Model-Based-Systems-Engineering for bridges: Representing modular precast bridges with SysML

Benedict Harder[1] and Sebastian Esser[1]
[1]Chair of Computational Modeling and Simulation, Technical University of Munich, Arcisstraße 21, 80333 Munich, Germany
E-mail(s): benedict.harder@tum.de, sebastian.esser@tum.de

Abstract: Model-Based-Systems-Engineering (MBSE) and its corresponding modeling language, SysML, have been widely used in mechanical and systems engineering. While its underlying structure was designed with systems engineering in mind, buildings in the AEC domain can also be represented using the same principles of structure, behavior, requirements, and parametrics. Specifically, bridge constructions can benefit from representation through clearly defined constraints and specifications. This paper focuses on developing a concept to adapt system engineering principles to the modular precast bridge construction space, resulting in a preliminary method of representing bridges with the system modeling language. SysML diagrams representing the core functions of an exemplary bridge were created to showcase the viability of using MBSE and the necessary modifications to be appropriately adapted into the AEC domain.
Keywords: Design Automation, MBSE, SysML, BIM

Erschienen in Tagungsband 35. Forum Bauinformatik 2024, Hamburg, Deutschland, DOI: 10.15480/882.13505

1 Introduction

In recent years, construction projects have started to transition from document-based information management, such as 2D plans and specifications, to model-based information management. The method of choice for this new approach is Building Information Modeling (BIM), which aims to reduce costs and errors, as well as provide a suitable base for modeling data adhering to paradigms such as *single-source-of-truth* [1]. Meanwhile, a similar paradigm shift has happened in the field of systems engineering. Model-Based-Systems-Engineering (MBSE) promises similar advantages as BIM by eliminating the need for document-based system specifications.

Naturally, there are stark differences between BIM and MBSE. BIM is designed to manage all the necessary information concerning planning, construction, operation, and demolition [1] accurately and consistently. In contrast, MBSE was created with precise, time-critical system procedures and complex component interactions in mind. Thus, the tools created to assist in the design process of

systems on the one hand (such as the Systems Modeling Language, SysML) and buildings on the other focused on differing aspects.

The fundamental differences between these two perspectives are difficult to overcome, and implementing MBSE concepts in the construction sector is anything but simple. Yet, MBSE's emphasis on interactions, data flow, and time-sensitive behavior can prove useful for specialized construction subdomains. Cyber-Physical-Systems (CPS) aim to bridge the gap between the digital world of data and the physical world of manufacturing and installation. With the help of clearly defined interfaces between the data models and authoring tools on the one hand and fabrication/installation systems on the other, feedback loops can be created that optimize processes for Design for Manufacturing and Assembly (DfMA) and the propagation of changes for design automation. Modeling this system with all its necessary components, ranging from the general structure of the building to the fabrication machinery and various sensors, can prove to be complicated. While BIM aims to primarily describe what the model *is*, MBSE primarily describes how it *behaves*, and thus offers the tools necessary to model such a CPS. This paper attempts to develop a method of representing built structures such as bridges by means of an appropriate system modeling language.

2 Related Works

2.1 Model-Based Systems Engineering

Model-Based Systems Engineering is the practice of utilizing a central *system model* to design a system, as opposed to using a conventional *document-based* approach [2]. In detail, this means that the system model stores all necessary components and associations while employing paradigms such as *single-source-of-truth*. Contrary to the conventional approach, where system specifications are spread across various documents ranging from PDFs, spreadsheets, and diagrams, the system model manages all data in a single place, with human interface views (such as diagrams) being automatically generated based on the model. This way, changes to a single component can be immediately updated in all views, ensuring consistency and mitigating human error. Changes in one part of the system are immediately propagated to all other corresponding parts.

2.2 The Systems Modeling Language

While MBSE is the underlying concept, including necessary paradigms and best practices, it does not offer a concrete design method or provide the necessary tools to develop the said model. This is where modeling languages, such as SysML, fill the gap. SysML was developed from the ground up with MBSE in mind and employs those same paradigms while additionally clearly defining diagram types with which it is possible to model a system in a human-readable manner. Being a dialect of UML, it makes use of various diagram types to define the system architecture, such as the block definition diagram (*bdd*), requirements diagram (*req*), use case diagram (*uc*), and more. These diagram types are either reused or extended from the UML 2.0 specification [3]. In Figure 1, an example of a block definition diagram can be seen. It includes multiple elements already known from UML, such as blocks (classes), associations, generalizations, and compositions.

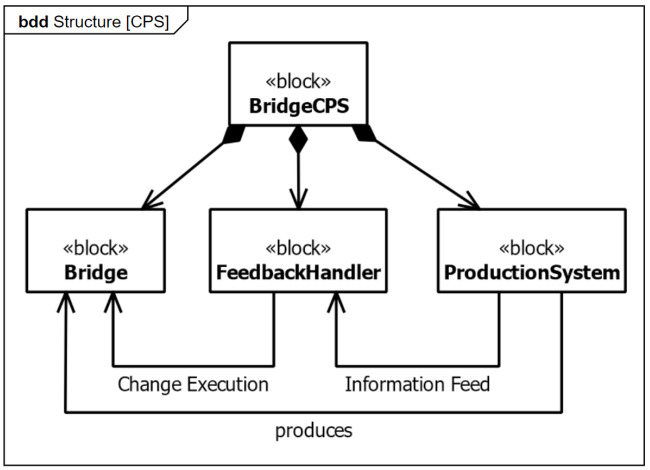


Figure 1: Example of a SysML Block Definition Diagram (*bdd*)

While SysML specifies an ontology as to what possible elements we can add to our diagrams, it does not directly create a system model. Since it is only a language, the underlying model creation is the job of the various pieces of software that implement the SysML specification and, based on the user-defined diagrams, build the model.

2.3 Cyber-Physical-Systems

Cyber-physical systems bridge the gap between the physical components of a system, such as machines, and the digital software that operates them. CPSs are tightly integrated and often embedded directly within physical components, such as sensors, but can also integrate more big-picture systems, such as directly relating planning and design tools to the physical hardware that executes upon that data. To stay in the Architecture, Engineering, and Construction (AEC) context, CPS can strongly link the planning and design software, such as BIM authoring tools, to the manufacturing and assembling machinery, thus creating a more robust data delivery between components and the possibility of intelligent feedback loops for iterative design improvements, based on real-world data [4]. To design these systems, a modeling method such as the earlier mentioned MBSE is suitable.

2.4 Specific Related Research Contributions

The adoption of MBSE in the AEC domain has been explored scientifically in numerous contexts. Many application areas make systems engineering an attractive way of describing and designing highly complex systems and designs that conglomerate different disciplines into one project. The AEC domain has many of these potential use cases, each differing from the next. A select few research articles will be presented below.

Nasserdine et al. explore the general transformative potential of adopting a model-centric approach in the AEC industry. Specifically, they establish concrete objectives and concepts that the industry needs to abide by to ensure a successful and advantageous transition, such as *single-source-of-truth*. The industry can achieve significant improvements by shifting to a model-centric approach, where digital models serve as the central repository of information throughout the project lifecycle. In Detail, these possible improvements include better collaboration possibilities among stakeholders, enhanced

and more nuanced decision-making based on integrated data, greater productivity, and the resulting cost and time savings. Numerous case studies and practical examples are provided to illustrate the successful implementation and positive outcomes of model-centric practices in real-world projects [5].

Philipp Geyer focuses on applying systems modeling to enhance the sustainability of building designs. He emphasizes the complex nature of sustainable building design, which requires balancing multiple criteria such as energy efficiency, environmental impact, and cost-effectiveness. Geyer proposes a systems modeling approach using SysML to evaluate and optimize building design performance holistically. By employing this approach, designers can better understand the interdependencies between different building components and make more informed decisions to achieve sustainability goals. Specifically, block definition diagrams (*bdd*), requirement diagrams (*req*), use case diagrams (*uc*), and activity diagrams (*act*) are created in the context of a residential building as a case study to explore the potential challenges and possibilities of systems engineering in the AEC domain poses. The study demonstrates the effectiveness of systems modeling through case studies, highlighting its potential to improve design outcomes and support the development of sustainable buildings [6].

O. Badreddin et al. introduce fSysML, an extension of the Systems Modeling Language (SysML) tailored specifically for modeling cyber-physical systems (CPS) to manage their complexity and ensure reliable performance robustly. The authors present fSysML as a foundational and executable version of SysML, designed to enhance the expressiveness and precision of models used in CPS development. fSysML incorporates formal semantics to enable the execution of models, facilitating early verification and validation of system behavior. This executable capability allows for simulation and testing of the interactions between the cyber and physical components of the system in a virtual environment. The paper outlines the architecture and features of fSysML, illustrating how it can be used to model various aspects of CPS, such as control systems, communication networks, and physical processes. Cast studies such as a monitoring system from the healthcare domain demonstrate the practical application and benefits of fSysML, highlighting its potential to improve the design, analysis, and implementation of complex cyber-physical systems [7].

2.5 Identified Research Gap

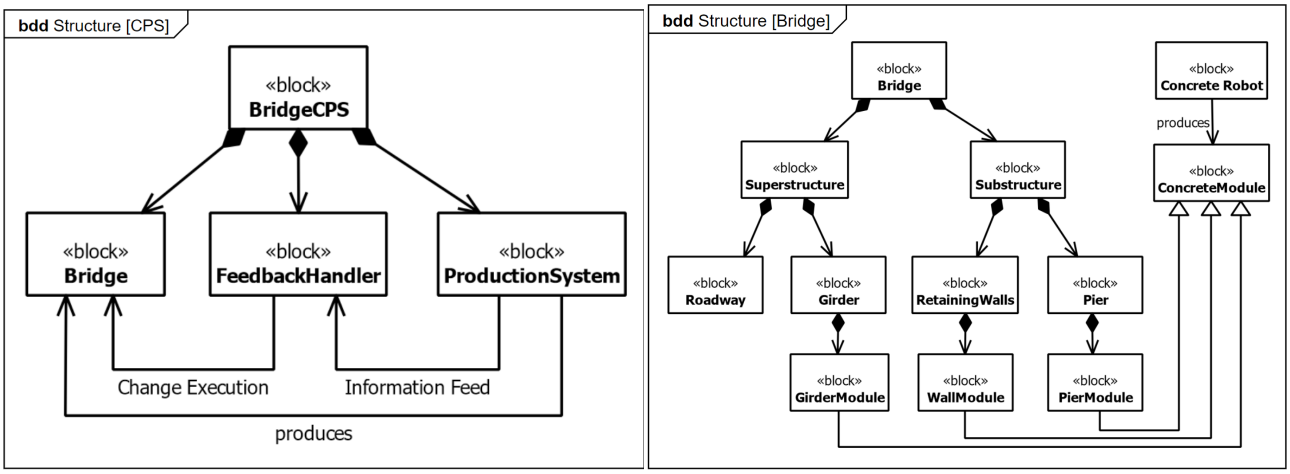
In general, it can be said that the transition to model-centric methods has numerous benefits. Yet, real-world attempts have been made to adopt methods such as MBSE into the AEC domain. While this is partially due to BIM filling that role, there are use cases where other established modeling methods and languages, such as SysML, can prove advantageous, specifically when confronted with situations with numerous in- and outputs and complex behavior. The SysML extension fSysML also introduces the idea of executable and simulatable system models. This paper attempts to model a basic bridge structure fabricated by the likes of a concrete printing robot with multiple interconnected levels of detail. A prototype model, including product structure representing the bridge, fabrication systems, and sensor groups, will be modeled using SysML to use MBSE principles. This example model will provide the necessary basis for answering the following research questions: (1) How can model structures of a bridge be combined with those of a fabrication system with the help of SysML? (2) Can the resulting model be used to perform model analysis?

3 Modeling a CPS through SysML diagrams

To make use of SysML as our modeling language, we must first analyze the possibilities and capabilities the language offers for our use case. Crucial to our bridge-CPS example are diagram types that concern (1) the necessary requirements, such as structural integrity, (2) the intended behavior, such as the production of modules, (3) the general structure of the bridge. With this in mind, sample diagrams of the requirements of the type and use case (*req*,*uc*), block definition (*bdd*), and activity (*act*) attempting to model the previously mentioned points will be created.

3.1 Structure

To begin, the general structure of a bridge and its corresponding CPS is modeled. SysML allows for multiple diagrams of the same type to exist in parallel, such as the block definition diagram. This is convenient when wanting to adequately separate different aspects of the same system. Nonetheless, components can be a part of multiple diagrams, even though the underlying unit stays the same. This allows the modeling of associations between different parts of the system without defining unnecessary details in a certain diagram view - these details can be described in a different diagram. To explain this further, let us have a look at a high-level basic representation of our bridge CPS:



(a) (b)

Figure 2: Two examples of a block-definition-diagram. (a) shows the high-level structure of the CPS through *blocks*((rectangles), *compositions* (black diamond) and *associations* (arrows). (b) details the structure of the bridge component in more detail through more granular composition, as well as *generalization* (white arrow).

Figure 2 shows how multiple *bdd*-diagrams can be used to granularly define different aspects of the system. Meanwhile, the `bridge`-block in (a) shares the same object with the `bridge`-block in (b) (as in: a change in that particular block will propagate to both diagrams). Separating aspects into different diagram views is a suitable way to illustrate the system while still maintaining necessary associations by storing them in the underlying model.

3.2 Requirements

The requirements of the bridge can either be defined at the beginning of the project lifetime and extended later on or derived from the modeled behavior. SysML also allows for the deriving and refining of requirements, as well as their satisfaction through other components. Many other system model components can satisfy a particular requirement, such as a block, an activity, or others. These requirements, defined by plain text and an id, are also part of the resulting model. Further detailing of the requirements can be achieved through *refining*, *tracing*, and *verification*, all of which are part of the SysML specification.

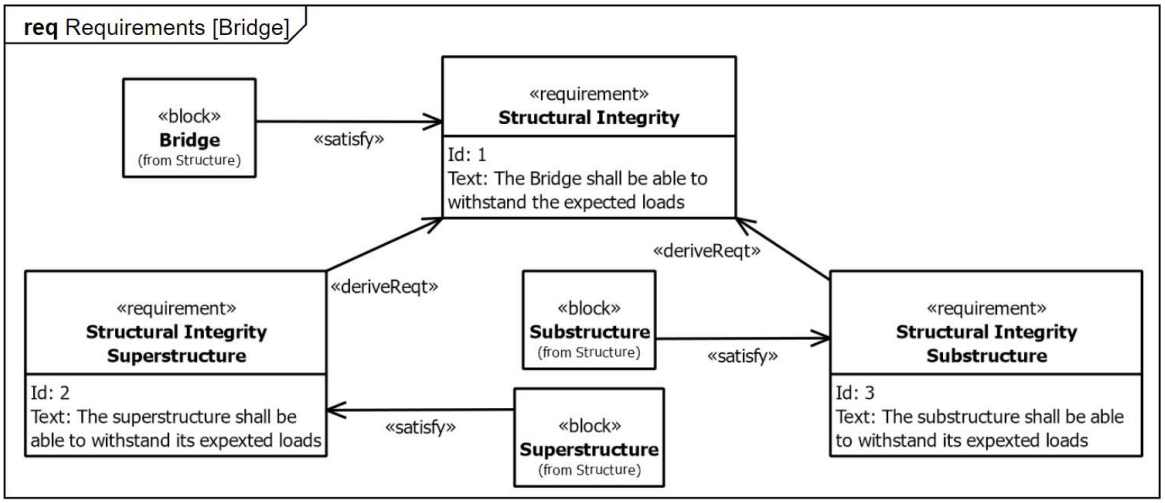


Figure 3: Example requirements of the bridge. Each requirement is currently satisfied by a respective block from the systems *structure*.

4 Model Testing

As several diagrams have now been created, the underlying model is also defined. Most SysML software provides the user with the tools to create the diagrams from which the modeling software derives the model. This is the case with the modeling software *Gaphor*, an open-source UML and SysML modeling tool. The graphical application used to model the diagrams also includes a Python library with which the objects within the model can be accessed [8]. *Gaphor* saves its models using an XML file, which can then be parsed by using the Python library. This allows the creation of a testing framework with which certain conditions can immediately be assessed. A prototypical implementation was developed to prove the concept of a simple system-level analysis of the model, ensuring model consistency and compliance across its development. Two examples of this would be (1) checking for requirement satisfaction and (2) testing the semantic embedding of new components. The first example is simple to implement and checks if all requirements inside the model are satisfied, i.e., have a 'satisfies' relationship with another component. In the following *req* diagram example in Figure 4, the test script will throw an error since not all requirements are satisfied.

Furthermore, a check for correct semantics can be implemented by automatically testing if new blocks that inherit from the block ConcreteModule are part of the bridge assembly - by traversing the composition associations until we arrive at the `bridge` block. This way, modeling consistency can be

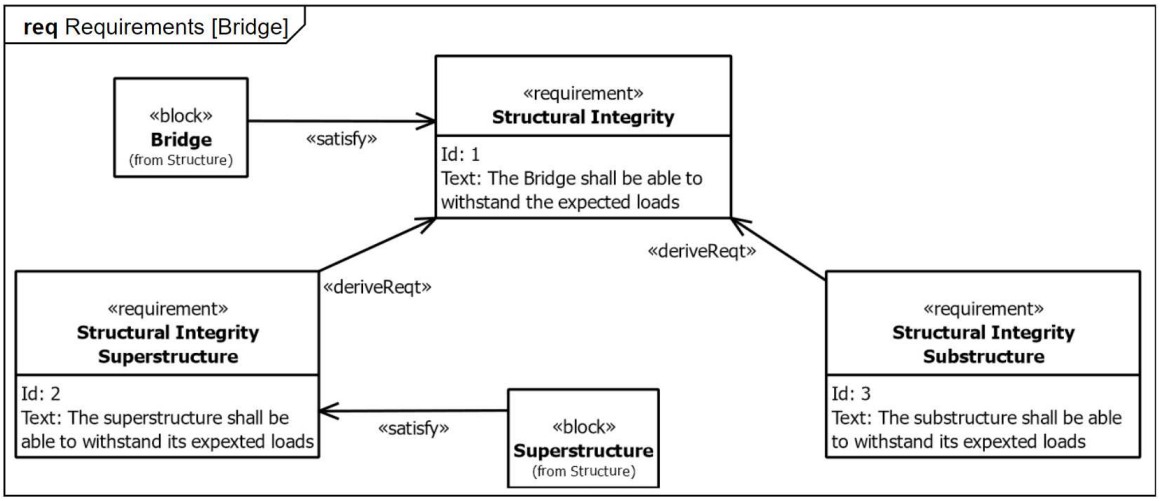


Figure 4: Example *req* diagram with one missing satisfaction relationship. In this case, the test would warn the user that not all requirements are satisfied. This can ensure model compliance

assured as components of a certain type must belong to a certain assembly. An example of this can be seen in Figure 2 in the (b) subfigure: If either `WallModule`, `GirderModule`, or `PierModule` were not part of the `bridges` composition, the text would warn the user.

The testing was scripted and executed using the *pytest* framework while using the *Gaphor* python library to extract and represent the model. One interesting point to make about this setup concerns the Object Modeling Group's (OMG) Meta Object Facility (MOF): Languages such as UML and SysML are defined on the M2 level and used on the M1 level. Concrete instances, on the other hand, are situated within the M0 level [9]:

Table 1: The Object Modeling Group's (OMG) Meta Object Facility (MOF) Hierarchy. [9]

M3 (MOF)	Defines a language for specifying a metamodel.
M2 (SysML)	Defines a language for specifying models.
M1 (User Models)	Defines a language That describes semantic domains.
M0 (Instance Models)	Contains run-time instances of the model.

The *pytest* framework exclusively tests on the M0 level, as it can only assert values or states during the runtime of the script or program. However, a workaround is necessary since we want to check for compliance on the M1 level (Where the SysML diagram exists). The *Gaphor* python library's solution is to map SysML objects to instantiated python objects when the model is loaded. An example of this would the SysML's *block*, which would be the equivalent to a UML *class*. When a model containing a *block* is loaded using the library, an object of type `SysML.Block` is instantiated. This kind of abstraction is a solution to make model testing with pre-existing frameworks simple and convenient.

5 Conclusion and Final Remarks

Adapting tools for systems engineering into the AEC domain is a challenging task, and this paper has only scratched the surface as to what is possible. SysML includes 9 different diagram types to model various aspects of a system [3], of which only 2 have been discussed here. Analyzing and dissecting these diagrams to find out which are necessary and which are not already a challenge in 'classical' systems engineering proves even greater of a challenge when adopting this method to a somewhat foreign domain. Yet, the use case of cyber-physical systems looks to be a promising starting point for the adoption of the method. Furthermore, existing software, including powerful libraries or APIs such as *Gaphor*, can prove to be useful for further automation of not only the design of the system itself but also for executing and simulating said system. One such example could be the previously mentioned feedback loops that can automatically propagate changes in one component to others, expressed through the system's behavior. Specifically, more research needs to be conducted on this kind of use case.

Acknowledgements

We gratefully acknowledge the financial support of the Deutsche Forschungsgemeinschaft (DFG) as part of the program SPP 2187 "Adaptive modularized constructions made in flux" (Project 423969184).

References

[1] A. Borrmann, M. König, C. Koch, and J. Beetz, *Building Information Modeling - Technologische Grundlagen und industrielle Praxis*, 2nd ed. Springer Vieweg, 2021.

[2] L. Delligatti, *SysML distilled: A brief guide to the systems modeling language*. Addison-Wesley, 2013.

[3] *Sysml faq: What is sysml?* [Online]. Available: https://sysml.org/sysml-faq/what-is-sysml.html.

[4] M. Broy, *Cyber-physical systems: Innovation durch softwareintensive eingebettete Systeme*. Springer, 2011.

[5] H. Nassereddine, M. E. Jazzar, and M. Piskernik, "Transforming the aec industry: A model-centric approach", Eigenverlag, 2020, pp. 13–18. DOI: 10.3311/CCC2020-076. [Online]. Available: https://repositum.tuwien.at/handle/20.500.12708/62873.

[6] P. Geyer, "Systems modelling for sustainable building design", *Advanced Engineering Informatics*, vol. 26, pp. 656–668, 4 Oct. 2012. DOI: 10.1016/J.AEI.2012.04.005.

[7] O. Badreddin, V. Abdelzad, T. Lethbridge, and M. Elaasar, "Fsysml: Foundational executable sysml for cyber-physical system modeling", 2016.

[8] *Gaphor: About*. [Online]. Available: https://gaphor.org/#about.

[9] O. 2. E. Team, "Meta-modeling and the omg meta object facility (mof)", 2017. [Online]. Available: https://www.omg.org/ocup-2/documents/Meta-ModelingAndtheMOF.pdf.