WRT-1028

VALIDATION FRAMEWORK FOR ASSURING ADAPTIVE AND LEARNING-ENABLED SYSTEMS

Principal Investigator:
Dr. Bryan Mesmer, The University of Alabama in Huntsville

Final Technical Report SERC-2021-TR-021
April 4, 2022

SYSTEMS ENGINEERING RESEARCH CENTER

The Networked National Resource to further systems research and its impact on issues of national and global significance

FINAL TECHNICAL REPORT: SERC-2021-TR-021

WRT- 1028

VALIDATION FRAMEWORK FOR ASSURING ADAPTIVE AND LEARNING-ENABLED SYSTEMS

Date: April 4, 2022

PRINCIPAL INVESTIGATOR: Dr. Bryan Mesmer, The University of Alabama in Huntsville

Sponsor(s): Office of the Under Secretary of Defense for Research & Engineering

DISCLAIMER

The Systems Engineering Research Center (SERC) is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology.

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Office of the Assistant Secretary of Defense for Research and Engineering (ASD(R&E)) under Contract [HQ0034-19-D-0003, TO#0671].

RESEARCH TEAM

Name	Org.	Labor Category
Bryan Mesmer	UAH	Principal Investigator
Daniel Shapiro	UAH	Research Scientist
Nicholaos Jones	UAH	Research Faculty
Jacob Smith	UAH	Graduate Research Assistant
Jennifer Stevens	NASA / MSFC	Doctoral Student
Jennifer Petrillo	UAH	Research Assistant II
James Walsh	Stevens	SERC Program Manager
Allison Moore	UAH	UAH Program Manager

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF (TABLES, SEQUENCES)

EXECUTIVE SUMMARY

Validation in Systems Engineering is the process of determining that an artifact will perform its intended task in its deployment environment. Unlike verification (which compares measured properties against written specifications) validation concerns abstract and prospective claims about systems that are not yet built, as well as stakeholder preferences for their performance. As engineers commonly pursue validation by stating, challenging, and refining claims about system properties, we employ Toulmin argumentation models to represent this discourse in terms of the premises, claims, and warrants (rationale) that support or refute them.

We focus on the practical benefits of this view and contribute four ideas that enable construction of a tool for managing validation arguments.

(1) We define a vocabulary of primitive argument types, organized into a type-hierarchy by inheritance of critical questions (constraints on the warrant's premises and conclusions). We claim that a handful of basic warrant types span the arguments found in systems validation and note that validation arguments commonly cite forms of trust (e.g., relying on expert opinion, delegating to a reputable firm, employing a vetted process).

Table 1 Warrant Types Summary

Warrant Type	Subtype	Subtype	Critical Questions
AGREEMENT			Is claim subject to agreement?
	Attestation		Is the claim knowable?
		Expert Opinion	Is the expert relevant to the claim?
		Common Knowledge	Does the rule apply in this context?
	Assumption		Is the claim reasonable, material & convenient?
	Declaration		Does the agent have the authority to assert the claim?
INSPECTION			Is the claim knowable via inspection?
	Demonstration		Is the demonstration representative of the use case?
	Test		Does the test address the claim? Are there defeating cases?
ANALYSIS			Is the claim subject to analysis?
	Analogy		Are the source and target environments, tasks, and systems sufficiently close?
	Modeling & Sim		Does the simulation address the claim? Are there defeating conditions?
	Inference		Is the inference cogent? Are there defeating facts?
CONSTRUCTION			Can the claim be made true by taking action?
	Procedure		Will executing the procedure produce the claim in this context?
	Organizational process		Will pursuing the organizational process produce the claim?
	Delegation		Are the principal and delegee value aligned?

(2) Given that warrants embed knowledge about the structure and content of validation arguments, we show that a warrant type-hierarchy supports a template-based editing model. We identify two main operations, for (a) identifying warrants that can support or refute specific aspects of the model, and (b) instantiating the elements of a warrant (its premises and conclusion) in a way that guarantees completeness and consistency of the result.

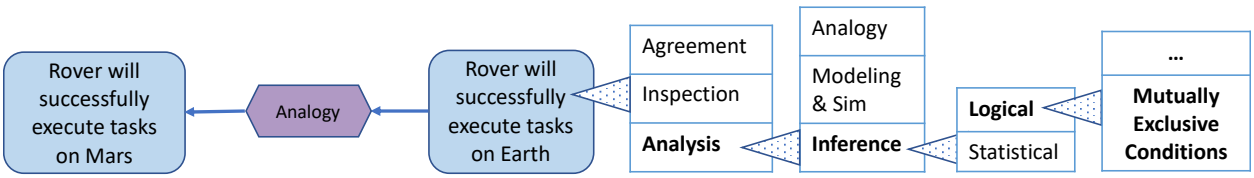


Figure 1 Warrant Type Example

(3) We provide a means of evaluating validation arguments into a joint distribution of beliefs. We employ a Bayesian interpretation that sums over the conditional probability of premises, conclusions, and the probability that each warrant is *apt* (meaning it is a relevant model for drawing conclusions in the current context). The probabilistic interpretation of warrants is new, as is the capability to resolve argument models into a distribution over beliefs (vs a single belief vector with associated confidence values).

(4) We extend our representation of validation arguments to include alternatives and uncertainties. This makes it possible to employ the full machinery of decision analysis to evaluate organizational choices in service of systems validation, in the presence of conflicting arguments about claims and their underlying rationale.

We illustrate this capability with a worked example that computes the value of conducting a physical test to resolve a conflict between a simulation result, and an expert who believes the simulation should not be trusted.

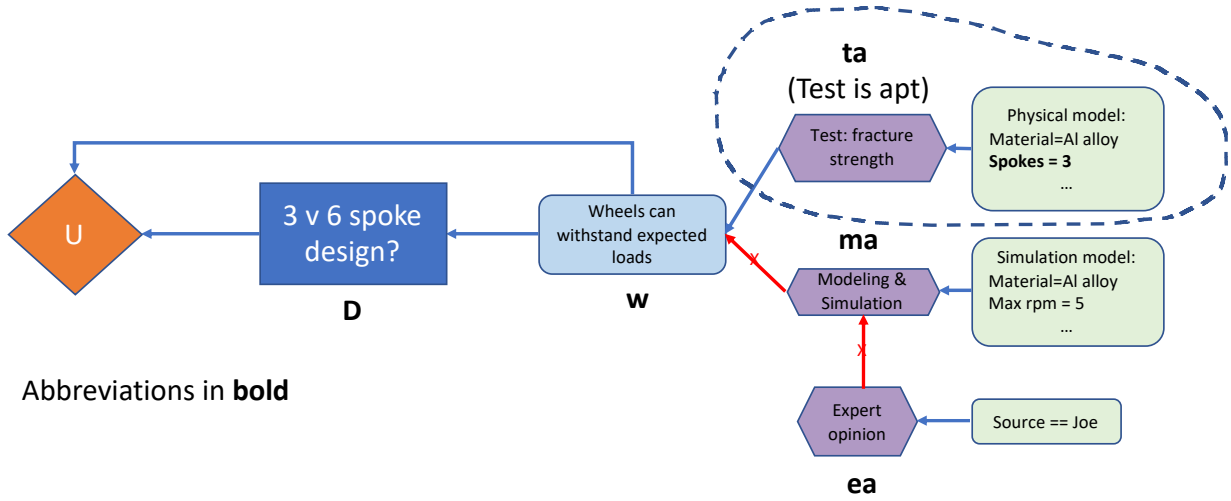


Figure 2 Worked Example

NEXT STEPS

Our work views systems validation as the iterative refinement of arguments for and against claims about a prospective artifact, made across the lifetime of a systems engineering project. In the process, we have provided the groundwork to develop a practical tool for managing systems validation arguments. We identify several next steps:

- Instantiate a larger portion of the warrant hierarchy. The resulting catalogue will demonstrate that a handful of warrant types cover systems validation arguments, clarify the arguments employed in systems engineering across project phases, and organize them into a functional form.

- Employ the warrant hierarchy to prototype a template-based editor for composing validation arguments. The resulting editor will ensure standards of completeness and consistency, while automating elements of validation argument construction.

- Demonstrate the benefits of unifying argumentation models with decision models. This work will illustrate decision analytic manipulations of validation arguments and identify systems validation activities where the combined machinery of these two models supplies new value.

- Design and develop a tool for constructing, evaluating, visualizing, and tracking validation arguments. This work will employ a model-based systems engineering framework to create a validation view into a repository of machine-readable system descriptions that ground its operations. The natural first step is a prototyping effort.

BACKGROUND

As stated in the most recent National Defense Strategy, success doesn't go to the countries that develop new technologies first. Rather, it goes to those that most rapidly integrate those technologies into their warfighting systems and change their way of fighting to take advantage of the new capabilities.

Modern systems are increasingly valuing adaptation and rapid capability deployment over other success attributes. With a continuous increases in software-enabled capabilities, the Department of Defense (DoD) is requiring the ability to provide updates to systems in the field, and is emphasizing continuous development and deployment practices for all DoD programs. This requirement responds to a continuously adapting threat in today's software-intensive systems, and the race to develop systems that use artificial intelligence and machine learning to rapidly adapt in the field. There is a pressing need for new methods and tools to continuously validate developed and deployed systems, and an emerging need to create methods and tools for validating autonomous systems, including those that learn from their experiences.

Previous research proposed a method for validating the spectrum of continuously adaptive and learning enabled systems through the use of a formal, logical argument method known as the Toulmin Argument Method. The concept of argument analysis has generally only been used in systems engineering by specialty engineering disciplines, such as safety and security. Use in those domains has more focused on the design of systems, and not validation. Traditional deterministic analyses and decomposition approaches often fail when addressing uncertainty and mathematically intractable state spaces. Argument analysis is well-suited to support decisions when faced with these conditions. The Toulmin Method could augment widely used SE methods focused on confirmation of static behaviors, by establishing a structured, logical argument that an evolving capability will succeed in the field. The Toulmin Method could be the basis for a formal argument process which could be codified in a user-guided toolset specifically designed to support validation of continually adaptive and learning enabled systems.

The Toulmin method establishes a language that provides a logical basis for validating behavior when evidence supports warrants that are key to supporting overall claims. The formal "soft" proofs of the Toulmin method show promise for validating the behavior of adaptive and learning systems in a rigorous, logical and consistent manner.

Initial research defined a methodology and process using Toulmin's work that promises to be more formal, hierarchical, and repeatable than current validation methods, based in the fundamental concept of validation as collection of knowledge to build confidence that a system enables a capability or creates a desired effect in the world. Confidence comes from an argument that the system will enable the capability and create the effect. This argument starts at the very beginning of concept development, e.g., why do we believe this design is a good idea, and grows throughout system definition and realization, becoming more compelling in each step. The structure of the argument is created as a part of initial feasibility studies to enable reasoning about system validation and provide grounds to justify or refute heuristics. Throughout development, the validation process

collects, adds and limits rebuttals, and builds a warrant that the system meets its defined capabilities, using evidence collected across the development process.

Validation in Systems Engineering is the process of determining that an artifact will perform its intended task in the world. It is a difficult assessment because it concerns abstract and prospective claims about artifacts, the interests of multiple stakeholders, and the performance of artifacts in environments that are often partially modeled and understood. We adopt the perspective that argumentation, vs proof, is the appropriate model for assessing system validity, and we introduce four ideas based on Toulmin argumentation theory that enable development of a software tool for managing validation arguments: a vocabulary of primitive argument types, a template-instantiation method for composing validation arguments, a mechanism for evaluating argument models into a joint probability over claims/beliefs, and a means of weighing decision options in the presence of the arguments and counterarguments commonly encountered in engineering activities. We conclude with a worked example clarifying the value of conducting a physical test in the context of a conflict between a simulation result, and an expert who asserts that the simulation is not valid. This example illustrates a more general integration of argument models with decision theory.

RESEARCH FINDINGS

1.1 OUTLINE

We begin by setting context; we distinguish validation from verification in systems engineering and clarify that validation activities occur from the inception of an engineering project through its design, implementation, and deployment.

Next, we introduce the structure of Toulmin argumentation models [1], and their reliance on *warrants,* which we equate to standards of proof that extend well beyond the notions of inference and test.

The bulk of this report concerns the four ideas we developed during the project, which we discuss in the order shown in Figure 3. We claim that a handful of basic warrant types are sufficient to capture the forms of argument commonly encountered in engineering practice, that warrants fall into a specialization hierarchy, and that this hierarchy enables a template-based instantiation model for composing (and refining) validation arguments.

In discussing the evaluation of argument models, we develop a probabilistic interpretation of warrants that is essential to our more general integration of argumentation models with decision theory.

We illustrate the power of this unification by providing a worked example, which computes the value of conducting a test in the presence of arguments for, and against claims, and the rationale for making them (as might be encountered in a systems validation context).

We examine related work at the boundary of argumentation theory and systems engineering, discussing application-motivated work on argument construction tools, argument evaluation, efforts to catalog arguments, and AI argumentation theory more broadly.

We conclude by outlining next steps that will apply our results to facilitate the systems engineering process; we propose to develop a software tool for composing, tracking, evaluating, and refining validation arguments in service of decisions throughout the lifecycle of complex engineered artifacts.

- Validation in Systems Engineering
- Validation argument structure
- Four ideas enabling an argumentation tool
  - A vocabulary of primitive argument types
  - Constructing validation arguments by template instantiation
  - Evaluating argument models into probabilities over beliefs
  - Adding uncertainties and decisions to argument models
- A worked example (the value of a test given conflicting arguments)
- Related work
- Next steps

Figure 3 Outline

1.2 VALIDATION VS VERIFICATION IN SYSTEMS ENGINEERING

Verification: the process of determining that an artifact meets its stated requirements

Validation: the process of determining that an artifact will perform its intended tasks in the world

Figure 4 V&V Definitions

While there are several formal definitions of validation and verification [2] [3] it is not hard to clarify their meaning in a colloquial sense: verification asks if an artifact meets specifications, while validation asks if it will perform its intended function in its intended environment.

These two processes are distinct. Verification concerns measurement of artifact properties relative to written specifications. Some inferences are involved, but the activity is largely (and ideally) independent of human judgment. As an archetypal example, a specification that calls for a bolt to be tightened to a given torque can be verified with an appropriate torque wrench. Similarly, a specification that calls for a solar panel to lose no more than 0.5% of its maximum power production in one year can be verified by measuring production over a short period and extrapolating to obtain its annual fall-off (i.e., by employing a well-grounded inference).

In contrast, validation concerns the performance of artifacts relative to their intended tasks in their deployment environments. For example, we would like to know that a Mars rover will be able to navigate to desired locations in a reasonable amount of time, conduct various tests, and report the results back to scientists on earth.

As a result, while verification concerns measurement, validation concerns performance in the field and has very different properties.

1.3 PROPERTIES OF VALIDATION

Because systems validation relates the performance of an artifact relative to its intended task, the process has distinct properties.

First, validation involves stakeholder preferences. In the case of a complex artifact, there are likely to be many users with different goals in mind, implying that validity is subjective and can differ across stakeholders. Taking the example of a Mars rover, a geologist might declare that the rover is incapable of conducting a desired test (the rover is invalid for that deployment decision) because it cannot position itself with sufficient accuracy. However, the exo-biologist might be perfectly happy with the rover's positioning capability.

These assessments involve judgment calls over abstract properties of artifacts. For example, the rover's ability to achieve a desired position is enabled by multiple component systems, so validating a claim to positional accuracy will require opinions from experts in each. Their assessments will necessarily concern the performance of the rover in its actual environment whose properties may be partially modeled/understood – like the traction afforded by the Martian soil. This introduces uncertainty into the validation of performance claims.

Validation also concerns prospective judgments. While the above example concerns a deployment decision, we would like to know, years in advance, that the design for a Mars rover will afford a particular positional accuracy once it has been refined, developed, and deployed. This is an example of a validation concern that arises during design.

Lastly, note that the rover might fully meet its specifications but fail to be valid for some stakeholders and tasks. This clarifies that validation and verification are separate processes with distinct goals.

- Involves stakeholder preferences
- Requires judgment calls
- Concerns abstract and prospective claims about artifacts
- Concerns performance in environments that are often partially modeled and understood
- A system can meet requirements and still not be valid

Figure 5 Validation Properties

1.4 VALIDATION ARGUMENTS BY CONTEXT

Validation activities span the lifecycle of a systems engineering project.

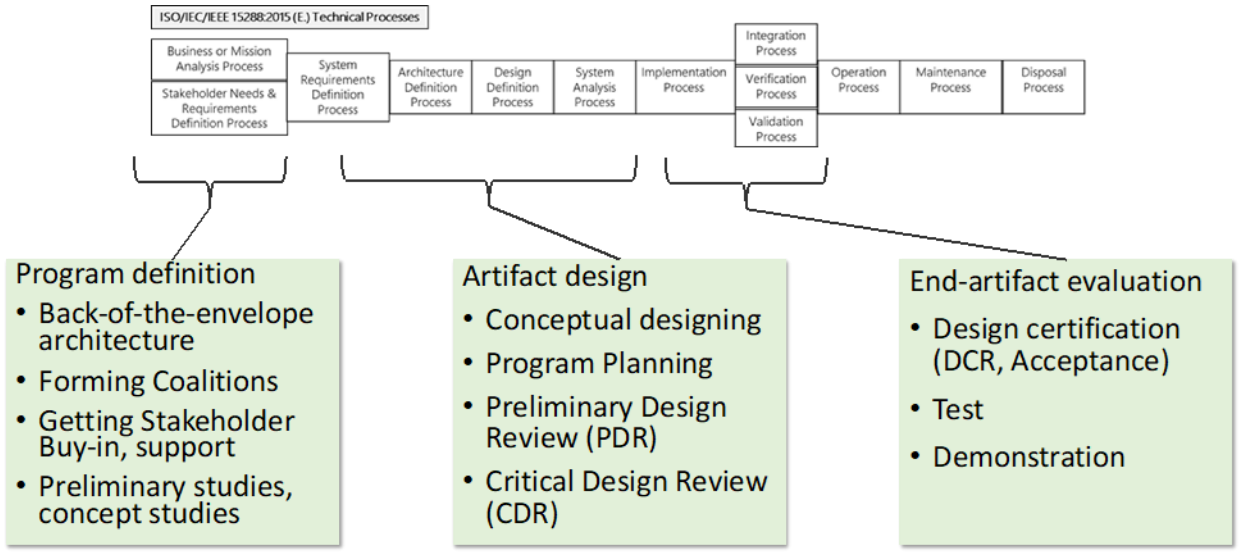


Figure 6 Validation spans lifecycle

The early stages of program definition focus on broad questions of technical, financial, social, and (for very large projects) political feasibility. Discussions at this stage typically clarify and resolve arguments for and against claims about program viability. This can be

seen as a means of validating the program as a mechanism for producing the desired end artifact, well before the artifact is well-defined.

The conceptual design stage of a systems engineering project examines major approach options and plans for achieving them. A simple example is the choice between a 4 vs 5 engine design for a launch vehicle, where the validation task is to determine whether each option leads to an end artifact that will achieve the intended function within an acceptable cost and timeline. As a result, design validation is a prospective task involving somewhat abstract claims. A design review also has the character of an argumentation process, involving claims, justifications, critical questions, counterarguments, resolutions, and decisions for how to proceed.

The final acceptance stage of a systems engineering project contains the most familiar examples of validation activities, which focus on testing an end-artifact in life-like scenarios. In the case of a Mars rover, it involves baking the vehicle in a vacuum chamber to simulate transport conditions, testing its mobility and navigation functions in the Mars yard, plus a host of other integrated systems tests. Validation at this stage concerns generalizing from feasible tests to likely performance in operation, recognizing that there will always be a gap between those settings.

In summary, validation activities occur at all stages of a systems engineering project, and the validation process involves marshaling and resolving arguments for and against claims.

1.5 A VALIDATION ARGUMENT EXAMPLE

The purpose of systems validation is to determine whether an artifact will perform its intended function in its intended environment. Because this objective is unobtainable in the strictest sense (due to the limitations of models), we have adopted the perspective that validation concerns arguments for and against claims about system function. We represent these arguments using a variant on Toulmin diagrams [1] as illustrated in the attached Figure 7.

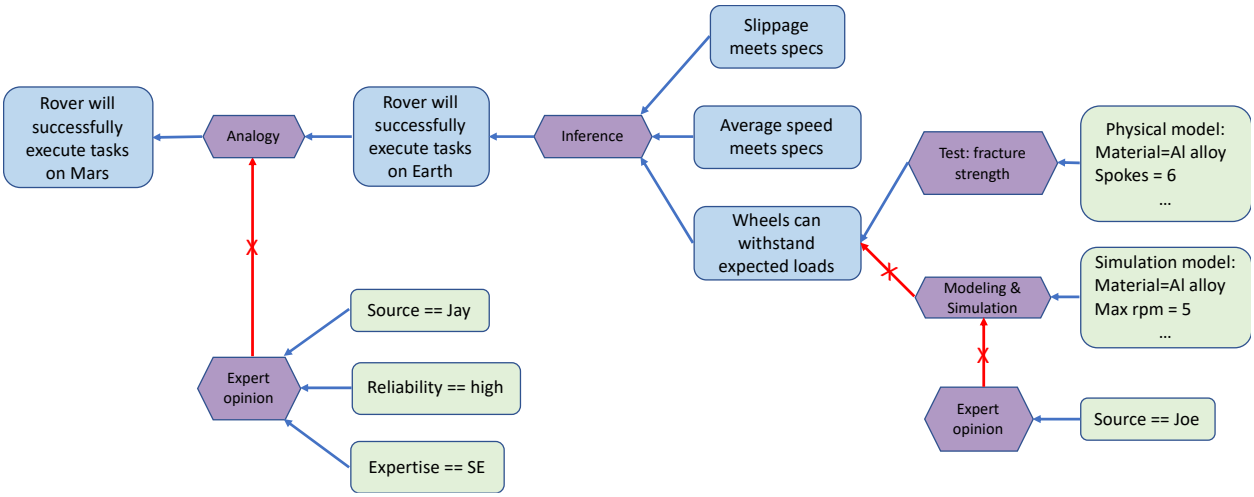


Figure 7 Arguments for and against claims example

Toulmin diagrams consist of predicates (shown as rounded rectangles), warrants (hexagons), and links (denoting input and output relations). Predicates are statements about the world. Primitive predicates (shown in green) encode observations or other inputs to the model, while non-primitive predicates encode intermediate or final conclusions of the model. Warrants are justifications; reasons why we should believe or disbelieve the predicate output by the warrant. Links in blue denote confirmation, while links in red denote disconfirming relationships.

In Figure 7, *Joe is an expert* is a primitive predicate (an observation), as is *Material=Aluminum alloy* (a chosen parameter). *The rover will successfully execute tasks on earth* is an intermediate conclusion, while *the rover will successfully execute tasks on Mars* is a final conclusion. The warrant linking these two predicates asserts that we should believe the rover will be functional on Mars because it successfully executes tasks on earth. The essence of the argument is analogical reasoning.

Taken as a whole, Figure 7 illustrates a validation argument surrounding the claim that the rover will succeed on Mars. It represents a chain of reasoning from detailed statements to abstract conclusions, and it includes both confirming and disconfirming relationships. For example, the physical fracture test (a warrant/justification based on physical evidence) supports the conclusion that the wheel can withstand expected loads, while the warrant that justifies its conclusion on the basis of a simulation study argues in the opposite direction.

Warrants can act to confirm or disconfirm predicates *or other warrants*. For example, Joe's expert opinion states that the simulation is an inappropriate analysis (meaning its conclusions cannot be trusted in the current situation). Jay similarly opines that it is inappropriate to argue by analogy that the rover will perform well on Mars, presumably because the difference between the terrestrial and Martian environments is too large.

Figure 7 contains an incomplete validation model. It represents the state of an argument under development at a particular moment in time, which will be refined as the project

progresses. Elements of the model can represent past results/conclusions, plans (e.g., to conduct a physical test), and expectations (e.g., that the test will confirm the conclusion).

We employ this representation to support mechanisms for editing and evaluating validation arguments, as discussed in the following sections.

1.6 WARRANTS AS STANDARDS OF PROOF

The concept of a warrant is relevant in everyday life whenever we justify why something is true. For example, we might state that it is warm outside because we observed it, that too much sun is bad for your skin because doctors have said it, or that a particular brand of sunscreen is good because the company is reputable and $3 Billion has been sold. Each rationale identifies a warrant, and warrants have types – they can be segmented into categories corresponding to standards of proof.

In the examples above, the reason provided for believing it is warm outside is a form of proof by inspection. The explanation that too much sun has ill effects is a form of proof by attestation, in this case by an appeal to a trusted authority. Both rationale for the quality of the sunscreen are forms of proof by demonstration, grounded in the longevity of the company and the popularity of the product (it wouldn't be so popular if it were bad).

The set of proof standards in common use goes beyond these examples. We frequently justify beliefs by inference and by appeal to social norms (we've always done it this way). In the European Middle Ages, trial by combat was sanctioned by law (my cause is right because my champion prevailed). Figure 8 illustrates a fanciful warrant to demonstrate the point that many standards of proof exist - one of the authors (Shapiro) used proof by pumpkin to justify that a machine learning algorithm was effective at controlling a watering system (if it produced a 600 lb pumpkin it must be good).

Figure 8 Proof by pumpkin

We argue, next, that a handful of fundamental warrant types underlie the arguments found in systems engineering.

1.7 WARRANT TYPES IN SYSTEMS ENGINEERING

We can identify the warrant types employed in systems engineering and organize them in a way that fosters the creation and refinement of engineering arguments. As a starting point, we mine the vocabulary of warrants cataloged by Walton [4] (drawn from common and legal use) to extract a subset that appears in systems validation arguments. Next, we take advantage of the fact that warrants in Toulmin models are coupled with *critical questions*, specific to the warrant's type, that challenge its premises, conclusions, and/or its relevance in the current context. For example, given an expert opinion lawyers will ask: who is the expert, what is their expertise, is the conclusion in their area of expertise, is the expert biased, and is the conclusion something they could reasonably infer? We interpret these questions as constraints on the content and structure of engineering arguments and use them to define a type-hierarchy of warrants characterized by the inheritance of critical questions.

Table 2 identifies four warrant categories that we believe span systems engineering arguments, plus instances of the next two levels in the hierarchy. ***Agreement, inspection, analysis,*** *and* ***construction*** are the fundamental argument types. ***Agreement*** contains warrants that ground belief in the social acceptance of a claim. It includes *attestation*, which cites explicit or implicit testimony, *assumption,* which adopts a belief for convenience or to bound further reasoning, and *declaration,* where an authority sets a parameter that is subject to choice. ***Inspection*** covers arguments based on examination of an actual artifact, with proof by *demonstration* and proof by *test* as subtypes. The *test* subtype can be further subdivided into a variety of strategies for representative sampling of test cases (not shown). ***Analysis*** concerns beliefs justified by analytic processes. It includes *inference*, reasoning by *analogy*, and *modeling and simulation*. ***Construction*** concerns beliefs that are made true by taking action. For example, engineers bolster the claim that wheel slippage is within tolerance by defining a risk avoidance *procedure* to circumnavigate loose soil. A rigorous safety review is an *organizational process* which supports a claim that the rover will successfully execute tasks on Mars. *Delegating* electric motor construction to a reputable firm supports the claim to a rover average speed, as does delegating path planning tasks to validated rover software[1].

Table 2 Warrant Types

Warrant Type	Subtype	Subtype	Critical Questions
AGREEMENT			Is claim subject to agreement?
	Attestation		Is the claim knowable?
		Expert Opinion	Is the expert relevant to the claim?
		Common Knowledge	Does the rule apply in this context?
	Assumption		Is the claim reasonable, material & convenient?
	Declaration		Does the agent have the authority to assert the claim?
INSPECTION			Is the claim knowable via inspection?
	Demonstration		Is the demonstration representative of the use case?
	Test		Does the test address the claim? Are there defeating cases?
ANALYSIS			Is the claim subject to analysis?
	Analogy		Are the source and target environments, tasks, and systems sufficiently close?
	Modeling & Sim		Does the simulation address the claim? Are there defeating conditions?
	Inference		Is the inference cogent? Are there defeating facts?
CONSTRUCTION			Can the claim be made true by taking action?
	Procedure		Will executing the procedure produce the claim in this context?
	Organizational process		Will pursuing the organizational process produce the claim?
	Delegation		Are the principal and delegee value aligned?

All forms of *agreement* share the critical question, “is the claim subject to agreement?”. The choice of material in a rover wheel is a positive example, while the tensile strength of an aluminum alloy is not[2]. All subtypes of *attestation* share the critical question, “is the claim observable (even in theory)?”. This is true of plausible world states, but false for mental constructs like assumptions and declarations. The critical questions become more

[1] Shapiro [5] clarifies when delegations to an autonomous agent are optimal.

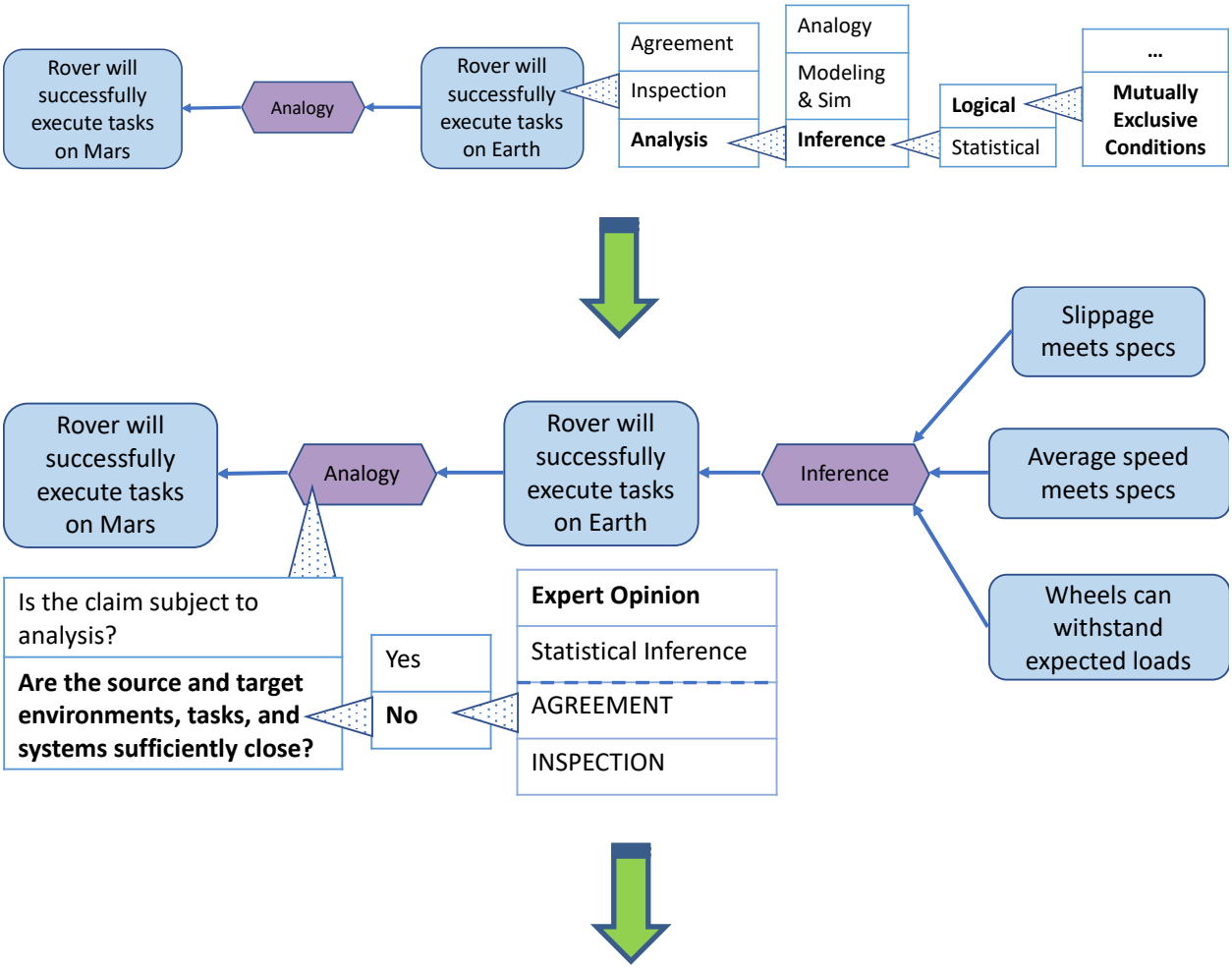
[2] The Indiana Pi Bill is an infamous attempt to apply *agreement* (inappropriately) by legislating a mathematical truth, i.e., that $\pi = 3.2$ https://en.wikipedia.org/wiki/Indiana_Pi_Bill

specific as we descend the hierarchy – Table 2 associates “is the expert relevant to the claim?” with expert opinion for brevity, but multiple critical questions apply (as discussed above).

The next section discusses the use of critical questions to facilitate the construction and refinement of engineering arguments.

1.8 CONSTRUCTING VALIDATION ARGUMENTS BY TEMPLATE INSTANTIATION

Given that warrants embed knowledge about the structure and content of validation arguments, a type-hierarchy of warrants can support a template-based editing model. We identify two main operations, for (a) identifying warrants that can support or refute specific aspects of the model, and (b) instantiating the elements of a warrant (its premises and conclusion) in a way that guarantees completeness and consistency.



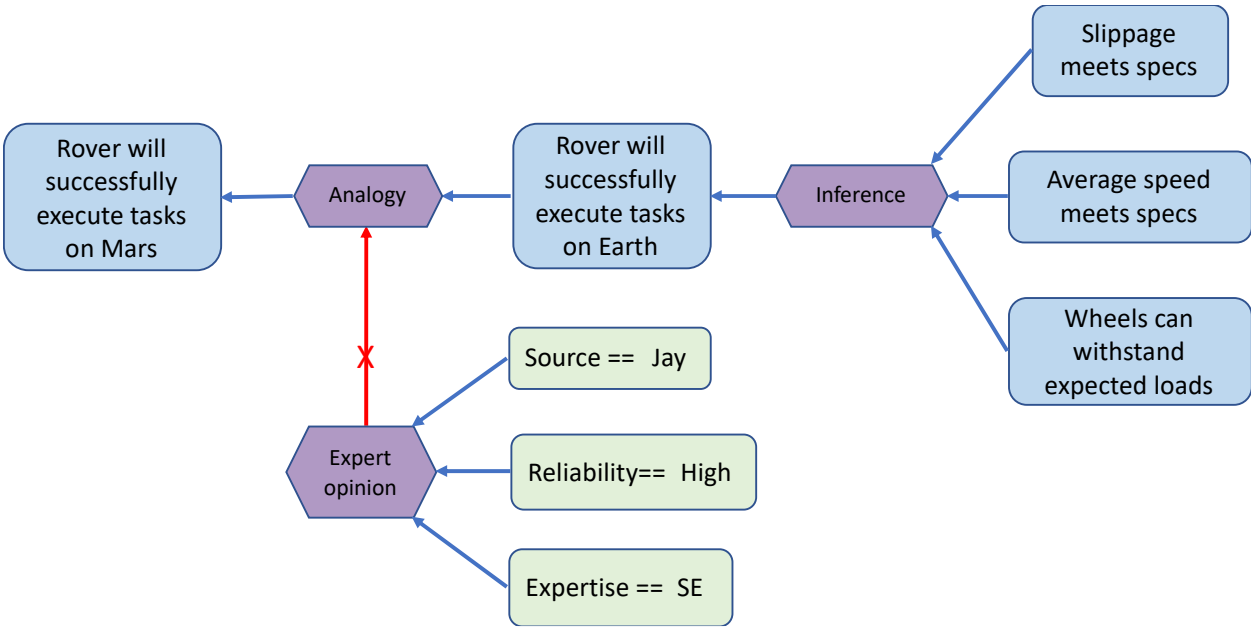


Figure 9 Example of template instantiaion model to refine validation argument

Figure 9 illustrates the use of a template instantiation model to successively refine a validation argument, beginning at the stage where the argument contains exactly one warrant, one premise, and one conclusion. In the top diagram, the user employs the editor to walk the warrant hierarchy, from analysis through modes of analysis, forms of inference, and ultimately to inference over a mutually exclusive set of conditions. Once selected, the editor prompts the user to enumerate the conditions, leading to the middle figure with its predicates for slippage, average speed, and expected load.

The scenario continues when a second engineer questions the validity of the warrant justifying the conclusion that the rover will successfully execute tasks on Mars by analogy with its behavior on Earth. As shown in the middle diagram, the editor retrieves all critical questions associated with the *analogy* warrant (via inheritance through the type-hierarchy) and asks the user to select one. The user indicates that the distance of the analogy is the concern, and '*No'* indicating that the distance is too large. In response, the editor employs experience gained from past interactions to offer two likely forms of counterargument. The user selects the *expert opinion* warrant, which invokes a dialogue for instantiating its elements, leading to the bottom of the three diagrams in the figure. The resulting validation model is far more refined than the initial, and it grows in response to the arguments, pro and con, that are common in engineering contexts. Note that the diagram is by no means complete – new claims will be introduced, and additional elements will be challenged/supported as the validation argument is refined.

In principle, the predicates and constraints referenced by critical questions can be represented in computational form. When this is feasible, the editor can provide additional operations to detect constraint violations (e.g., that the conclusion is not in the expert's area of expertise), validate slot fillers, and instantiate slot fillers when implied. The editor can also be aggrandized to include operations for tracking the status of

validation models (identifying completed, ongoing, and planned activities), searching validation models, and presenting them to stakeholders with interests in specific aspects of the model.

1.9 EVALUATING ARGUMENT MODELS

We take the perspective that a validation model expresses an argument over what to believe in the presence of conflicting rationale. With that in mind, we define a mechanism for evaluating validation arguments into a joint probability of beliefs over the predicates they contain. As mentioned earlier, these predicates typically describe abstract and prospective properties of artifacts under construction, especially their intended behavior in the world.

This is a non-standard treatment of argument models, which are commonly taken to express a single relationship (attacks), carry a binary truth value (accepted or rejected), and analyzed to find consistent subsets within a network of interrelated arguments [6]. Instead, we build a Bayesian interpretation that captures the probability of premises, conclusions, and the relevance of each warrant to the case at hand.

As shown in Figure 10, we represent argument models as directed acyclic graphs consisting of warrants (purple hexagons), predicates (rounded rectangles), and links denoting attacking (red) and supporting relationships (blue). Without loss of generality to the evaluation machinery that follows, we restrict these models to trees that reflect the structure of validation arguments observed in systems engineering. In this setting, each warrant inputs one or more predicates as premises and produces a single predicate representing its conclusion.

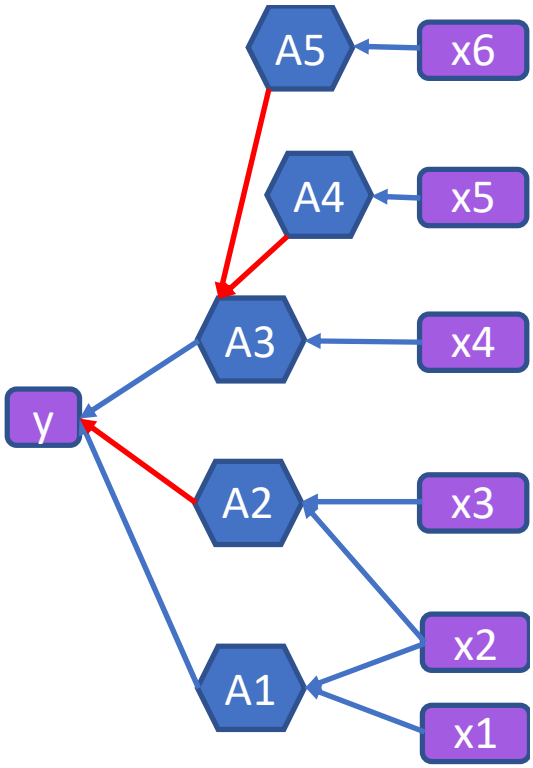


Figure 10 Argument model

$$p(y) = \sum_{\vec{E}} p(y|\vec{E})p(\vec{E}) \quad \text{for } \vec{E} \text{ in truth values of antecedents(y)} \tag{1}$$

We give validation arguments a probabilistic interpretation by associating a probability with each predicate, interpreting each link to indicate conditional dependence[3], and the absence of a link as conditional independence. We further associate a probability with each warrant, called its *aptness* (discussed in the next section), which reflects its relevance as a model for drawing conclusions. A probability that a warrant is apt can depend upon other warrants, as indicated in the figure.

We employ a simple sum of conditional probabilities to evaluate argument models. We apply this equation recursively to aggregate information from leaf predicates (observations or chosen parameter values) through warrants to intermediate predicates and final conclusions.

The following sections discuss the meaning of aptness and the implied probability assessment tasks. Next, we expand our treatment of validation arguments to encompass decision models, and the machinery they contain. We conclude with a worked example

[3] As a visualization aid, red links denote probabilities below 0.5, blue links >= 0.5.

illustrating one of the many capabilities that result, e.g., to determine the value of conducting a test in the presence of arguments for and against the warrants, premises, and conclusions that populate the validation model.

1.10 THE APTNESS OF A WARRANT

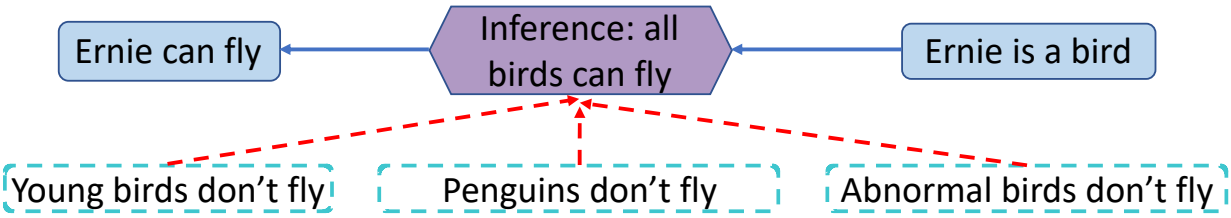


Figure 11 Aptness example

A warrant is *apt* if it is a relevant model for drawing conclusions in the current context. As a simple example, the warrant in Figure 11 captures the inference rule "all birds fly". Given the premise "Ernie is a bird", the warrant concludes that "Ernie can fly". This warrant is *apt* if we can trust its inference. More broadly, we say that a warrant is *apt* if we can trust its conclusion, whatever it might be, given its premises.

The warrant for "all birds fly" may not be apt as it represents a rule of thumb with familiar exceptions, e.g., young birds don't fly, penguins don't fly, and abnormal birds don't fly (like birds with injured wings). However, the inference is valid most of the time, suggesting a sense in which the warrant is mostly apt.

We assign warrants a probabilistic interpretation by abstracting over exceptions. In the case of "all birds can fly", this means assessing the probability that the rule of thumb is apt. Whether this is accomplished by statistical sampling or subjective assessment, the key question is this: if all you know is "Ernie is a bird", what is the probability "all birds can fly" is a good model for drawing conclusions about Ernie? If you believe the rule likely holds, you might answer "0.8". To arrive at this conclusion, you might consider one or two of the exceptional cases (it is unlikely that Ernie is a penguin in North America), characterize them as unlikely, and assign the remainder of the probability mass to the normal case. Implicitly, you must also summarize across unarticulated exceptions, like the possibility that Ernie is an ostrich, emu, cassowary, kiwi (or other flightless bird), a commercial chicken (which doesn't have room to fly unless it is free range), a dead bird, or a carving of a bird (unless it was thrown).

Formally, all subjective assessments of uncertain quantities (such as the likelihood that it will rain tomorrow) require summarization over an unknowably large quantity of conditioning facts, called the background state in probabilistic models. We apply this process to a defeasible inference rule (a rule that can be invalidated by additional knowledge) vs a fact when assessing the probability that a warrant holds.

Validation arguments in systems engineering consist of many defeasible inferences that support or refute claims, rarely with unassailable conclusions. Each of these arguments corresponds to a warrant with a probabilistic interpretation.

1.11 ASSESSING THE PROBABILITY THAT A WARRANT IS APT

Let A == Analogy is apt
R == Rover will successfully execute tasks on earth
E == Expert opinion is apt

$$p(A) = \sum_{R,E} p(A|R,E)p(R)p(E)$$

Requires user assessment

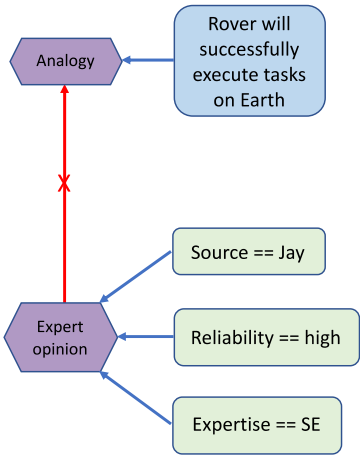


Figure 12 Probability that a warrant is apt

The aptness of a warrant can depend upon premises and other warrants. In Figure 12, the aptness of the analogy warrant depends upon the performance of the rover on Earth and expert opinion that reasoning by analogy is inappropriate in the current context. As a result, determining the aptness of the warrant requires four conditional probabilities:

- that the analogy (A) is apt given the rover successfully executes tasks on Earth (R) and the expert opinion (E) is apt
- that the analogy is apt given the rover successfully executes tasks on Earth and the expert's opinion is not apt
- that the analogy is apt given the rover does not successfully execute tasks on Earth and the expert's opinion is apt
- that the analogy is apt given the rover does not successfully execute tasks on Earth and the expert's opinion is not relevant

The probability that the analogy is apt follows from these assessments and the priors on each of the antecedents.

The prior that the expert's opinion is apt comes from a separate analysis, e.g., a subjective assessment or a statistical summary of the expert's track record in similar circumstances.

The prior probability that the rover successfully executes tasks on earth might come from thorough investigation of representative test cases in simulation and with a physical robot.

Note that the probability that the analogy warrant is apt is distinct from its conclusion (that the rover will successfully perform its tasks on Mars). The aptness of the expert's opinion is also distinct from its conclusion (that the analogy warrant is not apt). The probability that a warrant is apt concerns the relevance of the warrant as a model for drawing conclusions.

1.12 Assessing the Probability of a Premise/Conclusion

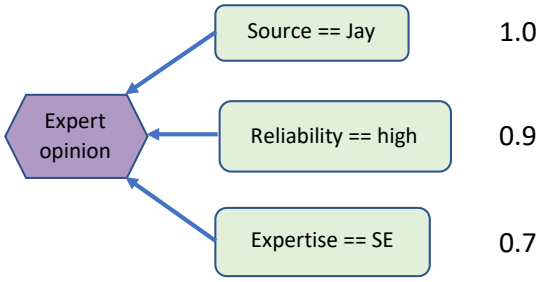
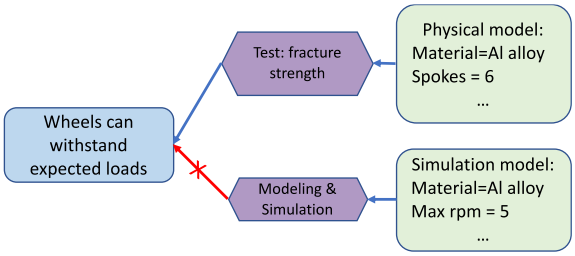


Figure 13 Expert opinion example

Let TFS == test warrant showing fracture strength < load is apt

MS == simulation warrant showing wheels withstand loads is apt



$$p(y) = \sum_{TFS,MS} p(y|TFS, MS)p(TFS)p(MS)$$

Requires user assessment

Figure 14 User assessment example

The probability that a leaf premise is true (a predicate with no antecedents) is determined by assessment. As shown in Figure 13, Jay is the expert in question (with probability = 1.0), he can be counted on to provide an honest opinion regardless of the circumstance (with probability = 0.9), but his expertise is not a perfect match to the topic at hand (p=0.7). Given these numbers, the probability that Jay's expert opinion is apt can be determined by the process described in the previous section.

The probability of an intermediate or final predicate is determined by a conditional probability calculation, as before. Figure 14 illustrates the required assessments, here, of the conditional probability that the wheels can withstand expected loads given all combinations of the aptness, or inaptness of the fracture strength test and simulation analyses.

1.13 UNIFYING ARGUMENTATION AND DECISION MODELS

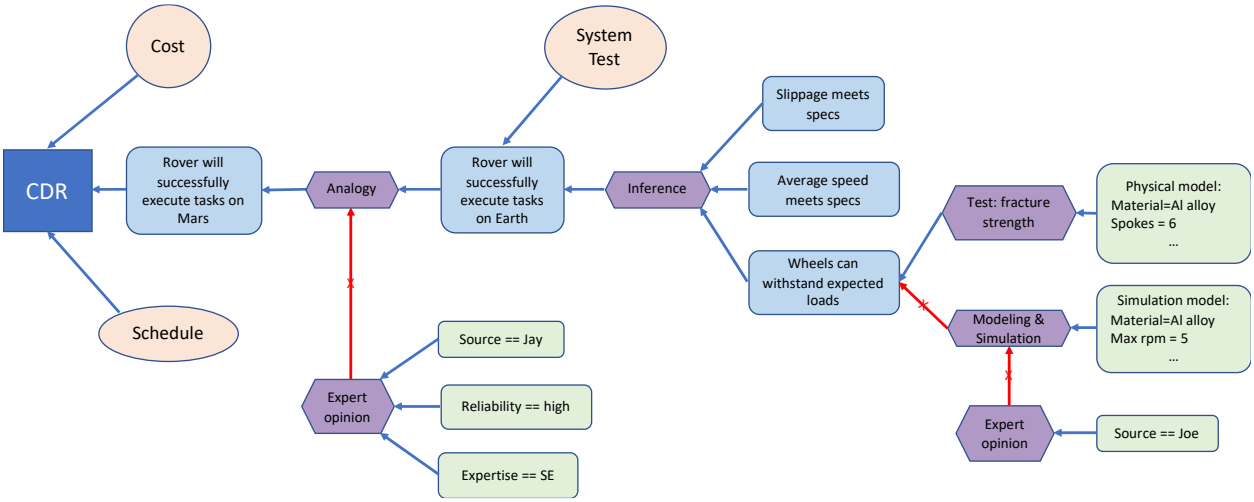


Figure 15 Example including uncertainties and decision points

We can extend the representation of argument models to include uncertainties and decision points. Uncertainties are probabilistic quantities that are not a premise or conclusion of a warrant. Figure 15 illustrates three examples (shown as ovals): cost, schedule, and the results of system tests. Decisions are choice points, as described in decision theory, here illustrated by a Critical Design Review.

We incorporate these new elements by augmenting the evaluation machinery. We sum over uncertainties in the conditional probability calculations that assign probabilities to claims, and we perform a utility maximization at decision points (invoking the presence of a utility model over claims/uncertain outcomes).

These extensions make it possible to evaluate decisions in the context of validation models that contain arguments and counterarguments about claims and the warrants (rationale) that underly them. Formally, this completes an integration with influence diagrams [7].

This unification makes the full machinery of decision analysis available for evaluating systems validation arguments. We illustrate this capability with a worked example that

computes the value of conducting a physical test to resolve a conflict between a simulation result, and an expert who believes the simulation should not be trusted.

1.14 THE VALUE OF A TEST WITHIN A VALIDATION MODEL

We posit a decision problem that would naturally arise in the process of validating a design for vacuum cleaning robot's wheel; should the firm employ a 3-spoke approach (a risky, less expensive solution), or a 6-spoke design (which is an expensive, sure thing)?

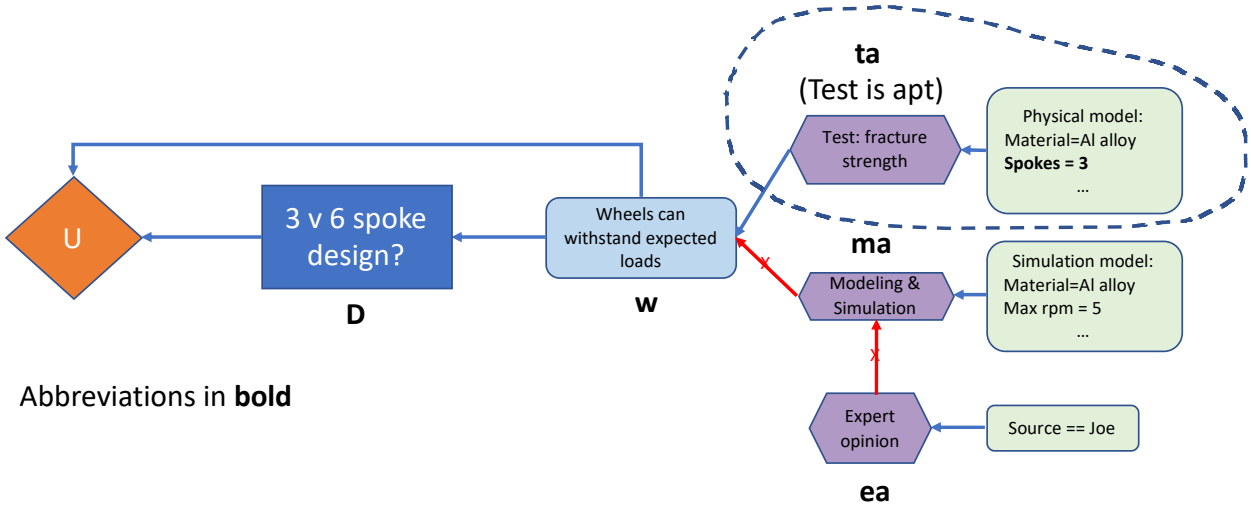


Figure 16 Value of performing test

As shown in Figure 16, we assume the firm has conducted a simulation study casting doubt on the 3-spoke design, but that an expert also casts doubt on the validity of the simulation (perhaps it wasn't designed to represent the types of surfaces and transitory loads that can appear in a household environment).

Engineers can gain more information by physically testing the wheel's fracture strength. The advisability of conducting this procedure can be determined by a value-of-test calculation:

> What is the expected value of testing the fracture strength of a 3-spoke wheel design in the 3-vs-6 spoke wheel decision, given the simulation results? Does that value exceed the cost of the test?

We assume a simple utility model to guide this evaluation:

6 Spokes: An expensive, sure thing

- p(w) = 1.0
- cost = $100,000 over production run

Utility = 100,000 p(w) – 100,000 = 0

3 Spokes: A risky, less expensive solution

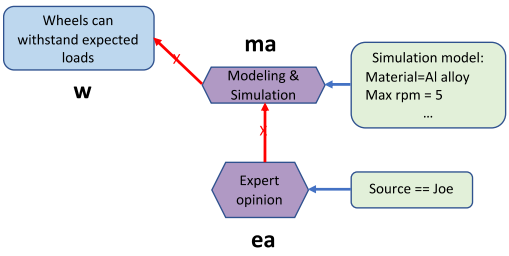
- p(w) is subject to argument, including physical test
- cost = $40,000 over production run

Utility = 100,000 p(w) – 40,000 – cost(test)

The key variables are p(w), the probability that the wheels will withstand expected loads, and the cost of the test, which is absent in the 6 spoke alternative. The utility models include a constant that scales p(w) relative to costs.

We evaluate the 3-spoke validation model to determine *p(w)* using a prior on the results of the fracture strength test and solve for *cost(test)* at the indifference point between the two designs (at Utility = 0). If the physical test is less expensive than this reservation price, it is worthwhile.

1.15 THE UTILITY OF A 3-SPOKE DESIGN WITHOUT A FRACTURE STRENGTH TEST



Abbreviations in **bold**

$p(ea) = 0.8$	Expert is trustworthy (opinion is apt)
$p(ma\|ea)$ = 0.4	Expert casts doubt on modeling & sim
$p(ma\|\neg ea)$ = 0.6	Prior is that modeling & sim likely apt
$p(w\|ma)$ = 0.4	Modeling says 3-spoke wheels won't hold
$p(w\|\neg ma)$ = 0.1	Prior is that 3-spoke wheels will not hold

$$p(ma) = p(ma|ea)p(ea) + p(ma|\neg ea)p(\neg ea)$$
$$= 0.2 * 0.8 + 0.6 * 0.2 = \mathbf{0.28}$$
$$p(w) = 0.184$$

Figure 17 Wheels can withstand expected loads probability calculation without test

We confirm that the 3-spoke design has lower utility than the 6-spoke design absent the fracture strength test by evaluating its argument model. This requires several assessments. As shown in Figure 17, an engineer (or other evaluator) determines that the expert's opinion is apt with high likelihood (p=0.8), and that the simulation is likely not apt (p=0.4) given the expert's disconfirming opinion. For comparison, the engineer assigns a higher probability to the simulation being apt (p=0.6) if the expert's opinion is not apt.

The simulation results indicate that the wheels will probably not withstand expected loads (p=0.4), but this conclusion is conditioned on the simulation being apt. If the simulation

is not apt, the engineer's prior on the probability that the wheels will withstand expected loads is very low (p=0.1), which is plausible in the absence of any data.

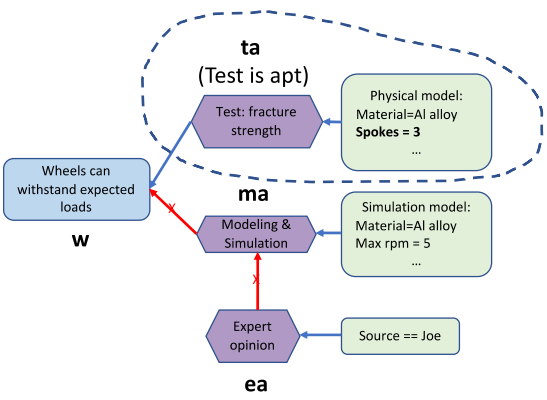
Given these assessments and simulation results, we determine that the simulation is apt with p=0.28 by summing across the cases where the expert opinion is apt, and not apt. An analogous calculation yields p(w) = 0.184 by summing across cases where the simulation is apt, and not apt.

After substituting p(w) = 0.184, and cost(test) = 0 into the utility model

$$EU(3\ spoke\ design) = 100000\ p(w) - 40000 - \text{cost(test)}$$

we obtain **EU(3 spoke design without the fracture strength test) = -21600**, meaning the 6 spoke design (with its utility of 0) is preferred to the 3-spoke design absent the test.

1.16 THE UTILITY OF THE 3-SPOKE DESIGN AND THE VALUE OF THE TEST



$p(ea) = 0.8$	Expert is trustworthy (opinion is apt)
$p(ma\|ea)$ = 0.4	Expert casts doubt on modeling & sim
$p(ma\|\neg ea)$ = 0.6	Prior is that modeling & sim likely apt
$p(ta) = 0.9$	Fracture strength test is definitely apt
$p(w\|ta, \neg ma)$ = 0.8	Expect physical test to show wheels hold
$p(w\|ta, ma)$ = 0.6	If both are apt, trust physical test more
$p(w\|\neg ta, ma)$ = 0.4	Modeling says 3-spoke wheels won't hold
$p(w\|\neg ta, \neg ma)$ = 0.1	Prior is that 3-spoke wheels won't hold
$p(ma) = 0.28$	
$p(w) = 0.688$	

Abbreviations in **bold**

Figure 18 Wheels can withstand expected loads probability calculation with test

We obtain the expected utility of the 3-spoke design with the fracture strength test by utilizing priors for the consequences of that test. Figure 18 illustrates the computation, showing new assessments in blue, and assessments common to the 3-spoke design (absent the test) in black.

The engineer must make two additional assessments:

- the probability that the fracture strength test is apt, *p(ta)*
- the probability that the wheels can withstand expected loads given the two new cases where the test is apt, while the simulation is apt/not apt

We show a high expectation that the test is apt, $p(ta) = 0.9$, and a high prior that the physical test will confirm that the wheels can withstand expected loads, $p(w|ta, \neg ma)$ = 0.8. We show a somewhat lower expectation if the simulation is also apt, $p(w|ta, ma)$ = 0.6, as we know it to be disconfirming.

The net result, after summing across conditional distributions, is that $p(w) = 0.688$, a much higher value than without the fracture strength test. Substituting this value into the utility model yields:

$$\begin{aligned} EU(3\ spoke\ design\ with\ test) &= 100000\ *0.688 - 40000\ - \text{cost(test)} \\ &= 28800\ - \text{cost(test)} \end{aligned}$$

We obtain the value of the fracture strength test by solving for *cost(test)* at the indifference point between the 3-spoke and 6-spoke designs. Setting the above to zero (the utility of the 6-spoke design) indicates that the test is worth performing if $cost(test) \leq \$28{,}800$.

This example illustrates a new capability to perform a decision analytic calculation over a systems validation model in the presence of conflicting arguments. While we have described a 'value of test' computation, other decision-analytic manipulations should follow, e.g., value of information calculations, sensitivity analyses of utility to the probability of premises and warrants, and dominance analyses over decision alternatives.

RELATED WORK

Other researchers have noticed that systems validation shares features with the process of marshaling arguments for and against claims. This has generated interest in the application of argumentation models [1] [8] and AI argumentation theory [6] [9] [10] to systems validation tasks.

Research on intelligent editing tools for constructing validation plans is possibly the closest in spirit to our own. For example, the AdvoCATE system [11] employs Goal Structuring Notation [8] (GSN), which is a form of argument model, to partially automate assurance case creation and validation. AdvoCATE defines assurance cases as action plans containing *goals* (claims)*, strategies, solutions, assumptions, justifications,* and *context*. It provides templates that decompose common assurance cases into component parts (i.e., by enumerating hazards or requirements), plus operations for automatically instantiating those templates to a case at hand, inferring properties of the assurance case model (e.g., do all branches terminate in solution nodes), and for querying the model to present views that are specialized to stakeholders. D-Case [12] provides a typed editor for manual construction of assurance cases, while ACSE[4] (Assurance and Safety Case Environment) is a commercial tool that imports safety cases and associated documents in a variety of representations (including GSNs) and manipulates them primarily from an information management vs evaluation perspective. Smith et al. [13] automatically generate test cases for a Mars rover following argumentation strategies encoded in GSNs. These tools illustrate the power that an intelligent editor model brings to validation reasoning, as well as several more specific ideas that we employ in our work: typed arguments, model construction by template instantiation, and the evaluation of argument models.

Many authors have examined the problem of evaluating validation arguments, largely to ascertain their implications for system safety. Graydon and Holloway [14] survey approaches for quantifying confidence in assurance arguments, including methods that develop and evaluate Bayesian interpretations of arguments from GSNs [15] and Toulmin diagrams [16]. Wang et al. [17] employ Dempster-Shafer theory to assess the confidence that the goals of a safety assurance case are met. Goodenough et al. [18] measures confidence by counting the number of defeating arguments eliminated while refining an assurance case encoded as a GSN. Bjornander et al. [19] flag instances of a GSN that are not satisfied by an underlying system architecture. These examples illustrate a diversity of approaches to evaluating argument structures. For comparison, our work proceeds by unifying Toulmin argumentation models with influence diagrams [7] and applies the machinery of decision analysis to evaluate them.

Work in argumentation theory also includes efforts to categorize argument models. For example, Luo et al. [20] define GSN patterns for six classes of safety arguments, Szczygielska and Jarzębowicz [21] extract 45 assurance case patterns via a literature meta-study, while Walton et al. [4] identify 96 argument schemes found in common

[4] https://www.adelard.com/asce/choosing-asce/index/

discourse. These categorizations informed our efforts to define an inheritance hierarchy of warrant types.

AI research on argumentation theory has focused on the formalization and evaluation of arguments models and the development of computational machinery. Prakken et al. [22] grounded the relations in Toulmin argument models (attacks undercuts, rebuts, undermines) in formal logic. Dung [6] formalized a representation of a network of atomic arguments (absent claims, premises, and warrants) that engage in attack relationships, and the criteria for their acceptance/rejection. The goal in this setting is to extract consistent subsets of arguments per one of several possible semantics [23] (e.g., complete, grounded, preferred, stable, or semi-stable semantics). This problem has generated significant attention, including an annual competition[5]. However, the absence of argument substructure in this representation makes it less applicable for capturing the interrelations among arguments found in systems validation. In addition, the evaluation machinery typically imposes a binary truth value on arguments (they are either accepted or rejected) as opposed to the treatment of arguments as evidence that is more appropriate to the systems engineering context.

[5] http://argumentationcompetition.org/2021/

SUMMARY

This project began with the intuition that the reasoning in systems validation could be productively framed as argumentation vs proof. Our work evolved to articulate a vision for applying Toulmin-style argumentation models to organize, compose and evaluate validation models within systems engineering projects.

We have made the following contributions:

- We clarified, by example, that validation activities occur at all project stages from systems conception through final artifact evaluation (viewing validation as the task of determining that an eventual artifact will perform its intended function in the world).

- We illustrated the use of Toulmin style argumentation models to capture validation reasoning for and against claims about system properties. In contrast, most work on argumentation in systems engineering employs GSNs, which do not represent conflicting evidence or opinion.

- We argued, by construction, that systems validation involves a handful of fundamental warrant types, which can be organized into a type-hierarchy based on the inheritance of critical questions.

- We showed, via example, that a type-hierarchy of warrants and critical questions enable a template-based editor for composing validation arguments.

- We defined a novel probabilistic interpretation of warrants as defeasible inferences. A warrant's aptness is the probability that it is a relevant model for drawing a conclusion.

- We integrated argument models with influence diagrams, creating the capability to evaluate decisions in the presence of uncertainties and arguments for/against claims and warrants.

- We illustrated the capability to evaluate validation decisions via a worked example.

These building blocks enable creation of a tool for managing validation argumentation in systems engineering, which is the subject of our following discussion of next steps.

NEXT STEPS

Our work has generated several results that are ready for publication. These papers will largely translate the material in this report into conference form. Plausible titles include:

- Framing systems validation as argumentation model construction
- A formal semantics for defeasible probabilistic reasoning in systems validation
- Towards a tool for managing validation arguments in systems engineering

Our work has made several conjectures that need further investigation. First, we claimed that a handful of warrant types suffice to cover systems validation arguments. To demonstrate that point we will need to instantiate a significantly larger portion of the warrant hierarchy and show that inheritance of critical questions is an informative and general organizing principle. We will examine validation arguments that are as disparate as possible (e.g., drawn from different phases of systems engineering projects), compare them with arguments found in the Walton catalogue [4], and identify both the subset and diversity of arguments employed in validation contexts.

Second, we should investigate the generality of our claim that systems validation can be productively framed as argumenta model construction. This calls for a breadthwise examination of argumentation uses (as above), but with an emphasis on testing bounds. We should consider validation arguments in program inception and requirements specification (which are underexamined contexts) and clarify the value that an argumentation tool would provide in each.

Third, we claimed that a library of warrants supports a template-based editing model. This suggests a design study fueled by multiple worked examples, culminating in a prototype of an intelligent editing tool. In principle, we can automate further aspects of argument construction. For example, given that critical questions specify constraints (is the claim within the expert's expertise?), a tool with appropriate background knowledge can automatically apply such tests.

Fourth, we should investigate the broader impact of our unification of argumentation models with influence diagrams. In addition to a value-of-test computation, we should illustrate a value of information calculation over validation arguments, a sensitivity analysis of claims on decisions, and a dominance analysis of decisions in the presence of uncertainties and conflicting arguments. More broadly, we should identify systems validation contexts where the combined machinery of argument models and decision analysis can supply value, as the capability is new.

Finally, our work to date has been motivated by desire to produce a tool for managing systems validation arguments. We should pursue the design and development of such a tool and demonstrate its application benefit. One promising avenue is to cast argumentation as a view within a model-based systems engineering framework [24] [25]. This will provide a tool for constructing, evaluating, visualizing, and tracking validation arguments with a repository of machine-readable system descriptions that it can employ to ground its operations. In parallel, systems engineers will gain an ability to track a

validation perspective across a project's lifetime. Overall, the construction of a tool for managing systems validation arguments implies significant effort, with a prototyping project as its first step.

APPENDIX A: LIST OF PUBLICATIONS AND PRESENTATIONS RESULTED –

Shapiro, D., Mesmer, B., Jones, N., Collopy, P., Stevens, J., “Towards a Tool for Managing Validation Arguments in Systems Engineering”, 2021 AI for SE & SE for AI Workshop, October 20th, 2021

Shapiro, D., Mesmer, B., Jones, N., Collopy, P., Stevens, J., “Towards a Tool for Managing Validation Arguments in Systems Engineering”, 13th Annual Systems Engineering Research Center Sponsor Research Review, Online, November 3rd, 2021

APPENDIX B: CITED AND RELATED REFERENCES

[1] S. Toulmin, *The Uses of Argument*, 2nd ed. Cambridge University Press.

[2] R. M. O'Keefe, O. Balci, and E. P. Smith, "Validating Expert System Performance," *IEEE Expert*, vol. 2, no. 4, pp. 81–90, Jan. 1987, doi: 10.1109/MEX.1987.5006538.

[3] B. W. Boehm, "Verifying and Validating Software Requirements and Design Specifications," *IEEE Softw.*, vol. 1, no. 1, pp. 75–88, Jan. 1984, doi: 10.1109/MS.1984.233702.

[4] D. Walton, C. Reed, and F. Macagno, *Argumentation Schemes*. Cambridge: Cambridge University Press, 2008. doi: 10.1017/CBO9780511802034.

[5] D. Shapiro and R. Shachter, "User-Agent Value Alignment," Jan. 2002.

[6] P. M. Dung, "On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and *n*-person games," *Artif. Intell.*, vol. 77, no. 2, pp. 321–357, Sep. 1995, doi: 10.1016/0004-3702(94)00041-X.

[7] R. A. Howard and J. E. Matheson, "Influence Diagrams," *Decis. Anal.*, vol. 2, no. 3, pp. 127–143, Sep. 2005, doi: 10.1287/deca.1050.0020.

[8] T. Kelly and R. Weaver, "The goal structuring notation–a safety argument notation," *Proc Dependable Syst Netw. Workshop Assur. Cases*, Jan. 2004.

[9] T. J. M. Bench-Capon and P. E. Dunne, "Argumentation in artificial intelligence," *Artif. Intell.*, vol. 171, no. 10, pp. 619–641, Jul. 2007, doi: 10.1016/j.artint.2007.05.001.

[10] P. Besnard, C. Cayrol, and M.-C. Lagasquie-Schiex, "Logical theories and abstract argumentation: A survey of existing works," *Argum. Comput.*, vol. 11, no. 1–2, pp. 41–102, Jan. 2020, doi: 10.3233/AAC-190476.

[11] E. Denney and G. Pai, "Tool support for assurance case development," *Autom. Softw. Eng.*, vol. 25, no. 3, pp. 435–499, Sep. 2018, doi: 10.1007/s10515-017-0230-5.

[12] Y. Matsuno, H. Takamura, and Y. Ishikawa, "A Dependability Case Editor with Pattern Library," in *2010 IEEE 12th International Symposium on High Assurance Systems Engineering*, Nov. 2010, pp. 170–171. doi: 10.1109/HASE.2010.26.

[13] B. Smith, M. Feather, and T. Huntsberger, "A Hybrid Method of Assurance Cases and Testing for Improved Confidence in Autonomous Space Systems," presented at the 2018 AIAA Information Systems-AIAA Infotech @ Aerospace, Kissimmee, Florida, Jan. 2018. doi: 10.2514/6.2018-1981.

[14] P. J. Graydon and C. M. Holloway, "An investigation of proposed techniques for quantifying confidence in assurance arguments," *Saf. Sci.*, vol. 92, pp. 53–65, Feb. 2017, doi: 10.1016/j.ssci.2016.09.014.

[15] E. Denney, P. G. J, I. Habli, P. G. J, and I. Habli, "Towards Measurement of Confidence in Safety Cases," presented at the 5th International Symposium on Empirical Software Engineering and Measurement (ESEM), Sep. 2011. Accessed: Feb. 15, 2022. [Online]. Available: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20110016239.pdf

[16] X. Zhao, D. Zhang, M. Lu, and F. Zeng, "A New Approach to Assessment of Confidence in Assurance Cases," Sep. 2012, vol. 7613, pp. 79–91. doi: 10.1007/978-3-642-33675-1_7.

[17] R. Wang, J. Guiochet, G. Motet, and W. Schön, "Modelling confidence in railway safety case," *Saf. Sci.*, vol. 110, pp. 286–299, Dec. 2018, doi: 10.1016/j.ssci.2017.11.012.

[18] J. B. Goodenough, C. B. Weinstock, and A. Z. Klein, "Eliminative induction: A basis for arguing system confidence," in *2013 35th International Conference on Software Engineering (ICSE)*, San Francisco, CA, USA, May 2013, pp. 1161–1164. doi: 10.1109/ICSE.2013.6606668.

[19] S. Bjornander, R. Land, P. Graydon, K. Lundqvist, and P. Conmy, "A Method to Formally Evaluate Safety Case Evidences against a System Architecture Model," in *2012 IEEE 23rd International Symposium on Software Reliability Engineering Workshops*, Dallas, TX, USA, Nov. 2012, pp. 337–342. doi: 10.1109/ISSREW.2012.101.

[20] Y. Luo, Z. Li, and M. van den Brand, "A Categorization of GSN-based Safety Cases and Patterns:," in *Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development*, Rome, Italy, 2016, pp. 509–516. doi: 10.5220/0005734305090516.

[21] M. Szczygielska and A. Jarzębowicz, "Assurance Case Patterns On-line Catalogue," 2017, pp. 407–417. doi: 10.1007/978-3-319-59415-6_39.

[22] H. Prakken, "An abstract framework for argumentation with structured arguments," *Argum. Comput.*, vol. 1, no. 2, pp. 93–124, Jan. 2010, doi: 10.1080/19462160903564592.

[23] P. Baroni, M. Caminada, and M. Giacomin, "An introduction to argumentation semantics," *Knowl. Eng Rev.*, vol. 26, pp. 365–410, Dec. 2011, doi: 10.1017/S0269888911000166.

[24] A. W. Wymore, *Model-based systems engineering: an introduction to the mathematical theory of discrete systems and to the tricotyledon theory of system design*. Boca Raton: CRC Press, 1993.

[25] B. P. Douglass, "Chapter 1 - What Is Model-Based Systems Engineering?," in *Agile Systems Engineering*, B. P. Douglass, Ed. Boston: Morgan Kaufmann, 2016, pp. 1–39. doi: 10.1016/B978-0-12-802120-0.00001-1.