Systems Engineering for the Human Factors Engineer: A Workshop

George M. Samaras

Samaras and Associates, Inc., Pueblo, Colorado USA

Abstract

Systems Engineering (SE) provides a structured, systematic approach to business and technical risk reduction and hazard mitigation. The emphasis of the workshop is on the use of SE by the human factors engineer (HFE) and the role of the HFE in an SE process. The IEA2006 workshop consists of a 90-minute didactic session (summarized here) and a 2-hour practicum, which consists of the detailed analysis of a simplified safety-critical system.

Keywords: Systems Engineering, Requirements Engineering, User Measurements, Cognitive Systems, System Hazards

1. Introduction

Humans and their organizations are continually intimately involved in every phase of the system lifecycle (development, deployment, and disposal). These humans, and the organizations in which they function, are critical determinants of both the structure and the behavior of their systems and processes. It is logical that, if we want the system behaviors to be *well understood and predictable (*"High Confidence Systems"), at a minimum, we need to understand and manage the influence of all those humans and their organizations on the development, deployment, and disposal or upgrading of the systems and processes. Absent the detailed appreciation and the management of the micro-ergonomics (design and management of tools for human use) and the macro-ergonomics (design and management of human organizations), we will remain unable to control the critical human/organizational influences on system design parameters and system sensitivities to external factors.

At the beginning of the last century, quality was managed by inspecting the product before it was sent to the customer. Towards the middle of the last century, statistical process control was employed, but still maintained a retroactive perspective – fix the "problem", not prevent it from ever occurring! With the advent of quality assurance, the "problem" was attacked proactively (seeking to detect or prevent defects), but was still inward looking - the focus was on the entire production chain, up to the point that it reaches the customer. Today, Strategic Quality Management (SQM) focuses not on initial price but on lifecycle costs (Total Cost of Ownership, TCO), which include manning costs, operating costs, service costs, and disposal or upgrade costs. The development effort focuses on not only system design, but on parameter design and tolerance design. This is also the perspective of classical systems engineering.

2. The Systems Engineering Paradigm

Classical systems engineering (SE) arose sometime in the middle of the last century. The "cradle to grave" approach of classical SE is well documented [1,2,3], but seems to be no longer used in a rigorous manner. It can be shown (e.g. [4,5]) to be a structured, systematic approach that, when applied in a disciplined manner, can accommodate both micro-ergonomic and macro-ergonomic aspects of system development, deployment and disposal. Classical SE is the structured, systematic approach to the development (Research, Design, De-

velopment, Testing & Evaluation or RDDT&E), deployment, and disposal of products and processes [1, 2, 3]. The SE space may be depicted as in Figure 1 [5].

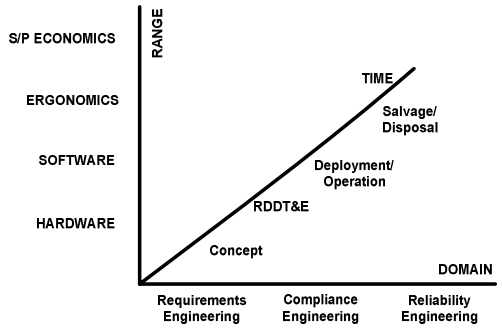

Figure 1: The SE Space

For microergonomics products, the disciplines range from hardware engineering to software engineering to human factors engineering to seller/purchaser economics. For macroergonomics processes, the disciplines range from management to operations to personnel to finance. The domain in both cases is the triumvirate of Requirements Engineering, Compliance Engineering, and Reliability Engineering [5].

Requirements Engineering is that engineering activity of (1) discovering stakeholder needs, wants, and desires (NWDs), (2) selecting those NWDs that will be translated into requirements, and (3) formulating system-focused requirements, so that they satisfice stakeholders, inform designers, and provide a complete and correct basis for validation. Points in the 3-D space (Figure 1) are SE activities, often facilitated by specialized tools.

2.1. Validation vs. Verification

Validation is about solving the correct problem; verification is about solving the problem correctly. Validation reduces system complexity by reducing the degree of uncertainty in system behavior. Validation is not essentially different from the general scientific procedures for developing and supporting theories [6]. System engineering validation is based upon proper, operationally-defined, formulation of system-focused requirements; it consists of empirical measurements to *refute* or *corroborate* compliance with these requirements [7, pg. 48]. As the development process progresses, the set of requirements evolve with each iteration (and intermediate validations occur) until the final set of requirements are developed (and validated) and the product or process is deployed.

Figure 2 identifies the four general user measurement categories from which we may derive operational definitions for system-focused human factors requirements. By overt we mean openly observable, not hidden or concealed; conversely, by covert we do mean hidden or concealed. In order to measure covert phenomena, we need to identify and measure overt resultants (e.g., *force*, a covert physical quantity, is related to the second time derivative of a *displacement*, an overt physical quantity).

	OVERT	COVERT
PHYSICAL	Anthropometry	Biomechanics
BEHAVIORAL	Verbal, Non-Verbal	Affective, Cognitive, Physiological

Figure 2: User Measurement Categories

Overt physical measurements include such things as length and mass (available as tabulated anthropomorphometric data) related to essentially static human characteristics, whereas covert physical measurements include such things as force and acceleration related to the dynamics of the human body. The technology for making such measurements is well developed (e.g., [8,9]). Properly constructed system-focused requirements that operationally define overt physical measurements might include the dimensions of a cockpit or an infusion pump buttons' dimensions; those that operationally define covert physical measurements might include the forces necessary to operate a yoke or install a pump cassette.

Overt behavioral measurements include such things as verbal and non-verbal responses related to external or internal stimuli. The measurement technology is routinely used in experimental psychology. Often these verbal and non-verbal responses are videotaped for later analysis. Properly constructed system-focused requirements that operationally define overt behavioral measurements might include the requisite elements of the conversational content with air traffic control or the layout of push buttons on the pump that would minimize sequence errors.

Covert behavioral measurements include analytical (based on prior information), subjective (based on self-reporting), performance (using a secondary task), and psychophysiological measures (measuring physiological functions that are believed to covary with cognitive functions). There exists a large body of work in cognitive work analysis (CWA) and cognitive systems engi-

neering (CSE) (e.g., [10,11,12,13]). CSE is not, as the name implies, "systems engineering"; it is a requirements engineering approach (identification of activities, actors, and agents [10]). Consistent with long established nomenclature, it should be termed "cognitive requirements engineering". It is a research strategy whose outputs support formulation of *requirements* for the development of tangible products; these requirements must be system-focused, not user-focused and they must support quantitative validation. At present, there still exist gaps between these CWA/CSE outputs and properly formulated system-focused requirements (e.g., [14,15]). As Woods [16] pointed out recently, "Ultimately, the test of CSE as a research strategy is its ability to identify basic requirements for how to support cognitive work that must be met, if new technology will be useful to practitioners in context".

2.2. The SE Lifecycle

From a lifecycle perspective (Figure 3), we can represent SE in either expanded or condensed notation. Buede [17, pg. 18-19] discusses the various equivalent development models (waterfall, spiral, Vee, and rapid prototyping), pointing out that Forsberg & Mooz have shown that "the spiral activities can be mapped onto the Vee model without swapping any activities in time".

The condensed notation (on the right) is "less messy" and clearly delineates the verification and validation loops. This shorthand representation of the SE lifecycle does NOT state that all the needs must be assessed, then all the requirements must be established, then all the specifications determined, and then all the implementation work must be accomplished. This is a frequent and mistaken reading of this shorthand notation.

The expanded notation (on the left) shows the multiple iterations in the project evolution; classical SE may be highly agile with proper project management. As seen from the expanded notation of Figure 3, classical SE possessed many of the attributes of modern agile methods. The systems engineering process is a learning process [18, pg.157]. In each iteration, some NWDs are identified or discovered. Following a hazard analysis (HA), some or all of the NWDs are selected and formulated as requirements for the product or process under development. The evolving requirements (*some* → *more* …→ *final*) define the stakeholders' evolving understanding of the "correct design project". Complete and correct requirements satisfice all stakeholders, inform all designers, and provide a basis for all validation measurements. Defective requirements are a principal cause of incorrect or inadequate system designs [18, pg.4], that can dramatically raise the TCO. Corrective and Preventative Actions (CAPA) may identify some, but not all defects.

2.3. The SE Process

The SE process, from a project management perspective, is shown in the Discrete Event Model (Figure 4; the hazard analyses and the post-CAPA iterations are omitted intentionally to simplify the diagram).

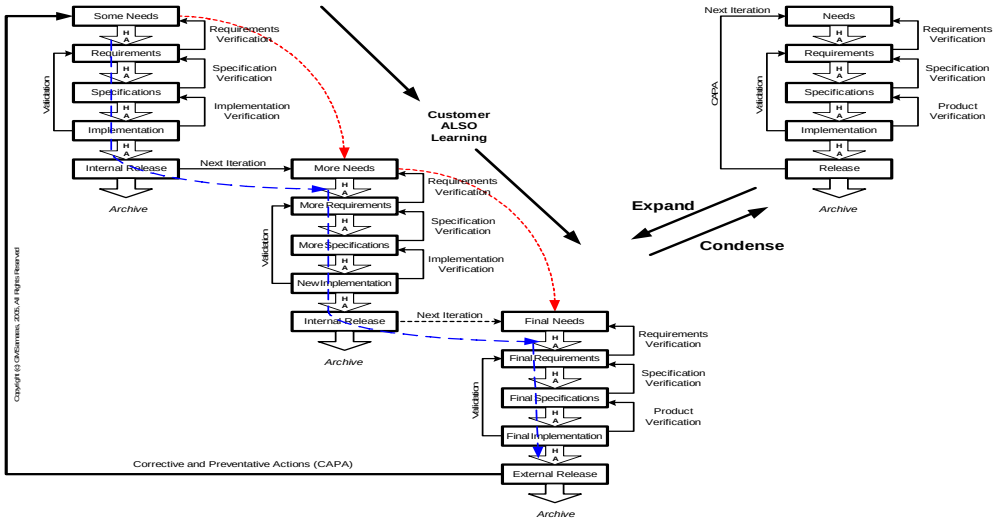

Figure 3: SE Lifecycle Notations – Expanded versus Condensed

The purpose of these periodic re-validations in the deployment/operations phase is to ensure the continuing integrity of the product or process; they are based upon the final set of requirements established, and successfully validated, in the development phase.

We apply the SE process not just in development, but also in deployment/operations and disposal or upgrading. From a human factors engineering perspective, the activities do not end, for example, with the development of the operation, maintenance, and requisite manuals, but continue with the delivery of training, testing and the associated organizational learning and the impact on planning and managing workload. From a macro-ergonomic perspective, strategic and tactical assessments – prior, during, and after deployment – are a continuing activity and a driver of modifications, improvements, and upgrades. These human-centered activities drive hardware, software, and seller/purchaser economic considerations – since in the final analysis, the tools and organizations only exist to support the cooperation and coordination of the human actors in achieving specific strategic and tactical objectives.

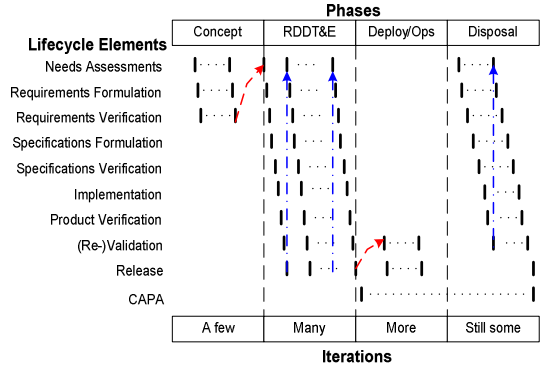

Figure 4: SE Process Discrete Event Model

As the project evolves, the customer is learning (Figures 3 & 4) along with the developers. This learning process is reflected in the multiple iterations. Regardless of the target system (product or process), increasing levels of validation increase the predictability and reliability of system behavior, thus reducing system complexity and increasing *confidence* in the predictability of system behavior. The engineering of complex systems and processes involving human actors benefits from the proper validation of both micro-ergonomic and macro-ergonomic considerations.

2.4. Macroergonomics and the SE paradigm

The SE paradigm also applies to macroergonomic activities. For example, Macroergonomic Analysis and Design (MEAD) is a method of assessing work system processes for organizational design and management activities [19]. Figure 5 [4] shows MEAD mapped to SE (condensed notation); it expands in the same manner as shown in Figure 3.

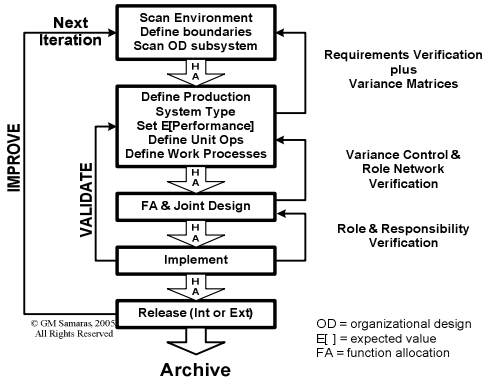

Figure 5: MEAD mapped to SE

2.5. Requirements Formulation

Satisficing stakeholders (e.g., customers, users, disposers, developers, producers and managers) requires understanding each group's NWDs and then prioritizing the resultant requirements, so that appropriate tradeoffs can be made in a systematic and traceable fashion. There are two general approaches: design-dependent and design-independent.

2.5.1 Design-Dependent Formulation

Quality Function Deployment (QFD) is a design-dependent approach of formulating requirements. Figure 6 shows QFD in the context of the SE condensed notation [4].

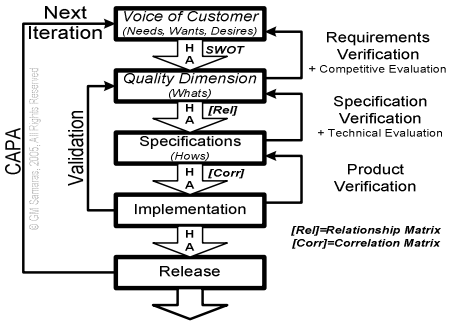

Figure 6: QFD mapped to SE

QFD employs a process of listening to the "voice of the customer" [20, pg 9] to discover, identify and understand NWDs. These NWDs are used to develop the *quality dimension* (synonymous with SE requirements), the "whats". The "whats" are prioritized based upon their importance to each stakeholder. Putative designs (the "hows") are identified and a relationship matrix – relating the "whats" and the "hows" – is constructed. A correlation matrix is also constructed - among the "hows" – permitting identification of conflicts between putative design elements. It is important to note that QFD does not result in design-independent requirements, since putative designs participate in the selection of requirements.

2.5.2. Design-Independent Formulation

Informing designers means that the requirements refer to the system [21, pg 272], not to the user(s); these are system-focused requirements! There exist at least two equivalent methods of defining design-independent requirements: (a) Use Cases that define requirements in context and (b) Requirements Specification that do not specify the use context. In both cases, the synthesis phase consists of organizing the results of the analysis and elaboration phases into a logical, understandable whole. For Use Cases, the synthesis consists of writing a "set of detailed stories" describing the use of the system; for Requirements Specification, the synthesis consists of enumerating the system-focused requirements in a logical, understandable document. From one perspective, they correspond (respectively) to a top-down (deductive) and bottom-up (inductive) approach to requirements elucidation. While typically only one or the other are used, employing both in parallel greatly contributes to achieving increased completeness; this same paradigm is used elsewhere (e.g., *reliability engineering* – fault tree analysis vs. failure mode effects analysis [22]; *physics* – thermodynamics vs. statistical mechanics [23, pg 9-17]; *psychology* – cognitive vs. behavioral [24, pg 38]).

2.6. Latent Failures and Drift

There exist two very important limitations to system engineering validation related to missing or defective requirements. First, if a requirement is absent (a latent failure [25, pg 173, 208] – a hole in Figure 7), the system will incorrectly pass the validation. In the graphical example (Figure 7), routine use of standardized specifications obfuscate the existence of the missing requirement and block the hazardous state; when the specification is subsequently changed (such as during a manufacturing quality engineering optimization), the unanticipated hazardous state "unexpectedly" appears. A means of minimizing this is the iterative HA; however, ruthless enforcement of Requirements Engineering is fundamentally the best approach. Formal methods also may be used to mitigate this.

Second, "drift" [26, pg 35-7] during manufacturing or post-deployment maintenance will expose unanticipated hazards that may not be susceptible to traditional validation studies. An example of this might be a variation in maintenance either that was not envisioned (a missing requirement) or that was outside of the control of some of the stakeholders (a defective requirement) (e.g., [26, pg 31-33]).

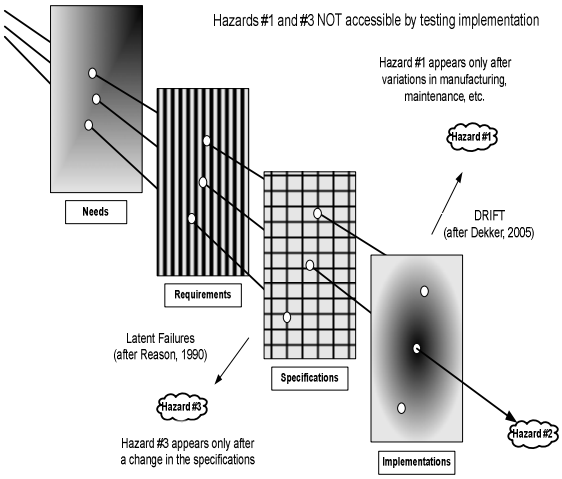

Figure 7: Hazards due to latent failures & drift

Such unanticipated events (e.g., Hazards #1 and #3 in Figure 7) are typically not accessible during validation of the system implementation; they may only be accessible during the periodic re-validations and this will be highly dependent on the design of the re-validation protocol. Once again, the iterative HA, ruthless enforcement of Requirements Engineering, and, possibly formal methods, are indicated.

3. Conclusion

A cardinal rule of system development has to be ruthless enforcement of Requirements Engineering. Complete and correct requirements satisfice all stakeholders, inform designers, and provide a basis for quantitative validation measurements. Without properly formulated, system-focused requirements, there can be no meaningful validation. Without proper

validation, system development cannot be certified as completed. Operationally defined, system-focused human factors requirements will inform designers and enable quantitative validation studies. Inclusion of HFE engineers throughout the system lifecycle will encourage incorporation of HFE knowledge and allow them to contribute throughout the complete system development process.

It has been said that humans are for tasks that designers are incapable of automating [25, pg 180]. Human behavior is profoundly influenced by artifacts and organizations [27, pgs 221-222]. If designs are hypotheses about how artifacts shape human behavior [28], then it is imperative that the HFE community become engaged in influencing designs by formulating proper system-focused requirements (using HFE conjectures that are derived from scientifically-acquired data) and by taking responsibility for their validation measurements of human-centric design. Ignoring Chapanis' advice to produce system-focused requirements [21, pg 272] and not contributing throughout the complete development process, we will only continue with the non-human-centered [10, p.172-3] or the "left over" design approach [29, pg 607].

References

[1] Hall, A.D. Systems engineering from an engineering viewpoint. IEEE Trans Syst Sci Cyb, SSC-1:4-8, 1965

[2] Hall, A.D., Three-Dimensional Morphology of Systems Engineering, IEEE Trans. Syst. Sci. Cyb. SSC-5:156-160, 1969

[3] Nadler, G. Systems Methodology and Design. IEEE Trans. Syst. Man, Cyb. 15(6):685-697, 1985

[4] Samaras, G.M. Engineering Complex Systems: Validating the Human Factors. Proc. 7th Human Interaction with Complex Syst. Symp., Greenbelt, MD, 2005, *in press*

[5] Samaras, G.M., & Horst, R.L. A systems engineering perspective on the human-centered design of health information systems, J. Biomed. Info, 38: 61-74, 2005

[6] Cronbach, L.J. & Meehl, P.E. Construct validity in psychological tests. Psych. Bull, 52:281-302, 1955

[7] Popper, K.R. Conjectures and Refutations: The Growth of Scientific Knowledge.London:Routledge. 1963/2005

[8] Chaffin, D.B. & Andersson, G.B.J. Occupational Biomechanics, 2nd Edition. NY: Wiley-Interscience. 1991

[9] Kroemer, K.H.E. Engineering Anthropometry, in Handbook of HFE, 2nd Edition, (G. Salvendy, Ed). New York: Wiley-Interscience. 1997

[10] Rasmussen J, Pejtersen AM, Goodstein LP. Cognitive Systems Engineering. New York: Wiley. 1994

[11] Vicente, K. J. Cognitive work analysis: toward safe, productive, and healthy computer-based work. Mahwah, NJ: Lawrence Erlbaum Associates. 1999

[12] Hollnagel, E. (Ed.). Handbook of cognitive task design. Mahwah, NJ: Lawrence Erlbaum Associates. 2003

[13] Hollnagel, E. & Woods, D.D. Joint cognitive systems: foundations of cognitive systems engineering. Boca Raton, FL: Talyor & Francis/CRC Press. 2005

[14] Eggleston, RG, Roth, E., Whitaker, R., & Scott, R. Conveying Work-centered design specifications to the Software Designer: A retrospective analysis. Proc. HFES 49th Annual Meeting, Orlando, FL. 2005, pgs. 332-336.

[15] Lintern, G. Integration of cognitive requirements into system design. Proc. HFES 49th Annual Meeting, Orlando, FL., 2005, pgs. 239-243.

[16] Woods, D.D. Generic support requirements of cognitive work: Laws that govern cognitive work action. Proc. HFES 49th Annual Meeting, Orlando, FL., 2005, pgs.317-321.

[17] Buede DM, The Engineering Design of Systems: Models and Methods, New York : John Wiley and Sons, Inc. 2000. p. 18-9.

[18] Eisner, H. Essentials of project and systems engineering management. New York: Wiley-Interscience. 1997

[19] Kleiner, B.M. Macroergonomic analysis and design for improved safety and quality performance. Intl. J. Occ. Safety and Ergonomics, 5(2), 217-245, 1999

[20] Madu, C.N. House of quality in a minute. Fairfield, CT: Chi Publishers. 1999

[21] Chapanis, A. Human factors in systems engineering. New York: Wiley, 1996

[22] Long A. Beauty & the Beast – The use and abuse of fault tree as a tool. Accessed on: October 22, 2005. Available at: http://www.fault-tree.net/papers/long-beauty-and-beast.pdf.

[23] Pathria RK. Statistical Mechanics. Oxford: Pergamon Press, 1972

[24] Grossberg S. How does a brain build a cognitive code. in: Studies of Mind and Brain. Dordrecht, Holland: D. Reidel Publishing Co., 1982

[25] Reason J. Human error. Cambridge: Cambridge University Press. 1990

[26] Dekker SWA. Ten questions about human error: A new view of human factors and system safety. New Jersey: Lawrence Erlbaum Associates. 2005

[27] Boulding, K.E. Ecodynamics: A New Theory of Societal Evolution. Beverly Hills: Sage, 1978.

[28] Woods, D.D. Designs are hypotheses about how artifacts shape cognition and collaboration. Ergonomics. 41: 168-173. 1998.

[29] Hendrick HW. Organizational Design and Macroergonomics, Handbook of HFE, 2nd Edition, (G. Salvendy, Ed). NY: Wiley-Interscience. 1997

Contact:
G. M. Samaras, PhD, DSc, PE, CPE, CQE;
george@samaras.eng.pro