A COMMONALITY TOOLKIT FOCUSED ON NASA APPLICATIONS

prepared by the MIT Space Systems Architecture Group

August 2010

Professor Ed Crawley

Ryan Boas
Bruce Cameron
Wilfried Hofstetter
Richard Rhodes
Anthony Wicht

Comments should be directed to
Bruce Cameron (bcameron@mit.edu)

PLANNING BETTER PROJECTS: THE EFFECT OF COMMONALITY

Commonality is a strategy that can be used during system architecting. Commonality reuses identical design elements like systems and components in multiple places throughout the architecture. Treating commonality as a strategy allows system architects to plan appropriately from the outset to maximize the benefit from reuse and the likelihood of the reuse occurring. Treating commonality as a strategy also identifies the circumstances where independent development is a better strategy than commonality.

Properly implemented, a commonality strategy can lower cost and risk over the entire product lifecycle. Commonality as a successful product strategy has proven its effectiveness in a range of industries from automotives to electronics. This toolkit illustrates its applicability to space-focused projects.

Consider for a moment the alternative to commonality: developing a series of products which meet requirements tailor-made for each one. Inevitably, some aspects of early designs will be reused for later products by risk-averse engineers. Commonality occurs naturally when its benefits are obvious. Developing a systematic process to assess the full lifecycle impact of potential commonality across a project allows system engineers and managers to extend commonality benefits to less obvious, but equally helpful, cases throughout product development.

This toolkit concentrates on planning for commonality. That is, designing products with an entire product family in mind, and with conscious thought as to whether sections of the product should be designed for reuse in later products in the family. This planning process is a balancing act between the sometimes complex advantages and disadvantages of commonality.

The toolkit is intended as a practical guide, and in the interests of clarity it does not discuss many of the aspects of commonality which we would describe as "possible, but unlikely". The references and further reading suggested for each of the chapters supply much of this extra detail and can be read thoroughly if needed.

The focus of the following chapters is on NASA cases and the examples and illustrations are space related. Through this toolkit, we seek to show that NASA flagship projects like launch vehicles and exploration architectures can benefit from the same considerations of product family design that lead to efficient design in more commercial fields. We also show the key management approaches that are required in the NASA context to make sure that technically feasible commonality opportunities are successfully and economically implemented.

The learning in this toolkit draws from ten cases studied at MIT over the past four years, which included four aerospace cases and three additional NASA cases. The aerospace cases were drawn from the F-35 Joint Strike Fighter, a commercial aircraft manufacturer, a business jet manufacturer and a communications satellite manufacturer. The NASA cases were the International Standard Payload Rack flown on the ISS, the Constellation Space Suit System and the Core Flight Software System developed at Goddard Space Flight Center. The toolkit also draws on a separate detailed study of commonality in NASA ground infrastructure.

In our experience in working on NASA case studies, almost all design teams are aware of the benefits of commonality, and make some attempt to consider commonality during design as a routine part of good engineering. However, these early efforts usually don't address all possible opportunities for commonality for a range of reasons. Even once identified, commonality may not be implemented in the final products because of an absence of incentives to pursue commonality, and an inability to prove the future potential benefits of commonality against immediate and obvious penalties to common development. This toolkit is designed to arm system architects, design teams and project managers with the information about commonality that they need in order to make the right strategic decisions about commonality.

COMMONALITY IN THE NASA CONTEXT

Much of the attention given to commonality has focused on its application to relatively small, simple, mass produced goods. Computer printers, automotives and furniture are some of the famous (and infamous) illustrations of commonality built on product platforms.

There is a temptation to think as a result that commonality does not apply to NASA systems. In our view this is a mistake for at least four reasons. Many of the benefits of commonality are most pronounced when building only a small number of systems. For example, the incremental learning benefit in moving from a third to a fourth identical launch vehicle engine is much greater than that in moving from the three hundredth to the three hundred and first.

Some commonality benefits are especially relevant to NASA. The decreased logistics cost associated with using common spares has a disproportionately high benefit to NASA as no other industry has such extreme shipping costs for its spares. The reduced operational risk from common operating procedures is an important safety consideration for human spaceflight.

NASA has a strong and interactive relationship with its contractors and suppliers. This gives NASA the opportunity to implement commonality throughout its supply chain.

Finally, the scope of potential future projects where NASA might implement commonality is wide. Figure 1 illustrates just some of the potential applications of commonality in future exploration architectures. Each appropriate instance of commonality, if well implemented, reduces total development time, risk and cost while still preserving acceptable performance.

Commonality has been a design strategy for NASA from the time of the Apollo program, when Saturn launch vehicles shared common engines and the lunar lander and the command and service module had very similar guidance computers. NASA recognized the importance of commonality in reusing ground infrastructure at Kennedy Space Center between the Apollo program and the Shuttle program. Most recently, the Constellation program plans to use common avionics and engines between the Ares I and Ares V launch vehicles. This toolkit aims to consolidate tools developed at MIT and elsewhere to help NASA continue using commonality in a sophisticated manner.

More specifically, the toolkit is designed to improve the understanding of *how* to make commonality happen within NASA. Through the course of our work with NASA on commonality, the overriding theme has been an acceptance of the benefits of commonality, hand in hand with a difficulty in finding effective techniques for realizing commonality. Perhaps the best encapsulation of this attitude is the 1969 report from President Nixon's Space Task Group which reported *"Commonality, Reusability and Economy"* as the three prongs on which more economical space exploration should take place: the emphasis on commonality was present from the start, but the realization of commonality probably fell short of initial goals.

During our studies, reasons like the following were common:

- "I can't identify technically feasible opportunities for commonality early enough in the design process to implement commonality – by the time I realize what we could have done, the opportunity is gone."
- "I already look for commonality, with reasonable results, and I don't know how to improve the process."
- "I can't identify whether it will be cost-effective to try to design for commonality in my case."
- "Even when I've identified beneficial commonality, how do I get my team to actually implement it?"

The aim of the toolkit is to provide answers to these questions.

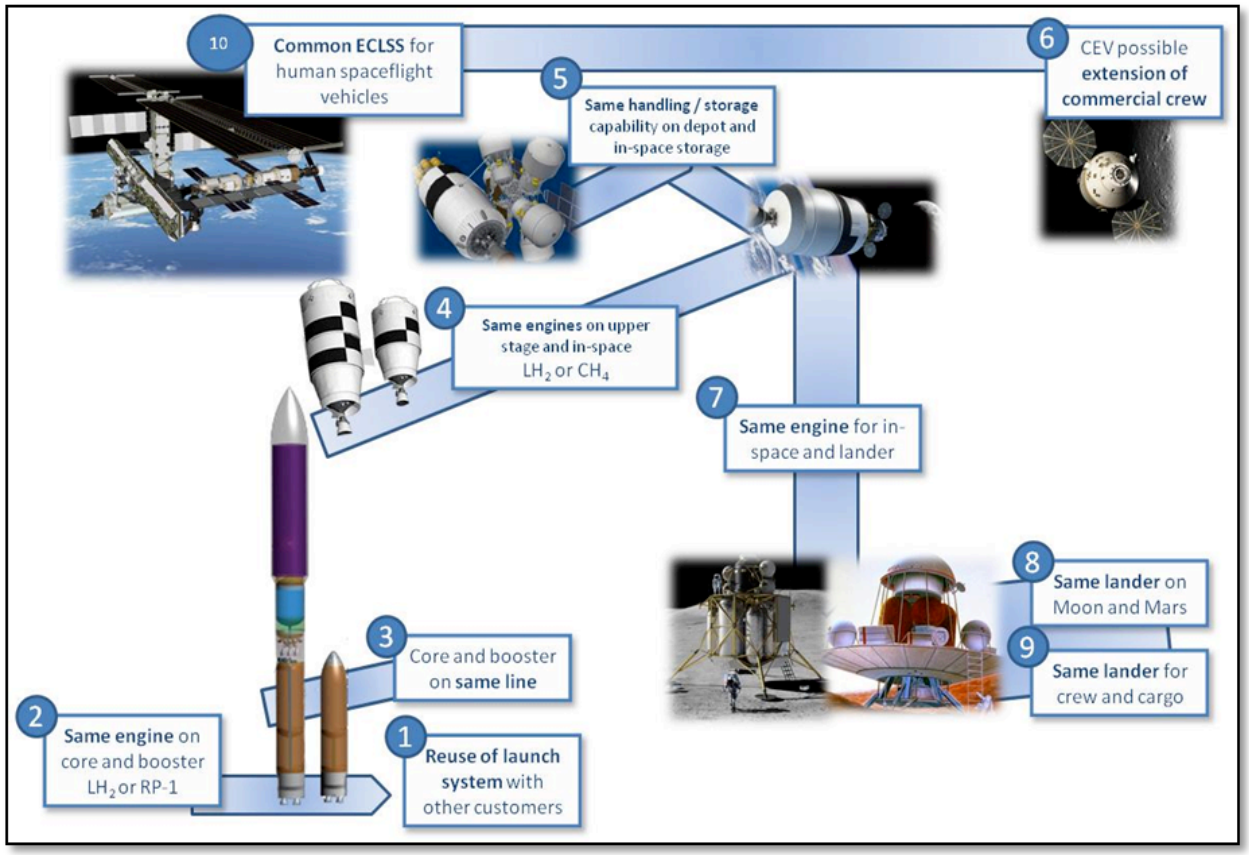

FIGURE 1: POTENTIAL AREAS OF COMMONALITY IN NASA'S HUMAN EXPLORATION PROGRAM

CONTENTS

TOOLKIT OUTLINE

This toolkit is divided into four parts. The initial section examines the basic features of commonality, including its benefits and drawbacks, how it can be measured and how it interacts with other design strategies like modularity and flexibility. It also illustrates some of the obstacles to commonality that were encountered in the case studies we examined. The toolkit then goes on to present strategies, tools and heuristics for solving those commonality obstacles. Those solution tools are broken up into three types:

- Tools to help **identify** potential technically feasible opportunities for commonality at the system architecting level
- Tools to help **evaluate** whether the technically feasible commonality is better than independent development
- Strategies to help **manage** projects with commonality, both by encouraging the identification of commonality and by maintaining optimum commonality as the project progresses

Chapter 1 presents a review of the major ideas behind commonality, since these are used for all the subsequent chapters. Chapters 2 to 4 then go into detail on tools and strategies. Figure 2 outlines the major tools and strategies presented.

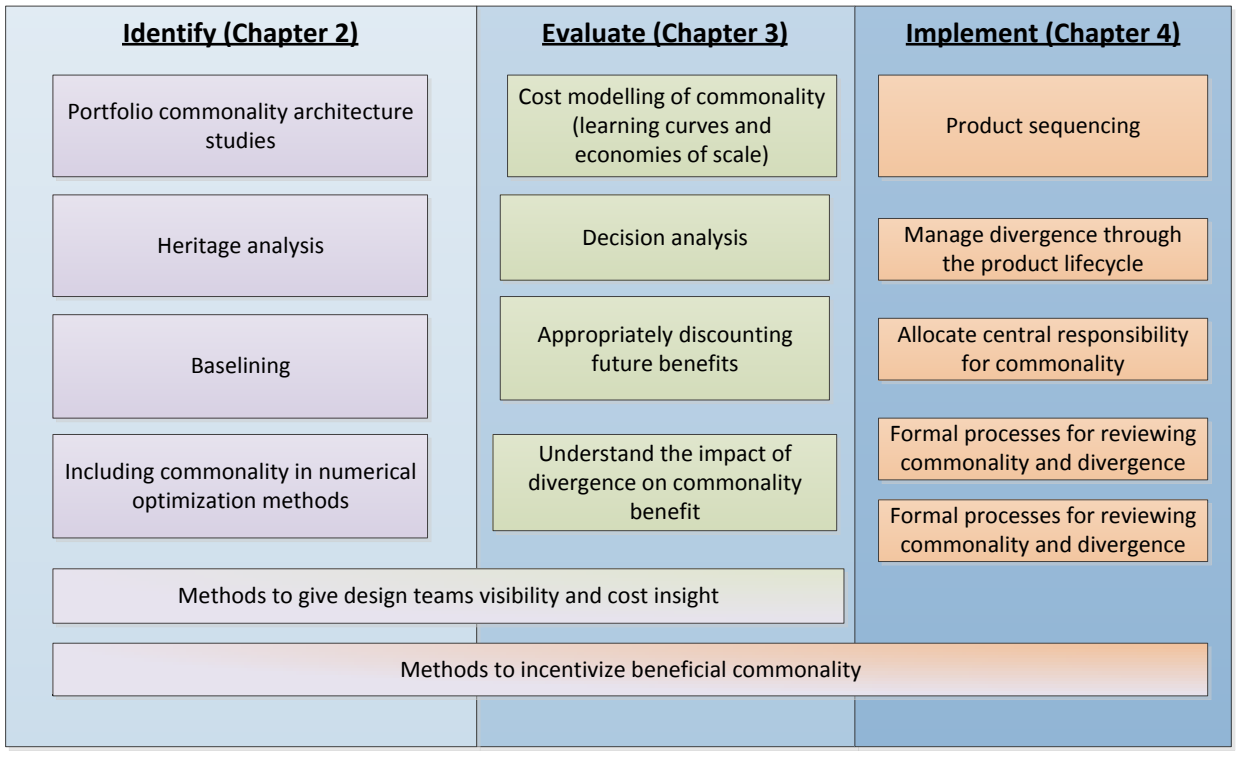

FIGURE 2: TOOLKIT OUTLINE

TOOLKIT TERMINOLOGY

The terminology surrounding commonality can be confusing. Many of the terms used to describe commonality are defined in different ways by different writers. For clarity and consistency throughout this toolkit, we have shared as much terminology as possible with the NASA Systems Engineering Handbook.

We use the following hierarchy when describing engineering programs:

Program: the highest level development effort, for example the Apollo program. It contains several projects which work in concert to produce the program outcomes. It is unusual to have a family of variants at this level.

Project: a self-contained development effort, for example, the Saturn V launch vehicle project. Often there will be a family of variants at the project level. Interoperable projects form a program. A particular output from a project is a **product** – each Saturn V would be a product. Commonality can occur at the project level, when a single product is used to meet two different sets of needs. For example, if an identical launch vehicle were used to service LEO and GEO markets, there would be commonality at the project level.

System: a system is a complex engineering development which interacts with other systems to produce a project. Systems are grouped by function. For example the Saturn V second stage propulsion assembly would be a system. Systems are usually the highest level at which commonality occurs.

Component: the smallest useful part of an engineering design, for example a bolt.

A **part** or an **element** is not level-specific. It merely means some smaller portion of any of these levels.

A **product family** or **portfolio** is a collection of different products which share common systems (or sometimes components) but which serve different needs. The different products are known as **variants**. The common systems are known as a **platform**. *For example, the Delta IV product family has medium, medium plus, and heavy variants which share a common main engine platform. See Figure 5.*

Descriptions of other commonality terms can be found in the glossary. A good reference for more detail about the terminology of commonality is the paper produced by RAND Corporation (Held et al. 2007).

CHAPTER 1: INTRODUCTION TO COMMONALITY CONCEPTS

1.1. WHAT IS COMMONALITY?

The obvious starting point in learning about commonality is to ask what we mean by commonality.

Commonality is a deliberate design strategy that aims to use identical parts for slightly different purposes, where appropriate. Those parts could be entire systems, they could be sub-systems or they could be components. In some cases, the parts could be the entire end product. The framing of the commonality proposition is deceptively simple. Deciding "where appropriate" occupies much of the rest of this toolkit.

1.2. THE POTENTIAL BENEFITS OF COMMONALITY

An analysis of commonality usually begins with the potential upside from using common systems. Properly implemented, commonality could deliver some or all of the benefits shown in Table 1.

TABLE 1: COMMONALITY BENEFITS

Benefit	Reason
Cost	
reduced development cost	commonality can reduce the number of separate development projects
reduced manufacturing cost	economies of scale in component procurement; reduced variation in the tooling and manufacturing equipment required for the product family spreads fixed, non-recurring capital costs
reduced operating cost	sharing operating resources like ground facilities or maintenance plants spreads fixed recurring costs over more products; variable recurring costs are reduced through economies of scale and learning; common products share common spare parts, reducing spares and logistics costs
reduced product volume risk	common components can be allocated to any of the variants, reducing the loss associated with a single variant performing poorly
Schedule	
reduced development time over the whole product family	in design, reuse of well understood elements allows faster design; in production, manufacturing multiple identical elements becomes faster due to learning
Cost and schedule	
reduced development risk over the whole product family	reuse of proven technologies; fewer separate development projects
reduced product testing and evaluation for later products	reuse of testing infrastructure; reuse of results from earlier identical products
reduced training time	common interfaces or operational procedures reduce operator training time
Product performance	
reduced operational risk	common products can be operated in common ways, reducing operational complexity and operator error
increased survivability	commonality improves the chances of success in using non-intended spares to replace critical parts in emergency situations

These benefits do not come free, however. Commonality has downsides also.

1.3. DRAWBACKS TO USING COMMONALITY

Table 2 presents some of the negative aspects of a commonality strategy. Many of the drawbacks to commonality are intrinsic to the strategy itself and are present to some degree even if commonality is properly identified and managed. Poor implementation and management can magnify these drawbacks or mute the positive aspects of commonality. Management methods for commonality are discussed in Chapter 4.

Therefore commonality has complex effects on the performance, risk and cost associated with a project. All three aspects should be considered when evaluating commonality as a design strategy.

The interplay between the upside and downside of commonality is shown graphically in Figure 3.

TABLE 2: COMMONALITY DRAWBACKS

Drawback	Reason
Cost	
Increased cost and risk for initial production	engineering a part to be common across multiple products often makes the part more complex
Increased cost for the initial product if future variants do not appear	commonality benefits are often based on sharing with future variants; if these future variants do not appear then the added cost and complexity of making the parts common is wasted
Cost and Schedule	
Increased cost and time for the design and manufacture of the first unit	the process of identifying and managing commonality increases the cost of producing the first unit
Increased risk of obsolescence	technology or market changes may make the common component obsolete, necessitating an upgrade of the entire product family
Product performance	
Sub-optimal performance	common parts imply either overperformance in one variant or underperformance in another variant, because the part is not optimized for the particular use case
Increased performance risk	a single design failure will affect more products if it occurs in the common part
Increased risk during operation	common parts may be more complex and therefore more prone to failure

EFFECTS OF COMMONALITY THROUGH THE PRODUCT DEVELOPMENT PROCESS

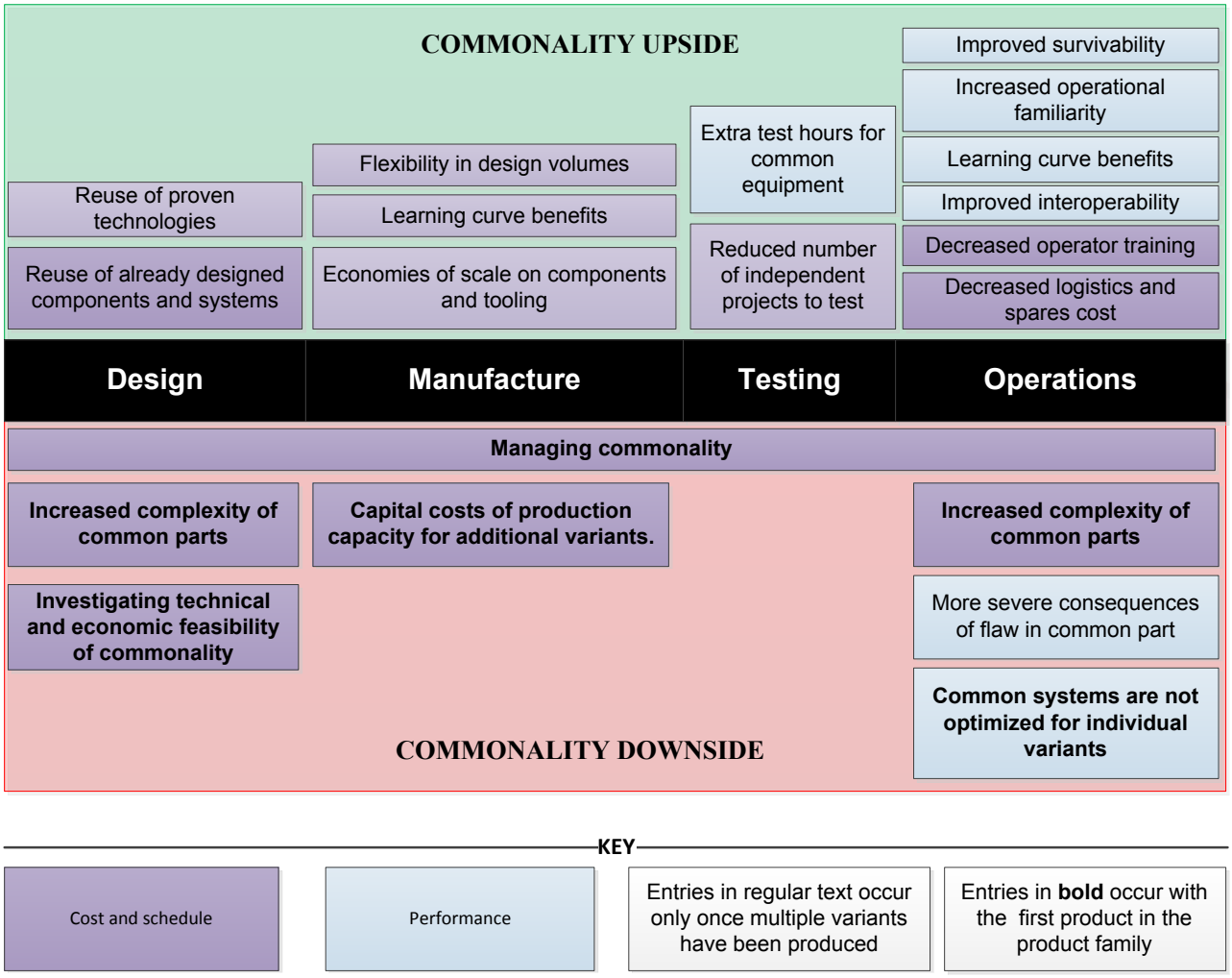

FIGURE 3: BENEFITS AND DRAWBACKS TO COMMONALITY THROUGH THE PRODUCT DEVELOPMENT PROCESS

Commonality is not always the right solution – but it should only be discounted when a thorough analysis shows that over the product lifecycle its disadvantages are likely to outweigh its advantages. It should not be discounted simply because it is more difficult to manage, more challenging initially or more poorly understood than conventional, independent design. The reverse is also true: commonality should not be pursued solely because it sounds good or it looks like an attractive solution.

1.4. WHAT IS THE GOAL OF COMMON DESIGN?

The goal of common design in this paper is to **reduce the total lifecycle cost of product families, while still meeting adequate performance targets and acceptable levels of risk**.

The methods in this toolkit would be less relevant to products which aim for performance at any price. Commonality always involves a performance trade on the product level, for a

better overall balance between cost, schedule and performance at the program level.

1.5. DIVERGENCE

It is critical to note that commonality is not an end in itself. It is a tool for building cheaper, more reliable products. Products with 80% commonality are not necessarily better products than those with 60% commonality. There are good reasons for reducing commonality between products, including to take advantage of new technologies, to incorporate learning into subsequent products, and to adapt to changing requirements. In those cases, the decrease from the initially planned commonality, known as "divergence", will produce better products. Tools to help choose when divergence is the best option are presented in Chapter 3. Strategies to manage divergence can be found in Chapter 4. When considering whether a reason for divergence is sufficiently strong to permit the designs to diverge, it is important to work through the same set of tradeoffs as were considered when making the initial commonality decision (ie those illustrated in Figure 3).

There are two things about divergence which are clear. One, divergence always increases (in other words commonality decreases) as product development proceeds. This is illustrated in Figure 4.

Commonality is a strategy for developing better products. Commonality is not an objective in itself.

Two, clinging to commonality as an absolute performance metric is a poor strategy. In the words of Senator McClennan, investigating the TFX Contract in 1970, *"As a result of not having sacrificed commonality when it was necessary to do so, you have a failure in one of the planes, a complete failure…"* For this reason, the toolkit does not go into detail on methods for quantifying the amount of commonality in a system. If this is required there are some summary papers available on the subject, for example (Simpson & Thevenot 2004).

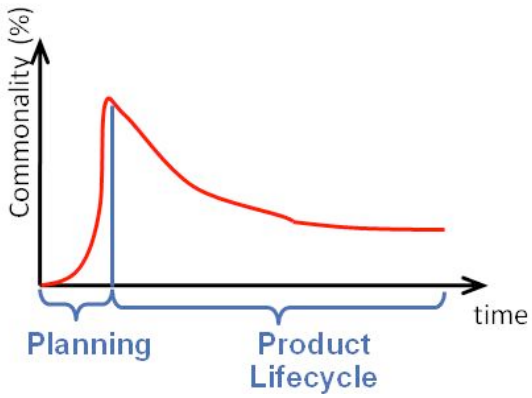

FIGURE 4: DIVERGENCE OCCURS

The following table shows common reasons for divergence, classified into acceptable reasons and unacceptable reasons based on whether or not they make the product better.

TABLE 3: CAUSES OF DIVERGENCE

Acceptable reasons for divergence	Unacceptable reasons for divergence
Technology developments mean that a better solution for the lifecycle of the whole family has been identified	Divergence to make **unnecessary performance improvements** unique to that variant
Market change necessitates a different second product than first anticipated	Divergence to **improve the cost or schedule of that particular variant**
Learning through development of the first product suggests changes for the second product	

1.6. SIMULTANEOUS VERSUS OFFSET DEVELOPMENT

When designing a product family, the variants can be designed in parallel or in series. Our work has shown that the most common method of development across a range of industries is offset development. This occurs for a number of reasons, but primarily because limited financial and human capital in an organization must be spread relatively smoothly across the whole development effort. For example, the development of the Ares I and Ares V rockets were offset, despite sharing some common hardware.

Most complex product families are developed sequentially, rather than in parallel. This increases the difficulty of successfully implementing commonality.

The implications for commonality are threefold:

1. in identifying commonality, the common system will need to be designed at a time when the detailed design of future variants is not available;
2. in evaluating the impact of commonality, there will be a strong penalty (possibly in all three of cost, schedule and performance) to the product designed first in time, and a benefit to subsequent variants;
3. in managing the implementation of commonality, the first product must consider subsequent products.

These points are examined in further detail in the later sections. Offset development is a key contributor to divergence.

1.7. STAGES OF THE PRODUCT LIFECYCLE

It is clear from the analysis of the benefits and drawbacks of commonality above that commonality has different impacts in different stages of a product's lifecycle. A simple breakdown of the product lifecycle into design, manufacture, testing and operation is enough to show that the beneficial impacts of commonality tend to occur over the lifetime of the product, while the drawbacks to commonality tend to be concentrated in the early stages of the product (refer to Figure 3). It also shows that the

The benefits of commonality occur later in time: through the manufacturing and operating phase of the initial product, and through all phases of subsequent variants. The drawbacks to commonality occur up-front. All stages of the product lifecycle must be considered in evaluating the impact of commonality.

beneficial impacts of commonality are most pronounced for subsequent variants of the product while the penalties are most pronounced for the initial product.

The presence of commonality across all phases of the lifecycle means that accurate analyses of commonality must look at uncertain future benefits and drawbacks as well as immediate concerns. The tools in Chapter 3 give some guidance as to how this can be done.

1.8. TYPES OF COMMONALITY

There are a range of different types of commonality. The most tangible expression of commonality is physical commonality, where identical elements are used across different systems or projects. Other types include:

- **Technological commonality:** the same technology is used, perhaps without any part commonality (for example two ablative heat shields could use completely different components, but the same basic technology). The benefits of this type of commonality focus on risk reduction during design and development.
- **Operational commonality:** the same operator interface is used, perhaps without any common parts. An example is using the same keyboard layout across different computers. This is important for operator training time. It also has benefits for operational risk in high risk environments such as on the International Space Station where common operation interfaces reduce the complexity of operations, which in turn reduces the risk of operator error.
- **Functional commonality:** the same function is delivered, perhaps without common parts, operating methods or technology. An example would be oxygen provision using ilmenite reduction or stored high pressure oxygen. This type of commonality is often used to reduce risk or logistics and maintenance costs during operations.

This toolkit focuses on physical commonality, because it is the most widespread and the easiest to conceptualize. However, the learning in this toolkit applies equally to the other types of commonality.

1.9. COMMONALITY BENEFITS ARE A MATTER OF PERSPECTIVE

The benefits of commonality also depend on the viewpoint of the potential beneficiary. An end-use operator looks for commonality benefit largely in decreased maintenance and logistics costs and more efficient training and operations. A manufacturer sees the potential benefits of commonality as lower design costs for future variants, lower capital costs for manufacturing, economies of scale and learning benefits during production, and decreased development risk for future variants. It is important to realize that not every organization over the product life cycle is looking to implement the same types of commonality. For example, NASA's contractors will not, of their own volition under traditional procurement contracts, be incentivized to look for the same types of operational commonality that will most benefit NASA.

1.10. COMMONALITY COMBINES TECHNICAL, MANAGERIAL AND PROGRAMMATIC EXPERTISE

Commonality is difficult to successfully realize. Chief among the reasons for this difficulty is that commonality is a design strategy that needs a full team effort to be successful. Commonality must be technically feasible, demonstrably cost effective over the lifecycle of the program, and supported through the program management. A

failure of any one of these different program sectors will jeopardize successful commonality.

Commonality requires real buy-in from the whole project team: technical, managerial and programmatic. Effort should be spent proving that commonality is a worthwhile strategy at the outset.

1.11. COMPLEMENTARY STRATEGIES TO COMMONALITY

In the next sections, the interactions between commonality and three other design strategies often linked with, or confused with, commonality are summarized. Specifically, the sections address standardization, flexibility and modularity.

1.11.1. COMMONALITY AND STANDARDIZATION

Standardization is certainly a type of commonality. Usually it refers to the use of common parts or processes across entire industries. Because of its breadth of application, it usually looks only at the commonality necessary for successful interoperability. It does not often consider more sophisticated common design to take advantage of learning curves, economies of scale and common manufacturing. However, when "standardized parts" within a single organization are discussed, "standardization" becomes closer to the type of commonality we consider in this paper. Semantically, "commonality" is often used where only a small number of products are specifically designed to have common elements; "standardization" on the other hand refers most commonly to the situation where almost all of a particular category of product has the common part.

Standards appear to be particularly effective in implementing commonality between the products of multiple corporations, and in implementing process commonality.

Standards were important for implementing commonality in Ground Support Equipment (GSE) at Kennedy Space Center. The analysis by Quinn (2008) shows that lack of commonality in GSE during Apollo and subsequent programs caused significant inefficiencies and sometimes damage to flight elements. GSE shipped by contractors was unsuitable for the KSC environment or incompatible with existing resources and interfaces at KSC. One part of the answer to this problem was the agency-wide GSE standard STD-5005 released in the mid 1990s.

1.11.2. COMMONALITY AND FLEXIBILITY

Commonality is one tool to permit flexibility in design. Flexibility refers to designs which can change to adapt to changing markets or use cases. A product family is usually designed to cover a suite of different use cases, with each variant targeted at a particular application. By implementing common design, the common platform can be produced to meet any of the market segments. This means that fluctuations in market segments do not affect the overall economy of producing the common parts so long as the fluctuations are balanced out by other market segments. Of course, commonality cannot deliver flexibility when the total market for all variants decreases.

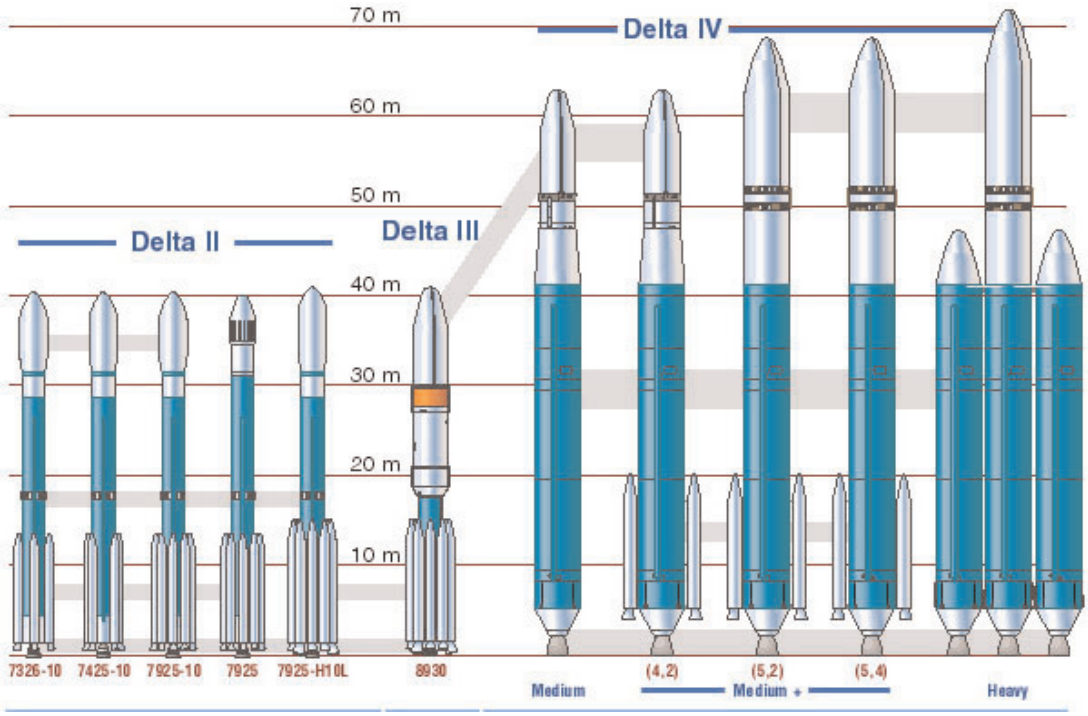

FIGURE 5: THE DELTA IV LAUNCH VEHICLE FAMILY (FEDERAL AVIATION ADMINISTRATION)

For example, in designing the Delta IV launch vehicle family shown in Figure 5, there may be uncertainty around the market split between the Medium, the Medium + and the Heavy variants. By designing a common core and engine the family is more economically adaptable to variations in this market split. Common elements between vehicles are shown with grey lines in Figure 5

1.11.3. COMMONALITY AND MODULARITY

Modularity is sometimes confused with commonality. They are not the same. Commonality is the reuse of parts from one product to another, whether planned or unplanned. Modularity is the delineation between the systems that form a product, with clearly defined interfaces.

However, modularity and commonality are complementary. If a subsystem is modular, then the amount of work to reuse that subsystem in a second product is reduced. The best example of this is software: modular sections of code can be reused without even needing to read the code (as for example in library functions), while with non-modular code it is often simpler to start from scratch than to try and reuse portions of the existing code. Modularity also makes it easier to estimate the cost of re-using the system.

As with commonality, there is often an up-front cost associated with building a modular system. The planning problem as to whether to invest this upfront cost is similar to the problem of whether to design a system to be common. Forward planning commonality can make the cost of building modularity more attractive, because the cost of building in modularity also leads to commonality benefits. The benefits of modularity are largely in risk and flexibility through the change-out of modular parts.

It is possible to have a modular system with no commonality, and vice versa. However, having both gives the benefits of commonality and

modularity for less than the cost of implementing each individually.

CHAPTER 2: TOOLS TO IDENTIFY TECHNICAL OPPORTUNITIES FOR COMMONALITY

2.1. OVERVIEW

Architecting systems with commonality begins with the identification of technically feasible opportunities for commonality. Without technical opportunities for commonality, the evaluation and management approaches in chapters 3 and 4 of this toolkit are redundant.

This section of the toolkit does two things. First, it gives an overview of a variety of different methods observed during the case studies for identifying technically feasible opportunities for commonality. It aims to give enough detail on the methods for a system architect to assess whether the method is appropriate to a particular situation. Actual implementation will probably require further reading of the relevant references. Second, it describes in detail a method developed at MIT for architecting systems at the conceptual design stage. This method was specifically designed with NASA's commonality needs in mind, and fills a perceived gap in commonality identification methodologies at the very early stages of development when little detailed information is available about the way the system will perform.

The summary table below is intended to help system architects choose an appropriate approach to the identification of commonality.

TABLE 4: APPROACHES TO IDENTIFYING TECHNICAL COMMONALITY

Approach	Applicability	Comments
Portfolio architecting	Very early in the design process Technology choices are understood, but little is known about detailed system performance	Approach is suitable only for identifying candidate systems for more detailed commonality analysis Requires some understanding of the costs of the system Can use iso-performance or iso-cost analysis to develop relative rankings between the options without knowing absolute cost or performance values
Heritage analysis	Common system designed to replace several existing independent systems, and some future systems Use when the requirements of future systems are uncertain *Example: Goddard Common Flight Software*	Relies on future system needs being analogous to past system needs Does not rely heavily on cost data

Approach	Applicability	Comments
End-item analysis	After concept design of both variants Future variant needs are well understood *Example: Constellation Space Suit System*	Limited opportunities for forward commonality Relies on cost data
Baseline and improve	When resources or knowledge do not permit full design of all systems concurrently *Example: F-35 Joint Strike Fighter*	May not identify all opportunities for commonality until it is too late to implement Requires incentivization of design teams to actively search for commonality opportunities
Incorporating commonality directly into numerical optimization	When the performance of the system in relation to its design is well understood When there is a narrow set of commonality opportunities being investigated	Cannot be used for searching very large tradespaces due to the need for a very detailed model
Visibility across design teams	As an adjunct to improve the effectiveness of other commonality methods	Possibly effective alone for small or simple design projects, where inter-system interactions are not complex; must be used with other methods for planned forward commonality
Component databases	When the primary benefits are manufacturing learning or economies of scale on a component level Example: *F-35 Joint Strike Fighter*	Can be used as an adjunct to project-level commonality optimization; Insufficient on its own to generate optimum commonality because it misses opportunities to make whole systems common

Once opportunities for technical commonality are developed using one of the methods outlined above, further analysis must be undertaken as to whether that commonality is cost-effective. This cost-benefit analysis is dealt with in Chapter 3, where methods for estimating the cost of commonality are discussed.

2.2. BACKGROUND

Technically feasible commonality occurs when a pair of projects or systems could both use an identical part instead of two uniquely designed ones, and still function.

In practice, technically feasible commonality includes some preliminary down-selection based on the difficulty in making the system common, since in almost all cases if enough time and money is spent on engineering an system it can be made to perform multiple purposes. Therefore the technical evaluations in this chapter and the economic evaluations in the next are best used linked together in a single iterative model for commonality design.

Although an engineering system is never fully understood at the time of concept design, this is when the first commonality analyses should be carried out. The disincentives to retrofit commonality at a later date mean that commonality analysis at the outset is invariably the best course of action, despite uncertainties in the models or data for early analysis.

It is more usual to discover that there are technical reasons why products cannot be common rather than reasons why products can become common as design progresses. Commonality usually represents a performance compromise and always entails additional work on the first variant. Cost and schedule pressures

almost always increase throughout a project due to unexpected events, and technical difficulty is more usually underestimated at the outset of a project than overestimated.

For these reasons, it is more likely that commonality will decrease over time, and establishing the maximum level of commonality early is critical. This was borne out in the case studies, where every case showed divergence occurring.

As emphasized in Chapter 1, trying for a system with too much commonality is as inefficient as designing a system with too little commonality. The approach in this chapter is intended to help system architects steer between these two shoals of under- and over- common design when initially architecting the system. Of course, such a path at the outset will not be perfect. Chapter 4 of this toolkit deals with management methods that can control the constant fine adjustments during the product development process which are required to keep the project on track to an optimum level of commonality.

Technical commonality must be sought out at the beginning of the project to be most effective.

2.3. PORTFOLIO ARCHITECTING

The first approach to be examined in this section is that of portfolio architecting. This is a method developed by Hofstetter at MIT which will be examined in detail. As the name suggests, the method is most often used at the outset of a product family design, in determining which aspects of the variants within a portfolio of projects should be made common. The portfolio architecting method focuses on commonality at the system level, not at the component level.

The portfolio architecting approach does not insist on a perfect understanding of the design and performance of the systems. This is a particularly useful approach for architecting space flight systems which often require significant research and development over timeframes of decades. It is impossible to have detailed knowledge of the system at the outset when initial commonality decisions must be made. The initial uncertainty does not mean that commonality decisions should not be made at the outset - instead, it means that initial commonality decisions need to be made and then supported by the ongoing commonality management described in Chapter 4. In using this type of approach, it is important to realize that the commonality study is aimed at selecting potential candidate architectures for commonality from the entire tradespace of potential commonality. The more detailed design methods explained in this chapter can then be used on those architectures. In many cases it would be impossible to use those detailed design methods on the entire tradespace.

The portfolio architecting approach uses a four step architecting methodology:

1. **Portfolio definition:** Identify which final products the commonality search will occur between.
2. **Architecture analysis without commonality:** Identify the optimum architecture for each variant separately, assuming no commonality between the variants.
3. **Commonality screening:** Decide on the level of similarity that indicates potentially common systems, and re-assess the architectures assuming certain parts are common. The required similarity must fit through four filters:
   a. **Complete functional commonality:** the systems must perform the same functions;

b. **Complete technological commonality:** the systems must use the same technology to achieve the functions;
c. **Similar operating environments:** systems which share operating environments are more likely to be made common;
d. **Similar design sizing:** systems which are closer in size to each other are more likely to be made common;

Sensitivity analysis is used to assess appropriate levels of similarity in 3(c) and 3(d).

4. **Preferred portfolio selection:** Select sets of interesting architectures with commonality. These architectures can then be passed to a more detailed level of design.

Figure 6 shows how these steps relate to the inputs and outputs described in more detail below, and how the size of the solution space changes as the commonality solution searches the tradespace.

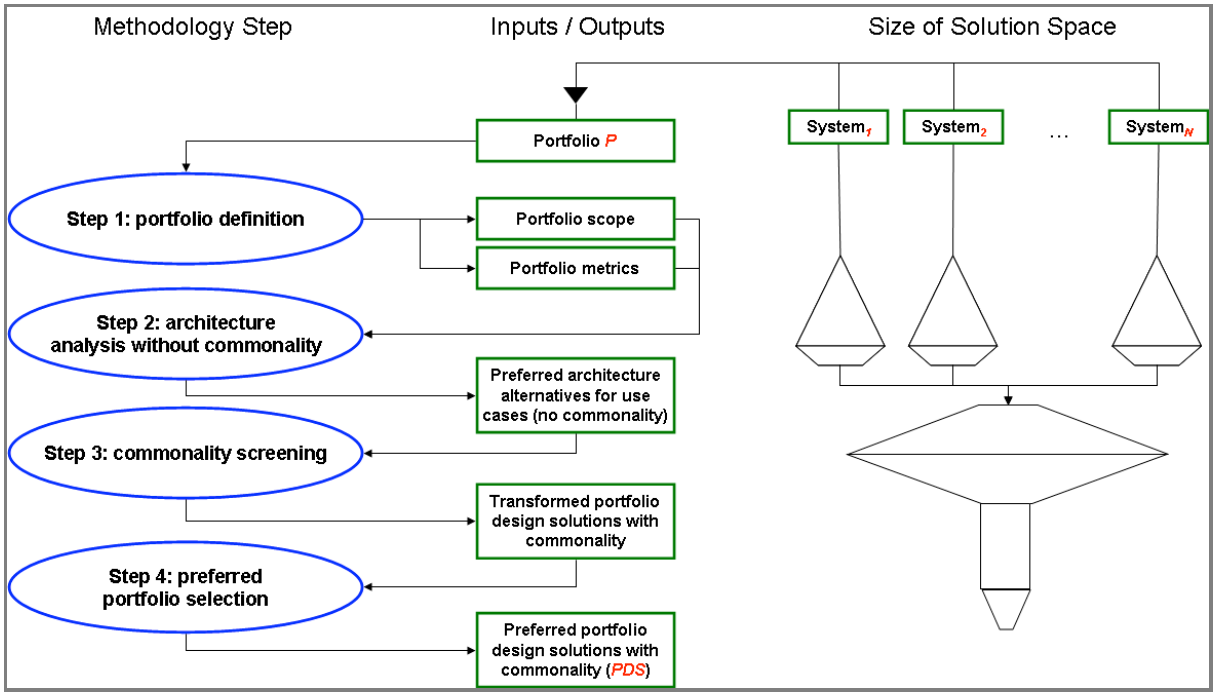

FIGURE 6: THE PORTFOLIO ARCHITECTING METHODOLOGY

2.3.1. Step 1: Portfolio definition and selection of metrics

As shown in Figure 6, the first step in the commonality identification process is to define the portfolio and select the metrics that will be used to compare different design solutions.

2.3.1.1. Portfolio Definition

In identifying which products should comprise the portfolio, the most elementary criteria is that the portfolio must share a beneficiary from commonality. The two systems should have a common customer, operator or manufacturer which is the organization that achieves benefit from the commonality and is incentivized to implement the commonality. The variants to be included in a portfolio should include relevant current projects, and projected future projects. It should also include any historical projects if those systems are well known and well understood, or if their systems are readily available. The variants may meet different purposes, but they should share at least one common function, otherwise the portfolio method of identifying commonality will fail. More detail on portfolio selection appears in Appendix A.

The portfolio will then consist of a **set of products sharing a common function which, when taken together, meet the customer's current and future needs**. The entire portfolio will be examined for commonality opportunities between any two of the products in the portfolio; it is not necessary that *all* of the products could share the same common system.

2.3.1.2. Selection of metrics

Once the portfolio is developed, the next task is to construct the metrics which will evaluate the individual architectures in the portfolio and also the portfolio performance as a whole.

The metrics should accurately assess the relative cost, risk (developmental and operational) and performance of the architectures. Absolute metrics are not necessary because the portfolio architecting methodology merely compares architectures to determine relative performance. Relative metrics are often the only choice at the early stages of design to assess architectures about which little is known, particularly with respect to cost.

If the commonality analysis is to be effective, the metrics must include some functions of quantity or time. Usually these are captured through learning curve benefits or economies of scale or both. Other benefits (or detriments) of quantity could also be included. A sophisticated model would take into account all of the life cycle effects identified in Chapter 1, such as decreased logistics or operations costs from commonality. The simplest model may only assess the number of independent development projects required, and weigh performance decrements against the decreased project risk implicit in a reduced number of development projects.

The commonality analysis is only as good as the metrics developed at this stage. Hofstetter's work presents metrics for total lifecycle cost for launch vehicles, planetary surface exploration vehicles and Environmental Conditioning and Life Support Systems (ECLSS).

The number of metrics required can be reduced by holding certain system characteristics constant throughout the analysis. This is known as an "iso-" approach. For example, in assessing launch vehicles, the analysis might require that the delta-v and payload performance for each variant within the portfolio does not change as the design of the variant is changed. Therefore each portfolio would have identical performance

characteristics and the portfolios could be ranked on life-cycle cost or some measure of operational risk.

2.3.1.3. TOOLS TO ASSIST PORTFOLIO SELECTION AND METRICS DEVELOPMENT

In developing the function descriptions and metrics, there are a variety of useful tools. Parametric cost estimating relationships, such as those available in (Wertz & Larson 1999) and the 2008 NASA Cost Estimating Handbook can help with developing metrics. The tools in the NASA Systems Engineering Handbook (NASA SP 6105) can assist with developing mission use cases. Learning curve models can also be found in the NASA Cost Estimating Handbook and in (Rhodes 2010).

2.3.2. STEP 2: ARCHITECTURE ANALYSIS WITHOUT CONSIDERATION OF COMMONALITY

The second step is to conduct an analysis of all possible architectures to decide which are the optimal architectures without any consideration of commonality. This step yields two results that are taken forward into the following steps:

1. it provides a cost and performance benchmark against which common architectures can be judged;
2. it identifies relatively good architectural combinations which are then taken through into the next stages of commonality analysis – the very poor combinations are left behind at this stage because commonality is unlikely to turn a low performing system into the top class of performance.

To conduct this analysis, first the variants are listed with their internal functions. Then the possible technology choices for each function of the variant are also listed. There may be some constraints between the technology choices. Figure 7 shows an example of the technology choices for a commonality study between the ECLSS functions of various projects, for example an in-space Crew Exploration Vehicle (CEV), a lunar rover intended for multi-day use and a fixed location lunar habitat. Only the ECLSS functions are shown.

Function	Technology choice 1	Technology choice 2	Technology choice 3	Technology choice 4	Technology choice 5	Technology choice 6
Food provision	Fully hydrated food	De-hydrated food				
CO2 removal	LiOH	4BMS	Solid amine, pressure-swing			
Oxygen provision	Stored, high-pressure	Stored, cryogenic	Water electrolysis	Electrolysis + Sabatier reactor	Electrolysis + Sabatier + methane pyrolysis	Ilmenite reduction (ISCP)
TCC	Expendable	Partially regenerative				
Clothing	Expendable	Washing machine + dryer				
Humidity removal	CHX + separator	Silica gel, expendable	Solid amine, pressure-swing			
Water management	Stored	Multifiltration	Multifiltration + VCD	Ilmenite reduction (ISCP)		

FIGURE 7: TECHNOLOGY CHOICES FOR ECLSS FUNCTIONS

For each variant, there is a set of feasible architectures based on the combination of each technology choice with each other feasible technology choice. For example, for the Crew Exploration Vehicle variant, an architecture might use De-hydrated food, LiOH CO_2 removal, stored high pressure oxygen, partially regenerative trace contaminant control (TCC), expendable clothing, silica gel humidity removal and stored water. The architectural choices which are made for this particular architecture are shown with red dots in Figure 7. A change to one of these technology choices would yield a different architecture.

Each of these architectures should be evaluated for its performance against the metrics. This involves optimization of the parameters of the design for each variant. The problem can be expressed as: *given* these customer needs for this variant, and *given* that this option for the variant *must* use these technology choices, what is the best design? This mini-optimization is then repeated for each technology choice option for each variant. To continue the example shown by the red dots in Figure 7, there will be a particular amount of high-pressure oxygen that is optimal for the Crew Exploration Vehicle to meet its requirements – this amount will be determined by technology-specific modeling not covered here. The results of that technology specific modeling, perhaps in terms of mass or cost, will be run through the metrics for the system, and an optimal point design for the CEV variant using the specified technology choices and measured according to the metrics will emerge.

Once optimization is complete, each variant will have an optimum design for each set of technology choices, ranked according to performance. The next step is to take a number of "interesting" architectures for each variant through to commonality analysis. The concept behind taking only a subset of architectures to the next stage is that the optimal technology choices for independent development may not be the optimal technology choices for common development, but a possible variant with highly suboptimal technology choices is unlikely to be an optimal choice for common development. Our research shows that taking approximately 10 different high-performing architectures for each variant through to the commonality analysis usually yields good results. Each of the architectures should involve different technology choices, and each should be optimized for that set of technology choices.

2.3.2.1. Dealing with existing or historical architectures

There are two aspects of the foregoing analysis which are slightly different for existing or historical architectures. Existing or historical architectures will not have technology choices associated with them. The technology choices and architectures are already determined.

Second, the metrics such as cost are often available directly, but it may give insight into the accuracy of the chosen metrics to analyze these known cases according to the developed metrics.

More information on using existing architectures in a portfolio can be found in Appendix A.

2.3.2.2. Tools to assist optimization

Step 2 is essentially an optimization problem, and methods abound to solve it. The key factors in choosing a solution method are:

- the number of variants and technology choices
- complexity of the relationship between the technology choice and the cost and performance (and perhaps risk) of the system will determine the approach
- the number of metrics by which the choices will be evaluated

Some guidelines and tools for solving the problem follow.

Simple problems of up to about 5 use cases with up to 5 technology choices each, and where the complexity of the optimization is limited to iterative sizing of the components can be solved using spreadsheet calculations. More complex problems may require bespoke programming as constraint satisfaction problems.

If an iso-performance or iso-cost approach is not used then a multi-variable optimization will be required. One way to do this is to explicitly provide a mapping of combinations of cost and performance to stakeholder value. An alternative is to use Pareto fronts.

At this point, analysis can also be conducted on sensitivity to assumed constants in the analysis. For example, sensitivity to changes in mission duration (for example on human spaceflight optimization), payload mass requirements (for example on launch vehicle analysis) or budget (for example for earth observing missions). The analysis can then be reworked to remove any solutions which are unduly sensitive to changes in uncertain parameters, replacing them with slightly suboptimal but more robust architectures.

2.3.3. Step 3: Commonality analysis of the portfolio

The third step is to take the interesting architectures identified in the optimal independent design analysis, and evaluate opportunities for commonality.

The portfolio analysis proposes four rules for determining if commonality is feasible between two systems:

1. the two systems must deliver the same internal function
2. the two systems must use the same technology choice for delivering that function
3. the two systems must be mostly operated in the same environments (the required *threshold factor* gives definition to "mostly")
4. the two systems must have similar quantitative design parameters (the required *overlap factor* gives definition to "similar")

Figure 8 shows this sequential approach graphically. These rules are fundamental to the methodology and deserve explanation in more detail. Rules 1 and 2 are very clear. No commonality is possible if the functional goals of the systems are different, or if the systems use different technologies to achieve their objectives. Certainly, there may be serendipitous instances where the sub-systems of systems using different technologies overlap, or where small tweaks to otherwise common systems enable different functional delivery. This is particularly the case with simple systems. On systems of the level of complexity of aerospace systems, this is less likely, and is in any case very difficult to analyze at this level of low detail design.

Rules 3 and 4 are more subjective. Rule 3 requires that most of the operational environments of the product are the same. It does not require complete congruence of operational environments. The reason is that adapting a technology to work in additional environments may not be difficult. There will be some "false positives" with this approach, where detailed design analysis confirms that the penalties associated with creating a system that works in the union of the set of environments for each system are too great, relative to the anticipated benefits of commonality. That is acceptable, as the purpose of this commonality screening tool is simply to identify likely candidates for further commonality analysis.

Rule 4 requires that the quantitative design parameters for the systems are close. The basis of this rule is that it is easier to make two systems common if they are trying to achieve the same function on the same scale. The parameters are often system "requirements" (though at a more detailed level than customer requirements). For example, the fuel storage function of a launch vehicle using liquid fuelled technology may use volume of fuel as the metric, the propulsion function may use thrust, and the ECLSS of the crew capsule might use oxygen throughput. Again, engineering judgment along with parametric sizing rules used in concept design[1] is necessary to guide selection of an appropriate overlap factor. Sensitivity analysis can also help.

A flowchart showing how these four rules are logically applied is shown in Figure 8. Commonality pairings drawn from interesting architectures in Step 2 are fed into the flowchart, and run through the four rules which filter out infeasible commonality opportunities. The feasible commonality opportunities are stored for evaluation in Step 4. More detail on the tools that can be used to efficiently run these filters can be found in Appendix B and in Hofstetter's work.

[1] For example those presented by Wertz and Larson

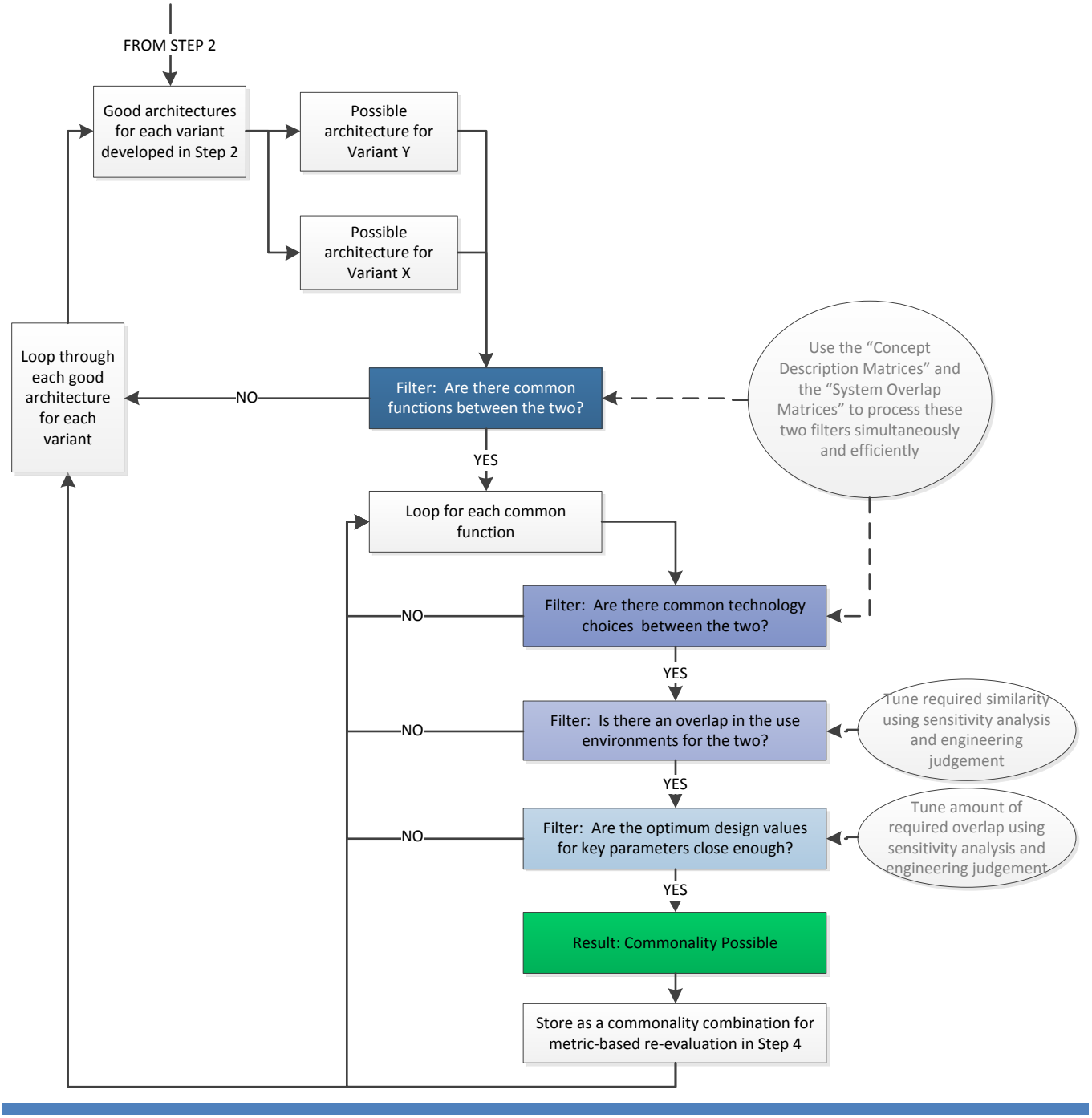

FIGURE 8: FLOWCHART SHOWING THE PROCESS FOR COMMONALITY FILTERING IN STEP 3

2.3.4. STEP 4: SELECTION OF PREFERRED PORTFOLIO DESIGN SOLUTIONS

The final step in the methodology is to analyze the portfolio of design solutions, from independent design through all feasible combinations of common design, to evaluate the portfolios which score well on the metrics.

The logic in selecting the portfolios of systems is to cycle through each identified commonality opportunity, first calculating the metrics for the project with maximum commonality, then progressively trying each combination of independent design against commonality.

Selection of those portfolios of common systems produces a figure similar to Figure 9, which shows the portfolio lifecycle cost against the number of development projects for the Saturn launch vehicle family. The figure illustrates the lifecycle cost of projects from totally independent development (14 projects) through 1 common project (13 development projects) to 7 common projects (7 development projects in all). This then represents the tradespace from common to independent development, from which a decision maker can select preferred projects for investigation in greater detail. More detail on this example can be found in (Hofstetter 2009).

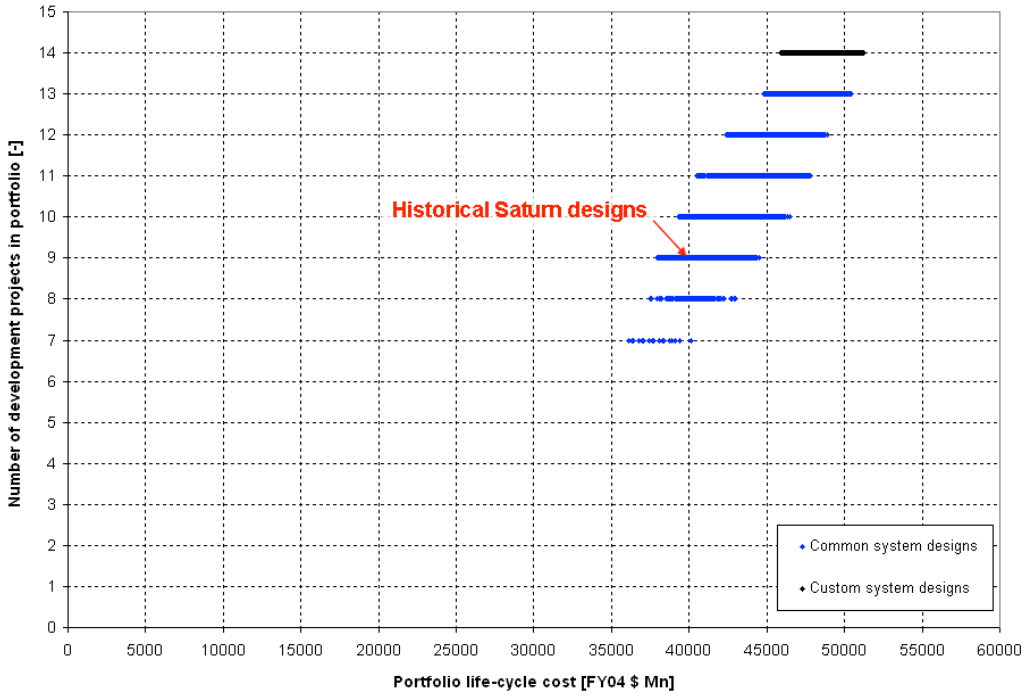

FIGURE 9: LIFECYCLE COST AGAINST DEVELOPMENT PROJECTS FOR THE SATURN LAUNCH VEHICLE FAMILY

2.3.5. FURTHER READING

Three worked examples for this method can be found in Hofstetter's 2009 dissertation: the Saturn launch vehicle family illustrated above, commonality in planetary surface mobility systems, and ECLSS design for a range of exploration applications.

2.4. ALTERNATIVE METHODS FOR IDENTIFYING COMMONALITY

Hofstetter's method presented above is an effective method when little is known about the system. There are other methods of identifying technical commonality which are useful when more information is known about the systems.

2.4.1. HERITAGE ANALYSIS

Analyzing a group of closely related systems which are already built helps to forecast the elements that should be common in a future system.

Engineers working on NASA Goddard's Common Flight Software were able to gain insight on architecture design decisions by analyzing previously developed systems in what they referred to as a heritage analysis (Rhodes 2010). Essentially, if the new system, which was planned to be common across a future family of satellites, could be developed with the flexibility to meet the previous systems' requirements than it is likely to be flexible enough to meet future system requirements. This method is most useful when there are a relatively large number of similar previous systems, and mission requirements and technologies are not changing rapidly.

If there are likely to be increases in requirements, heritage analysis can still be used. The pace of requirements change can be estimated from examining how systems have evolved in the past. The common system can be designed to cope with the predicted evolution over the time period between development of the common system and development of the variant systems.

2.4.2. END ITEM ANALYSIS

End item analysis can be used where the requirements for possible variants are already well known. The objective is to determine which components could be made common between two projects before fully designing either.

End item analysis was used to design commonality in the Constellation Space Suit System (CSSS) between the Configuration 1 Suit (designed for Launch, Entry and Abort and microgravity EVA) and the Configuration 2 Suit (designed for surface EVA). Commonality was managed in the CSSS case study through a formalized iterative process. This process began with the identification of commonality opportunities: the original system architects evaluated each expected Contract End Item (CEI) to determine whether or not the CEI could feasibly be common. If it was determined that it could become common, the CEI was baselined as such. Thereafter, any proposed changes to the architecture must be evaluated by the architecture and analysis group and approved in a multi-tier approval process (Rhodes 2010). The commonality search was able to be effectively performed on a piece-by-piece basis because the paper design of the two items was well understood.

2.4.3. BASELINE-AND-IMPROVE

The approach taken in the Joint Strike Fighter case represented an amalgam of approaches, and is suitable for projects with multiple variants where resource limitations mean that the variants are developed sequentially rather than in parallel. The variants were designed together to the Preliminary Design Review stage, then detailed design worked on each variant sequentially. The approach used started with a previous variant and then modified the baseline variant as required and justified by performance

and lifecycle cost analysis (Boas 2008). This sequential approach takes advantage of learning throughout the design and manufacture process; the counterpoint to this approach is that an increased rate of beneficial divergence due to learning should be expected.

2.4.4. TRADESPACE DESIGN WITH COMMONALITY

When the design of the system is well known, and the candidate systems or components for commonality have been identified, then commonality can be included in the system design and evaluated across the tradespace.

The additional commonality analysis can be implemented as a constraint in the optimization. The parameters of the common part must be constant across all variants. For example, instead of analyzing the tradespace of solutions and attempting to optimize three core diameters d_1, d_2, d_3 for a launch vehicle family, the tradespace is constrained and only one core diameter is analyzed, that of d_c. It is necessary to do further analysis to show that the likely lifecycle cost of using a common core diameter is less than that of using three individually optimized diameters.

For more details on this method, see (Willcox & Wakayama 2002).

2.4.5. VISIBILITY

Giving system engineers visibility across different systems improves the likelihood of reactive reuse between systems. It also increases the likelihood that potential new opportunities for forward commonality will be found. This is particularly the case if incentives are given to the system engineers to explore opportunities for commonality. An alternative way of achieving this is to conduct detailed analysis of the whole-of-lifecycle costs of the system under development, and to encourage engineers to minimize those costs, rather than just those of the particular system they are designing.

However, it is insufficient on its own for good commonality management. The absence of incentives for forward commonality will discourage project managers from suffering the up-front penalty for forward commonality. The absence of mechanisms to discourage detrimental divergence may encourage later systems to design new, performance-optimizing parts instead of using the common parts.

2.4.6. COMPONENT DATABASES

The idea of "visibility" is also present at the component level. Some projects have developed lists of components to help maintain commonality and increase economies of scale (for example the Joint Strike Fighter (Boas 2008)). Without incentives to implement commonality, however, such part databases are unlikely to significantly increase commonality. More tangible metrics like weight or cost on which an engineering team's performance is judged will prevail. The most extreme example of this occurred during a launch vehicle development program, where identical bolts differing only in specified hardness, and only by one gradation, were not made common because the engineers resisted the microgram differences in weight. Presumably this was because the team was formally evaluated on the weight of their system but not on its commonality.

Component databases should be seen as an adjunct to realizing the benefits of commonality, rather than a solution to commonality in themselves. (Yeager 1987) goes more deeply into the theory of component databases, looking chiefly at the Station Commonality Analysis Tool developed in the 1980s.

2.4.7. SYNOPSIS OF MANAGEMENT METHODS THAT ASSIST TECHNICAL COMMONALITY

The following techniques, described in greater detail in Chapter 4, can improve the effectiveness of teams working to identify commonality opportunities:

- give teams visibility into the technical work on other projects or systems
- give teams visibility into the lifecycle cost implications of their designs
- create incentives for identifying technical commonality opportunities

2.5. CONCLUSIONS

There are a wide variety of methods available for evaluating technically feasible opportunities for commonality. The main characteristic which distinguishes between the applicability of the methods is the degree of knowledge about the system's design and performance which is required to implement the method. For example, Hofstetter's approach can be used when only very basic metrics about the system are known, whereas Willcox and Wakayama's tradespace exploration methods require a more detailed model of system performance.

In practice, a combination of these methods should be used throughout the project as knowledge matures. Remember that not all commonality methods will be implemented because some fail the net benefit tests described in Chapter 3. Also, not all initially planned commonality will occur, and the commonality planned for a project will be continually revised to take into account the changes to commonality opportunities. This means that the identification of commonality is an ongoing process, and system architects should have at least a passing familiarity with all of the tools mentioned in this toolkit.

CHAPTER 3: TOOLS FOR EVALUATING THE ECONOMIC IMPACT OF COMMONALITY

3.1. INTRODUCTION

A constant theme of this toolkit is that commonality carries advantages and disadvantages. A key step in successfully working with commonality is evaluating whether a given commonality opportunity provides a net benefit. This section provides tools to help make that evaluation. Specifically, it contains two approaches for modeling and comparing the advantages and disadvantages of commonality:

- a **cost-benefit model** for commonality which draws on the work of Boas and Rhodes; and
- a **decision analysis tool** based on the work by Rhodes which allows analysis of the best development strategies accounting for the realities that divergence may occur and planned commonality may not be realized.

3.2. COMMONALITY AS A TRADE

Commonality can improve the cost, schedule, risk and / or performance of a particular program or series of programs. This improvement does not come for free. Hard decisions need to be made in implementing commonality, often leaving performance "on the table." The most obvious example is where not using a common part would allow a specifically tailored solution and yield a small performance improvement. Management discipline is absolutely required under these circumstances to evaluate the longer-term effect of not using the common part on the whole-of-life costs of the product family, appropriately discounted, and to make the decision as to whether to sacrifice commonality at that point.

Figure 3 in Chapter 1 illustrates the trade between advantages and penalties of commonality.

3.3. COST MODEL

Using a cost model for commonality is essential to demonstrating that commonality is a worthwhile investment in a particular case. This section presents one commonality cost model developed at MIT by Boas and Rhodes. Like all cost models, the one presented here depends heavily on the input data and the experience of those compiling the cost model. This cost model should be used as a framework to identify the components of costing which are directly affected by commonality, and which may be overlooked in a conventional cost model. It does not present all the detail necessary to run a full cost analysis of a common system because factors like learning benefits, performance penalties and initial investment in common elements are project-specific.

Figure 10 shows how the cost model combines the input data from traditional cost estimation (in the left hand boxes) with the essential features of commonality largely discussed in Chapter 1 (lower box) to produce a cost model incorporating commonality.

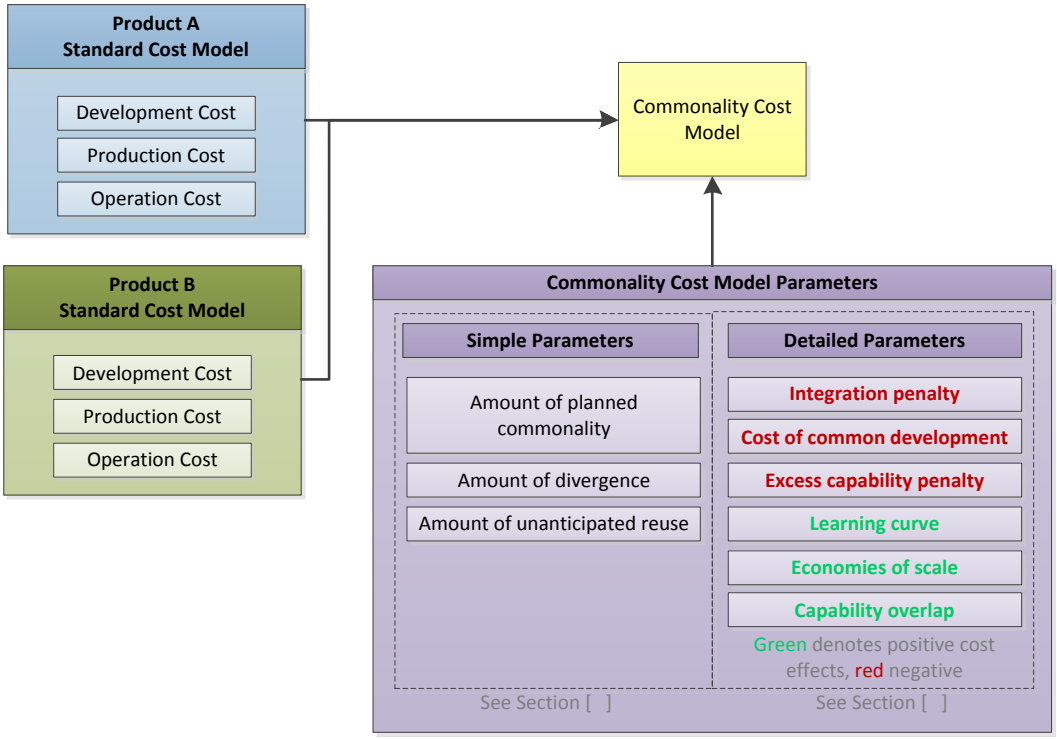

FIGURE 10: INTERACTION BETWEEN TRADITIONAL COST ESTIMATION AND COMMONALITY MODELLING TO PRODUCE A COST MODEL FOR COMMONALITY

3.3.1. THEORETICAL BASIS FOR COST MODEL

A summary of the features of commonality addressed by the cost model are presented in the "Commonality Cost Model Parameters" box of Figure 10. Many of these features have been explained in the theoretical discussion of commonality in Chapter 1. This chapter will explain them in more detail. The cost model presented in this chapter first looks at a simple model using only the simple parameters, then expands that model to include the detailed parameters.

The elements of the cost model are divided according to the classification of common and unique parts between product A and the variant, product B, into five classes. Those classes are:

1. Parts which are unique to product A, and which remain unique to product A;
2. Parts which are intended to be common, but which are only in fact used in product A;
3. Parts which are intended to be unique to product A, but which are reused in product B;
4. Parts which are intended to be common, and which do in fact become common to both products; and
5. Parts which are unique to product B.

Figure 11 and Figure 12 show how commonality affects the costs of each of those classes. Figure 11 shows the effect on non-recurring costs and Figure 12 shows the effect on recurring costs. Common development affects non-recurring and recurring costs differently because product A, being first, usually bears the brunt of the design

and manufacture costs of the common components, while during operation both product A and product B benefit from sharing costs like operations facilities and from decreased spares.

The cost model does not distinguish between common components and common systems. It could be used for either level of detail. For simplicity of explanation, it assumes that only two products are being developed, but the model could easily be extended to multiple products from the basis presented here.

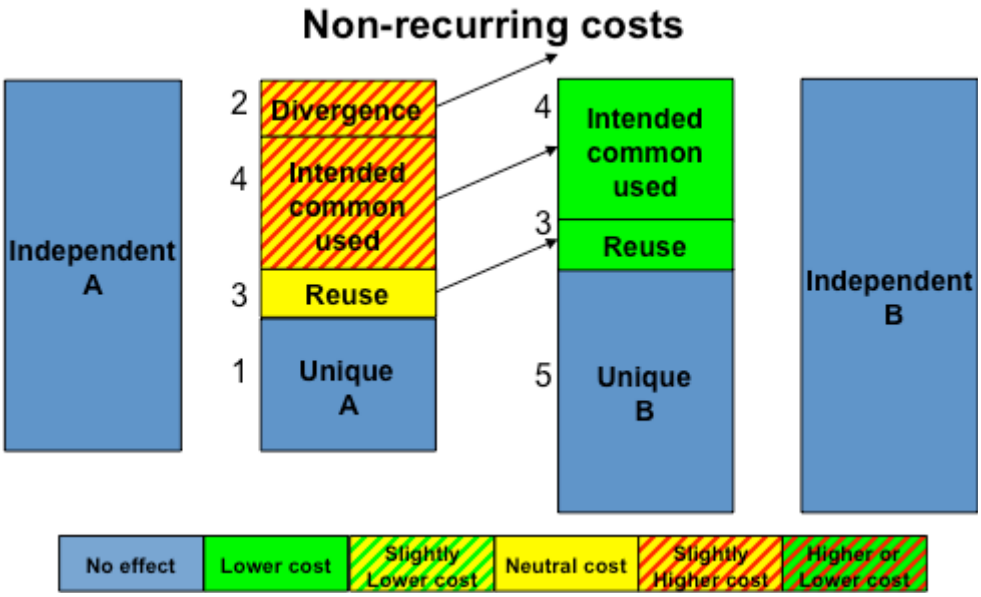

FIGURE 11: BREAKDOWN OF TYPES OF NON-RECURRING COST IN A TWO-PRODUCT PARTIALLY COMMON SYSTEM

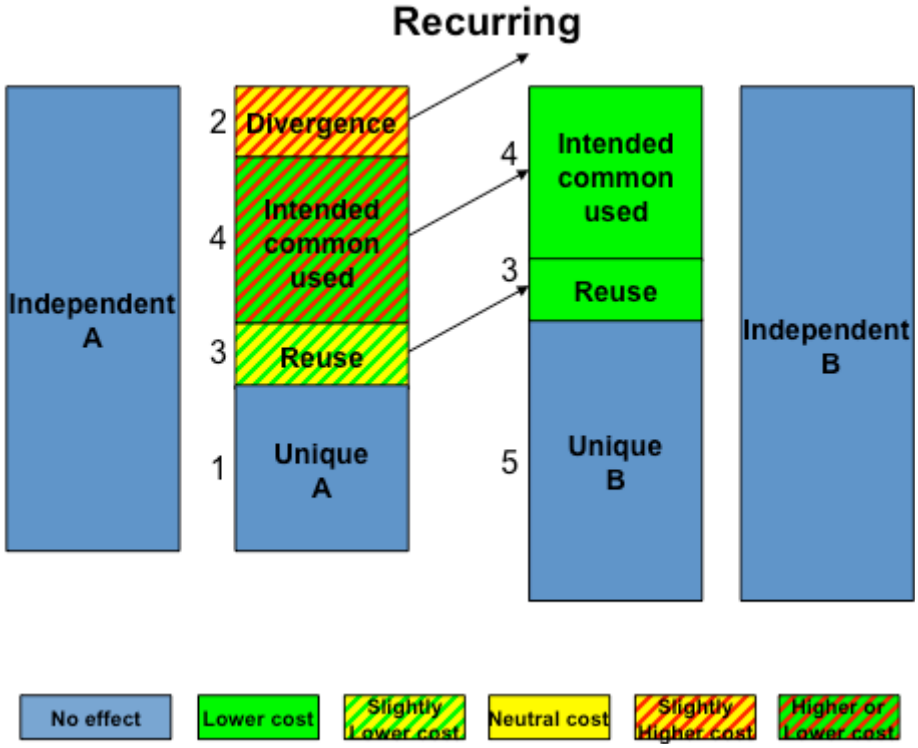

FIGURE 12: BREAKDOWN OF TYPES OF RECURRING COST IN A TWO-PRODUCT PARTIALLY COMMON SYSTEM

3.3.2. SIMPLIFIED COST MODEL

3.3.2.1. INDEPENDENT DEVELOPMENT COST INPUTS

The first inputs required are the cost estimates for developing, manufacturing and operating the systems independently (without commonality). These cost estimates are best made by traditional cost estimation methods, such as parametric cost models and cost estimation by analogy. The two independent cost estimates are denoted as A and B in the equations which follow.

An alternative approach to separately costing manufacturing and operations is to represent production and operations costs as some function of development costs. For example, some cost models assume that the first unit of production will cost 10% of the total development (Hofstetter, 2009).

3.3.2.2. COMMONALITY MODEL INPUTS

In addition to the independent development cost estimates, the simplified model takes as inputs the percent of intended commonality, divergence, and unintended reuse. These factors dictate the relative size of each of the five classes in the model.

- Commonality (C) – the percentage of system A that is developed with the intent that it becomes common with system B. This factor should be known at the time that the first system is developed and should be quantified.
- Divergence (d_{iv}) – the percentage of the intended common components of A that actually become unique to system A. Divergence may be more difficult to estimate because divergence will not have occurred yet and management will attempt to control divergence if possible. Divergence values from analogous projects could be used.
- Reuse (r_{iu}) – the percentage of the intended unique components of A that actually become common between the two systems. Reuse may also be difficult to estimate because if management anticipated that a component could be reused than it would be marked as common. Reuse values from analogous projects are probably the best methods to estimate this percentage.

By quantifying each of the above values, the classes can be quantified into a cost basis using the equations in Table 5. This table shows how to update the independent cost estimates for the two products to take account of common development, divergence, and unanticipated reuse.

The formulas for the cost basis divide the cost development estimates for the independently developed systems into the five categories illustrated in Figure 11 and Figure 12. The more detailed cost model which follows further divides the classes into development, production, and operation of the systems.

TABLE 5: SIMPLIFIED COST BASIS FOR COMMON PRODUCTS

Class	Class description	Cost basis for the class
1	Intended Unique A which remains Unique A	$A*(1-C)*(1-r_{iu})$
2	Intended Common which becomes Unique A	$A*C*d_{iv}$
3	Intended Unique A which becomes Common	$A*(1-C)*r_{iu}$
4	Intended Common which remains Common	$A*C*(1-d_{iv})$
5	Intended Unique B	$B-A*(C*(1-d_{iv})+(1-C)*r_{iu})$

Key to Table 5

A - cost of developing independent A
B - cost of developing independent B
C - intended common % of A (ie prior to reuse or divergence)
r_{iu} *- reuse of intended unique*
d_{iv} *– divergence of intended common components*

TABLE 6: COST MODIFIERS USED IN THE DETAILED COMMONALITY COST MODEL

Class	Development		Production		Operations	
	Product A	Product B	Product A	Product B	Product A	Product B
1	1	0	1	0	1	0
2	p_{cd}	0	p_{capAp}	0	p_{capAo}	0
3	1	p_{int}	$(L_{AB})_A$ eos	p_{capBp} $(L_{AB})_B$ eos	$(L_{AB})_A$ cap_o	p_{capBo} $(L_{AB})_B$ cap_o
4	p_{cd}	p_{int}	p_{capAp} $(L_{AB})_A$ eos	p_{capBp} $(L_{AB})_B$ eos	p_{capAo} $(L_{AB})_A$ cap_o	p_{capBo} $(L_{AB})_B$ cap_o
5	0	1	0	1	0	1

Symbols used in Table 6 are explained in section 3.3.3

3.3.3. DETAILED COMMONALITY COST MODEL

As Figure 10 illustrates, it is possible to delve into the cost model in an additional level of detail. This level of detail captures integration penalties, learning curve effects, economies of scale and other benefits of commonality.

Table 6 shows the benefits and penalties for each class and product phase. The paragraphs that follow explain the notation, and address the rationale behind the benefits and penalties.

3.3.3.1. CLASS 1: INTENDED UNIQUE A THAT REMAIN UNIQUE TO A

The life cycle costs of items that are developed with the intent that they are unique to product A and do in fact remain unique are unaffected by commonality. As a result, the cost of this class is best estimated with traditional cost estimation tools, such as parametric cost models or analogy. The cost estimates for this fraction of the system feed directly through the cost model without adjustment, as demonstrated by Table 6 in which a '1' indicates that the cost basis is not affected by a benefit or penalty related to commonality and a '0' indicates that the cost basis is not included in the cost estimate for that phase of the product.

3.3.3.2. CLASS 2: INTENDED COMMON TO UNIQUE A (DIVERGENCE)

Development

Developing an item with the intent that it becomes common requires the developers to develop the component to meet the requirements of the current product and the requirements of the expected future variant products. As a result, the development cost of the intended common elements of product A are higher, increased by the penalty cost of common development (p_{cd}). The cost of common development represents the additional labor and coordination cost of design and development of the components that are expected to be common with product B. However, divergence occurs, causing the elements in class 2 to be unique to product A. Therefore, class 2 penalizes product A and is not a part of the cost estimate for product B, as indicated by Table 6.

Production and Operations

An element that is developed to meet additional requirements is assumed to be more complex than independently developed elements and as a result is more expensive to produce and operate. The resulting added expense on product A is described as the excess capability penalty (p_{cap}). The excess capability penalty likely does not affect the different life cycle phases equally, therefore the excess capability penalty is unique to the phase and product to which it is applied. As a result, Table 6 shows p_{capAp} for the capability penalty for product A in production and p_{capAo} for product A in operations. In class 2, product B does not have an excess capability penalty because divergence occurs, causing the elements to be developed independently.

3.3.3.3. CLASS 3: INTENDED UNIQUE A TO COMMON (REUSE)

Development

In class 3, elements are developed with the intent that they are unique to product A. Despite this intent the elements in class 3 are reused on product B. As a result, product B obtains development benefits from commonality without the upfront cost of common development on product A. The reuse will benefit the development phase of product B by decreasing the development work required, however an integration penalty (p_{int}) will likely be created by the development work required to integrate the previously developed elements into product B.

Production

This class of elements is not developed with the intent that they become common, as a result product B is benefitted without imposing an excess capability penalty on system A. However, there may be an excess capability penalty (p_{capBp}) on product B in production and operations because the components that are reused may be sub-optimal for product B, depending on which system is more capable. The production portion of the system is composed of labor costs and materials cost. The labor portion of production benefits from learning (L_{AB}), while the materials portion is benefitted by economies of scale (eos). Learning benefits are related to the learning that

occurs through repetition of the same tasks, resulting in decreased time or cost to conduct the task. Learning can be incorporated into the model using a learning curve equation such as those presented in (Newman et al. 2004).

If product A and product B are produced simultaneously and use the same component, both systems derive more benefit and share equally in the benefits from learning. If there is a time offset, the system that is developed later obtains greater benefits from learning than the first system because production of the common components in the later system begins farther down the learning curve.

The economies of scale factor represents the benefit obtained from purchasing or producing more components in the same amount of time. Many equations are available for developing economies of scale. One suggestion is those presented by (de Neufville 1990).

Operations

The operations portion of the system is composed of fixed recurring and variable recurring costs. Operations fixed recurring costs are required regardless of the operations conducted and often include sustaining engineering and operations infrastructure, while variable recurring costs depend on the operations conducted and often include operations labor, training, and logistics. The variable recurring costs in operations are benefitted by accelerated learning (L_{AB}), similar to the production costs described above. The fixed recurring costs are benefitted by a capability overlap (cap_o). The capability overlap is the benefit created by an overlap in fixed recurring costs. For the same components used by multiple systems the fixed recurring costs of the components are not duplicated because they do not depend on the number of operations per year, but only on the ability to operate the component. The economic model assumes that products A and B split the benefits from the capability overlap based on the operation rate of each system. The benefits of the capability overlap could have been assigned completely to product B, but doing so may artificially penalize product A and benefit product B. The distribution of benefits does not affect the total life cycle costs.

3.3.3.4. CLASS 4: INTENDED COMMON TO COMMON DEVELOPMENT

This class of elements is developed with the intent that the components become common between the products, and in fact the elements do actually become common. This class has an upfront cost of common development (p_{cd}) on product A, as in class 2, and an integration penalty (p_{int}) on product B, as in class 3.

Production and Operations

The production and operation phases of this class are marked by the possibility of excess capability penalties (p_{cap}) on both product A and product B, as the common product may be optimized for neither. In addition, the labor portion of the production phase on both products benefits from learning (L_{AB}) and the materials portion in the production phase of both products benefits from economies of scale (eos). The variable recurring costs of the operations phase are benefitted from learning (L_{AB}), while the fixed recurring costs are reduced by a capability overlap (cap_o) as in class 2.

3.3.3.5. CLASS 5: UNIQUE B

At the time that product B is developed, some elements will be developed to be unique to product B and will in fact remain unique. As a result, the independent cost estimate for this class passes through the model without any modification.

3.3.4. COST MODEL CONCLUSION

The cost model presented above is one way of analyzing the costs and benefits of commonality. It obviously depends strongly on the accuracy of the initial cost models for the systems, and the estimates for each of the parameters like the amount of commonality and divergence. However, cost modeling using this framework is likely to focus precise attention on the plans for, and likely cost effects of, commonality.

3.4. DECISION ANALYSIS

3.4.1. INTRODUCTION

The preceding discussion in this toolkit has illustrated that there are both advantages and disadvantages associated with commonality. It is rarely obvious at the outset which way commonality will trend in a given project, especially given that external uncertainties like stakeholder needs may change during a project. How then does a project manager or program architect evaluate whether to try for commonality?

One method is to look at commonality in the system as being a point of flexibility. Common design can be treated as a design decision that can be reversed as the project unfolds if the payoffs from commonality shrink below those of independent design. The project manager could also decide on some intermediate strategies, like reducing the scope of the common development to items which showed more benefit from commonality or which were less likely to become divergent.

The power of the decision analysis method presented here is that it can be conducted at the start of the project to determine the best opening strategy, and also refined during the project to provide a rational decision making strategy as events unfold.

3.4.2. DECISION ANALYSIS INPUTS

There are two ingredients for successfully implementing decision analysis:

1. a comprehensive cost model, which may contain uncertainties;
2. an understanding of the probabilities of the outcomes from any uncertain events in point 1.

These ingredients are not trivial, and emphasize the importance of a good initial cost model and a solid understanding of the way commonality is likely to behave in practice.

3.4.3. UNCERTAIN EVENTS

Decision analysis is based around "relevant uncertain events". These are events which affect the result of the decision to implement commonality. The usual time to use decision analysis is at the outset of the project, when deciding whether to pursue common development between two products or to develop them independently. Taking the simple case of products A and B, some relevant uncertain events include:

- **Cost and budget unknowns**
  - the start time of product B
  - the end time of product B (reflecting scope changes or budget variations)
- **Technical unknowns**
  - the amount of divergence between A and B (ie decrease from currently expected commonality)
  - the integration penalty to include the common components in product B
- **Future demand unknowns**
  - the number of units of A and B actually produced (including whether product B is even produced at all)
  - whether stakeholder demand changes such that a third product, C, which could use the same common base is produced

Notice that most of these uncertainties deal with product B. This is usual, because product B is produced later in time and therefore is less certain at the time of initial decision analysis. Some of the uncertainties can be refined using aspects of commonality theory discussed in Chapter 1. For example, the uncertainties should

reflect that the further offset product B is, the greater the likelihood that product B might diverge (see section 1.6). As a separate example, the benefits which product B sees from using the common component, which are linked to the relative production volumes of A and B, should affect the likelihood of divergence (see section 4.4).

These uncertain events are modeled as discrete events, with a range of outcomes with associated probabilities. If the uncertain events are more accurately described using continuous variables (for example the market price of oil) then a similar approach called Real Options Analysis can be used. It is also possible to use hybrid models incorporating both Real Options Analysis and Decision Analysis (Neely & de Neufville 2001). The factors which give rise to the greatest uncertainty in NASA projects tend not to be easily quantifiable, price continuous or price visible; contrast this with commercial enterprises where the price of raw materials or interest rates strongly affect project viability. Therefore this toolkit concentrates on Decision Analysis.

The next step is to apply probabilities to each of the events. for example, the following might be the probability table for the uncertain event "divergence between product A and product B":

TABLE 7: EXAMPLE PROBABILITIES FOR DECISION ANALYSIS

Event	Probability
5% divergence	0.2
25% divergence	0.5
50% divergence	0.3

These uncertainties should also affect the outcome of the cost model as discussed in the previous section. For example, increased divergence will negatively affect the economies of scale, lifecycle logistics costs and efficiencies due to learning in most cost models. If an event does not affect the outcome of the cost model, it is not relevant in this analysis. If intuitively it should be relevant, then the cost model may need to be updated.

3.4.4. USING DECISION TREES

In order to conduct the analysis of the decision pathways, a decision tree is constructed. The decision tree links decision points with chance nodes until an outcome is reached. A simple decision tree is shown in Figure 13.

The initial decision is whether to pursue common or independent development. Then a chance event is simulated as occurring, for example the delay of Product B from its proposed development start date. The probability of outcomes for this event could be represented as:

TABLE 8: PROBABILITY OUTCOMES FOR DECISION ANALYSIS EXAMPLE

	Probability	Outcome
Outcome 1	0.5	0 year delay in start date for B
Outcome 2	0.5	2 year delay in start date for B
Outcome 3	0.6	0 year delay in start date for B
Outcome 4	0.4	2 year delay in start date for B

In this case, the initial decision affects the likelihood of a 2 year delay in project B, because of the emphasis by thoughtful program executives on a timely start to minimize divergence. In general however, decisions need not affect the chance nodes. The outcomes are possible delays in project B. By itself, this event doesn't give the decision maker any insight into

whether commonality is still a better strategy than independent development. In the event of delay, assuming divergence, the benefits of commonality are likely to be lower than before, but on balance it may still be a better solution than independent development.

After the start date of project B is finalized, the projects which decided to use common development could make the decision to switch back to independent development. (It is assumed in this example that the costs of switching back to common development from independent development are prohibitive, but this may not always be the case.) There is still no guidance on how to best make this decision.

A subsequent chance node might deal with the actual amount of divergence between the two products. For example, possible outcomes of 10% and 50% divergence at probabilities of 0.7 and 0.3 respectively for the no-delay case, and 0.5 and 0.5 for the delay case. This then leads to the final boxes in Figure 13 labeled "Outcome".

The commonality cost model can be applied at these final outcomes because the amount of divergence at each outcome is known. Now, each of the initial decision and the subsequent decision lead to clearly defined cost outcomes at certain probabilities. This is now a decision making tool. By multiplying outcomes by probabilities, each decision making node has an "expected value" associated with the decision choices. The decision maker should make the decision which leads to the highest expected value (or lowest expected cost).

Note that the cost of switching from common to independent design is not currently captured in the cost model presented earlier in this chapter. Some allowance may need to be made for this switching cost when conducting the analysis.

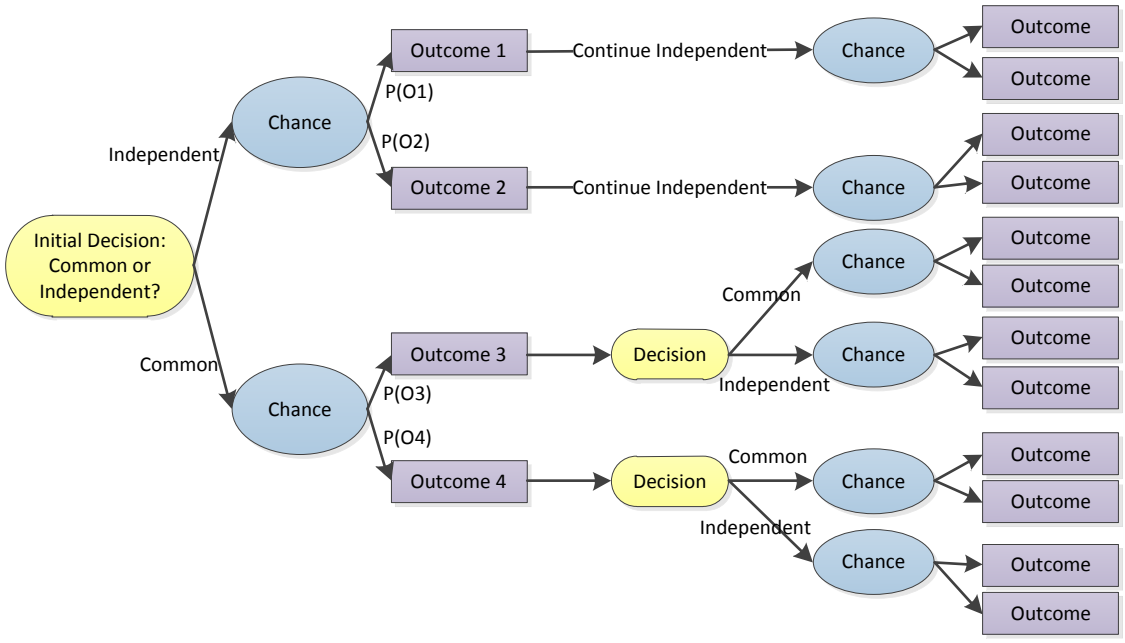

FIGURE 13: SIMPLE COMMONALITY DECISION TREE

3.4.5. CONSTELLATION SPACE SUIT SYSTEM WORKED EXAMPLE

Figure 16 shows a more realistic scenario modeled by Rhodes for the Constellation Space Suit System,[2] which demonstrates the advantages of being able to calculate expected outcomes. At the outset, there is the opportunity to make a decision to pursue common or independent development. Then a chance event occurs which is the delay of development start of product B from its planned start date. Possible outcomes are taken to be 0, 2, 4 years. The effect of this offset on the cost model is that increasing the offset decreases the benefits to production and operations phases (due to less overlap), and also increases the chance of later divergence.

After the simulated chance event of the delay, a further decision can be made as to whether independent or common development is pursued. The decision is followed by a chance node for the common development options only, representing the amount of divergence between product A and product B. Finally, a third decision point to decide whether to switch to independent development is offered, followed by a chance that measures the extent of scope change in product B.

The red arrows on the diagram show that commonality is the preferred initial strategy (having a lower cost than the independent branch). Commonality is also the preferred strategy after the delay to product B is known, except in the case where the delay to product B is 4 years or greater. In that case, the expected cost of common development exceeds independent development and independent development should be pursued.

[2] The analysis presented by Rhodes uses estimated numbers for the study, based on his experience in analyzing the commonality in CSSS. The analysis should be updated with more accurate numbers by the relevant NASA groups before any decisions are made as a result of the analysis.

3.4.6. UNDERSTANDING RISK USING VALUE-AT-RISK GRAPHS

Further insight into decision making for commonality can also be found through value-at-risk graphs. These graphs display the range of outcomes from a particular strategy, with associated probabilities. Therefore comparisons can be drawn between very variable strategies with potentially high pay-offs, and more conservative strategies with lower expected outcomes but with less variance in the results.

A value-at-risk graph can be produced by plotting the cumulative probability of each outcome, starting with the lowest cost outcome and moving towards higher cost outcomes. The discrete nature of the decision tree gives the graph its characteristic stepped appearance.

Figure 14 is a value-at-risk graph taken from Rhodes' CSSS study. It takes all the possible development paths represented in the Figure 16 decision tree and categorizes the outcomes into three categories:

- always choosing independent design;
- always choosing common design; or
- switching from common to independent design at some point.

The model shows that while always choosing commonality has the best outcome on average (dotted red line is to the left of all other dotted lines), it also has some outcomes which are potentially more costly than any other strategy (red line in top right). Always choosing independent development will be more expensive on average, but the best and worst outcomes are closer to the average. The green line, representing switching, includes all outcomes which are arrived at through a switch

from common development to independent development at some point. The switch is not based on the strategy of choosing the lower cost pathway – it is simply the sum of the outcomes of all possible outcomes involving switching.

Figure 15 shows the effect of introducing the smart decision making at each node based on the expected costs of the outcomes from each decision. In Figure 15 the green line represents this responsive decision making. The graph shows that it produces a slightly better expected result overall than either commonality or independent development. The probabilities of lower costs are higher, though the worst-case outcomes are still worse than purely independent development. Rhodes presents more detail on these graphs in his discussion of the CSSS study.

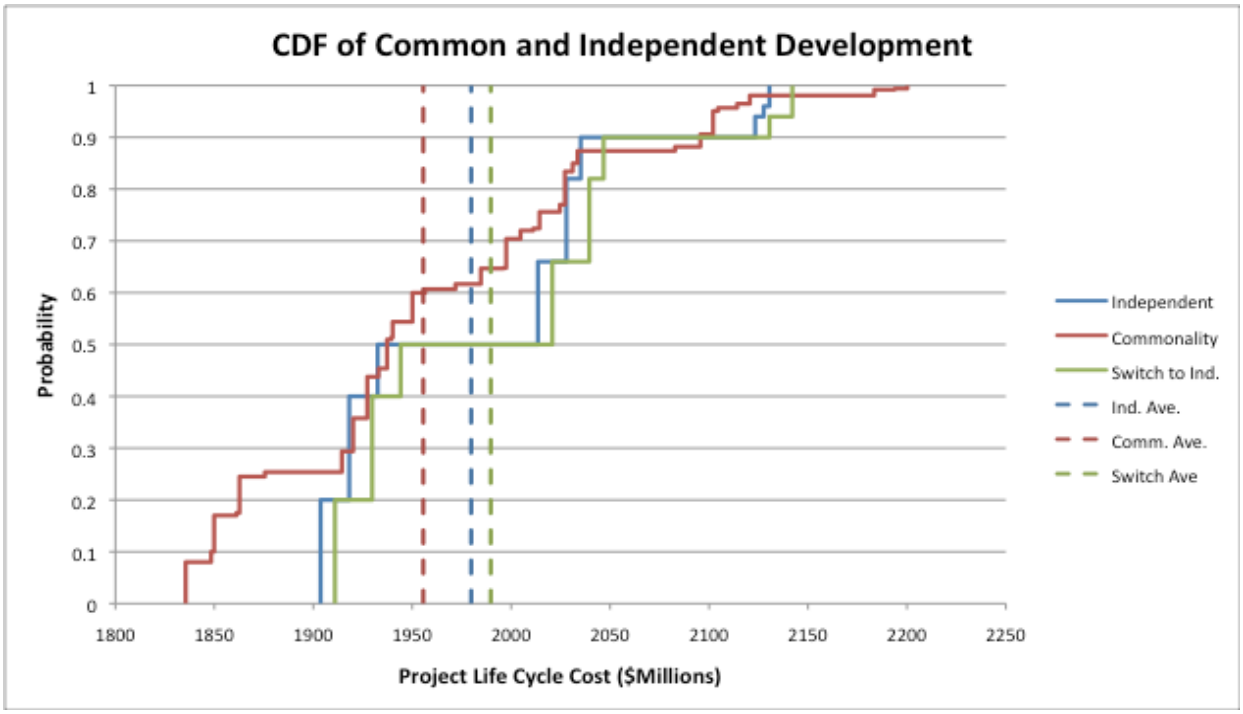

FIGURE 14: CDF OF COMMON AND INDEPENDENT DEVELOPMENT

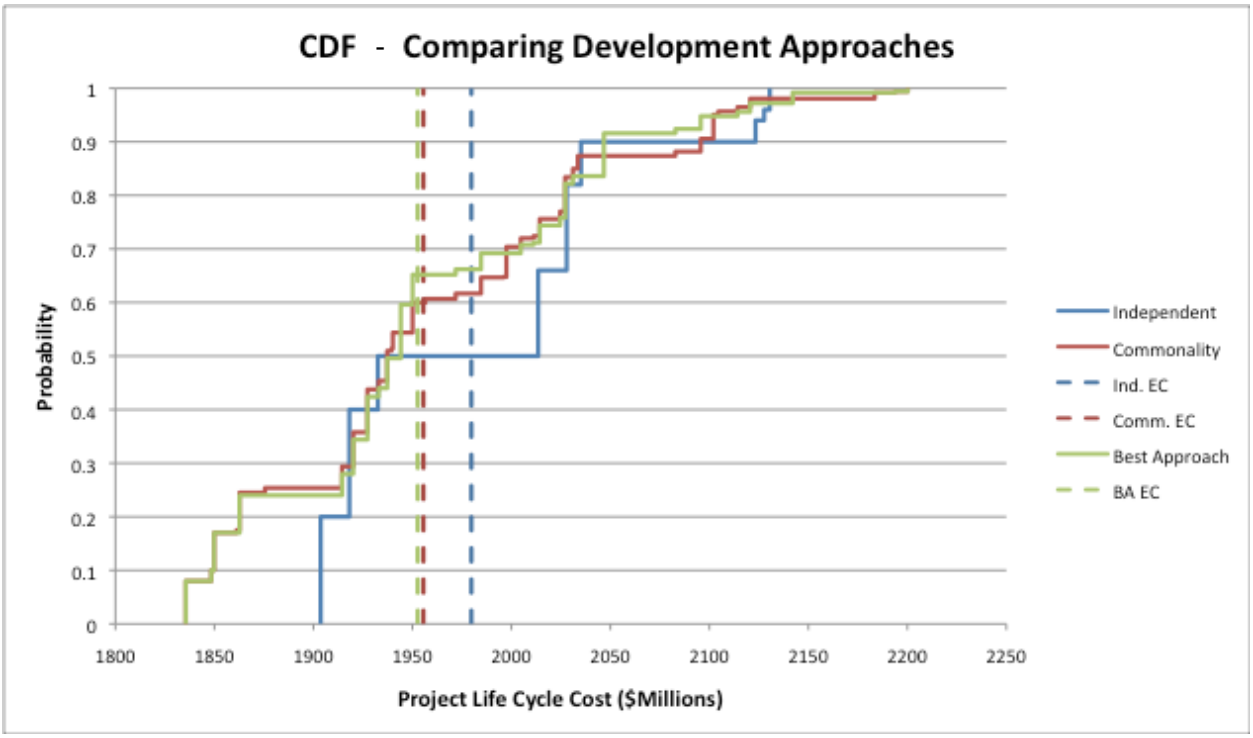

FIGURE 15: CDF SHOWING THE BEST APPROACH IS NOT PURELY COMMON OR PURELY INDEPENDENT

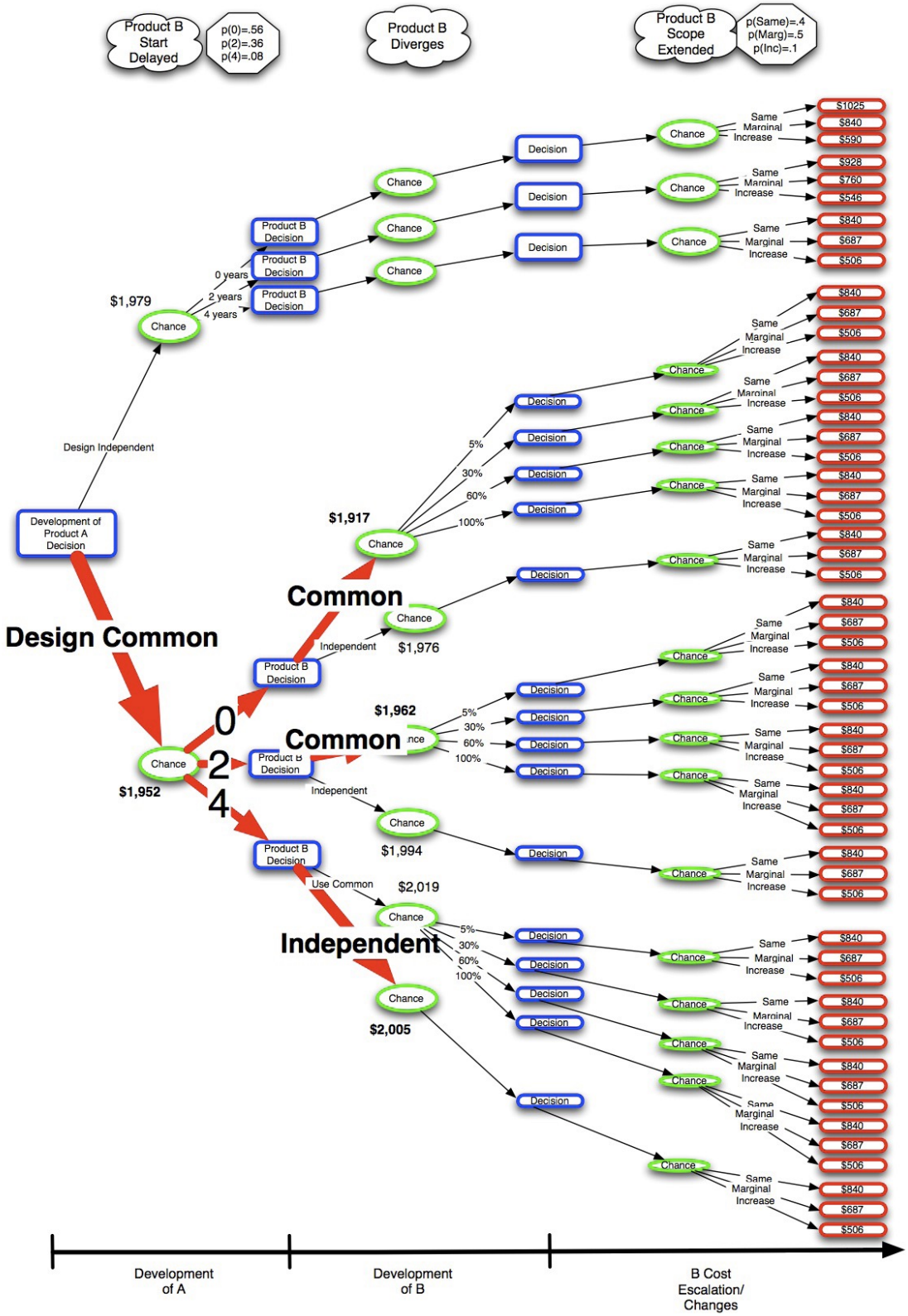

FIGURE 16: DECISION TREE SHOWING PREFERRED PATHS AFTER MEASURING EXPECTED VALUES

3.5. THE VALUE OF FLEXIBILITY

The analysis from the sample case on CSSS showed that there was benefit in keeping the flexibility in the system. The best results achievable with the expected value approach were marginally superior to the best results from sticking rigidly to commonality, and much better than choosing independent design at all times.

The flexibility to move from common design to independent design can be valuable, and this option should be preserved in project planning.

Therefore the flexibility to change between common development and independent development is important. Initial contract negotiations should not lock the project developers into mandatory common development, and there should be options to review the amount of commonality.

3.6. CONCLUSIONS

The first step in evaluating whether commonality is the right strategy is to perform a lifecycle cost analysis. The cost model presented at the beginning of this chapter offers one way to collect relevant information about commonality and include it in the analysis. With the cost information, decision analysis can be carried out to determine optimal development strategies.

Cost models, decision trees and value-at-risk graphs are all tools for demonstrating the effectiveness of commonality strategies. The tools can be used to complement the technical analysis of Chapter 2. The tools are also essential for logical demonstrations to project teams and external stakeholders of the potential benefits of commonality.

CHAPTER 4: STRATEGIES TO BETTER MANAGE PROJECTS INVOLVING COMMONALITY

4.1. INTRODUCTION

This section of the commonality toolkit focuses on the management techniques which can maximize beneficial commonality. These management heuristics are drawn from the case studies examined and also reflect some of the learning about commonality theory developed in Chapter 1.

4.2. IMPLEMENT COMMONALITY FROM THE START OF THE PROJECT

Commonality must be considered as early as possible in the design of the whole product family. Emphasizing commonality before any design decisions are locked in gives the greatest range of potential commonality opportunities to choose from. Contract structure, incentive and reporting structure, team interactions and senior management buy-in are all important influences on commonality and are most easily implemented at project inception.

Identify commonality opportunities early: Maintaining commonality requires strong management; rolling back independent designs to implement commonality is probably beyond even exceptional managers.

The identification of potential forward common components should take place early in the development process in order to maximize the potential benefits from commonality. As time progresses, the ability to influence development and costs decreases, and the chances of actually implementing commonality also decrease. The time at which commonality opportunities are identified affects the potential benefits. The method developed by Hofstetter summarized in Chapter 2 offers a simple way of analyzing commonality benefits very early in the architecting process.

4.3. OFFSET VERSUS PARALLEL PROJECTS

As briefly discussed in Chapter 1, when designing two products to be common the basic question arises as to whether to use offset or parallel development.

With NASA projects, almost invariably the development projects will be offset. Often the budget is not available to work on the entire family simultaneously. Some projects require specialists who cannot work on both projects at once, or particular design, manufacture or test facilities. Smoothing labor resources is also important. Perhaps both projects could be designed at once, but then there would be a lumpy project flow through the manufacture and test pipeline, which would involve hiring and laying off people as the project matured. Finally, many NASA projects involve R&D risk. A common method to reduce R&D risk is to fully design and evaluate one variant before completing the design of the other. The decision to offset the development of the Ares I and Ares V launch vehicles in the Constellation project could probably be traced in some part to all of these reasons. Arguably commonality is easier to implement in parallel development, but offset development is a reality and commonality can also be applied to offset projects.

4.4. Determining the order of offset products

Offset projects pose a problem for developing commonality: how do you develop a system to be common between two products at very different stages of design? This is an additional complication to the problem of identifying commonality analyzed in Chapter 2 regarding identification of feasible opportunities for commonality. The product first in time has better knowledge of its requirements and design constraints, so any common system developed by that team is (even without any inherent bias) likely to be better developed for product A. Implementing commonality in offset projects requires greater attention and discipline from management than parallel projects.

In offset design, always begin with the highest volume product to ensure that subsequent products have maximum visible incentive to maintain commonality.

One way identified in our research of incentivizing a full analysis before common systems are abandoned is to always design the lower volume product second. The reason is that the benefits of the economies of scale achieved by the lower volume product tagging along with the higher volume product are much greater. If the lower volume product were built first, the higher volume product sees smaller benefits from maintaining commonality. This goes directly to the manufacturing cost of the lower volume product, a cost that managers of the second project are more likely to take account of during design. Other benefits like logistical simplicity or operational learning are less likely to be "internalized" by the managers of the second project when making a decision as to whether to pursue commonality or not.

As an example, suppose the team developing product A knows that there is potential commonality between the life support system in product A and product B. They may also know that product A needs to have an additional 30 liters per hour oxygen production over what would be required for product B uniquely. What they do not really know are the details of interface and constraint for product B that may lead to the "common" system being unusable by product B. For example, a previously unknown vibration mode might be discovered in the finite element analysis of Product B that requires structural mass to be added; consequently every other system is ordered to shed mass. The overdesigned life support system might be one of the first areas that the mass could be removed from, and the commonality benefits would be lost. Remember, commonality is not an end in itself, so it may be that the life support system is the right place to remove the mass from. However, this shouldn't be done without considering the full life cycle impact of the decision and the other, less obvious, places to save mass.

4.5. Accelerate common portions of the design where possible

Designing effective common systems can be difficult because of the uncertainty about future systems. In working with seven cases of offset design, Boas observed that *"future variants are at best staffed with conceptual teams"*, who have less detailed analysis with which to support the performance requirements of the future variants. Consequently, the common system is biased towards the needs of the system first in time. If human and financial capital allows, accelerating the design of the intended common systems of future variants can result in a common system which better meets the needs of

all variants and is less likely to diverge in future. This technique was successfully used on the International Standard Payload Rack, a case study more fully described by Rhodes.

Accelerating development of common portions of design can give the commonality benefits of parallel development in a largely offset project.

4.6. ORGANIZE TEAMS BY DISCIPLINE, RATHER THAN BY PRODUCT

Offset projects do have a significant advantage over parallel projects. The same design teams can work on the same systems for each variant of the project. This is one of the best ways of giving the teams visibility across previous work and implementing commonality. Commonality is most likely to be possible within systems developed by teams from the same disciplines, because this indicates technological commonality.

As an example, this was a lesson successfully learned between the Apollo program and the Constellation program. In the 1960s, the nine swing arms connecting the launch umbilical tower to the Saturn rocket and Apollo capsule were designed by separate teams as separate products. For Constellation, the analogous four Tilt Up Umbilicals were designed by teams on a matrixed structure, with cross-cutting lead design engineers. Quinn describes this approach as leading to additional commonality and uniformity between the swing arms (Quinn 2008).

4.7. ESTABLISH MANAGEMENT SUPPORT FOR COMMONALITY

Management support for commonality from the outset is very important in order to encourage system engineers to seek out opportunities for commonality. The effort of identifying and analyzing commonality is an up-front cost that is only recovered over the lifecycle of the product family. Incentives for system engineers, whether formally in terms of remuneration or informally in terms of promotion prospects, perceived success, or respect, are usually based on cost or schedule performance. Strong commonality leadership from management, especially in recognizing the important role commonality plays in successful products that are cost effective in the long term, is required to create an environment where efforts to search for beneficial commonality are encouraged.

Establish ongoing, high level support for commonality beginning at project inception.

An up-front effort must be made in order to seek out opportunities for commonality and to eventually implement those opportunities. Project managers in the development stages will only rarely put themselves at risk of going over budget or overdue in order to apply commonality. Therefore management support is needed in order to promote the identification process. Strong management support for the development of commonality was observed in all three of the NASA case studies. Further, at the outset of the project it is not usually clear how a commonality plan will affect the design teams, and it is likely to be easier to get agreement on general points of commonality. Later in the design process implementing commonality will clearly mean additional work for particular teams which will make general agreement more difficult.

Ongoing management support for commonality is also critical. Almost all development projects

will be under cost, schedule or performance pressure. Each of these pressures can be temporarily alleviated by sacrificing commonality. Usually management has control over this decision, either directly or through the plans and incentives it puts in place. Making management aware from the outset of the potential benefits of commonality and the necessity to evaluate decisions over the full lifecycle of the project is essential to managing divergence. The tools presented in Chapter 3 can help to make the case for commonality supportable, and establish a rational framework to decide when to diverge from commonality.

Two practical methods for encouraging management support for commonality are:

- create a central point of responsibility for commonality at the outset of the project with the authority to plan, evaluate and implement beneficial commonality; and
- create a formal process for approving divergence which involves senior management.

Early design of the International Space Station recognized these approaches were essential to commonality. The Space Station Commonality Program developed through the 1980s stated *"All SSP items shall be common unless approval has been granted to make the items unique, or unless waivers or deviations have been granted by the Systems Integration Board…"*

4.8. CREATE INCENTIVES FOR COMMONALITY

Careful structuring of incentives for commonality can help lower life-cycle costs for the product family. Rewarding beneficial commonality within a company, or developing contract structures that properly incentivize contractors to develop product solutions that minimize cost over the whole product lifecycle is important.

The difficulty in assessing whether commonality is beneficial at the time the common system is developed makes it difficult to create quantitative incentive structures for commonality. Rewarding commonality in itself can be counterproductive because it discourages divergence even in cases where the divergence is beneficial. This is an ongoing area of research at MIT.

Incentivize minimum lifecycle cost over the entire product family, not upfront cost for a particular variant. This will encourage commonality development.

A very simple tool which focuses thinking on total lifecycle costs for the product family in order to manage divergence is to produce a table like the following. The example is for a three variant family. The table sets out the relative cost impact of a change to a common component based on the full life cycle costs of all variants. This case is intended to simulate the use of a smaller, simpler component in Variant 1 and 2, with the original component continuing to be used in Variant 3.

TABLE 9: EXAMPLE MATRIX FOR IDENTIFYING FULL LIFECYCLE COSTS OF TECHNICAL CHANGES

Product Lifecycle Phase	Variant 1	Variant 2	Variant 3
Performance Impact (could be sub-divided or quantified)	Positive	Positive	No change
Development	-$0.5m	-$0.5m	+$1.2m
Manufacture	-$0.2m	-$0.2m	+$2.0m
Operations	+$4.0m	+$3.2m	+10.0m
Total cost impact	**+$3.7m**	**+$3.1m**	**+$13.2m**

In his study of the Joint Strike Fighter (JSF) program, Boas consistently refers to the creation of a "commonality culture" in Lockheed Martin's JSF team. There are a number of technical and management strategies that seem to contribute to this. One is *"the 8-Square template which requires the lifecycle costs of a change to a common component to be considered from the perspective of each variant."*

The significance of this table is that if it becomes institutionally required to evaluate the cost impact of every change to commonality in this form, then thinking about the full lifecycle costs across all variants becomes more widespread. The designer proposing the common change must then explain why the positive performance increase across variants 1 and 2 justifies the increased cost. There is no reason why such a change could not be justified, however it must be justified on the basis of the full data provided in Table 9 – not by a single variant view.

APPENDIX A: SOME SUBTLETIES AROUND PORTFOLIO DEFINITION FOR ARCHITECTING

ASSESSING CUSTOMER NEEDS

The needs of the customer and the common function should be expressed in solution neutral language.[3] This may help to bring products into the portfolio that would otherwise have been overlooked.

Often, the future needs of the customer are uncertain. The heritage analysis presented later in this chapter is one way of determining uncertain future needs.

COMMON FUNCTIONALITY

A further limitation is the need for a common function. In some cases, all products share all functions, and the differentiator between the products is how extensively they perform those functions. For example, a family of launch vehicles all share a common externally delivered function of delivering payload to orbit or escape; they all share internal functions as well like guidance, propulsion, structural support and payload release. These functions will deliver a different scale of performance for different variants. For example, a heavy lift rocket will have a more powerful propulsion system than a lighter class of rocket. This is the classic platforming approach, often seen in consumer products, where some systems remain identical, and other systems have additional parts added or existing components enlarged, extended or made more capable, to deliver the higher performance.

Alternatively, the variants may not share all functions. There may only be one common system between the projects. An example of this type of portfolio would be the set of Environmental Conditioning and Life Support Systems (ECLSS) on space vehicles. The ISS, Earth escape vehicles and lunar rovers all have very different externally delivered functions, and many internal functions are different. However, one function which all of these systems share is human life-support. Because of that shared functionality, these systems are suitable candidates to be examined as a portfolio.

Importantly, the object of the commonality exercise is not just to match systems with the potential for identical systems but to match up systems with potential for commonality between systems. The difference is subtle but important: we are not trying to find systems which have a chance of being the same, we are trying to find systems with a chance of having sub-systems or key components that could be similar.

INCLUDING LEGACY SYSTEMS

The portfolio should also include all current and some legacy systems. Including current architectures ensures that all opportunities for re-using existing systems are considered. Reuse can be an effective way of capturing some commonality benefits with little up-front expense. In some circumstances it may be wise to include some legacy systems. Legacy

[3] For a more generalized discussion of the form and purpose of solution neutral language see, for example, Rechtin and Maier's book, "The Art of System Architecting".

systems are systems which are no longer used, but for which significant information already exists. Examples include re-using a design to reduce design costs, re-using a standard diameter tank to take advantage of existing manufacturing facilities, re-using surplus hardware from the legacy product and re-using a product for which the testing or operational performance is well understood. Care must be taken to not include legacy systems which have become so obsolete that information is so out of date or facilities so disused that there are not significant savings in reuse. This is an area for engineering discretion during the portfolio selection.

NARROWING PROJECTS TO THOSE CONTROLLED BY THE CUSTOMER

In framing the portfolio there is a tradeoff between casting the portfolio as widely as possible to capture many commonality possibilities, and narrowing the portfolio to reduce the computational burden of evaluating all the different designs. In practical terms, there is no point including in the commonality analysis projects future projects are a few rules for limiting the projects that should be included:

1. For future projects, the organization must have
    a. a strong interest in keeping the cost, schedule or risk of the project low
    b. decision making power as to the systems that comprise the project
    c. ability to influence the project management of that project as it is developed
2. For existing or historical projects, the organization must have good knowledge of that project, usually through developing it itself.[4]

It would be unrealistic, for example, to include a planned or existing Ariane rocket in a US launch vehicle analysis because it would fail on both rules.

LIKELIHOOD OF DEVELOPMENT

It is not necessary that future variants are almost certain to be developed. In fact, the speed and simplicity of this portfolio architecting approach makes it easy to consider even unlikely products for possible commonality. However, in each case the economic consequences and probabilities of common design against independent design of the systems must be evaluated using the tools in Chapter 3. This is particularly if there are unlikely variants in the portfolio, because the economic case for common development is less strong if future variants are less likely. Using the technical analysis in this section alone will not pick that up.

OVERLAPPING PORTFOLIO ANALYSES

One project could be part of multiple portfolio analyses. A human-rated Mars rover could form part of the portfolio described above for evaluating life support systems, and form part of the portfolio of human and robotic surface vehicles when looking for commonality for surface drive systems for example, as shown in Figure 17.

[4] There may be some exceptions to this, such as where a very modular part such as a solid rocket booster can be purchased directly. In that case the organization would need to have less knowledge of the common part.

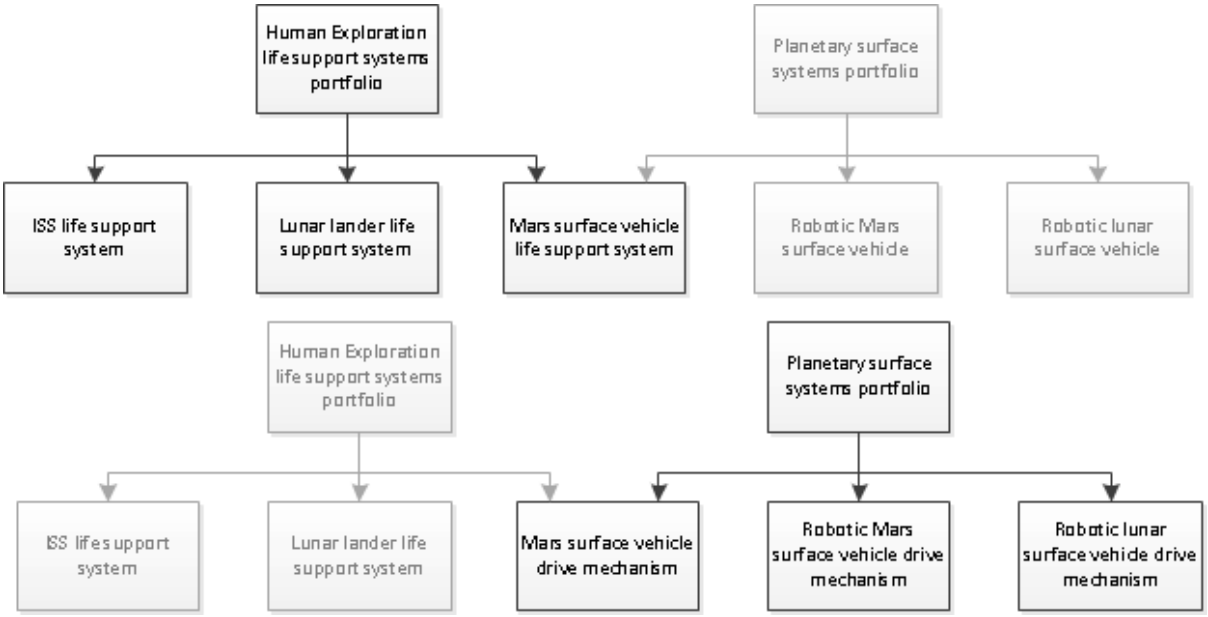

FIGURE 17: THE SAME END PRODUCT CAN BE INCLUDED IN DIFFERENT COMMONALITY ANALYSES DEPENDING ON THE SYSTEM BEING CONSIDERED

APPENDIX B: USING MATRICES TO SPEED COMMONALITY SORTING

The analysis of the commonality choices for Step 3 of the Portfolio Architecting method was outlined in principle in Chapter 2 of this toolkit. This Appendix provides more detail with regard to the mechanics of efficiently running variants through the four commonality filters.

CONCEPT DESCRIPTION MATRIX

To implement the four rules described in Step 3 of the Portfolio Architecting method, a system of overlapping matrices can be used. First, each possible architecture for a variant can be described in matrix form as a "Concept Description Matrix" (CDM).

The CDM shows the technology choice for each internal function. It also shows the environments in which that technology must operate for that particular architecture. Ones and zeros could be used for encoding since that usually assists in the programming.

Each CDM represents a design alternative for a project. Rules 1 to 3 can be assessed by overlapping CDMs, producing a combined matrix of the same dimensions called a "System Overlap Matrix".

Each SOM is generated by pair wise comparing each CDM with each other CDM from the set of interesting project architectures. Commonality opportunities occur when the number of environments in which there is overlap for a given technology choice and function, divided by the total number of environments for which there overlap, is greater than the threshold value chosen by the team conducting the commonality analysis. Effectively, this is the number of twos for a given function, divided by the total of the number of ones and the number of twos. Figure 18 shows this in more detail. If that ratio is greater than the threshold, then commonality is possible, though not certain. The commonality screening must then pass rule 4.

Rule 4 can be assessed using the same CDM framework with pair wise comparison. The rule is that the parameter which sizes the system must be within a certain overlap parameter of the same metric on the other CDM in order for commonality to be possible. The overlap parameter could be varied between different systems.

In Table 10, only Internal Functions 1, 3 and 4 are considered, since these were the only functions to pass the common technology screening shown in Figure 18. Further, only 3 environments are shown for simplicity. One parameter is used to size each of the internal parameters (it would be possible to use multiple parameters if necessary).

There is complete overlap in the parameters for internal function 1 for variants 1 and 2. At any values of the overlap parameter, these two systems could be made common. Internal function 3 could only be made common if the overlap factor was 1.2 or greater. Internal function 4 could only be made common if the overlap factor was set at 3.3 or greater.

Higher values of the overlap factor lead to more systems being made common; higher values also indicate more design effort required to produce the common system or more under-optimization of the performance of each variant, or both. Sensitivity analysis can be used to analyze how sensitive the preferred solution is to values of the overlap factor. Solutions which are robust to changes in the overlap factor should be

preferred, but ultimately the decision as to an acceptable value of the overlap factor remains with the system architect.

Concept Description Matrix for System 1

	Operational environments	Environment 1	Environment 2	Environment 3	Environment 4	Environment 5	Environment 6	Environment 7	Environment 8
Internal function 1	Technology choice 1	1	1	1	1	1	1		
	Technology choice 2								
	Technology choice 3								
	Technology choice 4								
	Technology choice 5								
Internal function 2	Technology choice 1	1	1	1	1	1	1		
	Technology choice 2								
	Technology choice 3								
	Technology choice 4								
	Technology choice 5								
Internal function 3	Technology choice 1								
	Technology choice 2	1	1	1	1	1	1		
	Technology choice 3								
	Technology choice 4								
	Technology choice 5								
Internal function 4	Technology choice 1								
	Technology choice 2								
	Technology choice 3								
	Technology choice 4	1	1	1	1	1	1		
	Technology choice 5								

Concept Description Matrix for System 2

	Operational environments	Environment 1	Environment 2	Environment 3	Environment 4	Environment 5	Environment 6	Environment 7	Environment 8
Internal function 1	Technology choice 1	1		1	1	1	1		1
	Technology choice 2								
	Technology choice 3								
	Technology choice 4								
	Technology choice 5								
Internal function 2	Technology choice 1								
	Technology choice 2								
	Technology choice 3								
	Technology choice 4	1		1	1	1	1		1
	Technology choice 5								
Internal function 3	Technology choice 1								
	Technology choice 2	1		1	1	1	1		1
	Technology choice 3								
	Technology choice 4								
	Technology choice 5								
Internal function 4	Technology choice 1								
	Technology choice 2								
	Technology choice 3								
	Technology choice 4	1		1	1	1	1		1
	Technology choice 5								

System Overlap Matrix for System 1 and System 2 (overlapping entries highlighted in red)

	Operational environments	Environment 1	Environment 2	Environment 3	Environment 4	Environment 5	Environment 6	Environment 7	Environment 8
Internal function 1	Technology choice 1	2	1	2	2	2	2	0	1
	Technology choice 2	0	0	0	0	0	0	0	0
	Technology choice 3	0	0	0	0	0	0	0	0
	Technology choice 4	0	0	0	0	0	0	0	0
	Technology choice 5	0	0	0	0	0	0	0	0
Internal function 2	Technology choice 1	1	1	1	1	1	1	0	0
	Technology choice 2	0	0	0	0	0	0	0	0
	Technology choice 3	0	0	0	0	0	0	0	0
	Technology choice 4	1	0	1	1	1	1	0	1
	Technology choice 5	0	0	0	0	0	0	0	0
Internal function 3	Technology choice 1	0	0	0	0	0	0	0	0
	Technology choice 2	2	1	2	2	2	2	0	1
	Technology choice 3	0	0	0	0	0	0	0	0
	Technology choice 4	0	0	0	0	0	0	0	0
	Technology choice 5	0	0	0	0	0	0	0	0
Internal function 4	Technology choice 1	0	0	0	0	0	0	0	0
	Technology choice 2	0	0	0	0	0	0	0	0
	Technology choice 3	0	0	0	0	0	0	0	0
	Technology choice 4	2	1	2	2	2	2	0	1
	Technology choice 5	0	0	0	0	0	0	0	0

FIGURE 18: CONCEPT DESCRIPTION MATRICES COMBINE TO GIVE A SYSTEM OVERLAP MATRIX

TABLE 10: SHORT CDM SHOWING PARAMETER VALUES

VARIANT ONE			Environment 1	Environment 2	Environment 3	Parameter Limiting Extreme
Internal Function 1	Technology Choice 1	Parameter 1 (eg Oxidiser throughput)	1000 L/s	0	0	1000 L/s
Internal Function 3	Technology Choice 4	Parameter 1 (eg Specific Impulse)	0	300s	300s	300s
Internal Function 4	Technology Choice 2	Parameter 1 (eg Maximum operating temperature)	1400K	1000K	1000K	1000K

VARIANT TWO			Environment 1	Environment 2	Environment 3	Parameter Limiting Extreme
Internal Function 1	Technology Choice 1	Parameter 1 (Oxidiser throughput)	1000 L/s	0	0	1000L/s
Internal Function 3	Technology Choice 4	Parameter 1 (Specific Impulse)	0	250s	250s	250s
Internal Function 4	Technology Choice 2	Parameter 1 (Maximum permissible temperature)	1400K	1700K	300K	300K

APPENDIX C: CASES STUDIED

TABLE 11: SUMMARY OF THE CASES STUDIED AND THE LEARNING DEVELOPED

	Constellation Space Suit System (CSSS)	**International Standard Payload Rack (ISPR)**	**Core Flight-Software System (CFS)**	**Previous Industry Cases**
Program	Constellation Program	International Space Station	Unmanned Scientific Spacecraft	4 Aerosapce, 3 Others
Product Type	Hardware	Service	Software	Hardware
Development Type	Contracted/External	In-house	In-house	In-house development
Organizational Structure	Single Project	International Peer Organizations	Single Project	Single Project
Life Cycle Offsets	Yes	Yes	Yes	Yes
Divergence	Yes	Yes	Yes	Yes
Management Approach	Widespread Forward Commonality	Common Building Block	Widespread Forward Commonality	Prodominately Reactive Reuse
Culture	Family, two configurations for the same common suit	A single standard system to be reused	A common core with common application modules to be continuously maintained and reused	Single-product culture, with large attempts to transition to forward commonality
Identify				
Pursuit of Forward Commonality	Yes, goal was to create two highly common configurations of a single suit system	Yes, goal was to create international standardization of payload systems	Yes, goal was to create a flight software system that can be applied to all systems	Mixed effort for forward comon
Identification Method	Analyzed each expected CEI and baselined if feasible	Identified as a result of utilization rights	Identification resulted from increased competition for satellite development	Previous system was baseline for future development
Management Methods	- Timing - Program Support - Assign Responsibility - Increased Visibility - Incentives	- Timing	- Management Support - Assign Responsibility - Heritage Analysis - Increased Visibility - Incentives	Varied
Evaluate				
Awareness of potential commonality benefits	Yes	Yes	Yes	Yes
Formal analysis of commonality opportunities	Yes, trade studies were conducted for each potential change. Economic analysis was limited.	Limited, goal was to efficiently utilize payloads on international laboratories with limited logistics onground and inspace	Limited, heritage analysis was used to inform decisions made by the deliberation of experts. Economic analysis was qualitatively estimated based on time.	Limited
Implement				
Formal management of commonality	Yes, maximize commonality in ininitial architecture and control divergence	Yes, negotiation and formalization of standards	Yes, CFS is continuously updated and maintained from feedback for each satellite	Informal management
Formal owner of commonality	Yes, Architecture and Analysis Group	Yes, Interface Control Working Group	Yes, CFS development team	Usually at executive level
Control Process for Divergence	Mult-tier approval process for changes to the baseline architecture	All changes must be presented and negotiated at ICWG	CFS is maintained by an independent development group	Limited
Management Methods	- Flexibility - Open Architecture - Standardization - Avoid Proprietary Restrictions	- Flexibility - Compromise - Standardization - Accelerated Development	- Flexibility - Traceability - Self-sustaining	Varied

SELECTED REFERENCES

Boas, R.C., 2008. *Commonality in Complex Product Families: Implications of Divergence and Lifecycle Offsets*,

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

Hofstetter, W.K., 2009. *A Framework for the Architecting of Aerospace Systems Portfolios with Commonality*, Massachusetts Institute of Technology.

Neely, J.E. & de Neufville, R., 2001. Hybrid real options valuation of risky product development projects. *International Journal of Technology, Policy and Management*, 1, 29-46.

de Neufville, R., 1990. *Applied Systems Analysis: Engineering Planning and Technology Management*, McGraw-Hill.

Newman, D., Eschenbach, T. & Lavelle, J., 2004. *Engineering Economic Analysis*, New York: Oxford University Press.

Quinn, S.M., 2008. *Commonality of Ground Systems in Launch Operations*, Massachusetts Institute of Technology.

Rhodes, R., 2010. *Application and management of commonality within NASA systems*, Massachusetts Institute of Technology.

Simpson, T. & Thevenot, H., 2004. A Comparison of Commonality Indices for Product Family Design. *Proceedings of DETC04 ASME 2004 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, DETC2004/DAC-57141.

Wertz, J. & Larson, W., 1999. *Space Mission Analysis and Design* Third edition., Space Technology Library.

Willcox, K. & Wakayama, S., 2002. Simultaneous Optimization of a Multiple Aircraft Family. *American Institute of Aeronautics and Astronautics*.

Yeager, D., 1987. Expert System Development for Commonality Analysis in Space Programs.