Leading Teams for Successful Clean Sheet Programs
Johnathan C Lewis, Sr. Systems Engineer, jclewis@rockwellcollins.com
Michael G Manninen, Pr. Systems Engineer, mgmannin@rockwellcollins.com
Harold G. Tiedeman Jr., Pr. Systems Engineer, hgtiedem@rockwellcollins.com
Rockwell Collins, 400 Collins Road N.E., Cedar Rapids, IA 52498

ABSTRACT

Clean sheet programs are characterized by little to no reuse, complex architecture and requirements, organizational complexity, and multiple hardware and software components. These programs encounter cost and schedule problems during execution more often than traditional programs due to lack of domain expertise, organizational challenges, and higher levels of risk in new technology. Clean sheet rotary wing programs, like new integrated cockpits, often bring together hundreds of engineers, all with various levels of domain knowledge and skills. Complex, clean sheet design efforts require not only high performing teams and detailed planning, but also agreed to process and methods for capturing requirements, defining operational concepts, system design, build, integration, verification, and validation. Without strong system engineering leadership, system and subsystem domain knowledge, and agreement on the methodology to execute a program, teams can quickly run into unforeseen challenges during system integration. This paper will focus on three strategies that mitigate some of the unique challenges of clean sheet programs: organization approach, onboarding of systems engineers, and design methodology. The organization approach will describe lessons learned on managing the impact of multiple hardware and software components within the system. Onboarding of systems engineers will describe an approach to address organizational complexity. Finally, design methodology will provide guidance and lessons learned on system engineering process and tools for complex architectures.

INTRODUCTION

It is intuitively understood that the larger and more complex a project is, the level of planning and management required will have to be increased if an organization wants the project to be successful. What are not so intuitive are all the factors that contribute to that “risk” or failure. What factors other than the sheer magnitude of dollars and people can cause a program to have difficulty or fail? Rockwell Collins recognized that teams working large projects were faced with challenges not encountered on smaller programs. This was not the case in all instances, but the issue was pronounced enough that it required some additional study. What was discovered is that projects that exhibit certain key traits were more prone to encounter problems.

Rockwell Collins refers to project that exhibit these traits as ‘Clean Sheet Programs’. So that potential projects can better identify themselves as clean sheet programs, a method to evaluate these traits was developed. The method evaluates the extent to which these traits are present. The evaluation is done during the pursuit phase of a project and is done prior to any formal bid or proposal effort.

It was also recognized that these clean sheet programs were exhibiting similar symptoms as a result of the project traits. These symptoms included problems with organizational structure and lack of a clear understanding of each organization’s roles and responsibilities, a lack of experience within systems engineering, especially as it relates to engineering methods and technology understanding, and poor project performance. Many of the problems associated with poor project performance could be traced back to late identification of requirements and late definition of the system design, such that elements of the design work were not understood and not bid or planned when the project proposal was developed. As a result, if a project is classified as a clean sheet program during the proposal phase, a formal series of system engineering activities are required before a project bid and proposal can be submitted to a customer.
This paper will describe our system engineering best practices for clean sheet programs instituted by Rockwell Collins. Rockwell Collins has witnessed improved program performance on clean sheet programs applying these best practices and are currently gathering evidence to support these claims. However, prior to gathering the hard evidence, Rockwell Collins can claim solid performance and the avoidance of many of the classical program misses on many of our most challenging programs following application of these best practices. Rockwell Collins is confident that these best practices are making a difference.

Design Process and Methodology

Academia, Government and Industry organizations recognize the application of a structured systems engineering design methodology early in a projects concept exploration, requirements development and design is critical to understanding and selecting the best solution from all those that are possible. Study and analysis performed on defense acquisition programs can be extrapolated to multiple tiers of solution iteration in industry. The 2006 Defense Acquisition Performance Assessment (DAPA) Project Report Survey Results showed that requirements instability was the most mentioned problem area and is critical to maintaining program cost, schedule, and performance.

<table>
<thead>
<tr>
<th>Source</th>
<th>Phase Requirements Issue Found</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Requirements</td>
</tr>
<tr>
<td>Boehm (1981)</td>
<td>1</td>
</tr>
<tr>
<td>Hoffman (2001)</td>
<td>1</td>
</tr>
<tr>
<td>Cigital (2003)</td>
<td>1</td>
</tr>
<tr>
<td>Rothman (2000)</td>
<td>1</td>
</tr>
<tr>
<td>Rothman Case B (2000)</td>
<td>1</td>
</tr>
<tr>
<td>Rothman-Case C</td>
<td>1</td>
</tr>
<tr>
<td>Rothman (2002)</td>
<td>1</td>
</tr>
<tr>
<td>Pavlina (2003)</td>
<td>1</td>
</tr>
<tr>
<td>McGibbon (2003)</td>
<td>5</td>
</tr>
<tr>
<td>Mean</td>
<td>1</td>
</tr>
<tr>
<td>Median</td>
<td>1</td>
</tr>
</tbody>
</table>

Rockwell Collins own internal analysis of clean sheet projects showed a correlation between not enough early systems engineering involvement and program cost and schedule performance issues later in time. By not performing enough early systems engineering activities, projects suffer from incomplete requirements, design detail not being captured or communicated, and therefore not bid during the proposal phase, and programs ‘discover’ additional work required when it is often too late and very costly to make adjustments. Often, since design is not captured, design changes made at lower tiers have unintended impact on the system. To mitigate these issues, Rockwell Collins implemented a design methodology that begins well in advance of project proposals being submitted and cost and schedule commitments being made.

Identification

Early in the concept development and pursuit phase, projects are evaluated to determine if they meet the characteristics of ‘clean sheet’ project. This evaluation includes not only an evaluation of the basic technology or existing product maturity, but also looks at the following areas:

- Organizational issues such as organizational complexity (Conway’s Law) and execution team experience
- Domain experience of the organization, the executing team and industry as a whole, is it a new domain area not widely understood by anyone
- Program concerns such as overall program size, risk, security and certification requirements, and dependencies on other projects or technology development

If projects exhibit some of these characteristics to a high degree, then they are characterized as ‘clean sheet’.

Early System Engineering Activities

As described earlier, the lack of early system engineering activities can cause significant rework and cost/schedule growth as requirements and design change or are refined after contract award. For this reason, certain early design activities are required for all projects that are pursued at Rockwell Collins and additional activities are required for ‘clean sheet’ programs. These initial activities are...
completed prior to teams launching on a bid or submitting a formal proposal to the customer. This process is illustrated in Figure 2 showing the sequence and timing of early design steps and additional checkpoints put in place to monitor the progress of a project throughout its life cycle.

Rather than prescribe a specific methodology for capturing architecture, the focus remains on purely capturing the information in the form familiar to the pursuit team. This information is then presented in a Review of the Design Approach (RDA), referred to as RDA1. The purpose of RDA1 is to:

- Clearly communicate the proposed baseline architecture and system interfaces to all stakeholders (e.g. bid team, proposal writers, customers, Rockwell Collins leadership, development team)
- Provide information to include in customer proposals that will give the customer confidence in the solution proposed
- Provide the bidding engineers enough information to complete accurate bids

The RDA1 for every project must contain a minimum set of architecture views to communicate structure, behavior and data flow. Since a number of projects at Rockwell Collins are based on reused system hardware and software, the baselines are well understood and little (if any) additional information other than the required modifications of the baseline need to be defined; however, for ‘clean sheet’ projects, where the baselines are not well defined, the role of the system engineer becomes much more pronounced. The lead systems engineer, or design authority, for the project, must lead the pursuit team in creating the design artifacts that will help the execution team in eventually being successful once the project is won through a competition or funded through internal investment.

The process of creating architecture involves balancing a set of often conflicting constraints to achieve the most successful design approach. Program requirements are one set of constraints. In addition to specific program requirements, it is necessary to take into account other constraints levied by the business and/or the customer. Considerations for budgetary constraints such as cost/price, reuse, business strategic objectives, available technology, exportability, security, safety, and process or manufacturing constraints, all present additional factors within which a design must be formulated. The architectural views created must communicate how these constraints are satisfied and how all project requirements can be met. Although tedious and time consuming, and sometimes challenged by a less then complete understanding of all the requirements, the process and methods used to create the architectural views in all cases provide clarity for stakeholders outside the core pursuit team so that accurate cost and schedule estimates can be made. This then can help insure a higher probability of success.

**Role of the Architect**

As Rockwell Collins became more focused on open systems and product line management, the role of the system architect was created. The architects are heavily used in the pursuit of ‘clean sheet’ projects. In addition to being generally well seasoned and connected engineers, architects take on additional responsibilities and maintain a unique perspective when defining architecture. Their role is illustrated in below, and is unique in that they must maintain broad knowledge of the industry, technology and trends as well as shaping internal company technology roadmaps and investments to improve the company’s ability to capture and execute programs. Architects maintain oversight of key projects, but generally become less involved as the architecture is implemented and risk of the team running into unanticipated roadblocks has subsided. One primary reason to maintain architect involvement is to ensure that product lines are maintained and evolved as planned. This is particularly important as many subsequent projects create dependencies on these developing product lines and a change in direction can have an unanticipated affect. Hence
it is important that architecture is accurately represented and maintained through the project life cycle.

Design Artifact Definition

In order to maintain a clear and consistent guidance regarding architecture views, a combination of best practices was tailored to form guidance for various DoDAF, UML, and standard architecture views to be developed. Though these views may not provide all of the needed detail for every clean sheet and reuse program, they provide a majority of the information within various functional, static, and behavioral views as shown in Figure 4. Of significant importance are interfaces, allocations, high level system and software architecture, and critical system data flows and timing.

Non-Advocacy Peer Reviews

So once the architecture is developed, the pursuit team should then be able to throw it over the fence and go on with life, right? No, not really. Refer to Figure 2 for a view of the complete sequence of events in the life of a clean sheet program. Recognizing that even the best engineers can misinterpret or read different meaning into any form of communication, including the best architecture model, various checks and balances are needed to make sure everyone is on the same path. This includes engineers and engineering leadership who are tasked with estimating work and executing projects.

Rockwell Collins has instituted these checks in the form of cost and schedule bid reviews and system development reviews, each performed by non-advocates of the project. The purpose is to get an unbiased assessment of the bid, requirements, design and planning artifacts early in the project, such that change can be affected before it is too late. The bid non-advocate review, or Bid Peer Review, purpose is to have experienced peers from engineering and management ensure that no major errors, omissions or inconsistencies exist in the bid. The Bid Peer Review also verifies that the bidders followed the instructions given for the pursuit team, including the requirements and design guidance provided in the approved review of the design approach. If there are problems, the Bid Peer Review team provides recommendations at the review and the pursuit team has to provide resolution before the bid can be submitted to engineering leadership for approval.

The System Development Peer Review (or SDR) is performed at roughly the same time as the Bid Peer Review; prior to the proposal being submitted to program management and executive leadership for approval. It is critical that the team has a credible plan and approach to project execution at this stage as the company prepares to commit to its customers that it can perform to the proposed cost and schedule. It is also equally critical that the requirements and architecture are accurate and complete and
programs was first instituted by the Department of Defense. One proven approach to organizing large teams and engineering managers, program managers and customers.

avionics program with hundreds of engineers, dozens of have a high performing organization on a clean sheet than 6 people. This makes one wonder how it is possible to state that high performing teams usually consist of no more

defined as cross-functional teams that are formed for the development programs become successful. “IPTs are and Process Development (IPPD), helped many large organizations.”

In “The Wisdom of Teams: Creating the High-Performance Organization”³, J. R. Katzenbaum and Douglas K. Smith state that high performing teams usually consist of no more than 6 people. This makes one wonder how it is possible to have a high performing organization on a clean sheet avionics program with hundreds of engineers, dozens of engineering managers, program managers and customers.

One proven approach to organizing large teams and programs was first instituted by the Department of Defense (DoD) in the early 1990s. Integrated Product Teams (IPTs), the core organizational idea to support Integrated Product and Process Development (IPPD), helped many large development programs become successful. “IPTs are defined as cross-functional teams that are formed for the specific purpose of delivering a product for an external or internal customer. IPTs members should have complementary skills and be committed to a common purpose, performance objectives, and approach for which they hold themselves mutually accountable. Members of integrated product team represent technical, manufacturing, business, and support functions and organizations which are critical to developing, procuring and support the product. Having these functions represented concurrently permits teams to consider more and broader alternatives quickly, and in a broader context, enables faster and better decisions.”²³

The main idea behind IPTs was to get people’s perspective to change from one that focuses their supports to functional organization to one that supports the product or program. “Team members work together to achieve the team’s objectives.” According to Kenneth Crow of DRM Associates, “Key characteristics of IPTs include:

- Team is established to produce a specific product or service
- Multidisciplinary - all team members/functions working together towards a common goal.
- Members have mutual, as well as individual accountability
- Integrated, concurrent decision-making
- Empowered to make decisions within specific product or service goals and project parameters (budget, schedule, and business results).
- Planned integration among teams towards system goals.”²³

Rockwell Collins uses IPTs on most programs and we believe that clean sheet programs are especially vulnerable to organization structures that do not support the key characteristics of IPTs. Though most programs use IPT structures, some have not effectively made use of IPTs. Some large clean sheet programs have attempted to organize IPTs with engineers loaned in from a larger matrix organization. An example matrix organization structure is shown in Figure 5. In this organization, engineers often support multiple programs and also have strong product line ties. This organizational strategy does not meet the key characteristics of IPTs and IPPD, because the engineers have competing priorities that makes them less efficient. Rockwell Collins has experienced challenges on some large clean sheet programs when IPT members are part of a matrix organization and not working towards the same objectives. Rockwell Collins has found the matrix organizations are best used to support mature products and can be very effective when continuous evolution of existing product lines is the primary function of the organization.
Conway’s Law and Collocation

Rockwell Collins learned some valuable lessons on collocation of engineers on clean sheet programs. We believe that collocation of at least a core team is critical to the success of clean sheet programs and should be considered a key characteristic of IPTs. Collocation is especially critical in the early stages of a program.

The need for organizing clean sheet programs into collocated IPTs is supported by Conway’s Law. Conway’s Law, written by Melvin Conway in 1968, asserts that “Organization’s which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations.”[7] If modular designs are desired, then it is acceptable for there to be separation between teams since interfaces between modules tend to be less complex. For teams designing complex integrated products, communication barriers tend to negatively manifest themselves during integration, proving Conway’s Law. Another version of Conway’s law states, "If the parts of an organization (e.g. teams, departments, or subdivisions) do not closely reflect the essential parts of the product, or if the relationship between organizations do not reflect the relationships between product parts, then the project will be in trouble. ... Therefore: Make sure the organization is compatible with the product architecture."[8] There is supporting evidence of Conway’s Law that has been published by a team of Harvard Business School researchers.

Their study reveals significant differences in modularity, consistent with a view that distributed teams tend to develop more modular products. Distributed teams have proven to result in modular architectures. For this reason we believe that for highly integrated avionics programs, it is critical that teams be collocated.

In our experiences, organizations that have poor communication structures will produce poor integrated designs and have difficulty with integration. On one large, clean sheet avionics program, teams were geographically collocated, but distributed among the different buildings. Even this minor separation of engineers, who needed to collaborate early and often, created challenges for the teams to communicate effectively. System interface definition suffered on this program and it took some effort to recover. The program temporarily collocated the key engineers, facilitating communication that did not happen earlier in the program. After months of facilitated communication, the designs of the many subsystems came together and integration proceeded. In another example of a large clean sheet program, a single building was used for the key engineers from all domains supporting the program. Despite its many challenges, the program performed very well during initial integration and through customer testing, delivering the end product on schedule.

As rotary wing avionics systems grow in complexity, teamwork within the system engineering team is essential. Though utilization of the latest telecommunications
technologies continues to bridge the gaps of physical separation, “facetime” through collocation is, and always will be, an important aspect of the engineering process. Having co-located teams, provides clean sheet rotary wing programs an opportunity to realize higher efficiencies in the system engineering process.

**Facilitating Design Decisions**

There are two roles that Rockwell Collins believes are critical for execution of clean sheet organizations. They are the Design Authority and the Process Authority. Most programs have a Design Authority, but few have the formal role of Process Authority. We believe that programs with both have a higher chance of success.

**The Role of the Design Authority**

On a large clean sheet program thousands of design decisions are required following requirements capture and architecture definition. Without a clear understanding among IPT members of how and who will make design decisions, work will not progress and bad decisions may be made causing rework later in the program. Design Authorities are mandatory leadership roles for IPTs. There is often a hierarchy of decision making on clean sheet programs. For system wide decision, it is critical to define a system level Design Authority. This person is often the System Level Architect or Chief Systems Engineer. A Design Authority should also be defined for the individual domains or subsystems. All program Design Authorities on the program should work together and decide on rules and methods for sharing design responsibility. Design Authorities may also lead a Configuration Change Board (CCB).

Rockwell Collins learned that on programs where the design authority is not assigned or not accepted as the authority by the team, many design and integration problems are encountered. In areas where common design is required between domains, the design authority should enforce the design, and be the final decision maker. The Design Authority can be multiple people as long as there is agreement how to function as a team. Whether a program is Clean Sheet or Re-use, this role is critical to programs since all programs have some level of design decisions to make.

Attributes of a System Level Design Authority include:

- Broad system wide knowledge including top level requirements, requirements allocation, and system interfaces.
- Good understanding of the system context, customer needs and system Use Cases
- Strong leadership and decision making skills

Attributes of a Subsystem Design Authority include:

- Domain subsystem and product line knowledge
- Good understanding of functions that cross subsystems
- Knowledge of domain HMI standards
- Formal education in Human Machine Interface or domain experience (e.g. pilots, flight instructors for clean sheet avionics programs)
- Strong leadership and decision making skills

For programs that require a great deal of Human Machine Interface (HMI) definition, it is useful to have a single person fill the role of HMI Design Authority. This person can then ensure that the man-machine interface is consistent across formats supporting all system functions. Again, if the program is large enough and supported by multiple domains, there may be multiple levels of HMI Design Authorities, all who need to work together as a single IPT.

Attributes of a HMI Design Authority include:

- Knowledge of domain HMI standards
- Formal education in Human Machine Interface or domain experience (e.g. pilots, flight instructors for clean sheet avionics programs)
- Strong leadership and decision making skills

**The Role of the Process Authority**

In addition to a Design Authority, a Process Authority is a critical role on a clean sheet program. The role of the Process Authority is to help define and enforce methodologies. The Process Authority defines “How” the team will execute the technical process. Processes, like the Technical Consistent Process (TCP) used at Rockwell Collins, are universal and often very similar from company to company and program to program. The RC TCP is a traditional V model. Unlike Processes, methodologies to support the process are “local” or specific to a team and program. We’ve found that programs where methods are not defined and enforced suffer during execution. Tools to support methods are also critical and need to be common among all domains on a program.

A Process Authority makes sure the team has an understood and agreed-to methodology and toolset for executing the program plan. Like the hierarchy for Design Authorities, there should also be Process Authorities for each level or the organization. This System Level Process Authority can lead a process committee that assures alignment of methods, tools and languages on the program.

Attributes of a System Level Process Authority include:

- Broad system wide knowledge including top level requirements, requirements allocation, and system interfaces.
- Knowledge of process, tools and languages to support methodology
- Good understanding of the system context, customer needs and system Use Cases
- Strong leadership and decision making skills
Attributes of a Subsystem Process Authority include:

- Domain subsystem and product line knowledge
- Knowledge of process, tools and languages to support methodology
- Good understanding of functions that cross subsystems
- Strong leadership and decision making skills

For re-use programs, a Process Authority should evaluate tool and methods used to develop the previous designs and may make improvements only if required. Incremental tool and methodology improvements are warranted if supported by broader enterprise wide initiatives and proven to improve efficiencies.

Rockwell Collins does not believe that the Design Authority and Process Authority need to be the same person. They do need to be coordinated with one another. In fact, it is better to have these roles filled by different people since they both tend to be full-time roles on large clean sheet programs.

**Onboarding of Systems Engineers**

To this point the paper has detailed various key program roles including the architect, design authority, and the process authority. These are critical roles on any program, however, the sustainment and detailed work is typically carried out by a team of systems engineers. Of specific interest is the role of the platform system engineer. The program leadership team, show in Figure 6, provides guidance and leadership to the engineering teams on the program.

![Figure 6 - Common Program Structure](image)

Rockwell Collins does not believe that the Design Authority and Process Authority need to be the same person. They do need to be coordinated with one another. In fact, it is better to have these roles filled by different people since they both tend to be full-time roles on large clean sheet programs.

**System Engineering Environment**

In this environment, the platform system engineering team in charge of system level requirements, design, integration, and test. The team provides oversight and guidance to subsystem engineers and support organizations (maintainability, safety, information assurance, etc.). Additionally, the responsibilities of the platform system engineer include, but are not limited to:

- Domain knowledge of systems and products
- Understand all aspects of the system engineering process
- Leadership
- Innovation
- Decision making
- Roadmap development
- Coordinate, communicate, and champion strategies

As Rockwell Collins engineering organization continues to grow at a rapid pace, these platform systems teams become staffed with a significant percentage of less experienced engineers. All of these responsibilities can be overwhelming and unrealistic for junior engineers to accommodate. This section will focus on an approach used on a recent rotary wing program which focuses on quickly maturing platform systems engineers by clearly defining and communicating expectations of each engineer’s role and responsibility on programs.

**SME Approach**

With a large focus on maintaining Work Breakdown Structure roles and accountability for customer deliverable items, platform system engineering teams are often organized with a document-centric focus. This method has an inherent benefit of enabling a one-stop point of responsibility. That is, when it comes time to delivering documents, it is easy to know who owns each product. Though critical for success, it also has some undesirable side-effects. Because the engineer(s) becomes focused on meeting the milestones associated with the delivery of a broadly scoped document, the focus can quickly become on meeting cost and schedule with insufficient focus on the contents of the material. When considered with the knowledge base of a team of junior platform engineers, the unintended result becomes a team well versed in document deliveries without a deep understanding of the system which they support.
In order to correct this, a revised approach was taken. Because the platform system engineering team so often interacts with internal and customer stakeholders who are well versed in subject matter, it made sense to assign individuals to subject matters in which their talents and interests reside. In effect, the platform system engineering team adopted a Subject Matter Expert (SME) approach. The SME approach is a method utilizing key individuals each with a responsible area/specialty/topic. The SME is responsible for keeping current with new developments and the subject matter field. An example SME area is the ever growing and complex Flight Management System domain; where being able to speak the language requires extensive knowledge and experience. The SME approach will be described in terms of knowledge base, system engineering task flexibility amongst individuals, and impact on integration and test.

In order to ensure document responsibilities were not ignored, individuals retained their document ownership status while gaining SME roles. This approach ensures the output of the platform requirements, design, and test do not get overlooked while the team is working closely with the various stakeholders (customer, subsystem, program management, etc.). Along with supporting the various stakeholders, the SME is responsible for content within each of the platform deliverables, any documents they own, integration and testing of their SME area, and review of the relevant subsystem documentation. SME roles were assigned based on individuals’ strengths and interest, and were assigned out in functional segments to correlate with the system specification and to follow Conway’s law. When assignments were made, roles, responsibilities, and expectations were made clear. Per Stephen Covey, “The cause of almost all relationship difficulties is rooted in conflicting or ambiguous expectations around roles and goals.” This is very much applicable to system engineering tasks as well.

The approach broadened skills of our platform system engineers and had other positive results which included a very quick increase in the platform system engineering team’s knowledge, an increase in work task flexibility, an increase of each individuals exposure to a variety of activities, enablement of more successful integration and test, and a “one stop shop” for functional knowledge.

Knowledge

With complex avionics systems consisting of dozens of LRUs/WRAs, software applications, and non-trivial architectures, less experienced engineers can quickly become overwhelmed with a sea of data. What tends to happen is engineers learn random pieces of the system and complete the tasks they are told to do. The so called “magic number” of seven years to become a systems engineer often takes much longer than any organization can tolerate. To break this paradigm and bring engineers up to speed faster, our focus shifted to a better understanding of the system engineering process, that is, a complete picture, and an in depth plunge into a large functional area. The amount of knowledge systems engineers acquired over time seemed to increase rapidly. Given the responsibility for an area rather than one particular problem, ownership of the subject matter jumped dramatically. This seemed to be a self-fulfilling prophecy where the SME would continue to grow exponentially in knowledge and have an overall understanding for the subject area and how the system as a whole enables the functionality.

Flexibility

Initial impressions of the SME approach may be that it is artificially limiting in what system engineers can do to learn. While this is a somewhat valid point, the results have shown that the engineers learn more and learn it faster, than with a document centric approach. One key advantage is the ability to have most any systems engineer work on any piece of the process when in a bind. The key to this is the fact the each engineer is not only learning the complete system engineering process, but they understand it because they live it every day. They understand the entire context around their specific subject matter and in time will move on to the next piece(s) to continue to be grow and be effective broad-based systems engineers.

Increase in Exposure

Though the approach limits short term exposure to various functional areas, the SME approach increases individuals’ exposure to a variety of system engineering tasks. Each systems engineer takes ownership of one subject matter but works each angle/task of the function. As mentioned earlier in the paper, there are various views developed in the early stages of the system engineering process. Starting with “Capture Originating Requirements”, show in Figure 7, all the way through the end of the cycle to product delivery, the SME takes ownership of the subject area to ensure the intended product is delivered to the customer.

**Figure 7 - System Engineering Vee Model**

Successful Integration and Test

One key area of growth for systems engineers is growing knowledge through validation and verification. With the outlined SME approach, engineers are able to work through problems from conception all the way through validation and verification. What this provides is a holistic view of the system engineering process and how the various engineering
tasks are complimentary. For example, as system engineers are working on the requirements phase of a program, they will have a better grasp and ownership of creating verifiable requirements. The knowledge of not only what needs to be tested, but how it is tested is in the back of their minds ready to influence good requirement writing; even detailing out test cases during the task of documenting the verification method.

This same philosophy trickle through the entire process. As the engineer is involved in design work, he or she knows exactly the requirements and their intent and can detail a system to solve the given prose. As design evolves into implementation, the engineer is able to efficiently detail test procedures without having to dig into detailed information or rehash the validity of the originating requirement.

During the verification testing, the systems engineer has a good understanding of how the SME function/product works and is able to perform system level testing with an effective outcome having understood the requirements and design.

**One Stop Shop**

As systems continue to grow in complexity and schedules remain critical to program success, its vitally important to be able to quickly respond to design actions, problem reports, and customer questions. The SME approach adds the ability, over time, to surround the program with skilled experts who can quickly respond with confidence in the subject matter. This quick problem resolution not only saves time and money, but puts aside deeper questions regarding team capabilities.

**SME Method Results**

It was also initially thought and concluded that some documents do not lend themselves to being shared by SMEs with one overall owner. Mainly, ICD related documents/repositories were maintained by an ICD SME to maximize efficiency and minimize the overhead placed on specialized engineering tools.

Positive internal stakeholder response to the SME approach has been overwhelming. Truly, that is an understatement when it comes to the feedback of our customer. Response from the SMEs themselves has also been consistently positive.

To ensure continued growth and to mitigate personnel rotations, cross-training was added 12 months into the implementation of the SME approach.

**Conclusion**

The complexity and novelty of clean sheet programs warrants additional emphasis on system engineering through the entire project life cycle. Without early attention to architecture, there is a higher than normal risk that a project will encounter problems in maintaining cost and schedule performance. This is attention to architecture is essential to provide leadership and guidance for engineering teams at all tiers of the system. Additional checkpoints are used to raise awareness of how a project is performing against the architectural guidance provided. With this value added attention to architecture, engineering teams can and do increase their chances for success.

During execution, clean sheet programs that collocate teams and follow the original intent of Integrated Product Teams have a much higher chance for success. Designating a Design Authority and a Process Authority helps IPTs make sound design decisions and ensure that all domains are efficiently collaborating. At Rockwell Collins, we have seen improved teamwork and efficiencies on programs that have instituted these organizational practices.

With the mentioned onboarding process, it was found that the engineers were able to gain deeper knowledge more quickly than traditional approaches and have more impact to the overall progress of development. These platform system engineers continue to gain knowledge and have a positive impact on the program. With this knowledge they can have a greater impact from the beginning of programs, the time in which making changes are more manageable and has a large positive impact to the program.

Overall, Rockwell Collins and our customers are benefiting from the improved performance of our teams conducting clean sheet programs. We are confident that information being gathered on clean sheet program performance will provide evidence supporting our system engineering best practices. We look forward to providing an update to this paper with this evidence and more lessons learned in the near future.

**References**

7. Conway, Melvin E., "How do Committees Invent?", *Datamation*, 14 (5) April, 1968, pp. 28-

