DOT/FAA/TC-17/26

Federal Aviation Administration
William J. Hughes Technical Center
Aviation Research Division
Atlantic City International Airport
New Jersey 08405

Definition and Measurement of Complexity in the Context of Safety Assurance

September 2017

Final Report

This document is available to the U.S. public through the National Technical Information Services (NTIS), Springfield, Virginia 22161.

This document is also available from the Federal Aviation Administration William J. Hughes Technical Center at actlibrary.tc.faa.gov.

U.S. Department of Transportation
Federal Aviation Administration

NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof. The U.S. Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report. The findings and conclusions in this report are those of the author(s) and do not necessarily represent the views of the funding agency. This document does not constitute FAA policy. Consult the FAA sponsoring organization listed on the Technical Documentation page as to its use.

This report is available at the Federal Aviation Administration William J. Hughes Technical Center's Full-Text Technical Reports page: actlibrary.tc.faa.gov in Adobe Acrobat portable document format (PDF).

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
DOT/FAA/TC-17/26		

4. Title And Subtitle	5. Report Date
DEFINITION AND MEASUREMENT OF COMPLEXITY IN THE CONTEXT OF SAFETY ASSURANCE	September 2017
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Sarah Sheard, Michael Konrad, Chuck Weinstock, and William R. Nichols	

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
Software Solutions Division Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	11. Contract or Grant No. DFACT-14-X-00010

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
Federal Aviation Administration William J. Hughes Technical Center Aviation Research Division Atlantic City International Airport, NJ 08405	Final Report 14. Sponsoring Agency Code AIR-130

15. Supplementary Notes

The FAA William J. Hughes Technical Center Aviation Research Division Technical Monitor was Srini Mandalapu.

16. Abstract

Airborne system complexity has been increasing rapidly with the advent of newer technologies. There are multiple interpretations of complexity, and it is difficult to measure system complexity in a meaningful way. The goal of this research effort was to define an appropriate type of complexity and to come up with a measure to estimate the complexity of avionics systems. This measure would help the FAA predict when systems are too complex so that it can assure their safety for use on certified aircraft.

Although the literature did not provide succinct definitions of the term complexity from which to select a small number of definitions, the readings did provide a rich canvas of concepts and approaches that serve as a solid foundation from which to select or define candidate measures and to identify impacts on flight safety. In fact, the word “complexity” implicitly suggests that some causes create some effects; without elucidation of which causes and which effects, the word is being used in a casual manner and is not adding facts to the discussion. After a detailed review of the literature, a taxonomy of complexity was developed, and this taxonomy is used to determine the impacts and measurement of system complexity.

A measure of complexity was developed using a number of ways that an avionics system error (fault) could propagate from one element to another. Because each potential propagation requires another sub-argument in the safety case, the number of such arguments should be linear with certification effort. Therefore, the ability to show that the system is safe, through the assurance process, should depend on whether a system has small enough complexity (number of ways for errors to propagate).

The results of this research include a formula for calculating the error-propagation complexity from a system design, the results of using that formula for small and medium systems, and steps for using the formula. The formula was tested on small system (stepper motor), a redesigned stepper motor systems, a medium complex system (wheel braking system), and a larger system (SAAB-EH-100).

To get a better feeling of the complexity measure, an attempt was made to convert the measure into an estimate of assurance safety review effort. The review effort is a good surrogate measure to compare systems of different complexities and to determine whether or not a system is too complex to assure safety. It estimated the review time for the small cases and extrapolated up to larger cases, assuming a spread of small, medium, and large designs included within a typical avionics system. Many of the numbers used are not tested and validated in terms of relationships and assumptions. Therefore, the proposed boundary of systems “too complex to assure safety” should be treated with caution.

17. Key Words	18. Distribution Statement
Airborne systems, System complexity, Safety assurance, Assurance case, Complexity measurement, Error propagation complexity, Safety case, Fan-out, Error model	This document is available to the U.S. public through the National Technical Information Service (NTIS), Springfield, Virginia 22161. This document is also available from the Federal Aviation Administration William J. Hughes Technical Center at actlibrary.tc.faa.gov.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	49	

ACKNOWLEDGMENTS

The SEI team would like to thank our excellent Business Manager Gregory Such for his continuous support and help during the past 2 years. Tamara Marshall-Keim provided excellent and timely technical editing and process guidance, supported by Erin Harper and Claire Dixon when Ms. Marshall-Keim was not available. Donald Firesmith, Julian Delange, John Goodenough, Peter Feiler, and David Zubrow (of the SEI), Maureen Brown (University of North Carolina), and Mike DeWalt, Barbara Lingberg, John Strasburger, and Srini Mandalapu (of the FAA) provided technical contributions, and David Zubrow and James Over provided excellent management support when needed. Our FAA point of contact, Srini Mandalapu, has been a steadying force, contributing and supporting throughout.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ACRONYMS

AADL	Architecture Analysis & Design Language
CAR	Certification authority review
DAL	Design Assurance Level
FC	Failure conditions
P-points	Propagation points
SCI	Safety Case Inspection
SEI	Software Engineering Institute
SCOR	Safety Case Ordinary Reading
SCR	Safety Case Review
STPA	Systems Theoretic Process Analysis

EXECUTIVE SUMMARY

This report defines complexity measures for avionics systems to help the FAA identify when systems are too complex to assure their safety.

The project selected a measure of complexity related to the number of ways that an avionics system error (fault) could propagate from element to element. As each potential propagation requires another sub-argument in the safety case, the number of arguments should be linear with certification effort. Therefore, the ability to show system safety through the certification process depends on this kind of system complexity.

The results include a formula for calculating the error-propagation complexity from system designs and its results for small and medium systems. It was tested on a second design for each system and on a larger design from a NASA report.

The complexity measurement must be matched to available review time to determine if a system is “too complex to assure safety.” Review times for small cases were extrapolated to larger ones, assuming that a typical system includes small, medium, and large designs. As many numbers and their relationships are speculative, the boundary of systems “too complex to assure safety” should be treated very cautiously. Finally, future research areas are discussed.

1. THE PROBLEM

The purpose of this report is to publish an algorithm for connecting the complexity of an avionics system to its likelihood of safety problems and its ability to be assured for use on a certified aircraft.

This has been a 2-year research project. The tasks in the research project were the following:

- Task 1[1]–Manage project
- Task 2–Conduct a literature review to define complexity for avionics systems
- Task 3–Identify candidate measures of complexity
- Task 4–Identify the impact of complexity on safety
- Task 5–Quantify the effects of system complexity on aircraft safety and identify relevant complexity measures
- Task 6–Test the identified measures on a representative avionics system
- Task 7–Make recommendations for the assurance of complex systems

A summary of the results for each of the prior tasks can be found in appendix A.

The literature of complexity [1] shows not only how things, tasks or situations can be complex, but also the various kinds of things that can have those kinds of complexity. When something is complex, it is assumed that there are factors behind that complexity and consequences that result from it. Some of the causal factors include size (e.g., number of sub-pieces), interrelationships among the pieces, structure of the pieces, dynamism, and even socio-political factors [2]. Consequences include difficulty building, difficulty predicting behavior, commission of errors, and even the type and amount of emergent properties.

The research initially started by measuring [3] the size of the system (e.g., in terms of number of lines of code or of architectural elements), but a clear link between measurements of those characteristics and safety could not be made. After a careful study of how safety is assured [4], it was apparent that what needs to be measured was not the avionics hardware or software, but the act of assuring that a system is safe (assuring that installed systems meet the applicable assurance standards and regulations).

Although Task 2 resulted in many potential directions for defining "complexity", one definition[2] (in the context of estimating complexity) was selected during Tasks 3-5 and has been used since then. This definition of complexity is the complexity of the assurance case that must be made to consider the avionics system as safe.

An assurance case is a structured argument, supported by a body of evidence providing a compelling, comprehensible, and valid case that a system exhibits a specified property (e.g., safety) in a given application in a given operating environment. Assurance cases consist of claims, subclaims, and evidence. A claim might be that the system is safe. Subclaims that must be satisfied for the claim to be true would include the software and the hardware being safe. There can be many levels of subclaims constituting a subclaim hierarchy, going down to the level where

[1] Tasks 1 through 7 are called Tasks 3.1 through 3.7 in the project plan and the report titles.

[2] From Task 3 document: Complexity is a state or quality of being composed of many intricately interconnected parts, in a manner that makes it difficult for humans, supplemented by tools, to understand, analyze, or predict behavior [3].

evidence can be obtained to show directly that the subclaim is true. A Safety Case is a special kind of assurance case that shows how the safety of a system is argued and evaluated.

Measurement will be useful in guiding selection of tooling, technology, and training needed to assess safety. The number of concepts that are needed to describe complexity makes it a complex topic, and because measurement, by definition, is a simplification, it may not be possible to measure complexity directly. A proxy for complexity was used to measure and describe the length of time (or equivalently, the amount of resources) it will take to achieve a level of confidence in the system's safety. It is possible to calculate the effort, cost, or equivalent staff years to review the entire assurance case given the number of sub-arguments in the system's assurance case (e.g., each sub-argument relating to one instance of a fault or error in one component propagating to another), some estimate of reviewer capability, and variables (e.g., a typical length of time to review a sub-argument). The theory is that, as a system gets more complex (e.g., has more pieces, interconnections, potential unanticipated interactions, or emergent properties), it will have more potential ways for hazards to cause unsafe situations; therefore, its assurance case will take longer for a certification team to review [3].

The bottom line is a complexity formula and a set of steps for evaluating the elements of the formula. The output of the formula is a number, which is an estimate of the number of ways that errors can propagate out from one component of the system to another. This number can then be turned into resource or time requirements via the factors mentioned above.

The FAA requested a definitive[3] statement as to a level of complexity above which the systems can be presumed to be too complex to assure for inclusion on a certified aircraft. Optionally, the Software Engineering Institute (SEI) could state a range of complexity numbers, below which systems are acceptable, and above which systems should be considered too complex (between these levels are "borderline systems").

Systems in flight today are safe; therefore, they fit in the lowest group of systems ("not too complex to assure safety"). Tomorrow's system can be determined to be safe only if its safety case can be reviewed by a certification team within program schedule, cost constraints, and availability of expertise and tools. Systems whose certification effort will exceed program budget cannot be assured. Systems whose certification effort is expected to exceed the budget the program has allocated to the certification effort also cannot be assured, although most programs can allocate a few more resources to certification if necessary to sell and operate the system.

The complexity formula counts the number of potential error propagations whose assurance arguments must be examined independently to assure that the proposed system will behave safely. Multiplied by conversion factors such as the typical length of time and typical effort to review the assurance argument that the error propagation will not result in an unacceptable hazard, this results in a number for total time (minutes) and resources (staff years) that it will take to assure that a system is safe. The authors guessed at what such conversion factors might be in a typical case, but because research like this has never been done before, data to support it does not exist; therefore, these numbers are speculative. Actual distributions for such factors would need to be determined by additional research.

[3] Though it must be definitive, the statement is allowed to be provisional.

The results of this research are as follows:

1. The algorithm to calculate complexity (of an avionics system):

    a. Inputs: Avionics system measurements taken from a formal statement of an avionics system's high-level architecture. The measurements are: the number of failures that can propagate out from one component to another and the related fan-out for each. These are to be measured for all propagation points (P-points), for all components, and for all system modes.
    b. Outputs: Complexity (as counted by number of ways errors can propagate from one component to another) and estimated review time for the system's assurance case.
    c. Algorithm:

Error Propagation Complexity = Sum over all system modes (i in $1..m$), Sum over all components (j in $1..n$), Sum over all P-points of component $C[j]$ (k in $1..q[j]$), of $[\ OutPropagateFailureCount(i,j,k) \cdot FanOut(i,j,k)\ ]$

2. The calculated boundary between systems that are too complex to assure safety and those that are not.

The above complexity number can be used to determine whether a system can be assured for use on a certified system. The review time required for a system of such complexity was compared to FAA reviewer availability and capability.

For an FAA staffing level comparable to that for the Boeing 787 (i.e., actual 12.5 full-time people for 8 years, and assuming approximately half [six people] deal with avionics), the system Error Propagation Complexity that can be reviewed within allowed resources is approximately 6,400 in the worst case and approximately 21,000 in the best case. There are many caveats associated with these numbers. For calculations, see section 2.3; for discussion about why these were chosen and the caveats, refer to appendix B.

2. METHODOLOGY

This section describes why the chosen methodology was selected, how complexity was calculated, and how the maximum value of complexity that an avionics system can have to be assured for certification purposes was calculated.

2.1 WHY THIS ALGORITHM?

A common motivation for a measure of complexity is practical: a measure is intended to capture some essence of a task that correlates with the amount of effort it will take to perform that task. The measure can then be used to predict how much effort that task will take in the future. The motivation behind this research is the need for the ability to predict how much effort (review time)

a capable certification authority review (CAR) team, equipped with appropriate training and tools, will need to expend to confidently determine both issues below:

- Whether a system (or component) design resolves a particular system (or component) hazard
- More broadly, whether the applicant is on track to creating a design that will mitigate a set of system hazards

This research focused more on the former, short-term, and more narrowly focused problem; some insights for the latter, longer term, and broader problem were drawn.

2.1.1 What to Measure: Design Complexity Versus Review Complexity

A measure of design complexity would have some of the attributes being sought. Verification of a design can get more challenging as its complexity rises, but the biggest burden is often borne by the creator of the design, who has had years of experience in the domain and analytic tools to identify promising candidate solutions and then evaluate those candidate solutions to show satisfaction of needed constraints. Arguments will need to be developed that use evidence to support assertions that the constraints are satisfied. The resulting design and parts of arguments are then reviewed by those carrying out verification activities to identify the set of evidence that uses the best possible combination of inputs and modes and best sequences the tests to maximize early discovery of defects and minimize later rework. For both architects and testers (and for those who develop architectural analysis tools), design complexity measures are important to estimating difficulty in working with and reasoning about the system.

The CAR team has a different problem to address. The team is provided a system (or component) design and the results of verification activities, including analyses (including modeling and simulation), inspections, and testing. The team reviews these and determines whether some possible hazards have been overlooked or not fully mitigated. Team members must review not only the design but also the test plan, records of how testing was conducted, and the results. They must determine that the evidence does show what the submission claims that it does, and that the system will not be subjected to unsafe situations or conditions from any of the hazards. They need to familiarize themselves with the behavior of components and to reason about them in context.

Therefore, design complexity is essential to characterizing the complexity of the task that the CAR team performs. Consequently, standard design complexity measures (e.g., the number of parts or percent test coverage) are inadequate for identifying potential causes of safety issues.

After a review of the literature on definitions and measures of complexity and safety [1, 3], the researched focused on a measure of the number of ways that failure conditions (FCs) can propagate from one component to another in a design. This measure estimates the number of error propagation events that may require review and resolution to develop an understanding of whether a particular system hazard is prevented or reduced to be very improbable in the system design. This quantity is called Error Propagation Complexity. The units are potential error propagations.

This measure is also a design complexity measure, although it is not a common one. This measure presumes that, as part of the design, an error model has been specified to identify the FCs for each

component of a system design, which can propagate outward from that component to influence the behavior of another component. The measure could also be applied to existing systems. When tracked over time, the measure would provide information about the complexity of systems and their parts and allow a benchmark for additional scrutiny during the assurance process.

2.1.2 Correlation with Review Effort: Caveats

Although the measure appears to correlate with review effort, it is not perfectly correlated, for two reasons.

First, the measure, which counts the number of possible propagations and therefore safety arguments to analyze, may overstate the number actually required to properly evaluate the likelihood of a system hazard. In practice, some system hazards will likely need only short arguments; whereas other system hazards (e.g., those addressing the emergence of undesired system behavior) might require much longer arguments, potentially touching on every possible propagation.

Second, the difference in two designs is not as high as might be expected. The measure used in this research assigned approximately 6% less complexity to an elegant redesign for a stepper motor-based engine control system. In reality, however, the original design requires 100%–200% more effort to review. The redesign uses a hypothetical hardware component that eliminates a particularly problematic FC, a so-called replication error [5] (in the original design, the possibility of this failure created additional derived requirements on intermediary design components, and therefore additional argumentation, to eliminate them as a source for a system hazard; to keep it simple to understand, the formula to address these was not revised, because it still accounted for some fraction of the additional complexity in what might be an untypical example).

However, the measure generally conforms to the intuition of argument complexity, is reasonably easy to operationalize, has reasonable interrater reliability, is automatable, and serves as a basis for answering the two questions at the start of this section.

2.2 ALGORITHM FOR ESTIMATING THE ERROR PROPAGATION COMPLEXITY OF A SYSTEM DESIGN

Section 2.2.1 summarizes the formula developed in the remainder of section 2.2. Section 2.2.2 covers the main assumptions and inputs. Section 2.2.3 covers the terminology used. Section 2.2.4 explains the notation used in the formula and then presents the formula. Section 2.2.5 provides the rationale for the formula. Section 2.2.6 addresses generalizations and limitations of the formula, and addresses several other important points. Section 2.2.7 presents a procedure for applying the formula in a practical fashion to a system design and error model.

2.2.1 Bottom Line

Error Propagation Complexity can be characterized as the number of ways in which an error created inside a system component can propagate to another component. Because Error Propagation Complexity is used to estimate how long certification will take, the number of separate analyses that must be done was counted. A system model (which can be high level) that shows components, connections, and possible errors (failures and faults) was used. This model was

perhaps created in a formal system modeling language such as Architecture Analysis & Design Language (AADL).

To estimate the number of ways an error can propagate from one component to another, the steps below were followed:

- First, if there is more than one mode, the analyses will be repeated for each mode. Therefore, a sum over all modes will be provided.
- Second, each component that is active in that mode may be the source of an error that propagates to another component. A sum over all components active in the mode will be given.
- Third, an error that originates within a component will exit (or escape from) the originating component through a P-point. The errors propagating through each P-point separately will be looked at, and a sum over all P-points will be presented.
- Finally, propagation events escaping from each P-point and transiting to other components must be estimated. Each P-point has two attributes of interest, called Failure Count and FanOut. The system's error model will tell how many errors there can be (failure count) associated with a given P-point, and the system's interconnection model will tell how many components these errors can transit to through the given P-point (the fan-out). Each error associated with a P-point and each component that error can transit to will again be evaluated separately.

To determine the total number of cases that must be evaluated (to ensure that a failure propagating from one component to the next cannot cause an unsafe condition), the following formula to compute the potential size of a safety argument (number of distinct cases that may need to be considered) will be used:

Error Propagation Complexity =
Sum over all system modes (i in $1..m$),
Sum over all components (j in $1..n$),
Sum over all P points of component $C[j]$ (k in $1..q[j]$), of
[$OutPropagateFailureCount(i,j,k) \cdot FanOut(i,j,k)$]

2.2.2 Assumptions

The main assumption is that there is a system architectural design that may be preliminary[4] but identifies system modes, components and their interconnections, and, for each component, possible FCs that have the capability to propagate outward.

Section 2.2.3 explains the points leading to the Error Propagation Complexity formula in section 2.2.4.

[4] For existing systems, design documentation should be available. The first diagram to be examined should be one that addresses major components as black boxes, rather than something more detailed.

2.2.3 Terminology Used in the Formula

The terminology used in expressing and discussing the formula is as follows:

- A mode is an operational phase of a system, during which the system is expected to experience different inputs and behave differently than when in other modes.
  - Examples of modes include takeoff, flying, and landing; full-strength, partial, and degraded; or coupled and uncoupled autopilot.
  - This definition for mode generally agrees with common use, but the Error Propagation Complexity formula only refers to system modes and not component modes. The ways to adjust the formula to address the situation is discussed in section 2.2.6.

- A P-point (also called interaction point) is any interface feature of a component through which an FC experienced by that component can emanate out to, and influence the correct functioning of, other components.
  - Examples of P-points include output ports, sockets, bus connectors, and possibly just a region of the component's surface through which that influence might occur.
  - Within a given system mode, an FC can potentially emanate out through none, one, or many of that component's P-points. The FC is said to be bound to this subset of a component's P-points.
  - In common use, a design will identify inward and perhaps bi-directional P-points too, but when applying the Error Propagation Complexity formula, focus on those P-points through which FCs can propagate out, regardless of whether or not they are also a conduit for the entry of FCs.

- Interconnections are represented in this work as logical relations[5] indicating in a given system mode which P-points of which components of a system are connected to which other components. Interconnections are considered to convey FCs from the component in which they arise through one or more P-points (to which that FC is bound) of that component to other components.
  - Note that interconnections may be only active during certain system modes. In some architectural design languages, interconnections can experience FCs [5] and/or can be active during mode transitions [6]. The methods to adjust the formula to address these situations is discussed in notes #4 and #5 in section 2.2.6 [7].
  - Because interconnections are logical relations, it is possible to conceptually relate a given P-point of a component to other components (called "recipient components") using just one interconnection but with a fan-out to all recipient components; alternatively, multiple interconnections can be introduced to indicate the conveyance of FCs from that given P-point. The former interpretation allows for simplicity of expressing the formula. It is therefore assumed that each P-point

[5] This applies to all interconnections, including physical ones, because they are represented by a logical representation in the system model.

has at most one interconnection, but with whatever fan-out is required to reach all recipient components.

2.2.4 Error Propagation Complexity Formula

Begin with a system architectural design (e.g., characterized as a formal model in a language, such as AADL) that specifies the following:

- m **system modes**: M[i] (i in 1..m), indicating for each:
  - Which system components and interconnections are active.
  - Which FC a system component may experience.
- n **runtime components**: C[j] (j in 1..n), indicating for each:
  - Its q[j] **P-points**: P[j,k] (k in 1..q[j]), where q[j] = number of C[j]'s P-points.
  - Which FC are bound to which P-points and for which system modes.
  - In which modes each component is active.
- **interconnections** between P-points and components:
  - Note that for a given system mode, determining which components an FC can initially reach can be done by tracing the FC through any P-points it is bound to (and therefore can emanate from) and then on to all components reachable from those P-points, as determined by the interconnections active in that same mode.

This formula also requires definitions of two factors, as follows:

- **OutPropagateFailureCount**(i,j,k) number of FC that can propagate out through P-point P[j,k] when C[j] is active in mode M[i] (note that this value can be 0). This number is available in an error model.

- **FanOut**(i,j,k) in mode M[i], the number of components that P[j,k] can transmit an FC to, through an active interconnection (there is either an active interconnection at P[j,k] in mode M[i] or not; if there is none, then FanOut(i,j,k) = 0).

Then **the potential size of a safety argument** (i.e., number of distinct cases that may need to be considered)—called Error Propagation Complexity—is:

```
Sum over all system modes (i in 1..m),
        Sum over all components (j in 1..n),
                Sum over all P-points of C[j] (k in 1..q[j]), of
                        [ OutPropagateFailureCount(i,j,k) * FanOut(i,j,k) ]
```

2.2.5 Rationale

Arguing that a system is safe based on a runtime characterization of its components and their interconnections potentially requires considering four kinds of items:

1. Every system mode (M[i])
2. Every system component C[j] active in that mode
3. Every FC that one of those components might experience that the component cannot be assumed to fully handle (and therefore may propagate out)
4. Every component that escaping FC might propagate to

Then it requires arguing how each component that receives an FC addresses it (i.e., fully resolves, mitigates, ignores, or transfers it, etc.). The formula is therefore structured to consider each of the bullets above and count all the resulting arguments. Each of these resulting arguments is potentially a distinct case requiring a distinct argument, and the time it takes to review these arguments (to demonstrate safety) should be approximately linear with the number of arguments.

2.2.6 Notes

During development and initial use of the Error Propagation Complexity Formula, and the subsequent analysis of results, the following generalizations, limitations, and observations were made:

1. The system environment in its entirety should generally be treated as one or more of the system components (i.e., one of the C[j]); otherwise, any FC arising in the system environment and propagating into the system will be excluded from consideration. In the examples considered thus far, what to use as P-points has not been in doubt, and is unlikely to be an issue when the Error Propagation Complexity Formula is applied to subsystems embedded in a larger engineered system. However, derivation of P-points for the system environment could become an issue when applied to a system that directly interacts with

people or the atmosphere, for example, unless the design includes a careful and precise definition of the system boundary.

2. The formula treats the components of a system design as "black boxes," in which certain FCs can be assumed to be fully handled by the implementers of that component (and therefore support compositional verification). This allows for the focus of the research to be on the system complexity (added safety issues created by interconnections) rather than just size. In addition, it avoids some forms of double counting and helps to keep the formula simple.
3. A system's component can also have modes nested within (i.e., as sub-modes of) a particular system mode—or perhaps independent of the system's modes. In the case of nested component modes, it is best to revise one of the summed items, replacing the contribution of a given component in a given system mode with a sum of the contributions by that component over all its modes that fall within that particular system mode. In the case of a component having modes independent of the system's modes, it may be necessary to consider pairs of modes (or more generally, n-tuples of modes).
4. If any of the interconnections can be the source of (i.e., not just pass through) an FC, then that interconnection should be re-characterized as a runtime component (i.e., one of the C[j]) rather than an interconnection. The rest of the system design topology (e.g., components, P-points, and interconnections) should be updated accordingly. (Appendix B of the Task 5 report recommends a way to handle this situation [8].)
5. Likewise, if a particular interconnection is active only during a mode transition, then for purposes of applying the formula, treat the mode transition as its own distinct system mode (one of the M[i]).
6. Related to Note 2, the formula explicitly ignores internal component complexity. For example, a number of state variables or computations could introduce error. This is one of the issues that exhaustive unit test or verification in DO-178B/C [9] is intended to address. The choice to focus on outgoing propagation implies that this formula applies to the integration of components. Nonetheless, a weight could be applied to each component. The weight could perhaps be the Design Assurance Level (DAL) or the likelihood of error. Alternatively, if a component is itself designed as a system of subcomponents, its complexity could be separately calculated using the formula and used as a weight.
7. During development, periodically ensure that the analysis is still correct. As long as all of the system's components are implemented in a manner that is faithful to the analyzed system design, the formula need not be re-applied. If any of the components of the system design are later implemented in a way that does not conform to the system design (e.g., FCs that were originally assumed to be handled by the component in fact "leak out"), then the design and associated fault model should be revised and the formula applied again.
8. It is possible that several of the distinct arguments mentioned under section 2.2.5 can in fact be more efficiently argued together within a system safety argument, thereby reducing the number of truly distinct cases to consider. When this is the case, the summations can be re-factored to sum over equivalence classes. This would reduce the effective complexity of such a system so it may be less complex (i.e., less problematic to build and review) than another system that cannot collapse the review effort completely.
9. Use an FC (error) taxonomy (e.g., see the SAE AADL Annex Volume 1 [5]) appropriate to the domain and design being considered. A very coarse error taxonomy will result in a smaller number of situations to consider, but larger (and possibly much more convoluted)

individual arguments. The error taxonomy used should be both relevant (appropriate to domain and design) and sufficiently granular to distinguish FC that might differ in impact or resolution. It should also be applied uniformly across the system design. The selection (and any tailoring) of the error taxonomy and its subsequent application should be performed by knowledgeable safety analysts. Another way to assess whether an error taxonomy is sufficiently granular is to relate it to the top-level system hazards: The fault model derived from the design and error taxonomy should be sufficiently detailed and specific to compose an argument that a given system hazard is impossible, very improbable, or its effects largely mitigated.

10. Automate the calculation of the formula where possible.
11. Based on using the formula on other examples, researchers may want to update the formula to address some of the issues noted (especially, Notes 2 and 6), so the formula may be revised.

2.2.7 Procedure for Applying the Error Propagation Complexity Formula

In the absence of an automated analysis tool, the formula needs to be applied manually. Often the model (i.e., the system design and fault model) is expressed in some kind of semi-formal or formal architecture description language that makes direct application of the formula by those unfamiliar with the language problematic. The procedure introduced in a report by Konrad et al (addressing Task 5) [8] were further refined by introducing some initial preparatory steps to simplify the application of the formula. The procedure, further simplified by organizing the information needed into a table, is presented here (the full procedure without such simplifications can be found in Sections 3–5 of the Task 6 report [7].)

2.2.7.1 Step 1. Initialize the Table for Calculating Complexity (One Table per Mode)

This table will have a top row of column headers followed by multiple rows underneath, one for each major class of components appearing in the system design.

In general, only put into the same row those components with identical P-points, FCs, and fan-out, so the cells in the row can be easily filled in with the correct values, and the meaning of the value in each cell of the table is clear. When there are different numbers for different elements of a type, put each element into the table separately.

The table, when populated, will look something like table 1 (at the time of initialization, it will not have data in the rows). In the column header, these abbreviations were used: *P* = propagation;, *FO* = FanOut.

Table 1. Example of table for calculating complexity

Component (or class) (1)	Number of Components (2)	Number of P Points per Component (3)	#FC (4)	FanOut (5)	#Components * #P points * #FC * FO (6)
cpu mem	1	4	1	1	4
Par1..Par4	4	1	2	1	8
Mon1..Mon2	2	1	1	1	2
Cmd1..Cmd2	2	1	1	1	2
		1	1	3	6
Pdl1..Pdl2	2	1	1	1	2
Select	1	1	1	2	2
Green Pump, Blue Pump, Accum	3	1	1	1	3
Shutoff	1	1	1	1	1
Selector	1	2	1	1	2
Green Skid, Blue Skid	2	1	1	1	2
Wheel	1	0	n/a	n/a	0

Components: 20 Error Propagation Complexity: 34

The table summarizes the information for only one system mode. A similar table will generally need to be constructed for each additional system mode.

2.2.7.2 Step 2. For Each Component (or Class of Components), Record its Information

The information for each component (or class of component) appearing in a runtime view of the architecture should be contained in its own row.

For example, the first two rows in the example table characterize the computational infrastructure used by the system encompassing two different classes of components: CPUs (there is just one) and processor partitions (there are four).

Likewise, continue filling in the first column with each class of component used.

The rest of each row is then filled in with information found in the model pertaining to that component (or component class) as follows:

- Number of components: Consult the runtime view of the model to see how often components of a particular class are utilized.

- Number of P-Points per Component: Consult the declarative section of the model to see how many P-points are declared (see section 2.2.3 for terminology) for a component of that class.
- FCs: Consult the error model for how many FCs are declared for the component. If the component contains P-points having different FCs or fan-out connectivity, follow the steps below:

    - Split the row into sub-rows as was done for the command components (Cmd1..Cmd2) in table 1, one sub-row per distinct P-point.
    - Identify the number of FCs bound to that P-point within the sub-row associated with a particular P-point (this is the value of the OutPropagateFailureCondition factor introduced in section 2.2.4).
    - Exercise care when entering information to be sure the subsequent information entered for that component is associated with the correct P-point (or add a column to hold the names of P-points).

- FanOut: Determined by inspecting the model to see how design components are interconnected (remembering that not all interconnections and components might be active in all modes).

2.2.7.2.1 Note: Do Not Record Higher-level System Representations that Group Components Together

These higher-level system representations slightly inflate the "score" returned by applying the Error Propagation Complexity formula without significantly affecting the effort expended in reasoning about system safety. It is recommended to not enter information about these higher-level system representations into the table, but focus instead on the lower-level runtime components that they contain (an example is given in the report for Task 6 [7]).

2.2.7.2.2 Note: Do Not Record Orphaned Inputs

The system design might include some components whose outputs are never used; and whose FCs can never propagate (e.g., because there are no connections to P-points to which they are bound; or they are not bound to any P-points). If, indeed, they generate no FC that can affect other components, eliminate these components from consideration (an example is given in the report for Task 6 [7]).

2.2.7.2.3 Note: Record Multiple Connections from the Same P-Point as a Fan-Out

Based on the definition of interconnections in section 2.2.3, for a given mode, each P-point should only have a single interconnection that fans out to other components. If in the model being evaluated there are truly multiple interconnections emanating from the same P-point—and active in the same mode—then split that P-point into multiple P-points (sub-rows) in the table so that the constraint is still met. Otherwise, regard all these interconnections as constituting the fan-out from the same P-point.

The result so far is a table (or tables if there are multiple modes) very similar to table 1. Note that there are no P-point names or FC names in the table. If the table is used for the sole purpose of applying the formula, such names and interconnection names or labels are not needed.

The table can also be depicted graphically in a manner that is neutral to the architecture description language used. Steps 1–7 in the Task 6 report by Konrad described how to achieve this [7]. The formula can then be applied directly to either the resulting graph(s) or table(s).

2.2.7.3 Step 3. Apply the Formula

To apply the formula:

1. Consider each system mode (table) separately in the following steps.
2. Consider each component (or component class) given its own row in the table separately in the following steps.
3. If there are no split rows, simply multiply the values in columns 2–5 and enter the product in the last column.
4. Otherwise, if there are split rows (e.g., Cmd1..Cmd2 in table1), within each sub-row, simply multiply the values in columns 3–5 of that sub-row together with the value in the parent row column 2, and enter the product in the last column.
5. The total Error Propagation Complexity for a particular system mode is the sum of all the entries in the last column of the table.
6. The total Error Propagation Complexity for the model is then the sum of all such table totals resulting from Step 8, one per system mode.

2.3 IS A SYSTEM TOO COMPLEX TO BE ABLE TO ASSURE SAFETY?

This section estimates the complexity numbers for a system that helps determine whether the system: 1) can be assured for safety; 2) is borderline too complex to assure; or 3) is almost certainly too complex to assure. The answer was first given in this research, with notes on some of the uncertainties in the estimation. Descriptions of various factors in the estimate were developed, including the rationale for the numbers used and where they were taken from. Because this research is groundbreaking, many of the numbers going into this estimation could only be grossly approximated. In the future, thresholds may be established using more concrete data from measured evidence. Additional research should be performed to solidify knowledge of some of the factors.

The numbers used depend on a number of other assumptions and may change. See appendix B for a more rigorous description of the factors that will affect the estimation and what assumptions were made.

2.3.1 Bottom Line

To compute whether a system will be able to be certified, the research started with the premise that a system can only be certified as safe if the length of time certification process is reasonable compared to the length of the program.

The authors have defined an "error propagation event" as one error propagating from one component to another. Error Propagation Complexity is then computed as[6] the number of arguments that will need to be made to demonstrate that the system cannot become unsafe because of a possible error propagation event. This quantity estimates how large the certification argument will be. Each event will require certifiers to review (or to create) a section of an assurance case [4, 8] demonstrating that the specific event is safely handled. The relationship between this quantity and the total certification time is assumed to be linear: The more arguments that must be made, the proportionately more resources the certification effort will take (in effort hours).

This report intends to find a value of Error Propagation Complexity that equates to the FAA allocating all (100%) of its certification resources on assurance activities for that one program. A system cannot be confidently determined to be safe if it takes more resources to prove it safe than the resources available for that one program (this Error Propagation Complexity value can easily be adjusted if safety assurance is only allowed to consume 1% or 10% or some other fraction of the available resources).

2.3.1.1 Error Propagation Complexity Equation Applied to Estimating Assurance Effort

The system's Error Propagation Complexity (i.e., number of error propagation events) times the amount of effort it takes to determine the safety of an average (or typical) error propagation event equals the total amount of review effort that must be expended to determine the whole system's safety. Therefore,

EPC * effort to resolve typical event = effort to assure the system as safe

That effort will then be compared to an estimate of the total certification effort available to FAA.

The calculation of effort to resolve a typical error propagation event makes many assumptions, (see sections 2.2, 2.3, and appendix B). Nevertheless, after deliberation over assumptions, the amount of effort required as 2.83–7.80 staff-years per 1000 events was calculated (see section 2.3.2.5).

Assuming a typical amount of resources for program certification, the Error Propagation Complexity for a system that would make it "too complex to assure safety" are 21,000 in the best case and 6,400 in the worst case.

These numbers come from estimating how quickly certifiers can resolve possible error propagation events (i.e., best case and worst case)—assuming a mix of small, medium, and large subsystems that are approximately the same size and complexity as the stepper motor case, the Wheel Brake System case, and the top-level SAAB-EII-100 (hypothetical) case, respectively—and converting given assumptions about the composition of an FAA review team from review minutes to years and to staff-years.

In practice, one would calculate or estimate the Error Propagation Complexity for the system of interest and also estimate the average effort to resolve a typical event from actual review performance data or estimations from other sources, such as those described above. Multiplying

[6] The authors recognize that arguments other than error propagation may need to be made to fully consider a safety claim.

these two figures gives a value of effort for certification, which should be then compared with the available certification effort to determine whether the system can reasonably be certified as safe. The number of review minutes can also be converted to the duration (calendar time) to accomplish such reviews, taking into account estimated program duration and available staffing to conduct such reviews.

2.3.2 Rationale for Factors Appearing in Effort Estimates

The calculations proceed in a number of increments. First, two review rates are estimated: 1) based on trials the authors have conducted with reviewing safety cases, and 2) based on rates known for code inspections.

Then the review times for best case and worse case for the examples of small, medium, and large system designs are estimated.

Then a hypothetical example system was constructed whose complexity and review time will be calculated. The review time was converted from minutes spent reviewing to time for a team of four reviewers and then staff-years overall. From these numbers, a conversion factor that produces the staff-years of effort for a given system complexity was calculated. These numbers are then compared to an actual case (The Boeing 787 Dreamliner) for partial validation.

Finally, the complexity of a system that can just be assured for safety in the effort allotted for the best- and worst-case review rates was calculated, assuming the time allotted matches the numbers found for the Boeing 787.

2.3.2.1 Review Rates

Two review rates (i.e., units: error propagation events considered per minute) were calculated. Their inverse (i.e., minutes it typically takes to review a safety case) per unit of Error Propagation Complexity of the design was also determined.

Table 2 displays complexity and safety case (assurance case) review statistics for the two designs of the stepper motor example (see appendix A, section A-4), including the number of nodes[7] and the review time for two trials.[8]

[7] Nodes are structural elements of an assurance case. In the case of the reviews described in the text, the particular type of assurance case reviewed is eliminative argumentation, in which a top-level claim of safety is decomposed into sub-claims, definitions or other context, explanations, defeaters (assertions of a condition countering a claim), and still other sub-claims or evidence (see report by Feiler for more on this [6]). Nodes generally appear in the assurance case as a single sentence.

[8] One author reviewed each safety case (there was one safety case for each design alternative) twice and timed these reviews.

Table 2. Input values for Safety Case Inspection rate calculation

Safety case for:	Complexity	#Nodes	Review time (min.)
Design A	17	76	9.4, 11.0
Design B	16	40	5.7, 7.5

2.3.2.1.1 Safety Case Inspection Rate

Review rate information from table 2 (complexity value divided by number of nodes) is copied into the fifth column of the Safety Case Inspection (SCI) Rate Calculation table (see table 3) along with estimates of the number of pages per minute for code reviews (i.e., four pages per hour based on author experience), number of sentences per page (author estimate), and the number of safety case nodes per sentence (typically one). The first four values in each row in table 3 are multiplied to get the fifth (complexity reviewed per minute), which is then inverted, to arrive at minutes per unit of complexity.

Worst case calculations in the following section use the average of the two bold "min/Cxty" values using the SCI rates (i.e., 1.31).

Table 3. SCI rate calculation

SCI calculation						
	Pages /min	sentences /page	nodes /sentence	Cxty /node	=Cxty /min	Min /Cxty
Assumptions	4/60	40	1			
				17/76 =		
Design A	0.067	40	1	0.22	0.60	1.68
				16/40 =		
Design B	0.067	40	1	0.40	1.07	0.94

2.3.2.1.2 Safety Case Review Rate Calculation

Design complexity and review trial information for the stepper motor Example is also copied from table 1 into the top part of table 3. From this, the Safety Case Review (SCR) rate can be calculated, as presented in table 4, from two trials performed by one of the authors on each safety case from the stepper motor example (in the Task 5 report [8]). Note that this method does not use estimates about the number of nodes as the SCI calculation does.

Best case calculations in the following section use the average of the four bold "min/Cxty" values using the SCR rates (i.e., 0.51).

Table 4. SCR rate calculation

	Error Prop. Complexity	Trial 1 Minutes	Trial 2 Minutes
Design A	17	9.4	11.0
Design B	16	5.7	7.5

		Cxty/min:		min/Cxty
Case 1	Design A	1.81		0.55
Case 2	Design A		1.55	0.65
Case 3	Design B	2.81		0.36
Case 4	Design B		2.13	0.47

2.3.2.2 Best-Case and Worst-Case Review Times

Three categories of system design sizes were defined to estimate the time needed to review the safety case for a hierarchically designed large, complex system. Stepper motor is an example of small systems [8]; Wheel Brake system is an example of "medium" systems [7], and the SAAB-EII-100 system is an example of "large" systems [10].

The first two systems were used as minimum and maximum estimates in table 5 because they had two different designs and therefore different values for complexity. The third system had 203 error propagations, but the design was incomplete, a minimum of 200 and a maximum of 300 were assigned.

To calculate review effort for the best case, the minimum system design complexity number was multiplied by the best-case inspection review rate (time per complexity unit) determined above. To calculate review effort for the worst case, the maximum system design complexity number was multiplied by the worst-case inspection review rate.[9] Table 5 shows the best- and worst-case times to review a small, medium, or large system's safety case.

[9] Using the rate from code inspections gives a slower review rate than using the rate from author trials reviewing a safety case; therefore, the SCI rate is used as the minimum, and the SCR rate is used as the maximum.

Table 5. Design class and best/worst times to review

	Complexity number		Review Times: Best case (minutes)		Review Times: Worst case (minutes)	
	Minimum	Maximum	Mean SCR	0.51	Mean SCI	1.31
Small	16	17		8.1		22.2
Med	30	34		15.2		44.4
Large	200	300		101.3		392.1

2.3.2.3 Hypothetical Example System: Complexity and Review Time

This section characterizes a hypothetical system and calculates its overall complexity[10] and its review time. This hypothetical system consisted of 100 small-sized subsystems, 30 medium-sized subsystems, and 10 large-sized subsystems (see table 6).[11] The total complexity, to be used later,[12] and the review time per hazard, to be used now, were calculated. Thirty hazards and two modes per hazard were assumed for this hypothetical system. This resulted to a number consistent with the hypothetical SAAB-EII-100 example (i.e., 55 overall-system hazards total).

Table 6. Review time and total complexity for hypothetical system

Hypothetical Example			Total Cxty	Review Time/system: Min	Review Time/system: Max
Avionics has	100	Small-sized systems	1650	8.1	22.2
Plus	30	Medium-sized systems	960	15.2	44.4
Plus	10	Large-sized systems	2500	101.3	392.1
Hypothetical system			5110	Total	

For each size of subsystem, the total review time will be the number of hazards[13] times the number of modes (i.e., 60, in the hypothetical and simplified case), times the review time for each category of system size, times the number of systems of that size. These results are shown in the last two columns, based on the maximum and minimum review times from the previous step. Best case

[10] For the more general case, given an estimated range for the Complexity measurement of a subsystem design, for example, m..n, the Best/Worst-case Review Time has range 0.51*m..1.31*n.

[11] The authors used the mean complexity for each size system in calculating the Total Cxty column (Column 4).

[12] There is no reason for a midpoint calculation of the estimated range for Complexity as was used for the Total Cxty column of table 6 when estimating the complexity of a specific system. The authors used this midpoint for the specific purpose of coming up with a conversion from overall system complexity to review time for our hypothetical system.

[13] Every propagation event must be considered with respect to each hazard and each mode.

(minimum) and worst case (maximum) total review times for each kind of subsystem are given in table 7.[14]

Table 7. Review time for each size set of components

				Min	Max
Review Time		(minutes)	Small	48,600	133,313
			Med	27,338	79,988
			Large	60,750	235,257

2.3.2.4 Review Time Converted to Duration and Staff-Years

The minutes of review were converted to years, assuming that there are 250 work days for a year and only 180 minutes per day can be dedicated to review (i.e., other time is other activities including preparation for the review, certification meetings, and so forth) for each of four people on a review team, with three reviewing different material and the fourth acting as overseer (see appendix A for further description). Table 8 shows that for the hypothetical system, the small subsystems as a group take a minimum of 1.1 years to review (i.e., using best-case review rates) and a maximum of 3.0 years, the medium subsystems take 0.6–1.8 years, and the large subsystems take 1.4–5.2 years. The entire hypothetical system will take 3 (best case) to 10 (worst case) years to review. Again assuming a team of four, this means 12.2–39.9 staff-years must be allocated just to do certification review.

[14] In the general case, it is best to continue with the table described previously, extending it significantly as follows:
- Add one extra column for each system mode (i.e., the modes of the overall system).
- Add a single extra row to indicate the number of overall system-level hazards relevant to each system mode.
- For each subsystem, enter 0 in the cell corresponding to a particular system mode, if the subsystem is not active in that mode; otherwise, enter the same number as in the single extra row to indicate that all system-level hazards relevant to the corresponding system mode potentially apply to that subsystem (this number can be reduced by additional argument, but it is assumed to be the most direct-to-calculation case).
- Add a single extra column to indicate the total number of system-level hazards potentially relevant to that particular subsystem. This is the total of the cell entries described in the previous bullet.
- Add a single pair of extra columns. Each potential system-level hazard and system mode pairing constitutes a context for considering each error propagation and whether its occurrence could somehow trigger a chain of error propagations leading to the particular system-level hazard. Therefore, multiply the total number of relevant system-level hazards and mode parings (column described in previous bullet) by the estimated best/worst-case review times (columns 5 and 6) to obtain a minimum and maximum review time for each subsystem, analogous to the last two columns in table 7.

Table 8. Conversion to years and staff-years

	Best Case					Total	If Team of 4
X minutes	*1 day/180 min			*1 year/250 days		Years	Staff-Years
48,600	270.0	days	1.1	years	Small		
27,338	151.9	days	0.6	years	Med		
60,750	337.5	days	1.4	years	Large		
						3.0	12.2
	Worst Case						
X minutes	*1 day/180 min			*1 year/250 days			
133,313	741	days	3.0	years	Small		
79,988	444	days	1.8	years	Med		
235,257	1307	days	5.2	years	Large		
						10.0	39.9

2.3.2.5 Conversion Factor: Complexity Number to Staff-Years Of Effort

Because the total complexity number for this hypothetical system was 5110, the review times are 3.0–10.0 years, and resources needed are 12.2–39.9 staff-years, computing a typical conversion factor from complexity number to time (in years) and resources (in staff-years) is now possible.

Typical time (years) per complexity unit:

Minimum = 3 years/5110 = 0.59 years/1000 complexity units

Maximum = 10 years/5110 = 1.95 years/1000 complexity units

Typical resources (staff-years) per complexity unit:

Minimum = 12.2 staff-years/5110 = 2.4 staff-years/1000 complexity units

Maximum = 39.9 years/5110 = 7.8 staff-years/1000 complexity units.

This means that a system with Error Propagation Complexity of 1000 will take 0.59–1.95 years to certify (at four people full time) and 2.4–7.8 staff-years of effort. Systems of smaller and larger complexity should scale linearly due to the definition of Error Propagation Complexity and associated assumptions.

Note that because an Error Propagation Complexity number is defined as the number of ways that errors can propagate between one unit and another (and therefore a key factor for the number of different scenarios requiring safety analysis), these numbers also represent the typical effort required to resolve 1000 error propagation events.

2.3.2.6 Complexity of a System that is Borderline (Just Barely Assurable)

To determine the complexity of a system below which systems are likely to be assured with some degree of confidence and above which they most likely are not, an assumption about time and effort available for safety assurance must be made. Assuming the same as reported for the Boeing 787 example: 200,000 FAA staff hours at 2,000 staff hours/year (standard business value) expended over 8 years:[15]

FAA Certification staffing: 100 (= 200,000/2,000) staff years / 8 years = 12.5 full-time staff

It is assumed that half (6.25) full-time staff were dedicated to avionics.

This calculation is used to find what system complexity values correspond to 8 years and 6.25 staff (= 50 staff years).

Borderline Complexity = staff years / staff years per complexity unit, which is the value in the previous section.

Borderline Complexity = 50 / 7.8 * 1,000 ~6,400 (worst case) or 50 / 2.4 * 1,000 ~21,000 (best case)

Therefore, the complexity of a borderline-certifiable system is between approximately 6,400 (worst case) and 21,000 (best case).

The conclusion is that systems with a certification budget of 50 staff years and systems with complexity numbers under 6,400 should be considered assurable, those with complexity numbers between 6,400 and 21,000 should be considered borderline, and those with complexity numbers above 21,000 should be considered "too complex to assure safety."

2.4 RECOMMENDATION

This research tentatively recommends that an avionics system (with 50 staff years for assurance effort) whose Error Propagation Complexity (calculated by the formula in section 2.2.4) is less than 6,400 be considered not too complex to assure safety, between 6,400 and 21,000 be considered borderline, and whose Error Propagation Complexity is more than 21,000 be considered too complex to assure safety.

The "tentative" quality of this recommendation relates to the many qualities on which there was no available relevant research and for which assumed values had to be used. It is strongly recommended for future research to test some of the assumptions that went into the calculations listed in this report and in appendix B.

2.5 RECOMMENDED FUTURE RESEARCH

The following projects are recommended to continue this research.

[15] See appendix A for a fuller description and source for this example.

2.5.1 Apply and Validate Error Propagation Complexity Formula

In FY16, the SEI team developed a complexity formula for estimating the potential size of an argument (or assurance case) about system safety for use as a proxy when a design might be too complex to assure safety. Applying the formula involves examining a system design and determining the number of cases in which an FC can propagate from one component of the system to another. The formula was exercised on two designs each for two small but detailed examples: 1) a stepper motor controlling engine thrust and 2) a wheel brake system. The formula was found to be easy to apply and was related to various existing complexity measures. Inter-rater reliability was also found to be high. Guidance was included on how to apply the formula on a larger scale design typical of those in applicant submissions for certification and on how to apply when a design is incomplete. However, the guidance has not been tested, and the formula has not been exercised on a full-scale aircraft design description.

For FY17, it is proposed to pilot the complexity formula on a real program at a real-life scale. If it is too difficult to get an "applicant" (e.g., aircraft manufacturer) to provide the FAA with current data, it is suggested to use a historical/closed program. The following questions need to be answered:

1. Scale up and feasibility: What issues arise when you scale up the size of the system whose complexity is being estimated? How can the formula best be applied given the various plans and artifacts included in an applicant's submission for aircraft/supplemental type certificate certification? What is the level of effort involved in such application?
2. Calibrate and validate formula-based thresholds: How closely do formula-produced estimates relate to actual time and costs incurred by a designee in verifying correctness relative to the system requirements? Do the thresholds agree with expert FAA designee experience as to when applicant errors occur or when further interaction with the applicant may be required?

Finally, to help the FAA move forward with deploying this tool for estimating complexity of a safety argument, the research addressed what guidance, consistent with current standards, should be given to contractors in the following areas:

1. What is the process in recognizing when particular parts of the design (particular subsystems or particular subsystem interactions) are very complex, and what preventative steps might be taken?
2. What additional verification and validation activities should be applied?
3. Should subsystem or interaction-focused assurance cases be required specifically when complexity exceeds some threshold?

2.5.2 Extend Today's Complexity and Safety Work

To understand more about special cases and assumptions made in the safety case, the below questions, addressing the viability of the complexity formula and to what extent it can be used, were posed:

1. Consider the difficulties of a certifier attempting to understand the complexity of a proposed system and its model. There is a fundamental limit to human cognition that

restricts people's ability to determine a big picture from small pieces. The research has suggested that creating a safety argument is very difficult ("NP-exponential"), but verifying that someone has asserted it correctly is less difficult ("NP-polynomial"–well behaved). The SEI would model how hard it is to verify any claim, based on human short-term memory capacity (there is a bandwidth limitation to integrating models in our mind). It is suggested that the complexity formula be augmented to reflect this.

2. Complexity that challenges the ability of humans to reason about the situation is often mitigated by a combination of formalization, tools, education (which takes time), decomposition into subclaims, and review of pieces of the safety case with others. These steps are taken by designee or certification authority personnel, therefore design complexity and fault-model complexity drive effort. However, mitigations that are put in place to make the system safer also drive effort, so that total certification effort calculations must include efforts for both.
3. As part of this, investigate the limitations on determining a global attribute like "safety" from a number of local attributes. Although there are continuously many little arguments such as evidence X proving subclaim Y, no such little argument is 100% proved; in general, certainty in a little argument is often based on a preponderance of evidence. There is a small probability that the little argument is actually erroneous and therefore does not lead to belief that the system is safe. This research asks the question, "How do probabilities and uncertainties propagate through the big model, the system safety case?" In particular, when all the analysis is there in black and white, are there any "black swans" (highly unlikely situations that would have high consequence if they did occur)?
4. (An alternative approach, not yet worked out): How can the quality and appropriateness of the inputs into the formula be assessed? Is it possible to compare safety cases from one applicant to another? How will the data be normalized?

2.5.3 Expand the Fault Model

This research approached complexity and safety based on a safety-oriented fault model that assumes the failure of components and the propagation of these failures to other components. Whereas this approach encompasses many safety hazards, others do not fit within this traditional fault model. It is therefore proposed to extend this approach to cover mishaps due to emergent behavior, concurrency, and cybersecurity.

Specifically, the research will address faults and failures due to the unexpected emergent behavior resulting from the interaction of properly functioning components. It will also address hazards due to standard concurrency defects such as race conditions, starvation, deadlock, livelock, and priority inversion due to the use of multiple processors (e.g., multicore and multiple interconnected computers). Cybersecurity-related hazards will be addressed by looking at the system with Nancy Leveson's Systems Theoretic Process Analysis (STPA) and extending the STPA by interpreting the nodes and arcs of the Systems Theoretic Accident Model and Processes model as constituting an attack surface; through this surface, the integrity of safety-critical messages and data can be attacked and access to safety-critical services can be denied. Although some of the challenges described may have answers based on straightforward extrapolations of the existing work, others might not.

2.5.4 Additional Research Ideas

2.5.4.1 Optional Work Not Completed

The statement of work for this research project included the following optional tasks. They should be completed as an extension of this work.

- Guidelines on practices for safety certification–This task suggests ways to approach safety certification of systems that are borderline or too complex according to the recommendation in section 2.4. Guidance includes ways to keep estimates of complexity for a system updated during various system phases (e.g., requirements, specification, implementation, validation and verification, and sustainment) and methods and approaches to overcome the underlying complexity. A prototype tool was also suggested to identify, from development artifacts, what the complexity is (both amount and type) and propose which strategies for overcoming the complexity would be most fruitful.
- Design Guidelines to improve safety through the reduction of software complexity–This would address improvement of the software development process so that complexity is identified and reduced throughout the development life cycle. The report may include a checklist to determine whether software should be developed at all based on estimates of resources required to build, test, validate, and certify it.

2.5.4.2 How to Address Precedents?

It seems reasonable that an avionics module that has already been certified represents less risk than one that has just been built, and one can say the subjective complexity of the tested module is lower than the one with an equivalent number of piece parts and lines of code that has not completed the certification process. How can this factor be expressed mathematically? Can "subjective complexity" be quantified at all, and if so, how are the quantified values validated? More germane to the point, how should one account for the fact that a given module is in use and has been validated? Does this actually reduce the complexity, or does the interaction complexity of that module with other new modules swamp the individual model's complexity?

2.5.4.3 Benefit of Organizing an Assurance Case

The complexity formula generates a number that is indicative of the complexity of the task of reviewing a system. Setting a bound for when a system is too complex to reasonably review depends on a number of variables. One of these is the manner in which review documents are organized and delivered. Reviewing a stack of evidence is much more difficult than reviewing a stack of evidence that has been organized into an argument showing how that evidence supports the claim of aircraft (or subsystem) safety (an assurance case). In the former case, a complexity number of N might be consider to be too complex to review, whereas it could be handled easily in the latter case. Research should be performed to understand the exact benefit of organizing the arguments and evidence for safety in an assurance case. It seems clear that the provision of an assurance case will make complex review activities much easier to handle than studying miscellaneous evidence that is poorly organized.

2.5.4.4 Architectural Level of Detail

If one looks only at a high level of detail, one will see less complexity than if one analyzes a fully finalized and implemented model of every component in the system. Is there something that can be said relevant to the increase in measured complexity to be expected with every additional layer of system modeling that one completes? If so, then can this "complexity expansion" be predicted and perhaps guidelines be created that keep this complexity from ballooning too much? Does the expansion, perhaps variable in the first few layers, then settle down to a more predictable number? (If so, then what is the first layer after which complexity is predictable?)

3. CONCLUSION

This report has reviewed the research to date on the FAA Complexity and Safety Project. As documented in the first three internal reports, the Software Engineering Institute (SEI) investigated the types, causes, and effects of complexity per the literature; surveyed and reported on possible metrics of complexity; and discussed aspects of complexity, safety assurance, and certification as they apply to avionics.

As documented in the two most recent internal reports, the SEI created and tested a formula for the complexity of an avionics system that relates safety to complexity by assessing the complexity of a safety case in a manner that can apply early on when the only available data about the system is an architectural description (e.g., parts, connections, and propagation points) and a fault model. This kind of complexity is called "Error Propagation Complexity." The formula assesses the number of ways in which a failure or fault can propagate from one high-level component to another, representing the number of "propagation event" discussions that must be held (or reviewed) to resolve whether such propagation could result in a lack of safety.

In this report, the SEI documents how this formula was applied to small, medium, and large system designs. For the first two systems, inter-rater reliability was measured and was proven high. For the third case, the complexity of the large system was assessed as one step in a series of calculations meant to tie the other two examples to real avionics.

Assuming a number of assumptions are true, including a budget of 50 staff years for avionics assurance review, avionics systems that have an Error Propagation Complexity above 21,000 should be considered too complex to assure safety. Those with an Error Propagation Complexity above 6,400 but below 21,000 should be considered borderline, because they may or may not be certifiable depending on where in the best-case to worst-case spectrum their projects lie. Those below 6,400 should be considered certifiable, although the amount of certification effort predicted by the formula should be compared to the actual budget available.

The SEI is grateful for the opportunity to perform this research and hopes to do some of the necessary "further research" to gain insight into many factors whose contribution at this point had to be assumed.

4. REFERENCES

1. Konrad, M., & Sheard, S. (2015). *FAA research project on system complexity effects on aircraft safety: Literature search to define complexity for avionics systems.* Software Engineering Institute, Carnegie Mellon University. Retrieved from http://resources.sei.cmu.edu/asset_files/WhitePaper/2016_019_001_484321.pdf.

2. Sheard, S. (2012). *Assessing the impact of complexity attributes on system development project outcomes* (Doctoral dissertation). Stevens Institute of Technology. Retrieved from seir.sei.cmu.edu/sheard/SarahSheardDissertationSubmitted2.pdf.

3. Nichols, W. R. & Sheard, S. (2015). *FAA research project on system complexity effects on aircraft safety: Candidate complexity metrics.* Software Engineering Institute, Carnegie Mellon University. Retrieved from http://resources.sei.cmu.edu/asset_files/WhitePaper/2016_019_001_484344.pdf.

4. Sheard, S., Weinstock, C., Konrad, M., & Firesmith, D. (2015). *FAA research project on system complexity effects on aircraft safety: Identifying the impact of complexity on safety*. Software Engineering Institute, Carnegie Mellon University. Retrieved from https://resources.sei.cmu.edu/asset_files/WhitePaper/2016_019_001_484330.pdf.

5. SAE International. (2015). SAE AS-5506/1A:2015, *SAE Architecture Analysis and Design Language (AADL) Annex Volume 1: Annex A: ARINC653 Annex, Annex C: Code Generation Annex, Annex E: Error Model Annex*. 2015.

6. Feiler, P., & Gluch, D. (2012). *Model-based engineering with AADL: An introduction to the SAE architecture analysis & design language*. Part of the SEI Series in Software Engineering series. Addison-Wesley Professional. ISBN-10: 0-321-88894-4.

7. Konrad, M., Sheard, S., Weinstock, C., & Nichols, W. R. (2016). *FAA research project on system complexity effects on aircraft safety: Testing the identified metrics.* Software Engineering Institute, Carnegie Mellon University. Retrieved from http://resources.sei.cmu.edu/asset_files/WhitePaper/2016_019_001_484340.pdf.

8. Konrad, M., Sheard, S., Weinstock, C., & Nichols, W. R. (2016). *FAA research project on system complexity effects on aircraft safety: Estimating complexity of a safety argument*. Software Engineering Institute, Carnegie Mellon University. Retrieved from http://resources.sei.cmu.edu/asset_files/WhitePaper/2016_019_001_484336.pdf.

9. RTCA, Inc. (2012). *DO-178C Software Considerations in Airborne Systems and Equipment Certification*. Retrieved from http://www.rtca.org/store_product.asp?prodid=803.

10. Peterson, E. M. (2015). *Application of SAE ARP4754A to Flight Critical Systems.* NASA. Retrieved from http://ntrs.nasa.gov/search.jsp?R=20160001634

APPENDIX A—SUMMARY OF PREVIOUS REPORTS

This section describes the five previous reports delivered as part of this research effort. All of them are internal reports deliverable only to the FAA. Readers interested in seeing them should request copies from the FAA.

A.1 REPORT 1. FAA RESEARCH PROJECT: SYSTEM COMPLEXITY EFFECTS ON AIRCRAFT SAFETY. TASK 3.2: LITERATURE SEARCH TO DEFINE COMPLEXITY FOR AVIONICS SYSTEMS (VERSION 2) [A-1]

This report addressed what kinds of things could be considered "complex" (according to the literature) and in what ways they could be considered complex. In this systematic literature search, the author found that, although many problems are blamed on "complexity," the term is usually left undefined. As a result, the author modified the focus of the task from simply collecting definitions to describing a taxonomy of issues and general observations associated with complexity.

The literature review revealed that complexity is a condition associated with trying to understand the cause-and-effect relationships within a system. There is a large taxonomy of different kinds of causes and another taxonomy of different kinds of effects. To prevent the impacts that complexity creates, one must reduce the causes of complexity.

Causes of complexity include the following:

- Causes related to system design (the largest group of causes)
- Causes that make a system seem complex (i.e., contributors to cognitive complexity)
- Causes related to external stakeholders
- Causes related to the system requirements
- Causes related to technological change
- Causes related to teams

Effects of complexity include the following:

- Increases confusion, likelihood of error, and excessive optimism
- Makes the design process harder
- Fails to provide adequate safety coverage of high performance, real-time systems as existing design and analysis techniques get more complex
- Makes project planning more difficult
- Makes it harder to predict system properties from the properties of the components because emergent behavior arises and because system behavior can be unstable and non-replicable
- Makes it harder to identify, report, and diagnose problems
- Makes it harder for people to follow required processes
- Drives up verification and assurance efforts and reduces confidence in the results of verification and assurance
- Makes changes more difficult (e.g., in software maintenance) and makes it harder to service the system in the field because the system is hard to understand.

The report includes descriptions of each of the causes and effects, and an appendix with the detailed taxonomy of complexity causes, effects, measurement, and mitigation.

The literature review also looked at the literature on measuring and mitigating complexity, in preparation for future tasks. Regarding measurement, no framework exists for measuring all of the complexity that could be relevant to aircraft safety, so developing such a framework, or at least a measurement, is an important step in the next task. Regarding mitigation, most good practices for systems and software engineering are good to the extent that they reduce and mitigate complexity. Promising mitigation activities include problem, system, and software modeling; data abstraction and typing; process discipline; incremental assurance; and early attention to assessment and reduction of complexity.

After submitting Version 1 of this report, the authors rewrote it to address FAA reviewer comments and delivered Version 2.

A.2 REPORT 2. FAA RESEARCH PROJECT: SYSTEM COMPLEXITY EFFECTS ON AIRCRAFT SAFETY. TASK 3.3: CANDIDATE COMPLEXITY METRICS [A-2]

This report listed a large number of candidate metrics that had been gathered from the literature of software, complexity science, systems engineering, and other fields. The point was not to determine metrics to be used but rather to understand the breadth of metrics that would be possible to use.

This report offered a definition of complexity tailored for this research project:

> Complexity is a state or quality of being composed of many intricately interconnected parts, in a manner that exceeds the ability of humans, supplemented by tools, to understand, analyze, or predict behavior.

Thirty-five candidate metrics were described in a table, along with notes about the utility of each. Each is further described in the text. At the time, there was a consideration to create a composite measure of complexity that would build on several of these candidate metrics. Many were not directly countable; the ones that were assessed as countable ended up eventually toward the selected metric called “Error Propagation Complexity.”

The report showed how complexity measures for components, component interfaces, and for subsystems, systems, and related interfaces will have to be combined to predict something about the safety case. A section shows how Nancy Leveson’s notion of control loops is related to the complexity measures.

The notion that the difficulty of proving the safety case would be a good measure of complexity for the purpose of determining aircraft or avionics safety can be seen in nascent form in this report. Eventually the concept of Error Propagation Complexity, and its estimate in the early system life cycle, was chosen to compute the value of a safety-related measure of complexity.

A.3 REPORT 3. FAA RESEARCH PROJECT: SYSTEM COMPLEXITY EFFECTS ON AIRCRAFT SAFETY. TASK 3.4: IDENTIFY THE IMPACT OF COMPLEXITY ON SAFETY (DRAFT REPORT, REVISED) [A-3]

This report was, per contract, a draft report that had a skeletal form consisting mostly of a call to the appendices, in preparation for the next report, which defines how to calculate Error Propagation Complexity.

The appendices included definitions of safety and assurance cases, requirements, claims, and evidence, and factors in a hazard mitigation-focused versus safety-requirements-focused approach to determining safety. It describes the difference between early and late life-cycle assurance cases including augmentation with evidence. It provides knowledge from the software work about the generation and detection of software errors, along with the relationship of that question to complexity, and of typical software concerns such as scheduling, redundancy, and the inability of tests to assure there are no remaining flaws. One section addresses cognitive aspects of certification, including the mental representations required for a certification authority to determine that the case for declaring an aircraft safe is acceptable.

A revision to this report was delivered because the customer asked for a more technical Executive Summary.

A.4 REPORT 4. FAA RESEARCH PROJECT: SYSTEM COMPLEXITY EFFECTS ON AIRCRAFT SAFETY. TASK 3.5: ESTIMATING COMPLEXITY OF A SAFETY ARGUMENT [A-4]

This report presented the formula for computing the Error Propagation Complexity of an avionics system, based on a high-level architecture and an understanding of possible errors. The report called this quantity just “Complexity;” the exact name, “Error Propagation Complexity,” came later.

As was evident from Report 1, there are many varied definitions of complexity; the challenge was to find or create a measure that related complexity to safety. It was decided that complexity in this situation is best represented by how much work would be required to review an assurance of safety. A much more complex system will necessarily be harder to assure as safe. The size of the actual safety case would be a good indicator of its complexity, and therefore a reasonable rationale for drawing a line and saying, “Systems more complex than this line will be too complex to assure safety,” as per contract request.

By the time the size of the safety case is accurately known, the program will be complete. A means to estimate this size was needed much earlier. It was eventually recognized that for every propagation of an error, there would need to be a follow-up analysis to determine whether anything unsafe happened as a result of that propagation and to create a way to count these propagations. The sum would be multiplied by a notional, average, or calibrated parameter estimating for how long each of these many cases would take to resolve.

The formula is repeated here in this final report, although guidance on how to apply it was modified a bit during the next step: testing of the formula on a (somewhat) larger system.

A.5 REPORT 5. FAA RESEARCH PROJECT: SYSTEM COMPLEXITY EFFECTS ON AIRCRAFT SAFETY. TASK 3.6: TEST THE IDENTIFIED METRICS [A-5]

This report defined the type of complexity as "Error Propagation Complexity" and applied it to a second example, a Wheel Brake System. The method used to assess Error Propagation Complexity for the Wheel Brake System was to obtain the architecture model of the entire Wheel Brake System, simplify the interconnections, then (as in Report 4) count the ways that errors could propagate from one element to another. The initial model is shown in Figure 3 of Report 5, and the model with simplified interconnections is shown in Figure 4 of Report 5.

The architecture was simplified to focus on aspects important to application of the Error Propagation Complexity formula, namely: modes, components, propagation points, failure conditions, and fan-out. As before, the Error Propagation Complexity formula essentially estimates the size of the safety case: assuming an average analysis time for the follow-through to determine whether a failure can propagate in an unsafe manner, the estimate of total time for safety case analysis can be created by multiplying this average time per failure propagation by the number of ways a failure can propagate, which is estimated by our formula for Error Propagation Complexity.

This report showed that the formula for Error Propagation Complexity can be applied consistently to multiple well-defined architectures and results in reasonable answers. It also showed how to simplify the formal model in a way that makes it easier to calculate Error Propagation Complexity (although automated approaches are recommended).

A.6 REFERENCES

A-1. Konrad, M. & Sheard, S. (2015). *FAA research project on system complexity effects on aircraft safety: Literature search to define complexity for avionics systems*. Software Engineering Institute, Carnegie Mellon University. Retrieved from http://resources.sei.cmu.edu/asset_files/WhitePaper/2016_019_001_484321.pdf.

A-2. Nichols, W. R., & Sheard, S. (2015). *FAA research project on system complexity effects on aircraft safety: Task 3.3: Candidate complexity metrics.* Software Engineering Institute, Carnegie Mellon University. Retrieved from http://resources.sei.cmu.edu/asset_files/WhitePaper/2016_019_001_484344.pdf.

A-3. Sheard, S., Weinstock, C., Konrad, M., & Firesmith, D. (2015). *FAA research project on system complexity effects on aircraft safety: Identifying the impact of complexity on safety*. Software Engineering Institute, Carnegie Mellon University. Retrieved from https://resources.sei.cmu.edu/asset_files/WhitePaper/2016_019_001_484330.pdf.

A-4. Konrad, M., Sheard, S., Weinstock, C., & Nichols, W. R. (2016). *FAA research project on system complexity effects on aircraft safety: Estimating complexity of a safety argument*. Software Engineering Institute, Carnegie Mellon University. Retrieved from http://resources.sei.cmu.edu/asset_files/WhitePaper/2016_019_001_484336.pdf.

A-5. Konrad, M., Sheard, S., Weinstock, C., & Nichols, W. R. (2016). *FAA research project on system complexity effects on aircraft safety: Testing the Identified Metrics.* Software Engineering Institute, Carnegie Mellon University. Retrieved from http://resources.sei.cmu.edu/asset_files/WhitePaper/2016_019_001_484340.pdf.

APPENDIX B—MATHEMATICAL ARGUMENT BEHIND SECTION 2.3

This appendix calculates the acceptable complexity numbers for a system if it is to be considered to be approvable and borderline versus too complex for safety to be assured. First, the answer, was given, including some of the uncertainties in the calculations. Then the descriptions of various factors in the calculations were developed, including the numbers used and their respective sources. Because this research is groundbreaking, many of the numbers going into this calculation could only be estimated. In the future, some may have more concrete data from measured evidence. Additional research should be performed to solidify knowledge of some of the factors.

Note: Whereas the calculations performed here are all justified, discussion subsequent to this writing led to the slightly-different values in the main text. In this appendix, the authors have left the original calculations as is to provide much of the rationale that did not fit in the main text and have included the numbers the authors agreed to use in the main text in parentheses.

B.1 A FORMULA-DRIVEN TAKE ON THE PROBLEM

The answers to three more precise formulations of how to use the formula on an aircraft design, starting with a very narrow, short-term perspective and broadening to the more general were explored:

1. Given a properly elaborated safety case, how long would it take to review the safety case to determine that a particular system hazard is impossible or very improbable?
2. Given a single-diagram-based system design (with separate error model) and verification results, how long would it take to review these to determine that a particular system hazard is impossible or very improbable?
3. Given the design and verification results (safety case evidence) for an avionics system (and its components), how long would it take a certification authority review (CAR) team to review these to determine that an entire set of system hazards are very improbable?

B.2 AN IMPORTANT CAVEAT

The answers were based on estimates derived from very limited data. The real goal is not in the answers themselves, but in identifying the particular set of factors, which if known, would help establish more precise answers to these questions. Whereas ranges and thresholds from very limited experimental data in this report have been derived, it must be emphasized that there is such high uncertainty around many of these factors, especially in the answer to the third question, that the ranges given for thresholds need to be interpreted very cautiously. When applying the formula in an actual situation, some of these identified factors will be much more precisely known or can be estimated, resulting in range estimates having some practical value in estimating the total-review effort that will be involved. Additional research could also be conducted to provide better guidance in the general case (see section 2.5).

B.3 TIME TO REVIEW A SAFETY CASE

The question, again, is this:

1. Given a properly elaborated safety case, how long would it take to review the safety case to determine that a particular system hazard is impossible or very improbable?

Properly elaborated safety case means that an argument has been constructed by the applicant on why the system hazard is very improbable based on: 1) features of the design and error model, and 2) verification activities performed. The argument makes careful reference to both in establishing a case as to why all possible conditions that might result in the system hazard cannot occur, or are very unlikely. The argument that relevant failure conditions (FCs) (i.e., those that can give rise to the system hazard) are themselves very improbable may be supported by a simulation, a test result, and/or a probability argument.

Here are factors affecting the time it takes to review a safety case, which are excluded in the estimate of review time:

- Reviewer has relevant domain knowledge and experience–The reviewer needs to understand the system design and how the system is supposed to operate sufficiently to identify and reason about possible hazards and possible causes.
- Time to develop confidence in the results of the verification activities reported–It takes the reviewer time to assess the verification evidence provided and become confident that it shows what it purports to show.

The review time estimates provided in table B-1 were based on only two trials of the same two safety cases and are only intended to provide an initial rough calibration (more information about the stepper motor designs, error models, and safety cases on which these data are based is found in the report for Task 6 [B-1]).

Table B-1. Review time estimates from two trials of two safety cases

Safety case for. . .	Complexity	#Nodes	#Words	Review Time (min.)
Design A	17	76	1169	9.4..11.0 (10.2 mean)
Design B	16	40	468	5.7..7.5 (6.6 mean)

This suggests a range of approximately 0.4–0.6 minutes review time per unit of complexity. This is called the Safety Case Review (SCR) Rate.

B.4 HOW DOES THIS SCR RATE COMPARE WITH ORDINARY READING RATE AND DESIGN INSPECTION RATES?

- Ordinary reading rate–The average reading rate is approximately 200–400 words per minute [B-2]. Assuming 200 words per minute because the material is relatively difficult to understand in places, and referring to the #Words column, it is expected that the safety case for Design A and 2.3 minutes for Design B can be read in approximately 5.8 minutes.

Dividing this by their respective "Error Propagation Complexity" results in a range of 0.15–0.34 minutes reading time per unit of complexity. This is called the Safety Case Ordinary Reading (SCOR) Rate.

- Inspection rate–According Bill Nichols, one of the co-authors of this report, detailed reviews of software design documentation is approximately four pages per hour. At 40 sentences per page, that works to 160 sentences per hour. Equating safety case nodes with sentences, then referring to the #Nodes column, it is expected that it will take approximately 28 minutes to inspect the safety case for Design A and 15 minutes to inspect the safety case for Design B. Dividing these by their respective Error Propagation Complexity results in a range of 0.9–1.6 minutes inspection time per unit of complexity. This is called the Safety Case Inspection (SCI) Rate.

Therefore, based on limited data, the SCR rate is approximately half the SCOR rate but approximately three times the SCI rate. Why is the SCR rate much lower than the SCI rate, which it presumably resembles? In addition to the small sample size being an issue, it is possible that the review times were conducted more from a perspective of comprehension and less from a perspective of detecting defects, because the trials were conducted from previously reviewed confidence maps. If so, the time reported might under report what a more careful review needed to detect defects might require. It is also possible that a properly elaborated safety case, which organizes verification-related information, together with referenced design and error model features into the structure of a concise and organized argument, might actually reduce the time needed to comprehend and review for defects. Only more experimentation will determine which is the case, which should include a measure of defects detected.

Table B-2 shows three estimates of the time it takes to review a safety case based on three different approaches to estimating the rate:

Table B-2. Time required for safety case review using three approaches

Review Rate per Unit of Error-Propagation Complexity:	Min	Max	Mean
SCR rate	0.4	0.6	0.5
SCOR rate	0.15	0.34	0.24
SCI rate	0.9	1.6	1.2

Two of these three rates, namely the empirically-derived review rate (SCR) and experientially observed inspection rate (SCI) will be referred to in the following sections.

B.5 SINGLE-SYSTEM DESIGN AND ONE SYSTEM HAZARD

The question:

1. Given a single-diagram-system design (with error model) and verification results, how long would it take to review these to determine that a particular system hazard is impossible or very improbable?

A single-diagram-system design is a design that can be presented as a single (flat) diagram on a standard-size sheet of paper versus a design that is best depicted using multiple diagrams: one diagram for the top-level system and additional diagrams for selected subsystems. Therefore, 25–30 components in the diagram were chosen for reasons of legibility, because anything more than that might be best represented hierarchically and not as a single diagram (no assumptions about the error model size or verification results were made when "single diagram" was used).

The team has studied designs in three size ranges (with some variation around sizes and complexity given the different purposes for the examples; see table B-3).

Table B-3. Time to review single diagram system designs in three size ranges

Design Class (and Example System)	#Components	Error Propagation Complexity
Small (stepper motor System)	5	16-17
Medium (Wheel Brake System)	17-20	30-34
Large (SAAB-EII 100 [B-3])	28	~ 200 (flight mode only)

B.6 CALCULATING REVIEW TIME FOR EACH "UNIT OF ERROR PROPAGATION COMPLEXITY"

To estimate the time required to review such a single-diagram-system design, a best-case/worst-case approach for each design class was taken.

For best case, the minimum empirically derived SCR rate of 0.4 minutes per complexity unit was assumed. This is best case also because this rate is derived assuming a properly elaborated safety case has been provided, which organizes the system design into a form that helps the reviewer directly evaluate the likelihood of a system hazard.

For worst case, the maximum experientially observed SCI rate of 1.6 minutes per complexity unit (C-unit) was assumed. This is worst case also because this rate corresponds to that achieved when reviewing software designs with the purpose of detecting defects. In summary:

Time to review a single-diagram design per unit of complexity: 0.4–1.6 mins/C-unit

The derived review times for each design class (i.e., multiplying the worst-case-to-best-case range identified immediately above by the estimated mean number of complexity units for that design class) are shown in table B-4.

Table B-4. Time to review single diagram system designs per unit of complexity

Design Class (and Example System)	Review Times
Small (stepper motor System)	6.6..26 minutes
Medium (Wheel Brake System)	13..51 minutes
Large (SAAB-EII 100 [B-3])	80..320 minutes

These review times must be interpreted very cautiously. Issues include the following:

- The review times for each class are not based on data from industry.
- The medium-class review time is derived from a design that was probably incomplete (i.e., the error model for most components in the design indicated only one to two FCs).
- The large-class review time is based on an incomplete design with minimal information about almost all components [B-3].

B.7 AVIONICS SYSTEM DESIGN AND MULTIPLE SYSTEM HAZARDS

The question:

1. Given the design and verification results for an avionics system (and its components), how long would it take a CAR team to review these to determine that an entire set of system hazards are very improbable?

The answer to this question is based on many factors: what is expressed quantitatively must be treated as very conjectural.

B.8 FACTORS THAT AFFECT HOW MUCH TIME IT TAKES TO REVIEW AN AVIONICS SYSTEM DESIGN AGAINST A SET OF SYSTEM HAZARDS

- Quality of the material submitted as part of certification:
  - The degree to which the error model is detailed and complete and sufficiently granular to support reasoning about the likelihood of particular system hazards
  - Whether a properly elaborated confidence map has been provided
  - The mix of components sizes (i.e., how many large, medium, small components)
    - Precedentedness (i.e., off-the-shelf and proven versus unprecedented)
- Whether the review rates identified above are representative: 0.4–1.6 mins/C-unit
- Number, heterogeneity, and volatility of system hazards that need to be evaluated
- Number, heterogeneity, and volatility of system modes that need to be considered
- Number of times that (new or revised) design baselines need to be reviewed
  - Volatility in the designs (and error models) and verification reports
  - Volatility in the technology and components used
- Capability of CAR team to sustain reviews with a defect-detection mindset in a managed, measured, and well-paced and efficient way
  - Assumes reviewers have strong domain knowledge and relevant experience
  - Assumes knowledge of conditions that lead to high-quality reviews with high-defect yield
    - Not trying to review too much in one day
    - If necessary, seed designs with defects that help sustain reviewer attention

    - Multiple reviewers evaluate same designs for selected critical-defect types
    - When results of earlier reviews can be reused versus review from scratch

- Capability of the applicant's design and verification teams to achieve and sustain the development and evolution of high-quality and well-verified designs

    - Capability to also manage a diverse supply chain

Given these factors that drive review time (and others not mentioned), establishing general lower and upper thresholds for Error Propagation Complexity measure has the following problems:

- Below the lower threshold, there is probably no issue reviewing the submitted design against a set of system hazards.
- Between the lower and upper threshold, the situation is highly uncertain.
- Above the upper threshold, the submitted design is probably too complex to adequately evaluate it against a set of system hazards.

However, by specifying one's assumptions about the previous list of factors based on an actual aircraft certification project, supplemented with some empirical research (e.g., confirming the estimated worst-case/best-case review rates), reasonable thresholds might possibly be achievable. In the absence of such an example and the empirical research needed to more firmly establish the assumptions, a hypothetical example was used instead.

B.9 ASSUMPTIONS RELATED TO THE ABOVE FACTORS

Assume a complete avionics system design review situation that has these parameters:

- CAR team is very knowledgeable, experienced, and has a very capable review process.
- Applicant and suppliers use capable designers, verifiers; and mature engineering processes.
- The hierarchy of subsystem and component designs consists of this mix of design size:

    - 10 large-size designs
    - 30 medium-size designs
    - 100 small-size designs

- Designs, verifications, and technology will not change for the duration of the review.
- A set of system hazards of size 60 exists (assumed equivalent to 30 distinct safety cases).

    - The example in reference [B-3] assumes approximately 55 system hazards
    - CAR team's review process is sufficiently capable to correctly exploit reuse possibilities

- An average of two system modes must be considered.

    - Half of components (in each size class) require only one mode be considered (i.e., Flight).
    - The other half require all three modes be considered (i.e., Climb, Mid-Flight, and Descent)

- Furthermore, the following review-rate parameters for the CAR team as a whole were assumed:

    - One review team member seeds the material to be reviewed (to estimate and monitor by capture-recapture the quality of the reviews).
    - Approximately 180 minutes of review time is spent per day (two 90-minute periods), day after day.
    - Three reviewers focus on different aspects of same design material (intentional overlap).

The above hypothetical example was then subjected to a best-case analysis to identify a lower threshold and a worst-case analysis to identify an upper threshold. However, the team's daily review rate must be determined first.

B.10 SUSTAINABLE REVIEW RATE

Sustainable review rate for a team of four full-time reviewers using a mature, measured, monitored, and high-defect yield review process: Each reviewer can spend no more than 180 minutes of review time/day.

- Years of experience with reviewing software designs indicates that the "sweet spot" for the number of inspectors needed to identify a large percentage of the defects is three; therefore, the 180 minutes of review time/day is for each of the people on the team, not just a single reviewer.

Dividing the number of review minutes (presented in table B-5) needed by the sustainable daily review rate for the team, an estimate of the number of years that such a review would encompass in calendar time was obtained:

- Worst case: 440K review min / (180 min/day) = 2.4K days or approximately 10 years calendar time (and 40 full-time equivalent years of person effort)
- Best case: 111K review min / (180 min/day) = 0.6K days or approximately 2.5 years calendar time (and 10 full-time equivalent years of person effort)

Table B-5. Volume of review material and number of review minutes needed by the CAR team

Design Class (and Example System)	Number of Items Reviewed	Worst Case	Best Case
Small (stepper motor System)	100 design items * 30 hazards * 2 modes = 6000 small-size design reviews	6000 reviews * 26 min = 156K review min	6000 reviews * 6.6 min = 40K review min
Medium (Wheel Brake System)	(Likewise) 1800 medium-size design reviews	1800 reviews * 51 min = 92K review min	1800 reviews * 13 min = 23K review min
Large (SAAB-EII 100 [B-3])	(Likewise) 600 large-size design reviews	600 reviews * 320 min = 192K review min	600 reviews * 80 min = 48K review min
Total	140 items reviewed 60 times each for different hazard and mode combinations = 8400 reviews	440K review min = x years, x hours	111K review min

In actuality, many of the assumptions made (e.g., that there be no changes) are unrealistic. In addition, if both reuse of reviews and automation of some parts of the review process can be introduced, then the total elapsed calendar time for the reviews would be less.

Conversely, if there is wait time and changes made, then the encompassed calendar time would be greater.

What is the total Error Propagation Complexity of the above hierarchical, mixed-size system design?

- 100 small-size components of Error Propagation Complexity 16.5 (mean) = 1.6K C-units (approx.)
- 30 medium-size components of Error Propagation Complexity 32 (mean) = 1.0K C-units (approx.)
- 10 large-size components of Error Propagation Complexity 200 (approx.) = 2K C-units (approx.)
- Total Error Propagation Complexity for the hypothetical system = 4.6K C-units

Therefore, under the previous assumptions, a design of approximately 20K C-units was estimated to be too much to review, which led to an upper threshold of approximately 20K C-units to be set up.

Under worst-case situation, a design of approximately 5K C-units would also take approximately 10 calendar years to review, but that was under optimistic assumptions; perhaps approximately half that amount might be sustainable.

B.11 NOT TOO COMPLEX, BORDERLINE, AND TOO COMPLEX

Therefore, under the given hypothetical assumptions not directly based on any actual example, the following can be made for the total system design package if the error-propagation complexity is in the specified range:

- Below 2.5K C-units = probably can sustain a quality review of all the safety cases
- Above 20K C-units = probably cannot sustain a quality review of all the safety cases
- 2.5K–20K C-units = highly uncertain

B.12 VALIDATION: COMPARISON TO ACTUAL PROGRAM

How does the above error-propagation complexity numbers compare with the Boeing 787 Dreamliner review statistics? We might make these points:

- The FAA logged approximately 100 person years of effort over the 10 years.
- Recognizing that perhaps 20%–40% of the FAA effort was spent on reviews of the avionics, a figure of approximately 20–40 full-time equivalent person-years spent in avionics review over 10 calendar years was obtained, corresponding reasonably well with the gross general assumptions about the scale of design review (effort and calendar time).
- However, the assumptions made are completely hypothetical and no doubt do not correspond to the Boeing 787 certification review circumstances.

B.13 REFERENCES

B-1. Konrad, M., Sheard, S., Weinstock, C., & Nichols, W. R. (2016). *FAA research project on system complexity effects on aircraft safety: Testing the identified metrics.* Software Engineering Institute, Carnegie Mellon University. Retrieved from http://resources.sei.cmu.edu/asset_files/WhitePaper/2016_019_001_484340.pdf.

B-2. Rayner, K., Schotter, E. R., Masson, M. E. J., Potter, M. C., & Treiman, R. (2016, May). So much to read, so little time: How do we read, and can speed reading help? *Psychological Science in the Public Interest, 17*(1), 4–34.

B-3. Peterson, E. M. (2015). *Application of SAE ARP4754A to Flight Critical Systems*. NASA. Retrieved from http://ntrs.nasa.gov/search.jsp?R=20160001634

APPENDIX C—GLOSSARY

- Assurance Case–An assurance case is a structured argument, supported by a body of evidence, providing a compelling, comprehensible, and valid case that a system exhibits a specified property (e.g., safety) in a given application in a given operating environment.
- Complexity–Complexity is a state or quality of being composed of many intricately interconnected parts in a manner that exceeds the ability of humans, supplemented by tools, to understand, analyze, or predict behavior.
- Error Model–A model of the system that shows the number of errors inside each component that can potentially propagate outward to another component. These values tell the OutPropagateFailureCount(i,j,k) that is in the complexity estimation formula for each mode, component, and propagation point.
- Error Propagation Complexity–Complexity caused by errors propagating from one component to another. This is the kind of complexity the research was focused on to determine whether a system can be certified as safe. This quantity is computed in section 2.2. The computation estimates how many assurance arguments will have to be made by counting the number of ways a defect or fault originating in one component can propagate out to another component.
- Interconnections–Relations indicating, in a given system mode, which propagation points (P-points) of which components of a system are connected to which other components. Interconnections convey failure conditions from the component in which they arise through one or more P-points (to which that failure condition is bound) of that component to other components.
- Mode–An operational phase of a system, during which the system is expected to experience different inputs and behave differently than when in other modes.
- Propagation point (P-point, also called interaction point)–Any interface feature of a component through which a failure condition experienced by that component can emanate out to, and influence the correct functioning of, other components.