The Use of Systems Engineering and Open Architectures to Reduce Development Cycle Time in Complex Systems

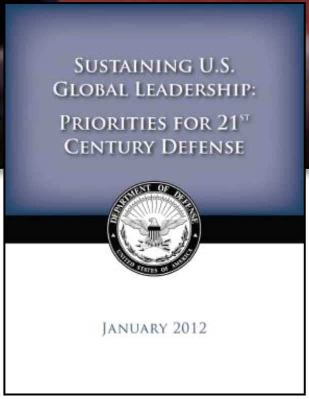
Defense Acquisition University

Foundational Learning Workflow Learning Performance Learning

Dr. Craig Arndt
Ann Wong

24 Oct 2016

POTUS and SECDEF: "DoD Will Be Agile"


SUSTAINING U.S. GLOBAL LEADERSHIP: PRIORITIES FOR 21ST CENTURY DEFENSE

JANUARY 2012

"The United States is going to maintain our military superiority with armed forces that are agile, flexible and ready for the full range of contingencies and threats."

- President Obama

"The US joint force will be smaller and leaner. But its great strength will be that it will be more agile, more flexible, ready to deploy quickly, innovative, and technologically advanced. That is the force for the future."

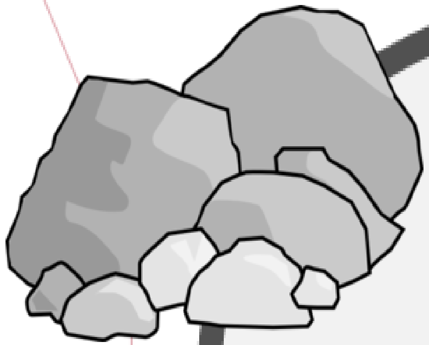
- Secretary Panetta

Defense Security Review, 5 Jan 12

Objectives

- Constraints
- Modeling Systems Engineering as a Control System
- Feedforward Toolbox
- Putting the “A” into Systems Engineering
- Summary

Constraints – Between a Rock….



Evolving Threats

Pace of Technology

COTS

Increasing Complexity

Defense Acquisition System

Funding Restraints

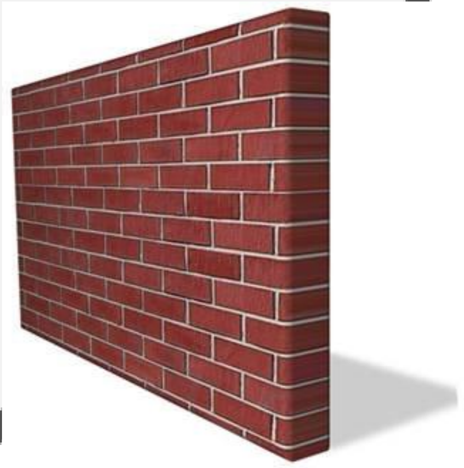
Accelerated Fielding

Increasing Software-related Demands

Risk

Changing Requirements

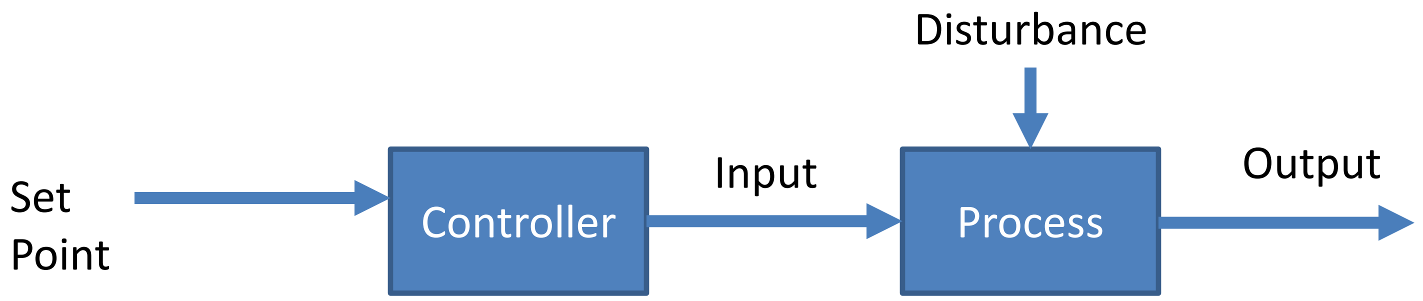
Interoperability



Systems Engineering

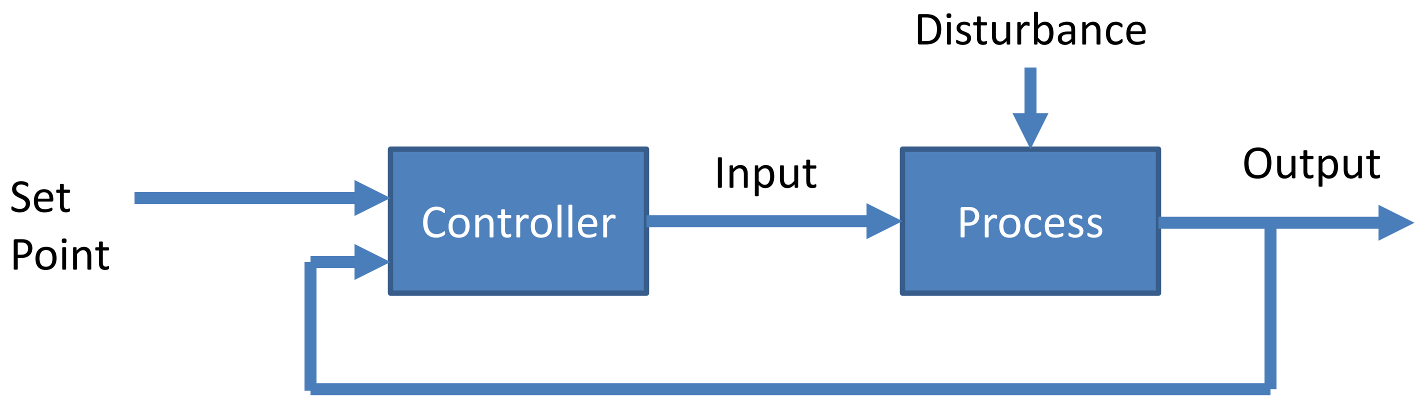
- DAG: Systems engineering (SE) is a methodical and disciplined approach for the specification, design, development, realization, technical management, operations, and retirement of a system.
- Systems engineering is the “art and science” of developing an operable system capable of meeting a holistic set of often conflicting requirements.
- Systems Engineering is:
  - Interdisciplinary
  - Integrative
  - Holistic
  - Iterative and recursive
  - Socio-technical (hardware, software, people, facilities, data, documentation)

System Engineering as a Open Loop Control System



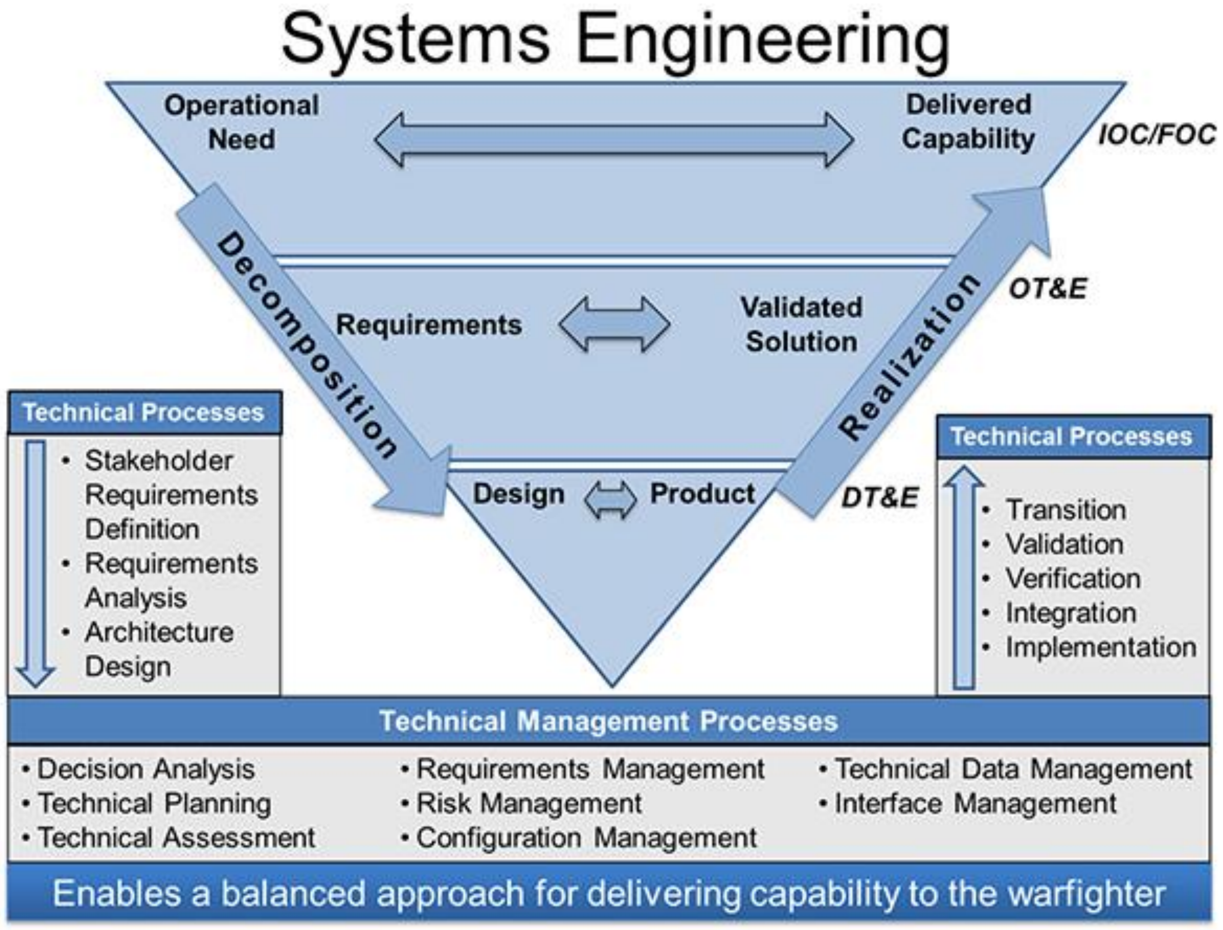
- Wild, Wild, West
- Pros:
  - Simple
  - No need for any sensors
- Cons:
  - Would work only for very stable development, where all variables are known
  - Requires a near-perfect model of the system
  - Deviations to final product quality is allowed
  - Cannot compensate for unknown disturbances
  - Would not work for complex systems

System Engineering as a Closed Loop Feedback Only System



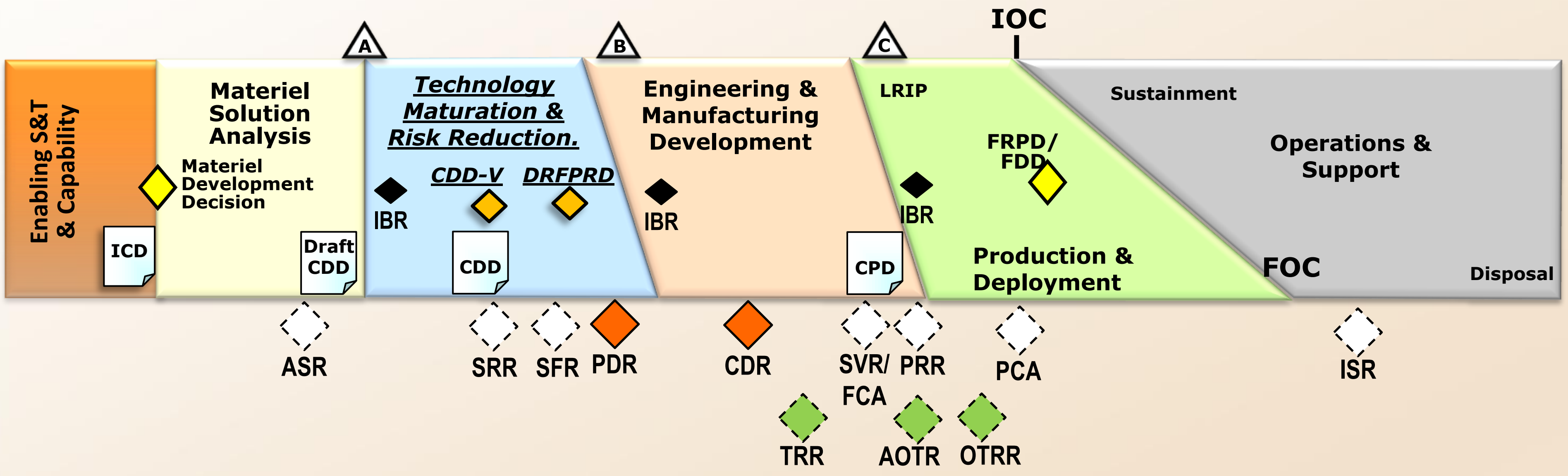
- Evaluation and Feedback occurs at output after a process is complete
- Pros:
  - Product under evaluation is more stable, complete to that point
  - Expected to meet a certain set of known, set requirements
- Cons:
  - Time delay incurred when reacting to corrective feedback
  - More extensive rework
  - Reactive Process
  - Emphasizes “perfection” → absolute delivery
- Represents Status Quo

DoD Systems Engineering Process Model



SE in Lifecycle Framework

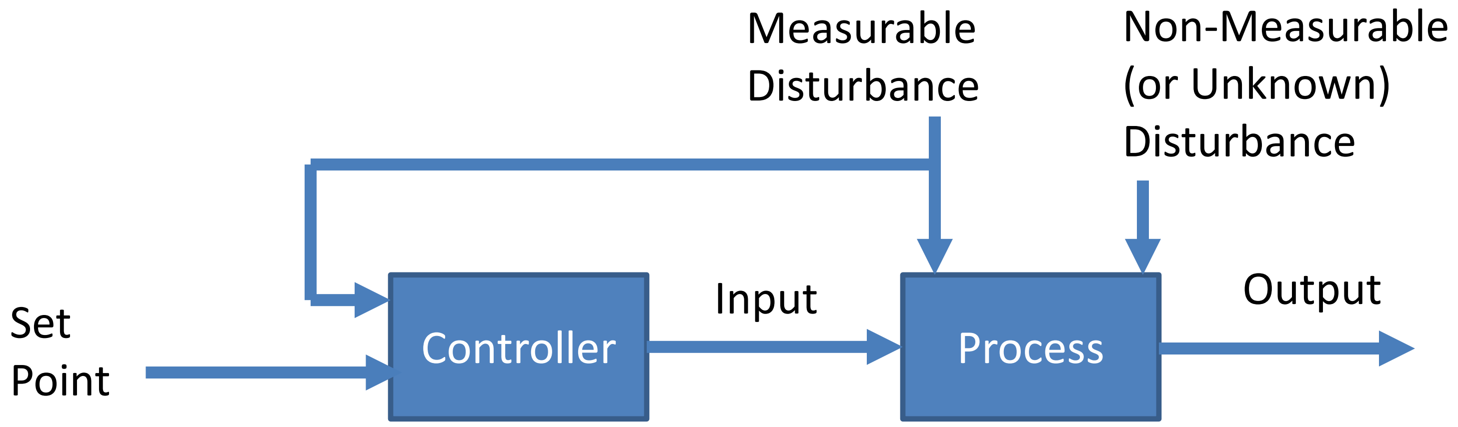
Technical Reviews & Decision Points



AOTR	Assessment of Operational Test Readiness	IBR	Integrated Baseline Review
ASR	Alternative System Review	ISR	In Service Review
CDD	Capability Development Document	MDD	Materiel Development Decision
CDD-V	CDD Validation Point	OTRR	Operational Test Readiness Review
CDR	Critical Design Review	PCA	Physical Configuration Audit
CPD	Capabilities Production Document	PMB	Performance Measurement Baseline
DRFPRD	Development RFP Release Decision	PRR	Production Readiness Review
EMD	Engineering and Manufacturing Development	S&T	Science & Technology
FCA	Functional Configuration Audit	SRR	System Requirements Review
FDD	Full Deployment Decision	SFR	System Functional Review
FRPD	Full Rate Production Decision	SVR	System Verification Review
ICD	Initial Capabilities Document	TRR	Test Readiness Review
IOC	Initial Operational Capability		

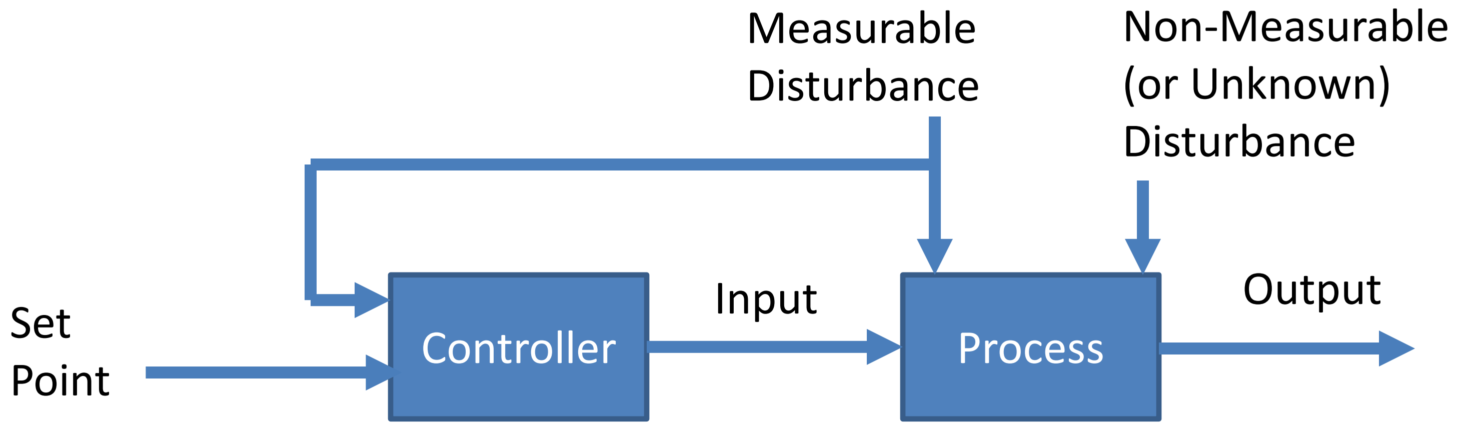
🔶	Mandatory technical Reviews
◇	Best practice technical reviews and audits
🟩	Test reviews (see DAG chapter 9)
◆	IBR is a PM review of KTR PMB for a contract with EVM. Has a major technical component

System Engineering as an Open Loop With Feedforward System



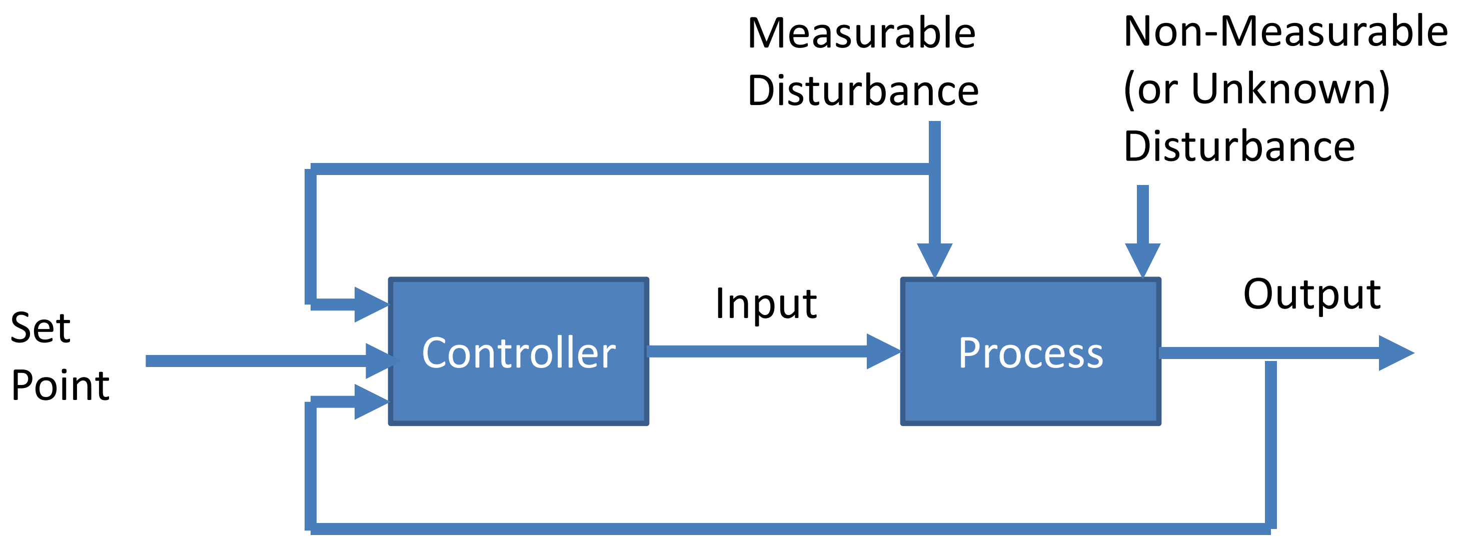
- Pros:
  - Minimum reliance on set goals and satisfying specific requirements
  - Allows change in outcomes
- Cons:
  - Not necessarily a repeatable process
- Feedforward systems can seem out of control, that’s the idea. For truly rapid, innovative systems development we need to maintain visibility not control

System Engineering as an Open Loop With Feedforward System



- New systems designed to be a feedforward will exhibit significantly different behavior. This means that decisions made at the lowest level will amplify themselves creating the possibility for both rapid failure and rapid development of high quality products

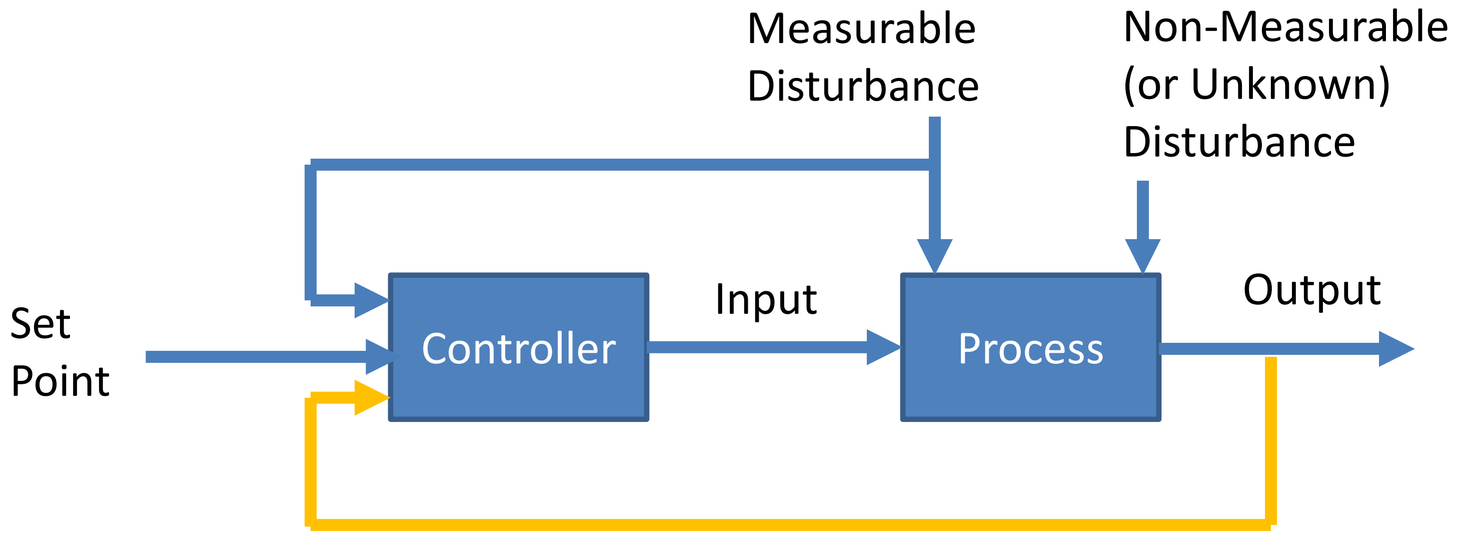
System Engineering as a Closed Loop With Feedforward System



- Best representative of the system engineering process as practiced
- Best of both worlds
  - Reap the benefits of the Pros
  - Mitigates the Cons

- What does DoD do well?

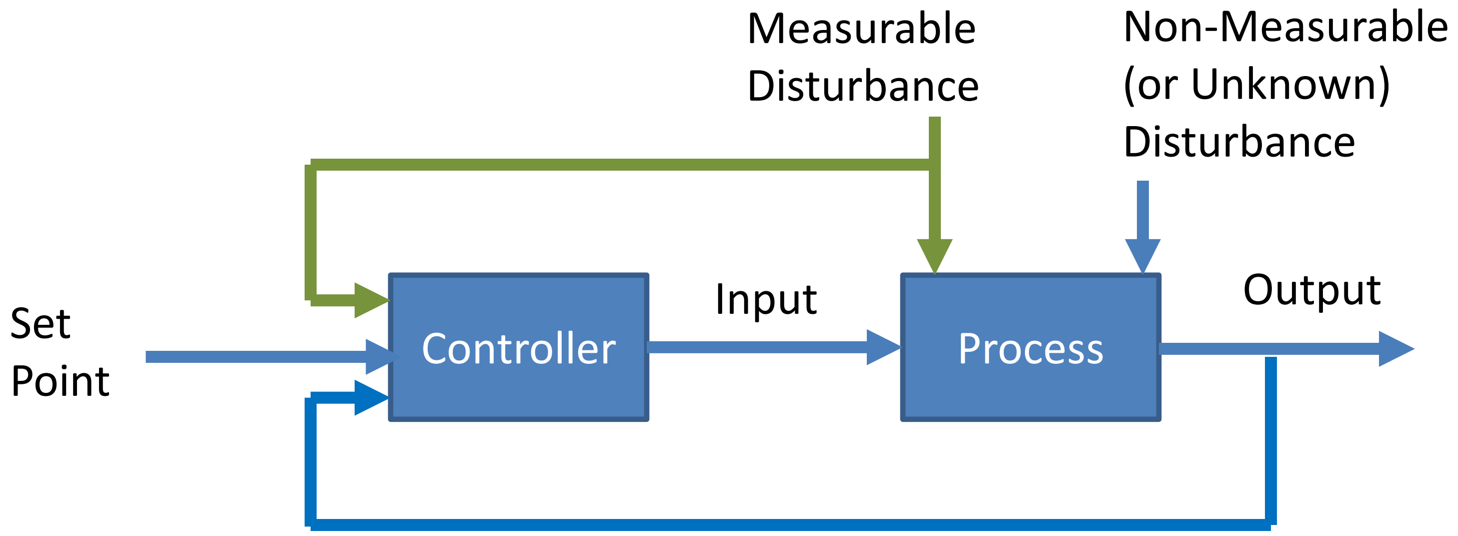
System Engineering as a Closed Loop With Feedforward System



- DoD generally executes the Feedback Loop well enough:
  – Technical Reviews
  – Milestone Decisions Points
  – Stage Gate process
  – Entry and Exit Criteria

Defense Acquisition University

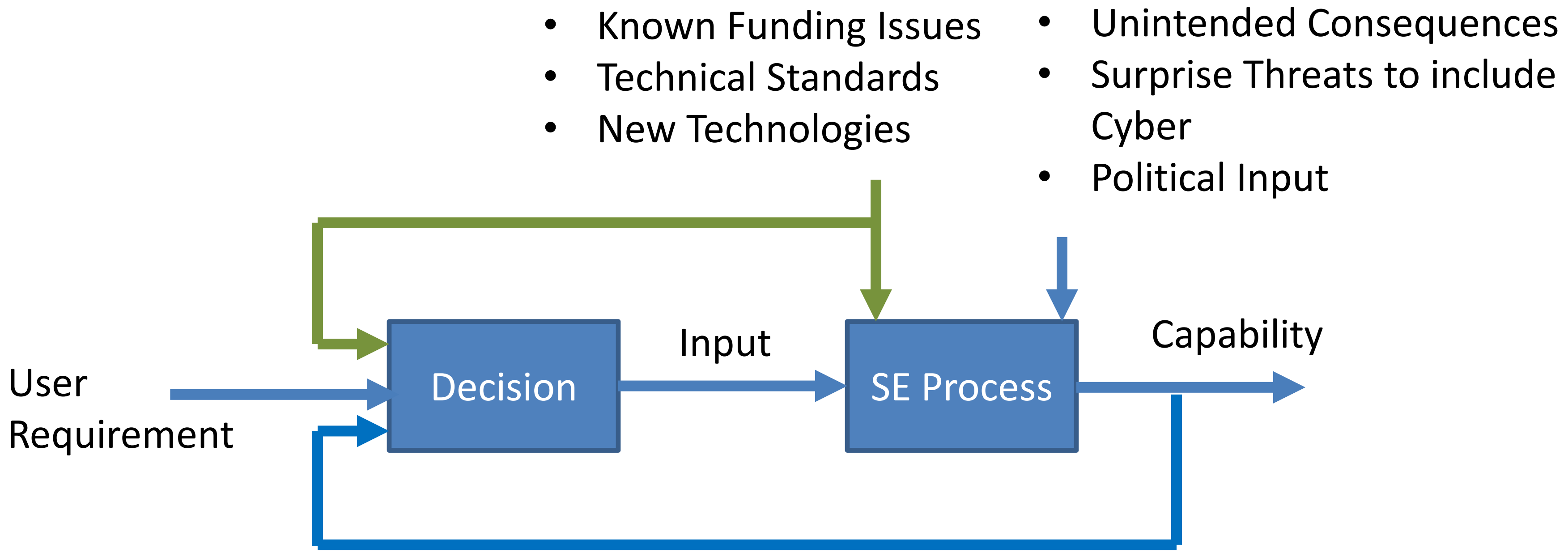
System Engineering as a Closed Loop With Feedforward System



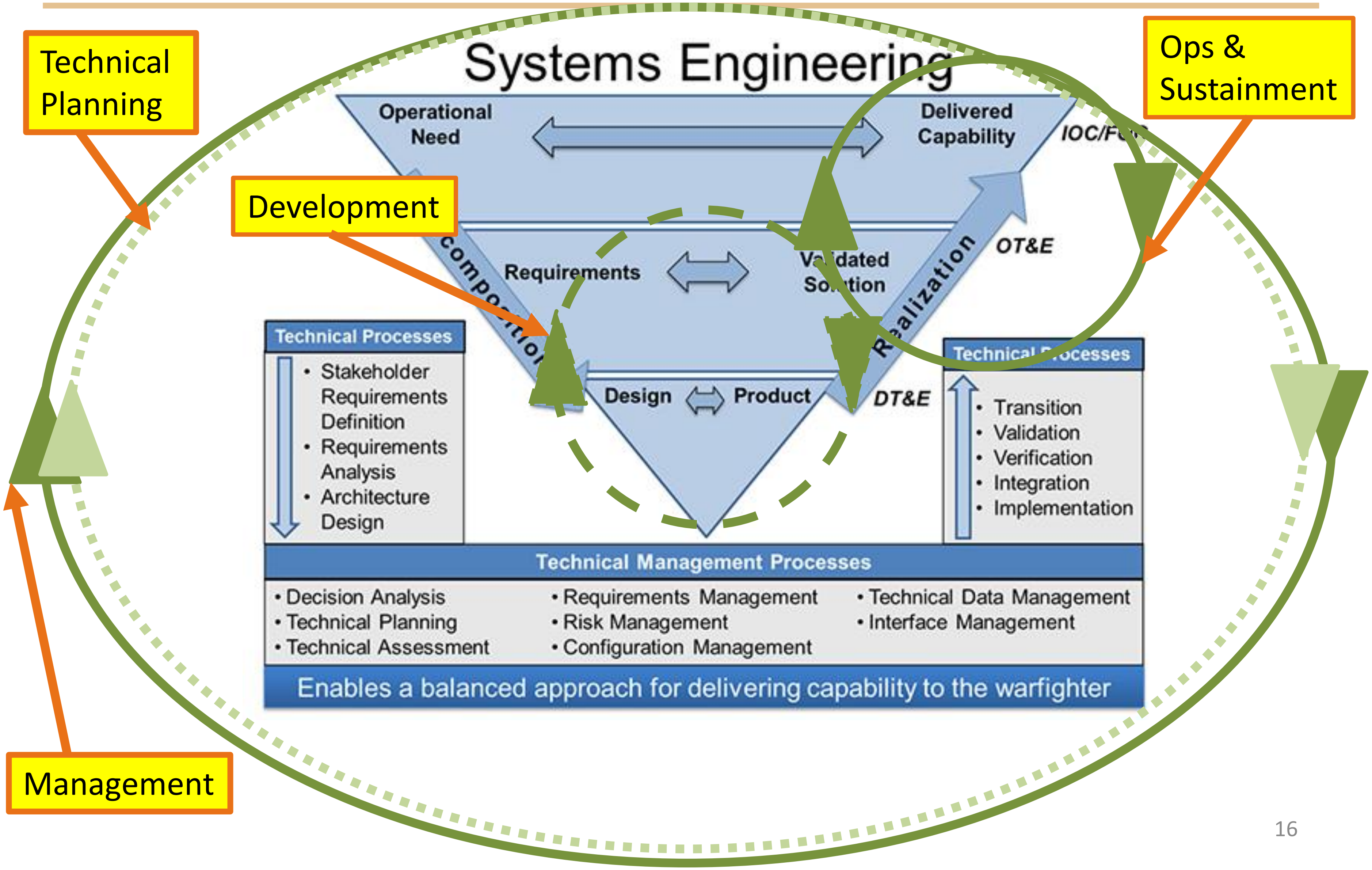
- But what can DoD leverage the goodness of
- How can DoD leverage Feedforward to accelerate our programs and remain malleable to changing user needs?

- There is opportunities to shift towards reaping the benefits of Feedforward!

System Engineering as a Closed Loop With Feedforward System

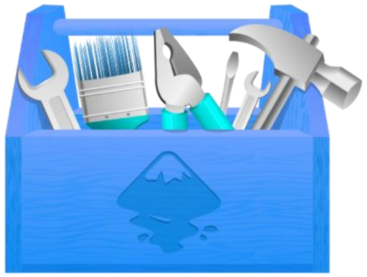


How Do We "Feedforward" the Systems Engineering Model?



Defense Acquisition University

Feedforward Toolbox


Open Systems Architecture

Agile Management Concepts

JCIDS IT Box Concepts

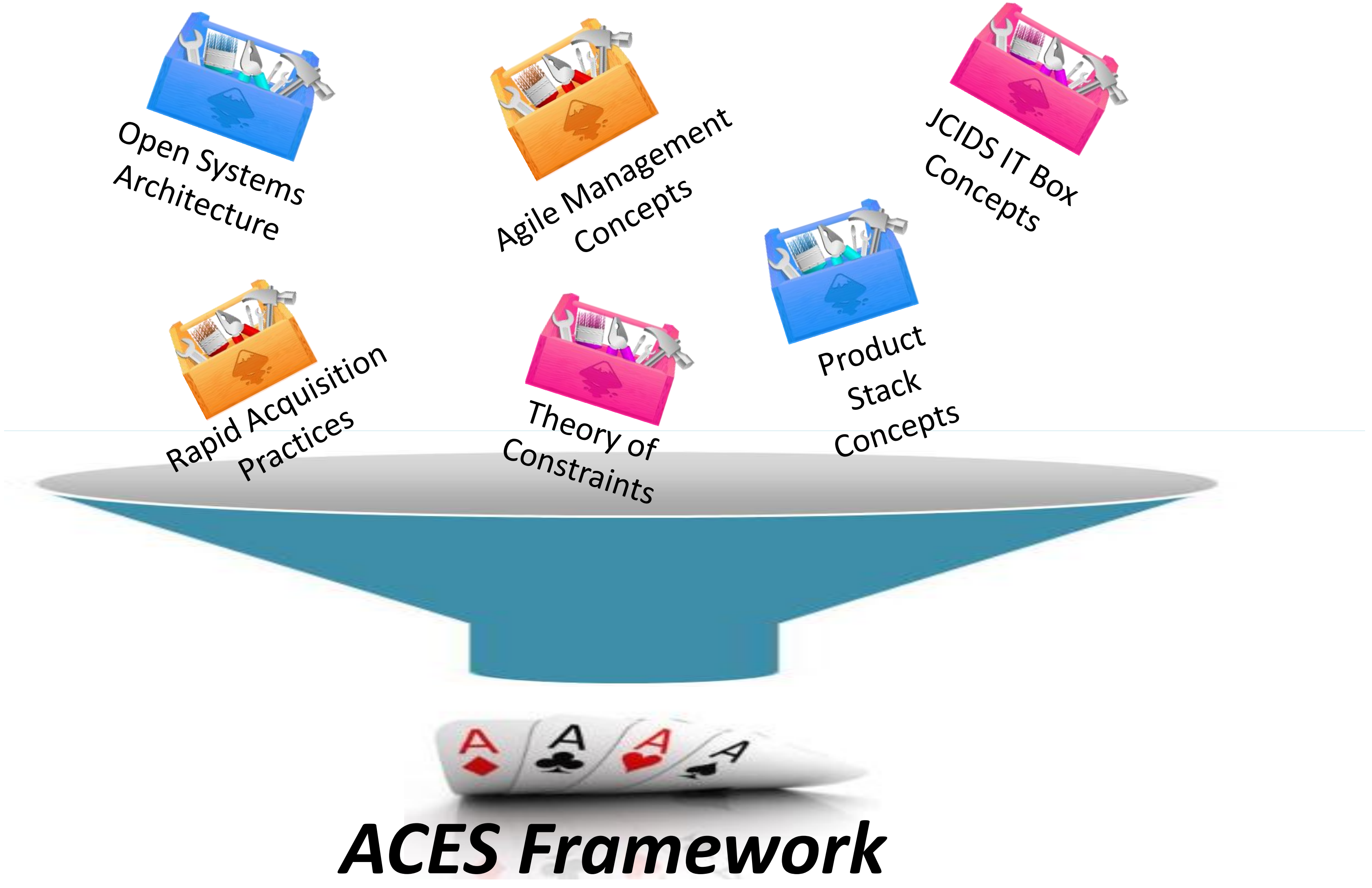
Rapid Acquisition Practices

Theory of Constraints

Product Stack Concepts

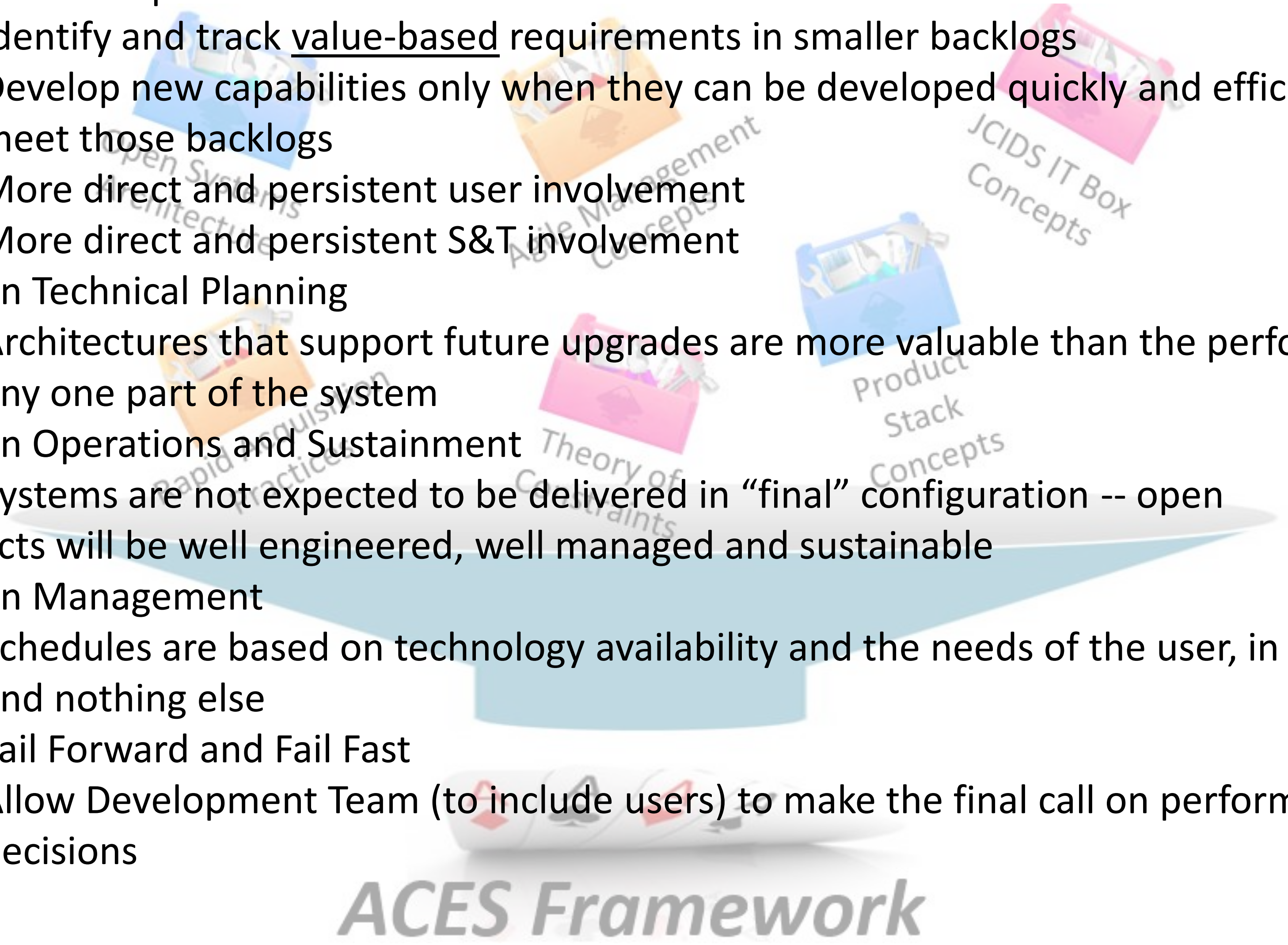
Others

Feedforward Toolbox



How Can We Put the "A" into Systems Engineering?

- Agile in Development
  - Identify and track value-based requirements in smaller backlogs
  - Develop new capabilities only when they can be developed quickly and efficiently to meet those backlogs
  - More direct and persistent user involvement
  - More direct and persistent S&T involvement
- Agile in Technical Planning
  - Architectures that support future upgrades are more valuable than the performance of any one part of the system
- Agile in Operations and Sustainment
  - Systems are not expected to be delivered in "final" configuration -- open
- Products will be well engineered, well managed and sustainable
- Agile in Management
  - Schedules are based on technology availability and the needs of the user, in that order and nothing else
  - Fail Forward and Fail Fast
  - Allow Development Team (to include users) to make the final call on performance decisions


Open Systems Architectures
Agile Management Concepts
JCIDS IT Box Concepts
Product Stack Concepts
Rapid Acquisition Practices
Theory of Constraints
ACES Framework

Summary

- Taking a different look at the Systems Engineering Process Model
  - Control Systems
  - Focus on Feedforward vs Feedback
- Finding ways to insert Agile concepts
  - With the Toolsbox we have now

Acquisition:

It's not Rocket Science

It's harder

"Comment form acquisition class at NRO"

Back-ups

Problem / Issues

- The current Defense Acquisition systems are designed around deliberate point solutions to specific user requirements (inputs).
- Issues in funding and requirements stability can extend the length of programs
- Interoperability can limit both initial designs and technical refresh
- The use of COTS has proven not to increase the passé of development and delivery
- Our Acquisition processes were developed and design around traditional large scale weapon systems development (Ships, Fighters, Tanks, etc.)
- The current systems engineering processes were not based on software systems.

Many inside and outside of the Acquisition community have questioned the ability of the DoD to develop high tech systems in a timely manner.

Requirements

- One of the largest issues that has been identified with the failures of government acquisition is that requirements, are not
  - Stable
  - Complete
  - Achievable
  - Well matched with threats
- Changes in requirements over the course of programs have been shown to dramatically affect program cost and schedule
- Traditional Acquisition programs attempt to have a complete and detailed set of requirement before starting to develop.

Given the continuously changing nature of requirements for DoD systems the traditional requirements process does not support rapid or agile development

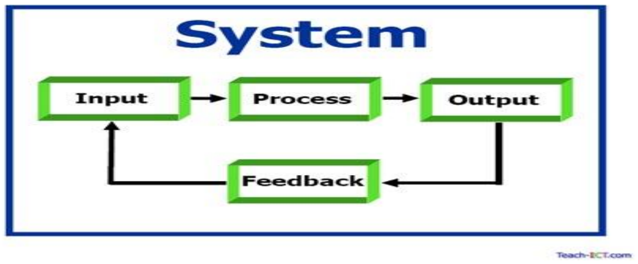
Constraints (schedule, cycle time)

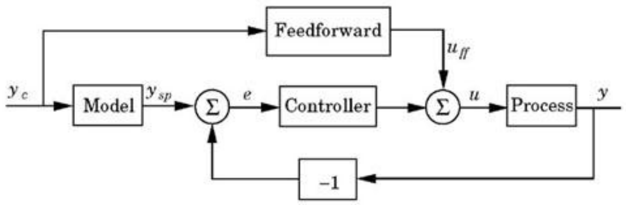
- Funding restraints
- Pace of threat changes in a complex environment
- Pace of technological change
- Ability to field new capabilities
- Ability to adequately test new capabilities before fielding to warfighters

The Key Constraint should be the time needed to train the warfighters in new the new capabilities

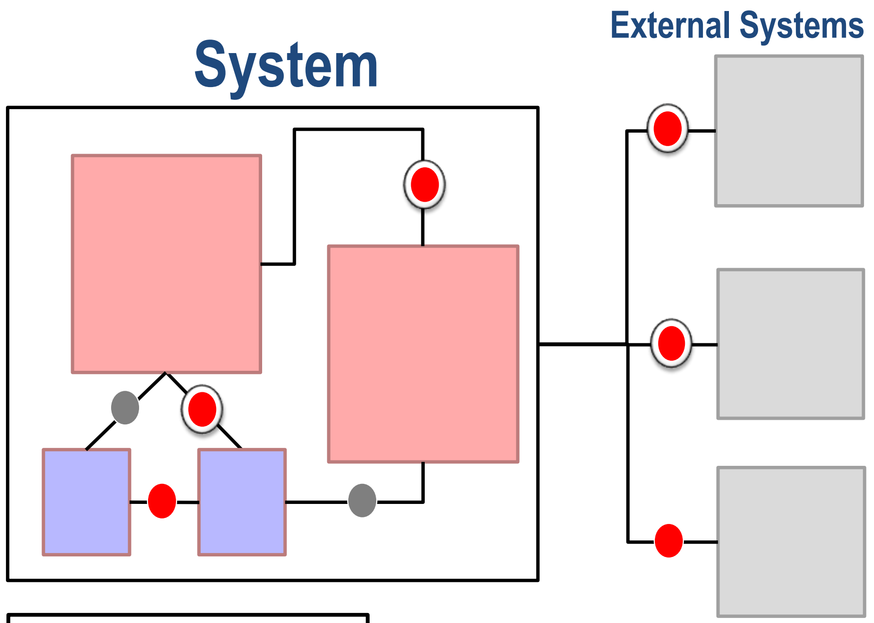
Feedback vs. Feedforward (in control vs. out of control)

- The current system is very much a feedback control system. This means that system uses significant levels of oversight and review to control the process and is managed from above.
- New systems designed to be a feedforward will exhibit significantly different behavior. This means that decisions made at the lowest level will amplify themselves creating the possibility for both rapid failure and rapid development of high quality products
- Feedforward systems can seem out of control, that's the idea. For truly rapid, innovative systems development we need to maintain visibility not control.





Open Systems Architecture



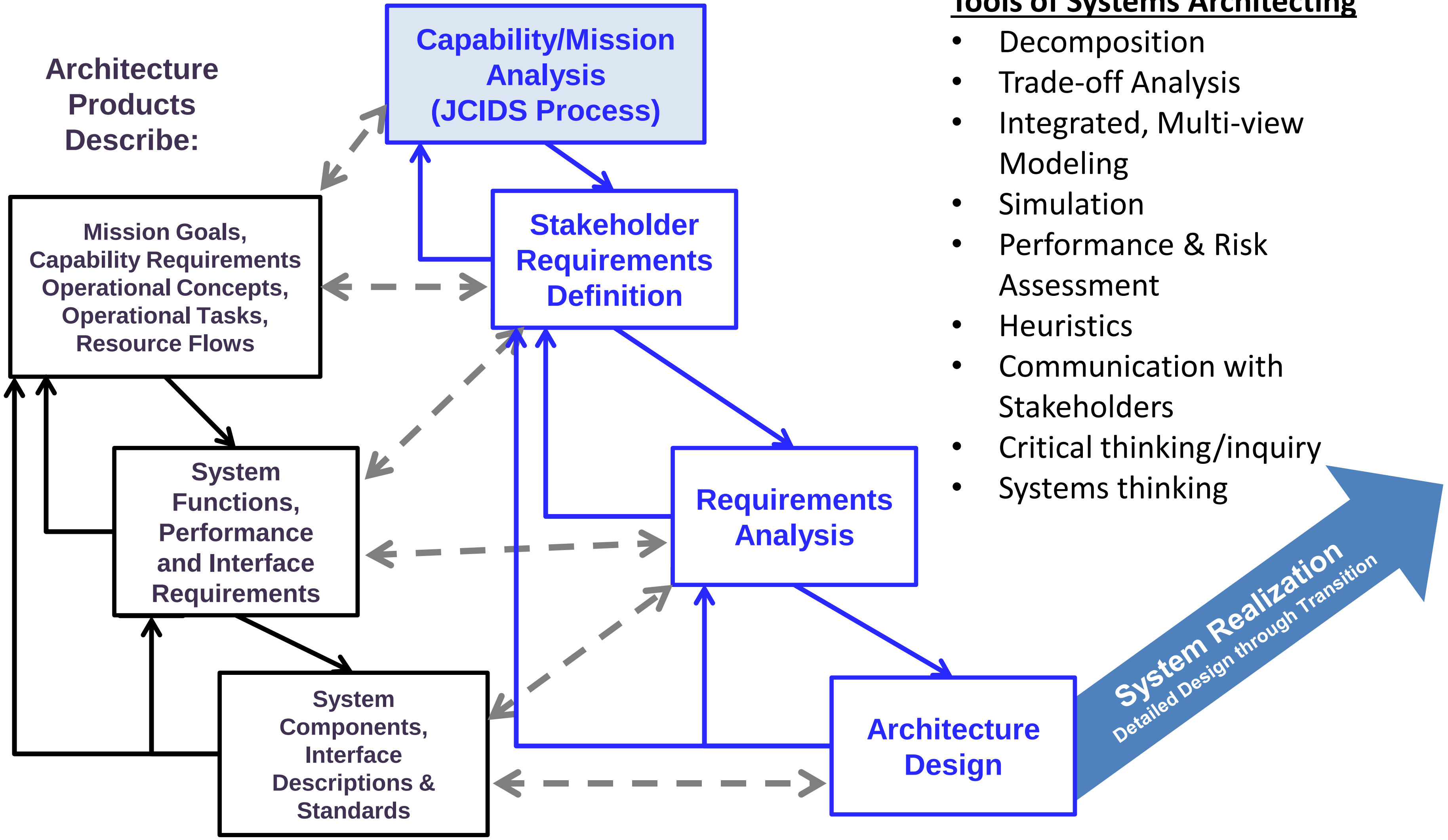
- ● Interface
- 🔴 Key Interface
- 🔴 Key Interface using Open Standard

Limited use of proprietary interfaces

Open Systems Architecture (OSA) uses modularity and defined/ published boundaries to support a business model that facilitates competition and prevents vendor lock.

- **Employ modular design**
  - **low coupling, high cohesion**
- **Designate key interfaces**
  - **Which interfaces will frequently need to change?**
  - **Which interfaces impact future re-procurement/ competition?**
- **Develop Business Case for use of open interfaces and acquisition of technical data.**
- **Use Open Standards**
  - **Published, widely supported, consensus based standards**
- **Certify Conformance**
  - **Develop verification & validation plans to confirm "openness"**

Architecture and the SE Design Process


Architecture Products Describe:
Capability/Mission Analysis (JCIDS Process)
Stakeholder Requirements Definition
Requirements Analysis
Architecture Design
Mission Goals, Capability Requirements Operational Concepts, Operational Tasks, Resource Flows
System Functions, Performance and Interface Requirements
System Components, Interface Descriptions & Standards
System Realization
Detailed Design through Transition

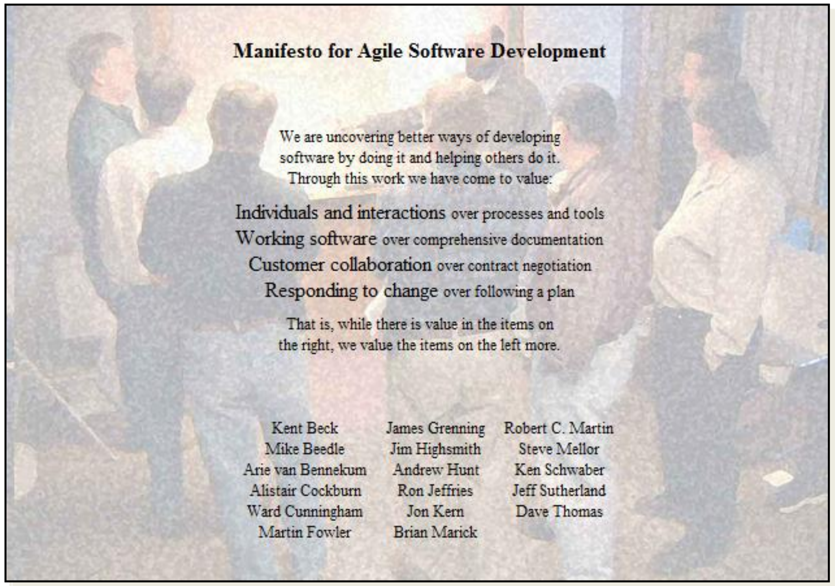
Tools of Systems Architecting

- Decomposition
- Trade-off Analysis
- Integrated, Multi-view Modeling
- Simulation
- Performance & Risk Assessment
- Heuristics
- Communication with Stakeholders
- Critical thinking/inquiry
- Systems thinking

Development Aspect

Agile Manifesto

- The foundational document for Agile software development
- Signed by 17 software developers in Feb 2001
- Core Values
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan



Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

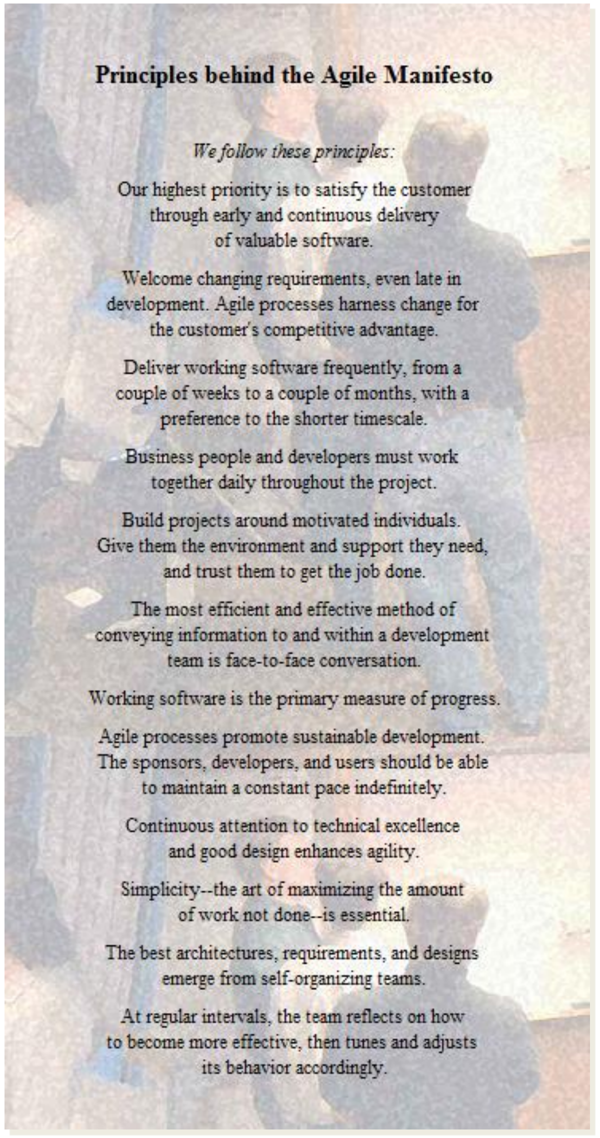
Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

http://agilemanifesto.org/

Defense Acquisition University

12 Principles of the Agile Manifesto

Development Aspect



Principles behind the Agile Manifesto

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity--the art of maximizing the amount of work not done--is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

1. Continuous delivery of valuable software
2. Welcome changing requirements
3. Deliver working software in weeks/months
4. Work together daily
5. Build projects around motivated individuals
6. Face-to-face conversation
7. Working software is the measure of progress
8. Promote sustainable development
9. Good design enhances agility
10. Simplicity is essential
11. Self-organizing teams
12. Reflect on how to become more effective

http://agilemanifesto.org/

Defense Acquisition University

Agile Software Development Methodologies

Development Aspect

- Scrum
- eXtreme Programming (XP)
- Dynamic Systems Development Method
- Rapid Application Development
- Crystal
- Kanban
- …

WE'RE GOING TO TRY SOMETHING CALLED AGILE PROGRAMMING.
www.dilbert.com scottadams@aol.com
THAT MEANS NO MORE PLANNING AND NO MORE DOCUMENTATION. JUST START WRITING CODE AND COMPLAINING.
© 2007 Scott Adams, Inc./Dist. by UFS, Inc.
I'M GLAD IT HAS A NAME.
THAT WAS YOUR TRAINING.

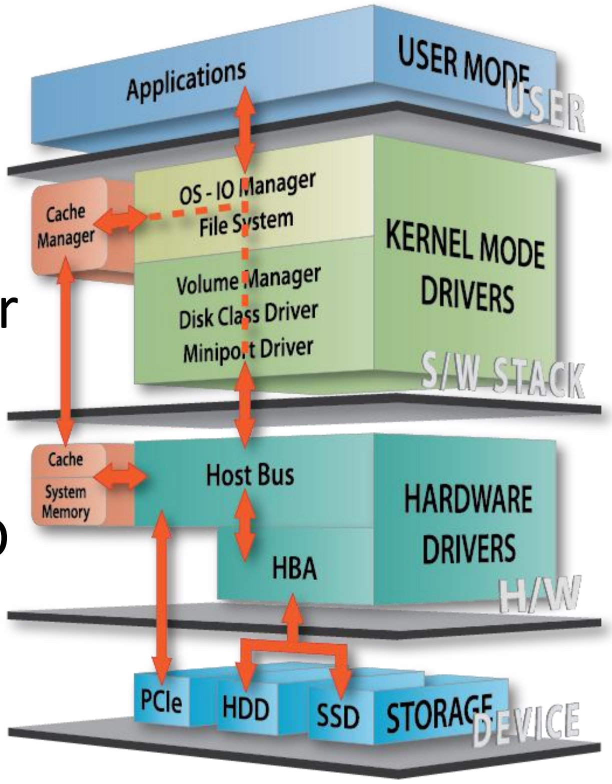
Rapid Acquisition

- Traditional acquisition:
  - Future focused
  - Very structured process
  - Evolved requirements
  - Analysis of alternatives
  - Lengthy development
  - High visibility on program
  - Large investment

- Rapid acquisition:
  - *Now* focused (meet immediate warfighter needs)
  - More *ad hoc* process
  - Broad requirement
  - Quick assessment of alternatives
  - Limited development
  - High visibility on results
  - Limited investment

Product Stack concept

- Agile development can draw many things from commensal systems development approaches
- Architectures need to be structured to change out different elements on differ schedules based on the availability of new technology.
- Cloud and SaaS can be used in the DoD to provide flexibility in solutions architectures.
- A flexible upgradable architecture can be used to solve a wide range of application problems in a cost and schedule efficient manner.



calypsotesters.com

Understanding Value (Theory of Constraints)

- The only Value in the system is the value to the customer
- Bottlenecks govern both throughput and inventories
- An hour lost at a bottleneck is an hour lost for the entire system
- An hour saved at a non-bottleneck is a mirage
- Waste in the system drives the cost
- From the customers (DoD / Warfighter) stand point the only cost is opportunity cost (how often am I going to get a system upgrade and how good will it be)

Key implications

- The current system is very much a feedback control system. This means that system uses significant levels of oversight and review to control the process and is managed from above.
- ACE'S is designed to be a feedforward system. This means that decisions made at the lowest level will amplify themselves creating the possibility for both Rapid failed program and

Lessons Learned from Agile

Lesson	How to Apply
1. Welcome new requirements	If requirements are defined correctly in classes and categories, new requirements are really clarifications.
2. Close and continues relationship with users	Developers (including the government team) need to understand product, task and environment (embed with users)
3. Only some architectures support Agile in development and sustainment	Successful architectures need to be defined by the product and the lifecycle, not the vender development approach.
4. Prioritization is a must	Traditional Specifications tend to end up as a long list of must dos. Prioritization allows for good design.
5. Communication must happen and often between the wright players	The users and developers must communicate and prototype products.
6. Team and Product before Organization and Process	Process and leadership must not be allowed slow down good design and good decision

Defense Acquisition University

Lessons Learned from Rapid Acquisition

Lesson	How to apply
1. Requirements directly from users, and users needs	Developers need to be in the field collecting threat data in real time as development is going on.
2. Early, continues, and innovative testing	Testing continues to be critical, to producing good products, but can be conducted at different times.
3. Small, highly skilled teams providing visibility to leadership, but not being managed from above	Small high capability teams can make critical decisions quickly and move to the next task
4. Collaboration between users, venders, and government	The government engineers need to doing design and development along side of the venders and users
5. High risk development, mitigated by mix of mature products and highly resourced new development supervised by top technical experts	By knowing the details of the underlying technology and building the right architecture development and integration risks can me actively managed.

New Acquisition principles

- It is about creating value for the customer by investing resources (house flopping model). Return on investment.
  - In and out fast
  - Add maximum value given available resources
  - Value is assessed based on capabilities that the customer can use
  - Product and Schedule is based on prioritized set of capability blocks not requirements
- Only develop new products or systems when they can be developed and delivered quickly, efficiently and provide value to the end customer in teams of capability

Defense Acquisition University

Functional Requirements for Agile DoD Acquisition

- New Systems and new capabilities be made available to the warfighter based on their ability to effectively use them.
- Government contracting will not inhibited the rapid development.
- The DoD will buy value blocks of capabilities in specific areas defined over the course of the development by integrated teams of government and contractor developers.
- The architectures of new systems will be evaluated first for enhancement to future upgrade and maintained.

Change in Mindset

- Success is measured by the core team, (users and developers only)
- Failure is expected and not-attributed
- Systems are not expected to be delivered in final configuration
- An architecture that supports future upgrade is more important than the performance of any one part of the system
- Schedules are set based on technology availability and the needs of the user, in that order and based on nothing else.
- The decision about the maturity of the technology belong to the engineer.
- Managing technical development requires direct involvement in all aspects of the

Defense Acquisition University

New Manifesto

1. There are no pure managers, if you do not have a technical role in the project you are not part of the team, just support staff.
2. Parallel all activities that can be parallel (if it requires iteration, review, or update it can be done in parallel).
3. Requirements are developed as needed to characterize capabilities needed by and delivered to users.
4. Its alright to fail, and it alright to decide that the program should be shut down.
5. New systems can be delivered fast and effectively only when the technology and the users are ready
6. The first solution will not complete and will need to be updated, therefore the upgrade and modification architecture is more important that the first solution

ACES Key Systems Engineering principles

1. Use of state of the art tools
   - Automated real time requirements definition
   - Deploy and update prototypes in the field
   - Development teams deploying sensors and other data collection tools to field to test and gather new requirements
2. Agile principles
3. Open systems principles
4. Just in time requirements
   - Requirements will be defined as they are needed
   - Interfaces first
   - Performance, iteratively as capabilities are available
5. Continues testing
6. Technical management best practices
   - Teams lead by technical experts also doing development
7. Lean development principals
   - Value is defined by the operator
   - Remove non-value added processes

Architectural Concepts

1. Architectures will assume that the first solution will not be final and will not be the best solution.
2. Architectures will support operations while development is ongoing.
3. Architectures will support continues operations while different parts of the systems is being tested.
4. Architectures will support remote operations of parts of the system.
5. Architectures will support parallel development of different parts of the system.

ACES Key Contracting principles

- We will contract for numbers of capability Value blocks, not deliveries of specific systems with specific per-determined requirements.
- New elements of "value"
  - ➢ Each element of requirement and delivery will be valued in units
  - ➢ A particular delivery might be 3.2 units of value, and be contracted and payed based on the value
- Commodity based developmental contracting. Scope controlled by the development team and end users

ACES Key Program Management principles

- Schedules will be based on availability of capability and the ability of the customer to accept delivery of new systems
- Program management existed to coordinate efforts of the technical teams and provide resources

What has to change to make this work

If we want to agile systems that follow the ACES processes then we need to change some basic concepts:

1. The users and the developers need to be directly connected
2. Get managers out of the decision making process
3. Government decision makers need to part of the development team
4. Smaller high capability teams of users, and government and contractor developers
5.