Considerations for Successful Reuse in Systems Engineering

Jared Fortune[*]
University of Southern California, Los Angeles, CA

Ricardo Valerdi[†]
Massachusetts Institute of Technology, Cambridge, MA

Reuse is the idea of leveraging previously developed capabilities into a new project for the purposes of improving project characteristics (i.e. cost, schedule, risk). While reuse is a fairly well-known concept in domains such as software and product development, almost no research has been conducted on reuse being applied to the systems engineering domain. This investigation examines the potential considerations for the successful utilization of reuse in systems engineering.

I. Introduction

As the demand for systems with greater performance, lower cost, wider compatibility, higher reliability, and shorter development cycles continues to grow, the challenge of pushing the state-of-the-art becomes increasingly difficult and the result is a trend towards more complex systems. However, developing systems with increasing complexity can lead to longer schedules and higher costs. In order to alleviate some of the cost and schedule pressures on a development effort, elements of a previously developed system are frequently reused in some form in the new system. As will be discussed later, the reused elements are not limited to products or subsystems but can be knowledge, processes, people, designs, etc. By reusing elements of a previous system, the intention is to leverage a previously developed capability and therefore reduce the resources required for the new development effort.

In addition to systems becoming increasingly complex, very few systems are developed completely new from the ground up or "built from scratch". Even if a system is mostly new, nearly every system element, process, or component is based on knowledge or capabilities derived from previous experience. Therefore, although one could argue that reuse is inherent in the development of every system, this research is focused on the effects of specific instances of reuse; occasions where reuse is actively utilized with the aim of reducing the development effort or improving the performance of a new system.

II. Research Motivation

The motivation behind this research is twofold. First, based on the author's experience, to meet cost or schedule requirements, system architects often attempt to reuse systems elements from related programs or leverage experience in developing similar products. For example, due to cost and schedule constraints, the system architects of a NASA lunar mission were recently exploring the possibility of leveraging or reusing existing terrestrial technology, components, and designs in systems that would operate on the lunar surface. The argument behind this approach was that the reuse methodology would result in a cost or effort savings; however, there was limited to no evidence to support the proposed savings. Considering the architects were attempting to reuse capabilities that were possibly not designed for reuse and definitely not intended to be operated in a lunar environment, the actual benefits of reuse were suspect. Therefore, this research proposes considerations for the successful reuse of elements in the systems engineering domain, which could enable a better assessment of the viability and benefits of reuse in a system.

Second, one of the authors developed the Constructive Systems Engineering Cost Model (COSYSMO), which estimates the amount of systems engineering effort required for a project.[27] Currently, the model relies on a "built

[*] Doctoral Student, Industrial and Systems Engineering Department, 3715 McClintock Avenue, GER 240, Los Angeles, CA 90089.
[†] Research Associate, Engineering Systems Division, 77 Massachusetts Avenue, Cambridge, MA, 02139, Member.

from scratch" philosophy, meaning that when inputting values for the model drivers, the output effort will be to complete an entirely new system; however, this assumption will not always hold true. In some cases, a substantial amount of capabilities will be reused from a previous development effort, which should (but not in all cases) reduce the amount of new effort estimated by COSYSMO. Furthermore, the industrial community has identified the incorporation of reuse into the model as the top priority for future research.

III. Methodology

In order to address the effect of reuse in the systems engineering domain, this investigation examined how other closely-related domains define reuse. By understanding how the concept is addressed in domains such as software and product development, where research into reuse is more established, a methodology for successfully implementing reuse within the systems engineering domain could then be obtained. The fundamental questions addressed by this research are:

1) What principles about reuse can be drawn from the literature?
2) What are the considerations necessary for successful reuse in the planning, implementation, and measurement phases of systems engineering?
3) What are the drivers of successful reuse in systems architecture or systems engineering?

To answer these questions, the current state of reuse in other domains was assessed, the key factors or drivers of reuse were determined, and the main challenges or limitations of reuse in other domains were identified, although not all are applicable or scalable to the systems engineering domain. Both the software and product development domains were previously known to have research on reuse and were therefore likely to provide insight on the topic. From the review of reuse in these domains, observations on reuse were collected and the results were organized to better inform how reuse can be incorporated in the systems engineering domain.

IV. Principles of Reuse

This investigation consists of three sections: a brief background on reuse, reuse in the software domain, and reuse in the product development domain. Based on this review, possible "principles of reuse" are presented.

A. Background

The origin of reuse is not new; the notion of not "reinventing the wheel" can be traced back to ancient times, and still today, engineers continually try to solve problems by applying solutions from related situations to new ones.[22] Early forms of reuse include mathematical models and algorithms, instances where the consistency of calculations was critical and developers would reuse functions in order to ensure precise results.[22,26] A more recent example of architecture and systems engineering reuse is the Joint Strike Fighter, having multiple model variants based on a single core concept.

One of the fundamental motivators behind the concept of reuse is the idea that no one wants to reinvent an application for every project.[26] As mentioned above, given the increasing complexity of systems, leveraging as much previously developed capabilities as possible is extremely advantageous. In the software domain, project cost savings between 10-35% have been attributed to well-planned reuse strategies.[26] In the product development domain, reusing core platforms across product families have resulted in substantial design and development savings.[11] *Principle #1: Reuse is done for the purpose of economic benefit, intending to reduce schedule, reduce cost, or increase performance.*

While utilizing reuse can have its benefits and will usually result in less resources being required for new development projects, an often overlooked aspect is the fact that reuse is not "free" (similar to the idea of quality not being "free, as theorized by Juran).[19] *Principle #2: Reuse is not free, upfront investment is required.*

In both the software and product development domains, reuse of a program element or technology platform is typically decided on from the onset; meaning that the aspect of the project is specifically designed for reuse. Reusable components are typically more expensive than their non-reusable counterparts because the costs associated with the component are front loaded and need to operate within multiple application types.[13] Therefore, while reuse is often touted as easy way to reduce the cost or schedule of a project, it cannot be applied to all environments or situations and still deliver the promised returns. *Principle #3: Reuse needs to be planned from the conceptualization phase of programs.*

So what exactly is reuse? Pfleeger states that anything can be reused: tools, training, experience, requirements, designs, components, etc.[17] Basili defines reuse as the use of everything associated with a project, including knowledge.[22] Prieto-Diaz claims that reuse is the ability to apply a solution to specific instances in other domains.[22] Other researchers explain reuse as the opportunity for an existing asset to satisfy an unforeseen need or that it is simply something specifically produced to be reused or derived from a legacy system.[5,7] In yet another sense of the word, the lessons learned from past failures could also be examples of reuse; the success of a project can be just as dependent on what processes or components should not be used as it is to what should be used.[6] Similarly, when a process or component is finally developed and is proved reliable, reusing that capability can increase the likelihood of success. *Principle #4: Anything can be reused.*

Successful reuse is driven by business need, well designed architecture, commonality between applications, and adequately documented process.[7] However, the capability of utilizing reuse is not only limited by technical factors, but non-technical factors as well, such as personnel knowledge.[13,26] Without employees who understand the details of the item being reused and the intricacies of applying it into new domain, a reuse initiative may not be successful. In addition to difficulties associated with the successful implementation of reuse, measuring the benefits of such efforts can be challenging as well. A critical aspect of successfully proposing a reuse plan and then delivering on it is to accurately capture the amount of resources saved on the project.[19] This can be extremely difficult because while some domains such as software can use metrics like the reduction in new source lines of code, a systems engineer would have a much more difficult time measuring the reduction of resources specifically attributable to reuse. *Principle #5: Reuse is as much of a technical issue as it is an organizational one.*

B. Software

Software reuse is the process by which existing software products are reused in a new development effort, usually with minimal modification.[2] Similar to other instances of reuse, software reuse seeks to create new products using previously developed architectures and components.[24] To accomplish this, the software domain touches on aspects similar to those described in product development, domain and application engineering by typically creating software that are either elements designed to be reused in their entirety or generic enough to be applied to specialized instances.[16] Interestingly, in terms of successfully reusing software, Hollenbach argues that a software module can only be called "reusable" if it has been successfully reused.[9]

Contrary to what one may think, software reuse is not just limited to design libraries and modules. Bollinger and Pfleeger argue that any software project characteristic, for example personnel work experience, that can affect the outcome of the project is potentially reusable.[2] Mili agrees with this, citing reuse of products and processes.[15] Selby also supports this idea, citing personnel and processes, in addition to design objects, histories, and procedures as opportunities for reuse.[24] Therefore a common theme not only with software reuse, but reuse across multiple domains, is more than just the technical capability of reusing a product in another application that affects successful reuse, it also depends on the people and the processes (see Principle #5).

The software domain has definitions for types of reuse that may be applicable to other areas, in particular the systems engineering domain. Poulin claims that assembling code, tailoring code, and porting code are the possible forms of software reuse.[19] Stephens defines the types of software reuse as framework, template, object, or information; all of which vary in degrees from fully reusing products, to reusing with modifications, to inheritance, to passing along information about the product.[26] Davis states that reuse can be opportunistic (reuse strategies for individual projects), integrated (standardized process incorporated into organization), leveraged (designed to optimize reuse over a set of related products), and anticipating (creating new opportunities around reuse).[5] In general, the definitions for software reuse presented in the literature highlight the fact that reuse can be individual project orientated or strategic-focused. Poulin and Stephens both define reuse from a project perspective with the amounts of reuse varying. Davis takes a more strategic view of reuse in the sense that it can be focused at the individual project level or increase all the way to the organizational level where multiple product families are being reused. Therefore, unless the development of new systems changes to where designed for reuse is a consideration, many of the concepts and ideas presented in the literature of other domains may not be directly applicable.

The motivation most often stated in the literature for reuse in software is a reduction in the cost of developing new products by avoiding redevelopment of capabilities and increasing productivity by incorporating components whose reliability has already been established.[2,20,24] However, to be worthwhile, the benefits must outweigh the costs, which several models attempt to address. Bollinger and Pfleeger propose a model that calculates the net benefit of reuse by taking the sum of savings on development and adaptation, and subtracting the upfront development investment.[2] An issue with this model is that while it may be applicable for instances of products designed for reuse, not all reused products were intended to be reused. Therefore, determining the upfront development investment could be difficult in some situations. Gaffney and Durek suggest a cost model for evaluating reuse similar to the model by Bollinger and Pfleeger; however, it assumes that the cost of development of the reused product is amortized across several instances of the product.[17] Again, this calculation could be problematic if the product was not originally intended to be reused or if the product is only reused a limited number of instances. Poulin and Caruso also developed a reuse model with a return on investment perspective, but it faces similar drawbacks as the models mentioned above.[17] In general, software cost models that address reuse appear to require either the upfront development cost for the reusable product or multiple instances of the product being reused. These are drawbacks as the values of these parameters are not always known or even calculable. In addition, most of the models examined assumed the economic benefit of reuse to be linear, when in many cases it is incorrect.[2,24] For example, from a lifecycle perspective, in almost any project phase, the amount of resources expended to provide for reusability could be greater or less than the amount of resources saved or benefit derived. *Principle #6: The benefits of reuse are not linear.*

In general, most of the literature states that for software reuse to be successful, significant effort has to be expended during the upfront design and development phases to make the product readily reusable (see Principle #3).[2] Mili states that reusable software must be reusable by design and also be developed from reusable software.[15] One explanation for this is that because reusable software tends to be larger in size and not as specialized in order to operate across multiple applications.[16] Therefore, in addition to upfront costs or design effort, it may be difficult for reuse to scale with complexity (see Principle #4).

To justify the development of software for reuse, the benefits have to be correctly presented, typically in financial terms; although these are not always available, such as the cost of developing the product without reuse (see Principle #1).[19] In order to support a reuse investment decision, metrics such as reuse percent, reuse cost avoidance, and reuse value added are used.[20] However, in cases where detailed project knowledge is not available or a product was not designed for reuse, the values of these parameters may not be supportable or quantifiable. Further complicating matters is the fact that there is no industry standard for reuse metrics, which inhibits the ability of a company to make objective assessments on how and where to best invest resources for reuse.[2,19]

C. Product Development

Reuse in product development typically consists of designing core product platforms that can serve as the basis of multiple product versions across different application domains (i.e. Joint Strike Fighter). The motivation for reuse in the product development domain is to reduce the development time for new products and retain market agility across various product families.[7] This is achieved by either developing a generic product that can be further specialized for specific applications or a product that is intended to be reused.[11]

As is the case in the software domain, product reuse is more than just technology or product platform related. While discussing an information technology system in particular, Boehm cites four factors that contribute to building a successful system: product, success, process, and property.[1] It is possible that these same categories can be generalized to product development, and more specifically to reuse.

Process reuse needs to be considered in this discussion because it appears successful reuse is partially dependent on the ability to recreate or incorporate a previously developed system, which could potentially not have been developed in-house, based on available documentation or personnel knowledge. Adequate processes are essential to all types of businesses as they enable better communications, efficient interactions, and performance benchmarking.[4] Almost all organizations have experience with process reuse, frequently purchasing packaged enterprise systems that are then adapted to suit their particular domain.[4]

A problem with process reuse is that different types of systems may require entirely different processes, therefore the opportunities for reuse may be limited.[17] This is observed with systems have varying levels of complexity and an inappropriate level of process detail could inhibit reuse, such as defining a process at too high of a level.[17] Furthermore, process reuse is difficult to impose on an organization that is fragmented and does not have the ability to absorb new processes.[8]

Problems also arise when organizations approach process reuse as an independent collection of tools and technology, or when an organization focuses on technical issues of reuse without addressing the non-technical ones.[5] Recently, organizations appear to becoming less interested in *improving* processes and more focused on *maintaining* current processes.[13] Overall, the capability for reuse is a function of proficiency, efficiency, and effectiveness that can be achieved by the organization.[5]

Knowledge reuse also appears to be another key factor that influences the success of reuse. Defined as the process by which an entity is able to locate and use shared knowledge, knowledge reuse improves the product development process and help to foster innovation.[14] Typically, knowledge reuse is enabled by a knowledge management system, a means of formally facilitating the creation, access and reuse of knowledge, which will enable an organization to more effectively solve problems and make decisions.[3,23] A knowledge management system assists in the capturing and recording of existing business knowledge, which is a strategic resource for a competitive organization.[14,25]

In terms of knowledge and personnel, successful reuse may reduce total effort required for a project, but likely requires additional and often costly human capital to do so. These people are likely more valuable to an organization than most and while reuse could reduce a project's cost or effort, other projects may suffer due to the shuffling of such personnel. For example, highly capable people are required to recognize the opportunities for reuse not only at the initial design of the product, such as during architecture development, but also sporadically throughout the life of the project, such as during the systems engineering activities.[14]

Such capabilities are indicative of tacit knowledge that employees hold but do not always document. Most companies' training and development programs are designed to transfer explicit technical or managerial knowledge, but not so called "deep smarts".[12] In addition, it is very difficult for personnel to become knowledgeable in a passive way, actively experiencing something is considerably more valuable than having it described.[6] Therefore, attempting to pursue a reuse strategy without sufficient knowledge of the previously developed system (or in some instances even with thorough documentation available) may prove difficult and potentially very costly.

V. Observations

Based on the investigation conducted on reuse in domains such as software and product development, some generalizations can be made about reuse. First, it appears that most instances of reuse are situations where products were designed to be reused from the beginning and specifically for that purpose (see Principle #3). Strategies have been created to invest resources upfront to ensure a product was reusable and cost models have been developed to assess the benefits of utilizing reusable products (see Principles #2 and #1). The fact that most previous research focuses on products designed for reuse is problematic for reuse in the systems engineering domain (see Principle #4). Since most system engineering activities are not designed specifically to be reused but instead leveraged when the opportunity is available, it would appear that only the high level concepts and lessons presented on reuse in other domains would be applicable to systems engineering. For example, developing a cost model for assessing reuse in systems engineering would be worthwhile, but following the methodology presented in the software literature may not be able to deliver reasonable results (see Principle #6).

In addition to the theme of designed for reuse, the idea of reuse metrics and quantifying the benefits of reuse was common in the research reviewed. Reuse metrics were proposed as a means of quantifying how much of a resource a reuse strategy was saving. By subtracting the investment or development cost associated with the reuse strategy from the savings generated by it, a net benefit of reuse could be generated (see Principle #1). Although quantifying the benefits of reuse sounds trivial, from a systems engineering domain perspective, this could be rather challenging. For example, not only would the benefits of reuse be difficult to quantify because the full implications to the architecture may not be immediately known, but also the costs associated with developing a reusable system element may be impossible to capture. Since a system element may not always be designed for reuse, trying to assess the cost of reuse associated with a product that has already been developed for another application could be filled with uncertainty (see Principle #6).[28] One potential solution to this issue is the development of unit of measure for systems development, equivalent to source lines of code for software. By creating a measure that can better define systems development, the value of reuse in the domain can be more accurately assessed.

A final observation from the literature review is that a large number of sources supported the belief that successful reuse requires more than just a reusable product, it also takes knowledgeable and capable personnel as well as adequate documentation, organization, and production processes (see Principle #5). Research from both the software and product development domains stated that having the correct people and processes were just as important as the right product platform to achieving a successful reuse program. The identification of people and processes as keys to successful reuse could be extremely applicable for the systems engineering domain, since those two factors can significantly impact systems development with or without reuse.

While investigating reuse in the product development domain, it became apparent that successful reuse in this area was not only a function of applying a proven technology across multiple products, but also applying the associated knowledge and process capabilities. This does not imply that such factors are not relevant to software development (they definitely are) but more emphasis was given to the discussion of reuse in the product development domain. Therefore, most of the discussion related to product development comes in the proceeding paragraphs on knowledge and process reuse.

Lastly, throughout the study, several challenges to the successful utilization of reuse became apparent. Primarily, challenges exist in planning for reuse, implementing reuse, and measuring reuse.

VI. Success Factors

Reuse is highly dependent on an established product, heritage design or experience, well documented processes, and personnel with knowledge of the product. Therefore, the successful development of a product as well as the reuse of a product may share common factors. They both rely on an existing product foundation, experience that the product functions correctly, a development process that is repeatable, and the intellectual capacity to both understand the existing product and develop the new product. Therefore, this investigation has identified successful reuse in the systems engineering domain as a function of three project factors:

a) Platform
b) People
c) Processes

Platform refers product or technology that is intended to be reused. Fundamentally, for a reuse program to be successful, the product needs to capable of being reuse. Successful reuse can be defined as instances where the benefits of reusing a product outweigh the costs. While Principle #4 states anything can be reused, not everything can be reused successfully.

People are the available personnel for the reuse project and their knowledge of the heritage/new products. Personnel needs to be knowledgeable enough to avoid the pitfalls associated with reuse as well as have a familiarity with both the heritage and new products. Because of this, successful reuse may require an organization to take experienced individuals, ones who are able to see the whole picture but still focus on a particular detail, such as a true systems engineer, and ensure their knowledge is captured.[12]

Processes encompass the organization, documentation, and production. The organization needs to have the flexibility for reuse, possibly utilizing a matrix structure. Documentation needs to be available from heritage products that are being reused and new projects should be well documented to enable reuse in the future. Lastly, production capabilities need to exist so the system can actually be developed and eventually delivered. These are three possible drivers to capture the likelihood of successful reuse and potentially indicate the expected savings of such a strategy.

In addition, reuse may have the most leverage when applied early in the development of a system, such as during the architecture design phase, one of the earliest systems engineering activities. By the time some of the other systems engineering activities are occurring, the “window of opportunity” for reuse may be limited and attempts at reuse could inadvertently affect other aspects of the system in the process.

VII. Conclusion

This investigation examined the potential considerations for the successful utilization of reuse in systems engineering by reviewing how related domains, such as software and product development, handle reuse. By studying how reuse is addressed in other domains, six principles were identified as key considerations for reuse:

1) Reuse is done for the purpose of economic benefit, intending to reduce schedule, reduce cost, or increase performance
2) Reuse is not free, upfront investment is required
3) Reuse needs to be planned from the conceptualization phase of programs
4) Anything can be reused
5) Reuse is as much of a technical issue as it is an organizational one
6) The benefits of reuse are not linear

In addition to identifying these six principles, three factors appeared to contribute significantly to the successful utilization of reuse:

a) Platform – appropriate product or technology, primed for reuse
b) People – adequate knowledge and understanding of both the heritage and new products
c) Processes – sufficient documentation to acquire and capture knowledge applicable to reuse as well as the capability actually deliver a system incorporating or enabling reuse

Collectively, these principles of reuse and success factors need to be considered in future reuse efforts and incorporated into systems architecture/engineering tradeoffs and cost models, such as COSYSMO. This and other future areas of research include:

- Validating the platform, people, and process factors as drivers of reuse; potentially incorporating these as drivers in a systems architecture/engineering cost model
- Assessing the potential for a reuse optimum; program cost or risk could potentially be reduced by utilizing a certain level of reuse in a system, although not likely along a linear relationship
- Determining the viability of a systems architecture/engineering unit of measurement; would assist in justifying the benefits of reuse
- Examining how successful reuse scales with system complexity; reuse may deliver varying amounts of benefit depending on system characteristics, although not likely along a linear relationship
- Measuring tacit knowledge in the systems engineering domain; a better understanding will likely improve the assessment of people (knowledge capacity) driver
- Identifying how scalable the reuse principles are to the systems engineering domain, architecture tradeoff studies in particular

Acknowledgments

The authors wish to thank the Air Force Space & Missile Systems Center for their support of this research, as well as the USC Center for Systems and Software Engineering Corporate Affiliates and the MIT Lean Aerospace Initiative Consortium Members.

References

[1]Boehm, B., "When Models Collide: Lessons from Software Systems Analysis," *IT Professional,* Vol. 1, No. 1, 1999, pp. 49-56.

[2]Bollinger, T. and Pfleeger, S., "Economics of Reuse: Issues and Alternatives," *Information and Software Technology,* Vol. 32, No. 10, 1990, pp. 643-652

[3]Huang, C. and Liang, W., "Explication and Sharing of Design Knowledge Through a Novel Product Design Approach," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews,* Vol. 36, No. 3, 2006, pp. 426-438.

[4]Davenport, T., "The Coming Commoditization of Processes," *Harvard Business Review,* Vol. 83, No. 6, 2005, pp. 100-109.

[5]Davis, T., "The Reuse Capability Model: A Basis for Improving an Organization's Reuse Capability," *Second International Workshop on Advances in Software Reuse Proceedings,* 1993, pp. 126-133.

[6]Garvin, D., "Building a Learning Organization," *Harvard Business Review,* Vol. 71, No. 7, 1993, pp. 78-92.

[7]Griss, M., "Architecting for Large-Scale Systematic Component Reuse," *21st International Conference on Software Engineering Proceedings,* 1999, pp. 615-616.

[8]Hammer, M. and Stanton, S., "How Process Enterprise Really Work," *Harvard Business Review,* Vol. 77, No. 6, 1999, pp. 108-119.

[9]Hollenbach, C., "Software Process Reuse in an Industrial Setting," *Fourth International Conference on Software Reuse Proceedings,* 1996, pp. 22-30.

[10]Kontio, J., Caldiera, G., and Basili, V., "Defining Factors, Goals and Criteria for Reusable Component Evaluation," *Conference of the Centre for Advanced Studies on Collaborative Research Proceedings,* 1996, pp. 21-33.

[11]Lam, W., "A Case Study of Requirements Reuse Through Product Families," *Annals of Software Engineering,* Vol. 5, 1998, pp. 253-277.

[12]Leonard, D. and Swap, W., "Deep Smarts," *Harvard Business Review,* Vol. 82, No. 9, 2004, pp. 88-97.

[13]Lynex, A. and Layzell, P., "Organisational Considerations for Software Reuse," *Annals of Software Engineering,* Vol. 5, 1998, pp. 105-124.

[14]Majchrzak, A., Cooper, L., and Neece, O., "Knowledge Reuse for Innovation," *Management Science,* Vol. 50, No. 2, 2004, pp. 174-188.

[15]Mili, H., Mili, F., and Mili, A., "Reusing Software: Issues and Research Directions," *IEEE Transactions on Software Engineering,* Vol. 21, No. 6, 1995, pp. 528-562.

[16]Mili, A., Yacoub, S., Addy, E., and Mili, A. "Toward an Engineering Discipline of Software Reuse," *IEEE Software,* Vol. 16, No. 5, 1999, pp. 22-31

[17]Perry, D., "Practical Issues in Process Reuse," *Software Process Workshop Proceedings,* 1996, pp. 12-14.

[18]Pfleeger, S. and Bollinger, T., "The Economics of Reuse: New Approaches to Modeling and Assessing Cost," *Information and Software Technology,* Vol. 36, No. 8, 1994, pp. 475-484.

[19]Poulin, J. "A Reuse Metrics and Return on Investment Model," *Second International Workshop on Software Reusability Proceedings*, 1993, pp. 152-166.

[20]Poulin, J., "Determining the Value of a Corporate Reuse Program," *First International Software Metrics Symposium Proceedings*, 1993, pp. 16-27.

[21]Poulin, J., *Measuring Software Reuse,* Addison-Wesley, Reading, Massachusetts, 1997.

[22]Prieto-Diaz, R., "Status Report: Software Reusability," *IEEE Software*, Vol. 10, No. 3, 1993, pp. 61-66.

[23]Satyadas, A., "Knowledge Management Tutorial: An Editorial Overview," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews,* Vol. 31, No. 4, 2001, pp. 429-437.

[24]Selby, R., "Enabling Reuse-Based Software Development of Large-Scale Systems," *IEEE Transactions on Software Engineering,* Vol. 31, No. 6, 2005, pp. 495-510.

[25]Sharp, D., "Knowledge Management Today: Challenges and Opportunities," *Information Systems Management,* Vol. 20, No. 2, 2003, pp. 32-37.

[26]Stephens, R., "Measuring Enterprise Reuse in Large Corporate Environment," *Proceedings of the Thirty-Sixth Southeastern Symposium on System Theory*, 2004, pp. 565-569.

[27]Valerdi, R., "The Constructive Systems Engineering Cost Model (COSYSMO)," PhD Dissertation, University of Southern California, 2005.

[28]Valerdi, R. and Davidz, H., “Empirical Research in Systems Engineering: Challenges and Opportunities of a New Frontier," *Systems Engineering*, Vol. 12, No. 2, 2009.