Applying Design Diversity to Aspects of System Architectures and Deployment Configurations to Enhance System Dependability

Matthew J. Hawthorne
Dewayne E. Perry
The University of Texas at Austin

Introduction

- Dependable Systems: Software-based systems with very high reliability requirements
- Examples (current and potential):
  - Aerospace applications
  - Nuclear power plant controls
  - Other industrial production and transportation
    - Especially environmental and safety-critical systems
  - Web servers, application servers
    - Critical for many companies
    - May be sole means of service delivery, transaction processing

The Challenge of Dependability

- Software is an increasingly integral part of the systems on which we depend
- Two characteristics of software-based systems:
  - Pervasiveness: Automation ≈ software-based systems
    - Even embedded "hardware" systems usually include significant software components
  - Complexity
    - Functional complexity
    - Legacy complexity
    - Application and component frameworks
    - Hardware and operating system complexity

Enhancing Software Dependability

- Process improvement: ISO 9001, SEI Maturity Model, Unified Process, Agile methods, …
- Architecture and design: CBSA, MDA, UML, …
- Engineering testing (component/unit testing)
- Verification and validation (QA, field testing)

Redundancy

- Used to enhance dependability
- Software-based systems present special challenges
- Software errors or vulnerabilities are almost always the result of development errors, e.g.:
  - Incorrect or incomplete requirements
  - Design or implementation errors
- Major problem: Positive failure correlation
  - Different versions tend to fail under the same, or overlapping, sets of conditions (inputs)

Design Diversity

- Try to reduce inter-version error correlation with "diversity-enhancing" development decisions
  - Mutual isolation of development teams
  - Different programming languages
  - Different architecture and design patterns
  - Different development and testing methodologies
- Design diversity research usually considers only the application under development

➢ Limited by the scope of the diversity-enhancing development decisions

Extending Design Diversity: Layered Components

- Non-trivial software components are almost certain to include unknown defects and vulnerabilities
- As development environments become more component and framework oriented, underlying systems become more complex
  - Most of the complexity of many systems is *below the application level*
- Layered component diversity can help protect against system and third-party defects

Extending Design Diversity: Hardware and Operating Systems

- Hardware and operating systems are also becoming more complex
  - Viruses, worms, etc., often attack only certain operating systems, operating system families, or different operating systems on the same hardware platform
  - Example: Dozens of security-enhancing fixes for the Windows OS
- Operating system and hardware diversity can help protect against OS- or hardware-specific errors or vulnerabilities

Extending Design Diversity: Network and Infrastructure

- Modern systems depend on connectivity
  - Network outage → system/node inoperative
- Systems depend on power supply, other infrastructure
  - Power outage → system/node inoperative
- Diversity in networking, power supply, and other infrastructure can help protect against infrastructure-induced system failures

Diversity-Enhancing Properties

- Modal diversity
- Geographical diversity
- Ecological diversity
- Other diversity properties:
  - Temporal diversity
  - Control diversity
  - Combinational diversity

Modal Diversity

- Provide for diverse modes of accomplishing system functions
- Example: Diverse UI modes
  - Power plant operator alert system
  - Primary UI mode
    - Graphical user interface (visual, auditory signals)
  - Backup UI modes
    - Operator's digital pager
    - Supervisor's mobile phone

Geographical Diversity

- Distribute hardware-software components geographically to avoid local failures
- Example: Diverse locations
  - Web application server-based system
  - Distributed redundant servers in London, Paris, Milano, New York and San Francisco

Ecological diversity

- Use diverse hardware, software, network and infrastructure components to protect against hardware or software-specific errors or vulnerabilities
- Example: Diverse networks (also modal diversity)
  - Primary Network: T1 line via Ethernet
  - Backup Networks: DSL modem, leased satellite link

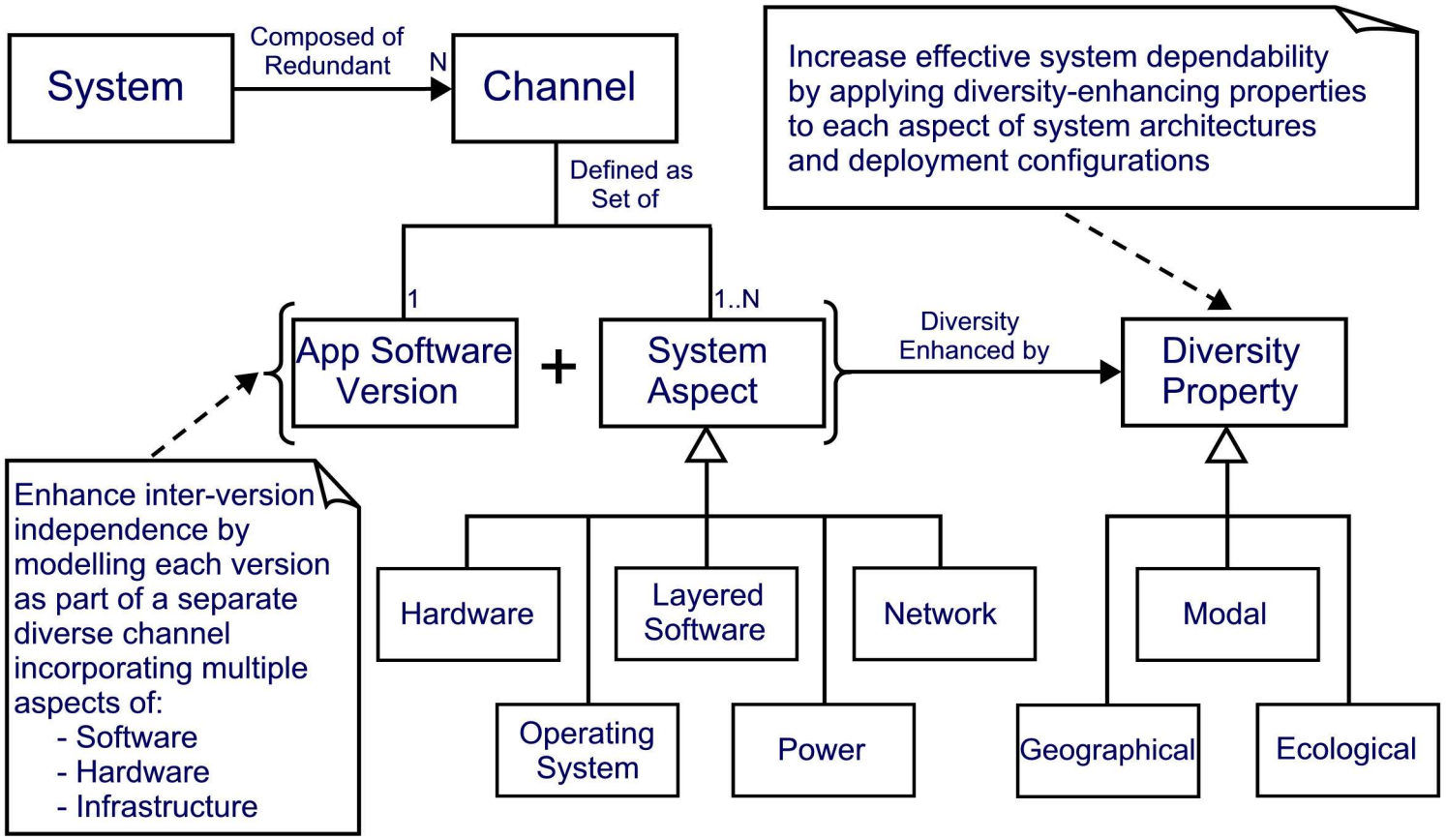
Other Diversity Properties

- Temporal diversity: Ability of system to adapt to temporal variability (variable event delays; temporal decoupling)
- Control diversity: Diverse automatic and human control systems (control decoupling)
- Combinational diversity: *Combination* of hardware-software components is diverse, even if not all the individual components are unique

Architectural Framework

- Diverse redundant hardware-software-infrastructure “channels”
- Channels ideally incorporate *top-to-bottom* design diversity
- May also leverage combinational diversity: Diverse *combinations* of hardware and software in different channels

Conceptual Model for Diverse Systems



Diverse Channel System Architecture Example

Channels (Diverse System Nodes)

Aspects of system being modelled:	Lon_1	Mil_2	NY_3	Aus_4	Diversity Properties
App. Version	V1	V2	V3	V4	Ecological
App. Framework	.NET	EJB	EJB	EJB	Ecological
Operating System	XP	Linux	Linux	OSX	Ecological
Hardware Platform	x86	x86	Mac	Mac	Ecological
Network	EurBB1, Sat2	EurBB2, Sat1	NABB1, Sat1	NABB2, Sat2	Modal, Ecological
Power Supply	Local, Gen	Local, Gen	Local, Gen	Local, Gen	Modal
Location	London	Milano	New York	Austin	Geographical

Lon_1, NY_3: Mutual diversity with respect to all modeled hardware-software aspects of system
Mil_2, Aus_4: Partial hardware-software diversity, some commonality (combinational diversity only)
All channels: Geographical diversity, modal network and power supply diversity

Conclusions

- Top-to-bottom design diversity for dependable systems incorporates the whole system:
  - **Software**: Applications, layered components, and operating systems
  - **Hardware**: Processors, storage units, etc.
  - **Infrastructure**: Networks, power supplies, etc.
- Use properties like *modal*, *geographical*, *ecological*, and *temporal diversity* to evaluate dependable system designs
- *Diverse hardware-software-infrastructure channels* can provide multi-level redundancy

Current and Future Developments

- Architectural frameworks to enable the design and development of systems with top-to-bottom diversity
  - Aspect-oriented approaches show some promise to help configure multi-level diversity in the software parts of the system
- Distributed intelligent service provider based self-directed system
  - Diverse nodes
  - Common request/reply/routing protocol