IEEE Intelligent Systems
www.computer.org/intelligent

HCC Implications for the Procurement Process

Robert R. Hoffman, Florida Institute for Human and Machine Cognition
William C. Elm, Aegis ManTech Research Corporation

Vol. 21, No. 1
January/February 2006

This material is presented to ensure timely dissemination of scholarly and technical work. Copyright and all rights therein are retained by authors or by other copyright holders. All persons copying this information are expected to adhere to the terms and constraints invoked by each author's copyright. In most cases, these works may not be reposted without the explicit permission of the copyright holder.

IEEE COMPUTER SOCIETY

© 2006 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

For more information, please see www.ieee.org/portal/pages/about/documentation/copyright/polilink.html.

HCC Implications for the Procurement Process

Robert R. Hoffman, *Florida Institute for Human and Machine Cognition*
William C. Elm, *ManTech International*

Most system designers and human factors engineers have participated in projects that culminated in systems that were highly constrained by short-term cost considerations. In the procurement of information processing and intelligent technology for complex sociotechnical domains, the focus on short-term cost considerations at the expense of human-centering considerations always comes with a hefty price down the road. This price weighs much more heavily on users' shoulders than on those of the technologists or project managers.

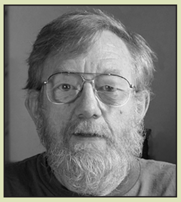
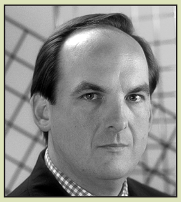
We illustrate this with just one of many examples: design of the US National Weather Service's Advanced Weather Information Processing System (AWIPS). Evidence from cognitive task analysis (CTA) had clearly shown that forecasters inspect on the order of seven data types or fields per minute.[1] The traditional meteorological chart wall lets forecasters inspect multiple data types, flip through charts over time, make annotations using colored markers, conduct weather briefings, and so on. When the National Weather Service was revamping the traditional chart wall as a computerized workstation, it was clear from a task analysis that forecasters would need at least four large-screen CRTs, each dedicated to particular data types depending on the day's weather.[1] One might be for computer model outputs, one for showing the satellite image loop, one for showing radar, and one for composing forecasts. However, the initial AWIPS prototype had a single CRT. Subsequent versions had more than one CRT, and the now-operational AWIPS has three. But throughout the reprototyping process, there was a momentum to limit the number of CRT displays because of cost considerations, despite considerable reference to human-factors issues.

A "solution" the designers adopted was to screen-sector the views of various data types. This quick fix didn't go very far for the graphically and symbolically dense displays involved in weather forecasting (that is, sector-minimized displays are illegible). Moreover, sectoring requires a great deal of make-work that burdens forecasters—pointing and clicking to minimize and maximize particular data types' views. And it ignores the fundamental point: that forecasters must be able to see at a glance a number of diverse data types, with the types depending on the forecasting situation at hand.

So, somehow the process of procuring the AWIPS deviated from its purported human-centered intentions. And by the way, following the introduction of the new electronic workstations, some forecasters have reinvented their traditional chart wall by tacking traditional paper charts to their cubicle dividers.

There are many additional stories about how reality hasn't matched information technology's promise.[2–4] Here's a passage from "Out of CAOCs Comes Order," which appeared in *Jane's International Defense Review*, May 2003:[5]

> New technology and revised procedures are greatly enhancing the capabilities of [Air Operations Centers].… "The buzzword for this decade is going to be 'integration.' Why can't we do that today? Why aren't we integrated now?" So said General John Jumper, US Air Force (USAF) Chief of Staff.…"[All] the stovepipes in each segment of the chain have to work in separate ways to make it all happen. Certain tribes within each of those stovepipes have taken steps to make sure they can't be interfered with by any other segment. We have formed antibodies to integration. You go into an Air Operations Center (AOC) today, and what will you see? Tribal representatives sitting down in front of tribal workstations, interpreting tribal hieroglyphics to the rest of us who are on watch. And then what happens? They stand up and walk over to another tribal representative, and reveal their hieroglyphics, which are translated by the other tribe into its own hieroglyphics and entered

Editors: Robert R. Hoffman, Patrick J. Hayes, and Kenneth M. Ford
Institute for Human and Machine Cognition, University of West Florida
rhoffman@ai.uwf.edu

into its own workstation. What if machines talked to one another? That would break down the stovepipe ….”

The author has nailed a problem—the AOC systems aren’t human-centered. At the risk of being accused of summoning a root cause analysis out of a cauldron, we hasten to add that everyone involved in this AOC project is smart, well-intentioned, and highly motivated: “The USAF leadership is considering how to proceed with further enhancements of its AOCs, with emphasis on achieving true integration of systems rather than mere interoperability.”[5]

Clearly the procurement process, at least in the US, leads to the creation of systems that are anything but human-centered. In this essay, we reflect upon the process by which information technologies (including “intelligent” decision aids) are procured, in light of human-centered computing.[6]

Down into the weeds

There is no single procurement process, of course. Looking at any one large organization—say, the US Department of Defense (DoD)—how technologies are procured depends on many guidelines and procedures, including literally thousands of specifications for everything from interface font sizes to stepwise budget-reporting processes. Many technical reference manuals, memoranda, addenda, appendices, and architecture framework standards specify data formats, communication exchange formats, interoperability requirements, software documentation requirements, and more, in a mind-boggling panoply of procedures and acronyms.[7–10]

Standing back from the gory details, the procurement process, or at least some large chunks of it, is typically summarized with reference to either the waterfall or spiral model.[11] (The literature offers a number of waterfall variations, including the rapid-prototyping model and the incremental model.[12]) Figure 1 presents the idealized waterfall model, and figure 2 presents the idealized spiral model.

In the waterfall model in figure 1, the steps are in iterative pairs. For instance, software requirements analysis feeds into preliminary design, but effort at that second step can feed back into software requirements analysis, leading to changes in the requirements. There’s also a review process at each step.

In the spiral model in figure 2, the quadrants show four activities that are presumed to be fundamental, and sequential. The spiral model is explicit about system evaluation, but the typical evaluations and verifications for both spiral and waterfall modeling are often based just on a “satisficing” criterion. That is, when users are asked to work with a system prototype for a while and are then queried about their opinions, results show that some people like it, more or less, at least some of the time.

Evidence suggests that we can attribute many of the breakdowns in human-computer interaction (such as automation surprises) to the procurement process. Every technologist has seen it. Even some program managers with whom we’ve talked bemoan the situation and argue that we must scrap the mandated, legacy procurement process. But they admit that they themselves are handcuffed by it. On the other hand, we’ve heard strident claims that we must couch the development of information processing systems, including intelligent systems, in terms of the waterfall or spiral model. These models, so it’s claimed, express the categories and process that system developers actually follow, that developers must follow, or (for the candid ones among them) that program managers follow because they’re forced to. Upstarts who point out nasty empirical facts (“Yeah, but what you say you do is not what you really do!”) hear that they must recast their ideas into the waterfall or spiral lingo because that’s what system developers use and are comfortable with.

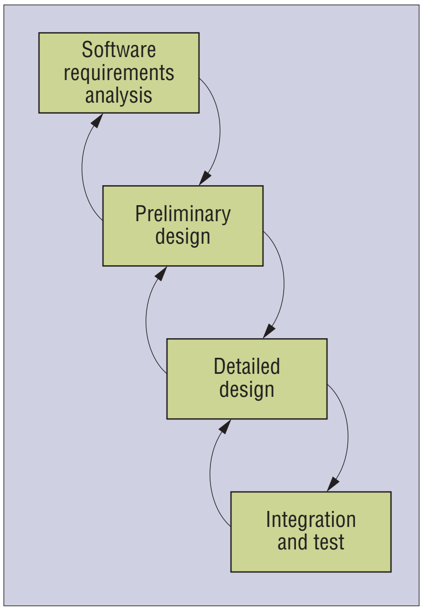


Figure 1. The idealized waterfall model.

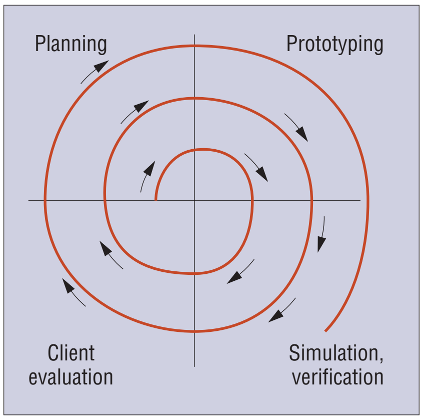


Figure 2. The idealized spiral model.

The trap of designer-centered design

Designer-centered design, whether conducted under the guise of either model, goes essentially like this:

1. Specify the requirements.
2. Design the automation to enforce the requirements.
3. Deliver the system as (what is believed to be) a finished product.
4. Force the human user to execute the designer’s plan.

The result is just the sort of ubiquitous technology (for example, VCR remote-control devices) that frustrates people at home and at work. Following are the negative and usually unanticipated consequences:

- *Effort*. The user must adapt to “tasks” defined by the designer and the machine.
- *Bewilderment*. Computers can be difficult to understand or use.
- *Roadblocking*. Computers provide impoverished feedback, limit users’ ability to explore and integrate information, and restrict the ability to detect and recover from error.
- *Overload*. Computers don’t help people cope with data or mental overload, and in many circumstances actually contribute to it.
- *Error*. Although a new computer system or interface might help you avoid certain kinds of error, they invariably create new forms and patterns of error—error attrib-

utable to the human-machine system, not to the human.

- *Clumsiness*. New technology might make some jobs easier but usually makes some jobs harder. Computers often reduce motivation and create a need for kludges, local solutions, and other means for avoiding make-work.
- *Surprise*. Users are sometimes surprised by the actions that automated agents take (or don't take). Worse, the automated agents don't make their mechanisms (or intent) apparent.

The net result is that people yell at machines, even simple ones such as VCR remote-control devices.

The fundamental problem here is the trap of designer-centered design: The road to user-hostile systems is paved with user-centered intentions, even on the part of smart, well-intentioned people who are aware of this trap.

Acquisition reform?

Reform isn't new to the acquisition process. A streamlining in DoD acquisition occurred in the mid-1990s at the US Secretary of Defense's direction. This new way of doing business included canceling some blocks of specifications without replacing them and leaving more things to contractors' discretion.[10] The effect was to reduce the number of specifications by 62 percent. However, this meant that the burden merely shifted to the documents that laid out non-DoD standards (such as NASA and Human Factors and Ergonomics Society documents). Our point here isn't about the necessary and gory details—font size is a consideration in the human factors of all display design. Rather, our point falls at a higher level, one dealing with human-centering. DoD Instruction 5000.2-R states,[7]

> Program managers shall initiate a comprehensive strategy for [human-system integration] early in the acquisition process to minimize ownership costs and ensure that the system is built to accommodate the human performance characteristics of the user population that will operate, maintain, and support the system. (para. C5.2.3.5.9)

Here we see that human-centering considerations are reduced to maintainability, safety, performance metrics, and training efficiency (with cost-effectiveness always being the cart pulling the horse). All these are surely important. However, in all the procurement documents, we only occasionally see a reference to guaranteeing on the basis of empirical evidence that the eventual technologies will help domain practitioners work on problems rather than forcing them to fight with the technology. Even then, the requirements are stated as "physical/cognitive requirements" or "human performance effectiveness." The acknowledgment that systems should be both usable and useful, that they should motivate and not frustrate, is rarely made explicit. Where the rubber meets the road, information technology needs to support human reasoning, knowing, perceiving, and collaborating. We know that expertise comes from extensive, continuous, deliberate practice, including practice at difficult tasks.[13] But in the standards and procurement documents, we see

The acknowledgment that systems should be both usable and useful, that they should motivate and not frustrate, is rarely made explicit.

no consideration that systems should support the achievement and expression of expertise. Quite to the contrary (and to our shock), there's actually a push, intended or not, to prevent users from achieving or exercising expertise: "Design efforts shall minimize or eliminate system characteristics that require excessive cognitive, physical, or sensory skills"[7] (para. C5.2.3.5.9.1).

Rethinking requirements

There's wide recognition that the development of information technologies hinges upon the interaction of users, systems designers, and systems developers (including systems analysts, computer scientists, and engineers).[14] Designers must understand users' needs and the goals for the system being created.[15] However, the process of designer-user interactions isn't grounded in the empirical methodologies of CTA. Thus, "poor or error-prone communication between the user and analyst remains a major problem"[16] (p. 257).

Miscommunication results in misinterpreted user needs.[17] This leads to design glitches that force users to create work-arounds and cope with user-hostile features such as brittleness and automation surprises when the machine does things the user doesn't understand. Consequently, it's the rare system that doesn't have to go through redesigns, often costly ones.[18] The need to improve this process is a major concern to the entire information systems development industry.[19,20]

A driving factor here is technological backlash—the inevitable negative consequence of "intelligent" technologies that aren't created under human-centering methodologies. The technologies don't work, so it's little surprise that this sometimes creates scandal. The most recent (in a depressingly long series of cases) is the US Federal Bureau of Investigation's discovery that the costly new Trilogy information technology modernization program resulted in software that does not support analysts' cognitive work (referred to as "operational needs").[21] The problems were blamed on a lack of adequate prototyping and testing in the operational context and on the fact that system requirements changed over the course of development.

This critique made us wonder about the assumption that requirements should be fixed, especially in a world that is not. In fact, change in the world seems to vastly outpace our ability to build and adequately test large-scale decision- and performance-support systems. Yet, the default belief seems to be that the world can be frozen and that requirements must be frozen, or nothing will get built. Technological backlash always results in a blame game seeking simple, clear-cut human errors or human limitations, or criticizing a contractor when the root cause is systemic. "Requirements creep" is not a nasty thing to eradicate, but an empirical inevitability to accommodate and understand empirically.

The shortcomings of traditional software engineering and system development, with respect to their ability to support decision making, aren't inherently linked to any particular software engineering approach. The research community has suggested and is vigorously pursuing alternatives to the traditional spiral and waterfall models, including

Extreme Programming,[22] Rapid Application Development,[23] and Joint Application Development.[24] All these procedures

- involve representative end users (but only minimally, mostly in the early stage of system development and basically in the form of focus groups), and
- assume that requirements specification is a clear-cut starting point for the system development process.

In general, software engineering does not regard requirements specification as a process. All the alternative models assume that the requirements are correct, and they seek to build to the requirements in high-quality ways. The system developers might therefore be building the wrong system, even though they might be building it well. The earliest formal approach, the waterfall model, was instantiated in a series of IEEE and military standards that focused on the content of each sequential design artifact in the process. The development effort "flowed downhill" from one approved document to the next. Virtually no attention went to the work activities needed to actually develop those documents' content. This was particularly evident in waterfall's initial requirements definition phase.

Spiral processes addressed some of the limitations of a downhill flow but still assumed that some miraculous insight would provide good system requirements concerning what to build for each spiral. Evaluating the previous spiral improved the odds to some degree, but it was still a long shot. This was true whether the programming language was procedural or object-oriented. An example is James Rumbaugh's Object Modeling Technique,[25] since merged with the Booch method[26] to become the Unified Modeling Language.[27] It's an example of an OO software engineering methodology that still assumes that requirements for effective decision support are divined.

As this recurrent "build the wrong thing well" syndrome became more evident, the community began to offer various techniques in response. Use cases are one example.[28] This approach tries to capture users using the proposed system in an easy-to-understand language—that is, objects. The assumption is that if enough cases are captured, all the needs of the users' interactions with the intended system would be exposed, and these use cases could then be the basis for system requirements definition. Use case notation comes in various forms—for example, what the Object Modeling Technique calls event trace diagrams, now called sequence diagrams in UML. Although this method focuses on users' interactions, it does nothing to help system developers determine what the correct interaction should be. So, in essence, it might be a better way to ensure completeness but not correctness.

Rapid application development and joint application development are becoming fashionable as ways to (supposedly) ensure that developers explicitly incorporate "user needs" into the system development process. The theory is that by holding a user captive within the development team for the first few months of system development, you will produce a design that is correct for the user. At best, such a method allows the program manager to say, "We had user involvement, so therefore it is a user-centered system." But this claim doesn't hold water. The difficulty in determining good decision support, even with a domain practitioner in the design team, is evident by the desperate measures that developers employ to achieve those requirement epiphanies. For example, we found the following advice regarding how to get users to "be creative" in describing system requirements:[29]

Use case notation does nothing to help system developers determine correct system-user interaction. It might be a better way to ensure completeness but not correctness.

> The first JAD I facilitated involved a three-day business trip to Florida with the JAD team. At the start of the first session after we returned, I mysteriously found a two-foot rubber alligator on the overhead projector. This "gator" has since become legendary and attends all JAD sessions. I use him as a tool by "speaking through him." Via this character, I get people to speak—or not speak. We consult him on important issues, and involve him in helping the group reach consensus.

And on the cover of this book is the reviewer's statement, "This book is a gold mine of practical advice on the organization and conduct of JAD sessions." Go figure.

Our point is that although software engineering methodologies focus on good construction techniques, they do not yet effectively address what's needed to provide good intelligent technologies or decision support systems. The mistaken belief is that if the initial specifications are correct and complete, the rest of the development process can proceed and will lead to a "final" system.[16]

Both the waterfall and spiral models have advantages and disadvantages (for example, whether they minimize certain kinds of risks, how they trade off effort and cost, or what project scale they're suited for).[12] Both involve feedback and review. Both have worked to the satisfaction of some researchers, at least some of the time. Even when a façade, they still satisfy someone, as indicated by the repeated references to spiral development in DoD documents.[8,9] Both of the models have worked insofar as some of the resulting software products have been put to actual use. However, both models

- represent a trajectory through the wrong kind of design space—the technology-centered, designer's space;
- include some notion of evaluation, but the wrong kind of evaluation—satisficing; and
- have some notion of involving domain practitioners, but the wrong kind of involvement—the designer-users might also be serving as domain experts, or it's too little involvement, or too late.

OK, what if we reach for a better model?

HCC principles as design challenges

By becoming part of the mind-set of how to design and develop systems, HCC might impact the numerous systems that engineers are routinely constructing using desperation measures. That very desperation becomes an opportunity. The challenge to the cognitive systems engineering community is how to adapt the HCC paradigm and methods to produce the outputs needed to integrate the system development processes. The complexities of the modern sociotechnical workplace, and the computational systems that inhabit it, behoove us to reflect on all of HCC's principles and the laws of cognitive work to discover the implications for procurement.

First, HCC mandates using CTA—that is, analyzing work in terms of its macrocognitive functionalities. These include knowledge, reasoning strategies, data integration skills, and decision-making skills. HCC also mandates studying domain practitioners in their actual work context. A string of recent cognitive systems engineering success stories includes using CTA to support the redesign of workstations aboard US Air Force aircraft and US Navy vessels, using CTA to create improved training methods for various important jobs, new interfaces in air traffic control, and new control systems for power plants.[13,30–32]

Glaring by its absence in the spiral and waterfall models is any emphasis on supporting procurement and specification by conducting CTA with domain practitioners (especially experts). The models don't even acknowledge that developers should conduct CTA before, during, and after systems engineers perform requirements specification, planning, and system design. And, we must note, CTA must involve much more than simple interviews or one-off focus groups.[16,33]

The work that people really need to accomplish often differs from their "actual work," since the latter is shaped by legacy technologies and mandated procedures.[32] A palette of CTA techniques from human-factors engineering and cognitive anthropology involves ways of revealing the true work by studying people at work.[13] What are the patterns and cues that decision makers perceive? What are the tough decisions? What knowledge is needed to make good decisions?

A second way in which HCC offers value to procurement will derive from taking HCC principles and the laws of cognitive work as design challenges or policies.

The Zero Tolerance Challenge

If we look closely at procurement standards documents and ferret out a list of what was really important to their authors, we see demands for interoperability and integration, portability, reusability, development efficiency, vendor independence, and manageability. We don't contest the importance of these things, but nowhere do we see a specification like this: *The system must be proven to be usable, useful, and understandable. It must be user friendly and have no user-hostile aspects.* Countless systems (including most VCR remote-control devices) would literally disappear overnight if we applied such a stricture. And we do not lack reports from domain practitioners on what works, what works well, and what needs fixing.[34]

The Sacagawea Challenge

This principle is named after the Indian guide of the Lewis and Clark expedition: *Human-centered computational tools need to support active organization of information, active search for information, active exploration of information, reflection on the meaning of information, and evaluation and choice among action sequence alternatives.*[35] Practitioners must be shown information in a way that's organized in terms of their major goals. Information needed for each particular goal should be shown in a meaningful form and should allow the human to directly comprehend the major decisions associated with each goal.

The work that people really need to accomplish often differs from their "actual work," since the latter is shaped by legacy technologies and mandated procedures.

The Envisioned World Challenge

New technologies are hypotheses about the effects of technologies on work patterns.[36] The introduction of new technology will bring about unanticipated changes in the cognitive and collaborative work of individuals and teams.[37] If new technology does indeed change cognitive work, then the handover of a deliverable can't be the end of procurement; it must be the beginning of a next wave of empirical research, to assure that cognitive work changes for the better.

Hold on, that's not good—procurement would actually take longer than it does now! Worse, if technology changes the work, practitioners in the current world of work won't necessarily carry their expertise over into the envisioned world. The jobs, roles, or tasks that they will conduct in the envisioned system might not be fully formed until after at least some of the technology is in place. Practitioners will have to backpedal to adapt to the new technologies.

The Envisioned World Challenge is this: *Exploration of the envisioned world must be folded into the system development process*. From design notion to design sketch, to initial mock-up, to first prototype, then refined prototype, new technology is embedded in the actual workplace. Designers work with users as users work with and comment on new devices. As the new devices come closer to matching the envisioned world, the users find themselves spending more time conducting their work using the new devices, and over time the legacy systems collect dust. At that point, the new world is essentially ready to become operational.

We've heard a few cases of advanced decision support systems development where, at every iteration (from design sketch to rough mock-up, and so on), users evaluated the prototype in or near their normal work context. They reached a point where the prototype was good enough, even better in some respects than their current systems. At that point, they kept the prototype in the operational facility, and as it was improved, they used it more and more as the primary tool. The old world of practice and the envisioned world coevolved. And the envisioned world included a cohort of individuals who learned within the evolving system context.

The Challenge of Adaptive Design

Two additional empirical facts have strong implications for procurement. The first is expressed by the notion of trickle-off ergonomics: *Prototyping never ceases, it just trickles off.* Complex technologies always undergo iteration and rebuilding after delivery. Software is always upgraded. Individuals, organizations, and teams always create local solutions. They always use Post-Its. The second nasty empirical fact is the Moving Target rule: *The* *socio-technical workplace is constantly changing, and constant change in environmental constraints may entail constant change in cognitive requirements, even if domain constraints remain constant*.[38] Given the empirical reality of trickle-off and the moving target, why then do procurement processes assume that deliverables must be essentially finished products? Why not admit reality and assume some notion of adaptive design?[36] In adaptive design, deliverables must have built-in (dare we say intelligent?) capabilities to enable them to be easily adapted and evolved so as to better meet the needs of particular organizations, teams, or individuals. The process goes like this:

1. The designers and practitioners identify a constraint space based on domain constraints and support for user control.
2. The designers build the tool so that it shows the constraint boundaries and data for ongoing situations in a way that helps practitioners choose among action sequences.
3. The practitioners "finish" the design on the basis of local information, knowledge, and expertise.

Adaptive design supports workers in tailoring their systems and in adapting to them. Practitioners must be empowered to control and modify their tools. Some factors that practitioners must consider can only be discovered by those practitioners during operation-in-context, which is itself always a moving target. A design should support the continual adaptation of the functionality of the sociotechnical system to local and contextual contingencies.

For example, process control workers sometimes deliberately alter alarm thresholds for various reasons, but usually to either reduce false-alarm rates or to have an alarm serve as an ad hoc notifier. In the Guerlain and Bullemer system,[39] workers could define temporary, context-specific notifications of occurrences or alarm states without altering or overriding the standard alarms. The system supports workers' local adaptation and thus mitigates their need to conduct ad hoc tailoring activities that could otherwise lead to errors, especially in unforeseen circumstances.

The notion of adaptive design brings us back to the problem with requirements.

Stella's Challenge

This principle is named after the character in Tennessee Williams' play, *A Streetcar Named Desire*: *Human-centered systems have a built-in capability to be expanded in the future so as to include functionalities that users desire but that aren't or can't be included in initial versions of the system.* Both the spiral and waterfall models fail to acknowledge that users not only have immediate, definite needs (that should be captured as requirements) but also that they have *desirements*. John MacNamara of the US Air Force Rome Labs suggested this concept (in a personal communication) to refer to desired functionalities that currently can't be included as system requirements but might be at some future time. These are not "bells and whistles" to be scratched off the project's accounting ledgers because managers think users don't really need them or because budget constraints preclude expending time and effort. While appreciating the trade-offs involved here, we're also guided by the common finding from studies of expertise in context that domain practitioners' candid statements about what they really need are often not given due consideration.[13]

> Studies of expertise in context have found that domain practitioners' candid statements about what they really need are often not given due consideration.

To build for desirements, we must create technology from the start so as to anticipate, and not merely allow for, subsequent modification. Modifiability is generally thought of in terms of common operating systems, interoperability, and modularity. A new software bundle might plug and play, making the technology adaptable from an engineering point of view. But there's also a largely neglected human-centering and human-machine system aspect. Because of the complex interactions and contextual dependencies that are always involved in complex cognitive and collaborative work, adding a new capability on the basis of a desirement might alter an existing system capability's work demands. For example, it might lead to goal conflicts.

We submit that regarding HCC notions as design challenges or policies for procurement has promise for making information technologies more intelligent by virtue of making them human-centered. One thing is clear: The legacy procurement process bears some responsibility for resulting in systems that are not human-centered.

We have one final recommendation—perhaps our most far-reaching one. Simply stated, it is to begin now to develop a program to educate a new cohort of individuals in both cognitive systems engineering and information technology project management. The goal will be to groom a generation of project managers who advocate for domain practitioners; who prioritize the creation of demonstrably usable, useful, and understandable technologies; who appreciate CTA's critical role; and who appreciate the significance of empirical generalizations such as the Envisioned World Principle. ■

Acknowledgments

We thank John McNamara of the US Air Force Research Lab for illustrating the concept of a "desirement" and how the notion can have utility in the system design and procurement process.

Robert Hoffman's contribution was made possible in part through his contributions to Chi Systems Inc. on the contract "Embedding Cognitive Systems into Systems Engineering Practice," DoD SBIR topic AF05-071, and through his participation in the Advanced Decision Architectures Collaborative Technology Alliance, sponsored by the US Army Research Laboratory under cooperative agreement DAAD19-01-2-0009.

References

1. R.R. Hoffman, "Human Factors Psychology in the Support of Forecasting: The Design of Advanced Meteorological Workstations," *Weather and Forecasting*, vol. 6, 1991, pp. 98–110.

Robert R. Hoffman is a senior research scientist at the Institute for Human and Machine Cognition. Contact him at IHMC, 40 So. Alcaniz St., Pensacola, FL 32502-6008; rhoffman@ihmc.us.

William C. Elm is the executive director of the Cognitive Systems Engineering Center of ManTech International. He has over 20 years of experience in applying cognitive systems engineering, adapting the science of David Woods, Jens Rasmussen, Erik Hollnagel, and others into a pragmatic systems engineering process. Contact him at Cognitive Systems Eng. Center, ManTech Int'l, 501 Grant St., Ste. 475, Pittsburgh, PA 15219; welm@mindsim.com.

2. G.A. Jamieson, "Bridging the Gap between Cognitive Work Analysis and Ecological Interface Design," *Proc. 47th Ann. Meeting Human Factors and Ergonomics Soc.*, Human Factors and Ergonomics Soc., 2003.

3. D.R. Jones and M.J. Smith, *Implementing New Technology: Proc. Human Factors and Ergonomics Soc. 48th Ann. Meeting*, Human Factors and Ergonomics Soc., 2004, pp. 1601–1604.

4. T.K. Landauer, *The Trouble with Computers*, MIT Press, 1995.

5. M. Hewish, "Out of CAOCs Comes Order," *Jane's Int'l Defense Rev.*, May 2003, p. 22.

6. R.R. Hoffman, P.J. Hayes, and K.M. Ford, "Human-Centered Computing: Thinking in and outside the Box," *IEEE Intelligent Systems*, Sept./Oct. 2001, pp. 76–78.

7. "Mandatory Procedures for Major Defense Acquisition Programs and Major Automated Information Systems Acquisition Programs," Instruction 5000.2-R., US Dept. of Defense, 1996.

8. W. Kaworski, *Handbook of Standards and Guidelines in Ergonomics and Human Factors*, US Dept. of Defense, 2001.

9. *Technology Transition for Affordability: A Guide for S&T Program Managers*, tech. report, Office of the Deputy Undersecretary of Defense (Science and Technology), US Dept. of Defense, 2001.

10. L.M. Adams et al., "Human Factors Engineering and Acquisition Reform," *Ergonomics in Design*, Summer 2003, pp. 10–15.

11. A. Behforooz and F.J. Hudson, *Software Engineering Fundamentals*, Oxford Univ. Press, 1996, chapter 1.

12. S.R. Schach, *Object-Oriented and Classical Software Engineering*, McGraw-Hill, 1996.

13. B. Crandall, G. Klein, and R.R. Hoffman, *Working Minds: A Practitioner's Handbook of Cognitive Task Analysis*, MIT Press, 2006.

14. G.I. Green, "Perceived Importance of Systems Analysts' Job Skills, Roles, and Non-Salary Incentives," *Management Information Systems Quarterly*, vol. 13, 1989, pp. 115–133.

15. K. Holtzblatt and H.R. Beyer, "Requirements Gathering: The Human Factor," *Comm. ACM*, vol. 38, no. 5, 1995, pp. 31–32.

16. L.A. Freeman, "The Effects of Concept Maps on Requirements Elicitation and System Models during Information Systems Development," *Proc. 1st Int'l Conf. Concept Mapping*, Universidad Pública de Navarra, 2004; http://cmc.ihmc.us/papers/CMC2004-126.pdf.

17. Y.-G. Kim and S.T. March, "Comparing Data Modeling Formalisms," *Comm. ACM*, vol. 38, no. 6, 1995, pp. 103–115.

18. P.C. Scott, "Requirements Analysis Assisted by Logic Modeling," *Decision Support Systems*, vol. 4, no. 1, 1988, pp. 17–25.

19. P.J. Guinan, J.G. Cooprider, and S. Faraj, "Enabling Software Development Team Performance during Requirements Definition: A Behavioral versus Technical Approach," *Information Systems Research*, vol. 9, no. 2, 1998, pp. 101–125.

20. D. Teichroew, "A Survey of Languages for Stating Requirements for Computer-Based Information Systems," *Proc. AFIPS 1972 Fall Joint Computer Conf.*, AFIPS Press, 1972, pp. 1203–1224.

21. J.C. McGraddy and H.S. Lin, eds., *A Review of the FBI's Trilogy Information Technology Modernization Program*, tech. report, National Academies Press, 2004.

22. K. Beck, *Extreme Programming Explained*, Addison-Wesley, 1999.

23. M.A. Hirschberg, "Rapid Application Development: A Brief Overview," *Software Tech News*, vol. 2, 2002; www.dacs.dtic.mil/awareness/newsletters/technews2-1/toc.html.

24. M.C. Yatco, *Joint Application Design/Development*, 2004, www.umsl.edu/~sauter/analysis/JAD.html.

25. K.W. Derr, *Applying OMT: A Practical Step-by-Step Guide to Using the Object Modeling Technique*, Prentice Hall, 1995.

26. G. Booch, *Object-Oriented Analysis and Design with Applications*, 2nd ed., Addison-Wesley, 1993.

27. C. Kobryn, "UML 2001: A Standardization Odyssey," *Comm. ACM*, vol. 42, no. 10, 1999, pp. 29–37.

28. I. Jacobson et al., *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley, 1992.

Erratum

J. Frank Yates's biographical information in the printed version of "Decision(?) Making(?)," *IEEE Intelligent Systems*, vol. 20, no. 4, 2005, pp. 76–83, was inaccurate. The following one is correct:

J. Frank Yates, a past president of the Society for Judgment and Decision Making, is a professor of psychology and of business administration and marketing at the University of Michigan. He is also coordinator of the University of Michigan Decision Consortium, associate editor of the *Journal of Behavioral Decision Making*, and author of *Decision Management* (Jossey-Bass, 2003). His research focuses on decision making, with current emphases on decision management, judgment accuracy and processes, indecision, consumer choice, risk perception, and culture/decision-making interactions. Contact him at jfyates@umich.edu.

29. J. Wood and D. Silver, 1995, *Joint Application Development*, 2nd ed., John Wiley & Sons, 1995, p. 225.

30. P.C. Cacciabue and J.-M. Hoc, eds., *Expertise and Technology: Issues in Cognition and Human-Computer Cooperation*, Lawrence Erlbaum & Associates, 1995.

31. M.R. Endsley, B. Bolte, and D.G. Jones, *Designing for Situation Awareness*, Taylor and Francis, 2003.

32. K.J. Vicente, *Cognitive Work Analysis*, Lawrence Erlbaum Associates, 2000.

33. C.T. Bowles et al., "Knowledge Engineering: Applying the Process," *Proc. Human Factors and Ergonomics Soc. 48th Ann. Meeting*, Human Factors and Ergonomics Soc., 2004, pp. 2411–2415.

34. P. Koopman and R.R. Hoffman, "Work-arounds, Make-work, and Kludges," *IEEE Intelligent Systems*, vol. 18, no. 6, 2003, pp. 70–75.

35. M.R. Endsley and R. Hoffman, "The Sacagawea Principle," *IEEE Intelligent Systems*, vol. 17, no. 6, 2002, pp. 80–85.

36. D.D. Woods and S.W.A. Dekker, "Anticipating the Effects of Technology Change: A New Era of Dynamics for Human Factors," *Theoretical Issues in Ergonomics Science*, vol. 1, 2001, pp. 272–282.

37. R.R. Hoffman and D.D. Woods, "Studying Cognitive Systems in Context," *Human Factors*, vol. 42, no. 1, 2000, pp. 1–7.

38. S.W.A. Dekker, J.M. Nyce, and R.R. Hoffman, "From Contextual Inquiry to Designable Futures: What Do We Need to Get There?" *IEEE Intelligent Systems*, vol. 18, no. 2, 2003, pp. 74–77.

39. S. Guerlain and P. Bullemer, "User-Initiated Notification: A Concept for Aiding the Monitoring Activities of Process Control Operators," *Proc. Human Factors and Ergonomics Soc. 40th Ann. Meeting*, Human Factors and Ergonomics Soc., 1996, pp. 283–287.

IEEE Software

www.computer.org/software

IEEE Intelligent Systems

Advertiser/Product Index
January/February 2006

	Page No.
MIT Press	10
SMC-IT 2006	19

NEXT ISSUE

March/April: Self-Managing Systems

Information systems are becoming more and more complex, distributed, heterogeneous, and dynamic. Consequently, they're becoming too expensive to build and manage. The articles in this special issue will report recent results addressing the problem of designing self-managing systems. More specifically, it will contrast two competing approaches: self-organization, which is based on emergent phenomena, and reflection, which is based on the application of control loops and explicitly represented knowledge. While the latter approach is rather natural and accepted, approaches based on self-organization are less known in this context. They're motivated by the fact that we're surrounded by self-organizing systems, such as multicellular organisms, social insects, and ecosystems, that successfully solve self-management. Distilling the key ideas from these systems and incorporating them into information systems could lead to cheap, straightforward, and highly robust solutions.

Advertising Sales Offices

Matthew Bertholf
Phone: +1 785 843 1234 ext. 267
Fax: +1 785 843 1853
mbertholf@acgpublishing.com

Sandy Brown
10662 Los Vaqueros Circle, Los Alamitos, CA 90720-1314; phone +1 714 821 8380; fax +1 714 821 4010; sbrown@computer.org

For production information, conference and classified advertising, contact Marian Anderson, *IEEE Intelligent Systems*, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720-1314; phone +1 714 821 8380; fax +1 714 821 4010; manderson@computer.org; www.computer.org.