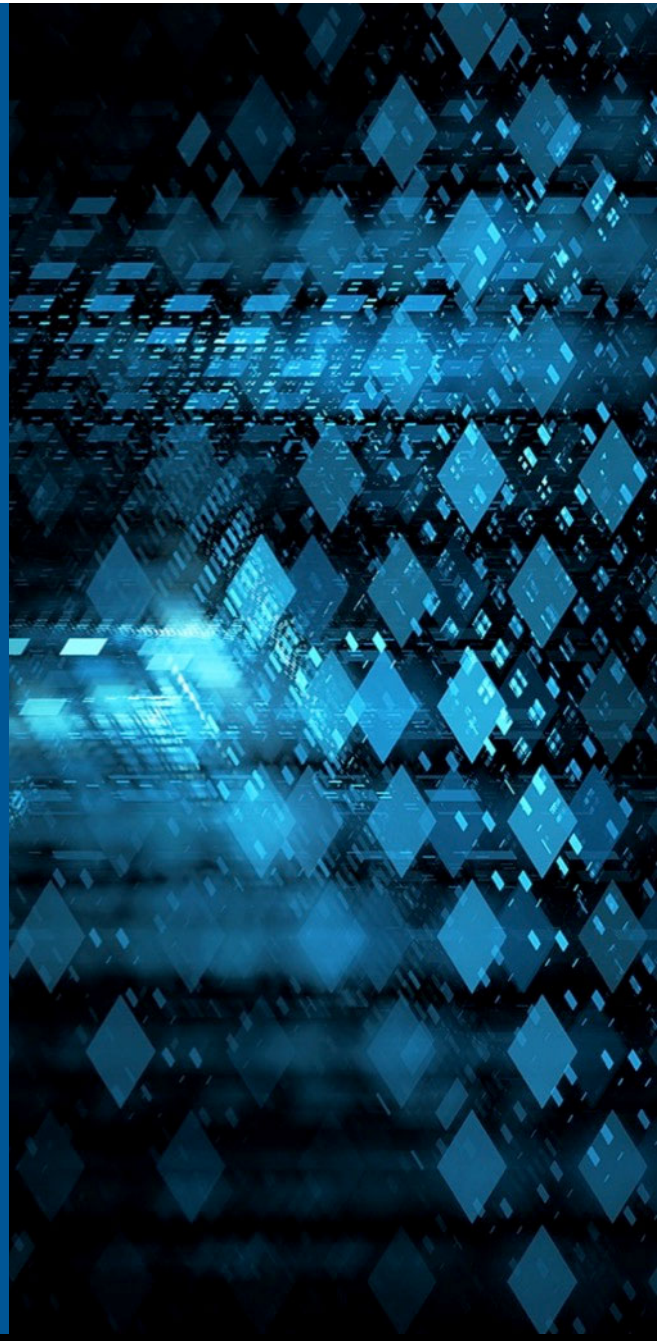
Measuring Complexity for System Safety Assurance

Sarah Sheard

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213



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.

Context: Complexity and V&V

John Frederick, at the 2016 V&V summit, said challenges to V&V-ing NextGen systems include:

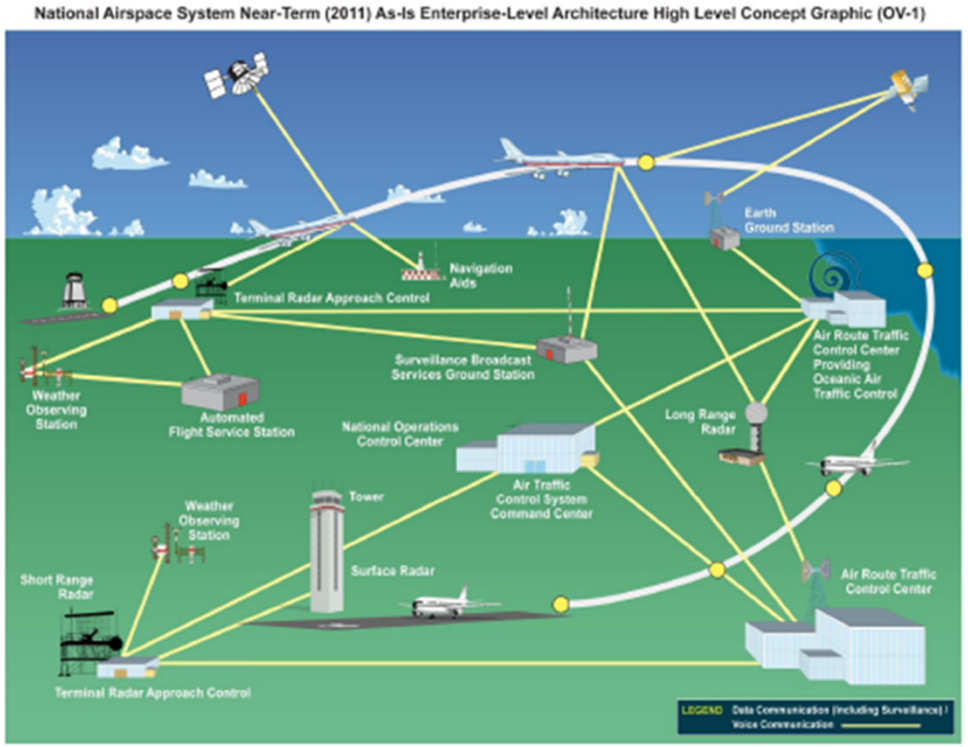
- SoS architecture rather than system ("subsystem point-to-point") architecture
- Multiple organizations and multiple systems need to cooperate
- Operational effectiveness and suitability refers to entire enterprise
- Consistent test practices, standardization and consistency across domains is more important than ever

How best can the FAA address these problems?

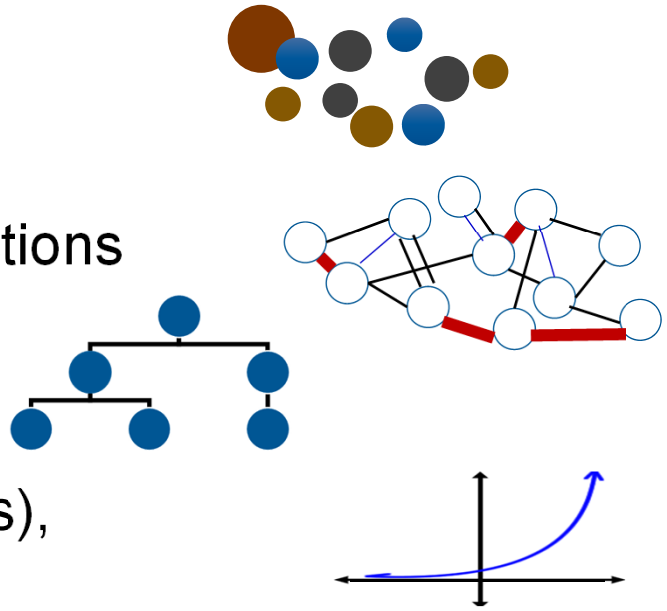
How complex is this situation? (2012)

There are 200 aircraft in the sky above you within an hour, and each:

- Has own origin, destination, speed, altitude, type
- Is full of human beings
- Burns 50 gallons of fuel per minute: cost, environmental, economic, & political impact
- Wants optimum, shortest time and lowest cost path
- Must be absolutely safe



How to assess complexity?



Numbers of things and of “different” things

Numbers and kinds of relationships/connections

Structure of systems and structure of organizations managing the scenarios

Dynamic complexity (complexity of changes), immediate through long-term

Other complexities:

- Organizational, legal, societal, business
- Who should be blamed, and who should pay, when there are problems with a program that is sold from one company to another?

How to manage complexity?

Technological: Augment human intelligence with tools

- Warnings: wind shear, pull up, terrain closure, traffic
- Checklists
- Flight strips
- Airspace simulations
- Decision support systems

But:

- Must trust tools (bit errors, power outage, safety…)
- Tools add to complexity
- Tools must reduce human's complexity
- Quest for ease-of-use → higher complexity adaptable tools
- "Pilot has gradually become barricaded from the reality of the flight task by an array of automated systems and is often swamped by the complexity of surrounding systems" – Noyes et al., 2000

Non-technological:

- Connect with others; understand other viewpoints

Research project 2015-2016

How complex can things get before we can no longer assure ourselves that they are safe?

- How should “Complexity” be defined? (literature search)

- How to measure complexity? (Many measures; must use one directly related to property you’re looking to assess)
- Algorithm
  - Looks at complexity *of safety case* (Case made for assuring a property, in this case safety)
  - Counts number of ways an error could propagate from one component to another, since every propagation must be followed up by reviewers
  - Estimates review resources required

Reports

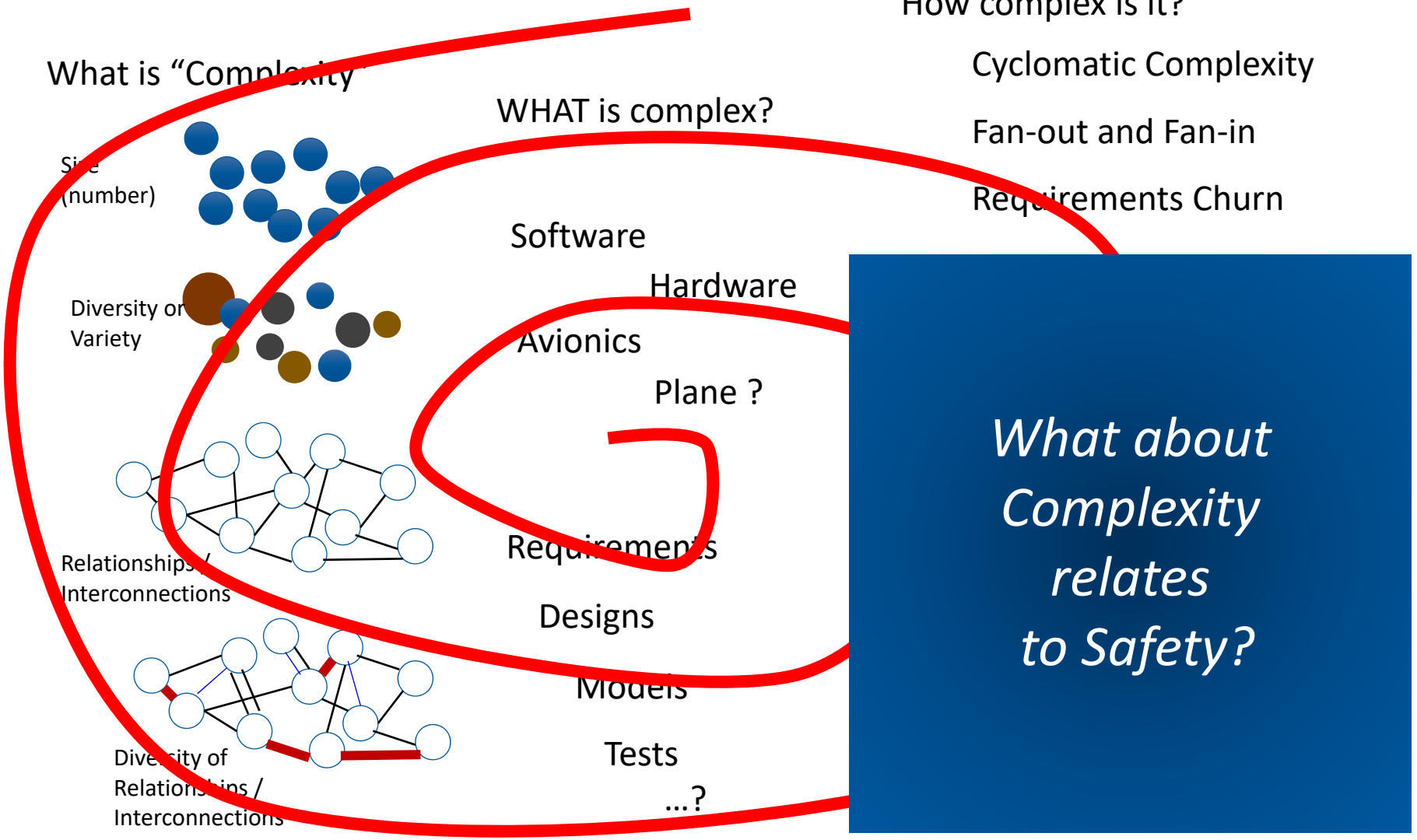
http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=483758

Definition and Measurement of Complexity in the Context of Safety Assurance *(*overall report, pdf)

5 previous and more detailed reports (pdf):

- Complexity Definition Literature Review
- Candidate Complexity Metrics
- Impact of Complexity on Safety
- Estimating Complexity of Safety Argument
- Testing the Identified Metrics

Complexity is complex

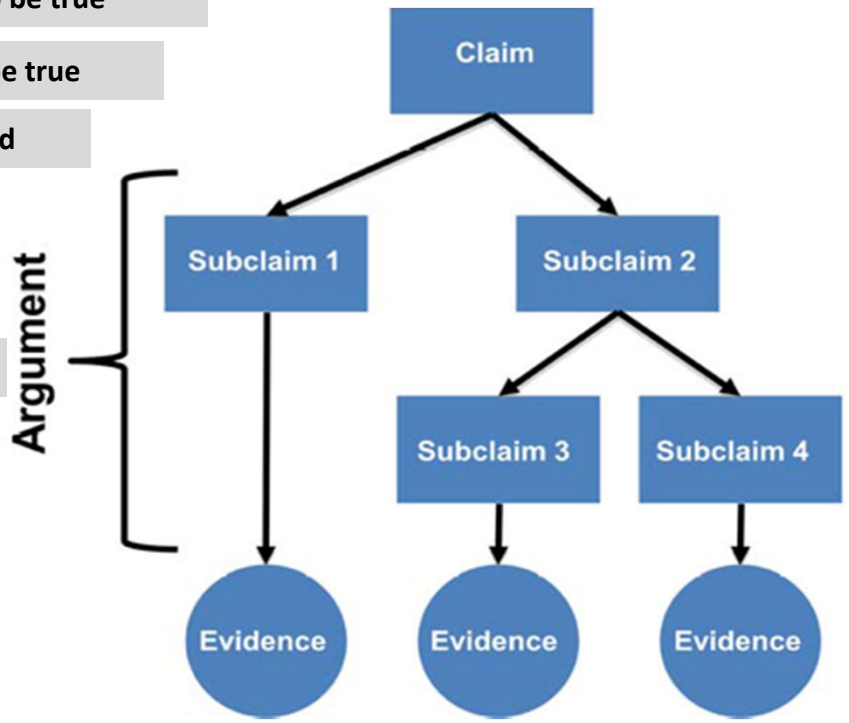


Safety case (type of assurance case)

For "The System Is Safe" to be true

Subclaim 1 and 2 must be true

Argument must be sound



For Subclaim 1 to be true

There must be X evidence

For Subclaim 2 to be true

Subclaim 3 and 4 must be true

For Subclaim 3 and 4 to be true

There must be Y evidence

There must be Z evidence

Two main contributions

1. Evaluate the complexity ***of the safety case***

   But: the safety case isn't "complete" until the aircraft is designed, built, tested, with all software on board…

2. **Estimate** the size of the safety case **early**

   How much work (analysis, documentation, meetings etc.)

   will it take to assure that the system is safe?

   (# potentially cascading error conditions)

   - ☞ Assume component assurance process will remain as is
   - ☞ Big unknown is result when errors cascade from one component to another
   - ☞ Order of magnitude probably ok

Dealing with complexity in general

Understand how complex it is…how many, how interconnected, how varied. System, Project, and Environment, maybe Process.

Where are the biggest centers of confusion? They are risks.

Identify stakeholders: Who has a stake in this? Who can help, who can hurt?

Work with stakeholders to understand:

Interface points: What needs to be communicated & when

Priorities: What do they care most about, & why?

Use System Thinking:

- Understand (parts, functions, relationships, risks, inputs, outputs, …)
- Interconnect (Of people at least as well as systems)
- Standardize (Reduce variety)

Recommendations to address V&V across interfaces of complex systems of systems$_1$

Assess *complexity of "components"* of V&V problem – to focus on biggest problems, and the nature of the complexity

- Ensure these get extra attention during development and during test planning
- Problem is still harder if components are stand-alone systems (SoS)

Assess *complexity of* systems of systems *interfaces*: number, variety, where things could go wrong, dynamics…

- Safety hazards and security threats, e.g.

Understand risks: create safety cases and get broad review

Make parts as reliable as possible: Do audits, reviews, inspections, during development

Insist on static analysis of design documentation, algorithms, code

Also non-static V&V (useful for testing combined components and system)

- Modeling, Simulation, HW/SW benchmark testing, HWIL,HIL

Recommendations to address V&V across interfaces of complex systems of systems$_2$

Track what you do and do not know at each level of test (box, subsystem, system…)

- May know: Inside the component (assuming precedented and tested)
- Do not know: Interfaces to other components
- May know somewhat: Interfaces to the environment

Understand what has already been tested about the component, and what needs to continue being tested

Practically speaking:

- Mission thread workshops focused on identifying software risks in general and also V&V-related risks

Contact Information

Sarah A. Sheard, Ph.D.
Principal Engineer
Office: (412) 268-7612
sheard@sei.cmu.edu

Software Engineering Institute
Carnegie Mellon University

Backup Slides

#1 Recommended future research: precedence

- Study complexity “discounts” that we should give to known or precedented system components because they are familiar
  - How many error propagations (from model) have already been proven not to be unsafe and thus need less review?
  - How can this be applied to, say, *slightly* different configurations? How do you measure “slightly”?
  - How can this be applied to slightly different hazards?
  - What is safety effect of higher-capability component compared to existing?
- Other areas can contribute:
  - How organizations today currently allow credit for testing already done
    - FAA and aircraft re-certification (e.g. longer fuselage)
    - FDA and medical devices
    - Regression testing
  - Estimate of the amount of impact caused by a change (hardware, then software)
  - Understanding how much of the problem could be solved by nearly-independent, modularized, proven-correct components

Recommended future research

1) Apply to larger system and validate at real-life scale.
2) Study special cases, assumptions, and limitations more specifically
   a) Including what about precedented system components: should these count as less complex because we are familiar with them? How?
   b) Including tweak numbers for whether the Applicant has provided an organized safety case or not. How does this affect FAA effort?
   c) Determine effect of having models to different levels of detail. Is there a notional “complexity reduction” curve?
3) Expand fault model to include more than error propagation: emergent behavior, concurrency, and cybersecurity
4) Develop guidelines for safety assurance practices and design guidelines to reduce software complexity

Our method

Primary Assumption:

Early design work on new system* has resulted in a model of the system architecture at a high level including

- system modes
- active components and their interconnections in each mode
- possible failure conditions that could propagate outward

Many additional assumptions made to arrive at notional thresholds for between systems that are assurable as safe and systems that are too complex to assure as safe

*For future research: precedented systems

Assume

Multiple modes; errors can propagate in each

- Sum over all modes

Multiple components; errors can propagate from each one

- Sum over all components active in that mode

Multiple propagation points on components

- Sum over all (outward-) propagation points

Then,

For each propagation point, each component, each mode:

- Multiply number of failures that could propagate out by number of places the failures could reach (Fanout)

Algorithm

Sum over all system modes:

Sum over all components active in a given mode:

Sum over all propagation points (p-points) for this component:

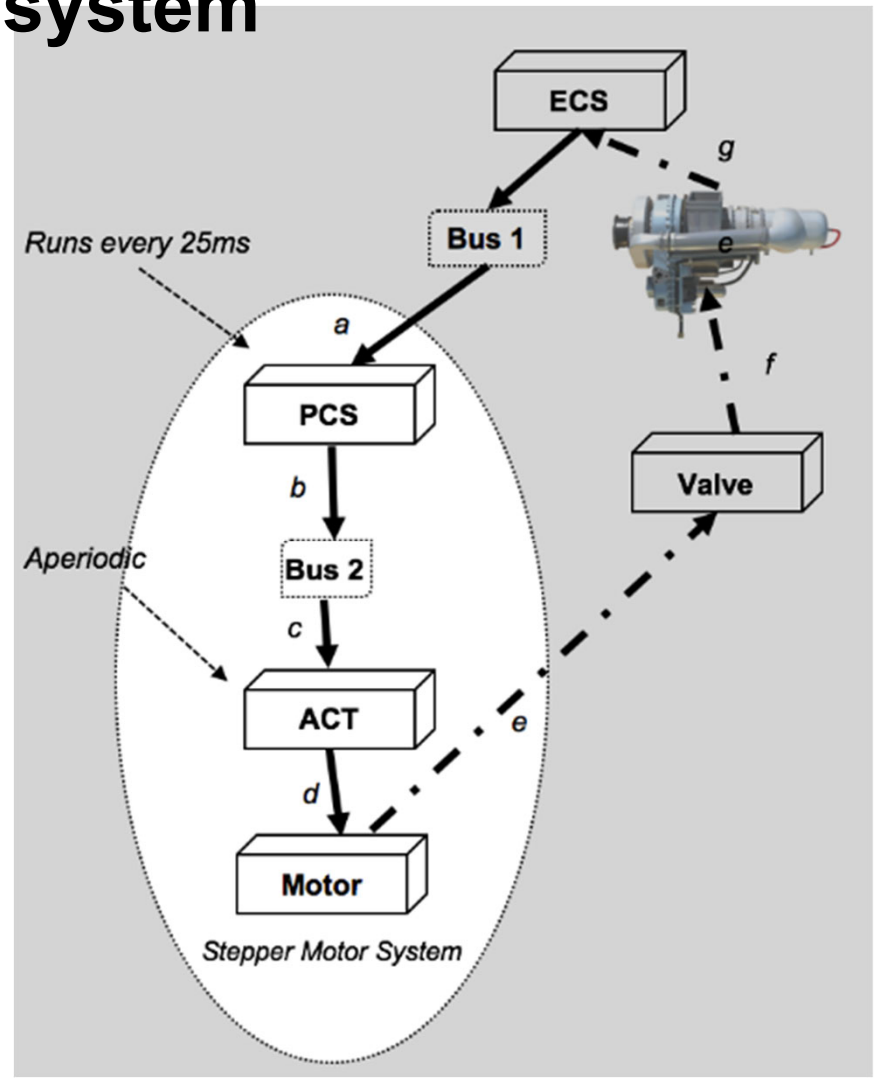
of:

{ Number of failures that could propagate out from this p-point } times { Fanout from this p-point }

Example 1: stepper motor system

1. From High Level design:
- 1 mode
- Interfaces shown
- Treat Bus 2 as a component*
- 4 components plus Environment
- #P-points = 1 for all components
- Fanout always = 1
- - - - - - - - - - - - -

2. From Error Model:
- Errors from Environment to SMS: 3
- Errors from PCS to Bus 2: 4
- Errors from Bus 2 to ACT: 3
- Errors from ACT to motor: 3
- Errors from Motor to Envt.:3

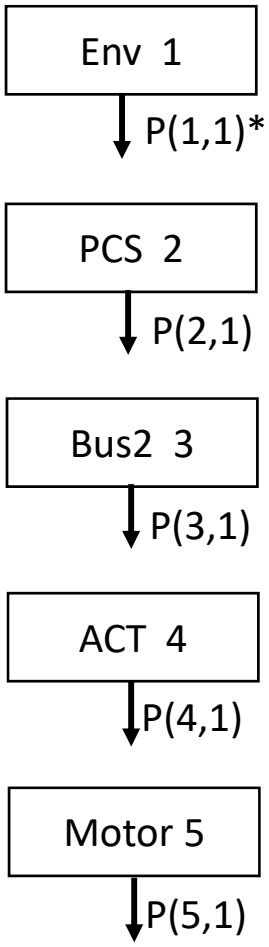


Ref: Konrad 2015b of Final Report

*Since it can be a source of a failure condition

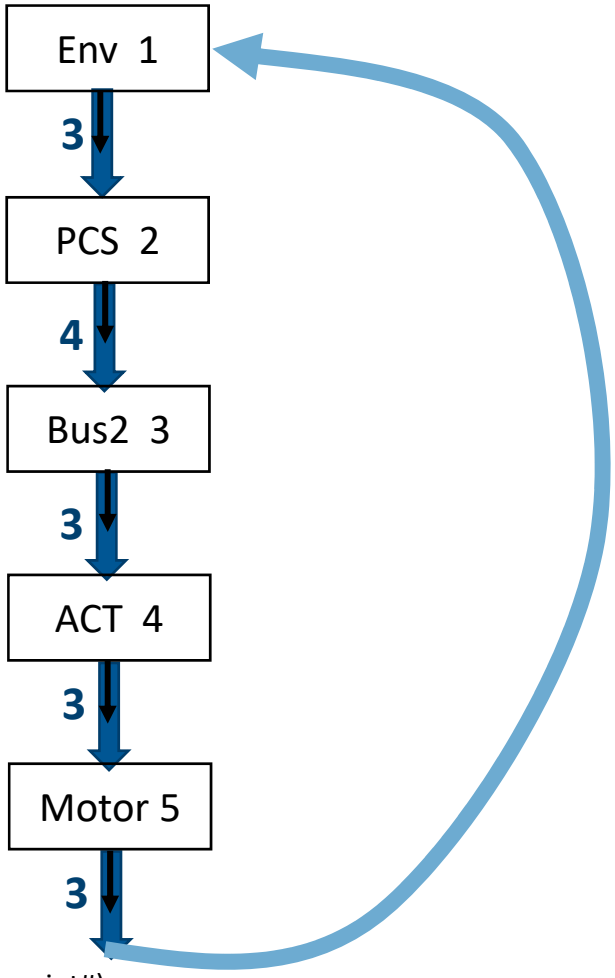
Calculating EPC (for one mode)

First step



*Notation P(component#, p-point#)

Second step



Third step

Sum of (#failures*Fanout for all P-points of Component x)

x	Sum
1	3*1 = 3
2	4*1 = 4
3	3*1 = 3
4	3*1 = 3
5	3*1 = 3

Total all components

Error Propagation Complexity = 16

Air traffic controller complexities

- Must track all these aircraft
- Know where they came from (final and previous sector) and where they are going (next sector and final)
- Account for takeoffs and landings adding and removing aircraft, with rapidly changing altitudes
- Account for problematic weather
  - Cumulonimbus clouds (turbulence, icing, lightning, hail, vertical and horizontal winds)
  - Squall lines (tornados, gust fronts, microbursts)
  - Constantly changing in intensity and moving
- Avoidance scenarios:
  - Non-standard traffic flows (vertical, horizontal): weather intensity changes, flight crew deviate from route, e.g.; thus new conflict points; inability to maintain headings in changing winds
  - Reduction in available airspace
  - Longer contact of pilots with ATC, increased phone coordination