Software Engineering in CREATE – Lessons Deployed

The Application of Engineering Rigor to Software Development—
Systems Engineering for Software

DOD HPC MODERNIZATION PROGRAM

Richard P Kendall, Ph.D. (Presented by Dr. Saikat Dey)
November 2012

Gov't Software: A Legacy of Risk Management Failure!

SEER by GALORATH

DAN GALORATH ON ESTIMATING

RSS Feed Comments

- Home
- About Galorath Incorporated
- SEER User Conferences

GO

Software Project Failure Costs Billions Estimation & Planning Can Help

June 7, 2008 · Filed Under Project Management - 18 Comment(s)

There are so many studies attempting to quantify the cost of software failures. The percentages but they generally agree that the number is at least 50 to 80 billion dol

Standish Chaos Reports: Standish is probably the most referenced. They defin budget, of cost, and with expected functionality. There are several updates to the The 2004 report shows:

- Successful Projects: 29%
- Canceled projects cost $55 Billion Annually?
- Challenged Projects: 53%
- Failed Projects: 18%

Standish Findings By Year Updated for 2009

	1994	1996	1998	2000	2002	2004	2009
Succeeded	16%	27%	26%	28%	34%	29%	32%
Failed	31%	40%	28%	23%	15%	18%	24%
Challenged	53%	33%	46%	49%	51%	53%	44%

Most projects cost more than they return, Mercer Consulting: When the many as 80% of technology projects actually cost more than they return. It is costs are always underestimated and the benefits are always overestimated. I

Oxford University Regarding IT Project Success (Saur & Cuthbertson, 2

- Successful: 16%
- Challenged: 74%

Mexico Considers Legalizing Drugs » « Foodstamp Use Breaks Reco

Billions Wasted on DoD Software

The victors in battles are those who create, modify and deploy ideas faster and more nimbly than opponents. Regrettably, limiting the U.S. military's access to ideas risks failure.

For years, the U.S. military has been losing an asymmetric battle that involves not improvised explosive devices, bullets al-Qaida, but instead swarms of defense industry contractors seizing control of taxpayer-funded ideas because governme policy and regulations were engineered to buy iron and steel, not to deploy a software-based military.

Much like the battles in Iraq and Afghanistan, the rapid and continual evolution of technology demands that the military accelerate just as rapidly, and the only way is to manage the ideas it has funded.

A common theme since 9/11 is that the U.S. government lacks imagination. We have not misplaced our imagination; we are simply unable to deploy new ideas as effectively or as quickly as we could. This loss of agility stands in stark contras to private industry, foreign governments and nonstate actors, who are adopting and deploying software technologies once exclusively in the military domain.

For instance, China deploys advanced electronic warfare technologies, Iran builds unmanned aircraft, al-Qaida evolves explosive devices, and private companies like FedEx and eTrade create complex, redundant and failsafe command-and-control systems.

Software is the fabric that enables planning, weapons and logistics systems to function. It might be the only infinitely renewable military resource. New software builds on the raw material of previous software, evolving capabilities. Software is pervasive, from ground sensors to satellites; it is the final expression of a military idea transformed into human readabl source code and deployed to a battlefield.

Wasted Billions

The Department of Defense spends tens of billions of dollars annually creating software that is rarely reused and difficult t adapt to new threats. Instead, much of this software is allowed to become the property of defense companies, resulting in DoD repeatedly funding the same solutions or, worse, repaying to use previously created software.

The lack of a coherent set of policies and regulations for the DoD's intellectual property has eroded the U.S. military competitive advantage, leading to compromised missions and lost lives. Improvised explosive device countermeasure systems can't be upgraded rapidly without replacing entire systems; personnel position systems can't update in real time; billions are wasted on software radios that don't interoperate.

The byzantine rules governing the military's intellectual property portfolio use an antiquated rights structure where the contractor always retains copyright, and therefore effective monopoly, control over taxpayer-funded software ideas. By contrast, commercial industry ruthlessly exercises control over its own software ideas.

The U.S. government has legislated a belief that the defense industry will do right by the military. However, the defense industry will, understandably, do what is best for its shareholders: maximize profit.

Monopolies via copyright ultimately increase costs and decrease adaptability and agility in military software. Examples include the General Atomics Predator and the recently canceled Future Combat Systems, where only one company can control these platforms and manipulate the software. Imagine if only the manufacturer of a rifle were allowed to clean, fix, modify or upgrade that rifle. This is where the military finds itself: one contractor with a monopoly on the knowledge of a military software system.

A first step would be to require all taxpayer-funded software ideas to be licensed with an open source software copyright. An open source license would define the rights, roles and responsibilities for the military and defense industry and simplify how military software ideas can be shared. To keep the U.S. military ahead of its adversaries, the DoD and defense industry must end this dysfunctional partnership of nonsharing.

Defining a modern software intellectual property regime would broaden the defense industrial base by enabling industry access to defense knowledge, thereby increasing competition and eventually lowering costs. Over time, DoD would evolv common software architectures and industrywide baselines to increase the adaptability, agility and – most important – capacity to meet new dynamic threats.

Who Killed the Virtual Case File? - IEEE Spectrum

COMPUTING / SOFTWARE

FEATURE

Who Killed the Virtual Case File?

How the FBI blew more than $100 million on case-management software it w

By HARRY GOLDSTEIN / SEPTEMBER 2005

n the early 1990s, Russian mobsters partnered with Italian Mafia families in Newark, N.J., t deral and New Jersey state gasoline and diesel taxes. Special Agent Larry Depew set up eration under the direction of Robert J. Chiaradio, a supervisor at the Federal Bureau of In C., headquarters.

ew collected reams of evidence from wiretaps, interviews, and financial transactions over years. Unfortunately, the FBI couldn't provide him with a database program that would help mation, so Depew wrote one himself. He used it to trace relationships between telephone c illance, and interviews, but he could not import information from other investigations that m So it wasn't until Depew mentioned the name of a suspect to a colleague that he obtained a had been holding since 1989.

opened it up, it was a treasure trove of information about who's involved in the conspiracy, family, the Genovese family, and the Russian components. It listed percentages of who g ere supposed to pay, the number of gallons. It became a central piece of evidence," Depe w at the FBI's New Jersey Regional Computer Forensic Laboratory, in Hamilton, where he just picked up the phone and called that agent, I never would have gotten it."

ter, Depew's need to share information combined with his do-it-yourself database skills ar rvisor, Chiaradio, would land him a job managing his first IT project–the FBI's Virtual Case

ointment to the FBI's VCF team was an auspicious start to what would become the most hi e in history. The VCF was supposed to automate the FBI's paper-based work environment e analysts to share vital investigative information, and replace the obsolete Automated Ca Instead, the FBI claims, the VCF's contractor, Science Applications International Corp. (SAI d 700 000 lines of code so bug-ridden and functionally off target that this past April, the bur 70 million project, including $105 million worth of unusable code. However, various gover orts show that the FBI—lacking IT management and technical expertise—shares the blame

-page audit, released in 2005, Glenn A. Fine, the U.S. Department of Justice's inspector g tors that contributed to the VCF's failure. Among them: poorly defined and slowly evolving y ambitious schedules; and the lack of a plan to guide hardware purchases, network deplo pment for the bureau.

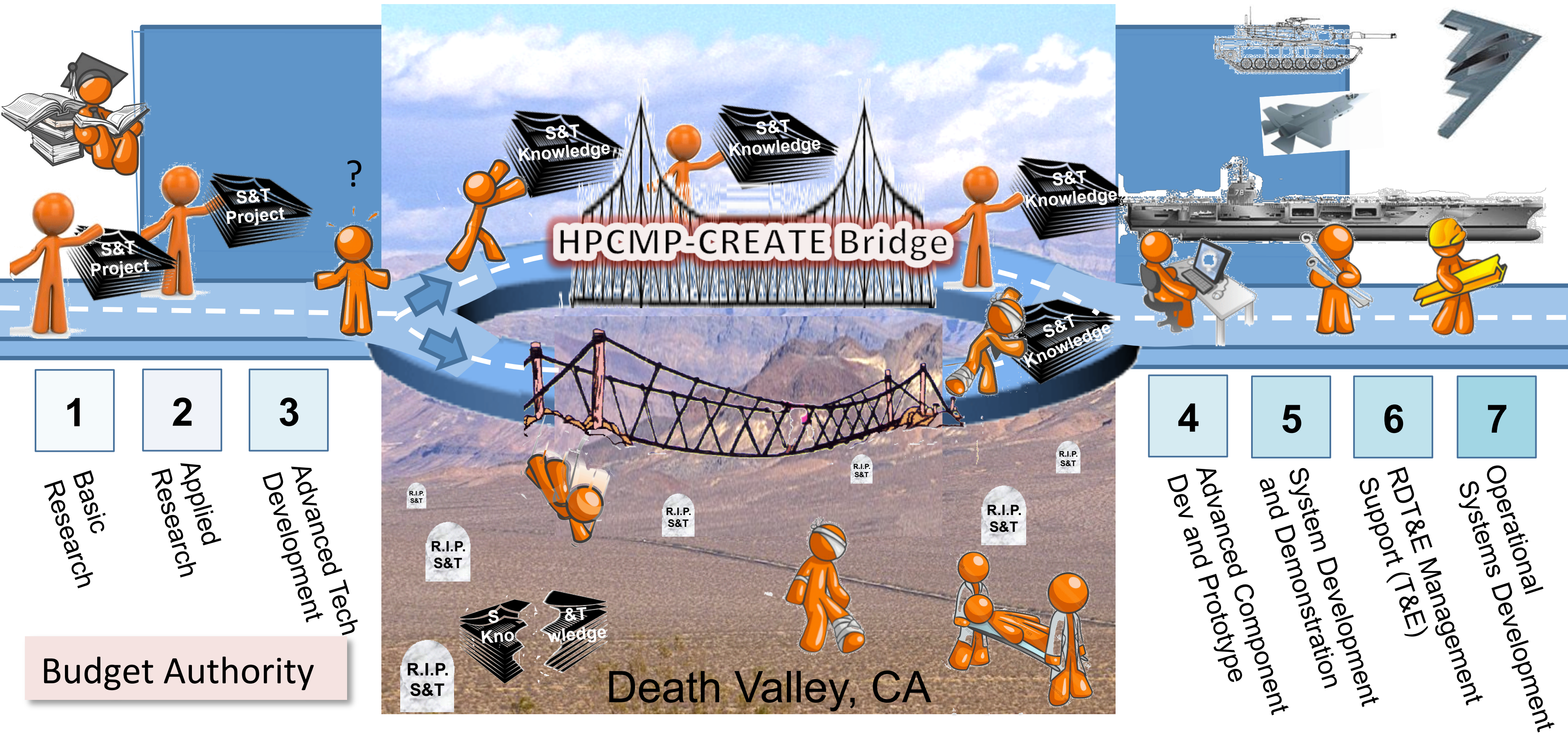
our years after terrorists crashed jetliners into the World Trade Center and the Pentagon, th zed for not "connecting the dots" in time to prevent the attacks, still did not have the softwar ny new dots that might come along. And won't for years to come.

Case Support system—which some agents have avoided using—is cumbersome, ineffici ities, and does not manage, link, research, analyze, and share information as effectively or wrote. "[T]he continued delays in developing the VCF affect the FBI's ability to carry out its

er it officially ended the VCF project, the FBI announced that it would buy off-the-shelf cost to be deployed in phases over the next four years. Until those systems are up and

-killed-the-virtual-case-file

HPCMP Computers, Networks and CREATE Tools Bridge the Gap over the "Valley of Death"



Distribution C. Please see Page 1 for additional information

Compounding Risk Factors

The Four CREATE Program Complexities:

1. Complex Physics (Integrated Multi-Scale, Multi-Physics)
2. Complex Computing (networks, security, architectures)
3. Complex Development Organizations (Distributed)
4. Complex Customers (Multi-Service, Multi-Community)

Managing Risks in CREATE

Software Engineering provides the framework for managing risk

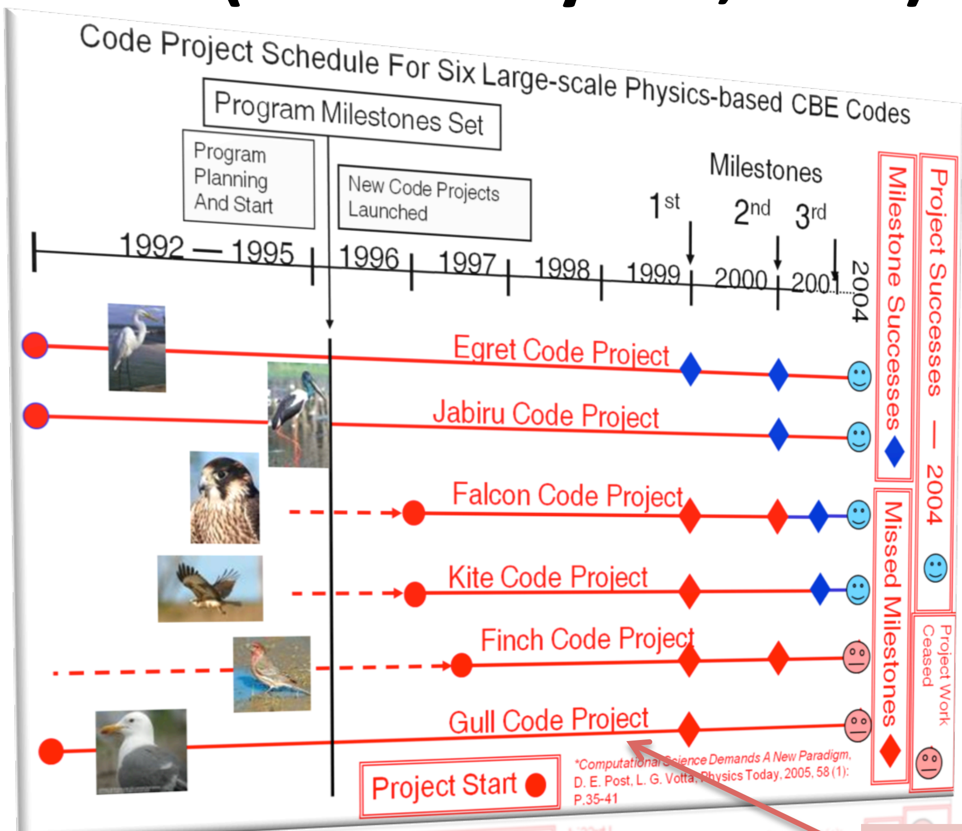
- Based on experience from DoD, DOE, Industry, Academia case studies
- Adapts best practices to physics-based software:

•Program Management
•Requirements Management
•Configuration Management
•Quality Assurance
•Verification, Validation and Uncertainty Quantification

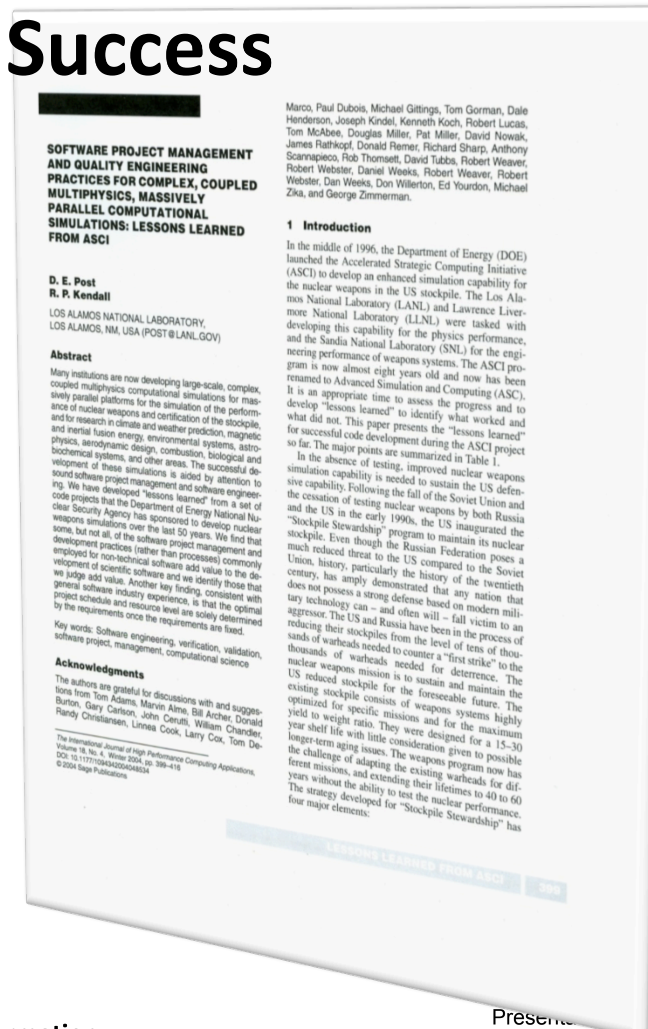
after CMMI (SEI)

An Example Similar to CREATE

- ASCI (Multi-Physics, HPC) < 50% Success



CREATE Scale



[redacted]

SOFTWARE PROJECT MANAGEMENT AND QUALITY ENGINEERING PRACTICES FOR COMPLEX, COUPLED MULTIPHYSICS, MASSIVELY PARALLEL COMPUTATIONAL SIMULATIONS: LESSONS LEARNED FROM ASCI

D. E. Post
R. P. Kendall

LOS ALAMOS NATIONAL LABORATORY,
LOS ALAMOS, NM, USA (POST@LANL.GOV)

Abstract

Many institutions are now developing large-scale, complex, coupled multiphysics computational simulations for massively parallel platforms for the simulation of the performance of nuclear weapons and certification of the stockpile, and for research in climate and weather prediction, magnetic and inertial fusion energy, environmental systems, astrophysics, aerodynamic design, combustion, biological and biochemical systems, and other areas. The successful development of these simulations is aided by attention to sound software project management and software engineering. We have developed "lessons learned" from a set of code projects that the Department of Energy National Nuclear Security Agency has sponsored to develop nuclear weapons simulations over the last 50 years. We find that some, but not all, of the software project management and development practices (rather than processes) commonly employed for non-technical software add value to the development of scientific software and we identify those that we judge add value. Another key finding, consistent with general software industry experience, is that the optimal project schedule and resource level are solely determined by the requirements once the requirements are fixed.

Key words: Software engineering, verification, validation, software project, management, computational science

Acknowledgments

The authors are grateful for discussions with and suggestions from Tom Adams, Marvin Alme, Bill Archer, Donald Burton, Gary Carlson, John Cerutti, William Chandler, Randy Christiansen, Linnea Cook, Larry Cox, Tom DeMarco, Paul Dubois, Michael Gittings, Tom Gorman, Dale Henderson, Joseph Kindel, Kenneth Koch, Robert Lucas, Tom McAbee, Douglas Miller, Pat Miller, David Nowak, James Rathkopf, Donald Remer, Richard Sharp, Anthony Scannapieco, Rob Thomsett, David Tubbs, Robert Weaver, Robert Webster, Daniel Weeks, Robert Weaver, Robert Webster, Dan Weeks, Don Willerton, Ed Yourdon, Michael Zika, and George Zimmerman.

The International Journal of High Performance Computing Applications, Volume 18, No. 4, Winter 2004, pp. 399–416
DOI: 10.1177/1094342004048534

1 Introduction

In the middle of 1996, the Department of Energy (DOE) launched the Accelerated Strategic Computing Initiative (ASCI) to develop an enhanced simulation capability for the nuclear weapons in the US stockpile. The Los Alamos National Laboratory (LANL) and Lawrence Livermore National Laboratory (LLNL) were tasked with developing this capability for the physics performance, and the Sandia National Laboratory (SNL) for the engineering performance of weapons systems. The ASCI program is now almost eight years old and now has been renamed to Advanced Simulation and Computing (ASC). It is an appropriate time to assess the progress and to develop "lessons learned" to identify what worked and what did not. This paper presents the "lessons learned" for successful code development during the ASCI project so far. The major points are summarized in Table 1.

In the absence of testing, improved nuclear weapons simulation capability is needed to sustain the US defensive capability. Following the fall of the Soviet Union and the cessation of testing nuclear weapons by both Russia and the US in the early 1990s, the US inaugurated the "Stockpile Stewardship" program to maintain its nuclear stockpile. Even though the Russian Federation poses a much reduced threat to the US compared to the Soviet Union, history, particularly the history of the twentieth century, has amply demonstrated that any nation that does not possess a strong defense based on modern military technology can – and often will – fall victim to an aggressor. The US and Russia have been in the process of reducing their stockpiles from the level of tens of thousands of warheads needed to counter a "first strike" to the thousands of warheads needed for deterrence. The nuclear weapons mission is to sustain and maintain the US reduced stockpile for the foreseeable future. The existing stockpile consists of weapons systems highly optimized for specific missions and for the maximum yield to weight ratio. They were designed for a 15–30 year shelf life with little consideration given to possible longer-term aging issues. The weapons program now has the challenge of adapting the existing warheads for different missions, and extending their lifetimes to 40 to 60 years without the ability to test the nuclear performance. The strategy developed for "Stockpile Stewardship" has four major elements:

Risk Factors: Why Software Projects fail.

SEI Risk Taxonomy (2007)

A. Development Cycle
1. **Requirements**
   a. Evolvability
   b. Completeness
   c. Understandability (clarity)
   d. Fidelity
   e. Technical Feasibility
   f. Execution Performance
2. **Design**
   a. Difficulty
   b. Factorizibility (Modularity)

B. Development Environment
1. **Development Processes**
   a. Repeatability (formality)
   b. Suitability
   c. Control of Process
   d. Familiarity
   e. Environmental Change Control
2. **Development System**
   a. Hardware

C. Program Constraints
1. **Resources**
   a. Schedule
   b. Staff
   c. Budget
   d. Facilities
   e. Management Commitment
2. **Contract**
   a. Type
   b. Restrictions
   c. Dependencies
3. **Program Interfaces**

ors
actor
gt.

Sources of Risk (*)

1. Project complexity
2. Project goals
3. Acquiring needed resources and skills
4. Requirements synthesis
5. Reporting
6. Communication among developers, customers and users
7. Technology maturity
8. Development practices
9. Project management, especially Planning & Execution
10. Stakeholder/Customer politics

(*) after IEEE Spectrum On-line, "Why Software Fails," 2010

CREATE Core Software Engineering Practices

Development Team

1. Lean (<10), close-knit development teams led by technical experts. **[3,8,10]**

2. Emphasis on transparency in development across CREATE projects. **[6]**

Customer Focus

3. Stakeholder-driven requirements through Boards of Directors comprised of stakeholder and user representatives. **[2,4,6,10]**

4. Pilots to solicit customer reaction and input to feature and attribute implementations. **[4]**

5. Frequent reporting to stakeholders. **[6]**

CREATE Core Software Engineering Practices

Technical Maturity

6. Reliance on proven technologies to satisfy customer-defined use cases. **[1,2,7]**

7. VVUQ in alignment with NRC recommendations for scientific code. **[10]**

Development Methods

8. Milestone-driven workflow management with flexible workflow execution and annual releases. **[8,9]**

9. Configuration management, including configuration control boards (CCBs), code management, build automation, continuous integration, and issue tracking. **[8]**

10. No code checked into the development branch without an accompanying test. **[8]**

.

11. Documented code with user's manuals, technical descriptions, tutorials, example problem setup and user forums. **[6]**

CREATE Core Software Engineering Practices

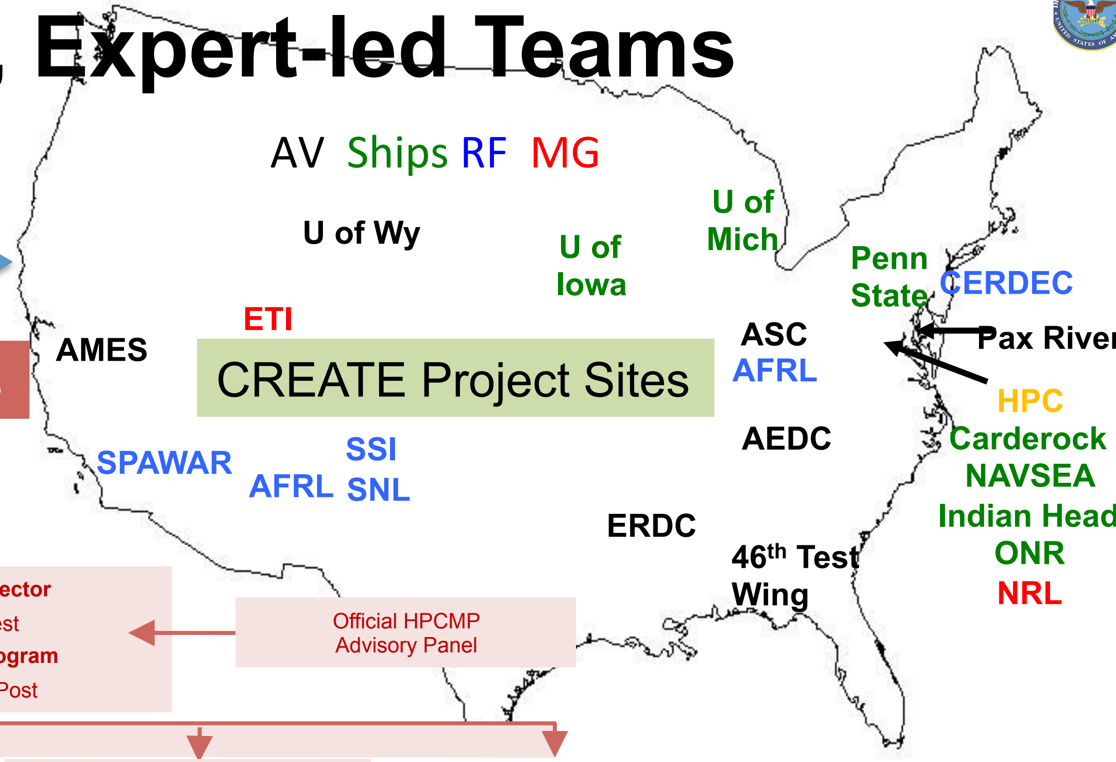
Requirements Definition

12. Reliance on prototypes to solidify difficult-to-specify, or possibly ambiguous requirements. [4]

Lean, Distributed, Expert-led Teams

Geographic

Embedded in Stakeholder Orgs



Organizational

HPCMP Director
John West
CREATE Program
Douglass Post

Official HPCMP Advisory Panel

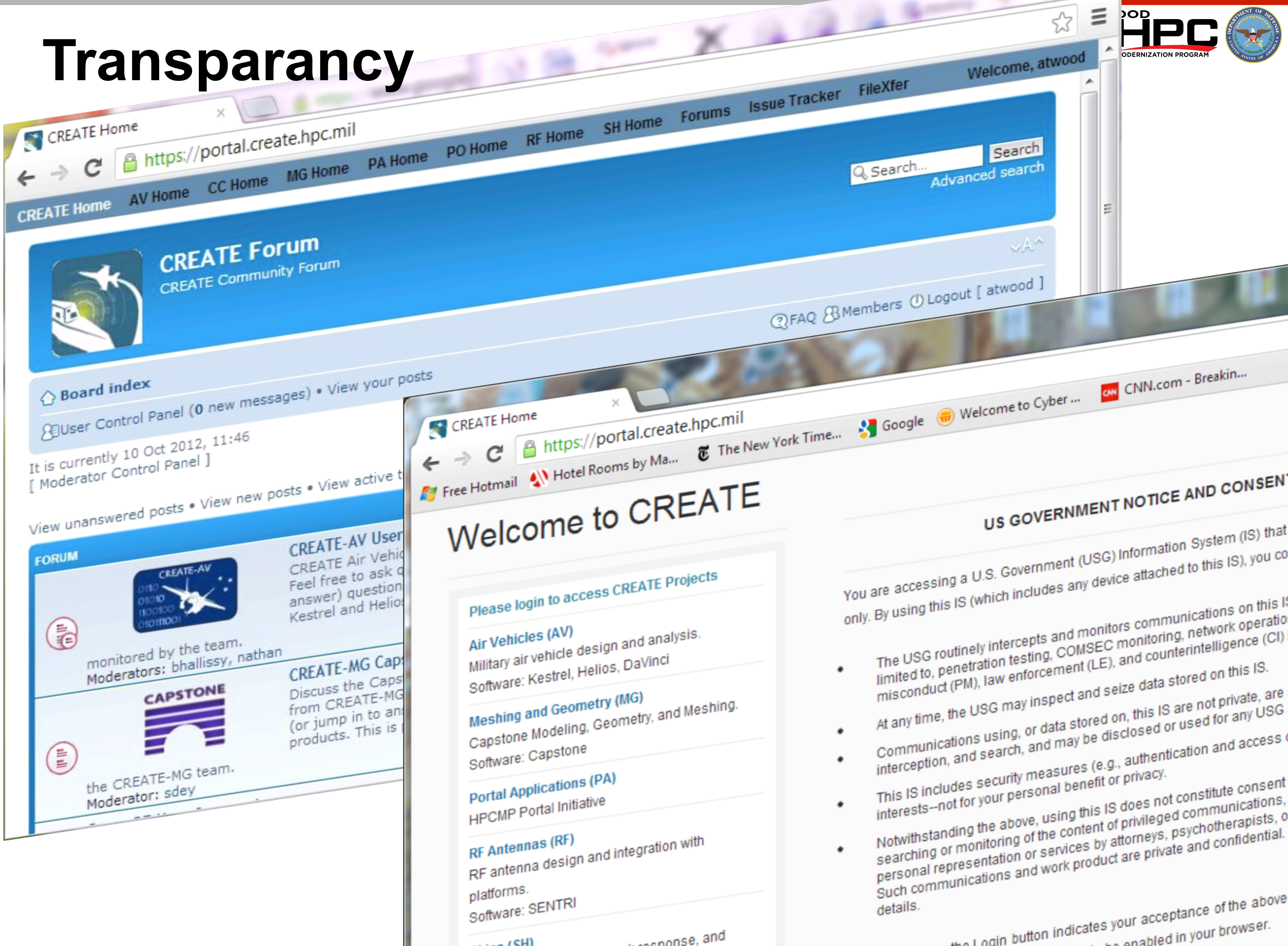
Ships Project – Myles Hurwitz, Carderock
- **NavyFOAM** – Joseph Gorski, Carderock
- **Integrated Hydro Design Environment** – Joseph Gorski, Carderock
- **Navy Enhanced Sierra Mechanics** – Tom Moyer, Carderock
- **Rapid Ship Design Environment** – Adrian McKenna, Carderock

Air Vehicles Project – Robert Meakin, NAVAIR, Patuxent River
- **Kestrel** – Scott Morton, Eglin AFB
- **Shadow-Ops** – Joe Laiosa, NAVAIR, Pax River
- **Helios** – Roger Strawn, Army, Ames
- **DaVinci** – Greg Roth, ASC & AFRL
- **Firebolt** – Robert Nichols, AEDC

RF Antennas Project – John D'Angelo, AFRL, WPAFB
- **SENTRI-core** – AFRL, WPAFB
- **SENTRI-V&V** – Andrew Greenwood, AFRL, KAFB

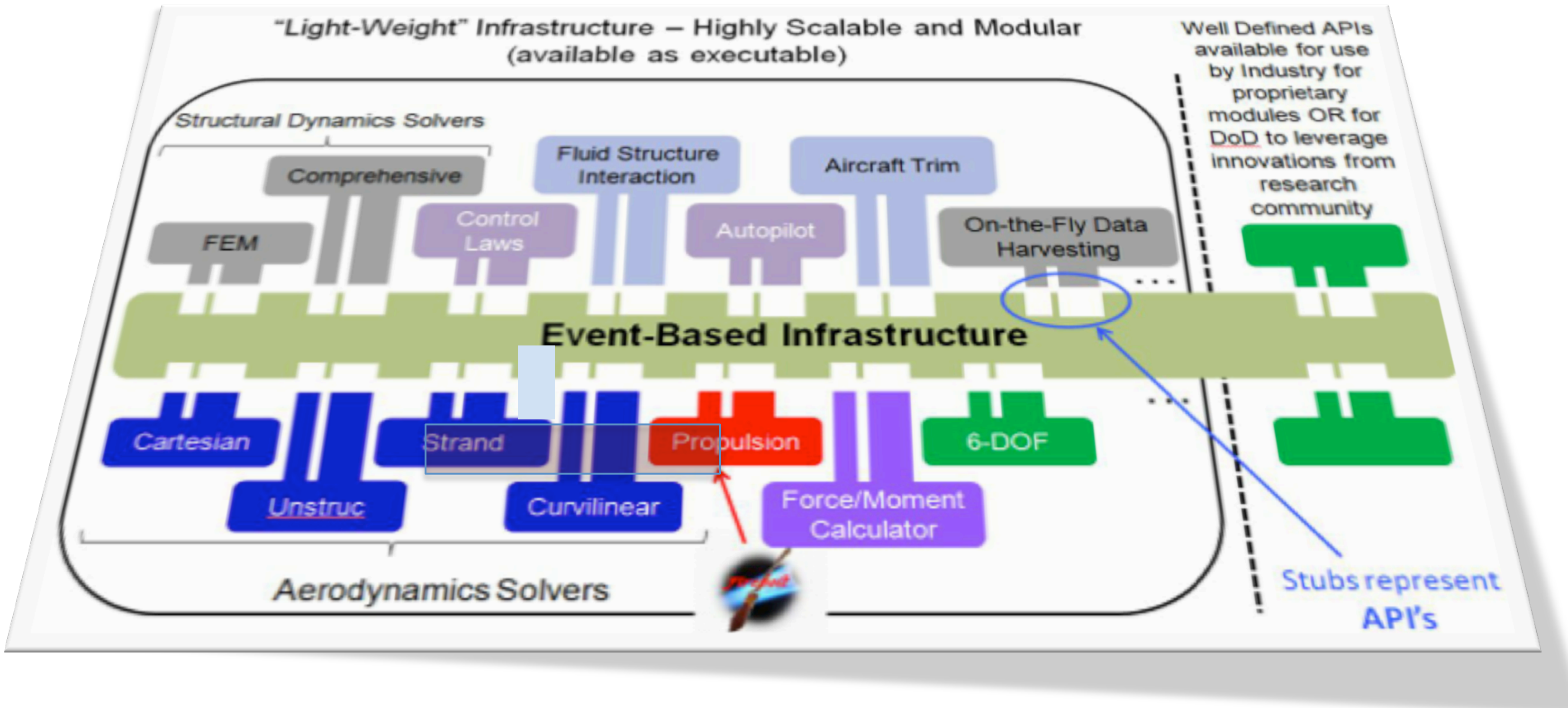
Mesh & Geometry Project – Saikat Dey, NRL
- **Capstone**

Transparancy



DOD HPC MODERNIZATION PROGRAM

Product Architectures Visible Across CREATE



Customer-Driven Use Cases

- Capture Requirements in customer-oriented language
- Clearly identify the "user"
- Describe the Goal of the user
- Specify Minimum functionality or performance expectations
- Describe main success scenarios

[from Cockburn, 2001, "fully-dressed" use cases]

Customer Driven "Use Cases"

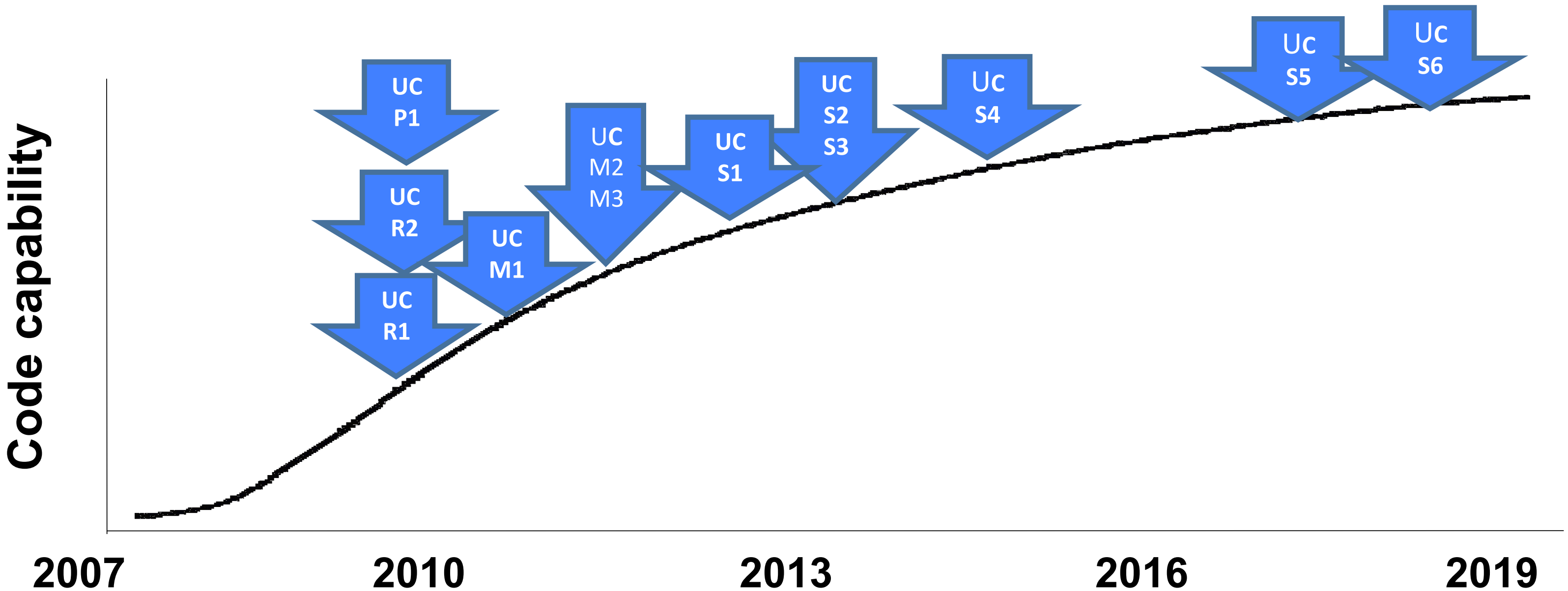
Use Case Example: Ship Shock (NESM)

Support Survivability Analysis (by Navy Structural Engineers) from Shock Damage from Explosives when

- Use Case I
  - Structural Response is Essentially Linear Elastic
  - Local Nonlinearities (mounts, joints, etc.)
- Use Case II (includes SURFEX)
  - Structural Response is Elastic/Plastic w/ Damage
  - Local Nonlinearities Included
- Use Case III
  - S... w... Structural

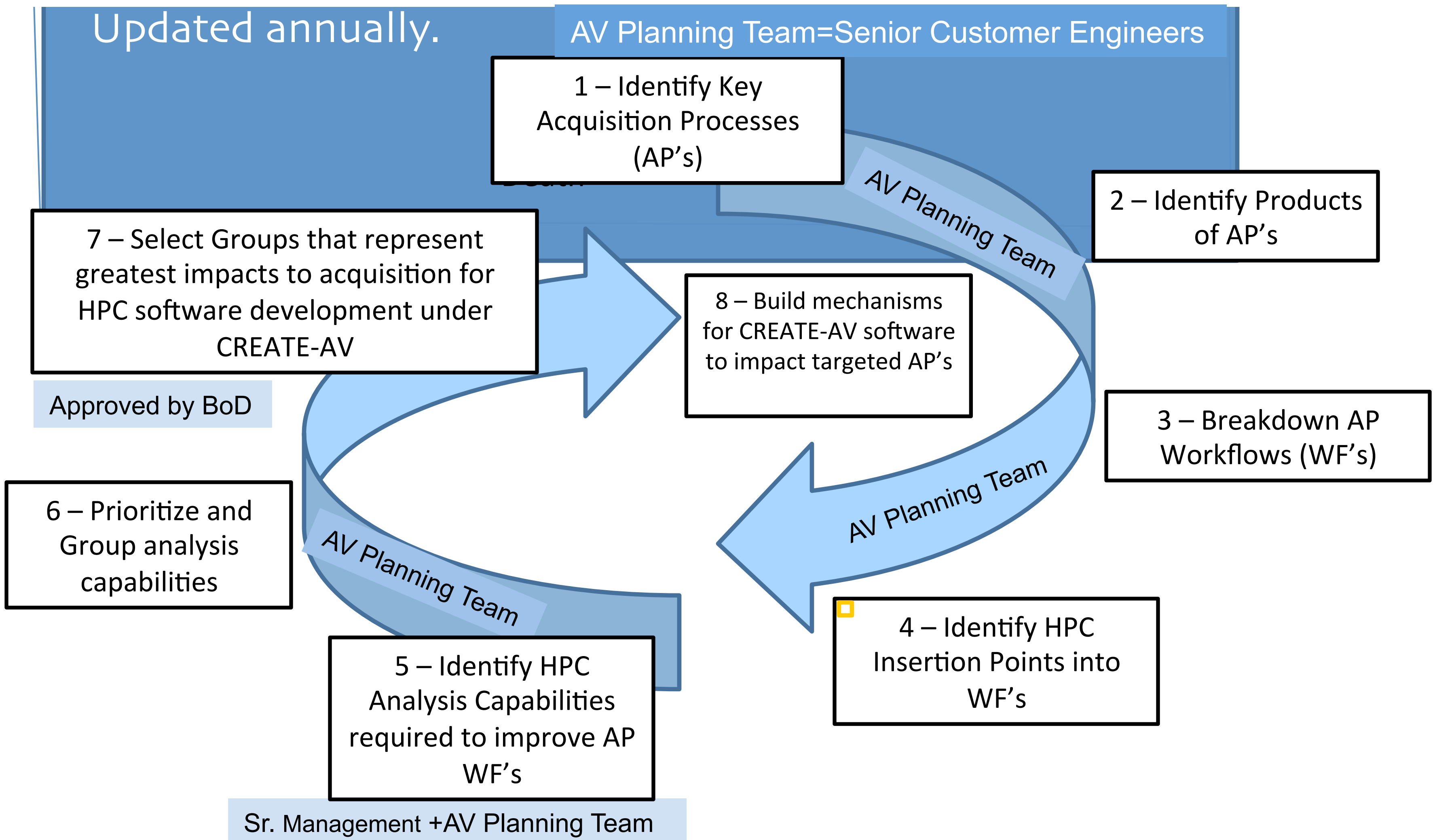
CREATE Project Roadmaps

Example from Ships Hydro



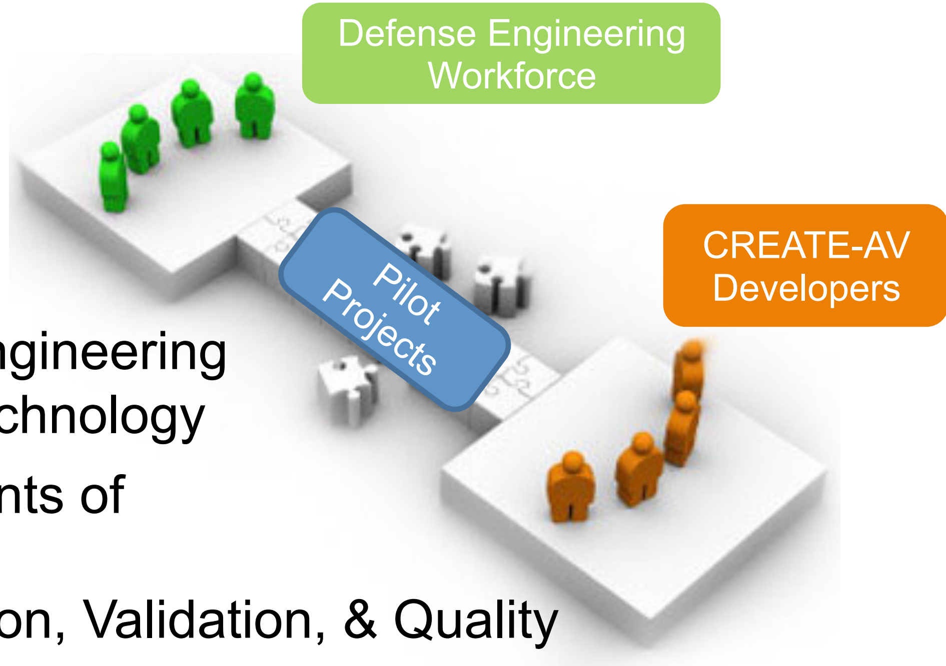
- Resistance Related
  - UCR1: Hull with fixed ship sinkage and trim
  - UCR2: Hull with computed sinkage and trim
- Powering Related
  - UCP1: Body force model for propulsor
  - UCP2 : Full propulsor/hull modeling
- Maneuvering Related (motions in calm water)
  - UCM1: Rotating arm steady turning motion
  - UCM2 : Planar Motion Mechanism (PMM)
  - UCM3 : Moving appendages and controller
- Seakeeping Related (involves waves)
  - UCS1 : Prescribed trajectory in regular waves
  - UCS2: Hull responds to regular waves
  - UCS3 : Prescribed trajectory in irregular waves
  - UCS4 : Predicted motions with moving appendages in waves
  - UCS5: Seaway loads with one way coupling to structures code
  - UCS6: Seaway loads with two way coupling to structures code

CREATE-AV Process to Manage Capability Gaps (Customer Requirements)



Pilots in AV

Annually execute between 4 and 6 Pilot Projects to "shadow" acquisition programs engineering workflows– 26 Pilots since 2008!



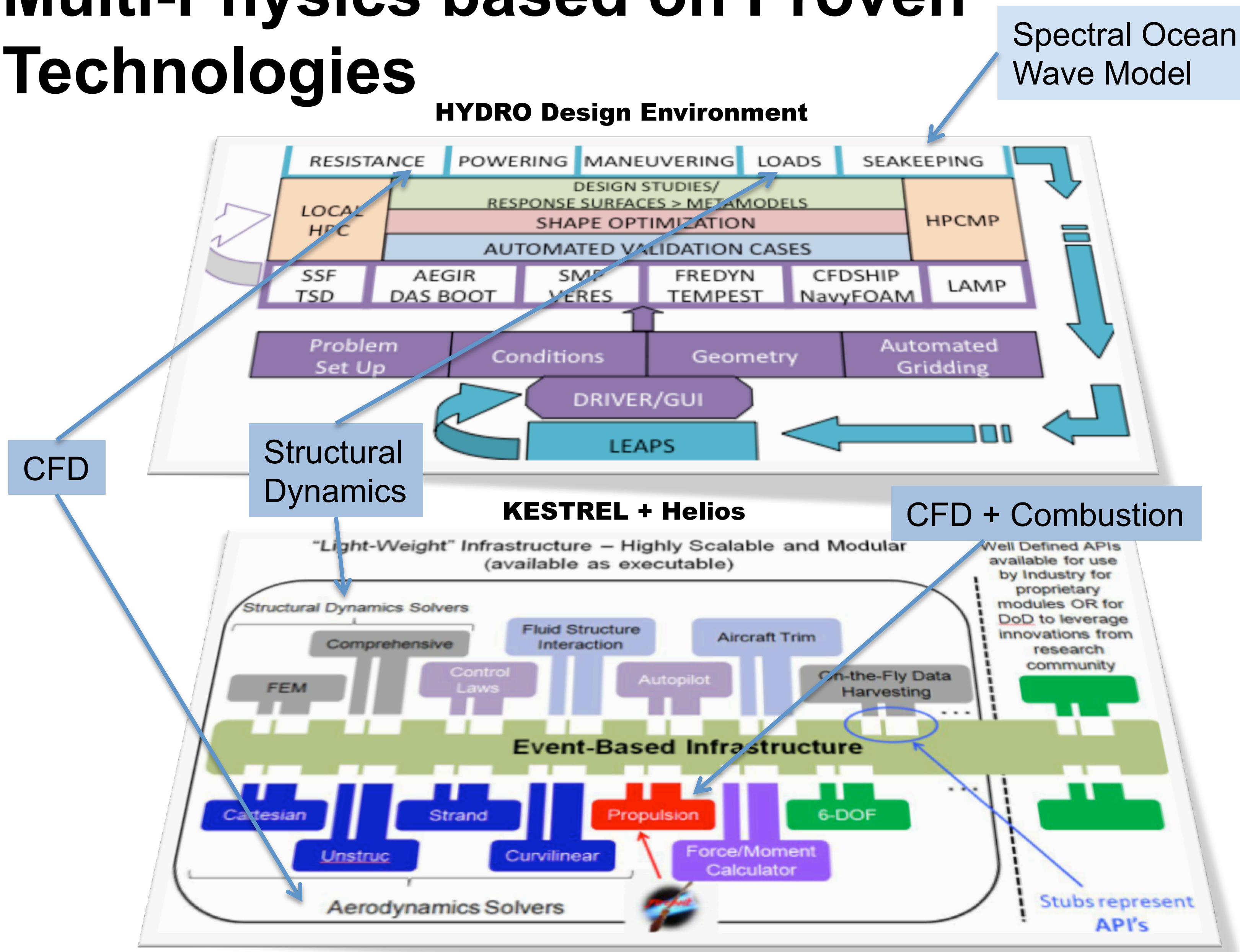
– Build bridges of trust between product developers and targeted acquisition engineering orgs in order to deploy CREATE-AV technology

– Learn workflows and actual requirements of targeted orgs

– Key roles in product VV&QA (Verification, Validation, & Quality Control)

- Build computational baselines
- Build archive of validation cases (VERY big deal)

Multi-Physics based on Proven Technologies



VVUQ aligned with NRC Best Practices

Cross-Walk to NRC Best Practices[1]

Verification Principles and Best Practices

- Principle: Solution verification is well defined only in terms of specified quantities of interest, which are usually functionals of the full computed solution.
  - Best practice: Clearly define the QOIs for a given VVUQ analysis, including the solution verification task. Different QOIs will be affected differently by numerical errors. CREATE VVUQ Practice 11
  - Best practice: Ensure that solution verification encompasses the full range of inputs that will be employed during UQ assessments. CREATE VVUQ Practice
- Principle: The efficiency and effectiveness of code and solution verification can often be enhanced by exploiting the hierarchical composition of codes and mathematical models, with verification performed first on the lowest-level building blocks and then on successively more complex levels.
  - Best practice: Identify hierarchies in computational and mathematical models and exploit them for code and solution verification. It is often worthwhile to design the code with this approach in mind. CREATE VVUQ Practice 8
  - Best practice: Include in the test suite problems that test all levels in the hierarchy. CREATE VVUQ Practice 8
- Principle: Verification is most effective when performed on software developed under appropriate software quality practices.
  - Best practice: Use software configuration management and regression testing, and strive to understand the degree of code coverage attained by the regression suite. CREATE Core Practice 11; CREATE VVUQ Practice 4
  - Best practice: Understand that code-to-code comparisons can be helpful, especially for finding errors in the early stages of development, but that in general they do not by themselves constitute sufficient code or solution verification. CREATE VVUQ Practice 6
  - Best practice: Compare against analytic solutions, including those created by the method of manufactured solutions—a technique that is helpful in the verification process. CREATE VVUQ Practice 6
- Principle: The goal of solution verification is to estimate, and control if possible, the error in each QOI *for the problem at hand.* (Ultimately, of course, one would want to use UQ to facilitate the making of decisions in the face of uncertainty. So it is desirable for UQ to be tailored in a way to help identify ways to reduce uncertainty, bound it, or bypass the problem, all in the context of the decision at hand. The use of VVUQ for uncertainty management is discussed in Section 6.2. "Decisions within VVUQ Activities").
  - Best practice: When possible in solution verification, use goal-oriented a posteriori error estimates, which give numerical error estimates for specified QOIs. In the ideal case the fidelity of the simulation is chosen so that the estimated errors are small compared to the uncertainties arising from other sources. Not addressed
  - Best practice: If goal-oriented a posteriori error estimates are not available, try to perform self-convergence studies (in which QOIs are computed at different levels of refinement) on the problem at hand, which can provide helpful estimates of numerical error. CREATE VVUQ Practice 6.2

Validation and Prediction Principles and Best Practices

- Principle: A validation assessment is well defined only in terms of specified quantities of interest (QOIs).
  - Best practice: Early in the validation process, specify the QOIs that will be addressed. CREATE VVUQ Practice 9
- Principle: A validation assessment provides direct information about model accuracy only in the domain of applicability that is "covered" by the physical observations employed in the assessment.
  - Best practice: When quantifying or bounding model error for a QOI in the problem at hand, systematically assess the relevance of supporting validation assessments (which were based on data from different problems, often with different QOIs). Subject-matter expertise should inform this assessment of relevance.
  - Best practice: If possible, use a broad range of physical observation sources so that the accuracy of a model can be checked under different conditions and at multiple levels of integration. CREATE VVUQ Practice 11
  - Best practice: Use "holdout tests" to test validation and prediction methodologies. In such a test some validation data is withheld from the validation process, the prediction machinery is employed to "predict" the withheld QOIs, with quantified uncertainties, and finally the predictions are compared to the withheld data. Not included
  - Best practice: If the desired QOI was not observed for the physical systems used in the validation process, compare sensitivities of the available physical observations with those of the QOI.
  - Best practice: Consider multiple metrics for comparing model outputs against physical observations. CREATE VVUQ Practice 11
- Principle: The efficiency and effectiveness of a validation assessment are often improved by exploiting the hierarchical composition of computational and mathematical models, with assessments beginning on the lowest-level building blocks and proceeding to successively more complex levels.
  - Best practice: Identify hierarchies in computational and mathematical models, seek measured data that facilitates hierarchical validation assessments, and exploit the hierarchical composition to the extent possible. CREATE VVUQ Practice 10
  - Best practice: If possible, use physical observations, especially at more basic levels of the hierarchy, to constrain uncertainties in model inputs and parameters. CREATE VVUQ Practice 10
- Principle: The uncertainty in the prediction of a physical QOI must be aggregated from uncertainties and errors introduced by many sources, including: discrepancies in the mathematical model, numerical and code errors in the computational model, and uncertainties in model inputs and parameters.
  - Best practice: Document assumptions that go into the assessment of uncertainty in the predicted QOI, and also document any omitted factors. Record the justification for each assumption and omission. CREATE Practice 9
  - Best practice: Assess the sensitivity of the predicted QOI and its associated

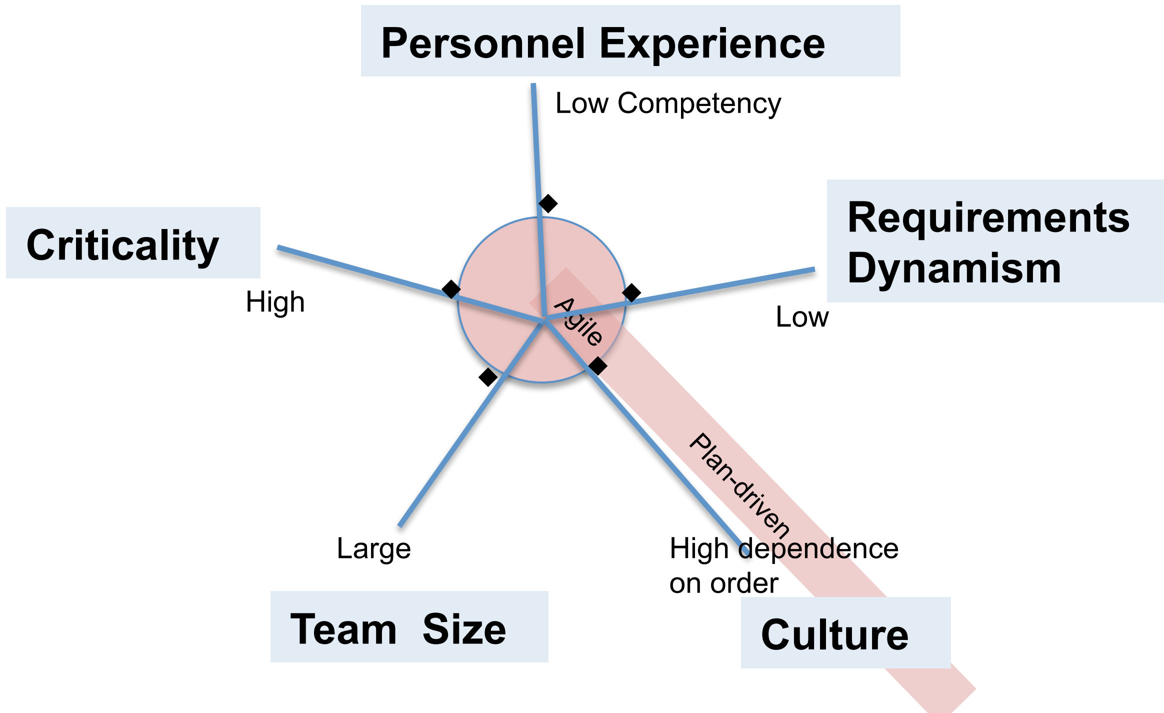
[1]From Chapter 7, Assessing the Reliability of Complex Models: Mathematical and Statistical Foundations of Verification, Validation, and Uncertainty Quantification

Workflow Management

Our Analysis

Notional Home Ground Chart for CREATE

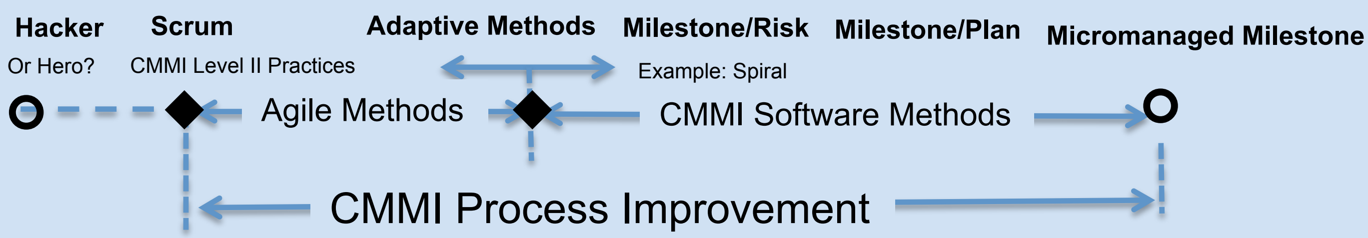
after Boehm, Using Risk to Balance Agile and Plan Driven Methods, IEEE Computer Society, 2003



Workflow Management

Our Approach

Milestone-driven, but flexible execution



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

Workflow Management

Our Approach

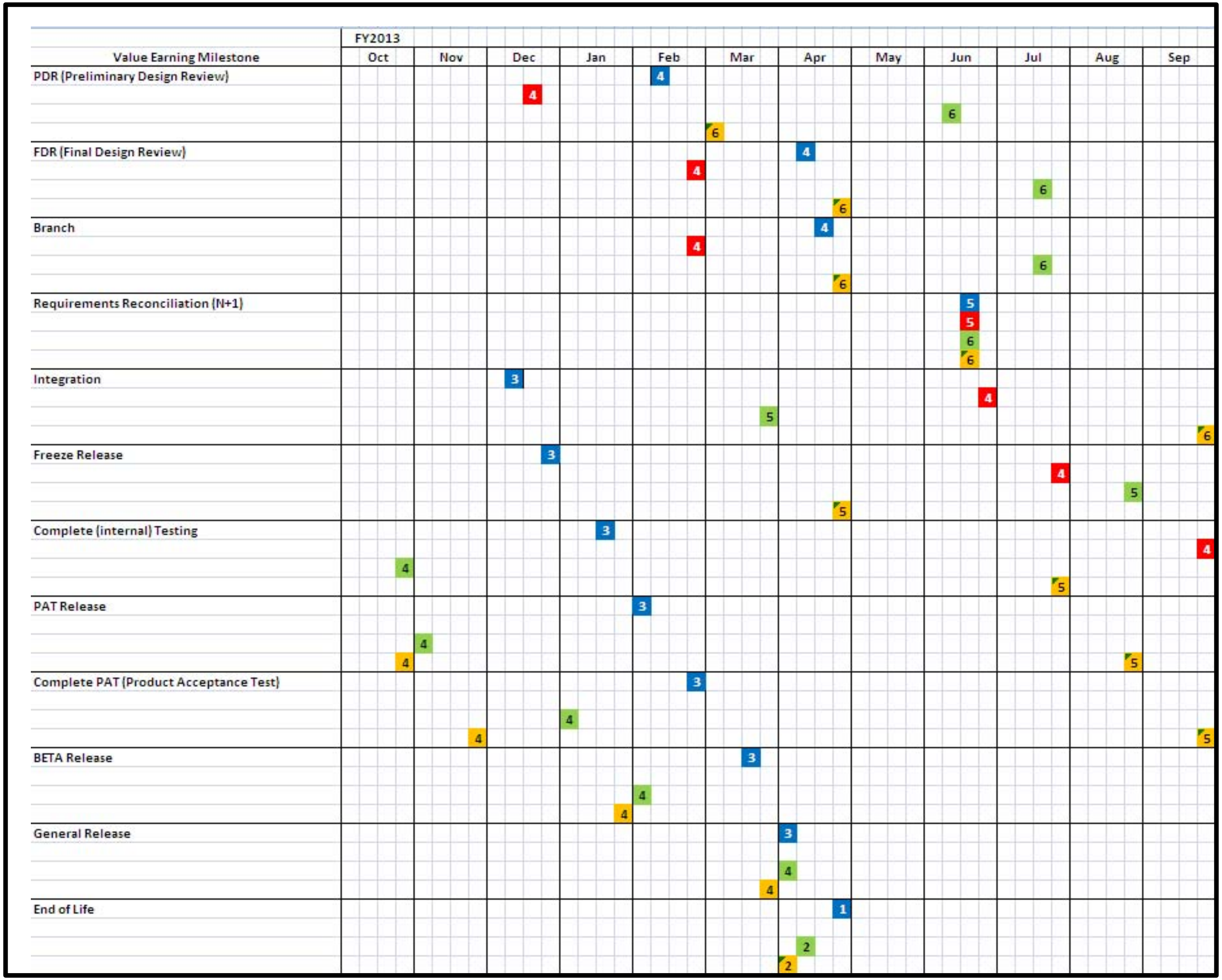
Iterative with Annual Releases

Canonical Milestones
- Preliminary Design Review
- Final Design Review
- New Development Branch
- Freeze of Development Branch
- Alpha Testing
- Beta Testing
- CCB assessment of readiness for release
- Release
- End-of-life for version

CMMI Best Practices for:

- Program Management
- Requirements Management
- Configuration Management
- Quality Assurance
- Verification, Validation & UQ

AV Integrated Milestone Chart (Covering 4 Codes) -- FY2013



CREATE Product Release Cadence

Fiscal Year	FY2010				FY2011				FY2012				FY2013-planned			
Quarter	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
AV-DaVinci							1				2				3	
AV-Helios		1					2			3				4		
AV-Kestrel		1				2						3			4	
MG-Capstone				1				2				3				4
RF-SENTRI	1	1.5						2				3				4
Ships-IHDE	1				2		2.1		3				4			
Ships-NavyFoam				1					2				3			
Ships-NESM	0.1					1				1.1			1.5			
Ships-RSDE										0.5			1			

Benefits of Annual Releases

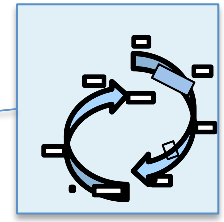
- Reach Closure on Incremental Capabilities
- Provides annual demonstration of significant progress
- Creates prototypes which facilitate customer testing and input
- Mitigation for Requirements Creep

NESM Version	Capabilities
NESM v0.1	• Preliminary UC I
NESM v1.0	• Verified UC I • Preliminary UC II
NESM v1.1	• Partially Validated UC I • Verified UC II
NESM v2.0	• Partially Validated UC II • Preliminary UC III
NESM v3.0	• Partially Validated UC III • Preliminary UC IV
NESM v4.0	• Verified UC IV
NESM v4.1	• Partially Validated UC IV • Preliminary UC V
NESM v5.0	• Partially Validated UC V • Preliminary UC VI
NESM v6.0	• Partially Validated UC VI

Note: Startups may take longer

Shared Development Practices

- Requirements Management
- Software Quality Attributes
- Design & Implementation
- Software Configuration Management
- Verification & Validation
- Release Practices
- Customer Support



Maintainability
Extensibility
Performance

Workflow Management:
Agile, Iterative, Spiral

Central code repository
Configuration Management Tools(Subversion)
Document Repository (Confluence)
Configuration Control Boards

Support only 2 releases
Issue Tracker (JIRA)
CREATE Community Web Services
User Forums (CREATE Forum)
On-line Application Documentation

Annual Releases

Well-Documented Software Development Plans

Capstone Backlog

Appendix B: Backlog for FY11

ID	Feature/Capability
	Geometry Database
501	Geometry Basic API enhancements
502	Geometry Query API enhancements
503	Geometry Modify API enhancements
504	Geometry AdvModify API enhancements
505	Geometry Visualization API enhancements
506	Super-Topology Data Structure (on the top of standard kernel topo)
507	Super-Topology Implementation In SMLIB Geometry Database
508	Super-Topology Implementation in Discrete Geometry Database
509	Super-Topology Implementation in Parasolid Geometry Database
510	Handling of Automatic Volume Creation Issue in SMLIB (via Super-Topology?)
511	Composite patch reparametrization
512	Super-Topology base topology discrete parametrization via unstructured triangular meshes & segmented edges
513	Super-Topology 2 tiers patch/base parametrization (for quick evaluation)
514	Functionality to automatically & manually define "Gap" face definition (closing boundaries etc.)
515	"Gap" face implementation in super-topology
516	"Gap" face (closed boundaries) using Delaunay point insertion
517	"Gap" face meshing using Nurbs evaluation (Itho's style)
518	Interpolation scheme for patch > base topologies for fast evaluation
519	Functionality to automatically define composites based on attribution, angle threshold etc.
520	Function to manually define composites
521	Super-Topology Workflow (Version #1) The super-topology is only created and maintained when generating meshes
522	Super-Topology Workflow (Version #2) the super-topology is always maintained while working on the model

Work Breakdown

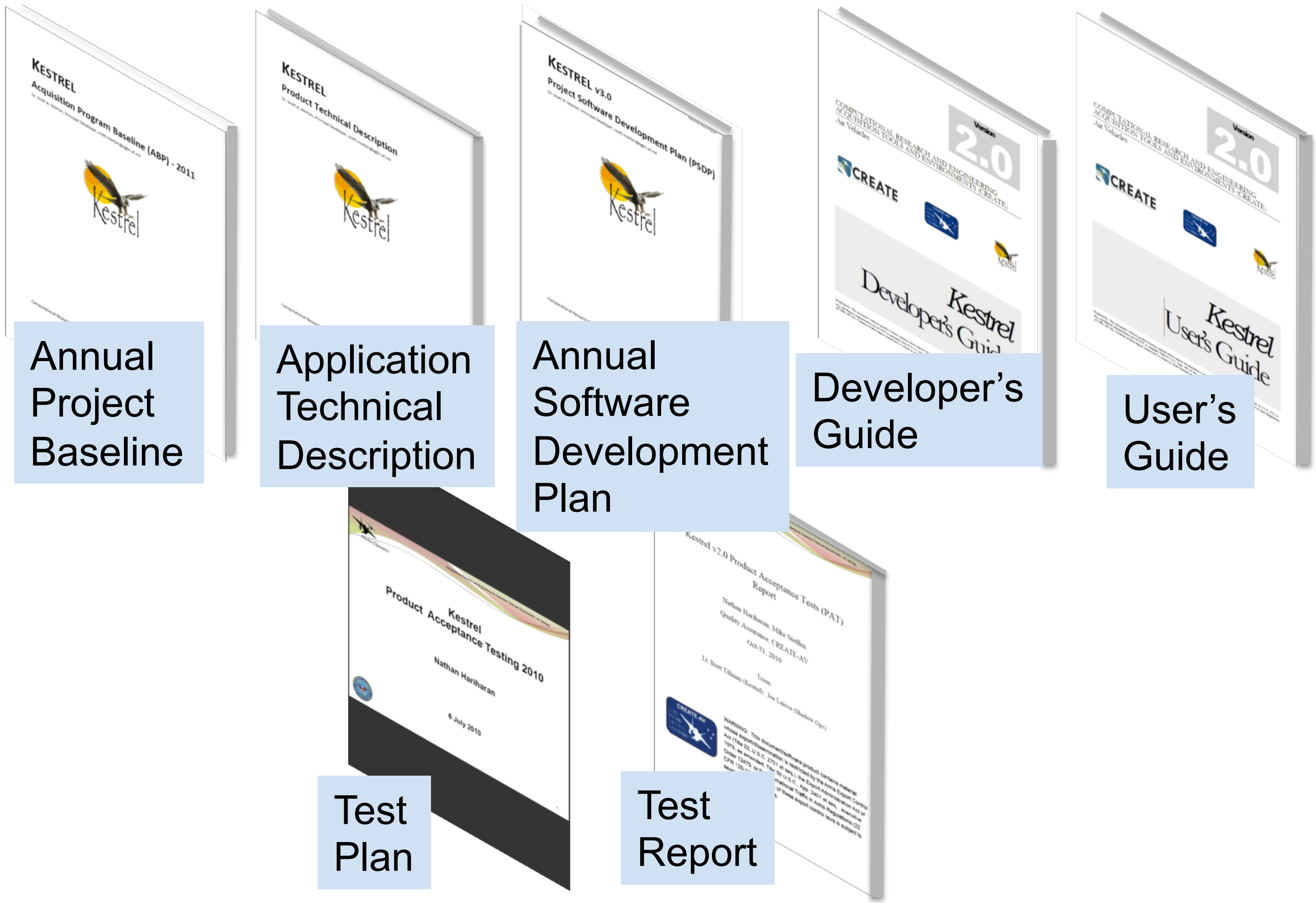
Work Breakdown for FY10 Features in V 1.0 based on V 1.0 FDR

PSTONE ature* (ritical feature for elease)	Implementing Work Package	Sub-package (Backlog)	Dependencies
rk			
	Produce developer documentation		
	Framework Implementation		
	Refactor/Optimize Framework		9
Module*			
	Geometry API Design		
	Geometry API Documentation		12
	Geometry I/O Infrastructure		9, 12
	Implement Geometry API: SMLIB Reference		9, 12
		Implement Query	9, 12
		Implement Modify	9, 12
		Implement Modify Advanced	9, 12
		Implement Visu	9, 12
		I/O Implementation	9, 12
		Evaluation of Tests (SMLIB)	15
	Implement Geometry API:		

Baseline Schedule

ption	Dependency and type	Expected End Da
e 1.0 release review and 2.0 PDR	Capstone 1.0 Release	Nov 30, 2010
ser cases for Capstone	Depends on AV, RF, Ships	Nov 30, 2010
asolid Software and tegration with	Depends upon PO contract with NGIT	Nov 30, 2010
e for exterior on and meshing		Dec 30, 2010
Boundary-layer (MG-FY11-02)	Depends on PETTT support for AFLR integration	Mar 30, 2010
support	Depends on 3	Mar 30, 2010
FDR	Depends on 1 and 2	Mar 30, 2011
mesh d 04)	Depends on 2	Apr 30, 2011
and MG-	Depends on 2	Apr 30, 2011
2.0 alpha-testing	2-8	
12 Begin Capstone 2.0 beta-testing	9	May 30, 2011
Release Capstone 2.0	10	Jul 30, 2011
		Sep 30, 2011

Well-Documented Applications



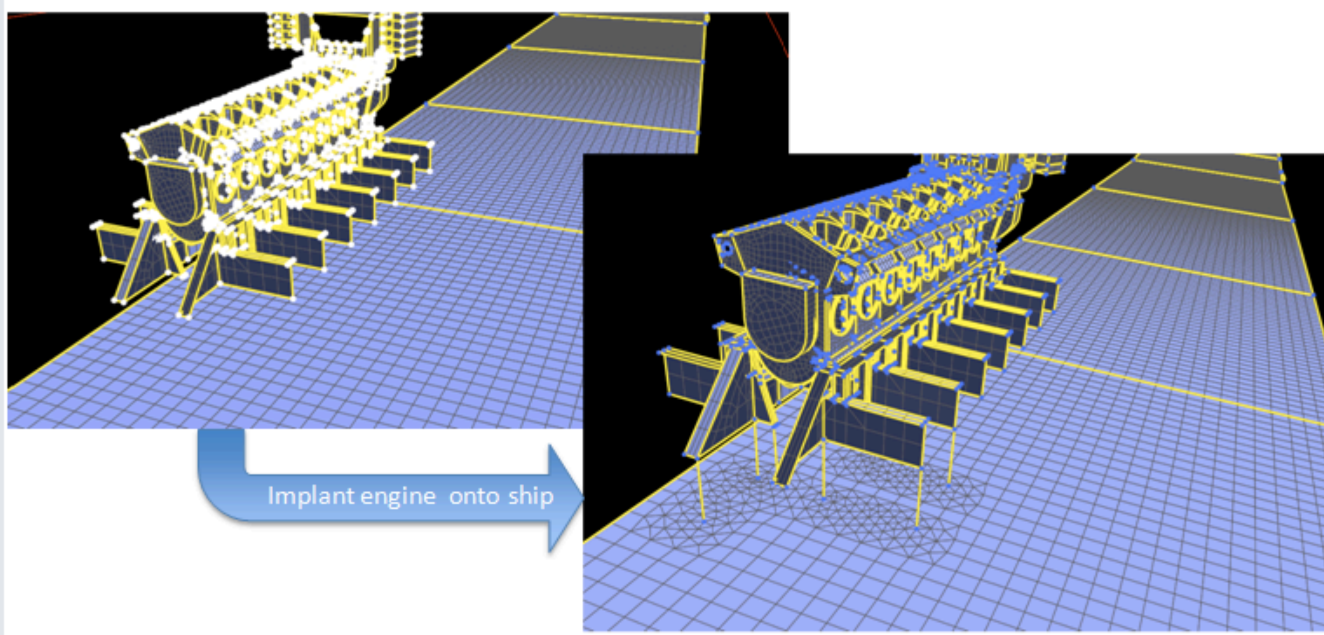
Prototype: A Example from MG- Capstone

- Original Requirement: Non-penetrating component implant (MG-09-UC-01)

1: non-penetrating implant (imprint) [MG-09-UC-01]
  a) exact vertex imprint (recover designated vertices of the component in the ship mesh)
  b) exact edge imprint (recover designated vertices and edges of the component in the ship mesh)
2: penetrating implant (boolean) [MG-09-UC-02]
  a) surface mesh only (both component and ships are surface meshes) (second slide)
  b) mixed-dimension (component and ships may have mixture of edge/face/region entities)

Ship Component Connecting Tutorial

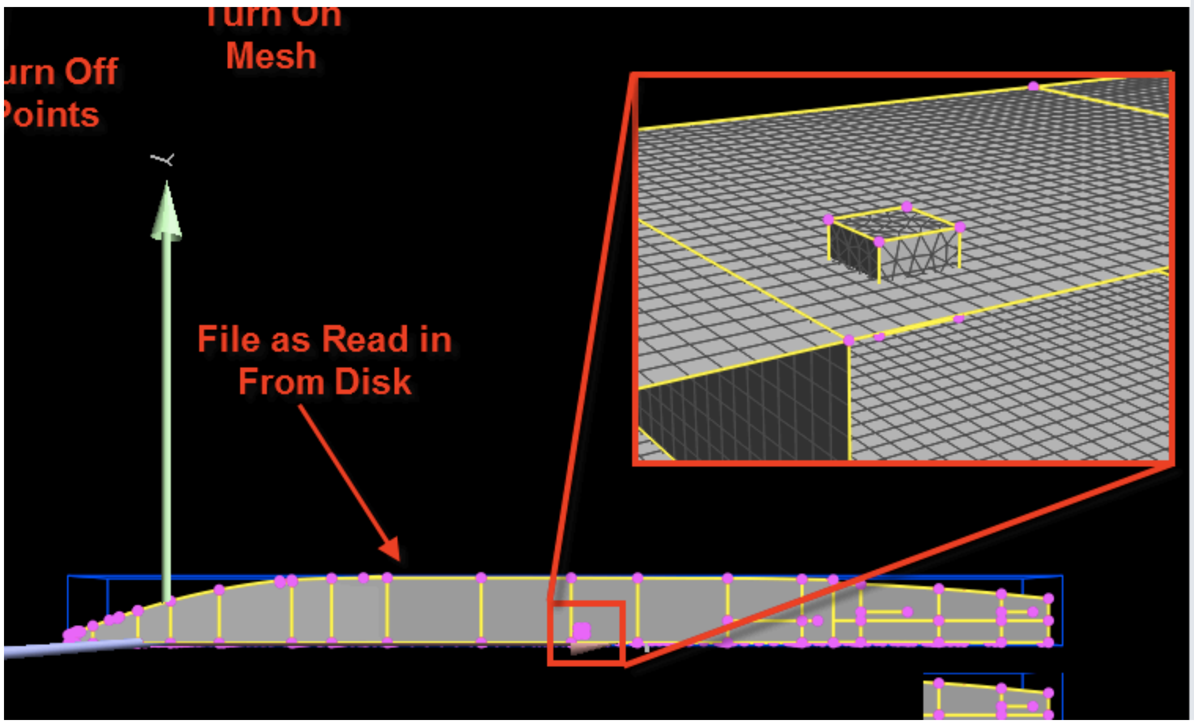
The goal of this tutorial is to demonstrate MG's capabilities to connect disjoint meshes (component and ship) through selected vertices/edges by projecting along link directions over selected faces (surfaces). For a detailed description of the algorithm developed for Create MG, select this Ship_Connection_Algorithm.pdg.

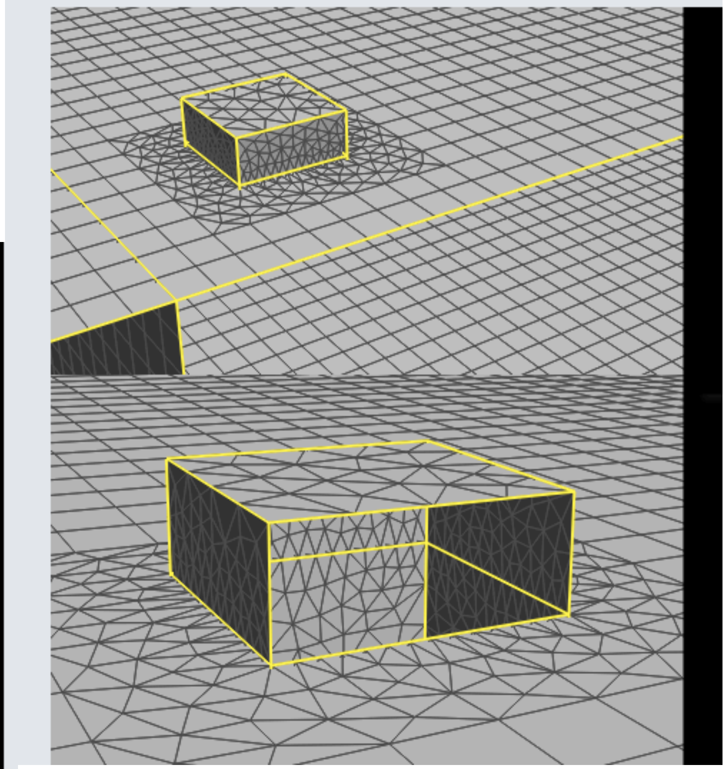


Connection of Disjoint Meshs

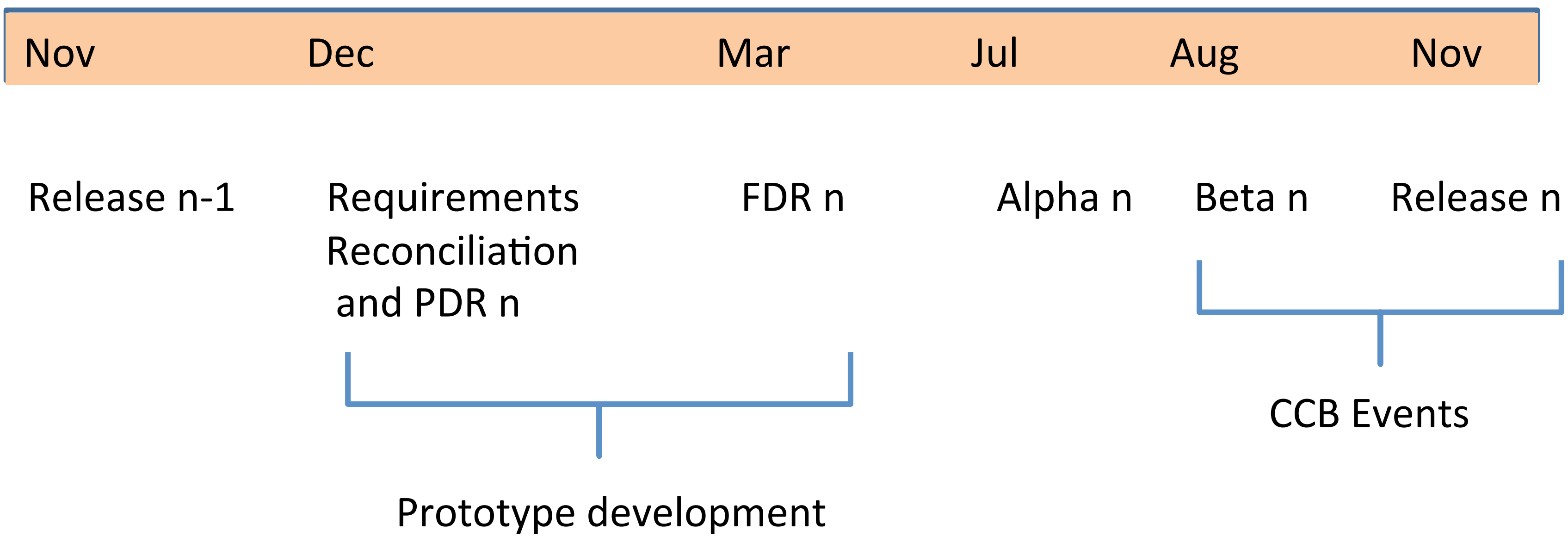
Prototypes: An Example from MG-Capstone

- Penetrating component implant (MG-09-UC-02)





MG Prototype Cadence



Summary

CREATE has successfully managed risk with sound software Development practices --

1. Based on lessons learned from real scientific code projects
   [from DARPA HPCS Case Studies and others of both successes and failures]

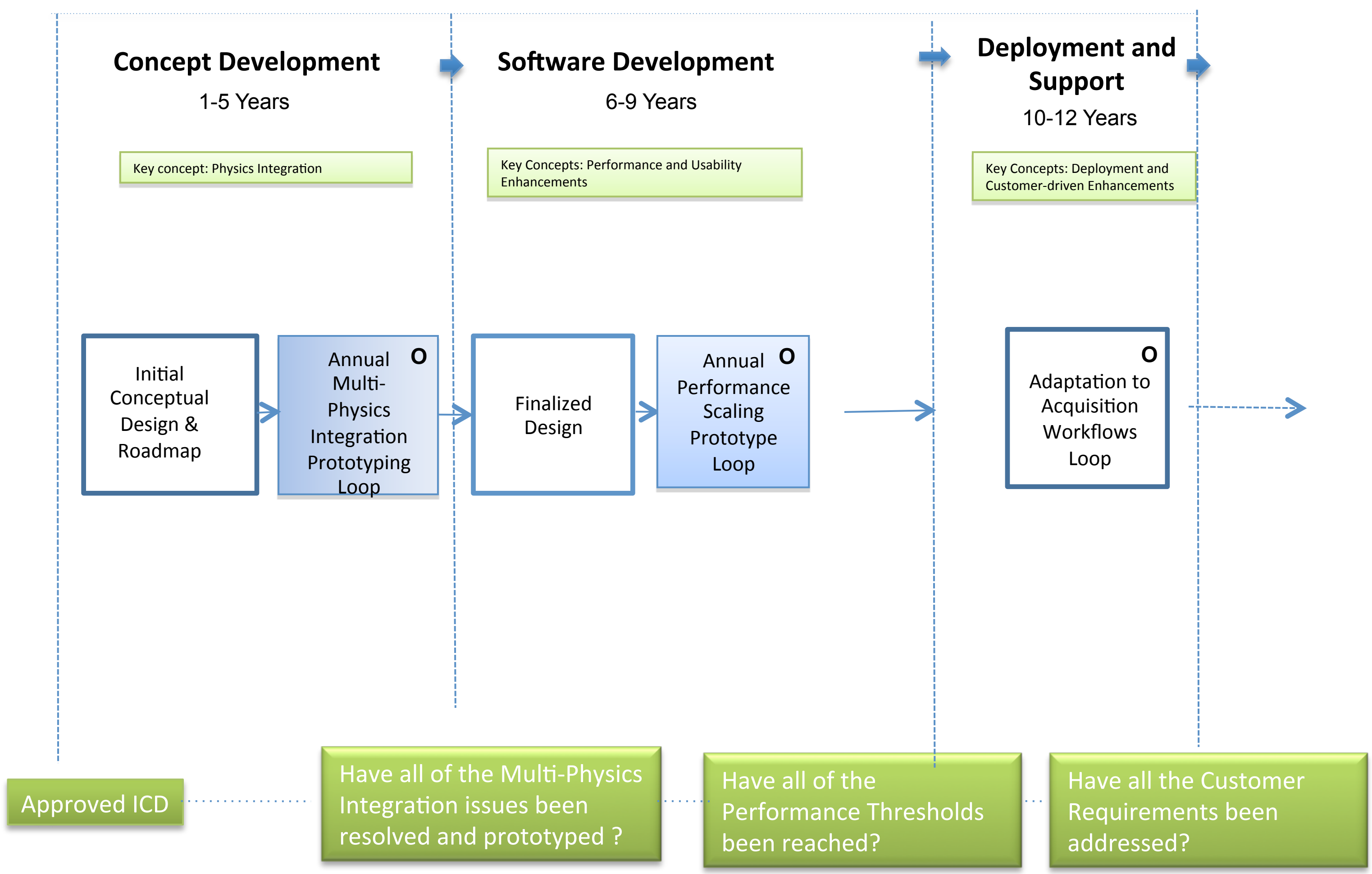
2. Addressing documented risks inherent in these projects
   [based on Software Engineering Institute Risk Taxonomy]

3. Documented in CREATE Software Development Guidance[SEPP and PMP]

4. Resulting in 100% success to date

Backup Slides

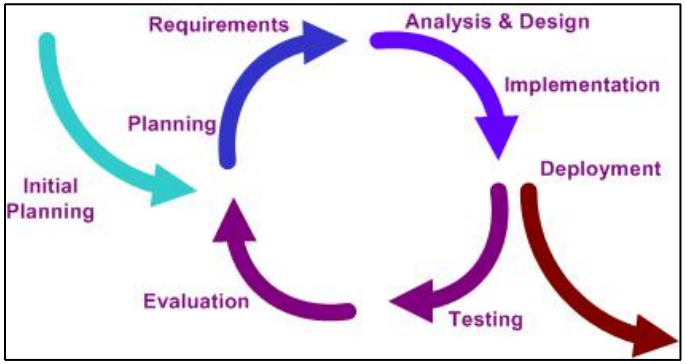
CREATE Development Rhythm



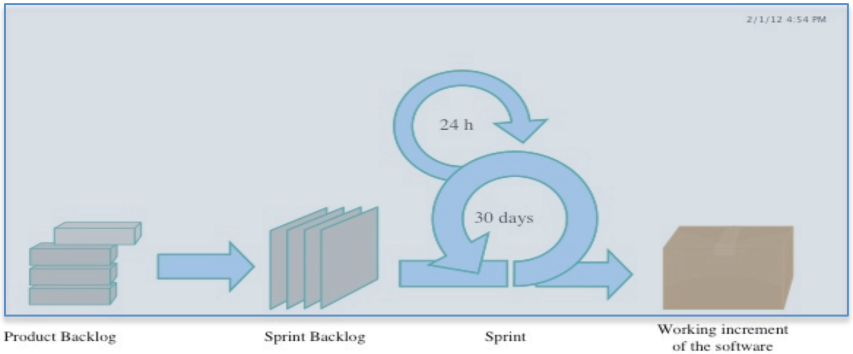
Distribution C. Please see Page 1 for additional information

Flexible Workflow Execution

The Management of CREATE Development Workflow



Iterative Workflow



Scrum Workflow

- **Iterative (and Incremental)**
  All
- **Agile (Scrum-like)**
  RF,Capstone, NESM, NavyFoam, RSDI, IHDE, DaVinci
- **Spiral**
  Capstone

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.

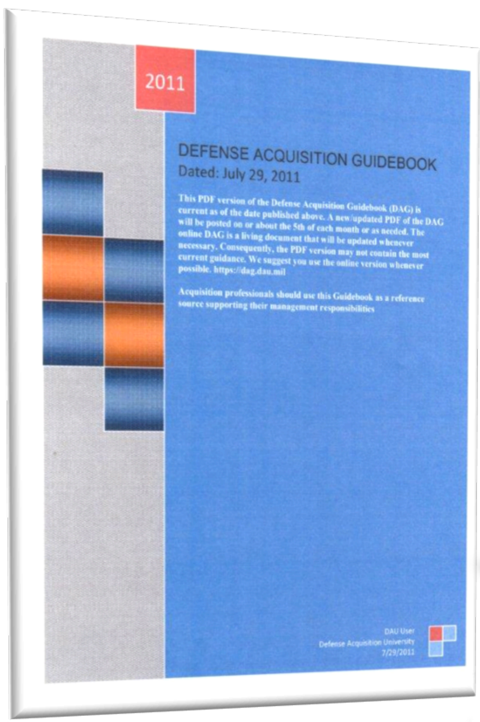
CMMI - Scrum Mapping: Some examples

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 Burndown 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

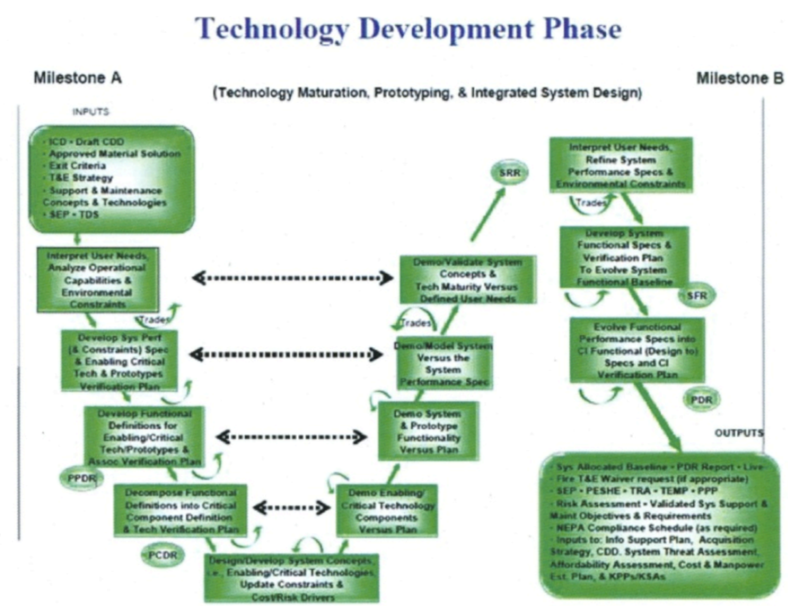


Figure 4.3.2.3.F1. Systems engineering-related steps during Technology Development

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

A Software Engineering History of CREATE:

ICDs

Startup Planning

Software Engineering Practices, v 0.1

Software Engineering Practices, v 1.0

Software Engineering Practices, v 2.0

3 Releases For Some CREATE Tools

FY07

FY08

FY09

FY10

FY11

FY12

Recruit PMs

Recruit Development Teams

Project Management Plan, v 1.0

Pilots Start

Initial Releases

All tools Have Had at Least 1 release; 8 two or more Releases