[KKN+24] H. Kausch, K. Koppes, L. Netz, P. O'Brien, M. Pfeiffer,
D. Raco, M. Radny, A. Rath, R. Richstein, B. Rumpe:
Applied Model-Based Co-Development for Zero-Emission Flight Systems Based on SysML.
In: Deutscher Luft- und Raumfahrtkongress (DLRK 2024),
Deutsche Gesellschaft für Luft- und Raumfahrt - Lilienthal-Oberth e.V., Oct. 2024.

# APPLIED MODEL-BASED CO-DEVELOPMENT FOR ZERO-EMISSION FLIGHT SYSTEMS BASED ON SYSML

Hendrik Kausch[1], Kristiaan Koppes[2], Lukas Netz[1], Patrick O'Brien[2], Mathias Pfeiffer[1], Deni Raco[1], Matthias Radny[2], Amelie Rath[1], Rebecca Richstein[2], Bernhard Rumpe[1]

[1]RWTH Aachen University, Chair of Software Engineering, Aachen, Germany.
[2]Airbus Aerostructures GmbH, Hamburg, Germany.

**Abstract**

In context of the BMWK LuFo Project "APPLIED MODEL-BASED CO-DEVELOPMENT FOR ZERO-EMISSION FLIGHT SYSTEMS", a model-based methodology applied in avionics is presented. SysML is used as modeling language and for automatic model analysis. A case study on a data link uplink feed system demonstrates our model-based approach for specifying and verifying safety and security properties. The key feature of the underlying formal dataflow theory compared to competitors is that refinement is compositional.

**Keywords:** Trustworthy AI, Co-Development, Model-Driven

## 1 INTRODUCTION

The AMoBaCoD project ("Applied Model-based Co-Development for Zero-Emission Flight Systems") works towards finding an approach for co-development in engineering using model-based methods. The goal is to collaboratively develop a zero emission demonstrator aircraft and test the applicability of the model-based systems engineering (MBSE) approach. Modern aerospace systems combine both software parts and hardware parts. Their specification was typically done separately, posing challenges for their integration related to compatibility, safety, and security. Co-development [1], sometimes referred to as co-design or integrated design aims to remedy this problem. Typically, informal system requirements are developed iteratively over many granularity layers towards a design, e.g., mechanical structures. In co-development, a system design is derived using generic components, i.e., components that are neither assigned to hardware nor software. The generic components are then assigned to software and hardware. Model-driven systems engineering (MDSE) is an enabler for co-development, as the specification of systems requires specification techniques and languages that are hard- and software-agnostic. An approach where refinement is compositional is highly desirable, i.e., when parts are refined separately, then the reassembled system should be a correct refinement of the original system.

However, the introduction of MBSE poses challenges. Especially for complex systems, such as modern airliners, the application of the MBSE method represents a major initial entry challenge. Airbus employs SysML-based modeling solutions from Dassault Systèmes. But creating the necessary and extensive system models with SysML is very time-consuming, especially at the beginning. Potential long-term advantages must thus outweigh this additional effort. Artificial intelligence (AI) systems promise interesting new capabilities with regard to system specification. If the initial modeling workload can be reduced through AI, the probability of making MBSE accessible for complex overall aircraft systems will increase.

Integration and use of AI systems in Aerospace Engineering, structural analysis and manufacturing has the potential to disrupt existing processes in unforeseen ways. However, the safety-driven nature of this domain demands that AI systems meet the highest standards of trustworthiness, explainability, and traceability to ensure safety, robustness, and regulatory compliance. Trustworthiness is paramount in the development of flying structures, as AI models must handle complex data, coming from numerous



doi: 10.25967/630489

sources. Moreover, the ability to trace data origins is crucial for justification. Explainability is critical for ensuring that AI-driven decisions are transparent and justifiable. In Aerospace Engineering, where safety is crucial, engineers must be able to trace AI decisions back to specific data inputs and model processes. Besides gaining regulatory approval, there is the aspect of public trust in AI systems, which includes the demonstrated ability to meet industry standards and safety protocols. In the case of large language model (LLM), traceability can be obfuscated by the trained model. This leads to challenges in keeping the digital thread whole.

Simulations and tests are critical for both calculating and validating designs during the development of mechanical structures. These processes must follow a clear, justifiable chain of reasoning to meet the stringent standards set by aeronautical authorities, which are predominantly safety-driven. For instance, when new structures are subjected to various critical load scenario calculations, the resulting reserve factors for each structure must be carefully determined. Each step in the process has to be justified to ensure compliance. Fully automated solutions offer advantages, but only if each intermediate result can be transparently shown and explained, ensuring every outcome adheres to the safety and regulatory standards. Unlike more traditional machine learning approaches, modern AI systems, such as those used for structural simulations and optimization, often operate as black boxes, obscuring the logic behind their decisions. In structural analysis, explainability becomes crucial for engineers who need to justify AI-generated predictions or decisions regarding stress distribution, material deformation, or load capacity. For instance, if an AI recommends design changes or alterations in manufacturing processes to optimize the performance of an aircraft component, engineers must be able to explain the reasoning behind the AI's recommendations clearly.

Not only safety, but also security needs to be designed into avionics to demonstrate not just the initial airworthiness, but also the maintenance of continuous airworthiness and protection from unauthorized interaction. Seven levels of security trustworthiness (Evaluation Assurance Level) [2] are used for system classification with respect to criticality. A high criticality raises the demands on the depth to which the manufacturer must describe and test their product. The highest level (so called level 7/7+) requires formal correctness arguments and is used in extremely high-risk situations, in particular when the high value of the goods justifies the higher costs. Formal model analysis has been applied successfully in other domains such as automotive [3, 4] or avionics [5].

The AMoBaCoD project is driven by the question "How do we want to use MBSE for future aircraft development?" and will investigate various steps in the product development of aircraft systems, such as the creation and management of requirements, the development and evaluation of solution concepts, and the detailed design of products and manufacturing systems using collaborative MBSE methods.

### Contribution

AMoBaCoD uses model-driven engineering to counter increasingly complex systems and enable co-development. To more efficiently create models and aid engineers, an AI model creator (chatbot) is presented. Because the aerospace industry has particularly stringent requirements on safety and security, the need for a trustworthy AI is established. Model analysis, specifically formal verification, is introduced as a means to check model correctness. To analyze the models, a precise, mathematical meaning (semantics) is required. A suitable formalism is established. A case study on a data link uplink feed system demonstrates our model-based approach for specifying and verifying safety and security properties. A formal verification toolchain is leveraged to process and check models for correctness. The key feature of the underlying formal dataflow theory compared to competitors is that refinement is compositional.

## 2 CO-DEVELOPMENT

Co-development of distributed systems [1, 6, 7], i.e., systems consisting of multiple subsystems, is achieved by first specifying generic, i.e., neither hardware nor software, components. The act of specifying these components is called co-specification and entails specifying both their interface and their behavior. These generic components can be further decomposed to build an architecture. In the partitioning phase, the atomic, i.e., not further decomposed, components are finally allocated to hardware or software. Allocations thus mark the mode of implementation. Co-development presents challenges such as finding a suitable specification mechanism as well as the integration of co-developed system parts.

Specification mechanisms need to accurately represent, i.e., create a model of, system requirements and system designs. They also need to be able to establish traceability, i.e., provide means to connect design decisions with requirements and vice versa. Schobbens et. al. [1] propose the use of algebraic notation to model both hardware (VHDL programs) and software (C code). This idea is not new as formalisms such as Communicating Sequential Processes (CSP) ([8, 9], as used in e.g. [10]),

Calculus of Communicating Systems (CCS) [11], $\pi$-calculus [12], Ptolemy [13], Temporal Logic of Actions (TLA) [14], Petri Nets [15] or FOCUS [16, 17] all follow this same principle. Such mathematical underpinnings are favorable because they support important aspects of system development, such as non-determinism/underspecification, a notion of behavioral refinement, time-sensitive specifications, and hierarchical decomposition. Decomposition is especially useful to facilitate the specification of complex systems. Integration of parts is a major challenge for distributed system design. To be able to efficiently verify the correctness of such decomposed systems, a compositional[1] verification, i.e., co-verification, is highly desirable.

# 3 TRUSTWORTHY AI

Trustworthy AI is a term addressing the data privacy, explainability, accountability, and robustness of AI and its products and services [19, 20]. All data submitted to a trustworthy AI, be it during training or use, remains private. Products and decisions of the AI are analyzable and it is possible to disclose and justify rationale. This includes the ability to explain outcomes to the user when/directly after they are produced, but also all other stakeholders, e.g. regulators, at a later date. Inevitable erroneous outputs are also explainable and there exist ways of correcting them. Harsh working conditions, such as erroneous or even malicious input, do not lead to unforeseen consequences. Data privacy can be achieved. Fallback to non-AI tools and services like human experts provides robustness. Following this definition, a trustworthy AI is similar to a trustworthy colleague. One would expect their colleague not to tell secrets (data privacy), be able to justify their work (explainability), be accountable for made mistakes (accountability), and be robust to stressful situations (robustness). Failing to meet any of the criteria would render a person untrustworthy.

## Model-Driven Approach

As size and complexity of manufacturing systems are continually growing, building, updating, and maintaining them becomes increasingly challenging. To manage complex systems effectively, appropriate mechanisms such as divide-and-conquer or suitable modeling techniques are essential. Models relate to an original entity that they represent. But, models do not capture every attribute of the original. Models abstract certain aspects as needed for specific use cases. Models thus serve as substitutes under specific conditions. Abstraction to focus on key aspects is what enables modeling to be such an effective tool against complexity. Model quality has also been shown to have an impact on product quality [21, 22].

To design and develop large manufacturing systems, we propose the use of model-driven techniques whereby requirements and designs are expressed using models. The de-facto standard modeling language for systems engineering is SysML [23]. We propose the use of the newest version, SysML v2, because SysML v1 is no longer actively developed[2].

## Conversational AI for Model Creation

While modeling provides many benefits, manual model creation requires expert knowledge of the modeling notation. A SysML v2 model creator (chatbot) was thus developed to aid engineers in the modeling process. The conversational AI is able to create, modify, and analyze SysML v2 textual models.

The concept of utilizing natural language for generating code or models has been investigated in several studies, highlighting its potential in bridging the gap between domain experts and developers [24–28]. We used a limited amount of examples to teach OpenAI's ChatGPT4 LLM the rules for SysML v2. A refined Few-Shot Learning Approach [29] was providing the LLM with a fixed set of example conversations of users asking for models or explanation thereof. Further instructions were added to improve the reliability of the output [30]. A prototype of the trained chatbot is openly available[3]. Next to OpenAIs LLMs, open source models can be constrained to only produce output in a given modeling language. The approach described in [31] is capable of reliably producing models for a given DSL while running on local hardware.

## Towards Accountability

AI-generated models are generally not trustworthy. To achieve trustworthiness, the chatbot's accountability needs to be tackled. This can be done by verifying the output, i.e., the models, using formal model analysis.

The aerospace industry has successfully adopted formal methods [32] for verifying properties at the code level, using model checkers and abstract interpretation, e.g. for worst-case execution time analysis [32, 33], which is also regulated by EUROCAE ED-216, or for replacing testing efforts for certain properties [32]. Testing as a method for verifying system correctness can only address a finite number of cases. Systems with numerous components and states, especially those operating over long periods, can easily surpass the capabilities of exhaustive testing. Formal methods offer a solution to this limitation.

[1] Carnab introduced compositionality as *Frege's principle* [18, p. 120-121].

[2] https://www.omgsysml.org/news-articles.htm

[3] https://chatgpt.com/g/g-3DkGCDlNM-sysml-v2-model-creator

Effective use of formal methods requires models to have precise, mathematical meaning (semantics). But model notation, i.e., syntax like SysML v2, has no inherent semantics. We thus propose to use FOCUS because, compared to its competitors, FOCUS has the property that refinement is fully compositional [17, 34]. This means that if a system design has been decomposed and the parts were refined separately, then the reassembled refined parts are by design a correct refinement of the original system design. FOCUS is widespread in industry and has been used in a range of consortia projects for the development of distributed cyber-physical systems (BMBF projects [7, 35, 36][4]).

## An Accountable Toolchain

While the chatbot can create SysML models based on given informal requirements, it currently cannot distinguish correct from incorrect models. This means that it may classify correct models as incorrect (false negative) or classify incorrect models as correct (false positive). The chatbot thus cannot be held accountable for its products.

We propose the use of an automatic model analyzer (the MontiBelle framework [34]) as an oracle for model correctness. For a model to be correct, it first has to be valid with regard to the modeling notation, i.e., the syntax. We call this syntactical correctness.

To better grasp the difference between syntax and semantics, consider the following example. The English sentence "Wingspan must below 5m" is syntactically incorrect, as the verb "to be" is needed. Similarly, the use of incorrect keywords in SysML renders models syntactically incorrect. Compared to syntax, the meaning of something depends on interpretation, i.e., semantics. We call this semantical correctness. The English sentence "Wingspan must be below 3m but above 5m", while syntactically correct, is semantically incorrect. Examples for semantical properties are consistency and redundancy. Consistency of requirements is defined as the ability to provide a compliant realizable system design. An example of an inconsistent set of requirements is "wingspan $<$ 3m" and "wingspan $>$ 5m" as there is no wing design fulfilling both requirements.

Figure 1 shows the proposed toolchain: Humans input informal requirements into the chatbot. The chatbot produces SysML v2 textual models. Refinement links between components of different granularity levels and decomposition relations between components of the same granularity level establish traceability from system designs to their origin requirements. For traceability and model management (navigation, etc.), we created an Formal Integrated Development Environment (F-IDE). The models are loaded into the F-IDE.

The F-IDE automatically encodes the models into theorem prover syntax [4, 37]. These theorems include system encodings and theorems with outstanding proofs, so called proof obligations. The proof obligations result from traceability links. The F-IDE automates proof finding for these proof obligations. At the same time as proof finding, counterexample finders try to falsify the claims. The result is either a formal proof, a counterexample, or neither (timeout). In the latter case, no statement on correctness can be made, neither positive nor negative. A theorem prover expert may be able to manually help along the proof finder. Testers could write additional tests in hopes of demonstrating the incorrectness of claims. In case a proof was found, the models produced by the chatbot are found to be correct and are safe to use for further development. In case of counterexamples, these examples can be turned into tests. While the design models are reworked, these tests should be run regularly. Once no more tests fail, the verification can be re-attempted.

The (potentially repeated) application of this toolchain results in verified system design models. As no partition needs to be done, this approach is inherently compatible with co-development. Models produced by the chatbot whose correctness is disproven by counterexamples need to be reworked or discarded.

## Aerospace Case Study

An MDSE generative-AI approach can for example be used in the development of avionics systems [38]. The development of an avionics data link is used as an example [39–41]. This Data Link Uplink Feed (DLUF) system is representative for safety-critical, time-sensitive systems in the avionics domain and will be used to demonstrate the capabilities of our approach to trustworthy AI. The DLUF system should enable components using a wireless connection (e.g. between an Unmanned Aerial Vehicle (UAV) and its ground station) to transfer prioritized data packets. Co-development is required as the DLUF system consists of hardware, i.e., the UAV, and software, i.e., prioritization protocol.

One of the 18 total requirements, the non-starvation (or liveness) property, required the use of formal methods. It was derived a.o. from a security requirement, namely the need to prevent a denial-of-service attack by an overflow of low-priority messages. This property could not be tested for, as it required checking an unknown and potentially infinitely long time frame. It had to hold for the overall system and could not be sufficiently verified by only checking properties of the system's parts, but

[4] https://spesml.github.io

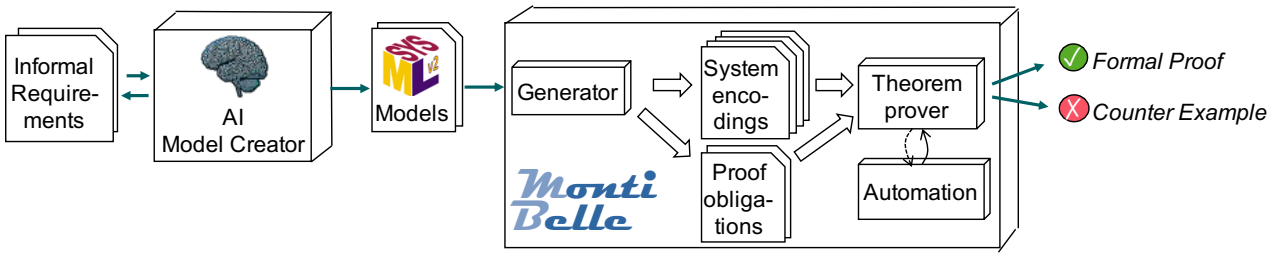


**Figure 1** Proposed toolchain combining AI model creator (chatbot) with automatic model analyzer as an accountability gate.

required the integration of all artifacts into a single coherent claim.

DLUF could previously be modeled in two ways. Either in SysML v2 directly or in Cameo Systems Modeler as fig. 2 shows. From Cameo, the model could then be exported to SysML v2 textual notation through a custom plugin.

First, the chatbot is fed the informal non-starvation system requirement. The chatbot's response begins with a minimal explanation as to the output's structure. It explains that it first defines a system context (i.e., a box with inputs and outputs). The non-starvation is then enforced using constraints. Next, it begins listing a complete SysML model. Figure 3 shows the chatbot in action after the initial user query demanding a DLUF system with provision of prioritization and non-starvation. The chatbot is then fed the remaining 17 informal system requirements. The chatbot is able to update its previously created model, i.e., it keeps track of context.

When fed information from external analysis, such as the error report of a syntax checker, the chatbot can correct syntactical errors. The created model included a transition which wrongfully started not with a state but rather an action. We told the chatbot that in the specific transition, the keyword $first$ should be followed by a state. The chatbot responded with a thank you, followed by a short summary of our request. It then concluded with an updated SysML model where exactly the mentioned error was fixed. Similarly, the chatbot can improve models if the design doesn't satisfy the system requirements.

The chatbot can answer questions regarding the task at hand, e.g., questions about the model, and general questions, e.g., questions about SysML. When asked what a packet type with star-cardinality ($Packet[*]$) meant within the model, it explained the respective concepts from the SysML language, i.e., types and cardinalities (multiplicities). It gave examples on how to use them and summarized how $Packet[*]$ was used in the SysML model previously created by the chatbot. It also compared the star-cardinality with other cardinalities ($[1]$, $[0..1]$, $[1..*]$). When more generally asked what a transition was, it seemed to summarize the documentation, listed key features of transitions, gave examples, mentioned the visual representation, and summarized its answer.

Next, the model's refinement chain can be automatically verified. The final, most granular design is shown to be compatible with the first, most abstract requirement. For this, models created by the chatbot are loaded into and then verified using the F-IDE.

All resulting proof obligations [22, 39] were verified, i.e., all proof obligations were met using the F-IDE (indicated by green light bulbs). Thus, the models are correctly refined from highest-level requirement to most granular decomposed design.

### Tool-Qualification

The proposed approach aims to reduce the testing effort by incorporating formal verification. Consequently, this necessitates the qualification of the tool, e.g., in accordance with RTCA DO-330/EUROCAE ED-215 standards.

Isabelle is used to prove the functional correctness of DLUF and must therefore be qualified. The EASA and the theorem proving communities published in [42] and particularly [43] how Isabelle proofs could be certified. In conclusion, Isabelle is one of the easiest tools to certify because of its axiomatic structure.

## 4 DISCUSSION AND LIMITATIONS

The EASA classifies AI tools in three categories, ranging from the human being augmented by the AI but firmly in charge (level 1), over a near fifty-fifty split of control (level 2), up to an AI that is under the oversight of a human, but otherwise independently operable (level 3) [44]. While AI advances in recent years might lead to dream scenarios about advanced automation, the presented approach is merely a level 1 AI. The human remains in control and actively queries the chatbot for help. The presented conversational AI is currently in a prototype stage and could be improved at multiple stages. A RAG approach could be added [45] in order to enable referencing and incorporating already existing models into the modeling process.

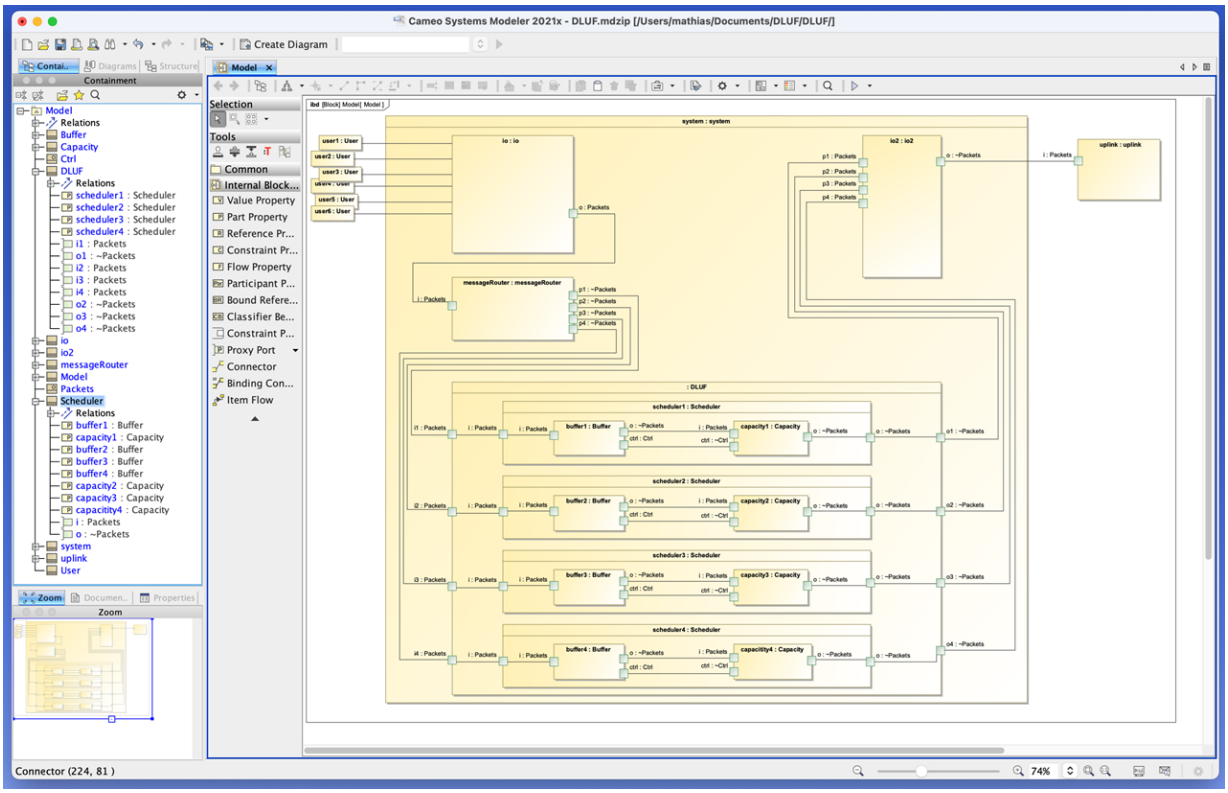


**Figure 2** Screenshot of the DLUF system in Dassault Systemes Cameo Systems Modeler.

Furthermore, the approach presented here does not cover all phases of system development. Other validation, most likely in the form of human review, must take place to ensure the correct formalization of informal system requirements. This is an inherent limitation of (current) computer aided requirement management as machines are (not yet) capable of interpreting human thought. Similarly, while a machine is able to determine the consistency of formalized requirements, it is unable to determine the completeness. Human review of both the informal and formal requirements and their translation is necessary and cannot currently be replaced by AI tools.

Finally, the development of system design down to a hardware level could potentially be automated, e.g., using formalizations of hardware, but is currently typically a process that involves humans.

# 5 CONCLUSION

In the context of the AMoBaCoD project, MBSE applied in aerospace was evaluated. Model-driven engineering counters the increasing complexity in aerospace systems and enables co-development. Adoption cost of MBSE in aerospace as a challenge was identified and AI chatbot for model creation was proposed. If the initial modeling workload can be reduced through AI, the probability of making MBSE accessible for complex overall aircraft systems will increase.

Use of AI in model creation requires analyzable models and thus clear model semantics for trustworthiness, particularly because of stringent requirements on safety and security in the aerospace domain. A formal dataflow theory was introduced whose key feature compared to competitors is the compositionality of refinement. A formal verification toolchain is proposed that formally verifies model correctness. Model analysis can increase the applicability for the aerospace industry and thus provides added value.

The applicability to the aerospace domain is demonstrated using an UAV case study on a DLUF system. Security properties can be specified at a higher-level, and the system design can be formally checked to respect these properties.

## Outlook

An AI-supported design process could be further evaluated using relevant case studies, especially considering current trends towards fast-paced development to take on challenges such as emissions and zero-E aviation. Feeding an AI the informal system requirements for a new aircraft has potential to significantly speed up the initial design process. Models could be created rapidly (rapid prototyping). Verification results could immediately be fed back to the

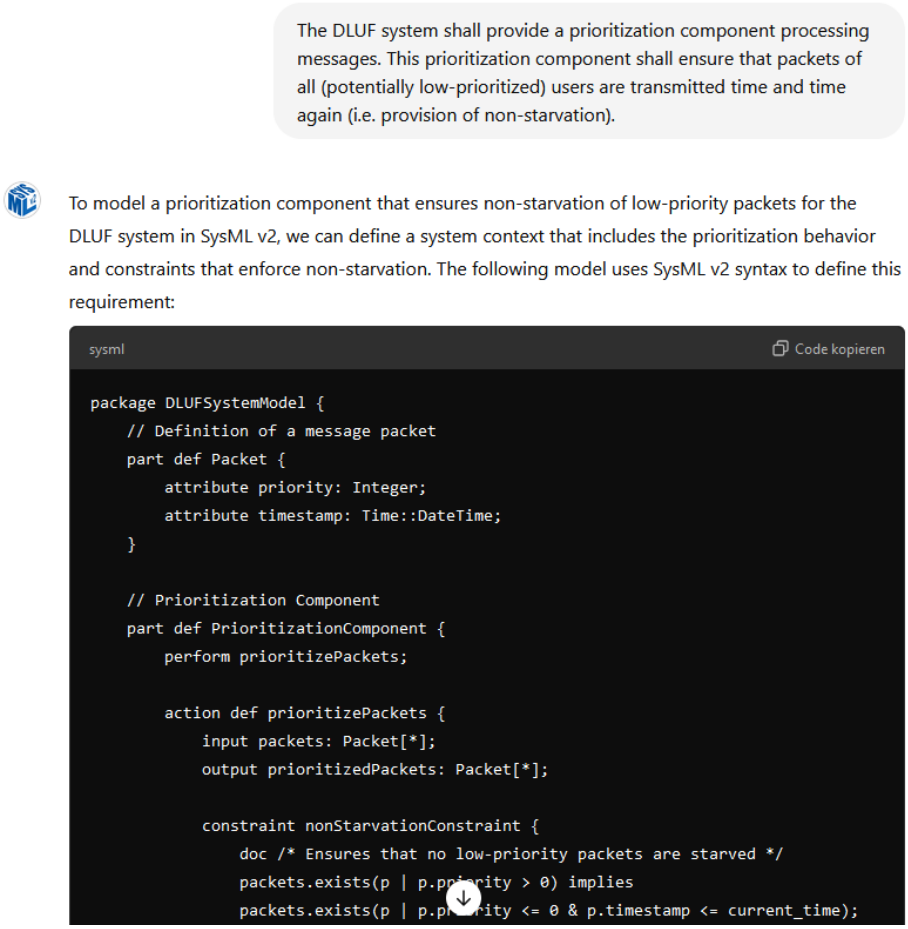


**Figure 3** Screenshot of the chatbot creating a SysML v2 textual model for the non-starvation property.

AI-assistant. The design process could be fast-paced and more agile, as the verification provides accountability. The capabilities of LLMs to summarize and explain could also be leveraged. Feeding the LLM with aerospace- or even project-specific information such as a glossary could streamline workflows and minimize communication breakdowns. Key challenges include regulatory compliance, managing unintended behavior, and optimal performance of AI systems. Despite the potential in the industry, significant challenges remain. One prominent issue is the lack of established frameworks for AI compliance. EASA's road map outlines standards for the Aerospace sector, with guidelines regarding a high level of transparency in AI-driven decisions, especially as they begin to issue new guidelines for Level 3 AI systems, which involve advanced automation, expected by 2025 but final standards may not be in place until 2028, leaving a gap in the regulatory framework for AI-driven design and manufacturing systems.

Modeling tools for SysML v1 such as MagicDraw/-Cameo Systems Modeler[5] or Catia No Magic[6] could be integrated into the toolchain because modeling tools for SysML v2 are not yet established. Previously, SysML v1 graphical models could be created and then exported to SysML v2 textual. DLUF for example was previously modeled in Cameo (fig. 2) and then exported to SysML v2 through a custom plugin. The ability to directly create SysML v1 (graphical) models using AI tools such as chatbots could be a promising next step. Their underlying data structure is often a derivative of XML, which is verbose and thus error prone. The error proneness of generating large XML files will make it necessary for the chatbot to be trained to create such XML files. Instead, building bridges between SysML v1 and v2 seems promising. It is feasible to translate between the two languages. One could treat SysML v2 models as ground truth, while SysML v1 models serve for human interaction and communication. Existing SysML v1 models could also be modernized by translating them to SysML v2.

Finally the efficiency gains of our proposed trustworthy AI assistant need to be quantitatively measured. Using an established aerospace case study, two teams could separately be given the informal system requirements and be tasked with

[5] https://www.3ds.com/products/catia/no-magic/cameo-systems-modeler
[6] https://www.3ds.com/products/catia/no-magic

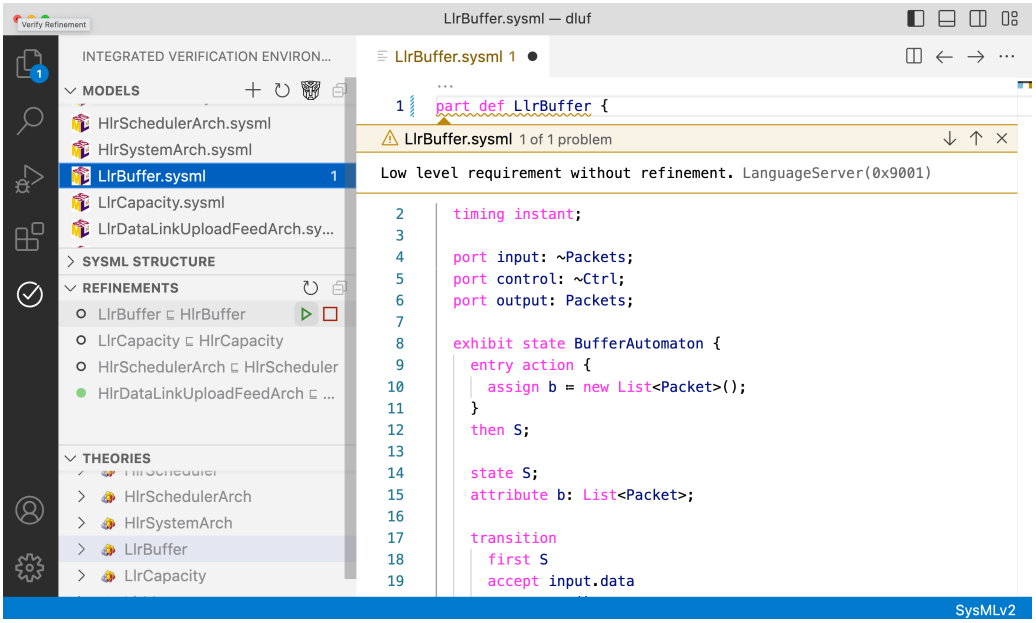


**Figure 4** Formal-IDE (F-IDE). The left-hand side shows from top to bottom: the list of models, proof obligations, as well as theorem prover encodings. The right-hand side shows the model editor with a warning displayed as a result of hovering the mouse cursor over a warning marked with squiggly lines under the model text.

developing and verifying a realizable system design. Correctness of models as well as development time could be measured to gauge time and cost savings for system engineers.

**Funding**: German Federal Ministry for Economic Affairs and Climate Action, AMoBaCoD-Project (Grant No. 20X2201C).

**Corresponding Author**: raco@se-rwth.de

## References

[1] M. Aiguier, S. Béroff, P.Y. Schobbens, An algebraic approach for codesign, in *Theoretical Aspects of Computing - ICTAC 2004*, ed. by Z. Liu, K. Araki (Springer Berlin Heidelberg, Berlin, Heidelberg, 2005), pp. 415–430

[2] Common Criteria Management Board (CCMB), *Common Criteria for Information Technology Security Evaluation, Part 3: Security Assurance Components* (2012). URL https://www.commoncriteriaportal.org/. CCMB-2012-09-003

[3] S. Kriebel, D. Raco, B. Rumpe, S. Stüber, Model-Based Engineering for Avionics: Will Specification and Formal Verification e.g. Based on Broy's Streams Become Feasible?, in *Proceedings of the Workshops of the Software Engineering Conference. Workshop on Avionics Systems and Software Engineering (AvioSE'19)*, *CEUR Workshop Proceedings*, vol. 2308, ed. by S. Krusche, K. Schneider, M. Kuhrmann, R. Heinrich, R. Jung, M. Konersmann, E. Schmieders, S. Helke, I. Schaefer, A. Vogelsang, B. Annighöfer, A. Schweiger, M. Reich, A. van Hoorn (CEUR Workshop Proceedings, 2019), pp. 87–94

[4] J.C. Bürger, H. Kausch, D. Raco, J.O. Ringert, B. Rumpe, S. Stüber, M. Wiartalla, *Towards an Isabelle Theory for distributed, interactive systems - the untimed case.* Aachener Informatik Berichte, Software Engineering, Band 45 (Shaker Verlag, 2020)

[5] H. Kausch, J. Michael, M. Pfeiffer, D. Raco, B. Rumpe, A. Schweiger, Model-Based Development and Logical AI for Secure and Safe Avionics Systems: A Verification Framework for SysML Behavior Specifications, in *Aerospace Europe Conference 2021 (AEC 2021)* (Council of European Aerospace Societies (CEAS), 2021)

[6] G.N. Khan, A. Awwal, Codesign of embedded systems with process/module level real-time deadlines, in *2009 International Conference on Computational Science and Engineering*, vol. 2 (2009), pp. 526–531

[7] W. Böhm, M. Broy, C. Klein, K. Pohl, B. Rumpe, S. Schröck (eds.), *Model-Based Engineering of Collaborative Embedded Systems* (Springer, 2021)

[8] C.A.R. Hoare, Communicating sequential processes. Communications of the ACM **21**(8), 666–677 (1978)

[9] C.A.R. Hoare, *Communicating Sequential Processes* (Prentice Hall International, Englewood Cliffs, N.J., 1985)

[10] T. Murray, G. Lowe, On refinement-closed security properties and nondeterministic compositions. Electr. Notes Theor. Comput. Sci. **250**, 49–68 (2009)

[11] R. Milner, *A Calculus of Communicating Systems* (Springer-Verlag, Berlin, Heidelberg, 1982)

[12] J. Parrow, *Handbook of Process Algebra* (Elsevier Science, Amsterdam, 2001), chap. An introduction to the pi-calculus, pp. 479–543

[13] E. Lee, Fundamental limits of cyber-physical systems modeling. ACM Transactions on Cyber-Physical Systems **1**, 1–26 (2016)

[14] M. Abadi, L. Lamport, Open Systems in TLA, in *Proceedings of the thirteenth annual ACM symposium on Principles of distributed computing - PODC '94*, ed. by J. Anderson, D. Peleg, E. Borowsky (ACM Press, New York, New York, USA, 1994), pp. 81–90

[15] W. Reisig, *Petri Nets: An Introduction* (Springer, Berlin, Heidelberg, 1985)

[16] M. Broy, K. Stølen, *Specification and development of interactive systems: Focus on streams, interfaces, and Refinement* (Springer, New York, 2001)

[17] M. Broy, B. Rumpe, Modulare hierarchische Modellierung als Grundlage der Software- und Systementwicklung. Informatik-Spektrum **30**(1), 3–18 (2007)

[18] R. Carnab, *Meaning and Necessity: A Study in Semantics and Modal Logic* (The University of Chicago Press, Chicago, IL, USA, 1947)

[19] I. Saifarchive, B. Ammanath, 'Trustworthy AI' is a framework to help manage unique risk. MIT Technology Review (2020)

[20] ITU-T, Trustwothy AI. Tech. rep., International Telecommunication Union, Geneva (2022)

[21] F. Fieber, M. Huhn, B. Rumpe, Modellqualität als Indikator für Softwarequalität: eine Taxonomie. Informatik-Spektrum **31**(5), 408–424 (2008)

[22] H. Kausch, M. Pfeiffer, D. Raco, B. Rumpe, A. Schweiger, Enhancing System-model Quality: Evaluation of the MontiBelle Approach with the Avionics Case Study on a Data Link Uplink Feed System, in *Avionics Systems and Software Engineering Workshop of the Software Engineering 2024 - Companion Proceedings (AvioSE)* (Gesellschaft fÃ¼r Informatik e.V., 2024), pp. 119–138

[23] Object Management Group. SysML Specification Version 1.0 (2006-05-03) (2006). URL http://www.omg.org/docs/ptc/06-05-04.pdf

[24] A. Desai, S. Gulwani, V. Hingorani, N. Jain, A. Karkare, M. Marron, S. R, S. Roy, Program synthesis using natural language, in *Proceedings of the 38th International Conference on Software Engineering* (Association for Computing Machinery, New York, NY, USA, 2016), ICSE '16, pp. 345–356

[25] M. Ibrahim, R. Ahmad, Class Diagram Extraction from Textual Requirements Using Natural Language Processing (NLP) Techniques, in *2010 Second International Conference on Computer Research and Development* (2010), pp. 200–204

[26] X. Pang, Y. Zhou, P. Li, W. Lin, W. Wu, J.Z. Wang, A novel syntax-aware automatic graphics code generation with attention-based deep neural network. Journal of Network and Computer Applications **161**, 102636 (2020)

[27] M.D. Ernst, Natural Language is a Programming Language: Applying Natural Language Processing to Software Development p. 14 pages (2017)

[28] J.J. Thomas, V. Suresh, M. Anas, S. Sajeev, K.S. Sunil, Programming with Natural Languages: A Survey, in *Computer Networks and Inventive Communication Technologies*, ed. by S. Smys, R. Bestak, R. Palanisamy, I. Kotuliak (Springer, Singapore, 2022), Lecture Notes on Data Engineering and Communications Technologies, pp. 767–779

[29] T.B. Brown, Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020)

[30] B. Wang, Z. Wang, X. Wang, Y. Cao, R. A Saurous, Y. Kim, Grammar prompting for domain-specific language generation with large language models. Advances in Neural Information Processing Systems **36** (2024)

[31] L. Netz, J. Reimar, B. Rumpe, Using grammar masking to ensure syntactic validity in llm-based modeling tasks. arXiv preprint arXiv:2407.06146 (2024)

[32] Y. Moy, E. Ledinot, H. Delseny, V. Wiels, B. Monate, Testing or formal verification: Do-178c alternatives and industrial experience. IEEE Software **30**(3), 50–57 (2013)

[33] U. Schöpp, A. Schweiger, M. Reich, T. Chuprina, L. Lúcio, H. Brüning, Requirements-based code model checking, in *2020 IEEE Workshop on Formal Requirements (FORMREQ)* (IEEE Computer Society, Los Alamitos, CA, USA, 2020), pp. 21–27

[34] H. Kausch, M. Pfeiffer, D. Raco, B. Rumpe, MontiBelle - Toolbox for a Model-Based Development and Verification of Distributed Critical Systems for Compliance with Functional Safety, in *AIAA Scitech 2020 Forum* (American Institute of Aeronautics and Astronautics, 2020)

[35] K. Pohl, H. Hönninger, R. Achatz, M. Broy (eds.), *Model-Based Engineering of Embedded Systems* (Springer Berlin, Heidelberg, 2012)

[36] K. Pohl, H. Hönninger, H. Daembkes, M. Broy (eds.), *Advanced Model-Based Engineering of Embedded Systems* (Springer, Cham, 2016)

[37] H. Kausch, M. Pfeiffer, D. Raco, B. Rumpe, Model-Based Design of Correct Safety-Critical Systems using Dataflow Languages on the Example of SysML Architecture and Behavior Diagrams, in *Proceedings of the Software Engineering 2021 Satellite Events*, vol. 2814, ed. by S. Götz, L. Linsbauer, I. Schaefer, A. Wortmann (CEUR, 2021)

[38] H. Kausch, M. Pfeiffer, D. Raco, B. Rumpe, An Approach for Logic-based Knowledge Representation and Automated Reasoning over Underspecification and Refinement in Safety-Critical Cyber-Physical Systems, in *Combined Proceedings of the Workshops at Software Engineering 2020*, vol. 2581, ed. by R. Hebig, R. Heinrich (CEUR Workshop Proceedings, 2020)

[39] H. Kausch, M. Pfeiffer, D. Raco, B. Rumpe, A. Schweiger, Correct and Sustainable Development Using Model-based Engineering and Formal Methods, in *2022 IEEE/AIAA 41st Digital Avionics Systems Conference (DASC)* (IEEE, 2022)

[40] H. Kausch, M. Pfeiffer, D. Raco, A. Rath, B. Rumpe, A. Schweiger, A Theory for Event-Driven Specifications Using Focus and MontiArc on the Example of a Data Link Uplink Feed System, in *Software Engineering 2023 Workshops*, ed. by I. Groher, T. Vogel (Gesellschaft für Informatik e.V., 2023), pp. 169–188

[41] H. Kausch, M. Pfeiffer, D. Raco, B. Rumpe, A. Schweiger, Model-driven development for functional correctness of avionics systems: A verification framework for sysml specifications. CEAS Aeronautical Journal (2024)

[42] D. Cofer, G. Klein, K. Slind, V. Wiels, Qualification of Formal Methods Tools (Dagstuhl Seminar 15182). Dagstuhl Reports **5**(4), 142–159 (2015)

[43] J. Andronick, Please check my 500K LOC of Isabelle, in *Qualification of Formal Methods Tools – Report from Dagstuhl Seminar 15182* (2015)

[44] EASA, ARTIFICIAL INTELLIGENCE ROADMAP 2.0. Tech. rep., Cologne, Germany (2023). URL http://www.easa.europa.eu/ai

[45] N. Baumann, J.S. Diaz, J. Michael, L. Netz, H. Nqiri, J. Reimer, B. Rumpe, Combining Retrieval-Augmented Generation and Few-Shot Learning for Model Synthesis of Uncommon DSLs, in *Modellierung 2024 - Workshopband*, ed. by H. Giese, K. Rosenthal (GI, 2024), pp. 1–15