Defense Acquisition University

An EA Process Template for PMs

Robert M. Flowe, Major, USAF
Defense Acquisition University

This presentation was given 28 October, 2002 at the Evolutionary Acquisition and Spiral Development Conference hosted by the International Quality and Productivity Center.

Major Robert M. Flowe currently serves as Professor of Systems Acquisition Management at the Capital and Northeast Campus of Defense Acquisition University, Fort Belvoir, VA. In this capacity, Major Flowe teaches Level III certification courses, as well as serving as Course Manager for the Intermediate Software Acquisition Management course (SAM-201). As Course Manager, Major Flowe maintains and delivers the intensive 2-week SAM-201 course throughout the US. Major Flowe also develops and delivers tailored software acquisition training to the Australian DoD and other customers.

Prior to coming to DAU, Major Flowe served as an Operations Research Analyst in the Office of the Secretary of Defense, Program Analysis and Evaluation (PA&E), as Cost Analysis Improvement Group (CAIG) analyst for a variety of programs across all Services. Major Flowe's particular interests involve the acquisition management implications of emerging software acquisition strategies, such as Evolutionary Acquisition, System-of-Systems (SoS) acquisition, and commercial-off-the-shelf (COTS)-intensive systems.

Other EA-related presentations by Major Flowe include:

•"Cost Estimating Implications for Evolutionary Acquisition and Spiral Development", presented at the 34 Annual DoD Cost Analysis Symposium at http://www.ra.pae.osd.mil/adodcas/Presentations%202001/EA-SD%20Cost-Flowe.pdf

•Integrating Evolutionary Acquisition with PPBS: provided in support of OUSD(AT&L) initial training on EA at: http://www.ncat.com/ea/Revised_PPBS.pdf

DAU NE-TED/SM

9820 Belvoir Road, Suite G-3

Fort Belvoir, VA 22060-5565

Phone: (703) 805-5419; DSN 655-5419; FAX: (703) 805-5082; DSN 655-5082

Email: robert.flowe@dau.mil or flowe_robert@hotmail.com

Agenda

Defense Acquisition University

- Purpose
- Definitions: Evolutionary Acquisition & Spiral Development
- Why an EA Process Template?
- Key Features of this Template
- Discussion

Purpose

Defense Acquisition University

- Purpose: To encourage a dialog within acquisition community regarding EA issues
- Process: To share insights on the EA management and related issues, in an open, constructive manner
- Goal: To identify and elaborate issues, insights, ideas of value to the acquisition community (government and industry)
- Key insights and conclusions to be considered for incorporation into curricula, and/or publication

1/14/2003 3

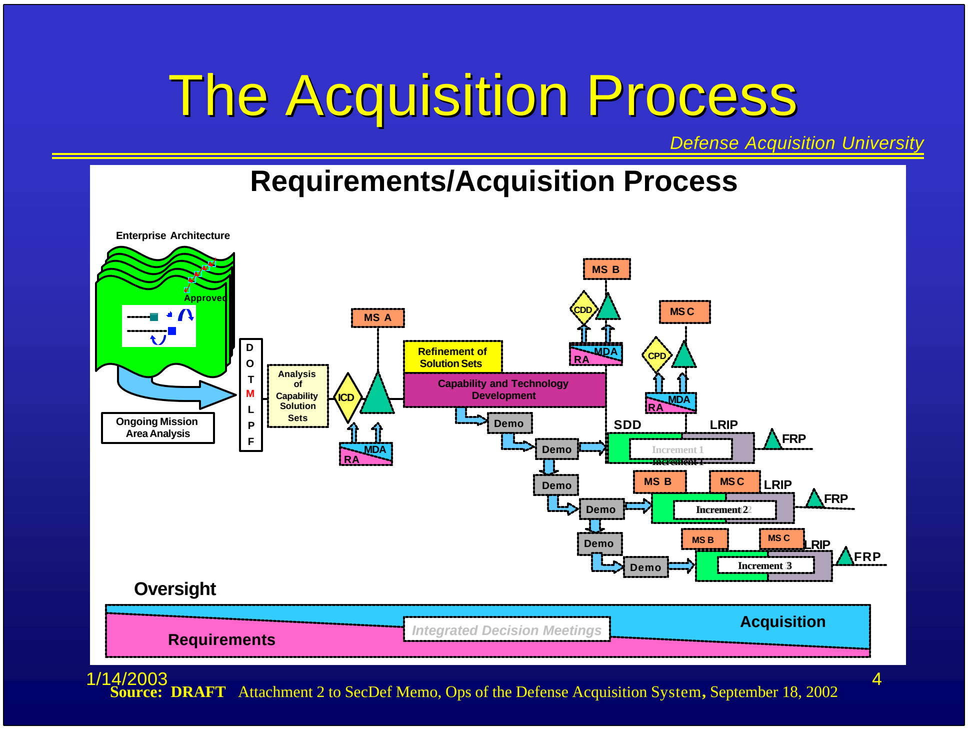
With the recent emphasis on Evolutionary Acquisition and Spiral Development from the DoD Acquisition leadership, there is great interest within the acquisition workforce regarding what these terms mean, and how to apply these concepts to improve the acquisition process.

However, there no definitive "canon" on what these terms mean, and how best to apply these principles.

DAU is participating with OSD and industry to develop training on EA and SD, with the initial focus on the "what" and "why" of EA. Developing awareness within the acquisition community of what EA is, and why the Department is interested in pursuing it.

This presentation is intended to stimulate dialog on the "how" of EA, which I feel is the more challenging and meaningful question.

This presentation reflects my views and opinions, which are not necessarily the views of DAU or the Department of Defense.



This is “hot off the presses”—part of the rapid iteration of the “interim guidance” replacing the rescinded DoD 5000.2 documentation. In my opinion, this chart is significant in that it depicts a continuum between Requirements and Acquisition. By implication, the acquisition process has a major stake in the requirements process. Finally, we’re getting away from the idea that the acquisition process starts at the ORD.

Another important take-away is that it encourages, in my view, the consideration of non-material alternatives as part of the overall process of satisfying User needs. This has always been a blind spot in the acquisition process—that we’ve only narrowly considered the material alternatives without much regard for how that impacts the user in the non-materiel aspects.

This opens the door to the overdue and very necessary dialog between the “Requirements Community” and the acquisition community. There are new concepts in this diagram which need explanation:

DOTMLPF: this is a term that arises from the requirements community, and represents the spectrum of modalities to satisfy operational needs—materiel solutions being only one—namely: Doctrine, Organization, Training, Materiel, Leadership, Personnel, Facilities.

RA/MDA: Requirements Authority/Milestone Decision Authority—this represents the shared responsibility of the requirements and acquisition decisionmakers regarding what requirements are validated, and how they are to be satisfied.

ICD: Initial Capabilities Document—the original articulation of the capabilities required by the user, expressed in a time-phased manner.

CDD: Capabilities Development Document—prepared at the conclusion of Technology Development, as a precursor to System Development & Demonstration and Milestone B.

CPD: Capabilities Production Document—necessary for entry into Production and Deployment phase, and Milestone C.

Definitions: EA

Defense Acquisition University

Evolutionary acquisition is DoD's preferred strategy for rapid acquisition of mature technology for the user. An evolutionary approach delivers capability in increments, recognizing, up front, the need for future capability improvements. The success of the strategy depends on the consistent and continuous definition of requirements and the maturation of technologies that lead to disciplined development and production of systems that provide increasing capability towards a materiel concept.

1/14/2003 Source: **DRAFT** Attachment 2 to SECDEF memo, *Operation of the Acquisition System*, dated 18 September 2002 5

Key concepts:

Incremental delivery of capability with future improvements

Continuous requirements definition

Maturation of technologies

Definitions: SD

Defense Acquisition University

A process in which a "…desired capability is identified, but the end-state requirements are not known at program initiation. Those requirements are refined through demonstration and risk management; there is continuous user feedback; and each increment provides the user the best possible capability. The requirements for future increments depend on feedback from users and technology maturation."

1/14/2003 Source: DRAFT Attachment 2 to SecDef Memo, Ops of the Defense Acquisition System, September 18, 2002 6

Key points:

End-state requirements not known at the outset.

Demonstration and risk management

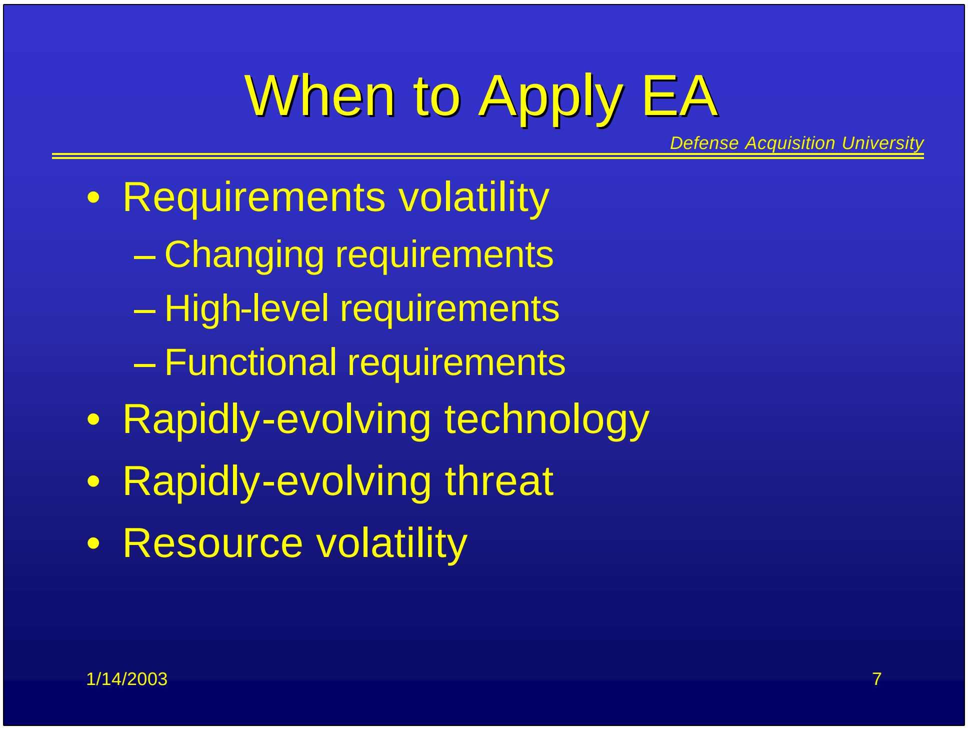
Continuous user feedback

Issues:

Establishing a clear distinction between Evolutionary Acquisition and Spiral Development

Providing clear guidelines for applying EA and SD

Possible confusion over/conflict with established SD principles (in Software community)



Joint Logistics Commander's EA Guide (1998) identified 6 conditions wherein EA was an appropriate approach:

1. Requirements Uncertainty: EA is favored in environments where requirements are uncertain or are likely to change over the development period. This dynamic was observed in C4I systems development over the past decades.
2. Technology change: EA is favored when technology is likely to offer significant increases in capability over the projected development timeframe.
3. User Involvement:
4. Schedule Urgency
5. Funding Instability
6. Other Constraints, such as interoperability, status of current system, availability of commercial products, etc.

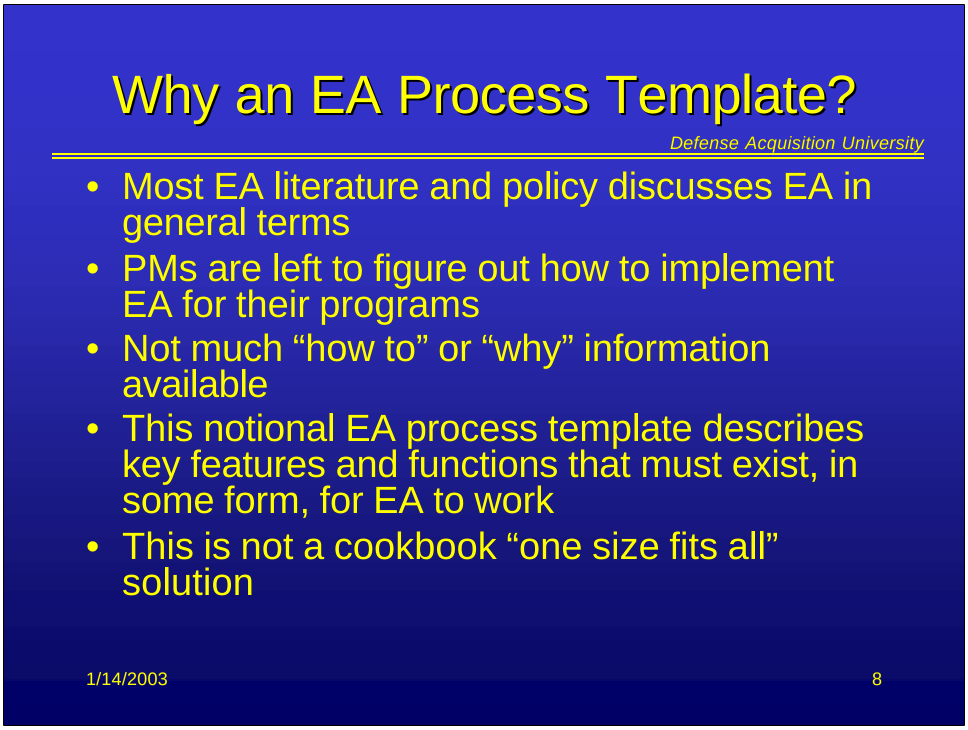
Classically, in the single-step to full capability, requirements, technology, threat, or resources would change over the development period, requiring costly and inefficient re-planning and redesign.

In its simplest terms, EA is an acquisition strategy that is less vulnerable, yet more responsive to these sorts of changes. By so doing, inefficiencies due to unplanned rework, missed schedules, blown budgets, and crisis management are reduced.

This is not to say that EA is easy or cheap to do.

It's kind of like how Churchill described democracy: "It's absolutely the worst form of government, save all others"

So EA is a difficult and expensive way to acquire capability, but is ultimately less difficult and costly than any other method (under the conditions described above).



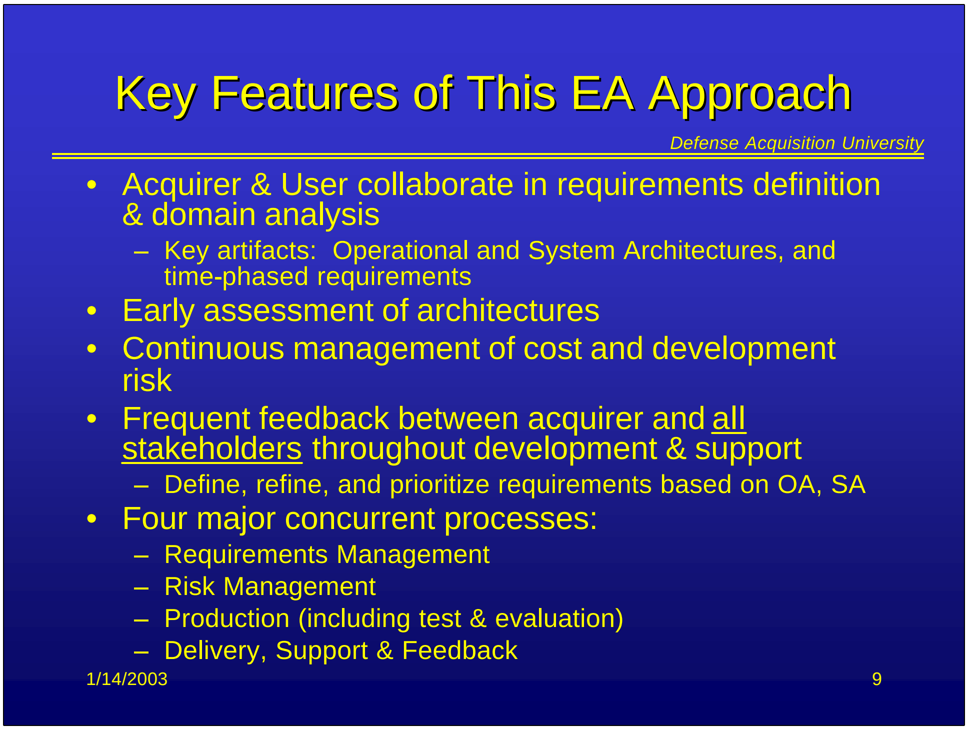
At this stage of the game, the big push is to get the word out to the acquisition community regarding the “what” of EA.

So the focus is on definitions, and the desired end state (faster delivery of capability to the warfighter), with not much on the “how”.

So this template is an attempt to break down the acquisition process into its component steps and describe how these steps are applied in the EA environment.

These steps are generally the same as in any systems engineering/systems acquisition process, but I will attempt to describe how these processes will change for EA.

The reason this is called a “template” is because each program will have to adapt these processes to fit their particular situation, so it’s not a prescriptive “cook book” where one can mindlessly apply the steps in sequence and expect success. But I hope the template will alert you to issues that, if successfully resolved, can improve your chances for success.



To implement EA according to this template, significant changes to the structure of, and workflow within the program office will be necessary. This will be reflected in the contract structure and strategy, and will also impact resource allocation and phasing.

To be successful, a much greater investment in establishing and maintaining relationships with users and other stakeholders will be necessary. This expanded participation will require substantial resources.

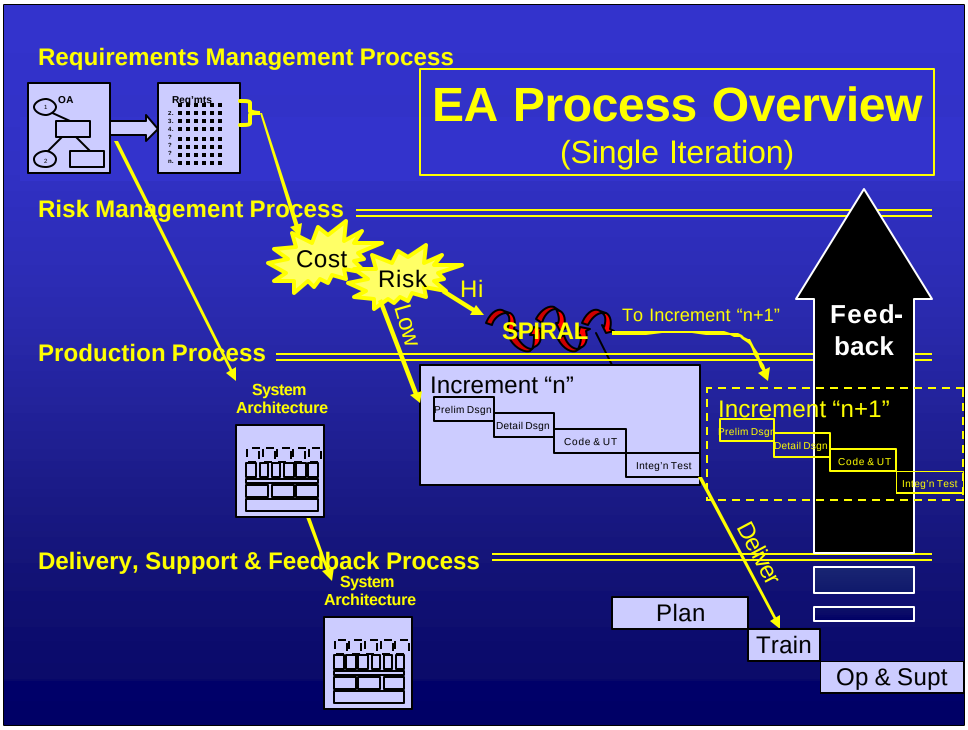
For the purposes of this discussion, I break out 4 major concurrent and highly coupled processes:

Requirements management—deals with eliciting, elaborating, validating, prioritizing, time-phasing, tracing and tracking requirements.

Risk management—deals with assessing implementation risk of each requirement, identifying the appropriate level of urgency, recommending appropriate mitigation strategies, and integrating the risk mitigation strategy with the other three concurrent processes.

Production—a highly efficient, optimized process for transforming risk-mitigated, high priority requirements into material solutions.

Delivery, Support, and Feedback—the process of transitioning the developed capability to the end user, ensuring the user is trained and equipped to use the capability, that the maintainer is trained and equipped to support the capability, and managing the incorporation of the user's feedback into the requirements process.



Here is one iteration of the EA process template. The four concurrent processes are show as zones, the timeline projects from left to right. Though it isn't explicitly shown, the systems engineering process is suffused throughout this template.

The key goal of the requirements management process is to ensure that a current, accurate, validated, prioritized listing of user-focused requirements is always available. Requirements include functional capabilities required, interoperability, security, safety, reliability, maintainability, environmental compliance, etc. etc.

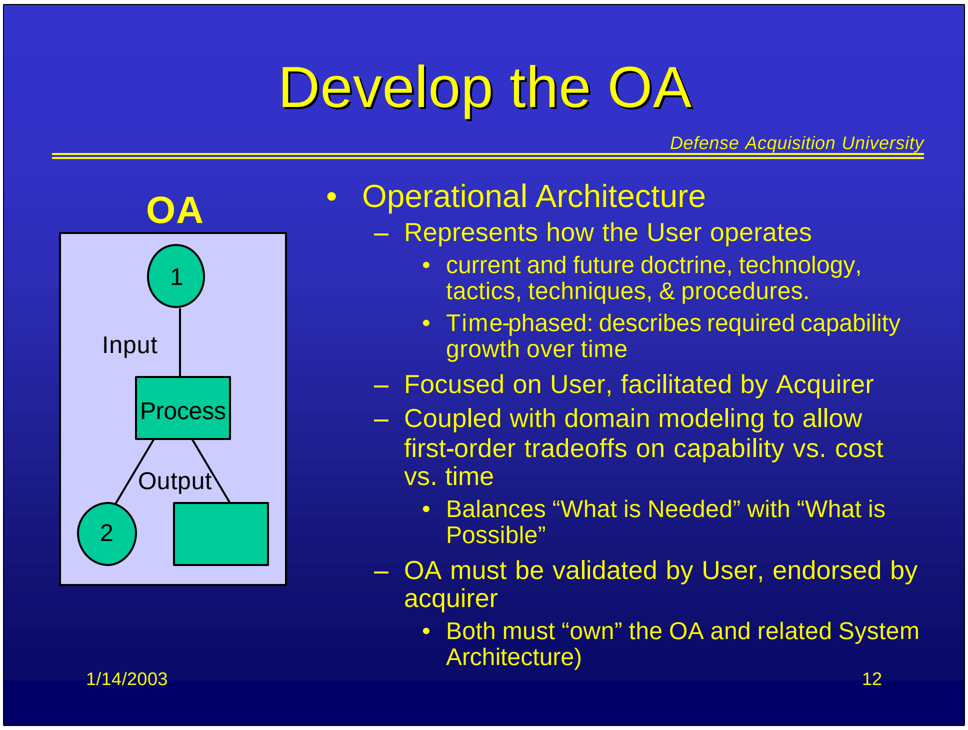
The key goal of the risk management process is knowing the cost, schedule, and implementation risk of each requirement, what risk mitigation steps should be undertaken to reduce the implementation risk, the likelihood that a given requirement can be incorporated at low risk within a given increment.

The key goal of the production process is to efficiently transform the requirements allocated to a given increment into an operationally robust, suitable, effective, supportable capability on time, on schedule, and within budget.

The key goal of the Delivery, Support and Feedback process is to ensure the Users are prepared to accept the capability, to deliver the capability, and to ensure the capability, once delivered, will be supported. This process facilitates feedback from the users and other stakeholders into the requirements process.

Implementing such a strategy would have to use a modular contract strategy.

Process:
Requirements Management



Why should the acquirer concern his/herself with the development of the OA?

Simply put, the OA is the origin of all requirements that must be satisfied either with materiel or non-materiel solutions.

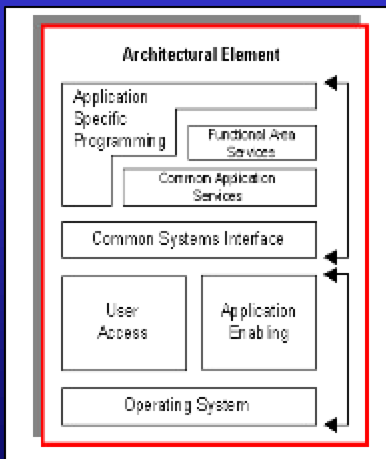
Without access to some representation of how the User operates, there is no objective means to determine the priority of requirements, determine the time-phasing of requirements, make the materiel vs. non-materiel solution decision, understand how to derive detailed requirements from top-level requirements, etc.

The OA can form the basis for a common understanding of the capability required. This reduces the likelihood of miscommunication between user and acquirer. This simplifies the development of operational test requirements.

There are automated tools available to facilitate the creation of the OA, one of which being the IDEF (Integrated Definition) process modeling methodology. IDEF is a family of methods that supports the modeling needs of an enterprise and its business areas. The most basic of these is IDEF0, which supports function modeling, wherein the user states "what I do" in a formal symbolic language, comprised of "inputs", "controls", "outputs", and "mechanisms", collectively referred to as ICOMs. More detailed models of interest include IDEF1, information modeling; IDEF1X, data modeling; and IDEF3, process description capture, each of which can provide valuable insights into the user's processes, and the requirements derived from them. A significant benefit of undergoing a rigorous IDEF modeling exercise, is that it helps illuminate irrelevant, redundant, or counterproductive elements of the user's processes. Thus, the IDEF process is a mechanism for process reengineering, an important "sanity check" to achieve before embarking upon a materiel acquisition. Without having this "sanity check", there is no guarantee that the technology investment will result in actual improvement in the user's performance.

Develop a Systems Architecture

Defense Acquisition University



- Conduct domain analysis & modeling concurrently with OA definition
- Establish, distribute the enterprise SA
- SA describes functional “boundaries” of the program, interaction rules, and key interfaces
- SA reflects goals of flexibility, scalability, interoperability, robustness, security, etc.
- Provides scalable framework within which operational capability will evolve
- Analysis & design of SA informs the development of lifecycle cost model

1/14/2003 Graphic: Barry M. Horowitz, Ph.D.ESC-TR-94-208, September 1994 13

Systems Architecture includes: “…the composition of hardware and software components, the structure that interconnects them, and the rules by which they interact.”

architecture is the main determinant of a system’s characteristics. The efficiency of the system, and thus its performance, depend on how the architecture handles resource utilization; architecture determines how the system sustains operations when parts of the system fail. The architecture also determines how maintainable the system is; that is, (1) how much effort is required to find and fix errors; (2) how easy it is to add new capabilities through software; and (3) how much is required to move the software to different computer hardware. Although they may be invisible to the user, these characteristics, which are all determined by architecture , are very visible to developers and maintainers who must modify and add to the operational capabilities of the system.

If the system is properly structured, then hardware components can be added or upgraded without expensive changes to the rest of the system. A good architecture allows a system designed to counter one threat to counter a different threat through localized modifications to the software that change the functional capability of the system or allow it to interoperate with other systems.

…

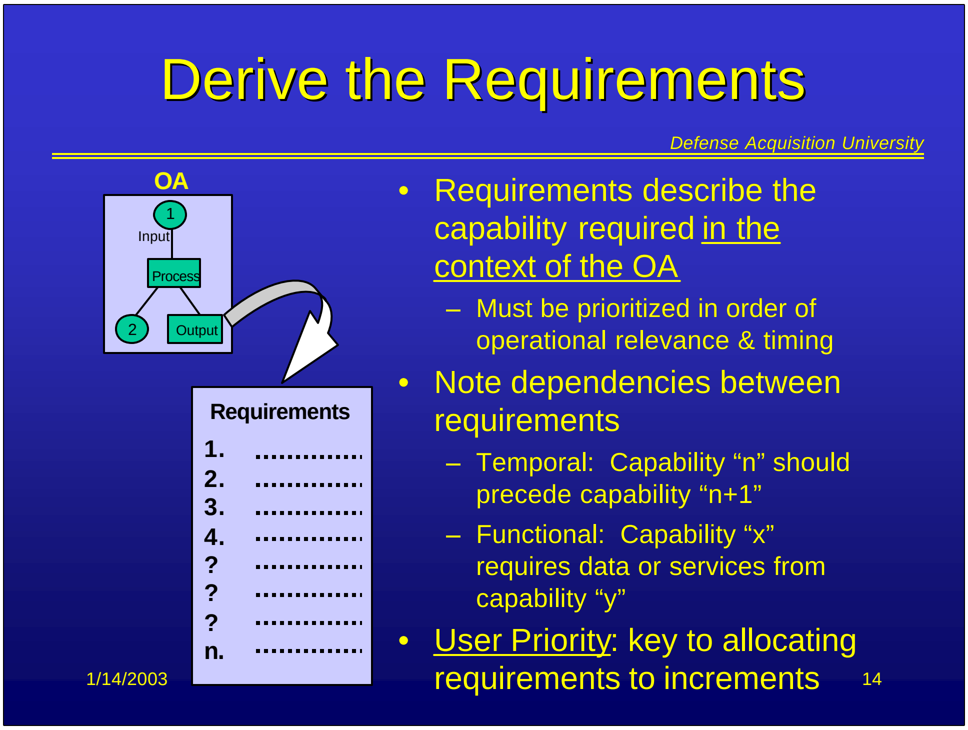
Barry M. Horowitz, Ph.D.
ESC-TR-94-208, September 1994

A substantial investment must be made in domain analysis and domain engineering so that the full and ultimate capability of the system can be defined, and the system architecture selected to enable that ultimate growth. Of particular concern at this point is the adoption of a commercial or open systems architecture. Plans to incorporate commercial-off-the-shelf (COTS) software and hardware will directly impact the selection of the systems architecture, and vice versa (think Windows versus Unix).

Once developed, the SA should be made known to the user community, the T&E community, and the S&T community, the security accreditation community, and others as applicable, as well as any potential industry partners that may wish to develop capability to integrate into the architecture.

The S&T community, in particular, should be advised of the systems architecture, so as they mature technologies, they can do so mindful of the integration requirements the SA will impose.

One of the important artifacts that will evolve with the development of the OA and SA is the C4I support plan. This document identifies all information exchange requirements between the elements of the system, and between the system and external entities.



System Requirements must always support, and therefore be derived from the operational requirements. Having developed the OA as a collaborative effort between the user and acquirer, much of the hard work in requirements development has been accomplished. The OA answers the perennial question: "Why do you need that?" Historically, before it became OK for the acquirer and user to collaborate on requirements development, the answer would typically be "Because I say I need it". This type of dialog, as you can imagine, did not make for productive Cost-as-independent-variable (CAIV) discussions.

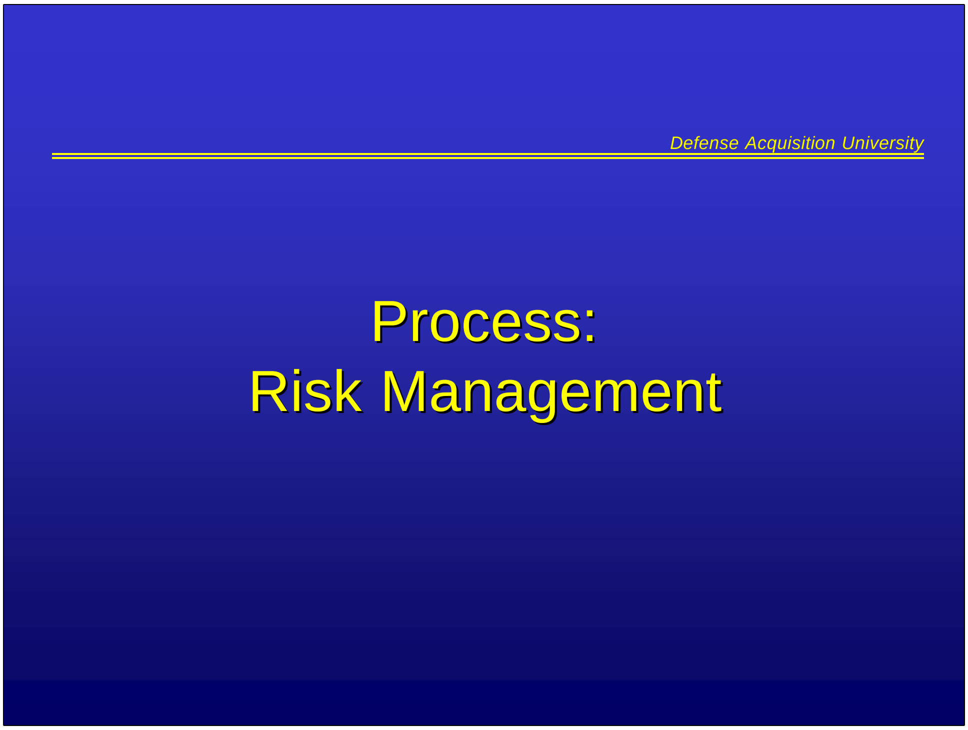
The OA describes "what must I do", the requirements describe "what I need', and together they explain "What I need, and why I need it". With this information, the linkage between requirements, capabilities, and mission effectiveness is more clearly drawn, and meaningful discussions can be held regarding priority, dependency, and impact of each requirement.

Priority and dependency are critical determinants of when a particular requirement will be satisfied. A high priority requirement should be addressed before a low priority requirement. Historically, without a prioritized list of requirements, developers typically gravitate toward doing the easy stuff first. It rarely occurs that the most important stuff is also the easiest to do, so the result is delivering a 20% solution, and saying "trust me" for the remaining 80%.

Dependencies are also important, particularly if a high-priority function is dependent upon services or data from something rated "low". Interoperability requirements often fall into this category, wherein the user may focus on the those requirements that are meaningful in the context of their OA, but must rely on other systems to provide that functionality, and thus must share that priority.

Note that the degree of detail and specificity of the requirement will vary depending upon its priority. High-priority requirements, which should be implemented earlier, must be wrung out in greater detail. This will become evident in subsequent steps.

In summary, the OA is the origin, and the rationale for all requirements in the SOW.



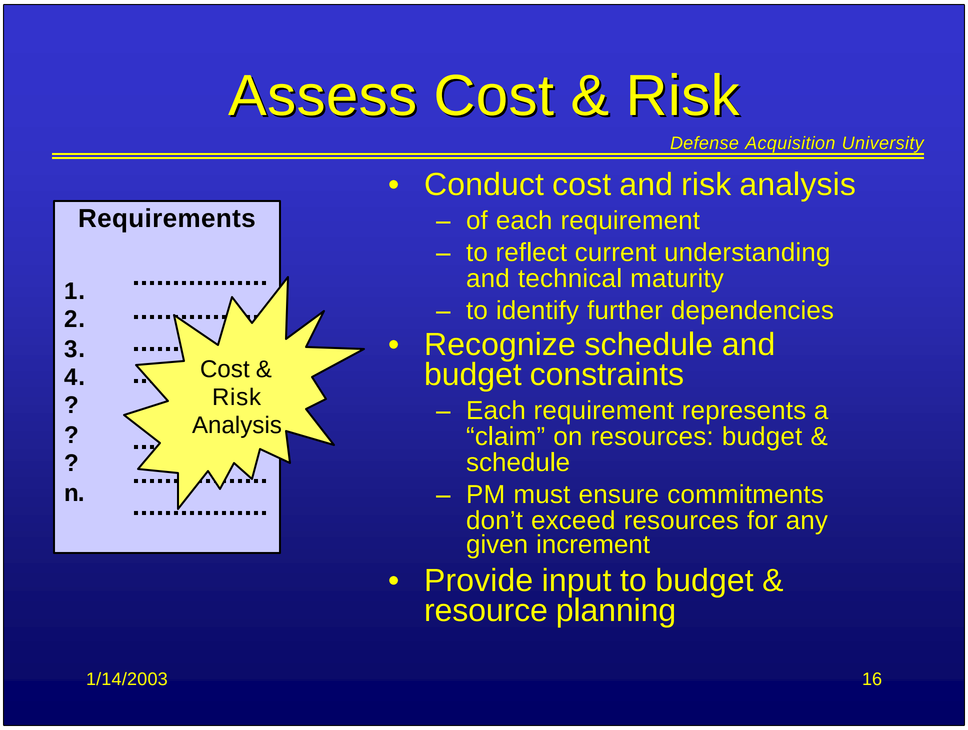
In the following discussion I use an intuitive, but somewhat non-standard concept of risk. The risk I'm concerned about is the risk of not being able to figure out what is required and implement it according to the specified schedule and budget.

The goal is to reduce the risk of satisfying any given requirement to nearly zero prior to committing to production.

In this conceptualization, risk has two primary drivers: requirements risk: do I understand what is required? And implementation risk: do I know how to build it? Is the technology mature and available? Does it conform to my systems architecture? Etc. etc.

For each of these general types of risk, different mitigation strategies apply. For requirements risk, further elicitation/elaboration of requirements is necessary. This must be done with the user, often in studies, focus groups, or rapid prototyping sessions (the last, in particular, for "look and feel" requirements, which Barry Boehm refers to as "IKIWISI" requirements: "I'll know it when I see it").

Implementation risk deals with technology maturity, and compatibility with the systems architecture, which responds well to the spiral development process of iterative prototyping, assessment, and refinement. One particular subset of implementation risk deals with Commercial off-the-shelf (COTS) products. Often a great deal of effort is required to determine if a promising product can actually be successfully integrated into the systems architecture.



To attempt to implement a requirement without first assessing the clarity of the requirement, and the likely cost and technical feasibility of the requirement is sheer madness. Yet this is what we have typically done. Often programs have committed to a program budget and schedule without the foggiest notion of whether the specified requirements can be met within those constraints.

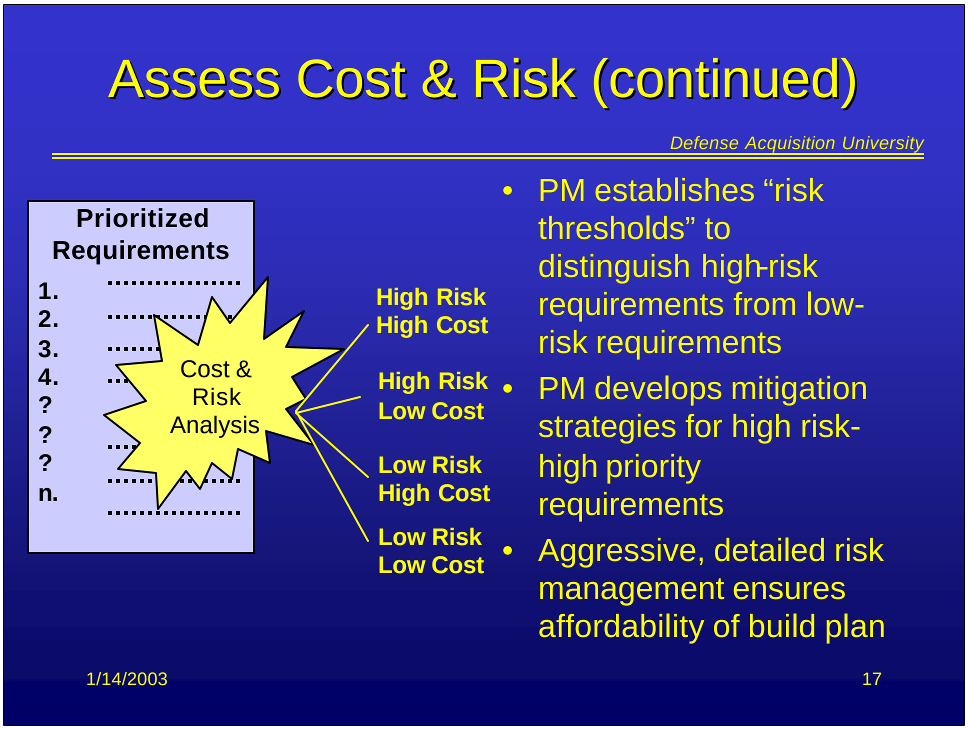
It's helpful to think of each requirement as a "check" that must be written against the "bank" of time and money. As with personal finances, bad things happen when more "checks" are written than are resources available to cover those demands.

Risk analysis is a very technical exercise that tends to ask nasty questions like "what do you mean by this requirement?", and "how to you intend to test this requirement?", and "how do you plan to implement this requirement?". But these are critical questions to ask early in the program before expectations are set regarding what is feasible.

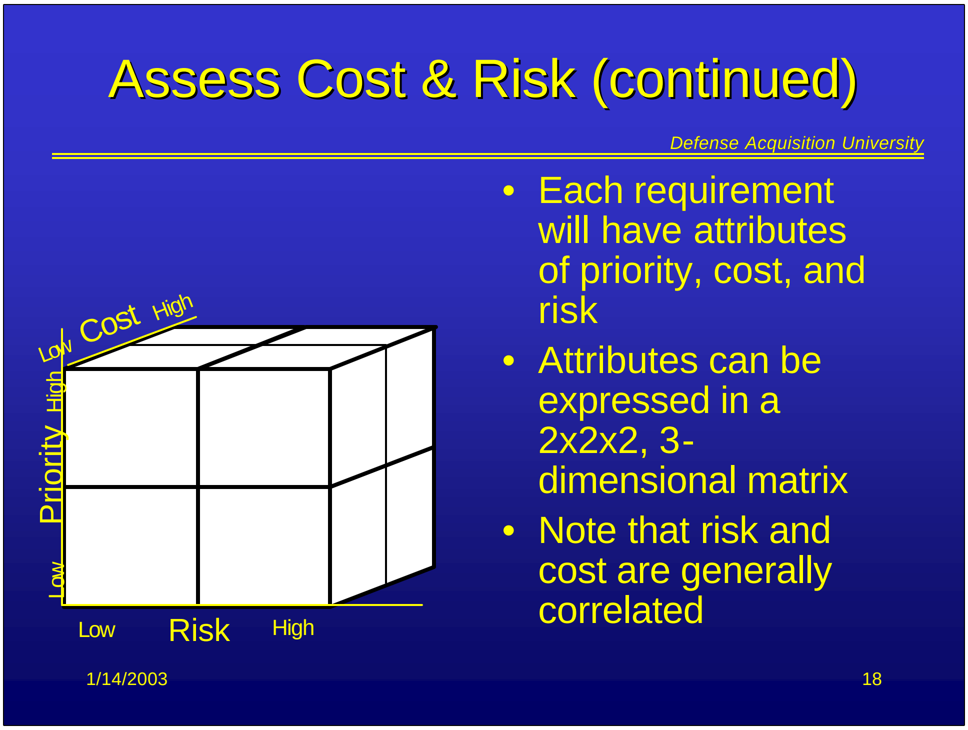
Note that priority and timing of the requirement will determine the degree of detail and specificity necessary in determining risk and cost. Near-term, high-priority requirements must be carefully analyzed for cost and risk, whereas requirements that will not be addressed for several increments can be more "fuzzy".

Make no mistake, however, this is a difficult and laborious exercise, and will result in multiple iterations, going back to the OA, and validating the requirement against the operational rationale, looking for easier and more cost-effective ways of achieving the same objective.

During this time, investment in automated requirements tools, linked to the OA and SA, will prove vital. Also, using automated means to creating "virtual teams" will provide necessary insight and feedback from other key stakeholders such as testers, producers, and supporters.

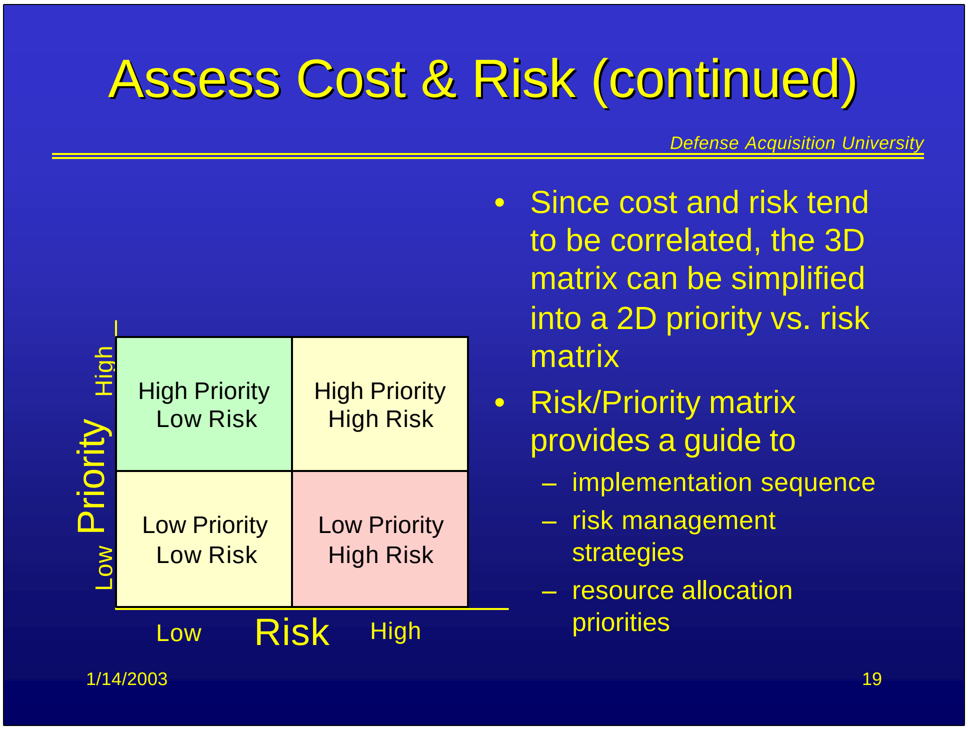


As a result of the preceding analyses, ach requirement will have attributes of priority, cost and risk. These attributes determine the specific development approach for each requirement.

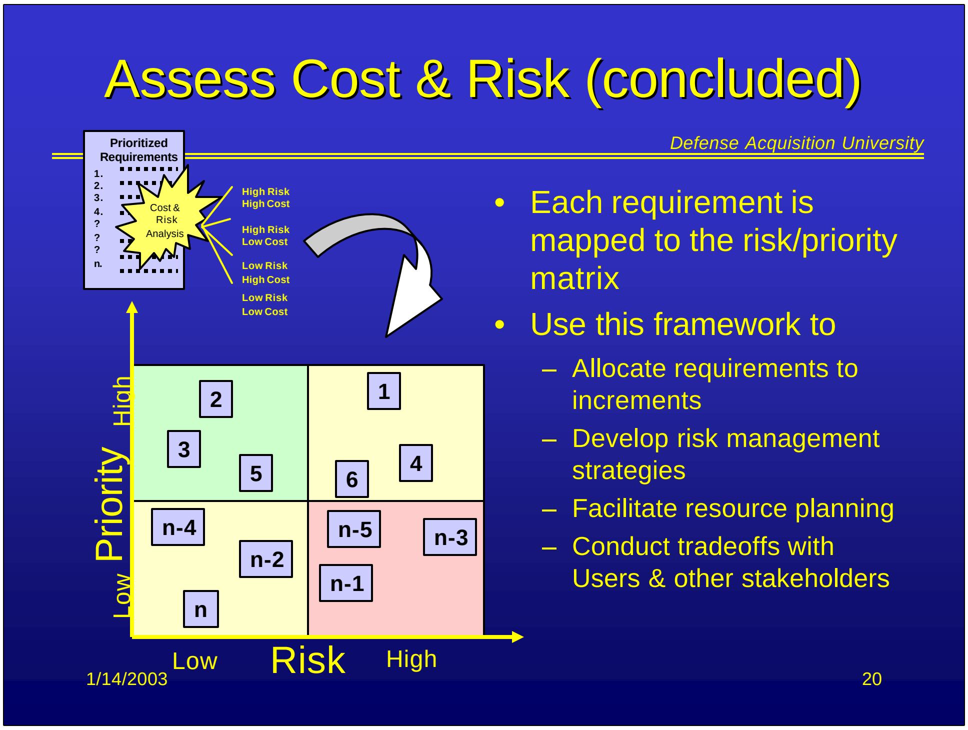


Conceptually requirements, characterized by their priority, cost and risk can be mapped to a 3 dimensional matrix. Requirements with similar attributes will cluster in the various regions within the matrix.

Attributes of risk and cost are generally correlated. This is because if the program intends to satisfy a risky requirement, resources must be allocated to mitigate the risk before it can be implemented. Therefore, for simplicity and clarity's sake, the 3-Dimensional matrix can be collapsed to 2 dimensions of priority and risk.



This risk versus priority matrix is a useful tool to determine the development approach for all requirements. This approach will ensure that the program delivers the most important capability to the user in the quickest time, and at the lowest cost.



This risk versus priority framework is the major tool for the Program to determine what efforts to undertake when. It also communicates clearly to all stakeholders the rationale for the development approach. Specifically, this framework describes:

The sequence of requirements to be implemented

What requirements will be subjected to risk mitigation

What requirements will be allocated to subsequent increments

How many requirements will be implemented in any given increment

Note that priorities and risks are relative, and will change over the implementation period

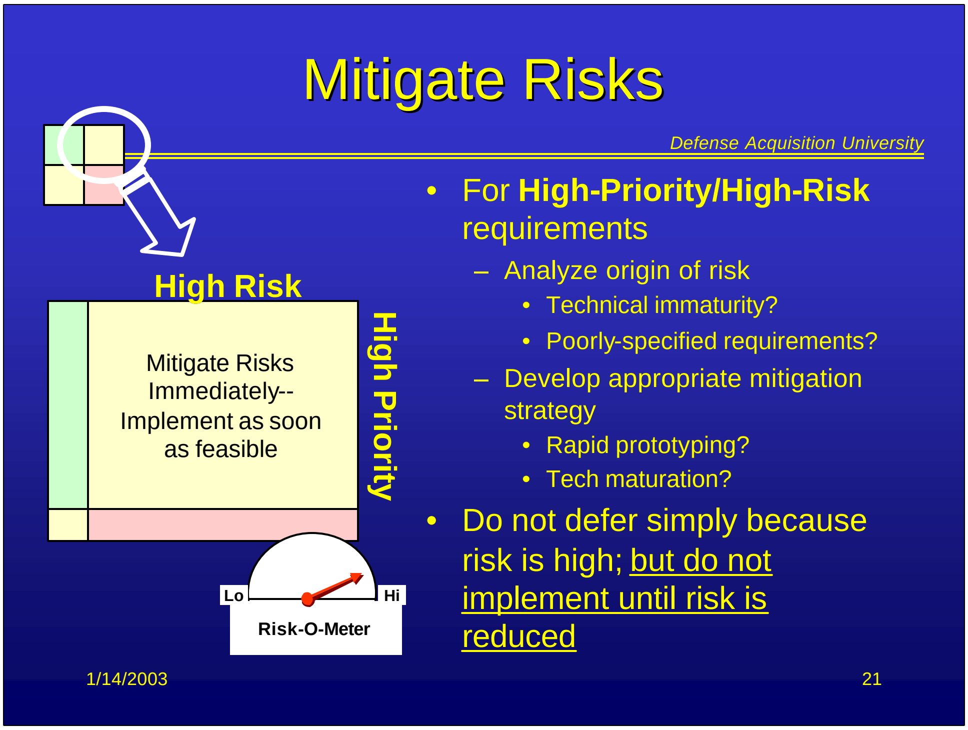
As top priorities are satisfied, the requirements with lower priorities will rise in relative importance priority

As technology progresses, high risk requirements will become feasible

As technology progresses and/or new threats emerge, low priority “nice to haves” may become essential

As doctrine, policy evolve, the OA may have to be modified, giving rise to new, or re-prioritized requirements

As a result of the fluid nature of relative risks and priorities, this priority/risk/cost matrix must be reassessed regularly.



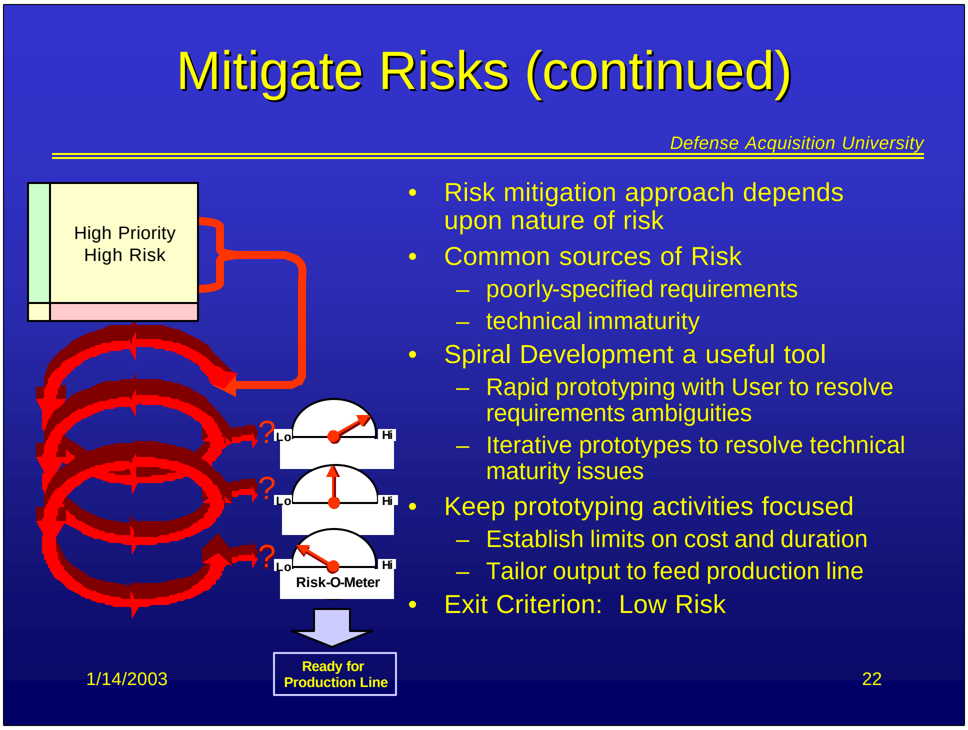
Let's tackle the toughest case first: High Risk, High Priority.

This is where Programs can get in trouble right off the bat. In the laudable desire to deliver an "80% solution", the PM may attempt to satisfy all the high priority requirements in the initial increment. But if these high-priority requirements are risky, the program will almost certainly end up delivering far less capability than promised, take much longer to develop it, expend more money to do it, and will destroy the program's credibility in the eyes of the user and other stakeholders.

So do you sweep all the risky requirements downstream, and just implement the easy stuff? Many programs do that to "log an early win, to build confidence in the program". However, this tends to result in a "20% solution", and the credibility of the program is again suspect.

The right answer is to begin to mitigate the risk of the high-risk, high-priority requirements, with the intent to implement them as soon as the risk is sufficiently mitigated to do so. This ensures these high priority requirements are addressed as soon as possible, but the program is not "betting the farm" that these will be delivered in the first increment.

A quick note on the concept of the "Risk-o-Meter". This is a tongue-in-cheek concept that represents the multidisciplinary assessment of risk, that includes concepts like technical maturity, producibility, maintainability, compatibility with the selected systems architecture, scalability, etc. etc. The program and risk managers must consider all aspects of risk—whatever will jeopardize the program's ability to deliver an operationally effective and suitable product to the warfighter.



Right at the start, the PM must implement effective risk mitigation activities to drive down the implementation risk of these high-priority requirements.

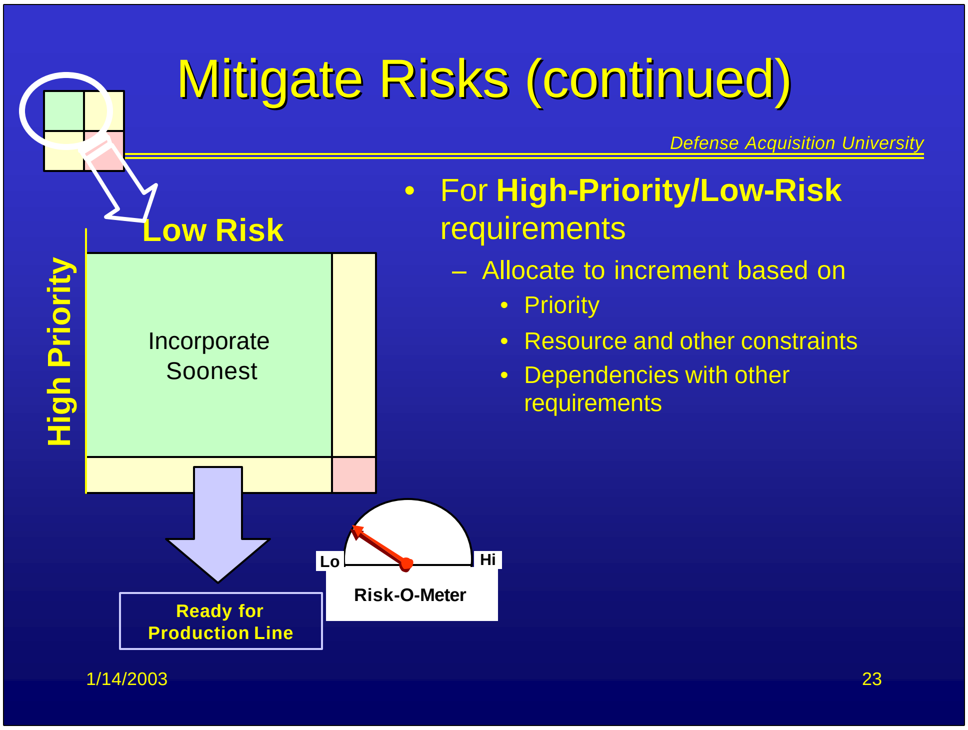
Spiral development, as is shown here, is a flexible, risk-based method to quickly reduce the uncertainties of producing the required capability.

The specific activities within the spiral will depend upon the nature of the risk. For example, if the risk arises from poorly specified requirements, or an IKIWISI user interface, then the Spiral will likely take the form of rapid prototypes that allow the User to review and correct. The output of these activities is a highly refined requirement set, and probably a prototype that unambiguously reflects the user's priorities.

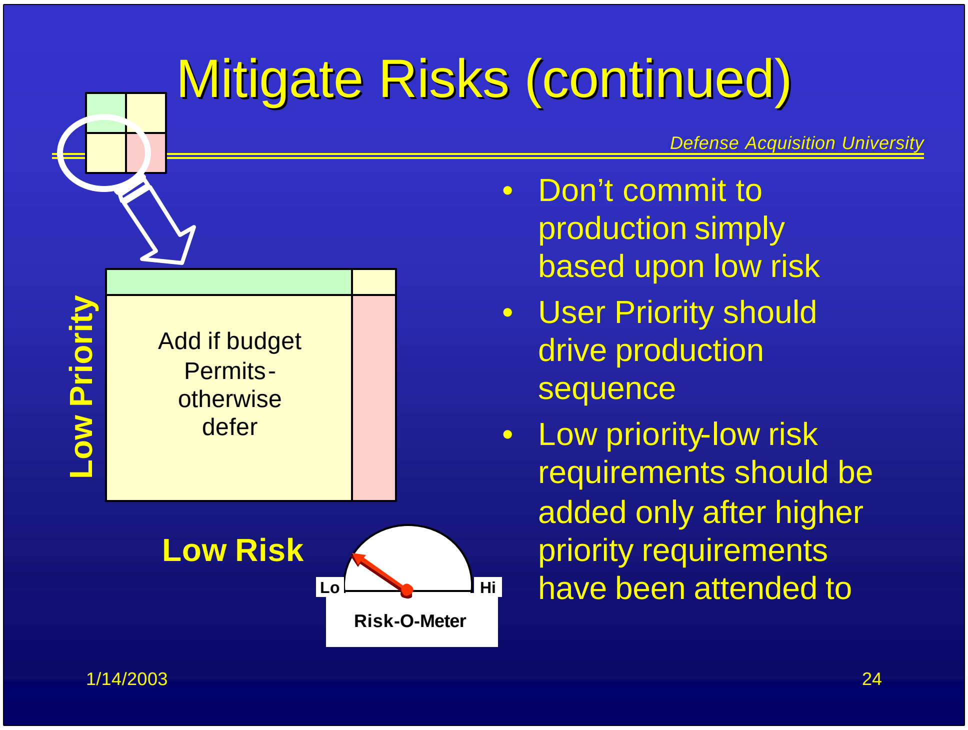
If the risk for a particular requirement results from technical immaturity, the spiral will likely take the form of iterative prototypes that mature the technology, taking it from the lab to the field, to an operationally-relevant environment.

As is portrayed by the "Risk-o-Meter", each spiral increases knowledge, and therefore reduces the risk of implementing the requirement. When the risk is deemed sufficiently low, then the requirement is approved for entry into the production pipeline, and is allocated to an increment.

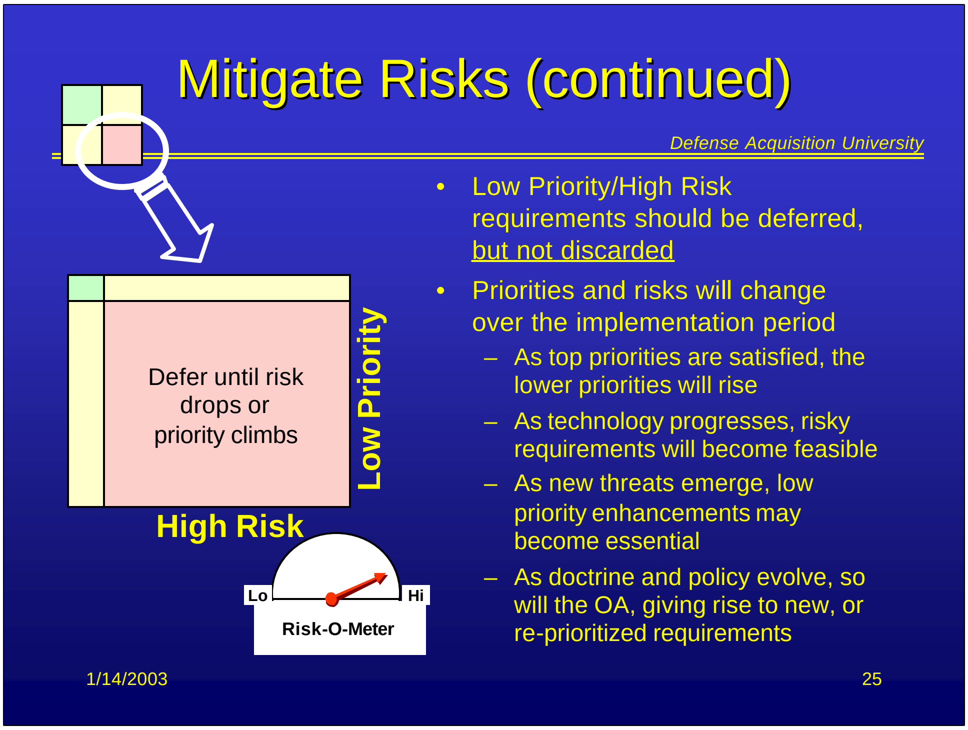
When structuring the risk mitigation strategy, ensure the output of the prototyping process, for example, is a product that can efficiently feed the production line. Using production-representative processes in the prototyping effort will help mitigate production risk. This is not a "theoretical exercise", the purpose is to prepare the requirement for incorporation into a deliverable increment.



The high-priority, low-risk requirements should be incorporated as soon as schedule and budget permit. These should pose little implementation risk, but still, it is wise to reserve some schedule/budget margin for “unknown unknowns”, particularly in the early increments.

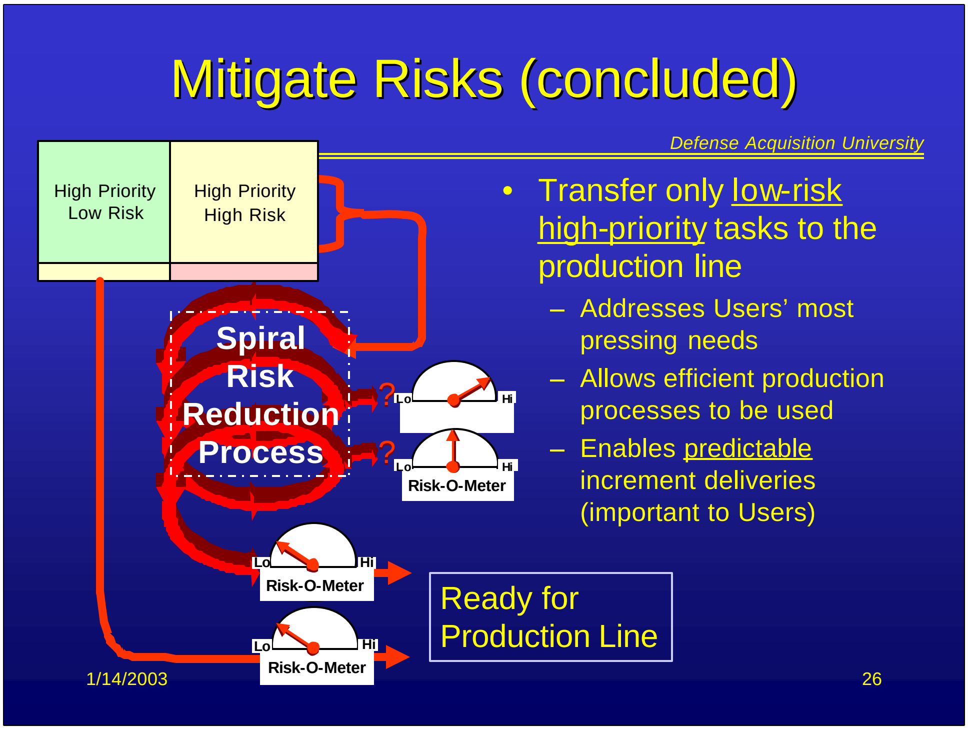


Resist the temptation to fill up the early-increment "job jar" with low-risk/low-priority requirements. Even if these low-priority requirements appear simple, they introduce complexity, and may increase the aggregate risk of the increment. Rather, if possible, allocate excess production resources to mitigate risk in high-risk/high-priority requirements.



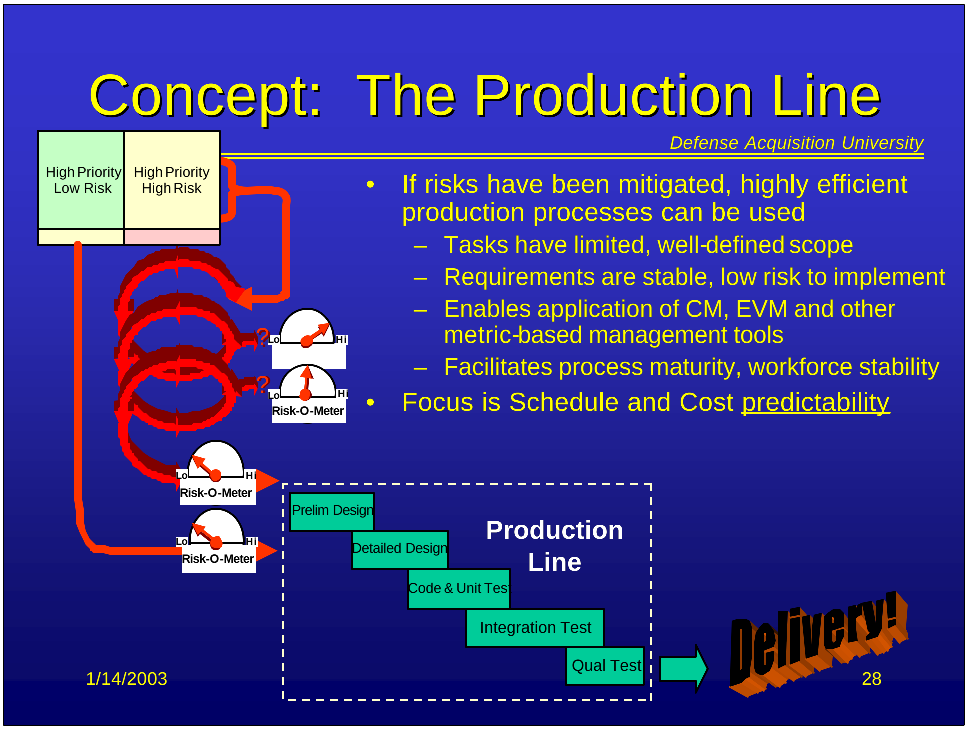
These are the risky requirements that are sufficiently low in priority that they don't merit aggressive risk mitigation with precious program resources. However, bear in mind that these requirements will eventually bubble to the top in priority, and therefore will have to be addressed. The best approach is to try to mitigate the implementation risk of these requirements with OPM (Other People's Money). For example, float a list of these long-term requirements to the Science and Technology community, to see if the labs can help mature the technology such that it will be available when the program requires it. Also, if there is a commercial market for the technology in question, keep track of how industry is maturing the technology, or perhaps encourage defense contractors to invest IR&D money to mature the technology.

Another reason not to completely ignore these high-risk, low-priority requirements is that the world might change—new threats might emerge, user doctrine might evolve, or the technology marketplace may leap forward, resulting in a dramatically different risk/priority rating.



Users place a high value in predictability when it comes to fielding new technology. Training cycles, deployment cycles, range time, and other features of the User's calendar are not, in general, flexible. Delay in fielding, or delivering substantially less capability than planned causes disproportionate disruption to the Users, incurs additional cost or operational risk, and erodes the User's confidence in the acquisition process. Therefore, the acquirer must place tremendous emphasis in the predictability of delivery—making sure the expected capability is delivered on the date projected. To accomplish this, only low risk, high priority requirements should be allocated to the production process. Not all of the User's high priority requirements are low risk, however, so the Acquirer must take active steps to mitigate the risk of these requirements. The Spiral risk reduction process is one method for accomplishing this. Through iterations of the analyze, build, test, evaluate cycle, requirements ambiguities can be resolved, and technical maturity can be enhanced. When the risk for satisfying a particular requirement is mitigated in this method, it is only then ready for allocation to the implementation process. The allocation decision includes rationalizing the requirement against resources and competing requirements. But by having mitigated the risk, the high priority requirement now is feasible with high predictability.

Process:
Production



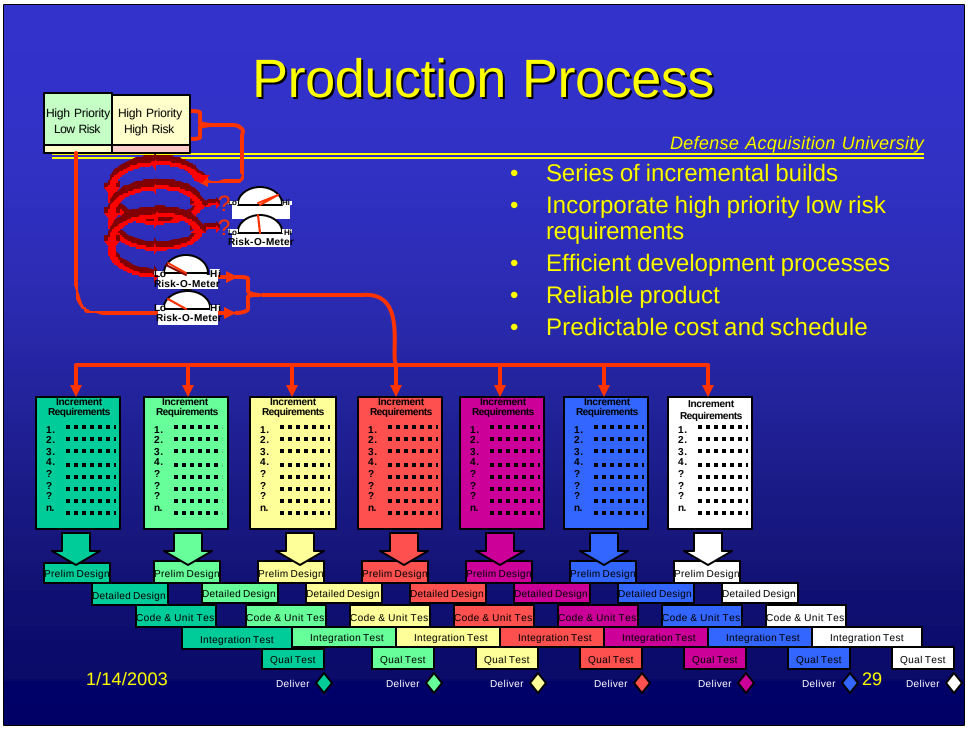
Once all the analysis is done, this is how the process runs. High priority, low risk requirements are fed into an efficient production process, where they are transformed into a robust, well-documented, stable, supportable, operationally effective and suitable capability that is delivered to a trained, well-prepared user. High priority, high-risk requirements are fed through a spiral development process which reduces their implementation risk such that they can be fed into the production process with similar results. Note that the risk mitigation process takes time, and so these requirements may not be incorporated in the same increment as the lower risk ones.

The production process, since it only implements low risk requirements, can be very efficient, with a stable, high expertise workforce, implementing mature development processes. Such a production process is also very predictable, delivering the expected output on time, and within budget, which is very important to all the stakeholders.

Note this graphic depicts how Spiral and Waterfall development can be used to support an evolutionary acquisition strategy.

Spiral development is used to rapidly mitigate the risk of high priority, but risky requirements. The output of the spiral is a waterfall production process.

Waterfall development can efficiently turn low-risk, and risk-mitigated requirements into delivered capability. This can be done because the requirements are clear, the technology is mature, and the development period is sufficiently short to prevent requirements from changing during the development period.



In order to maintain the efficiency of the production process, the production "pipeline" must stay full. This means that the planning for future increments must occur simultaneously with the production of the current increment. This planning means allocating requirements to increments based upon priority to the user, and the expected time for the risk to be mitigated sufficiently to allow implementation.

Here's a rule of thumb: When planning for increment "n", allocate high priority, low risk requirements to increment "n" and if necessary, "n+1". Allocate high-priority, high-risk requirements that will undergo risk mitigation immediately to increment "n+1" or "n+2" depending upon the anticipated time to reduce risk. Allocate high-risk, high-priority requirements that are not yet being risk-mitigated to increment "n+2" or later.

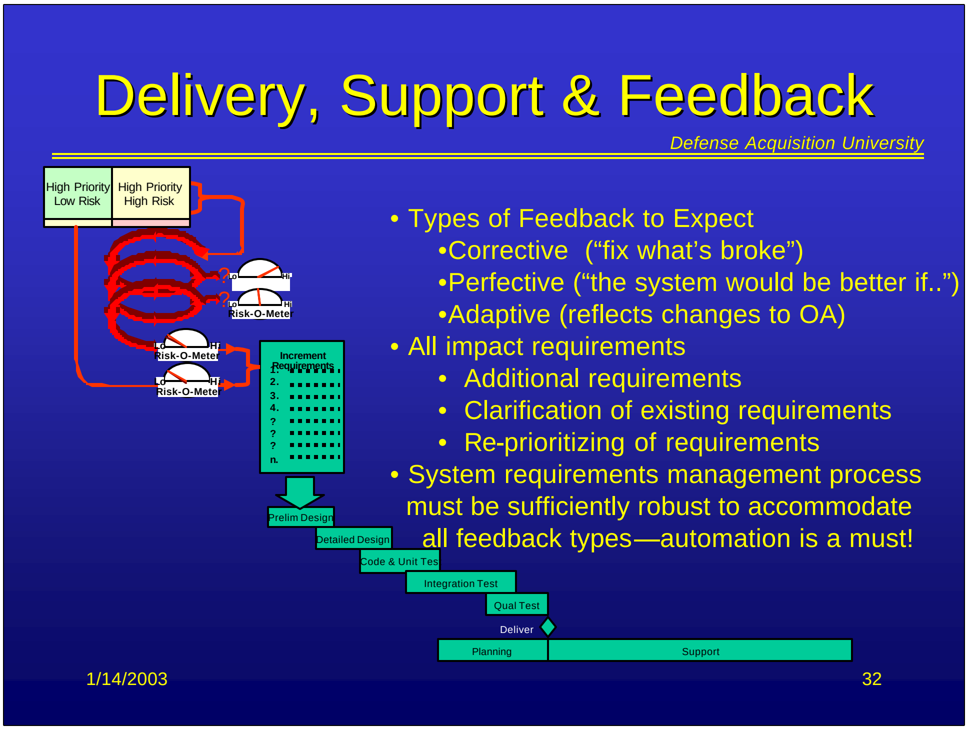
Process:
Delivery, Support & Feedback

As mentioned before, predictability is a critical to the user, so the acquisition process must deliver what is promised when it is promised.

All the planning and risk mitigation is designed to provide the most useful capability at the earliest possible time, with a high degree of predictability.

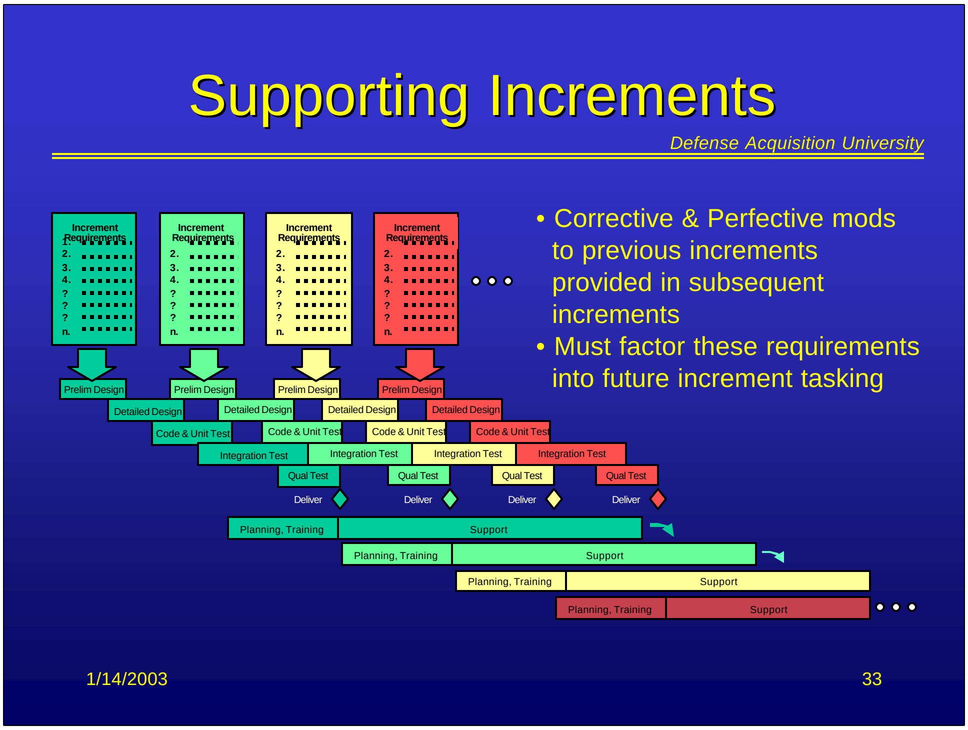
The product must be both supportable (a design requirement), and supported (requires planning and resources).

Once the product is in the field, the user will quickly identify everything that is wrong with it. This is a good thing, and the acquirer must provide a robust and responsive mechanism to a) fix what breaks, b) feed design deficiencies back into the requirements process, and c) track the evolving requirements of the user as he/she uses the technology in ways never before imagined.



Note that changes in the post-deployment phase will be corrective (fix this, it's broken), perfective (it would work better if it did this), and adaptive (my new ops concept requires me to do this, can you add this capability to the system?). From a systems engineering perspective, there is little distinction between responding to requirements in a development versus support mode. As a result, it makes sense to use the same process for addressing post-deployment requirements as was used to develop the system in the first place. This makes more sense when you consider in an evolutionary acquisition, the development and support processes run concurrently.

And I mean the WHOLE PROCESS: from revalidating the OA, to ensuring the SA is still valid, to refining, validating, and prioritizing the requirements, to risk and cost analysis, risk mitigation, allocation to an increment, and commitment to the production line. Figure on having at least an annual stakeholders meeting to revalidate the OA and overall requirements set. You can also use this venue to demo the new prototypes and to gain feedback from users on the current build.



As a general rule, corrective issues (errors, bugs, etc.), and perfective issues (better ways to accomplish the mission) are high priority, and will displace additional functionality in subsequent increments (in other words, if you have a choice between shipping a patch and adding a new cool capability, guess which gets deferred to the next increment?). The program should plan for this dynamic when allocating requirements to increments. For each succeeding increment, the amount of capability out in the field is greater, so the number of corrective/perfective requirements will likely grow. Therefore, plan on reducing the amount of new capability in later increments, to provide production margin for the inevitable corrective and perfective effort.

Note that the program must plan for delivering the product to the user, so implementation of a materiel fielding plan is necessary for each increment. Typically, users and support staff must be trained, facilities must be established and prepared, help desks set up, etc. These plans must be developed in parallel with the definition of the first increment, and be fully implemented with the delivery of the first increment. Support will continue throughout the remainder of the development period, and must be able to accommodate the addition of functionality as new increments are delivered.

Summary

Defense Acquisition University

- Acquirer must collaborate with user in requirements definition
- Acquirer must stay engaged with user throughout product lifecycle
- Establish and distribute SA as first “deliverable”
- Implement functionality in order of operational priority using OA as reference
- Mitigate risk of high-priority requirements using spiral development
- Send only low-risk high-priority requirements to production line
- Integrate test and support early and throughout increment development
- Institute formal support & feedback over lifecycle

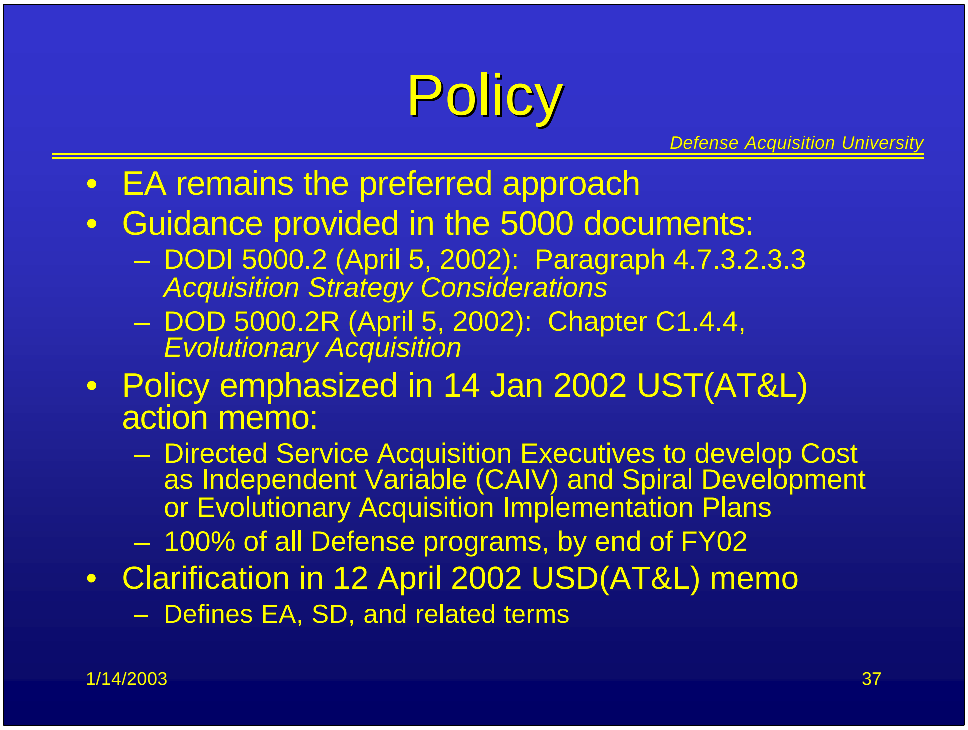
1/14/2003 34

The success of EA depends largely upon the Acquisition Program Manager taking a proactive, risk-focused approach. EA is management and resource intensive. It will demand extraordinary leadership and management skills on the part of the acquisition workforce, and will demand vigorous support on the part of the Defense Acquisition leadership.

Discussion

Background

- Evolutionary Acquisition (EA) as a formal acquisition approach initiated ~1993
  - Documented in Joint Logistics Commanders' (JLC) sponsored publication
  - Primary applications in C^4I community
- Updated in 1995 and included in DoD 5000 series as an alternate program acquisition strategy
- Evolutionary Acquisition became preferred approach in Jan 2001 update to DoDI 5000.2, Change 1



4.7.3.2.3.3. Acquisition Strategy Considerations

4.7.3.2.3.3.1. The acquisition strategy shall define not only the approach to be followed in System Development and Demonstration, but also how the program is structured to achieve full capability. There are two such approaches, evolutionary and single step to full capability. An evolutionary approach is preferred. Evolutionary acquisition is an approach that fields an operationally useful and supportable capability in as short a time as possible. This approach is particularly useful if software is a key component of the system, and the software is required for the system to achieve its intended mission. Evolutionary acquisition delivers an initial capability with the explicit intent of delivering improved or updated capability in the future.

4.7.3.2.3.3.2. The approach to be followed depends on the availability of time-phased requirements in the ORD, the maturity of technologies, the relative costs and benefits of executing the program in blocks versus a single step, including consideration of how best to support each block when fielded (e.g., whether to retrofit earlier blocks, the cost of multiple configurations, how best to conduct new equipment training, etc.). The rationale for choosing a single step to full capability, when given an ORD with time-phased requirements, shall be addressed in the acquisition strategy. Similarly, the rationale for choosing an evolutionary approach, when given an ORD with no time-phased requirements, shall be addressed in the acquisition strategy. For both the evolutionary and single-step approaches, software development and integration shall follow an iterative spiral development process in which continually expanding software versions are based on learning from earlier development.

4.7.3.2.3.3.3. In an evolutionary approach, the ultimate capability delivered to the user is divided into two or more blocks, with increasing increments of capability. Deliveries for each block may extend over months or years. Block 1 provides the initial deployment capability (a usable increment of capability called for in the ORD). There are two approaches to treatment of subsequent blocks:

4.7.3.2.3.3.3.1. The ORD includes a firm definition of full capability, as well as a firm definition of requirements to be satisfied by each block, including an IOC date for each block. In this case, each block shall be baselined and the acquisition strategy shall define each block of capability and how it will be funded, developed, tested, produced, and operationally supported.

4.7.3.2.3.3.3.2. The ORD includes a firm definition of the first block, but does not allocate to specific subsequent blocks the remaining requirements that must be met to achieve full capability. In an evolutionary acquisition, the specific requirements for Block 2 are defined in the ORD, based on the user's increased understanding of the delivered capability, the evolving threat, and available technology, lead-time-away from beginning work on Block 2, and so on, until full capability is achieved. Requirements that cannot be fulfilled during a specific block development, with the approval of the requirements authority, may be delayed to the next block development. The first block, and each subsequent block, is baselined in conjunction with the MDA authorizing work to proceed on that block. The acquisition strategy shall define the first block, of capability, and how it will be funded, developed, tested, produced, and supported; the full capability the evolutionary acquisition is intended to satisfy, and the funding and schedule planned to achieve the full capability to the extent it can be described; and the management approach to be used to define the requirements for each subsequent block and the acquisition strategy applicable to each block, including whether end items delivered under earlier blocks will be retrofitted with later block improvements.

Latest Policy Status

- DoDD 5000.1, DODI 5000.2, and DOD 5000.2R rescinded as of 30 Sept 2002
- Interim Guidance: yet to be signed
- Bottom Line: DO WHAT MAKES SENSE
  - Apply EA and SD where appropriate
  - Regardless of status of 5000, we still need to meet the same standards

EA: 2 Approaches

- ORD includes a Firm Definition of Full Capability as well as a *firm definition of Requirements to be satisfied by Each Block*: Acquisition Strategy describes how each Block will be Baselined, Funded, Tested, Produced and Supported

- ORD includes a *Firm Definition of the First Block but Does Not allocate to subsequent Blocks the remaining Requirements:* Subsequent requirements based on User's increased understanding of Threat, Available Technology
  - MDA authorizes work to begin on subsequent Blocks in consideration of above as well as Full Funding, Test and Sustainment Strategy, etc.

Source: DOD 5000.2R, 5 April, 2002