A Co-Simulation Based Approach for Developing Safety-Critical Systems

Daniella Tola[1] and Peter Gorm Larsen[1]

DIGIT, Department of Engineering, Aarhus University, Aarhus, Denmark
{dt,pgl}@eng.au.dk

Abstract. When developing safety-critical systems it is important to provide evidence that such systems do not pose a significant hazard to their surroundings. One way of providing such evidence is known as safety cases and here the Goal Structuring Notation is one of the most popular notations. Evidence can take many different forms and in this paper we explore whether co-simulation of critical scenarios can be used with advantage as a beneficial approach. This is illustrated with a combine harvester using different tools in a Functional Mockup Interface setting.

Keywords: Goal Structuring Notation · Co-simulation · Functional Mockup Interface.

1 Introduction

With the growing development in technology, the complexity of systems nowadays is increasing. More systems are operating in dynamic environments, where living creatures reside. If these systems fail, they may damage their surroundings, therefore they are called safety-critical systems [16].

It is crucial that such safety-critical systems operate in a safe manner. The increase of software in these types of systems makes the process of ensuring the system safety a challenge. This is due to the complications followed with the integration of software and systems engineering. A typical complication could be incomplete software specifications, where important details or requirements may be missing or stated incorrectly [16].

Developing safety-critical systems is a complex process, where a large amount of time is spent on demonstrating the safety of the system, either by performing physical tests or formal verification [7]. The effort required to develop safety-critical systems using traditional processes makes it hard to keep up with the shrinking time-to-market that is expected of such systems [16]. Performing complete physical tests of a system to demonstrate its safety can be challenging due to the uncontrollable operating environments, such as changes in the surrounding atmosphere, and unethical tests, such as testing a pilot-controlled aircraft during a thunderstorm. Instead other Validation and Verification (V&V) techniques must be used, such as simulation and co-simulation [24, 22]. The focus of this paper is how co-simulation can be incorporated in the development process of safety-critical systems.

The rest of this paper starts with background information for the reader in Section 2. Afterwards, Section 3 gives a brief overview of the developed process, followed by Sections 4 and 5 which go further into detail of the main steps of this process. Finally, Section 6 presents the concluding remarks and the potential future work.

2 Background

This section describes the required background knowledge to understand the subsequent sections in this paper.

2.1 Case Study

The case study is based on a system where the goal is to develop a combine harvester, which can autonomously drive within the boundaries of a crop field and harvest crops, without the need for a driver controlling the vehicle.

Figure 1 illustrates how the environment of the autonomous combine harvester may look, including a number of the sensors that will be used in the system. The figure also demonstrates how animals may hide in between the crops, which in some cases can be difficult to detect. The system can potentially injure or kill living creatures in the field, damage the harvested crops (if the vehicle runs over an animal, the harvested crops will be contaminated), or catch fire [26]. These are few of the reasons that this system is categorised as safety-critical. The motivation for using this system as the case study when researching how to incorporate co-simulation in the development of a safety-critical system, is:

- It is challenging to test the system in the different atmospheric environments where the conditions cannot be controlled, such as extremely windy, rainy, or foggy.
- It is unethical to test the system with living creatures on the field, to determine if the system operates as it should under these specific circumstances.

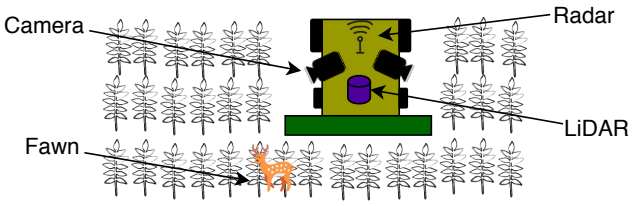


Fig. 1: An illustration of the autonomous combine harvester that is used as the case study in this paper.

2.2 Safety Case

The safety of a safety-critical system must typically be demonstrated to a government regulatory body, by creating a safety case [23], which is defined as [3]:

> "*A documented body of evidence that provides a convincing and valid argument that a system is adequately safe for a given application in a given environment.*"

The three main elements used in a safety case are [3, 15]:

Claim: a safety requirement or claim about the system, for example "System X is safe".

Evidence: used to support the claims about the safety of the system, for example physical tests of the system operating safely in specific environmental conditions.

Argument: used to describe how the evidence supports the safety claims.

The main stages of a safety case, that are relevant for this paper, are illustrated in Figure 2. The information obtained at each stage is used in the following stages. For example, the system description can be used in the process of deriving hazards. A hazard is defined as a system state, where the system may potentially damage its surroundings. It is important to note that a system can be in a hazardous state without causing an accident [23].

The hazard analysis stage consists of identifying hazards and their root causes. Different methods exist for finding root causes of hazards, and usually multiple methods will be used. In this paper we chose to only use one method, *fault-tree analysis*, due to its simplcity and graphical view. It is a top-down approach, starting with a hazard and moving down towards potential events that may lead to the occurrence of the hazard. In the fault-tree analysis approach, events that may lead to a hazard are presented graphically using logical gates, such as AND and OR gates [23].

The risk of each of the identified hazards is then assessed by determining the severity and likelihood of the hazards [6]. The determined risk of each hazard is then used to order the hazards from highest to lowest risk, and thereby determine the acceptable cost of reducing the risk of a hazard [23].

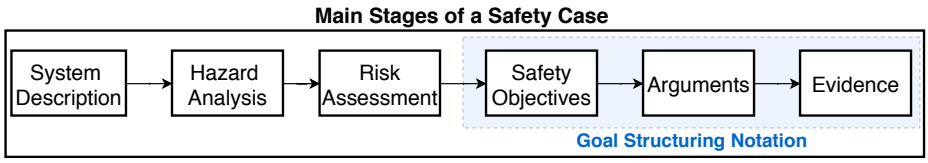


Fig. 2: The relevant main stages of a safety case.

One of the existing safety case processes, is to develop the system first and then create the safety case. The disadvantage of this process is that if safety

issues are found after the development of the system, then a system redesign may be necessary. Therefore, another process has been adopted, which is the phased safety case process, where the system and safety case are both developed simultaneously. This process is beneficial to use when a novel safety argument is being used, which is the case in this paper. There are typically three main phases to be developed in the phased safety case, Preliminary, Interim and Operational. The details of the safety case increase with each phase [14]. We only consider the Preliminary safety case in this paper.

2.3 Goal Structuring Notation

Creating explicit and clear safety arguments using plain text is not always a simple task, and can therefore end in misinterpretations by the regulator. The Goal Structuring Notation (GSN) attempts to solve these potential problems by using a graphical structure, that shows the connections of arguments, safety claims and evidence.

The relevant GSN elements used in this paper are illustrated in Figure 3, where the *goal*, *solution* and *strategy* are equivalent to the *safety objective*, *evidence* and *argument* in a safety case, respectively [15]. The *away goal* uses a claim from another module to support the argument in the current module [11]. The usage of an *away goal* in a GSN for arguing that a simulation tool adequately models the environmental conditions affecting part of a system, can be found in [24]. This method of argumentation, using an *away goal*, is also used in this paper.

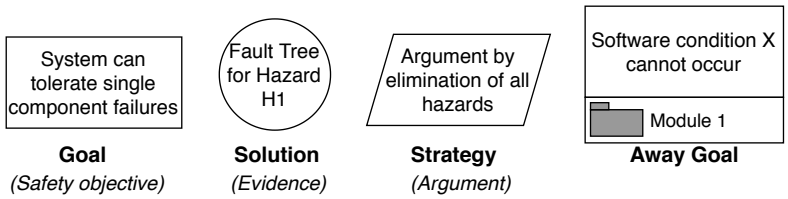


Fig. 3: The relevant elements used in *GSN*. The figure is inspired from [15].

2.4 Modelling and Co-simulation

A model is defined by J. Van Amerongen as [1]:

> "*A simplified description of a system, just complex enough to describe or study the phenomena that are relevant for our problem context.*"

As Amerongan explains, a model should be a simple representation of a system, that is easy enough to understand, but also complex enough to include the important characteristics of the system [1].

Performing simulations of the model with various parameters allows testing the model under different conditions. A challenge of obtaining a simulation of a complete system is that they are composed of several parts; often modelled using different mathematical formalisms. For example, the combine harvester consists of software, electrical and mechanical parts. Co-simulation allows to use models developed in different tools and execute them simultaneously [10], where a co-simulation engine is responsible for exchanging the data between the different models [9].

For the co-simulation engine to be able to communicate with the models, each model must implement an interface, such as the Functional Mock-up Interface (FMI). A model that implements the FMI is called a Functional Mock-up Unit (FMU) [4]. Figure 4 gives an idea of how FMUs developed using different tools can be run in a co-simulation, and also illustrates the main files within in an FMU. An FMU encapsulates a model as a black box with only the necessary parameters, inputs, outputs and binaries exposed in the interface. This allows Original Equipment Manufacturers (OEMs) to share models of their equipment, and at the same time protect their intellectual property [4].

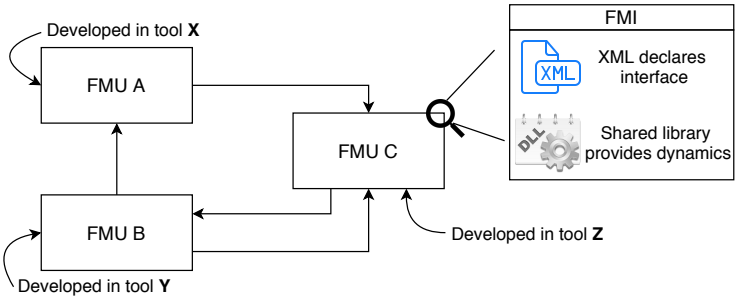


Fig. 4: An illustration of how co-simulation can combine FMUs developed using different tools, and also some of the main files inside an FMU.

3 Process Overview

This section gives a brief overview of the proposed process; more details can be found in [25]. This process provides clear guidelines on how to incorporate co-simulation in the development of safety-critical systems. The process does not yet include how to ensure the validity of the evidence obtained using co-simulations of the system.

The six main stages of the process are illustrated in Figure 5. These stages describe how to start from a hazard analysis and potentially end up with parts of evidence for a safety case. The first two stages are part of the traditional process of creating a safety case, where a safety and hazard analysis is performed,

followed by defining the safety goals of the system. Before proceeding to the third stage, the hazards must be studied to determine which method should be used for creating evidence. How to determine this, is described in the following section. If co-simulation is chosen as an appropriate method for creating evidence, then the next step is to specify test scenarios and create models of the different parts of the system. This is followed by co-simulations of the specified test scenarios, leading to results that if prove the system is safe under the specified conditions, can be used as evidence in the safety case. If the results imply the system is not safe under the specified conditions, then safety improvements must be made and the system must be co-simulated in the failed test scenarios again.

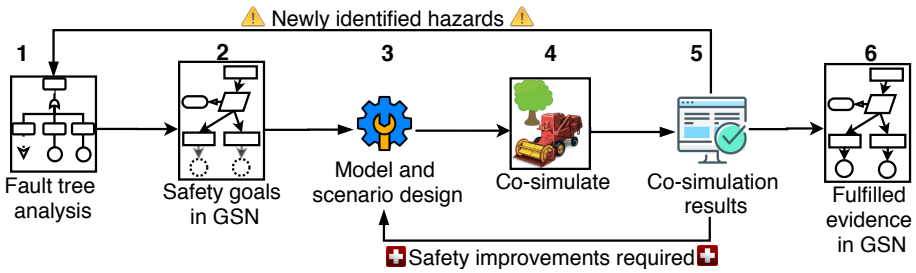


Fig. 5: An overview of the method of developing a safety case using co-simulation.

The contributed stages of the process of creating safety evidence are stages three, four and five, meaning that other safety processes construct evidence using other techniques in these exact stages. It is important to note that determining which elements of the safety evidence that can be constructed using co-simulation results, must be defined in stage two, before moving on to stage three. If it appears that no evidence can be created using co-simulation, then another evidence creation technique must be used, for example mathematical analysis.

The following sections describe how the process was used on the case study, where Section 4 describes the safety analysis, and Section 5 describes how the evidence was created using co-simulation.

4 Safety Case Analysis

This section describes how the safety case analysis of the autonomous combine harvester was performed. The steps illustrated above in Figure 2 are followed, where the first step of describing the system can be found above in Section 2.1. The details of the safety case and system are at a high abstraction level, since a Preliminary safety case is the objective.

Hazards were identified by consulting with an agricultural engineer from AGCO and reading articles. Some of the identified hazards are: *H1: collision with object*, *H3: fire ignition* [26], and *H5: contamination of harvested crops*. A

fault-tree analysis was performed on the hazards to determine potential root causes, see Figure 6a for a simplified fault-tree view. This was followed by a risk analysis to determine which hazards posed the highest risks. The risk analysis showed that hazards *H1* and *H3* posed the highest risks, which are the two hazards considered in this paper.

By analysing these two hazards with regards to how they can be mitigated, it was possible to determine that *H1* can be mitigated using software and redundant sensors, while *H3* can be mitigated using mechanical structures for isolating surfaces that may catch on fire. Comparing the mitigation methods of these two hazards, it is clear that *H1* is a relevant safety hazard to provide evidence for using co-simulation since it is possible to model the sensors, vehicle dynamics, and simulate the software for mitigating the hazard. *H3* is somewhat more difficult to co-simulate since the model would be more focused on the mechanics of the system.

A compact fault-tree analysis of the hazard *H1* is demonstrated in Figure 6a, where the events of the camera view being obscured or a decreased LiDAR performance could potentially lead to *H1* occurring. Possible root causes of these two events are illustrated in the fault-tree as heavy rain and dense fog. This information is used when describing the safety goals that are formalized using GSN, illustrated in Figure 6b. The evidence for fulfilling the safety goals *G2*, *G3* (from Figure 6b) and *G4: The system can tolerate inaccurate sensor measurements* is defined as follows:

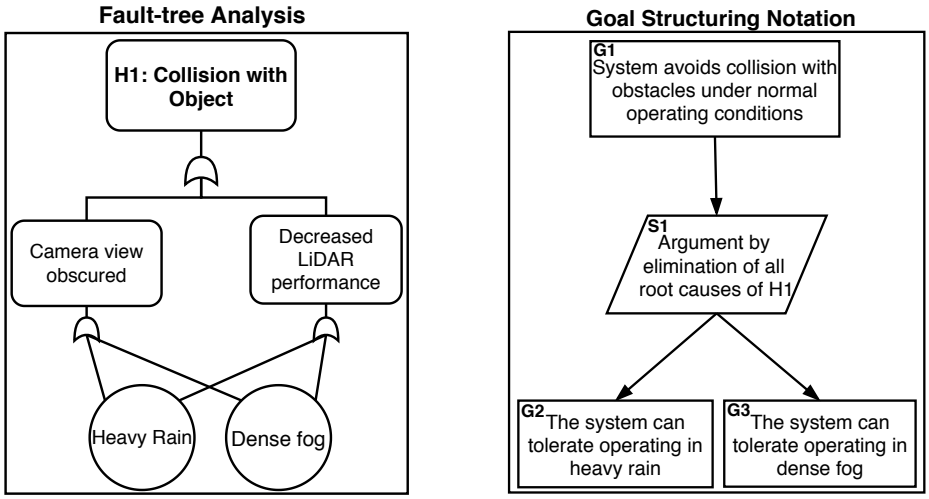
Evidence for G2: Co-simulation results of heavy rain scenarios.
Evidence for G3: Co-simulation results of dense fog scenarios.
Evidence for G4: Co-simulation results of scenarios with inaccurate sensor measurements.

5 Producing Evidence using Co-simulation

This section describes the process of creating evidence using co-simulation, by demonstrating how to follow the steps 3, 4, 5, and 6 presented above in Figure 5.

The first step consists of designing test scenarios and creating the relevant models. The test scenarios must be relevant to the physical system and possible to be executed in a co-simulation. The scenarios should be defined clearly in a way that makes it possible to reproduce the results of the co-simulation. They must also be directly related to the safety goals in a GSN, which in some cases are related to the hazard root causes. Table 1 describes the defined test scenarios, related to the GSN in Figure 6b, where the specific hazardous events are defined together with the vehicle speed and distance to the obstacle. These scenarios are typically unethical or expensive to test on the physical system, and therefore can beneficially be co-simulated instead. The goal is to co-simulate the system in these scenarios, and demonstrate its safety under the hazardous conditions.

In order to co-simulate these test scenarios, models of the vehicle dynamics, vehicle controller, sensors, environment and a safety controller must be created.



(a) Fault-tree analysis of *H1* with only few components included.

(b) Initial GSN of *H1* with only few components included.

Fig. 6: Example of how the fault-tree analysis can directly be translated to GSN. Further details of the diagrams can be found in [25].

Hazardous event	**Vehicle speed**	**Initial distance to obstacle**
Dense fog	{1,2,3} [m/s]	>1 meter
Heavy rain	{1,2,3} [m/s]	>1 meter
Inaccurate sensor	{1,2,3} [m/s]	>1 meter

Table 1: Defined test scenarios.

Each of the models created in this project are described in Table 2. The FMUs containing the models of the vehicle dynamics and controller were taken from an example project from INTO-CPS[1] [8]. The remaining FMUs were modelled using the python library, PyFMU [19]. The models of the system parts were created with only the necessary details that are relevant in this stage of the project, since a Preliminary safety case is the objective.

The main parts of the process of creating the sensor model are described below. This information is used to illustrate how to combine knowledge from the hazardous events and the sensors of the system. As described above in Section 2.1, a camera, LiDAR and radar are mounted on the system. The effect of dense fog and heavy rain on cameras and state-of-the-art LiDARs is described as

[1] `https://github.com/INTO-CPS-Association/example-autonomous_vehicle`
[2] `https://www.20sim.com/`
[3] `http://overturetool.org/`
[4] `https://pypi.org/project/pyfmu/`
[5] `https://opencv.org/`

Model	Program	Description
Vehicle	20-sim[2]	Models the vehicle dynamics using a bicycle model.
Controller	Overture[3]	Controls the vehicle using the pure-pursuit path following algorithm.
Sensor	PyFMU[4]	Models the three types of sensors combined, i.e. LiDAR, radar and camera. A more detailed explanation of this model is presented below.
Environment	PyFMU	Models the obstacles in the environment surrounding the vehicle. The environment was modelled using the python library OpenCV[5]. The environment can be loaded using a white background image, with obstacles defined as coloured objects on the image. This allows easily creating different environments.
Supervisory Controller	PyFMU	Safety controller that performs a safety stop before crashing into an obstacle. The controller performs a safety stop if a detected obstacle is within the range of the calculated braking distance of the vehicle. The obstacles are detected using the sensor model. Therefore, if the sensor model does not detect an obstacle, then the Supervisory Controller will not receive information about an obstacle, and will therefore not perform a safety stop.

Table 2: Models used in the project, together with the program that they were modelled in, and a description of each model.

a reduction in the maximum viewing range of the sensors [2, 12, 20, 17]. Figure 7 illustrates how dense fog and heavy rain can be modelled using a single sensor model at a high abstraction level.

Figure 8 illustrates how the sensor model combines all the sensors into one, instead of creating a model of each sensor. This allowed modelling the complete effect of the environment, using adjustable parameters defining the minimum and maximum sensor range. The minimum range of the sensor represents the hazard of inaccurate sensor measurements [18, 5]. In the future, when more knowledge about the sensors and the system is obtained, details can be added to the constructed models iteratively [21].

In total, 2 scenarios were co-simulated, with the parameters described below, leading to in total 12 co-simulation runs:

Environment: Two different obstacle maps, with varying obstacle locations, densities and sizes.

Vehicle speed: 1, 2, and 3 [m/s]. These values are taken from [13], which describes the optimim combine harvester speed is around 2.2 [m/s].

Minimum sensor range: When operating correctly, the range was modelled as 0.5m. When operating with inaccurate measurements, the range was modelled as 1m [18, 5].

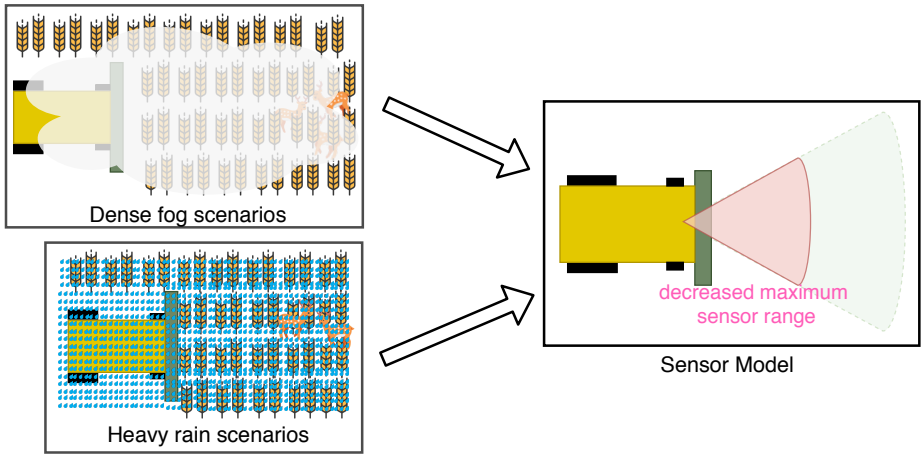
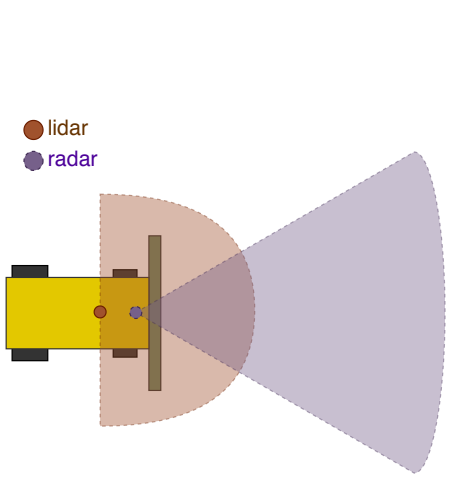
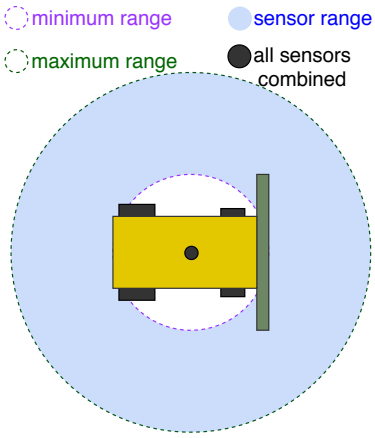


Fig. 7: An illustration of how the scenarios related to dense fog and heavy rain may be modelled simultaneously, since they affect the system in similar ways. The figure demonstrates that the scenarios where dense fog or heavy rain occur affect the maximum viewing range of the sensor.



(a) A mounted lidar and radar on a combine harvester, showing the range of each of the sensors. This is just an example, showing two mounted sensors, but in reality there will be several types of sensors mounted in different places on such a system.



(b) An example illustrating how the sensors are modelled, by generalizing a combination of all of the sensors on a combine harvester. The range of the generalized sensors together is modelled as a circular area around the vehicle, with a minimum and maximum range.

Fig. 8: Generalization of sensors on the autonomous combine harvester.

Maximum sensor range: When operating in dense fog or heavy rain, the range was modelled as 2m, which is the worst case values found in the articles [2, 12, 20, 17].

The different values of the parameters and number of co-simulation runs must be sufficient in order to be able to use the results as evidence. The worst-case values were chosen, but potentially more confidence in the evidence of these hazardous conditions could be obtained using best-case values and at least one value in between.

Figure 9 shows one of the live plots of a co-simulation run, where the vehicle successfully performs a safety stop before colliding with the detected obstacles. In the cases where the vehicle failed to perform a safety stop, the system had run over the obstacles.

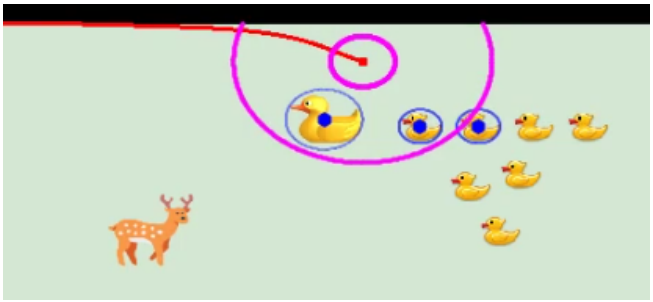


Fig. 9: An example of a co-simulation run of the system with visualization of the minimum and maximum range of the sensor, and the currently detected objects. The red lines displays the route the vehicle has driven so far. The two pink circles represent the sensor minimum and maximum range. The blue circles represent the detected objects. The minimum and maximum ranges are set to 0.5m and 2m respectively in this co-simulation, and the vehicle has performed a safety stop.

The co-simulation results were successful for the scenarios simulating the dense fog and heavy rain, but unfortunately failed in a number of the scenarios simulating the inaccurate sensor measurements. The successful results were used as evidence in the GSN shown in Figure 10; notice the validated model is used as part of the argument for justifying the use of co-simulation as evidence. The model is not yet validated, and therefore the away goal, *G0_ModelValidated*, describing it is greyed out. The unsuccessful co-simulation results mean that improvements on the safety mechanisms must be made, in order to obtain the evidence *Sn3*. It is important to note that the safety properties derived by co-simulation results will only be valid to be used as evidence, if the model of the system is validated and sufficiently represents the true system.

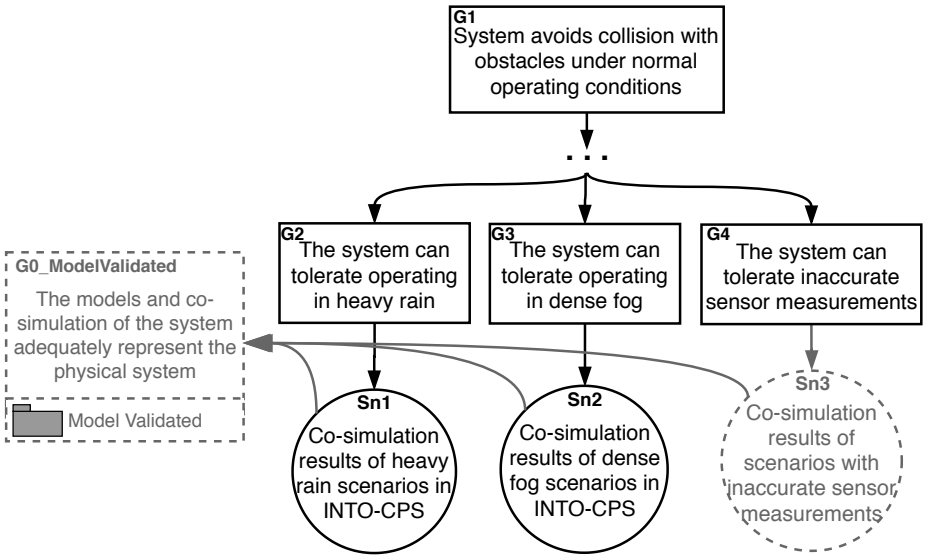


Fig. 10: The relevant parts of the GSN diagram, illustrating which evidence was obtained using co-simulation. The dashed border of *Sn3* and *G0_ModelValidated* illustrates that these components are yet to be constructed, i.e. the evidence for *Sn3* must be obtained, and the model used in the evidence *Sn1* and *Sn2* must be validated against the physical system.

6 Concluding Remarks and Future Work

This paper has demonstrated guidelines for incorporating co-simulation into the process of developing a safety case using FMI-based co-simulation. This was demonstrated by applying the process to a case study of an autonomous combine harvester.

Co-simulation is a strong tool for experimenting with system behaviour, when the system consists of software, electrical and mechanical parts that work together. It is especially relevant to use in the safety case of a complex system, where physical tests of the system are either unethical or challenging to perform. The combine harvester was beneficially co-simulated in scenarios such as dense fog and heavy rain, which are conditions that cannot be controlled in the physical world. The co-simulation was visualized, allowing to create videos of the results, which can advantageously be presented to the regulatory bodies, to help them easier understand how the safety mechanisms ensure system safety under specific conditions.

Different techniques for producing evidence exist, as illustrated in Figure 11, where it is important to consider which technique is most suitable to use depending on the nature of the problem. Looking at co-simulation, the time spent on modeling, co-simulating, analyzing the results and validating the models used, is important to consider. In some cases co-simulation may require too much time, and there may be other techniques that can formally prove the safety of the system, such as a mathematical proof.

This paper has shown how co-simulation can be used to demonstrate that the design of a system has adressed the identified hazards. This was achieved by running co-simulations of a system model in different hazardous conditions, and using the co-simulation results as evidence in the GSN of a safety case.

Future Work

The work presented in this is based on a specific case study with focus on one hazard, and therefore there are multiple potential areas that can be researched:

Extend process: The process could be extended to both interim and operational safety cases, potentially providing guidelines on how the model fidelity can be increased between each phase.

Amount of parameters per scenario: Determining how many parameters should be experimented with is important, to ensure sufficient results for evidence, but also avoid running an endless amount of tests.

Model validation: The model used in this project must be validated, and the methods used to validate it can be explained. Choosing the specific scenarios and parameters to use during validation should be part of this research.

Evaluate process on other systems: Using this process on other types of systems, such as medical devices which require human operators, could demonstrate if the process can directly be transferred to other areas or if adjustments must be made.

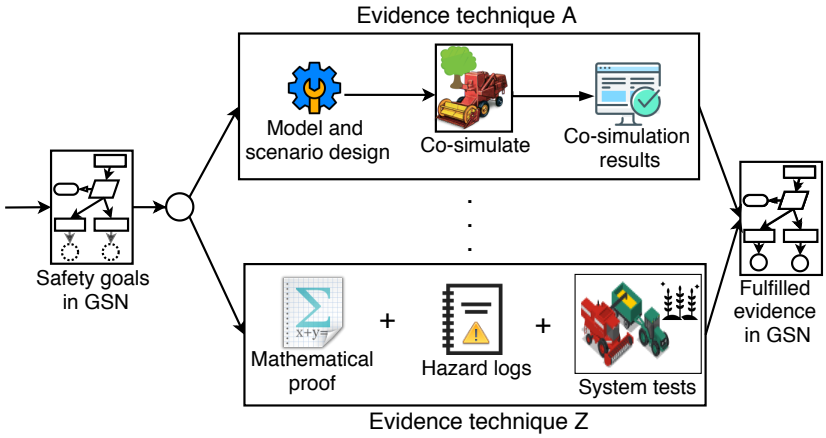


Fig. 11: An illustration of the possible techniques for producing evidence that can be used in a safety case. The specific technique must be determined after defining the safety goals. Note that evidence technique A also requires model validation in order to be able to use the co-simulation results as evidence.

Acknowledgements. We would like to thank Martin Peter Christensen from AGCO for the discussions about the challenges for an autonomous combine harvester. We would also like to express our thanks to the anonymous reviewers.

References

1. van Amerongen, J.: Dynamical Systems for Creative Technology. Controllab Products, Enschede, Netherlands (2010)
2. Bijelic, M., Gruber, T., Ritter, W.: A benchmark for lidar sensors in fog: Is detection breaking down? In: 2018 IEEE Intelligent Vehicles Symposium (IV). pp. 760–767 (6 2018). https://doi.org/10.1109/IVS.2018.8500543
3. Bishop, P.G., Bloomfield, R.E.: A methodology for safety case development. In: Safety-critical Systems Symposium. Birmingham, UK (February 1998)
4. Blochwitz, T.: Functional Mock-up Interface for Model Exchange and Co-Simulation. `https://www.fmi-standard.org/downloads` (July 2014)
5. Clark, R.: Selecting a lidar system, `http://www.acuitylidar.com/PDFs/Selecting\%20a\%20LIDAR\%20System.pdf`
6. Despotou, G., White, S., Kelly, T., Ryan, M.: Introducing safety cases for health it. In: Breu, R., Hatcliff, J. (eds.) Proceedings of the 4th International Workshop on Software Engineering in Health Care, SEHC 2012, Zurich, Switzerland, June 4-5, 2012. pp. 44–50. IEEE Computer Society (2012)
7. D'Silva, V., Kroening, D., Weissenbacher, G.: A survey of automated techniques for formal software verification. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **27**(7), 1165–1178 (2008)
8. Fitzgerald, J., Gamble, C., Larsen, P.G., Pierce, K., Woodcock, J.: Cyber-Physical Systems design: Formal Foundations, Methods and Integrated Tool Chains. In:

FormaliSE: FME Workshop on Formal Methods in Software Engineering. ICSE 2015, Florence, Italy (May 2015)
9. Fitzgerald, J., Larsen, P.G., Verhoef, M. (eds.): Collaborative Design for Embedded Systems – Co-modelling and Co-simulation. Springer (2013)
10. Gomes, C., Thule, C., Broman, D., Larsen, P.G., Vangheluwe, H.: Co-simulation: a Survey. ACM Comput. Surv. **51**(3), 49:1–49:33 (May 2018)
11. Group, T.A.C.W.: Goal structuring notation community standard (2 2018), `https://scsc.uk/r141B:1?t=1`
12. Heinzler, R., Schindler, P., Seekircher, J., Ritter, W., Stork, W.: Weather influence and classification with automotive lidar sensors. pp. 1527–1534 (06 2019). https://doi.org/10.1109/IVS.2019.8814205
13. Isaac, N., Quick, G., Birrell, S., Edwards, W., Coers, B.: Combine harvester econometric model with forward speed optimization. Applied Engineering in Agriculture **22** (06 2006). https://doi.org/10.13031/2013.20184
14. Kelly, T.: A systematic approach to safety case management (01 2003). https://doi.org/10.4271/2004-01-1779
15. Kelly, T., Weaver, R.: The goal structuring notation – a safety argument notation. In: Proc. of Dependable Systems and Networks 2004 Workshop on Assurance Cases (2004)
16. Knight, J.C.: Safety critical systems: Challenges and directions. In: Proceedings of the 24th International Conference on Software Engineering. p. 547–550. Association for Computing Machinery, New York, NY, USA (2002)
17. Koyluoglu, T., Hennicks, L.: Evaluating rain removal image processing solutions for fast and accurate object detection. Master's thesis, KTH ROYAL INSTITUTE OF TECHNOLOGY, Sweden (2019)
18. Lee, S., Cho, H., Yoon, K.J., Lee, J.: Intelligent Autonomous Systems 12: Volume 1 Proceedings of the 12th International Conference IAS-12, Held June 26-29, 2012, Jeju Island, Korea. Springer Publishing Company, Incorporated (2012)
19. Legaard, C.M., Gomes, C., Larsen, P.G., Foldager, F.F.: Rapid Prototyping of Self-Adaptive-Systems using Python Functional Mockup Units. SummerSim '20, ACM New York, NY, USA (2020)
20. Liu, Z., He, Y., Wang, C., Song, R.: Analysis of the influence of foggy weather environment on the detection effect of machine vision obstacles. Sensors **20**(2), 349 (1 2020). https://doi.org/10.3390/s20020349, `http://dx.doi.org/10.3390/s20020349`
21. Maria, A.: Introduction to modeling and simulation. In: Proceedings of the 29th Conference on Winter Simulation. pp. 7–13. IEEE Computer Society (1997)
22. Martin, H., Tschabuschnig, K., Bridal, O., Watzenig, D.: Functional Safety of Automated Driving Systems: Does ISO 26262 Meet the Challenges?, pp. 387–416. Springer International Publishing (September 2017)
23. Sommerville, I.: Software Engineering. Addison-Wesley, 5th edition edn. (1996), iSBN 0-201-42765-6
24. Sun, L., Kelly, T.: Safety arguments in aircraft certification. In: 4th IET International Conference on Systems Safety 2009. Incorporating the SaRS Annual Conference. pp. 1–6 (2009)
25. Tola, D.: A Co-Simulation Based Approach for Developing Safety-Critical Systems. Master's thesis, Aarhus University, Department of Engineering (June 2020)
26. Val, J., Videgain, M., Martin-Ramos, P., Cortés, M., Boné-Garasa, A., Ramos, F.: Fire risks associated with combine harvesters: Analysis of machinery critical points **9**, 877 (12 2019). https://doi.org/10.3390/agronomy9120877