Verification, Validation and Uncertainty Quantification in CREATE – A Case Study

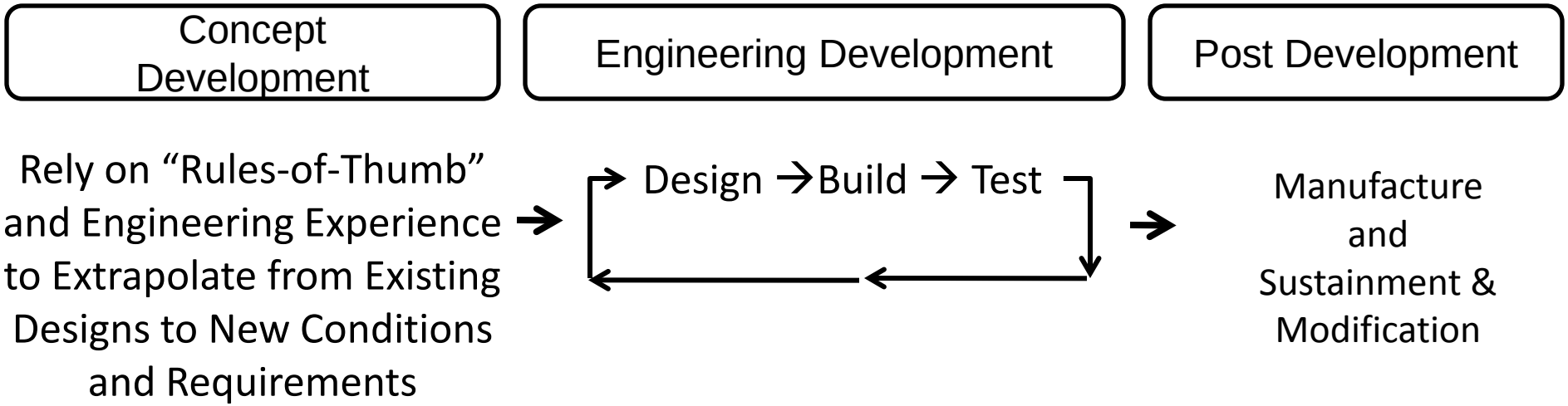
Lawrence G. Votta, Ph.D.
November 2012

Outline

- CREATE Project
- Verification, Validation and Uncertainty Quantification
- Case Study
- Survey Instrument
- Findings
- CREATE 2010/2012 Guidance
- Summary
- References

CREATE Project

- **The Classic Product Development Process**

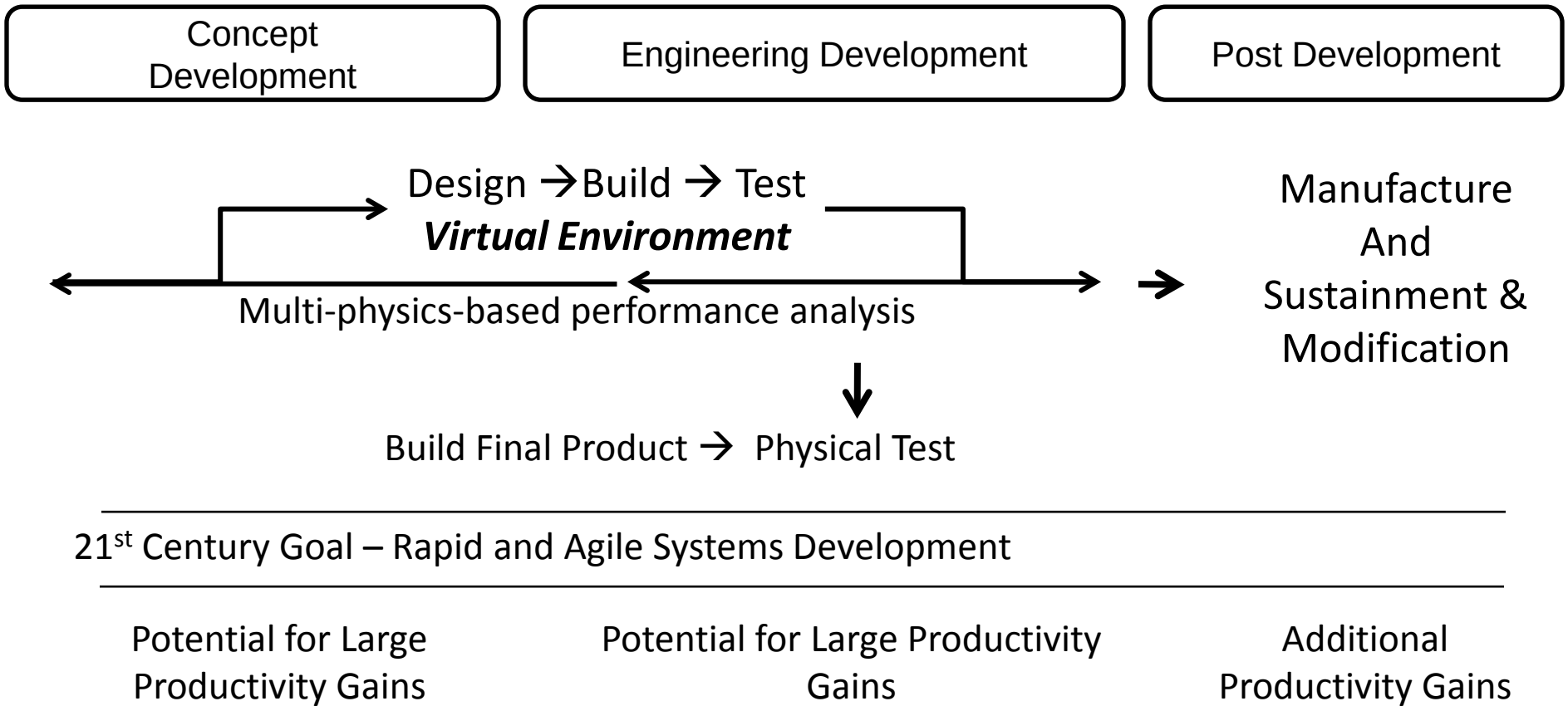


Industrial revolution has focused on mass production

Limited Productivity Gains	Limited Productivity Gains	Enormous Productivity gains

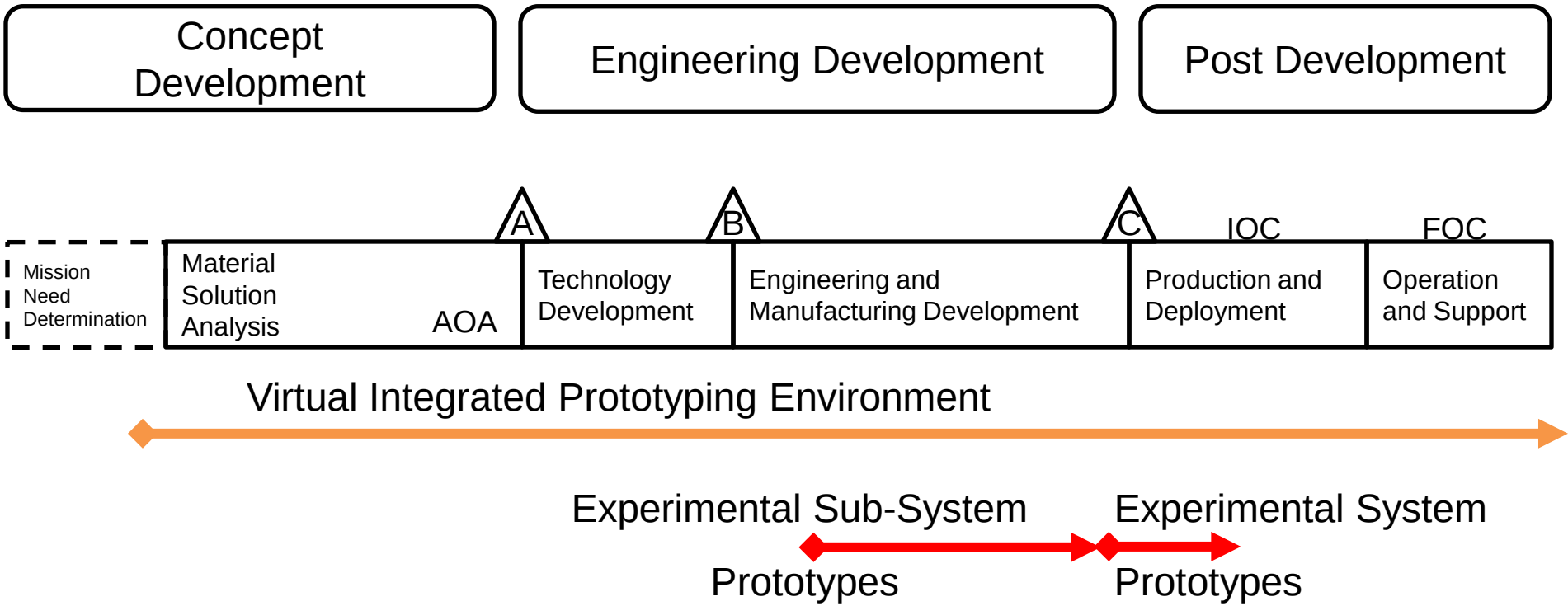
CREATE Project

- **Multi-physics-based performance analysis increases productivity for complex systems**



CREATE Project

- **Performance analysis of virtual prototypes is the key**



Inject physics into design early and all through the process!

CREATE Project

- What is CREATE?

DoD program to develop and deploy multi-physics-based software for engineering design and analysis of:

Air Vehicles (AV)

- Aerodynamics, structural mechanics, propulsion, control, …

Ships

- Hydrodynamics, structural mechanics, shock, …

CREATE tools support all stages of acquisition from rapid early-stage design to full life-cycle sustainment

Radio Frequency (RF) Antennas

- RF Antenna electromagnetics and integration with platforms

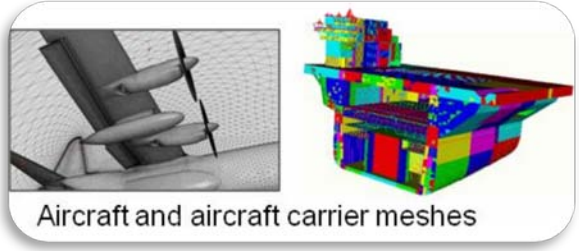
Mesh and Geometry (MG) generation

- Rapid generation of mesh and geometry representations needed by analysis

Design concept

Seakeeping and resistance

Shock vulnerability

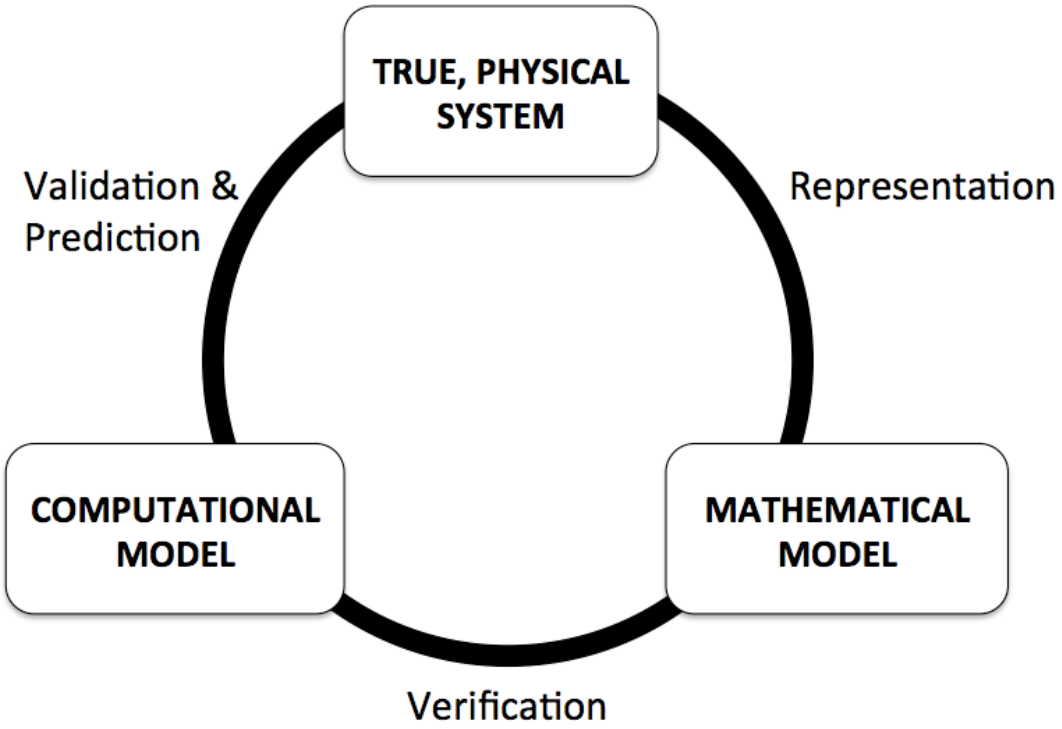


Aircraft and aircraft carrier meshes

Military platforms with antennas

VVUQ

The Modeling and Simulation Ecosystem



Verification, validation, and prediction as they relate to the true, physical system, the mathematical model, and the computational model. (Adapted from American Institute for Aeronautics and Astronautics. 1998.)

VVUQ

Important Terms and Concepts - 1

- **Quantity of Interest (QOI)** – are the output(s)/result(s) of computational models and are used in the engineering and study of modeled systems
- **Verification** – how accurately a computer program ("code") correctly solves the equations of the mathematical model
- **Validation** – the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model
- **Uncertainty Quantification (UQ)** – quantifying uncertainties associated with model calculations of true, physical QOIs

VVUQ

Important Terms and Concepts - 2

- **Community of Interest –** A community of domain experts, computational users and modelers that maintain detailed domain knowledge, shared validation test suites, and benchmarks for problems of interest
- **Intended Use –** A computational model cannot “be proven” correct. Usually a community of interest defines problems in a domain and sets an acceptable level of testing to insure that the computational model is validated. An intended use is defined by the set of problems.

Case Study

- **Goals**
  - Assess the compliance level of the 10 CREATE projects with 2010 practice guidance
  - Assess VVUQ practice
- **Experimental Design**
  - All 10 CREATE software projects (exhaustive sample)
  - Survey instrument
  - Initial self assessment
  - Capture results with software engineering team interviews
  - Resolve discrepancies and clarifications in follow-ups
- **Analysis**
  - Summary Compliance Matrix
  - Comparative analysis with NRC report

Survey Instrument

CREATE Software Engineering Principles

- **Development Team**
  - Lean (<10 person) teams
  - Development transparency and visibility across CREATE projects
- **Customer Focus**
  - Stakeholder-driven requirements with use case
  - Pilots to converge on customer needs
  - Frequent reporting to stakeholders
- **Technical Maturity**
  - Use proven technologies to satisfy customer-defined use cases
  - VVUQ in alignment with NRC recommendations for scientific code
- **Development Methods**
  - Milestone-driven workflow with flexible execution and annual releases
  - Vertical configuration management and configuration control boards (CCBs)
  - Code delivery requires software and test
  - Documented code with user's manuals, tutorials, and user forums
- **Requirements Definition**
  - Prototypes solidify difficult-to-specify, or possibly ambiguous requirements

Findings

Summary Compliance Matrix - 1

PROJECT		A	B	C	D	E	F	G	H	I	K
Code Mgt											
	Config Mgt Tools										
	Auto Build Tools										
	Continuous Integration										
	Debugging										
	Bug Tracking										
	Cross Compile										
Verification											
	Documented Plan										
	Verifi before validate										
	Coverage measured										
	Coverage numb drives										
	Exter components test										
	ST tied to use cases										
	Auto compiler feat test										
	Tests req' at Check-in										
	Check-in Test Coverage										
	Convergence Studies										
	Regression Tests										

LEGEND

- Yes
- Some Issues
- Needs Work
- Not Applicable
- Investigating

Findings

Summary Compliance Matrix - 2

PROJECT	A	B	C	D	E	F	G	H	I	J
Validation										
Documented Plan										
Community Prov Tests										
Independent Vetting										
Embed Empirical Models										
Doc Simple Assumpts										
Hierarchical Validation										
Doc Domain of Applicabi										
Tesst Coverage Measure										
Use Case QOI Valid										
QOI measured by expt										
QOI agree metrics										
Access to valid data										
Measure Error Bars										
Uncertainty Quantification										
Doc Key Assumptions										
Assess sensitivity QOIs										
Community stds various QOIs										
Incomple valid data sets										

LEGEND

- Yes
- Some Issues
- Needs Work
- Not Applicable
- Investigating

Findings

Case Study Results

- Adherence to 2010 guidance was excellent, with many factors playing a role in the quality of implementation of the guidance
- We compared the 2010 guidance to the 2012 NRC report and found that the CREATE 2010 V&V guidance is in very good agreement except for UQ
  - Expanded/added one more practice to leverage code organization for hierarchical verification
- We have found it necessary to add an explicit UQ section to CREATE's guidance and to start adopting the guidance as soon as development opportunities present themselves

Findings

CREATE 2010/2012 VVUQ Guidance

Verification:

Practice 1: Document the domain and range of targeted applicability of the code

Practice 2: Verify the code prior to validation

Practice 3: Generally, verify the code as much as practical, and document the verification

Practice 4: Specifically, conduct unit, integration, system and regression tests and document the results

Practice 5: Compile the code with compilers from several different vendors

Practice 6: Use as many types of verification tests as are feasible.

Practice 7: Develop a verification test plan

Practice 8: ***Design code with hierarchical code verification in mind***

Findings

CREATE 2010/2012 VVUQ Guidance

Validation:

Practice 9: Validate for the full range of the targeted use of the code

Practice 10: Develop archival database for validation

Practice 11: Validation should be focused on the behavior and accuracy of QOIs associated with use cases

Practice 12: Develop validation project plans, review them with independent experts and users, and execute them

Practice 13: ***Formally assess the V&V status and progress***

Findings

CREATE 2010/2012 VVUQ Guidance

Uncertainty Quantification:

Practice 14: All QOIs important to the intended use (and described in use cases) are defined in the computational tool's manual

Practice 15: For each QOI, the computational tool must support investigations of sensitivity to input values and computational techniques

Practice 16: CREATE computational tools should support models of uncertainties and should provide feedback to the user about the important sources of variance for each QOI

Practice 17: Periodic assessment of the tool's capability to predict QOIs and the variance of QOIs in the areas of intended use and feedback provided for verification and validation improvement

Summary

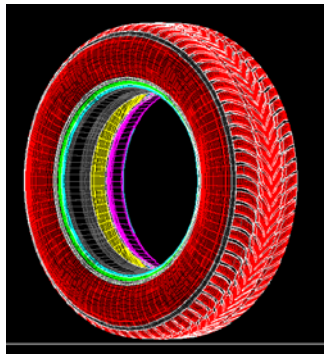
- This presentation/report implements Practice 13 for this CREATE development cycle
- NRC report shows the leverage of using structure (hierarchy) in testing strategy and implementation – added one practice to Verification
- NRC report highlights importance of Uncertainty Quantification, added 5 practices

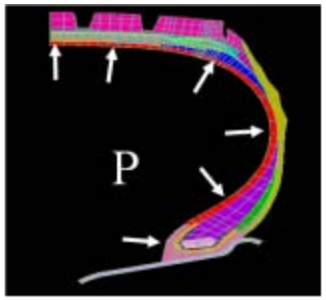
References

- National Sciences Academy Report, "Assessing the Reliability of Complex Models: Mathematical and Statistical Foundations of Verification, Validation, and Uncertainty Quantification (VVUQ)“, ISBN 978-0-309-25634-6, March 2012
- Roache, P. 1998. Verification and Validation in Computational Science and Engineering. Socorro, NM: Hermosa Publishers

Backup Slides

This Paradigm is Proving Successful for Some Companies





Inflation and Seating

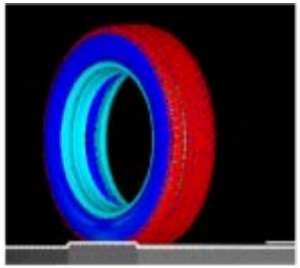
Hydroplaning Tests

Flight Radial

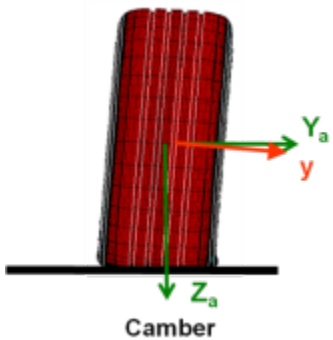
Requirements → Design and Mesh Virtual Product → Analyze and Test Virtual Product → Build and Test Physical Product → Market

Design iterations

- Reduced time to market from 3 years to less than 1 year
- Increased new products delivery from 1 every 3 years to 5 per year



Running Over Obstacles



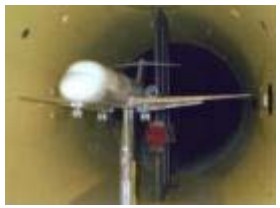
Camber

Fortera TripleTred

Miller, L.K., "Simulation-Based Engineering for Industrial Competitive Advantage." *Computing in Science and Engineering*, 12, pp. 14–21, 2010.

Present Product Development Process
Iterated Design → Build → Test Cycles



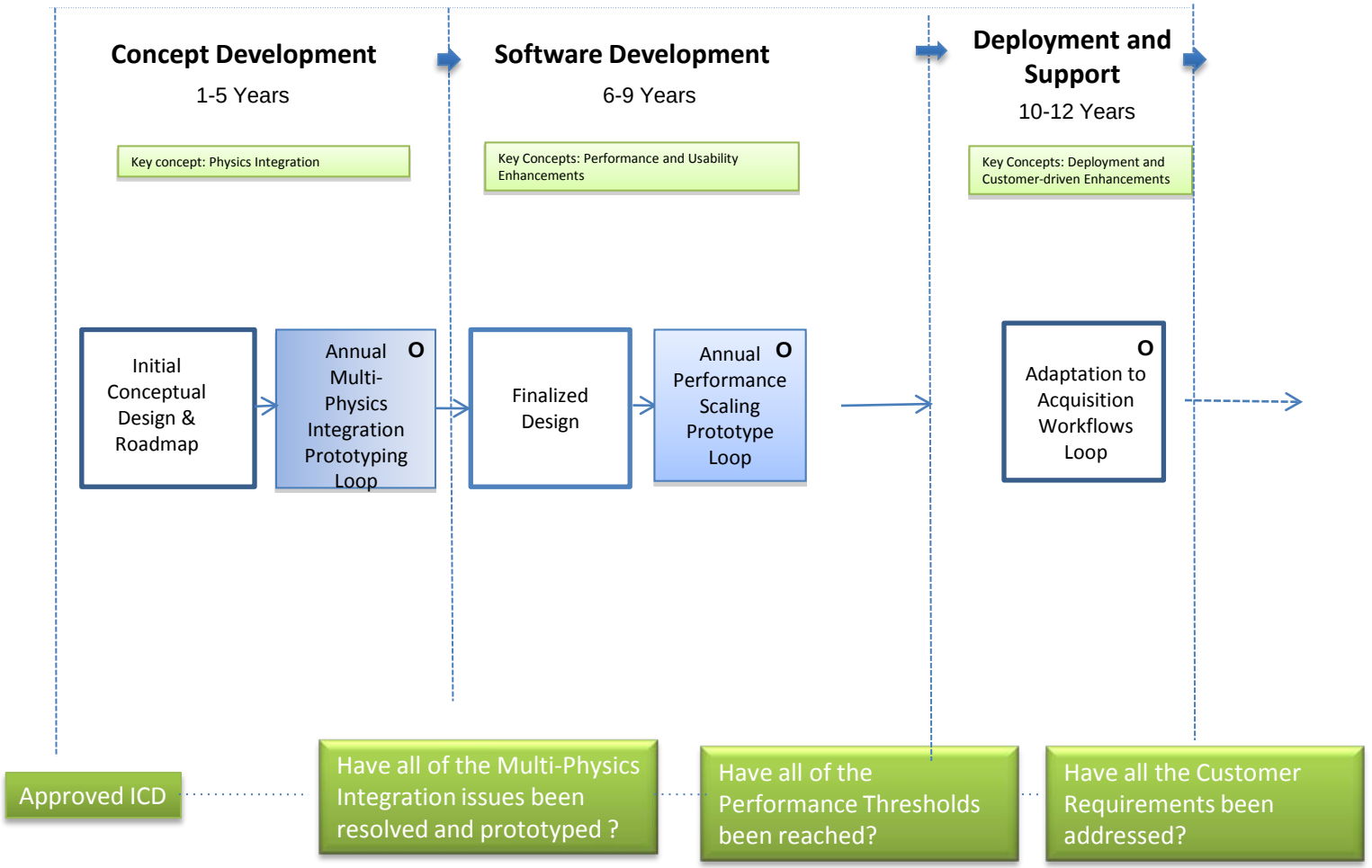




Requirements → Design → Build Physical Product → Test Physical Product → Manufacture and Market

(Many) Design iterations

- **Long time to market**
  - Requires many lengthy and expensive design/build/test iteration loops
- **Process converges slowly, if at all**
  - Process is rigid, not responsive to new requirements
  - Design flaws discovered late in process leading to rework
  - Systems Integration happens late in process

CREATE Development Rhythm



CREATE Canonical Milestones (from Annual Software Engineering Plan)

A. *Baseline Schedule*: Includes design question milestones, if applicable. This schedule should conform to the guidance provided in the Guidance for Product Development Measurement. This schedule must include the software development milestones listed below for each version of the product under active development or support (illustrated in Figure 2) during the fiscal year of the plan:

   a. Completion of Initial Design Review (set specifications for design of release).
   b. Completion of Final Design Review.
   c. Creation of a new development branch of the program library for the annual development cycle (alternatively, spinoff of production release branch with all development in the trunk)
   d. Freeze of the product development branch to new product features.
   e. End of alpha testing.
   f. Completion of beta testing.
   g. Completion of readiness assessment for production release (including version requirements reconciliation).
   h. Completion of production release.
   i. End of life for version.

CMMI - Scrum Mapping: Some examples:

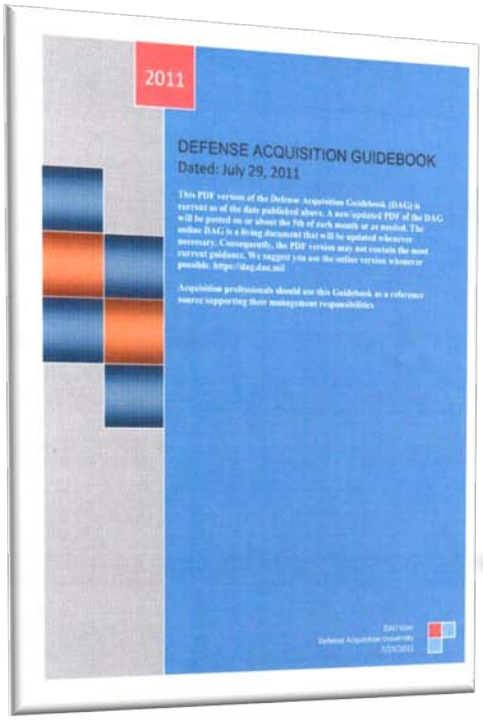
Requirements	CMMI Practice	Scrum Practice
SP 1.1	Develop understand on meaning	Review Backlog with Product owner
SP 1.2	Obtain participant commitment	Sprint planning sessions that seek team commitment
SP 1.3	Manage requirements changes	Add stories to product backlog
SP 1.5	Identify inconsistencies	Daily Stand-up meetings Sprint planning sessions Burn down charts

Project Planning

SP 1.1	Establish top-level WBS	Scrum backlog expanded into tasks
SP 1.2	Estimate work content of tasks	Story points (used to estimate size of stories)
SP 1.3	Define life-cycle phases	The Scrum Process itself
SP 2.1	Establish budget and schedule	Scrum estimates (in Ideal Time) Estimates of work in each release Sprint backlog
SP 2.6	Plan involvement of stakeholders	Scrum process roles (Scrum master, Product Owner)

Our Customer's Expectations

Chapter 4: Systems Engineering



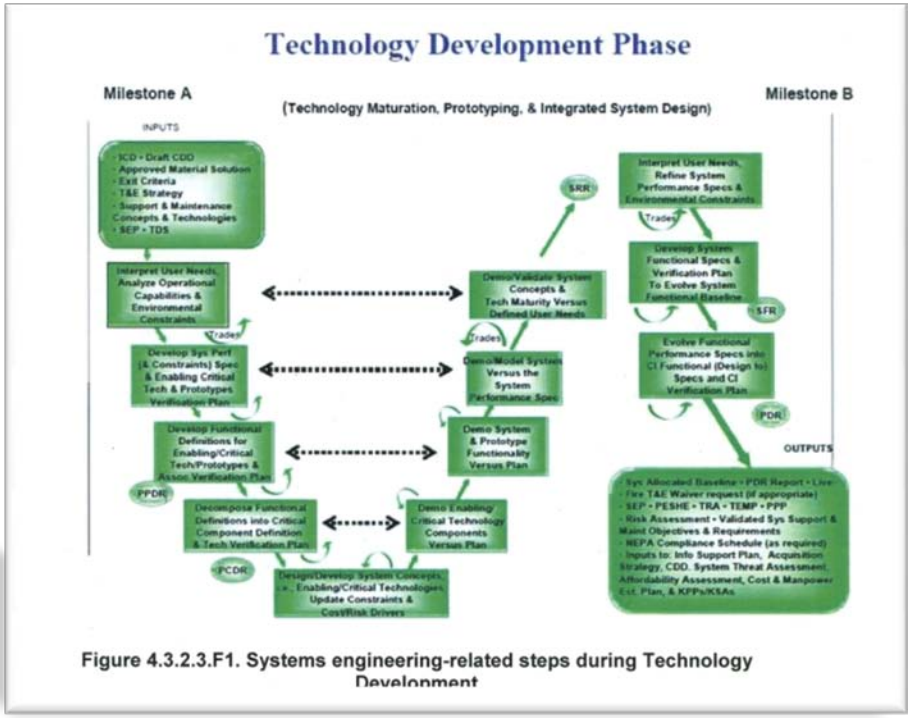
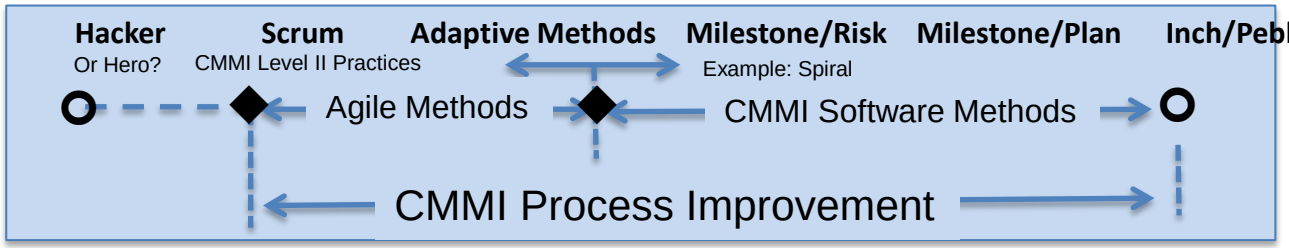


Figure 4.3.2.3.F1. Systems engineering-related steps during Technology Development

Defense Acquisition Guidebook, https//: dag.dau.mil

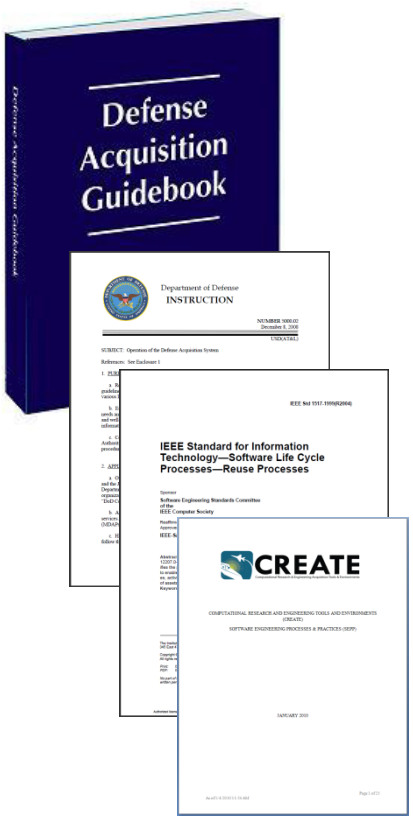
Our Approach

- **Deploy light-weight software development processes with the best features of Milestone-based methods**



after Boehm, "Getting Ready for Agile Methods with Care," IEEE Software, 2002

Product Test Plan References



- Defense Acquisition Guidebook
  CH. 9 -- Test and Evaluation (T&E)
- DoDD 5000.1, DoDI 5000.2, DoD 5000.2-R (Defense Acquisition Directive, Instruction, and Regulation)
- IEEE/EIA 12207.0, “Standard for Information Technology – Software Life Cycle Processes” *Converted to DoD standard on 27 May 1998*
- CREATE Software Engineer Process Guidance Jan 7, 2010
- Others: A Guide to V&V of CREATE Codes, etc…

Growth in Computers Since 1945 Provides Unparalleled Problem Solving Power

- The 10^{15} increase in computer power since 1945 can enable us to develop and deploy codes during the next decade that are much more powerful than past tools:
  - Utilize accurate solution methods
  - Include all the effects we know to be important
  - Model a complete system
  - Complete parameter surveys in hours rather than days to weeks to months
- In ~10 years, workstations will be as powerful as today's supercomputers
- Greatest opportunities for 2020 (and 2010) include large-scale codes that integrate many multi-scale effects to model a complete system