NPS-AM-13-C10P01R06-035

EXCERPT FROM THE

PROCEEDINGS

OF THE

TENTH ANNUAL ACQUISITION RESEARCH SYMPOSIUM ACQUISITION MANAGEMENT

Fewer Mistakes on the First Day: Architectural Strategies and Their Impacts on Acquisition Outcomes

Linda McCabe and Anthony Wicht
Massachusetts Institute of Technology

Published April 1, 2013

Preface & Acknowledgements

Welcome to our Tenth Annual Acquisition Research Symposium! We regret that this year it will be a “paper only” event. The double whammy of sequestration and a continuing resolution, with the attendant restrictions on travel and conferences, created too much uncertainty to properly stage the event. We will miss the dialogue with our acquisition colleagues and the opportunity for all our researchers to present their work. However, we intend to simulate the symposium as best we can, and these *Proceedings* present an opportunity for the papers to be published just as if they had been delivered. In any case, we will have a rich store of papers to draw from for next year’s event scheduled for May 14–15, 2014!

Despite these temporary setbacks, our Acquisition Research Program (ARP) here at the Naval Postgraduate School (NPS) continues at a normal pace. Since the ARP’s founding in 2003, over 1,200 original research reports have been added to the acquisition body of knowledge. We continue to add to that library, located online at www.acquisitionresearch.net, at a rate of roughly 140 reports per year. This activity has engaged researchers at over 70 universities and other institutions, greatly enhancing the diversity of thought brought to bear on the business activities of the DoD.

We generate this level of activity in three ways. First, we solicit research topics from academia and other institutions through an annual Broad Agency Announcement, sponsored by the USD(AT&L). Second, we issue an annual internal call for proposals to seek NPS faculty research supporting the interests of our program sponsors. Finally, we serve as a “broker” to market specific research topics identified by our sponsors to NPS graduate students. This three-pronged approach provides for a rich and broad diversity of scholarly rigor mixed with a good blend of practitioner experience in the field of acquisition. We are grateful to those of you who have contributed to our research program in the past and encourage your future participation.

Unfortunately, what will be missing this year is the active participation and networking that has been the hallmark of previous symposia. By purposely limiting attendance to 350 people, we encourage just that. This forum remains unique in its effort to bring scholars and practitioners together around acquisition research that is both relevant in application and rigorous in method. It provides the opportunity to interact with many top DoD acquisition officials and acquisition researchers. We encourage dialogue both in the formal panel sessions and in the many opportunities we make available at meals, breaks, and the day-ending socials. Many of our researchers use these occasions to establish new teaming arrangements for future research work. Despite the fact that we will not be gathered together to reap the above-listed benefits, the ARP will endeavor to stimulate this dialogue through various means throughout the year as we interact with our researchers and DoD officials.

Affordability remains a major focus in the DoD acquisition world and will no doubt get even more attention as the sequestration outcomes unfold. It is a central tenet of the DoD’s Better Buying Power initiatives, which continue to evolve as the DoD finds which of them work and which do not. This suggests that research with a focus on affordability will be of great interest to the DoD leadership in the year to come. Whether you’re a practitioner or scholar, we invite you to participate in that research.

We gratefully acknowledge the ongoing support and leadership of our sponsors, whose foresight and vision have assured the continuing success of the ARP:

- Office of the Under Secretary of Defense (Acquisition, Technology, & Logistics)
- Director, Acquisition Career Management, ASN (RD&A)
- Program Executive Officer, SHIPS
- Commander, Naval Sea Systems Command
- Program Executive Officer, Integrated Warfare Systems
- Army Contracting Command, U.S. Army Materiel Command
- Office of the Assistant Secretary of the Air Force (Acquisition)
- Office of the Assistant Secretary of the Army (Acquisition, Logistics, & Technology)
- Deputy Director, Acquisition Career Management, U.S. Army
- Office of Procurement and Assistance Management Headquarters, Department of Energy
- Director, Defense Security Cooperation Agency
- Deputy Assistant Secretary of the Navy, Research, Development, Test, & Evaluation
- Program Executive Officer, Tactical Aircraft
- Director, Office of Small Business Programs, Department of the Navy
- Director, Office of Acquisition Resources and Analysis (ARA)
- Deputy Assistant Secretary of the Navy, Acquisition & Procurement
- Director of Open Architecture, DASN (RDT&E)
- Program Executive Officer, Littoral Combat Ships

James B. Greene Jr.	Keith F. Snider, PhD
Rear Admiral, U.S. Navy (Ret.)	Associate Professor

Acquisition Management

Fewer Mistakes on the First Day: Architectural Strategies and Their Impacts on Acquisition Outcomes

Linda McCabe—McCabe is a member of the technical staff in the Airborne Networks Group at MIT Lincoln Laboratory. Since joining the laboratory in 2005, McCabe has participated in a wide range of research programs from developing new methods to evaluating advanced tactical networks to planning and executing airborne networking field experiments. More recently, she has been involved in an evaluation of technology transfer, identifying both positive and negative lessons learned from a wide range of programs. Prior to joining Lincoln Laboratory, McCabe led a division at SAIC that focused on wargame design and execution. McCabe holds a BA in international relations from Boston University and an MA in security policy from The George Washington University.

Anthony C. Wicht—Mr. Wicht completed a Master of Science in Aeronautics and Astronautics at Massachusetts Institute of Technology (MIT) in 2011. He also holds a Bachelor of Laws degree and a Bachelor of Engineering degree with First Class Honours from the University of New South Wales, Australia. His research at MIT focused on architecting complex engineering systems like space infrastructure. Prior to attending MIT, Mr. Wicht worked as a project finance lawyer at a major Australian law firm on legal aspects of financing and constructing large engineering projects. He is a former president of the National Space Society of Australia and is co-chair of the Australian Space Development Conference series.

Abstract

Reducing cost and development time, while preserving acceptable levels of performance, is a priority for all government-sponsored complex product development. One avenue for improving outcomes is to use architecting strategies to guide development decisions. Frequent examples are commonality, interoperability, modularity, flexibility, extensibility, robustness, openness, and adaptability. A second avenue for improving outcomes is better acquisition strategies. The two are often considered in isolation. This paper begins an examination of how the choice of architecting strategy affects the choice of acquisition strategy, and vice versa.

As a first step, the paper synthesizes existing literature and provides straightforward definitions of each of the architecting strategies. As a second step, the paper maps each of the defined architecting strategies against two common axes of acquisition design, specifically openness to competition and sensitivity to requirements change. The conclusions, while tentative, show that increasing attention to the interaction between how systems are designed and how they are acquired may have a significant effect on the cost, schedule, and performance of complex product development.

Introduction

Reducing cost and development time, while preserving acceptable levels of performance, is a priority for complex product development in both military and civilian systems. One avenue for improving development is to use particular architecting strategies to guide high-level development decisions. These strategies reflect the priorities that the customer places on different aspects of the product. Different strategies lead to different design decisions and ultimately different outcomes. For example, developments based on the “robustness” strategy might trade high performance under specific conditions for acceptable performance over a range of conditions. This paper considers common, modular, open, flexible, adaptable, robust, extensible, and interoperable architecting strategies. Many of these strategies have at one time or another been exhorted as the solution to acquiring complex products.[1]

However, there appears to be a disconnect between the use of these strategies and their effect on complex product development, particularly in the government sphere. New architecting strategies are encouraged at the highest levels, but the cost of acquiring new complex systems continues to climb (Berteau et al., 2010; Peters, 2009; Moore, 2011).

We observed two factors that we hypothesized to be likely contributors to the lack of impact of these architecting strategies on project acquisition costs. The first factor is a lack of effective communication about what the architecting strategies mean, caused by a scarcity of definitions for the strategies, and compounded by the application of the strategies in engineering disciplines far removed from where the terms were first used. The second factor is a universal application of the strategies to all acquisitions, rather than to just those acquisitions where the strategy would be particularly relevant and helpful.

The existing literature does not provide sufficient guidance on the architecting strategies, or the types of product acquisitions where they should be applied. The Defense Acquisition University (DAU) provides an excellent start, with brief definitions of many of the important terms in its *Glossary of Defense Acquisition Acronyms and Terms* (Hagan, 2009), although *flexibility*, *adaptability*, and *extensibility* are not defined. A symposium paper from the MIT Engineering Systems Division discussing architecting strategies generally is also useful, but does not specifically define the strategies and the differences between them (Crawley et al., 2004). The DoD dictionary, which consolidates definitions provided in doctrine documents, defines only commonality (Joint Chiefs of Staff, 2010). Finally, even within the top 100 articles found doing a Google Scholar search for articles containing the words *robust*, *flexible*, *common*, *interoperable*, and *extensible*, none sets out definitions for these terms. Although these searches may not be exhaustive, they represent a much more detailed search for definitions than a professional is typically able to conduct when investigating architectural options.

The specific objectives of this paper, therefore, are twofold. First, the paper aims to synthesize existing definitions of the design strategies into a single definition. This synthesis will provide a starting point for discussion of what the design strategies mean. Extensive comment and discussion about these definitions is anticipated, but consolidating all definitions into a single document and posing possible definitions for discussion is a prerequisite for this discussion, and represents an advance on the existing literature.

Second, the paper aims to provide a coarse analysis of the acquisition scenarios to which each strategy is well suited. Such an analysis makes the broad point that different acquisition scenarios merit different design strategies, and not one design strategy is a panacea for all acquisition challenges. The analysis also makes more specific findings about the regions of suitability of each design strategy, in terms of certainty of requirements and openness to competition.

In Section I, the paper examines the literature in detail. It presents a number of observations about the potential for confusion in the existing literature, and also highlights

[1] *Commonality*: “Commonality is the key to affordability” (DoD, 2013). *Interoperability*: “It is DOD Policy that . . . Department of Defense pursue materiel interoperability with allies and coalition partners” (Carter, 2009). *Open Systems Architectures*: “[Acquisition Professionals within DOD will ...] require open systems architectures” (Carter, 2010). “Program managers shall employ MOSA [Modular Open Systems Approach] to design for affordable change, enable evolutionary acquisition, and rapidly field affordable systems that are interoperable in the joint battle space” (DoD, 2008).

how reference is frequently made to the design strategies without an accompanying explanation of what is meant by those strategies. In Section II, standard definitions for each of the chosen architecting strategies (common, modular, open, flexible, adaptable, robust, extensible, and interoperable, referred to as the “Eight Strategies”) are proposed. Each definition is illustrated with examples from previous government acquisitions. Section III presents an analysis of the newly defined design strategies against two important acquisition parameters: (1) certainty of requirements, and (2) number of organizations involved.

Finally, Section IV concludes the paper and presents suggestions for further work in this area.

The Existing Literature Confuses More Than It Clarifies

If architectural strategies are to be used for complex government acquisition projects, there is a need for all involved to understand the meaning of these strategies. This section of the paper reviews the existing definitions of architectural strategies and considers whether the definitions are consistent and easy to find. Definitions that are consistent and easy to find would be expected as a prerequisite to effectively using the architectural design strategies across government acquisitions.

At the outset, it is important to be precise with terminology used in this paper. An architecture is “an abstract description of the entities of a system and the relationships between those entities” (Crawley et al., 2004). Put another way, architecture is “the arrangement of the functional elements into physical blocks” (Ulrich & Eppinger, 2008). Architecture is the underlying concept of how a complex system is brought together, the process of relating form to function in order to create value where value is benefit at cost.

Different architectures have different properties. When the architecture clearly brings about a certain property in the final system, the final system is often referred to as having a “property-architecture.” For example, if the architecture is such that the final system is robust, then the system is described as having a robust architecture. We refer to the process of designing an architecture for a desired result as an architecting strategy.

Surprisingly, we were able to find only a small number of references that presented and compared all or most of the architecting strategies. De Weck, Ross, and Rhodes (2012) investigated most of the architecting strategies presented in this paper but were concerned with them as “system properties” rather than architecting strategies. Their paper also focused on the interrelationships between the strategies rather than describing the strategies themselves. The symposium paper by the MIT Engineering Systems Division, *The Influence of Architecture in Engineering Systems* (Crawley et al., 2004), investigated definition in more detail. The paper described how architecture influences the properties of created systems, resulting in robustness, adaptability, flexibility, safety, and scalability. However, Crawley et al. focused on the importance of architecture rather than detailing different outcomes from the architecting process. Fricke and Schultz (2005) presented an excellent side-by-side view of adaptability, agility, flexibility, and robustness, but they did not extend their analysis beyond these “changeable” architectures.

Finding other papers that compared architecting approaches proved difficult. A Google Scholar search for articles containing the words *robust*, *flexible*, *common*, *interoperable*, and *extensible* gives a surprising number of results—13,800—but no paper in the top 100 sets out definitions for these terms. In other cases, papers defined a few of the terms in domain-specific areas. For example, Ferguson, Siddiqi, Lewis, and de Weck (2007) examined flexible and reconfigurable systems in product design, but their definitions would require thought and interpretation before application to another area, for example, information architectures.

Nor is there assistance from the key textbooks in the area. The Art of System Architecting mentions some of these architecting strategies but does not contrast them or provide extensive detail. In *Architecture and Principles of System Engineering*, Dickerson and Mavris (2010) did not present definitions for any of the Eight Strategies. However, the terms themselves are mentioned, in some cases, in the sense we use them ("interoperable and cost effective military systems" [Dickerson & Mavris, 2010, p. 148], "methods and techniques to … design for robustness relative to uncertain operational environments" [p. 313]), and in other cases in very different contexts (for example, *openness* is used in the context of stakeholder discussions, and *flexibility* in the context of "development flexibility, such as environmental limitations or regulatory standards").

In the government context, the situation does not improve greatly. A complete set of definitions does not exist. Of the Eight Strategies, the DoD dictionary, which consolidates definitions provided in doctrine documents, defines only commonality (Joint Chiefs of Staff, 2010). The DAU provides brief definitions of many of the important terms in its *Glossary of Defense Acquisition Acronyms and Terms*, although flexibility, adaptability, and extensibility are not defined (Hagan, 2009). Further, some of the terms are narrowly defined. For example, *module* is used only in the context of software architectures. The DAU's online "Terms and Definitions" (2013) defines three out of the Eight Strategies, and defines the substance of extensibility, though referring to it as scalability.

Compounding the problem, the same term is used in the same community to mean different things. Defense Directive 5000.01 (Wolfowitz, 2007) emphasizes five key acquisition policies, one of which is flexibility. Dickerson and Mavris (2010) summarized flexibility in this context as the "need to structure each acquisition program according to the set of strategies, documentation, reviews, and phases that make sense for this program" (p. 290). This is a different definition of flexibility than used by system architects in describing the properties of their systems. However, there are some bright spots in the government landscape. The push towards a "modular, open-systems architecture" by the Open Systems Joint Task Force (2004), shows significant development of the modular and open systems concepts through tens of pages of principles, definitions, and examples.

Definitional confusion is not entirely due to a deficiency in the literature, however. The architecting strategies are often mentioned in the same breath but in fact are concerned with quite dissimilar things. *Adaptability*, *flexibility*, and *robustness* are characteristics of an end-product that describes how the product interacts with its environment, especially as that environment changes. *Extensibility* describes how the product is able to improve over time. *Interoperable* describes how the product interacts with other products in the operations phase. *Commonality* describes similarities with other products, usually in the development and operations phases. *Modularity* describes the physical structure of the product. *Openness* describes the process of acquiring the product. Therefore, it is not surprising that a single paper does not cover the range of architecting strategies, because they are quite different.

Further, the architectural strategies inter-relate. Modularity emphasizes simple, well-understood interfaces and so enables commonality (through reuse of existing products) and openness (by more easily tying together the contributions of different participants). Interoperable architectures require knowledge of the systems that interoperate, implying some level of openness. Interoperable architectures also work because of some degree of commonality, usually in the patterns of information exchange, so an interoperable architecture could also be described as having, for example, a common communications protocol.

The difficulties with using the existing literature to define architecting strategies can therefore be summarized as follows:

- No single reference presents and compares all the architecting strategies.
- Several key references refer to architecting strategies without defining them, assuming that they are well understood.
- Where definitions are provided, they are often domain-specific.
- The words chosen for the architecting strategies are sometimes used by the same communities, with different meanings.
- The strategies interrelate, and multiple architecting strategies are often used to achieve a given result.

Although this may appear a formidable list of obstacles, clear, widely available definitions of the architecting strategies will assist with resolving all of these difficulties. In the defense acquisition context, referring to the definitions of these strategies when proposing them as mandatory considerations in acquisition would improve communication of the desired outcomes. There is a precedent for such definitional foundation in the commonality literature. The RAND Corporation produced a report containing a standard commonality lexicon (Held, Lewis, & Newsome, 2007). A similar report examining the definitions described in Section II, with more detail and rigor, presents a possible solution to the current confusion in the literature.

Defining Architectural Strategies

In an attempt to remedy some of the confusion outlined in the previous section, this section provides an overview of architecting strategy definitions from the engineering literature, a relevant DoD example of each definition, a discussion of the definition as it relates to process versus architecture, and, because these strategies are often painted as a panacea for all new-product development ills, a description of the possible downsides of the approach. In the definitions that follow, we begin by discussing a simple example of each architecture strategy. Because low complexity examples are rare in the real world, we also discuss the application of each strategy to a more complex, and where possible, “system of systems” example.

Flexible Architectures

A flexible architecture is one that is easily modified to respond to changing requirements (Crawleyet al., 2004; Fricke & Schultz, 2005; Ferguson et al., 2007). The modification requires work to be done on the system. For example, an architecture that allows different external stores (often referred to as pods) to be loaded on military aircraft to provide different functionality for different missions would be described as a flexible architecture. External stores can provide numerous functions, including—but not limited to—weapons, additional fuel, electronic counter measures (ECMs), communications, and sensors. A more complex example is the ability to load different software onto pre-defined hardware, such as is expected from software programmable radios. Loading new software changes the functionality of the radio to suit the operating environment. In each case, the designers considered that easily changing the system performance was important, and allowance for such changes was built into the architecture. The architectural choices permit product flexibility and therefore are described as a flexible architecture. The benefit of a flexible architecture is that a particular design continues to perform even as the environment changes. For example, an entirely new airplane is not required simply because the range requirements for a certain mission exceed the internal fuel storage capacity of the aircraft.

Flexibility is often associated with modularity if the flexibility arises through the swapping of modular parts (for example, weapons).[2] Flexibility need not be dependent on a modular architecture, however. A flexible system could allow for software changes to be inserted without even unpacking the part from the case, as in the case of the Block III High Speed Anti-Radiation Missile (HARM), a system designed to destroy radar equipped air defenses. The Block II HARM has its own software operating system, which can be upgraded in the field—and in the crate—to redefine its flight profile, its function, and how it interacts with the targeting system onboard the aircraft system.

Flexibility is not always a positive attribute, however, as there is a price associated with designing systems to be flexible. Flexibility should be used only where uncertainty of requirements for the system means that the strategy is required. Crawley et al. (2004) put this succinctly:

> In some cases, flexibility comes at a price—namely, efficiency in some form. Flexibility may require over-design, generic components, extra interfaces, or changeover time. A less flexible system might have more focused components, fewer interfaces, and no loss due to changeover.

Flexibility can in fact increase overall lifetime costs of a system, especially if the product lifetime is shorter than expected, due to the significant up-front cost. As the Army's Future Combat System program office pointed out in its reaction to a GAO (2009) report,

> Because of the significant amount of new technology development and the emphasis on laying a good, flexible architecture foundation, development effort/costs may not follow typical expenditure rates as other projects, and a larger percentage will be needed in the early stages of the program.

Adaptable Architectures

Fricke and Schultz (2005) described adaptable systems as systems that "deliver their intended functionality under varying operating conditions through changing themselves." In other words, an adaptable architecture modifies itself to meet a changing environment. An example from the commercial world is commercial power generation, which automatically brings additional power production online during high demand periods. In the defense context, an example of an adaptable system is radar. Most radar systems are able to change their receiver gain automatically in order to filter out noise generated by jamming.

The difference between flexible and adaptable is subtle, and in the experience of the authors, those using the terms do not always grasp the difference. In particular, either term is often used as a catch all for the meaning of both terms. The difference between adaptable and flexible architectures has important cost implications for DoD projects. Adaptability usually places significantly greater demands on a system than flexibility but may be warranted in some cases, for example, where human intervention is impossible[3] (such as a pacemaker), or where human reaction times are too slow (for example, the ACESII ejection seat, which automatically changes its ejection profile based on the altitude and airspeed of the aircraft at the time of ejection).

---

[2] De Weck, Ross, and Rhodes (2012) showed this as a strong link in their diagram of "ility co-occurrence in the literature."

[3] In the DoD context, adaptability in the context of situations where human intervention is impossible is tied to autonomy, which is commonly not acceptable given the high stakes involved in warfare and the unwillingness to take the human decision-maker out of the loop.

There is also overlap with other terminology. An open architecting process could be described as a flexible architecture because it allows new design implementations to be introduced over time. An “extensible” architecture can also be changed over time and therefore be characterized as “flexible,” albeit at the most inflexible end of flexible.[4] Finally, a modular approach enables flexibility but is not enough in itself to guarantee flexibility.[5] To conceptualize this, imagine a modular system, such as the aircraft with weapons discussed previously, where the weapons racks were welded to the aircraft frame. The design is no less modular, but the architecture is no longer flexible.

Robust Architectures

A robust architecture is one that is able to meet its performance specification over a wide range of, often unanticipated, external conditions and still perform well (Hagan, 2009; Crawley et al., 2004; Fricke & Schultz, 2005). This design strategy is often used when there is high uncertainty over the future performance requirements of a product (Thomke, 1997), or when the system itself is complex and not well understood (Crawley et al., 2004). The design approaches to achieve robustness are not well understood (Crawley et al., 2004), particularly in the area of software design. The benefit of a robust architecture is that the product keeps performing even as the external environment changes. Robustness may be preferred to flexibility or adaptability for a number of reasons. Robustness may be a lower cost approach because the system never needs to change. Robustness may also be preferred for situations where the range of environmental challenges is not well known. An example of a robust architecture from the defense context is the design of the Link-16 protocol, which assumed that message traffic might get lost in the dynamic airborne environment. Therefore, it built significant redundancy into its message traffic, sending positional data and other messages multiple times per second to ensure delivery. A classic example of a robust architecture is the nuclear command and control architecture. Built into mountains and underground silos, and designed to operate in a post-nuclear attack radiation environment, robustness was clearly a main design criteria.

System designs described as “robust” are more widely used than the strict definition above would allow. Some consider a robust design to be anything that copes with environmental changes and continues to perform. Robustness is also used as a synonym for survivability, to indicate continued performance when components of the system are damaged. Finally, some members of the defense community (perhaps showing some pessimism with the acquisitions process) use a robust design to mean one that actually works as designed under field conditions, using it interchangeably with “ruggedized” (Hawkes, 2013; Sherborne Sensors, 2013). To add to the confusion, Thomke’s (1997) paper, which contains excellent case studies into what we would call “robustness” in the design stage, describes the cases as “design flexibility.”

It is obvious that a robust architecture will usually be more expensive upfront than a conventional architecture. The greater span of requirements often necessitates more time preparing better designs or more cost in manufacturing, as more exotic materials are used. Therefore, as with any architectural design choice, there is a cost-benefit tradeoff for a robust design.

[4] For clarity, systems that are intended to be changed back and forth many times are usually referred to as “flexible and reconfigurable” (Ferguson et al., 2007).
[5] A simple illustration of the link between flexibility and modularity can be seen with a Google Scholar search for (“flexible and modular” or “modular and flexible”), which yields ten times more results than (“flexible and robust” or “robust and flexible”).

Open Architectures

Open architectures are becoming increasingly popular due to their prevalence, and success, in the software industry. Silver (2010) examined the history of open architectures in detail. An open architecture is one where the necessary information to design a part of the system is made accessible to the public or a wide group of possible designers. Hagan (2009) goes into more detail, defining an open system as

> a system that implements specifications maintained by an open, public consensus process for interfaces, services, and support formats, to enable properly engineered components to be utilized across a wide range of systems with minimal change, to interoperate with other components on local and remote systems, and to interact with users in a manner that facilitates portability.

The essence of these definitions is that open architectures allow any interested organizations to participate in the design and development of parts of the system. This is not a new idea, but the fact that individuals anywhere, equipped with only a computer, can contribute to open software development, combined with the increased importance of software to complex projects, has meant that the pool of potential contributors to open architectures has widened over recent decades. The benefit of an open architecture is that better solutions can sometimes be found because more organizations have the chance to examine the problem and propose design solutions. The increased competition also has the potential to lower costs.

An example from the defense context, though not yet officially sanctioned, is the growth of "Tactical iPhone apps" that have been developed both by soldiers and by small companies (Tactical Nav, 2013). These are built using the open interface exposed to applications developers by Apple.

Openness is generally well understood and difficult to confuse with any of the other terms presented here. It is important to recognize that openness is more concerned with the process of development than the attributes of the end-state of the product. However, the system architect is concerned with process as well as end-state because the development process affects affordability by changing development cost. Therefore, in developing and comparing architectural strategies, it is valid to consider strategies that affect process.

Open architectures have some significant drawbacks that are sometimes overlooked in the current enthusiasm for their use. Open architectures present coordination challenges for the government customer who must ensure that the products developed on the open market can interface to produce a usable end product. The broad dissemination of information about the end product may also present security concerns.

Common Architectures

A common architecture focuses on reuse of proven systems, or the design of platforms for later reuse (Wicht, 2011). With relatively simple systems, the key benefit is cost reduction, as much of the work from the first system is reused. Reusing systems and/or system components also decreases the development time associated with the system. With more complex systems, other benefits also become obvious: reliability increases because proven designs are reused and each part is used more often; maintenance and logistics are more affordable because there are fewer unique parts; less training is required as operators are familiar with previous instances of the product.

Examples from a DoD perspective include the Joint Strike Fighter (JSF), which was designed in three variants with as many common parts as possible.[6] Another example is the M61A1 20 mm cannon. This automatic weapon, often called the Vulcan, has been used in numerous Air Force aircraft (F-104, F-105, F-106, F-4, F-14, F-15, F-16, F-18, A-7, F-111D, and most recently the F-22), the Navy PHALANX system, and the Army C-RAM.

Commonality is straightforward in principle; however, it has significant overlap with modularity and interoperability. In a modular system, multiple common modules are often used to incrementally increase performance. The resulting system has strategies of both modularity and commonality. For example, some launch vehicles have a modular configuration with respect to the number of solid rocket boosters clustered around the vehicle core. For example, the Atlas V launch vehicle can have from zero to five solid rocket boosters attached to the core, depending on the particular payload and orbit of a launch.[7]

Modularity also makes reuse easier and therefore enables commonality at a lower cost. Software modules are the canonical example of this, because good practice software writing encapsulates particular software tasks into modules, with defined inputs and outputs. If that functionality is required in a subsequent development, the module can be easily transplanted into the new context. Commonality and interoperability are also blurred. Interoperability generally requires a common (or "standardized") interface. Therefore, the two systems are interoperable, or they share a common interface. The outcome is the same, but the terminology could be used differently. We suggest that if commonality is used solely for standardizing communications protocols for interoperability, the guiding strategy is interoperability. However, if the rationale is life-cycle cost savings from common design of terminals, hardware, or training procedures, then "commonality" is probably more appropriate.

Commonality does not always produce benefits. For example, if requirements change and a new system is required, the additional up-front investment in designing a common system is lost. In some instances, the cost of designing and enforcing a common system outweighs the life-cycle cost savings of having the common system. This may be the case with the F-35, which has had "continuing manufacturing inefficiencies, parts problems, and technical changes [that] indicate that the aircraft's design and production processes may lack the maturity needed to efficiently produce aircraft at planned rates" (GAO, 2011). The program was restructured in 2011, triggering a Nunn-McCurdy unit-cost breach. Performance is often penalized with a common system, with both systems having to share a system that suits neither of them perfectly.

Modular Architectures

To deal with complexity in systems, the idea of modularity is as old as engineering itself. Modularity allows a complex problem to be tackled in pieces. At its most basic, a modular architecture focuses on dividing the form of the system to reflect the functions of the system. This means that the system can be divided into chunks, each of which performs a distinct function. Baldwin and Clark (1999) had an elegant definition: "A module is a unit whose structural elements are powerfully connected among themselves and weakly connected to elements in other units." This design strategy tends to produce "tidier" designs

[6] "The JSF program goals are to develop and field a family of stealthy strike fighter aircraft for the Navy, Air Force, Marine Corps, and U.S. allies, with maximum commonality to minimize costs" (GAO, 2009).
[7] United Launch Alliance described the Atlas V under the heading "Modular System for Maximum Flexibility and Reliability" as using "a standard common core booster™ (CCB), up to five strap-on solid rocket boosters (SRB)" (United Launch Alliance, 2012).

with associated benefits to reliability and re-work cost. Modularity standardizes interfaces and minimizes the amount of information that needs to travel across those interfaces. More advanced modularity defines standard interfaces between aspects of the form of the system, and allows those pieces of form to be swapped out to produce different functions. This allows the product to perform across a greater range of external environments and to be upgraded more quickly and cheaply. For example, the computer and USB-peripheral architecture now used on personal computers is a modular architecture. The defined interface is the USB, and different forms with different functions can be connected to the computer via USB to improve the function of the system as a whole. A defense example of modularity is the guided bomb unit (GBU). GBUs are basically a series of modular parts, including guidance systems, ordnance, and fuses, among others, that can be assembled from the modules based on the need. Depending on the target and the desired effect, weaponeers basically build munitions from standard modular parts. There are a number of different approaches to modularity (Crawley et al., 2004), but all revolve around the same idea of neatly encapsulating product functions inside aspects of form.

Discussions about modularity usually imply that modularity is beneficial for product development; however, this is not always the case. Modularity is beneficial if it assists the product in meeting its cost and performance goals, for example through enabling commonality, flexibility, or simply neater design with less re-work. Modularity requirements can be detrimental in applications where performance, space, or weight is at a premium. In these cases, modularizing the system may introduce unacceptable performance penalties. For example, an iPhone is a tightly integrated system. The touchscreen and camera are built into the casing, and the batteries are such an integral part of the unit that they cannot be separately replaced. This allows the iPhone to be made smaller, but makes it more difficult to reuse sections of the phone from model to model. Changes to the internal design between the iPhone 4 and the iPhone 4S meant that the positions of buttons on the case needed to shift.

This tight interaction, where changes to one part of the product necessitate other changes, is typical of tightly integrated systems.[8] A second example is writing high-performance software. The use of "libraries" (pre-existing code, the software equivalent of modules) is minimized, and their functionality often re-written completely in order to optimize it for a particular application. Only the code absolutely required for the program to run is included.[9]

Modularity also shows significant interaction with other architecting strategies, particularly open architectures. This is because openness usually outsources many of the design tasks, reducing the ongoing communication between the system architects and the product design teams, and increasing the risk of integration difficulties. Modularity's emphasis on clearly defined interfaces and each module performing a single function mitigates integration risk, and therefore works well with an open architecture. An example of modularity and openness working together is the development of apps for smartphones. The apps are modular add-ons to improve the functionality of the phone and can be developed by anyone (i.e., a partially open architecture). In the defense context, modularity has been combined with open systems, which modularity enables. The result is "modular open-systems architectures." DoD Directive 5000.1 states that "acquisition programs shall be managed through the application of a systems engineering approach that optimizes total

[8] See, for example, Giffin et al. (2009), who found less change propagation through a system where "the architecture of [the] system was carefully crafted to be modular from the start."
[9] "In structured software design, functionality and data is arranged in software modules" (Chakrabati, de Alfaro, Henzinger, Jurdzinski, & Mang, 2002).

system performance and minimizes total ownership costs. A modular, open systems approach shall be employed, where feasible" (Wolfowitz, 2007). This is another example of constructive interaction between two architecting strategies.

Interoperable Architectures

Almost all systems are interoperable with some other systems because nothing works in absolute isolation. Most electronic devices are interoperable with mains power; most computers are interoperable with Internet servers; most vehicles are interoperable with the highway systems in the countries for which they were built. Exceptions to these rules exist, but only in specialized applications. When we use the word *interoperable architecture*, therefore, it is not to describe these common situations where the interoperability is implicit, but rather to describe systems where the interoperability is a key requirement of the user.

Further, interoperability is what defines systems in the sense that if there is no interaction, there is no system. If a broader perspective is taken, any product that is interoperable with another can therefore be seen as simply two parts of a single system. For example, one type of radio mounted in a ship could be described as interoperable with another type of radio mounted in an aircraft. Or, a broader system could be considered that includes both radios, in which case the interoperability is internal to the system. Therefore, simply depending on where the boundaries of the system are drawn, an interoperable architecture can refer to interoperability with systems outside the architecture or interoperability with systems internal to the architecture.

The first view of "interoperable" is used to describe architectures capable of interfacing with specified systems external to the architecture under consideration, in order to improve its functionality. This is the usual level of consideration of the architecture and is the substance of Hagan's (2009) definition of interoperability:

> The ability of systems, units, or forces to provide data, information, materiel, and services to and accept the same from other systems, units, or forces and to use the data, information, materiel, and services so exchanged to enable them to operate effectively together.

Making a system interoperable usually increases the usefulness of that architecture. For example, designing a radio handset that can use existing waveforms increases the number of other radios with which it can communicate. The ability to interoperate external to the architecture under consideration permits wider communication than developing a new, unique waveform. This would usually make a more useful product than developing a new radio in isolation.

The second way is a high-level view in which the elements of the architecture under consideration are themselves interoperable. This second view was referred to as "intra-operability" by the Open Systems Joint Task Force (2004). For example, in designing a military communications network like the Joint Tactical Radio System (JTRS), a guiding principle was that any radio on any platform running the same waveform could communicate. The JTRS architecture could therefore be described as an interoperable architecture, with the interoperation occurring within the system. To be more specific, this high-level view is often used for systems with separate physical elements that communicate information and where interoperation is not essential to the design. It would not be common to say that a set of radios designed for use by groups of infantry was an "interoperable architecture" because radios that are not interoperable with each other are generally useless. However, in the case of the JTRS, where radios on aircraft could interface with radios on ships and in the hands of infantry, this was an unprecedented degree of interoperation that was central to the JTRS project. Therefore, describing the JTRS as an "interoperable architecture" adds information about the central design strategy.

Interoperable architectures also present some disadvantages. The increased cost and complexity of design involved in making an architecture interoperable should not be overlooked. In particular, interoperable architectures are difficult to test because the boundaries of the system under test are often unclear or difficult to simulate in real-world conditions.

A common issue with implementing interoperability is different implementation of the standard. An example is the Link-16 standard message set, which has been implemented differently across various systems, resulting in suboptimal interoperability.

Extensible Architectures

An extensible architecture is one that makes provision for additional elements to be added in the future. In contrast to a flexible architecture, where the guiding strategy involves addition and removal as needed, an extensible architecture generally contemplates permanent additions.[10] A striking example of extensibility is the practice of constructing future on-off ramps at the time of construction of highway overpasses. These "ramps to nowhere," which extend only a short distance out from the main bridge, minimize the cost and disruption of traffic if another road needs to be connected to the overpass at a future time.

In a DoD context, an example of an extensible architecture is the F-15E Strike Eagle, which, when it was built, was built and architected to support four radios but was initially fitted only with two. However, the space, the physical interface, and the interface with the Operational Flight Program (the software) were all developed and built in at the start. One of the two remaining slots has been subsequently filled. The disadvantages to the extensible architecture are primarily the additional up-front expense and time of building in the extensibility. The extensibility offers an easy target for scope reduction under cost or schedule pressure. Extensibility can also be difficult to test in complex systems because the elements to be extended are often not created; therefore, testing the interface under realistic conditions is difficult. When the government, not the contractor, is the ultimate beneficiary of the cost savings of a well-engineered extensible solution, there is little incentive apart from compliance testing to ensure that the extensibility is done well.

Extensibility has a relatively clear definition. It can be distinguished from flexibility through the permanence of the extensible addition. It can be distinguished from interoperability because at the time the extensible system is created, the system it will interoperate with is not yet created. Note that extensibility is very similar to scalability, and the two are often interchanged. Two criteria to distinguish the terms are proposed here based on our reading of the nuance in usage between the two, but these are by no means hard rules. First, *extensibility* usually refers to a bounded addition, where *scalability* usually refers to arbitrarily large increases in quantity. For example, extensibility could be used in the context of adding a second story to a building or an additional lane to a freeway. Scalability is more commonly used in information systems when unbounded increases in quantity are more feasible. For example, in a computer network architecture, a scalable system indicates the ability to add on more nodes arbitrarily. Secondly, scalability also

[10] No satisfying formal definitions of *extensibility* could be found in the literature, presumably because the term was widely used and understood. Wikipedia states, without citation in its entry on extensibility, that "in systems architecture, extensibility means the system is designed to include hooks and mechanisms for expanding/enhancing the system with anticipated capabilities without having to make major changes to the system infrastructure."

connotes additions that are similar or common to what already exists, where extensibility could include provision for something different. For example, an architecture that envisaged adding a garage to the side of a house might easily be described as extensible but less comfortably as scalable. The ability to duplicate an existing garage would be easier to describe as scalable but could probably also be described as extensible.

Summary of Engineering Literature Definitions

The definitions suggested previously are summarized in Table 1, highlighting the engineering focus of the design strategy, as well as some of the confusing overlaps of the terminology used to describe the end result. Note that the end goal is always to deliver the desired performance at required costs, and the architectural strategies should be considered as a range of tools to achieve that end.

Table 1. Architectural Strategies

Architectural Strategy	Main Focus	Major Benefits	Major Drawbacks
Common	Parts, rather than interfaces	Increased life-cycle affordability Manufacturability Reliability	Higher upfront costs Sub-optimal performance
Modular	Interfaces (designed, minimized, and standardized) One-to-one mapping of function to form	Leads to scalability Leads to flexibility Leads to commonality	Sub-optimal performance Added weight (in some cases)
Adaptable	Changes itself based on variations in the environment	More affordable than developing different products May improve survivability, reliability, or other performance characteristics	Requires well-defined requirements May require over-design, generic components, extra interfaces
Flexible	Gets changed by people in reaction to changes in environment	More affordable than developing different products May improve survivability, reliability, or other performance characteristics	Requires well-defined requirements May require over-design, generic components, extra interfaces
Robust	Continues to deliver performance despite substantial variations in environment	More affordable than developing different products May improve survivability, reliability, or other performance characteristics	Usually more expensive Lower performance
Interoperable	Standardizes interfaces	Improves performance May improve affordability through reuse of existing network infrastructure	Effort to correctly interface with existing systems Perpetuation of legacy standards
Open	Necessary design information made public	Encourages innovation that may improve affordability or performance Encourages competition, which may improve affordability or performance	Loss of design control, intellectual property, and project influence by customer
Extensible	Provisions made for future permanent additions	Improves affordability, assuming the extension is used	Higher upfront costs Difficult to test in development

Selecting and Aligning Architectural Strategies With Acquisition Goals

Equipped with a better understanding of the architectural strategies, we return to the challenge of how the acquisition community can make sense out of these terms and best apply an acquisition strategy to achieve the desired end state. The premise of this section is that some acquisitions are better suited to some architectural strategies.

Against the backdrop of several acquisition reform efforts, including the Weapon Systems Reform Act of 2009, the 2008 reissuance of DoD Instruction 5000.02 and the Under Secretary of Defense for Acquisition, Technology, and Logistics (USD[AT&L]) “Better Buying Power” memorandum (Carter, 2010), understanding the interrelation between acquisition strategies and architecting strategies becomes increasingly critical. These reforms place an increased emphasis on the systems engineering phase, as well as focus on cost performance throughout a program’s life cycle (GAO, 2012). The Weapon Systems Reform Act of 2009, in particular, places an emphasis on competition throughout the program life cycle (GAO, 2012). The result of these efforts is that more time and money is being spent prior to system development or production, and more emphasis is being placed on competition at all phases, to reduce cost.

Each acquisition is unique. In attempting to give broad guidance to the acquisition community, this paper focuses on two variables that change how acquisitions are conducted and which architectural strategies may be most appropriate:[11] First, the degree to which requirements and environment change from the initial planning to the field-conditions of the system; and second, the number of contractors separately involved in delivering the end system. We consider a contractor separately involved in the acquisition if it is directly responsible to the government customer, rather than acting as a subcontractor. Multiple contractors may be introduced because the system under consideration is too large (in terms of cost or complexity) to give to a single company or to increase competition in the procurement process. Deputy Secretary Carter (2010) has already made the point that he wants increased involvement by a larger number of firms under the theory that it lowers costs, increases buying flexibility, increases the strength of the industrial base, and leads to company-driven innovation (in support of competition). The Weapon Systems Reform Act of 2009 requires the use of competitive prototypes prior to systems development to be a part of the acquisition strategy (GAO, 2012).

These two variables lead us to ask the following two questions for each of the Eight Strategies:

1. If this architectural strategy is used, how flexible can the procurement be to changes in the anticipated operating environment and/or requirements?
2. If this architectural strategy is used, how difficult is it to involve multiple, separate companies?

The results of asking these questions are presented in Figure 1.

[11] Of course, other variables may also affect the choice of the architecting strategy, for example, the remuneration structure of a contract (choosing from fixed-fee, cost-plus, and incentive-fee, among others). The two variables we chose are not as well controlled by government than many other factors that affect acquisitions; therefore, the architectural approach needs to be tailored to the acquisition variables, rather than the acquisition variables being tailored to the architectural approach. For detailed examples on how acquisition variables could be tailored, assuming a commonality approach was taken (Wicht, 2011).

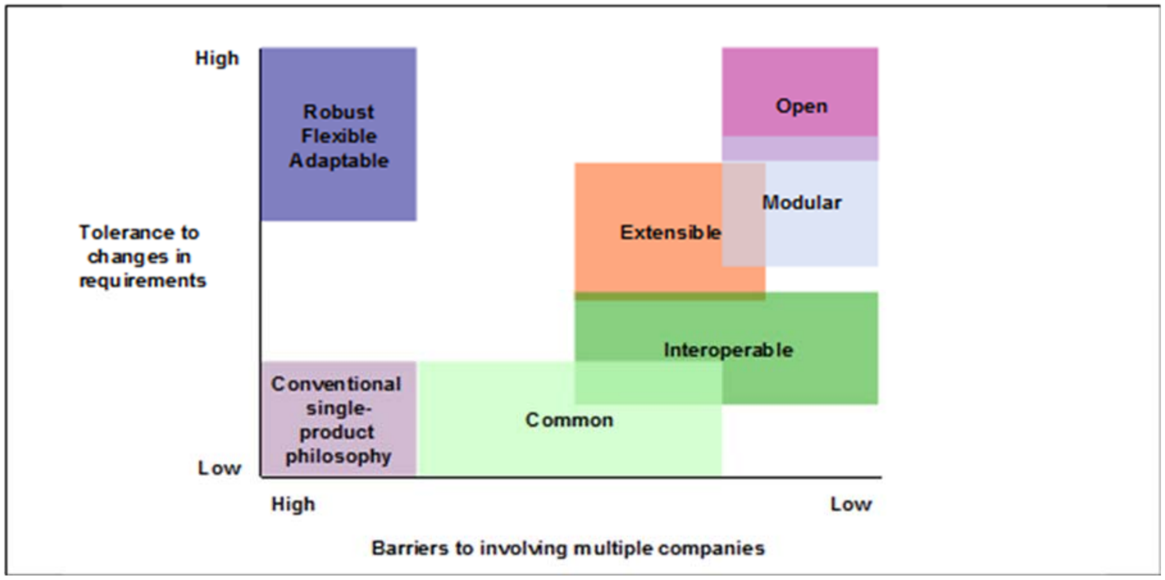

Figure 1. Architecting Strategies in Context

Figure 1 shows that, depending on where across the spectrum a given procurement falls, there are generally multiple architecture options that will achieve a good result, but there are also architectural approaches that are not well suited. The proposed framework described previously is offered as a starting point for identifying potential architecting strategies based on where on the spectrum a given acquisition is likely to fall. The architecting strategy is directly tied to cost-benefit trades for the product, and as a smart buyer and/or as a systems architect, the government must be aware of these architectural considerations. A detailed rationale for the position of each entry on the chart in Figure 1 follows.

Conventional, Single-Product Design Strategy. Conventional, single-product design strategy describes a conventional single product, single contractor development process where the government specifies the requirements up front and a single contractor produces the product. It has low tolerance to changes in the initial requirements because the contractor has no incentive to design outside the requirements given. There are no defined interfaces at the government-contractor level, which makes simultaneous competition difficult. The intellectual property usually rests with the contractor, which makes competition over time difficult. This is the paradigm that the DoD is attempting to leave, but it has a place in acquisition. For some small, non-complex procurement, it might be the right strategy.

Common Strategy. Common design makes it a little easier to introduce multiple companies, for example, because a government furnished equipment (GFE) process can be used across the common elements of the architecture. One company supplies the equipment, and another uses it in the systems it is developing. However, the common design is "locked-in," making it very intolerant to changes in requirements. Any changes need to be cascaded through two contract mechanisms, between the government and the GFE supplier, and the contractor building the current system. This increases time and cost.

Interoperable Strategy. The interoperable architecture strategy is intended to allow multiple different products to interoperate. Therefore, it is helpful for lowering barriers to involving multiple companies. However, the interoperable standard needs to be defined at the outset because it defines what aspects of the system must be the same in order to have interoperability. The standard is effectively common and brings the inflexibility, which is both the strength and the limitation of a standard. The standard is very difficult to update as requirements change, and systems usually ignore the standard and break the chain of interoperability in cases of significant requirements change. Note that changing the standard is not the same as changing other aspects of the elements that interoperate. For example, an aircraft may be upgraded to fly further in response to evolving requirements, but so long as the communication system remains unchanged, the interoperability will remain.

Robust, Flexible, or Adaptable Strategy. These design strategies evolved to allow systems to meet changing requirements, even if the requirements are not known at the time they are developed. Therefore, the strategies score high on the changing requirements axis. However, the principles that are used to evaluate robust, flexible, or adaptable approaches must be applied to the system as a whole, using rigorous system engineering techniques across the entire end product. This makes it difficult to fragment the system and use multiple companies.

Extensible Strategy. An extensible architecture builds in allowances for changes in requirements. However, the changes need to be anticipated at the outset in a way that, for example, a flexible architecture does not. It is difficult to build an extensible architecture without an idea of what will be extended. However, building a flexible architecture, such as a software-defined radio, allows decisions to be made about the changes once the new requirements are better known, for example, writing new software. Whether an architecture is extensible does not appear to have a significant effect on the involvement of different companies in the development of the architecture. Arguably, it makes it slightly easier to include additional companies if the extension can be “re-competed.” However, in many cases, the degree of knowledge of the original contractor about the system makes it difficult for new contractors to be competitive.

Modular Strategy. A modular architecture minimizes the interfaces between parts of a product or system and groups functional areas together. Therefore, a modular architecture is more suitable for the involvement of multiple companies because of the ease of partitioning work packages. A modular architecture also allows aspects of the architecture to be changed out, if necessary, without redesigning the whole system, which makes it reasonably tolerant to changes in requirements.

Open Strategy. An open architecture has low barriers to involving multiple companies. There are fewer intellectual property barriers, and companies are free to submit bids for pieces of work. An open architecture is ideally changed quickly as requirements change because there is a short development cycle due to competition and a minimum of formal requirements. It should be noted that open architectures are heavily dependent on agreed standards to manage the interfaces between the open development and other parts of the system. If the requirement changes necessitate changes in the interfaces and standards, then the benefits of openness to dealing with the requirements change are lost.

The previous analysis suggests that architecting strategies that are chosen largely on “hard engineering” concerns actually have implications for the cost and other programmatics of the project, and the architecting community needs to start coming to grips with which architectures are most useful in which situations. No one acquisitions approach can be universally applied to all architecting strategies. The architecting strategies suit different acquisition scenarios, and therefore, much thought should go into which type of architecting strategy is appropriate for each acquisition. However, due to the overlap of some architecting strategies, more than one strategy may be successful for a given acquisition. Figure 1 highlights where those architectures are more likely to be successful choices. This underscores Maier and Rechtin's (2009) central thesis that "engineering is more of a science, and architecting is more of an art."

Summary

The terms we have called architecting strategies in this paper—commonality, interoperability, modularity, flexibility, extensibility, robustness, adaptability, and modularity—have all been used at various times as preferred solutions for reducing the cost and schedule of government acquisitions of complex systems.

There has not been a wide and consistent understanding of the full meaning of these terms across the acquisition community. In order to use these terms to communicate approaches and strategies, all personnel involved must share a common understanding of the terminology. Sections I and II of this paper attempted a first step in this direction by surveying the literature and engineering practice to arrive at definitions, strengths, and weaknesses for the architecting strategies. Even with a common understanding of the strategies, a second danger presents itself. That danger lies in a belief that particular architecting strategies are the solution for all acquisitions. In fact, as Section III of this paper showed, some architecting strategies are better suited to particular acquisition scenarios than others. Understanding the interconnections between the architecting strategies and acquisition scenarios is essential to making the right decisions at project initiation. The importance of getting the architecting strategy right, through good communication of ideas and solid understanding of these interconnections, cannot be overemphasized. As Robert Spinrad said, "In architecting … all the serious mistakes are made on the first day" (Maier & Rechtin, 2009). Spinrad was talking about software, but the apothegm applies equally to other forms of complex systems. Better communication and understanding of terminology cannot eliminate mistakes altogether, but they represent a good first step.

References

Baldwin, C., & Clark, K. (1999). *Design rules: The power of modularity*. Cambridge, MA: Massachusetts Institute of Technology.

Berteau, D., Ben-Ari, G., Hofbauer, J., Sanders, G., Ellman, J., & Morrow, D. (2010, April 20). *Cost and time overruns for major defense acquisition programs*. Washington, DC: Center for Strategic and International Studies.

Carter, A. (2009, July 29). *Materiel interoperability and standardization with allies and coalition partners* (DOD Instruction 2010.06). Washington, DC: OUSD(AT&L).

Carter, A. (2010, September 14). *Memorandum for acquisition professionals: Better buying power: Guidance for obtaining greater efficiency and productivity in defense spending*. Washington, DC: OUSD(AT&L).

Chakrabati, A., de Alfaro, L., Henzinger, T., Jurdzinski, M., & Mang, F. (2002). *Interface compatibility checking for software modules* (Vol. 2404/2002).

Crawley, E., de Weck, O., Eppinger, S., Magee, C., Moses, J., Seering, W., … Whitney, D. (2004, March 29). *The influence of architecture in engineering systems*. MIT Engineering Systems Division Symposium.

Defense Acquisition University (DAU). (2013). Terms & definitions. *Acquisition Community Connection.* Retrieved from https://acc.dau.mil/CommunityBrowser.aspx?id=22108

deWeck, O., Ross, A., & Rhodes, D. (2012). *Investigating relationships and semantic sets amongst system lifecycle properties (Ilities)*. Paper presented at the Third International Conference on Engineering Systems, TU Delft, the Netherlands.

DoD. (2008, December 8). *Operation of the defense acquisition system* (DoD Instruction 5000.02). Washington, DC: Author.

DoD. (2013). F-35. Retrieved from http://www.jsf.mil/f35/f35_technology.htm

Dickerson, C., & Mavris, D. (2010). *Architecture and principles of systems engineering*. Auerbach Publications.

Ferguson, S., Siddiqi, A., Lewis, K., & de Weck, O. (2007). Flexible and reconfigurable systems: Nomenclature and review. *Aerospace Engineering*, 1–15.

Fricke, E., & Schultz, A. (2005). *Design for changeability (DfC): Principles to enable changes in systems throughout their entire lifecycle, 8*(4), 342–358.

Giffin, M., de Weck, O., Bounova, G., Keller, R., Eckert, C., & Clarkson, P. (2009, August). Change propagation analysis in complex technical systems. *Journal of Mechanical Design, 131*.

GAO. (2009, March). *Defense acquisitions: Assessments of selected weapons programs* (GAO-09-326SP). Washington, DC: Author.

GAO. (2011, March). *Defense acquisitions: Assessments of selected weapon programs* (GAO-11-233SP). Washington, DC: Author.

GAO. (2012, March). *Defense acquisitions: Assessments of selected weapon programs* (GAO-12-400SP). Washington, DC: Author.

Hagan, G. (2009, November). *Glossary of defense acquisition acronyms and terms*. Defense Acquisition University.

Held, T., Lewis, M., & Newsome, B. (2007). *Speaking with a commonality language: A lexicon for system and component development*. Santa Monica, CA: RAND Arroyo Center.

Held, T., Newsome, B., & Lewis, M. (2008). *Commonality in military equipment: A framework to improve acquisition decisions*. Santa Monica, CA: RAND.

Joint Chiefs of Staff. (2010, November 8). *Department of Defense dictionary of military and associated terms* (Joint Publication 1-02). Washington, DC: Author.

Maier, M., & Rechtin, E. (2009). *The art of systems architecting* (3rd ed.).

Moore, J. (2011, March 31). DoD acquisition official: Cost over-runs “intractable” problem. *Government Executive*. Retrieved from http://www.executivegov.com/2011/03/dod-acquisition-official-cost-over-runs-%e2%80%98intractable%e2%80%99-problem/

Open Systems Joint Task Force. (2004, September). *Program manager’s guide: A modular open systems approach (MOSA) to acquisition* (Version 2.0).

Peters, K. (2009, March 31). GAO: Staggering cost overruns dwarf modest improvements in Defense acquisition. *Government Executive.* Retrieved from http://www.govexec.com/defense/2009/03/gao-staggering-cost-overruns-dwarf-modest-improvements-in-defense-acquisition /28872/

Sherborne Sensors. (2013). LSOC/P ‘L’ single axis rugged servo inclinometer, ±1° to ±90°. *Sherborne Sensors*. Retrieved from http://www.sherbornesensors.com/ international/products/view/LSOC-P-L-Single-Axis-Rugged-Servo-Inclinometer

Silver, M. (2010). *Open collaborative system design: A strategic framework with application to system biology*. Cambridge, MA: Massachusetts Institute of Technology.

Tactical Nav. (2013). Army captain builds iPhone app for soldiers in Afghanistan. Retrieved from http://www.tacticalnav.com/?press=army-captain-builds-iphone-app-soldiers-afghanistan

Thomke, S. (1997). The role of flexibility in the development of new products: An empirical study. *Research Policy, 26*, 105.

Ulrich, K., & Eppinger, S. (2008). *Product design and development* (4th ed.). Boston, MA: McGraw-Hill Higher Education.

United Launch Alliance. (2012). Atlas V Modular System for maximum flexibility and reliability. Retrieved from http://www.ulalaunch.com/site/pages/ Products_AtlasV.shtml

Wicht, A. (2011). *Acquisition strategies for commonality across complex aerospace systems-of-systems*. Cambridge, MA: Massachusetts Institute of Technology.

Wolfowitz, P. (2007, November 20). *The defense acquisition system* (DoD Directive 5000.01). Washington, DC: DoD.