Presented at the 5th Polytechnic Summit
Wentworth Institute of Technology, Boston, Massachusetts, June 5-7, 2013

Improving the Systems Engineering Requirements Analysis Process: A Few Tools and Techniques

Karl Snyder[a], Adeel Khalid[b,*]

[a]*Graduate Student, Southern Polytechnic State University, 1100 South Marietta Parkway, SE, Marietta, GA. 30060, USA*
[b]*Assistant Professor, Southern Polytechnic State University, 1100 South Marietta Parkway, SE, Marietta, GA. 30060, USA*

Abstract

The technological complexities of today's engineering breakthroughs require a robust engineering process to promote successful product, systems, or software development. Systems engineering is that process. Employed across a diverse set of engineering fields, and inextricably linked to the tenets of project management, the rigor in which systems engineering is applied may often make the difference between project success and failure. In this paper, we review the systems engineering tools and techniques widely known throughout the community of interest but that are not necessarily known or applicable to working with customers to generate requirements or to provide feedback to the customer to vet the stated and derived requirements. The tools and techniques presented in this paper are just a subset of those available to systems engineers. Many more exist that may be customized to assist systems engineers in requirements analysis. Systems engineers should not have to conduct extensive research to locate and evaluate such tools and techniques. Instead, it would behoove the systems engineering community to develop a tools and techniques bible; that is, a source for systems engineers to review tools and techniques applicable to particular processes, where they can then determine, based on a balance of time and cost to the project, those best suited to advance and support the requirements process.

Keywords: Requirements Engineering, System Development, Derived Requirements, Customer Feedback

1. Main text

Systems Engineering, is defined by the International Council on Systems Engineering (INCOSE) as "an interdisciplinary approach and means to enable the realization of successful systems, focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation, while considering the complete problem.[1]" Employed across a diverse set of engineering fields, and inextricably linked to the tenets of project management, the rigor in which systems engineering is applied may often make the difference between project success and failure. For a given system, success is the completion of a project within the constraints of cost, schedule and performance, as well as delivering the desired service (i.e., product, system, system of systems, etc.) to the customer.
Systems engineering processes include, but are not limited to, requirements analysis, validation, functional and design verification, synthesis, and trade and assessment studies [2]. In this study, we focus on the inputs to requirements analysis, and more specifically, the tools and techniques systems engineers can utilize to foster communications with customers and better understand their requirements.

2. Motivation

Too many large-scale projects have failed over the past 20 to 30 years, in part, due to a failure to define, scrutinize, verify, and/or control requirements. The 2007 Standish Group CHAOS report documented that 35% of software projects started in 2006 were successful, 46% were challenged (having cost or time overruns or not fully meeting user requirements), and 19% outright failed [3]. Failed projects occur in the business sector as well as government. From 1981 to 1994, the Federal Aviation Administration (FAA) spent between $1.5 and $2.6 billion on the Advanced Automation System (AAS), which was never used and coined as "useless." The Government Accounting Office (GAO) studied AAS and stated multiple reasons for the failure, including "the FAA did not effectively control system requirements [4]." Another high profile project that failed was the Federal Bureau of Investigations (FBI) Virtual Case File (VCF). Designed to automate the FBI's paper-based work environment, VCF was a $170 million failure. In 2005, a U.S. Department of Justice Inspector General report described eight factors that contributed to the failure of VCF. Among them, "poorly defined and slowly evolving design requirements [5]."

In the recent years, systems engineers have improved the requirements analysis process. A 2010 report by the National Defense Industrial Association (NDIA) titled Top Systems Engineering Issues in US Defense Industry noted that there had been improvements in requirements analysis since 2006. This report compared engineering issues identified in 2006 to improvements made by 2010; two of five major issues noted in 2006 were requirements related [5]:

- Insufficient systems engineering is applied early in the program life cycle, compromising the foundation for initial requirements and architecture development.
- Requirements are not always well- managed, including effective translation from capability statements into executable requirements to achieve successful acquisition programs.

While the NDIA has noted progress, projects still reach failure where perhaps more clearly defined systems engineering processes, especially the inputs to requirements analysis, could lead to more systems engineers being successful. No single methodology in engineering has formally proven more effective in leading to project success than any other. However, the systems engineering life cycle process appears dominant in today's engineering culture. There are many high-level steps in systems engineering and at the root of this holistic process is requirements analysis. Requirements analysis establishes system capabilities, performance measures, system interfaces, physical attributes, and constraints [7]. At the core of requirements analysis is its input: defining stakeholder expectations; all other requirements analysis sub-activities must link back to the needs of the stakeholders, making defining stakeholder expectations a foundation for systems engineering.

As the primary input to and fundamental step in requirements analysis, there exists little formal documentation from the standards community (e.g., IEEE, ISO, and DoD) to guide the systems engineering workforce in communicating with and collecting requirements from stakeholders. Often times, the poor execution of the 'define stakeholder expectations' step is a root cause of project failures; but no formal source supports this idea. Poor requirements execution, in general, often contributes to project failures, especially when projects are extremely complex. The lack of formal guidance for systems engineers in using tools and techniques to better communicate with and understand stakeholders' requirements has motivated the authors to highlight this shortcoming.

3. Objective

In this paper, we address the lack of formal guidance offered to systems engineers in the use of tools and techniques to promote optimal requirements analysis. Specifically, tools and techniques available to address customer interview and customer feedback processes within requirements analysis are discussed. The application of robust tools and techniques to requirements analysis in a customer interactive and understandable format will result in higher success rates in engineering projects.

The objectives of this paper are twofold: (1) identify where improvements in requirements analyses are necessary; and (2) provide analyses and summarize tools and techniques best suited for requirements analysis. The qualitative and quantitative comparative analysis of tools and techniques will be suitable for systems engineers to determine which tools and techniques to implement within their systems engineering management plan (SEMP). The authors strive to make the case for the systems engineering community that current international and government standards lack the depth of guidance necessary to make noticeable improvements in systems engineering processes.

4. Literature Review

Requirements analysis is mandatory in the conduct of any engineering task. Even without following a formal process or knowing that requirements analysis is occurring, people determine, often on their own, the components necessary to complete their task. For example, to clean a floor a janitor knows that he requires a mop, bucket, water (usually hot) and some type of detergent. These are just the basic requirements of the janitor's task, although janitors probably do not call these elements "requirements." Given this general line of reasoning, there is significant literature available that provide insights into requirements analysis, a function every engineer must competently understand to be successful.

International standards and government guidance top the list of documents assumed by the author to be at the heart of systems engineering acumen. Three Institute of Electrical and Electronics Engineers (IEEE) standards are most relevant to systems engineering requirements analysis. IEEE Std 15288-2008 addresses systems and software engineering life cycle processes [8]. Germane to this research, section 6.4.1 develops the stakeholder requirements definition. Defined by three sub-activities (elicit, define, analyze and maintain stakeholder requirements), the stated process is vague. IEEE does this purposefully. IEEE qualifies it's vagueness by stating the "standard does not prescribe a specific system life cycle model" and "does not detail life cycle processes in terms of methodology," leaving it to the engineer to determine how such processes should be performed [9]. IEEE Std 1220-2005 is the IEEE Standard for Application and Management of the Systems Engineering Process. This standard is similar in context to IEEE 15288, but specifically states the meaning for defining stakeholder expectations. However, the remainder of the text is just as vague as 15288. IEEE 1220 qualifies its vagueness through the statement "…terminology of the standard are defined at an appropriately high level of abstraction (iii)." Lastly, the standard in which the author expects to document the most detail in methodology is IEEE Std 1233-1998, IEEE Guide for Developing System Requirements Specifications. This standard identifies customer documents that provide the systems engineer a sense of raw requirements (i.e., not formatted). An example of these documents includes concept of operations, system concept, and request for proposal. This standard provides the largest singular example within this literature review in describing requirements collection methods.

The U.S. government and associated defense groups have a formal set of documents in which Program Managers (PM) are required to adhere. Within these documents exists requirements analysis descriptions, some of which appear derived from the IEEE standards, although not explicably stated. Within the Department of Defense (DoD), the Defense Acquisition Guidebook (DAG) is the proverbial PM bible [10]. Chapter 4 of the DAG specifically addresses systems engineering principles, and hence, the stakeholder requirements definition process. It highlights the Joint Capabilities Integration Development System (JCIDS) as the input process to stakeholder requirements definition. There exists significant gaps in guiding the PM on how to interact with customers to define requirements.

MIL-HDBK-520 is the DoD's Systems Requirements Document Guidance (SRDG). This guide states where input for requirements should come from (e.g., warfighter needs, objectives, etc.), but does not state by what methods to get them. At best, SRDG acknowledges, "requirements analysis is a core SE process that begins with definition of the warfighters Capabilities Based Requirements [11]." This guidance is also based on high level and abstract processes.

The National Aeronautics and Space Administration's (NASA) Systems Engineering Processes and Requirements manual offers similar breadth of stakeholder definition information as other documents, stating that the process inputs include customer expectations, other stakeholder expectations, and customer flow-down requirements [12].

The NDIA has conducted extensive research on the effectiveness of systems engineering. Although not a government agency, NDIA is a legal forum for the exchange of information between industry and the government (NDIA website). In NDIA's Report on Systemic Root Cause Analysis of Program Failures, a study of 44 Acquisition Category (ACAT) 1D programs (defined as RDT&E ≥ \$365M or Procurement ≥ \$2.19B) discovered that 11 of 48 systemic issues (23%) were related to requirements (e.g., requirements were not identified, not understood, were vague, or were unreasonable) [13]. This is the first of the literature stated here not as a reference guide, but as a document stating the gaps in requirements analysis processes. In another NDIA document, Top Issues 2011 [14, 15], NDIA noted various improvements were necessary in requirements analysis. Specifically, NDIA stated the need to "ensure that requirements define the true needs, recognizing that nice to haves raise cost," and that "EVMS cannot fix a contract with ill-defined requirements and cannot prevent cost growth unless requirements are clearly understood and requirements creep is controlled."

The above literature covers a wide variety of requirements analysis from case studies to the use of tools and techniques used to analyze large-scale projects. Case studies reviewed as part of requirements analysis were difficult to find where software was not the primary topic of interest. While software is certainly a topic with systems engineering applications, the authors are interested in case studies and documents reaching across engineering fields, where the outcome of a study could be applied with breadth of application, not software alone. During this research, software is a predominant topic for requirements analysis and engineering, highlighting that more work needs to be done for general engineering approaches.

The case study of Knowledge-based Approach for Selection of Requirements Engineering Techniques

(KASERT) outline a list of requirements elicitation techniques that apply to varied engineering disciplines [16]. Examples of techniques include focus groups, laddering, brainstorming, and document mining. Parviainen and Tihinen [17] state that despite a significant amount of papers based on requirements engineering methods, "we could not find any published research giving the big picture of available methods…describing how the available methods support and cover the requirements engineering activities as a whole [17]" The authors find this statement to be a direct parallel of current research findings in requirements elicitation and collection methodologies. Furthermore, Parviainen and Tihinen [17] state that their aim is "to gain an overall view of the available requirements technologies." In this study, we hope to contribute to systems engineering by bringing to light various tools and techniques systems engineers can use to collect more accurate and useful requirements definitions from their customers.

5. Methodology

"Problem definition is the most difficult activity of the systems engineering process [18]." The literature review section clearly identified that international standards, DoD guidance, and technical papers on requirements analysis lack the rigor for developing customer interaction and requirements collection methodologies. Sheard and Lake compare systems engineering standards and models [19] and note the difficultly practitioners have in keeping current with the extended amount of standards available, determining how they differ, and which versions of each is applicable. Today, systems engineers must sift through a plethora of engineering publications and technical papers to develop methodologies that may help communicate to their customers (i.e., stakeholders) the potential requirements for their project.

The methodology for this paper moves past knowledge gained from literature reviews and into a discussion of tools and techniques appropriate for working with customers of varied technical competencies to collect, define, and review requirements. Even though much of the literature reviewed shows gaps in specific methodologies currently used in requirements collection, some relevant material is found that might be applied to general engineering practices. This material is reviewed more closely in the tool and technique analysis section. The primary sources of data for this research include textbooks, technical papers, and websites designed to promote the use of particular systems engineering tools and techniques.

In this paper we review systems engineering tools and techniques widely known throughout the community of interest but that are not necessarily known or applicable to working with customers to generate requirements or to provide feedback to the customer to vet the stated and derived requirements. These processes should occur before moving into more detailed requirements analysis processes.

The next section brings to light, for the systems engineering community, proven tools and techniques useful in working with customers to collect requirements. Specifically, we address the application of methods for interaction with customers in understandable ways.

For analysis purposes, tools and techniques are broken into two categories: customer interview and customer feedback. Customer interview techniques analyzed in this paper include the Quality Attribute Workshop, Use Case Diagramming, User Stories, Architecture Tradeoff Analysis Method (ATAM), Delphi Brainstorming, and Objective Hierarchies. Customer feedback tools and techniques discussed are System Requirements Specifications (SyRS), SWOT-TOPSIS, Hybrid TOPSIS-FAHP, AbsrRM, Cost-Value Prioritization, and Cost-Driven Design Optimization.

6. Tool and Technique Analysis

6.1 Customer Interview

The customer interview process to develop and understand initial requirements is the first active step a systems engineering team undertakes. As such, the success or failure of the interview weighs heavily on the outcome of the project.

6.1.1. Quality Attribute Workshop (QAW)

There are multiple methods available to work with the customer on initial requirements; one method is the Quality Attribute Workshop (QAW). Developed by Carnegie Mellon's Software Engineering Institute (SEI) as a tool to address software-intensive systems, it has applications in systems engineering requirements development [20]. The purpose of the workshop is to develop an initial set of quality attributes used to describe high-level system roles and purpose, thus QAW is primarily a qualitative tool. For instance, QAW would help to determine the "ilities" a system or product requires, such as scalability, availability, reliability, and usability. For each quality attribute (ilities), the workshop participants (i.e, the customer) develop scenarios to refine each of the priority attributes. For the systems engineer, each attribute would necessitate a requirement or multiple requirements to support it. While QAW uses a tablet methodology to develop scenarios for attributes, a use case is pictorial and serves to augment the tablet format. In collaboration with the QAW, use cases can significantly enhance the development of customer stated scenarios.

6.1.2. Use Case Diagrams

Use case diagrams represent the high-level interaction of four elements: actor, system, services, and relationships. Based on the attributes developed through the QAW, use cases show the functionality of each attribute by depicting who would use it, what it would provide, and the relationships involved. Use cases provide a communications stream between engineer and customer to ensure the customers idea is accurately understood by the engineering team.

6.1.3. Use Stories

In lieu of use case diagrams, a systems engineer might choose to develop user stories. User stories are an agile software tool used to develop the customer's story about required functionality. It is not an end-all be-all story. In fact, it is only a start and not meant to be complete. User stories are developed for a particular role, in this case a system function, and provides the who, how and why of the function. This is done at a high-level and is not technical in nature. One example of a user story is, "As a bank business owner, I want to set the ATM's withdrawal parameters so the ATM will provide funds to customers but protect against fraudulent activities." Nazzaro et. al. assert that systems engineers will take these functionality describing stories and develop requirement sets to achieve the goals [21].

6.1.4. Architecture Tradeoff Analysis Method (ATAM)

Another technique utilized in similar efforts as QAW and use cases is the Architecture Tradeoff Analysis Method (ATAM). ATAM follows the QAW, as the QAW is an integral input. Originally designed for software development, ATAM is a method that "assesses the consequences of architectural decisions in light of quality attribute requirements," such as those developed in QAW [22]. The same workgroup that discussed QAW should participate in the ATAM. ATAM is easily applied with a focus on the requirements of any type of system, not only software. Kazman [22] notes that ATAM is an inexpensive exercise and produces results commensurate with the level of detail of the architectural specification. As applied to requirements analysis, the ATAM effort is customizable to the amount of time available to work on requirements development. ATAM includes processes to develop requirements trees (based on functionality), utilize scenario brainstorming, and perform tradeoff analyses. Not only is ATAM capable of assisting the systems engineer in developing initial requirements with the customer, it would also be useful as a tool during the feedback of derived requirements.

6.1.5. Delphi Technique

There are other methods not directly included in the QAW -> Use Case -> ATAM stream that are less advanced but just as valuable to the systems engineer when interviewing customers. The easiest is the brainstorming technique called Delphi. Commonly referred to as the Delphi Method, it is a method to achieve a consensus on a topic of interest among a group of experts through a series of anonymous questionnaires. Not often used in requirements collection, a well-informed systems engineer might look to gain consensus on requirements for individual system or product functions using Delphi. Utilized at length, Delphi would enable a wide audience of stakeholders to participate in developing requirements without retribution (due to anonymity). The downside is that this method would take an extended period to complete, as each distinct topic (product functional area) would require several iterations of Delphi.

6.1.6. Objective Hierarchy

Another simple tool to develop requirements with a customer is the objective hierarchy. An objective hierarchy is an arrangement of objectives that a customer expects a new system or product to achieve. Similar to a Work Breakdown Structure (WBS), an objective hierarchy used to develop requirements would list top-level objectives at the top of a page, while supporting requirements for each objective would be branched off below. For each objective, the engineer and customer would answer, “How could you achieve this?” by stating a requirement. Similarly, each primary requirement would be broken down with sub-requirements until a root requirement was established. The result of a well-developed objectives hierarchy is a requirements tree traceable to the customers’ high-level objectives.

6.2 Customer Feedback

The customer interview and requirements collection process is the first event in a series of interactions with the customer but clearly not the last. The entire systems engineering process is iterative, never being satisfied without feedback loops to refine and validate information gained. Requirements analysis is laden with feedback loops and there are tools available to assist the systems engineer in promoting effective dialogue with the customer to determine which initial requirements are necessary, affordable, and achievable within customer-defined constraints. Similar to customer interview techniques, international and DoD standards in systems engineering only promote the process, not the methods. This section analyzes tools to document initial requirements developed during the customer interview process and to provide the means to gain active customer participation in refining the baseline requirements.

6.2.1 System Requirements Specification (SyRS)

The System Requirements Specification (SyRS) is perhaps the best-documented process for reaching back to the customer with engineered and derived requirements. Standardized by IEEE Std 1233 [2], SyRS defines the documentation of requirements based on customer needs, operational concepts, and systems analysis. SyRS is used to communicate requirements to both the customer and the technical community. As such, there are different derivations of SyRS based on the intended audience. High-level requirements support customer objectives for customer reviews but a more technical and qualitative deep-dive version is developed for the engineers that will actually use SyRS to build a system.

6.2.2 7.2.2. Strength, Weaknesses, Opportunities, and Threats - Technique for Order Preference by Similarity to Ideal Solution (SWOT-TOPSIS)

TOPSIS (Technique for Order Preference by Similarity to Ideal Solution) is a technique widely used for strategic decision-making; however, we have not necessarily seen it applied to requirements development. Its strength in quantitative analysis has application in determining which requirements better fit customer scenarios.

For instance, one technique integrates the (SWOT) model with TOPSIS, and is aptly dubbed the SWOT-TOPSIS Integration Method. In a requirements scenario, SWOT is used during the customer interview process to develop system scenarios based on internal and external (environmental) factors to promote focus on the real problem to be solved. Using the technique described by Yang Ying [23], a requirements SWOT chart is developed. The customer and systems engineer fill in the SWOT quadrants by strategizing the problem to be resolved. Next, using the SWOT, options (alternatives) are developed as a solution to the customer problem. Following this, each element in the SWOT is evaluated on a 5-point scale, based on their ability to contribute to a solution for each option. Thus, an m x n array is completed. Finally, the proximity of optimal value of each alternative is computed. The result is a sorted index of options from 1 through n. The SWOT-TOPSIS tool is therefore used to provide a quantitative method to evaluate alternative solutions to the customer problem from which further requirements solicitation can occur.

6.2.3 Fuzzy Analytic Hierarchy Process (FAHP)

Analytic Hierarchy Process (AHP) is a decision-making tool that has applications in requirements selection and analysis. Traditionally used in pairwise analysis to determine best choices, provide ranking, and as far-reaching as conflict resolution, engineers can apply AHP using customer input to prioritize capabilities of a proposed system as compared to alternative systems [24]. Capabilities are factors that drive requirements. Taking AHP one-step further, Fuzzy AHP (FAHP) is a tool that incorporates not only the expert knowledge required to develop AHP, but also a component of fuzzy set theory to account for imprecision and vagueness [9]. FAHP uses a linguistic scale to assign triangular fuzzy numbers (TFN) in the pairwise scheme normally associated with AHP. So for instance, requirements may be structured into a pairwise table and the customer(s) assign a TFN to each requirement pair. Next, engineers calculate weights for all requirements through synthesis, comparison, and calculation of priority weights. After normalization of requirements, the resultant scores are calculated as a distance from the positive ideal solution (PIS) and negative ideal solution (NIS), a closeness coefficient is determined, and all requirements are ranked. As may be noted, TOPSIS is the quantitative engine that drives FAHP.

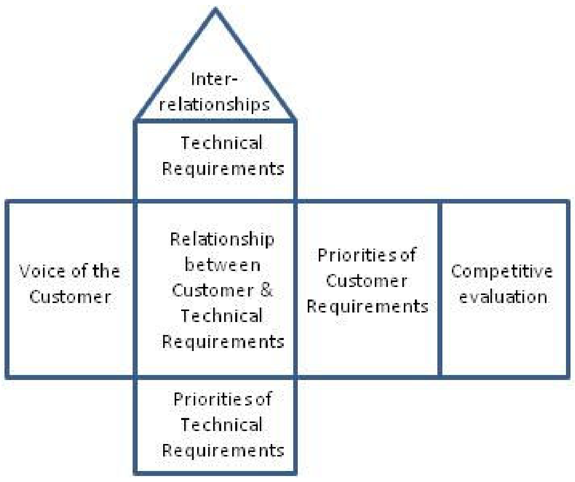
Inter-relationships
Technical Requirements
Voice of the Customer
Relationship between Customer & Technical Requirements
Priorities of Customer Requirements
Competitive evaluation
Priorities of Technical Requirements

Fig 1: House of Quality

6.2.4 *Quality Function Deployment (QFD)*

A methodology, which is robust enough to utilize the input gained from customer interview processes, and then assist in subsequent iterative customer interactions is the Quality Function Deployment (QFD) process. A complex tool that captures the voice of the customer and engineered requirements (i.e., derived and technical requirements developed by the engineering team), QFD promotes customer interaction through the prioritization of requirements lines. QFD pictorially represents requirements in what is known as a House of Quality. Essentially, the House of Quality represents a comparative picture of the elements of (1) voice of the customer, (2) technical requirements, (3) relationships between customer and technical requirements, (4) priorities of technical requirements, (5) priorities of customer requirements, and (6) a competitive evaluation. Both the customer and engineers work in partnership to establish relationships between requirements, such as determining very strong, strong, or weak relationships. The QFD process is beneficial since both the customer and engineering teams participate in determining the baseline requirements. Elements and relationship of a House of Quality are depicted in Fig 1. [25]

6.2.5 *Abstraction-Based Requirements Management (AbstRM)*

All previous methods for determining, documenting, and working with the customer have required significant manual efforts. One method, Abstraction-Based Requirements Management (AbstRM), is an automated process that uses signal-processing algorithms to detect customer abstractions of requirements from such diverse documents as Requests for Proposals and Concepts of Operations, as well as helping to analyze the impact of changes to requirements baselines. The primary benefits realized through AbstRM are (1) tighter integration between requirements elicitation and requirements management and (2) requirements impact analysis [26].

6.2.6 *Cost-Value Approach for Prioritizing Requirements*

Applying a cost value to a requirement early on in the systems engineering process is difficult, at best, especially if there is no comparative system for parametric estimates. Karlson and Ryan [27] discuss an approach to prioritizing requirements based on cost in A Cost-Value Approach for Prioritizing Requirements. Their five-step process is (1) review candidate requirements; (2) work with the customer to develop and apply AHP; (3) use AHP to estimate the relative cost of implementing each requirement; (4) plot the values on a cost-value diagram; and (5) provide customers the cost-value diagram to determine which requirements to implement and which not to implement. Step 5 is always tough for customers as they typically aim to maximize system capabilities. A cost-value diagram helps to put the cost of “needs” over those of “wants” in a customer understandable format.

6.2.7 *Cost Driven Design Optimization*

A similar method as cost-value is cost-driven design optimization. Using cost as a determining factor, engineers develop cost algorithms to determine which, in a set of related and comparable requirements, meet the customer’s objectives at minimal cost. During this process, engineers (1) determine optimal values for design variables, and (2) select the best alternative. The three main cost factors considered are fixed costs, direct costs, and indirect costs. An example cost model is Cost = aX + b/X + k, where a = direct cost; b = indirect cost; k = fixed cost; and X = the decision variable, i.e., transmitter power for a radio [28].

7. Comparative Analysis of Tools and Techniques

Table 1: Tools and Techniques Comparison (Qualitative)

	Customer Interview	Customer Feedback	Customer Input	Engineer Input	Qualitative	Quantitativ e	Complexit y	Cost Componen	Manual	Software
QAW	*		*	*	*		L		*	
Use Case	*		*	*	*		L		*	*
User Story	*		*	*	*		L		*	*
ATAM	*		*	*	*	*	M		*	
Delphi	*		*		*		L		*	
Objective Hierarchy	*		*	*	*		L		*	*
SyRS		*	*	*	*	*	H		*	*
SWOT-TOPSIS	*	*	*	*	*	*	H	*	*	*
FAHP		*	*	*	*	*	H	*	*	*
QFD		*	*	*	*		M		*	*
AbstRM		*		*	*		H		*	*
Cost-Value		*		*		*	H	*		*
Cost Driven Design Optimization		*		*		*	H	*		*

7.1. Qualitative Analysis

There is no single application for the tools and techniques analyzed in this paper. Some tools are discussed as they apply to architecture, strategic decision-making, or defining system purpose. However, each tool or technique discussed in this paper also has clear application within requirements analysis. The knowledgeable systems engineer should be familiar with a complete set of tools and techniques such that they may apply the best approach to requirements analysis, or any other phase of systems engineering.

Table 1 lists the tools and techniques reviewed, analyzed for good fit to requirements analysis, and discussed in this paper. Each tool or technique is analyzed for its fit to a series of attributes to assist the systems engineer in selecting the most appropriate one based on time, cost, and complexity. Customer interview and feedback refer to whether a technique is better utilized to collect requirements or to present a baseline set of requirements back to the customer for review. The customer and engineer input simply reflect which participant can directly contribute to the development of the information within that tool or technique. Qualitative or quantitative attributes are an indication of the type of information necessary to determine output. Some tools receive qualitative input, derive a quantitative output, and are the reason some tools reflect the capability for both attributes. The scale of complexity of each tool or technique is a subjective grade. Low (L) indicates the tool or technique requires little experience, is easy to use, and self-explanatory; Medium (M) indicates that the user should have some experience using like tools or techniques; while high (H) indicates the user most likely requires some form of training and extended experience in the application of the tool. Tools that are capable of incorporating a cost factor into its analysis are considered to have a cost component. Table 1 provides an indication of whether a tool or technique has either a manual or software approach, or both. The author recommends the use of software, where possible, to improve the accuracy and timeliness of the analysis being undertaken.

7.2. Quantitative Analysis

A quantitative analysis of the tools and techniques described in section 7.1 is also performed. Using TOPSIS and AHP, and following commonly accepted processes, a rank order analysis of the tools results QAW to be on the top. The remaining tools rank in the following order: Use Case, User Story, ATAM, Delphi, Objective Hierarcy, SyRS, SWOT-TOPSIS, FAHP, QFD, AbstRM, Cost-Value, and Cost Driven Design Optimization. The processes for completing TOPSIS are not shown, as it is not germane to this paper.

The outcome of quantitative analysis is based on the assignment of scores made only by the authors. As such, the rank order should not be considered authoritative. However, it does provide for some conclusions based on the author's research. First, QAW and ATAM, which received the highest two scores, are specifically used in partnership. That is, QAW is an input to ATAM. Second, the results generally show that those tools and techniques with a mixture of attributes score higher than those that lean more toward either the customer or engineer.

An exhaustive list of tools and techniques available to assist the systems engineer in improving the requirements analysis process is not analyzed in study. Instead, this analysis highlights a subset of tools and techniques available to promote the successful customer interview process to collect requirements and to provide feedback to customers after systems engineers have developed and derived technical requirements. The systems engineers' job is to enable the customer to be able to understand the baseline requirements to prevent excessive requirements changes farther into the systems development process. Each project in which a systems engineer embarks is unique, and therefore requires analysis to determine which tool is most applicable to solve the problem, given the constrains of time, money, and complexity.

8. Conclusion

Projects of all complexities are founded on requirements. Requirements drive the development of products, software, and systems. However, international standards and DoD guidelines, which provide some of the most prolific systems engineering documentation available, only describe high-level systems engineering requirements processes. These documents do not attempt to describe "how" a process should, or could, be implemented. The problem with this approach is that systems engineers are left to interpret how to carry out each requirements process.

Systems engineers, out of necessity, require well documented and easy to find tools and techniques to interact with customers in the collection of initial requirements, as well as to provide feedback to customers, throughout the iterative requirements analysis process, in a customer responsive and understandable way.

The tools and techniques presented in this paper are just a subset of those available to systems engineers. Many more exist that may be customized to assist systems engineers in requirements analysis. It behooves the systems engineering community to develop a tools and techniques bible; that is, a source for systems engineers to review tools and techniques applicable to particular processes, where they can then determine, based on a balance of time and cost to the project, those best suited to advance and support the requirements process.

This study only touches the surface of describing systems engineering tools and techniques available to promote better requirements analysis. Further research should continue to catalogue tools and techniques with comparative analysis as to their particular application.

References

1. Systems Engineering Body of Knowledge (SEBoK), <http://www.sebokwiki.org>, Last accessed on Sep 23rd, 2012
2. IEEE Computer Society. IEEE Guide for Developing System Requirements Specifications. IEEE Std 1233-1998. New York. The Institute of Electrical and Electronics Engineers, Inc. 2005
3. Cerpa, Narciso and June Verner. "Why Did Your Project Fail?," Communications of the ACM 52 (2009): 130-134.
4. Cone, Edward. "The Ugly History of Tool Development at the FAA – Out of Control" : < http://www.baselinemag.com/c/a/ Projects-Processes/The-Ugly-History-of-Tool-Development-at-the-FAA/>, Last accessed on Sep 23rd, 2012
5. Goldstein, Harry. "Who Killed the Virtual Case File?," IEEE Spectrum, Inside Technology. September 2011 <http://spectrum.ieee.org/ computing/ software/who-killed-the-virtual-case-File>, Last accessed on Sep 23rd, 2012
6. National Defense Institute Association, <www.ndia.org>, Last accessed on Sep 23rd, 2012
7. IEEE Computer Society, IEEE Standard for Application and Management of the Systems Engineering Process. IEEE Std 1220-2005. New York. The Institute of Electrical and Electronics Engineers, Inc. 2005.

8. IEEE Computer Society. Systems and Software Engineering – System Life Cycle Processes. IEEE Std 15288-2008. Piscataway. The Institute of Electrical and Electronics Engineers, Inc. 2008
9. Balli, Serkan and Serdar Korukoglu. "Operating System Selection Using Fuzzy AHP and TOPSIS Methods." : Mathematical and Computational Applications 14 (2009): 119-130.
10. Defense Acquisition Guidebook, <dag.dau.mil>, Last accessed on Sep 23rd, 2012
11. Department of Defense. "Department of Defense Handbook – Systems Requirements Document Guidance." MIL-HDBK-520 (USAF). March 5, 2010
12. National Aeronautics and Space Administration. "NASA Systems Engineering Process and Requirements." NASA Procedural Requirements. NPR 7123.1A. March 26 2007.
13. National Defense Institute Association. Report on Systemic Root Cause Analysis of Program Failures. December 2008
14. National Defense Institute Association. Top Issues 2011.
15. National Defense Institute Association. Top Systems Engineering Issues in US Defense Industry. September 2010.
16. Jiang, Li, Armin Eberlein and Behrouz Far. "A Case Study Validation of a Knowledge-based Approach for the Selection of Requirements Engineering Techniques." : Requirements Engineering 13 (2008): 117-146.
17. Parviainen, Paivi and Maarit Tihinen. "A Survey of Existing Requirements Engineering Technologies and Their Coverage." : International Journal of Software Engineering and Knowledge Engineering 17 No. 6 (2007): 827-850.
18. Agouridas, Vassilis, Alison McKay, Henri Winand, and Alan de Pennington. "Advanced Product Planning: A Comprehensive Process for Systemic Definition of New Product Requirements." : Requirements Engineering 13 (2008): 19-48.
19. Sheard, Sarah A. and Jerome G. Lake. Systems Engineering Standards and Models Compared. Proceeding s of the Eighth International Symposium on Systems Engineering. INCOSE. 1998.
20. Barbacci, Mario R., Robert Ellison, Anthony J. Jattanze, Judith A. Stafford, Charles B. Weinstock, and William G. Wood. Quality Attribute Workshops (QAW). Third Ed. Pittsburg: Carnegie Mellon, 2003
21. Nazzaro, William. "Requirements Made Easy with User Stories." Online posting. 16 Oct. 2011 <http://www.develop.com/media /pdfs/RequirementsMadeEasywithUser Stories.pdf>, Last accessed on Sep 23rd, 2012
22. Kazman, Rick, Mark Klein, and Paul Clements. ATAM: Method for Architecture Evaluation. Pittsburg: 2000.
23. Ying, Yang. SWOT-TOPSIS Integration Method for Strategic Decision. 2010 International Conference on E-Business and E-Government. 2010.
24. Wikipedia. Analytic Hierarchy Process. < http:// en.wikipedia.org/wiki/Analytic_ Hierarchy _Process>, Last accessed on Sep 23rd, 2012
25. Evans, James R. and William M. Lindsay. Managing for Quality and Performance Excellence. 8th Ed. South-Western Cengage Learning; Mason. 2011.
26. Goldin, Leah, and Anthony Finkelstein. Abstraction-Based Requirements Management. Proceedings of the 2006 International Workshop on Role of Abstraction in Software Engineering. New York. 2006.
27. Karlson, Joachim and Kevin Ryan. "A Cost-Value Approach for Prioritizing Requirements." : Software 14 No. 5 (1997): 67-74.
28. Sullivan, William, Elin Wicks and C. Patrick Koelling. Engineering Economy. 14th Ed. Upper Saddle River: Pearson. 2009