Balancing the Risks and Benefits of Using LLMs to Support Assurance Case Development

S. Diemert, E. Cyffka, N. Anwari, O. Foster, T. Viger, L. Millet, J. Joyce

Balancing the risks and benefits of using large language models to support assurance case development

Simon Diemert[1,2][0000−0001−9493−7969], Erin Cyffka[1], Naweed Anwari[1], Olivia Foster[1], Torin Viger[1,3], Laure Millet[1], and Jeffrey Joyce[1]

[1] Critical Systems Labs Inc., Vancouver, Canada
[2] University of Victoria, Victoria, Canada
[3] University of Toronto, Toronto, Canada

Abstract. Large Language Models (LLMs) have been shown to aid human experts in the development of Assurance Cases (ACs), including supporting tasks like defeater generation and generating AC argument structures. As a result, LLMs have the potential to both improve the quality of ACs and reduce the cost of their development. However, ACs are often prepared for high-risk systems and applications, and the consequences of poor judgement or errors in the assurance process can be severe. Therefore, it is important to consider both the potential risks and benefits of using LLMs to support AC development. To this end, the contribution of this paper is four-fold. First, the paper surveys the literature on the use of LLMs to support ACs and identifies four LLM use cases. Second, this paper suggests seven additional novel LLM use cases. Third, this paper proposes a method for assessing the risk-benefit trade-off of a specific LLM use case. Finally, the paper applies the proposed method to the identified use cases, which demonstrates the applicability of the method and provides an opportunity to evaluate the benefits and consequences associated with each use case.

Keywords: Safety Cases· Assurance Cases· Large Language Models· Generative AI· Risk Analysis

1 Introduction

Assurance Cases (ACs) are engineering artifacts that are produced to demonstrate that a system has satisfied one or more essential quality attributes such as safety (a "safety case") or cybersecurity (a "security case"). ACs consist of two fundamental parts: 1) a structured argument describing the rationale for why a system possesses the desired quality attribute(s), and 2) supporting evidence [2]. Various approaches exist for structuring assurance arguments, including graphical notations such as Goal Structuring Notation (GSN) [17], Claims-Argument-Evidence (CAE) [4], and Eliminative Argumentation (EA) [10], and textual notations such as the Friendly Argument Notation (FAN) [13]. Regardless of the approach taken, it is important that the AC argument be expressed in a clear manner that can be understood by a wide range of interest holders, including engineers, safety assessors/auditors, and business decision makers.

Preparing an AC is required for compliance with numerous engineering standards and regulations in the automotive, nuclear, defense, rail, and oil & gas industries. Of

note is the recently released ISO/PAS 8800 – *Road vehicles – Safety and artificial intelligence* standard, which requires the preparation of a safety argument for automotive technology that depends on AI for safety-critical applications [15]. Additionally, there is interest in ACs in the aerospace and medical industry [6, 1]. The continued, and increasing, use of ACs is consistent with the view that modern systems require goal-oriented assurance, in addition to engineering rigor in the development process.

Given the importance of ACs, there is value in developing methods and tools that can improve the quality (correctness, completeness, etc.) of ACs and reduce the effort required to produce and maintain them. Indeed, preparing and managing ACs is a demanding endeavor, requiring technical expertise, argumentation skill, reviewing large amounts of evidence, and the application of engineering judgment. Consider also that ACs have lifetimes that start during early concept development and extend through systems operation [12, 7], and that the case will likely be revised over time to account for changes to the system's design or operation. Without appropriate tool support, accounting for such changes is also a challenging undertaking.

Many methods and tools have focused on improving the quality of ACs and efficiency of their preparation, both for initial development and addressing the impact of changes. These include model-based methods that integrate with other assurance activities, formal methods, qualitative and quantitative confidence assessments, notations and reasoning methodologies, and tool support for automating various aspects of AC management. Broadly speaking, these methods depend on having structured data for the AC and related items such as the argument structure. However, natural language is still used to describe detailed reasoning, even if notations like GSN, CAE, FAN, or EA are used to describe the structure of the argument. The approaches described above struggle to cross the "natural language barrier" to examine the meaning of each element of an argument (e.g., the textual contents of the "boxes" in a GSN diagram). Thus, to realize the potential of many of the approaches outlined above, users are required to build models or otherwise expend significant effort to input data in a format that enables analysis functions. At scale, this limits the type of automation that can be applied to support the development and management of ACs in an industrial setting.

Large Language Models (LLMs) became widely available in late 2022 and have demonstrated remarkable capabilities for processing and generating unstructured natural language text [24]. They have since been used in a wide range of applications, including performing specific tasks related to AC synthesis and analysis [5, 9, 18, 21–23, 27]. With LLMs, methods and tools can be developed that break through the natural language barrier and result in higher quality ACs that can be created or maintained more efficiently.

LLMs are a complex and, at times, controversial technology. They pose inherent risks that must be considered, especially in matters of assurance. Further, they are limited by their nature as "stochastic parrots" [3], with their fundamental function being to generate the next most plausible token in a sentence. They are prone to hallucinations, where they generate outputs that are plausible, but factually incorrect [16]. So, while LLMs enable new forms of natural language processing that may have significant benefits in terms of quality and efficiency, they also pose risk with errors having potentially

significant consequences. It is important to understand the risk versus benefit trade-off being made when using LLMs to prepare or analyze ACs.

Overview of Contribution

Overall, as a way of framing the contribution of this paper, our position is that LLMs hold significant potential to improve the state-of-the-art for AC development and management in ways that improve the overall quality of ACs while also enabling new forms of automation that reduce effort. We view LLMs as a supportive technology that allows users to (re-)focus their efforts on value-adding activities such as critically engaging with AC arguments and evidence and effectively communicating to interest holders. However, since there is risk associated with the application of LLMs to ACs, we do not advocate wholesale application of this technology. Rather, our objective in this paper is to provide researchers and practitioners with a method to assist them in making informed decisions about whether using LLMs in their AC practice is acceptable for their specific use cases and context.

To this end, this paper makes four contributions. First, it reviews published literature focused on using LLMs to support AC development or management and identifies use cases that have been considered by researchers in this field. Second, it proposes several novel use cases for applying LLMs to ACs. Third, it proposes a method for systematically evaluating the risk versus benefit trade-off of using an LLM to perform a specific task for an AC. Importantly, the proposed method is both flexible and applicable in a range of settings, and simple to use to rapidly determine which use case(s) are acceptable for practitioners or researchers who are considering incorporating LLMs as part of their AC practice. Finally, to both illustrate and validate the method, this paper applies the proposed method to the use cases from the literature and our novel use cases.

2 Use Cases for LLMs

This section describes a combination of previously published and novel use cases for LLMs for the development or management ACs. Existing use cases were identified through a literature review. Novel use cases are proposed based on the authors' industrial and research experiences with AC and LLMs.

2.1 Existing Use Cases for LLMs in ACs

We performed several literature searches using databases (Compendex, IEEE Xplore) to identify an initial corpus of papers that describe topics related to ACs and LLMs. Our search focused on titles, abstracts, and keywords and used search strings that combined common AC terminology ("assurance case", "safety case", "goal structuring notation", etc.) with terms used to describe LLMs ("generative ai", "large language model", etc.). We limited our search to papers published after 2017, since LLMs were first introduced in June 2017 [24]. In total our search returned 25 prospective papers, 20 of which remained after removal of duplicate search results. These 20 were fully reviewed. We excluded papers that did not describe an application of LLMs to support the development

or management of ACs in some way. Also, some papers were repeated publications of the same results, which we removed. The final corpus for our review consisted of 11 papers which collectively describe four use cases.

Use Case: Defeater Generation. Defeaters (also referred to as "challengers" or "counterclaims") were introduced by Goodenough et al. as a way of increasing confidence in an AC and are now widely used to mitigate confirmation bias and drive critical thinking about AC arguments or evidence [10]. In their initial proposal to use LLMs to generate defeaters Viger et al. point out that the nature of defeaters as doubts in AC means that the consequence of worrisome hallucinations is greatly diminished, but the benefits are significant, such as detecting incorrect or incomplete reasoning in an AC [26]. Viger et al. proposed the AI-Supported Eliminative Argumentation (AEA) framework for using an LLM to generate defeaters, and they showed that 25% of defeaters generated by an LLM represent reasonable doubts in an argument [27]. Gohar et al. proposed the CoDefeater method, which similarly uses an LLM to generate defeaters, and they also found that LLMs can perform zero-shot defeater generation and that they can be used to support the practitioners in finding novel (i.e., previously unidentified) defeaters [9]. However, Gohar et al. found LLMs might still struggle to identify some types of defeaters that require advanced reasoning over implicit information. Shahandashti et al. used GPT-4 Turbo to generate defeaters for AC arguments and found that the GPT-4 Turbo LLM can produce "moderately reasonable" defeaters [21]. Of particular interest is the use of Chain-of-Thought prompting by Shahandashti et al. to guide the reasoning of an LLMs, which they found improved performance when comparing their results to ground-truth defeaters. Finally, Ghahremani et al. also suggest using LLMs to generate defeaters as part of preparing an AC for a safety management system [8].

Use Case: Argument Synthesis. Several authors have explored the possibility of using an LLM to synthesize AC argument structures based on a range of information sources. Sivakumar et al. conducted an experiment where they prompted an LLM with descriptions of a system, the argument's top-level objective, and descriptions of available evidence [22]. They compared the generated ACs with ground truth ACs and found that LLMs have a moderate capability to perform this task. Odu et al. also generated AC argument structures, but used AC argument patterns/templates to guide the LLM, which they expressed as logical predicates [18, 19]. They found that patterns helped the LLM produce "relatively good" ACs compared to a set of ground-truth ACs; however, they also noted the LLMs occasionally missed relationships in the provided pattern structures. Chen et al. proposed the Trusta method, which generates argument structures using an LLM using the CAE "building blocks" to guide the LLM to perform rational argument decompositions [5]. They then subject the generated arguments to formal proof to check correctness (see next use case).

Use Case: Argument Formalization. Before LLMs were available, researchers formalized argument structures and then subjected them to automatic correctness proofs using tools such as theorem provers. However, such approaches were limited in that they could not automatically translate the natural language text from the argument into proof inputs. Since the advent of LLMs, two groups have published results where LLMs are

used to translate natural language aspects of arguments into formal constraints that can be provided to theorem provers. Chen et al.'s Trusta method includes a step where an LLM optionally translates natural language sentences within GSN argument nodes into formal constraints that can be checked by the Z3Prover [5]. Their results show that LLMs can perform these translations if the text in the argument nodes is unambiguous, but that human assistance is required where the contents of the nodes are ambiguous. Chen et al. used their proof results to find and close gaps in argument structures. Similarly, Varadarajan et al. used an LLM as part of their CLARRISA approach to assist with the automated evaluation of AC arguments in an activity they refer to as "Semantic Analysis" [23]. Their analysis is performed in two steps: 1) an LLM assists with the translation of natural language text from each node in the argument into Object-Property-Environment tuples, and 2) using a conventional software procedure to translate the argument (with the tuples) into a Prolog program which is run to check correctness properties of the argument.

Use Case: Change Impact Assessment. The assurance rationale for many systems is complex, often resulting in large AC arguments that are difficult to maintain. It can be difficult to assess the impact of changes on an AC, such as when a system's design is changed. Viger et al. have suggested that LLMs could be used to support AC change impact assessment using a four step procedure: 1) provide the LLM system background information and the AC argument structure, 2) provide the LLM a description of the changes, 3) ask the LLM to identify the impacted nodes in the argument, and 4) a human reviews the identified nodes and applies the appropriate change [25].

2.2 Novel Use Cases for LLMs in ACs

In addition to the use cases in literature, we propose several novel use cases of LLMs to support the development or management of ACs. These use cases are based on our practical experience developing ACs for real-world systems, considering how LLMs might contribute to this activity; this is not an exhaustive list of use cases. These are briefly introduced below and are then analyzed using the method proposed in Section 4.

Use Case: Compliance Matching. ACs are often prepared for systems that are subjected to regulations or technical standards, and an AC can be used to demonstrate compliance requirements have been satisfied by linking relevant requirements directly to the part of the AC that addresses them. In this proposed use case, an LLM is provided with the full list of compliance requirements and a specific node (or collection of nodes) in the argument. The LLM outputs suggested compliance requirements for the node(s).

Use Case: Text Rephrasing. The natural language text of an AC, such as the text that appears in a GSN Goal node, is typically written by human authors. When multiple authors are contributing to an AC, this can introduce variability in the argument's expression. For example, some authors might be more forceful/direct in their assertions, while others might adopt a more passive voice. In this use case, an LLM is given the text of a node and asked to rephrase to have certain properties related to tone, length, or terminology. The LLM outputs a rephrased version of the node's text.

Use Case: Concept Searching. As ACs become large or complex, especially if they are authored or maintained by multiple contributors, it can be difficult to find all of the content in an AC that is related to a specific topic. Keyword searches are inadequate if the terminology in the AC is variable. For example, finding all of the information in the AC related to "software unit verification" might require searching for words such as "unit test", "inspection", "code review", etc. For this proposed use case, the LLM is provided with the entire AC as input along with a short phrase describing a concept to search for in the argument. The LLM outputs a list of nodes that address this concept.

Use Case: Summary Generation. When communicating with non-expert interest holders (e.g., business management), it is often necessary to summarize the contents of the AC in a narrative form. The summary might cover the entire AC or a fragment addressing a specific topic. In this use case, an LLM is provided with the AC as input (either in full or in part). The LLM outputs a summary of an argument and evidence.

Use Case: Reverse Engineering. Within a larger AC argument, there are typically small argument fragments that perform reasoning steps over specific pieces of evidence as a way of connecting them to higher-level claims in the argument. In this proposed use case, the LLM is provided with a description of the available evidence. The LLM outputs an argument structure that makes use of the evidence, and which can be connected to a higher-level claim in the argument.

Use Case: Confidence Assessment. Many methods exist for quantifying confidence in ACs. However, these methods are laborious to apply, often requiring users to provide several inputs per node in the argument. In this proposed use case, an LLM is provided with two inputs: a small fragment of an argument and a description of a confidence assessment method. The LLM is asked to suggest a confidence score for the top-level claim in the argument fragment, by applying the method to the structure. The result is returned to a human user for review.

3 Method for Evaluating LLM Use Cases

This section proposes a novel method to analyze the potential consequences and benefits of LLM use cases. In this context, "benefits" are positive outcomes where an LLM adds value to an activity. For example, using an LLM might help users produce higher quality ACs in terms of completeness by suggesting defeaters. Conversely, "consequences" refer to negative outcomes, where using an LLM might have an adverse effect. For example, if used in an unchecked manner, an LLM might introduce factual errors into an AC that degrade its quality.

Four principles were used to guide the development of the method. They serve as high-level (non-functional) requirements for the method:

1. **Familiarity.** To allow the method to be easily adopted by practitioners or researchers, the method should resemble existing methods that are familiar to this group.

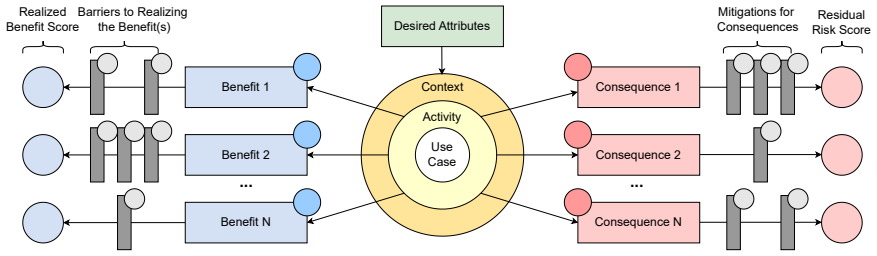

Fig. 1. Generic bowtie for assessing benefits and consequences of an LLM use case.

2. **Flexibility.** The method should be applicable across a range of different settings, that vary across, the tasks performed on the AC, different roles (e.g., reviewer, assessor), capabilities for interacting with LLMs, and organizational contexts.
3. **Simplicity.** The method should be simple enough such that it can be easily applied "by hand" to rapidly determine an LLM use case is reasonable: scoring should use whole numbers or fractions, and that it should be possible to quickly document the results in a graphical or tabular manner.
4. **Collaborative.** Since ACs are usually developed by organizations and the decision to use LLMs is often taken at the organizational level, the method should facilitate collaboration between different groups with varying objectives and opinions.

3.1 Description of Method

The proposed method is based on the Bowtie risk assessment method, which describes how conditions in a system's environment (sometimes called "threats"), in combination with the occurrence of a hazardous event, can precipitate into losses (sometimes called "consequences") [20]. The analysis is visualized bowtie-shaped diagram with the left side of the bowtie describing threats, the center containing a hazardous event, and the right side representing the consequences of the hazard occurring. Barriers are included that either prevent threat conditions or mitigate consequences.

Towards Principle #1 (Familiarity), the proposed method for evaluating LLM use cases modifies the bowtie method as shown in Fig. 1 The LLM use case is shown in the center of the bowtie. The left side depicts potential benefits and barriers that might prevent the benefits from being realized. The right side of the modified bowtie depicts potential consequences of the use case, with mitigations that might limit the risk associated with each consequence. The periphery contains scores for the realized benefit (after accounting for barriers) or residual risk (after accounting for mitigations).

Step 1: Establish Desired Attributes. The first step is to define the overall objectives or concerns related to using an LLM to support the AC. These objectives and concerns are expressed as "Desired (Quality) Attributes" and may include attributes of the AC itself, such as completeness, correctness, and soundness; desired aspects of the process, such as saving time during review or improving communication; and broader concerns,

such as legal or ethical issues like data privacy or regulatory compliance. The Desired Attributes are used to frame all subsequent analysis, especially the enumeration of benefits and consequences in Steps 3 and 4 of the method. They should be expressed as short phrases and used to annotate the modified bowtie diagram as shown in Fig. 1.

At this point, it is also important to consider the role of the AC within the organization. As pointed out by Graydon, there are various purposes for developing an AC, such as showing compliance, organizing evidence, or assessing confidence in claims [11]. For instance, if an AC is to be publicly disclosed as part of a campaign to build trust for a system, then the Desired Attributes might emphasize communication with a non-technical audience or legal/ethical concerns, as opposed to including extensive technical argumentation about a system.

Step 2: Describe LLM Use Cases, Activities, and Context. The second step is to define the LLM use case, the activity being performed, and the larger context for the AC, which are depicted in the center of the bowtie, as in Fig. 1.

The LLM use case should be described in a few sentences at a conceptual level with a focus on the inputs to the LLM, the task(s) being performed by the LLM, and the expected outputs. The use case should assume the LLM provides idealized output and should avoid describing technical details, such as prompting strategies. In the defeater generation example, the use case might be described as follows: *The LLM will be used to suggest potential defeaters for a selected claim in the AC. The input provided to the LLM will be the selected claim and additional contextual information (e.g., context or assumption nodes, the parent claim, or neighoubring claims). The output returned by the LLM is a list of candidate defeaters for the selected claim.*

Next, the activity being performed on the AC by the (human) user of the LLM should be defined. Examples of activities include developing (i.e., authoring) the AC, reviewing evidence, reviewing the argument, performing a change impact analysis, assessing the case for compliance, and so on. Importantly, the activity being performed will influence the potential benefits and consequences of using an LLM and will also impact which barriers or mitigations are reasonable. For the defeater generation example above, the potential benefits of the LLM suggesting defeaters during AC development are different from the benefits during AC maintenance.

Finally, the larger context of the AC should be described, focusing on factors that might impact the LLM use case. Such factors might be technical or organizational. For instance, some organizations might have in-house LLMs that have been fine-tuned for a corpus of topics related to the system in question, which might improve the performance of the LLM for the specific task. Contextual factors might impact the scoring, or affect the barriers and mitigations that are available.

In performing this step, it is likely that many combinations of use cases, activities, and context will be identified. Each combination should be analyzed separately using this method, i.e., a separate bowtie for each combination.

Step 3: Enumerate Benefits and Barriers. In the third step potential benefits and barriers are listed. Benefits should refer to the stated LLM use case, activity, and relevant contextual factors. Further, benefits should be framed in terms of the desired attributes

established in Step 1. For defeater generation, for a review activity, a potential benefit is: *The defeaters suggested by the LLM might not have been considered by a reviewer, thus improving the completeness of the review.* Barriers should express reasons that a stated benefit might not be realized, either in part or in full. Though they are not limited to these topics, barriers often describe limitations of an LLM's performance (e.g., hallucinations) or the capabilities of human user(s) to interact with the LLM. A barrier for the benefit above is: *The reviewer might not have sufficient system context and discards genuinely novel defeaters suggested by the LLM.*

Step 4: Enumerate Consequences and Mitigations. In the fourth step, consequences and mitigations are listed. Consequences describe potential negative outcomes from using an LLM, and should refer to the stated use case, activity, and relevant contextual factors. Further, consequences should be framed in terms of the impact on desired attributes established in Step 1. For defeater generation, for a review activity, a consequence is: *When using the LLM to suggest defeaters, a reviewer becomes overconfident in the completeness of the AC and falsely concludes that the AC is adequate.* Mitigations should express actions or measures that are intended to reduce the risk associated with a consequence. Mitigations might be technical, social, or organizational in nature. Technical mitigations might adjust how the LLM is prompted or how the output of the LLM is handled by software. Social or organizational mitigations might require that special procedures be performed by users. For defeater generation, a technical mitigation might be: In the prompt, provide the LLM with a list of defeater categories to increase the likelihood that important topics are covered.

Step 5: Scoring (Optional). Steps 1 through 4 give a qualitative analysis of the benefits and consequences for a use case. For some, this level of analysis might be adequate, especially if the objective is to determine mitigations to enable the use case. However, a scoring procedure can be applied as a fifth step to deepen the analysis. To improve the simplicity of the method (Principle #3), the scoring uses integers and simple fractions, which can be combined with basic multiplication and addition such that a user of the method could quickly compute ("on paper"). This procedure also enables collaborative decision making, by allowing interest holders to articulate their opinions numerically (Principle #4). First, the scoring procedure for a single rater is described, and then it is generalized to multiple raters.

Scoring Potential Benefits and Consequences. An integer score from 0 to 3 may be assigned to each benefit and consequence, without considering the barriers or mitigations. For benefits, a score of 0 means there is no appreciable benefit and 3 means there is significant benefit in terms of the desired attributes. Similarly, for consequences, a score of 0 means there is no risk to the consequence and a score of 3 means the risk is significant. The scoring criteria should be adjusted to account for the desired attributes, optionally expressing a priority between attributes. For example, if there are two attributes (e.g., completeness, and effort), the scoring criteria can prioritizparagraphe one, as in Table 1.

Table 1. Sample scoring criteria for Benefits and Consequences when desired attributes are completeness of the AC and effort required to develop the AC.

Score	Benefit	Consequence
0	No benefit in terms of completeness or effort.	No impact on completeness or effort.
1	Reduces effort required to develop the AC.	Increases effort required to develop the AC.
2	Improves completeness of the AC.	Reduces completeness of the AC.
3	Improves completeness and reduces effort.	Reduces completeness and increases effort.

Scoring Barriers and Mitigations. In this part of the scoring procedure, fractional scores (0, $\frac{1}{4}$, $\frac{1}{2}$, $\frac{3}{4}$, 1) are assigned to each barrier and mitigation to describe its reduction on the respective consequence or benefit, i.e., as a "discount factor" describing effectiveness or impact. For barriers, a score of 0 means the barrier does not reduce the benefit and a score of 1 means the barrier fully negates the benefit. For mitigations, a score of 0 means the mitigation does not reduce the risk associated with the consequence and a score of 1 means the mitigation fully mitigates the risk. Fractional scores express partial effect: $\frac{1}{4}$ is a minor reduction, $\frac{1}{2}$ is a moderate reduction, and $\frac{3}{4}$ is a major reduction.

Tabulating Scores. The overall "net score" is computed as the difference between the realized benefit and residual risk scores. The realized benefit is the sum of all individual realized benefit scores (product of the benefit score and the related barrier scores). Similarly, the residual risk score is the sum of all residual risk scores for each consequence (product of the consequence score and the related mitigation scores):

$$\Big[\sum_{\substack{b_i \in \\ \text{benefits}}} (b_i \cdot \prod_{\substack{r_{i,k} \in \\ \text{barriers}}} 1 - r_{i,k})\Big] - \Big[\sum_{\substack{c_j \in \\ \text{conseq.}}} (c_j \cdot \prod_{\substack{m_{j,l} \in \\ \text{mit.}}} 1 - m_{j,l})\Big] \tag{1}$$

For example, consider a use case with two benefits, each with one barrier: $b_1 = 2$ ($r_{1,1} = \frac{1}{2}$) and $b_2 = 3$ ($r_{2,1} = \frac{3}{4}$), and then one consequence with $c_1 = 3$ with two mitigations $m_{1,1} = \frac{1}{4}$ and $m_{1,2} = \frac{1}{4}$. Then the realized benefit is $2 \cdot (1 - \frac{1}{2}) + 3 \cdot (1 - \frac{3}{4}) = 2 \cdot \frac{1}{2} + 3 \cdot \frac{1}{4} = 1 + \frac{3}{4} = 1.75$ and the residual risk is $3 \cdot (1 - \frac{1}{4}) \cdot (1 - \frac{1}{4}) = 3 \cdot \frac{3}{4} \cdot \frac{3}{4} = \frac{27}{16} = 1\frac{11}{16} = 1.6875$. Then the net score is $1.75 - 1.6875 = 0.0625 = \frac{1}{16}$.

If multiple raters are participating in scoring, then the procedure may be amended as follows: 1) each rater scores all benefits, barriers, consequences, and mitigations independently, 2) the raters meet to discuss differences and agree on a new score, if required, and 3) compute the net score from the agreed scores.

Step 6: Decision. Finally, a decision should be made about whether the LLM use case is acceptable for the activity, if the stated mitigations are implemented. If scoring was used, then a threshold can be set for the net score. The default threshold is naturally a net score of zero. Alternative thresholds might be set to indicate varying risk tolerances: an organization with higher risk tolerance might accept any net score of at least -1.

Table 2. Analysis of results for LLM use cases.

Use Case	Activity	# Ben.	# Barr.	# Conseq.	# Mit.	Net.
Defeater Generation	Develop AC	3	6	3	10	2.56
Argument Synthesis	Develop AC	2	3	3	11	1.46
Argument Formalization	Develop AC	3	4	3	8	1.12
Change Impact Assessment	Maintain AC	4	6	4	11	1.12
Compliance Matching	Review AC	3	5	4	10	0.88
Text Rephrasing	Develop AC	2	5	3	5	0.33
Concept Search	Review AC	3	6	2	7	-0.02
Summary Generation	Assess AC	4	4	3	11	1.26
Reverse Engineering	Develop AC	3	7	2	10	2.21
Confidence Assessment	Develop AC	4	6	5	8	-2.58

4 Analysis of LLM Use Cases

This section applies the analysis method described in Section 3 to the ten LLM use cases from both the literature and the novel use cases in Section 3. The analysis results are summarized Table 2. Each use case was analyzed from the perspective of a specific activity. For brevity, the Defeater Generation use case is analyzed in detail below. The analysis results combined scores from three authors (N. Anwari, O. Foster, and L. Millet), all of whom have contributed to ACs for real-world systems. The analysis was performed for two desired attributes: *AC Quality* and *Level of Effort*.

4.1 Use Case Analysis: Defeater Generation

In the Defeater Generation use case, an LLM suggests potential defeaters for claim(s) in the AC. The input to the LLM is the claim(s), its parent and neighbors, and additional assumptions or context. The output returned by the LLM is a list of candidate defeaters for the selected claim(s). The use case was analyzed in terms of an initial AC development activity in the context of a small team preparing an AC for a complex system. The resulting bowtie diagram is shown in Fig. 2.

On the left side, three potential benefits are considered: 1) the suggested defeaters might help improve the quality of the argument as the developers prepare counter-arguments; 2) the level of (system) risk awareness increases if new defeaters relating to critical defects in the system are discovered; and 3) as changes are introduced to the AC during development, new defeaters can be generated to help analyze the changes. Several barriers to these benefits are considered: none eliminate a benefit, but they significantly reduce the overall realized benefit. For example, for the first benefit, three barriers consider the impact that limited knowledge of the LLM or reviewer might have. The realized benefit is computed as $\frac{221}{64}$.

On the right side, four potential consequences are considered: 1) long-term use of the LLM might reduce critical thinking abilities; 2) the presence of many defeaters might make the argument too complex; 3) some team members might resist use of the LLM; and 4) the argument becomes incomplete or biased. Several mitigations are provided for these potential consequences. Many of the mitigations describe specific

user-LLM interactions where the user critically engages with the LLM, which increases the likelihood of detecting problems and reduces complacency. However, human-led reviews of LLM generated content are imperfect, a fact we reflect in our scoring for mitigations. Using a systematic method, like the method used by Viger et al. in [27] which asked users to consider specific criteria for each defeater during the review, can improve the rigour of a review. Another point of interest is that the third consequence (about adoption), is scored as 0 risk in terms of the desired attributes of AC Quality and Effort. If another attribute was considered, such as "increased communication in the team", then perhaps this would score differently. The residual risk is computed as $\frac{57}{64}$.

Combining the two scores, the net score for Defeater Generation is $\frac{221}{64} - \frac{57}{64} = \frac{164}{64} \approx 2.5625$ points, which indicates there is a net benefit to using an LLM to suggest defeaters during an initial AC development activity in a small team preparing an AC for a complex system, if the identified mitigations are implemented.

5 Discussion

LLM's have significant potential to support the development and management of ACs, throughout the systems engineering lifecycle. Researchers have begun exploring their use for several use cases, and this paper suggested several additional use cases. However, using LLMs also introduces risk that must be considered alongside the potential benefits. The aim of this paper is not to advocate for any single use case of LLMs or advocate that a specific use case is ultimately "good" or "bad". Instead, the objective is to equip researchers and practitioners with the tools to systematically consider risk-benefit trade-off and make informed decisions about their use of LLMs within their specific context. To this end, this paper presents a modified bowtie analysis method to consider benefits and consequences of LLM use cases. Our proposed method accounts for the benefits of LLM use, potential barriers to realizing the benefits, consequences of LLM use, and mitigating factors that might be employed. The method is also flexible enough to consider a variety of LLM use cases and activities related to AC development or management. The method includes an optional scoring step that allows users

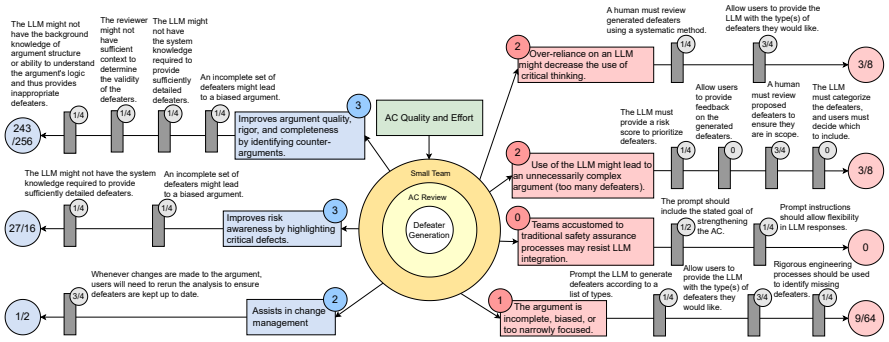

Fig. 2. Bowtie diagram from defeater generation LLM use case.

to weight the benefits and consequences (and barriers and mitigations) and compute an overall "net benefit" score for the use case. To demonstrate the applicability of the method, we applied it to ten use cases, four of which were identified from the literature and seven of which are novel use cases that have, to our knowledge, not been published.

Observations on Scoring and Mitigations. The majority of LLM use cases in Table 2 received positive scores, suggesting that, if the mitigations are properly implemented, then the realized benefits likely outweigh the residual risk. One might take this result to mean that most use cases are in fact "good ideas". However, our interpretation is different: enumerating mitigations helps to enable use cases that would otherwise be untenable. The same idea is foundational to functional safety engineering, where control measures are added to reduce the level of risk such that the benefits of the system may be realized.

Recurring Analysis Themes. During our analysis of the LLM use cases, we identified recurring consequences that are worth considering. First, many consequences relate to the LLM omitting details that are often important in ACs. For example, in the Text Rephrasing use case, one consequence was that "rephrasing could introduce some bias by oversimplifying the intent of the node." Mitigations for these consequences require the supervising human to be attentive to details in some way (e.g., reviewing the output). Second, some of the identified consequences relate to "chronic use" of LLMs, instead of the immediate consequences of usage. For instance, in the Defeater Generation use case, a consequence was: "over-reliance on this feature might decrease use of critical thinking". As this method is further validated and eventually applied in practice, recurring analysis themes for both benefits and consequences might emerge. Overall, there is an opportunity to develop a list of common benefits and consequences for using LLMs, like existing lists of failure modes for technical failure analyses.

Limitations. This paper has proposed a new method of assessing risk for using LLMs for ACs. The application of the method to several use cases, including those published by other authors, shows that the method is applicable to this problem. However, more comprehensive validation is still necessary. One challenge with further validation is the lack of an obvious "ground truth" compare against. It is not clear what constitutes a "correct" result, so one cannot easily demonstrate that this method produces correct results. Alternatively, the repeatability of the method could be assessed by having different users apply the method to the same use cases; with the ideal outcome being that the results between different users are similar.

Future Work. Further validation of the proposed method (e.g., repeatability) is one avenue for future work. Another is to find other areas where LLMs are being used where the consequences of incorrect or incomplete work are substantial and then generalize the proposed method to those areas. Additionally, there might be benefit in developing common lists of LLM benefits and consequences, to aid analysis applying our method.

Yet another avenue is to explore an iterative application of the proposed approach, possibly including small experiments with an LLM, to validate the proposed mitigations and justify the scores for mitigations and barriers. Similarly, additional work could be undertaken to develop more sophisticated score aggregation methods, especially for large groups of interest holders.

Finally, the emerging area of "Safe LLMs" has motivational overlaps with this work, see for instance work on trustworthy LLM-based agents by Hua et al. [14]. It is likely that mitigation strategies used by authors working in this area could also be applied to LLMs for ACs.

Closing Remarks. In the spirit of advancing the state-of-the-art in applying LLMs to ACs risk-aware manner, we suggest that other researchers working in this field undertake some risk-benefit analysis (using our proposed method or otherwise) as part of research or industrial projects they undertake in this area.

Disclosure of Interests. All the authors are currently, or were recently, employed by Critical Systems Labs Inc., a Canadian firm that provides engineering services for high-risk systems and produces *Socrates - Assurance Case Editor*, a commercial product for preparing assurance cases.

References

1. Association for the Advancement of Medical Instrumentation: Medical device safety assurance case guidance (2019)
2. Assurance Case Working Group: Goal Structuring Notation Community Standard (Version 3). Tech. rep., Safety Critical Systems Club (2021)
3. Bender, E.M., Gebru, T., McMillan-Major, A., Shmitchell, S.: On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In: Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency. pp. 610–623. Association for Computing Machinery (2021)
4. Bishop, P., Bloomfield, R.: A Methodology for Safety Case Development. In: Industrial Perspectives on System Safety. Springer (1998)
5. Chen, Z., Deng, Y., Du, W.: Trusta: Reasoning about assurance cases with formal methods and large language models. Science of Computer Programming **244**, 103288 (Sep 2025)
6. Daw, Z., Beecher, S., Holloway, M., Graydon, M.: Overarching Properties as means of compliance: An industrial case study. In: 2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC). pp. 1–10 (2021)
7. Fenn, J., Hawkins, R., Nicholson, M.: A New Approach to Creating Clear Operational Safety Arguments. In: Computer Safety, Reliability, and Security. SAFECOMP 2024 Workshops. pp. 227–238. Springer (2024)
8. Ghahremani, E., Joyce, J., Lechner, S.: Avoiding Confirmation Bias in a Safety Management System (SMS). In: 2024 Integrated Communications, Navigation and Surveillance Conference (ICNS). pp. 1–11 (2024)
9. Gohar, U., Hunter, M.C., Lutz, R.R., Cohen, M.B.: CoDefeater: Using LLMs To Find Defeaters in Assurance Cases. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. pp. 2262–2267. ASE '24, Association for Computing Machinery (2024)
10. Goodenough, J.B., Weinstock, C.B., Klein, A.Z.: Eliminative Argumentation: A Basis for Arguing Confidence in System Properties. Tech. rep., Software Engineering Institute (2015)
11. Graydon, P.J.: The Many Conflicting Visions of 'Safety Case'. In: 47th International Conference on Dependable Systems and Networks Workshops. pp. 103–104 (2017)
12. Hobbs, C., Diemert, S., Joyce, J.: Driving the Development Process from the Safety Case. In: Safe AI Systems. Safety-Critical Systems Club, Bristol, UK (2024)
13. Holloway, C.M.: The Friendly Argument Notation (FAN): 2023 Version. Tech. Rep. NASA/TM-20230004423, National Aeronautics and Space Administration (2023)

14. Hua, W., Yang, X., Jin, M., Li, Z., Cheng, W., Tang, R., Zhang, Y.: TrustAgent: Towards Safe and Trustworthy LLM-based Agents (2024), http://arxiv.org/abs/2402.01586
15. International Organization for Standardization: ISO/PAS 8800:2024 Road vehicles — Safety and artificial intelligence (2024)
16. Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y.J., Madotto, A., Fung, P.: Survey of Hallucination in Natural Language Generation. ACM Comput. Surv. **55**(12), 248:1–248:38 (2023)
17. Kelly, T.P.: Arguing safety - A Systematic Approach to Safety Case Management. Ph.D. thesis, University of York (1998)
18. Odu, O., Belle, A.B., Wang, S., Kpodjedo, S., Lethbridge, T.C., Hemmati, H.: Automatic instantiation of assurance cases from patterns using large language models. Journal of Systems and Software **222**, 112353 (2025)
19. Odu, O., Beltrán, D.M., Gutiérrez, E.B., Belle, A.B., Sherafat, M.: SmartGSN: a generative AI-powered online tool for the management of assurance cases (2024), http://arxiv.org/abs/2410.16675
20. de Ruijter, A., Guldenmund, F.: The bowtie method: A review. Safety Science **88**, 211–218 (2016)
21. Shahandashti, K.K., Belle, A.B., Mohajer, M.M., Odu, O., Lethbridge, T.C., Hemmati, H., Wang, S.: Using GPT-4 Turbo to Automatically Identify Defeaters in Assurance Cases. In: 2024 IEEE 32nd International Requirements Engineering Conference Workshops (REW). pp. 46–56 (2024)
22. Sivakumar, M., Belle, A.B., Shan, J., Khakzad Shahandashti, K.: Prompting GPT-4 to support automatic safety case generation. Expert Systems with Applications **255**, 124653 (2024)
23. Varadarajan, S., Bloomfield, R., Rushby, J., Gupta, G., Murugesan, A., Stroud, R., Netkachova, K., Wong, I.H., Arias, J.: Enabling Theory-Based Continuous Assurance: A Coherent Approach with Semantics and Automated Synthesis. In: Computer Safety, Reliability, and Security. SAFECOMP 2024 Workshops. pp. 173–187. Springer Nature Switzerland (2024)
24. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.: Attention Is All You Need. In: Advances in Neural Information Processing Systems. Long Beach, CA, USA (2017)
25. Viger, T., Murphy, L., Diemert, S., Menghi, C., Chechik, M.: Supporting Change Impact Assessment with LLMs. In: 2024 IEEE 35th International Symposium on Software Reliability Engineering Workshops (ISSREW). pp. 203–204 (2024)
26. Viger, T., Murphy, L., Diemert, S., Menghi, C., Di, A., Chechik, M.: Supporting Assurance Case Development Using Generative AI. In: SAFECOMP 2023, Position Papers (2023)
27. Viger, T., Murphy, L., Diemert, S., Menghi, C., Joyce, J., Di Sandro, A., Chechik, M.: AI-Supported Eliminative Argumentation: Practical Experience Generating Defeaters to Increase Confidence in Assurance Cases. In: 2024 IEEE 35th International Symposium on Software Reliability Engineering. pp. 284–294 (2024)