26th Annual INCOSE International Symposium (IS 2016)
Edinburg, Scotland, UK, July 18-21, 2016

A Process for Capturing the Art of Systems Engineering

"Skip" Clark V. Owens III
NASA-LSP Kennedy Space Center
Mail Code: VA-G2
Kennedy Space Center, FL 32899
321-867-2935
skip.owens-1@nasa.gov

Carrie Sekeres
Embry-Riddle Aeronautical University
386-275-9015
sekeresc@my.erau.edu

Yasmeen Roumie
Stuyvesant High School
917-669-3554
yasmeenroumie@gmail.com

Abstract. There is both an art and a science to systems engineering. The science of systems engineering is effectively captured in processes and procedures, but the art is much more elusive. We propose that there is six step process that can be applied to any systems engineering organization to create an environment from which the "art" of that organization can be captured, be allowed to evolve collaboratively and be shared with all members of the organization. This paper details this process as it was applied to NASA Launch Services Program (LSP) Integration Engineering Branch during a pilot program of Confluence, a Commercial Off The Shelf (COTS) wiki tool.

The Art and Science of Systems Engineering

There is both an art and a science to systems engineering. In the NASA paper "The Art and Science of Systems Engineering" (Bay et al. 2009), systems management is described as the science of systems engineering, while technical leadership is identified as the art of systems engineering. The management of systems is easily captured in the processes and procedures that an organization accumulates and implements from year to year and from project to project. These bits of knowledge are immortalized in formal documents and passed down from systems engineer to systems engineer. The art of systems engineering is much more elusive. Technical leadership is an art that is not easily defined, cannot be easily taught and is quite challenging to bound. Organic would be another way to describe an art. Across all professions, an art is something that is tailored by the individual and therefore is much more complex and harder to directly pass down from one individual to the next. What works for one person may not be the most effective technique for the next. Art also has many different pieces and techniques

from which an individual must evaluate and choose to use in any given situation. Science is more procedural and it is the art of a discipline in the hands of a master craftsman that implements that science. So how do we go about training and developing master craftsman in systems engineering?

One of the main components of systems engineering is the famous “why.” “Why” is used by the systems engineer as a way to get from what a customer or a stakeholder thinks they want to what they actually need. But the “why” is also an essential aspect of the art of the systems engineering. For example, the rationale statement that often goes along with an individual requirement is the “why” behind the requirement. But what about the “why” behind a technical leadership tactic or a method for driving the technical team to consensus? The “why” behind something can’t be captured unless it is within the context of the “what.” In the example of the requirement rationale statement, the rationale is side by side with the requirement (the “what”), so the two are linked together. However, requirements are part of the science of systems engineering and are much more naturally documented in a formal manner. How do we as systems engineers go about documenting the “why” behind the art of systems engineering contextually with the “what?”

A Process for Knowledge-Capturing the Art of Systems Engineering

We propose that there is a process that can be applied to any organization to create an environment from which the "art" of that organization can be captured, be allowed to evolve collaboratively and be shared with all members of the organization. A process for capturing the art of systems engineering is as follows:

Step 1: Create a Functional Architecture
Step 2: Identify Current Knowledge Capture Methods
Step 3: Map Existing Knowledge Capture Methods
Step 4: Enhance Knowledge Capture Methods
Step 5: Introduce Enhanced Knowledge Capture
Step 6: Evolve the Knowledge Capture

The remainder of this paper will use the Integration Engineering (IE) Branch within NASA's Launch Services Program (LSP) as an example of how these six steps can be implemented. Just as with Systems Engineering, there is both an art and a science to these six steps. Simply following these steps is not a guarantee of success. The process itself is iterative and each organization will need to tailor this process and its implementation to best suit the specific environment of the organization. These six steps have recently been applied to LSP's IE Branch as part of a pilot program carried out during the summer of 2015. This pilot program was testing the effectiveness of a collaborative wiki tool called Confluence and was led by two summer interns, Carrie Sekeres and Yasmeen Roumie, who are both co-authors on this paper.

Why Confluence?

Wikis have been around for quite a few years and have proven to be a very effective way to informally capture large amounts of technical information, with Wikipedia being one of the most recognized examples of a large-scale wiki. Several NASA Centers, including The Jet Propulsion Laboratory’s JPL Wired (Rober, 2009) and The Goddard Space Flight Center (GSFC), have implemented center-wide wikis as a way to capture tribal

knowledge across their organizations. Both JPL and GSFC have evaluated the commercial wiki software options and found that the collaborative wiki software suite called Confluence to be the most suitable option. In addition, NASA has also decided to publish its latest version of the NASA Software Engineering Handbook in form of a wiki using Confluence (NASA, 2016). The Confluence wiki environment is an example of a wiki tool that combines the traditional knowledge capture aspects of a wiki with social networking tools to establish a collaborative environment. Confluence appears to be a good fit for attempting to collect the organic pieces and parts that make up the art of systems engineering, and, for this reason, it has been the subject of a small pilot within NASA's Launch Services Program (LSP) for the last several months.

Confluence is a web or server-based, centralized team collaboration platform, structured as a wiki. Each instance of Confluence has separate "spaces" (a Confluence specific term for a wiki structure) and pages within these spaces. Users can make customized templates of pages they intend to use often, such as meeting notes, tutorials, guides, and peer reviews. This collaborative software has a simple user interface, which helps minimize the learning curve that usually comes with new tools. Email notifications remind users to use the tool and are sent when the pages they are watching are edited, when they are mentioned, or when a new question is asked. Atlassian, the company that created Confluence, has a marketplace with a large variety of add-ons that can be used to customize a group's experience with the tool. Custom add-ons can be programmed using Atlassian's Software Development Kit (SDK) and developer tutorials. There are many macros that are already built into Confluence, including some that report the usage statistics of a space. Putting up a leaderboard of contributors makes group members aware of their colleagues' contribution frequency, which will help to motivate them to contribute on a more regular basis. Most importantly, Confluence has the ability to integrate with SharePoint, which is LSP's current program-wide collaboration tool of choice. All information added into Confluence will show up in a unified search across both SharePoint and Confluence once the add-on capability called SharePoint Connector is setup and the two systems are integrated.

A common question that comes up when discussing tool selection is why other wiki platforms (free or open source wikis) or SharePoint wasn't used instead. In our case we were already using SharePoint and chose to use Confluence as a way to supplement SharePoint, rather than replace it completely. Knowledge capture requires two very critical components that Confluence does well, contextual information and social networking. When trying to capture something as challenging as the "art of systems engineering," there isn't a well-defined boundary or set of items that encompass an art. Context becomes key. Knowledge capture is very ethereal, which makes it extremely difficult to perfectly capture. In order to capture an "essence" or an "art," the capture method must be both powerful and flexible. Confluence has a mix of contextual based tools and social networking options that make it an ideal choice for an effort like this. The rest of this paper will focus on the major areas of systems engineering knowledge capture our team utilized during the pilot and highlight the technical aspects of Confluence that aided in that knowledge capture.

Step 1: Create a Functional Architecture

The first step in the process we have established is to identify the organization's main responsibilities or functions by creating a functional architecture. The NASA LSP's purpose is to procure and manage commercial launch services for NASA. When NASA

LSP procures a launch service we are not "buying a rocket," but instead we are purchasing a service that includes everything (including the use of a rocket) that is needed to process and launch a spacecraft into space. Most robotic space missions across NASA come to the LSP to gain access to space through the use of a launch service. We have identified three main functions of the IE Branch: Support Procurement of Launch Services, Support Management of Launch Services and Provide Advanced Mission Support.

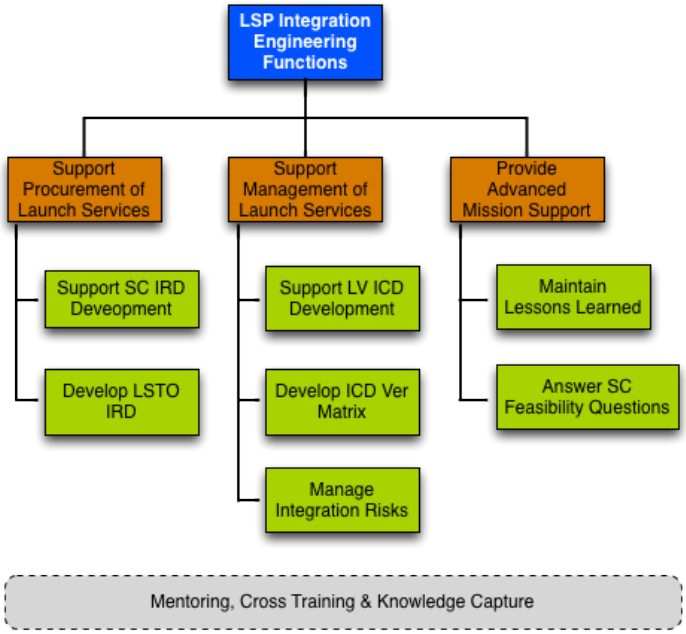

Figure 1. LSP IE Branch Functional Architecture

The first and arguably the most important task is the procurement of the launch service. The LSP Integration Engineer is responsible for assembling a reduced Interface Requirements Document (IRD) for the purposes of the launch services procurement. This document is leveraged heavily from the spacecraft project's full IRD, but, in order for the document to be useful during a competitive procurement, it must be non-launch vehicle specific and the requirements within must be free from specific implementation direction as to not overly constrain the potential bidders and maintain a competitive environment. This reduced IRD for the procurement of a launch service is one of two main systems engineering products that the LSP Integration Engineering group is solely responsible for creating.

Managing the launch services is the largest function we have within the IE Branch. Most of the work we do within the branch is covered by this function. The first sub-function under managing launch services is supporting the development of the Launch Vehicle (LV) Interface Control Document (ICD). The LV ICD is where all of the interface requirements for a specific mission's launch service are documented. While the LV ICD is not a product that LSP produces, we are involved heavily in the development of the requirements and are the technical authority for its approval, along with the spacecraft (SC) project.

The second sub-function or product that the IE Branch is responsible for is an independent verification matrix. Once a launch service is put on contract, the Launch

Vehicle Contractor (LVC) is responsible for developing and maintaining the ICD (the first sub-function). This ICD is then used to define all the interfaces and common environments between the spacecraft and the launch vehicle and is the basis for almost everything that takes place in the years leading up to the launch. This approximate time frame, which starts shortly after the launch service is awarded and extends through launch, is called the mission integration cycle. Even though the IE Branch does not write and maintain the ICD, we are responsible for creating and maintaining our own independent set of verifications against all the requirements in the ICD.

The final sub-function is to manage integration risks. In addition to providing NASA a launch service, the LSP also serves as the mission's insurance policy against launch failure. The way we protect against failure is by managing the risk that is associated with launching a spacecraft into space on a launch vehicle. Risk mitigation is a key component to managing risk and a large portion of mitigation is accomplished by performing both insight and oversight of our LVCs. The IE Branch is the primary path for our spacecraft customer to interface with our LVC as the mission progresses through the mission integration cycle. Oversight is mostly a program management function, consisting of review and approval of the LVC products and deliverables. Insight, independent analysis, and verification are where we start getting into more traditional systems engineering functions. The LSP insight role includes maintaining an in-depth knowledge of the contractor's engineering review and approval practices, as well as their engineering analysis capabilities. LSP engineers will also perform independent analysis throughout the mission integration cycle.

The final function of the IE Branch is to provide advanced mission support. Advanced mission support is defined as any engineering or programmatic activities that take place before a spacecraft mission is taken through the launch service procurement process (i.e. before development of the spacecraft IRD and the Launch Services Task Order (LTSO) IRD are started). Two sub-functions under advanced mission support are maintaining and communicating lessons learned and answering SC feasibility questions. Since LSP manages most of NASA's unmanned commercial launch services, we have a wealth of experience concerning spacecraft integration. The best time to apply spacecraft development lessons learned is as early in the mission integration process as possible. In this case, we try to communicate lessons learned before we even begin supporting spacecraft IRD development. Early in the mission development and mission feasibility phase, questions come up concerning compatibility with various launch vehicles. The LSP IE Branch supports is then responsible for answering these technical feasibility questions from spacecraft projects.

The final item depicted in the LSP IE Functional Architecture is an underlying function that every technical organization has, whether it is formally recognized or not. That function is a combination of mentoring, cross training and knowledge capture. This function is all about maintaining and growing a technical capability, the technical capability that is required to carry out all the other functions that the IE Branch provides. This "knowledge" function is shown as detached from the main architecture because it permeates everything and does not fit in one specific place in the hierarchy.

Step 2: Identify Current Knowledge Capture Methods

This second step is very important and is a step that many will be tempted to skip altogether because it is believed the organization already has all the methods and tools required to create an environment suitable to capture the art of systems engineering. It is critical to understand the current state of knowledge capture and knowledge sharing

even it is known to be insufficient. As we will discuss more in step 5, any changes made to the existing knowledge map must be made methodically, else a reduction in capability could jeopardize the main goal of building upon and expanding knowledge capture capabilities that are already in place.
Within the IE Branch, we already had the following knowledge capture and knowledge sharing mechanisms in place:

- SharePoint Site
  - Lessons Learned Database (Database within SharePoint)
  - Branch Meeting Minutes (Microsoft Word Documents)
  - IE Training Resources (Microsoft Word Documents)
  - Peer Review Meeting Invites & Documents (Microsoft Word Documents)
  - Processes & Guides (Microsoft Word Documents)
- Verification Matrix Template in FileMaker Pro (FileMaker Pro Template File)

Step 3: Map Existing Knowledge Capture Methods

The next step in the process is to map the organization's existing knowledge capture methods against the functional architecture. We choose to use our existing functional architecture diagram and overlay the elements identified in Step 2 as red circle elements with a legend.

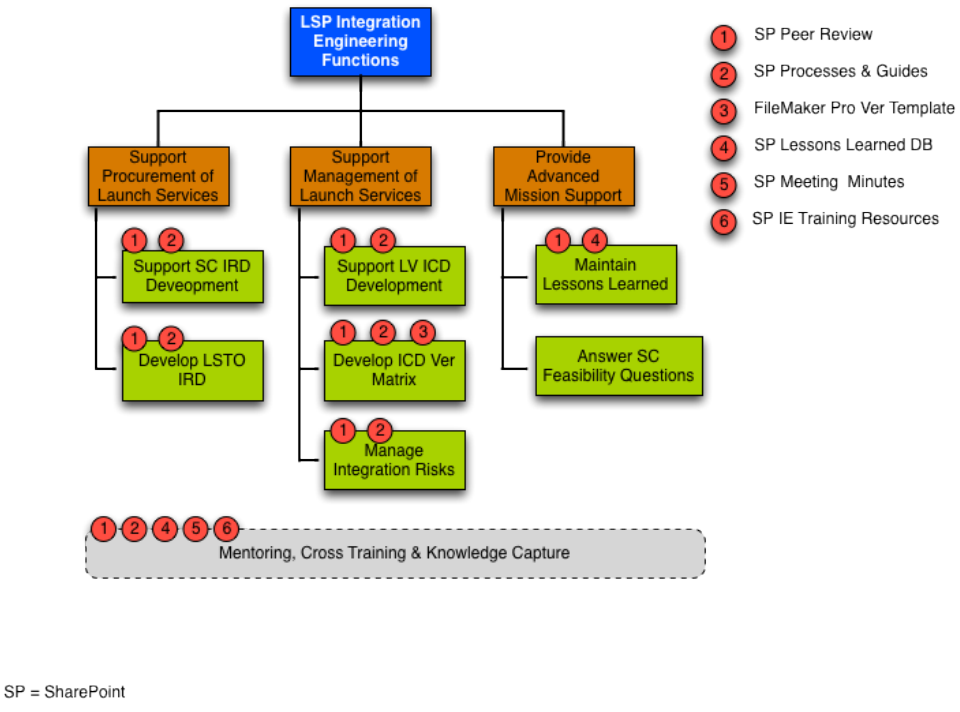

Figure 2. LSP IE Branch Original Knowledge Capture Map

The most prevalent items in our IE Branch original knowledge capture map are the use of our peer reviews and our processes and guides. Peer reviews are the most important technical meetings we have as a team. A peer review is required before we are able to complete and publish several of the major systems engineering products we are responsible for creating as LSP Integration Engineers.
Below is a list of the most common types of peer reviews we conduct:

- ICD
- IRD (Interface Requirement Document)
- Our LSP Independent Verification Matrix
- Reduced IRD for LSTO
- Program Risk Generation
- Engineering Review Board (ERB) Presentations
- Launch Vehicle Readiness Review (LVRR) Presentations

Before we introduced Confluence as a new way to capture knowledge, we were capturing peer review artifacts in a meeting minutes style Microsoft Word document. SharePoint was used to collect all of these Word documents into one main library and the notes within that document were mostly taken by our group's engineering assistant and sometimes the notes would be updated after the meeting by the engineer actually going through the peer review. The other commonly used knowledge capture elements were the IE processes and guides. These were also formal Word documents that were written primarily by just a few individuals within our group and most of the guide documents had not been updated for several years or more. Despite not having been updated in a while, these processes and guides were still very much the centerpiece of our workflows and were used by every engineer in the group as a way to generate consistent products.

Perhaps what is most telling about this mapping of our existing knowledge capture system is what is missing. Our mission feasibility support function did not have a direct link to any of our captured knowledge products. We were also not efficiently tying in our meeting minutes and training resources into our primary functions, they only existed as stand alone documents within SharePoint.

Step 4: Enhance Knowledge Capture Methods

As we discussed earlier, there are two very critical components that Confluence does well, contextual information and social networking. Capturing the "art" within a systems engineering organization requires that the "why" be captured contextually with the "what." That is where the contextual information and social networking features of Confluence become so crucial. While there are certainly other ways to enhance an organization's knowledge capture methods, Confluence was an obvious choice for the IE Branch because of its ability to integrate into the existing LSP program-wide collaboration tool SharePoint. The enhanced method of knowledge capture chosen should be minimally disruptive to current workflows and have a user interface that does not impede user input and engagement. If the enhanced knowledge capture method is non-intuitive or adds any type of a “barrier” between the user and the content, then it is less likely to be used. In our case, we could have simply enhanced how we were using SharePoint instead of shifting much of our knowledge capture over to Confluence, but SharePoint was missing some of the contextual and social features that we knew Confluence contained. We also had the advantage of having two other NASA Centers (GSFC and JPL) with established instanced of Confluence from which to leverage. There is no single solution for enhancing knowledge capture, but keep in mind that small details like ease of use, social networking and commenting can make or break an initiative like this. Shown below in Figure 3 is our IE Branch's updated knowledge capture map.

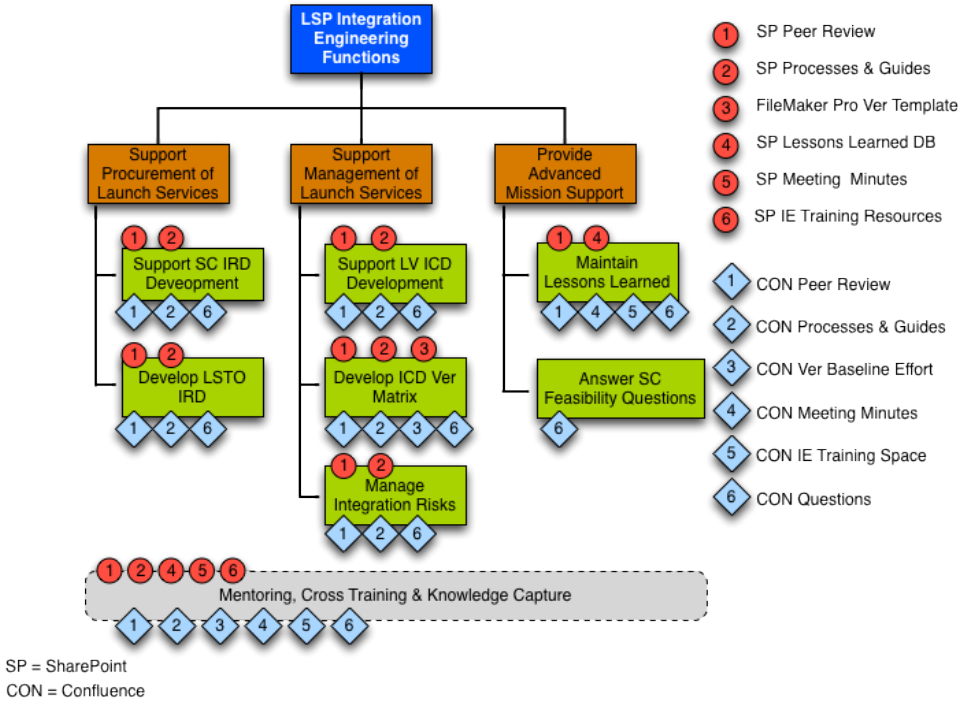

Figure 3. LSP IE Branch Enhanced Knowledge Capture Map

In most cases there was an element in Confluence that was a direct replacement for the old knowledge capture method, but some methods were replaced with multiple methods. For example, before enhancing our knowledge capture methods with Confluence; we did not have a good way of capturing information related to spacecraft feasibility questions. With the addition of Confluence we are now using Confluence Questions to capture that knowledge. Mentoring and cross training is another good example, because every new method we introduced with Confluence is now supporting our mentoring and training. What follows are summaries of how these enhanced knowledge capture techniques were implemented as part of our pilot and why they were found to be effective. Step 4 of the process we are presenting here is the most difficult and variable step in the process. It is not feasible to define a process for enhancing knowledge capture and expect that process to be effective in every type of systems engineering organization. So instead, we are presenting methods that have proven effective during our pilot in the hope that it provides a foundation from which to start. Use the following examples as a guide when starting to develop a strategy for this step in the process.

Confluence Questions. One of the very first features in Confluence we started using was the Confluence Questions add-on module. Confluence Questions is a way to crowd-source a question with answers from multiple people within the organization, similar to other popular online systems today like Yahoo Answers and Stack Overflow. We chose to start using this feature first because it was the most simple and least disruptive change we could make, but, at the same time, it was the most effective. Before using Confluence Questions, our method of polling the group concerning a particular technical question or situation was either email or walking around and talking with people one-on-one. While both of these methods were effective at getting answers, they were not effective at capturing the results for use by others in the future. Answers that would be sent back to the individual asking the question would then be captured only in that individual's email inbox and would not be accessible to the rest of the group. The

same thing would happen with individual discussions unless the results of those conversations were documented. The IE Branch did not have an effective method to capture the knowledge from these really useful conversations. Now, instead of relying on individual email inboxes to capture these kinds of group technical questions, we shifted those interactions over to Confluence Questions.

Enhancing our knowledge capture in this area with Confluence Questions did two things for our IE Branch very quickly, it gave everyone in the group a reason to start using Confluence and it immediately put a stop to the loss of organizational knowledge that was occurring. Now, when someone has a question we still email the entire group with that question but the email now links to the question within Confluence Questions instead of relying on email responses. Confluence Questions is a prime example of how the "art" of systems engineering can be captured. In systems engineering, there is almost always more than one to treat a problem or situation. Each of those individual approaches is a valid way to solve the problem and each answer is an example of the "art" that was applied by each person to solve that particular issue or situation. By crowdsourcing our group's technical questions, we are assured that the person seeking an answer will have multiple approaches to consider as part of the solution. Now that individual can evaluate all of the potential approaches and choose a method that best suits how they most naturally practice the art of systems engineering. Notice that Confluence Questions is now a knowledge capture mechanism for each and every one of our IE Branch functions, as shown in Figure 3.

Branch Meeting Notes. Like any other group in the work environment, we have weekly group (Branch) meetings. The purpose of these meetings is for our Branch Chief to share information from higher level LSP management meetings with the group and for the individuals within the Branch to share information about the spacecraft missions we are all assigned. Most of the Integration Engineers are assigned multiple missions (between two and four missions) and we are typically the only Integration Engineer working those missions (unless we are assigned a backup Integration Engineer for training purposes). Because of this “one Integration Engineer to a mission” assignment strategy, it is imperative that as a group we share important information regarding our mission work with others, so the experience we gain working that individual mission (the "art") can be applied by other Integration Engineers to their assigned missions. Before using Confluence, our Branch Chief would bring in a combination of handwritten and typed notes taken from his weekly management meetings and use these notes as an agenda for our weekly Branch meetings. The meeting would take place and our group’s engineering assistant would take meeting minutes (typically handwritten notes) and would later type these notes into a Word Document and upload the meeting minutes into our Branch’s SharePoint site. After our group started using Confluence, we transitioned not only the notes and agenda content going into the meeting, but also the meeting minutes themselves into a single page within Confluence. A template with placeholders for individual updates from each engineer, the Branch Chief's updates, notes from the meeting, and action items to be completed was developed. Each set of minutes was also labeled with relevant tags to associate these minutes with the missions and documents being discussed. The Branch meeting minutes template was created with the tag "meeting-notes," so the entire collection of meeting minutes could be displayed automatically using the "Content Report Table Macro" within Confluence. This macro will automatically display all content with user-specified labels on a page, similar to a table of contents can be automatically generated in Microsoft Word simply by using the correct "Style" of text for each section and figure within the document.

There are numerous macros similar to the content report table macro, which make it easy to display information to the user within the context of the portion of the Confluence site they are currently working.

It was important to start capturing our Branch meeting minutes in a more collaborative and accessible format because this meeting is our primary methods of formally sharing knowledge with our group in a face-to-face setting outside of our peer reviews. The rest of the time we are individually leading our engineering teams in the integration of one of our assigned missions. By shifting our meeting documentation to Confluence, we were able to time shift our meetings. Our Branch Chief makes his talking point available before the meeting, which lets the group skim the notes and get up to speed on what is going to be discussed. Our group is now more prepared coming into our meetings so our discussions are more focused and efficient. Some of our peer review meetings have even had success in taking meeting minutes live within Confluence, which ensures a more accurately and timely capture. After the meeting ends we can each go into Confluence and add points our engineering assistant may have missed or link to content on Confluence that pertains to points covered in the meeting. I have personally stopped taking notes at our group meetings since we started using Confluence because I know that we have a centralized record of knowledge that I can go in and add to after the meeting, just like having my own personal notes. Confluence has converted our weekly Branch meeting notes from a document that used to be stored in SharePoint into a central reference point of collaboration for our team.

Peer Reviews. Prior to using Confluence, peer reviews were done in multiple meetings lasting a few hours each until we had made our way through the complete review of a document or product. Comments were emailed to product owner after the meeting by our group's engineering assistant, and it was up to the product owner to decide what to do with those peer review minutes and actions. Within LSP we follow an engineering process that directs us to document important engineering decisions, including all of the artifacts such as meeting minutes, individual engineer comments and supporting documentation. The discussion that takes place in our peer review, the formal comments made and the disposition of those comments are all very important aspects of documenting the engineering decisions that lead to the completion of the product. Before we started using Confluence, the documentation that came out of our peer reviews was inconsistent. The product owner in the peer review would always summarize the results of the peer review, but the capture and documentation mechanisms were far from perfect. With Confluence, we have set up the structure of the peer review page within Confluence to be compatible with an export out to PDF, so that with a single press of a button we can export all the content documented in the Confluence peer review page out to a single PDF. Now, in addition to the high-level summary we used to write we also have a detailed PDF product that exports out of Confluence. This results in a much more professional result and is now consistent across our engineers and even across the types of peer reviews we conduct.

Peer Reviews in Confluence are labeled with several different tags, which help users to sort and search through these pages. Each document is tagged with a status (open or closed), as well as with a label specifying the type of peer review (ERB, ICD, IRD, LVRR Risk, Verification Matrix or Other). A separate page holds expanding sections that sort peer reviews by status, type, and date added to aid in discoverability. Templates were created for each type of peer review to ensure uniform labeling, but also so each peer review generally follows the same structure. The product being reviewed is attached at the top of the page, while comments, before and after the in-person review, are located

in tables further down the page. The tables are used to capture the names of the reviewers, comments made during the review, and the section of document each comment is referring to. This document is formatted to enable the author to export to pdf, allowing all the comments, discussion, and dispositions to be easily captured and stored externally (which is required by our engineering process).

Verification Baseline Collection of Wiki Pages. Before using Confluence, our primary method of knowledge capture with respect to our independent ICD verification matrix was the FileMaker Pro template. This template contained the standard verification plan language we would use to start the verification matrix for a new mission once we had established an ICD. However, we would often run into the same question when we finished the verification matrix and took it through our peer review…"Why are we doing the verification this way?"

The FileMaker Pro template was very good at capturing the verification plan wording, but there was not an easy way to efficiently and completely capture all the rationale behind the "why" (e.g. the number and type of verifications for a given requirement). With the addition of Confluence we were able to capture the "why" behind these verifications so when we encounter a mission unique situation we can adapt the verification plan to address the unique nature while still maintaining the original intent behind the verification. A collection of wiki pages within Confluence was created and a separate wiki page was then established for each discipline, mission type and launch vehicle. For each of these disciplines, mission type and launch vehicle wiki pages we started to conduct peer reviews and slowly began populating a rationale statement behind each of our verification plans. Once the group had agreed upon the number of verifications, the verification plan wording and the rationale statements, that particular set of verifications was considered "baselined." This verification baseline effort is still in its very early stages of development, but we feel this is an excellent example of how an existing knowledge capture method can be slightly modified to better capture the "art" in parallel with the science.

Process and Guides. Our enhancement to our existing processes and guides, which up until the point of the Confluence pilot were Microsoft Word documents, was quite simple. Processes and guides were transitioned from Microsoft Word into the collaborative wiki environment as separate wiki pages. Now, instead of being static documents, our processes and guides are living wiki sites that can take full advantage of social and contextual features of Confluence. Shifting this content over to Confluence also enabled it to be searchable with all the rest of our Confluence content. Now, when a process or a guide is mentioned during a staff meeting, peer review or as part of a Confluence Question, everything is both hyperlinked and connected (greatly enhancing the usefulness of search results). These IE Branch processes and guides are essential to the science of what we do as LSP IEs and now we are able to link them together with all of our other knowledge capture material, enabling us to link the "what" (processes and guides) with the "why" (the "art").

IE Training Resources. Out of all the areas where our group enhanced knowledge capture for the purpose of capturing more of the art, the area of mentoring and training has the most potential art to be captured. The purpose of the mentoring and training space (both the original collection of Microsoft Word Documents in SharePoint and the new collection of wiki pages in Confluence) is to establish a set of training resources and both formal and informal guidance for mentors and mentees. Our group felt it was important to try and establish a common set of ground rules for everyone involved in the training process and then provide additional informal guidance to further

strengthen the formal guidance. Training an individual person is not something that can be done with a "one size fits all" approach. Both the mentor and mentee are individuals, each with unique ways of learning and teaching. Therefore, each case of mentor/mentee requires customization. That customization is up to the mentor to decide, which is why we have included the informal guidance as part of the IE Training Resources. When common guidance is combined with multiple approaches from individual IEs, the mentor is given a variety of approaches from which to choose. This allows the mentoring IE to follow the common formal guidance, while at the same time customize the mentoring approach to best fit both the individual characteristics of the mentor and the mentee.

Step 5: Introduce Enhanced Knowledge Capture

This is a very important step in the process to approach in a way that is both effective and least disruptive to the group's current workflows. Change, even when this change is a large improvement compared to the baseline, can have unintentional negative consequences. When introducing the enhanced knowledge capture methods identified in Step 4, it is extremely important that these enhancements are carefully phased into the organization's workflows over time. Start with the least disruptive enhancement and slowly introduce additional enhancements. We chose to start with items that were very similar to how we were already conducting business, like branch meeting notes and peer reviews. Confluence Questions was also a good early adoption feature to roll out because it rapidly showed the group the value of using this new feature. Knowledge capture value is only realized later when the information is searched for and the knowledge can be easily accessed. This delayed value proposition makes it a hard sell up front where all of the knowledge capture is being performed. Branch meeting notes, peer reviews and Confluence Questions all gave the users who were adding content into the system an immediate benefit. With branch meeting notes our group was able to come into branch meetings better prepared and then, after the meeting, add to the notes instead of having to take their own notes. The peer review enhancements made our face-to-face peer review meetings more effective and reduced the time spent in the physical meetings. Confluence questions gave our group a centralized way to capture knowledge that we were not formally capturing in the past, which gave us some relief from having to rely on email as the capture mechanism. The other recommendation for introducing the enhancements from Step 4 is to start with tasks that must be performed by the group. If it is a required task then it ensures a very quick and wide adoption of the new techniques.

Step 6: Evolve the Knowledge Capture

The final step in the process is to continue to evolve the enhanced knowledge capture techniques. Successfully capturing the "art" of the organization is going to drive innovation. As the organization continues to improve, knowledge capture techniques must also improve. This entire 6-step process is iterative and must be re-assessed on a regular basis to ensure it remains effective. For example, if the organization takes on new functions or changes some of the functions it is responsible for, then those changes must be reflected in the knowledge capture methods. Go back to step 1 and update the organization's functional architecture and re-evaluate from that point forward. The continuous evolution of knowledge capture methods will take significantly less effort than the original enhancement and each evolution will give the organization another

opportunity to build upon the benefits of the knowledge that has been captured up to that point.

Conclusions

Systems engineering is a challenging field due to its multi-disciplinary nature. It is impossible to capture all the bits of technical knowledge needed to carry out the job of a systems engineer, much less all of the "art" and rationale behind all of the technical knowledge. The process for capturing the "art" of systems engineering presented here is a starting point for any systems engineering organization that wants to enhance their existing knowledge capture techniques in order to grow beyond just the science and to start capturing the "art" of systems engineering as well. Knowledge capture is just the beginning. The next step is to ensure that it is available for everyone in the organization to find and use as they perform their jobs on a daily basis. It is in this area of discoverability and reapplication of knowledge that we are finding the contextual and social networking features of Confluence to be most valuable. What we have presented in this paper is a high-level six-step process, using the example of our LSP IE Branch Confluence pilot program to aid in the instruction of how to carry out this process. Just like the "art" of systems engineering, the execution of this process is more "art-like" than it is science. Each organization will need to tailor this process to the unique culture and functions of their organization, as well as the tool or tools they choose to use, to enhance their knowledge capture.

References

Bay, Michael, Bill Gerstenmaier, Mike Griffin, Jack Knight, Wiley Larson, Ken Ledbetter, Gentry Lee, Michael Menzel, Brian Muirhead, John Muratore, Bob Ryan, Mike Ryschkewitsch, Dawn Schaible, Chris Scolese, and Chris Williams. "The Art and Science of Systems Engineering." NASA.gov. January 18, 2009. Accessed October 3, 2015. http://www.nasa.gov/pdf/311199main_Art_and_Sci_of_SE_SHORT_1_20_09.pdf.

NASA Software Engineering Handbook. NASA.gov. Web. 16 Mar. 2016. <http://swehb.nasa.gov/display/7150/Book A. Introduction>.

Rober, Mark. "Wired Overview." Vimeo. 20 Dec. 2009. Web. 16 Mar. 2016. <https://vimeo.com/8303614>.

Biography

“Skip” Clark V. Owens III is a systems engineer in the Integration Engineering Branch of the Launch Services Program at NASA Kennedy Space Center. Mr. Owens graduated from Wichita State University with a B.S. in Aerospace Engineering and has worked as both a spacecraft and launch vehicle trajectory/mission design engineer. Mr. Owens is currently pursuing his M.S. in Space Systems Engineering from the Stevens Institute of Technology.

Carrie Sekeres is currently pursuing her B.S. in Aerospace Engineering, with a concentration in Astronautics, at Embry-Riddle Aeronautical University, where she also works analyzing engineering education practices for the Engineering Fundamentals Department. Ms. Sekeres interned in the Integration Engineering branch of the Launch Services Program Directorate working to develop and implement a working online collaboration space for several of the branches at Kennedy Space Center. She plans to pursue her M.S. in Systems Engineering after her graduation from ERAU.

Yasmeen Roumie is a senior at Stuyvesant High School in New York City. She was an intern for the Integration Engineering branch of the Launch Services Program at the Kennedy Space Center during the summer of 2015. For the past decade, she has been engineering and programming robots and has traveled around the world to attend robotics competitions. On the weekends, Ms. Roumie likes to build websites at hackathons and teach young girls how to build and program robots as well. Ms. Roumie plans on going to college soon to study engineering and/or computer science.