THE NECESSITY OF FUNCTIONAL ANALYSIS FOR SPACE EXPLORATION PROGRAMS

A. Terry Morris, NASA Langley Research Center, Hampton, Virginia 23681

Julian C. Breidenthal, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California 91109

Abstract

As NASA moves toward expanded commercial spaceflight within its human exploration capability, there is increased emphasis on how to allocate responsibilities between government and commercial organizations to achieve coordinated program objectives. The practice of program-level functional analysis offers an opportunity for improved understanding of collaborative functions among heterogeneous partners. Functional analysis is contrasted with the physical analysis more commonly done at the program level, and is shown to provide theoretical performance, risk, and safety advantages beneficial to a government-commercial partnership. Performance advantages include faster convergence to acceptable system solutions; discovery of superior solutions with higher commonality, greater simplicity and greater parallelism by substituting functional for physical redundancy to achieve robustness and safety goals; and greater organizational cohesion around program objectives. Risk advantages include avoidance of rework by revelation of some kinds of architectural and contractual mismatches before systems are specified, designed, constructed, or integrated; avoidance of cost and schedule growth by more complete and precise specifications of cost and schedule estimates; and higher likelihood of successful integration on the first try. Safety advantages include effective delineation of must-work and must-not-work functions for integrated hazard analysis, the ability to formally demonstrate completeness of safety analyses, and provably correct logic for certification of flight readiness. The key mechanism for realizing these benefits is the development of an inter-functional architecture at the program level, which reveals relationships between top-level system requirements that would otherwise be invisible using only a physical architecture. This paper describes the advantages and pitfalls of functional analysis as a means of coordinating the actions of large heterogeneous organizations for space exploration programs.

I. Introduction

After the termination of the Constellation program (CxP) and subsequent to the retirement of the Space Shuttle program (SSP), the United States intends to move in a new direction characterized by increasing commercial interaction and participation in space exploration with the goal of sustained human presence in low Earth orbit (LEO) and beyond. NASA's Exploration Systems Mission Directorate (ESMD) is tasked with developing the systems and capabilities required to enable affordable commercial crew access to the International Space Station (ISS) and to launch crewed vehicles for missions beyond LEO. The primary aim of increased commercial spaceflight to LEO will be the development and operation of vehicles that could become the nation's primary means of ISS crew transportation thus reducing America's reliance on foreign systems [1]. This new direction to achieve affordable space exploration capabilities mandates the need to perform systems integration correctly, particularly among government and commercial entities that are heterogeneous by nature with sometimes competing goals. In its initial planning, ESMD has already identified three major risks for commercial crew capabilities (see Table 1, first three rows): failure of a commercial partner, uncertainty regarding emerging commercial market demand and requirements unique to NASA [1].

The purpose of this paper is to argue the need for establishing functional architectures as a means of weaving and integrating the goals and requirements of public-private space exploration partnerships. Functional analysis, as expressed in this paper, can be seen as a critical mitigation approach for the risk of NASA-unique requirements (see Table 1, row three) and for the risk of inadequate program integration of government and commercial crew (see Table 1, row four).

Table 1. NASA Programmatic Risks Concerning Commercial Crew

Risk Title	*Risk Statement*
Failure of a Commercial Partner	Commercial partners may not be able to complete the demonstration phase and thus NASA's investment would not result in available commercial services.
Uncertainty Regarding Emerging Commercial Market Demand	With a minimum of only two flights per year from NASA and an uncertain non-NASA market, potential providers may be wary of the commercial business potential.
Requirements Unique to NASA	NASA-unique requirements will increase the cost to provide services such that the commercial providers may not be able to capture non-NASA markets.
Inadequate Program Integration of Government and Commercial Crew Products	Program management may not integrate divergent government and commercial interests resulting in increased costs, inefficiencies and/or program termination.

II. Background

A. Managing the Interfaces, Interactions and Influences via Effective System Integration

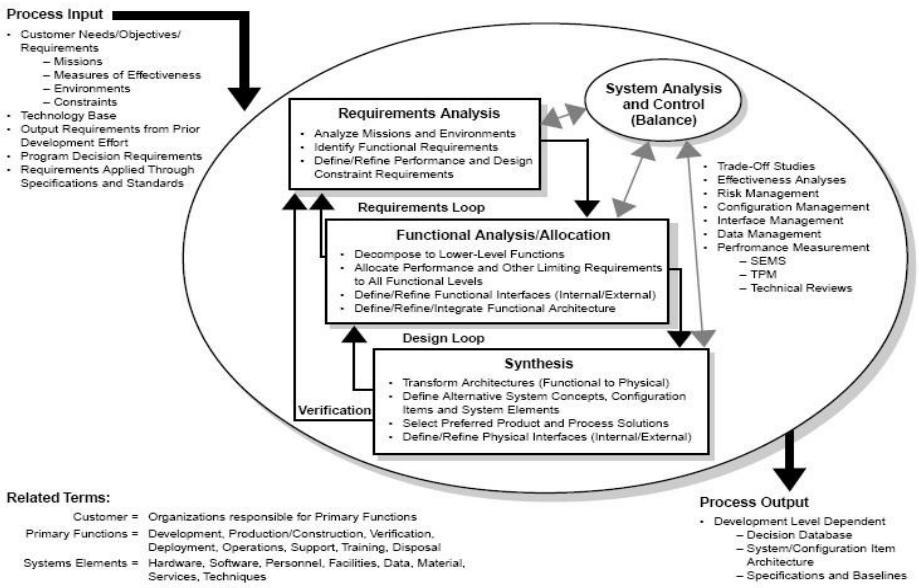
System integration, in general, involves managing the interfaces, interactions and influences of the players and components of a system in order to achieve the end product or mission. Integration management, in the context of large scale systems, is an arduous and time-consuming process, particularly between heterogeneous stakeholders with divergent agendas. We assert that both good program management and effective system integration are fundamentally necessary to achieve commercial crew exploration goals within funding and timing constraints. This point is often overlooked because many managers looking to save costs believe that physically independent or (physically) loosely coupled systems do not require system integration effort. The purpose of good program management is to direct the program toward program success. The purpose of effective system integration is the success of the end product. Broadly speaking, system integration is the process of bringing together the component subsystems into one system and ensuring that the subsystems function together as a coordinated whole. Simply stated, system integration is more than system assembly. A foundational tool in the system integration arsenal is functional analysis which can be used at both the programmatic system level and the lower project subsystem levels. In this paper, we will use the term "system" to refer to the coordinated product at the program level and the term "subsystem" to refer to project-level products. Depending on the scope of the project, its subsystem can be very large and can be sub-divided into multiple lower-level subsystems. As a high level risk mitigation tool, the authors intend to focus primarily on the programmatic benefits of functional analysis.

B. What is Functional Analysis?

The goal of the systems design process is to transform explicit and latent requirements into a coherent description of system functions that can be used to guide a design (Figure 1). In this endeavor it is very common to decompose a system into smaller pieces that can be more easily understood. Those smaller pieces are divisions of the system using a separation criterion: typically spatial[1] in the case of hardware or facilities, administrative in the case of organizations, or in the case of software some other criterion such as boundaries of configuration items. From this division arises the concept of interfaces, regions of interaction between the decomposed pieces.

In principle there are as many possible separation criteria as there are distinctions within human thought. We will concentrate here on just two commonly used criteria: spatial and functional. Choosing to decompose a system spatially leads to one particular kind of system architecture, while choosing to decompose a system functionally leads to another, as sketched in Figure 2. Each of these potential decompositions is both useful, and by itself, inadequate. The spatial decomposition has the ability to reveal spatial relationships, and is convenient for organizing characteristics that have a clear relationship to spatial boundaries such as mass, temperature, or energy flow. The functional decomposition reveals functional relationships, and is

[1]Footnote 1 The term "physical" is a common substitution for "spatial", though strictly speaking there are other physical distinctions that might be used as criteria for decomposition such as time, density, temperature, etc.



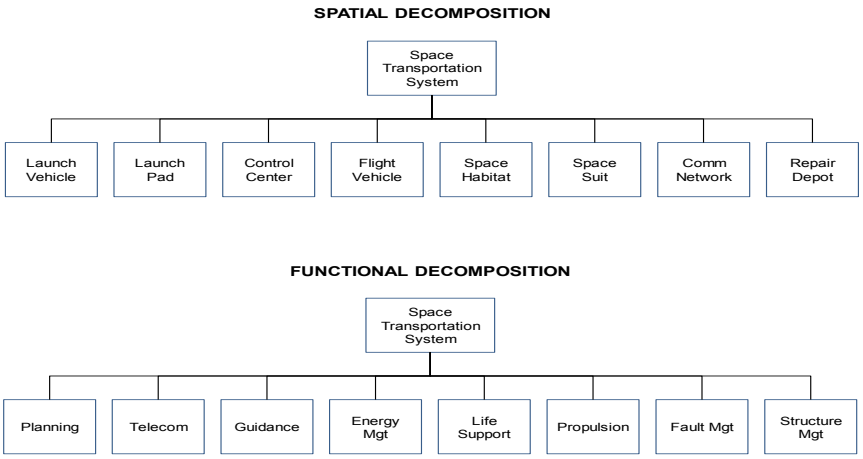


Figure 2. Two Decompositions of the Same Space Transportation System

convenient for organizing characteristics that have a clear relationship to functional boundaries such as computational loading, response time, or data volume. However, neither decomposition type can reveal the relationships accessible using the other. To the extent that success of a system depends on management of spatial and functional characteristics at the same time, both decompositions are necessary. Functional analysis, then, is a systematic process of, decomposing, describing and relating the functions a system must perform in order to achieve end product success. It is important to note that functional analysis at the top program-level probably should not address how these functions will be performed. This is because large scale systems generally have multiple levels of planning, coordination and implementation. The program level utilizes requirements to determine what it wants. These requirements describe the "what". In many cases, the requirements are provided to other organizations (contractors, vendors, integrators, etc.) who compete to determine the "how to build the what". There are generally multiple design implementations that can satisfy requirements. Good requirements attempt not

to be overly prescriptive so as not to restrict the design space. And since the set of core high-level functions are extracted and derived from these high-level requirements, they, too, should describe "what actions are required" and not "how to do the actions".

Skilled systems engineers will try to select physical subsystems that align well with functions, but there are limits to how completely this can be done because functions usually overlap each other when mapped to physical subsystems. This invariably forces a compromise in which some functions must be split between subsystems. It is these cross-subsystem functions that are amenable to functional analysis. Functions that truly can be contained completely within one subsystem should be left for that subsystem to manage on their own.

In the early phases of the program life cycle, functional analysis reveals top-down, system-level functions that need to be performed by the system, where these functions need to be performed, and how often they need to be performed [2]. This is accomplished by examining functions according to various criteria such as sequence, data exchange, or resource usage; decomposing higher-level functions into lower-level functions; and allocating functions from the program to any number of dependent projects [3]. Functions, in this context, are processes that take inputs in and transform them into outputs [4] to achieve the system goals and objectives. These functions may be stated explicitly in the source requirements, or they may be derived from requirements. They may also be covered partly in operational concepts and are at times called by different titles like "logical decomposition." The functions will eventually be performed through the use of hardware, software and/or people.

C. The Concept of Inter-Functional Architecture

In general a functional architecture contains at least a statement of the existence of functional subdivisions of a system, and a statement of a set of relationships between them. There may also be descriptions of principles and guidelines that govern their design and evolution over time [5], or additional information describing rationale and stakeholder concerns [6]. The artifacts used to express the relationships between functions include object models, data flow diagrams, state models, activity diagrams, functional flow block diagrams (FFBDs), and performance budgets, to name a few. These artifacts help produce an interim product called the functional architecture description (FAD), which describes the system in terms of a functional decomposition rather than a spatial decomposition.

We have found useful the concept of an inter-functional architecture (IFA). An inter-functional architecture is a design structure matrix [7] applied to the functional decomposition of the system. Design structure matrices document relationships between components, and are the parent class of the commonly used N2-diagrams. We believe that this article is the first recognition of applicability of the design structure matrix to a functional decomposition of the system. The matrix may be represented in tabular form as in Table 2 (the X's represent links to other information describing the relationships between functions), or in the form of a directed graphical representation showing the interrelationships between high-level system functions (see Figure 3). The latter is more convenient when the matrix is sparse. It is important to note that use of data flows in the examples is notional; any other kind of relationship can be represented in an inter-functional architecture.

Design structure, in this context, is a statement about a set of functional relationships. Formally, a design structure matrix is a set of ordered pairs $\boldsymbol{G} = \boldsymbol{(F, L)}$, where $\boldsymbol{F}$ is a finite set of functions, and $\boldsymbol{L}$ is a link associated with a rule that assigns relationships to groups of functions that are a subsets of $\boldsymbol{F}$. The rows and columns of the matrix, or the nodes on the graph, represent various system-level functions in $\boldsymbol{F}$, while the links, $\boldsymbol{L}$, describe dependencies between various functions. The relationships pointed to by $\boldsymbol{(L)}$ can represent data flows, state transitions, event triggers, open information streams, timing of events, etc. The IFA differs from the functional flow block diagram, the former being more general and the latter being more specific to sequencing and triggering of activities. First, the FFBD describes the sequential relationship of all functions that must be accomplished. The IFA can show any type of relationship and is amenable to any of the techniques for detecting architectural mismatches that are used

Table 2. Design Structure Matrix Notional Example for Data Flow Relationships

	Guidance	Thermal	Display & Controls	Fault Mgt	Recovery Mgt	Sequencing	Comm	Miss Plan
Guidance							X	X
Thermal	X		X					X
Display & Controls				X				
Fault Mgt								
Recovery Mgt							X	
Sequencing			X				X	
Comm								X
Miss Plan								

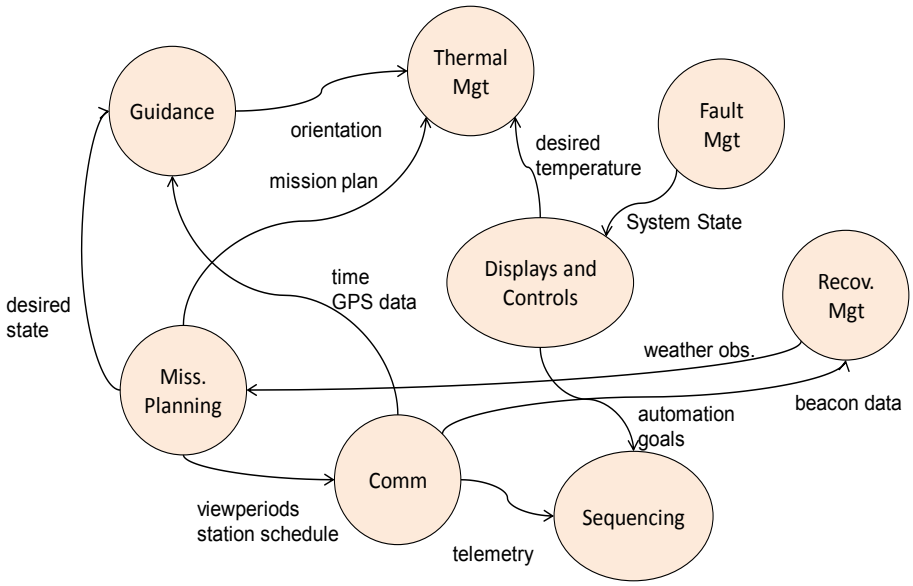


Figure 3. Inter-Functional Architecture Notional Example for Data Flow Relationships

on N2-diagrams; for instance, interface balancing or mechanization comparisons. This is an important advantage as it quickly reveals mismatches between system-level functions. Second, the FFBD arranges functions in a logical sequence whereas the IFA does not necessarily. The IFA can be useful in revealing missing requirements in addition to architecture insights that are more difficult to view using single function or physical architectures.

D. Can Functional Analysis and Spatial (Physical) Analysis be Combined?

Any hierarchy created by decomposing a system depends on the perspective taken by the viewer, and subsequently any number of decomposed hierarchies can be defined for the same set of systems [8]. The logical subsystems for spatial (physical) analysis are the physical components. The logical subsystems

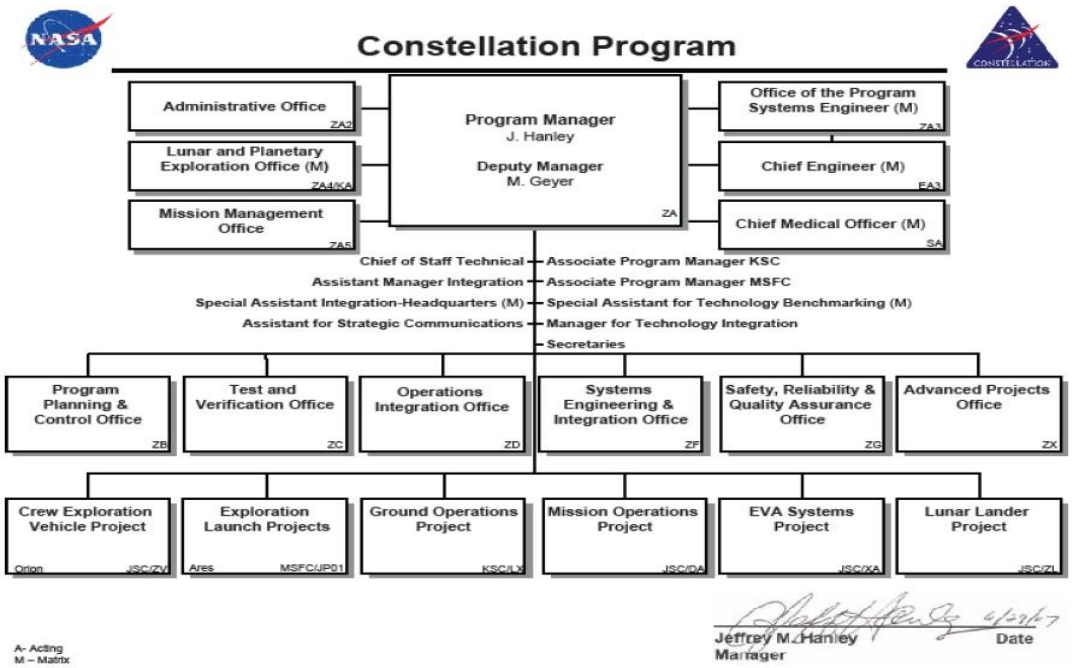


Figure 4. Initial Organizational Structure for NASA's Constellation Program

for functional analysis are the functions. Each viewpoint has advantages and disadvantages. Similarly, either approach may be able to yield an acceptable system solution if due diligence and sound system engineering principles are consistently applied throughout the life cycle given enough time and money. But, in the real world, where divergent organizational cultures, different mental models, competing paradigms and perspectives exist, some organizations prefer one approach over the other or choose a combination of both. For example, in the recently terminated Constellation program, the program's organizational structure was decomposed functionally using Apollo's "5-box" structure plus an additional "Advanced Projects Office" box (Figure 4, second row) whereas the projects were decomposed physically based on various system components in the physical architecture [9] (Figure 4, third row). The functional decomposition in Figure 4 (row 2) is primarily focused on the people and personnel tasks. The functional decomposition described in this paper focuses on the required functions of the integrated end product, that is, the functions that integrate the physical components. Good program management serves the people. Good systems engineering serves the end product. Functional analysis is needed in both regimes.

E. What is Program-Level Functional Analysis?

Program-level functional analysis is functional analysis applied at the highest level within a program to capture the fundamental set of system functions required for the end product to accomplish system-level goals and objectives. This fundamental or core set of functions are established based on explicit and derived requirements. Depending on the program's organizational structure [10], projects of the program can be decomposed functionally, spatially (physically) or both.

F. How is Program-Level Functional Analysis Different from Project-Level Functional Analysis?

Whereas program-level functional analysis utilizes requirements to establish the core or fundamental set of system-level functions, project-level functional analysis starts with the subset of functions allocated from the program. Functional analysis at the program-level inherently describes the aggregated characteristics of the whole system, as

opposed to project-level functional analyses which inherently describe the partial characteristics for a single subsystem. Furthermore, the program-level should only describe "what actions to perform", while most if not all project-level functional analyses should include "how to perform the allocated function" as well. The program-project relationship often represents a parent-child dependency. Traceability between the chain of functions between the program and the projects must ensure that there are no disconnects. The project's goal is to develop products or deliverables that perform the functions allocated to them by the program. The programmatic goal is to harmonize the partial functions allocated to the set of projects to meet program goals and objectives. Though program-level functional analysis contains high-level functions and should stop decomposing when project-level functions are reached, project-level functions can be extremely detailed, exhaustively including all sub-functions required to realize complete implementations of the partial functions allocated to them by the program.

III. The Necessity of Functional Analysis for Space Exploration Programs

The philosopher, poet and novelist George Santayana once wrote [11], "Those who cannot remember the past are condemned to repeat it." This wisdom embodies the efforts of many organizations and individuals (managers, engineers, scientists, etc.) who want to learn from past mistakes in order to grow, mature and improve. NASA is one such organization that attempts to explicitly learn from past faults and failures. A few years before Shuttle retirement, NASA leaders addressed several technical and management/leadership lessons learned from the Shuttle in order to leverage this knowledge on future human space exploration efforts. One of the technical lessons learned [12] was called "Simplification by functional integration versus complexity by decomposition." In this lesson, the leaders preferred the benefits of functional integration as a means of providing a strong connection between system components. In the reference, the term "complexity by decomposition" refers to a phenomenon where excessively segmented spatial (physical) subsystems add functional complexity elsewhere in the system. As we have already discussed earlier, functional analysis has definite advantages over spatial analysis alone. One of the management/leadership lessons learned from the Space Shuttle involved "cost control versus cost assessment." The primary lesson is leadership must control costs across the entire life cycle. This is accomplished by leadership not allowing costs to be neglected or to become reactionary. The neglect of utilizing effective integration tools (like functional analysis) has a profound impact on costs. In another report describing lessons learned during Space Shuttle Integration, Boeing's Space Shuttle Orbiter Program Director exclaimed [13], "We were not as smart as we thought we were! Develop and maintain a strong integration team throughout the program life cycle. Empower integration to challenge the elements and program on issues of design flaws and interaction between the elements." Strong integration is relevant and required from both technical and leadership perspectives. Technically, strong integration requires tools and techniques that manage the interfaces, interactions and influences between the parts of the system. From a leadership perspective, program integration and project engineering should be staffed with respected peers who have the courage to tackle issues. It is safe to infer from these and other lessons that neglect of strong integration (via people and engineering tools) has a significant impact on costs and schedule.

It is in the same manner that we focus on functional analysis as a means of coordinating the actions of large and diverse organizations for space exploration. This section will address functional analysis as a partial solution to heterogeneous program integration, the theoretical advantages of functional analysis, various approaches to developing an inter-functional architecture, as well as the pitfalls which may be encountered. The consequences of neglecting functional analysis at the program level will also be discussed.

A. A Partial Solution for Heterogeneous Exploration Program Integration

Most of the lessons learned from previous space exploration programs have been clustered into categories generally termed "management/ leadership" and "technical". Both categories are

complementary and both are required to achieve program end product success. Functional analysis, particularly at the program integration level, appears to provide part of the solution that enables strong product integration. This is because functional analysis efficiently bridges the gap between system-level requirements (what is holistically wanted from the system) and lower-level project implementations. Functional analysis at the system level (for long term space exploration ventures) is usually technology-independent which means it is an excellent tool for guiding product behavior over an extended life cycle. Given the unpredictability and uncertainty involved with space exploration, functional analysis is also evolvable and traceable vertically (from program to project and vice-versa) and horizontally (from project to project via a higher-level functional allocation). When implemented with sound engineering principles, functional analysis can provide increased costs savings (compared to political or ad hoc interface management) as well as efficiencies in communication which positively impact schedule. Functional analysis effectively serves as a technical communications tool between the program and the projects.

B. Theoretical Advantages of Functional Analysis

There are three main advantages to a program that result from including functional analysis in the system design process: improved performance of the organization, reduced risk to the program, and enhanced safety of the resulting system. These advantages are theoretical in the sense that there is a train of logic that leads one to expect the advantage, but these advantages have not, to our knowledge, been experimentally verified. There have been some anecdotal reports of success which we will point out, and there is also a large body of both experimental and anecdotal information that supports the converse assertion: that there are disadvantages to omitting functional analysis.

1. Performance Advantages

One important measure of the performance of a program team is the accuracy of its cost estimates. In a study carried out by the NASA Space Propulsion Synergy Team (SPST), it was found that controlling the lifecycle cost of a launch vehicle depended heavily on having a complete definition of the requirements at the beginning [14]. Also, among the most common factors for failure of software projects is an inaccurate estimate of needed resources [15]. The accuracy of a cost estimate depends to some degree on the completeness of the requirements; that is, if requirements are missing, they will not be included in early cost estimates, setting the stage for an upward trend of the estimate over time. The SPST recognized, and we independently recognized, that a functional analysis is essential to finding all the requirements. One reason is that the top-level system requirements, even if they happen to be complete, only express the requirements on the system as a whole. They do not express derived requirements that result from relationships between physical subsystems. Another reason is that the top-level requirements do not express derived requirements that result from relationships between functions, e.g. the need for communication functions to prioritize certain kinds of control or fault management data, or for power and thermal management to consider the constraints imposed by vehicle pointing for propulsion. The functionally-derived requirements can be significant drivers on both system cost and engineering cost, and they can only be found using a functional architecture.

Another important measure of the performance of a program team is its rapidity of converging to a solution. If a system were decomposed only along spatial lines, as is often done, it is easy to get insight into the physical relationships between the subsystems. It is much harder to get insight into the functional relationships starting from only a spatial decomposition. Our experience with the Constellation program was that functional relationships could be discovered eventually, starting from a purely spatial decomposition. The process used was to allocate high-level requirements to subsystems, synthesize subsystem designs, take note of their dependencies, organize the dependencies into functional clusters, analyze the functions to determine the functional relationships, and only then update the subsystem requirements to include the functionally-derived ones. However, this was a very long and expensive process in terms of personnel and time. The same thing could have been accomplished directly, in a shorter time, by starting the requirements analysis at the top level using a

functional decomposition in parallel with the spatial one.

A program team using functional analysis will often encounter design situations where they recognize that the same function must be performed in many different locations, typically having to do with a distributed function such as communication, data management or fault management. This is an opportunity to introduce commonality into the overall system design so as to achieve a superior solution in terms of cost and complexity.

Interface development can be aided with the use of a system-level functional analysis. The cross-subsystem functions, once understood, set requirements on the partial functions allocated to the subsystems. In turn, these define expectations on the interfaces that must be met if the top-level functions are to be realized. These expectations become interface requirements that can be stated as a target for the subsystem to meet. This method avoids the substantial time required for projects to discover otherwise unknown interface requirements through their own mutual interaction, and is usually less costly.

Fault management also benefits from functional analysis. Fundamentally, fault management has to make decisions about which functions to preserve in the event of a fault. Fault management analyses are often linked and interdependent with hazard analyses efforts with respect to must-work and must-not-work functions. So, of necessity, a fault management analysis depends on a functional analysis in its quest to determine which functions to preserve in the event of a fault. In the absence of an explicit functional analysis for the top-level system, the functional management effort has to perform a functional analysis of its own. This, however, is often done incompletely and solely for the purposes of fault management. The results can be 1) excessive time spent on fault management design while the functional analysis is being completed; 2) incomplete functional analysis, or 3) inconsistent functional analyses across disciplines. Overall engineering costs should be less by performing a single complete, consistent functional analysis for use by all technical disciplines.

Use of a functional decomposition at the top-level can help a program escape a peculiar trap when it comes to designing physical systems to match operations roles, and vice-versa. This trap is organizationally enticing because it gives the illusion that managing the people is the same as managing the operation. At the top-level, spatial boundaries inherently separate machines from human operators. The top-level owner, either the program manager or the system engineer, becomes responsible for managing the engineering transactions between the people (the machine engineers and the operations engineers). On large programs, the traffic between the people often overwhelms the manager who subsequently neglects the management of lower-level details. Under this circumstance, essentially no one is managing the match between human and machines. The way out of this trap is to use functional boundaries that keep both machines and operators together. An engineer given authority within the function can then define the allocation of sub-functions between machines and operators, and make sure that they play together effectively.

We believe that the greatest organizational performance advantage offered by functional analysis is the ability to exactly define the top-level engineering responsibility, authority, and accountability. The cross-project functions are those things that make the top-level system more than the sum of its parts. The cross-project functions are the reason for having a top-level system rather than merely a loose assemblage of subsystems under the care of a custodial parent organization that has little say in their development. The top-level organization is responsible for defining the cross-project functions; has authority over their design, verification, and integration; and is accountable if the cross-project functions are not realized. If only a physical decomposition is used, there is no concrete representation of the cross-project functions, and the top-level organization is at an extreme disadvantage in trying to create them. In fact, there is little the top-level organization could point to as its unique role beyond physical assembly, and even that could be delegated to one or a few of the projects. There would be nothing to design, and anything further in terms of controlling budget, schedule, performance, or risk would duplicate what the projects could do for themselves. Conversely, a concrete representation of the cross-project functions facilitates communicating goals, allocating requirements, and verifying the overall system.

2. Risk Advantages

Every program faces the risk that it will make errors that create gaps, duplications, and inconsistencies in the system architecture early in the design process. If the team is not aware of functionally-derived requirements, then analysts cannot detect functional errors in the architecture, with the result that the errors persist into the final product. Such errors will not be found until the system validation process which occurs very late in the development cycle. The errors can be corrected at that time, but at higher cost than if detected early. Thus, being able to discover functionally-derived requirements is important in correcting errors during the design phase. Indeed, many requirements-driven errors can be found using functional analysis without any investment in testing, development, simulations, or even the system design process. This makes it less costly than other techniques to overcome requirements errors.

The integration phase is often the time that latent errors are discovered. With a spatial decomposition, integration of the system can begin when the spatial subsystems are complete. When subsystems are connected, the system-level functions exist for the first time, and tests on them can begin. The down side is that failures of the system-level functions are discovered in a high-cost environment, with all machines and all teams up and running, often in a co-located scenario under severe time constraints. As an alternative, top-level system functions can provide useful integration insights due to the way functions are bounded. One system-level function can be assembled at a time using partial versions of the spatial subsystems as they become available, using only as much co-location or system fidelity as is needed to verify that particular function. If extra instances of the partial subsystems are available, the integration of single functions can be done in parallel. The early testing of system-level functionality increases the level of confidence that the full system will be integrated successfully.

3. Safety Advantages

An issue common to all safety analyses is whether the analysis is complete. If only physical components are considered, then only physical hazards can be addressed. Addition of a functional analysis enables identification of functional hazards as well as hazards that are physically obvious. In the case of integrated hazard analysis on the Constellation program, identifying the complete set of must-work and must-not-work functions was critical to providing a complete set of hazard mitigations because distributed systems must be analyzed for hazards in terms of the functions that rely on them [16]. Using a companion functional hazard analysis, we found a number of otherwise-unrecognized hazards, which completed the hazard list. Once these were identified, a set of hazard controls were developed to mitigate the known hazards. Completeness can be judged against the list of known hazards, and dependability cases can be constructed to prove completeness of the analysis [17, 18]. Certification of readiness is then straightforward: if the logic of the dependability case is correct, then the system is dependable for the intended use and can be certified given the verification evidence.

A final safety advantage may be available in some circumstances: finding system solutions that cut down on physical redundancy, but that still meet safety requirements. The risk in physical redundancy is common cause errors. When a common defect is found in one physical component, confidence is lost in all similar physical components. Functional redundancy can be viewed as a mitigation against common cause errors because it increases the robustness of fault tolerant designs by increasing dissimilar legs of redundancy. Additionally, functional redundancy also provides cost/volume/weight advantages. For example, suppose that having accurate time on a system clock is critical to the safety of a space vehicle. Rather than carrying a number of physical clocks, the navigation function may be able to determine time from external observations.

C. Approaches to Development of an Inter-Functional Architecture

We have observed three successful approaches to developing inter-functional architectures.

1. Fiat Approach

In the *fiat* approach, a set of functions is asserted by the analyst. This can be done top-down, asserting a decomposition and then from that deriving the relationships between the functions. Alternatively,

this can be done bottoms-up, in which the analyst asserts a composition from known sub-functions. In either direction, the analyst may refer to functions as expressed by customers or suppliers of the system under consideration, or may refer to previously-identified design patterns such as the Hatley-Pirbhai template [19].

2. Use Case Approach

In the use-case approach, the analyst identifies scenarios describing the system operation using actor-verb-object statements. The verbs are then analyzed to identify functional decomposition (or composition).

3. Temporal Approach

In the temporal approach, the analyst describes the system operation using required timelines of events. The analyst then identifies transformations that must occur between events. The transformations are used to generate a candidate set of functions, which then may be organized as either a decomposition or a composition (see comment on the pitfall of temporality below).

D. Pitfalls in Functional Analysis

1. Failure to Consider Alternatives

The first functional architecture found is rarely the best. Buede [4] recommends trying alternate decompositions, disaggregating the functions differently, bundling and unbundling transactions differently, re-evaluating functional dominance in terms of feedback and control, catching interface errors. Of course the judgment of "best" is a problematic issue, but some reasonable factors to consider are sufficiency for the problem space, interface balance, freedom from gaps and undesired overlaps, complexity of mapping to the physical decomposition, and verifiability.

2. Temporality

Particularly when following the temporal approach, it may be tempting to make a one-to-one identification between periods of activity and the functions. To see the consequence of succumbing to this temptation, consider an elevator system. One potential timeline for the system might be:

Passengers request use of the elevator

Elevator goes to the first floor

Passengers ingress car on the first floor

Elevator goes to the second floor

Passengers egress car on the second floor

…

Elevator goes to the top floor

Passengers egress car on the top floor

Using a strict identification of activity into function, the functions would be:

Request use

Go to first floor

Ingress first floor

Go to second floor

Egress second floor

…

Go to top floor

Egress top floor

There are at least three deficiencies of the resulting set of functions. Perhaps the most obvious is that there are too many nearly identical functions; "go to floor" would be more useful than "go to floor N". The next deficiency is that the timeline only expressed one possible sequence of operation. There are many possible sequences for an elevator to execute, so the timeline does not express that possibility well. This is very typical of "happy day" analyses that ignore the possibility of error conditions that require the system to execute contingency sequences. The third deficiency is that many functions are missing: support of riding passengers, emergency communication, or maintenance are a few that come to mind. It is usually necessary to go back and consider alternative decompositions after starting on a temporal approach to alleviate these kinds of deficiencies.

3. Excessive Abstraction

In the functional decomposition or recomposition processes, functions can become too general or obscure to be used as meaningful kernels of allocation, integration, or testing later in system development. The functions should be checked to ensure that they make sense for the later uses.

4. Poor Timing

The work of functional analysis can be intellectually challenging or unfamiliar, and as a result can end up taking too long considering the surrounding program constraints. The work of functional analysis should keep up with the prevailing schedule, making contributions to the surrounding development effort as increments of understanding appear. On the other hand, work that is done too soon may lack ability to influence decisions when they are eventually made (decision makers may not, or may not know how to, use the artifacts resulting from functional analysis). Discussion of functional analysis results is most fruitful in the context of an active design process.

E. The Consequences of Neglecting Program-Level Functional Analysis

The consequences of neglecting program-level functional analysis can be measured quantitatively as well as qualitatively. Quantitative measures include increased costs and expanded schedules. Qualitative measures include nonproductive and ineffective use of resources due to misexpectations and miscommunications between program teams. Because of the surrounding reality of changes of direction (in terms of funding and guidance), the exploration program often received significant impacts in unexpected and inadvertent ways. NASA leadership just recently released a compilation of Constellation lessons learned [20] where they focused primarily on nontechnical challenges.

One of the unintended impacts on the Constellation program occurred at the very beginning during the start-up phasing between the projects and the program.

> "Two of the primary projects, the *Orion* Crew Exploration Vehicle and the *Ares* Crew Launch Vehicle, were begun well in advance of the Constellation Program. This enabled rapid start-up and early results from each project. They functioned for some time without an integration function between them...These factors led to costly contract changes when the appropriate level of integration for a complete architecture was better understood. This approach also provided an additional integration challenge. The projects had formed requirements, budgets, design approaches, acquisition strategies, etc., that needed to be integrated retroactively. The late addition of the integrating functions had a long-term effect that was not fully resolved by the end of the program. Strategies that were in a collective self-interest for the program were extraordinarily difficult to implement due to the cost of contract changes...Moreover, integrated analyses by the program uncovered technical issues that could have been caught earlier (and been resolved in a more efficient manner) if the start-up timing had been different. Incentives on contracts were developed primarily to benefit each project rather than the overall program. Missing were incentives for contractors to participate in technical solutions that benefit the integrated program. This drove results that were less than optimum, and sometimes not mutually compatible" [20].

One of the technical issues (not mentioned in the report) that had a delayed effect on the program was the deferred use of functional analysis during early stages in the program life cycle. Though a partial product had been developed, the impact of deferring functional analysis was not noticed for a number of years which significantly impacted cost and schedule due to retroactive rework. Only after progressing toward the end of the preliminary design was it manifest that multiple program-level teams were experiencing difficulty in justifying a complete end-to-end integrated story. Even though program requirements were traceable to the projects, there was difficultly in consistently translating the requirements into actions and difficultly in determining if there were gaps or missing system-level functions which led to (in some cases) omissions of duties and unnecessary redundancies between program-level teams. In due time, program-level teams identified the need for a complete end-to-end functional

analysis. This issue, however, was not fully resolved by the end of the program.

Without a functional analysis, it may be difficult for a program to converge to an acceptable system design, to recognize opportunities for commonality, to identify gaps in contracts with vendors for subordinate systems, to avoid discovering unintended emergent behaviors or major errors late in the lifecycle, to manage the human-machine interface, to identify system-level verification and validation criteria, or to certify that its system is safe. If any of these difficulties are encountered, we would suggest looking back over the history to see whether a functional analysis was done; if not, it is essential to complete one before such situations can be corrected. Of course, it is undesirable to go without the benefits of performing functional analysis early, but to some extent, functional analysis can be performed later in the life-cycle for emergency redesigns of the program or the system.

The absence of a functional analysis is likely to produce a situation where the top-level engineering organization is not aware of the emergent qualities introduced by the interactions of the system components. The organization may erroneously come to the conclusion that the top level has no legitimate engineering function, and managerially has no reason to exist beyond double-checked accountability imposed on the lower-level organizations. The sequel is a self-fulfilling prophecy, in which the organization believes there is no need for system-level qualities, and the final product ends up having none. Essentially, this situation is an abrogation of the entire concept of systems engineering.

IV. Conclusions

As NASA evolves its human space exploration capability to include diverse partners, there is an increased need to integrate and effectively allocate responsibilities between government and commercial stakeholders to achieve coordinated results. The system engineering and integration technique of functional analysis will provide performance, risk, and safety advantages to such a large-scale government-commercial partnership by helping to manage the interfaces, interactions and influences between diverse components and heterogeneous players. Lessons learned from the Space Shuttle and Constellation programs all attest to the sustained need for strong program-level functional integration across the life cycle as a means of achieving mission success within cost and schedule constraints. For large-scale exploration programs, there needs to be a paradigm shift away from a tendency to see cost and schedule as separated from the way the organization goes about realizing the end product (via people or the technical end product). The organization (of the people and the end product) and the methods by which the product is integrated both have direct and indirect effects on cost and schedule. Functional analysis is an effective solution that can both improve and clarify the connections between costs, schedule and organization of the technical effort.

References

[1] *FY 2012 Complete Budget Estimates, National Aeronautics and Space Administration,* NASA Headquarters, Washington, DC, February 14, 2011, http://www.nasa.gov/pdf/516675main_NASAFY12_Budget_Estimates-508.pdf [accessed 2 Aug 2011].

[2] *Systems Engineering Fundamentals,* http://www.dau.mil/pubs/pdf/SEFGuide%2001-01.pdf [accessed 25 Aug 2011], Defense Acquisition University Press, 2001, page 31.

[3] Guerra, Lisa, *Space Systems Engineering: Functional Analysis Module*; Department of Aerospace Engineering at the University of Texas at Austin, Austin, Texas, 2008, http://spacese.spacegrant.org/index.php?page=functional-analysis [accessed 3 Aug 2011].

[4] Buede, Dennis M., *The Engineering Design of Systems – Second Edition*, p. 214, Wiley, 2009.

[5] *DoD Architecture Framework v. 1.5*, Vol. I, Dept. of Defense, Washington D.C. 23 April 2007, http://www.health.mil/Libraries/Captions/DoDAF_Volume_I.pdf [accessed 4 Aug 2011].

[6] *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, Figure 1, IEEE Std 1471-2000, IEEE Standards Association, Piscataway, N.J., 21 September 2000, http://standards.ieee.org/findstds/standard/1471-2000.html [accessed 4 Aug 2011].

[7] Browning, Tyson R., "Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions", *IEEE Transactions on Engineering Management,* Vol. 48, No. 3, August 2001,

http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=946528&tag=1 [accessed 4 Aug 2011].

[8] Whitcomb, Clifford and Szatkowski, John, *Functional and Physical Decomposition for Ship Design*, Symposium paper, Massachusetts Institute of Technology, Cambridge, MA, 21-23 March 2000.

[9] Rhatigan, J. L., Hanley, J. M., and Geyer, M. S., "Formulation of NASA's Constellation Program," NASA SP-2007-563, NASA Johnson Space Center, Houston, Texas, October 2007.

[10] Morris, A. T., and Massie, M. J., "Structuring the Organization for Large Scale Integrated Hazard Analysis," *AIAA Infotech@Aerospace Conference,* St. Louis, Missouri, 29–31 March 2011.

[11] Santayana, George, *Life of Reason*, "Reason in Common Sense," Scribner's, 1905, page 284.

[12] Zapata, Edgar, Levack, Daniel J. H., Rhodes, Russel E., Robinson, John W., "Shuttle Shortfalls and Lessons Learned for the Sustainment of Human Space Exploration", 45th AIAA/ASME/SAE/ASEE Joint Propulsion Conference, Denver, Colorado, August 2-5, 2009, AIAA 2009-5346.

[13] Bejmuk, Bo, "Space Shuttle Integration Lessons Learned - An Insider's View," The Boeing Company, April 5, 2006.

[14] Robinson, John W.., "Controlling Launch Vehicle Life Cycle Costs", *Aerospace America*, October, 2010.

[15] Charette, Robert M., "Why Software Fails", *IEEE Spectrum*, September, 2005.

[16] Morris, A. T., and Massie, M. J., "Analyzing Distributed Functions in an Integrated Hazard Analysis," AIAA-2010-3486, *AIAA Infotech@Aerospace Conference,* Atlanta, Georgia, 20–22 April 2010.

[17] Weinstock, Charles B. et al., *Dependability Cases*, Technical Note CMU/SEI-2004-TN-016, Carnegie-Mellon University Software Engineering Institute, May 2004.

[18] Goodenough, John B., and M. Barry, "Evaluating Hazard Mitigations with Dependability Cases", AIAA 2009-1943, *AIAA Infotech @Aerospace Conference*, Seattle, Washington, 6 April 2009.

[19] Buede, Dennis M., *The Engineering Design of Systems – Second Edition*, p. 214, Wiley, 2009, figure 7.3.

[20] Rhatigan, Jennifer L., Neubek, Deborah J., Thomas, L. Dale, and Stegemoeller, Charles, "Constellation Program Lessons Learned; Volume I: Executive Summary," NASA SP-2011-6127-VOL-1, Johnson Space Center, Houston, Texas, June 2011.

Acknowledgements

We would like to thank Mahyar Malekpour of the NASA Langley Research Center, Michael Massie of the ARES Corporation, Houston, Texas, as well as Ron Morillo and Lorraine Fesq of the Jet Propulsion Laboratory for their helpful comments and discussions in the development and critique of this paper. The research described in this publication was carried out in part at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.