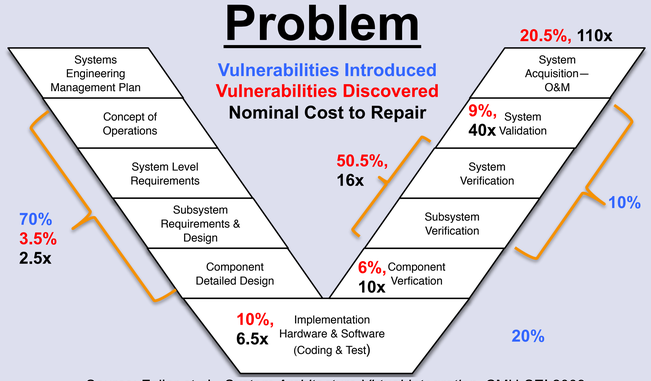
Towards a Process to Forecast Vulnerabilities in Systems of Systems

Problem

- Complex System of Systems (SoS) implementations rarely match SoS designs.
- The impact of subsystem design choices on SoS is difficult to forecast.
- Design tools used across subsystems are stovepiped and cannot be readily integrated in simulation.



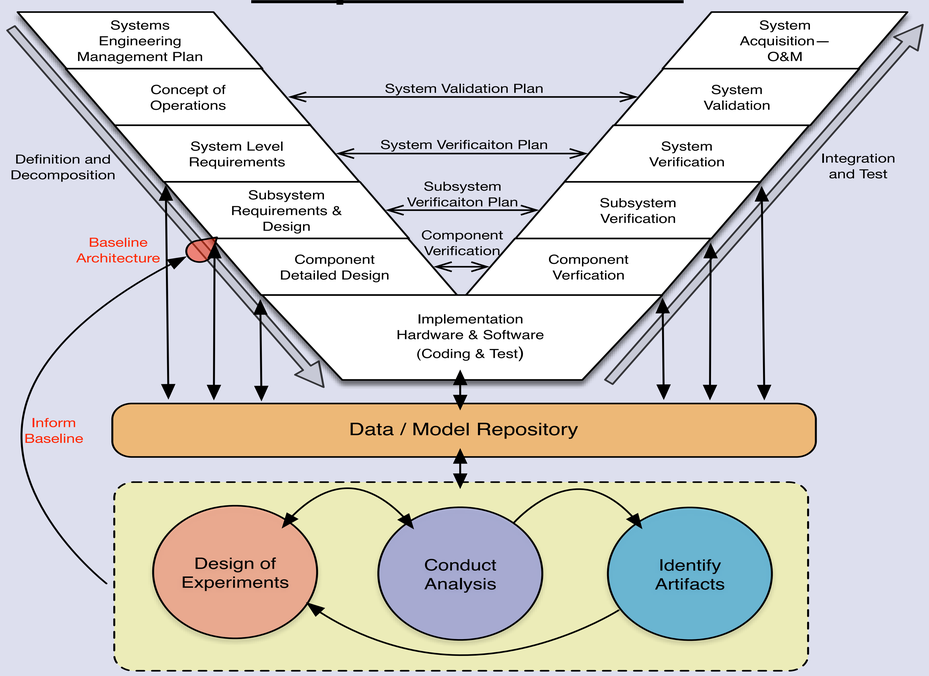
- 70% of vulnerabilities are *introduced* in the architecting phase
- ~80% of vulnerabilities are *discovered* after design lock
- Cost to counter vulnerabilities after design lock is 10-100x the cost of the vulnerable component / subsystem.

Source: Feiler et al., *System Architecture Virtual Integration*, CMU-SEI 2009

Approach

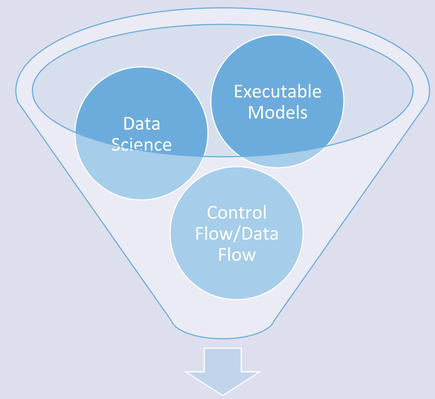
- Integrate design tools that permit end-to-end simulations of SoS “digital twins”.
- Commercial component design tools, such as semiconductor design automation tools (EDA), and system design tools, such as model based systems engineering (MBSE) tools, can be leveraged to create these digital twins.
- Corpus of trace data artifacts from tools in each domain provide opportunity for integration into executable SoS model.
- Subject SoS model to design of experiments, varying design and process parameters to produce statistical distribution of designs and design performance. Experiments include Random Assignment, Bayesian Methodologies, Failure Covariance, supersaturation tests, etc.
- Experimental results are subjected to analytical techniques such as failure propagation, link analysis, uncertainty quantification, etc.

Proposed Process



- Integration in Simulation, requires the *integration of design tools*
- Analysis of design artifacts *reveals vulnerabilities* during the architecting phase.
- Bayesian model that *forecasts vulnerabilities* based on design artifacts and their corresponding location in the overall architecture.
- Heuristics derived from this model will inform *new SoS design principles*.

Path Forward



Vulnerability Forecasting

1. Capture architecture in executable model
   - Perti Nets
   - SySML
   - AADL
2. Identify interface misalignments between components
   - Formal methods
3. Analyze control flow and data flow across the system for unintended behavior
   - Statically abnormality

HCSS

Joseph Natarian, AFRL; Marvin Worst, NASIC; Bruce Howard, WSRI

2016