Defining a DSL for Transmission Pipeline Systems Metamodeling

Bunakiye R. Japheth
Department of Computer Science
Edo University Iyamho
Auchi, Nigeria
jbunakiye@gmail.com

Acheme I. David
Department of Computer Science
Edo University Iyamho
Auchi, Nigeria
ijeggs@gmail.com

***Abstract*—Transmission pipeline systems metamodeling is simply reengineering pre-constructed notations and abstractions of the pipeline engineering domain in a form that offer expressive power for the domain expert to create designs that suits the intended transmission pipeline project. The required formality that can provide such expressive power is a domain specific language (DSL), the domain specific modeling approach, therefore is adopted to create a domain specific platform where the specification primitives represent abstractions and conceptual modeling processes in the design and implementation of transmission pipeline configurations. Domain specific languages, which are centered on meta-modeling raises the level of abstraction beyond programming by specifying the solution directly using domain concepts. The conceptual DSL definition brings to bear domain abstractions, and expressive power restricted to, the domain of transmission pipelines for the related products in the petroleum industry and in water supply. Consequently this can be achieved only by taking advantage of specific properties of the pipeline engineering application domain that pertain to transmission. The description of these specific properties therefore represents the domain concepts, which will be useful in creating the abstractions and in the semantic mappings of the elements of the DSL modeling platform.**

Keywords—Formal specifications; semantic mappings; petroleum industry; pipeline design; modeling platform

I. Introduction

Transmission pipelines are the most common means of transporting oil or gas [1]. They are used to transport large volumes over long distances to major markets. These oil and gas products are introduced into a pipeline transmission system at various terminals, processing plants near supply fields, and interconnections with other transmission pipelines. Transmission pipeline also delivers natural gas to large industrial end-users, to homes and businesses for heat and energy. Major characteristics of transmission pipelines are that they are long and continuously welded flow lines with a number of curves and no sharp bends. These properties of transmission pipelines mean that small sections of pipeline are not easily removed for maintenance and consequently great care is taken to prevent problems arising in the first place. A pipeline is extremely expensive to lay, especially in the case of offshore pipelines [4]. Though maintenance on pipelines is expensive; they frequently form the most efficient and cost-effective method of transporting the quantities of oil or gas produced.

The industry encompasses a range of different activities and processes which jointly contribute to the transformation of underlying petroleum resources into useable end-products valued by industrial and private customers. To address the global competition, some midstream operators, which link the upstream and downstream entities mostly, include resource transportation and storage to strive to deliver more quickly through transmission pipeline systems designed from better and cheaper platforms [20]. For complex systems such as transmission pipelines, the design is fundamentally an overwhelming task often involving multiple computation-intensive processes for both discrete and continuous design variables. Taking the design computation challenge with AutoCAD, a computer aided design (CAD) system as an example, over the years such CAD computing environments did map domain concepts to specific abstraction levels that concentrates and relates only to the computer aided design technologies but cannot express domain concepts appropriately. It is reported that it takes a stakeholder in the pipeline engineering design domain to always seek for the services of CAD systems for solving pipeline design problems, and for a favorable project, assuming an average computation time would be several days to months, which is unacceptable in real practice [14].

Despite continual advances in CAD computing power, the complexity of usage seems to increase. In recent years, the domain specific languages of the model had driven engineering based method for design representation in a language has attracted many attentions [6]. This approach represents physical model functions with simple domain concepts and attributes. With a simple model, and transformed into a meta-model, classic design, for example of a transmission pipeline that can be easily fabricated to effect smooth conveyance of products can then be effectively achieved. Such a method is therefore referred as meta-modeling the pipeline systems through the DSL definitions. Section 2 of this paper describes related work for variability in the domain specific modeling approach. In Section 3 the concepts of model driven architecture and domain specific modeling in the model driven engineering development spaces is described, followed by the domain specific modeling architecture for supporting the actual DSL specification definitions. Section 4 briefly describes the transmission pipeline domain with regards to tracing the domain concepts and the domain model in a DSL system. Section 5 describes the primitives' specifications, followed by discussion on metamodeling and case study in Section 6. Section 5 concludes the paper and describes future work.

A. Motivation and Open Problems

The strategy is utilizing domain specific modeling technologies in pipeline systems meta-modeling. In the domain of transmission pipeline engineering, there have been constant demands for more cost-effective and efficient tools and methodologies that could aid better, and provide faster and productive solutions to production of artifacts for pipelines systems. Though GPLs and common interactive CAD systems are effectively utilized for modeling, they do lack the necessary power to express the specifications of these models in a language (i.e., formal notations), which could enable the systematic representation of the various facets of the specific pipeline domain. In addition, they cannot express domain concepts appropriately, which means they are also characterized with the complexities of time consuming, syntax oriented and code centered development to achieving results in particular problem spaces [13].

The motivation therefore, is to define a DSL that could tackle the identified complexities of conventional software development tools. The DSL structure simply offers primitives whose semantics are familiar only to transmission pipeline mechanisms. With this well-defined DSL through domain specific modeling (DSM) approach, non-programmers and domain experts will be provided the resource to operate on very familiar notations and achieving great results without bothering on how the system is working, and without being burdened by its syntactic or semantic requirements [2].

II. Related Work

A pipeline system modeling language is simply a domain specific language whose pre-constructed notations and abstractions only offer expressive power to the pipeline engineering domain. As usual it has its own definition, which to some extent is presented in this work. However a few more formally defined schemes have been identified. Defining the domain directly as a language is [1].

Another exemplary language definition based on domain analysis is that of [2]. He translated a feature diagram to both grammars and propositional formulae. The semantics of the grammar is a set of iterative tree with string tokens, and thus repetition was possible. His definitions are close to ours but differs in the respect that there is no clear separation of decomposable features. Some research works have dealt with domain specific modeling languages. The Model-Driven Testing (MDT) work on automation of software testing emerged from the project of [10], which resulted in a domain specific modeling language (DSML) for Mobile Phone Applications Testing. They were interested in providing a platform, where test scenarios such as downloading an application, installing it, launching it, navigating in menus, validating user permission requests that are typically repeated on several devices can be performed by as a suite of actions by the tester or a non-programmer on one phone. The DSML uses models as instances of the language metamodel to express and execute tests [10]. One significant advantage we have over this method is their use of UML diagrams that traditionally restricts the user with its diagram definition standards. With DSM- DSL approach a potential user is not restricted but has the freedom to express their viewpoints clearly to achieve desired results.

There has been a surge of interest in applying model engineering and DSMLs to tool integration, with the benefits of model transformations [11]. The novel idea we are bringing to bear in this scenario is to greatly simplify the two issues of syntactic and semantic interoperability via tool utilization instead of defining each dynamics separately on a different framework before integrating to achieve desired model transformations. MetaEdit+'s implementation of the GOPPRR meta-modeling language provides useful metamodeling flexibility [8]. The heart of the environment is the MetaEngine, which handles all operations on the underlying conceptual data through a well-defined service protocol. The different tools request services of the engine in accessing and manipulating repository data [9]. The Graphical Editing Framework (GEF) provides technology to create rich graphical editors and views for the Eclipse Workbench UI [3]. GEF makes no restrictions on the underlying model; it can be an EMF model, Java code, etc. [7]. GEF follows the MVC (model-view-controller) concept, meaning that there is a separation between the model, its graphical representation (view) and the program logic (controller) [3].

III. Methodology

A. Model Driven Engineering (MDE)

Model Driven Engineering (MDE) technology is a suit of methodologies that support the development of domain specific languages (DSLs). There are [5] two approaches to MDE; the Model Driven Architecture (MDA) and Domain-Specific Modeling (DSM). The Domain-specific modeling approach [18] is characterized by a domain specific modeling language (DSML). The language formalism usually is about requirements within particular domains, such as oil and gas pipeline systems. All the models are defined in some language which defines the relationships among concepts in the domain and precisely specify the key semantics and constraints associated with these domain concepts [12]. MDE with DSML definition 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 [3]. Policies are specified at a higher level of abstraction using models and are separated from the mechanisms used to enforce the policies.

B. Domain-Specific Modeling (DSM)

Domain-specific modeling (DSM) is a new approach to model-based software development that defines and produces domain specific languages (DSLs). DSM is a top-down vertical approach that gives the developer the freedom to use structures and logic of a domain model that is specific to the target application domain, and thus, completely independent of programming language concepts and syntax. The application domain of consideration, for example, transmission pipeline systems can be represented in a domain model through metamodeling. The domain model usually represents the real world components, concepts and vocabulary relative to the core of the language definitions [6]. The necessity for a domain model is founded on the fact that not only is it the exact conceptual framework that showcases the semantics and the language workflow [9], but also contains domain classes and relationships as the basic defining components.

A well-defined domain-specific language provides abstraction mechanism to deal with complexity in a given domain; defining a DSL for transmission pipeline systems metamodeling is simply creating pipeline models from conception through a domain specific modeling language platform to produce an artifact. The definitions in our case, is to clearly identify and specify the concepts in the transmission pipelines domain as instances in the DSL such that the defining elements form the language metamodel with related domain notations [10]. In this way the design of a typical transmission pipeline tied to a particular conceived project system can achieved without the user facing any difficulty of how the policies are mapped onto the underlying mechanisms implementing them.

C. *DSM Architecture*

Illustrated in Fig. 1 is the DSM architecture for defining the DSL; whereas the left side describes the actual specifications, the right side describes use of the domain model [7]. The language is formalized into a metamodel and all models describing applications or features are instantiated from this metamodel. Thus models can’t express anything else other than what the language allows. This language instantiation ensures that developers for a typical transmission pipeline follow the concepts and rules of the domain model [9]. This flexibility is making sure the domain framework is not visible to developers, in a similar manner as basic input and output system (BIOS) code or primitives called by the running application are not visible to programmers in general purpose programming languages (GPL).

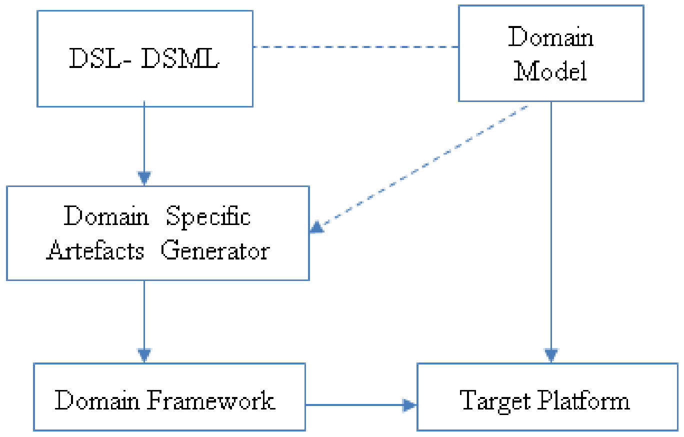


Fig. 1. DSM Architecture (source: Steven and Juha, 2008).

Another progression is in the aspect of clear definitions and use of the modeling language. In the simplest cases, each modeling symbol generates certain fixed artifact, including the values entered into the symbol as arguments. The domain framework also provides the interface between the generated artifact and the underlying target platform. It can directly call the platform components, whose existing services are enough to make the artifacts simpler. This domain framework can range in size from components down to libraries, which provide predefined building blocks [11]. In DSM, all the possible layers are hidden and not visible to developers yet the use of the domain model elements are made automatic.

IV. Defining the Primitives

Significantly, the system definitions involve the specification and evaluation of solutions to the specific problems of modeling transmission pipelines. The key issues here are the applicable steps of the interacting components realized from domain analysis. Conventionally [15], the DSL definition steps include defining the domain, designing the language that accurately captures the domain semantics, and describing the configuration rules of the features of the pipeline physical components within the domain model.

A. *Domain Definition*

The domain definition is a framework that describes the requirements engineering products resulting from the domain analysis. Shown in Fig. 2 are the key elements in the framework for the development of the new system. The rest of this section is dedicated to the explanation of the elements of the framework. As depicted in the framework, domain knowledge involving oil and gas pipelines, which invariably means transmission pipelines will come first. The next will be the description of concepts and the domain model specifics and so on. The concept description, which is part of the analysis, follows a precise path that moves into the formation of the model instances for the language design [16].

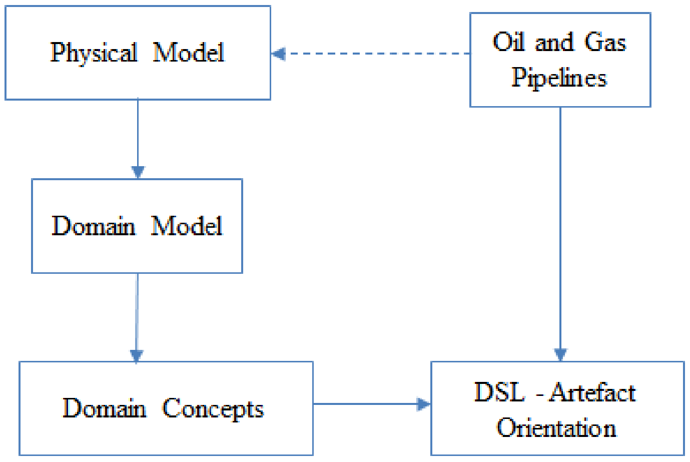


Fig. 2. Domain analysis framework.

B. *Oil and Gas Transmission Pipeline*

Oil and gas transmission pipelines here refer to domain knowledge; the specific information needed from stakeholders in the oil and gas pipeline engineering domain [19]. In order to achieve this very important step in the definition of a DSL, some domain knowledge about pipeline systems were gathered during domain analysis. Knowledge was provided by crew engineers from oil and gas pipelines servicing industries. Transmission pipelines are specifically designed to transport petroleum products along distances. The transmission pipeline as shown in Fig. 3 collects the specific petroleum products from any quality assured source along the pipeline and delivers the product to end users [5]. Transmission pipelines can convey unrefined crude oil from producing areas to large storage areas or directly to refineries, it can deliver water for town water supply, and could be for natural gas only or carry a number of processed or refined petroleum products such as gasoline, diesel, refined fuel oils.

Fig. 3. Typical transmission pipeline.

Most transmission pipelines are designed to the American Society of Mechanical Engineers or standards based on these. The design and operation of pipelines is usually regulated or subject to local laws, which detail design, construction, operation and maintenance requirements for pipelines. The pipelines are made by welding together lengths of steel pipe, typically constructed to meet the specific needs established by the marketplace.

The major components used to construct these lines include pipe, fittings, joints, flanges, gaskets, coatings, valves, compressors, drivers, meters, liquid management equipment, actuators, cathodic protection equipment, control equipment, and ancillary systems to provide compressed air. Compressors, drivers, and meters have already been discussed so this section will concentrate on the other components. The design and build process of transmission pipelines [20], involves determining the origin and destination of the pipeline, the approximate length of the pipeline, the product to be transported, diameter and type of pipe used, hydraulic factors such as type of flows expected in a pipeline, approximate capital cost and running expenses. It also involves route selection being requirements for right of way acquisition, testing of soils and data collection, and analysis and design of hydraulic and job scheme.

C. Description of Concepts

Engineers responsible for the preparation of design documents must, from time to time, review the current codes and standards in order to comply with and take advantage of the changes in the industry which are expected to continue as computerized drafting and isometric or orthographic pipeline sketches are made, as determined by project requirements. Structural, and control information are often included in these sketches, which form the basis for the working physical drawings. The sketches and composites are now transformed into the computer versions of the physical models in a computer-aided design (CAD) system. With the development of three-dimensional computer-aided design (3DCAD) software, the engineer can check for interference and can generate different views [8].

Once the orthographic drawings are completed, they may be issued for piping fabrication and construction. However, for complex pipeline systems, it is common practice to develop separate piping isometric drawings for each pipeline []. For pipe stress analysis, fabrication, and installation, the piping isometric drawings are easier to use than the orthographic drawings because all the information on the isometric drawing pertains to the pipeline of interest without cluttering with extraneous information [18]. It was observed however that a bottleneck [12], in their modeling operations is the inability of the conventional CAD 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. The systems provided poor facilities for keeping track of design rules from the stakeholder’s viewpoints. For example, RapidVu could only create a solution platform for maintenance needs, and then programming expertise is required all the time to leverage Solid Works with Excel whenever an interface and some calculation routes are needed in their schedules [9]. Now a carefully defined DSL with knowledge of pipeline physical components representations and design parameters can make the engineer achieve optimal performance in bringing a typical transmission pipeline design on an editor interface with relevant concrete syntax representations better than struggling all the time trying to understand isometric and other CAD mechanics to achieve same. The conceptual DSL definition brings to bear domain abstractions, and expressive power restricted to, the domain of transmission pipelines for the related products in the petroleum industry and in water supply. Consequently this can be achieved only by taking advantage of specific properties of the pipeline engineering application domain that pertain to transmission [10]. The description of these specific properties therefore represents the domain concepts, which will be useful in creating the abstractions and in the semantic mappings of the elements of the DSL modeling platform.

D. Domain Model

The semantic gap created due to inability of domain experts to manipulate artefacts orientation by using GPLs in their work place is closed by mapping the domain concepts to abstractions in the form of attributes of the CAD physical models representing the pipeline physical components [20]. The domain model is the repository for these concepts (i.e. its vocabulary) and their relations. In the domain model is the semantic model subset consisting of the classes of the events and their relationships with a focus on the user’s perspectives. An example of a typical event pertaining to user’s perspective is thus given:

```
event
name:elbowJoint
code:PipeBuild
```

```
end
state:active
elbowJoint ➡WaitingForParameters
end
state:join.this.elbow
translate ➡target
target:name ➡join.this.elbow
trigger:elbowjoint
end
```

As knowledge changes, the semantic model itself can change so as to ensure physical components continue to do what the users want them do at the editor interface of the DSL and then produce clear design specifications for pipeline physical assets such as pipes, valves, active equipment (pumps, compressors, etc.), insulation and supports [6].

V. Specifying the Primitives

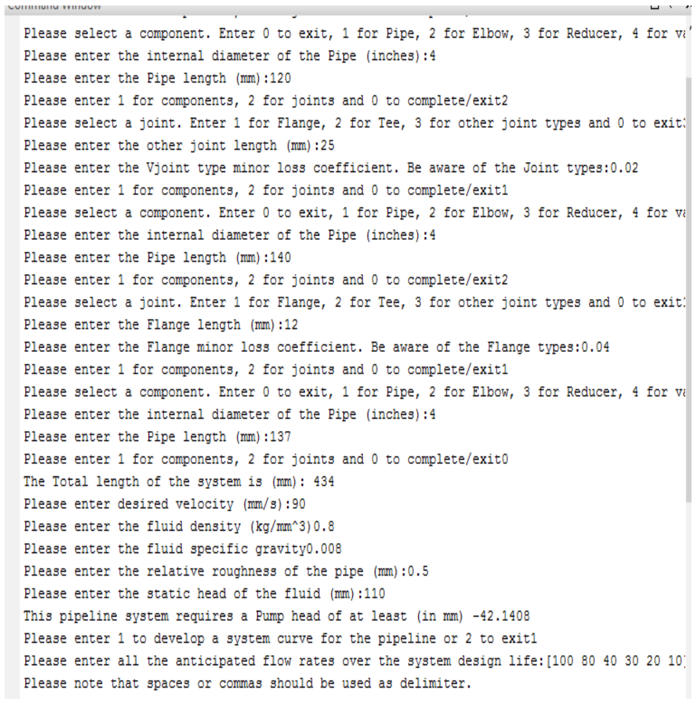
The modeling primitives in the language internal logic are specified to ensure the exertion of the linguistic power to manipulate input parameters from domain experts and as well display appropriate pipeline configurations [15]. It also displays the modeling language internal mechanics that reflects the abstractions; incorporating domain concepts and associated production and semantic communication rules. The system engine contains the interactive configurations implying possible assignment of features given the current state of the system, and propagating information whenever new choices are made. The Pipeline designated (r) is the parent root feature with mandatory features; Components (c), Fittings (f), Joint (j), and Support (s), and Optional pipeline bed location feture (b). These are the standard references that define the pipeline components attributes and relationships [2]. The standard reference definition can actually be an associated tree grammar with mandatory feature having dimension (d), point of intersection (p), and type of component (t) as child features. Syntactically directed; the standard reference definitions are made up of the context-free grammar with attributes and rules to calculate the attributes. The syntactic elements in the CFG are specified as input with productions specifying the symbol substitutions for the major objects in the pipeline model that can be recursively performed to generate new modeling sequences [19]. With each grammar symbol, a set of attributes are associated, and with each production, a set of semantic rules are defined for mapping values of the attributes corresponding to a typical artefacts of the pipeline model as follows:

Production	***Semantic Rules***
$r ::= c\|f\|j\|[\![b]\!]s$	$r.model = c.t.d.p\|f.t.d.p\|j.t.d.p\|s.t.d.$
$c ::= ct\|cd\|cp$	$c.val = ct.val\|cd.val\|cp.val$
$f ::= ft\|fd\|fp$	$f.val = ft.val\|fd.val\|fp.val$
$j ::= jt\|jd\|jp$	$j.val = jt.val\|jd.val\|jp.val$
$s ::= st\|sd\|sp$	$s.val = st.val\|sd.val\|sp.val$

Following the necessary structural framework that must be put in place for the language to implement its core operations, fragments of the syntactic elements of the grammar in BNF notation for defining the various pipeline build metrics of the language is created. The grammar is the pipeline components grammar; it is a collection of the modeling primitives and the rules connecting them as the syntactic elements [16]. The entire structure is a collection of pipeline components context free grammar split into varieties of lexemes corresponding to each token in the statements for processing as follows:

$$Program \coloneqq (M)$$
$$M \in Program$$
$$Id \in Identifier$$
$$E \in Expression$$
$$Cm \in Command$$
$$St \in Statement$$
$$M ::= Id, E, Cm, St$$
$$M \coloneqq R$$
$$R ::= Rt, B, C$$
$$Id \in R = \{Cdomain \wedge Rcodomain\}$$
$$Id = \{\{\{P,F,J,S,I,T\},\{p\},\{r,t,f\},\{b,g,e\},\{h,c,k,d\},\{u,v,m,n,a\}\}\wedge\{d,fn,tn\}\}$$
$$Cm ::= Id \coloneqq E \mid \text{if E then St} \mid \text{while E do St} \mid \text{St}_1;\text{St}_2 \ldots.\text{Stn} \mid \text{end}$$

The domain specific modeling methodology for creating domain specific language is presented with a focus mainly on modeling transmission pipeline systems [17]. This approach is flexible comparable to Computer Aided Software Engineering (CASE) and GPL tools (Steven, 2007). One notable difference is in the aspect of clear specifications formalized into a metamodel in the form of a collection of modeling primitives and the rules connecting them; there is informal domain description, recursively defined for pre-processing by the system functions.



```
Please select a component. Enter 0 to exit, 1 for Pipe, 2 for Elbow, 3 for Reducer, 4 for va
Please enter the internal diameter of the Pipe (inches):4
Please enter the Pipe length (mm):120
Please enter 1 for components, 2 for joints and 0 to complete/exit2
Please select a joint. Enter 1 for Flange, 2 for Tee, 3 for other joint types and 0 to exit
Please enter the other joint length (mm):25
Please enter the Vjoint type minor loss coefficient. Be aware of the Joint types:0.02
Please enter 1 for components, 2 for joints and 0 to complete/exit1
Please select a component. Enter 0 to exit, 1 for Pipe, 2 for Elbow, 3 for Reducer, 4 for va
Please enter the internal diameter of the Pipe (inches):4
Please enter the Pipe length (mm):140
Please enter 1 for components, 2 for joints and 0 to complete/exit2
Please select a joint. Enter 1 for Flange, 2 for Tee, 3 for other joint types and 0 to exit
Please enter the Flange length (mm):12
Please enter the Flange minor loss coefficient. Be aware of the Flange types:0.04
Please enter 1 for components, 2 for joints and 0 to complete/exit1
Please select a component. Enter 0 to exit, 1 for Pipe, 2 for Elbow, 3 for Reducer, 4 for va
Please enter the internal diameter of the Pipe (inches):4
Please enter the Pipe length (mm):137
Please enter 1 for components, 2 for joints and 0 to complete/exit0
The Total length of the system is (mm): 434
Please enter desired velocity (mm/s):90
Please enter the fluid density (kg/mm^3)0.8
Please enter the fluid specific gravity0.008
Please enter the relative roughness of the pipe (mm):0.5
Please enter the static head of the fluid (mm):110
This pipeline system requires a Pump head of at least (in mm) -42.1408
Please enter 1 to develop a system curve for the pipeline or 2 to exit1
Please enter all the anticipated flow rates over the system design life:[100 80 40 30 20 10
Please note that spaces or commas should be used as delimiter.
```

Fig. 4. Modeling action using DSL definition.

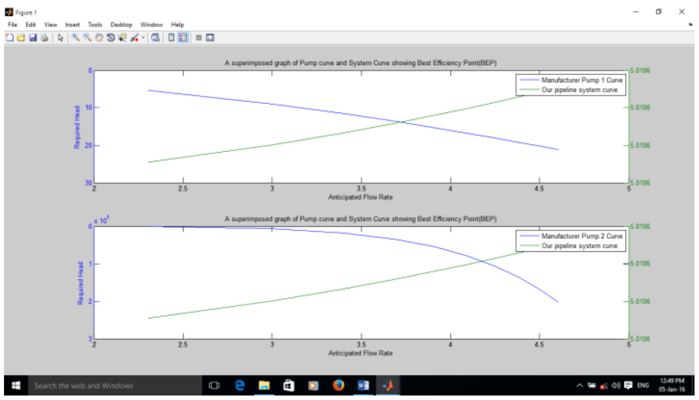


Fig. 5. Result of a design scenario.

A. *Design Scenario and Case Study*

In order to start building the different design criteria or routes that depicts stakeholders view points, the input values and attributes of the pipeline components has to be selected. Depending on the particular design operation, a stakeholder or a domain expert (pipeline engineer) can simply follow simple prompts by selecting any desired scenario such as pipe→joint→pipe→fitting→pipe→instrument→pipe, etc. to come up with a system curve that models and describes the fundamental requirements of the developed pipeline system for onward physical interpretation and fabrications respectively [5]. Fig. 4 is an example of a modeling action using the editor of a DSL definition, and a subsequent system curve depicting model selection for a particular pipeline project as presented in Fig. 5.

VI. Conclusion and Future Work

The DSM approach is adopted in this research to create a domain specific platform whose type systems and semantics simplify modeling processes in the design and implementation of transmission pipeline configurations. The domain specific representation is predicated on transmission pipeline graphics model as the entity during development, it is the model that reflects the prescriptive technical characteristics prevalent in the transmission pipeline engineering domain. It also represents the concepts of the domain within which the language formalism is created to control the flow of processes without including extra or unnecessary properties captured in the design analysis. A typical modeling activity with this system takes away complexities related with conventional modeling systems where the engineer has to rely on most of the times, but with this system, the engineer only need to follow simple instructions and achieve design intents. More activity and build process is possible in the future because computing science is yet applied to solving an engineering problem concerning pipeline design for fluid transmission operations. With given set of values in a system curve, the engineer is equipped with relevant information about the intended pipeline properties, and can now go ahead for acquisition of the physical components to start a pipeline build project.

Acknowledgment

First and foremost, my gratitude goes out to my research fellows in the field of domain specific modeling and requirements engineering. This work was part sponsored by the Nigerian Niger Delta Development Commission, and then fully supported by the Edo University Iyamho Edo State Nigeria through its provisions for research frontiers.

References

[1] Zhao, W., Bryant, B. R., Cao, F., Rajeev, R., Raje, M., and Auguston, C.C. (2004), Grammatically interpreting feature compositions, in: Proceedings of the 16th International Conference on Software Engineering and Knowledge Engineering (SEKE'04), Banff, Canada, 185-191.

[2] 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.

[3] Gustavo, C. M., Sousa, F. M., Costa, G., Peter, J., and Clarke, A. A. (2012), Model-Driven Development of DSML Execution Engines, *Proceedings of ACM Conference*, eduMRT '12, Innsbruck, Austria, 112 -118.

[4] A.Agrawal et al. "MILAN: A Model Based Integrated Simulation Framework for Design of Embedded Systems". In: Proceedings of LCTES, 2001.

[5] L.Bondé, C.Dumoulin, J.-L.Dekeyser. "Metamodels and MDA Transformations for Embedded Systems". In: Proceedings of the Forum on Design Languages (FDL), Lille, France, September 2004.

[6] EMF. Eclipse Modeling Framework. Available at: http://www.eclipse.org/emf. Accessed in May, 20010.

[7] Baar, T. (2006), Correctly defined concrete syntax for visual modeling languages, in: Proceedings of the 9th International Conference on Model Driven Engineering Languages and Systems (MoDELS'06), 111-125.

[8] Kelly, S., and Tolvanen, J. P. (2008), Domain-Specific Modeling. Wiley-IEEE Computer Society Press, NY, 2008.

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

[10] Youssef, R. (2010), A DSML for Mobile Phone Applications Testing, University of Pau Avenue de l'université 64013 Pau, France.

[11] Zekai, D., and Marjan, M. (2009), Verification of DSMLs Using Graph Transformation: A Case Study with Alloy, MoDeVVa'09, 74-82.

[12] M.F.S.Oliveira, E.W.Briao, F.A.Nascimento, F.R.Wagner. "Model Driven Engineering for MPSoC Design Space Exploration". Journal of Integrated Circuits and Systems,v. 3, n.1, 2008.

[13] OMG. UML Profile for Modeling and Analysis of Real-time and Embedded Systems (MARTE). 2007, available at .

[14] Kelly S, Tolvanen JP (2008) Domain-Specific Modeling: Enabling Full Code Generation, Wiley-IEEE Computer Society Pr.

[15] Gronback RC (2009) Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit, Addison- Wesley Professional, Upper Saddle River, NY.

[16] B.G. Technical LTD (2013), B.G. Technical Oil & Gas industry Port Harcourt, Nigeria; www.bgtechnical.com/ Annual Reports 2009 to 2013

[17] Zezula, F., and Durden, C. (2000), Piping Joints Handbook, Piping & Pressure Systems Consultant, UG, Sunbury, 2000

[18] Cook S, Jones G, Kent S, Wills AC (2007) Domain- Specific Development with Visual Studio DSL Tools, Addison-Wesley Professional, Upper Saddle River, NY.

[19] Mezhuyev V (2015) Metamodelling architecture for modelling domains with different mathematical structure. In: Advanced Computer and Communication Engineering Technology: Proceedings of the 1st International Conference on Communication and Computer Engineering, Lecture Notes in Electrical Engineering vol 315, Springer, pp 1049–56.

[20] Mezhuyev V, Sputh B, Verhulst E (2010) Interacting entities modelling methodology for robust systems design. In: Proceedings of 2010 Second International Conference on Advances in System Testing and Validation Lifecycle, pp 75–80.