A Process to System Engineering: Creation of a System Engineering Process Model and its Deployment on the Orion Program

S85-2010-5000-0802-007-1.0

Mitch Fletcher[1] and Bill Airo[2]
Honeywell International, DSES - Human Space, Aerospace, Glendale, AZ, 85308

Julee Clelland[3]
Honeywell International, DSES - Human Space, Aerospace, Glendale, AZ, 85308

The term systems engineering can be traced back to Bell Telephone Laboratories in the 1940s. In the same timeframe, General Electric had an in house educational course that taught system engineering and electrical design. All of these early system engineering efforts were created to deal with the complexity of newly emerging technology and its application to complex applications (e.g. flight dynamics of unstable systems, etc.). Since that time, many publications and textbooks have described the art of system engineering. Within the past decade, the system engineering process has been documented in many forms. In fact, the increased need for system engineering in a complex world resulted in the founding of International Council on Systems Engineering (INCOSE) in 1990. The most recent recognized best in class description of System Engineering has been institutionalized by Capability Maturity Model Integration (CMMI) models. In the same timeframe development of CMMI for System Engineering, architectural framework techniques have further been institutionalized by the government with the various releases of Department of Defense Architecture Framework (DoDAF) Figure 1. Many companies, including Honeywell, have created Engineering Process Models (EPMs) to institutionalize their system engineering process and allow for certification by the Software Engineering Institute established by Carnegie Mellon. This paper describes the process that Honeywell executed to establish an Engineering Process Modes, and our subsequent selection of CMMI for that model. In addition, the Honeywell 3-view system Architecture Framework process will be compared to DoDAF and the NASA Systems Engineering Handbook (NASA/SP-2007-6105 Rev 1). The Altair deployment of the 3-view framework is included as an exammple. A final segment of the paper covers the observations of the implementation of system engineering for the avionics system on the Orion program and lessons learned from that program.

[1] Chief Engineer, Electronic System Engr & Apps, 19019 N. 59th Ave./2S12, AIAA Senior Member.
[2] Senior Technical Manager, Electronic System Engr & Apps, 19019 N. 59th Ave./2S12, n/a.
[3] Engineer III, Electronic System Engr & Apps, 19019 N. 59th Ave./2S12, n/a.

Nomenclature

SLI = Space Launch Initiative
CR/ISP = Cycle Time Reduction / Integrated Systems Process team
SEMP = System Engineering Management Plan
TDP = Technical Development Plan
CMMIsm = Capability Maturity Model Integration
DRCL = Design Requirements Checklist
C4ISR = framework for Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance architectural development, presentation, and integration
QFD = Quality Functional Deployment
SBD = System Boundary Diagram
OIE = Operational Information Exchange Matrix
CEV = Crew Exploration Vehicle
MEL = Master Minimum Equipment List
LDAC-1 = Lunar Design Analysis Cycle
RMUX = Remote Multiplexer Unit
RDCs = Remote Data Converters
LM = Lockheed Martin
PDU = Power Data Unit

I. Introduction

THE term system has been discussed at many a conference and around the bar by many self named system engineers for decades. A common definition for the term system is "a group of units so combined as to form a whole and to operate in unison". A technical definition similar to the common one, but not very specific, is offered by Hall and Fagan: "A system is a set of objects together with relationships between the objects"[1]. The term systems engineering can be traced back to Bell Telephone Laboratories in the 1940s[2]. In the same timeframe, General Electric had an in house educational course that taught system engineering and electrical design. The author is fortunate to know someone in position of part of General Electric's original teaching guide. This course is typed and bound in multiple three ring binders. All of these early system engineering efforts were created to deal with the complexity of newly emerging technology and its application to complex applications (e.g. flight dynamics of unstable systems, etc.). In fact, the GE teaching guide includes early driving point impedance analysis modes for the early transistors. In 1989, General Dynamics hosted a meeting at the University of California, San Diego, to discuss the apparent shortage of *qualified* engineers, those who could think in terms of a total system (rather than just a specific discipline) and could implement the systems engineering process. The group concluded that this shortage was of national scope, and agreed to recruit more representatives of government, industry, and academia to examine the problem. The group agreed that a national organization was needed, adopted a charter, and formed the National Council on Systems Engineering (NCOSE).

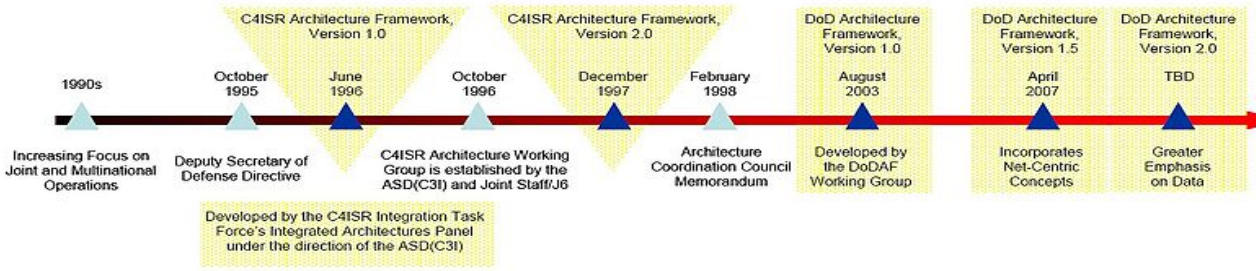

Figure 1. Evolution of DODAF System Engineering Decomposition Process[4]. *The institutionalization of System Engineering started in the late 1980s and early 1990s. As companies started to lose experienced engineers from the Apollo era, the need to document and pass on this wealth of system engineering experience became recognized as a cost benefit to companies.*

The most recent recognized best in class description of System Engineering has been institutionalized by Capability Maturity Model Integration (CMMI) models (evolved the Capability Maturity Model (CMM) concept). In the same timeframe of development of CMMI for System Engineering, architectural framework techniques have further been institutionalized by the government with the various releases of Department of Defense Architecture Framework (DoDAF) (see Figure 1. Many companies, including Honeywell, have created Engineering Process Models (EPMs) to institutionalize their system engineering process and allow for certification by the Software Engineering Institute established by Carnegie Mellon.

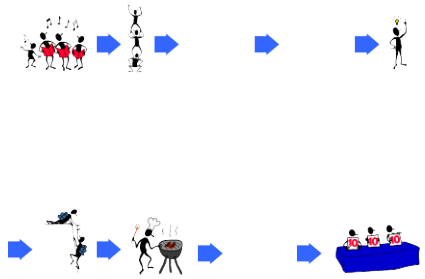

Figure 2. Early CRISP Process Model. *The Initial Honeywell Process Model work started in early 1999 focused upon EIA 731.*

II. The CR/ISP Team

During the 1980s, Honeywell experienced a large turn over in engineering staff in both the Human Space (then called Data Control Systems [DCS]) and Control Moment Gyroscope development teams. This resulted in the programs of the late 1980 and the early 1990s being staffed by only a handful of senior engineers. The remaining multitude of junior engineers had not received sufficient on-the-job training and there were not enough senior engineers to adequately mentor the staff. This had the result of program errors and design escapes. The most annoying of these were the requirement escapes. Young engineers tend toward implementing things "the same way we did last time" which resulted in many programs finding out they did not meet the performance requirements at CDR or even worse at qualification.

At the same time, the "radical" process (today commonly known as LEAN) was exercised with a goal of achieving a cycle-time reduction of 50%. Because there was significant, cost associated with cycle time reduction and the elimination of program escapes, funding was authorized to implement lessons learned and process improvement. The first finding of the team is that the company was using an adhoc process. The first order of business was to create as-is and to-be process maps. Additionally, the site leadership started looking at short term fixes.

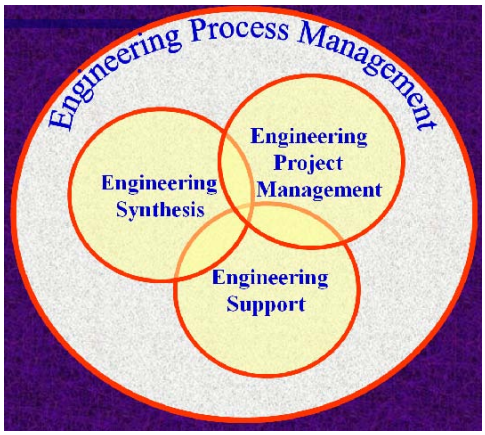

Figure 3. Engineering Process Model Concept. *The EPM provides a consistent engineering project management practices integrated with Program Management practices and provides an approach for managing the integrated engineering process with an interface to other Honeywell organizations and industry standards. A successful EPM provides forward thinking to support the future of engineering.*

A. Approach to the Process Model

Along with this training, formals meetings were held where all the senior engineers and Technical Directors met for sensing sessions and lessons learned. At this time, a "straw man" for the future state of system engineering was put together (see Figure 2). The "straw man" was modeled after the EIA 731 Process Model and started the Cycle Time Reduction / Integrated Systems Process team (CR/ISP) was kicked off. The charter of the CR/ISP team was to develop & deploy an integrated product development model that is flexible & applicable to all Glendale programs, incorporating best practices both within and outside of Glendale. The project was a four-phase project:

- Phase I - Identification and assessment (1Q01)
- Phase II - The new process (2-3Q01)
- Phase III - Pilot program (4Q01)
- Phase IV - Completion & full deployment (1Q-2)

In addition to the choice of a final process model, the CR/ISP Team was chartered with the culture change that would be required to accept the new procedures. It is widely recognized within the industry that organizational

culture must change to enable the implementation and institutionalization of process improvement. The culture needs to have shared values, beliefs, and understanding. This starts at the top, but needs buy-in at all levels. Engineering Processes are useless if the engineers cannot use them, or will not use them. It was essential that the CR/ISP team develop processes that are usable, understandable, and accessible. The vision of the new engineering environment was to achieve:

•PROCESS MODELS	•ASSESSMENT MODELS	•ASSESSMENT METHODS
Defines "What" Must be Done in An Organization to Achieve Product Development:	Identifies the Best Practices and Artifacts Expected of Processes at Various Capability Levels:	Recognized Step-wise Measurement Methods and Scoring:
–MIL-STD-2167	–SEI SW-CMM	–SEI SW-CMM
–MIL-STD-499B	–EPIC SECMM	–EIA-STD-731-2
–IEE-STD-1220	–INCOSE SECAM	–EPIC SECMM
–EIA-STD-632	–EIA-731-1 SECM	–SEI CMMI
–ISO STD 15288	–SEI SE/SW CMMI	

Figure 4. Process Model Attributes. *In the early 2000s, there were many competing process models. The only consistent industry standard was SEI CMM for software. The above table illustrates the difficulty in choosing a process model in this timeframe.*

- Proposals that contain the information needed to start a project plan
- Plans that are actually used throughout the project
  - – Reduce initial effort to develop a project plan
  - – Facilitate seamless updates and communications of plan changes
- Products, plans and work that can be tailored for projects from $1K - $100M
- Consistent between projects
  - – Incorporate lessons learned and "best practices" so that they are used on projects
  - – Reduce training when moving between projects
- Reduce redundant overlap between program management and engineering
  - – Projects detailed plans and status easily roll up to what PM needs
  - – PM needs easily flow down to project plans
- Reduce (eliminate) "heroic" efforts that we typically use to deliver products

An engineering process model is needed to provide an organized approach to identify, define, and incorporate "best practices". It also provides a structure that defines and organized activities so that staff from one project can move to a second project efficiently. Most engineers like the idea of "lessons learned". A process model provides the mechanism for developing and improving the processes so that "lessons learned" are integrated into the business model and do not have to be reviewed continually. Historically, a "lessons learned" list ends up being a "lessons forgotten" list.

B. Selection of CMMI

The general requirements for selection of a process model included a model based upon a current and future industry accepted standard. The choice could not negatively impact the current site rating (at that time, ISO 9000 and SEI Level 3 for Software). The model had to provide a process for architecture implementation and requirements management. The process had to be tailor-able from the standard based upon individual project needs. It also needed to provide a framework for continuous improvement and the ability for evaluation Honeywell capabilities compared with an industry standard. The process needed to support various product life cycle implementations. The final requirement is that facilitate the process institutionalization within the organization.

Figure 5. Model Candidates and Industry Standards. *A large number of candidates were investigated and CMMIsm for Systems Engineering was chosen for implementation.*

A number of six-sigma tools were utilized for the evaluation of the Engineering Process model. In the end, the Capability Maturity Model Integration (CMMIsm) for Systems Engineering/Software Engineering/Integrated Product and Process Development - Continuous Representation was selected.

The main reasons Honeywell choose to use the CMMI Continuous Model are:

- It provides a road map for process definition and improvement for multiple Engineering disciplines
- Combination of the two most recognized industry standards (SEI CMM for Software and EIA/IS-731 for Systems)
- Adoption of the CMMI as a standard is presently the trend in Industry and other Honeywell locations.
- SEI CMM will be phased out of existence in approximately four years and planned replacement by the CMMI.

III. CMMI Deployment

With all of the planning and study that went into the selection of the CMMI process model executed by the CR/ISP team, the Honeywell process to tailor the CMMI implementation and achieve certification should have been trivial. Unfortunately, planning and execution of a plan are not always the same. Honeywell's implementation of CMMI is a classic example of why an organization should have a process to follow. Unfortunately, there is not a process to implement a process.

A. The Initial Implementation

With all of the planning and study that went into the selection of the CMMI process model executed by the CR/ISP team, Honeywell leadership thought that deployment of the process was well in hand. A team of junior engineers was put together to review the CMMI requirements and create a process ready for prime time. The assignment of a team of junior engineers seemed to make since because there was a very good set of expectations and all the groundwork for implementation had been completed. Some of the major trades had already been completed, such as the requirement to require (or not) a System Engineering Management Plan (SEMP). Therefore, the leadership funded a burden project to implement the Honeywell CMMI process and the project started.

To make a long story short, the process documents were completed and ready for deployment, albeit six months late to the initial schedule. The initial Engineering Process Model was over 600 pages long and almost a rewrite of the CMMI documentation. In the interim, Honeywell was awarded the Orion program with the requirement that the process be certified by the Orion PDR. The Orion program and a small test equipment job were selected to be the pilot programs for the CMMI deployment. Since both of these programs were beyond the proposal stage, the first phase of the CMMI process that required an output was the creation of the SEMP. This first straw is the one that broke the camel's back. Both pilot programs discovered that the process was overly complex and required a tremendous amount of effort to interpret the document. It was clear that we had failed in out attempt to implement a process envisioned by the CR/ISP team.

B. The Next Steps

The next steps in CMMI implementation were obvious, but none the less painful. Two of the most experienced engineers in the company were assigned to re-write the CMMI process. Their approach was to take all the existing adhoc processes that worked and fit them into the CMMI framework. It took these senior engineers and their support team fourteen months to re-write the process. The Process Model was reduced to 72 pages. The initial pilot program created compliant SEMP documents and the organization has not looked back. To date, seven programs have deployed using the CMMI process since its release in October of 1988.

C. A Process Model That Works

Honeywell has studied and implemented an Engineering Process Model for both System Engineering and Software development. In considering architectural options, Honeywell draws upon our domain avionics architecture experience gained through the programs referenced previously herein. Honeywell analyzes Avionic Subsystem Architecture needs, including design to cost elements, using processes from our CMMI process model (see Figure 6) as tailored to align with the newly released NASA system engineering process.

The Honeywell (Aerospace Glendale) Engineering Project Model (EPM) is a description of the processes to be followed by engineers working on systems and/or hardware design and development programs for the Aerospace-Glendale (59th Ave) facility. It is a suitable reference for use by experienced technical directors and teams. The detailed guidelines for EPM Implementation contain additional guidance and application notes. The intent of the standard processes described in the EPM is to ensure high quality, low cost, and to reduce the project's lifecycle.

The EPM provides the framework in which all engineering projects operate. It describes how the individual processes are selected, integrated and consistently applied on a project to produce a system or product and manage its development. It is the goal of this document to be as simple and straightforward as possible for the engineering community that uses it.

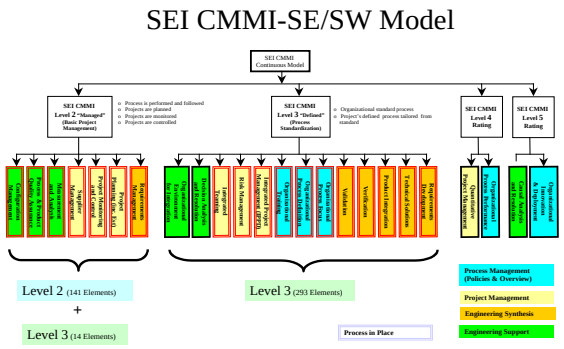

Figure 6. Honeywell CMMI Model. *Honeywell internal engineering process applies the discipline of its existing CMMI tailored to the new NASA architecture framework process to assure success in the architecture synthesis.*

The EPM is a comprehensive and practical description of how an engineering project is executed within the organization. It describes the fundamental process elements that each engineering project is expected to incorporate into its defined engineering process.

The EPM promotes uniformity within and across Aerospace-Glendale (59th Ave) projects by integrating location policies and procedures, into a framework that is aligned with Honeywell Aerospace Process Description - Design Factory (APD-04) and the SEI Capability Maturity Model Integration (CMMI, v1.2). This approach encourages practices that produce high quality products within schedule and budget. It also provides continuity in the organization's process activities and is the reference for the measurements and long-term improvement of the engineering processes used in the organization.

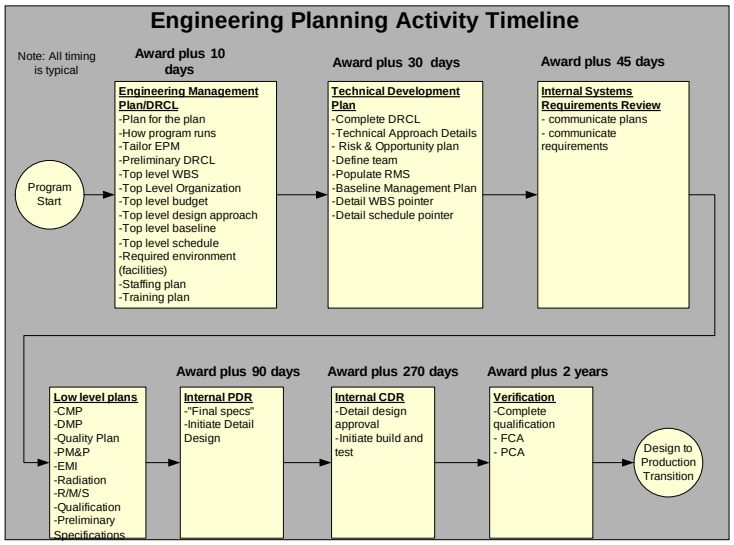

Figure 7. EPM Planning Sequence. *The EPM provides a complete life cycle process implementation for a program. Each required element of the CMMI process is covered throughout the lifecycle.*

A process is defined as a logical sequence of tasks performed to achieve a particular objective. Thus, it defines what is to be done, without strictly specifying how it will be accomplished. A lower level view of a process is that it is composed of one or more procedures, which describe how to perform the process, i.e., what steps to do or techniques to use. A procedure may or may not be automated by use of a tool. A procedure generates one or more work products, which are artifacts that can be deliverable items, inputs to another procedure, or simply evidence that the procedure was accomplished.

The EPM requires each project to develop a Technical Development Plan (TDP) with an appropriate level of technical detail. A separate TDP is typically developed for each Configuration Item (CI). The TDP provides an efficient means to capture the process planning

information for less complex programs. Complex programs may need to use a separate management plan to standardize processes across all CIs and document how the efforts of multiple development teams are to be integrated and coordinated.

A System Engineering Management Plan (SEMP) is used to simplify the planning of complex programs by communicating a common vision and coordinating common process requirements, which would otherwise be replicated in multiple TDPs, and Design Requirements Checklists (DRCLs). Complex projects continue to provide the necessary detail in their TDPs and DRCLs, but reference the program SEMP for common process requirements such as metrics, shared vision, stakeholder involvement, and resource management (staffing, training and facilities). A criterion that determines whether a program is complex and should use the SEMP is pragmatic and flexible. It is driven by the need to centralize management of an engineering program with a consistent common plan.

IV. Architectural Framework Process

Persons who have ever remodeled their home know how important building codes, blueprints, and city or county inspections are to successfully complete the project. The architect operates within a "framework" of building codes, preparing blueprints for each phase of the project, from the structural changes to the size and layout of the rooms. Detailed drawings specify plumbing, electrical, and building construction information for the entire structure. Enterprise Architecture works in a similar manner5. The same is true for systems engineering. Architectural Framework provides the ability to “simplify” the system by decomposing it into the minimized implementation.

A. Government Based Architectural Framework

The first release of a defined framework, the C4ISR Architecture Framework Version 1.0, was developed by the Integrated Architectures Panel of the C4ISR Integration Task Force, and was published on 7 June 1996. C4ISR is a framework for Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance architectural development, presentation, and integration. The C4ISR Architecture Framework, Version 2.0 was released in December of 1997. This specification was superseded by the Department of Defense Architectural Framework (DoDAF) in April of 2007.

B. Honeywell 3-View System Engineering

The Architectural Framework within Honeywell started concurrent with the DoD transition from C4ISR to DoDAF. At the time, Honeywell primarily used a software program called DOORS® as a requirements management tool. For level 3 architectural implementation (e.g. the black box level), the textual capture of “shall” requirements statements and the powerful traceability capability provided by DOORS® was sufficient. As Honeywell migrated from a black box supplier to a sub-system, system, and system-of-systems supplier, the textual capability provided by DOORS® was insufficient. At this time, there were no graphical tools available to do Level 1 and Level 2 system decomposition. Because of this, Honeywell created a customization of the C4ISR implementation based upon integration with a home grown tool. The SLATE language was selected to integrate the Honeywell custom 3-View process with a tool implementation.

The 3-View system approach focuses on “artifacts”, not documents and uses the views of the system to organize the artifacts and allow complete requirements allocation and management. The basic organization of the process is shown in Figure 8 and is designed to approach the system engineering from the three aspects as defined below managed by a management control plan defined within the CMMI process. Looking at the system from these three aspects will lower the cost and risk to the development program because the requirements are

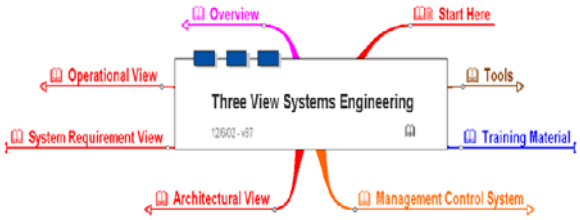

Figure 8. 3-View Architectural Framework. *The 3-View system solution approach focuses on an Operational View, a Systems Requirement (or Functional) View in addition to an Architectural View approach to assure all aspects of a customer's system needs are addressed and satisfied.*

clearly allocated and the likelihood of significant change later in development is substantially reduced

Honeywell uses the 3-view approach to elaborate and analyze implementation options that meet the requirements developed for the system, considering the hardware, software, interfaces and processing needs both on-board and off-board the vehicle. The 3-View approach assures that all aspects of the program from customer needs to laws of physics are addressed. The 3-View approach organizes the project as follows:

- Operational View
  – Defines WORK that customer/product wants/needs to accomplish
- System Requirements View
  – Defines WHAT the system must do to accomplish the work
- Architecture Design View
  – Defines HOW the components and interfaces implement the system requirements
- Management Control System
  – Binds the three views together

On completion of the development of the three architectural views, the System Design Requirements will be developed and provided as a part of the project study reports.

Operational View

The operational architecture view is the users perspective. The objectives and use cases will be defined, and shown within the context of the overall system. The operational view shows how the user's needs are met and establishes the boundaries of the system. Inputs will be received from the monitored subsystems and structures, and the sensors used to monitor them. Outputs will be sent to the flight crew, mission controllers, and ground operations and to other system applications.

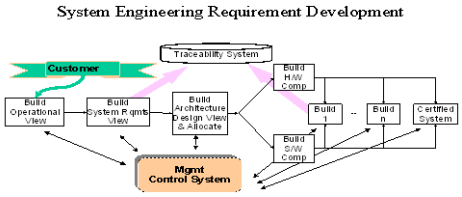

Figure 9. System Requirement Development. *An Architecture Framework process, such as the Honeywell 3-view system engineering provides a method to decompose complex systems.*

The operational view will capture concepts of operations. For example, use cases will trace the flow of information from the time a threat condition occurs, through capture of data about the event, the monitoring and processing required to determine that the event has occurred, and sending notification to the users. Evaluation of how the users apply the information to mitigation actions will help to make sure that the right information is gathered. The operational view is a key to validating the requirements, and establishing the boundaries of the system and its interactions with other systems.

Once the operational view is elaborated, decomposition to functional and physical views can be developed with confidence that the system will meet the objectives of the system. The operational views will be elaborated in the initial part of the program and presented at the mid-period technical interchange meeting so that feedback from NASA can be incorporated in the functional and physical architectures.

Systems Requirements (or Functional) View

The systems requirements view deals with a class of requirement type that modifies or constrains functional behavior and requirements. Typically, these are functionally related requirement types as illustrated below:

- Dynamic Behavior
- Logical Behavior
- Temporal Behavior
- Parallelism & Concurrency
- External System Interactions
- Information Relationships
- Flow of Control
- Flow of Data
- Flow of Material
- Persistence or Information History
- Performance

The philosophy within the system requirements view is to develop the "shalls" subordinate to the system model. This is where non-functional or quality requirements are dealt with and the "fitness for purpose" addressed. Within this category are reliability, maintainability, availability, survivability, durability, testability, supportability, produceability, privacy, security, safety, weight, power, size, design to cost, disposal, MTBF, MTTR, fault tolerance, human factors, legal issues, temperature, noise, vibration, EMI, humidity, and many other factors will be addressed.

Performance requirements such as timing and accuracy for the system will be addressed by utilizing brainstorming activity to aggregate large and diverse amounts of information into manageable partitions. The process will identify and aggregate/categorize the system functionality based on the information from the Operational View. The System View is meant to document the basic functional components that will be used throughout the system development process.

Top-level functional architecture of the system will span vehicle and ground systems, real-time and non-real-time, and in-flight and post-flight processing elements. Using our analysis and optimization models and processes, we will determine what elements are needed and how much of the threat detection problem will be solved in each component. Top-level requirements will be allocated to on-board, ground and post-flight processing components.

The interfaces and bandwidth needed to effectively communicate between elements will be developed, and analyzed to determine how the needs match existing communication infrastructure. Tradeoffs between cost and capability will be made, so that the program can make fully informed decisions prior to the start of the program about what threats will be detected, in how much time, and for what cost.

Architectural (or physical) View

The Architectural View is a physical architecture that includes the hardware and software components and interfaces to the system and the users of information. The physical architecture allocates functions to the actual components where they will be implemented. The focus of the physical architecture will be the on-board components that provide the processing and storage capacities. These architectures will describe interfaces to sensors, signal conditioners, avionics interfaces, processing capabilities, and mass storage.

Physical architecture is done within the context of the overall Architectural View system requirements. They focus on the on-board elements required to obtain the sensor data, process the data, store results, access storage of the data, and provide results to the crew and ground. The on-board sensors, signal conditioning, processing capacity, storage, display and other physical elements will be determined, including existing equipment, new equipment and modifications to interfaces. The physical architecture view is documented for the program and establishes the baseline for cost and schedule estimations.

The physical architecture view also addresses the performance of the system. The physical view will analyze several important attributes and identify architectures that meet the program requirements. Some of these key attributes are:

- Data rates. The threat condition analysis determines how much time may be allowed between an event and when the event is detected, annunciated and acted upon.
- Timeliness. Some events may require very high-rate monitors to ensure that the event is detected, but may not require immediate action. Others may have stringent time constraints that require immediate action. The analysis must consider the time from detection to response, and the architecture must ensure that these overall processing requirements are met. Threat conditions that require immediate action must be processed on-board, while other conditions may be processed on-board when processing time is available, or may be processed on the ground.
- Criticality. Requirements for handling high criticality will be met, addressing reliability and availability requirements with redundancy, error checking or other methods.

The architecture will identify existing equipment and new components required along with assessments of power, weight and cooling requirements. The study will recommend an architecture that best meets the requirements within the constraints. We will apply our experience with doing these types of analysis from commercial and military aviation programs as well as Shuttle and ISS avionics support that we have provided to vehicle prime contractors since the inception of the programs to assure complete and accurate architectures that meet all program objectives.

Architecture trade studies use the analysis and optimization models developed for requirements analysis, combined with Honeywell's Six Sigma and Systems Engineering processes to develop accurate quantitative analyses of program options. Quality Functional Deployment (QFD) models are used to describe, score and rank each option against an array of criteria.

V. Altair: An Architectural Framework Example

The Altair Avionics Subsystem Architecture Study resulted in multiple independent avionics architectural solutions through the use of a 3- Views Systems Engineering decomposition approach. The weight and power saving conclusions, arrived at through the use of 3-Views Systems Engineering and a Honeywell Architecture & Wiring and Analysis Tool (HAWAT), were significant when compared to the original Lunar Design Analysis Cycle-1 (LDAC-1) design requirements.

Throughout the 3-Views System Engineering analysis of the LDAC-1 requirements, assumptions were made when generating artifacts when insufficient data was available. This information was then consolidated and placed into several different Quality Function Deployment (QFD) artifacts to quantify the relative benefits of each approach based on customer requirements. The initial "winning" avionics architecture compared both the power and weight analysis to the Voice of the Customer against all five minimum functional architectures combined. The results indicate that a Honeywell-proposed architecture (Honeywell Configuration 1) represented the least amount of power and weight with five IO modules distributed throughout the Lander vehicle.

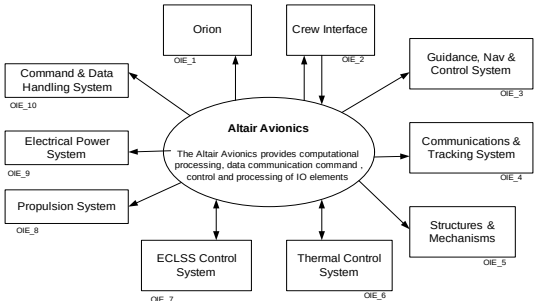

Figure 10. Altair System Boundary Diagram. *The System Boundary Diagram is the first element of understanding the intended operation of the system being decomposed.*

A. System Boundary Diagram

The System Boundary Diagram (SBD) was used to define the Altair avionics system as a whole as well as defining the boundaries of the Altair avionics system in relation to other spacecraft systems. Each subsystem in the avionics then had its own SBD to define the boundaries of those subsystems.

As the study progressed through various iterations of 3-Views analysis, the SBD was revisited and updated to reflect that the team had in fact accounted for every subsystem believed to be a part of the avionics. It was also beneficial to revisit the functions of each subsystem and update those as more subsystem information became available.

B. Operational Information Exchange Matrix

The Operational Information Exchange Matrix (OIE) is a systematic approach to capturing high level, key attributes for each external interface as defined in the SBD. The OIE matrix that was created included the source of where the vehicle was getting its information, the destination of the consuming system, the type of interface, type of data transmitted, and mission criticality including safety.

C. Mission Scenario Diagram

The Mission Scenario Diagram graphically represents the entire mission cycle from ground operations, launch, Earth Departure Stage separation, docking with the CEV, Lunar Orbit Insertion to lunar arrival for each Design Reference Mission. For the Sortie mission, this also includes Lunar ascent, rendezvous and dock, ascent module disposal, return and recovery. Included in the artifacts was a step-by-step walk-through of each phase, along with an evaluation of which avionics subsystems were active and which were on standby. This evaluation was helpful when determining function criticality later on in the trade study.

D. Function List

A Function List was created during the study defining functions per subsystem. The functions were defined as either an input or an output and then mapped to the Master Minimum Equipment List (MEL) for the avionics in the Lander vehicle. This ensured that each of the functions that the team defined was associated with a specific piece of hardware and that the ensuing artifacts were complete.

E. State/Mode Transition Diagram

The State Transition Diagram shows the transition between different system states and what is required to get to the next state. The diagram depicts what mode transitions the Lander vehicle would be allowed to go into and in what order this would be done. The Mode Transition Diagram depicts the various state transitions of the Lander Vehicle taking into account mission phases and vehicle configurations.

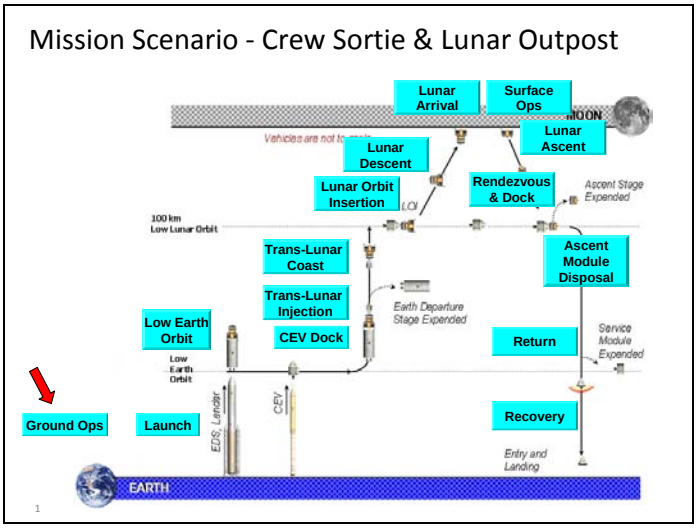

Figure 11. Altair Mission Scenario Diagram. *The Altair Mission Scenario Diagram was derived from the NASA mission Con-Ops. The Con-Ops were augmented to define the individual states for the mission.*

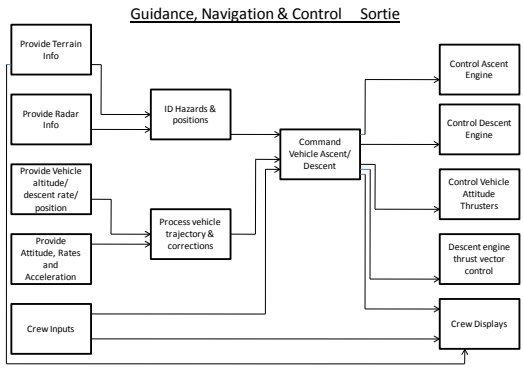

Figure 12. Altair Functional Flow Block Diagram. *The functional flow allowed the development of the data flow and starts the process of sizing the communication paths in the avionics implementation.*

F. Functional Flow Block Diagram

For the main functions, a Functional Flow Block Diagram was created to represent the flow from function to function in a particular subsystem in the Lander vehicle, taking into consideration the uniqueness of the various design reference missions.

G. Data Flow Block Diagram

A Data Flow diagram was drawn to graphically depict the flow of particular subsystem data throughout the Lander vehicle, taking into consideration the uniqueness of the various design reference missions. The various Data Flow diagrams also showed the involvement of various functions in the system and how they were related to the data flow, and color coding was used to identify portions of each subsystem.

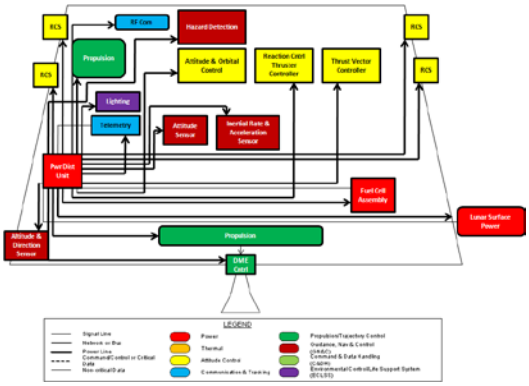

Figure 13. Altair Data Flow Diagram. *The Altair Data Flow Diagram identified the communication between the required functions. This provides the framework to complete the Super Bubble work.*

H. Super Bubble

The traditional Super Bubble was ineffective for purposes of this study. There were simply too many functions to attempt to combine with the various functions and hardware of the Lander. The Super Bubble diagram allows allocation of functional requirements to hardware. The critical functions were identified using a QFD. The QFD compared the functions to the perceived customer requirements. These were then separated into process functions, IO functions and hardware functions and then hardware was allocated.

Control vehicle attitude within commanded limits (AttitudeControl)											
Avionics Processing Modules											
Crew Interface				C&DH partition	GN&C partition						
accept control inputs from crew	accept positional and rotational	provide graphical informatio to crew	provide textual data informatio	process commands and send to appropriate	accept commande attitude acceleratio	measure vehicle inertial attitude	measure vehicle inertial rates	compute vehicle attitude and rate	compute control torques and forces	compute thruster firing sequence	generate GN&C subsystem telemetry

Crew Interface Hardware			Guidance,Navigation and Control Hardware					
Keyboard	Hand Controller	Display Monitor	Inertial Measurem Unit	Star Tracker	Reaction Control Thrusters			
enter textual commands from crew	enter position/ro commands from crew	display telemetry data	output inertial rate measureme at 200Hz	output inertial attitude at 2 Hz	receive RCS Pod 1 thrust commands	receive RCS Pod 2 thrust commands	receive RCS Pod 3 thrust commands	receive RCS Pod 4 thrust commands

Avionics IO Modules											
Sensor IOM				Thruster Pod 1 IOM		Thruster Pod 2 IOM		Thruster Pod 3 IOM		Thruster Pod 4 IOM	
send star tracker status and data to	receive star tracker status and data from	send IMU status and data to system bus	receive IMU status and data from IMU	receive firing sequence for	send firing sequence for RCS1-4 to	receive firing sequence for	send firing sequence for RCS5-8 to	receive firing sequence for	send firing sequence for RCS9-12	receive firing sequence for	send firing sequence for RCS13-16

Figure 14. Super Bubble. *The conventional Super Bubble was replaced with a data base function. The output of the data base function used to group the functions and create architectural implementations is shown here.*

I. Study Architectures

The Altair Avionics study included an evaluation of the LDAC-1 baseline architecture through a graphical representation of hardware and wiring with respect to vehicle module and locations in the vehicle. From this, four iterations of Honeywell proposed hardware, types and locations of IO units, and required wiring were then analyzed to determine if they could potentially reduce overall power and weight when compared to the baseline architecture. IO signal wiring was included, but wiring associated with power distribution to the various units was not.

The LDAC-1 Baseline architecture contains one Remote Multiplexer Unit (RMUX) in each module of the Lander. These are then connected to the subsystem processors and Vehicle Control Data Network.

The "Honeywell Configuration 1" included five IO modules connected to the High Integrity Data Network (combined Vehicle Control Network and High Rate Data Network) through switches. These IO modules were also connected, in a similar fashion as the Baseline, to the subsystem processors.

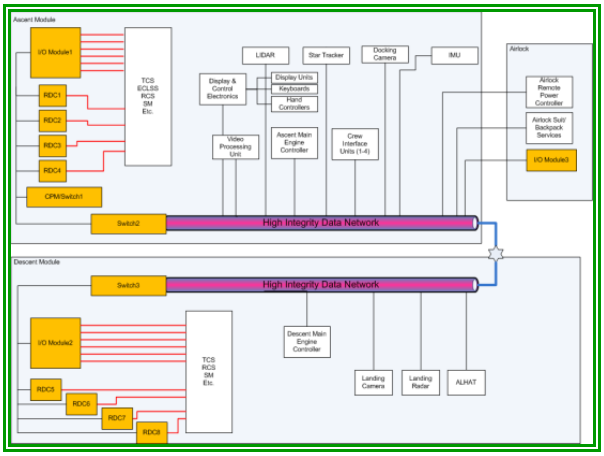

Figure 15. Configuration 4 Study Architecture. *Four architectural configurations were created to be used in the architectural trade based upon the super bubble results.*

The "Honeywell Configuration 2" architecture included 18 Remote Data Converters (RDCs) collocated in the modules near the subsystem processors. The individual RDCs are then connected through switches to the High Integrity Data Network.

The "Honeywell Configuration 3" architecture contains 18 RDCs, half of which are wireless. Some of the RDCs are wire-connected to a subsystem data processor, and also connected through switches to the High Integrity Data Network. The wireless RDCs are spread throughout the Lander modules making them more accessible to hardware in locations that otherwise would add to the overall wire weight.

The "Honeywell Configuration 4" architecture demonstrates a hybrid of IO Modules and RDCs collocated near the subsystem processors. These are also connected through switches to the High Integrity Data Network.

J. Quality Function Deployment (QFD)

The Voice of the Customer was obtained through a Figure of Merit Evaluation on customer needs in order of relative importance. The method of obtaining this data was through a Zoomerang survey submitted to NASA and contractors. The results for the customer needs used in this QFD came from NASA. The needs were listed in order of importance and rated accordingly. The importance rating value of 30 was given for the priority customer need and decreased by 2 per subsequent need. This method was a bit arbitrary and may need to be reevaluated for any future QFD ranking.

Once a power and weight evaluation was conducted on each avionics architecture, using approximate space hardware weights and power consumptions, a representative number was obtained for each. A "1,3,9" rating was then used to represent the sum of each customer need per architecture configuration. The QFD chart calculated the sum of the 1,3,9 ratings with respect to the relative importance of the customer needs. A ranking was then derived from these results and the ideal minimum functional architecture was revealed.

The results indicated that a Honeywell-proposed architecture (Honeywell Configuration 1) represented the least amount of power and weight with five IO modules distributed throughout the Ascent Module, Descent Module and Airlock. This particular result was unexpected when considering that it was being compared to configurations with multiple RDCs, including wireless. More information will be necessary to perform a more exhaustive analysis on this particular architecture for verification of the results. Location of equipment, configuration of IO Modules and RDCs, and quantity of IO signals and types can have a significant effect on the results, so as these inputs become better defined, the toolset will converge on the best candidate architecture.

Maximize, minimize, or target											
Customer Needs	**Importance Rating**	Baseline Config	HW Architecture 1	HW Architecture 2	HW Architecture 3	HW Architecture 4	Baseline Config	HW Archetecture 1	HW Architecture 2	HW Architecture 3	HW Architecture 4
AM Power	30	290	114	146	146	130	1	9	9	9	9
DM Power	28	130	78	110	110	94	1	9	3	3	9
AM Thermal Load	26						1	9	9	9	9
DM Thermal Load	24						1	9	3	3	9
Airlock Power	22	100	16	16	16	16	1	9	9	9	9
Airlock Thermal Load	20						1	9	9	9	9
AM Weight	18	44	33	47	47	40	1	9	9	9	9
DM Weight	16	21	26	41	41	34	3	9	1	1	3
Airlock Weight	14	10	10	10	9	10	9	9	1	1	3
Crew Safety	12						3	9	3	9	1
Technology Readiness	10						3	1	1	1	9
Cost	8	0.88	0.55	0.76	0.76	0.66	3	1	1	1	3
Mission Success	6						3	9	3	9	1
Relative Importance							450	1962	1302	1410	1734
Target Range											

Figure 16. Altair Architecture Trade Study QFD Results. *The QFD trade results produced an unexpected result. Architecture 1 configuration which contains a single large RIU was the highest rated option. All the options were significantly higher that the initial NASA architecture.*

VI. Lessons Learned on the Orion Program

To date the CMMI process has worked well for the Orion program. The Orion program completed a SEMP and seven TDPs to control the separate work packages of the program. The main observation is that it is a continuous battle to assure that the staff is utilizing the approved CMMI process and not trying to get their work done through previous adhoc processes with which they are familiar. The reason for the is that the Orion has had to significantly expand the existing ESEA staff with new hire engineers and engineers brought in from other parts of Honeywell. These engineers have good experiences and possible even a “best practice” not currently deployed, but process conformity is essential. It is just important to use the continuous improvement provisions of the existing CMMI process to incorporate these best practices.

One of the most frustrating realizations in the execution on Orion is the 3-view process tool. Unfortunately during the deployment of 3-view, the selected tool vendor (SLATE) went bankrupt. This caused the tool to be re-written and deployed as a tool named Team Center. The resultant implementation relies heavily on a shared network interface and interface to other COTS tools. For example, the boundary diagram feature utilizes Visio® at the visual capture tool. While the Visio® tool works very well, the real power of a data management tool is to provide tags and links throughout the system. Team Center provides the link capability, but to date the promised feature of impact analysis throughout the database is not realized. An early lesson learned is not to force a tool to implement features and deploy the tool prior to completion. The Visio® linking did not work and was the biggest waste of program time and money. Another frustration is that the tool is so special and complexities that the first line engineers see using the tool. The Orion program has had to implement a work around where many front line system engineers do their work using other tools and then pass off that input to be entered in the Team Center tool by a group of Team Center experts. The SEIT organization has developed program specific training for how ORION is implementing requirements management. Each of the functional groups maintains a little SEIT group that works in the tool to support the CRS and CSCI requirements. Big SEIT maintains core expertise to support others and provide common direction for implementation. Teams do work directly in team center for requirements management. While most of our customers use a variety of other requirements management tools, and even with the afore mentioned struggles, the LM lead PDU team has base lined Team Center as their requirements tool and uses it exclusively for the combined LM, HI and HS members.

An additional lesson is to maintain program specific expertise to lead and manage the team application. Tool specific experts should be made available to the program specific experts. We use to have better level of access that has been slowly eroded and constrained. Previous requirement management tool deployments do not necessarily scale up to meet the needs of a large program like Orion. The issues change with the size so a lead group needs to be able react and provide common direction. We must get experts for the deployment to work on the program and have to go use it. Academic knowledge with no tie out to the program constraints and realities is worse than useless it is also detrimental. Theory is nice but practical applications are not as forgiving. Make them use it.

Another lesson learned is to remember that all tools have their warts. As we all know the best tool is the one you once had or the one you are going to get. Quite complaining and make it work. Don't be shy to tailor it to your needs but get beyond the storming and into the solving periods as quickly as possible. This requires strong team leadership and management support. Remember that the team didn't get a choice in tool selection but told to go make it work. Personal preferences on the tools have no place in the leadership teams. It doesn't help the people who have to make it work to listen to the complaints about tools.

Afinal lesson learned is the interaction with other companies that do not use an Architectural Framework as their primary process for architecture decomposition. For the Orion program, Lockheed Martin has prepared a text based avionics specification with over 3000 “shalls” in the document. Because of this, it is hard to validate the need and proper implementation of the Honeywell subsystem requirement that is synthesized from this document. Without the function list and the data flow diagrams, sizing the system has been more complex with a greater risk that something has been missed in the requirements process that will be found later in the program during integration or qualification and system certification. Although Honeywell is not contractually responsible for these items, we want to do everything possible to assure the success of the program by finding gaps and addressing early.

VII. Conclusion

As presented in this paper, the journey to achieve a CMMI Level 3 rating was a long and laborious road. However, it would not be possible to perform a large-scale program such as Orion without this level of control. The main lesson learned through the process of implementing CMMI is that process is important to the organization and cannot to implemented successfully by anyone but the companies best engineers. The goal to reduce cycle time has resulted in Honeywell’s ability to be a first class performer on a scale unrealized until this time. The implementation of CMMI has allowed us to move from a “hero” company to a company with the ability to respond to any challenge.

VIII. Contacts

Mitch Fletcher, Chief Engineer
Honeywell International Inc.
Defense & Space
19019 North 59th Avenue
Glendale, Arizona 85308-9650
Telephone: (602) 822-3158
Cell: (602) 284-1715
Fax (602) 822-3680
mitch.fletcher@honeywell.com

Bill Airo, Sr. Technical Manager
Honeywell International Inc.
Defense & Space
19019 North 59th Avenue
Glendale, Arizona 85308-9650
Telephone: (602) 822-3357
Cell: n/a
Fax (602) 822- 4330
william.c.airo@honeywell.com

IX. References

[1] 1956, "Definition of System", with Robert E. Fagen, in: *General Systems*, 1 (1956)
[2] Schlager, J. (July 1956). “Systems engineering key to modern development”. IRE Transactions EM-3: 64-66.
[3] Genesis of INCOSE
http://www.incose.org/about/genesis.aspx
[4] DOD (2007) DoD Architecture Framework Version 1.5. April 2007.
[5] Rob Thomas and Phil Cullen (2001). "Building an Enterprise Architecture framework". US Customs Today April 2001.