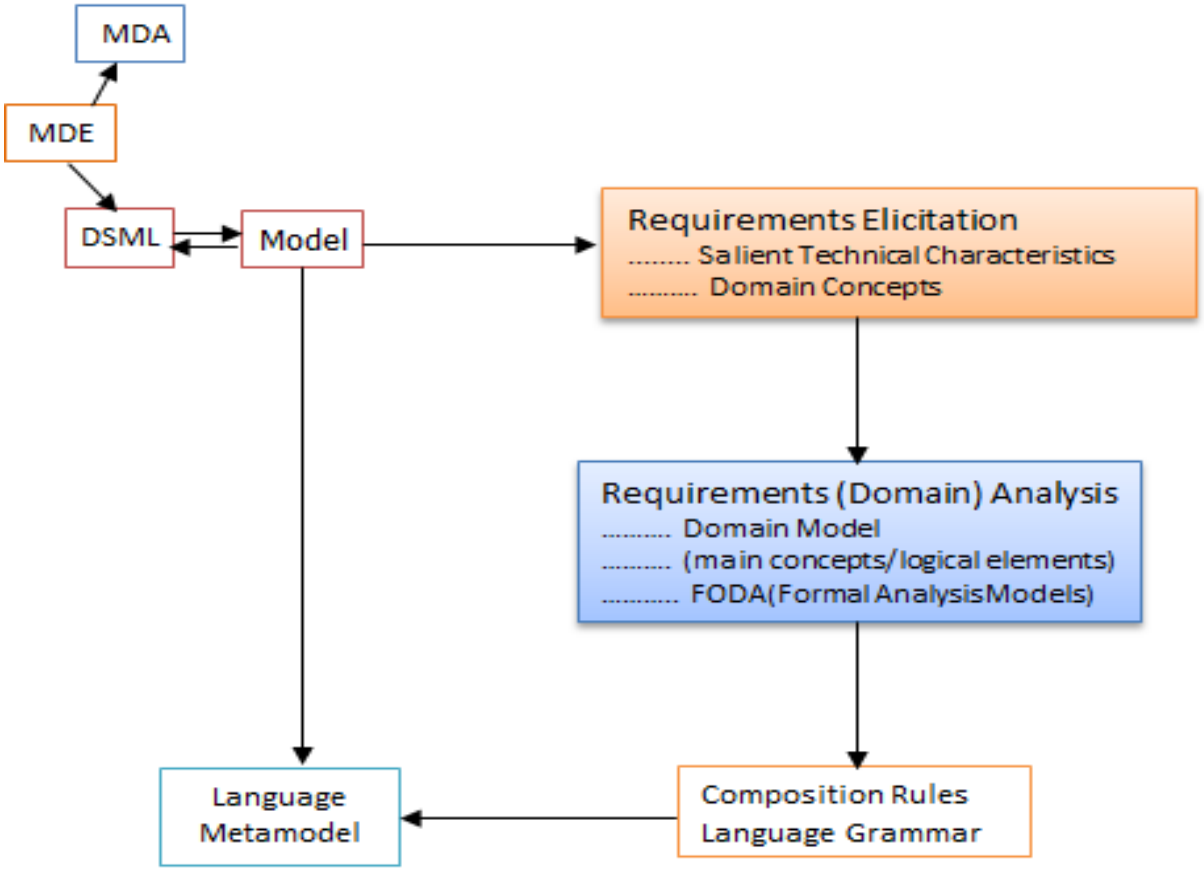

Figure 6: Research Roadmap

Industry knowledge were sequenced through the company's technical documents and crew engineers [6]. The acquired knowledge became the domain knowledge for the formal analysis and subsequent language construction via feature oriented domain analysis (FODA) [5]. Some of the key requirements for the system work flow include a semantic model for user perspectives, and a user interface component with familiar notations, permitting its users to represent their mental models about their design intents

5.2 Components Grammar and Case Study

The syntax and semantic definitions of the language were clearly defined to exemplify the approach and contribution to knowledge. The semantics are precisely defined as operational units to capture concurrency, and communication abstractions of the features of the pipeline product family [5]. The grammar also incorporated the vocabulary and associated attributes specified as denotational units representing feature relationships to be part of the solution model [4]. The theory of the internal working mechanism of the solution model of the system is a core component of DSM [11]. The pipeline components grammar is the collection of the modeling primitives and the rules connecting them as the syntactic elements [50]. They include pipeline components (c), pipeline fittings (f), pipeline joints (j), pipeline bed (b), pipeline supports (s), and the necessary interdependencies in the form of character set, expressions, and statements (see figure 7).

www.ajocict.net

Character Set	Statements	Expressions
$c \Rightarrow type, dimension, point(ct, cd, cp)$	$stmt_1 \rightarrow r = c * f * j + s - b$	$c\text{-}type \Rightarrow pipe, bolts, valve$
$f \Rightarrow type, dimension, point(ft, fd, fp)$	$stmt_2 \rightarrow c = ct * cd * cp$	$c\text{-}type \Rightarrow gauge, meter, gasket$
$j \Rightarrow type, dimension, point(jt, jd, jp)$	$stmt_3 \rightarrow f = ft * fd * fp$	$c\text{-}type \Rightarrow union, strainer, tank$
$b \Rightarrow law, route, location(bl, brt, bln)$	$stmt_4 \rightarrow j = jt * jd * jp$	$f\text{-}type \Rightarrow reducer, elbow, tee$
$s \Rightarrow type, dimension, point(st, sd, sp)$	$stmt_5 \rightarrow b = l * rt * ln$	$j\text{-}type \Rightarrow langed, grooveed$
		$s\text{-}type \Rightarrow hanger, anchor$

Figure 7: Statements, Expressions and Character Set

Figure 8 is an example of a modeling action, where a typical design intent is expressed with a resultant system curve depicting model selection for a particular pipeline project.

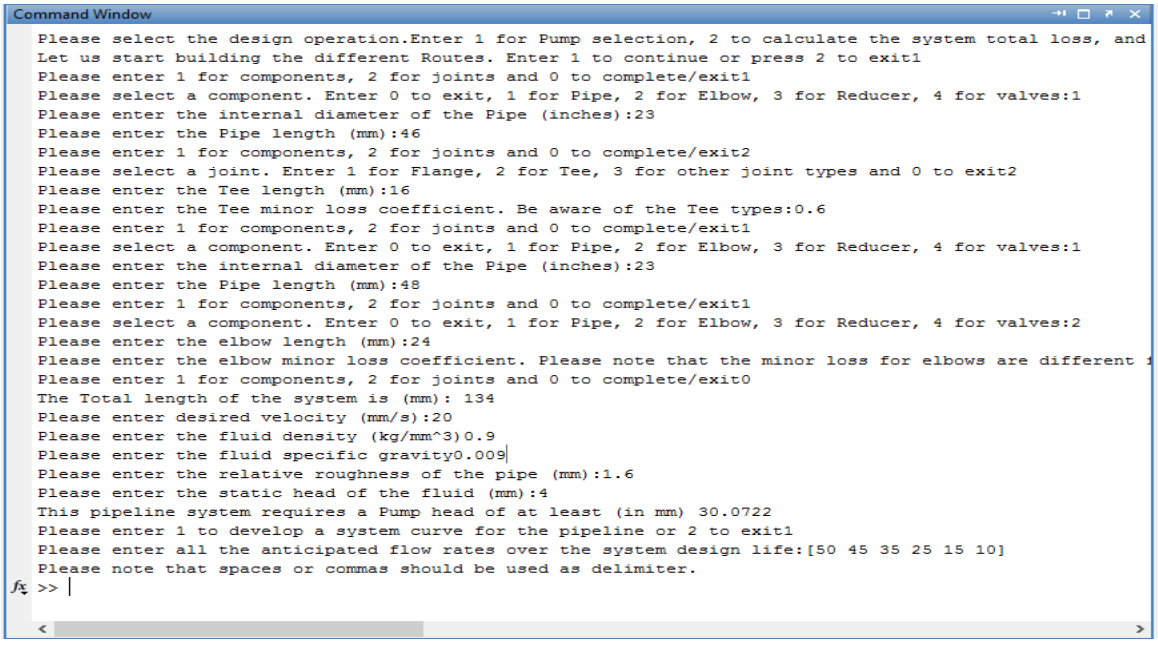

Figure 8: Expressing Stakeholder Viewpoints

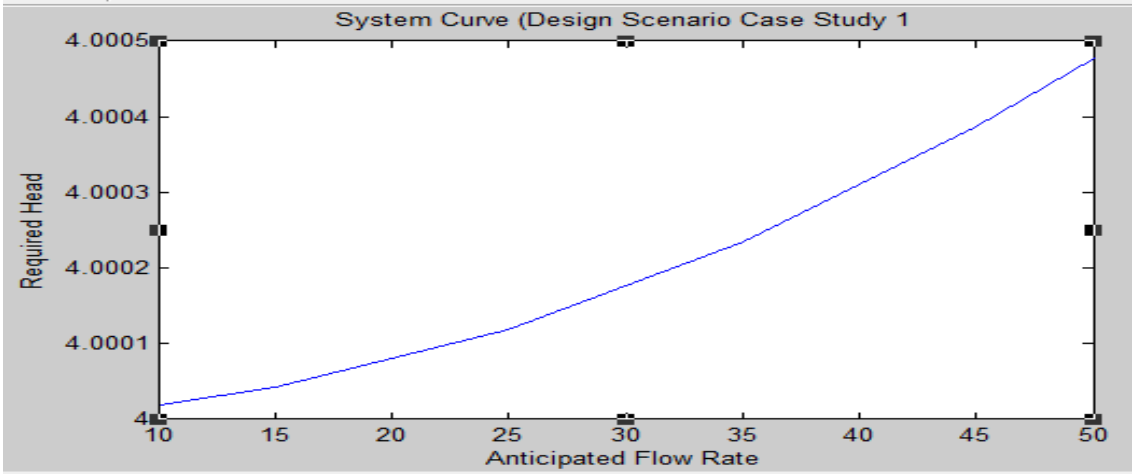

Figure 9: Design Scenario System Curve for Case Study

A case study for the particular design scenario signifying the intents of a stakeholder is given in figure 9. The case study illustrates pipeline physical components selection, cost effective control, productivity, and project quality control.

6. CONCLUSION AND FUTURE WORK

Adopting the domain specific modelling (DSM) approach, a methodology for creating domain specific languages based on domain knowledge and metamodeling is presented. Clear specifications of a domain model from the domain of transmission pipelines engineering are formalized into a collection of modelling primitives and the grammar rules connecting them. There is informal domain descriptions that incorporates all of the units and their relationships in the manner that captures user view, and depicts what the current systems must do in the pipeline design domain. The resultant effect of the internal working mechanism is that several design scenarios can be tested for pipeline route and pump selections. Each of these scenarios are case studies resulting into the evolution of a system curve depicting a particular pipeline project as viewed by a stakeholder. This approach saves time, does not require the engineer to have any programming or CAD expertise to achieve results.

References

[1] [1] Andrade, F.A. (2011), Asymptotic Model of the 3D Flow in a Progressing-Cavity Pump SPE Journal Volume 16, Number 2, 451-462.

[2] Autodesk Inc. (2013) *AutoCAD Release 2013 Programmers Reference Manual.*

[3] Anvil International. (2012), Pipe Fitters Handbook, University Park, IL United States.

[4] [4] Alfred, V. A., Ravi, S., and Jeffrey, D. U. (2007), Compilers *Principles, Techniques, & Tools,* Pearson Education, New York

[5] Bontemps, Y., Heymans, P., and Schobbens, P. Y. (2005), Generic semantics of feature diagrams variants, in: Proceedings of 8th International Conference on Feature Interactions in Telecommunications and Software Systems, 58-70.

[6] *B.G. Technical* LTD (2013), *B.G. Technical* Oil & Gas industry Port Harcourt, *Nigeria; www.bgtechnical.com/* Annual Reports 2013

[7] Bran, S. (2011), 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, 1-18.*

[8] Batory, D. S. (2005), Feature models, grammars, and propositional formulas, in: Proceedings of the 9th International Conference on Software Product Lines (SPLC'05), 7-20.

[9] [9] Christian, H., and Klaus, F. (2009), Domain Specific Modeling Language for Multiagent Systems, German Research Institute for Artificial Intelligence (DFKI); Springer-Verlag Berlin Heidelberg, 56 – 66.

[10] David, A. S. (1997), Denotational Semantics: A methodology for language development *Department of Computing and Information Sciences,234 Nichols Hall, Kansas State University, Manhattan, KS 66506. schmidt@cis.ksu.edu*

[11] Eric, E. (2003), Domain-Driven Design: Tackling Complexity in the Heart of Software Addison Wesley, USA.

[12] Juha-Pekka, T. (2011), Implementing Your Own Domain-Specific Modelling Languages: Hands-on, ICM-International Congress Centre Munich, Germany.

[13] Kaskil, D. J. W., Buxton, D., and Ferguson, R. (2005), Ten CAD challenges, IEE Computer Graphics and Applications 25(2), 81-92.

[14] Lee, E. A. and Zheng, H. (2005), Operational semantics of hybrid systems, in Proceedings of Hybrid Systems: Computation and Control (HSCC), vol. LNCS 3414. Springer, 25–53.

[15] Mark, N. (2012), Pipeline Route Selection Project, SR/WA Right of Way 2012.

[16] Martin, F. (2010), Domain Specific Languages, Addison-Wesley Professional. USA.

[17] ***Neil, C. K., Skidmore, O., and Merrill, L. P.*** (2007)**,** Parametric Modeling in AutoCAD, AEC bytes Viewpoint Issue #32.

[18] Philip, J., and Roggenbach, M. (2014), Encapsulating formal methods within domain specific languages: A solution for verifying railway scheme plans, Mathematics in Computer Science 8 (1) (2014) 11-38.

[19] Bernhard, S., and Fortiss, G. G., (2011), From Solution to Problem Spaces: Formal Methods in the Context of Model-Based Development and Domain-Specific Languages, 35th IEEE Annual Computer Software and Applications Conference, 445 - 455

[20] Joerg, K., Nicolas, G., and Sadaf, M. (2009), Crisis Management Systems: A Case Study for Aspect-Oriented Modeling School of Computer Science, McGill University, Montreal, Canada

[21] Bashar, N., and Steve, E. (2000). Requirements Engineering: A Roadmap Department of Computer Science Imperial College 180 Queen's Gate 6 King's College Road London SW7 2BZ, U.K.

[22] Markus Voelter (2010), Domain Specific - a Binary Decision ? Ötztaler Strasse 3870327 Stuttgart, Germany voelter@acm.org