Can SysML Be Used for Enterprise Modelling?

Kimberly Lai[1] and Michael Gruninger[2]

[1] Department of Mechanical and Industrial Engineering, University of Toronto, Canada
kimberly.lai@mail.utoronto.ca
[2] Department of Mechanical and Industrial Engineering, University of Toronto, Canada
gruninger@mie.utoronto.ca

Abstract. Although a variety of specialised formalisms have been proposed specifically for enterprise modelling, the use of existing modelling languages has not received as much attention. In this paper, we demonstrate that the systems modelling formalism SysML is in fact not sufficient to act as a standalone language for enterprise modelling. To demonstrate this claim, we show that there are four key enterprise modelling scenarios that cannot be addressed while adhering to SysML semantics: temporal representation, timing and scheduling, collaborations between two or more teams and decision trees.

Keywords: Enterprise modelling, SysML, Civil aircraft system case study.

1 Introduction

Within academia and industry, there are many different modelling formalisms or languages that can be use in enterprise modelling, including UML [1], SysML [2], BPMN [3] and the IDEF modelling language family [4]. Depending on the purpose and context of a model, different languages are chosen as each has its own strengths and weaknesses. In practice, an enterprise model is often made up of a combination of models and hence comprises of a combination of languages. Certain modelling languages are used to model certain aspects of the enterprise while another language is used to model another aspect. This results in a lot of inconsistency as concepts may not be clearly defined and interpretations can differ. Thus, in an ideal scenario, having one modelling language that can cover all aspects of the enterprise would solve this problem.

On the other hand, it is postulated in academia that an enterprise can be considered as a system. Treating an enterprise as a system and using a systems engineering approach can result in a more efficient and effective running of the enterprise [5]. In terms of a modelling language for system models, SysML is the standard modelling language that has been tailored for systems engineering applications. It supports the specification, analysis, design, verification, and validation of a wide range of complex systems [6].

As a result, it would be reasonable to hypothesise that SysML can act as a standalone language for enterprise modelling. However, in this paper, we claim that although SysML can be used to model many aspects of an enterprise, it is in fact not sufficient to provide a full depiction. There are four key enterprise modelling scenarios that cannot be addressed while adhering to SysML semantics: temporal representation, timing and scheduling, collaborations between two or more teams and decision trees. This will be demonstrated using

example diagrams in the context of an enterprise that designs and manufactures civil aircraft systems, with a specific focus on the system development and safety assessment processes.

2 Background and Related Work

2.1 Enterprise Modelling and Systems Modelling

Enterprise modelling is a field that has been gaining attention for the last few decades. As a result of digitalisation and globalisation, enterprises must become increasingly agile and continuously adapt to change in order to survive. One approach to remain competitive and achieve agility is to create an enterprise model to represent the organisation. As defined in [7], an enterprise model is a computational model that illustrates "the structure, activities, processes, information, resources, people, behaviour and goals of an enterprise". It provides all the information and knowledge necessary to achieve model-driven enterprise design, analysis, and operation.

Using a model-based approach to run an enterprise can bring about many benefits. This includes better integration and communication between various departments within an enterprise, as well as an improved understanding of the enterprise as a whole. It also results in better decision making as analysis can be performed before execution and the consequences of any sudden changes can be traced throughout the enterprise.

There are two key aspects that must be determined before any models are built. The first is the enterprise architecture/modelling framework which dictates how the enterprise will be broken down and represented. This is achieved by organising the model into various views or layers such as the business view, operations view, conceptual view, technical view and implementation view. Currently, there are a variety of existing frameworks that have been designed to support various modelling purposes and levels of granularity. Examples include: DoDAF, TOGAF, the Zachman Framework and CIMOSA [8]. The second key aspect is the modelling language that is used to draw the diagrams that make up the enterprise model. The modelling language dictates how certain concepts are graphically represented and the type of information that is presented in various types of diagrams. Examples of modelling languages include UML, which is used in software engineering, SysML which is used in systems engineering, and BPMN and IDEF3 which are used for process modelling.

It is often suggested in academia that an enterprise can be considered as a system. In the Handbook of Systems Engineering and Management [5] it discusses how treating an enterprise as a system and using a systems engineering approach can lead to a more efficient and effective running of the enterprise. Rouse [9] also suggests that understanding enterprises as systems is critical to addressing strategic challenges such as achieving growth and responding to change. Understanding interactions between different functions within an enterprise is also essential to fully leveraging an enterprise's assets. Following on from this logic, enterprise modelling can be considered as a variant of systems modelling and hence using SysML as an enterprise modelling language could be possible.

2.2 The Systems Modeling Language (SysML)

SysML is a visual modelling language that provides the semantics and notations for modelling a system. It was developed by the Object Modelling Group (OMG) specifically for systems engineering applications and is an extension of the Unified Modeling Language (UML), which was first developed as a generic modelling language in the software field. As it was developed specifically for systems engineering applications, there are certain concepts and diagram types that are introduced to support activities such as requirements specification and trade studies for design analysis. The diagrams defined by SysML semantics are also classified into four pillars: Behaviour, Structure, Parametric and Requirements.

SysML is well accepted by industry and academia as the de-facto standard for systems modelling. According to [10] it supports the "specification, analysis, design, verification and validations of systems that include hardware, software, data, personnel, procedures and facilities". Hence, it can describe a complex system from concept development and requirements writing all the way to design, implementation, verification, and validation activities.

SysML also has a profile extension mechanism that allows users to customise its profile so that it can be modified to suit the users' needs. This is a very useful capability, especially in the context of enterprise modelling, as every enterprise will have its preference for domain-specific vocabulary. With SysML, the names of pre-defined stereotypes can be modified, and specialisations of stereotypes can also be created to match the language typically used in the organisation. For example, for an enterprise that produces complex mechanical products, the requirement stereotype can be specialised into a functional requirement, performance requirement and constraint requirement to better capture all the requirements that need to be satisfied.

By examining the definitions of enterprise modelling and SysML, it can be seen that there is a large amount of overlap between the concepts covered by these formalisms. Hence, if systems modelling is indeed a variant of enterprise modelling, it is logical to consider SysML as a plausible language for enterprise modelling.

2.3 Related Work on Enterprise Modelling Languages

Ongoing and existing research in the field of enterprise modelling languages largely fit into two categories. The first category involves exploring how multiple languages representing different domains can be integrated together to cover all aspects of an enterprise model. For example, TOGAF provides a framework to integrate various domains within an enterprise including business, data, and technical architectures [11]. The second category is the introduction of a new modelling language that has been specifically created for the purpose of enterprise modelling. For example, the use of the Unified Enterprise Modelling Language (UEML) is proposed in [12] which acts as a simple universal language that functions as a standard user interface on top of existing languages and systems. Domain-specific models can then be translated into UEML models and interact with other domain models. On the other hand, The Open Group also introduce the use of the ArchiMate modelling technique [13] for describing enterprise architectures.

Works that contribute to assessing enterprise modelling techniques focus mostly on enterprise architecture/modelling frameworks. For example, [14] evaluates whether these frameworks provide a structure that will allow all aspects of an enterprise to be modelled.

There has also been some work done on evaluating the adequacy of languages that have been specifically created for enterprise modelling purposes. For example, [15] evaluates how well ArchiMate responds to common enterprise modelling challenges.

However, beyond the work in [16], limited research has been performed on evaluating whether modelling languages that are already widely used in industry, such as BPMN or SysML, can be used for enterprise modelling purposes instead. If an existing language is already sufficient for enterprise modelling, this would greatly simplify the most common challenge of language inconsistency for enterprise models. Therefore, since it is often suggested that an enterprise can be considered as a kind of system, an evaluation on the adequacy of SysML as an enterprise modelling language is needed.

3 Case Study

3.1 Method

To investigate if SysML can be used as a standalone language for enterprise modelling, the following steps were performed. First, a list of common enterprise modelling scenarios was created by examining existing enterprise modelling frameworks such as GERAM and CIMOSA. These scenarios were then modelled using SysML on the Papyrus tool, which is an open-source graphical editing tool developed by Eclipse that has been designed to support SysML and adheres to all its semantics and rules. Finally, each of the scenarios were then analysed in order to determine whether SysML was able to adequately portray all the intended information.

Table 1 below provides a summary of common modelling scenarios that contribute to describing an enterprise as well as the corresponding type of SysML diagram that was used to represent them. Of these scenarios, there are three that cannot be represented accurately which are indicated in bold below: interaction between entities/teams, process interaction and decision tree. The conclusions drawn and challenges faced from modelling these scenarios will be discussed in Section 3.3.

Table 1. Summary of modelling scenarios.

Enterprise modelling scenario	Type of SysML diagram to use
Organisation hierarchy	Block definition diagram [bdd]
Process breakdown	Block definition diagram [bdd]; Internal block diagram [ibd]
Interaction between entities/teams	**Activity diagram [act]**
Process interaction	**Activity diagram [act]**
Context & Stakeholders	Use case diagram [uc]
Deliverable lifecycle	State machine diagram [stm]
Decision tree	**State machine diagram [stm]**
Requirements flow down	Requirement diagram [req]
Trade studies	Parametric diagram [par]
Task allocation	Allocation table [alloc]

3.2 Context: System Development & Safety Processes for Civil Aircraft

The example scenarios shown in this paper are performed from the perspective of an enterprise that is involved in the development of civil aircraft systems. In particular, some of the information related to the system development and safety assessment processes for a civil aircraft system are modelled. Standard documentation of these processes can also be found in ARP4754A [17] and ARP4761 [18].

3.3 Diagrams

The purpose of this paper is to identify and describe the inadequacies of SysML as an enterprise modelling language. Therefore, in this section, only the three scenarios mentioned above that could not be adequately portrayed using SysML will be presented. The challenges faced when using SysML semantics to represent the desired information will also be discussed below. For reference, the scenarios that were successfully represented using SysML can be found in the full paper **here**.

Interaction Between Entities. For an enterprise to operate smoothly and efficiently, multiple entities within an enterprise will often work together to achieve a common goal. As such, the representation of interactions between various entities is a crucial part of an enterprise model. An *Activity Diagram*, such as **Fig. 1**, can be used for this purpose, where swim lanes can be used to represent the different entities, and activities are used to represent the activities, tasks and processes that are performed. A swim lane can be used to represent either a specific individual, a team, or even an entire department, and hence it is suitable for detailing high level processes as well as low level ones.

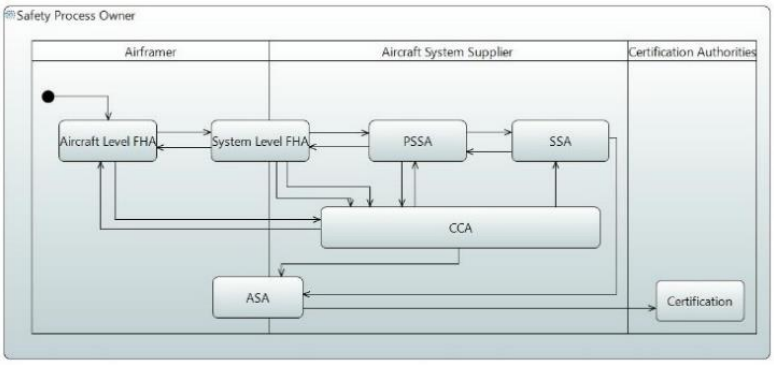

Fig. 1. Interaction between entities

However, one major drawback is that representing activities that are performed in collaboration between two or more entities or teams is impossible without violating SysML rules. In the context of aircraft system safety processes, the activities “System Level FHA” and “ASA” should be a collaborative effort by the Airframer and Aircraft System Supplier. Hence, as shown above, the activity is placed in between two swim lanes to reflect this. However, this could only be accomplished by disabling the simulation and model checking

feature in Papyrus as placing an activity between two swim lanes is prohibited by the modelling tool. According to SysML semantics, placing an activity within a particular swim lane is equivalent to allocating that activity to the actor or element the swim lane represents and hence, only one allocation can be made. Therefore, despite successfully showing the intended information, **Fig. 1** is semantically incorrect.

An alternative to an *Activity Diagram* is to use a *Sequence Diagram* to represent interactions between processes, with each lifeline representing a different entity. However, the same problem exists as each activity can only be drawn on top of one lifeline and cannot span across two or more.

Process Interaction. The representation of interactions between processes is another scenario that must be included within an enterprise model as it contributes to defining the operations of an enterprise. Of all the SysML diagram types, an *Activity Diagram* is best suited for this purpose. Processes can be represented by *Activities*, and *swim lanes* can represent the larger overarching super-process that a certain group of processes contribute to. For example, in **Fig. 2** the diagram describes how the two major processes, the system development process and safety process, interact with one another and what type of information is transmitted. By using swim lanes, it can be shown that the Aircraft level FHA, System level FHA and PSSA are all sub-processes of the overall Safety Process.

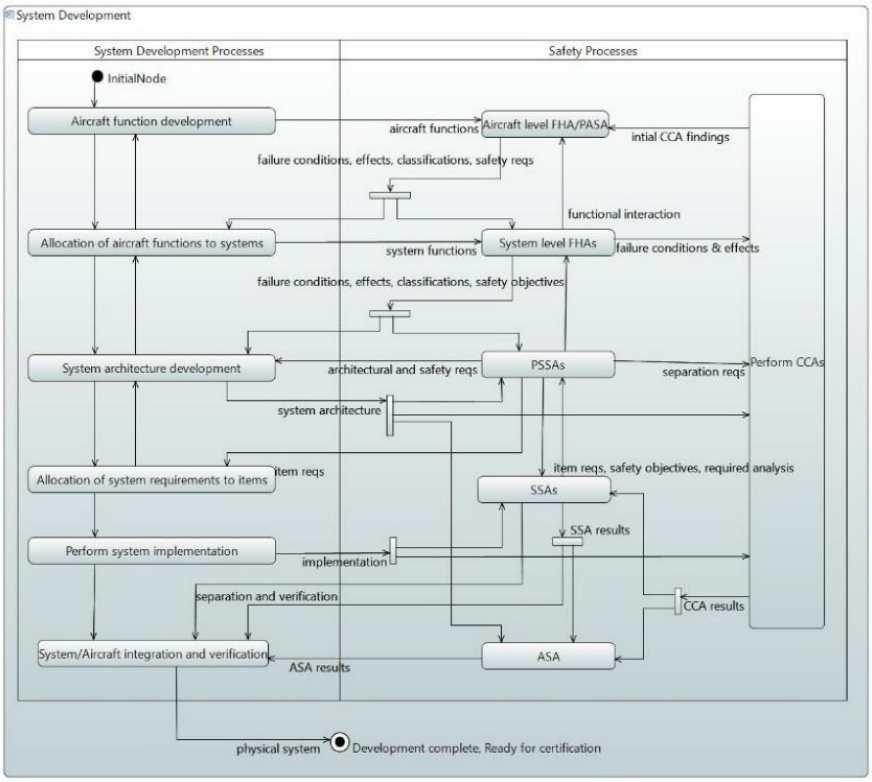

Fig. 2. Process interaction

However, one major problem in this representation is the lack of any temporal information. This diagram provides a great overview at all the important processes and how they interact, however, SysML semantics does not account for being able to represent the duration of each process. Knowing the expected duration of each sub-process and being able to predict the amount of time needed for completion can be extremely important, especially for time sensitive projects or tasks. Furthermore, without the support of temporal information, planning and scheduling is not possible, and hence this is another major drawback of SysML.

Moreover, it became apparent while drawing this diagram that the figure could very quickly become quite complex and visually confusing. SysML semantics dictate the use of *merge* and *split nodes* when one type of information needs to go from one element to two different ones or vice versa, thus adding to the complexity of the diagram. For scenarios of a higher complexity with much more information transfer than is shown in **Fig. 2**, the diagram would become very dense and disorganised which is impractical.

Decision Trees. Decision trees are useful in providing a common guideline on how certain actions or decisions are made within an enterprise. For example, it is specified in ARP4761 [18] that once failure conditions are identified as part of the FHA process, there are three possible next steps. This is determined by a decision tree which has the FHA failure conditions as the top-level input as shown in **Fig. 3**. A *State Machine Diagram* can be used to represent this, yet this only works visually. In terms of SysML semantics, what is shown in this diagram is not correct; it violates SysML semantics and cannot be executed. State machines are meant to represent states, operations, and events, and hence drawing a diagram in this format where the decision nodes are used to represent questions in a decision tree is not valid. Therefore, the conclusion can be reached that SysML fails to support the modelling of decision trees.

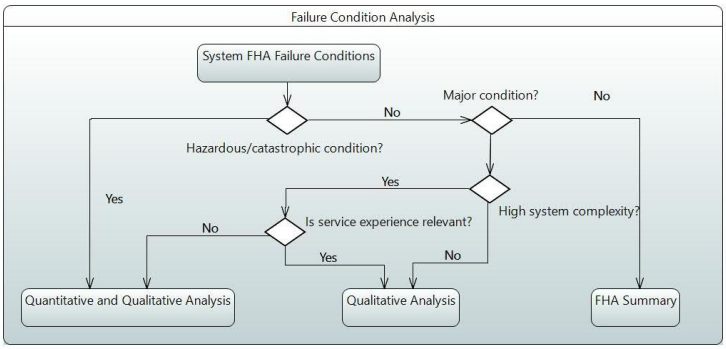

Fig. 3. Decision tree

4 Challenges of Using SysML for Enterprise Modelling

In summary, by drawing various enterprise model scenarios using SysML, we have identified four key enterprise modelling aspects that SysML is unable to represent. Although this

may not be exhaustive, it is enough to demonstrate some counterexamples that prove that SysML cannot be used as a standalone language for enterprise modelling despite it being a logical assumption as discussed in Section 1 and 2. While each of these modelling aspects have already been mentioned in the previous section, they will be summarised again here.

4.1 Temporal Representation

Firstly, SysML does not provide a means to incorporate time into the model. When modelling processes with an activity diagram or a state machine diagram, it is not possible to indicate of how much time is needed for each activity or how long the overall process could take. One potential approach could be to introduce a new attribute to represent time and use simulation to calculate how much time is needed for certain processes. However, from a graphical standpoint this does not solve the problem and is a very impractical method. There would still be no indication of time when looking at the diagram and the user would have to manually explore the properties of each model element to retrieve this information.

4.2 Timing and Scheduling

Similarly, representing timelines and creating schedules is also something that SysML cannot do. By definition, an enterprise model should supply the information and knowledge necessary to support the operations of the enterprise. Scheduling plays a crucial part in planning the operations of an enterprise and thus, without the capability to do so, this is quite a significant aspect of an enterprise model that cannot be represented.

4.3 Collaboration Between Teams

Representing activities as a collaboration between two or more teams is another scenario that SysML is unable to handle. In large organisations that have multiple teams, the scenario where multiple teams or individuals are responsible for one activity or deliverable is particularly common. Therefore, being able to indicate that something is under the joint responsibility of two or more stakeholders is necessary to avoid any confusion or misunderstandings. As discussed in Section 3.3, it is impossible to place an activity between two swim lanes without breaking SysML semantics that are implicitly enforced in the Papyrus tool. **Fig. 1** was only achieved by making the model non-executable, however this is obviously not an ideal solution as the diagram is no longer valid.

4.4 Decision Tree

Finally, is the challenge of representing a decision tree using SysML. Decision trees are useful in standardising how certain decisions are made in an enterprise and how to approach various scenarios. It can also be used to facilitate business decision making as they can be used to calculate probabilities and outcomes. A state machine diagram can be used to present the decision-making information graphically, however this causes the diagram to be semantically incorrect and invalid. As such, it can be concluded that SysML semantics do not support the generation of decision trees either.

5 Conclusion & Future Work

In conclusion, SysML is not sufficient to act as a standalone language for enterprise modelling. As demonstrated by this paper, there are several scenarios that should be described by an enterprise model which cannot be represented while still adhering to SysML semantics. In order to construct an enterprise model that will describe all the information necessary to support the operations of an enterprise, SysML will need to be supplemented with other languages or frameworks to help represent aspects that SysML cannot. For example, SysML could be combined with TimeML [19], a specification language developed for event and temporal expressions, or an ontology framework such as the Process Specification Language (PSL) [20] ontology which also supports temporal concepts [21].

Considering that the findings presented in this paper are a result of the authors' experience and expertise, this raises the challenge of being able to derive formal results about the expressiveness of modelling languages. In particular, how can one *prove* that temporal constraints and various other aspects cannot be represented by SysML?

Another question worth considering for further research efforts is whether an enterprise can indeed be regarded as a kind of system. As previously discussed, it is suggested by literature that an enterprise can in fact be treated as a system and using a systems engineering approach for an enterprise can result in a more efficient and effective running of the enterprise. Therefore, if an enterprise is considered a kind of system, then the results of this paper suggest that SysML is in fact not sufficient as a systems modelling language and needs to be extended such that all use cases are covered. Alternatively, an enterprise could instead be regarded as an extension of a system and hence it is only logical that SysML is not sufficient as an enterprise modelling language. Nevertheless, further exploration is needed on how an enterprise can be considered as a system and what sort of transformations need to be carried out for this to be true. Following from that, the shortcomings of SysML identified here will need to be evaluated once more to see if they are still valid.

References

1. Object Management Group, "Unified Modeling Language v2.5.1," Decemeber 2017. [Online]. Available: https://www.omg.org/spec/UML/2.5.1/PDF. [Accessed June 2021].
2. Object Management Group, "SysML v1.6," November 2019. [Online]. Available: https://www.omg.org/spec/SysML/1.6/PDF. [Accessed June 2021].
3. Object Management Group, "Business Process Model and Notation (BPMN), Version 2.0," January 2011. [Online]. Available: https://www.omg.org/spec/BPMN/2.0/PDF. [Accessed June 2021].
4. C. Menzel and R. Mayer, "The IDEF family of languages," in Handbook on Architectures of Information Systems, Berlin, Springer, 2006, pp. 215-249.
5. A. P. Sage and W. B. Rouse, Handbook of Systems Engineering and Management, 2nd ed., New York: Wiley, 2009.
6. Object Management Group, "SysML Open Source Project," [Online]. Available: https://sysml.org/. [Accessed 10 March 2021].
7. M. Gruninger and M. S. Fox, "The Logic of Enterprise Modelling," in Modelling and Methodologies for Enterprise Integration, Boston, MA, Springer, 1996, pp. 140-157.

8. L. Urbaczewski and S. Mrdalj, "A Comparison of Enterprise Architecture Frameworks," Issues in Information Systems, vol. VII, no. 2, 2006.
9. W. B. Rouse, "Enterprises as systems: Essential challenges and approaches to transformation," Systems Engineering, vol. 8, no. 2, pp. 138-150, 2005.
10. M. Hause, "The SysML Modelling Language," in Fifth European Systems Engineering Conference, 2006.
11. The Open Group, "The TOGAF Standard, Version 9.2," [Online]. Available: https://pubs.opengroup.org/architecture/togaf9-doc/arch/welcome.html. [Accessed June 2021].
12. F. B. Vernadat, "UEML: Towards a Unified Enterprise Modelling Language," International Journal of Production Research, vol. 40, no. 17, pp. 4309-4321, 202.
13. Visual Paradigm, "What is ArchiMate?," [Online]. Available: https://www.visual-paradigm.com/guide/archimate/what-is-archimate/. [Accessed June 2021].
14. S. Leist and G. Zellner, "Evaluation of current architecture frameworks," in SAC '06: Proceedings of the 2006 ACM symposium on Applied computing, 2006.
15. B. v. Gils and H. A. Proper, "Enterprise Modelling in the Age of Digital Transformation," in PoEM 2018, 2018.
16. A. Tsadimas, "Model-based enterprise information system architectural design with SysML," in 2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS), Athens, Greece, 2015.
17. SAE Aerospace, SAE4754A: Guidelines for Development of Civil Aircraft and Systems, 2010.
18. SAE Aerospace, ARP4761: Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment.
19. J. Pustejovsky, J. Castano, R. Ingria, R. Saurí, R. Gaizauskas, A. Setzer and G. Katz, "TimeML: A Specification Language for Temporal and Event Expressions," Kluwer Academic Publishers, Netherlands, 2003. [Accessed September 2021].
20. C. Schlenoff, J. Lubell, M. Gruninger, F. Tissot, J. Valois and J. Lee, The Process Specification Language (PSL) Overview and Version 1.0 Specification, Gaithersburg, MD: NIST Interagency/Internal Report (NISTIR), National Institute of Standards and Technology, 2000. [Accessed September 2021].
21. R. Sriram and M. Brady, "Ontology for Big Systems: The Ontology Summit 2012 Communique," Applied Ontology, vol. 8, no. 3, 2013.