Proceedings of the 2008 Industrial Engineering Research Conference
J. Fowler and S. Mason, eds.

Four Challenges, and a Proposed Solution, for Cognitive System Engineering – System Development Integration

Christopher R. Hale
Science Applications International Corporation
Dayton, OH 45431, USA

Vincent Schmidt
Air Force Research Laboratory
Wright Patterson Air Force Base
Dayton, OH 45433, USA

Abstract

Productively integrating Cognitive System Engineering (CSE) into system design and development processes depends on successfully addressing four challenges: determining which content produced by the CSE community adds specific value to system development, identifying CSE artifacts that communicate with other system development participants/stakeholders, selecting appropriate risk assessment tools to enable system developers to conduct relevant tradeoffs at each stage of design/development, and providing traceability to CSE cognitive/work requirements and first principles. We describe a philosophy and subset of tools for integrating CSE and system development by addressing each of these challenges.

Keywords

Cognitive systems engineering, CSE-SE integration, work analysis

1. Introduction

Integrating Cognitive System Engineering (CSE) with System Engineering (SE) and development methodologies is receiving renewed and widespread attention: systems are becoming more complex as networking technologies enable a functional linking of previously separate components into composite systems-of-systems. System performance, while still crucial, is no longer the sole criterion of acceptability. In addition, an emphasis on system effectiveness is now equally considered in assessing a system's desirability and acceptability. The teleological roles of technologies in complex activities (e.g., effects-based operations) have resulted in concerns about the purposive nature of designed artifacts. For years the CSE community has emphasized the need to consider these factors in system design [1-3]. As these two communities pursue paths to productive integration it has become increasingly clear that several challenges must be overcome.

Philosophically, the CSE practitioner's role on the overall development team is often in question. The CSE-as-lead viewpoint places the CSE practitioner in the lead role. In this model, early cognitive analysis and subsequent visualization/interface designs comprise the central activities of development. The SE-as-lead model reverses the relationship, placing the SE in a lead role with CSE providing support. The contributing team model places both CSE and SE in contributing roles reporting to, and directed by, a program manager who might have minimal technical expertise in either area. The model chosen for a particular development project will affect system evolution due to cultural factors arising out of the backgrounds of those placed in leadership positions. Inevitably, the two disciplines envision and evaluate design problems differently, emphasize different aspects of design, and adopt different criteria for success.

Technical challenges arise as well, specifically in the areas of content, communication, risk management and traceability. It is axiomatic in the human factors community that "we're not included in the design early enough to make a positive difference." Unfortunately, this community often produces data that are not useful to the needs of the evolving design. The content challenge is compounded by a challenge of communication, that is, the two communities speak different languages. Risk management represents a third challenge. While methods for risk management exist in the SE community, those used by the CSE community often are subjective and unreliable. The

fourth challenge for CSE-SE integration is traceability. This is the ability to "reverse engineer" an artifact, tracing it backward through the development process to its originating requirements. We focus on these four challenges in the next section of this discussion. The final section discusses a methodology designed to address these challenges.

2. Four Challenges for CSE-SE Integration

2.1 Producing Substantial and Actionable Content

A scan of the cognitive engineering and human factors literature reveals vast information addressing almost every aspect of human interaction with systems. Why does this information so seldom influence system design? Our position is that this information (1) often is produced in the service of theory testing rather than design, (2) is articulated at a level of fidelity that is of little use to the mission-oriented considerations of system design, (3) is not actionable, that is, does not provide the "content" needed to derive design commitments, (4) does not support the tradeoffs required during the course of design, and (5) does not support designers in understanding how adaptive behavior at one level influences or constrains behavior at other levels across total system operation.

Useful cognitive information should include content that is normative and mission-oriented. Rasmussen [3] and others [4-6] have provided a framework that helps cognitive engineers define design-relevant information that makes clear the constraint relationships among design levels. Information is organized into several levels of increasing abstraction designed to capture the problem space (or decision space) faced by the system being designed. These levels are typically referred to (moving from more to less abstract) as (1) functional purpose, (2) abstract function, (3) generalized function, (4) physical functions and (5) physical form. Functional purposes refer to the goals of the system and external constraints on its operation. Abstract functions capture the criteria the system uses for measuring its progress toward its functional purposes. Generalized functions encompass the processes required to achieve the system's functional purpose. Physical functions embody the capabilities and limitations of the physical objects in the system that enables the processes at the next higher level of abstraction. Physical forms articulate the physical artifacts of the system by which physical functions are realized. All information produced by a CSE analysis will reside at one of these levels of abstraction. Traditional human factors engineering (HFE) concentrated primarily on the lowest two, and occasionally on the middle, levels of this abstraction hierarchy. However, because the purpose of the system, as well as the ecology within which it must operate, is articulated in the upper three levels of the hierarchy, human factors analysis remained disconnected from these important teleological aspects of the system design. Cognitive system engineering corrects this deficit by providing the needed ecological, purposive information.

The cognitive systems approach begins, therefore, with a consideration of work environment ecology: Its properties and constraints as well as the system's "purpose space" within the work environment. This creates the work context within which all system activity is assumed to occur. When this has been specified, cognitive engineers will move on to more action-oriented analysis of activities taking place within the system. These include the control tasks in which the operators must engage; strategies that operators devise to improve their effectiveness and efficiency; socio-organizational factors affecting the definition, selection and execution of control tasks and strategies; and the competencies and limitations that operators bring to the work requirements. We emphasize again that all of these latter processes exist within the context of the work environment structure, opportunities, and constraints. The information collected by a thorough CSE process is shown at the left side of Figure 1.

The middle and right portions of Figure 1 summarize four broad categories of information typically collected, and design artifacts produced, by system engineers early in development. First is information capturing scenarios within which the system must operate, to include a consideration of system boundaries. The second category describes the constraints affecting system operations: external constraints as well as constraints arising from system structure and purpose. The third category includes functional and performance requirements. Finally, measures that the design team will use to assess the system will be described. These will include both effectiveness criteria and measures of technical performance. A comparison of the left and middle portions of Figure 1 shows much similarity in the content of CSE and SE analyses. A concern for the operational ecology of the work environment is present in both analyses, as is the identification of constraints in the operating environment, and the role of these constraints in affecting system dynamics.

2.2 Communicating CSE Substance to the SE Process

Communication between cognitive engineers and system developers takes two forms. One is comprised of various artifacts the CSE community produces from the analyses shown in Figure 1 [5][7][8]. These range from text descriptions of work environment ecology to specifications of human performance parameters. However, these artifacts often do not map unambiguously onto the needs of system development. For example, a CSE analysis that produces a set of concept maps might include information about work organization and constraints, or cognitive task analytic information about processes and information requirements. While the analysis is of value to the evolution of the system design, the artifact conveying the information is neither familiar to nor efficacious to system engineers. At the point in development that concept maps would be offered, the development team's focus is on requirements. Concept maps, while they suggest requirements, do not articulate them explicitly. Similar criticisms can be made of abstraction-decomposition networks, critical decision matrices and traditional task analysis.

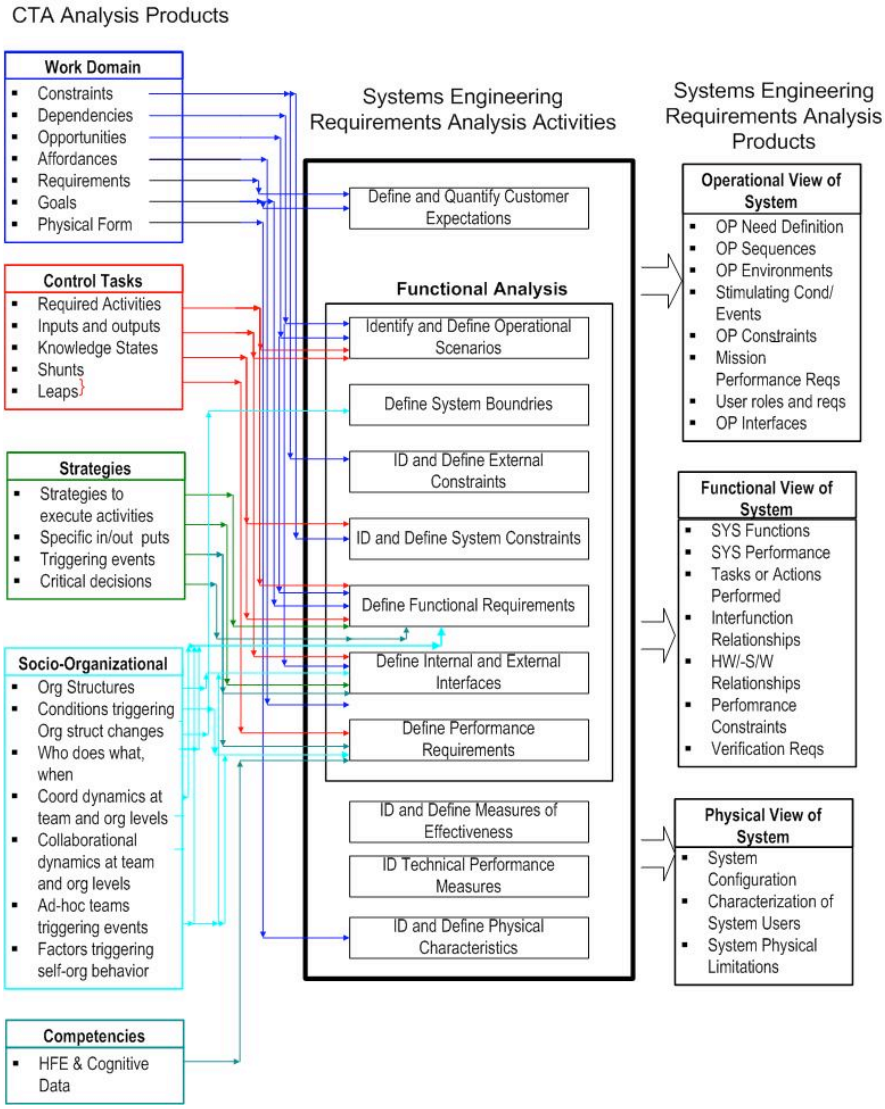

Figure 1: Conceptual Integration of CSE and SE Products in Early Design Stage

The second avenue of communication is comprised of the informal working relationships between cognitive engineers and system developers. Cognitive engineers often communicate empirical results centered on individual humans performing artificial tasks in artificial work environments, using measures unique to human performance that are unrelated to the needs of system development. Furthermore, the two disciplines speak different languages. The CSE community might use the language of theoretically motivated empirical results obtained under highly controlled experimental conditions. Alternatively, they might use high-level, often ill-defined concepts such as

"sensemaking." Our position is that there must be more rigor in the artifacts and languages of communication, not new terms.

2.3 Assessing Risk and Managing Tradeoffs

The National Research Council report on human-system integration [9] identified three categories of risk: technical, cost and schedule. In considering course-grain program risk management this study group presented a generic methodology consisting of risk identification, analysis, option evaluation, and mitigation. Tracking and communication were presented as activities taking place at each of these stages. One could argue that a major contribution of CSE is in mitigating risks early in the development cycle by providing the kinds of information needed to avoid incomplete or inaccurate requirements. In fact, this is a strength of the CSE contribution. However, CSE provides no method to compute the likelihood of such a risk or the nature of detrimental consequences if the risk is realized. Similar criticisms obtain for CSE contributions to other development stages.

Mature risk analysis techniques (e.g., Technique for Human Error Rate Prediction (THERP), Failure Modes, Effects and Criticality Analysis (FMECA), and Fault Tree Analysis (FTA)) often must be used in conjunction with a general theory of error (e.g., [10]). Causes of potential errors can vary greatly (poor user interface design, poor system architecture, inadequate training, selection issues, individual differences, etc.). None of these techniques help developers attribute errors to their potential developmental causes. Each technique relies on group consensus in building the analytical structure and assigning probabilities to elements of this structure, but it is often hard to achieve this group consensus. Furthermore, estimating likelihoods tends to be a very unreliable process. Finally, these methods rely on a degree of specification often available only late in the development process, after design commitments have been made. Ideally, one would like to make "free" design commitments early and then apply risk analysis so that problems then could be corrected cheaply and easily.

2.4 Traceability

The challenge of traceability is best illustrated by example. In the development of a visualization support system for operational assessment, a set of histogram-like visualizations were developed. Each individual histogram consisted of a color gradient ranging from red in the lower-left corner, through a yellow band, to green in the upper-right area. The challenge of traceability is contained in the questions about this visualization that were posed by the customer of this effort: (1) "What is the purpose of the color gradient?" (2) "Why do the symbols in each histogram take on the shapes and colors that they do?" (3) "Why does the visualization behave the ways it does over time?" Ideally, the member of the design team representing the CSE domain would base answers to these questions on clear relationships between the designed visualizations and the originating requirements developed early in the process. More typically, however, the answers are some variant of the "design is an art" position. The challenge, in our view, lies in defining clear relationships between the outcomes of the development process (the designed cognitive artifacts) and the process itself. The goal is to develop a method that delineates the steps taken from the designed artifact, back through the stages of the development process, to the originating requirements.

3. Fragments of a Solution to the CSE – SE Disconnect

Our approach addresses the four challenges discussed in the previous section through a staged methodology extending across the entire system development cycle. We begin with an analysis focused on the five areas shown at the left side of Figure 1. First, a work domain analysis is conducted to gather information about the design domain's goals and requirements. Second, control task analysis identifies required activities, tasks and workflow relationships. Third, we identify strategies used to execute activities and tasks, focusing on critical cues and other triggering conditions for each task, critical decisions, common errors, communication patterns, tools used to carry out the work, and data products used during task accomplishment. Fourth, we identify the socio-organizational factors present in the domain, to include organizational structures, conditions that trigger changes to these structures (e.g., conditions creating ad hoc teams), and coordination/collaboration dynamics. Fifth, we identify perceptual and cognitive competencies used to manage work, achieve goals, and collaborate with teammates. We collect this information into a set of structured concept maps containing the information shown in the template of Figure 2. This constitutes a work ontology in the physical, psychological and socio-organizational domains.

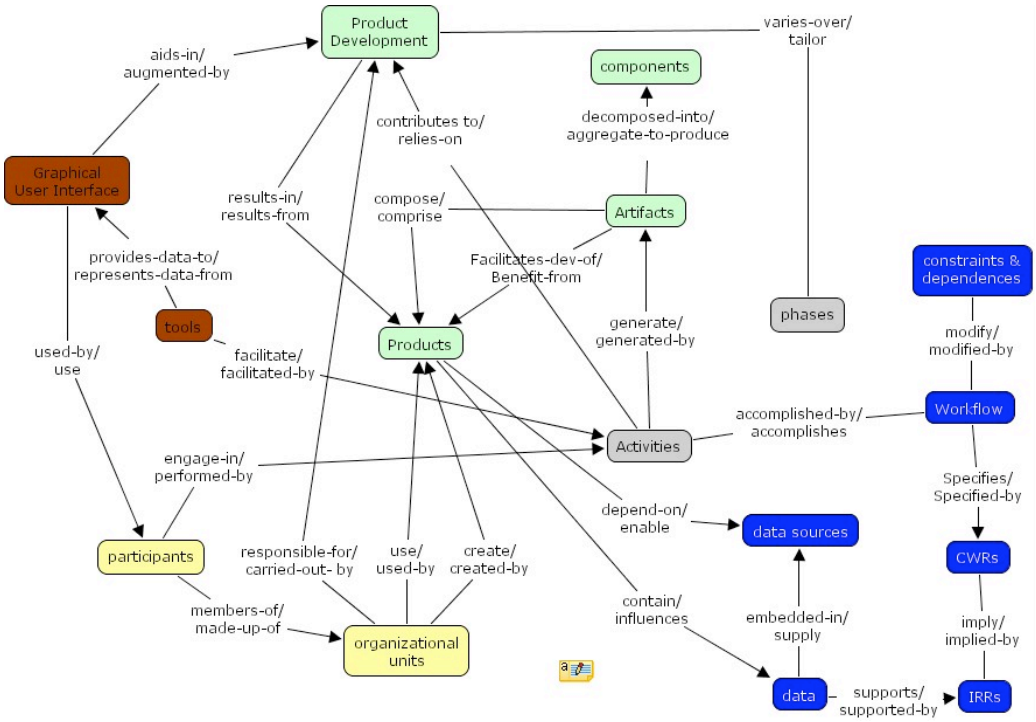

Figure 2: Ontology for Cognitive Work Analysis

The cognitive systems engineering analysis is then mapped to the 11 system engineering requirements analyses and products shown in the center and right portions of Figure 1. Cognitive and traditional system engineers participate in this mapping exercise, explicitly identifying where each result of the CSE analysis will be placed within the system engineering functional analysis. This often requires some re-casting of terminology and level of specification contained in the CSE analysis, and helps to translate the CSE analysis into a form appropriate for system requirements. Most importantly, performing this mapping ensures that information gathered during the CSE analysis is included in the system engineering requirements analysis products that will be used to guide subsequent system development.

All the information from the previous analyses is used to develop a set of system engineering models that can be captured in a computer-aided software engineering (CASE) tool. Working with subject matter experts (SME) we develop a set of hierarchically organized diagrams of the work domain that includes information about system functionality, inputs and outputs, sequence, constraints and dependencies, triggering conditions and end-states. These diagrams form the basis of requirements definition, and system requirements can often be generated automatically by the CASE system. These system requirements then are augmented with requirements identified by mapping the automatically generated set against the concept maps and other artifacts of the CSE analysis. Each requirement is labeled with a reference to its source and color-coded to indicate whether it is a system-only requirement or a requirement involving user participation with the system under design.

We can be confident at this point that the information gathered during CSE analysis is represented in the requirements that will guide system design. What remains is the crucial step of explicitly relating the requirements to the design of the artifact. We accomplish this in two steps. First, we map each requirement to the perceptual and cognitive work elements needed to satisfy it. We have discovered, across many projects, that a relatively small set of cognitive work elements are adequate to satisfy most requirements that define system development efforts. After verifying that this set covers the work analyzed in the work domain analysis, we carry out the mapping of requirements to cognitive work elements, capturing these in spreadsheet format. Once each requirement has been mapped to cognitive work elements, we create an executable model of each requirement. Imagine the following requirement for a visualization system designed to support effects-based assessment (EBA):

The system shall provide a way to derive intended and unintended effects from tactical assessment results.

Work elements needed to satisfy this requirement include: *inference, acquisition, classification, comparison and recognition*. The requirement's executable model will consist of submodels for each work element organized into a task network model. With executable models in hand, cognitive engineers can begin developing visualization concepts. Each work element has basic visualization requirements defined for it that are consistent across contexts. For example, *compare* involves examining two or more objects in terms of their similarities and differences. The visualization requirements for this element include displaying the attributes and values of objects being compared to users. Further, these "objects" should be displayed in a way that facilitates the comparison being carried out, that is, by highlighting the similarities and differences through some method of ranking, coding or some other means. Thus, there will be a basic structure associated with each element as well as performance parameters for the elements. Parameters for *compare* might include the number of to-be-compared elements that can be held in short-term memory and sensitivity limitations on attribute similarity. The requirements provide the context, constraints and boundaries for visualization design for each element. Thus, while the basic requirements will not change across elements the values of parameters associated with modeling of elements will change according to the context of each requirement.

4. Conclusions

We believe that the methodology described here provides a way to productively integrate the philosophical orientation, activities and products of cognitive system engineering into the larger development enterprise. While there are a number of CSE methodologies currently in use, most of these remain separate from the development process as defined by university programs, professional organizations and system development businesses. In part, the disconnect is due to philosophical differences. More substantially, however, we believe that a failure to adequately address the challenges outlined above separates the two communities. We believe the methodology presented here will begin bridging this gulf.

References

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

2. Rasmussen, J. (1983). Skill, rules and knowledge: Signals, signs and symbols and other distinctions in human performance models. *IEEE Transactions on Systems, Man and Cybernetics, SMC-13 (3)*, 257-267.

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

4. Flach, J.M., Schwartz, D., Bennett, A. Behymer, K. and Shebilske, W. (2007). Synthetic task environments: Measuring macrocognition. In Macrocognitive Metrics Conference Proceedings, Columbus, OH: Ohio State University, June, 2007.

5. Naikar, N. Hopcroft, R. and Moylan, A. (2005). Work domain analysis: Theoretical concepts and methodology. DSTO-TR-1665, Air Operations Division, DSTO Defense Science and Technology Organization.

6. Vicente, K. J. (1999). Cognitive work analysis: Toward safe, productive and healthy computer-based work. Mahwah, NJ: Erlbaum.

7. Elm, W. (2002). Applied Cognitive Work Analysis. Pittsburgh, PA: ManTech Aegis Research Corporation.

8. Canas, et al., in press

9. National Research Council (2007). Human-system integration in the system development process: A new look. Washington, DC: National Academies Press.

10. Reason, J.T. (1990). Human error. Cambridge, England: Cambridge University Press.