Software Engineering Institute
Carnegie Mellon University

Definition and Measurement of Complexity in the Context of Safety Assurance

Sarah Sheard
Michael Konrad
Chuck Weinstock
William R. Nichols

November 2016

TECHNICAL REPORT
CMU/SEI-2016-TR-013

Software Solutions Division

http://www.sei.cmu.edu

This material is based upon work funded and supported by Federal Aviation Administration under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center sponsored by the United States Department of Defense.

This report was prepared for the Federal Aviation Administration.

DM-0004119

Table of Contents

List of Tables

Acknowledgments

The SEI team would like to thank our excellent Business Manager Gregory Such for his continuous support and help during the past two 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.

Executive Summary

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 two-year research project. The tasks in the research project were

- Task 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 assurability of complex systems.

As a part of each task, we delivered a document summarizing our work.

While Task 2 elicited many potential directions for defining complexity, during Tasks 3-5 we selected one definition[1] (in the context of estimating complexity), which we have used since then. Our 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, that provides 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 that the software is safe and the hardware is safe. There can be many levels of subclaims constituting a subclaim hierarchy, going down to the level where 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. Because the many concepts that are needed to describe complexity make it a complex topic, and because measurement, by definition, is a simplification, it may not be possible to measure complexity directly. For our measurement we use a proxy for complexity, describing the length of time (or equivalently, the amount of resources) it will take to achieve a level of confidence in the system's safety. Given the number of sub-arguments in the system's assurance case (each sub-argument relating to one instance of a fault or error in one component propagating to another component), some estimate of reviewer capability, and variables such as a typical length of time to review a sub-argument, it is possible to calculate the effort or cost or equivalent staff

[1] 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 [Nichols 2015].

years to review the entire assurance case. The thinking is that when a system gets more complex (has more pieces, interconnections, potential unanticipated interactions, emergent properties, etc.), it will have many more potential ways that hazards can cause unsafe situations; thus its assurance case will take longer and longer for a certification team to review.

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 component. This number can then be turned into resource or time requirements via the factors mentioned above.

The FAA requested a definitive[2] 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 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.")

We have taken the position that systems in flight today are safe; therefore they fit in the lowest group of systems ("not too complex to assure safety"). More important, we have taken the position that tomorrow's system can only be determined to be safe only if its safety case can be reviewed by a certification team within program schedule and cost constraints. Systems whose certification effort will exceed program budget cannot be assured. One could even say that systems whose certification effort is expected to exceed the budget the program has allocated to the certification effort cannot be assured, although most programs can allocate a few more resources to certification if required to be able to sell and operate the system.

The results of this research are as follows:

1. The algorithm to calculate complexity (of an avionics system)
   - 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[3] for each. These are to be measured for all propagation points, for all components, for all system modes.
   - 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
   - Algorithm:

Error Propagation Complexity (EPC) =

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] While it must be definitive, the statement is allowed to be provisional.

[3] With a hyphen, "fan-out" refers to the concept. With capitals and no space, "FanOut" refers to a variable being calculated.

2. The 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. We took the approach of comparing review time required for a system of such complexity to FAA reviewer availability and capability.

For an FAA staffing level comparable to that for the Boeing 787 (12.5 fulltime people for 8 years, assuming about half—6—deal with avionics), the Error Propagation Complexity of a system that can be reviewed within allowed resources is, in the worst case, about 6400, and in the best case, about 21000. 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.

Abstract

This report describes research to define 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. Since each potential propagation requires another sub-argument in the safety case, the number of arguments should be linear with certification effort. Thus, the ability to show system safety through the certification process depends on this kind of system complexity.

Our results include a formula for calculating the "error-propagation complexity" from system designs and its results for small and medium systems. We tested it 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. Since 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 two-year research project. The tasks in the research project were the following:

- Task 1[4] Project Management
- 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 [Konrad 2015] shows not only many ways that things (or tasks, or situations) can be complex, but also many different kinds of things that can have those kinds of complexity. When something is complex it is assumed that there are factors that drive or cause that complexity as well as consequences that result from it. Some of the causal factors include size (number of sub-pieces), interrelationships among the pieces, structure of the pieces, dynamism, and even socio-political factors [Sheard 2012]. Consequences include difficulty building, difficulty predicting behavior, commission of errors, and even the type and amount of emergent properties.

While we initially thought of measuring [Nichols 2015] the size of the system (e.g., in terms of number of lines of code, or of architectural elements), we could not make a clear link between measurements of those characteristics and safety. After studying how safety is assured [Sheard 2015], we realized the thing that we had to measure 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 regulation).

While Task 2 elicited many potential directions for defining complexity, during Tasks 3-5 we selected one definition[5] (in the context of estimating complexity), which we have used since then. Our definition of complexity is the complexity of the assurance case that must be made to consider the avionics system as safe.

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

[5] 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 [Nichols 2015].

An assurance case is a structured argument, supported by a body of evidence that provides 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 that the software is safe and the hardware is safe. There can be many levels of subclaims constituting a subclaim hierarchy, going down to the level where 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. Because of the number of concepts that are needed to describe complexity makes complexity a complex topic, and because measurement, by definition, is a simplification, it may not be possible to measure complexity directly. For our measurement we use a proxy for complexity, describing the length of time (or equivalently, the amount of resources) it will take to achieve a level of confidence in the system's safety. Given the number of sub-arguments in the system's assurance case (each sub-argument relating to one instance of a fault or error in one component propagating to another component), some estimate of reviewer capability, and variables such as a typical length of time to review a sub-argument, it is possible to calculate the effort or cost or equivalent staff years to review the entire assurance case. The thinking is that when a system gets more complex (has more pieces, interconnections, potential unanticipated interactions, emergent properties, etc.), it will have many more potential ways that hazards can cause unsafe situations; thus its assurance case will take longer and longer for a certification team to review [Nichols 2015].

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 component. This number can then be turned into resource or time requirements via the factors mentioned above.

The FAA requested a definitive[6] 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 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.")

We have taken the position that systems in flight today are safe; therefore, they fit in the lowest group of systems ("not too complex to assure safety"). More important, we have taken the position that tomorrow's system can only be determined to be safe only if its safety case can be reviewed by a certification team within program schedule, cost constraints, availability of expertise and tools, etc. Systems whose certification effort will exceed program budget cannot be assured. One could even say that systems whose certification effort is expected to exceed the budget the program has allocated to the certification effort cannot be assured, although most programs can allocate a few more resources to certification if required to be able to sell and operate the system.

Our complexity formula counts the number of potential error propagations whose assurance arguments must be examined independently in order to assure that the system proposed will behave

[6] While it must be definitive, the statement is allowed to be provisional.

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 figure for total time (minutes), and resources (staff-years) that it will take to assure that a system is safe. We have guessed at what such conversion factors might be in a typical case, but since research like this has never been done before, data to support it does not exist, thus these numbers are speculative. Actual distributions for such factors would need to be determined by additional research.

The results of this research are as follows:

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

- 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, for all components, for all system modes.
- 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
- Algorithm:

Error Propagation Complexity (EPC) =

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. We took the approach of comparing review time required for a system of such complexity to FAA reviewer availability and capability.

For an FAA staffing level comparable to that for the Boeing 787 (actual 12.5 full time people for eight years, and we assume about half—6—deal with avionics), the system Error Propagation Complexity that can be reviewed within allowed resources is, in the worst case, about 6400, and in the best case, about 21000. 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 we selected this approach, how we calculate complexity, and how we calculated the maximum value of complexity that an avionics system can have to be able to be assured for certification purposes.

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. This is our motivation: we want to be able to predict how much effort (review time) a capable certification authority review 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

In our research, we have focused more on the former, short-term, and more narrowly-focused problem; we draw some insights for the latter, longer term and broader problem.

2.1.1 What to Measure: Design Complexity vs. Review Complexity

A measure of design complexity would have some of the attributes we are seeking. Verification of a design can get much more difficult 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 be able 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.

But the *certification authority* review 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 results. They must determine that the evidence really does show what the submission claims that it does, 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 be able to reason about them in context.

Thus design complexity is essential to characterizing the complexity of the task that the certification authority review team performs. Yet standard design complexity measures, such as 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 [Konrad 2015, Nichols 2015], we focused on a measure of the number of ways that failure conditions 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 in order for an understanding to develop of whether in the system design, a particular system hazard is prevented or reduced to be very improbable. We call this quantity Error Propagation Complexity. The units are potential error propagations.

In fact, 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 that identifies the failure conditions for each component of a system design that can propagate outward from that component to influence the behavior of another component. The measure could also be applied to existing systems; 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

As we will see, while 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; while other system hazards, for example those addressing 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. In our research, we came across an elegant redesign for a Stepper Motor-based engine control system to which our measure does assign about 6% less complexity, but in reality the original design requires 100-200% more effort to review. The redesign employs a hypothetical hardware component that eliminates a particularly problematic failure condition, a so-called replication error [SAE 2015, Annex E]. (In the original design, the possibility of this failure created additional derived requirements on intermediary design components, and thus additional argumentation, to eliminate them as a source for a system hazard. We did not revise the formula to address these, as it still accounted for some fraction of the additional complexity in what might be an untypical example, so as to keep the formula simple to understand.)

However, we have found the measure generally conforms to our intuition of argument complexity, is reasonably easy to operationalize, has reasonable inter-rater reliability, is automatable, and serves as a basis for answering the two questions at the start of this subsection.

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 the 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 we will be using this to estimate how long certification will take, we will count the number of separate analyses that must be done. We use a system model (which can be high-level) that shows components, connections, and possible errors (failures and faults). This model is perhaps created in a formal system modeling language such as AADL.

To estimate the number of ways an error can propagate from one component to another, we do the following:

First, if there is more than one mode, the analyses will be repeated for each mode. Thus we sum all of the following over all modes.

Second, each component that is active in that mode may be the source of an error that propagates to another component. Thus we will sum over all components active in the mode.

Third, an error that originates within a component will exit (or escape from) the originating component through a propagation point. We will look at errors propagating through each propagation point separately, and thus we will sum over all propagation points.

Finally, we must estimate propagation events escaping from each propagation point and transiting to other components. Each propagation point has two attributes of interest, called Failure Count and FanOut. The system's error model will tell us how many errors there can be (failure count) associated with a given propagation point, and the system's interconnection model will tell us how many components these errors can transit to through the given propagation point (the fan-out). Each error associated with a propagation point and each component that error can transit to will again be evaluated separately.

Therefore, 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), we will use the following formula to compute the potential size of a safety argument (number of distinct cases that may need to be considered):

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*) · *FanOut*(*i*,*j*,*k*)]

2.2.2 Assumptions

The main assumption is that we have a system architectural design that may be preliminary[7] but identifies system modes, components and their interconnections, and for each component, possible failure conditions that have the capability to propagate outward.

In the next section (Section 2.2.2), we more precisely explain these points, leading to the Error Propagation Complexity formula in Section 2.2.3.

2.2.3 Terminology Used in the Formula

The terminology we use 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; or 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. How to adjust the formula to address the situation is discussed under Generalizations, Limitations, and Discussion (Note 3).
- A **propagation point** (P-point, also called interaction point), is 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.
  - 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, a failure condition can potentially emanate out through none, one, or many of that component's P-points. The failure condition 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 propagation points as well, but when applying the Error Propagation Complexity formula, focus on those propagation points through which failure conditions can propagate out, regardless of whether or not they are also a conduit for the entry of failure conditions.
- **Interconnections** are represented in this work as logical relations[8] indicating in a given system mode which P-points of which components of a system are connected to which other

[7] 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.

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

components. Interconnections are considered to 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.

- Note that interconnections may be only active during certain system modes. In some architectural design languages, interconnections can experience failure conditions [SAE 2015] and/or can be active during mode transitions [Feiler 2012]. How to adjust the formula to address these situations is discussed in Section 3.5 of the Task 6 report (Notes 4 and 5, respectively) [Konrad 2016b].
- Because interconnections are logical relations, we can 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, we can introduce multiple interconnections to indicate the conveyance of failure conditions from that given P-point. For simplicity of expressing the formula, we choose the former interpretation. Therefore, in each mode, we will assume that each P-point 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 (perhaps 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 failure conditions 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 failure conditions 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, which components a failure condition can initially reach can be determined by tracing the failure condition through any P-points it is bound to (and thus 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 failure conditions that can propagate out through P-point P[j,k] when C[j] is active in mode M[i]. (Note 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 a failure condition 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, than FanOut(i,j,k) = 0.)

Then **the potential size of a safety argument** (number of distinct cases that may need to be considered) – this is what we call 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:

- every system mode (M[i]),
- every system component C[j] active in that mode,
- every failure condition that one of those components might experience that the component cannot be assumed to fully handle (and thus may propagate out), and
- every component that escaping failure condition might propagate to.

Then it requires arguing how each component that receives a failure condition addresses it (fully resolves, mitigates, ignores, or transfers it, etc.). Thus the formula is 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, we made the following generalizations, limitations, and observations:

1. The system environment in its entirety should generally be treated as one or more of the system components (one of the C[j]); otherwise, any failure conditions 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 failure conditions can be assumed to be fully handled by the implementers of that component (and thus support compositional verification). This allows us to focus 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) a failure condition, then that interconnection should be re-characterized as a runtime component (one of the C[j]) rather than an interconnection; and the rest of the system design topology (components, P-points, and interconnections) should be updated accordingly. (Appendix B of the Task 5 report recommends a way to handle this situation [Konrad 2016a].)
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 [RTCA 2012] 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., failure conditions 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 Rationale can in fact be more efficiently argued together within a system safety argument, thus 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 (less problematic to build and review) than another system that cannot collapse the review effort so well.
9. Use a failure condition (error) taxonomy (e.g., see the SAE AADL Annex Volume 1 [SAE 2015]) 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 failure conditions that might differ in impact or resolution; and should 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. As we learn more from 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. In the report addressing Task 6 [Konrad 2016b], we further refined the procedure introduced in Section 4 of an earlier report (addressing Task 5) [Konrad 2016a] by introducing some initial preparatory steps to simplify the application of the formula. The procedure has been further simplified by organizing the information needed into a table, and it is presented here. (The full procedure without such simplifications can be found in Sections 3-5 of the Task 6 report [Konrad 2016b].)

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, failure conditions, and fan-out, so that 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 the following (at the time of initialization, it will not have data in the rows). In the column header, we use these abbreviations: *P* = propagation; *FC* = failure conditions, *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.

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 employed 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 (consult Section 2.2.2 above for terminology) for a component of that class.
- Failure conditions: consult the error model for how many failure conditions are declared for the component. If the component contains P-points having different failure conditions or fan-out connectivity, follow the steps below:
  - Split the row into sub-rows as was done for the command components (Cmd1..Cmd2) in the table above, one sub-row per distinct P-point.
  - Within the sub-row associated with a particular P-point, identify the number of failure conditions bound to that P-point. (This is the value of the OutPropagateFailureCondition factor introduced in Section 2.2.3.)

  - 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 can be 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).

Note: Don't 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. We recommend not entering information about these higher level system representations into the table, but focusing instead on the lower-level runtime components that they contain. (An example is given in the report for Task 6 [Konrad 2016b].)

Note: Don't Record Orphaned Inputs

The system design might include some components whose outputs are never used; and whose failure conditions 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 failure conditions that can affect other components, eliminate these components from consideration. (An example is given in the report for Task 6 [Konrad 2016b].)

Note: Record multiple connections from the same propagation point as a fan-out.

Recall from the definition of interconnections in Section 2.2.3 (see last sub-bullet of the definition) that 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 in the table split that P-point into multiple P-points (sub-rows) so that the constraint is still met. Otherwise, regard all these interconnections as constituting the fan-out from the same P-point.

Results Thus Far. 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 failure condition names in the table. If the table is used for the sole purpose of applying the formula, such names as well as 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 describe how to achieve this [Konrad 2016b]. The formula can then be applied directly to either the resulting graph(s) or table(s).

Step 3. 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 (as there are for Cmd1..Cmd2 in our table example), 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. Then, 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 help determine whether the system (1) can be assured for safety, (2) is borderline too complex to assure, or (3) almost certainly too complex to assure. We first give the answer, noting some of the uncertainties in the estimation. Then we develop descriptions of various factors in the estimate, including the rationale for the numbers we used and where we got them. 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 explanations in this section are summarized for easy reading. In truth, the numbers we 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, we have taken the position that a system can only be certified as safe if the amount of time certification is going to take is reasonable compared to the length of the program.

We have defined an "error propagation event" as one error propagating from one component to another. Error Propagation Complexity is then computed as[9] the number of arguments that will need to be made to demonstrate that the system cannot become unsafe due to a possible error propagation event. Thus 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 [Konrad 2016a, Sheard 2015] 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).

We are looking for 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 are available for that one program. (We can easily adjust this Error Propagation Complexity value if safety assurance is only allowed to consume 1% or 10% or some other fraction of the available resources.)

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

2.3.1.1 Error Propagation Complexity Equation Applied to Estimating Assurance Effort

The system's Error Propagation Complexity (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. Thus,

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

We will then compare that effort to an estimate of the total certification effort available to FAA.

Our calculation of effort to resolve a typical error propagation event makes many assumptions, as shown in Sections 2.2, 2.3, and Appendix B. Nevertheless, after deliberation over assumptions, we calculate the amount of effort required as 2.83-7.80 staff years per 1000 events. (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 then, in the best case, 21000, and in the worst case, 6400.

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

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 these two figures gives a value of effort for certification, which should be then compared to the available certification effort to determine whether the system can reasonably be certified as safe. We could also convert the number of review minutes 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, one based on trials the authors have conducted with reviewing safety cases and the other based on rates known for code inspections.

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

Then we construct a hypothetical example system whose complexity and review time we will calculate. We will also convert that review time from minutes spent reviewing to time for a team of four reviewers and then staff-years overall. From these numbers we calculate a conversion factor that produces the staff-years of effort for a given system complexity. We compare these numbers to an actual case (The Boeing 787 Dreamliner) for partial validation.

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

2.3.2.1 Review Rates

We calculate two review rates (units: error propagation events considered per minute), and then calculate their inverse, namely minutes it typically takes to review a safety case, per unit of Error Propagation Complexity of the design.

Table 2 displays complexity and safety case (assurance case) review statistics for the two designs of the Stepper Motor example (see Appendix A, Report 4 description), including the number of nodes,[10] and the review time for two trials.[11]

2.3.2.1.1 SCI (Safety Case Inspection Rate)

Table 3 are multiplied to get the fifth (complexity reviewed per minute), which is then inverted, to arrive at minutes per unit of complexity.

"W*orst 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 2: Input Values for SCI 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

Table 3: SCI Rate Calculation

SCI calculation						
	Pages /min	**sen-tences /page**	**nodes /sentence**	**Cxty /node**	**=Cxty /min**	**Min /Cxty**
assump-tions	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 SCR (Safety Case Review) Rate Calculation

[10] 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 [Feiler 2015].) Nodes generally appear in the assurance case as a single sentence.

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

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 from two trials performed by one of the authors on each safety case from the Stepper Motor Example (in the Task 5 report)[Konrad 2016a]. 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

We must estimate the time needed to review the safety case for a hierarchically designed large complex system, so we will define three categories of system design sizes. "Small" systems would be like the Stepper Motor example of the fourth report [Konrad 2016a]; "medium" systems would be like the Wheel Brake system example of the fifth report [Konrad 2016b], and "large" systems would be most like the SAAB-EII-100 system described in Peterson's report [Peterson 2015].

Since the first two systems had two different designs and thus different values for complexity for each, we use those as minimum and maximum estimates in Table 5. The third system had 203 error propagations, but the design was incomplete, so we guessed at a minimum of 200 and a maximum at half again its size, or a complexity of 300.

To calculate review effort for the best case, we multiplied the minimum system design complexity number by the best-case inspection review rate (time per complexity unit) determined above. To calculate review effort for the worst case, we multiplied the maximum system design complexity number by the worst-case inspection review rate.[12] Table 5 shows the best- and worst-case times to review a small, medium, or large system's safety case.

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

Table 5: Design Class and Best/Worse Times to Review

	Review Times					
	Complexity number		Best case (minutes)		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

We will now characterize a hypothetical system and calculate its overall complexity[13] and its review time. Based on author experience, we have set this hypothetical system as consisting of 100 small-sized subsystems, 30 medium-sized subsystems, and 10 large-sized subsystems, as shown in Table 6.[14] We calculate the total complexity, for later use,[15] as well as the review time per hazard, which we will use now. We are assuming a total of 30 hazards and 2 modes per hazard for this hypothetical system. This brings us a number that is consistent with the hypothetical SAAB-EII-100 example (55 overall-system hazards total) and our other author knowledge.

Table 6: Review Time and Total Complexity for Hypothetical System

					Review Time/system:	
Hypothetical example			Total Cxty		Min	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		

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

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

[15] 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. We used this midpoint for the specific purpose of coming up with a conversion from overall system complexity to review time for our hypothetical system.

For each size of subsystem, the total review time will be the number of hazards[16] times the number of modes (i.e., 60, in our 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 (minimum) and worst case (maximum) total review times for each kind of subsystem are given in Table 7.[17]

Table 7: Review Time for Each Size Set of Components

			Min	Max
Review Time	(minutes)	Small	48600	133313
		Med	27338	79988
		Large	60750	235257

2.3.2.4 Review Time Converted to Duration and Staff-Years

We convert minutes of review to years, assuming that only 180 minutes per day can be dedicated to review (other time is other activities including preparation for the review, certification meetings, and so forth) and that is for each of four people on a review team; three of whom review different material and the fourth oversees. (This review approach is further described in Appendix A). We use 250 work days for a year. 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 3.0 (best case) to 10.0 (worst case) years to review. Again assuming a team of four, this means 12.2-39.9 staff years must be allocated just to certification review.

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

[17] In the general case, we would continue with the table described previously, extending it significantly as follows:

- Add one extra column for each system mode (we mean 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. (Of course, this number can be reduced by additional argument, but we will assume 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 of course 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, we will 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 yr/250 days		years	staff-years
48600	270.0	days	1.1	years	Small		
27338	151.9	days	0.6	years	Med		
60750	337.5	days	1.4	years	Large		
						3.0	12.2
	Worst Case						
X minutes	*1 day/180 min			*1 yr/250 days			
133313	741	days	3.0	years	Small		
79988	444	days	1.8	years	Med		
235257	1307	days	5.2	years	Large		
						10.0	39.9

2.3.2.5 Conversion factor: complexity number to staff-years of effort

Since 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, we can now compute a typical conversion factor from complexity number to time (in years) and resources (in staff years).

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 whose Error Propagation Complexity is 1000 will take 0.59-1.95 years to certify (at four people full time) and 2.4 to 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 represents the typical effort required to resolve 1000 error propagation events.

2.3.2.6 Complexity of a system that is borderline (i.e., 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, we need to make an assumption

about time and effort available for safety assurance. We assume the same as reported for the Boeing 787 example: 200,000 FAA staff hours at 2000 staff hours/year (standard business value) expended over 8 years:[18]

FAA Certification staffing: 100 (=200,000/2000) staff years / 8 years = 12.5 full-time staff.

We also need to assume how many of these people were dedicated to avionics; we guess half or 6.25 full-time staff.

For this calculation we need to see 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*1000 ~6400 (worst case) or 50/2.4*1000 ~21,000 (best case)

Therefore, the complexity of a borderline-certifiable system is between about 6400 (worst case) and 21000 (best case).

The conclusion is that for systems with a certification budget of 50 staff years, systems with complexity numbers under 6400 should be considered assurable; those with complexity numbers between 6400 and 21000 should be considered borderline; and those with complexity numbers above 21000 should be considered "too complex to assure safety."

2.4 Recommendation

We tentatively recommend 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 6400 be considered not too complex to assure safety, between 6400 and 21000 be considered borderline, and whose Error Propagation Complexity is over 21000 be considered too complex to assure safety.

The "tentative" quality of this recommendation relates to the many qualities on which we were unable to find relevant research and for which we therefore had to assume values. We strongly recommend future research to test some of the assumptions that went into the calculations, as listed in this report and in Appendix B.

2.5 Recommended Future Research

The following projects are recommended to continue this research.

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 a failure condition can propagate from one component of the

[18] See Appendix A for a fuller description and source for this example.

system to another. The formula was exercised on two designs each for two small but detailed examples: first, a Stepper Motor controlling engine thrust, and second, a wheel brake system. The formula was found to be easy to apply, did relate to various existing complexity measures, and inter-rater reliability was 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, we propose 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 and us with current data we would propose to use a historical / closed program. We would be looking to answer the following questions:

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/STC 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, we would address what guidance, consistent with current standards, should be given to contractors in the following areas:

1. How do we recognize 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

We would also like to understand more about special cases and assumptions made in the safety case. Below are questions addressing the viability of the complexity formula and to what extent it can be used:

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 our ability to determine a big picture from small pieces. Our research has suggested that the creation of a safety argument is very difficult ("NP-exponential"), but verification 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). The complexity formula to date does not reflect this; we would augment it to do so.

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, thus design complexity and fault-model complexity drive effort. But 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 always 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 question is "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 do we assess the quality and appropriateness of the inputs into the formula? Can we compare safety cases from one applicant to another? How would we normalize the data?

2.5.3 Expand the Fault Model

Currently, our approach to complexity and safety is 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. We therefore propose extending our approach to cover mishaps due to *emergent behavior*, *concurrency*, and *cybersecurity*.

Specifically, we will address faults and failures due to the unexpected emergent behavior resulting from the interaction of properly functioning components. We will 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). We will address cybersecurity-related hazards by looking at the system with Nancy Leveson's Systems Theoretic Process Analysis (STPA). We will extend STPA by interpreting the nodes and arcs of the Systems Theoretic Accident Model and Processes (STAMP) 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. While some of the challenges we have 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 how to keep estimates of complexity for a system updated during various system phases (requirements, specification, implementation, V&V, and sustainment) as well as methods and approaches to overcome the underlying complexity. A prototype tool was also suggested that would 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 one with an equivalent number of piece parts and lines of code that hasn't 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 upon 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 we might consider a complexity number of N to be too complex to review. In the latter we might be able to easily handle it. 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

As mentioned above, 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 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 (parts, connections, and propagation points) and a fault model. We call this kind of complexity "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) in order to resolve whether such propagation could result in a lack of safety.

In this report, the SEI documents how this formula was applied to three different system designs, one each "small," "medium," and "large." For the first two systems, inter-rater reliability was measured and proved 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 6400 but below 21000 should be considered borderline, as they may or may not be certifiable depending on where in the best-case to worst-case spectrum their projects lie. Those below 6400 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.

Appendix A Summary of Previous Reports

This section describes the five previous reports delivered as a part 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.

Report 1. FAA Research Project: System Complexity Effects on Aircraft Safety. Task 3.2: Literature Search to Define Complexity for Avionics Systems (Version 2) [Konrad 2015]

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, we found that while many problems are blamed on "complexity," the term is usually left undefined. As a result, we 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. We have 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
  - For example, existing design and analysis techniques may fail to provide adequate safety coverage of high performance, real-time systems as they 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 diagnosis 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 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 we submitted Version 1 of this report, we rewrote it to address FAA reviewer comments and delivered Version 2.

Report 2. FAA Research Project: System Complexity Effects on Aircraft Safety. Task 3.3: Candidate Complexity Metrics [Nichols 2015]

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 we were considering creating 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 guiding us, eventually, toward the selected metric, which we call "Error Propagation Complexity."

In the report we show how complexity measures for components, component interfaces, and for subsystems, systems, and related interfaces will have to be combined in order 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 we chose the concept of Error Propagation Complexity, and importantly, its estimate in the early system life cycle, to compute the value a safety-related measure of complexity.

Report 3. FAA Research Project: System Complexity Effects on Aircraft Safety. Task 3.4: Identify the Impact of Complexity on Safety (Draft report, revised) [Sheard 2015]

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 vs 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.

Report 4. FAA Research Project: System Complexity Effects on Aircraft Safety. Task 3.5: Estimating Complexity of a Safety Argument [Konrad 2016a]

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 came later.

As was evident from Report 1, there are many and very varied definitions of complexity; our challenge was to find or create a measure that related complexity to safety. We 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 thus a reasonable rationale for drawing a line and saying “Systems more complex than this line will be too complex to assure safety,” as was requested by our contract.

Of course, by the time the size of the safety case is accurately known, the program will be complete. We needed a means to estimate this size, much earlier. We eventually realized 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 created a way to count these propagations. The sum would be multiplied by a notional, average, or calibrated parameter estimating for each of these many cases, how long each one 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.

Report 5. FAA Research Project: System Complexity Effects on Aircraft Safety. Task 3.6: Test the Identified Metrics [Konrad 2016b]

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.

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 (though automated approaches are recommended).

Appendix B Mathematical Argument Behind Section 2.3

This appendix calculates the acceptable complexity numbers for a system if it is to be considered likely to be able to approve and borderline vs. too complex to be able to assure safety. We first give the answer, noting some of the uncertainties in the calculations. Then we develop descriptions of various factors in the calculations, including the numbers we used and where we got them. 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: while 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 we 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 we agreed to use in the main text in parentheses.)

A formula-driven take on the problem

We explore 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:

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 CAR Team to review these to determine that an entire set of system hazards are very improbable?

An important caveat

All of our answers are based on estimates derived from very limited data. Our real goal is to identify the particular set of factors, which if known, would help establish more precise answers to these questions; and not on the answers themselves. While we do derive ranges and thresholds from very limited experimental data in this report, it must be emphasized that there is such high uncertainty around many of these factors, especially in our 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. Also, additional research could be conducted to provide better guidance in the general case (see Section 2.5).

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?

By properly-elaborated safety case we mean 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 why all possible conditions that might result in the system hazard cannot occur, or are very unlikely. The argument that for relevant failure conditions (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 but not included in our 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 below 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 [Konrad 2016b].)

Table 9: 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 about 0.4...0.6 minutes review time per unit of complexity. We call this the Safety Case Review Rate (or SCR rate).

How does this SCR rate compare with ordinary reading rate and design inspection rates?

- **Ordinary reading rate.** The average reading rate is about 200..400 words per minute [Rayner 2016]. Assuming 200 words per minute because the material is relatively difficult to understand in places, and referring to the #Words column, we'd expect about 5.8 minutes to read the safety case for Design A and 2.3 minutes for Design B. Then dividing by their respective Error Propagation Complexity, results in a range of 0.15..0.34 minutes reading time per unit of complexity. We call this the Safety Case Ordinary Reading Rate (or SCOR rate).
- **Inspection rate**. According to one of the co-authors of this report (Bill Nichols), detailed reviews of software design documentation is about four pages per hour. Assuming about 40 sentences per page, that works to about 160 sentences per hour. If we equate safety case

nodes with sentences, then referring to the #Nodes column, we'd expect about 28 minutes to inspect the safety case for Design A and 15 minutes to inspect the safety case for Design B. Then dividing by their respective Error Propagation Complexity, results in a range of 0.9..1.6 minutes inspection time per unit of complexity. We call this the Safety Case Inspection Rate (or SCI rate).

Thus, based on limited data, the SCR rate is about half the SCOR rate but about three times the SCI rate. Why might the SCR rate be so 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, as 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. On the other hand, 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.

At any rate, we therefore have three estimates of the time it takes to review a safety case based on three different approaches to estimating the rate:

Table 10: Time Required for Safety Case Review Using Three Approaches

Review rate per unit of error-propagation complexity:	Min	Max	Mean
Safety Case Review Rate (SCR rate)	0.4	0.6	0.5
Safety Case Ordinary Reading Rate (SCOR rate)	0.15	0.34	0.24
Safety Case Inspection Rate (SCI rate)	0.9	1.6	1.2

We will refer to two of these three rates, namely the empirically-derived review rate (SCR) and experientially-observed inspection rate (SCI) in the following.

Single-system design and one system hazard

The question, again, is this:

2) 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?

By a single-diagram system design, we mean 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. Thus, for reasons of legibility, we're speaking of at most 25-30 components in the diagram; anything more than that might be best represented hierarchically and not as a single diagram. (We don't make any assumptions about the error model size or verification results when we say "single diagram.")

Thus far, the team has studied designs in three size ranges (with some variation around sizes and complexity given the different purposes for the examples):

Table 11: 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 [Peterson 2015])	28	~ 200 (flight mode only)

Calculating review time for each "unit of Error Propagation Complexity"

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

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

For worst-case, we assume the maximum experientially-observed safety case inspection rate of 1.6 minutes per complexity unit (C-unit). This is worst-case also because this rate corresponds to that achieved when reviewing software designs with the purpose of detecting defects. Summarizing:

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

We can then immediately derive these review times for each design class (multiplying the worst-case-to-best-case range identified immediately above by the estimated mean number of complexity units for that design class), and rounding:

Table 12: 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 [Peterson 2015])	80..320 minutes

Remember 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 (the error model for most components in the design indicated only 1-2 failure conditions).
- The large-class review time is based on an incomplete design with minimal information about almost all components [Peterson 2015].

Avionics System Design and Multiple System Hazards

Again here is 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 so many factors that what is expressed quantitatively must be treated as very conjectural.

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 (how many large, medium, small components)
    - precedentedness (off-the-shelf and proven vs. 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 review same designs for selected critical-defect types
    - when results of earlier reviews can be reused vs. 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 problem 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, we proceed with a hypothetical example.

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 employ 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 [Peterson 2015] assumes about 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 (Flight)
  - The other half require all three modes be considered (Climb, Mid-Flight, Descent)
- Further, we assume the following review rate parameters for the CAR Team as a whole:
  - One review team member seeds the material to be reviewed (in order to estimate and monitor by capture-recapture the quality of the reviews).
  - 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).

We now subject the above hypothetical example to a best-case analysis to identify a lower threshold; and a worst-case analysis to identify an upper threshold. But first, we determine the team's daily review rate:

Sustainable review rate

Sustainable review rate for a team of four full-time reviewers employing 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 % of the defects is three, thus the 180 minutes of review time/day is *for each of the people on the team—not just a single reviewer*.

Table 13: 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 [Peterson 2015])	(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 combi-nations = 8400 reviews	440K review min = x years, x hours	111K review min

Dividing the number of review minutes needed by the sustainable daily review rate for the team, we obtain an estimate of the number of years such a review would encompass in calendar time:

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

Of course, in actuality, many of the assumptions made (e.g., that there be no changes) are unrealistic. Also, 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.

On the other hand, 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

Thus, we can estimate (under the previous assumptions) that a design of about 20K C-units would be too much to review, allowing us to set an upper threshold of around 20K C-units.

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

Not Too Complex, Borderline, and Too Complex

Thus, **under the given hypothetical assumptions** not directly based on any actual example, we might conclude the following 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

Validation: Comparison to actual program

How does the above compare with the Boeing 787 Dreamliner review statistics? We might make these points:

- The FAA logged about 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, we obtain a figure of around 20-40 full-time equivalent person years spent in avionics review over 10 calendar years, which corresponds reasonably well with our 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.

Appendix C Glossary

Assurance Case. An assurance case is a structured argument, supported by a body of evidence, that provides 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. For each mode, for each component, and for each propagation point, this values tells the *OutPropagateFailureCount*(*i*,*j*,*k*) that is in the complexity estimation formula.

Error Propagation Complexity. Complexity caused by errors propagating from one component to another. This is the kind of complexity we focus 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 are relations indicating in a given system mode which 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 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.

Propagation point (P-point, also called *interaction point*), is 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.

Appendix D Acronyms

Acronyms	Definition
CAR	Certification Authority Review [team]
FC	Failure conditions
FO	Fan-out (concept) or FanOut (variable)
NP	Nondeterministic polynomial [time to solve]: type of mathematical problem
P-points	Propagation points
SCI	Safety Case Inspection [rate]
SCOR	Safety Case Ordinary Reading [rate]
SCR	Safety Case Review [rate]
STAMP	Systems Theoretic Accident Model and Processes
STC	Supplemental type certificate
STPA	Systems Theoretic Process Analysis
V&V	Validation and Verification

References

URLs are valid as of the publication date of this document.

[Boeing 2014]
Kaszycki, Michael; Rich Ptacin; Christopher B. Bergen; et al. *Boeing 787-8 Design, Certification, and Manufacturing Systems Reviews*. Boeing, March 19, 2014. http://www.faa.gov/about/plans_reports/media/787_report_final.pdf

[Feiler 2012]
Feiler, P. & Gluch, D., *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, 2012 (ISBN-10: 0-321-88894-4).

[Konrad 2015]
Konrad, M. & Sheard, S. *FAA Research Project: System Complexity Effects on Aircraft Safety: Literature Review Task 3.2: Literature Search to Define Complexity for Avionics Systems.* Software Engineering Institute, Carnegie Mellon University. 2015.
[Konrad 2016a] Konrad, Michael; Sheard, Sarah; Weinstock, Chuck; & Nichols, William R. *FAA Research Project: System Complexity Effects on Aircraft Safety. Task 3.5: Estimating Complexity of a Safety Argument*. Software Engineering Institute, Carnegie Mellon University. 2016.

[Konrad 2016b]
Konrad, Michael; Sheard, Sarah; Weinstock, Chuck; & Nichols, William R. FAA Research Project: System Complexity Effects on Aircraft Safety. Task 3.6: Test the Identified Metrics. Software Engineering Institute, Carnegie Mellon University. 2016.

[Leveson 2013]
Leveson, Nancy. 2013. *An STPA Primer, Version 1*, August 2013. http://sunnyday.mit.edu/STPA-Primer-v0.pdf

[Nichols 2015]
Nichols, William R. & Sheard, Sarah. *FAA Research Project: System Complexity Effects on Aircraft Safety. Task 3.3: Candidate Complexity Metrics.*Software Engineering Institute, Carnegie Mellon University. 2015.

[Peterson 2015]
Peterson, Eric M. “Application of SAE ARP4754A to Flight Critical Systems.” NASA, 2015. http://ntrs.nasa.gov/search.jsp?R=20160001634

[Rayner 2016]
Rayner, Keith; Schotter, Elizabeth R.; Masson, Michael E. J.; Potter, Mary C.; & Treiman, Rebecca. “So Much to Read, So Little Time: How Do We Read, and Can Speed Reading Help?” *Psychological Science in the Public Interest.* Volume 17. Number 1. May 2016. Pages 4-34.

[RTCA 2012]
RTCA, Inc. *DO-178C Software Considerations in Airborne Systems and Equipment Certification*, January 2012. http://www.rtca.org/store_product.asp?prodid=803

[SAE 2015]
SAE International. 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.

[Sheard 2012]
Sheard, Sarah. *Assessing the Impact of Complexity Attributes on System Development Project Outcomes*, PhD diss., Stevens Institute of Technology. 2012.

[Sheard 2015]
Sheard, Sarah; Weinstock, Chuck; Konrad, Michael; & Firesmith, Donald. *FAA Research Project: System Complexity Effects on Aircraft Safety. Task 3.4: Identify the Impact of Complexity on Safety*. Software Engineering Institute, Carnegie Mellon University. 2015.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	November 2016	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Definition and Measurement of Complexity in the Context of Safety Assurance	FA8721-05-C-0003

6. AUTHOR(S)

Sarah Sheard, Michael Konrad, Chuck Weinstock, William R. Nichols

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2016-TR-013Error! No text of specified style in document.

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFLCMC/PZE/Hanscom Enterprise Acquisition Division 20 Schilling Circle Building 1305 Hanscom AFB, MA 01731-2116	n/aError! No text of specified style in document.

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

This report describes research to define 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 sys-tem error (fault) could propagate from element to element. Since each potential propagation requires another sub-argument in the safety case, the number of arguments should be linear with certification effort. Thus, the ability to show system safety through the certification process de-pends on this kind of system complexity.

Our results include a formula for calculating the "error-propagation complexity" from system designs and its results for small and medium systems. We tested it 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. Since 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.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Safety, mission assurance, avionics	53

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL