Assurance Case Arguments in the Large: the CERN LHC Machine Protection System

Laure Millet[1], Simon Diemert[1], Chris Rees[1], Torin Viger[2], Marsha Chechik[2], Claudio Menghi[3], and Jeffrey Joyce[1]

[1] Critical Systems Labs, Inc.
{laure.millet,simon.diemert,chris.rees,jeff.joyce}@cslabs.com
[2] University of Toronto {torinviger,chechik}@cs.toronto.edu
[3] University of Bergamo and McMaster University claudio.menghi@unibg.it

Abstract. Most public assurance arguments are used to introduce, discuss, and present novel concepts and techniques related to structured argumentation. These examples often rely on generic claims such as "All hazards have been identified" and generic patterns of reasoning and are quite different from their fully developed industrial counterparts. This practical experience report describes a medium-size assurance case argument for the CERN Large Hadron Collider Machine Protection System expressed using Eliminative Argumentation. This assurance case with 509 nodes was created in approximately three months, validated in collaboration with CERN experts, and is now publicly available. We also report on our practical experience in creating this argument and reflect on the support provided by the features of the collaborative assurance case editor we used called *Socrates*.

Keywords: Assurance Case · Large Hadron Collider · Nuclear · Goal Structuring Notation (GSN) · Tools.

1 Introduction

Producing high-quality Assurance Case (AC) arguments for industrial systems is complex and time-consuming, especially when the system design is highly innovative and experts cannot benefit from a previously established structure for the argument. Examples of such complex industrial arguments are not generally publicly available: they are typically proprietary and protected by non-disclosure agreements. The absence of representative examples of industrial arguments is a severe limitation to the scientific and industrial communities. Scientists and researchers need representative examples to evaluate new methods and techniques. Instead, they can only rely on small-scale showcase examples that often do not represent the challenges faced by practitioners. On the other side, industry cannot often assess the maturity and applicability of the different research results since they lack validation on industrial-scale examples.

This problem motivated a collaborative effort involving Critical Systems Labs (CSL), the University of Toronto, McMaster University, and the European Organization for Nuclear Research (CERN) to produce a representative assurance

case argument to be publicly shared. We chose the CERN Large Hadron Collider (LHC) Machine Protection System (MPS) for this case study for four reasons. First, the MPS is a large safety-critical system that was the result of more than 10 years of effort involving highly knowledgeable and skilled personnel. As such, it likely shares some similarities with other safety-critical systems. Second, there is a substantial amount of openly available technical documentation about the MPS design [11]. Third, some of the authors from CSL were previously involved in technical reviews of the MPS during its commissioning and were already familiar with the system [3]. Finally, we could rely on the feedback from CERN experts to validate our argument.

We prepared this assurance case using the Eliminative Argumentation (EA) method and notation [4]. We selected EA since it explicitly supports the expression (and resolution) of doubt about the validity of the argument through the inclusion of *defeaters*. We have also used EA to prepare ACs for other safety-critical systems [1]. Tool support for EA-based arguments is also provided by *Socrates* [12], a collaborative assurance case tool developed by CSL.

This paper reports on the following contributions. First, we present our medium-size EA argument for the MPS and make it freely accessible by members of the scientific community and industry [2]. This argument consists of 509 nodes of different types, including 146 claim nodes and 105 defeaters. Second, we reflect on features the AC tool we used and their impact collaborative development of the argument by a geographically distributed team.

The rest of this paper is organized as follows. Following a brief overview of the CERN Large Hadron Collider MPS (Section 2) and an introduction to EA (Section 3), this paper briefly describes the structure of the assurance case argument for the MPS (Section 4). Then we discuss features of the assurance case editor that facilitated the collaborative development of the argument (Section 5). Finally, we present our conclusions (Section 6).

2 The Machine Protection System

Built by the European Organization for Nuclear Research (CERN), the Large Hadron Collider is the world's largest particle accelerator and collider. It is a 27-kilometer ring that contains thousands of superconducting magnets that increase the energy of two particle beams until they reach nearly the speed of light, and then make them collide. Collision experiments are performed to analyze phenomena related to the collision: testing theories and investigating unanswered questions in particle physics.

The hyper-accelerated particle beams generated during the experiments release a significant amount of energy which could damage to the system if their trajectories become unstable; this phenomenon is called "beam loss". The *Machine Protection System (MPS)* is a monitoring system composed of several interdependent sub-systems designed to ensure that the Large Hadron Collider does not become damaged during operation due to unstable particles [7]. When a beam loss is detected, the MPS is responsible for performing a "beam dump"

(i.e., extracting all particles before hazardous scenarios occur) within 400µs of the occurrence of a beam loss event. A beam permit signal is used by components of the MPS to communicate that the LHC can continue to operate safely.

The MPS is comprised of four main sub-systems: the Beam Loss Monitoring System (BLMS), the Beam Interlock System (BIS), the Beam Dumping System (BDS) and the Safe Machine Parameter Controller (SMPC).

• The *Beam Loss Monitoring System* consists of approximately 4000 monitors distributed within the LHC to measure the beam loss. It is designed to detect and communicate beam losses to the Beam Interlock System within 80µs and uses the Safe Machine Parameter Controller to track beam losses events accurately. When intolerable beam losses are detected, it signals the Beam Interlock System to initiate a beam dump by withdrawing the beam permit.

• The *Beam Interlock System* takes between 20µs and 120µs to receive, process, and transmit a beam permit signal to the Beam Dumping System and transmits loss of the beam permit in at most 100µs. It also determines whether the Beam Dumping System should initiate a beam dump due to other conditions, such as loss of redundancy of critical components.

• The *Beam Dumping System* is responsible for extracting the beams from the LHC's rings without damaging the system. The BDS directs the beams towards a large graphite sink designed to absorb the beam's energy. Components called "kicker magnets" are used to divert the beams from the main LHC ring towards the sink. The magnets also disperse the beams into smaller clusters to reduce the energy density when beam reaches the sink.

• The *Safe Machine Parameter Controller* computes Safe Machine Parameters used by the other systems to identify dangerous or spurious beam losses within the LHC.

Considering the consequence of a failure of the MPS, engineers need to ensure that risk is properly mitigated. The Eliminative Argumentation (EA) method and notation for preparing ACs can support engineers in this activity.

3 Eliminative Argumentation for Assurance Cases

Eliminative Argumentation (EA) [5,4] is a graphical notation to express assurance cases. EA allows engineers to reason about properties such as safety or security and and their confidence in the argument. It is similar to *Goal Structuring Notation* (GSN) [6,8]. Like GSN, EA uses a directed acyclic graph to organize the argument structure. EA and GSN arguments both approximate a tree data structure that starts with (at the "root") a high-level claim about the system that is decomposed into sub-claims until it can be directly supported by evidence, thus providing traceability between evidence and the claims. Unlike GSN, EA enables analysts to express "doubts", a.k.a., *defeaters*, they might have in the validity of claims, evidence, or inferences in an AC. EA is founded on the notion of 'defeasible reasoning' where confidence in the validity of a claim (or evidence or inference) increases as reasons to doubt that claim are resolved.

Thus, EA defeaters can express doubt about claims, evidence, if it does not accurately support its parent's claim, and inference rules to identify doubts one may have about the the soundness or completeness of the argument.

EA enables analysts to use different types of nodes [1] to build an argument: *claim nodes* express statements that must be supported by further argumentation to demonstrate their validity; *evidence nodes* express observations, data, or artifacts that support the argument; *strategy nodes* describe the approach used to organize a collection of nodes; *inference rules* describe how to logically combine a collection of nodes; *context nodes* provide background information or missing details that may be necessary to understand the argument; *assumption nodes* list conditions related to the system or its operational environment assumed to be true in the argument, and *defeater nodes* express doubts about the validity of an assurance argument.

In EA, confidence in the top-level claim is established by showing that reasons to doubt the case have been resolved. If a doubt not resolved, it is marked as "residual" and contributes to the overall doubt in the argument. In our experience, residual doubts are a helpful when communicating sources of risk with stakeholders, such as management, regulators, or customers [1].

4 The Machine Protection System Assurance Case

This section describes the AC for the Machine Protection System (MPS) created using the EA. The argument consists of 509 nodes and is publicly available [2] as both a PDF report and a machine-readable archive (JSON). Among the 105 defeaters that can be found in the argument, 10 expose unmitigated risks in the MPS that were confirmed to be valid by CERN experts.

Figure 1 presents the high-level structure of the EA argument for the MPS. The argument starts with a top-level claim (C0001) that asserts that "*The MPS protects the LHC against damage from potential beam losses*". This claim is decomposed into four sub-arguments ("branches"), one for each MPS sub-system (see Section 2).

For each sub-system, the corresponding sub-argument argues that design is acceptably safe. The argument relies on various decomposition strategies. For example, Figure 2 presents a portion of the argument that decomposes the claim C0030, that states that "*The BIS will transmit loss of the beam permit to the BDS in less than 100 microseconds*", via strategy S0654, by arguing over the foreseeable failure modes that may block, delay or otherwise interfere with the transmission of a beam dump request from the BIS to the BDS. These failure modes are represented by the defeater nodes D0031, D0036, D0438, and D0512.

Each defeater branch is followed by a sub-argument that addresses how the associated risk is mitigated (not shown in Figure 2). For example, the sub-argument rooted at D0031 takes into account the fail-safe design of the mechanism responsible for the transmission of the beam permit including redundancy of critical components.

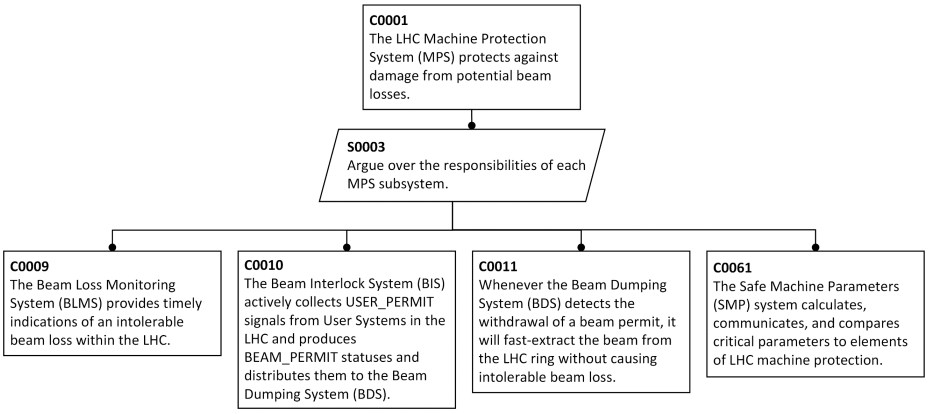

Fig. 1: High-level argument structure for the Machine Protection System.

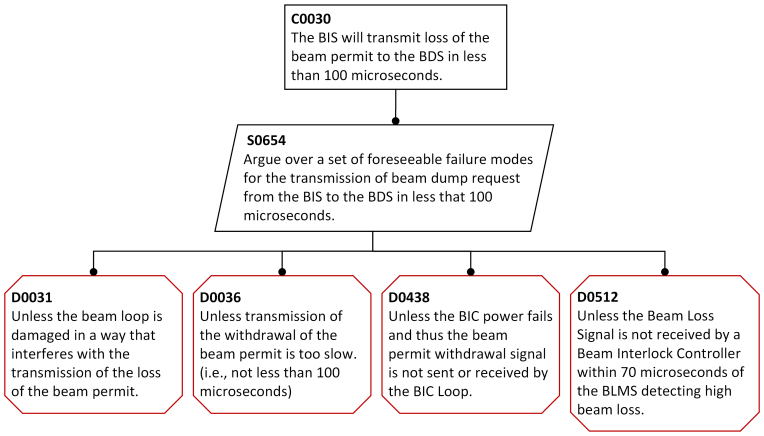

Fig. 2: A portion of the argument for the Beam Interlock System.

Through the argument, each claim is either refined into a set of evidence nodes or defeaters. Each evidence node is associated with an artifact that directly references specific details of the MPS design that justifies why the risk is sufficiently mitigated. For example, Figure 3 shows that the evidence E0543 ("*In the event of one or all transmission lines being damaged, the bean permit loop will have no 10MHz signal or noise and subsequently result in the request for a beam dump*") is contained in the document "Architecture of Beam Interlock System". In the AC tool *Socrates*, the user can click to open the corresponding document on CERN's file server.

In our argument, defeaters that are not resolved are marked as either *residual* or *undeveloped*. *Undeveloped defeaters* identify doubts for which no resolution is available. *Residual defeaters* identify doubts that are not resolved but are considered acceptable.

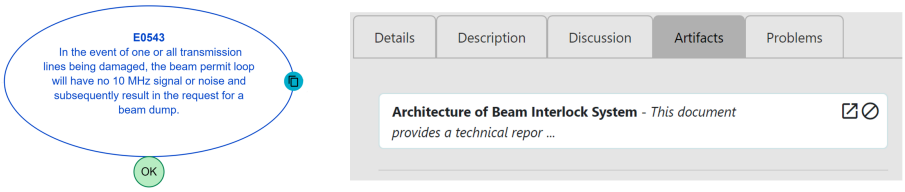

Fig. 3: Evidence node E0543 and its reference to its supporting artifact.

5 Lessons Learned from our Collaborative Development

In this section, we reflect on our practical experience developing an argument by a geographically distributed team. The development of our AC required approximately 92 effort-days. It was the output of a collaborative effort from four safety engineers with various degrees of experience and no prior knowledge of the MPS system. We developed our argument using *Socrates* [12], a web-based tool that enables collaborative AC development. In the following, we discuss our experience in the context of capabilities of AC tools identified by other authors [9].

Navigation Features. An industrial-scale AC is likely difficult to navigate without proper tooling. Unlike a figure created using a drawing tool, we developed our argument using a tool that automatically rendered a data structure representing the AC. This reduced the effort required from developers to layout the argument in a visually appealing format. Moreover, the tool we used provides several navigation features such as the ability to expand or collapse branches of the argument or the search function to navigate to specific sub-trees. These navigation features were increasingly useful as the size of the AC increased.

Collaborative Development. Collaborative tools, such as *Socrates*, enable developers to work on the AC in parallel, by editing the argument simultaneously. This feature was effective in preventing conflicts as this project's team was distributed over two continents and four different time zones. Our team extensively used a feature that enables discussion between developers by attaching comments to nodes in the argument.

Natural Language Processing. From our experience, we learned that tools with a grammar checker or other basic language processing might be helpful during the development of an AC. For example, to avoid ambiguous phrasing. The tool we used offered no language processing support. Like many real-world ACs, specialized terminology is used in the LHC MPS argument. Inconsistent use might be avoided by the presence of a built-in glossary.

Linking Artifacts. This feature allows developers to link nodes in the argument to external artifacts. As with many large ACs for complex systems, one of the main challenges in the development of the MPS argument was mapping aspects of the argument to details in the documentation. We linked AC nodes to the documentation that defines or acts as evidence for these nodes.

Version Control. This feature provides the ability for developers to save stable versions of the AC. When preparing and maintaining an AC, developers often

restructure the argument to improve its understandability or to reflect changes to the system. In our project, we found this version control useful, especially to consult previous versions of the AC.

Rule-based Static Analysis. This feature provides rule-based checks on the argument structure that address both the syntactic correctness of the argument and detects common "anti-patterns" in the argument. For most of the project team, this was their first experience with the development of a large AC using EA. Static analysis does not ensure that an argument is logically sound and complete, but it helped developers minimize the number of basic errors. The real-time feedback provided by this feature also helped team members learn EA.

Impact Analysis. Modifying, adding or deleting a node may affect the structure of the entire sub-argument in which it is contained. In such cases, an analyst may find it necessary to apply the same modification to other unrelated sub-arguments that share a similar structure, in order to ensure the overall cohesion of the argument. Beyond basic search and linking functions, the tool we used did not offer sophisticated impact analysis functions. However, had these been available our experience suggests they would have reduced the effort associated with argument maintenance, particularly in the later stages of development.

Metrics. The tool we used reports metrics about the argument, such as a timeseries showing how the number of nodes in the argument (stratified by node type) changed over time. For example, we used the rate at which new defeater nodes were created to understand when the developers had reached a level of technical understanding sufficient to pose probing questions to the CERN experts participating in this project. Overall, this feature helped us to gauge our understanding of the LHC's MPS and the maturity of the AC.

6 Conclusion

This paper presented a medium-size assurance case argument for the CERN Large Hadron Collider Machine Protection System that is expressed using Eliminative Argumentation. We reported on our practical experience in creating this argument and reflected on the support provided by the capabilities of the collaborative assurance case tool that we used.

In future work, we plan to develop a framework that can monitor the activity of the analysts and provide practical recommendations as mandated by our manifesto, which proposes using assurance cases as data [10]. We are also empirically studying assurance case development.

Acknowledgements

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) [funding reference numbers RGPIN-2022-04622, DGECR-2022-0040, RGPIN-2015-06366].
We thank Mateo Delgado and Rolf Lippelt for their contribution, and CERN

experts Jan Uythoven Markus Zerlauth, and Lukas Felsberger for their review and feedback on the AC.

References

1. Diemert, S., Joyce, J.: Eliminative argumentation for arguing system safety - A practitioner's experience. In: International Systems Conference (SysCon). pp. 1–7. IEEE (2020)
2. LHC MPS argument. `http://cds.cern.ch/record/2854725/files/` (02 2023)
3. Ghafari, N., Kumar, R., Joyce, J., Dehning, B., Zamantzas, C.: Formal verification of real-time data processing of the LHC beam loss monitoring system: a case study. In: Formal Methods for Industrial Critical Systems. pp. 212–227. Springer (2011)
4. Goodenough, J., Weinstock, C., Klein, A.: Eliminative argumentation: A basis for arguing confidence in system properties. Tech. Rep. CMU/SEI-2015-TR-005, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA (2015), `http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=434805`
5. Goodenough, J.B., Weinstock, C.B., Klein, A.Z.: Eliminative Induction: A Basis for Arguing System Confidence. In: International Conference on Software Engineering (ICSE). pp. 1161–1164. IEEE (2013)
6. Group, A.C.W.: Goal structuring notation community standard - version 3. Tech. rep., Safety Critical Systems Club (2021), `https://scsc.uk/r141C:1?t=1`
7. Holzer, E.B., Dehning, B., Effnger, E., Emery, J., Grishin, V., Hajdu, C., Jackson, S., Kurfuerst, C., Marsili, A., Misiowiec, M., Nagel, M., del Busto, E.N., Nordt, A., Roderick, C., Sapinski, M., Zamantzas, C.: Beam loss monitoring for LHC machine protection. Physics Procedia **37**, 2055–2062 (2012)
8. Kelly, T.P.: Arguing safety-a systematic approach to safety case management. DPhil Thesis York University, Department of Computer Science (1999)
9. Koopman, P.: The UL 4600 Guidebook: What to Include in an Autonomous Vehicle Safety Case. Carnegie Mellon University (2022)
10. Menghi, C., Viger, T., Di Sandro, A., Rees, C., Joyce, J., Chechik, M.: Assurance Case Development as Data: A Manifesto. In: International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER). pp. 135–139. IEEE/ACM (2023)
11. CERN Website. `www.cern.ch` (04 2022)
12. Socrates . `https://safetycasepro.com/welcome` (04 2022)