Graph-based modelling of systems interaction in model-based systems engineering environment

Maxim Filimonov[(1)], Pathmeswaran Raju[(2)], Craig Chapman[(3)]

[(1)]Knowledge Based Engineering Laboratory, Birmingham City University
Millennium Point, Curzon Street, Birmingham, B4 7XG, United Kingdom
Email: maxim.filimonov@mail.bcu.ac.uk

[(2)]Knowledge Based Engineering Laboratory, Birmingham City University
Millennium Point, Curzon Street, Birmingham, B4 7XG, United Kingdom
Email: path.raju@bcu.ac.uk

[(3)]Knowledge Based Engineering Laboratory, Birmingham City University
Millennium Point, Curzon Street, Birmingham, B4 7XG, United Kingdom
Email: craig.chapman@bcu.ac.uk

ABSTRACT

Systems engineering (SE) is a rapidly developing engineering field. Growing complexity of modern systems causes issues in the development process. Model-based systems engineering (MBSE) is an emerging area in SE and one of the most advanced ways of reducing the complexity. By definition MBSE is a formalised application of modelling to support all aspects of systems development on all lifecycle phases. This paper aims to tackle problems connected with overcomplexity of system models such as lack of communication, lack of understanding and lack of dynamic interaction. The aim is set to be achieved by developing new methods and tools for modelling logic in MBSE, which will allow to avoid problems such as inconsistencies and will be dynamic. Developing new ways of modelling logic will provide improvements for systems engineering.

INTRODUCTION

Modern systems in systems engineering (SE) are becoming more complex, therefore designing such systems is more challenging. Various issues arise in SE like overcomplexity, lack of communication or understanding of the design process on different stages of a lifecycle [1]. Model-based systems engineering (MBSE) has been introduced as a way to overcome these issues and reduce systems complexity. MBSE is "a formalised application of modelling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later lifecycle phases" [2].

According to decomposition principle, the main system model in MBSE is divided into multiple submodels, which correspond to each subsystem involved in the development process [3]. These submodels include design engineering process, computational analysis, cost model, manufacturing analysis, requirements model etc. All these models and systems constantly communicate and interact with each other. However, the interaction is not modelled in a way to dynamically check the consistency of the engineering and systems knowledge. Although systems modelling happens in early stages of SE, models remain static, interaction among different models is pre-defined, often there are inconsistencies in rules, and the dependencies among the rules are not captured [4]. Other issues include logic modelling in form of hard-coded rules, pre-defined relationships, strict constraints and fixed mathematical expressions.

While communication is defined as "means of sending or receiving information", interaction is seen as a two-way process of exchanging information that has an impact on both interacting components [5]. In engineering communication is always involved in an interaction process as any kind of communication either in a form of rules or relationships leads to interaction. Therefore, in scope of this research logic is a way of interaction among different submodels in MBSE, as shown in Fig. 1. For describing logic terms "interaction" and "communication" are utilised simultaneously. Thus, logic in MBSE is not dynamic enough to satisfy evolving nature of system models because of the growing complexity in modern systems [6]. This leads to challenges such as lack of communication, lack of understanding and inconsistencies in form of logical contradictions as well as conflicting knowledge in the development process [7]. Developing more applicable approach for defining interaction among submodels in MBSE is an essential part of systems engineering as a whole.

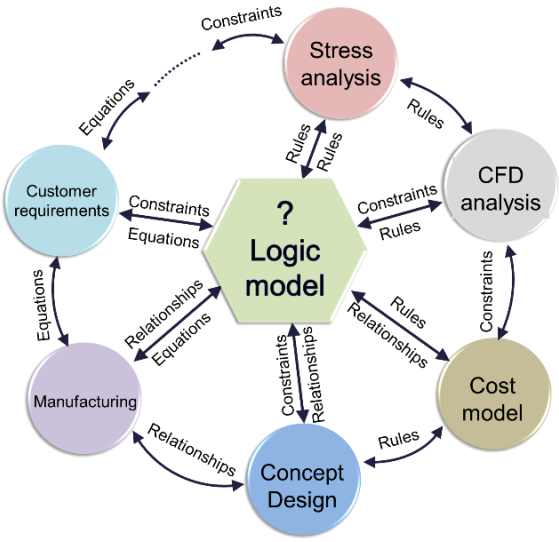

Fig. 1. Logic in the research context

This paper proposes a novel conceptual method where the interaction mechanism can be improved by substituting pre-defined rules and relationships with more sophisticated dynamic approach with utilisation of graph theory. Developing such dynamic ways of modelling logic has the potential to tackle overcomplexity issue as well as deal with inconsistencies of design process by storing all communication in a common interaction module in systems engineering.

RESEARCH CONTEXT

Systems engineering definition evolved through time while scientists added some new aspects as well as making old definitions clearer. Iterative process is as a compulsory part of any design process, which is true for all modern design techniques [8]. Holt spoke of SE simply as "the implementation of common sense" [9], although it is rather difficult to define the meaning of "common sense". Humans have also started to be seen as a component of SE [10]. Modern definition of SE is provided by International Council on SE (INCOSE) and in fact is a combination of various viewpoints – "An interdisciplinary approach and means to enable the realization of successful systems. SE integrates all the disciplines and specialty groups into a team effort forming a structured development process that proceeds from concept to production and operation" [11]. It is significant not to forget about the evolving nature of the core subject itself – systems engineering.

The term "model" is a key concept part of MBSE, which leads to the fact there is a need to define the model itself at first. According to Rumbaugh, a model is identified as a representation of a certain part of the world, which captures needed important aspects and does not include irrelevant features [12]. Model has to obtain three features: it has to be based on original, it has to reflect some properties accordingly and it has to have a purpose to be used in place of the original [13].

MBSE is an emerging approach and is considered to be one the most advanced approaches in engineering [14]. The key aspect of MBSE is the system model which consists of all aspects of the system being developed and used to support the development process on every stage of a lifecycle from meeting requirements to integrating design engineering and engineering analysis. It is recognised that MBSE is going to become the most applicable new generation approach in SE allowing systems engineers to model any kind of systems and support development of any product type [15].

Estefan provides a survey through leading MBSE methodologies [16]. There are numbers of methodologies but among them there is a Vitech MBSE methodology and INCOSE Object-Oriented System Engineering Method (OOSEM), which involve generation of a system model with numerous interacting components. Vitech methodology is based on four concurrently maintained SE activities that are linked together through a common System Design Repository [17]. It is necessary to adequately manage behaviour of model components whereas organised scheme or ontology is essential. OOSEM methodology utilises System Modelling Language (SysML) to support all aspects of systems engineering and it is used alongside object-oriented methodology in a hybrid approach [18].

Comparison between various MBSE methodologies lies beyond the scope of the paper. Nevertheless, one of the MBSE methodologies has to be utilised due to the fact that the field of interest is building improvements upon existing MBSE

basis. Thus, OOSEM methodology seems to be the most suitable candidate at the moment as it involves object-oriented mechanisms which are widely used in programming. Also Systems Modelling Language (SysML) is a key part of OOSEM and this particular language is going to be used in the conceptual framework.

SysML is defined by OMG as "a general-purpose graphical modelling language for specifying, analysing, designing and verifying complex system that may include hardware, software, information, personnel, procedures and facilities" [19]. Technically SysML is a language intended for systems engineers allowing them to model all aspects of systems engineering such as requirements, behaviour and structure. The main purpose of developing SysML is to unify various modelling languages that are used by systems engineers for better cooperation and understanding [1]. The work comes from initiative by OMG and INCOSE [20]. SysML is considered as a new language although it shares a close relationship to Unified Modelling Language (UML). Indeed SysML is based on UML and utilises same kind of diagrams used in UML. Nevertheless, UML was developed in 1997 and mainly aimed at software engineering which leads to the fact that SysML is more advanced, includes all advantages of UML and offers more for SE. SysML and UML relationship is illustrated in Fig. 2 [1].

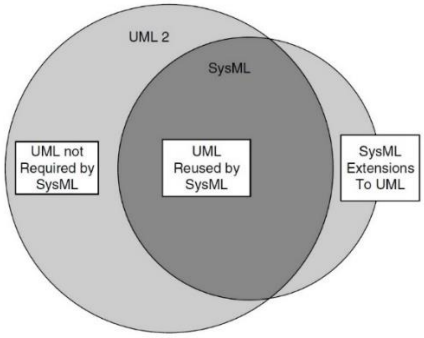

Fig. 2. Relationship between SysML and UML

Willard identifies usefulness of UML/SysML in SE field and recognises SysML as a step in right direction in evolution of modelling languages for systems engineering [21]. SysML is considered to be a more advanced tool for systems engineers than UML which is intended more for software engineers. Thus, SysML is more suitable for the research. Estefan recognises growing significance of SysML and argues that additional diagrams used in SysML allow engineers to model more complex systems for systems engineering purposes [16]. Moreover, Estefan emphasises that SysML is not a methodology itself but a tool that can be utilised in any environment. Actually UML and/or SysML are implemented in most of MBSE methodologies. There are nine diagrams in SysML, each of them represents certain aspects of a system model and can be utilised as a representation of a corresponding submodel/subsystem of the system model for the purpose of defining interactions between models.

RELATED WORK AND RESEARCH GAPS

Growing Complexity

In his book on SE Holt identifies "the three evils of engineering" [1] which are complexity, lack of understanding and communication problems.

- Complexity: Large systems have lots of relationships between its entities and adding more of them make complexity go even higher than it was before. As shown in Fig. 3 [1], to illustrate this problem a block-diagram is utilised to show the complexity increase with the addition of more relationships and rules where (c) is the most complex one.

- Lack of understanding: Concept of "lack of understanding" can arise on any stage of a lifecycle beginning with formulation of requirements which will lead to issues during the development stage and after that even during the operation of a product. Friedenthal argues that understanding of a system model is a major factor [22]. Models have to be dynamic due to the fact that usually there is a need to look at the system only from a particular aspect of certain design. The same thing can be said about overcomplicated models with lots of communication, rules and relationships.

- Communication problems: With the "complexity concept" in place another issue arise, which is the communication problem. This can happen on any level, between several people, groups of people, companies, systems or different

departments involved in process of development. This problem leads to lack of communication between different models which are parts of a main system model in MBSE.

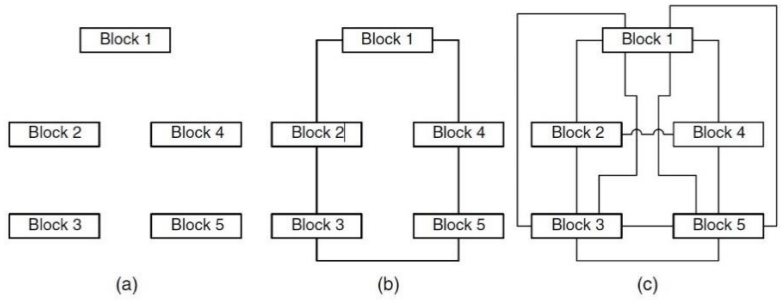

Fig. 3. Complexity description through relationships

Three of these problems cannot happen on their own but they actually generate one another. It is always significant to tackle these issues during early stages of lifecycle. Thus, creating a proper dynamic model of communication between different models of MBSE is a needed step in the development of the corresponding field.

Most related work revolves around implementing one of existing MBSE methodologies as well as making improvements over them. As mentioned before, MBSE is an advanced approach to overcome complexity issues and all the MBSE methodologies propose a way to organise development process by creating models of every aspect involved [16]. Also, as seen from survey through MBSE methodologies, all of them utilise interaction mechanisms where interaction is modelled by hard-coded rules, pre-defined relationships, strict constrains and mathematical equations. To overcome the complexity issues in design process Goossens recognises that modern design approaches tend to develop "virtual" prototypes of their systems and creating complete system model in one environment is the key [23].

In a paper from European Southern Observatory main issues of system projects complexity are stated as identified by NASA's Jet Propulsion Laboratory (JPL) [4], [24]. These issues consist of growing mission complexity, huge amount of pieces in system design without exact architecture, losing knowledge at lifecycle stage boundaries and different technical sides communicating poorly. Author argues that MBSE is an advanced modern approach and identifies SysML as a key for modelling and integrating all parts of systems engineering. The conventional V-model development process is seen as an outdated from experience in MBSE. Karban argues that validation stage comes too late therefore dealing with the problems of design is almost impossible due to the necessity of basically developing everything from scratch once again. MBSE allows the verification and validation of requirements, concept and full design on early stages of the lifecycle. For that purpose proper means of communication between different models are required to be integrated in MBSE.

Inconsistencies

Finkelstein identifies inconsistency as a logical contradiction [25]. As an example of inconsistency in MBSE any technical statement which includes ambiguous definition can be considered. With complexity of systems in industry increasing through time modern approaches tend to decompose main system model into simpler submodels. Thus, quite a lot of inconsistencies can arise when you have several submodels, which need to communicate with each other. Herzig proposes an approach for finding inconsistencies in MBSE and utilises exact matching problem of graph patterns to determine whether or not ambiguous definition of a property exists [7]. Applicability of this approach was shown with the use of actual implementation. However, this approach has its limitations due to the fact that graph patterns have to be exactly the same and, in some cases, even when the pattern is not exactly matching another pattern (only one link is missing), human engineer might be able to find enough evidence to classify this as an inconsistency.

Friedenthal recognises that design inconsistencies can arise especially when multiple people work on the same model [22]. To tackle this problem just a well-defined disciplined process is proposed which leaves space for human mistakes generating inconsistencies. Growing complexity of current system models leads to a high possibility of inconsistencies existence in the form of logical contradictions, which have to be dealt with on all stages of a lifecycle. Developing proper means for communication between all parts of MBSE process and stages of product development lifecycle will allow systems engineers to reduce quantity of inconsistencies and lower possible impact of mistakes in design originating from inconsistent management.

CONCEPTUAL APPROACH

Decomposition principle

According to Fosse, system is composed of interacting subsystems and components, where all of them have some useable functionality that is required by the main system [26]. Functionality of any component or subsystem is a feature, which is independent from connected components or subsystems. Therefore, in the early phases of design lifecycle classic approach to model and increase understanding of a large complex system, which can be any kind of process, product or organisation, is to follow three steps [27]:

1. Decompose main complex system into more simple subsystems and then into components about which we have more information. Ideally the components should be completely independent and correspond to one or limited number of tasks. All the subsystems and components are maintained independently by different means and even engineers involved in the development process. Yassine implies that proper decomposition of a complex system into small manageable parts allows to boost efficiency and making development process concurrent and easier maintained on all phases of a lifecycle [3].
2. Distinguish relationships among the subsystems to understand system behaviour. For that purpose "language of interfaces" can be utilised, where interface is "the system boundary that is presented by a system for interaction with other systems" and described by interface specification containing corresponding boundary properties [26]. Interaction itself can be seen as "operational entity interface", which connects to other operational entities and utilises interaction specification for that purpose.
3. Identify external influence on the system.

Identifying all the relationships among subsystems and components is essential to achieve full understanding of systems behaviour and is the key to create adequate interaction mechanism, which is the main point of interest of this paper and the whole research.

Design Structure Matrix

While the complex system can be decomposed into lots of components, the question still remains how to adequately track all the relationships among separate components. For the purpose of better understanding of these kind of dependencies Design Structure Matrix (DSM) can be utilised. DSM represents the interactions between different components of a system in a matrix form, advantageous for logic analysis in SE. DSM itself is a square matrix, as shown in Fig. 4 [27]. Rows, columns and diagonal elements are identical and stand for different system components. Each non-diagonal mark represents dependency existence between two components in one way or another. For further organisation of systems complexity the process called clustering is utilised. Clusters are special combinations of elements in DSM, where all the interaction is done inside of a cluster, minimising links between clusters.

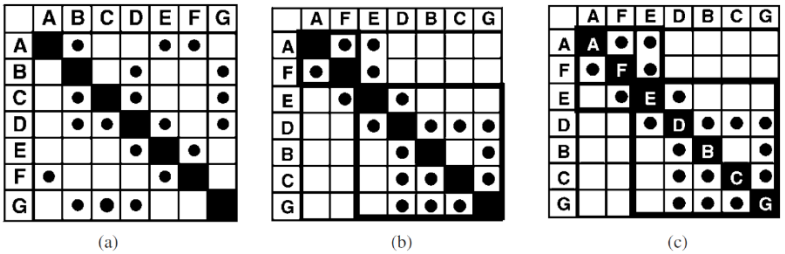

Fig. 4. Example DSM:
(a) Base DSM; (b) Clustered DSM; (c) Alternative Clustering DSM

Overall Yassine implies that DSM method helps to control complexity and makes a better formal representation of the interactions among different subsystems. Clustering and partitioning allow to minimise needed quantity of iterations, which as well leads to reducing complexity. DSM itself is a useful matrix representation of a directed graphs that can be integrated in the development process. This leads to the next stage – implementation of graph theory.

Graph Theory

Graph is a "representation of a set of objects where some pairs of objects are connected by links. The interconnected objects are represented by mathematical abstractions called vertices (also called nodes or points), and the links that connect some pairs of vertices are called edges (also called arcs or lines)" [28].

Basu recognises the fact that any big system with numerous related entities can be modelled by some kind of a graph [29]. By utilising graph constructs number of benefits is provided. First of all, basis for use of graphs is the reason that it is an effective way to represent, visualise and understand very complex systems. This might not be possible with conventional system description techniques either text-based or formal ones, which are useful only to those who can understand utilised formal language. Secondly, formal properties of graph structures provide an applicable way to analyse structure and behaviour of complex systems. Herzig recognises labelled directed multigraphs as the best way of model representation and utilisation as shown in Fig. 5 [7]. Every statement is divided into three parts – subject, predicate and object, vertices of a graph stand for subject and object as directed labelled edge is a predicate between subject and object [30].

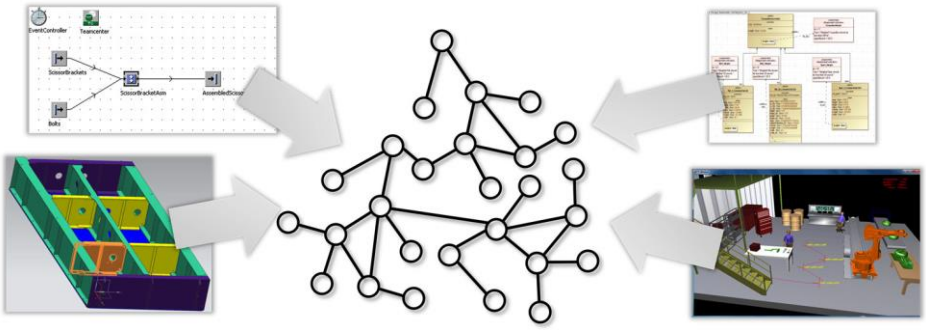

Fig. 5. Graphs as a way of model representation

The more sophisticated concept such as hypergraph is seen to be even more advanced way that can be utilised in SE. Hypergraph handles any generalised object that represents physical or abstract properties: particles, states, points in space etc. Gazdik's analysis of modern graph approaches shows that hypergraph has to be considered as one of the most adaptable tool for modelling systems [31]. Gazdik provides an illustrative example to show the applicability of hypergraph approach to model complex large-scale systems. From this example advantages of hypergraph utilisation can be deduced. Hypergraph can be represented in a matrix form and, therefore, easily utilised by a computer. Matrix representation in certain moment in time provides fixed complex system configuration in that moment. Vertices of a hyperedge represent generalised object with any kind of physical or abstract properties. The configuration of hyperedges of a hypergraph can be modified reflecting changes in any component or a subsystem of a complex system. This shows applicability of hypergraph approach for the aim of the whole research – propose novel approach to model systems interaction.

Metagraphs, an extension of directed graphs and hypergraphs, differ from conventional graphical constructs as each edge is an ordered pair of sets of elements. Metagraphs are considered to be a powerful method for decision support systems as they can be utilised for analysis of interaction between components no matter what these components are - models, relationships or rules [32].

Conceptual Framework

Based on the discussion above, a conceptual framework has been developed and is shown in Fig. 6. The first phase is decomposition phase at which system is decomposed into subsystems and components. In ideal case all subsystems and components are completely independent. The next phase is interface phase where boundaries that are presented by subsystems for interaction are distinguished. During that phase all relationships among subsystems and components are identified and classified. The third phase is design structure matrix phase where previously distinguished relationships are organised using DSM. Then the original DSM is clustered for further organisation. The next phase – graph construction phase – is the main phase of the framework at which DSM is reconstructed using hypergraphs/metagraphs concepts in order to create adequate logic model of subsystems interaction. The final phase is validation and verification phase where developed interaction graph model is tested in different scenarios with lots of changes in subsystems in components for errors and inconsistencies. Efficiency of the developed interaction mechanism is tested against chosen set of metrics.

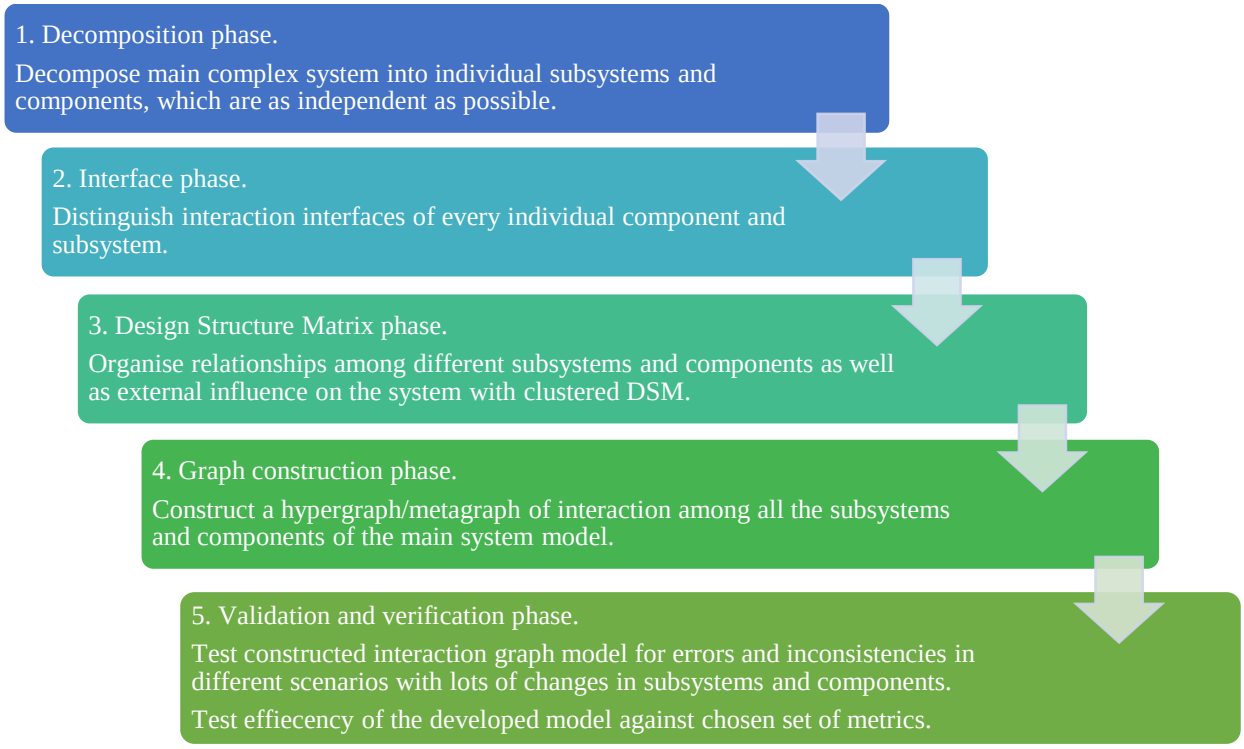

Fig. 6. Phases in the conceptual framework

SUMMARY AND CONCLUSIONS

In this paper, a concept approch to modelling logic of systems interaction is presented. Modelling systems interaction is an essential part of model-based systems engineering as modern systems are becoming more complex and involve large amount of interacting subsystems and components. Currently MBSE is an advanced approach to reducing systems complexity as seen from a survey from current MBSE methodologies. Despite the fact that all of them are succesful in reducing overall systems complexity to a certain extent, for modelling logic not dynamic mechasims are utilised with hard-coded rules, pre-defined relationships and fixed constrains with direct links between different subsystems and components. Moreover, different components and subsystems represent entirely different abstact of physical properties.

We argue, that old not dynamic concepts can be substituted by a more sophisticated dynamic approach with utilisation of graph theory. This paper provides definitions and general information about hypergraphs and metagraphs, extensions of conventional graphs, highly applicable for systems modelling in MBSE. Any kind of abstract and physical properties can be represented by certain elements of a hypergraph.

With systems and models becoming more and more complex in systems engineering field, issues of overcomplexity, lack of understanding and communication problems in MBSE lead to number of problems in the design process. The main requirement for everyone involved in a development process in industry is to create products that are high quality, less expensive, more advanced and manufactured in less time. For that purpose interaction mechanism based on hypergraph/metagraph theory can be utilised to improve systems modelling and design processes. Developing a new way of modelling logic in MBSE and improving over existing ways tackle all kind of identified problems and push forward state of SE field. This involves helping systems engineers to avoid inconsistencies in development process and require less time to develop complex system models with lots of interacting components as well as making a way of communication dynamic in its nature. The future works includes further exploring hypergraph and metagraphs theories as well as their applications for systems modelling. Proof-of-concept implementation of the proposed conceptual approach is set to be done in the nearest future to be able to assess how the framework works in actual real-life examples.

REFERENCES

[1] J. Holt and S. Perry, *SysML for systems engineering*, vol. 7. IET, 2008.

[2] "Systems Engineering Vision 2020," *INCOSE-TP-2004-004-02, Version 2.03*, 2007.

[3] A. Yassine and D. Braha, "Complex concurrent engineering and the design structure matrix method," *Concurr. Eng.*, vol. 11, no. 3, pp. 165–176, 2003.

[4] R. Karban, L. Andolfato, P. Bristow, G. Chiozzi, M. Esselborn, M. Schilling, C. Schmid, H. Sommer, and M. Zamparelli, "Model based systems engineering for astronomical projects," in *SPIE Astronomical Telescopes+ Instrumentation*, 2014, p. 91500L–91500L.

[5] "Oxford Dictionary." [Online]. Available: http://www.oxforddictionaries.com/.

[6] C. B. Chapman and M. Pinfold, "Design engineering - a need to rethink the solution using knowledge based engineering," *Knowledge-based Syst.*, vol. 12, no. 5, pp. 257–267, 1999.

[7] S. J. I. Herzig, A. Qamar, and C. J. J. Paredis, "An approach to identifying inconsistencies in model-based systems engineering," *Procedia Comput. Sci.*, vol. 28, pp. 354–362, 2014.

[8] H. Eisner, *Essentials of project and systems engineering management*. John Wiley & Sons, 2008.

[9] J. Holt, *UML for Systems Engineering: watching the wheels*, vol. 4. IET, 2004.

[10] D. Hybertson and S. Sheard, "Integrating and Unifying Old and New SE Elements," *INSIGHT*, vol. 11, no. 1, pp. 13–16, 2008.

[11] C. Haskins, *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, v. 3*. John Wiley & Sons, 2006.

[12] J. Rumbaugh, I. Jacobson, and G. Booch, *Unified Modeling Language Reference Manual*. Pearson Higher Education, 2004.

[13] H. Stachowiak, "General Model Theory [Allgemeine Modelltheorie]," p. 494, 1973.

[14] D. Rhodes, "Addressing systems engineering challenges through collaborative research," *SEARI-Systems Eng. Adv. Res. Initiat. Massachusetts Inst. Technol.*, 2008.

[15] A. L. Ramos, J. V. Ferreira, and J. Barceló, "Model-based systems engineering: An emerging approach for modern systems," *Syst. Man, Cybern. Part C Appl. Rev. IEEE Trans.*, vol. 42, no. 1, pp. 101–111, 2012.

[16] J. A. Estefan, "Survey of Model-Based Systems Engineering (MBSE) Methodologies." Jet Propulsion Laboratory, California Institute of Science. INCOSE MBSE Initiative, 2008.

[17] L. Baker Jr and J. E. Long, "Role of System Engineering Across The System Life Cycle," *Vitech White Pap. Vitech Corp.*, 2000.

[18] H. Lykins, S. Friedenthal, and A. Meilich, "Adapting UML for an Object Oriented Systems Engineering Method," in *in the Proceedings of the Tenth Annual International Symposium of the International Council on Systems Engineering*, 2000.

[19] "The Official OMG SysML site." [Online]. Available: http://www.omgsysml.org/.

[20] "INCOSE Model-Based Systems Engineering (MBSE) initiative." [Online]. Available: http://www.omgwiki.org/MBSE/doku.php.

[21] B. Willard, "UML for systems engineering," *Comput. Stand. Interfaces*, vol. 29, no. 1, pp. 69–81, 2007.

[22] S. Friedenthal, A. Moore, and R. Steiner, *A practical guide to SysML: the systems modeling language*. Morgan Kaufmann, 2014.

[23] P. Goossens, "Model-Driven Innovation in Machine Design," *Engineering Solutions, Maplesoft*, 2016. [Online]. Available: https://www.theengineer.co.uk/supplier-network/product/model-driven-innovation-in-machine-design/.

[24] "Integrated Model-Centric Engineering (IMCE) Workshop for JEO," 2011.

[25] A. Finkelstein, "A foolish consistency: Technical challenges in consistency management," in *Database and Expert Systems Applications*, 2000, pp. 1–5.

[26] E. Fosse and C. L. Delp, "Systems engineering interfaces: A model based approach," in *Aerospace Conference, 2013 IEEE*, 2013, pp. 1–8.

[27] T. R. Browning, "Applying the design structure matrix to system decomposition and integration problems: a review and new directions," *Eng. Manag. IEEE Trans.*, vol. 48, no. 3, pp. 292–306, 2001.

[28] R. Diestel, *Graph theory*. Springer, 2000.

[29] A. Basu and R. W. Blanning, "Metagraphs: A tool for modeling decision support systems," *Manage. Sci.*, vol. 40, no. 12, pp. 1579–1600, 1994.

[30] J. C. Giarratano and G. Riley, *Expert systems*. PWS Publishing Co., 1998.

[31] I. Gazdík, "Modelling systems by hypergraphs," *Kybernetes*, vol. 35, no. 9, pp. 1369–1381, 2006.

[32] A. Basu and R. W. Blanning, "Metagraphs in workflow support systems," *Decis. Support Syst.*, vol. 25, no. 3, pp. 199–208, 1999.