



A Framework for Requirements Engineering for Oil and Gas Pipeline Systems Modeling

Japheth Bunakiye Richard¹, Asagba Oghenekaro Prince²

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

²Department of Computer Science, Faculty of Physical Science and Information Technology, University of Port Harcourt, Port Harcourt, Nigeria

Email address:

rb.japheth@ndu.edu.ng (J. B. Richard), asagba.prince@uniport.edu.ng (A. O. Prince)

To cite this article:

Japheth Bunakiye Richard, Asagba Oghenekaro Prince. A Framework for Requirements Engineering for Oil and Gas Pipeline Systems Modeling. *American Journal of Software Engineering and Applications*. Vol. 4, No. 6, 2015, pp. 99-106. doi: 10.11648/j.ajsea.20150406.11

Abstract: Domain specific modeling services, especially when made available to pipeline systems that transport oil and gas, constitute an interesting but very challenging domain. It poses fundamental problems for requirements engineering, software architecture, and their relationship. We propose a novel, domain-based framework for requirements engineering for this class of applications. The framework addresses the key concepts in this field, such as changing complexities for design platforms and domain specific requirements. We report experimental lessons learned on this framework and suggest requirements analysis products for documentation and future system design directions.

Keywords: Requirements Specification, Oil and Gas Pipeline, Domain Analysis, CAD Models, Stakeholders Intents

1. Introduction

Models are human concepts that explain systems in the real world, they are simply outcomes from abstractions, and one good example is the use of computer graphics to model a system. A typical oil and gas pipeline system is one good example of such computer graphics created from the interactive aggregation of graphics primitives, graphics assemblies and subassemblies of CAD systems [17]. These graphics models then represent the pipeline design from the domain of pipeline engineering. The transformation (modeling) of the model which is now domain specific to a complete artifact is made possible because of a constrained language definition that could address specific needs of the problem space of the domain. Such a possibility is usually referred to as domain specific modeling (DSM) [7]. The domain specific modelling approach is model centered and requires the construction of a domain specific modeling language that can provide a means for expressing domain concepts in a model. This domain model is capable of being transformed into other formats that represents all aspects of a systems understanding in the domain [8]. Modeling goals arise when stakeholders establish interests along lines of disciplines, intents, and design bases such as physical attributes, and environmental and materials-related factors.

There are competing design requirements among stakeholders; each one has their own set of constraints, objectives and responsibilities resulting to varying designs. These determining facts have to be gathered and accommodated in a language formalism to come up with an optimal solution. These goals can be achieved by capturing critical aspects of the domain, the expressions of stakeholders design intent and the requirements analysis; reflecting the domain characteristics and then integrated as concepts to the policies and mechanics of software architecture. This paper is about modeling requirements, and is targeted at oil and gas pipeline engineering domain analysis and requirements for a cost effective and timely modeling by domain experts to achieve desired products in their line of business. It is expected that the contribution of these requirements analysis framework will result in a new layer of reusable software capable of processing these graphics models to produce desired artefacts [5]. A collection that supports further layers of accessible software connected through interfaces for productivity centred configurations. The requirements of course will make sense only if the design goals can be met. This means in general that computer aided design systems have a complex representation of domain concepts and as such are not applicable to any specific domain [15]. For all these reasons, we argue that a new approach is needed to tackle this kind of complexities

associated with these indispensable tools stakeholders must use. Such an approach is the subject of this paper, and will be described as follows. Section 2 will provide the reader with some background information. Section 3 explains the requirements engineering framework that will be used throughout the work. Section 4 outlines the specifications, while Sect. 5 sets out some of the key experimental lessons learned as well as requirements documentation for the design issues.

2. The Approach

The purpose of this section is to highlight the influence associated with requirements engineering in the area of modelling engineering systems. In order to properly define concepts in the domain of oil and gas pipeline engineering, we will adopt Jackson's terminology in his work on the "world and machine [4]. The requirement, which focus on the problem, is in the world; the machine is the solution we construct. In his paper Jackson [4] identifies four facets of the relationship between the world and the machine that represents a foundation in understanding the relationships between a software artefact and the surrounding world. The modelling facet is concerned with a model of some aspect of the world within the machine. The interface facet is concerned with interaction between the machine and the world. The engineering facet is concerned with the control the machine applies on the world, which allows us to distinguish between requirements, the environment, and specifications. Finally, the problem facet is concerned with the relationships between the structure of the world and the machine, where the nature of the problem and the shape of the world influence the shape of the machine and of the solution. The discussion of all the facets is useful to us; with a pipeline context model, the operational standards, environment and concepts of the pipeline domain can be modelled within the machine. The engineering facet concerns us particularly on the distinction between requirements, which identify the characteristics of processes in the world of oil and gas pipeline engineering, the environment surrounded with the processes, which implement the required support needed for the models, and specifications, which identify the collective occurrences between the requirements and the programs. This obvious distinction is hardly made in research on domain specific modelling software processes [16]. We have therefore clearly exemplified where requirements are concerned solely with the world of the domain; in this case with a model representing domain concepts, programs are concerned solely with the machine for design execution, specifications are bridging the requirements and the design. We will use these ideas in working out the boundaries within our framework.

3. Related Work

This section looks at related work on how requirements are defined and represented. It is common to express

requirements by describing the component elements in a framework. Anthony and Andrea [1] defined and presented a framework for requirements engineering for context-aware services. In their presentation, they described requirements for context-aware services, when made available to mobile devices. Requirements for the domain of mobile services, though an interesting study but constitutes lots of challenges. They therefore came up with a novel, reflection-based framework for requirements engineering for this class of applications that addresses the key difficulties such as changing context and changing requirements. We are more concerned with requirements for tackling platform complexities associated with conventional modelling systems. Comparing requirements analysis methods for developing reusable component libraries Alistair et al. [5] advocated the use of domain theory as a means of domain analysis for creating a reusable library of software components for constructing telemedicine applications. Reported in their work is the experience of applying each approach as a domain analysis method to specify abstract components that produced detailed specifications at different levels of abstraction. Instead of a comparison of approaches and analysing domains at different levels of abstraction, we are analysing domain as an integral part in the requirements engineering process to come up with abstractions that can meet the requirements goal. The World and the Machine by Jackson [4] clearly identifies the usefulness of the machines (i.e. the software solutions) developers make and the world, in which the machine serves a useful purpose. But there has to be a balance between the competing demands and attractions of these two concerns, and this balance can only be possible with requirements to solving a particular problem, which is in the world. We have clearly adopted this methodology in working out the boundaries within our framework. In his work on domain understanding the key to successful system development Ray [11] posited the need for developers to always look at the very significant advantages offered by developing a prior and adequate understanding of the surrounding application domain. He further argued that most practising engineers, be they pipeline, aeronautical, electrical, or maritime etc., would accept without argument that a clear, concise and unambiguous understanding of a development project's specific application domain is an essential precursor to successful systems development. This suggestion is quite useful to us because our requirements engineering framework is a necessary first step into eliciting knowledge about pipeline engineering and how a modelling system can be developed to ease technical work for the stakeholders in that domain. Charles et al. [12] presents a framework for security requirements elicitation and analysis, based upon the construction of a context for the system and satisfaction arguments for the security of the system. They started with enumeration of security goals based on assets in the system. They described the system context using a problem-centred notation, and then a satisfaction argument was constructed to validate the context against the security requirements. The satisfaction argument, which is in the form

of a formal argument now, verifies if the system can meet its security requirements.

4. The Framework

The requirements framework described in figure 1

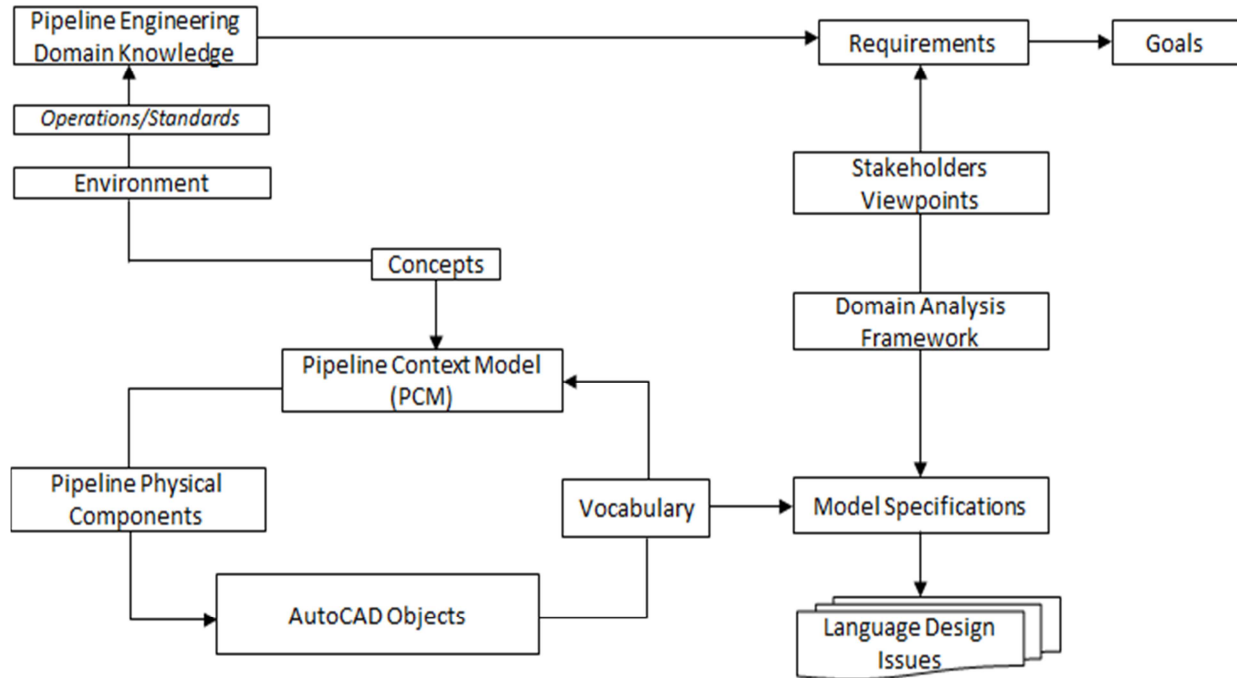


Figure 1. Requirements Engineering Framework.

The rest of this section is dedicated to the explanation of the elements of the framework [7]. As depicted in the framework, we will start with the goals and domain expert operational standards and environment, which are available in the world. The next will be the description of concepts, i.e. the salient technical characteristics prevalent in the domain of oil and gas pipeline engineering, all of which still belongs in the world. This concept description will follow a precise path that facilitates an understanding of the elicitation and formation of the pipeline context model (PCM) from AutoCAD objects and other domain knowledge. Integrating the concepts with the specifications, Will invariably be the bridge between the world and the machine for a modelling system [17].

4.1. Goal

The goal of modeling oil and gas pipeline systems is to provide domain engineers with hands on modeling tool with which they can build pipeline designs and related systems. It is a long-term objective achieved through cooperation of elements in the proposed software and in the environment. Essential aspects in the construction and coordination of the modeling software mechanism is [20] the application of domain specific modelling methodology (DSM) on the development of a domain specific modeling language (DSML) that can transform stakeholders input through

addresses the problems of a lack of distinction between requirements specifications and programs in software processes targeted at modelling engineering designs, and a general lack of a clear pathway for deriving requirements for modelling oil and gas pipeline systems from business goals in the pipeline engineering domain [19].

guided notations of the pipeline systems to desired artifacts. In our interpretation the goals do not change with the changing development context, and must represent the ultimate objective the modeling software system is meant to achieve through the cooperation of all the elements in the requirements engineering framework [15].

4.2. Domain Knowledge

Domain knowledge here refers to the specific information needed from stakeholders in the oil and gas pipeline engineering domain. Inputs regarding pipeline components physical attributes and design criteria are sequenced through the acquisition of qualitative information from pipeline engineers in their fields of operation. This was necessary in order to set the boundaries of the new system. The environment was taken into account because of its strong influence on the elicitation of the domain knowledge [19]. Environment in this context means the scope and operational standards of pipeline engineering basic work flow. A few of which entails Isometric or orthographic pipeline drawings and system descriptions based on functions, intents and major features such as design foundations, operating modes, performance ratings, and control concepts. Feature oriented domain analysis (FODA) of the domain detailed the common and variable parts of the embodiment of the entire relationships and associations of the features and attributes of

the pipeline context model in the problem space [18]. The problem space in its generic form refers to established feature constraints and their composition rules that guide the process of transforming the user requirements into a modelling

language that satisfies the requirement goals. Figure 2 is the domain analysis subsystem in the requirements engineering framework that determine the degree of correctness between the realization steps of the pipeline model.

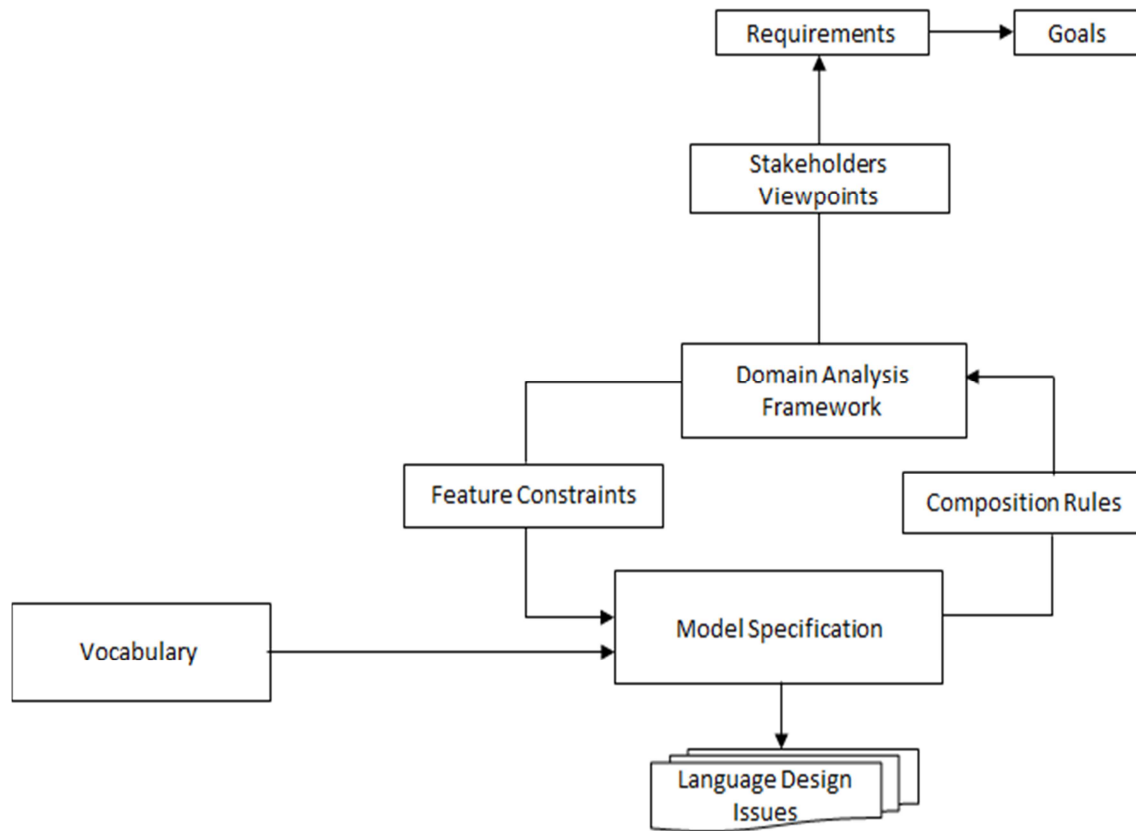


Figure 2. Domain Analysis Subsystem.

4.3. Description of Concepts

A typical oil and gas pipeline systems modeling formalism is expected to offer, through appropriate notations and abstractions, expressive power restricted to, the oil and gas pipeline design problem domain. Consequently this can be achieved only by taking advantage of specific properties of the pipeline engineering application domain that pertain to transmission [19]. The description of these specific properties therefore represents the domain concepts, which will be useful in creating the familiar notations and in the semantic mappings of the elements of the modeling platform. The design of a pipeline system requires the application of theory from a number of engineering principles [16]. It also requires the knowledge and application of a number of codes and standards such as physical attributes, loading and service conditions and environmental and materials factors that relate to pipeline design. Physical attributes are those parameters that govern the size, layout, and dimensional limits or proportions of the pipeline [2]. Dimensional standards have been established for most pipeline components such as fittings, flanges, and valves, as well as for the diameter and wall thickness of standard manufactured pipes. Physical piping

components in a pipeline project includes pipe, flanges, fittings and joints, bolting, gaskets, valves, and the pressure containing portions of other piping components. It also includes pipe hangers and supports and other items necessary to prevent over pressurization and overstressing of the pressure-containing components [19]. It is evident that pipe is one element or a part of piping. Therefore, pipe sections when joined with fittings, valves, and other mechanical equipment and properly supported by hangers and supports results into a pipeline system. A Pipe is a tube with round cross section conforming to the dimensional requirements of the American Society of Mechanical Engineering (ASME) B36.10M Welded and Seamless Wrought Steel Pipe and ASME B36.19M Stainless Steel Pipe [19]. Hangers and supports include elements which transfer the load from the pipe or structural attachment to the supporting structure or equipment. Pipeline valves are dedicated for controlling the flow of the fluid transported along the pipeline. Joint is a connection between two lengths of pipe or between a length of pipe and a fitting. There are quite a lot of types of joints in a pipeline system, each dependent on the service orientation in the pipeline project. Pressure containing components such as gauges and meters carries sufficient pressure metrics for the

inputted fluid /pipe surface relationships.

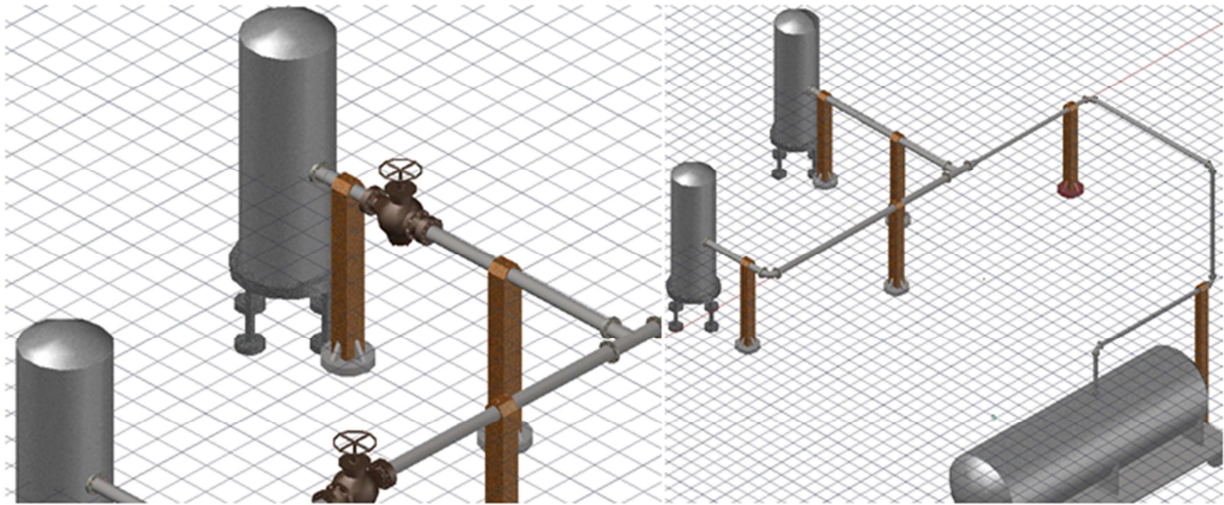


Figure 3. A Typical Pipeline Model with Fittings.

Figure 3 is a typical pipeline model with fittings that can convey oil or gas from oil wells and storage facilities to tank farms for storage and to points of utilization or to refineries for processing. A pipeline context model (PCM) can be achieved with AutoCAD or any CAD system by careful selection of appropriate fittings as specialized pieces that connects other pieces together [17]. The description of the concepts i.e. the pipeline physical components and necessary design parameters are often represented as AutoCAD objects. Two basic reasons for this are:

1. to make sure all the objects are collectively seen to be the model for the modelling system definition.
2. to make sure the vocabulary necessary for the subsequent modelling platform specifications are captured

4.4. Vocabulary of Components

Domain specific modeling (DSM) approach to language development involves the logical use of models as core entities throughout development. This usually results into the language metamodel where the model is specified in some formal notation. In our view this model at the core of development is the pipeline context model (PCM). Consequently the model offers insight into the needs that when met, oil and gas pipeline systems modeling is made possible productively through a modeling language platform [3]. It is again evident that models are simply outcomes from levels of abstractions, so developing a modelling language is an effort in creating a platform where the domain concepts can be mapped to the abstraction levels. Once this mapping is achieved, it will create a stakeholder or domain expert user friendly interface with notations from the concepts description; it will also hide all the syntax and semantic complexities associated with programming and CAD platforms [6]. The vocabulary of components captured as origin, termination, length, diameter_inner, diameter_outer, material type, points of intersection, slope, colour etc. serves

a useful purpose of mapping these concepts to the abstractions in the form of attributes of the AutoCAD objects representing the pipeline physical components as described in the pipeline context model (PCM). Shown in Figure 4 is a technical diagram explaining the typical real life pipeline fitted with the components. It is an example of the pipeline context model where all the necessary attributes such as size, shape, direction of loops, layout, and dimensional limits or proportions of the pipeline are captured and represented as the vocabulary of components.

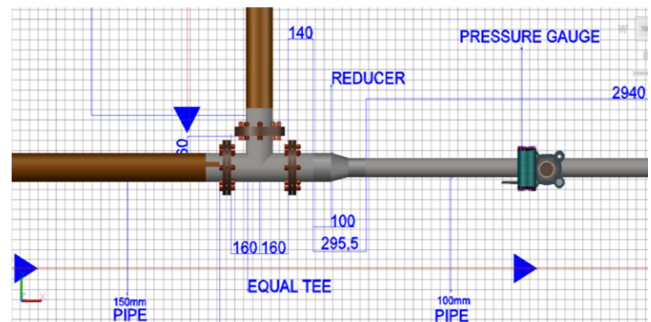


Figure 4. Technical Diagram Explaining Vocabulary Capture.

5. The Specifications

The modeling language specifications and design solution can be constructed in the sense of pipeline engineering product families of a set of basic items, in which the characteristically distinctive aspects of the products that are visible to various stakeholders are represented as features in the software framework [12].

The software framework or platform, embeds all feature variability and commonalities, and provides services common to all applications of the product family. While the variability indicates precisely what evidence is needed to specify an instance of a feature that could fit into a particular modeling scenario; the commonalities in this respect define the set of common operations and primitives of the language

[13]. Features such as fittings, joint and support, whose parent is the pipeline root concept, are common features. Then component dimensions and types, fitting dimensions and types, piping support features, points of joints and joint types and dimensions are the variabilities [10]. The piping components can be atomic or compound, access to the components as a service can be triggered by a design and modelling intent of a typical stakeholder; the variability and commonalities are considered as first step in designing the modelling language, and are therefore both built into the execution mode. The feature model of the composite forms enable transformation and for interactive configurations; interactive configuration implies possible assignment of features given the current state of the system, and propagating information whenever new choices are made [9]. These internal interactions adhere to the composition rules about configurability of the features that corresponds to the language definition.

6. Lessons Learned

The syntax and semantic complexities associated with programming and CAD platforms examined in the previous section represent a formidable challenge for any domain engineer [17]. One of the key issues in these systems is that stakeholders have competing design requirements. Therefore, requirements and system behaviour must adapt themselves to accommodate these determining factors. In order for this to be feasible, the vocabulary of CAD objects representing the domain specific oil and gas pipeline components must be specified in a domain specific language formalism for easy and timely artefact orientation of the pipeline systems configurations [14]. This representation must take place in a way that is both readily understandable by stakeholders and easily manipulatable by the machine, which means a suitable software architecture design phase should always precede the actual implementation. In other words, as far as experts could see through to a design scenario, the system will be able to capture it and evolve a design that meets their needs. The user could make some input through guided notations from the interface, and the system can then match these inputs with a parsing grammar specification to produce desired designs. Internal communication among the architecture components is enforced and made possible by utilizing the running system that implements a specification. This architecture, invariably defines the scope and environment of the new requirements. But some very important questions such as the ones listed below may arise [1].

1. What is the trade-off of switching from a CAD system to using the proposed language?
2. Could the benefits of expressing every detail of the pipeline overcome the simplicity that normally is associated to a CAD system in terms of usability for the user?
3. What clear benefit can be gained using the approach over existing approaches?
4. Why is a DSML needed at the first place?

The trade-off of switching to using this language is reduction in design time and cost of operations. For example, some trained CAD software professionals whose job is to oversee and attend promptly to varying engineering designs and fabrication considerations [20]. The simplicity in usability of this language is clearly exemplified in the vocabulary of components. Having captured the vocabulary and then translated into a modelling platform, the users need not any engineering design expertise to get their design intents achieved. In optimal cases, programming expertise must be needed to achieve some design intents [18]. The clear benefit is that in this language, the metamodel is a repository of concepts from the oil and gas pipeline domain. The models are constructed using these concepts and not the concepts of a given programming language. Therefore there is no semantic gap between design intent and the expression of this intent in several lines of codes [20]. The users have the freedom of expressing their intents through familiar domain representations; without facing any difficulties of how the policies are mapped onto the underlying mechanisms implementing them.

The obvious is that all the services in the framework cannot complete implementation cycle on its own. However, our current thought is that it should be possible to characterize stakeholders design intent as the view points of the input parameters; so that a complete requirements analysis products and system workflow can be documented. The term design intent refers to identified interests and disciplines of all the stakeholders of this system as it relates to pipeline design and highlights physical attributes that must be considered in completing a modelling process [15]. Figure 5 illustrates an overall view of the requirements analysis products for the requirements documentation and system workflow.

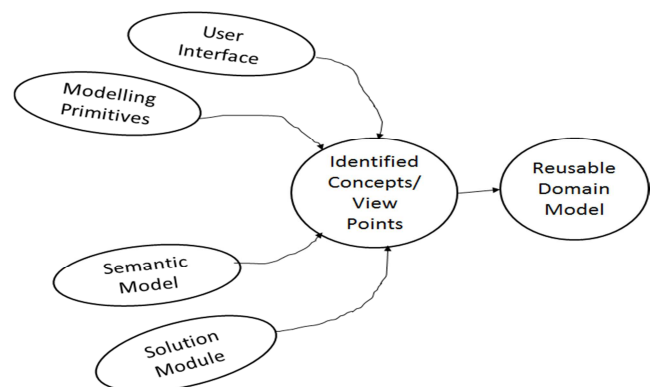


Figure 5. Requirements Analysis Products.

Some of the key requirements for the system work flow are described below.

1. A semantic model should be included as a feature of the new system. The semantic model in this instance is an abstraction that describes the meaning of the model instances that makes up the language metamodel. It also defines the way the model instances relate to the real world examples. It should be able to showcase the

expressive power to interpret meaning (semantics) from the instances.

2. The software has to have a user interface component with familiar notations, permitting its users to represent their mental models about their design intents. It should contain pipeline engineering features that can promote enhanced engineering design modelling for stakeholders without programming or CAD software expertise, and without being burdened by its syntactic or semantic refinements.
3. Users should be able to define modelling parameters in line with pipeline engineering principles. For example, if a pipeline design is constructed from the selection of a number of modelling primitives such as instrument components and support, then the artefact orientation may be considered as corresponding to the variability and commonalities of the primitives. Whereas the variability indicates precisely what evidence is needed to specify an instance of a feature that could fit into particular design intent, the commonalities define the set of common operations and primitives of the language.
4. Users should be able to interpret artefacts. For example, the system should have a knowledge based rule processing solution module, which can allow processing to generate products that reflect oil and gas pipeline real life operations.

7. Conclusion and Future Work

This paper has presented a framework for requirements engineering for oil and gas pipeline systems modeling where the domain of pipeline engineering is analysed and a requirements documentation effected to the goals of a modelling platform capable of transforming pipeline models to stakeholders and domain experts desired artefacts. Development of reusable software will necessitate advances beyond the current generation of programming and computer aided design systems. These systems provide little or negligible design and process guidance for expressing domain specific concepts; the contribution of this paper has been to apply a requirements engineering framework to the pipeline engineering domain for platform simplicity and for improving specification of domain specific libraries. In the future the experimental lessons learned on this framework will be used for design and implementation of a simplified modeling platform and for reuse of domain libraries.

References

- [1] Anthony Finkelstein, Andrea Savigni A Framework for Requirements Engineering for Context-Aware Services Department of Computer Science University College London Gower Street London WC1E 6BT United Kingdom.
- [2] M. Feather, S. Fickas, A. van Lamsweerde, and C. Ponsard. Reconciling System Requirements and Runtime Behavior. In *Proceedings of IWSSD'98 - 9th International Workshop on Software Specification and Design*, Isobe, Japan, April 1998. IEEE Computer Society Press.
- [3] S. Fickas and M. S. Feather. Requirements Monitoring in Dynamic Environments. In *Proceedings of the Second IEEE International Symposium on Requirements Engineering*, pages 140–147. IEEE Computer Society Press, 1995.
- [4] M. Jackson. The World and the Machine. In *Proceedings of the 17th International Conference on Software Engineering*, pages 283 – 292, Seattle, Washington, USA, April 24 – 28 1995.
- [5] Alistair Sutcliffe, George Papamargaritis, Liping Zhao Comparing requirements analysis methods for developing reusable component libraries; *The Journal of Systems and Software* 79 (2006) 273–289.
- [6] Hall, J. G., Jackson, M. J., Laney, R. C., Nusibeh, B., Rananotti, L., 2002. Relating software requirements and architectures using problem frames. In: Greenspan, S., Saddiqui, J., Pohl, K. (Eds.), *Proceedings of RE 02, 1st International Conference on Requirements Engineering*. IEEE Computer Society Press, Los Alamos, CA, pp. 137–145.
- [7] Jarzabek, S., Ong, W. C., Zhang, H., 2003. Handling variant requirements in domain modeling. *Journal of Systems and Software* 68 (3), 171–182.
- [8] MA. Lam, W., McDermid, J. A., Vickers, A. J., 1997. Ten steps towards systematic requirements reuse. In: *Proceedings ISRE '97: 3rd IEEE International Symposium on Requirements Engineering (AnnapolisMD)*. IEEE Computer Society Press, Los Alamitos CA, pp. 6–15.
- [9] Levi, K., Arsanjani, A., 2002. A goal-driven approach to enterprise component identification and specification. *Communications of the ACM* 45 (10), 45–52.
- [10] Mannion, M., Kaindl, H., Weadon, J., 1999. Reusing single system requirements from application family requirements. In: *Proceedings of the International Conference on Software Engineering, ICSE 99*. IEEE Computer Society Press, Los Alamitos CA.
- [11] Ray Offen Domain Understanding is the Key to Successful System Development Division of ICS, Macquarie University, New South Wales, Australia *Requirements Eng (2002) 7:172–175* Springer-Verlag London Limited.
- [12] Charles B. Haley Milton Keynes Jonathan D. Moffett A Framework for Security Requirements Engineering *SESS'06*, May 20–21, 2006, Shanghai, China. 2006 ACM 1-59593-085-X/06/0005...\$5.00.
- [13] McMenamin, S. M., Palmer, J. F., 1984. *Essential Systems Analysis*. Yourdon Press, Englewood Cliffs, NJ.
- [14] Papamargaritis, G., Sutcliffe, A. G., 2004. Applying the domain theory to design for reuse. *BT Technology Journal* 22 (2), 104–115.
- [15] Van Lamsweerde, A., 2001. Goal-oriented requirements engineering: aguided tour. In: *Proceedings of the RE_01—5th IEEE International Symposium on Requirements Engineering*, Toronto, August, 2001. IEEE Computer Society Press, Los Alamitos, CA, pp. 249–263.
- [16] Alessandro NADDEO, Cad Active Models: An Innovative Method in Assembly Environment, *Journal of Industrial Design and Engineering Graphics* Volume 5 Issue No. 1 – 2010.

- [17] Autodesk Inc. (2013) *AutoCAD Release 2013 Programmers Reference Manual*. L. Nayyar.—7th ed. p. cm. ISBN 0-07-047106-1 *McGraw-Hill* 2000.
- [18] Feature-Oriented Domain Analysis (FODA) <http://www.sei.cmu.edu/reports/90tr021.pdf>.
- [19] Nayyar, Mohinder L. Piping handbook / [edited by] Mohinder
- [20] Markus Voelter, Eelco Visser: Product Line Engineering using Domain-Specific Languages Independent, acm.org 2011 15th International Software Product Line Conference 978-0-7695-4487-8/11 2011 IEEE DOI 10.1109/SPLC.2011.25.