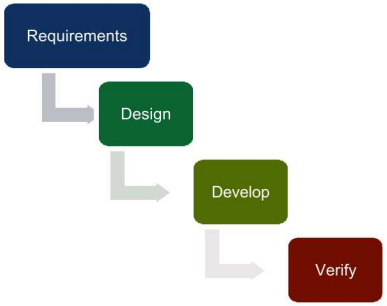
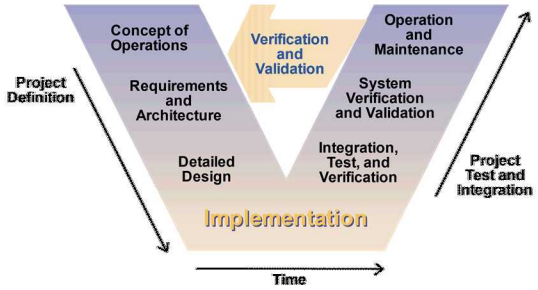
SAND2018-6843C





Requirements Efficiency: Challenges and Best Practices

Celeste A. Drewien, Patricia Hubbard
Systems Analysis Org.
Cheryl Bolstad
Human Factors Org.
Raymond Wolfgang
System Surety Engineering Org.

May 16, 2018

Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Sandia National Laboratories

Sandia National Laboratories

Albuquerque, New Mexico

SNL works National Security programs with other government agencies, industry, and academic institutions to accomplish missions in the following strategic areas:

Nuclear Weapons
Defense Systems & Assessments
Energy & Climate
Global Security

Livermore, California

For more information, please see: http://www.sandia.gov/about/index.html

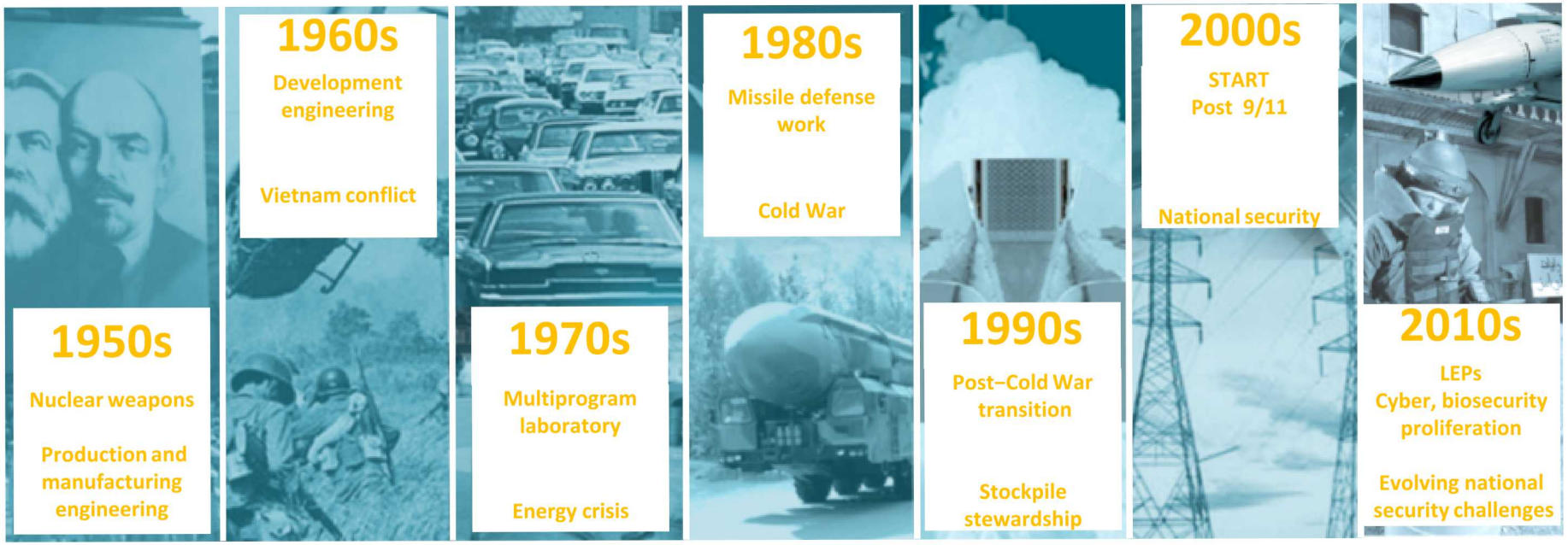
Kauai, Hawaii

Waste Isolation Pilot Plant, Carlsbad, New Mexico

Pantex Plant, Amarillo, Texas

Tonopah, Nevada

Sandia Has a 70 Year History in Working National Security Challenges



Initiate → Define → Design & Develop → Produce → Operate → Dismantle & Disassemble

Study Basis

Where can a program gain efficiency in its system engineering requirements processes?

Improve the efficiency of the requirements engineering and management processes, and thereby reduce organizational impacts, product realization delays, costs, and schedule delays due to requirements processes

Goals:

- Understand existing requirements processes
- Analyze inefficiencies in processes
- Identify opportunities for consistency in processes
- Identify opportunities to streamline processes

- Approach (3 parts)
  - Literature review → Current Presentation
  - Internal benchmarking - ongoing
  - External benchmarking - ongoing

Requirements Management
Elicitation
Specification
Derivation and Decomposition
Validation and Verification (V&V)
Change

External Literature Review

- >65 Articles from recent Peer-reviewed Journals and Conference Proceedings
- High Relevance Articles
  - ~60% by International authors
  - ~75% from Academia
- Majority of Articles Addressed:
  - General requirements topics
  - Requirements validation and verification
- Fewest Articles Found for Metrics

Relevance: Focus on improvement in requirements process for complex hardware systems with long realization timelines, long lifetimes, multiple stakeholders, and many authority requirements (safety, security, health, environmental, etc.)

Sandia National Laboratories

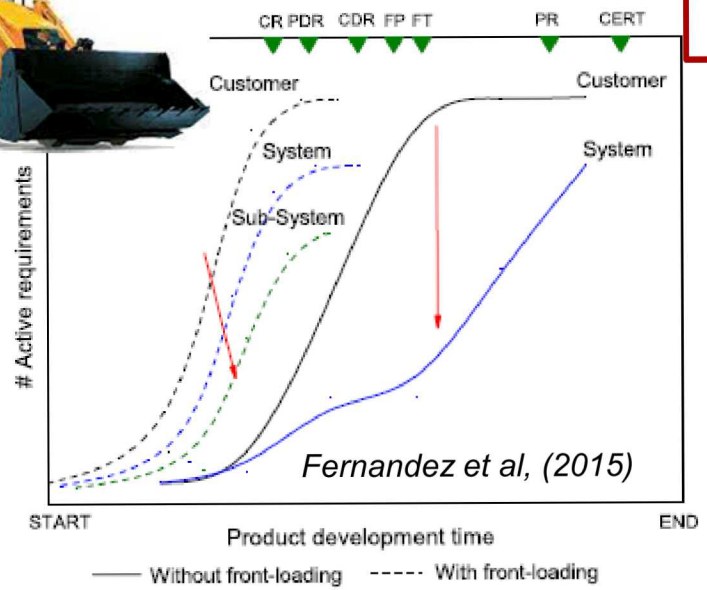
Outline

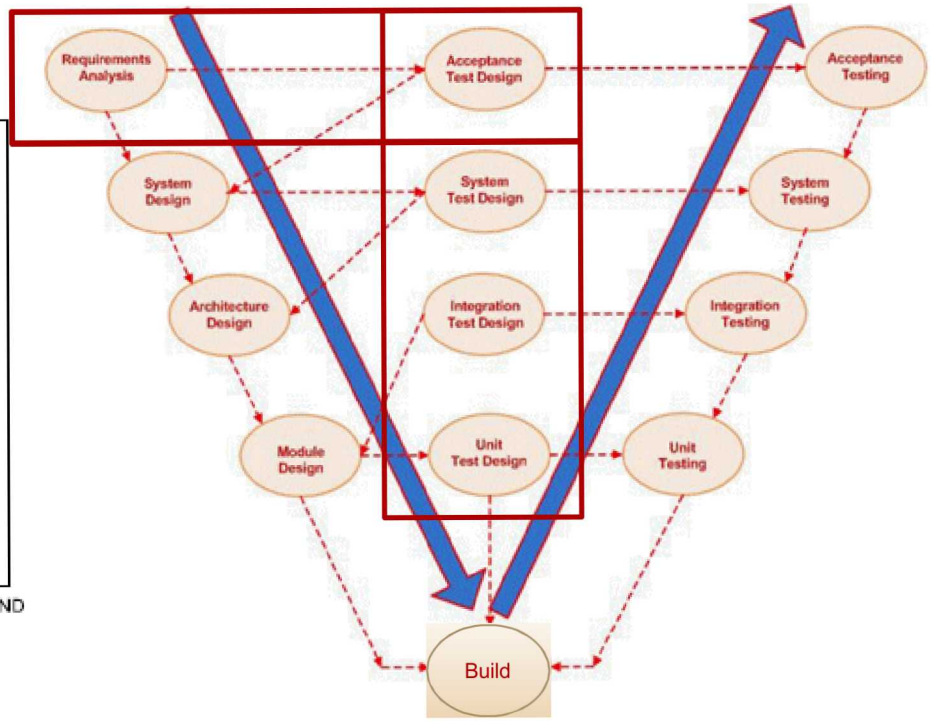
- Main Findings
  - Avoid Big Requirements Up Front (BRUF)
  - Anticipate Changing Requirements
- Improving Requirements Engineering and Management Processes
  - General
  - Requirements Elicitation
  - Requirements Specification
  - Requirements Derivation and Decomposition
  - Requirements Validation and Verification
  - Requirements Change
- Summary

Main Finding: Avoid Big Requirements Up Front (BRUF)

- BRUF is a large, detailed set of requirements developed from customer interactions and source documents by requirements engineers early in a program and then issued to design team
- Challenges
  - Requirements elicitation requires domain knowledge
  - Designers eager to design
  - Very hard to think through all the requirements up front
  - Can result in less team communication – communication by documentation instead
- Very costly to implement later requirement changes
  - Changes may require significant rework of requirements, design, and verification plans

Best Practice: CRAVE via Front Loading

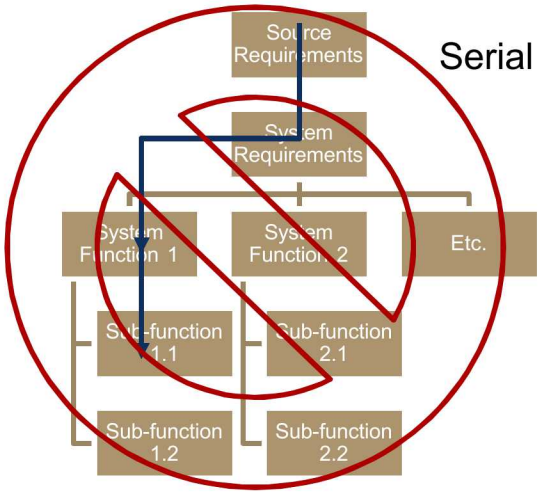


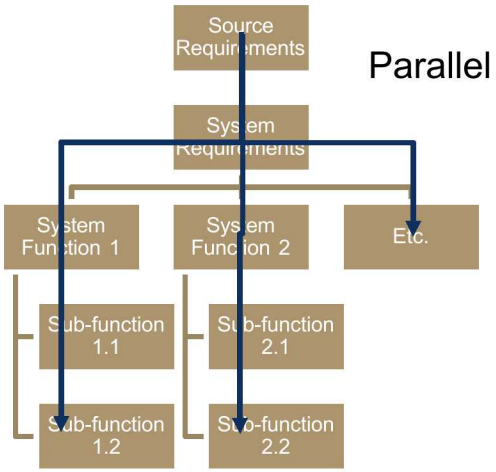


- Ensure participation of customers, system, and sub-system early in process
- Capture Critical Requirements and Verification Early (CRAVE)
- Develop functional architecture as basis of requirements
- Link requirements and testing up front
- Release requirements as needed by design team in timely manner
- Continue managing requirements after release

Implementing Front Loading

- **Use Increased Level of Concurrent Requirements Engineering**
  - Frequent requirements coordination/integration meetings
  - Standard procedures for how higher-level requirements teams engages with dependent lower-level requirements teams
- **Invest in Initial Increase in Allocation of Resources**
  - Prevent requirements activities from becoming serialized through one individual
  - Allow parallel flow down of requirements
  - Include all stakeholders, particularly domain experts, design engineers, and test engineers in process
  - Ensure verification of customer requirements if meet-in-the-middle approach is used
- **Focus on Critical Requirements Up Front (CRUF)**
  - Confirm required versus desired operational needs
  - Identify critical functionality
  - Establish Measures of Effectiveness (MOE) or success criteria for critical requirements
  - Establish Key Performance Parameters (KPPs) needed to verify critical requirements
- **Delay Non-Critical Requirement Details That May Change**



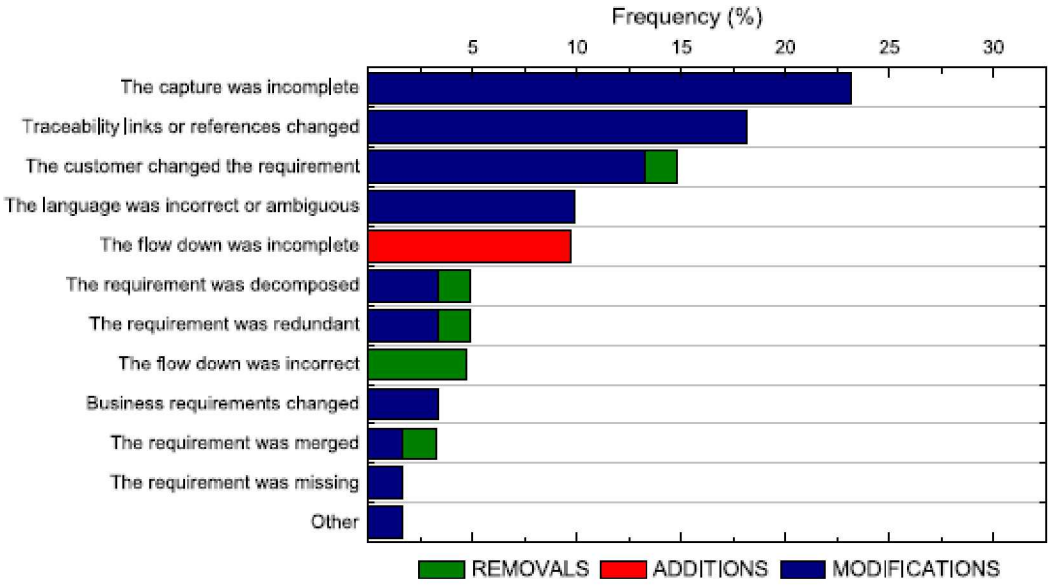


Comparison and Contrast of BRUF and CRAVE with Front-Loading

Sandia National Laboratories

	BRUF	CRAVE via Front-loading
Requirements Elicitation	Customer(s) Requirements Team	Customer(s) and other stakeholders, such as design and test engineers Requirements team
Requirements Approach	May be serial or parallel All source to all system requirements	Parallel Source to functional requirements
Requirements Management	Detailed from start	Detail limited to only what was needed immediately by design and test engineers, followed by more detail later on
Requirements Derivation and Decomposition	Performed by Requirements Team	Worked by key stakeholders and overseen by Requirements Team
Requirements Validation	Performed at Requirements Review	On-going validation
Requirements Verification	Verification requirement may be developed in conjunction or after the fact	Verification requirement developed in conjunction with requirement

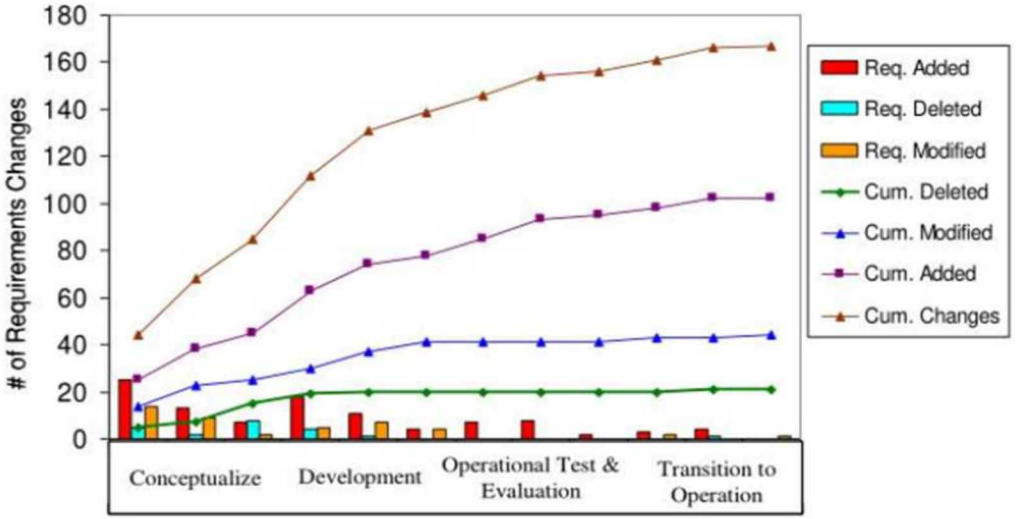
Main Finding: Anticipate Changing Requirements



Reasons for requirements volatility or changes are well documented

Requirements change across the lifecycle phases

Phase transitions tend to generate requirements changes



Fernandez et al, (2015); Pena, (2015)

Sandia National Laboratories

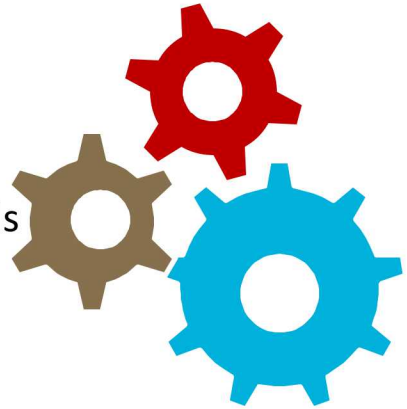
Outline

- Main Findings
  - Avoid Big Requirements Up Front (BRUF)
  - Anticipate Changing Requirements
- Improving Requirements Engineering and Management Processes
  - General
  - Requirements Elicitation
  - Requirements Specification
  - Requirements Derivation and Decomposition
  - Requirements Validation and Verification
  - Requirements Change
- Summary

Improving Requirements Management

Sandia National Laboratories

- Challenges
  - Establishing adequate requirements in timely manner
  - Requirements engineer is not an expert in the domain
  - Inadequate communication between requirements engineer, designers/developers, stakeholders
  - Unstable and changing requirements



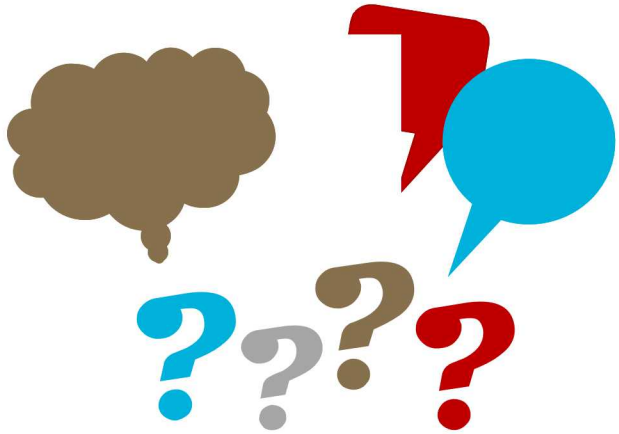
- Best Practices
  - Employ requirements engineer(s) with strong communication skills
  - Use consistency and uniformity
    - In plans, products, processes from system-level to lower level
    - Requirements specification template and reporting
    - Common taxonomy (e.g., SEBok)
  - Avoid BRUF
  - Limit use of requirements tools* until use cases and functional analysis begin to solidify
    - Track metrics (e.g., Count, Traceability, Volatility) to communicate trends

*Some individuals may be able to rapidly use tools

Improving Requirements Elicitation

Sandia National Laboratories

- Challenges
  - Multiple methods exist for requirements gathering
  - Missing requirements
  - Requirements apply at differing levels
  - Challenges of interfaces



- Best Practices
  - Develop a plan for the requirements gathering process
    - Ensure a lifecycle perspective is considered and proper stakeholders are included
    - Agree on prioritizing requirements through verbs or designated priority levels
    - Define approach for interface requirements
      - Analyze the requirement (inputs and responses) across both sides of the interface
      - Define accepted format and practices for interface specifications
      - Agree on who owns the interface requirements and has verification oversight
  - Study existing products and information for functional requirements, to understand various levels where requirements might apply, and to look for product improvement opportunities
  - Have on-going meetings and communication with stakeholders
  - Utilize various modeling techniques for requirements identification

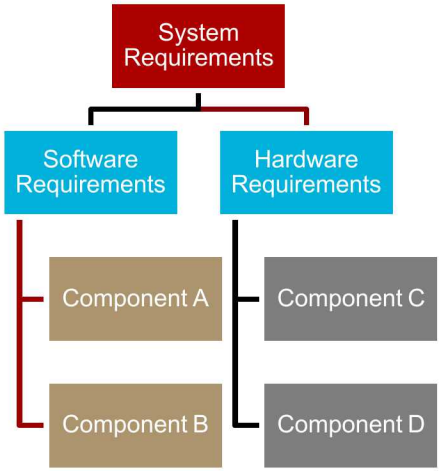
Improving Requirements Specification

- Challenges
  - Separating needs from solutions
  - Tackling inconsistency and ambiguity, e.g., standards and authority documents
  - Writing requirement with a sound verification pathway
  - Removing/avoiding redundancy

- Best Practices
  - Work with customer(s) to determine “what” instead of “how”
  - Assign responsibility and establish repository for standards and authority documents, including handling of revisions
  - Take advantage of patterns and automation
    - Use standard format/template and if possible requirements patterns
    - Consider requirements authoring tools, natural language parsing tools, auto-ballooning capture tools for documents or drawings
  - Couple verification testing and acceptance criteria with requirements
  - Use component databases for commonly used or already qualified parts
  - Use Fagan inspection for quality in requirements specification

Improving Requirements Derivation and Decomposition

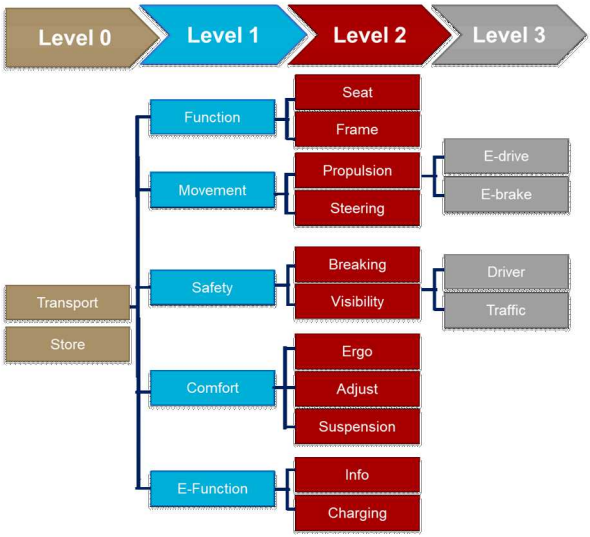
- Challenges
  - Requirements derivation is highest in early design stage, when designers are working out their design proposals
    - Detailed requirements may not be generated until a design is selected
  - Derived requirements may not include explicit state constraints (only implied)
  - Non-functional requirements may receive less attention
  - Designers/engineers as they work out the design:
    - May estimate requirements
    - Base requirements on feelings or intuitions
    - Skip some steps or sequences
    - Place prioritization/value on certain requirements
    - Use jargon
  - Requirement decomposition can lead to flow down errors



These can lead to requirement changes

Improving Requirements Derivation and Decomposition

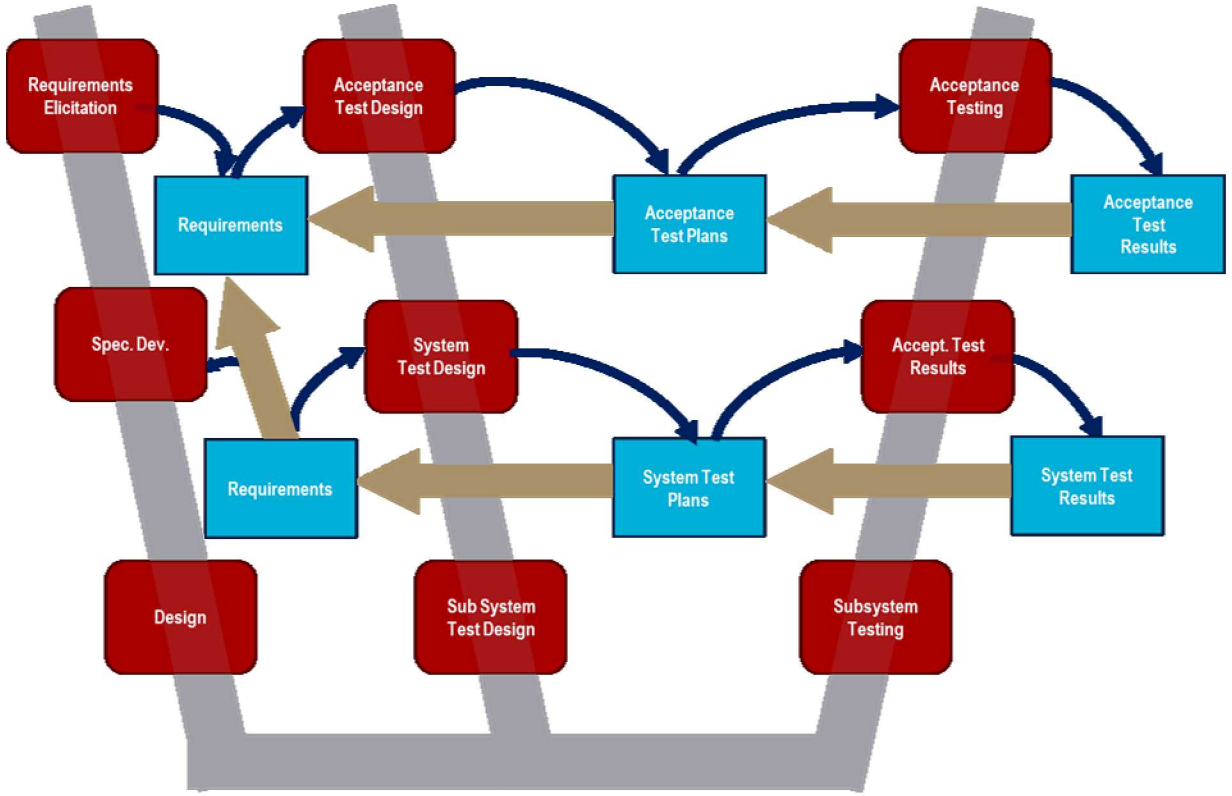
- Best Practices
  - Work with designers to properly specify requirements
  - Sequence or separate requirements to describe how the system behaves from one state to the next
  - Plan sufficient time for deriving non-functional requirements and environments for which functions apply
  - Understand and avoid common flow down errors
    - Duplicated requirements with differing performance parameters
    - Improper allocation to components
    - Non-KPPs overlooked
    - Parameter mismatches (e.g., units of measurement)
    - Dangling requirements (parentless, child missing)
    - Unwarranted assumptions



Sandia National Laboratories

Improving Requirements V&V

- Challenges
  - Assume similitude with other designs/projects
  - Failure to understand extreme loadings/environments
  - Inadequate review of test cases against requirements



System Engineering "W Model" Incorporates Verification Requirements, Planning, and Test Results into the Traditional "V Model" (A.J.J. Dick, 2012)

Sandia National Laboratories

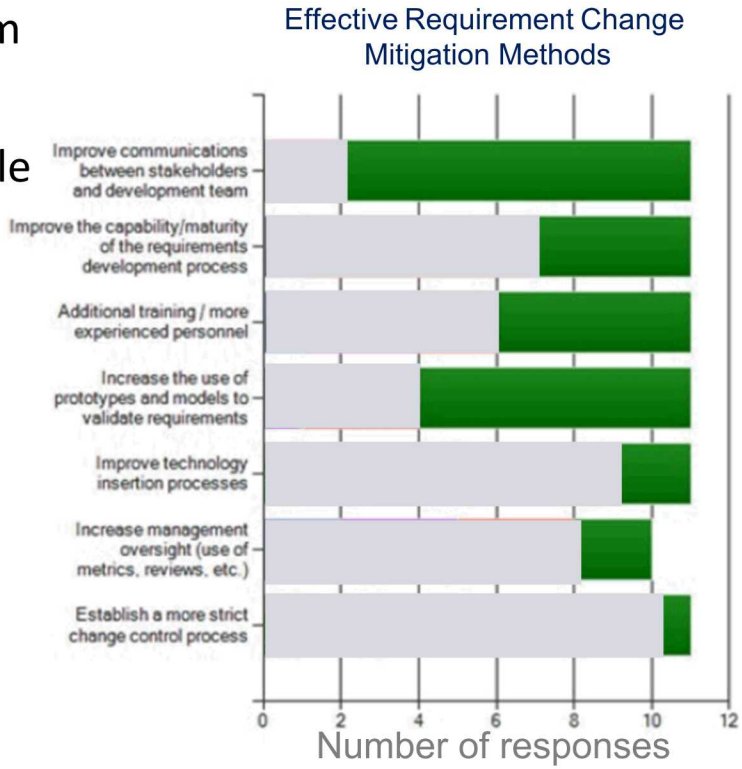
Improving Requirements V&V

- Best Practices
  - Validate requirements with variety of means
    - Use rapid prototyping – particularly useful for early validation efforts
    - Communicate with stakeholders
  - Review all requirements, including validation and verification plans
    - Include multiple organizations
    - Confirm traceability of all requirements (matrix)
    - Ensure full test coverage
    - Identify and quantify limit states or constraints
  - Use structured argumentation or assurance case for requirements verification
    - Capture context and assumptions

Effectively Managing Requirements Change

- Challenges
  - Requirements change throughout program
  - Not all parties informed of change
  - Greater impact later in the system life cycle
- Best Practices
  - Understand origins of requirements change
    - Change in customer(s) needs
    - Scope reduction or expansion
    - Part change or functionality enhancement
    - Derivation or decomposition mistakes
    - Verification issues
    - Defect fixing
  - Set requirements chill and freeze dates
    - Establish "critical no change" date
  - Improve communication between requirements engineers, test engineers, quality, and designers → involve all in change control process
  - Consider that change can be positive → increased quality, reduced costs or time

CSSE ARR Survey (2010)



Sandia National Laboratories

Outline

- Main Findings
  - Avoid Big Requirements Up Front (BRUF)
  - Anticipate Changing Requirements
- Improving Requirements Engineering and Management Processes
  - General
  - Requirements Elicitation
  - Requirements Specification
  - Requirements Derivation and Decomposition
  - Requirements Validation and Verification
  - Requirements Change
- Summary

Summary: Realizing Efficiency Gains

Sandia National Laboratories

- Avoid BRUF

- Front-load the requirements process to ensure earlier V&V and avoid late requirements changes
  - Allocate resources to ensure inclusion of all stakeholders—customer(s), domain experts, design and test engineers, etc.—in requirements
  - Specify verification requirement along with the requirement and ensure flow down
  - Have database for commonly used or reusable parts and for standards and authority documents to share across projects

- Manage requirement volatility and change
  - Understand major drivers of requirements change
  - Include stakeholders, design and test engineers, etc. in requirements change control process
  - Determine who owns and must verify interface requirements
  - Avoid common flow down errors