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

MAY 12, 2023
Sam Procter
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?

Essential = Superior to alternatives

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


Is Design Diversity Essential / Effective / Practical for...

<table>
<thead>
<tr>
<th></th>
<th>System Safety</th>
<th>Security</th>
<th>Dependability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Systems</td>
<td>Only as a follow-on effect of dependability</td>
<td>Automated diversification</td>
<td>Effective, but hard to measure, not universally practical</td>
</tr>
<tr>
<td>Systems</td>
<td></td>
<td>???</td>
<td>Essential, but easy to accidentally undo</td>
</tr>
</tbody>
</table>

Effective! (Subject to Caveats)

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

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

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:** \(0.9999 \approx 1 \approx 1.0001\)
- **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

---


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.”

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


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


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

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

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

Sam Procter
sprocter@sei.cmu.edu