Integrating Cognitive Systems Engineering Throughout the Systems Engineering Process

William C. Elm
James W. Gualtieri
Brian P. McKenna
James S. Tittle
Jay E. Peffer
Samantha S. Szymczak
Justin B. Grossman
Resilient Cognitive Solutions, LLC

Abstract: In order for cognitive systems engineering (CSE) to deliver the benefits of recent theoretical advances in actual systems being developed, the insights of CSE must be transformed into pragmatic engineering practices. The CSE engineering practices described in this article (using the applied cognitive systems engineering methodology as an exemplar) are typical of just such an engineering adaptation of revolutionary science and represent engineering practices that are dedicated to building effective systems. CSE research insights can have a significant impact on their corresponding systems engineering (SE) practices by expanding SE's concept of a system from just the technology components to a joint cognitive system (Hollnagel & Woods, 2005) and expanding the associated SE practices appropriately. This article uses the Department of Defense system life cycle and the SE V-model (Forsberg & Mooz, 1992) as SE process exemplars to illustrate how CSE engineering practices can be integrated into the SE process. Specifically, we propose four key integration points where CSE can contribute significantly to the SE process: concept refinement, software development, testing, and postsystem development (i.e., operations support, training, and maintenance). Our approach shows that the practice of CSE does not compete with SE but, instead, augments current SE practices to ensure that the technology components are engineered with the users' cognitive needs in mind.

Introduction

> Engineering makes a reality of the potential value of science by translating scientific knowledge into tools, resources, energy and labor to bring them into the service of man. . . . To make contributions of this kind the engineer requires the imagination to visualize the needs of society and to appreciate what is possible as well as the technological and broad social age understanding to bring his vision to reality.
>
> —Sir Eric Ashby (1958, p. 72)

ADDRESS CORRESPONDENCE TO: William C. Elm, Resilient Cognitive Solutions, LLC, 113 S. 26th St., Suite 200, Pittsburgh, PA 15203, welm@resilientcognitivesolutions.com. Visit the JCEDM Online Companion at *http://cedm.webexone.com*.

Journal of Cognitive Engineering and Decision Making, Volume 2, Number 3, Fall 2008, pp. 249–273.
DOI 10.1518/155534308X377108.

Both the scientific and engineering perspectives are essential to the development of advanced, effective systems. It is significant that in its 25th year, the field of cognitive systems engineering (CSE) is experiencing a renewed interest in exploring its role within systems engineering. This interest shows most prominently in the recent National Academy of Science panel report by Pew and Mavor (2007), which focuses on the need for the engineering perspective to "catch up" to its scientific counterpart. Hollnagel and Woods (1983) coined the term *cognitive systems engineering* in a paper subtitled "New Wine in New Bottles" while Woods worked in the research department of one of the largest engineering organizations in the world. The first textbook related to the field, Rasmussen's (1986) *Information Processing and Human-Machine Interaction,* was authored by a control systems engineer who was interested in building systems that were more capable of supporting human capabilities.

However, since that time the scientific perspective's advances have far outpaced the application of CSE by the engineering community. That is, far more studies have been conducted and papers written about CSE than there have been systems built that have utilized CSE as the basis for their design. Although CSE was born as an engineering necessity, it has been advanced primarily by a scientific mindset (i.e., studying and describing the theory). The number of fielded systems containing a significant CSE contribution is extremely small. For the most part, scientific advances have not been complemented by the rigorous engineering adaptation of those scientific concepts into sound engineering practices. Our purpose in this article is to discuss expansions to current systems engineering practices that allow the theoretical advances of CSE to be integrated into the systems engineering (SE) development process. These changes involve both the development of pragmatic CSE engineering practices as well as modification of SE practices currently in use.

The complementary relationship between engineers and their scientific brethren is found in every other engineering discipline, and it also must be found in CSE so that the research of cognitive psychologists and cognitive systems scientists can be adapted for engineering practices used by cognitive systems engineers and also the larger community of systems engineers. CSE has to complement the science with a focus on SE. Although researchers do well focusing on one area of the larger CSE process (such as knowledge elicitation or cognitive work analysis), the engineering implementation of CSE must be a holistic process that folds the insights of all aspects of CSE into the pragmatic engineering practices used by actual system developers.

The CSE concepts described in this article are the general principles that are necessary for adapting science concepts and findings for engineering practice. These concepts are instantiated in the practices described in a specific CSE methodology, applied cognitive systems engineering (ACSE). It is left to the reader to explore how these general CSE pragmatic engineering concepts are or are not part of other CSE methodologies. ACSE is the fourth generation of a set of CSE engineering practices that have evolved steadily since the near-catastrophe at Three Mile Island. It is the recent evolutionary improvement over the applied cognitive work analysis (ACWA) methodology, reflecting the larger SE focus of the methodology

and the misnomer implied by the word *analysis* in ACWA. Over that time, ACSE and its predecessors have resulted in more than 40 major systems (e.g., AWARE: nuclear power plant alarm management system; TRANSCOM Regulating and Command and Control Evacuation System [TRAC2ES]: regulation and evacuation of casualties; Information Warfare Planning Capability: assessment synchronization and evaluation; performance view: communications network monitoring) under six corporate umbrellas.

Given the evolution of CSE over the past 25 years, how is it now going to produce the kinds of engineering practices that will make it part of the mainstream SE community? The good news is that the SE community is amenable to much of what CSE is prepared to provide. Looking behind the various SE concerns and issues being discussed in the SE professional literature reveals the opportunity for CSE and the implicit requests for CSE assistance as a member of the SE community. This article is about making that connection between CSE and SE at the engineering practices level.

Systems Engineering Background

SE has been defined simply as an interdisciplinary approach and a means to enable the realization of successful systems (International Council on Systems Engineering, 2004). A more complete definition describes SE as a robust approach to the design, creation, and operation of systems that consist of the identification and quantification of system goals and requirements, creation of alternative system design concepts, performance of design trades, selection and implementation of the best design, verification that the design is properly built and integrated, and postimplementation assessment of how well the system achieved its goals (Institute of Electrical and Electronics Engineers [IEEE], 2000; National Aeronautics and Space Administration, 1995).

From an SE perspective, a system can be thought of as a collection of multiple technology elements and hardware subsystems that, in combination, produce a result beyond anything the individual elements could accomplish in isolation. Within the SE community, a *system* has been defined as

> a construct or collection of different elements that together produce results not obtainable by the elements alone. The elements, or parts, can include people, hardware, software, facilities, policies, and documents; that is, all things required to produce systems-level results. The results include system level qualities, properties, characteristics, functions, behavior and performance. The value added by the system as a whole, beyond that contributed independently by the parts, is primarily created by the relationship among the parts; that is, how they are interconnected. (Rechtin, 2000, p. 26)

SE is a top-down, comprehensive, iterative, and recursive problem-solving process that is applied sequentially through all stages of development by integrated multidisciplinary teams; it transforms needs and requirements into a set of system

products and process descriptions so that the system can be built tested, fielded, and supported. People are included as part of the definition of a system, but their role in that system is generally poorly specified, and the focus of the effort is on the technology components that can actually be "engineered." It is critical for cognitive systems engineers to be better integrated into these multidisciplinary system development teams if they are to be successful in transforming the operator's cognitive needs into a set of system engineering products by providing this critically needed input to the system development process.

Successful integration of CSE into the SE development process involves finding the proper scope for designating and understanding the systems being designed. One of the most important ways CSE can have a significant impact on the SE process is by expanding the very concept of a system. People are more than just another system component. They bring a unique set of demands, capabilities, and goals with respect to the system. Hollnagel and Woods (2005) argued that the proper scope of a system is actually the joint cognitive system (JCS): the combination of human problem solver and the automation and/or technologies that must act as coagents to achieve goals and objectives in a complex work domain.

In order for the designed system to achieve its goals, the human problem solver (or problem solvers) must work with the technology as a coordinated team. From the JCS perspective, the system is no longer simply the technology and hardware subsystems but also the human decision-making subsystem acting in concert with the technology and hardware to achieve the joint system's goals.

The key assumption of the JCS approach as a foundation for SE is that the operator and enabling technologies must be seen as a single, fully integrated object. The JCS must be the fundamental unit of analysis in system design (McKenna, Gualtieri, & Elm, 2006). The JCS is not a set of independent subsystems but consists of the entire set of humans, technology, and automation systems operating in conjunction. The potential power of a properly designed JCS can be seen from research showing that a human-plus-technology/automation team working together performs better than either working alone (Layton, Smith, & McCoy, 1994). Thus, although the human practitioners and the technology are separated physically, in a properly designed JCS they always should be coupled functionally. The people are not just interacting with the technology but, rather, using it to perform some function in order to accomplish some goal. Seen this way, the technology can be understood as not merely a separate subsystem component that can have some human interface bolted on after it is designed; rather, it specifies and constrains how human intentions will be transformed into actions in the domain of interest.

Systems Engineering and the Acquisition Process

The SE process formed the basis for the structure of the Department of Defense (DoD) system acquisition process. However, the large scale of DoD systems acquisitions has created inertial forces, which have circled back on themselves and now shape and control the SE process. Currently, they are almost inseparable, with key

acquisition decisions and payments coupled to key SE milestones and outputs. Therefore, to better understand how CSE can integrate into SE, it is useful to consider how the SE process is captured within the acquisition framework. This is used as a specific example of an SE methodology in order to more easily describe the integration of CSE with SE. Although the example discussed in this paper is specific (ACSE with DoD), the points being made are general. It is expected that any mature CSE process should exhibit the same opportunities for integration with any formalized SE process.

The fundamental phases of the DoD process are concept refinement, technology development, system development and demonstration, production and deployment, and operations support. These five SE activities are formalized as part of the DoD systems acquisition process (see Figure 1). At each stage of the DoD systems acquisition process, milestones and performance reviews occur to ensure that the SE process has been successful to that point.

Life-cycle integration is a continuous process that cannot be achieved by a one-time application of SE principles. For some systems, the development process (through initial operational capability [IOC]) can last 10 to 15 years (Defense Science Board, 2007). Throughout this period, there are myriad decisions to be made regarding the form the final system will take. In order for the SE process to successfully build a JCS, it is important that CSE be involved for the entire development process, not just as an afterthought at the end of a development process. CSE cannot be thought of as a consultant service whose contribution can be realized in a few meetings and a week of time.

Part of CSE's limited impact on major systems is that traditionally, if CSE has any role at all, it has been applied solely during the development of a user interface, often after that interface has been implemented by the software developers. This interface is typically "bolted on" after all the engineering decisions regarding the underlying technology and hardware have been made. This is too late in the process to have any meaningful impact, preventing true integration of the human and technology subsystems. Many of the large decisions regarding system architecture (i.e., the design of the set of relations between the parts of a system) and functionality have been locked down, leaving very little "maneuver room" for the cognitive systems engineer to have the needed impact. Systems integration is achieved only

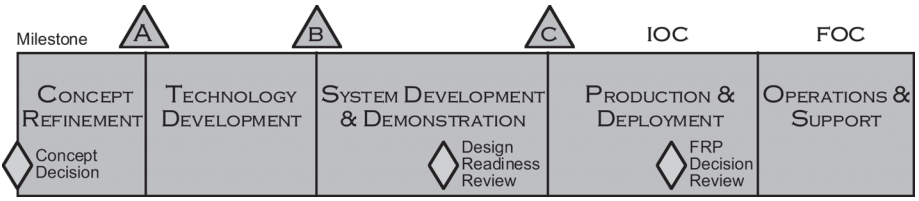

Figure 1. Overview of the Department of Defense system acquisition process. IOC = initial operational capability, FOC = final operating capability, FRP = full-rate production.

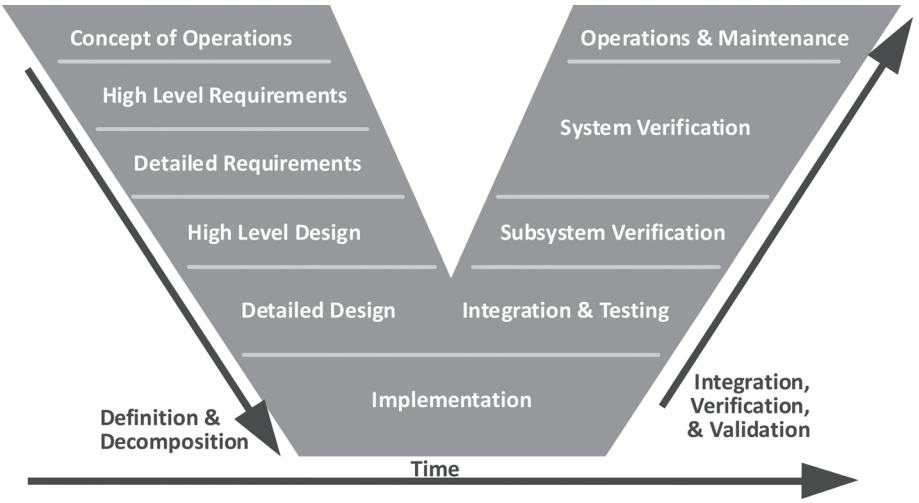

Figure 2. Overview of V-model of systems engineering.

through the continuity of subsystem development—that is, concurrent consideration of all SE needs during the development process, including the CSE needs.

This paper uses, in addition to the DoD system acquisition process, the V-model (Forsberg & Mooz, 1992) to illustrate how CSE can be integrated into the SE process (see Figure 2). Both illustrate the temporal sequence of events necessary for systems development. However, the V-model contains a bit more detail within each acquisition phase, and it emphasizes the coordination of testing and how testing at various levels of abstraction corresponds to the level of system design.

Others (Pew & Mavor, 2007) have suggested using an incremental commitment model (ICM; Boehm & Lane, 2007) for the integration of CSE with SE. The simpler V-model is more convenient to illustrate the points of this paper. Once this is understood, the reader is free to expand the discussion to the incremental SE processes of the ICM method.

The V-model is a graphical representation of the SE life cycle. It is designed to simplify the understanding of the complexity associated with developing systems within a system validation framework. It is used to define a uniform procedure for product development. The V-model reflects the simple fact that a system can be validated only if it "runs" (Cockburn, 1994). Before it runs, it must be built and debugged. Each piece is based on a design, which is based on requirements.

Time proceeds from left to right in the V-model. The left side of the V-model represents the decomposition of requirements and creation of system specifications. The right side of the V-model represents the integration of parts and their verification (Defense Acquisition University Press, 2001). In other words, the SE team must gather requirements before they design, design before they build, build before they test, and test before they validate the initial requirements.

The remainder of this paper discusses the generic opportunity for CSE's role during each phase of the SE process. Further, the ACSE CSE methodology and its connection to the DoD system phases and the SE V-model provide specific examples of the engineering practices to implement these opportunities. Only by integrating CSE throughout the larger system development process can (a) these systems be made truly effective at the intersection of agent, technology, and work; and (b) the field of CSE grow through the SE community's adoption of sound CSE engineering practices.

Integration Point 1: CSE's Role in Concept Refinement

Systems engineers, seeking to field an operational system, work from project infancy to the installation and support of that system. The initiation of the development of a new system requires that systems engineers become involved in the process of developing the requirements for the system to determine exactly what the system must do.

Nominally, the concept refinement process starts with customer inputs. The customer is asked, "What capabilities do you need?" Inputs consist primarily of the customer's needs, objectives, and project constraints. This process is typically informal and literally captures the subjective opinions of users (and pseudo-users). CSE offers a powerful complement: supporting the collection of these customer inputs through the use of its various knowledge elicitation techniques and subsequent analysis.

This raw knowledge elicitation is transformed through a cognitive work analysis into a representation of the goals and demands of the work domain (Elm, Potter, Gualtieri, Roth, & Easter, 2003; Rasmussen, 1986; Rasmussen, Pejtersen, & Goodstein, 1994; Vicente, 1999; Woods & Hollnagel, 1987). Although this transformation is routinely part of the CSE process, how to integrate it into the system development process has been more problematic. The results of this analysis produce a new class of requirements, embodying the CSE principles necessary to implement an effective JCS.

Brooks (1987) claimed the specification, design, and testing of the concept is the most difficult challenge within software engineering—more difficult than the actual code writing. Essentially, determining what a JCS is supposed to be has been more difficult than determining how to construct the JCS. However, building code gets the most focus in SE education and training; little education goes toward the requirements development process, despite its difficulty (Brackett, 1990). In fact, the challenge of developing effective requirements is reflected in the many and varied techniques that the SE community uses to try to fill this void, such as Jacobson use cases (Jacobson, 1992), Rapid Application Development (RAD)/Joint Application Development (JAD; Andrews, 1991), and extreme programming (Beck, 2000). CSE is ideally suited to be the key breakthrough in the development of good system requirements because of its focus on overall system goals and the associated cognitive work (including coordination) that needs to be accomplished by the people using the system to achieve those goals.

Requirements engineering seeks to elicit high-level goals, refine these goals, and assign the responsibilities to various agents in order to complete the goals (Feather, Fickas, van Lamsweerde, & Ponsard, 1998). Often, this requirements generation process is little more than recording what the customer requests. Although the process is more complex than this, the customer, as end user, is seen as the authority by which requirements must be developed. The designer asks the customer what he or she wants in a system, and the customer provides this information.

Although this seems a straightforward and logical approach, it has its flaws. Customers often know what they want their system to do at a very broad level but find defining the concept itself is a very difficult process. A simple question-and-answer approach (or questionnaire) does not adequately address the deeper issues inherent in the domain, and it may fall short of generating the proper requirements because of the inability of users and questioners to communicate effectively (Goguen & Linde, 1993). Additionally, asking the customers generally produces only the requirements that the customers think they need (or can think of at those particular times), ignoring other valuable requirements (Ernst, Jamieson, & Mylopoulos, 2006). Furthermore, this process is made more difficult by the fact that even knowledgeable and well-practiced experts can have great difficulty verbalizing their procedural knowledge regarding their own expertise (Ericsson & Simon, 1993). Beyond that, users have immense difficulty in describing an "envisioned world" much beyond current practices and systems (Hollnagel & Woods, 2005).

Current SE requirements practitioners struggle to discover information needs for the human decision-making agent portion of a system (Papasimeon & Heinze, 2000). Commonly, these attempts provide only lists of data that "shall be presented to the user." It is not adequate to provide users with only a data list display (Elm et al., 2003). Doing so adds cognitive work, forcing the user to process and integrate data on the fly. At best, this processing is done correctly but adds time to the decision-making process. At worst, the processing leads to a mistake attributable to an overburden of cognitive work placed on the user, possibly leading to a grave accident (McKenna, Gualtieri, & Elm, 2007). CSE can provide critical input, because the SE process demands requirements that are understandable, unambiguous, comprehensive, complete, and concise and that incorporate the human subsystem into the larger system (Turk, 2006).

CSE injects a different class of requirements into the SE process (Figure 3). Specifically, CSE provides requirements that focus on the decisions the envisioned system must support (in our terms, a JCS) and the coordination requirements necessary for the interface between the human subsystem and all the other subsystems. Traditional human-focused requirements address only what data need to be on the graphical user interface provided to the operator.

These data availability requirements tend to be very generic, and they do not provide sufficient guidance about frames of reference and information relationships to support an effective representation design process. This makes it impossible for the requirements process to drive the SE problem solvers to design novel, effective

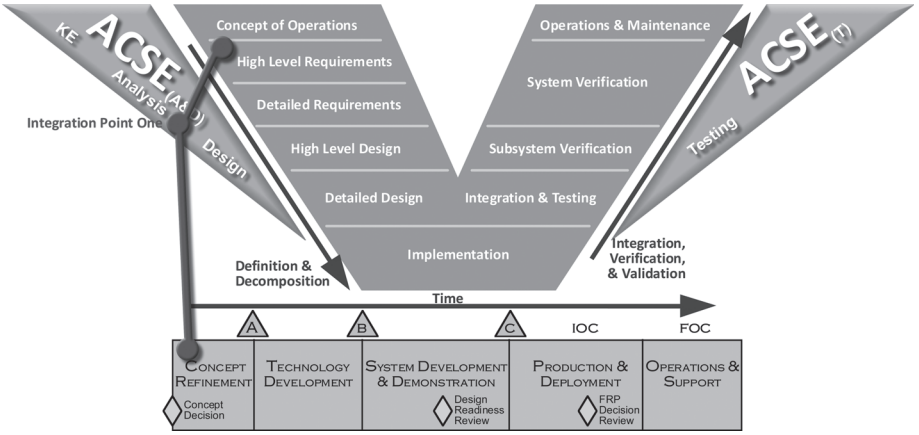

Figure 3. Cognitive systems engineering integration into the systems engineering requirements generation process. ACSE = applied cognitive systems engineering, KE = knowledge elicitation, IOC = initial operational capability, FOC = final operating capability, FRP = full-rate production.

decision support features, resulting in the repetition of designs based on traditional interface formats (e.g., maps, spreadsheets). CSE-based design requirements specify the relevant information relationships, appropriate frames of reference to highlight those relationships, and, finally, the properties of perceptual representations to portray that information. Both the traditional and CSE-based requirements have similar goals, but a CSE focus leads to a requirements specification that provides more design guidance based on understanding of the work domain and of the perceptual and cognitive demands of the people in a JCS.

To provide a more detailed exploration of how CSE integrates with the SE process, we will consider the ACSE methodology throughout the paper. ACSE (Elm, Gualtieri, Potter, Tittle, & McKenna, 2008) is the fourth generation of a set of engineering practices that have evolved directly out of the ACWA methodology (Elm et al., 2003). The ACSE methodology addresses the analysis, design, and evaluation of a JCS. ACSE has strong influences on the requirements generation process, changing the way requirements are developed and written. Standard "the system shall . . ." style requirements are insufficient to develop a truly effective JCS. It is not adequate to develop general statements of the way a system is supposed to behave and allow the software engineer to determine the best way to make the behavior happen.

CSE approaches concept refinement in a completely different manner: It begins with the concept development process as a cognitive analysis, which "support[s] the development of revolutionary systems unconstrained by previous solutions" (Sanderson, Naikar, Lintern, & Goss, 1999, p. 319). Cognitive analyses are powerful because they abstract away the organizational partitioning of the work

domain to model the underlying goals, functional relationships, and cognitive work. The cognitive work analysis phase of ACSE, shown as $ACSE_{(A)}$ in Figure 3, uses a focused approach to generate the understanding of the work domain in order to develop the proper system requirements for the JCS.

Whereas many of the standard knowledge elicitation approaches are used to create the baseline domain knowledge, the information gathered from these techniques is then combined with situated observation in the work domain to reveal key abstract concepts. These abstractions, and why a user employs them in performing the decision-making task, help to define the functional goals fundamental to the work domain. These functional goals, and the relationships between them, are captured within a *functional abstraction network* (FAN). This lays the foundation for the information space from which the system design should be based. The differences between an ACSE/ACWA FAN and a functional abstraction hierarchy (FAH)/attribute abstraction hierarchy (AAH; Vicente, 1999) are attributable in part to ACSE's focus on being part of a larger SE process.

Once the network of functional goals of the domain is represented in the FAN, the next step in the ACSE methodology involves overlaying *cognitive work requirements* (CWRs) on the functional model as a way of identifying the cognitive demands, tasks, and decisions that arise in the domain and for which the operator requires support. The system requirements that follow from these CWRs must allow users to accomplish this cognitive work in the JCS. Identifying the functional relationships in the work domain and their associated cognitive work leads to sufficiently detailed requirements specifications to support the relevant operator decision making within the resulting system design. This application of CSE results in an integrated set of requirements—with dependencies, shared functions, and related cognitive work—that all influence one another.

The next step in ACSE is to identify the information that the decision-making agent within the JCS needs to complete the specified cognitive work (see Elm et al., 2008, for a more in-depth description of the ACSE analysis process). These information relationship requirements specify key relationships among collected data that must be preserved within the JCS, ultimately shaping the system concept. The system concept is more than just the interface between the operator and the technology; the key relationships specified in the CSE analysis should affect all aspects of the system (e.g., sensors, business rules, and physical placement of components).

The requirements definition process ends the concept refinement phase. At the end of this phase, CSE has provided a definition for the JCS and a hypothesis about what is required to support effective decision making for the human operators in the JCS. Within ACSE, the analytic artifacts produced by the process include a functional model of the work domain, a specification of the cognitive work needing support, and the information sources required to conduct that cognitive work. CSE plays a key role during the concept refinement phase, complementing traditional SE requirements that excel at reliability, availability, and other requirements with an entirely new class of requirements that makes the needs of the user an explicit part of the SE process.

Following the concept refinement phase is the technology development phase. This phase is meant to reduce technology risk (i.e., by understanding the role that technology plays in supporting various system functions) and determines the appropriate set of technologies to be integrated into the final system. Although CSE is not traditionally thought to have a large role during technology development, in fact, the selection of embedded, enabling technologies dramatically changes the nature of the "teaming" between the user and the technology. Adding technologies adds cognitive work associated with understanding and controlling it. CSE insights during the technology development phase can dramatically improve the final success of the JCS influencing the selection and/or the role of the technologies within the system architecture.

Integration Point 2: CSE's Impact on Software Development

The emphasis during the system development and demonstration phase is to ensure operational supportability (i.e., requirements that extend through a system's deployment and retirement). The goal is to develop a prototype of the system, reduce integration risk between system components (including the human operator), ensure system productivity, and implement human-systems integration. At this point, the integration of CSE into the SE process is vital. During this phase, the system engineer defines the system architecture and the system elements down to the configuration item level (i.e., the smallest unit of the system that will be changed independently of other components and that is under the control of the configuration management process), based on the technology selected during the technology development phase. Also during this phase, system design requirements are allocated down to the subsystem level and are refined as a result of developmental and operational tests and iterative SE analyses.

Within system development and demonstration, systems engineers define the system in terms of its physical components and software elements design synthesis (i.e., the process by which concepts are developed based on the functional descriptions that are capable of performing the required functions within the limits of the performance parameters prescribed). The design synthesis produces both physical and software architectures for the envisioned system. Systems engineers create the architectures by revisiting the functional requirements to verify that the synthesized system design can perform the required functions at required levels of performance. This iterative process permits the reconsideration of how the system will perform its mission, and this helps to optimize the system design.

From the CSE perspective, this design process needs to extend this design synthesis (Figure 4) to consider the entire JCS. During this phase, CSE takes the functional analyses that were done in the concept refinement phase and transforms them into a decision-centered design that will support the work that the human operator must perform. Thus, the CSE engineers "take control" of the design aspects that directly relate to the user and affect the eventual success of the JCS. By folding in this CSE-based process, the designer ensures that the operator is able

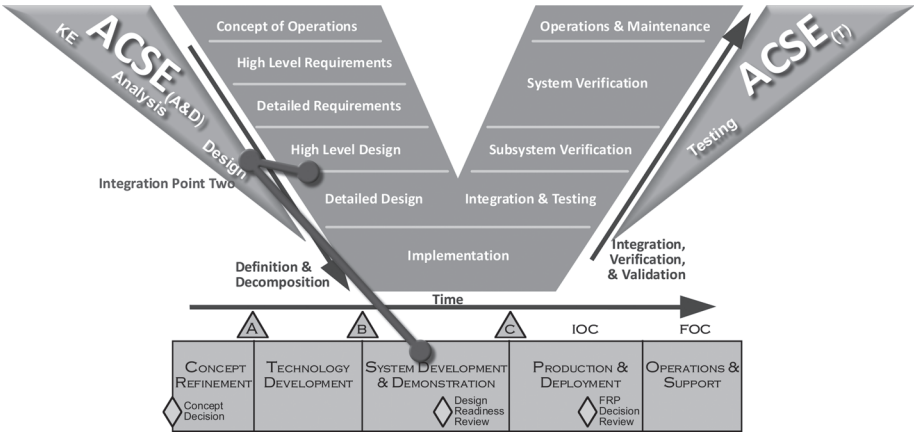

Figure 4. Cognitive systems engineering integration into the software development process of systems engineering. ACSE = applied cognitive systems engineering, KE = knowledge elicitation, IOC = initial operational capability, FOC = final operating capability, FRP = full rate production.

to complete work in the domain with the system that he or she is given. CSE's unique perspective allows for the consideration of the interaction of the decision-making humans, the available support technology, and the demands of the work domain in order to develop a more robust and reliable JCS.

Incorporating a CSE consideration for the decision-making human is critical for determining the interface between the human operator and the rest of the system (including other human decision makers). The system interface (or *presentation layer*) is not simply a device to exchange information between the human and technology; more important, it specifies how the work domain is conveyed to the humans in a JCS and constrains their actions in the world (McKenna et al., 2006). Accordingly, the interface design requirements for the system must capture the cognitive demands of the JCS effectively.

Traditional approaches to requirements analysis in the software engineering community often have treated the generation of these requirements as the specification of a software application programming interface (a message format used by an application program to communicate with the operating system or some other control program), merely defining the relevant data structures and the mechanisms for sharing those data across subsystems. As such, software engineers generally take the lead on coding the requirements into the system and designing the interface. However, leaving the design of a system to the software engineers can lead to problems (Cooper, 1999) with the degree of support provided to the users.

First, software engineers design the interface based on how they think about the problem, not how the actual end user thinks about the problem. In such cases, the resulting system is unlikely to reflect the experience of actual users (Goguen & Linde, 1993). Although the resulting design makes sense to the designer, users will

struggle to understand many of the concepts within the design. Second, when designing the system, software engineers fall back and rely on standard techniques for data portrayal, using nearly a one-size-fits-all design methodology. When code can be reused, Cooper (1999) argued, software engineers will reuse it, and because the resulting system preserves the stated requirements (i.e., assuming no CSE involvement, a set of requirements exists stating that "the required data are available somewhere in the system"), the system will be called a success. In most cases, this results in systems that are data-complete but do not consider how the representation affects human performance within the system.

From the CSE perspective, for information to be useful to support human understanding and decision making within the JCS, the designer must utilize human perceptual and cognitive capabilities (Elm et al., 2008). ACSE's *representation design requirements* (RDRs), as representation requirements derived from the CWRs, are objective specifications of the needed correspondence for the display, eliminating subjective and artistic arguments that may occur when the requirements underspecify the design process (Gualtieri, Szymczak, & Elm, 2006). These RDRs are the link between analysis and design, transforming knowledge elicited in the analysis phase into operational requirements for the working interface design. This step has proven to be the most difficult in the design of JCSs; it is where weak linkage exists between analysis artifacts and design artifacts. The design requirements specifically must link the display concept and the work that is being supported.

RDRs specify not only what is to be provided for the user but also how the system should be presented to the user (McKenna et al., 2007). This is an important difference, as it provides the system designer with clearer direction on the perceptual elements to appear in the completed design. When RDRs are properly specified, the designer has a more detailed and shareable definition of success. The set of RDRs guides the designer into the interface's final appearance and provides a more principled basis for determining when a design provides sufficient support to meet the requirements. This appears to be in contradiction with Brackett (1990), who argued that giving the designer more freedom is the proper way to design a system. In fact, there is no contradiction; a designer needs adequate freedom to explore novel solutions and at the same time enjoys the clarity of the guidance from the RDRs to both specify and stimulate the need for invention and creativity toward an effective JCS. Whereas design freedom can generate the proper design (by inspiration, chance, or attrition), specifying the design with a shareable artifact helps to ensure that all members of the SE development team understand the goals of the intended JCS.

Although a CSE analysis process such as $ACSE_{(A)}$ specifies the information relationships that must be portrayed within the interface, the cognitive systems engineer is not completely constrained regarding the interface design: The designer must determine the best way to present the required information to the user. When developing design requirements, one must make decisions about what frames of reference should be used to structure the presentation of the data, what events should be displayed, and what contrasts should be made salient. The CWRs

provide guidance on these decisions in order for the resulting system design to support the cognitive work.

The cognitive systems engineer must make data-encoding decisions when instantiating the RDRs as a presentation design. These decisions are influenced by the research on human perception and its interaction with various presentation techniques (Elm et al., 2008; Gualtieri et al., 2005). These designs form a kind of "requirement" that CSE contributes to the SE process. Typically, neither software engineers nor systems engineers have the necessary background in human cognitive and perceptual capabilities to fully understand how the requirements, as encoded in the interface, will affect the users' ability to make sense of the world. The cognitive systems engineer, as part of the multidisciplinary SE team, utilizing the lessons learned from years of studies in human perception and cognition when encoding requirements into an interface, can ensure that the abilities of the human user are taken into account during the development phase.

Although the focus of CSE is often on design during the system development phase, CSE also can affect software development beyond the representation within the presentation layer. These design decisions deeply influence the logical system architecture. In fact, CSE can influence the entire software development process, which, in addition to the presentation layer, includes the business, data, and input/output (I/O) layers (Figure 5).

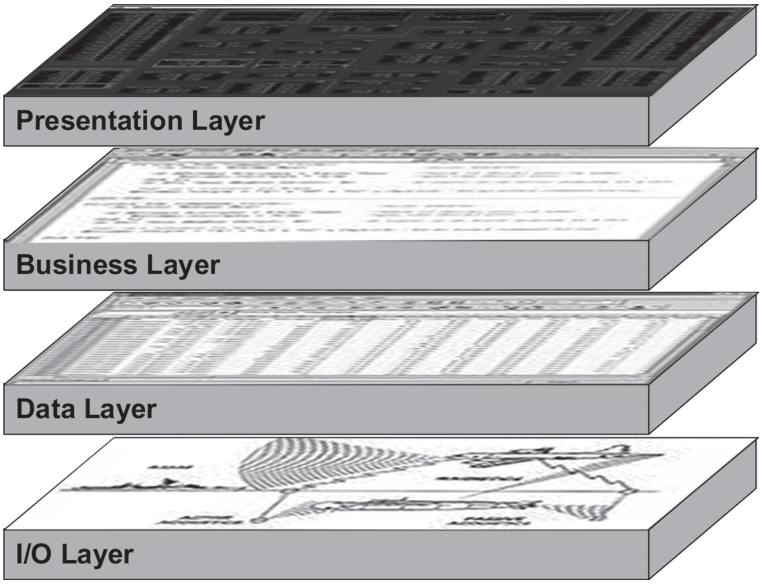

Figure 5. Applied cognitive systems engineering has a significant impact on every software layer, not just the encoding in the presentation layer. It affects what data are collected in the input/output (I/O) layer, how they are stored in the database layer, and the transformations that must occur in the business layer.

In classic SE practice, the business layer handles the exchange of information between the data layer and the presentation layer. The business layer performs all the transformations necessary to convert the available data into the form needed by the presentation layer. In a traditional SE process, data are usually presented in the same form in which they were stored: as data values. Users then are forced to convert data on their own in order to make decisions. Rather than directly address this mismatch, systems engineers typically have seen fit to add more to the business layer, using it as an automated agent that was nominally charged with replacing the human. Without CSE involvement, the workings of the business layer were hidden from the user, who must trust that what had happened behind the scenes were the proper actions to take and that the answer provided is correct and yet was ultimately charged with preventing the automation from committing an error (i.e., by manual override).

In not recognizing that system failures were attributable to poor design of the presentation layer, systems engineers swung the pendulum too far in the design of the business layer. Because users couldn't always properly convert pure data into the answer, the business layer was used to provide that answer, while giving no data to the user. This is a major problem with automation: Most systems engineers prefer to hide the details of the business layer from the users who are required to monitor and work with the automation. This becomes problematic when something goes wrong and the user is given no indication of what went wrong with the business layer, why it went wrong, or how to solve the problem.

From a CSE perspective, the proper business layer design is a middle ground between the two dichotomous ends of the spectrum (either present only data or fully automate the task). That middle ground is where powerful technologies are engaged when they excel, but in a manner that is fully observable and directable by the human (Hollnagel & Woods, 2005). To achieve this, cognitive systems engineers should use the functional analysis developed during analysis as a knowledge model (i.e., the FAN in ACSE) to inform the business layer how to convert the collected data into the representational forms described in the RDRs.

The *information relationship requirements* (IRRs) identify the proper context for the required data, turning it into information that the decision maker requires. Making these conversions in the business layer, and presenting this information to the user within the presentation layer, exempt the user from doing any data transformations in the head, greatly reducing the likelihood of error.

Additionally, the knowledge model created from the cognitive analysis can be used as the basis for any automation that is desired for the system (Bisantz et al., 2003). The analysis is created independent of who, or what agent, is performing the identified work. Its purpose is to identify the relevant work and the information that is needed to complete this work. When the required work is suited for automation, then automation should be built into the business layer.

However, adding automation also adds work for the human in the JCS to complete. The human must monitor the automation, initiate it, control its modes, and so forth, in order to ensure that it is running as expected. This requires that

the presentation layer support the user in understanding the work that is being done in the business layer (i.e., present the information necessary for decision making), so that the user is able to jump in and complete the work at a moment's notice should the automation be stretched beyond its capabilities. The presentation layer also must allow for the manipulation of the automated agent (i.e., provide feedback to its processes) in the business layer, so that users can control what it is doing rather than being controlled by it. With this CSE input, the business layer becomes more powerful than simply extracting data in the database and presenting it in the presentation layer: It presents the right information to the user in the right context at the right time.

Changing the way the business layer is developed affects how software engineers must develop the data layer. The way the data are stored in the database influences data access times in the business layer. The better laid out the architecture, the faster the data calls and, thus, the faster the program will run. Designing an effective data layer architecture, however, depends on having a good sense of how the user will move through the information space (and, via business layer transformations, the associated data space). Without this understanding, early decisions about what constitutes an effective business layer is not possible.

The I/O layer is similarly affected by the CSE influence. The IRRs from the CSE analysis, along with the associated transformation algorithms, lead the determination of what data must be acquired by the system. A give and take ensues between the information needs of the user (as specified by the IRRs) and what is technically possible in the I/O or business layers. IRRs become a requirement target to be met, in contrast to traditional SE processes, which approach the problem as "these are the available data."

The ACSE process identifies the data that are to be collected by the I/O layer in the domain to support good decision making (albeit in the form of needed information presentations for the user). ACSE identifies the information relationships that decision makers will need to perform cognitive work within the domain. These information needs are created independently of the current work practices and focus on user needs. Accordingly, the analysis is able to identify those data that are not currently part of the business practice and need to be collected. The data needs are not even limited to data that can be collected currently. If some data may be useful but currently no technology exists to do the collection, this may spur on the creation of the technology, especially if the data required are extremely valuable to the decision-making process.

Integration Point 3: CSE Insights for Testing

Providing design input is not the only way in which CSE contributes to the system development and demonstration phase. For each application of the SE process, the solution must be compared with the requirements. This part of the process is called *validation and verification*. Each requirement at each level of development must be validated and verified.

Validation is the process by which systems engineers confirm that the requirements allow for the system to perform a necessary action. Verification is the process by which systems engineers check the system to ensure that the requirements have been implemented. It is expected that this process will ensure both that the correct system is built and that it is built correctly. With a completely validated system, it is assumed that the proper requirements are in place. However, as stated earlier, from a CSE perspective the current requirements elicitation process is unable to ensure that good requirements have been developed. Adding CSE-based testing to verify decision support to the user expands traditional verification to include its highest-level requirements as a JCS.

With ACSE, validating requirements becomes a much different process. The cognitive systems engineer transforms the RDRs for the system into visual representations, which are early prototypes of the final system. As such, they serve as hypotheses about how the domain works (Hollnagel & Woods, 2005). Design reviews of these representations are used as validations of the CSE requirements. The designs are presented (and, to a lesser extent, exercised via storyboards, prototypes, etc.) to end users as a test of the design hypotheses. If the design does not sufficiently allow the user to accomplish the identified cognitive work, the requirements must be revised. If the JCS design is effective, then the system can be built, and verification can ensure that the system is built properly. This CSE validation has a twofold effect (see Figure 6). It makes sure that the requirements are sufficient to achieve system goals and that the requirements have been instantiated in a way that takes advantage of the unique capabilities of all elements of the JCS, leading to effective overall system performance.

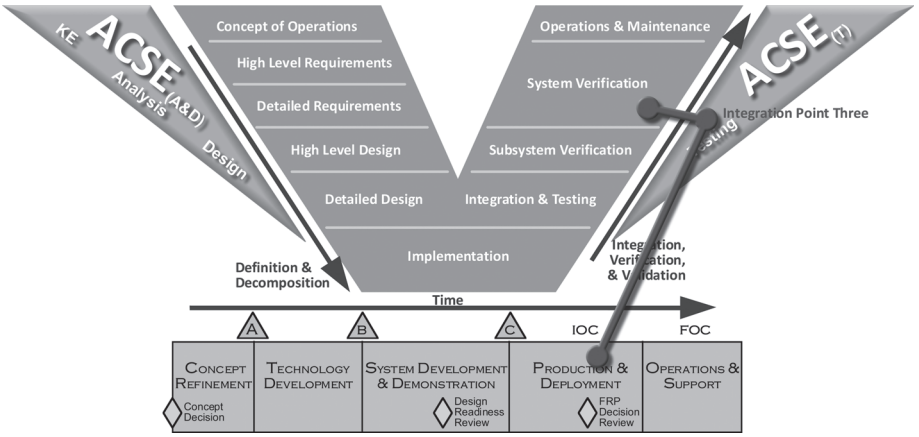

Figure 6. Extending the systems engineering process with verification of CSE-based requirements with edge-centered testing. ACSE = applied cognitive systems engineering, KE = knowledge elicitation, IOC = initial operational capability, FOC = final operating capability, FRP = full rate production.

Testing from a CSE perspective complements the current systems evaluation methods. Traditional SE tests consist of evaluating the usability of the system, evaluating the human performance within the system, and validating the software that was written. These tests must be complemented with a CSE-based evaluation of the net decision-making ability of the JCS in realistic, "decision-difficult" situations. Just as CSE is used in the creation of a system design, it also must be the foundation on which JCS designs are tested to ensure that cognitive work is being supported. The testing component of ACSE, $ACSE_{(T)}$, uses an approach called *edge-centered testing* (ECT). This CSE-based evaluation was developed to test the effectiveness of a JCS in any challenging work domain by reusing the CSE analysis as the basis to evaluate the decision-making effectiveness across identified decision-difficult regions in the JCS structure (Potter, Elm, & Tittle, 2007; Rousseau, Easter, Elm, & Potter, 2005).

ECT involves the explicit design and analysis of tests based on the key decision-making demands within the scope of the system. The result is an explicit test design describing the cognitive problem under test, the hypothesized "edge" or latent potential weaknesses in the JCS, and the events that need to be included in the scenario. In ECT, test scenarios are developed to specify a progressive evolution of events that would be expected to stress the defined edge. ECT has been remarkably effective at identifying JCS design weaknesses that were not evident in the results of more traditional SE-style testing. This decision-centered approach to testing has proven effective in discovering fundamentally new ways for evaluating the net decision-making effectiveness of the joint human-technology decision-making team. Ultimately, ECT is about evaluating the resilience of the JCS.

ECT becomes a mechanism for ensuring the JCS is designed in a manner that allows an operator to routinely make good decisions. Based on the analysis developed in ACSE, it pinpoints difficult decision areas and forces users to make these decisions under cognitive pressure. A well-designed display allows users to make good decisions even in novel, adverse, undersigned situations. This is different from human performance testing, which can state nothing about any of the particular aspects of the display or the JCS, only that the human did better in one system than in another. ECT specifically tests one aspect of the system, allowing conclusions to be drawn about how well the tested aspect supports the decision-making abilities of the JCS. Furthermore, by testing to identify the decision-making support provided by the JCS, the cognitive systems engineer can ensure that the functional analysis was executed properly.

By testing against the CSE-based requirements, one can attribute the success or failure of the visualization in supporting decision making to how closely the cognitive systems engineer approximated the domain in an analysis and how well the analysis artifacts were encoded into a display. If the overall system excels during the ECT, it will ensure a much more robust decision-making team and thus be better able to deal with the complex, unpredictable nature of the real world. ECT serves as an appropriate means to stress the components of the JCS to see if they will fail under pressure and verify and validate the integrated system against the

specified operational requirements within the required operational environment. ECT provides the necessary CSE complement to traditional SE testing to deliver test coverage of both the operation of the system and the satisfaction of its role as a JCS performing cognitive work in situ.

Integration Point 4: CSE Insights Beyond System Development

CSE is able to play a large role in the development of the system, generating the novel CSE-based requirements, achieving an effective representational design for the users, verifying and validating the requirements, and allowing decision-centered testing of the CSE requirements once the system is built. But it also can have a role that is larger than just the designed system. CSE provides a holistic design point of view that extends to such areas as role allocation, crew structures, training, procedure development, and operational business practices.

The purpose of the production and deployment phase is to achieve operational capability that satisfies mission needs. As the system becomes operational and is placed in the hands of users for actual use, CSE techniques such as in situ observation present a unique opportunity to discover issues not evident to the non-CSE-trained eye. Once detected, the traceability of the CSE-based requirements provides great benefits during the production and deployment phase, enabling the redesign of components that do not fully support the decision-making needs of the operators.

It is during this phase that the system attains its initial operational capability. It is also at this phase that training issues come to the forefront. Training has become a bit of a cure-all in the SE process, to the point that it can be viewed as an indicator of a latent system design issue for a CSE-trained observer. Systems engineers attempt to correct JCS performance issues by providing additional training, procedures, policies, and so forth. Whenever an accident occurs, one of the first corrective measures to make sure a similar accident does not happen again is to add incident-specific training to the training program. Correcting the JCS design constitutes a more enduring solution than requiring the user to recall additional training during a stressful situation.

The final phase of the system life cycle is the operations and support phase. The objective of this phase is to sustain the system over its total deployed life cycle. The role of CSE during this phase is to continue to observe and test to detect JCS issues and incrementally execute the CSE methodology to facilitate improvements to the design of the JCS.

Conclusion

SE has continually enhanced its processes without looking to CSE for additional insights. (This is attributable, at least in part, to the lack of pragmatic CSE practices suitable for adoption by the SE processes.) Successful systems design is possible only when the human is considered an integral component of the overall

system. The question for those in the CSE community is, Where should we influence the SE process?

The short answer is *everywhere.* However, there are key periods in the SE process when CSE input is particularly valuable. Based on the points made earlier in this paper, a prescriptive effort profile is proposed (see Figure 7).

By augmenting traditional SE teams and processes with the indicated CSE contributions, one adds a JCS perspective to the SE process, resulting in more supportive system designs. This improved support evidences itself in more robust decision-making performance under stressful conditions, as indicated by reduced error rates and, more important, faster error detection and recovery, particularly in novel, unanticipated situations. ACSE, as a specific methodology of CSE, is a structured methodology that specifies the structure of the analysis, the way to transform this analysis into a design, and a means for testing the resulting system.

CSE integrates naturally in the initial stage of the SE processes (i.e., concept refinement), when efforts focus on developing a vision for the system under development in the form of concept documents and the various levels of requirements. By explicitly exposing the cognitive work to be supported, CSE contributes a critical perspective on "what the right system is," which has remained a vexing problem for the SE community. By incorporating this decision-centered perspective into an already robust requirements management portion of the SE process, one combines

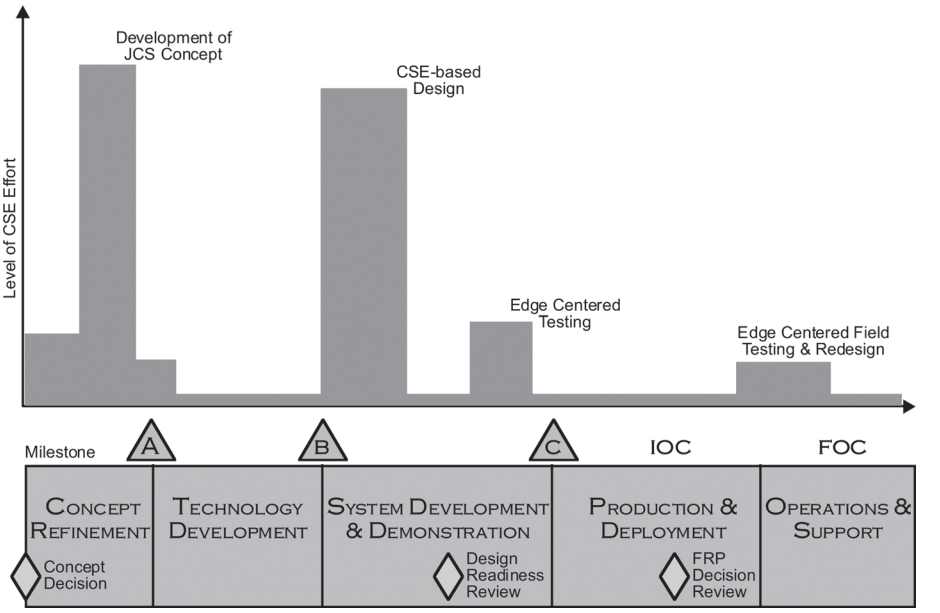

Figure 7. Relative level of cognitive systems engineering (CSE) effort in the Department of Defense system acquisition process required for effective joint cognitive system (JCS) development. IOC = initial operational capability, FOC = final operating capability, FRP = full rate production.

the best of both worlds, finally allowing not only "building the system right" (process focus) but also "building the right system" (effectiveness focus).

The addition of CSE at the concept development phase is the least contentious point of integration because the SE community continues to look for new techniques that address fundamental user issues (e.g., Rapid Application Development (RAD)/Joint Application Development (JAD), use case, Jacobson extreme programming). The highly visible CSE-based graphical user interface representational concepts appeal to advanced, "early adopter" users. CSE provides a principle-driven approach to the development of operational concepts (e.g., coordination of directed-attention subsystems with both functional and physical display systems) that contain a level of design that appeals to systems engineers.

During the technology development phase, when high-level or detailed systems requirements are generated, the SE community has been less accepting of CSE input. Traditional SE processes treat requirements as fundamental "shall"/"will" language statements that are expanded or decomposed during the initial phases of a program into more detailed statements. CSE-developed requirements (as instantiated by the particular requirements developed by the ACSE methodology) actually contain representational requirements and performance requirements that are normally thought of as "design" artifacts. For example, an ACSE-generated software requirement specification (SRS) is expanded beyond the standard IEEE SRS to explicitly include designed displays, needed data transformations, and look and feel descriptions. The proven advantages to the process of considering these design insights as system requirements have included (a) directly affecting early system architecture and technology design decisions while they are still malleable and (b) improving engineering work budget/schedule development estimates because specific capability requirements for presentation and interaction replace the vague "the system shall display all necessary data" type requirements.

During the system development and deployment phases, the CSE role is fundamentally one of supporting the handoff to engineers constructing the system. Our experience with ACSE has proven that even our best CSE design specification and its companions, the expanded system and software requirement specifications (SyRS and SRS), never fully communicate all the nuances of the design to the developers, and face-to-face interaction is essential. This is because the CSE derived specifications are focused on the human component of the system and lack some of the technical detail that is typically found in an SyRS and SRS. Further, engineering constraints, budget and schedule pressures, and so forth require a very closely coupled, highly iterative cycle between the cognitive systems engineer and the developers in order to negotiate design and implementation compromises and alternatives. Ultimately, the idealized initial design has to be resolved against an affordable, practical implementation as an operational system. That resolution requires both a cognitive systems engineer and a developer as part of that negotiation.

As the system development phase produces operational system components, CSE reengages in the SE process. As system components complete unit and system integration testing and are verified against the traditional SE requirements, they

provide the operational capability to allow a novel CSE-based form of system testing. Just as the initial phases emphasized the decision-centered analysis of the larger JCS (i.e., including the human within the scope of system analysis), CSE-based testing explicitly tests the combined performance of the user (or users) and the developed technology system as an integrated JCS. It is our experience with CSE-based ECT that it can detect latent decision support brittleness, even after successful system testing and even extensive time in deployed operational use. These ECT tests can detect this unique class of decision support weakness, which traditional system testing overlooks, thus eliminating these flaws before they result in an operational failure or disaster.

When a human user is involved in the operation of a system, as is the case in any JCS, the cognitive and perceptual capabilities of that user must be taken into consideration. To this extent, CSE does not compete with SE but, rather, augments and improves the SE process by ensuring that the users' cognitive needs are satisfied. Integrating these CSE principles throughout the SE process provides the necessary approach for developing truly user-centered systems, moving beyond merely user acceptance to a whole new generation of resilient, supportive decision support technologies working seamlessly with the user to form a joint cognitive system.

References

Andrews, D. C. (1991). JAD: A crucial dimension for rapid applications development. *Journal of Systems Management*, *42*(3), 23–31.

Ashby, E. (1958). *Technology and the academics: An essay on universities and the scientific revolution.* London: Macmillan.

Beck, K. (2000). *Extreme programming explained: Embrace change.* New York: Addison-Wesley.

Bisantz, A. M., Roth, E. M., Brickman, B., Lin, L., Hettinger, L., & McKinney, J. (2003). Integrating cognitive analyses in a large scale system design process. *International Journal of Human-Computer Studies, 58,* 177–206.

Boehm, B., & Lane, J. (2007, October). Using the incremental commitment model to integrate system acquisition, systems engineering, and software engineering. *CrossTalk—The Journal of Defense Software Engineering,* 4–9.

Brackett, J. W. (1990). *Software requirements* (SEI-CM-19-1.2). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

Brooks, F. P. (1987). No silver bullet: Essence and accidents of software engineering. *Computer, 20*(4), 10–19.

Cockburn, A. (1994). In search of methodology. *Object Magazine*, *4*(4), 52–56.

Cooper, A. (1999). *The inmates are running the asylum: Why high tech products drive us crazy and how to restore the sanity.* Indianapolis, IN: SAMS Press.

Defense Acquisition University Press. (2001). *Systems engineering fundamentals.* Fort Belvoir, VA: Author.

Defense Science Board. (2007). *21st century strategic technology vectors: Accelerating the transition of technologies into U.S. capabilities* (2006 Summer Study). Washington, DC: Author.

Elm, W. C., Gualtieri, J., Potter, S., Tittle, J., & McKenna, B. (2008). Pragmatic use of CWA in system design: Extending current thinking by adapting the mapping principle. In C. M. Burns & A. M. Bisantz (Eds.), *Advances in cognitive work analysis* (pp. 247–271). Boca Raton, FL: CRC Press.

Elm, W. C., Potter, S. S., Gualtieri, J. W., Roth, E. M., & Easter, J. R. (2003). Applied cognitive work analysis: A pragmatic methodology for designing revolutionary cognitive affordances. In E. Hollnagel (Ed.), *Handbook for cognitive task design* (pp. 375–382). London: Erlbaum.

Ericsson K. A., & Simon H. A. (1993). *Protocol analysis: Verbal reports as data*. Cambridge, MA: MIT Press.

Ernst, N. A., Jamieson, G. A., & Mylopoulos, J. (2006, April). *Integrating requirements engineering and cognitive work analysis: A case study.* Presented at the Fourth Conference on Systems Engineering Research, Los Angeles.

Feather, M. S., Fickas, S., van Lamsweerde, A., & Ponsard, C. (1998). Reconciling system requirements and runtime behavior. In *Proceedings of the International Workshop on Software Specification and Design* (p. 50). Washington, DC: IEEE Press.

Forsberg, K., & Mooz, H. (1992). The relationship of systems engineering to the project cycle. *Engineering Management Journal, 4*(3), 36–43.

Goguen, J., & Linde, C. (1993). Techniques of requirements elicitation. In *First IEEE International Symposium on Requirements Engineering (RE'93)* (pp. 152–164). Washington, DC: IEEE Press.

Gualtieri, J. W., Szymczak, S., & Elm, W. C. (2005). Cognitive systems engineering-based design: Alchemy or engineering. In *Proceedings of the Human Factors and Ergonomics Society 49th Annual Meeting* (pp. 254–258). Santa Monica, CA: Human Factors and Ergonomics Society.

Heinze, C., Papasimeon, M., & Goss, S. (2000). Specifying agent behaviour with use cases. In C. Zhang & V. W. Soo (Eds.), *Design and applications of intelligent agents: Third Pacific Rim International Workshop on Multi-Agents, PRIMA 2000. Proceedings* (pp. 128–142). Heidelberg, Germany: Springer.

Hollnagel, E., & Woods, D. D. (1983). Cognitive systems engineering: New wine in new bottles. *International Journal of Man-Machine Studies, 18,* 583–600.

Hollnagel, E., & Woods, D. D. (2005). *Joint cognitive systems: Foundations of cognitive systems engineering.* Boca Raton, FL: CRC Press.

Institute of Electrical and Electronics Engineers. (2000). *Recommended practice for architectural description of software-intensive systems* (Std-1471-2000). New York: Author.

International Council on Systems Engineering. (2004). *Systems engineering handbook* (Version 2a). Seattle, WA: Author.

Jacobson, I. (1992). *Object-oriented software engineering.* New York: Addison.

Layton, C., Smith, P. J., & McCoy, E. (1994). Design of a cooperative problem-solving system for en route flight planning: An empirical evaluation. *Human Factors, 36*, 94–119.

McKenna, B., Gualtieri, J. G., & Elm, W. C. (2006, July). *Joint cognitive systems: Considering the user and technology as one system.* Presented at the 16th Annual International Symposium of INCOSE, Orlando, FL.

McKenna, B., Gualtieri, J. G., & Elm, W. C. (2007, June). *Expanding functional analysis to develop requirements for the design of the human computer interface.* Presented at the 17th Annual International Symposium of INCOSE, San Diego, CA.

National Aeronautics and Space Administration. (1995). *NASA systems engineering handbook* (SP-610S). Washington, DC: Author.

Pew, R., & Mavor, A. (2007). *Human-system integration in the system development process: A new look.* Washington, DC: National Academies Press.

Potter, S. S., Elm, W. C., & Tittle, J. (2007, June). *Evaluating the resilience of a human-computer decision-making team: A methodology for decision-centered testing.* Poster session presented at NDM 8–Eighth International Conference on Naturalistic Decision Making, Pacific Grove, CA.

Rasmussen, J. (1986). *Information processing and human-machine interaction: An approach to cognitive engineering.* New York: North Holland.

Rasmussen, J., Pejtersen, A., & Goodstein, L. (1994). *Cognitive systems engineering.* New York: Wiley.

Rechtin, E. (2000). *Systems architecting of organizations.* Boca Raton, FL: CRC Press.
Rousseau, R., Easter, J., Elm, W., & Potter, S. (2005). Decision centered testing (DCT): Evaluating joint human-computer cognitive work. In *Proceedings of the Human Factors and Ergonomics Society 49th Annual Meeting* (pp. 322–326). Santa Monica, CA: Human Factors and Ergonomics Society.
Sanderson, P., Naikar, N., Lintern, G., & Goss, S. (1999). Use of cognitive work analysis across the system life cycle: From requirements to decommissioning. In *Proceedings of the Human Factors and Ergonomics Society 43rd Annual Meeting* (pp. 318–322). Santa Monica, CA: Human Factors and Ergonomics Society.
Turk, W. (2006). Writing requirements for engineers [good requirement writing]. *Engineering Management Journal, 16*(3), 20–23.
Vicente, K. J. (1999). *Cognitive work analysis: Toward safe, productive, and healthy computer based work.* Mahwah, NJ: Erlbaum.
Woods, D. D., & Hollnagel, E. (1987). Mapping cognitive demands in complex problem-solving worlds. *International Journal of Man-Machine Studies, 26,* 257–275.

William C. Elm, M.S. (electrical engineering, Carnegie Mellon University, 1978), is one of the most experienced practitioners of cognitive systems engineering, with over 25 years of experience, dating back to its inception with David Woods in the early 1980s. He recently retired as a military intelligence officer with almost 25 years of active and reserve service from tactical through joint intelligence levels. Elm is currently the cognitive systems engineering fellow, founder, and president of Resilient Cognitive Solutions, LLC.

James W. Gualtieri, Ph.D. (applied cognitive psychology, George Mason University, 1994), is a principal cognitive systems engineer with over 15 years' experience developing innovative decision support tools for business and mission-critical domains. He has conducted research in the areas of decision making, mental models, team information processes, and system utility. He has helped to design and develop 10 fielded systems and has written more that 50 publications and technical reports.

Brian P. McKenna, M.S. (human factors psychology, Wright State University, 2005), has been a cognitive systems engineer since 2004. Brian currently works with Resilient Cognitive Solutions, for which he has been a key member on several projects, utilizing his training in cognitive systems engineering to complete domain analyses, system designs, and system testing. His background and training focused on the coupling between perception and action.

James S. Tittle, Ph.D. (experimental psychology, University of California, Irvine, 1991) is a principal cognitive systems engineer with Resilient Cognitive Solutions. He has over 15 years' experience researching human cognitive and perceptual capabilities, including visual perception of spatial structure from multiple sources of information. For the past 6 years he has been applying cognitive systems engineering principles to the analysis and design of joint cognitive systems for complex work domains.

Jay E. Peffer, B.S. (information sciences and technology, Pennsylvania State University, 2003), is a cognitive systems engineer with Resilient Cognitive Solutions. Jay specializes in analyzing work domains from a functional perspective, making the fundamental decision support needs explicit. Using this analytic structure, he can define visualization design requirements to design decision support systems, utilizing factors of human perception, attention direction, and salience management. A recent focus has been evaluating and designing for coordination from a cognitive systems engineering perspective.

Samantha S. Szymczak, B.S. (systems engineering, University of Illinois at Urbana-Champaign, 2004), is a cognitive systems engineer with Resilient Cognitive Solutions. She studied systems engineering at the University of Illinois at Urbana-Champaign. She specializes in using cognitive work analyses as artifacts for designing decision support systems for complex work domains and inventing novel representations for visualizing information. Her recent work includes applying cognitive engineering principles for above-water warfare, various command and control domains, intelligence analysis, researcher collaboration, and enterprise/business awareness.

Justin B. Grossman, M.S. (cognitive systems engineering, Ohio State University, 2006), is a cognitive systems engineer with Resilient Cognitive Solutions. He has conducted research in complex work domains for more than 7 years, including information analysis, command and control, and robotic-assisted search and rescue. He has also helped design numerous decision support systems for these domains. For the past few years he has been embedded in a transformational element of a large agency, applying cognitive systems engineering principles to its mission, tools, and processes.