Raytheon

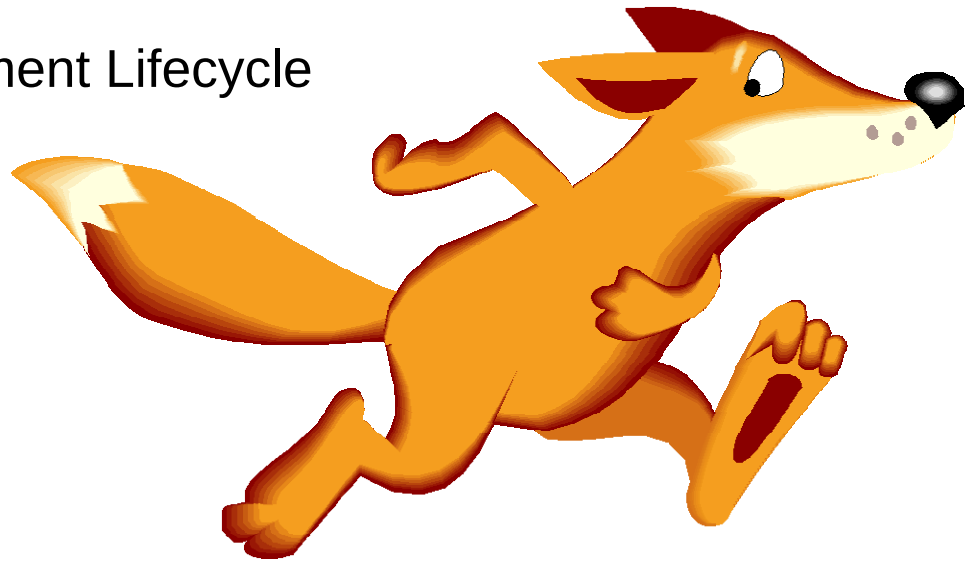
Customer Success Is Our Mission

Design for Six Sigma in the Software Lifecycle -- Did We Lose the Fox?

Jill Brooks
Sanjeev Venkatesan
11/19/2008

Agenda

- Definition of Design for Six Sigma (DFSS)
- How DFSS is Often Applied to Software
- DFSS Opportunities in Software Systems Lifecycle
  - Modeling and Simulation
  - Live Virtual Construction
  - Hardware in the Loop
  - Prototyping
- DFSS in the Software Development Lifecycle
  - Requirements
  - Architecture
  - Design
  - Code
  - Integration and Test
  - Production
  - Maintenance
- Summary

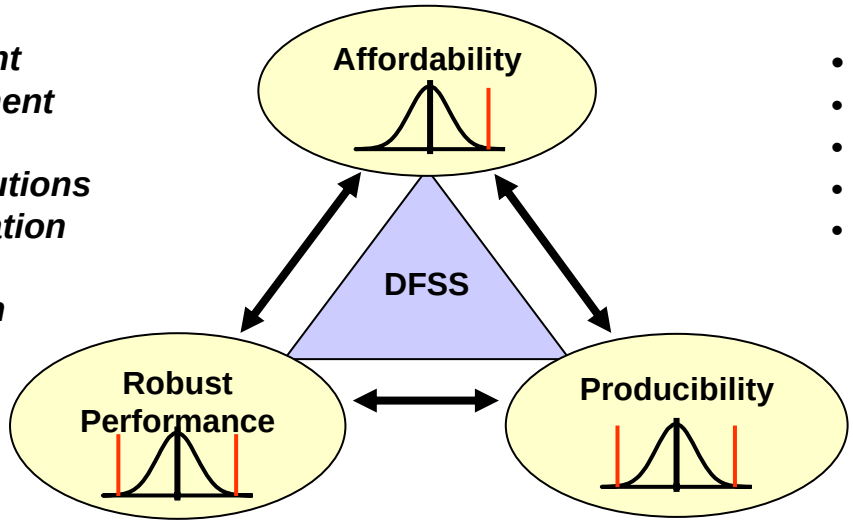


DFSS has greater application than is routinely used today

Definition of DFSS

Design for Six Sigma is a methodology used to predict, manage and improve **Producibility, Affordability**, and **Robust Performance** for the benefit of the customers.

- *Quality Function Deployment*
- *Critical Parameter Management*
- *Statistical Design Methods*
  - *General Orthogonal Solutions*
  - *Multi-Objective Optimization*
  - *Sensitivity Analysis*
  - *Requirements Allocation*
  - *Monte Carlo Analysis*
  - *Design of Experiments*
- *Measurement System Error Budgeting*
- *Reliability Prediction*

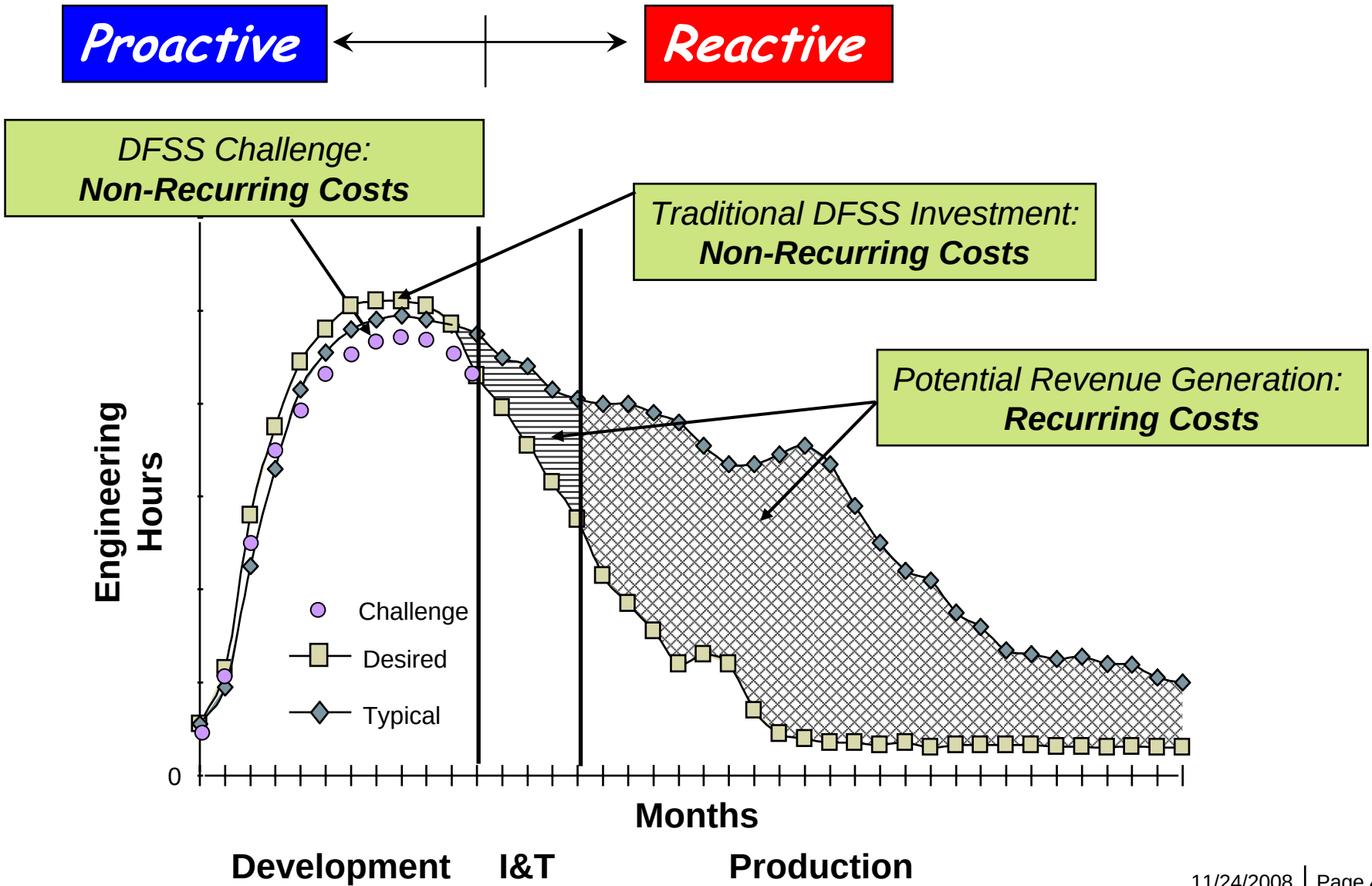


- *Design for Manufacturing*
- *Process Capability Analysis*
- *Timing and Sequencing*
- *Upgradeability Management*
- *Process Model Simulation*

- *Cost as an Independent Variable (CAIV)*
- *Cost Estimating & Tradeoff Analysis*
- *Design to Cost*
  - *Unit Production Cost*
  - *Operation & Support Cost*
  - *Life Cycle Cost*

Definition of DFSS: Why Use DFSS?

Raytheon



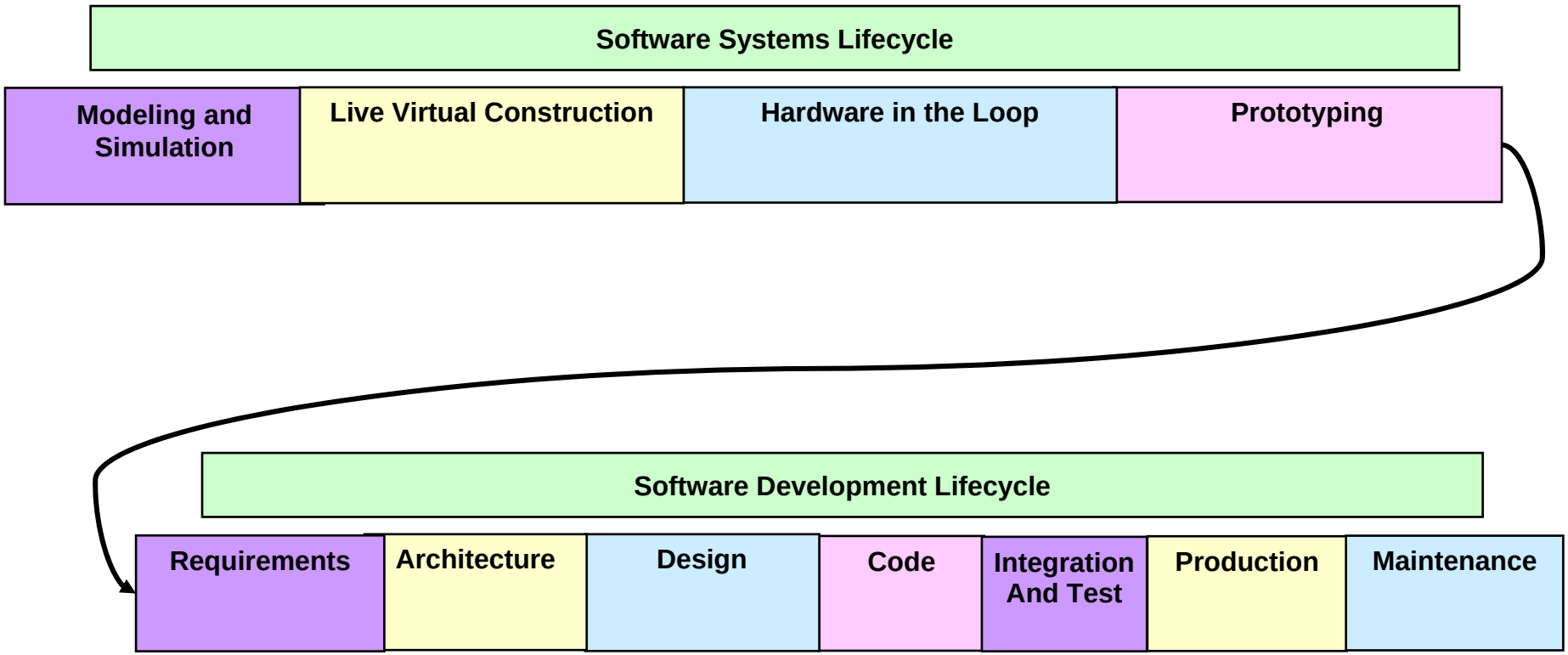
How DFSS is often applied to Software

- Historically, DFSS for Software is often considered to be:
  - Software contribution to System-Level Technical Performance Measures (TPMs) such as
    - Processor Utilization
    - Transactions per second
    - Memory Utilization
  - Defect Containment

- Some in industry say DFSS is about systems engineering and product design, independent of any particular technological domain
  - The Boeing 787 is a marvel of modern engineering…except for these 6.5 million lines of code, which were merely 'developed'.

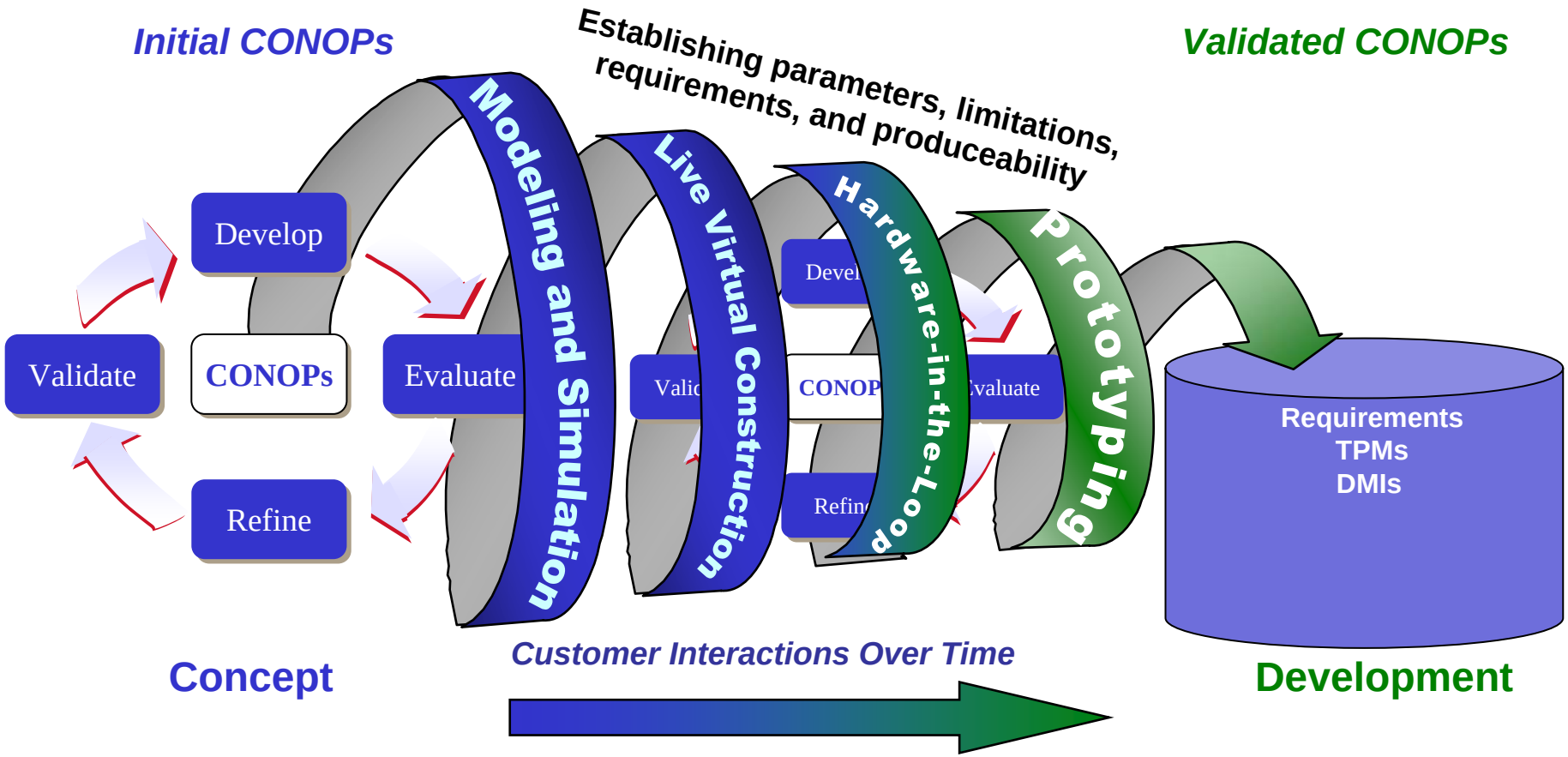
What is called DFSS for Software is often quite limited and reactive – We've lost the fox!

DFSS Throughout the Lifecycle



Need to consider DFSS opportunities through out the entire lifecycle

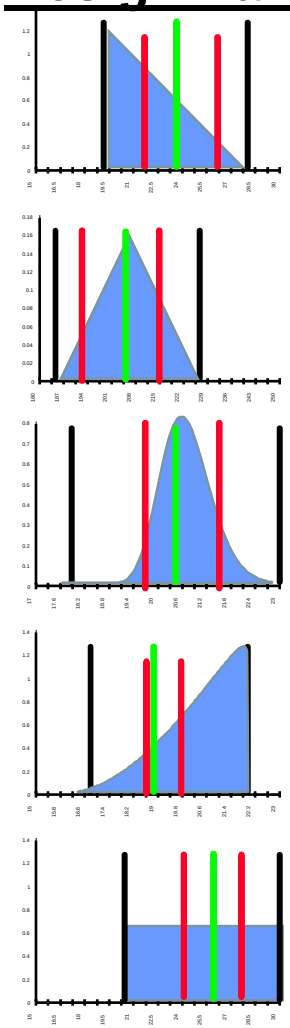
Infusing DFSS in the Software Systems Lifecycle



DFSS allows us to fine tune the design parameters

Statistical Software Systems Analysis

Design Variables



A

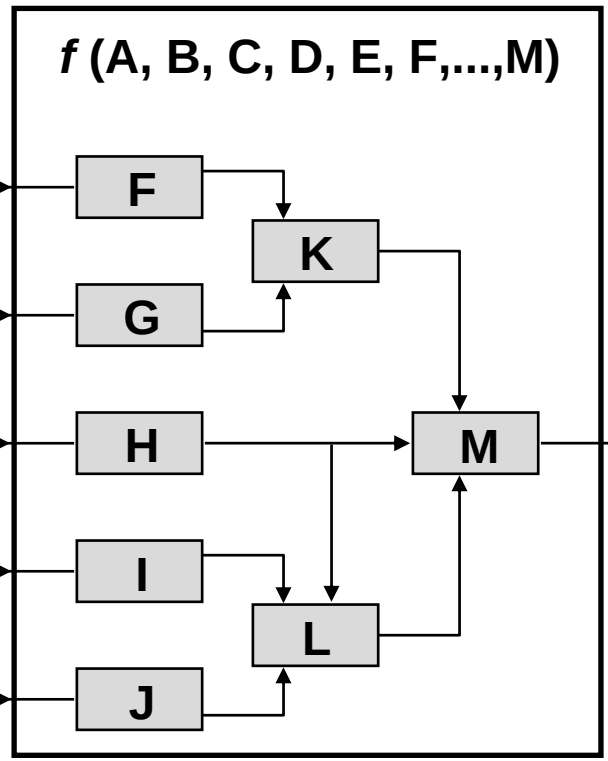
B

C

D

E

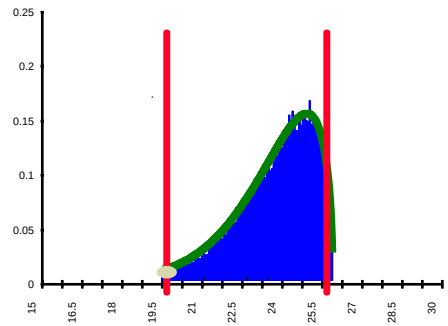
System/Sub-System Model



Response

Y = *f* (A, B, C, D, E, F,...,M)

Y

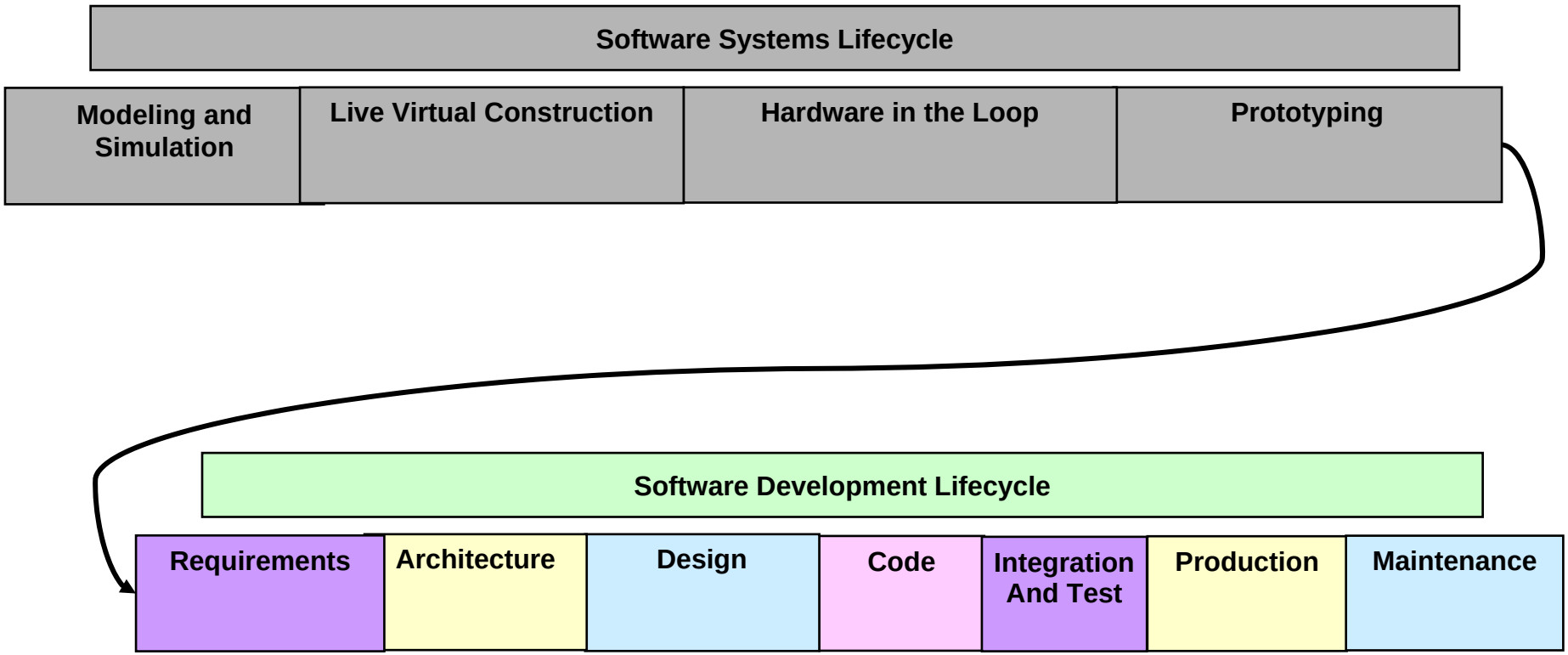


Flow Down/Allocation Based on Output Variability

DFSS Throughout the Software Development Lifecycle

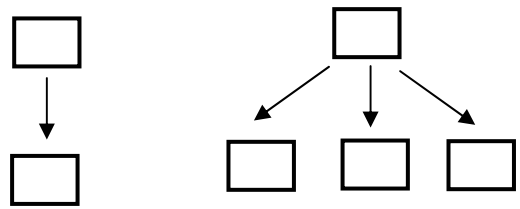
- Fundamental objectives of DFSS addresses to software development:
  - quality,
  - value,
  - reliability,
  - robustness.
- Key is to measure the maturity of the “ilities” through the life cycle
- Design for:
  - Producibility
  - Reusability
  - Maintainability
  - Upgradeability

DFSS Throughout the Lifecycle



DFSS Throughout the Software Lifecycle: Health of Software in terms of Requirements

- Requirements Creep
  - Requirements Maturity Index (RMI) = (# of req in the current build - # of req changed from the last Delivery) / # of requirements in the current build
- Level of decomposition and expansion (D&E) factors
  - D factor = # of requirements traced to the lowest component level / the total # of system requirements
  - E factor =



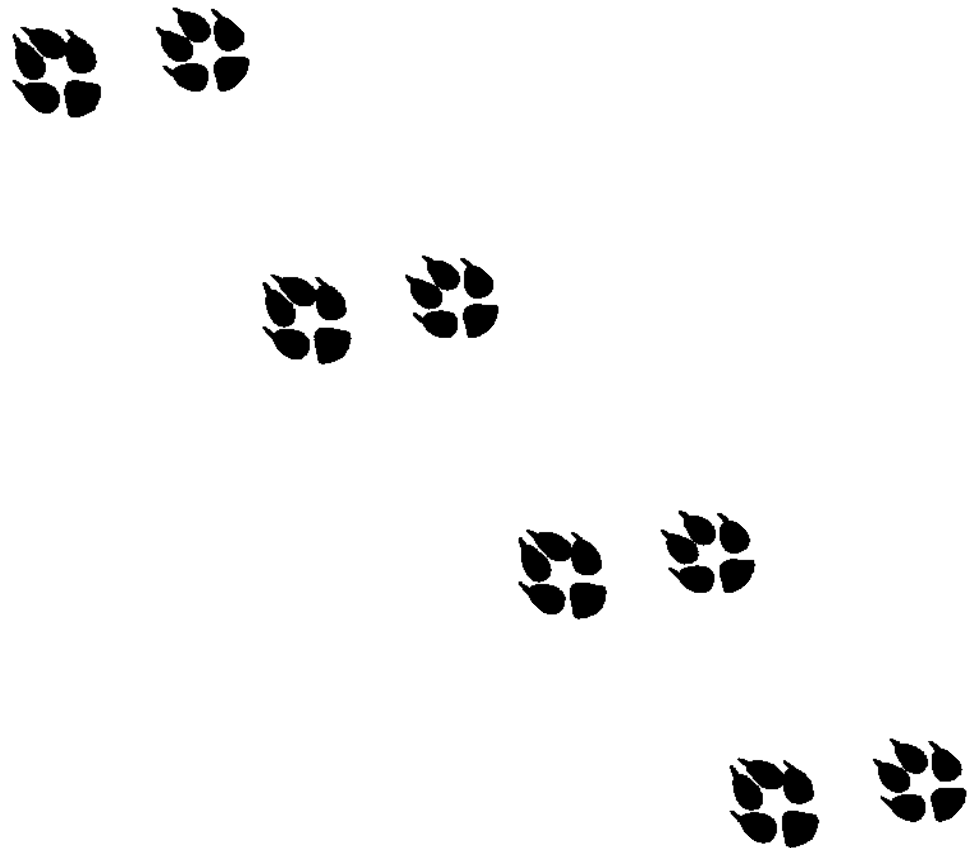
- Qualitative:
  - Effectiveness = (Total Useful Requirements *Budget available per requirement * 100)/Actual Requirements Engineering Cost per Unit
  - Efficiency = (Total Useful Requirements * 100) / Total Requirements
  - Useful Requirements do not include derived requirements

DFSS Throughout the Software Lifecycle: Requirements

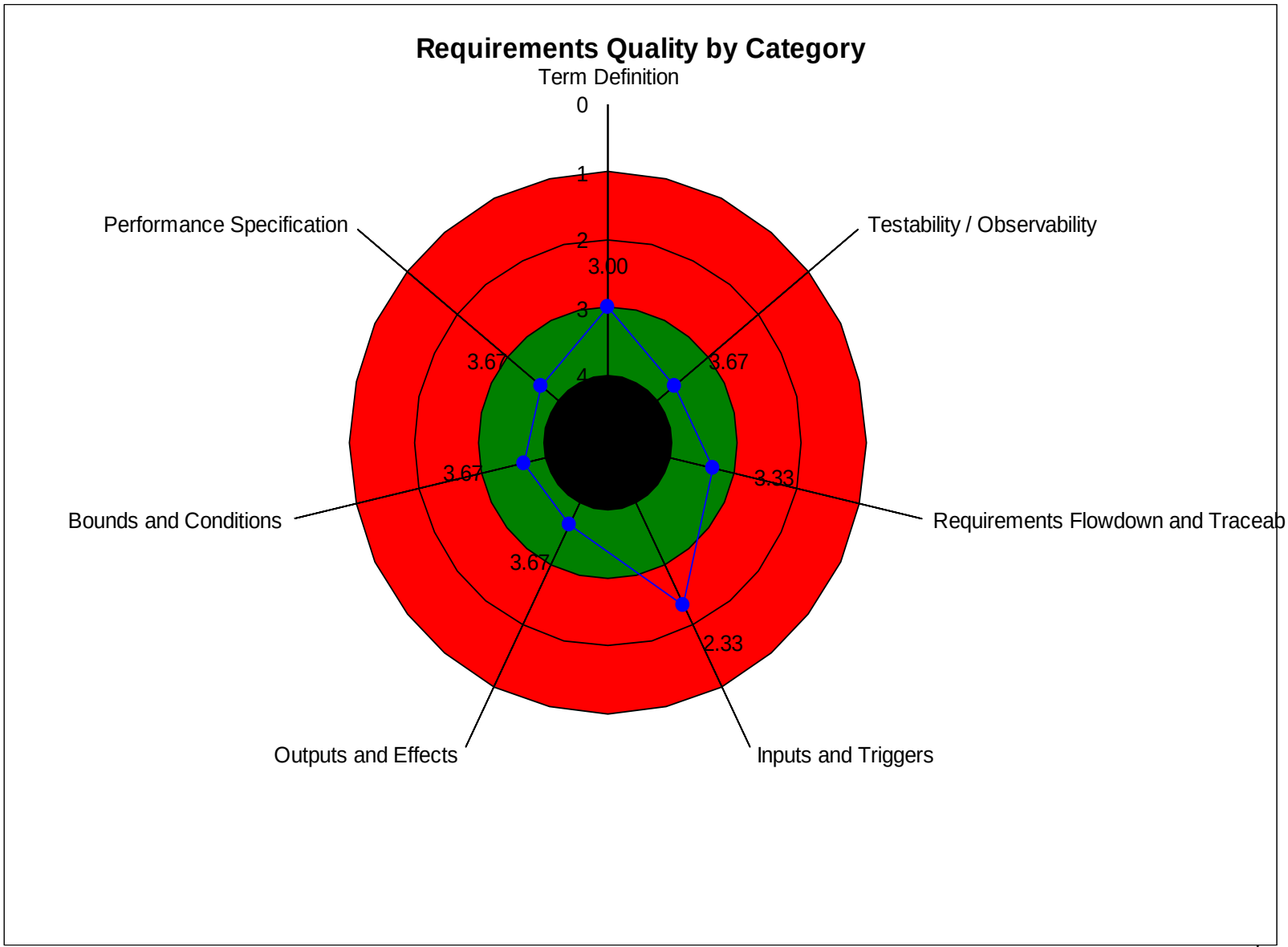
Raytheon

Quality of Requirements

- Correctness
- Completeness
- Consistency
- Clarity
- Non-ambiguity
- Traceability
- Testability
- Singularity
- Feasibility
- Freedom from product/process mix
- Relevant



DFSS Throughout the Software Lifecycle: Requirements



DFSS Throughout the Software Lifecycle: Architecture

Raytheon

- Present Architecture Evaluation techniques measure:
  - How well an architecture describes a system

- It does not measure:
  - Does not evaluate the quality of an architecture itself
    - Design Patterns VS New Construct
    - FMEA
    - Performance
    - Modifiability
    - Design Elegance

Current techniques don't give the whole picture

DFSS Throughout the Software Lifecycle: Architecture

Raytheon

Procedure for using FMEA to evaluate an Architecture

1. List the functions in the architecture
2. Look at potential failure modes of the current architecture
3. Weight the failure modes as to
   - ⇨ Probability of occurrence
   - ⇨ Severity of occurrence
   - ⇨ Probability of detection should the failure occur
4. Evaluate risk mitigation alternatives
5. Iterate the solutions and strengthen the architecture

DFSS in Software Systems activities sets the stage for refinement during architecture

DFSS Throughout the Software Lifecycle: Architecture Design Points

Raytheon

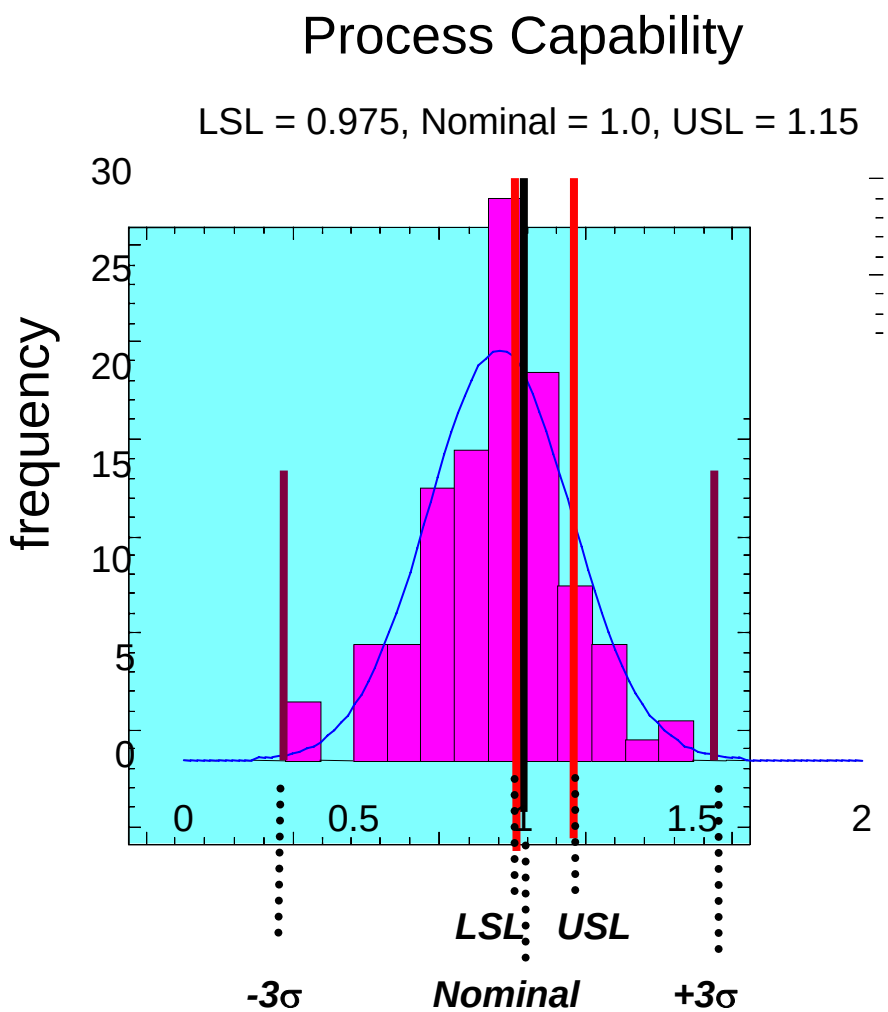
- Performance
  - Stimuli
    - Mode
    - Source
    - Frequency
  - Responses
    - Latency
    - Throughput
    - Precedence
  - Parameters
    - Resource
    - Resource arbitration
- Modifiability
  - Stimuli
    - Change to component
  - Responses
    - AMD
    - Resulting Complexity
  - Parameters
    - Indirection
    - Encapsulation
    - Separation

DFSS attributes to Design Elegance in Architecture

- **Availability**
  - Fault Tolerance
  - Fault Prevention
  - Graceful Degradation
- **Modifiability**
  - Modularity
  - Flexibility, Open, Standardized internal and external interfaces (Loose Coupling)
- **Security**
  - User Access
  - Cross domain security for information transfer
  - Exportability
- **Testability**
  - Manage Input/Output
  - Internal Monitoring (BIT and Instrumentation)
- **Usability**
  - Anticipated User errors
  - User confidence
  - Ease of Use
- **Sustainability**
  - Configurability
  - Composability
  - Maintainability
  - Deployability
- **Performance**
  - Loss of QoS
  - Latency
  - Bandwidth
  - Throughput
- **Interoperability**
  - Common infrastructure of systems/applications
  - Common interfaces
  - Interoperate between machines in common location
  - Interoperate between remote locations

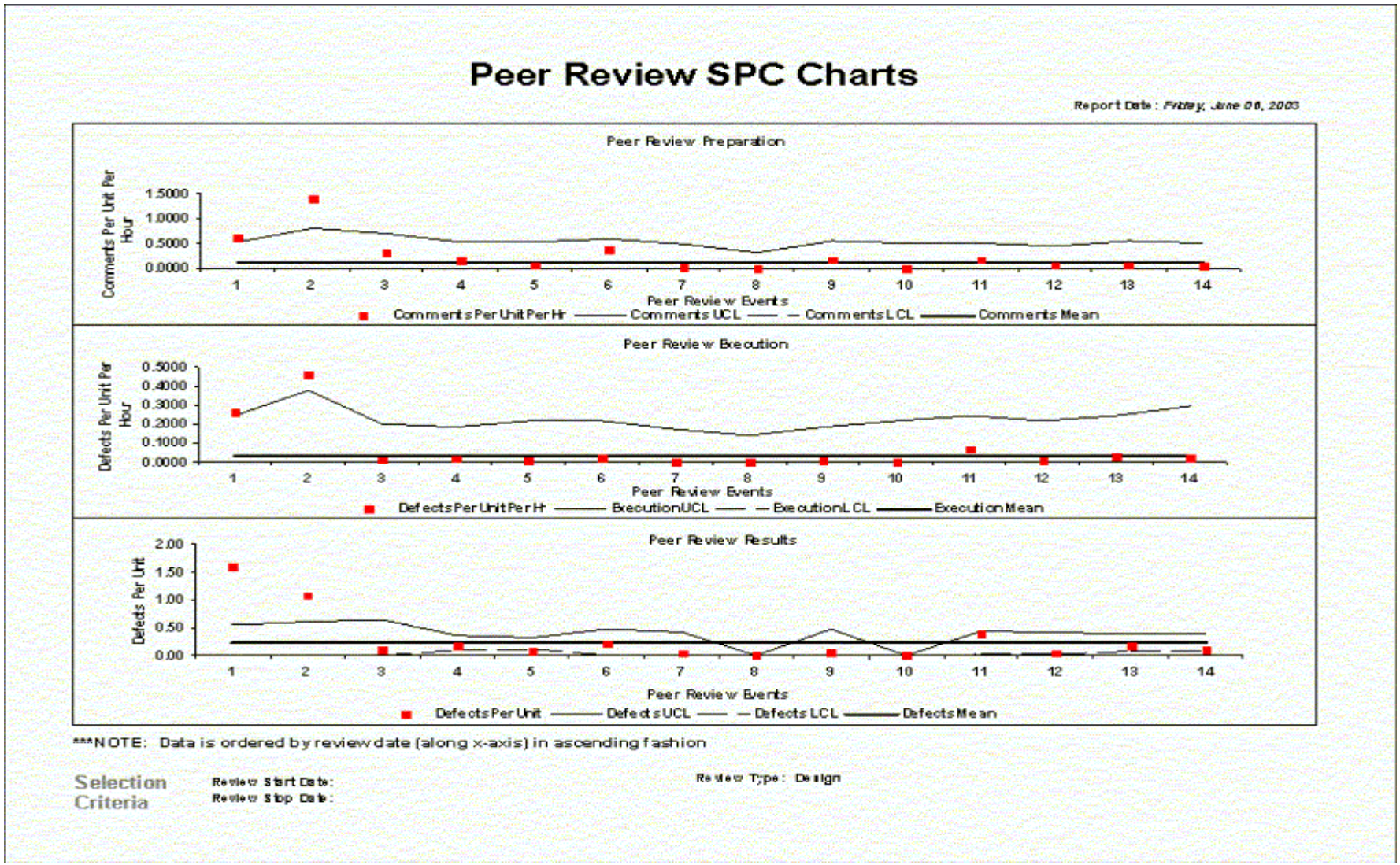
DFSS Throughout the Software Lifecycle: Design and Implementation

- Defect Containment and Defect Density
- A process is considered “capable” when the spread on the bell-shaped (normal) curve is narrower than the tolerance range.
- By definition, a process’ capability is six times the standard deviation (6s) or +/- 3σ around the mean



DFSS Throughout the Software Lifecycle: Design and Implementation

Raytheon



DFSS Throughout the Software Lifecycle: Integration and Test

Raytheon

- Instrumentation
- Built-in Test
- Fault Isolation
- Component Performance Testing
- Integrated Performance Testing
- Test Automation

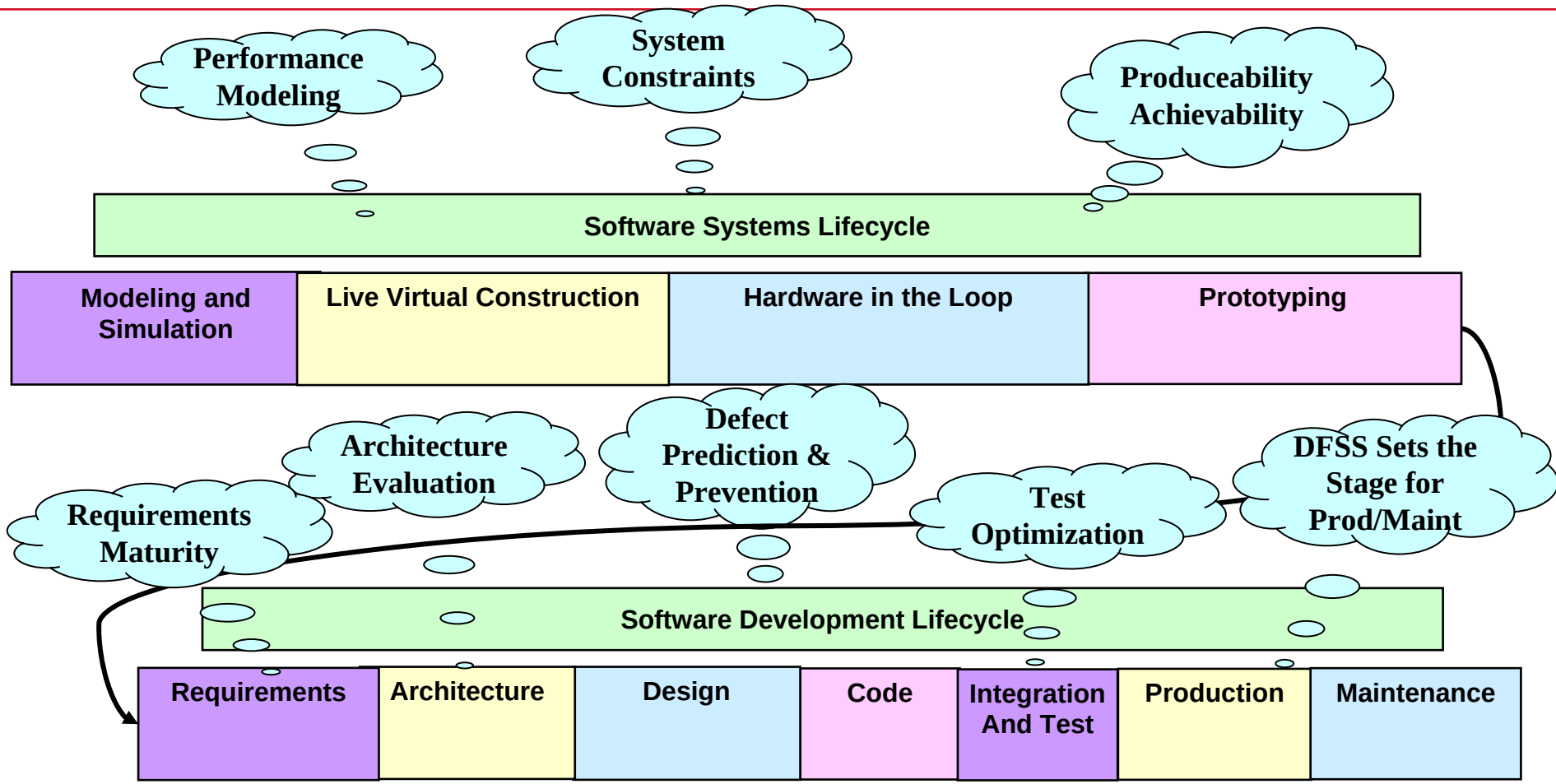
DFSS Throughout the Software Lifecycle: Production and Maintenance

Raytheon

- Maintainability
- Performance Based Logistics
- Upgradeability
- Technology Insertion
- Mean Time Between Failure
- Mean Time Between Replacement

Reaping the benefits of DFSS till the end

Summary



DFSS Provides Opportunities Throughout the Entire Lifecycle

Contact Information

- Jill Brooks
  - Senior Principal Quality Engineer (Six Sigma Expert)
  - jill_a_brooks@raytheon.com
- Sanjeev Venkatesan
  - Software Director, NTx, NCS
  - sanjeev@raytheon.com

References

- **Measuring and Influencing Requirements Engineering Process Quality in Organizations**
  - Bhavani Palyagar
  - *Information and Communication Sciences*
  - *Macquarie University*
- **DFSS in Software**
  - Bob O'Shea
  - *Raytheon*
- **Using Design for Six Sigma to achieve Lean Software Development at Raytheon Missile Systems**
  - Tony Strickland
  - *Raytheon*