Modularity as an Enabler for Evolutionary Acquisition

by

Nirav Bharat Shah

B.S. in Aeronautics and Astronautics
Massachusetts Institute of Technology, 2001

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2004

Author .....
Department of Aeronautics and Astronautics
May 14, 2004

Certified by .....
Daniel E. Hastings
Professor of Aeronautics and Astronautics and Engineering Systems
Co-Director, Engineering Systems Division
Thesis Supervisor

Accepted by .....
Edward M. Greitzer
H.N. Slater Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

Modularity as an Enabler for Evolutionary Acquisition

by

Nirav Bharat Shah

Submitted to the Department of Aeronautics and Astronautics
on May 14, 2004, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

The end of the cold war witnessed several significant changes in the defense acquisition environment. Budgets declined and the scope of missions expanded. At first, the DoD did not respond well to these pressures resulting in cost overruns and schedule delays becoming the norm. In an effort to change this situation, national security officials decided to fundamentally change the way systems were acquired, shifting the focus to systems that could evolve/adapt to changing resources and needs. To operationalize this shift they recommended implementing an evolutionary acquisition strategy using a spiral development process.

The fundamental characteristic of the evolutionary acquisition strategy is a focus on delivering a minimum capability early and then building upon that capability as risks are resolved. This imposes requirements on the acquisition process and the system architecture. From a process perspective, since needs and resources are changing over time, involving all relevant stakeholders is key to successful evolutionary acquisition. Since the objective was to prevent cost overruns and schedule slips, understanding and mitigating key risks is central. From an architectural perspective, the ability to update the system to allow for enhanced capability is important.

The MATE-CON process can be used to satisfy the process related requirements of evolutionary acquisition. MATE-CON uses a multi-attribute utility theory to capture the diverse and changing needs of decision makers. Then tradespace exploration is used fully reveal the impact of design decision on decision maker perceived value. By representing both value and expense in terms of metrics that all stakeholders can understand, MATE-CON facilitates effective collaboration between stakeholders.

A modular architecture provides the architectural flexibility required when using evolutionary acquisition. By separating system components into a few decoupled

modules, the modular architecture allows enhancements to be made to the modules without adversely affecting the rest of the system. Changes can be made to accommodate new needs or incorporate new capabilities. This flexibility in-service come at the price of a possible loss of performance and/or design efficiency. Thus valuation of the flexibility gained is crucial. An options framework that incorporates risk identified by the stakeholders should be used.

The Space Based Radar is a good example of a system that demonstrates the challenges faced by defense acquisition identified above. The most recent attempt at an actual hardware demonstration was canceled because of cost overruns and schedule slip. Using the MATE-CON approach and a modular constellation architecture, evolutionary strategies for space-based radar can be found.

Thesis Supervisor: Daniel E. Hastings
Title: Professor of Aeronautics and Astronautics and Engineering Systems
Co-Director, Engineering Systems Division

Acknowledgments

The author wishes to acknowledge the following people and groups for their aide in completing the research documented in this thesis.

Adam and Nathan for the MATE-CON process. Tim for his SBR model. Chris for developing with me the notion of transition options. Jason and Bobak for helping me better understand spiral development and evolutionary acquisition.

The students, staff, and faculty of the Lean Aerospace Initiative for their support over the past few years and making me aware of the non-technical side of engineering. The members of Prof. Hastings' research group for teaching me about space system architecture. Prof. Hastings for giving me the opportunity to learn so many different things from so many different fields and not giving up on me when things got tough.

All of my students for teaching me that I don't know nearly as much as it sometimes may seem. All of my instructors for convincing me that every so often I get things right. The students and staff of ESG for teaching me so much about learning and life.

Michael, Michelle, Cappy and Phil for being the brothers and sisters for whom I've always wished. Tanvi for the encouraging words when I needed them most. Jen for editing both my writing and thinking. Dave for teaching me how to think like a writer. Rakhi for hugs, smiles and keeping me sane.

For mom and dad

Also, for Joyce

Table of Contents

List of Figures

List of Tables

Chapter 1

Motivation and Introduction

1.1 Background and Motivation

The purpose of any engineering design process is to develop some application of scientific or technical principles in order to meet some human need. This capability must be delivered in a timely manner, and at a reasonable cost[1]. Over the past hundred years, the aerospace industry has delivered spectacularly on this promise. One need only consider such successes as the Apollo program and widespread passenger aviation to understand why the aerospace industry is often equated with 'doing the impossible'. One of the most significant areas of application for the aerospace industry is defense. Events following the cold war have posed new challenges to defense aerospace in living up to a history of great achievement.

A report on national security space programs by a joint task force of the Defense Science Board and the Air Force Scientific Advisory Board (DSB/AFSAB, 2003) identified several changes in the military aerospace sector that occurred during the 1990's following

[1] More quantitatively, the cost must be sufficiently low such that the value derived from the new capability exceeds the value that could be derived from other uses of the same capital.

the cold war. The end of the cold war brought about a decline in defense acquisition budget as the threat environment changed. The market for commercial uses of space also failed to materialize. As a result, business decreased in both the public and private sectors which led to consolidation in the aerospace industrial base. At the same time, the threat changed from a focus on the USSR to a more diversified and dynamic global threat.

In response to this evolving threat, national security officials embarked on an effort to update the national security infrastructure (the military and intelligence communities). Their goal was to create a more agile force that could quickly and effectively adapt to dynamic funding and threat environments. This effort attempted to foster two major changes: (1) increased integration and standardization across the military, and (2) reduced cost through more efficient management of limited resources. A 'systems of systems' approach became more common emphasizing the interrelationships between different national security assets. Such an integrated approach, they believed, would increase agility by reducing delays and waste caused by interfaces between disparate systems. A principal impact of the focus on integration was that many more uses were expected out of a given system. Faced with increasingly demanding and complicated requirements as well as decreasing budgets the acquisition process also needed to be reformed. [2]

The DSB/AFSAB (2003) report found that one of the major impacts of the mismatch between more requirements being imposed and fewer resources being provided was that the level of risk deemed acceptable increased. They came to this conclusion by looking at the acquisition system from several different perspectives. Facing budget pressure, the primary objective of an acquisition effort shifted from ensuring 'mission success' to staying within budget. This shift is particularly problematic in space programs. At their core, space systems involve both high energies and high complexity. As a result, even the slightest engineering error can have catastrophic consequences. When 'mission success' was paramount, such slight errors would be found early and corrected, thereby avoiding

[2] Similar changes can be seen when looking at air-based programs as well.

catastrophic failures. Such failures are not as common in air-based programs because greater technology maturity and lower energies results in less risk. Being overly focused on cost has, however, resulted in cost increases and schedule delays. For example, reduce orders have contributed to unit cost growth on both the F-22 and B-2 programs.

Having cost as the key objective could work if the budget process allowed for sufficient margin to ensure mission success in the face of greater uncertainty. The DSB/AFSAB concluded that such margin did not often exist. Rather, an over reliance on contractor provided, non-objective budget estimates and a practice of '50/50' budgeting[3] caused programs to have unrealistic budgets with underlying plans that were difficult, if not impossible, to implement. To understand the impact of unrealistic budgeting and planning upon a program, the DSB/AFSAB task force viewed a space program through four characteristics:

1. Requirements – *Receiving what we need?*
2. Schedule – *When we need it?*
3. Launch – *Where we need it?*
4. Cost – *At a price we can afford?*

The changes experienced during the 1990's put pressure on program managers on all four of these fronts. The larger user community for a given product resulted in a growing set of requirements. Unrealistic planning led to inadequate schedule and budget. Consolidation in the commercial sector limited the launch options available. Finally, there was often little or no margin included in budgets to allow a program to respond to these pressures. The task force found that program managers generally use one of two strategies when faced with these pressures, but lack the margin to respond the them: (1) they let the schedule slip and suffer significant cost penalties or (2) they take on greater risk incurring

[3] A program budgeted at a '50/50' level means that the program is expected to have a 50% chance of being under budget and 50% chance of being over budget.

a higher probability of mission failure. Neither strategy improves the situation. Instead, a more comprehensive reform of the theory and practice of acquisition is needed.

Of the four characteristics mentioned above, requirements instability seemed to be the most significant cause of schedule slip, cost increase, and greater risk. With the increase in requirements there has not been a corresponding improvement in the process by which requirements are managed. The effect of this can be seen in programs like SBIRS High.

SBIRS[4] High (Crock, 2002; DSB/AFSAB, 2003) is an excellent example of how requirement instability can lead to cost growth. Conceived in the mid-1990's as an early warning system for missile defense, SBIRS High suffered from a series of management mis-steps. The original requirements called for detection of long-range missile launches. As the program progressed, the Air Force and new users levied additional requirements. For example, intelligence agencies wanted ground surveillance capability to detect rocket-engine tests. The Army wanted short-range missile such as SCUDs to be detected. Tracking and decoy/warhead detection was also desired.These additional missions led to conflicting needs. For example, detecting rocket tests requires high acuity sensors that can discern small, low-contrast details; such sensors however have too high a false positive rate to be useful for a missile warning system. Solving/settling such conflicts is one of the reasons for cost growth from new/changed requirements (see Figure 1-1).

Requirements growth was not the only problem faced by SBIRS High; rather, most of the cost growth would have occurred even with the original requirements, which may have been too ambitious. In explaining the cost growth, the DSB/AFSAB found that the pressures on a program as described earlier did impact SBIRS High. To adapt to changing budgets, cost became the key driver. As a result, four planning efforts were needed during the program's short life. Contributing to the budget instability was a lack of credible estimates for the actual cost. In particular, the contractor's estimate (shown in Figure 1-1) was the lowest estimate produced. As part of a partnering strategy, this same contractor

[4]Space Based Infra-red System

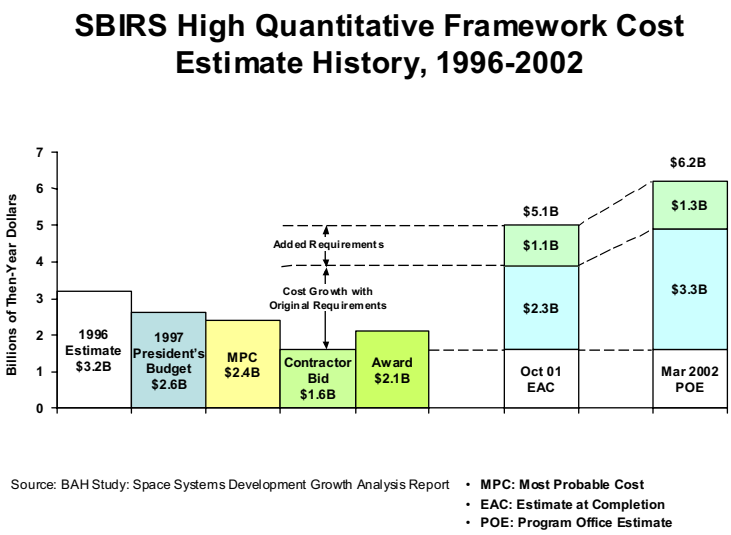


FIGURE 1-1: Cost Growth in SBIRS High from (DSB/AFSAB, 2003)

was given unusually wide latitude in engineering and management of the program. Some traditionally government oversight functions entrusted to the contractor. The government's intent was to eliminate oversight roadblocks that slowed down progress. The effect, however, was to limit the program manager's ability to respond to the changing funding and needs environment. Poor risk management was also an issue. For example, within a year of the launch date, well into the integration and test period, sensors on the spacecraft needed to be retrofitted with sun shielding. This problem was known five years prior to launch, but was not deemed a significant risk. With all of these difficulties, SBIRS High has become "an example of how not to run a space program".

The DSB/AFSAB committee made several recommendations for improving space acquisition programs including SBIRS High. First they recommend making mission success the 'overarching principal' governing acquisition. To ensure that success is tractable they recommend strict control over requirements and budget during the planning and implementation phases. They also recommend improvements in staffing policies so that learning takes place and a knowledge base develops.

Though the DSB/AFSAB report is recent, the Department of Defense (DoD) is aware of at least some of the problems identified and has enacted acquisition reforms to help mitigate them. The largest of these reforms were the DoD 5000 series of acquisition regulations (DOD, 2004). In DoD directive 5000.1 and DoD instruction 5000.2, evolutionary acquisition (EA) is established as the preferred acquisition strategy (Aldridge, 2002). EA attempts to deliver "core operational capability sooner by dividing a large, single development into many smaller developments or increments" (Delaney, 2000). This strategy reflects an acknowledgment by senior leadership that programs need to evolve when faced with a dynamic threat (and budget) environment. By making increments smaller, programs may be less vulnerable to the poor budgeting and planning cited in the DSB/AFSAB task force report. The evolutionary part of EA stems from the fact that requirements to be satisfied by each small increment are not necessarily defined at program inception, rather they reflect the needs (and budget) environment when the increment begins. As such, the system, like an animal, evolves in response to its changing environment. In principle EA does seem to be well suited to the new realities following the cold war. When implementing EA, however, difficulties may arise.

Aldridge cites two possible scenarios for the implementation of EA – one where the ultimate capability is defined at the start of a development effort, and another where requirements evolve to match changing needs and new technical capabilities. In the first case, a traditional acquisition process will suffice. Since the requirements of each increment are determined at the onset of the program, there is no need to allow for emergent requirements. This scenario is similar to the preplanned-product-improvement (P3I) (Unger, 2003) often citied in discussions of the acquisition system. The second scenario requires an alternative process that allows for feedback between users and developers so that changing needs and capabilities can be communicated. The Air Force recommends Spiral Development (SD) as such a process.

Barry Boehm, a software engineer who has studied EA/SD extensively, defines Spiral

Development as a risk-driven process used to guide multiple-stakeholder concurrent engineering. It has two distinguishing features: (1) a cyclic approach to allow incremental improvement of the system over time, and (2) a set of anchor or milestone points for ensuring stakeholder commitment throughout the product life cycle. By multiple-stakeholder, Boehm means that at each milestone, agreement must be achieved by not only those directly connected with the program (e.g. contractor and SPO), but also users and other relevant decision makers. Anchor points ensure that the spiral strategy, though iterative, makes progress toward a better-defined and improved system (Boehm, 2000).

Darien Unger (2003), in a study of product development (PD) processes, reiterates Boehm's focus of risk as the key driver of the spiral process. According to Unger, the distinguishing features of SD are that iterations span several stages of the PD process (e.g. cycling from design through manufacture and operation and then back to design) and that such large-scale iterations are expected at the onset of the development effort. SD's strong focus on risk indicates that it may help alleviate some of the problems found by the DSB/AFSAB task force. By identifying and mitigating key program risks early and requiring explicit buy-in from all stakeholders at each iteration, an SD process should be less susceptible to an unrealistic budget and schedule.

In non-spiral processes (e.g. a waterfall process), the cost of modifying early decisions tends to be very high. As a consequence, large-scale iterations common in SD are often avoided. Attempts to resolve issues discovered late in the PD effort are hampered by earlier decisions. A direct result of this limitation is that if needs change and 'mission success' is redefined, a very high cost penalty may be incurred in reorienting the program to the new mission. In SD however, the entire PD process is reviewed at each iteration and thus the remaining risks are more likely to be handled in a manner consistent with overall mission success.

The spiral process is, of course, not appropriate for every acquisition program. As will be

discussed in Chapter 2, programs using EA/SD successfully should have certain characteristics and present unique challenges. Boehm found that programs using EA have a product architecture that can accommodate changing user requirements, user and developer communities trust each other and be flexible to adapt to the pace of system evolution. Unger's cases that used a spiral process tended to be environments that had fewer less structured reviews and may not work well when high implementation costs and/or regulatory requirement necessitate a more structured decision process. Ferdowsi (2003) found that EA/SD is better suited to smaller programs. Johnson and Johnson (2002) identified several "perils of spiral development" including potential political risk of being more likely to face budget cuts since the flexible nature of SD allows for greater resilience to changes in budget. The logistics community faces challenges in supporting several versions of a system in the field simultaneously.

Given both Unger and Boehm's[5] characterization of the SD process, it is not surprising that the Air Force has chosen the Spiral Development process as the preferred process for implementing EA when the second of Aldrige's scenarios occurs. Though the Air Force has some experience with spiral development of software/computer intensive systems (avionics and C2 systems in particular see Delaney (2000)), little knowledge exists on applying spiral development outside of the software arena. To improve this situation, the Air Force established Acquisition Centers of Excellence, or ACE offices, at various bases around the United States. The mandate of the ACE offices is much broader than implementing EA/SD. They serve as a local resource to programs at a given base, providing guidance on running efficient programs. Their scope covers the whole range of acquisition-related activities from contracting and procurement to governing regulations (ACE Office, 2002). Collectively, these ACE offices are creating a new knowledge base for acquisition by gathering and sharing lessons learned.

In parallel with the 'learn by doing' approach taken by the ACE offices, MIT's Lean

[5] Additional definitions and perspectives on SD are provided in Chapter 2

Aerospace Initiative is attempting to close the knowledge gap through research into EA/SD. A series of recent theses, of which the current document is the last, explores several key facets of EA/SD within the aerospace context. Derleth (2003) looked at the application of Multi-Attribute Tradespace Exploration (MATE) to EA. MATE (Ross, 2003) uses multi-attribute utility theory in a structured design process to evaluate rapidly many different design options. Derleth argues that MATE is useful for EA because it provides a better understanding of the end user's requirements, an optimal design choice for the first iteration, an ability to understand possible optima for future iterations, and a short redesign time if user preferences or designer capabilities change. To demonstrate these advantages of MATE, Derleth built a three-iteration model of the Small Diameter Bomb.

One of the more onerous parts of the MATE process is the formal utility interview which reveals decision maker preferences. Spaulding (2003) explored simplifications of this process. He also compared a MATE based design for the Space Based Radar to an existing government concept study. He found that the MATE process revealed high value portions of the design space that were not explored in the previous government effort. Spaulding also commented on the usefulness of MATE within the existing acquisition process.

Roberts (2003) also considered Space Based Radar, but this time from a EA/SD perspective. Roberts' contribution is similar to that of Derleth, but for a space system. He also included the dimension on time into the MATE analysis. By considering a very large number of Space Based Radar deployment schedules, Roberts demonstrated that MATE concepts of utility and tradespace exploration could be used to study programs that deliver increasing value over time such as those that use EA/SD. These theses provide valuable insights into EA and SD; however, they do not directly address the system architecture implications of using EA/SD.

1.2 Contribution of this thesis

All three theses mentioned at the end of the previous section focused on the *process* by which systems are acquired under an EA strategy. Along with that process, however, there must be a *system architecture* that can cost effectively take advantage of the knowledge gained from one iteration to the next. A modular architecture provides such a facility. Modularity (most commonly found in the computer and IT industries) (Baldwin and Clark, 2000) is a design methodology that attempts to configure a system such that most of the interactions between components (both in the functional and physical sense) occur within each of a small set of 'modules' and few interactions occur between modules. In doing so, a set 'Design Rules' are established that govern the interfaces between the modules. With the inter-module interfaces fixed, the modules themselves can be replaced with new improved versions. As long as the existing 'Design Rules' are respected, other modules will not have to be changed in response to the upgrade.

A simple example of a modular system is a (somewhat idealized) freight train. There exist certain rules for coupling one car to the next. If these rules are followed, then the mix of cars in a particular train can be customized to the needs of the day. Furthermore, should needs change tomorrow, the engineer can, for example, add or remove a freezer car as needed. He need not redesign the entire train. From a system (train-wide) perspective, the functions of the individual cars is unimportant – only their coupling to the other cars matters. This 'hiding' of the within car functions that leads to the great cost saving associated with a modular design. As long as changes made within a module obey the 'Design Rules',i.e. the standardized interface connecting cars, the impact of those changes are limited to being within the module. The rest of the system is protected from any cost growth on account of those changes. This ability to swap[6] one module for another even after implementation is what makes modular design so useful in EA/SD. As new knowledge is gained from fielding the system, modules can be replaced by improved ones

[6]There are five additional operations one can perform on a modular design. See Chapter 4.

altering the systems performance without requiring significant system-level redesign each iteration.

This thesis argues that a modular system architecture can be a key enabler when using an evolutionary acquisition strategy and a spiral development process. Traditionally modular design has been used most successfully in the software industry; applying it to aerospace problems present new challenges. The applicability of such an architecture within the aerospace context will also be discussed.

1.3 Structure of the remainder of the document

Outlined below is the structure of the remainder of this document. The reader is encouraged to use this section as a guide in understanding the crux of the argument.

In chapter 2, evolutionary acquisition and spiral development are discussed in greater detail along with an overview of the use of MATE and modularity in implementing EA/SD. Chapter 3 is a more detailed description of the MATE process with a focus on valuation and Multi-Attribute Utility Theory (MAUT). Chapter 4 begins by defining modularity precisely in terms of a design/task structure matrix that exhibits a particular structure. Modular operators are introduced to provide a framework for design within a modular structure. The opportunities for design modification of a modular design over time are interpreted as a set of options on future designs.

Chapter 5 brings the MATE and modularity approaches together in a discussion of the spiral development of an aerospace system: Space Based Radar (SBR). SBR is an expensive space system that accomplishes several different missions. Being a high technology, high cost program, it suffers from both technical and budgetary risk. The ability to identify designs that allow limited capability immediately at relatively low cost,

yet still can be expanded in the future should resources become available, can help mitigate these risks. Hence SBR is an excellent case study for these ideas.

Finally, chapter 6 includes some caveats for applying this research in practice, an overview of the major conclusions, and several recommendations for future work.

Chapter 2

Evolutionary Acquisition and Spiral Development

To better understand the implications of using a spiral development process to implement an evolutionary acquisition strategy, the reader is encouraged to consult the following references: Boehm (2000) offers definitions and frameworks for identifying EA/SD; Maier and Rechtin (2000) and Unger (2003) discuss EA in the context of other strategies; Delaney (2000) provides a DoD/Air Force perspective, and Roberts (2003) discuses the implications of EA/SD specific to aerospace programs. The remainder of this chapter is a summary of these discussions with particular focus on the nuances of EA/SD that will be explored in later chapters.

2.1 Evolutionary Acquisition

Formally, the Air Force has defined EA as:

> "An acquisition strategy that defines, develops, produces or acquires, and fields an initial hardware or software increment (or block) of operational capability. It [the initial increment] is based on technologies demonstrated in relevant environments, time-phased requirements, and demonstrated manufacturing or software deployment capabilities. These capabilities can be provided in a shorter period of time, followed by subsequent increments of capability over time that accommodate improved technology and allowing for full and adaptable systems over time. (Aldridge, 2002)"

Though Aldridge's definition does not specifically mention the evolutionary part of EA, he does identify the key difference between a traditional acquisition strategy like 'preplanned product improvement' (P3I) and EA. Since the content of second and later increments are governed by the needs and capabilities that exist at the time of those increments, the system will evolve to meet changing needs and take advantage of new capabilities. In P3I, the content of the increments are planned in advance, technology development etc. are pursued to meet that schedule.

Boehm (2000) identifies four 'embedded assumptions' inherent to EA, echoing some of Aldridge's definition:[1]

1. "The initial release (or increment) is sufficiently satisfactory to key system stakeholders that they will continue to participate in its evolution (i.e. in the words of Aldridge, 'core operational capability' has been delivered)."

2. "The architecture of the initial release is scalable to accommodate the full set of system life cycle requirements (i.e. the second increment can build upon the first thereby meeting new needs apparent at the time of the second increment)."

3. "The operational user organizations are sufficiently flexible to adapt to the pace of system evolution."

[1] The stated definitions are from Boehm, less the parenthetical comments.

4. "The dimensions (both in time and space) of system evolution are compatible with the dimensions of evolving-out the legacy systems being replaced."

Each of these embedded assumptions needs to be satisfied for the successful implementation of EA. From an architectural perspective the third assumption is the most important since it specifies a need for the system architecture to be 'scalable'. In satisfying this assumption, a designer may choose to use a modular architecture. As will be elaborated in Chapter 4, a modular design allows the designer the freedom to make certain changes to a design without the need to trace their impact upon every possible inter-dependency within the system. Such a system is not infinitely flexible, i.e. there will be early design choices which cannot be easily changed in later increments. However, when compared to an integrated architecture, the number and scope of the such choices is minimized.

The textual definition from Boehm and Aldridge do not clearly define the relationship between increments in EA. The notional time-line from (Delaney, 2000)[2] in Figure 2-1 provides a clearer depiction.

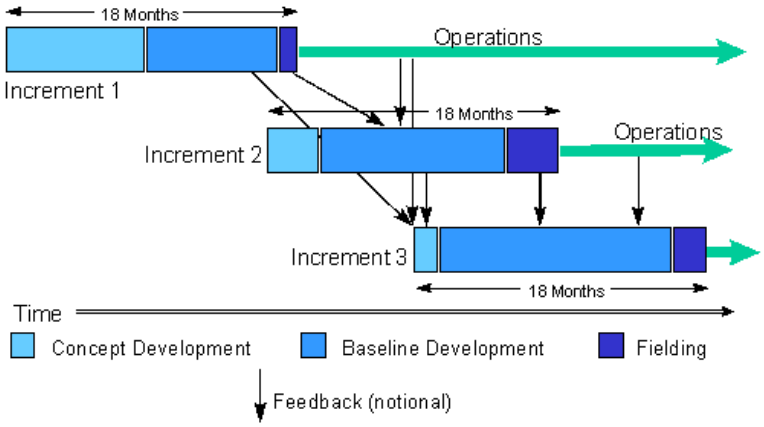


FIGURE 2-1: Notional EA increments from (Delaney, 2000)

[2]Delaney's discussion in AFI 63123 is specific to C2 systems, however the ideas expressed are applicable to non-C2 systems. Air Force documents that are not directly related to C2 systems, e.g. the Operational Test and Evaluation handbook and various program managers guides cite AFI 63123 as a good source for understanding EA and SD from an Air Force perspective.

Delaney breaks each increment into three phases. The first, Concept Development, takes operational needs and available solutions and produces well-defined requirements to be met by the end of the increment. In addition, a concept or concepts are endorsed as viable methods of achieving those operational needs. Should no further development be deemed necessary, the process is halted. If, however a need and a possible solution are identified, the second phase, Baseline Development, is initiated. In the second phase, the results of the Concept Development are refined into a system which can be fielded. Baseline Development concludes when the system is accepted for fielding. The final phase is Fielding and Operations.

As shown in Figure 2-1, when the the first increment completes development and is being fielded, concept development begins on the second increment. This time, the starting point is the recently fielded system. Lessons learned from the first increment's development (and later operation) are fed back into the concept and development for the second increment. The same is true for the third increment with respect to the second. Note that the increments need not proceed sequentially; rather, as is shown, two increments may be occurring in parallel. This makes sense since the instigating event for the increment was an emergent need and capability. For example, if in the middle of an increment a new stealth technology emerges work can immediately begin on incorporating it or if world events dictate operations in a new environment, an increment my be spawned to make needed changes. Once the new capability has matured, it is added to the product.

Another characteristic of an increment is the use of a fixed time horizon, i.e. the length of increment is kept fixed. A fixed time horizon ensures that capability is delivered to users at regular intervals. A question that immediately comes to mind is "What if some portion of the need specified in Concept Development cannot be met within the prescribed time limit?" For this reason feedback occurs between the baseline development phase of increment 1 and the concept phase of increment 3. There were some items which could not completed in increment 1, but then, because of some external development, say an

improvement in technology, those items left out now seem achievable and are included.

Note that the lengths of the three phases within each increment changes from one increment to the next. This reflects the fact that as capabilities are added, the degree to which the systems can change decreases. Eventually, the initial concept will become so dated that augmentation is no longer a option, and fresh effort should begin. Boehm identifies this reality in his fourth embedded assumption. When the 'evolving-out of legacy components' is no longer possible, the EA should cease[3]. The speed at which this decrease in ability to adapt the system occurs is largely dependent on the architecture initially chosen for the system. As will be argued later, a modular architecture offers a larger degree of flexibility[4] and can thus be 'evolved' through more iterations than an integrated architecture.

2.2 Spiral Development

With Evolutionary Acquisition well defined, the question of what goes on in each increment arises. Aldridge pointed out two possibilities (see Chapter 1). Scenario 1 involves a simple waterfall flow from concept through fielding. Since all requirements are known at the outset a traditional acquisition approach can be user. Much more interesting is Scenario 2 in which evolution is needed because of changing needs. Aldridge suggests Spiral Development as an appropriate process.

The remainder of this section will examine spiral development from several perspectives. Two are from academics: Boehm and Unger; one is from practitioners: Johnson and Johnson.

[3]This does not mean that the system should be abandoned, only that further development using an EA is not possible. The system could still operate for many years.

[4]Here, flexibility is defined as the ability to be adapted to environments and to satisfy needs that were not known at program inception.

2.2.1 Academic Perspectives on EA/SD

Boehm correctly links the incremental nature of EA with the cyclic nature of SD (see Boehm's definition of SD provided in chapter 1). In particular he identifies a set of six-invariants that define an SD process and ensure that the assumptions of EA stated above are satisfied. Before listing the invariants, a few definitions[5] are in order: An artifact is a specific instantiation of a design. Early in the design process these may be runs of a computer model, soon becoming prototypes and eventually delivered products. Stakeholders are those persons and/or organizations who seek to derive a benefit and in return contribute resources to the development effort. For example, users derive the benefit of use of the product in return for money, while designers derive payment in return for time and effort spent on the design. With these definitions in place, Boehm six spiral invariants are[6]:

1. Concurrent rather than sequential determination of artifacts in each cycle.

2. Consideration in each cycle of critical stakeholder objectives and constraints, product and process alternatives, risk identification and resolution, stakeholder review and commitment to proceed.

3. Level of effort on each activity within each cycle is driven by risk considerations.

4. Degree of detail of artifacts produced in each cycle is driven by risk considerations.

5. Managing stakeholders life cycle commitments via the LCO, LCA, and IOC anchor point milestones.

[5] Based on Boehm's definitions

[6] Boehm's terminology best applies to the first of Delaney's increments. In subsequent increments the same products are produced/refined, but the names may change slightly. For example, referring to the system fielded after the second increment at the 'Initial Operating Capability' doesn't make much sense. A better term might be 'Upgraded' or 'Enhanced Capability'

6. Emphasis on system and life cycle activities and artifacts rather than development activities and artifacts.

The message here is to proceed concurrently, while guaranteeing the inclusion of all relevant stakeholders and in appropriate sized chunks. Risk considerations determine when a stakeholder is involved when and the size of the chunks.

Roberts (2003), in referring to Boehm and Delaney, emphasizes the importance of three milestones or anchor points in Boehm terminology that serve as progress checkpoints and ensure proper level of commitment by stakeholders as the development progresses. Again using Boehm terminology these anchor points are: (1) Life Cycle Objectives (LCO), (2) Life Cycle Architecture (LCA), and (3) Initial Operating Capability (IOC). The first or concept phase of development (in Delaney's terminology) ends with the establishment of the LCO's. From the Air Force perspective (as described in (Delaney, 2000)), one can think of this phase as defining key yes/no questions that determine military utility of a design option. The goal in this phase is to define objectives such that any design meeting those objectives will have military utility. In addition, a feasibility rationale must be included ensuring that at least one feasible concept exists that can satisfy the objective. All stakeholders must agree on the rationale and the underlying objective for the development to proceed to the next phase, baseline development.

Whereas the concept development phase focuses on defining clear objectives and obtaining buy-in from stakeholders with regard to the feasibility of those objectives, the baseline development phase seeks to evolve the vague feasibility rationale into a specific architecture that can serve as a basis for initial operational capability. Within the spiral development process, such an architecture should allow not just a single operational instantiation, but several scalable versions. This life cycle architecture serves as a foundation for future development (note in Figure 2-1 the link between Baseline Development in the first increment and concept development in subsequent increments). Again, stakeholder acceptance is critical. Recalling that SD is a risk-focused activity, the

LCA must identify and either mitigate or manage all major program risks. The mitigation efforts could involve abandoning part of the concept if its success cannot of assured in the time allotted for the increment. In the end, however, a minimum level of 'core operational capability' must be delivered. The level of 'capability' required is determined by the stakeholder's willingness to field the system. This fact emphasizes the importance of the concept development phase. If there is a large gap between what is promised at in the LCO and what can actually delivered for IOC, users stakeholder will not accept the system for fielding. To prevent this, the length of each increment should be kept short. Of course in developing both LCO and LCA iteration will be necessary. This is clearly stated in the first invariant. However, once convergence is reached LCA is powerful object. Within the LCA is an assurance that the minimum objectives ('core operational capability') will be met, a plan for managing risk, and an architecture for implementing the plan while meeting the objective.

With a somewhat mature LCA in place, the development process moves to the third phase, fielding and operations. Here, the first operations occur. The LCA is instantiated into an actual product (known as the initial operating capability or IOC) that delivers actual value. Built from the LCA, the IOC can adapt to the changing environment envisioned in LCA. The spiral development process can be summarized in the diagram in Figure 2-2 which refers to a typical software development effort.

One begins at the center of the figure and then proceeds through each of the activities. Each activity refines and further specifies the three anchor points. Eventually, operational capability is reached.

Whereas Boehm discusses spiral development in an attempt to precisely define it and prescribe a structure for carrying it out, Unger focuses on the differences between SD and other product development processes. In trying to understand the design of product development processes, Unger defines several metrics to distinguish between processes. They are listed in Table 2-1.

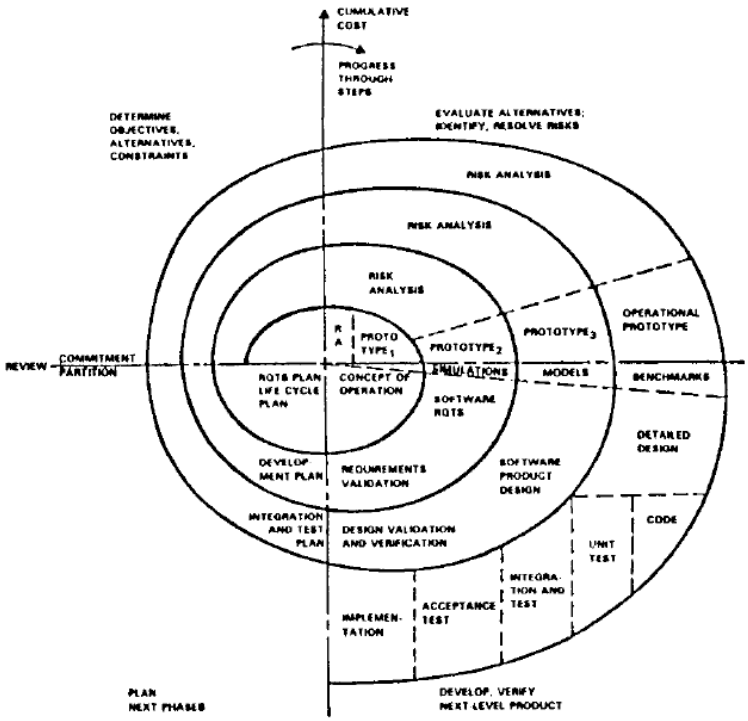


FIGURE 2-2: Boehm's spiral model from (Boehm, 2000)

Category	Metric	Score for SD and Meaning
Iterations	Breadth of Iterations	3 – Iterates over many phases
	# of inter-phase loops	Many
	Planning	5 – planned and scheduled
Reviews	Rigidity	5 – Less rigid
	Frequency	Unspecified
Risk	Profile of key Risks	Manages market risk well

TABLE 2-1: Spiral development as measured by Unger's Metrics

To validate these metrics, Unger completed several case studies. After reducing the metrics into a composite score, he found that among the organizations studied, those that used a spiral process had less frequent and less rigid reviews. The iteration metrics reveal that many, planned and comprehensive reviews are undertaken when using the spiral process. All of the organizations that used SD were software developers. Unger concludes that the few reviews and much iteration attributed to SD reflects the types of risks faced by organizations employing SD. In particular, he found market risk was significant. By having many iterations, and including in each iteration interaction with the users (see Figure 2-2), the organization is in constant communication with the market. The lack of rigidity imposed by review allows the process to be flexible to a changing market – old decisions need not be adhered to strictly.

Unger also looked at the types of products developed by each organization. One thing stood among those using SD – ease of integration. Being software companies, they essentially had zero building costs associated with integrating components into complete product. Since SD requires many 'builds' of the product to gain feedback from users, keeping the cost of such 'builds' low is clearly very important.

In applying SD to military[7] aerospace systems, Unger's results have implications both in acquisition policy and system architecture. The need for flexibility in the spiral process requires a certain amount of faith on the part of regulators that stakeholders will keep the process under control without having many reviews and rigid decision gates. Observing Boehm's invariants 2-4 can help accomplish that. In terms of architecture, Unger found that ease of integration is key. This may pose the greatest difficulty, since, typically, aerospace products are not known for their ease of integration; rather, enormous complexity is commonplace.

The perspectives of Boehm and Unger are primarily those of academics examining the process from the outside. To gain better insight into the practice of EA/SD, some

[7] Unger's notion of the market can be replaced by a dynamic threat environment.

perspective from the aerospace industry and the acquisition community is needed.

2.2.2 A Practical Perspective on EA/SD

Johnson and Johnson (2002) provide a practitioner's view of spiral development. They define SD, using terms similar to those of the authors discussed earlier, as:

> "A set of acquisition activities incrementally incorporated into an evolving baseline. Each increment or spiral increases capability and does so in a rapid pace, with each spiral building on the previous spiral and spreading risk and development costs over a longer period of time. Each spiral is made up of one or more projects developed independently to the maximum extent possible. When each of the developments is ready, it is dropped into the production baseline. Testing, both internal to the program (DT&E) and external (IOT&E) is done incrementally."

Their definition echoes concepts introduced by Delaney and Aldridge. In particular, reference to parallel development of new capability is also found in Delaney. Though both Boehm and Unger allow for such development, they do not put the same emphasis upon it. In comparing SD to other methods such as P3I, Johnson and Johnson reiterate Aldridge's point that capability delivered in a particular spiral is not fully specified at the beginning of the program; rather, it is defined based upon the success of previous spirals, changing requirements priorities, or changing budgets. Though not explicitly included by Johnson and Johnson, new technology should also be considered when planning an increment. With SD operationally defined, Johnson and Johnson move to the implementation of SD.

Echoing Boehm's requirement that stakeholders be involved in defining the needs to be satisfied and major risks to be resolved, Johnson and Johnson (as does Delaney) recommend forming a team composed of the user, the contractor and the Pentagon staff

supporting the program. This team should meet on regular basis reevaluating the goals of each spiral. Johnson and Johnson emphasize the need for agreement among these key stakeholders and proper documentation of the strategy they develop.

For the team structure to be effective, the user must trust the acquisition community and the developer to deliver the final capability at a later date even though IOC specifies a less capable solution. The user must be willing to accept this partial capability, test it, and use it reporting back to the other stakeholders lessons learned. Then, in conjunction with the other stakeholders, make appropriate changes to the requirements document. Though the benefits of the flexibility offered by SD are easy state, users can be unwilling to accept partial capability (though still militarily useful) especially when they are not used to getting what is requested.

Once requirements are properly specified, development of the product (or system) presents additional challenges. In SD, a variety of new capabilities can be developed in parallel becoming part of the product as they mature. For such a strategy to be successful, the parallel developments must be independent. Independence has an additional benefit in that it reduces development risk by reducing the number of items on the critical path. Johnson and Johnson give the example of a program with five parallel developments but one that is critical for the next increment. The developer and the government can pay special attention to the critical item and ensure its success without having to worry about the impact of failures in the other four efforts.

Obtaining such independence can be difficult, especially when users demand improvements in overall capability as opposed to specific sub-system modification. The flexibility gained by this independence has a hidden cost that Johnson and Johnson point out: there may be an increased risk of budget cuts since the program *can* adapt to less funding by, for example, delaying the least important of the parallel development efforts. Contrast this situation with a program in which the cuts will prevent the delivery of *any* operational capability. It is likely that leaders will choose to cut the spiral program. This,

of course, creates a perverse incentive toward program inflexibility. National leaders need to use their oversight authority to prevent this from occurring.

In fielding systems developed using a spiral process, Johnson and Johnson point out the unique logistics challenge faced by the user. Since SD involves upgrading fielded system over time, several variants of a system may be in use simultaneously. Training and logistics issues are significant in such a situation. Hence representatives of those communities need to be involved in the SD planning process.

Overall, Johnson and Johnson describe many challenges, including those enumerated above, faced by programs using a spiral process. Programs which for technical, political, or other reasons cannot meet these challenges should not use a spiral approach. If, however, SD can be used, the rewards are significant (as delineated by Johnson and Johnson):

- Capability delivered to the user quickly
- Risk spread across a series of spirals – less reliance on a particular 'miracle' technology to come to fruition
- Formal integration lesson learned and feedback from users making the process much more responsive to dynamic user needs
- Faster integration of technology enabled by a flexible architecture

To conclude the discussion of EA/SD, a real-world example, the Global Hawk is discussed below.

2.3 Real-world example of EA/SD: Global Hawk

The Global Hawk[8] is a high altitude, uninhabited aerial reconnaissance platform. With loiter times in excess of 24 hours, and the ability to be fitted with a variety of sensors, it represents the future of long-range, long-term aerial reconnaissance.

Global hawk transitioned from ACTD[9] to EMD[10] in the winter of 2001. The capability produced as a result of the ACTD was not sufficient to satisfy the user. Furthermore, the initial plan for EMD and subsequent phases would have required seven years to achieve the demanded performance. The Commander of Air Combat Command challenged Global Hawk to field additional capability much sooner than outlined in the initial plan.

To meet the Commander's challenge the Global Hawk team transformed the program in the summer of 2001 to take full advantage of the spiral approach. The various projects in the initial plan were logically broken down and prioritized to allow quick fielding and independent development (where possible). The user conducted a 'requirements conference' to capture evolving needs faster that the traditional ORD[11]-based process. The result was a dynamic program that could adapt to changing needs and field new capabilities much quicker that the seven years stated in the initial plan.

Global Hawk's first spiral involved developing a baseline platform which could enhanced in the future. Since all the elements of the platform were needed to reach IOC (in Boehm terminology), they all needed to the completed prior to production (see Figure 2-3). The baseline capability will be delivered in FY03.

Subsequent spirals, on the other hand, will bring new capabilities into production as they matures. See, for example, the plan for the second spiral in Figure 2-4. Note that there are

[8] This discussion of Global Hawk is based primarily upon Johnson and Johnson (2002).
[9] Advanced Concept/Technology Demonstrator
[10] Engineering & Manufacturing Development
[11] Operational Requirements Document

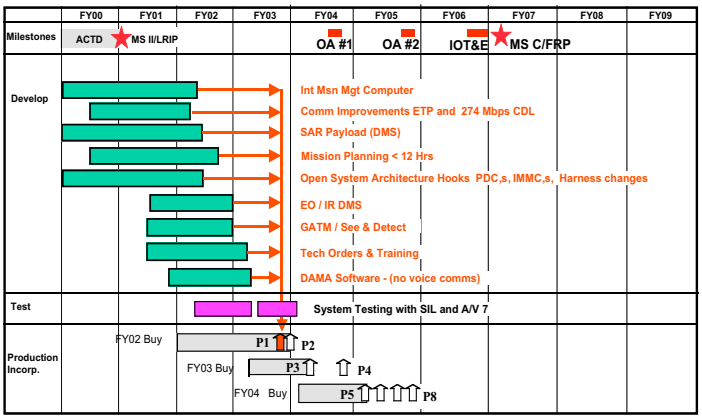


FIGURE 2-3: First Spiral plan for Global Hawk (from Johnson and Johnson (2002))

several operational assessment, or 'OA's, planned to obtain feedback from users. Statutory test requirements are still adhered to however, with IOT&E occurring prior to full-rate production.

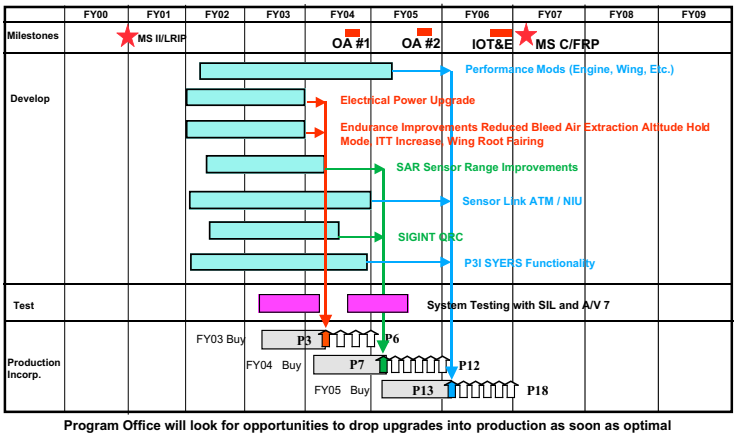


FIGURE 2-4: Second Spiral plan for Global Hawk (from Johnson and Johnson (2002))

2.4 The importance of MATE and Modularity

MATE and modularity are both useful in implementing the spiral development process. During the concept development phase, accurate capture of diverse user needs is crucial since it governs the objectives for subsequent phases. The MATE process formalizes needs capture. The MATE process can be used to bring together the needs of multiple stakeholders[12] (as advocated by Boehm) into unified measure of value.

Modularity can be a key enabler for the spiral process. The earlier a design decision is made, the less it benefits for the experience gained during the design process. As will be explained later, modularity allows one to delay certain design decisions in the design process by placing them within a single module. By isolating these decisions to different modules, the independence advocated by Johnson and Johnson is achieved. This creates the flexibility needed for concurrent design. Furthermore, by decoupling the system into small, manageable pieces, it decreases the need for many comprehensive reviews[13]. This also aids in integration. Both are factors that Unger identified as typical of SD.

The Global Hawk demonstrates these advantages of modularity. By using a modular architecture in which sensor and avionics hardware are kept largely independent of the air platform, the Global Hawk team was able to upgrade those capabilities without redesign of the underlying airframe. This independence enabled the rapid introduction of new technology into the production aircraft as shown in Figure 2-4. Modification of the airframe is also possible (see the final upgrade during spiral 2). Since such modification is more difficult and may impact other systems, it is left to later in the spiral.

Making a modular architecture may not be possible for all systems. Most modular systems suffer a performance penalty relative to an integrated system at the same cost.

[12] This assumes that there exists a supra-decision maker as is often the case.

[13] Instead, once the modular structure is defined, early reviews can focus on particular modules; later reviews on integrating the system as a whole

The performance penalty stems from the preference for decoupling between module over the precise tuning often needed to obtain the high efficiency found in high performance systems. For example, to obtain high rotation rates in jet engines, each piece of rotating turbo-machinery must be precisely balanced. Since the pieces must balance as a whole, this can create a coupling between the designs of the individual components. Lowering the rotation rate would allow for looser tolerances on balance, but there would be a performance penalty.

Convincing decision makers to accept this performance penalty in return for greater flexibility of a modular system can be difficult. Even if such a penalty can be avoided from a technical perspective, there may not be sufficient resources available to design a modular architecture in addition to the individual module development. For further discussion of the limitations of modularity see Chapter 4.

Chapter 3

Multi-Attribute Tradespace Exploration with Concurrent Engineering (MATE-CON)

The reader is strongly encouraged to refer to (Ross, 2003) for a complete description of the MATE-CON process.

In the previous chapters, Evolutionary Acquisition and Spiral Development were described. Furthermore the case was made for why the EA/SD approach can alleviate some of the difficulties faced by the US aerospace industry. This chapter introduces a formal process that will ensure satisfaction of Boehm's EA assumptions and SD invariants. Roberts suggests that the MATE-CON process developed by Ross and Diller is ideally suited to EA/SD as defined by Boehm. Before summarizing Roberts claim, a brief description of MATE-CON is in order.

3.1 Description of the MATE-CON process

Adam Ross and Nathan Diller developed the MATE-CON process in 2002. Their aim was to create a new method of space system conceptual and preliminary design that did not suffer from some of the short falls discussed in chapter one. In particular, they identify (1) the cost associated with premature imposing of requirements and (2) subsequent requirements creep as the key sources for cost growth and schedule slip. Recognizing that many requirements are decided during conceptual and preliminary design, they propose a new methodology for conceptual design that attempts to mitigate requirements creep. The new process, know as Multi-Attribute Tradespace Exploration with Concurrent Engineering (MATE-CON) breaks down the conceptual and preliminary design process into three stages[1].

The first stage, needs identification, results in a quantitative description of the needs to be met by the product of the design effort. This stage seeks to answer the question "What should it do for me?" The next stage, architecture-level analysis, defines the space of possible solutions and then evaluates them against the needs identified in the first stage. Architecture-level analysis seeks to answer the question "How well can I meet the needs?" The result of the second stage is a set of architectures[2] to be explored in greater depth. These first two stages constitute the MATE portion of MATE-CON and are depicted in Figure 3-1.

After the second stage, the space of possible solutions will have been reduced to a smaller, Reduced Solution Space (RSS), that represent the stakeholder perceived most promising ways of satisfying the needs. One may think that the RSS should consist of exactly one architecture, the 'optimal' method of satisfying the needs identified in stage one. This

[1] The first two stage constitutes MATE and the third is CON.

[2] Here the term architecture means a sufficient description of a method of satisfying needs to allow a design (or detailed description of the engineered product) to proceed. In common parlance, an architecture can be thought of a system-level design.

will, in general, not be the case. Each solution in the RSS will require the expenditure of some resources (time, material, money, etc.). As part of stage two, an estimate is made of the resources required to instantiate each architecture. The result is a space of resource expenditure vs. need satisfaction. The RSS represents those architectures that, for a given level of expense, maximize need satisfaction.

The third stage, design-level analysis, takes one or more members of the RSS and adds in the necessary sub-system detail to produce a preliminary design. It may seem that MATE-CON is largely sequential process. This is, in-fact, not the case. Rather, at each stage, opportunities exist to revise and update the results of prior stages. To more fully understand the MATE-CON process, each stage is described below. Figure 3-2 extends Figure 3-1 to include the third stage which constitutes CON. The process is not linear. There are feedback loops between steps.

3.2 Needs Identification

Two questions are answered during the needs identification stage. First, “Whose needs are being satisfied?” and second, “What are their needs?” The first question is process of stakeholder identification. Stakeholders are those who contribute some resource to the design process in order to extract some benefit (a need satisfaction) from the process.

3.2.1 Stakeholder Identification

In MATE-CON, stakeholders are divided into four major groups: Users, Customers, Firms, and Designers. Users are those who will directly interact with the product when it is in service. For example, consider a military imaging satellite. The users would be the air force personnel who operate the spacecraft on a day-to-day basis. The benefit they

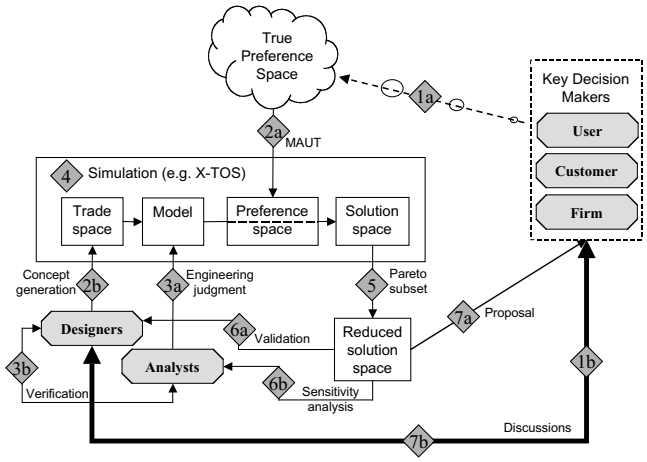


FIGURE 3-1: MATE Process overview (Ross, 2003)

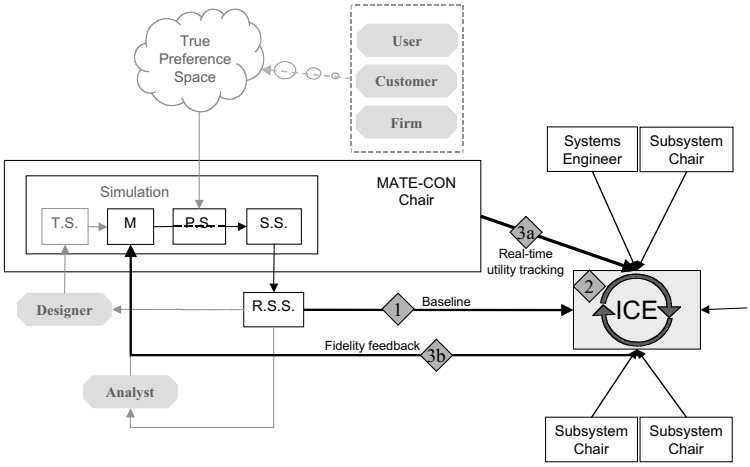


FIGURE 3-2: MATE-CON Process overview (Ross, 2003)

derive are images of a battlefield. The customers are one level removed from the product. They derive benefit indirectly from information produced by the users. The customer for the military imaging satellite would be the commander who uses the assessment of the battle made by the imagery experts to evaluate past and plan future operations. The benefit he derives is improved knowledge of the state of the battlefield as elicited from the images. The firm is the next level up in this hierarchy of needs. For now, they and any higher levels will be ignored since they have not been explicitly included in any case study of MATE-CON process[3]. Designers (and analysts[4]) derive the benefit of money for their services.

The distinction being made between users, customers, and firms creates a hierarchy of needs. Additionally, it highlights the fact that expenses are not always paid by the direct users of a system. In the case of the military imaging satellite, it is not the users who must pay for the system; rather, funds are provided through a higher-level source. This source, at a very abstract level, is the taxpayer, but, practically speaking, budget allocation for the system will be done at some command level within the Air Force. It is this level of stakeholder that is usually referred to as the customer.

Though all stakeholder derive value from the design effort, decision makers form an important subset of the stakeholders. Decision makers are those stakeholders that control the allocation of resources needed for the design effort and as such their assent is needed for the effort to proceed. MATE-CON therefore focuses on satisfying the needs of decision makers. Usually, decision makers exist at the customer or firm level of the hierarchy of needs; however, their principle need is to ensure that the needs of the users below them are met. Thus users are often used as proxy decision makers.

[3] In fact only the users needs have been explicitly included in examples of MATE-CON's use to-date. The customer is described here to provide the reader with a sense of the hierarchy of needs.

[4] A designer creates a design while an analyst evaluates it. For the purposes of this discussion, the difference is unimportant.

3.2.2 Stakeholder needs identification

Once decision makers are identified, their needs must be quantified. MATE-CON uses Multi-Attribute Utility Theory (MAUT) to accomplish this. MAUT is a framework for rational decision making under uncertainty developed by Keeney and Raiffa (1993). In MAUT, each decision maker defines a set of quantifiable attributes of their need(s) to be satisfied. These attributes put a quantitative face on the value derived from the system. Formally, an attribute is a decision maker perceived metric that measures how well a decision maker specified objective is met. For military imaging, user attributes may be image resolution, coverage level, targets reachable per day, etc.; a customer attribute could be percent improvement in battle damage assessments; a designer attribute could be profit for the design firm. The attributes need not represent real physical quantities as in this example; they only need to quantitatively reflect the degree to which a need of a stakeholder is satisfied as perceived by that stakeholder. Each attribute has a specified range which defines the minimum and maximum acceptable value.

To fully cover the decision maker's preference space, Keeney and Raiffa require that the set of attributes be complete, minimal, operational, decomposable, non-redundant, and perceived independent. Complete means that the decision maker is indifferent between any two alternatives that have the same attribute values. Minimal, in contrast, means that if any attribute is removed, completeness will be lost. Operational means that the decision maker has a preference over the attributes. Decomposable means that the attributes can be quantified. Non-redundant means that each attribute describes a different aspect of the preference space. Finally, 'perceived independent', means that, in the mind of the decision maker, variation of one attribute does not imply variation in another. Attributes may be coupled in reality – independence need only exist in the mind of the decision maker. This final characteristic is the most important since it allows evaluation of multi-attribute preferences without having to elicit preferences on all possible combinations of attribute values. Finding a set of attributes that strictly satisfy the above conditions is often

difficult. The designers must do their best to work with the decision makers to produce an acceptable set of attributes.

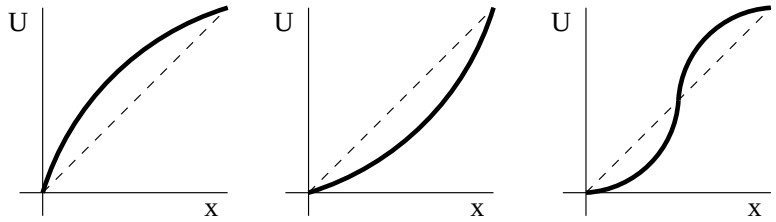
The attributes alone are not sufficient to quantify need. Since at the transition between MATE and CON, the value delivered by the various solutions found during stage 2 must be traded against each other in choosing members of the RSS. Since SD is a risk driven process, the risk preference of the stakeholder for each attribute must be captured. Multi-Attribute Utility Theory (MAUT) developed by Keeney and Raiffa provides a framework for doing exactly that.

First consider the single attribute problem, i.e. a decision maker whose preference can be quantified along a single dimension. To simplify the discussion, it is assumed that the value delivered increases as the attribute increases. It is unlikely that the decision maker has a single value of the attribute above which his needs are satisfied. Rather, there will be a range of attribute values over which the degree of need satisfaction increases. More precisely, the lower bound, x°, of the range is defined as the attribute value below which sufficient benefit will not be delivered to gain acceptance by the decision maker. The upper bound, $x^\star$, is defined as the attribute value above which additional benefit is not delivered with increasing attribute value. The utility function, $U(x)$, is then defined as follows:

$U(x^\circ)$ is set to zero; $U(x^\star)$ is set to one; for $x^\circ < x < x^\star$, the value of $U(x)$ is chosen to reflect the risk aversion of the decision maker. The exact definition of $U(x)$ is dependent upon the method by which the utility function is elicited from the decision maker. One simple method is the certainty equivalent probability [5]. Here, the decision maker is offered a choice between a system that delivers the attribute value x with certainty and one that delivers $x^\star$ with probability p, x° with probability $1-p$. $U(x)$ is set to the value of p at which the decision maker is indifferent between the two systems. Some notional utility

[5] Certainty equivalent probability is not the preferred method for utility function elicitation. Other methods, such as lottery equivalent probability perform better in terms of consistency across repeated interviews and reduced bias. See (Keeney and Raiffa, 1993; de Neufville, 1990; Seshasai, 2002).

curves are plotted in Figure 3-3. The x axis range is from x° to $x^\star$. The U axis range is from $U(x^\circ) = 0$ to $U(x^\star) = 1$. The dashed line represents a 'risk-neutral'



(a) Risk Averse (b) Risk Seeking (c) Threshold

FIGURE 3-3: Examples of single attribute utility functions with different risk aversions

As indicated in Figure 3-3, the decision maker's aversion to risk is evident in the shape of the utility function. For example, the risk averse case, (a), has a concave shape. This reflects the decision maker's aversion to risky situations. To understand why this concave shape indicates a risk averse decision maker, consider $p = U\left(x = (x^\circ + x^\star)/2\right)$. From (a) it is clear that $p > 0.5$. The certainty equivalent lottery that defined p indicates that the decision maker is indifferent between receiving x for certain, and a lottery that delivers $x^\star$ with probability p, x° with probability $1 - p$. The expected value of this lottery is given by $px^\star + (1-p)x^\circ$. Since $p > 0.5$, this expected value is larger than x. To be indifferent between a certain event and an uncertain event, the decision maker required that the uncertain event have higher expected value. Thus, the decision maker is risk averse. By a similar argument, the convex curve in (b) reflects risk seeking behavior. In graph (c), the decision maker transitions from being risk seeking to risk averse. Building upon this understanding of the single-attribute problem, the multi-attribute problem is now considered.

Drawing upon utility theory as developed by von Neumann and Morgenstern (described above), Keeney and Raiffa developed Multi-Attribute Utility Theory (MAUT). They define a metric, the Multi-Attribute Utility (MAU), which can be used to rank possible solutions in each decision maker's preference space. Each solution is mapped onto a scale

that ranges from zero to one where zero means 'minimally acceptable on all attributes' and one means 'fully meeting all needs'. By the axioms that define MAUT, the rational decision maker will prefer solutions which rank higher on this scale. Note that MAU is an ordered metric scale, e.g. an MAU of 0.5 is not twice as good as 0.25. This is analogous to centigrade temperature where 100 degrees is hotter than 25 degrees, but not four times as hot.

The two key assumptions required to compute the multi-attribute utility are pairwise-preferential independence and utility independence. Pariwise preferential independence means that if a decision maker, for example, prefers 3 units of attribute i to 1 unit of attribute i, then that preference holds regardless of the value of any other attribute. Utility independence means that the shape of single attribute utility function $U(x_i)$ does not depend on the values of the other attributes. Utility independence ensure that the single attribute utility curves need not be evaluated for all possible values of the other attributes. This makes the evaluation process tractable.

When these assumptions are satisfied, then the multi-attribute utility is of the form:

$$KU(\bar{x}) + 1 = \prod_{i=1}^{n} [k_i U_i(x_i) + 1] \tag{3.1}$$

where

$\bar{x}$	is the vector of n attribute values
$U(\bar{x})$	is the multi-attribute utility
x_i	is the value of the i^{th} attribute
$U_i(x_i)$	is the single attribute utility of x_i
k_i	relates to the importance of attribute i in relation to the others
K	is a normalization constant

Engineers often want to use a simpler function such as a weighted sum of the attribute values to aggregate performance on the attributes into a single metric. Such techniques, however, are arbitrary in form[6] and make implicit assumptions about the decision makers'

[6]The metric is quite sensitive to the definitions of the weighting factors.

preferences[7]. de Neufville (1990) discuses the limitations of such techniques which lack the axiomatic underpinnings of MAUT.

Though this framework allows for the involvement of multiple decision makers, for the sake of clarity, the remainder of the discussion will focus on the situation where the user is the only decision maker.

3.3 Architecture-level Analysis

Once a metric for the preference space has been defined (1a in Figure 3-1), the design process can move on to developing solutions to meet the expressed needs. With the hope of exploring as wide a variety of solutions as possible a parametric model is built. The model begins with a vector of design parameters that constitute a complete list of the system-level decisions that are needed to specify an architecture, i.e. each architecture corresponds to a particular choice of design vector. Each possible choice for the design vector is then processed through a simulation that estimates the attribute levels delivered by that choice of design vector. These attribute vectors collectively form the attribute space. Finally, the attribute space is mapped onto the decision maker(s) preference space by the MAU function. Taken as a whole, the model maps the parameterized space of possible designs[8] to the preference space of the decision maker. It is labeled (4) in Figure 3-1. Roberts (2003); Spaulding (2003) have discussions of the advantages and limitations of parametric models used in MATE.

Given the above mapping of designs onto a preference metric along with an estimate of the expense of designs, decision makers can rationally choose which design(s) form the reduced solution set and thus warrant further study. The most common method for

[7]For example, risk aversion is often not explicitly captured.

[8]This space is known as the tradespace since it consists of the major trades that must be made at the system level.

choosing the RSS is to plot all of the designs evaluated by the parametric model during trade space exploration on a cost vs. MAU space, and then include those designs that maximize MAU for a given cost (see Figure 3-4). Such designs are referred to as Pareto optimal or efficient. Improving upon MAU will requires greater cost, thus they make the best use of available resources among those designs considered.

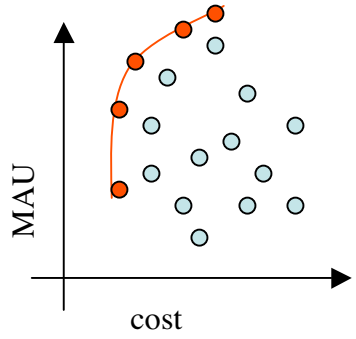


FIGURE 3-4: Plot of cost vs. MAU in which each point represents the cost incurred and MAU delivered for a particular choice of design vector. The reduced solution space, RSS, is indicated in red.

This process of producing a model and then finding an RSS will iterate until the stakeholders (in this case, users and designers) agree that the RSS constitutes a feasible set of architectures and that those architectures deliver the estimated utility at the estimated expense. Such iteration is often necessary because after seeing the 'best' solutions decision makers will realize that their preferences as expressed in the utility functions were mis-stated and will want to revise them. This iteration follows the loop 7A $\rightarrow$ 1A $\rightarrow$ 2A $\rightarrow$ 5 in Figure 3-1. Each iteration will allow for a better understand of the decision makers' true preference space as well as the dynamics of the tradespace.

As described above, the RSS can consist of multiple designs, however implementations of MATE-CON to date have always used a single member of the RSS for the CON portion of the process. The choice of which design to use during CON is made by the decision makers in consultation with other stakeholders.

3.4 Design-level Analysis

Once all stakeholders agree on an architecture(s) for continued development, the CON process occurs. In this process, the system-level design provided in the design vector corresponding to the chosen architecture in the RSS is built upon to produce a detailed design that can be carried forward into production. A concurrent engineering approach is used to ensure all stakeholders are represented in the design process. The MAU metric is used to measure progress of the design effort through the MATE-CON chair in the concurrent engineering effort. Feedback also occurs between stages 2 and 3. Knowledge from the tradespace is used in developing models for stage 3. The greater fidelity of models used in stage 3 can elucidate poor assumptions made in stage 2. In that case, re-evaluation of the tradespace will be necessary. Diller (2002) has greater detail on CON. The result of stage 3 is a design that all stakeholders agree can be instantiated.

3.5 MATE-CON and EA/SD

To date, the MATE-CON process has only been used for preliminary design of space systems. Roberts (2003), however, argues that the MATE-CON process is well suited for implementing EA/SD.

3.5.1 Methodologies for system architecting

In considering different system architecting methodologies for EA/SD, Roberts (2003) discusses four methodologies described by Maier and Rechtin (2000): normative, rational, participative and heuristic. A normative architecting methodology can be described as ‘doing things the way that they have always been done’. Here, there is reliance on technical standards and manuals where making design choices. While this approach may

be suitable for well-understood systems with simple designs, it does not work well with the systems for which the Air Force is using EA/SD. These systems often have vague requirements and complex design involving cutting edge of technology. The rational methodology refers to architecting strategies that rely upon engineering and computation seeking a technically optimal solution within some constraints. In the participative method, the "right" answer is defined by the assent of the key stakeholders. It seems to work better than the other methods when requirements are ill-defined and/or fluid. The final method, heuristic relies upon the architects' experience to devise a solution the "looks right". In the context of complex system architectures, Roberts points out that each of Maier's methodologies may be useful at various times in the design process. For example, at the start of the design effort, a heuristic approach may be taken to intuitively sketch possible solutions. This could shift to a more participative approach as greater stakeholder assent is needed to gain resources required to develop the design in greater detail. In turn, these stakeholders may demand a more rational approach to justify their expenditure. Finally, once the design is well-codified and understood, a normative approach may be taken.

Turning back to EA/SD, it would seem, as Roberts points out, that the participatory approach is best suited to EA/SD given the emphasis of stakeholder commitment in both Boehm and Delaney. A purely participatory approach risks becoming 'design by committee'. Without an objective guiding metric, such design methods often lead to sub-optimal solution if any solution at all. Traditionally, designers and customers have responded to this problem by imposing requirements early in the process so as to bring a sense of rationality to what is inherently a participatory process that will involve negotiation and learning. Such a need to jump from participatory to rational design is understandable given the large resource expenditures involved in the complex aerospace systems being discussed. However, as was discussed in chapter 1, such premature imposition of precise requirements can often lead worse results than if the requirements are left vague until more was known. The reason for this is that prematurely specified

requirements are bound to change, and there will be a significant cost penalty incurred as the design adopts the new requirements. The MATE-CON process solves this problem by allowing participative design within the rational framework of MAUT. As long as the decision makers agree with and satisfy the axioms underlying MAUT, then from their perspective, the design process is a rational method. Thus, Roberts concludes that the MATE-CON process is well suited to EA/SD.

3.6 Modularity

To review, thus far certain problems were identified with large aerospace programs. EA/SD was offered as a possible solution to these problems. The MATE-CON process was suggested as a formal framework for implementing EA/SD. A key step in the MATE-CON process is the selection of a baseline architecture that serves as the basis for further development. Traditionally, MATE-CON has chosen a single point in the tradespace to carry from MATE to CON. There are many criteria upon which this choice can be made. One may opt for the highest performing architecture that meets some budget constraint (the Pareto optimal choice). On the other hand, one may value flexibility leading to a choice that prefers room for expansion at the expense of lower initial performance. In reality, the best choice is a compromise between these two extremes. As stated in Boehm's EA assumptions, a high enough level of performance must be delivered to ensure key stakeholder commitment, while the architecture must also be 'scaleable', i.e. adaptable to an uncertain future. One approach to satisfying these competing objectives is to choose more than one design point for CON. By assembling a portfolio of designs that are investigated in detail, one can ensure a certain level of performance will be achieved while retaining design freedom. Such a solution may seem, at first glance, to be a recipe for increased cost. Modularity provides a framework by which such a cost increase can be minimized.

Chapter 4

Modular Design

As was stated earlier, the ability to 'scale' or adapt the current iteration of a design into the next iteration is a defining characteristic of a successful EA architecture (see Boehm spiral invariants). Modularity provides a framework for building such an architecture. Though rooted in the computer world, recent work on modular design (Baldwin and Clark, 2000; Maier and Rechtin, 2000) allude to broader application. The exact definitions of terms related to modular design can vary from source to source. This discussion will focus on modularity as described by Baldwin and Clarke.

4.1 The structure of design

To understand Baldwin and Clarke's approach to modularity, one must first understand their view of design. In 2000 they published Design Rules in which they attempt to explain modular design and relate it to the evolution of the modern modular computer.

They begin with the notion of an artifact or a physical object assembled/created to accomplish some purpose. A design of an artifact is defined as a complete description of

the artifact. Designs, in turn, are broken down into design parameters. These parameters reflect the individual decisions that must be made to instantiate a design into an artifact. The process of making these decisions is the process of design.

If these decisions could be made independently of one another and if the complete list of such decisions could be enumerated before committing to any choices, the design process would reduce to the task of finding the set of choices that leads to the highest value design. Of course, real design does not operate in this manner. Design choices are often interrelated. Baldwin and Clarke give the example of a coffee mug and its lid. Since the lid must fit snugly onto the mug, the choices of the diameters of the mug and lid are interrelated. If the designer changes one, the other may need to change as well.

There is a hierarchy of design parameters: making one choice creates an additional set of choices. Returning to the coffee mug example, if the designer chooses to include a handle, then he must choose values for the design parameters associated with the handle. These handle-related parameters were not included in the list of design parameters prior to her choosing the 'is there a handle' design parameter, i.e. choosing to include a handle. In this simple case, the designer could have foreseen the need to design the handle prior to deciding to include it. However, one could imagine more complex designs in which such consequences of design choices (especially when a sequence of choices leads to a new level of the design hierarchy) would not be so apparent.

Recalling that the intent of an artifact is to accomplish some purpose, the designer can use that desired purpose as a metric for evaluating the impact of a design choice. Does option A allow greater fulfillment of the desired purpose than option B? Quantitatively, this is exactly the notion of utility defined in the discussion of MATE-CON. Furthermore, the design variables in MATE-CON terminology map to the design parameters in Baldwin and Clarke's terminology. Such a performance-based value metric is not the only way to value a design option. One can use the market price[1] as an arbiter of value. Market price

[1] In fact, basic economics dictates that market price directly reflects the utility of the product to the pur-

is the valuation approach used in Design Rules.

Given a value metric, it seems that the individual design decisions can be made rationally by evaluating their impact on the value metric, and then making the value maximizing choice[2]. Even if such an evaluation method exists, the interdependence and hierarchy of design parameters would lead to a possibly endless cycle of design iteration. Furthermore, the designer can only estimate the value of design. In order to better understand the impact of the interrelationships between design parameters, Baldwin and Clarke use a Design Structure Matrix (DSM) to map the connection between design parameters.

4.2 DSMs, TSMs and the fundamental isomorphism

A DSM is a square matrix with binary entries where each row/column corresponds to a design parameter. A 'one' (often denoted as an 'x') in column j, row i indicates that a choice made for design parameter j in some way restricts choices on i. One can use a DSM to represent the relationship between mug and lid diameters.

	L	LD
Lid/No Lid = L	•	
Lid Diameter = LD	X	•

(a) Hierarchy

	MD	LD
Mug Diameter = MD	•	X
Lid Diameter = LD	X	•

(b) Interaction

FIGURE 4-1: Two simple DSM (from Baldwin and Clarke)

DSM (a) represents the relationship between the 'Lid/No Lid' design parameter (L) and the 'Lid Diameter' parameter (LD). Clearly, the designer's choice to have a lid affects the choice of lid diameter. The converse is not true. The designer can choose whether or not to have a lid regardless of its diameter. Therefore, the parameter LD is dependent upon L while L is independent of LD.

chaser.

[2]During the design of a product, the value of the market price used is an estimate with associated error.

DSM (b) demonstrates another possible relationship between design parameters: interdependence. Fixing either the diameter of the mug or the lid will restrict the possible values for the un-fixed parameter. This type of relationship between two parameters can lead to iteration in design. Consider two different engineers, Martha and Larry, are responsible for choosing MD and LD respectively. Martha chooses the mug diameter to be 2 inches and communicates this to Larry. Since the lid must fit upon the mug, Larry is now restricted in his choice of LD to LD > 2 in. Larry then chooses LD to be 3 inches. Martha realizing that she has some room to increase the diameter (thereby allowing more coffee to be held) makes MD 2.5 inches. To ensure a better fit, Larry then reduces LD to 2.51 inches. This new LD prevents Martha from increasing MD any further, hence stopping the iteration. The design then converges.

Several key lessons come out of this simple example. First, each designer made choices to increase performance (volume in Martha's case and fit of the lid in Larry's case) within their respective domains. Second, the solution converged to (MD = 2.5 and LD = 2.51) required compromises by both designers. Third, the final solution is in no way optimal, but is rather a side-effect of the particular sequence of choices made by the designers. One immediately realizes that the problems cited above are caused by a lack of 'good system level-engineering'. Each designer was focused on his or her part of the design and not on the system as whole.

DSM (a) imposed an order in which the design parameters needed to be chosen. DSM (b) demonstrated interdependence which led to the problems discussed in the preceding paragraph. The third possible relationship between two design variables is independence. This is the preferred relationship from a design point of view since it afford the greatest flexibility to the designer.

More complicated DSMs, such as the one in Figure 4-2, exhibit behavior similar to the mug example. There are interdependencies and possibilities for iteration. As emphasized in the figure, most complex, engineered artifacts tend not to have fully populated DSMs.

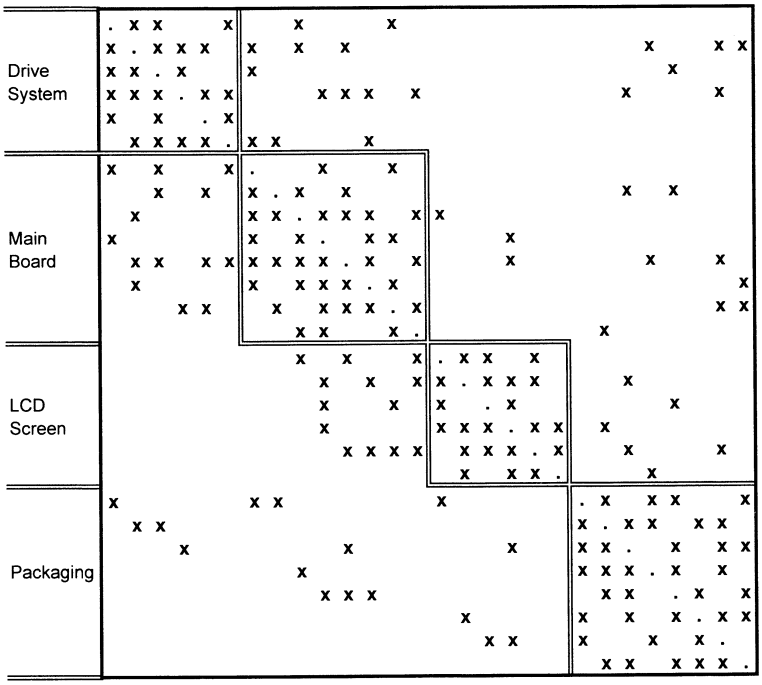


FIGURE 4-2: DSM for a Laptop Computer (from Eppinger-McCord as reproduced in Baldwin and Clarke)

Such a structure would be very difficult to manage and build. Rather most systems consist of several blocks (Main Board, LCD, etc.) of tight integration and a few connections between those blocks. This fact reflects the organizations that design these artifacts. They tend to have a hierarchical organizational structure, grouping designers by sub-system or function.

More generally, Baldwin and Clarke argue that there exists a 'fundamental isomorphism'[3] between a design structure (DSM) and the tasks needed to create and instantiate the design. They record these tasks in a matrix called a task structure matrix or TSM. TSMs work in a similar manner to DSMs except that the rows/columns become steps, or tasks needed to complete the design and produce the artifact. An 'x' in a TSM indicates a dependency relationship between two tasks (i.e. an 'x' in column *j* row *i* indicates that

[3]Baldwin and Clarke do not formally show that an isomorphism (a two-way map that is one-to-one and onto) exists. The term 'fundamental isomorphism' will therefore be kept in quotation marks. They do provide intuition for its existence, which is discussed in the main text.

task *i* depends on task *j* and hence *j* must to be completed before *i*). If a situation like Figure 4-1(b) occurs, then iteration may be needed since the tasks are interdependent. One can get a sense of why the 'fundamental isomorphism' exists by thinking of design parameters in the DSM as a list of decisions that must be made for the design to be instantiated. Since each of these choices constitutes a task, one can immediately see the isomorphic relationship between the TSM and DSM of an artifact – for every design parameter a 'choice' task must be completed. For the purposes of this thesis, the existence of the 'fundamental isomorphism' is not particularly important. The discussion of the 'fundamental isomorphism' is included here, however, since in some contexts it is more natural to think of design in terms of tasks rather than design parameters.

4.3 Modularity

DSMs (and TSMs) provide a descriptive framework for analyzing designs. They also elucidate problems related to iteration in the design process. In particular, for complex designs the degree of freedom available to the designer after several iterations becomes quite small. Once convergence occurs, the designer is so constrained by the interdependencies of prior choices with the choices made by other designers that he or she can only make small changes to the design. This paralyzing complexity leads to an inability of designers to adapt to a changing environment. They cannot capture emerging markets nor take advantage of new technology. Baldwin and Clarke argue that modularity offers one way of avoiding paralyzing complexity. In their words:

> "The driving force behind a quest for modularity is always a desire to achieve the right balance between fruitful uncertainty and paralyzing complexity. The architects of a modular design want to admit enough uncertainty and interdependence into the design process to allow new things to happen, but do

> not want to admit so much that the process cycles endlessly without converging, or settles to a frozen state."

4.3.1 A definition of modularity

Baldwin and Clarke divide the definition of modularity into two concepts: (1) the module and (2) the notions of abstraction, information hiding and interface. They define a module as:

> "A module is a unit whose structural elements are powerfully connected among themselves and weakly connected to elements in other units."

The larger system connecting its constituent modules must provide a framework, or architecture, that allows the modules to retain their independence yet integrate to deliver the system level value. Abstraction, information hiding, and interface relate to how such a modular architecture manages the modules within it:

> "A complex system can be managed by dividing it up into smaller pieces and looking at each one separately. When the complexity of one of the elements crosses a certain threshold, that complexity can be isolated by defining a separate abstraction that has a simple interface. The abstraction hides the complexity of the element; the interface indicates how the element interacts with the larger system."

In terms of a DSM, a modular design would exhibit a nearly block diagonal structure with the blocks representing the tight interconnections within a module. Baldwin and Clarke suggest that a process of defining a series of 'design rules' allows the system architect (a sort of meta-designer who designs the system architecture encoded in the DSM) that

eliminate coupling between proto-modules[4]. The coffee mug example can illustrate this idea.

Say the architect wished to create two modules, a mug module for the mug itself and a lid module for the lid. As we discovered earlier, these two proto-modules would be coupled by the requirement that the lid fit snugly onto the mug. This relationship between lid and mug diameter led to iteration and eventual design paralysis. If, for example, the architect proscribed the mug diameter[5] prior to the design of the mug and lid, the iteration and paralysis would be eliminated. The two proto-modules would be decoupled and become true modules. The specified dimension is a 'design rule' that abstracts the information about the mug relevant to the lid away from the mug module into the design rule thus eliminating the need for the mug and lid designers to iterate over choice of diameter.

			D	M	H	WW	WS	HS	LM	LS
Design Rule	Diameter	D	•							
Vessel	Material	M	x	•		x	x			
	Height	H	x		•			x		
	Width of Walls	WW	x		x	•		x		
	Type of Walls	WS	x			x	•	x		
	Handle Shape	HS		x	x			•		
Lid	Lid Material	LM	x						•	x
	Lid Style	LS							x	•

FIGURE 4-3: Notional modular DSM for the coffee mug

The result of this action has benefits as well as costs. The design process will be more efficient since an iteration loop has been eliminated. The cost, however, is that designers loose the ability to explore some of the design space. Whereas before any mug diameter could be used, the diameter is now restricted to the value specified in the design rule. The net value of the decision to impose the design rule depends on the degree to which the architect has ascertained that no high value regions exist in the portion of the design space

[4]Structures that are nearly modules, but still have a few tight connections to elements outside of themselves.

[5]Iteration could also be eliminated by first specifying that the lid be 0.001 larger in diameter than the mug and then insist that the mug be designed before the lid. Such an approach eliminates interaction but still retains dependency (as used in Figure 4-1) and does not allow independent design of the mug and lid.

being excluded by the design rule. The architect's ability to make this judgment is dependent upon the following criteria: (1) there knowledge of high value regions in the known design space and (2) the possibility that high value region, currently unknown, may be become apparent (e.g. through a change in the market). The first criterion is satisfied by extensive exploration of the design space such as that advocated in the MATE process; the second by ensuring that the design rules do not too severely the design space.

Both an integrated and modular architectures impose constraints on the portion of the design space that can be explored. In the integrated case however, the constraints often emerge through the design process as couplings are revealed, while in the modular case the design rules state some of the constraints explicitly. The advantage of explicit constraints is that one can estimate the cost associated with imposing them and then rationally determine whether or not they are worthwhile.

Looking at the modular architecture from a task perspective, one can partition the design process into three stages:

1. Formulate design rules
2. Work on hidden modules[6]
3. Integrate and Test

Since the discussion to this point has focused on the DSM representation of an artifact, testing and integration, which are tasks, have not explicitly discussed. The easiest way to understand the relationship between these three stages is to study Figure 4-4.

Figure 4-4 is a modular DSM for a hypothetical laptop computer. The main system components have had their interdependencies abstracted into a set of design rules creating the characteristic block diagonal structure of a modular architecture. The first stage formulates the design rules, ensuring that the module development in the second stage can

[6]Those modules that do not interact directly; rather they interface through design rules.

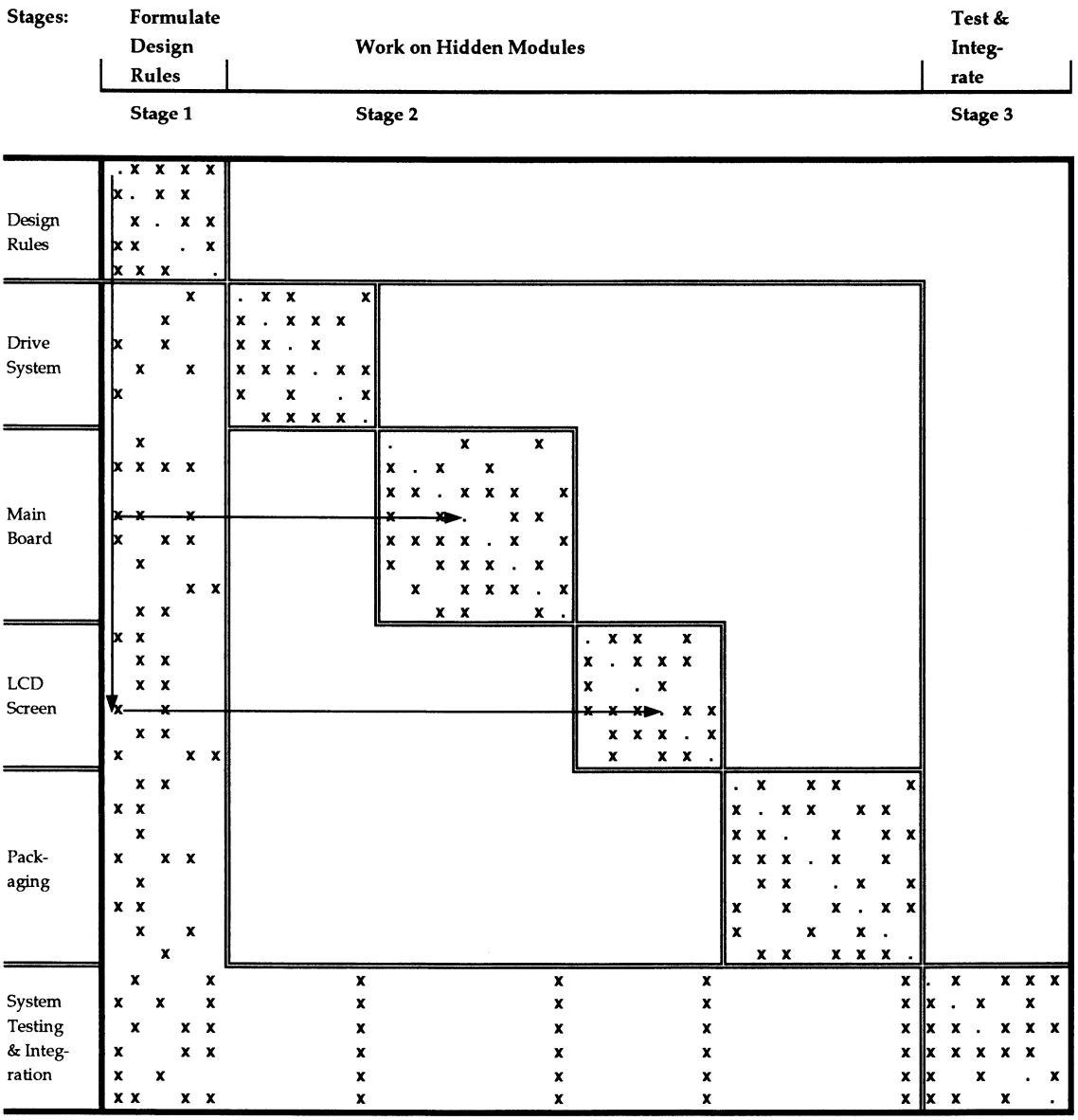


FIGURE 4-4: A modular DSM for a hypothetical laptop computer (from Baldwin and Clarke). The indicated design rule decouples an interaction between the motherboard and LCD

be carried out in parallel. The final stage, system integration and test, brings the modules together into a final product. Note that integration and test does not feed back to the previous stages. Since the modules are decoupled, all sub-system or module level issues are confined to the module. Any problem discovered during testing should be at the system level and therefore should not require further work internal to the modules.

Another way of looking at a modular design is through a design hierarchy. This view focuses on the notions of information hiding and interface mentioned in the definition of modularity.

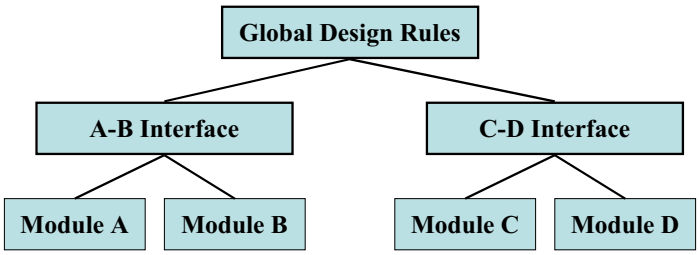


FIGURE 4-5: Design Hierarchy Example

In a design hierarchy such as the one depicted in Figure 4-5, each layer contains information visible to layers below it. A piece of information is visible if layers below it may use it. Changing visible information may require changes in modules further down the hierarchy. A design rule is an example of visible information since modifying it causes changes to those modules below it in the hierarchy. A link between layers implies design rules are being imposed by the higher layer onto the lower layer (e.g. the A-B interface imposes constraints via design rules on Module A and Module B). The modules at the lowest level have nothing below them, and therefore contain no visible information. These modules are called hidden modules.

In summary, Baldwin and Clarke identify three effects of modularity:

1. Modularity increases the range of "manageable" complexity. It does this by limiting the scope of interaction between elements or tasks, thereby reducing the amount and range of cycling (iteration) that occurs in a design or production process.

2. Modularity allows different parts of a large design to be worked on concurrently.

3. Modularity accommodates uncertainty.

The last effect is the crucial point in understanding why the modular architecture work well when using a spiral development process. A modular design partitions the design parameters into two groups, hidden and visible, the larger group being hidden. The hidden parameters are those contained within the bottom layer of the design hierarchy. The visible parameters are codified in the design rules. Since, by definition, the designer can make changes to the hidden parameters without affecting the rest of the system, hidden modules can be changed to take advantage of new opportunities of increasing value or to mitigate the effects of decreasing budgets without the need to redesign the whole system. In fact, any choice for a hidden that satisfies the design rules is acceptable.

Recall from Chapter 2 that one of Boehm's assumptions for successful EA is that the system architecture is scalable, i.e. the system delivered at IOC could be built upon in future increments. A modular architecture provides this capability. The designer can use lessons learned from the first increment to change any of the hidden modules and deliver higher value in the second. Also, the design rules are similar to the life cycle architecture in Boehm's terminology.

The capability to swap one module for another can be viewed as creating a portfolio of options on the set of possible modules that satisfy the design rules. The designer has the ability (but not the obligation) to use any of the modules available to him. Instantiating a particular design reduces to exercising a sub-set of these options. When viewed from this options perspective, the recently developed valuation techniques of real-options can be used to value the development of particular modules.

Mixing and matching hidden modules is not the only operation available to the designer. Baldwin and Clarke define six modular operators that specify the possible ways a designers can improve a modular design.

4.4 The Modular Operators

Given that modularity buys the designer a degree of freedom, what exactly can the designer do? Six modular operators defined by Baldwin and Clarke specify the possible operations a designer can apply to a modular system:

1. **Splitting** a module into two or more sub-modules
2. **Substituting** one module design for another
3. **Augmenting** the system by adding a new module
4. **Excluding** a module from the system
5. **Inverting** to create new design rules with existing modules
6. **Porting** a module from another system

Baldwin and Clarke develop criteria for determining value of each of these operators. In this thesis however only splitting, substituting, augmenting and excluding will be discussed since they are referred to in the case study in Chapter 5.

Splitting involves taking an interconnected DSM and specifying design rules so as to remove connections in order to break up the interconnected structure into two or more independent modules. It terms of a design hierarchy:

Note that formally the splitting operator is the generator of modular structure. If one views an interconnected design as one large module, then the process of breaking the

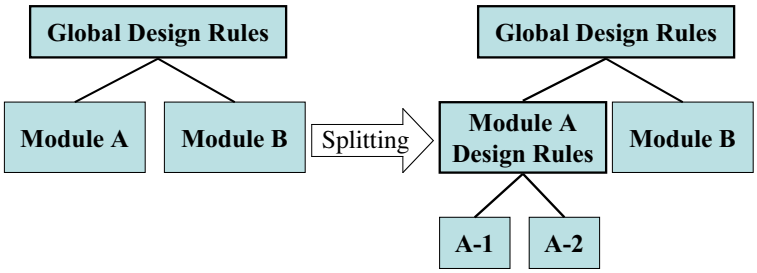


FIGURE 4-6: The Splitting Operator

interconnected design into smaller modules and establishing design rules along the way is really repeated applications of the splitting operator.

Baldwin and Clarke define splitting as a four-step process. First, the designer must "accumulate and organize" knowledge about specific parameter interdependencies, i.e. the form of the DSM to be split. Second, they must "perceive" that the time is right, i.e. that splitting now in the way desired will not exclude high value portions of the design space. Third, the designer must formulate design rules to break the connection between the new modules and specify the new architecture, interfaces and tests. Fourth, these new design rules must be enforced in future design work.

The advantages of splitting are the same as those of modularity in general. Before the split, the designer was forced to choose among design for module A as a whole, after splitting she can choose to change sub-module A-1 while retaining A-2. As long as the sub-set of possible designs for module A that can be represented using the A-1, A-2 decomposition contains high value solutions, there is increased flexibility in design without loss of performance.

Substitution is much simpler operator than splitting. It involves replacing one module's design with another, better design. Once again, any design that satisfies the design rules is a candidate for being substituted in.

Augmenting involves adding a hidden module to an existing system. The key here is that the modules are hidden and therefore do not impact the global design rules. A simple example of this might be adding a printer module to a computer. The printer obeys the global design rule of talking to the system bus and delivers the additional value of being able to make a hard copy. Exclusion is the opposite of augmenting. Modules are now removed from the hierarchy. Though the value that they generated is lost, the cost incurred in building, integrating and testing the excluded module is avoided. Excluding is commonly applied in situations involving product families.

Augmenting and excluding often work in tandem. When faced with limited budgets, designers will exclude some modules from an expensive design to bring it within budget and then later, when more resources become available, they will augment the excluded modules back into the hierarchy.

The preceding sections have defined modular architecture and described the structures and processes associated with using one. The decoupling of modules make such an architecture a natural choice when using EA/SD. In practice however, constructing aerospace products using a modular architecture has can be difficult. Whitney (2002) offers a possible explanation for this difficulty.

4.5 Modular Design for Aerospace Systems

Whitney defines two classes of systems: (1) those that are primarily signal processors[7] (referred to as 'signal processors' or 'low power systems') and (2) those that process and transmit significant power[8] (referred to as 'power processors' or 'high power systems'). The terms 'high' and 'low' power are used in the following way. Low power means that the power needed to accomplish the function of the system in much smaller than that used.

[7] Examples include microprocessors, computers, gas meters, typewriters and sewing machines.
[8] Example include laser printers, automobiles and aircraft.

High power means that the power needed is on par with the power used. The principal difference between these two classes is that signal processors use energy within them only as concrete representation of information, i.e. the signal. As such, the choice of representation can be tailored to allow other desirable effects such as the decoupling needed for modular design. For example, in digital circuits, the information being represented are bits, i.e. ones and zeros. Typically a nominal five volt signal is used to represent a one and a nominal zero volt signal for a zero. By having such a large spread between the two energy states that represent one and zero, the designer need not have exactly five volts to represent a one – anything between 4.5 and 5 volts works just as well. Such freedom typically does not occur with power processors. If a five volt source is being used to drive a motor, then a reduction to 4.5 volts will result in 10% less power delivered to the motor and consequent degradation of system performance. To understand the design differences between signal and power processors, Whitney compares the design of integrated circuits (signal processors) using the VLSI[9] design methodology to the design of certain complex-electro-mechanical-optical (CEMO) systems (power processors) such as those examples given above.

In a similar manner to modular design described above, VLSI proceeds in three stages. The process is summarized in Figure 4-7. The first stage, component design, involves developing a library of elementary devices such as logic gates that are the building blocks of any circuit. Each of these devices can be viewed as a physical instantiation of a mathematical operation. A set of design rules are established to ensure that these elementary components are decoupled in terms of design parameters. These design rules include specifications such as bus voltage, line width and size constraints. With these rules in place, the components can be connected to one another in an arbitrary manner without the need to worry about implementation difficulties[10]. Manufacturing processes are also

[9] Very-Large-Scale-Integration

[10] Of course, there will always be *some* implementation challenges. In practice, however, much less time is spent on these integration issues than on developing the library in the first place.

developed for each item in the device library. Constructing a versatile device library is a very difficult task that often takes years to complete.

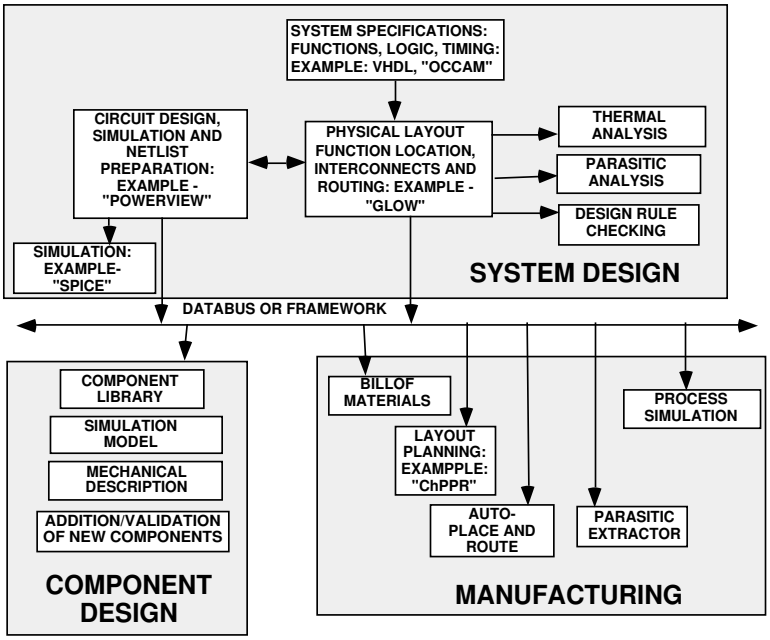


FIGURE 4-7: The VLSI design process (from Whitney (2002))

With the device library completed, the second stage, system design, involves choosing the correct arrangement of components from the library to accomplish the desired purpose. Since each elementary device has been reduced to mathematical object, system design can proceed in a top-down manner, developing logic diagrams whose operations correspond directly to the components in the library. This correspondence is so strong that often the process of going from mathematical representation to circuit layout can be entirely automated. Simulation of processes such as heat transfer ensure that side-effects during implementation do not prevent proper operation. In practice, design rules such as the ‘5 volt = 1’ specification described earlier cause the system to be over-designed to a point that such side-effect are minimized. Once the full system is designed, it is again simulated to correct any remaining timing or logic errors. Taking advantage of the independence gained from the modular architecture, system level simulations can be constructed using

the component level models found in the device library. In the final stage, the circuit is manufactured.

Compare the above description of VLSI design to the design of CEMO systems. In VLSI design, there is a clean progression from one stage to the next and near certainty that issues relevant to an earlier stage will not appear in a later stage. CEMO design, on the other hand, is characterized by feedback loops between different communities (e.g. customers, designers, manufacturers, etc.). In CEMO design, there is no analogue to the device library central to VLSI design. Rather, sub-systems must be developed for each new design effort. Ideas, concepts and structures may be re-used from previous design, but extensive work must be done to ensure their appropriateness within the new context. The few parts which can be found in a standard library are minor items such as fasteners, fittings and finishes. Whitney describes the CEMO process as consisting of the following tasks:

1. Specifying a set requirements that define the major functions of the product
2. Defining sub-systems that will carry out these functions
3. Allocating space to the sub-systems within the allowed space
4. Decomposing the sub-systems into individual parts
5. Determining allowable variation (e.g. tolerances) both in parts and system parameters
6. Predicting when the variation may be larger than allowed and then mitigating any off-nominal behavior
7. Developing manufacturing methods with their associated costs
8. Verifying each step taken in the process (e.g. through simulation and prototypes)

9. Revisiting earlier decision when knowledge gained later warrants revision

At first glance, these steps may seem similar to the process described for VLSI design. The key difference though is that they must repeated in at each stage of design for each part. For example, the requirements flow down process is a repetition of step 1. Step 2 must also be repeated with each sub-system until detailed engineering specification are arrived at. The reason for this difference is that in VLSI the main functions of the product are carried out by assembling pieces from a *general purpose* library. This same library can be used to accomplish many different function. In contrast, in CEMO systems each sub-system and consequently part must be developed anew for each design effort. Those few parts that can be reused, e.g. fasteners, tend to be over-designed given their application. For example, a fuselage skin is attached using far more rivets than theoretically necessary. Since the mathematics only partially (often quite poorly) approximates the behavior of parts, testing must be redone with each modification of a part. Even if the parts behave as designed when tested in isolation, their behavior in the context of other parts in an assembly may be different and difficult to predict. This is especially true of off-nominal behavior. Thus integration and test becomes a very costly portion of the program cost.

Typically, high performance power processors, in particular those found in aerospace, tend to be subject to severe volume, weight and energy constraints. As a result, one part often needs to perform multiple functions to gain volume/weight efficiency. Multiple function per part (coupled with multiple parts per function) leads to strong coupling between parts that makes modular decomposition via splitting difficult if not impossible.

Given the above distinction between signal processors which can often be modularized and power processors which cannot, designers who wish to take advantage of the benefits of the modular approach need to first determine the information interfaces within their system and then attempt to form design rules to split into modules components connected through those interfaces. Often this will be impossible since those same components will

share power interfaces as well. There are a few cases where such decoupling can be accomplished. The next section provides two examples from different parts of the space industry.

4.6 Examples of modular spacecraft

In this section, two examples of modular spacecraft are discussed. They represent different segments of the spacecraft industry, yet both demonstrate the advantages and challenges of using a modular architecture. One, the TRW Flexible Space Vehicle Production Line (FSVPL), produces, for example, large communication satellites. This is a well-developed market with mature products. The other example, the Stellenbosch UNiversity SATellite (SUNSAT) is at the other extreme of the space business. It is a small science satellite developed and constructed by academic and research institutions.

4.6.1 TRW Flexible Space Vehicle Production Line

TRW, in association with Air Force Mantech, realized that space systems face many of the problems that Whitney described for CEMO systems. Spacecraft tended to be custom designed for a single purpose around a given payload. As a result, the structural, electrical and thermal properties of each spacecraft was unique and lessons could not easily be applied from one craft to the next. Integration and test were very costly and time consuming. In developing the FSVPL, TRW set ambitious goals of a 50% reduction in cost and a 75% reduction in cycle time. The result was a modular spacecraft architecture that reduce costs through standardization. The modular architecture also provided a well defined structure for the manufacture of the spacecraft. Such a well defined structure enabled the creation of lean production line resulting in impressive cycle time reduction and cost savings. The architecture for the spacecraft bus is shown in Figure 4-8.

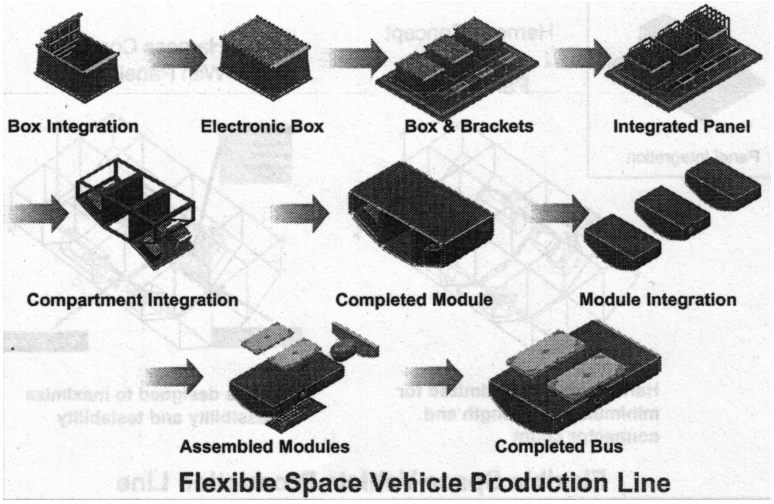


FIGURE 4-8: TRW Modular Spacecraft Bus Architecture for the FSVPL (Emery and Punatar, 2002)

Figure 4-8 clearly demonstrates the difference between signal processors (e.g. sensing and data processing systems) and power processors (e.g. propulsion and solar arrays). Those components that are signal processors form a well-defined modular hierarchy with higher level assembly imposing design rules on lower level assemblies. The hierarchy begin at the upper-left corner with individual boards. These board are assembled into boxes that impose size and interface constraints on the boards. The boards combine to form panel that are loaded into a frame to form a module. Modules are then connected forming the bus. The precise mix and number of boards, boxes, panels and modules can be tailored to specific mission needs. Since interfaces are standardized through design rules, boards, boxes, panels, even entire modules developed for one spacecraft can be used in another. Compare this to the power processors. The propulsion module, for example, carries more power that any other part of the spacecraft. Unlike the electronics, it exists as a single, monolithic module attached to the rear of the spacecraft in the last step depicted in Figure 4-8.

The modular structure developed by TRW also demonstrates the evolutionary potential of modularity. For example, consider a spacecraft constellation using a staged deployment

strategy to keep up with rapidly improving sensor and processing technology. Future generation could re-use much of the design from prior generation, replacing only those components that warrant an upgrade. Of course, for such an upgrade to be successful, the existing design rules would need to be followed. This creates an issue of legacy that is inherent in most evolutionary effort. As Boehm points out, the architecture must be flexible enough to keep up with the evolutionary pace. Thus, if the new processors cannot use the old interfaces, the benefits modularity and EA may be lost[11].

4.6.2 Stellenbosch UNiversity SATellite (SUNSAT)

Whereas the FSVPL reflects the 'big' side of the space industry, SUNSAT comes from the 'small', micro-sat world. SUNSAT was South Africa's first academic/research spacecraft. It carried a large variety of instruments ranging from an experimental NASA earth imager to temperature sensors designed by high school students. As such, having a spacecraft that could be flexible enough to accommodate so many different customers, yet still keep integration cost low became a key concern for the spacecraft designers. They chose a modular architecture depicted in Figure 4-9.

The tray-based system allowed rapid integration from the spacecraft components. Design rules were established to ensure easy integration of student designed experiments in the experiment tray. These included mass, volume, power, and environmental limits. The tray structure and design rules allowed easy routing of power, data and other sub-system flows throughout the spacecraft. The trays also demonstrate another important feature of modularity relevant to EA/SD – the ability to bring together diverse stakeholders (various student teams) with diverse needs (various experiments). All of them had to agree to abide by the design rules to make the spacecraft successful. Thus great care was taken to choose rules that allowed flexibility without sacrificing implementation ease. The fact that the

[11] When this is the case, one may be able to use the porting operator, though possibly at significant cost – see Baldwin and Clark (2000) for details.

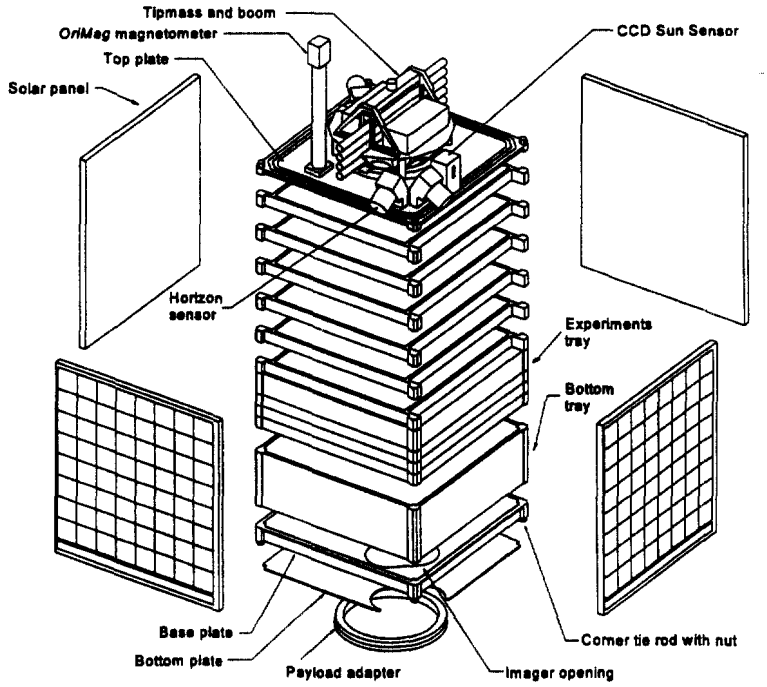


FIGURE 4-9: SUNSAT 'Tray' based modular architecture from Mostert and Koekemoer (1997)

four student experiments were both designed and manufactured at disparate locations demonstrates the success of the SUNSAT team's approach.

4.7 Modularity as an enabler for EA/SD

As was alluded to in the examples, modularity can enable an evolutionary acquisition strategy. Fundamental to an EA/SD strategy is a system architecture that is scalable to the changing needs and capability environment. This ability to scale should be aligned with the resolution of principle risks to which the system is exposed. A modular architecture has these properties.

The creation of a modular hierarchy can be interpreted as creating a set of options for the designer. The modular operators provide define the set of possible changes that can be made to the hierarchy. The term option here is used in a similar sense to financial option, i.e. it is the rite, but not the obligation, to make some change to the design at a later date. For example, the designer can choose to arrange the first two rings his satellite constellation so that two more may be added later if demand materializes. To create this option however, the designer cannot arbitrarily place the first two rings. He has two options: (1) include additional fuel on the satellites to allow for orbit changes after launch, or (2) arrange the first two rings so that they do not need to be changed when adding more rings. These options can be operationalized as design rules on the configuration of either orbits or spacecraft within the constellation design. In both cases, imposing the design rules limits that space of possible designs. For example, only a sub-set of possible orbits will allow additional rings to be added without reconfiguration of existing rings.

With the portfolio of options[12] purchased (i.e. the constellation configured as per option

[12]Formally an option is right but not the obligation to do some particular act in the future. Since, for example, deploying a two-ring architecture in 2002 allows transition to several different 2005 architectures (including leaving the 2002 architecture unchanged), the term portfolio of options is more appropriate.

(2)), the designer can choose at a later date whether or not to add rings. By allowing this choice to be made after uncertainty has resolved, the modular architecture allows risk to be the key driver of design evolution.

The previous two chapters have introduced two powerful tools, MATE-CON and modular design, for implementing evolutionary acquisition and spiral development. None of the examples given thus far have attempted to unify the two in single design effort. To date, no system has flown that uses both MATE-CON and modular design, however, the author along with several graduate students at MIT conducted a conceptual design of a Space Based Radar using the MATE-CON approach. The staged deployment of the constellation was cast as modular architecture. The next chapter discusses this case study.

Chapter 5

Space Based Radar: A case study in modular constellation design

The previous chapters introduced two tools for implementing EA/SD. The MATE-CON process provides a means for capturing user needs as well as ensuring stakeholder participation throughout the development process. The feedback both during the MATE and CON phases provide ample opportunity for stakeholders to gather additional information about the design problem as well as learn from prior efforts. For EA/SD to be useful however, one more ingredient is needed – a system architecture that allows the new information and learning to be incorporated into the design without having to redesign the entire system.[1]

When a system is evaluated while in service, there are three possible outcomes. First, the system meets all current and foreseeable future needs and should be left as is, second, the system meets none of the current needs and should be scrapped, and third, the system meets some current needs and can be improved. When an EA/SD process is used the third possibility is the most likely scenario. Since designers will be building upon an existing

[1] This case study was completed jointly by the author and Christopher Roberts in 2003.

system, a modular architecture is a good choice. By reducing the system to several independent (or nearly independent units), the modular architecture allows the designer to change/add modules to the system without redesigning the entire system. This ability to make changes is not limitless. The designer must obey existing design rules to ensure that module independence is maintained.

One such system that uses EA and exhibits the third scenario is the Space Based Radar (SBR). The high cost and high technology risk inherent to SBR drove the Air Force to pursue and EA strategy. As is described in greater detail below, initial capability will be built upon as technology and funds become available, hence a modular architecture is useful. In addition, SBR has a wide variety of uses and thus has several different attributes that quantify its value. MAUT within the MATE-CON can be used to determine value SBR designs.

5.1 An introduction to Space Based Radar

The ability to gain broad perspective over the battle-space as well as track the movements of enemy forces is key to successful warfare. For years, aviation has provided this capability (e.g. observation balloons and reconnaissance flights). In modern times as the scope of the battle-space has expanded and the operational tempo has increased, the need for timely, actionable information has become even more acute. One system designed to meet this need is the JSTARS (Joint Surveillance and Target Attack Radar System). One component of JSTARS is a modified Boeing 707[2] aircraft that carries a radar. First fielded in desert storm, the radar system provides commanders the ability to track moving targets in real-time. Being an airborne system, however, JSTARS suffers from several problems

[2]The JSTARS aircraft is another example of modular design. The aircraft consists of several racks and sensor ports on to which different mixes of sensors can be mounted depending upon the given mission needs. This modular approach allowed JSTARS technology to be developed and initially fielded on a 707 and then easily transitioned to a larger aircraft.

including airspace restrictions, limited loiter time and deployment delays to emerging areas of interest, since the aircraft needs to fly to the region of interest.

One possible solution to these issues is to move the radar asset into orbit forming a Space Based Radar. This solution has been proposed on various occasions over the pass few decades. Most recently, a study was conducted at Lincoln Laboratory during the summer of 2002 (Spaulding, 2003). The Lincoln Lab study tried to improve upon previous SBR design studies by more explicitly incorporating the tactical user's perspective into the design effort. To do this, they divided study participants into two teams. One team, the 'sellers', was composed on technical experts who focused on exploring the SBR tradespace and identifying promising system architecture and operational concepts. The remaining participants formed the 'buyers' team. They sought to understand the tactical users' needs and examined concepts proposed by the 'sellers'. Feedback between the two teams was facilitated by a series of scheduled meetings. When considered from an MAUT point of view, this approach is somewhat lacking. Since the actual users (Combatant Commanders[3]) could not participate in the effort on an on-going basis as the teaming arrangement would have required, the 'buyers' were forced to be proxies for the actual users. If a process such as MATE were used, the users preferences could have been captured directly using the interview techniques described in Chapter 3, one layer of interpretation would be removed. Now of course, MATE does not recommend interviewing and then not speaking to the user again. Rather feedback is expected from the user on a regular basis. However, since the MATE process highlights the particular details of the design relevant to the decision maker, much less time commitment would be required per feedback session than when the teaming approach is used.

Second Lt. Timothy Spaulding designed and implemented a MATE[4] model of space based radar in an attempt to compare the results of the Lincoln Laboratory study to results gained using the MATE approach (Spaulding, 2003). Unfortunately, classification of the

[3]These are among the highest level operational commanders in the US military.

[4]Spaulding did not complete the CON portion of MATE-CON. Thus, only MATE will be discussed here.

Lincoln Lab study results prevented a direct comparison. Nevertheless, Spaulding's study did provide a useful model for studying the SBR tradespace and hence was used in the current research.

One of the key roadblocks to successful development of the SBR system is the radar technology itself. To date no similar radar system has been operated from orbit. The most recent attempt to do so, Discoverer II, never progressed beyond concept development and was canceled by congress in 2000 (Goodman, 2002; Caceres, 2003, as cited in (Roberts, 2003)). Given the high degree of technology risk associated with SBR, the Air Force believes that the EA/SD paradigm is appropriate.

Beginning with technology demonstration similar to Discoverer II and the eventually building up to full global satellite system, the Air Force aims to have a SBR that meets all specified needs by 2010. Two possible evolution strategies that can accomplish this goal are (1) launch a series of successively larger and more complex constellations as the technology develops and (2) build upon previously launched constellations replacing spacecraft only at end-of-life. Given the high cost of the SBR spacecraft, the second option is much more attractive. Since future systems are built up from their predecessors, a modular architecture would provide a way manage the growth of the constellation. The remainder of the chapter explores such an architecture in terms of value delivered and cost incurred as functions of time.

5.2 Spaulding's SBR MATE model

Spaulding's MATE model of SBR consists of the three components found in a MATE model: (1) a set attributes and associated utility functions, (2) a set of design variables the parameterize the space of design choices available, and (3) a model to map specific design choices to their expected utility and cost.

5.2.1 SBR Attributes

The attributes (metrics of how well needs are met) of the SBR will be described before the design variables. This reflect the fact that the attributes are solution independent and should be viewed without regard to any particular architecture that may be reflected in the choice of design variables.

The first step in choosing attribute is to identify the relevant decision makers. In the case of SBR, the key decision makers are the military leaders who choose to allocate resources to SBR development and operation, and the congress who approves this choice. However, much of the utility derived by these decision makers is directly related to the usefulness of the SBR to its users. Thus, measuring the utility of the SBR as perceived by users is a good surrogate for measuring the utility perceived by the decision makers.

The users of SBR require it to accomplish three different missions: Digital Terrain and Elevation Data (DTED), Synthetic Aperture Radar (SAR) Imaging and Ground Movement Tracking Indication (GMTI). DTED involves gathering detailed elevation data to form maps. DTED data is only gathered during peacetime. SAR provides high resolution (<1m) ground imaging in all weather situations and is used both peacetime and during operations. GMTI allows real-time tracking of moving object and is used during operations. To operationalize these three mission within the MATE framework Spaulding, working with the user community, found a set of quantifiable attributes that described success for each mission.

Working in consultation with a member of the 'buyers' team from the Lincoln Lab summer study, Spaulding developed the following list of user attributes.[5]

For the attributes listed, Spaulding conducted interviews with the user representative to elicit single-attribute utility functions as well as the k_i-values needed for the

[5]The user representative was principally concerned with the SAR and GMTI missions. This is reflected in his choice of attributes.

Attribute	Worst	Best
TRACKING AREA: The number of 10 square mile boxes inside which objects can be reliably tracked	10 boxes	75 boxes
MIN DETECTABLE SPEED: The speed above which an object can be detected.	50 mph	5 mph
SAR RSESOLUTION: The best possible resolution of the SAR images	3 meters	0.5 meters
SAR AREA: The square mile size of SAR images possible (can be split into several smaller images).	1/4 by 1/4 miles	10 by 10 miles
GEOLOCATION ACCURACY: The average size of the error ellipse on a GMTI return.	500 meters	50 meters
GAP TIME: The average time during which the enemy can 'hide', i.e. when there is no coverage.	60 min	5 min
CENTER OF GRAVITY AREA: The number of 100 square mile boxes inside which center of gravity can be readily calculated.	1 box	5 boxes

TABLE 5-1: Attributes in Spaulding's SBR MATE Model

multi-attribute formulation (see Equation 3.1). The 'worst' value listed in Table 5-1 corresponds to the minimal acceptable value to the user (utility $= 0$) and the 'best' value corresponds to the value beyond which there is no additional benefit (utility $= 1$).

These attributes are particular good in terms of solution independence since they don't even imply that the mission is being accomplished from space. The design variables do however suggest a space based solution.

5.2.2 SBR Design Variables

There are five design variables associated with the SBR, three spacecraft related and two constellation related. The three spacecraft related variables are: (1) Scan Angle, (2) Technology Level and (3) Aperture Area. Scan angle refers to the size of the off nadir window that the system can image/track object within. Scan angles of 5×5, 20×15, 30×15 and 45×15 [all in degrees $\times$ degrees] were considered. Technology level was used as surrogate for improved technical performance of the radar technology over time. Instead of attempting to predict performance improvement over time, Spaulding used the technology level design variable to impact the cost incurred to deliver a particular level of

performance. A given level of performance would become cheaper to attain the later it was used. Technology in the years 2002, 2005 and 2010 was considered. Aperture area is the size of imaging aperture as measured in projected area on the ground. Aperture areas of 40, 70 and 100 meters^2 were considered. These spacecraft related design variables reflect the major technical trade-offs that must be made in any radar design.

The two constellation related design variables are constellation pattern and orbit altitude. The constellation pattern variable consisted of 13 different Walker Delta constellation that were deemed promising for this application by Lincoln Laboratory during their study. Each constellation was considered at four different altitudes, 800, 1000, 1200, 1400 km. The 13 Walker Delta constellation are listed in Table 5-2.

Number of Satellites	Number of Planes	Number of Satellites Per Plane	Angle Between Satellites per Plane (360/ # of Sats. Per Plane, Deg)	Offset between Planes (360/# of Planes, Deg)
8	4	2	180.0	90.00
9	3	3	120.0	120.00
10	10	1	360.0	36.00
12	6	2	180.0	60.00
13	13	1	360.0	27.69
15	5	3	120.0	72.00
16	8	2	180.0	45.00
17	17	1	360.0	21.18
18	6	3	120.0	60.00
19	19	1	360.0	18.95
20	5	4	90.0	72.00
21	7	3	120.0	51.43
22	11	2	180.0	32.73
24	8	3	120.0	45.00

TABLE 5-2: Walker constellations used in Spaulding's SBR model

The strength of the effect of each design variable on each attribute is documented in the QFD matrix produced by Spaulding. A larger entry in the matrix represents a stronger effect.

As they are key to staged deployment strategies suggested earlier, orbit altitude and constellation pattern deserve special attention. They are indicative of one of the fundamental trade-offs in SBR constellation design – that of power vs. persistence. Radar performance on tracking tasks will improve the longer the spacecraft loiter over the

CASES		Tracking Area	Min Detectable Speed	SAR resolution	SAR area	Geolocation accuracy	Gap Time	Center of Gravity Area	Cost	
4	Scan Angle	9	0	0	9	3	3	9	0	33
3	Technology Level	0	0	0	0	0	0	0	9	9
3	Aperture Area	9	9	9	9	9	3	9	9	66
4	Orbit Altitude	3	3	3	3	3	9	3	0	27
13	Constellation	9	0	0	9	0	9	9	9	45
1872		30	12	12	30	15	24	30		

TABLE 5-3: Spaulding SBR QFD

desired target. This tends to favor high altitude spacecraft as their earth central angle[6] is larger and orbital velocity is slower. However, radar performance is largely driven by the power reflected back from the target. At higher altitude, more power is needed to generate the same return (i.e. be able to track the same size targets) because the effective power seen at the target decreases as one over the fourth power of the distance between the spacecraft and target. This additional power leads to increased per spacecraft cost. On the other hand, lower altitude constellations, though requiring less power, need many more spacecraft to obtain the same level of coverage and persistence as the higher altitude constellation. More spacecraft means higher total constellation cost. As will be discussed later, the tension between power and persistence can be exploited to create a more flexible SBR architecture.

5.2.3 Limitations of the model

To ensure a just comparison between the MATE model and the results of the Lincoln Lab study, Spaulding made an effort to include as much commonality as possible between his model and Lincoln's. To that end, his spacecraft model interpolated over a table of radar

[6] A measure of the area on the earth visible to a spacecraft.

designs produced during the Lincoln Lab study. This removed biases caused by differing technical assumption about the spacecraft made by the two studies. Similarly, he only considered the 13 walker constellations listed earlier. Both these choices, however, limited the ability to explore evolution strategies for space based radar. Also, the model did not allow for easy introduction of uncertainty. Despite these limitations, interesting evolution paths for SBR were identified.

5.3 Results of Spaulding's SBR MATE model

After evaluating the utility and cost of the 1872 combinations of design variables enumerated above, Spaulding produced the utility vs. cost space plotted in Figure 5-1. In the plot and discussion that follows, costs are always expressed in year 2002 dollars.

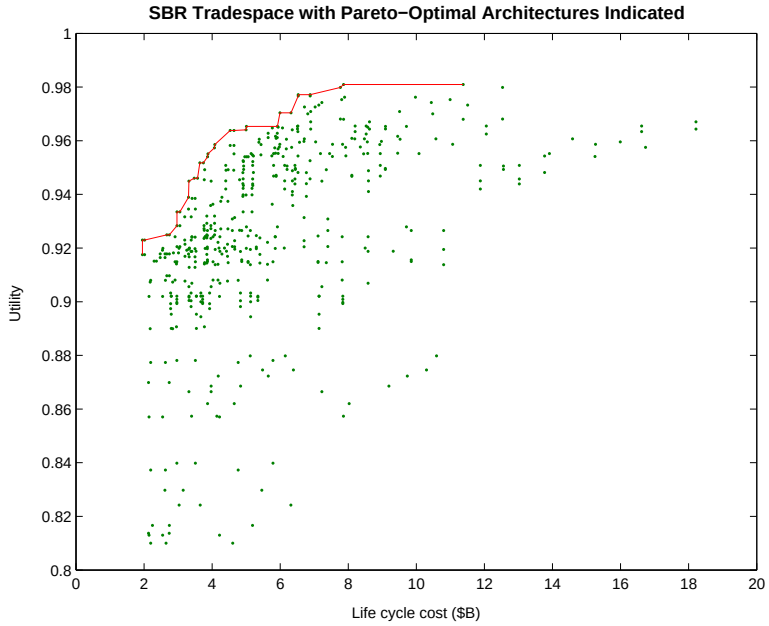


FIGURE 5-1: SBR Cost vs. MAU space produced by Spaulding. Pareto optimal architectures lie along the red curve.

Examining the Pareto optimal architectures reveals several key trends within Spaulding's SBR tradespace. The characteristics of the Pareto optimal architectures are listed in Table 5-4.

Architecture ID Number	Cost ($B)	Utility	Numb. of Satellites	Numb. of Sats per Plane	Altitude (km)	Aperture (sq. m)	Tech Level
1275	2.145	0.812	8	2	1200	40	2010
1262	2.145	0.857	8	2	1000	40	2010
1249	2.145	0.901	8	2	800	40	2010
15	2.328	0.917	9	3	1000	40	2002
28	2.328	0.922	9	3	1200	40	2002
184	2.577	0.922	9	3	1200	40	2005
30	2.742	0.924	12	2	1200	40	2002
186	3.292	0.924	12	2	1200	40	2005
19	3.502	0.938	16	2	1000	40	2002
1254	3.781	0.949	16	2	800	40	2010
10	4.263	0.951	20	4	800	40	2002
1256	4.503	0.955	18	3	800	40	2010
1258	4.600	0.958	20	4	800	40	2010
25	4.64	0.963	22	2	1000	40	2002
1260	5.009	0.965	22	2	800	40	2010
269	6.713	0.972	19	1	800	100	2005
113	7.025	0.972	19	1	800	100	2002
272	7.674	0.976	22	2	800	100	2005
285	7.674	0.977	22	2	1000	100	2005
129	8.024	0.977	22	2	1000	100	2002
1364	9.006	0.980	22	2	800	100	2010

TABLE 5-4: Characteristics of Pareto optimal SBR architectures

The first and most obvious trend is that none of the architectures that use a 70 m^2 aperture area are Pareto optimal. To understand this effect, it is useful to examine the tradespace coded by aperture area as is shown in Figure 5-2.

Below a life cycle cost of about six billion dollars, the reduced cost of the 40 m^2 aperture outweighs the performance improvement gained when using a 70 m^2 aperture. Above six billion dollars, the better performing 100 m^2 aperture dominates the more expensive 70 m^2 aperture designs. This example demonstrates the power of tradespace within the MATE approach. Being able to produce plots such as Figure 5-2 allows the designer to conclude that the 70 m^2 aperture should not be pursued as it is dominated at all cost levels.

The persistence vs. power trade-off discussed earlier is also apparent in Table 5-4. As cost increases, there is a trend from high altitude constellations with fewer spacecraft to lower

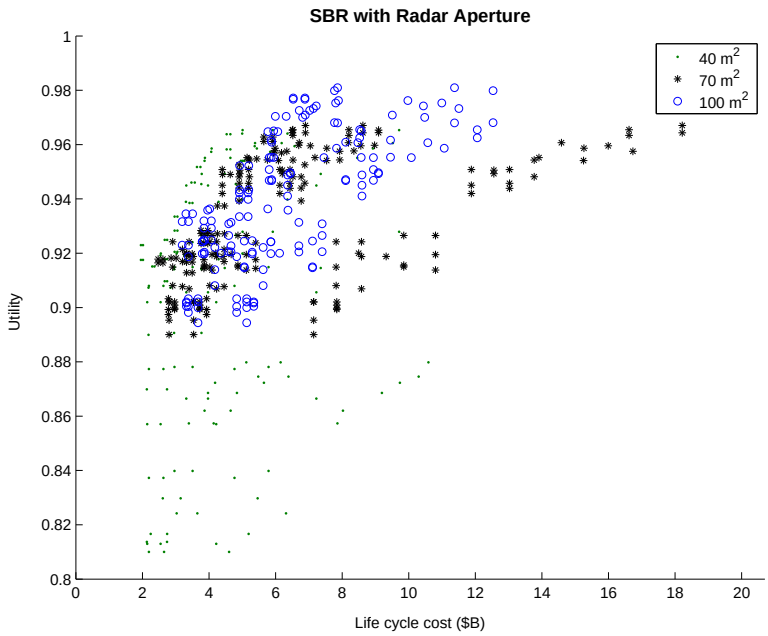


FIGURE 5-2: SBR Cost vs. MAU space coded by aperture area

altitude constellation with a greater number of spacecraft – exactly the trend predicted earlier. Roberts (2003) discusses the impact of other design variables on cost and utility.

5.4 Delivering improved capability over time

One of the more challenging aspects of interpreting the model results as described thus far is that architecture at different technology levels are plotted on the same space, i.e. those that do not deliver capability till 2010 are being compared to those that deliver capability in 2002. Given the attributes defined in Table 5-1, there is no preference expressed for having capability sooner rather than later. The Air Force's selection of an EA strategy, however, clearly demonstrate that such a preference exists. Unfortunately, including this preference within the MAUT framework is quite difficult.

One way to include the preference is to introduce another attribute, 'time at which capability is delivered'. This attribute does satisfy the preferential independence

assumption, i.e. capability earlier will be preferred to capability later regardless of the values of the other attributes. On the other hand, utility independence may not be satisfied. Recall that utility independence means that the shape of the utility function for an attribute does not vary when other attributes are changed. As was shown in Figure 3-3, the shape of the utility function reflects the decision maker's risk aversion. It is quite likely that the decision maker will be more risk seeking the greater the value of the other attributes. The decision maker will be willing to accept greater risk upfront for greater reward (capability). Since capability is defined by the other attributes, this is a violation of the utility independence axiom.

Another approach is to take the preference for earlier capability and treat it as a constraint on the design, i.e. instead of 'earlier capability is preferred', 'capability in 2002 is required'. This approach, though no longer strictly within the MAUT framework, does allow the study of the evolution of SBR architectures over time. Marking each architecture by the year requisite technology will be available (see Figure 5-3), it is clear that many of the highest value (low cost, high utility) architectures will not be available until 2005 or 2010[7]. The decision maker, however, requires capability at an earlier date. Hence, the best solution seems to be a scalable architecture which delivers some initial capability in 2002 and then builds upon that capability as 2005 (and later 2010) technology becomes available.

In transitioning from a 2002 architecture to a 2005 architecture (as well as from a 2005 architecture to a 2010 architecture), the possible transitions that can be considered are limited by the structure of Spaulding's model. Relaxation of these rules and consequent extensions of the analysis are discussed in Chapter 6. These limitations impose a set of transition rules that govern the time evolution of the SBR constellation (for details see Roberts (2003)).

1. All transitions must not decrease utility (i.e. systems must not lose capability)

[7]For the purposes of this discussion, the reader is asked to consider 2001 as the current year

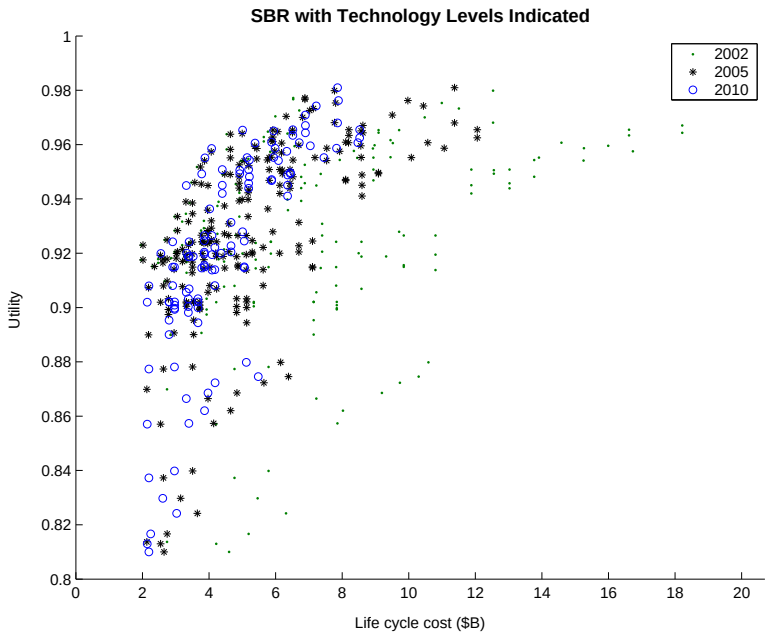


FIGURE 5-3: SBR tradespace marked by year technology is available

2. The satellites in the transition architecture must have the same scan angle and aperture area as the initial architecture.

3. The number of satellites per orbital plane of the initial architecture must be less than or equal to the number of satellites per orbital plane of the transition architecture (i.e. satellites cannot be removed from one orbital plane to populate another in the transition architecture)

4. The number of orbital planes in the initial architecture must be less than or equal to the number of orbital planes of the transition architecture (for Walker constellations this rule is the same as 3)

5. The offset angle between the orbital planes of the initial architecture when divided by the offset angle between the orbital planes of the subsequent architecture must be an integer value (i.e. the relative angle between orbital planes the are member of the initial architecture cannot be changed)

6. Transitions may not involve increasing the altitude of already deployed satellites

7. The altitude of all satellites in the transition architecture must be identical (i.e. the architecture cannot be composed of multiple constellations at different altitudes)

With these rules in place, the time evolution of the SBR constellation can be described in terms of a modular hierarchy. The transition rules determine the design rules in the hierarchy.

5.5 A modular structure for SBR

Recognizing the need to have future capability build upon from existing capability, the following modular structure was developed for SBR. The modular hierarchy has three layers (see Figure 5-4. The top layer represents global design rules that need to be followed by all members of the constellation. These include such specification as constellation wide communication protocols. In the next layer, the constellation is broken down into rings. All the spacecraft in a given ring must have the same altitude, inclination etc. to place them within the rings. These specifications are design rules imposed on members of a given ring. The final layer consists of the individual spacecraft which are the hidden modules in the hierarchy.

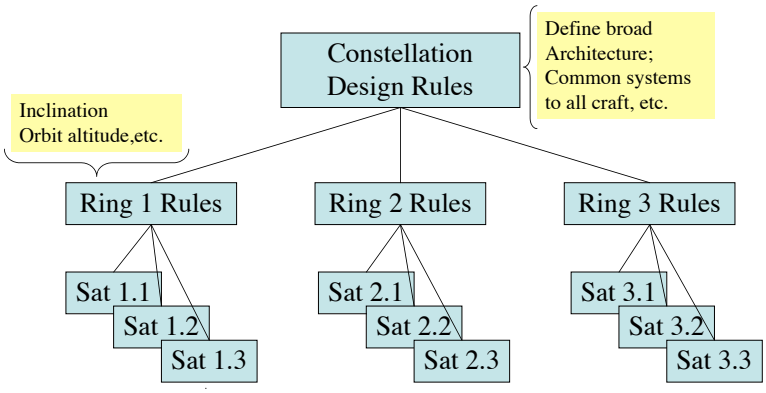


FIGURE 5-4: A modular hierarchy view of the SBR constellation

5.6 Transition from one modular structure to another

Transitions from a 2002 configuration to a 2005 configuration can be understood in term of modular operator being applied to the modular hierarchy in Figure 5-4. Adding new spacecraft to an existing ring is an example of augmenting. Moving a ring to a lower altitude and then adding spacecraft to it is an example of substitution at the ring level and augmenting at the spacecraft level. Adding a new ring is augmenting at the constellation level.

After applying the transition rules mentioned above, a plot was made of number of 2005 architectures that a given 2002 architecture could transition to (see Figure 5-5). Note that one possible 'transition' is to do nothing. Because of the structure of Spaulding's MATE model such a 'null' transition needed to be considered as a transition from a 2002 architecture to 2005 architecture even no real change was made. The number of transitions indicated in Figure 5-5 includes this 'null' transition. Thus, those 2002 architectures which can transition to only one 2005 actually do not change from their 2002 composition.

It is interesting to note that some of the most flexible (in the sense of having many difference transition options) 2002 architecture tend not be Pareto optimal. This reflects that fact the these highly transition-able architecture are over-designed given there altitude and orbital configuration. New spacecraft can be added while still remaining within the orbit/constellation space specified by Spaulding. The Pareto optimal architectures, on the other hand, are at the boundaries of that space using all available locations for spacecraft. Of course, looking at the number of transitions possibilities is not sufficient to ensure that by 2010 the constellation will be in a high value state. A 2002 architecture may allow many transitions, but only to 2005 architecture that marginally improve utility. The designer must explore the particular 2005 architectures to which he has the option of transitioning after launching a particular 2002 architecture.

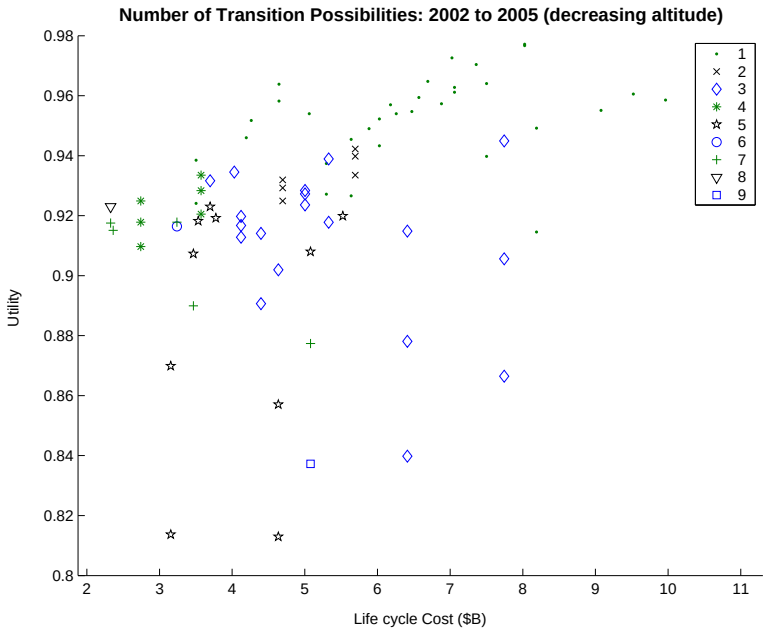


FIGURE 5-5: Each point is a constellation architecture that uses 2002 technology. The symbol represents the number of 2002 → 2005 transitions possible.

5.7 Transition Option Trees

As a starting point, the designer may consider the lone 9 transitions architecture in Figure 5-5. This architecture, ID # 964, maximizes the number of transitions possibilities. Applying the transition rules stated earlier, the designer can plot both utility and cost as function of time for each of the possible constellation growth paths (2002 architecture to 2005 to 2010). The results of architecture 964 are plotted Figures 5-6 and 5-7. The boxed numbers at each node in the tree are ID # of particular architecture. They allow the designer to compare the utility and cost graphs. In addition to transitions where spacecraft are added and/or orbits reconfigured, a 'null' transition is included in which the constellation is left unmodified. These correspond to the transitions that leave utility unchanged.

The cost of a transition reflect the modular nature of the constellation. The designer need

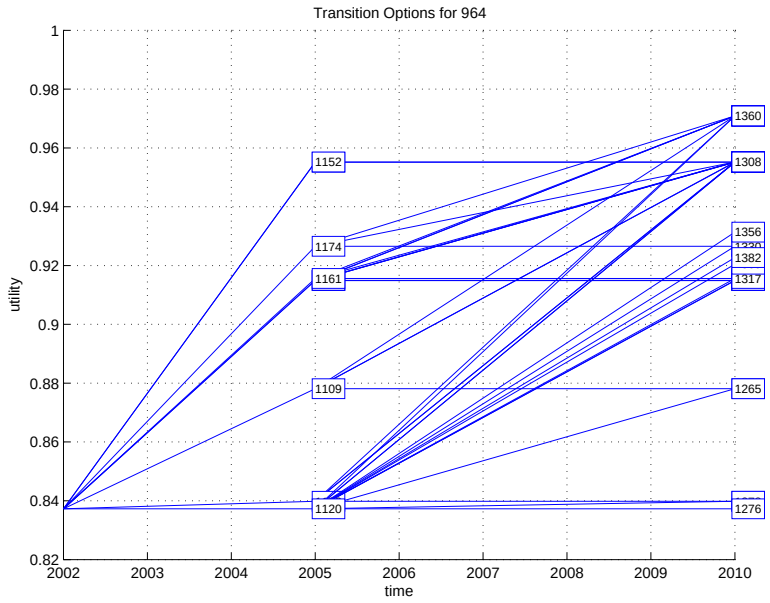


FIGURE 5-6: Utility delivered over time given one start with architecture 964 in 2002

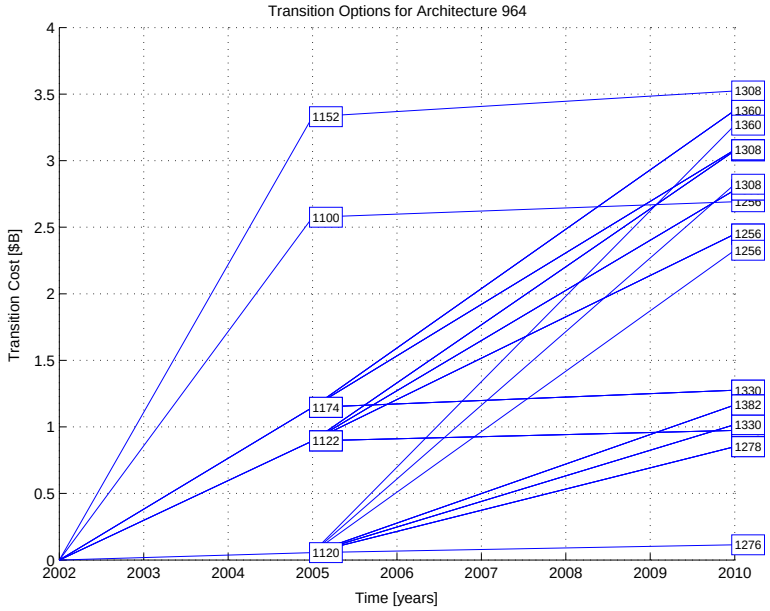


FIGURE 5-7: Cost incurred over time given one starts with architecture 964 in 2002. Only cost beyond 2002 is included. All cost are expressed in 2002 dollars.

only pay for new spacecraft launched into the constellation. This cost consists of non-recurring engineering cost associated with developing the 2005 (or 2010) technology and the cost of building and launching the spacecraft that use the new technology. Since the design rules were followed, there is no cost incurred on account of the existing spacecraft.

Returning to the transition cost and utility charts, the trees for architecture 964 demonstrate one of the peculiar feature of evolutionary development. Sometimes one must forgo benefit now to leave expansion capability for later. Notice that in Figure 5-6, if the utility maximizing choice is made in 2005, then it is impossible to transition to the utility maximizing choice in 2010.

The choice of which evolution path to follow should be driven by risk considerations. Unfortunately, the difficulty in incorporating uncertainty into Spaulding's model precluded inclusion of such analysis within the case study. The process would occur as follows. First, the principle risk at each decision point (2002, 2005) in the evolution strategy would be identified. Then distribution function for cost and utility given these uncertainties would be created. At each decision point, the expected value maximizing choice would be made starting at the 2005 decision and working backwards. The result would be a space of expected utility streams with corresponding expected cost streams. The decision makers must then decide, in consultation with other stakeholder, which strategy best balances cost and value.

Once all stakeholders agree on a particular transition strategy as laid out in plots similar to Figures 5-6 and 5-7, they will have established as Boehm calls it, a Life Cycle Architecture for the system. Included in this architecture is the baseline (2002) capability that constitutes the Initial Operating Capability. By agreeing to a particular strategy, the stakeholders are not committing themselves to dogmatically following that strategy. Rather, their only expenditure is the first step, the IOC. By deploying the chosen 2002 architecture, they purchase options on the possible 2005 (and by extension 2010)

architectures. They are under no obligation to exercise those options in the even that uncertainty does not resolve in an advantageous manner. Should they gain insight into the system (both from needs and solutions perspective) from fielding the 2002 version, they can take advantage of computational efficiency of the MATE process and re-evaluate transition options at a later date.

The above discussion has provided a flavor of how MATE and modularity can be used to analyze an EA/SD problem. Many issues however remain for future discussion, a summary and some suggestions are provided in the concluding chapter.

Chapter 6

Summary and Future Work

6.1 Summary of major findings

As the DSB/AFSAB report concluded, there are many new challenges facing the US aerospace industry today. Among them is a need to adapt to a rapidly changing threat and budgetary environment. In an attempt to meet this challenge, the DoD has chosen Evolutionary Acquisition as its preferred strategy.

The principal feature of Evolutionary Acquisition is incremental delivery of increasing capability over time, where the contents of each increments are determined as the program progresses. There is a strong emphasis incorporating feedback from when planning future development. The time between deliveries to users is purposefully kept short to ensure timeliness of both the deliveries and the feedback received from users. By being incorporating regular feedback, EA allows for rapid adjustment to the changing environment. Furthermore, the keeping increment short and relying on existing technologies within an increment, budget and schedule risk are reduced. To achieve these benefits however, an appropriate process is needed.

Aldridge recommends Spiral Development as such a process. Both Unger and Boehm discussions about SD support this recommendation. Boehm defines three anchor points, Life Cycle Objectives, Life Cycle Architecture and Initial Operating Capability. LCO specifies the needs that the system will satisfy. By revising the LCO with input from all stakeholder at each iteration, the LCO is kept current to changing. The LCA is chosen so as to be flexible enough incorporate changes to the system so that new objectives can be me without sacrificing old ones. Both changing objectives and including new capabilities to meet those objectives are consistent with an EA strategy. Hence spiral development seems well suited for EA. Unger, taking a more quantitative approach found that SD processes handled market risk (i.e. changing users needs) very well. This is because the frequent interaction with users provided continuous feedback to developers. To make such a continuous feedback stream possible, Unger recognized the need for ease of integration. If the program takes years and half its budget to deliver a prototype to the users, there clearly will not be many cycles of feedback and improvement. Though EA/SD seem to alleviate some of the problems mentioned in the DSB/AFSAB report, implementing them imposes new challenges. Two critical challenges are how to capture user needs given such diverse user base and finding a system architecture that delivers the flexibility required of the LCA and the ease of integration pointed out be Unger.

The first of these challenges is met by the Ross and Diller's MATE process. The MATE process is a three phase process that emphasize taking a global or system viewpoint when making design decisions. One of its most powerful features is the ability to aggregate disparate sources of value into a single overall utility metric. Multi- attribute-utility-theory defines such a metric, the multi-attribute utility. Unfortunately the MAU is an ordered metric and as such it only measures if one design option is preferred over another by the decision maker. It does not indicate degree of preference. Despite this limitation, its axiomatic foundations make it a powerful tool in decision analysis. To determine the MAU, the decision maker must make explicit a series of preference function over each of the attribute of the system they value. These preference functions are elicited through an

interview technique which though sometimes cumbersome, does provide a reliable measure of preference.

The second challenge is met by a modular system architecture. Baldwin and Clarke define modularity in terms of design structure matrices that exhibit a block-diagonal structure. By breaking the system into these independent blocks, the modular system architecture limits the portion of the system that can be affected by a design change thereby easing integration. This ease of integration allows the system to be upgraded in-service and thus enable the evolutionary strategy. The choice of whether of not to make changes to the system can be delayed until the risk related to such changes has been minimized. This benefit comes at the price of following design rules that allowed the decoupling to take place. The degree to which such decoupling can occur varies greatly from one aerospace product to the next depending upon the relative dominance of information to power interactions within the system.

Whitney identifies the 'power' of a system as being indicative of ease of decoupling. For example, a high power system like a jet engine is able to transfer large energies efficiently from one subsystem to the next (compressor to combustor to turbine) only by carefully tuning (or matching in propulsion terminology) each of the sub-systems to the others. This matching implies a strong interdependence between the sub-system – changing one will surely change the others. The design rules needed to decouple such a system while maintaining efficiency would severely limit the design. In contrast, all flows in software are purely informational and as such require little energy. Software and the jet represent the two end of modularity spectrum. The key differentiator between the two is the relative predominance of information or energy flows. If energy flows dominate, like in the engine modular design will be difficult. Conversely if information flows dominate modular design may be useful. The examples of modular design discussed bear out this viewpoint. For example, in the TRW FSVPL high power components such as the propulsion system were monolithic, while low power, information carriers such as electronic exhibited a high

modular structure.

The case study of Space Based Radar demonstrates the power of both MATE and modular approaches. By using the MATE approach, Spaulding was able to combine the desired attributes of several different mission into single utility metric. The tradespace exploration features of MATE revealed major trends in the SBR tradespace such as the lack of Pareto optimal designs that use a 70 m^2 aperture. By time phasing the trade-space and modeling the constellation as modular hierarchy, transition options from a given current year architecture to possible future year architectures could be studied. Such an approach allows for easy identification of architectures that deliver acceptable capability initially with little cost commitment up front, yet can still be expanded into high performing architectures in the future (see Figures 5-6 and 5-7).

6.2 Suggestions for future work

Though modular architecture seems well-suited to EA/SD, several areas for future work remain.

In the SBR case study only transitions from a single 2002 architecture were considered. In fact, each point in Figure 5-5 spawns cost and utility trees similar to those shown for architecture 964 in Figures 5-7 and 5-6. The choice of 964 was based on a heuristic that a large number transition option would tend to increase the likelihood of finding a favorable one.

In keeping with the MATE philosophy of tradespace exploration, all the trees should be considered. Unfortunately this leads to combinatorial explosion of possible evolution paths. Ideally, there would be a common metric to value the utility delivered over time for particular evolution path. Such a metric would be analogous to present value (PV) for the cost stream. Unfortunately discounting techniques such as PV are difficult if not

impossible to implement in multi-attribute problems or problems that do not involve consumption streams (Keeney and Raiffa, 1993). The required number and scope of utility interviews quickly becomes unmanageable. An alternative approach worthy of further study is the Quality Adjusted Life Year (QALY) often used in the medical community (Miyamoto et al., 1998). When evaluating treatment options, patients are faced with a similar utility delivered over time problem. Quality of life is a multi-attribute quantity that varies as a function of time. QALYs essentially scale the length of each year of remaining life by the quality of life expected in that year. The scaled years are then added to form QALYs. Thus many years of low quality are equivalent to fewer years of high quality. A similar technique could be applied to create a 'Quality Adjusted Service Life' metric for systems such as the SBR. With a such a metric in place, the decision maker could in one plot see the expected value delivered over time (QASL) and cost incurred by all possible evolution paths. This would allow the identification of a set of Pareto optimal evolution strategies for exercising the options purchased when a particular 2002 architecture is chosen.

Improvements should also be made in the underlying model for the SBR, in particular, the spacecraft and orbit models should rewritten in a parametric form to allow modeling of more complex transitions. Uncertainty especially in relation to technology development needs to incorporated. With these changes made, the options framework described in Chapter 5 can be implemented making risk the driver in architecture selection. Additional upgrade possibilities could also be explored with an improved model. For example, if additional fuel could be included on the satellites then plane changes and altitude increases could be considered.

Modular design of aerospace systems is another area in which more work needs to be done. The SBR case study did not consider upgrading spacecraft on-orbit. The high degree of coupling between systems within the satellites makes such servicing difficult. If spacecraft were constructed from modular components and on-orbit servicing

infrastructure was available, then upgrades both for life extension and technology improvement could be considered. The TRW FSVPL demonstrates that modular spacecraft are possible, however, extending that concept to in-space upgrades is an open area of research. The forthcoming paper by Long and Hastings (2004) is determining the value of servicing architectures and the demands they place on satellite design.

In the broader context of evolutionary systems, several open questions remain. The discussions in the SBR case study assumed that needs, as quantified by MAUT, remain fixed throughout system evolution. It was argued earlier, however, that responding to changing user needs was one of the key requirements of an evolutionary strategy. Nothing within the MATE framework prevents the decision maker from revising utility curves or introducing new attributes. In fact, because MAUT only captures a sub-set of the decision maker's true preference space, it is expected that such revision take place. Derleth (2003) examined the redefinition of attributes when modeling the evolution the Small Diameter Bomb. He also looked at the possibility of new technology introducing new capability in the form of new attributes. An open question, however, is which system architectures can most effectively adapt to changing user needs. The properties of modularity discussed above seem to make it a likely candidate; however, no studies have been done to demonstrate this. Not only do needs change over time, but those expressing the needs may change as well.

Though MATE-CON successfully includes multiple stakeholders, multiple decision makers has been a challenge. In most system with multiple users there is a supra-decision maker who can balance resources across different uses. The supra-decision maker's preferences can be used to form the MAU. For example, though many services may use an SBR, the joint staff can act as a supra-decision maker in balancing the needs of one service against another. In some cases however such a supra-decision maker cannot be readily identified. Arrow's impossibility theorem states that the preferences of the group may not be discernible from the expressed preferences of the members. Thus is situation

with multiple decision makers, negotiation may be necessary. Resolving this issue is another important area of research.

Finally, there is the question of technology development. EA/SD as described only makes wide use of a technology when its has been fully developed and is ready for production. The availability of the appropriate technology is crucial to any development effort. However, these technologies require investment in order to mature. Determining the best way to invest in promising technologies is crucial to the success of the DoD's vision of EA/SD as its preferred acquisition strategy.

References

ACE Office. New office gives Center an ACE. *Agile Acquisition Newsletter*, 2002.

E. C. Aldridge. Evolutionary acquisition and sprial development. Memorandum, April 2002.

Carliss Y. Baldwin and Kim B. Clark. *Design Rules: The Power of Modularity*. MIT Press, Cambridge, MA, 2000.

B. Boehm. *Spiral Development: Experience, Principles, and Refinements*. University of Southern California, 2000.

M. Caceres. Industry insights. *AIAA Aerospace America*, April 2003.

Stan Crock. Is this missile defense an eagle – or an albatross? *Business Week*, 2002.

Richard de Neufville. *Applied Systems Analysis*. McGraw-Hill Publishing Company, 1990.

Lawrence J. Delaney. Evolutionary acquisition for C2 systems. *Air Force Instruction 63-123*, 2000.

Jason E. Derleth. Multi-attribute tradespace exploration and its application to evolutionary acquisition. Master's thesis, MIT Aeronautics and Astronautics, 2003.

Nathan P. Diller. Utilizing multiple attribute tradespace exploration with concurrent design for creating aerospace systems requirements. Master's thesis, MIT Aeronautics and Astronautics, 2002.

DOD. DOD5000 series regulations, 2004. http://dod5000.dau.mil/.

DSB/AFSAB. *Report of the Defense Science Board/Air Force Science Advisory Board Joint Task Force on Acquistion of National Security Space Programs*. Department of Defense, 2003.

Beth Emery and Dilip Punatar. Flexible space vehicle production line (FSVPL). In *Lean Aerospace Initiative Plenary Conference*, 2002.

Bobak Ferdowsi. Product development strategies in evolutionary acquisition. Master's thesis, MIT Aeronautics and Astronautics, 2003.

G. W. Goodman. Space-based surveillance: Reconnaissance satellites are a national secutrity sin qua non. *Armed Forces Jounal, The ISR Journal*, 3(3), 2002.

Wayne M. Johnson and Carl Johnson. The promise and perils of spiral acquisition: A practical approach to evolutionary acquisition. *Agile Acquisition Newsletter*, 2002.

Ralph L. Keeney and Howard Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. Cambrige University Press, Cambridge, UK, 1993.

Andrew Long and Daniel E. Hastings. Catching the wave: A unique opportunity for the development of an on-orbit satellite servicing infrastructure. In *AIAA Space 2004*, 2004.

M. W. Maier and E. Rechtin. *The Art of Systems Architecting*. CRC Press, 2000.

John M. Miyamoto, Peter P. Wakker, Han Bleichrodt, and Hans J. M. Peters. The zero-condition: A simplifying assumption in qaly measurement and multiattribute utility. *Management Science*, 44(6):839–849, 1998.

Sias Mostert and Jan-Albert Koekemoer. The science and engineering payloads and experiments on SUNSAT. *Acta Astronautica*, 41(4-10):401–411, 1997.

Christopher J. Roberts. Architecting evolutinary strategies using spiral development for space based radar. Master's thesis, MIT Technology and Policy, 2003.

Adam M. Ross. Multi-attribute tradespace exploration with concurrent design as a value-centric framework for space system architecture and design. Master's thesis, MIT Aeronautics and Astronautics, and Technology and Policy, 2003.

Satwik Seshasai. A knowledge based approach to facilitate engineering design. Master's thesis, MIT Electrical Engineering and Computer Science, 2002.

Timothy J. Spaulding. Tools for evolutionary acquisition: A study of multi-attirbute tradespace exploration (MATE) applied to the space based radar (SBR). Master's thesis, MIT Aeronautics and Astronautics, 2003.

Darian W. Unger. *Product Development Process Design: Improving Development Response to Market, Technical, and Regulatory Risk*. PhD thesis, MIT Technology, Management and Policy, 2003.

Daniel E. Whitney. Physical limits to modularity. *MIT Engineering Systems Division Internal Symposium*, 2002.