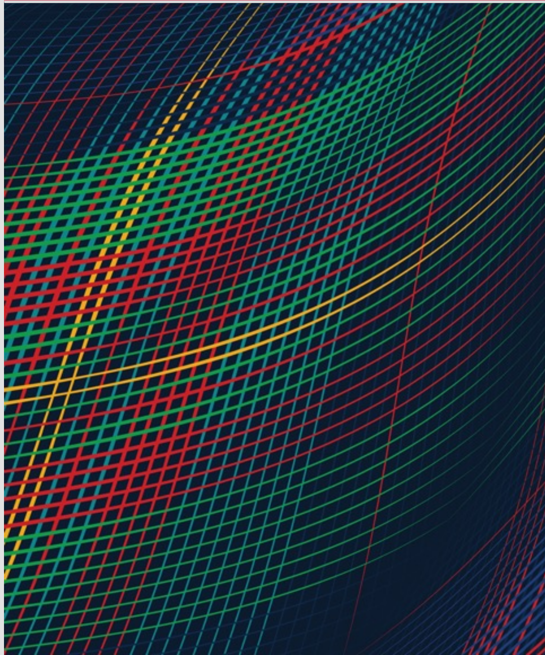
Carnegie Mellon University
Software Engineering Institute



Is Design Diversity Essential / Effective / Practical for Critical Systems?

MAY 12, 2023

Sam Procter

Document Markings

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

DM23-0472

Agenda

- Preliminaries
- Design Diversity for:
  - **Software Security**
  - **Software Dependability**
  - **System Dependability**
- Wrap-up

Preliminaries – Goals

- Design diversity is an area I’m interested in and have exposure to
- Want to avoid this being a “book report” (especially since the book’s authors are in the room!)

- I’ll discuss some connections to my area of expertise
- Hopefully this sets up the good discussions SCC is known for

Preliminaries – Definitions

Essential? Effective? Practical?

$$\text{Essential} = \text{Superior to alternatives}$$

$$\text{Practicality} = \frac{\text{Efficacy}}{\text{Cost}}$$

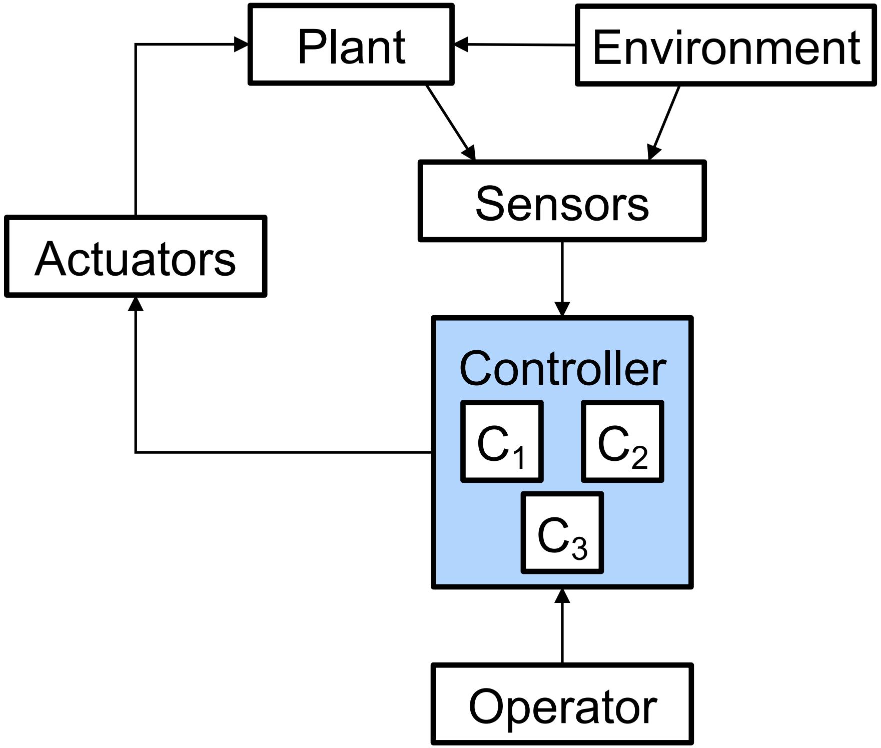
- Terms are somewhat interrelated
- Primarily a software talk: I’m a software guy, I work at a software institute, this is a software workshop

Preliminaries – Definitions

Critical Systems Traditions, Notional Cyber-Physical System

- Different "traditions" in critical systems, also conceivable as system design goals
  - Dependability
    - "Home" of redundancy / design diversity
  - System Safety
  - Security
  - … and Real Time
    - Also mentioned by Rushby, not really addressed in this talk

"Critical System Properties: Survey and Taxonomy." John Rushby. Reliability Engineering and System Safety, 1994.



"Fault-Tolerant Systems (Second Edition)." Israel Koren, C. Mani Krishna. Morgan Kaufman, 2021.

Is Design Diversity Essential / Effective / Practical for…

	System Safety	Security	Dependability	
Software	Only as a follow-on effect of dependability	Automated diversification	Effective, but hard to measure, not universally practical	Effective! (Subject to Caveats)
Systems		???	Essential, but easy to accidentally undo	Essential!

Design Diversity for Software Security

Design Diversity for: Software Security

- Variety of techniques for reducing software monoculture
- Recent technological advances have enabled automated program diversification:
    - Increased computing power (i.e., the cloud)
    - Online software delivery
- Modification types and timings:
    - Pre-distribution: Source code (via compiler, linker)
    - Post-distribution: Native / binary code (via installer, loader, executor, updater)
- Relevance for critical systems
    - Difference in goals: Fault tolerance vs attack resistance
    - Difference in motivation: Monoculture is not an issue (or less of one?)
    - High degree of automation / largely transparent to users

"SoK: Automated Software Diversity." Per Larsen, Andrei Homescu, Stefan Brunthaler, Michael Franz. IEEE Symposium on Security and Privacy, 2014.

Is Design Diversity Essential / Effective / Practical for Critical Systems?

Design Diversity for Software Dependability

Design Diversity for: Software Dependability

Challenges and Solutions

- Can be effective / situationally practical, but challenges include…
- *Consistent comparison problem*: $.9999 \cong 1 \cong 1.0001$
- *Non-independence between versions*:
  - Challenges:
    - Common specifications
    - Intrinsic difficulty of the problem
    - Common algorithms
    - Cultural factors
    - Common software and hardware platforms
  - (Potential) solutions are diverse…
    - Specifications
    - Programming languages, development tools, and compilers
    - (Cognitively diverse) teams

"Fault-Tolerant Systems (Second Edition)." Israel Koren, C. Mani Krishna. Morgan Kaufman, 2021.

"An Experimental Evaluation of Software Redundancy As a Strategy for Improving Reliability." Dave E. Eckhardt, Jr. Alper K. Caglayan, John C. Knight, Larry D. Lee, David F. McAllister, Mladen A. Vouk, John P. J. Kelly. NASA Technical Memorandum 102613, 1990.

"An Experimental Evaluation of the Assumption of Independence in Multiversion Programming." John C. Knight, Nancy G. Leveson. IEEE Transactions on Software Engineering, 1986.

Design Diversity for: Software Dependability

“The picture that emerged from this evidence—industrial experience, experiments, and theoretical modeling—has sometimes been taken to undermine claims for the efficacy of software diversity. While the industrial evidence is positive, it inevitably emerges only after years of operating experience, and even then does not easily allow quantitative claims for achieved reliability. The experimental and modeling evidence, on the other hand, has been taken to be negative. In fact, this negative view seems somewhat unfair.”

“Reasoning about the Reliability of Diverse Two-Channel Systems in which One Channel is ‘Possibly Perfect’.” Bev Littlewood, John Rushby. IEEE Transactions on Software Engineering, 2011.

Design Diversity for: Software Dependability

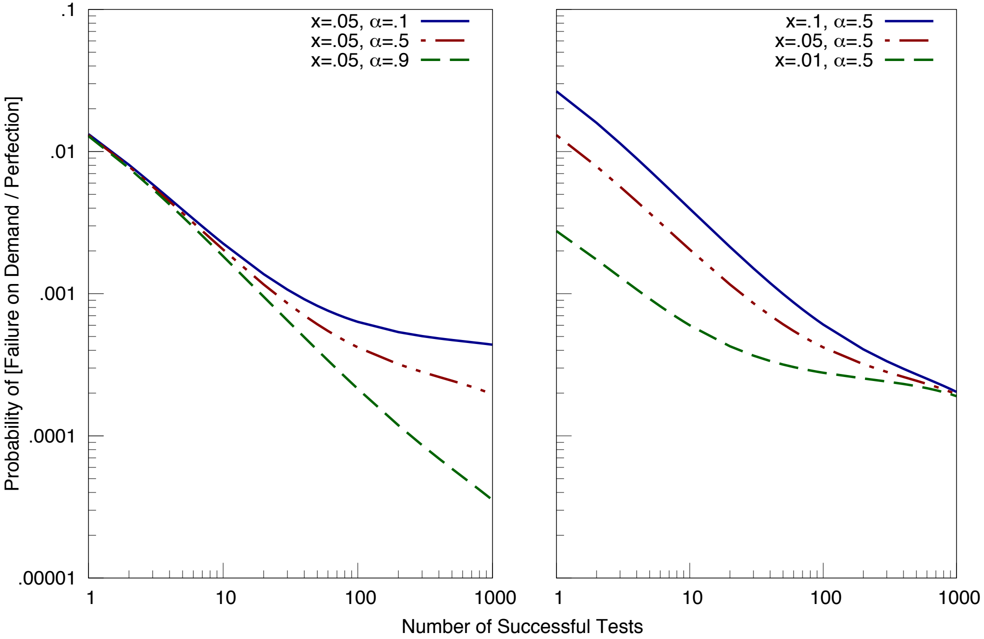
Q1: How effective is a particular set of implementations?

- Reliability models for software are not as mature as those for hardware
- Calculations for an error rate are based on number of bugs and input rate
- Alternative calculations include subjective assessments of:
  - Independence
  - “Perfect” implementation
- Refinements made based on successful tests

“Fault-Tolerant Systems (Second Edition).” Israel Koren, C. Mani Krishna. Morgan Kaufman, 2021.

“Reasoning about the Reliability of Diverse Two-Channel Systems in which One Channel is ‘Possibly Perfect’.” Bev Littlewood, John Rushby. IEEE Transactions on Software Engineering, 2011.

“Guided Architecture Trade Space Exploration: Fusing Model Based Engineering & Design by Shopping.” Sam Procter, Lutz Wrage. Software and Systems Modeling, 2021.



[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Design Diversity for: Software Dependability

Q2: When is it effective, especially compared to alternatives?

- Design diversity relies on precise, accurate requirements / specifications
  - … but so do various types of formal methods.
- In what situations is design diversity preferable to formal methods?
  - Types of systems?
  - Domains?
  - Implementation technologies / approaches?

Design Diversity for: Software Dependability

Q3: Can we lower the cost?

- What is the role of AI-assisted “low-code” or even “no-code” tools?
  - Cost reductions could (dramatically) change the practicality calculation
- Low-code:
  - Amazon CodeWhisperer, Github Copilot
  - “Still makes many mistakes—including critical errors” – Ruben Martins
- No-code:
  - AlphaCode (from Alphabet’s DeepMind)
  - Outperforms 45.7% of programmers in competitions

“AI Rewrites Coding.” Samuel Greengard. Communications of the ACM, 2023.

“AI learns to write computer code in ‘stunning’ advance.” Matthew Hutson. Science, 2022.

Is Design Diversity Essential / Effective / Practical for Critical Systems?

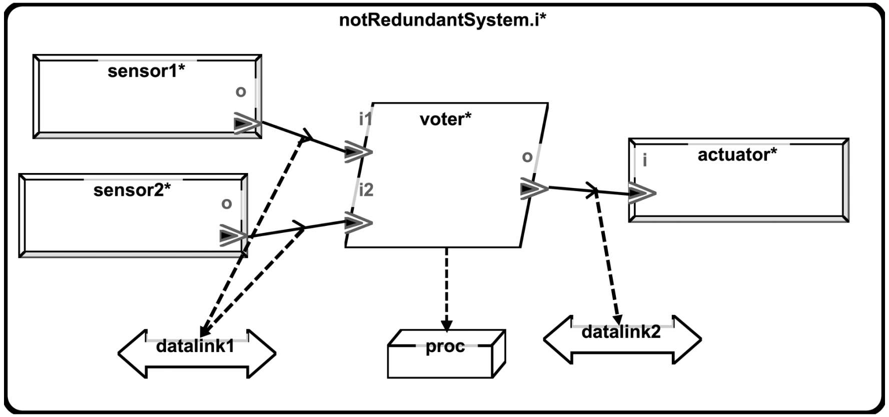
Design Diversity for System Dependability

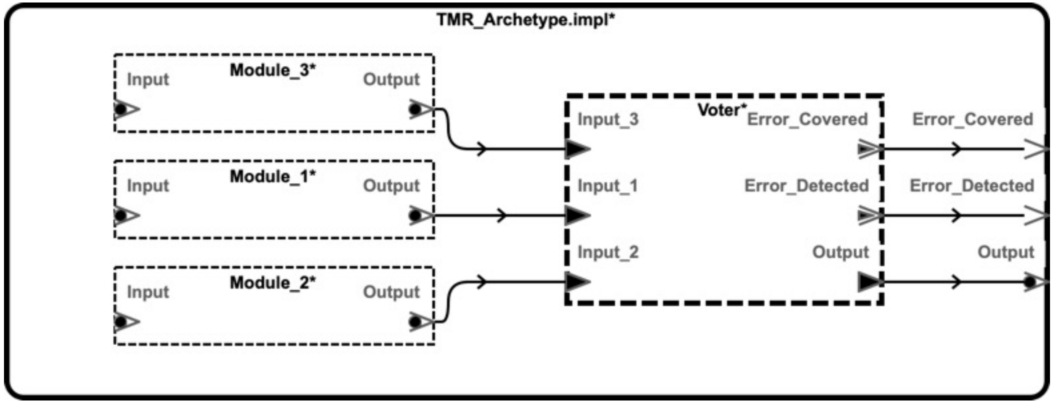
Carnegie Mellon University Software Engineering Institute

Design Diversity for: System Dependability

Avoiding implementation errors

- Design diversity – and redundancy more generally – can be challenging to implement correctly
  - Insufficient redundancy can undercut careful design
- Solution: Patterns

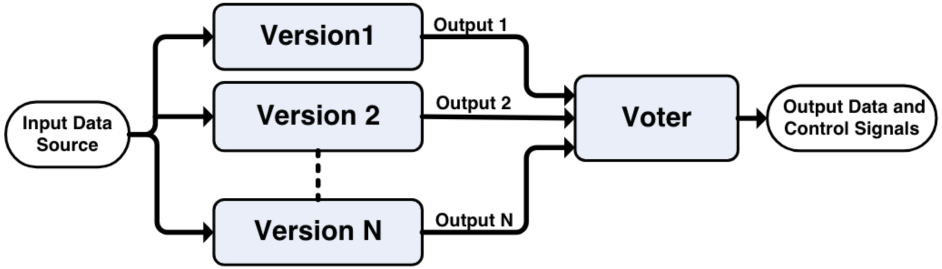




Design Diversity for: System Dependability

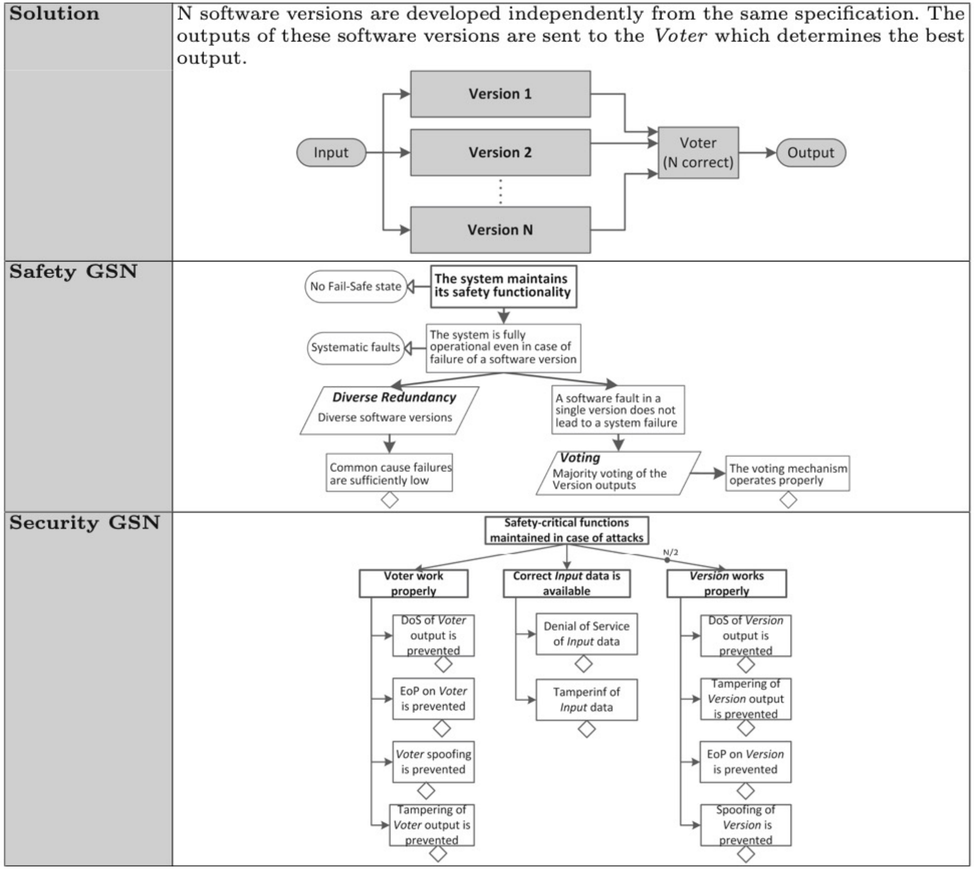
Pattern library and analysis

- Armoush's library of design patterns
  - Analysis for safety, alignment with various standards
- Preschern et al's analysis of patterns' safety and security



"Design Patterns for Safety-Critical Embedded Systems." Ashraf Armoush. Dr.-Ing. Dissertation, RWTH Aachen University, 2010.

"Safety Architecture Pattern System with Security Aspects." Christopher Preschern, Nermin Kajtazovic, and Christian Kreiner. TPLOP IV, 2019.



Solution N software versions are developed independently from the same specification. The outputs of these software versions are sent to the *Voter* which determines the best output.

Version 1
Input
Version 2
Voter
(N correct)
Output
Version N

Safety GSN

No Fail-Safe state
The system maintains its safety functionality
Systematic faults
The system is fully operational even in case of failure of a software version
Diverse Redundancy
Diverse software versions
A software fault in a single version does not lead to a system failure
Common cause failures are sufficiently low
Voting
Majority voting of the Version outputs
The voting mechanism operates properly

Security GSN

Safety-critical functions maintained in case of attacks
N/2
Voter work properly
Correct Input data is available
Version works properly
DoS of Voter output is prevented
Denial of Service of Input data
DoS of Version output is prevented
EoP on Voter is prevented
Tamperinf of Input data
Tampering of Version output is prevented
Voter spoofing is prevented
EoP on Version is prevented
Tampering of Voter output is prevented
Spoofing of Version is prevented

Design Diversity for: System Dependability

Language and Tool Support

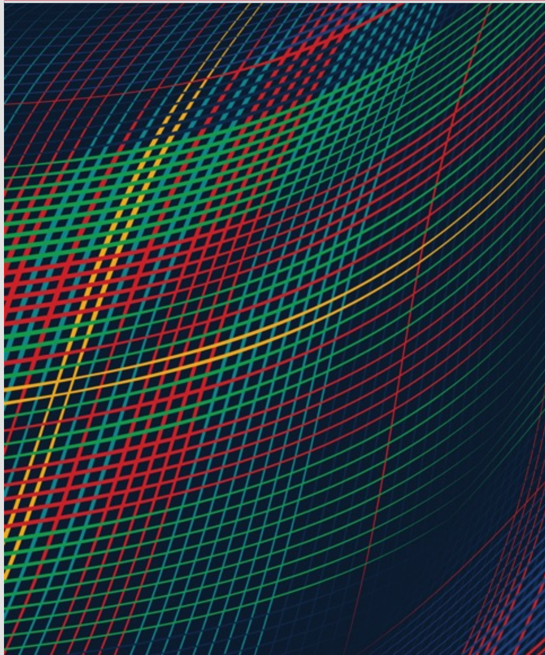
- Tool-based assistance for using patterns
  - Definition / templates
  - Enforcement / checking
- AADL Ecosystem:
  1. ReqSpec: Define requirements
  2. AADL: Defines static and dynamic pattern architecture
  3. AGREE: Contract definition and verification on individual components
  4. Resolute / Awas / OSATE Slicer: Definition of basic verification methods
  5. ALISA: Specify relationships between requirements, architecture, and verification methods

Wrap-up

Collected Discussion Questions

- What lessons can the safety (or dependability) community take from security's automation-first approach to design diversity?
- When is design diversity preferable over formal methods?
- What, if any, is the role for low- / no-code (generative AI) tools in producing diverse implementations?
- How can we predict:
  - Which systems might benefit from design diversity?
  - How much a particular system would benefit from diverse implementations?

Carnegie Mellon University Software Engineering Institute



Is Design Diversity Essential / Effective / Practical for Critical Systems?

Sam Procter

sprocter@sei.cmu.edu