

## USING WORK-CENTERED SPECIFICATIONS TO INTEGRATE COGNITIVE REQUIREMENTS INTO SOFTWARE DEVELOPMENT

Jeffrey Wampler§\*, Emilie Roth†, Randall Whitaker~, Kendall Conrad§,  
Mona Stilson§, Gina Thomas-Meyers§ and Ronald Scott‡

§Air Force Research Laboratory, Wright-Patterson AFB, OH

~ Northrop Grumman Information Technology, Dayton, OH

† Roth Cognitive Engineering, Brookline, MA

‡ BBN Technologies, Cambridge, MA

As Cognitive Engineering (CE) becomes mainstream, methods are needed to better integrate the unique cognitive requirements of the target users of a Human Computer Interface (HCI) into software development requirements and testing. This paper discusses a preliminary work-centered specification describing the key cognitive HCI elements of a complex work system that enhance situation awareness (SA) and decision making. The specification provides traceability from the cognitive requirements obtained during knowledge acquisition to specific display elements in the final design. This specification approach can be applied to any CE methodology. We are applying it to a working prototype currently being integrated into an operational system. We have elicited feedback from developers of the operational system regarding the content and usefulness of the specification as it applies to their software development processes. This paper highlights critical aspects of our inaugural work-centered specification.

### INTRODUCTION

This paper describes an approach for specifying the human computer interface (HCI) requirements of complex displays designed using cognitive engineering (CE) and work-centered design (WCD) principles. In CE analysis and WCD, the content and layout of the information on the display and automation supporting the HCI is purposely designed to provide affordances which can reduce cognitive burden (e.g. Burns and Hajdukiewicz, 2004; Endsley, Bolte and Jones, 2004; Eggleston, 2003; Woods and Roth, 1988; Woods, 1995). However, the rationale behind these intricate designs may not be readily apparent to a software engineer who is typically not trained in HCI design, Human Factors and/or cognitive psychology.

During implementation, software engineers perform numerous trade-offs due to technical, cost and schedule constraints that account for the software functional requirements, but do not explicitly account for or even accommodate display elements critical to supporting cognitive work. The results are that the HCI is often degraded during implementation and key aspects of the design may not make it into the final product. Due to this “gap” from design to implementation, the HCI of complex work systems often do not reach their full potential. Especially in large systems development projects, this can result from deficiencies in human factors expertise within the development team (which is often geographically distributed), developers' knowledge of the target work domain, or developers' sensitivity to cognitive aspects of the target work.

This breakdown is further complicated due to the failure of software testing to account for cognitive work requirements. Traditional software requirements and testing are functionally based (Wiegers, 2000), and cognitive requirements such as critical display elements and their relationships are treated as

arbitrary features not directly derived from functional concerns. As a result, cognitive work requirements have not been explicitly stated or used in software testing.

Traditional HCI specifications such as those used to develop Microsoft Office products and web pages focus on screen layout and control interactions (Torres, 2002). They are rarely based on operator's cognitive processes in the course of his / her work functions. They define a design in terms of system software requirements to accommodate the role of a ‘user’ (operating the system per se) rather than a ‘worker’ (employing the system in the context of a task). This focus is readily apparent in specifications addressing system manipulations such as form filling and menu selections (i.e., what the user does *to* the system). Unless such specifications address what the worker does *through* the system, the resultant product may be easy to operate yet unsuited to supporting a given task. This prospective deficiency is most problematical where a decision support system design fails to accommodate critical features of the decision space as addressed by the decision maker.

The most sophisticated attempt to provide traceability from cognitive requirements to display elements is the Applied Cognitive Work Analysis (ACWA) methodology (Elm, 2004). ACWA is a top-down systems engineering approach that uses a modified abstraction hierarchy called a Function Abstraction Network (FAN). The FAN is the foundation and structure of the work representation from which cognitive work requirements and related display requirements are derived. However, the products of the ACWA method are generally intended to support display design and traceability, with less focus on delivery of a specification to software developers for implementation.

In the Goal Directed Task Analysis (GDTA) method (Endsley, et al., 2004), user tasks (including decision making

tasks) are represented in a goal structure to aid in designing HCIs for situation awareness (SA). Cognitive requirements are represented as questions and the information required to answer those questions. The actual display layout and relationships among display elements are left to the designer to implement using SA design principles.

Eggleston, Roth, Whitaker and Scott (2005) have described a high level framework for WCD specifications that attempts to convey the cognitive requirements of the HCI to large scale development teams. Their heavy emphasis on work context in formulating design rationale formed the basis for this approach, in which the HCI is specified in three levels: a summary work description, an overview description of the work support concept and a detailed design specification. The research described in this paper aims to flesh out the detailed WCD specification and build on selected cognitive requirements traceability aspects of the ACWA method and the question based decision requirements approach of GDTA.

### TECHNICAL APPROACH

In this paper, we will focus on the third level or detailed work-centered specification. Our process derives design rationale from analysis of the target work activity’s cognitive requirements, and then applies that rationale to derive implementation features and requirements. Our specifications must therefore reflect this line of derivation from cognitive requirements to system features.

We used a recent work-centered visualization design effort as a concrete case to stimulate development of concise work-centered specifications that can be meaningfully handed-off to a software implementation team from another organization. The project involved developing a prototype work-centered support system (WCSS) for an Air Force airlift operations center (Wampler, et al., 2005). The prototype tool was recently evaluated and found to reduce replanning time and errors while increasing SA and reducing the mental workload associated with replanning tasks as compared to existing tools (Roth, et al., 2006).

Throughout the development of the work-centered prototype, the design team produced informal products for communicating among themselves regarding the cognitive support requirements that needed to be met and how the evolving design would satisfy them. In this case, the products included an interactive software prototype allowing the team to inspect and test basic concept features. Once they achieved consensus on the basic design features, they produced materials with which they could present stakeholders the design concepts, their prospective benefits, and scenarios illustrating how the concepts would be employed. The process of creating the inaugural work centered specifications began after these actions had been taken.

We developed an initial specification which links the cognitive work requirements to display elements and packaged it as a deliverable to the airlift planning system developer in order to transition this technology to the customer. We evaluated the specification with software engineers with

respect to its ability to communicate the key features of the WCD and to provide useful development requirements and testing criteria. Our goal was to structure the requirements so they can be added to the software requirements documents and software test plans.

Our specification contains top level cognitive requirements derived from the work domain analysis, associated display elements which directly support those cognitive requirements and screen shots exemplifying the requirements in an implemented solution (Figure 1). An appendix is included in the specification that contains a list of mouse-over contents, a list of violation messages, intended usage scenarios and a list of the technical trade-offs conducted in the prototype environment that may be considered for the production system.

Our example focuses on a timeline visualization of military airlift missions in which the operator must consider many factors and constraints when schedule changes arise, for example, the aircraft must land when the airfield is open and not during quiet hours (a time when aircraft are restricted from landing and taking off due to noise). Visibility of these constraints relative to the mission itinerary is necessary for rapid and accurate replanning decisions.

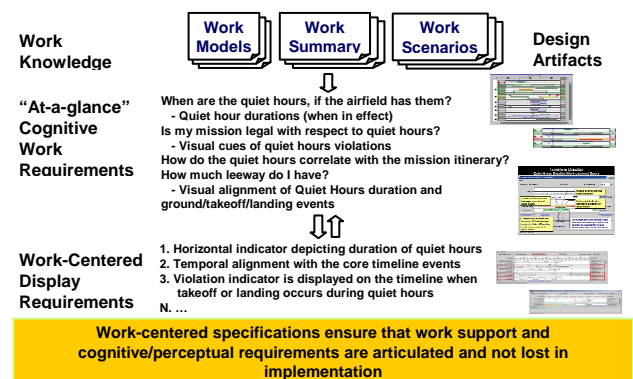


Figure 1. Detailed Work-Centered Specification Flow.

The top level cognitive requirements are expressed by the “At-a-Glance” Cognitive Work Requirements (CWRs). These requirements are essentially the questions that the user must be able to answer “at a glance” or without invoking other screens and performing mental calculations to attain SA and make efficient decisions.

At the next level, the CWRs are further refined into work-centered design requirements (WCDRs). WCDRs are the visual display elements and relationships between those elements which trace to the CWRs. WCDRs describe the interface elements (and relationships among these elements) necessary to permit a user to apprehend the state of the work subject matter. These requirements address such things as visualization issues (e.g., forms and colors) and cognitive issues (subject matter representational fidelity or clarity). The WCDRs are organized around the WCSS design principles (Eggleston, 2003) into four categories: central display, peripheral display, interaction requirements and automation requirements. The *central display* describes the primary

visualization in the center of the display that represents the visual form or aspects of the “work” to be performed. The *peripheral display* describes display elements that “frame” the problem in the work representation and allow the user to get situated. Peripheral display elements are typically labels in the workers’ terms and ancillary information contained in mouseovers. The *interaction requirements* are the selectable items on both the central display and controls on the peripheral displays involved in dynamic user manipulations of the interface. Finally, the *automation requirements* are display elements that are generated based on ‘intelligent’ computational processing (e.g., ‘rule-based’ processing) that augment central display elements to assist the user in attaining SA and conducting problem solving. Alerts generated from exceptions to business rules are the most common type of automation.

An important element of the ‘work-centered’ approach is that the results of the automated processes are displayed within work context visualizations, enabling users to better follow and evaluate the basis for automation outputs. This combination of central visualizations, peripheral labels/controls and automation (alerts) provide a coherent “picture” for framing the problems, attaining SA and making decisions.

**EXAMPLE WORK-CENTERED SPECIFICATION**

Our example involves the monitoring and replanning of airlift missions. Figure 2 depicts an airlift mission timeline visualization with mission itinerary (takeoff, landing, time in air (solid line) and time on ground (dotted line)) across the upper portion of the central display.

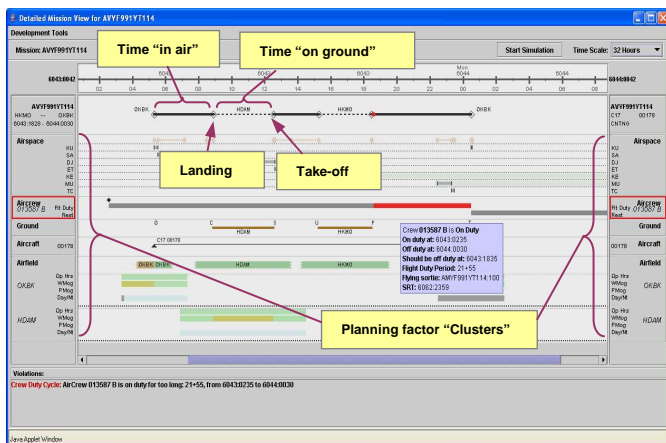


Figure 2. Airlift mission timeline prototype.

Related planning factors are grouped into “clusters” and displayed directly below the itinerary on a common timescale. This temporal arrangement allows the user to easily see the ramifications of making a change to the itinerary. The prototype contained the following clusters of planning factors: Country crossings (Airspace), crew rest (Crew), minimum ground times (Ground), Aircraft and Airfield availability. Our example focuses on the country crossing data presented in the Airspace cluster.

In order to fly across foreign countries, the Air Force must obtain formal permission from the country being overflown in the form of a Diplomatic Clearance or DIP. Legal overflight requires having a DIP whose timeframe of validity covers the period between entering and exiting that country’s airspace. The user monitoring the mission is alerted when any border crossing is outside of the DIP coverage so they can either adjust the itinerary to fall within the coverage or request an extension to the DIP coverage period.

In the specification, each cluster is described by both (a) the context for and characteristics of the work in which these are relevant as well as (b) the display element and its content. The specification for the airspace cluster is as follows:

**Description:** *The airspace cluster shows the durations spent in each country the mission transits. Country crossings indicate what countries are transited during a mission flight. Some countries require permissions to be in the airspace of that country, called a Diplomatic Clearance (DIP). Any available DIP information is provided visually per country.*

**Work Characteristics & Context:** *The airspace cluster supports work involved with determining if the mission has the proper DIP coverage. Any changes in the mission plan must account for DIP coverage. The user needs to know if any DIP is being violated and by how much time the violation involves. The user also has to be able to determine how much “slack” the DIP coverage allows in a replan.*

**“At-a-Glance” Cognitive Work Requirements (CWRs)**

- What national airspaces is this mission transiting?
  - Duration of projected time over each country (1, 2)
  - Labels of each country transited (7, 8, 9)
- When does the mission enter and exit each airspace?
  - Projected country entry and exit times (2, 4, 8, 9)
- What DIP coverage is in effect for each national airspace?
  - DIP permission duration per country transited (5, 10)
  - Slack in DIP coverage is graphically depicted (6)
- Are there DIPs violations concerning airspace?
  - Visual indication of airspace violation (11)
  - Time at which start of violation occurs (14)
  - Visual indication of violating country row (12)
  - Violation message (15)
- How does DIP coverage line up with national transits?
  - DIP permission duration visually aligned with projected country crossing duration (5)
- Does DIP coverage exceed or fall short of national over flight requirements, and by how much?
  - Duration of DIP violation is visually distinct. (13)

Table 1. CWRs for monitoring Country Crossings.

Our analysis determined that the user must be able to rapidly and accurately answer the questions in Table 1 to support the cognitive work involved in monitoring and replanning country crossings. The information required to answer each question is listed with a numbered cross reference to its WCDR (Table 2) and called out in a set of screenshots (Figures 3 and 4) to help developers understand the required display affordances for the CWRs.

\*This work is not subject to U.S. copyright restrictions

**Work-Centered Design Requirements (WCDRs)**

**Central Display:**

1. Country crossings are summarized at the top of the cluster with horizontal lines depicting transit durations.
2. Summary of country crossings has visual indication to depict entry/exit times
3. Vertical stack of countries being transited with visual distinction between rows.
4. Entering and exiting endpoints are clearly shown on beginning and end of country bars.
5. DIP coverage is shown as a coded sheath around the country crossing bar.
6. Relative durations of DIP coverage and country overflight are visually present for comparative inspection.

**Peripheral Display:**

7. Left & Right: Airspace cluster label, vertical stack of acronyms for countries being transited.
8. Mouseovers – Airspace: Entry/exit times of moused over country and countries crossed on the leg
9. Mouseovers – Country rows: Entry/exit times and DIP permission information

**Interaction Requirements:**

10. Clicking on country crossing summary bar expands/collapses to show individual country crossings and DIPs.

**Automation Requirements:**

11. Violations in the airspace cluster are visually coded by outlining the cluster label in red.
12. The violating country row's country label is turned red
13. DIP violation time duration is visually shown on country crossing bar.
14. The main timeline displays the time at which the DIP violation occurs
15. A violation message is displayed in the Violations portion of the display

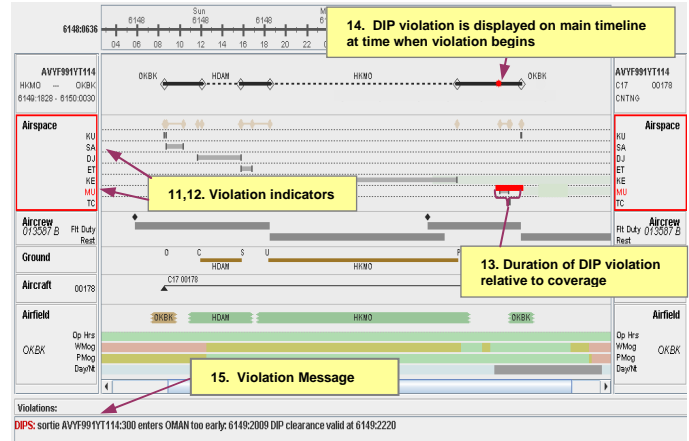


Figure 4. Example Screenshot Calling-out WCDRs for Monitoring Airspace and DIPs with Automated Alerting.

**INTEGRATING COGNITIVE WORK REQUIREMENTS INTO SOFTWARE DEVELOPMENT**

The developer's process was examined to identify potential insertion points for cognitive work requirements. In our case study, the developers of the operational system receive high level functional requirements from the Air Force airlift operations center customer. Initial concepts are prototyped to support the functional requirements and screen shots of the concepts are refined in Joint Application Development (JAD) sessions (Hoffer, 1999) with the customer. The final requirements are documented and test procedures defined to verify the requirements in a Standard Test Description (STD).

We propose to use the specification screen shots as a guide for JAD sessions. One of the shortcomings of streamlined software development approaches such as JAD is the lack of user involvement in the initial prototyped concepts (Carmel, 1993). Our approach allows the cognitive work requirements to become part of the initial requirements, to be vetted by customers and to become part of the final software requirements.

The STD is structured to capture screen level interactions and verification of display entities such as particular data, graphics or windows invoked through user interactions. Due to the user interaction and screen oriented nature of the STD, the WCDRs lend themselves to being verified in this manner. We propose that the STD procedures be extended to verify the interaction and display features of the WCDRs. Test procedures will be extended to account for both the normal condition and the alerted condition. The normal condition tests the display when the mission itinerary is valid and no automatic alerting is required. It will account for the WCD central display, peripheral display, and interaction requirements (items 1-10 in Table 2 and Figure 3). The alerted condition exists when exceptions to the itinerary's business rules are detected by the system and displayed to the user. It will account for the WCD automation requirements (items 11-15 in Table 2 and Figure 4). In our case study, two

Table 2. WCDRs for Monitoring Airspace and DIPs.

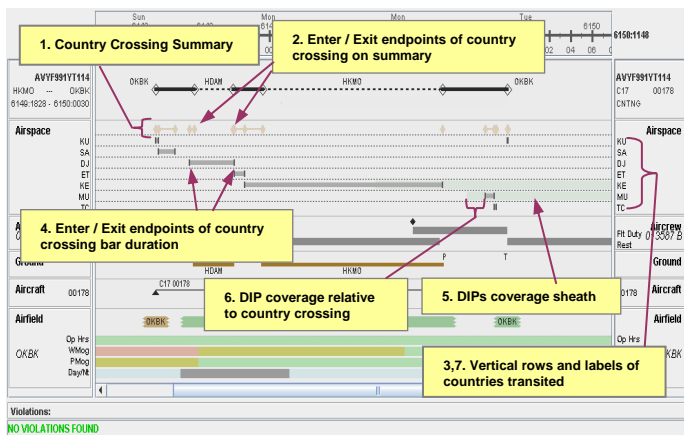


Figure 3. Example Screenshots Calling-out WCDRs for Monitoring Airspace and DIPs.

test procedures were written (normal and alerted) for each cluster on the primary display.

Finally, the specification will remain in its entirety as a stand alone document as a reference to the developers to provide traceability to the work context during development.

## DISCUSSION

Our analysis suggests that cognitive work requirements can be integrated into requirements and testing of large scale development systems. Our case study involved a fairly informal documentation process which allowed us to easily extend the current practice to accommodate cognitive requirements. It should be noted that each software development process is different and the challenge of inserting cognitive requirements into software development may become more difficult in more formal development efforts.

After the developers implement our prototype functionality into the operational system, we will conduct interviews to assess the usefulness of the specification and cognitive based extensions to the requirements and test procedures. The feedback from the developers will form the basis for further refinement of both the specification and its influence on software requirements and testing.

There are many advantages of work-centered specifications over traditional HCI specifications. First, the developers are provided with richer contextualization of the functionalities they are expected to implement and, are therefore, less 'isolated' from the original sources of design rationale traced to cognitive work. Second, the developers are better informed on the 'why' as well as the 'what', so there is arguably less likelihood of misunderstandings requiring laborious modifications, missed milestones, etc. Finally, the HCI is documented using a work-centered framework. The specifications are organized around the work-centered principles to aid the transfer of these principles to the final product. In our case, the work-centered principles are explicitly defined in the requirements and verified through inspections in the STD or similar test procedures.

Some of the disadvantages of work-centered specifications are the additional documentation plus the overhead needed to generate and maintain the specification. Also, the developers will require additional discussion / reading time to understand the specifications for implementation. It is our belief that the advantages far outweigh the disadvantages for complex work systems, where the features supporting cognitive work would otherwise not be a formal part of the development.

In conclusion, Work-Centered Specifications can help developers account for cognitive requirements by explicitly describing the key features of an HCI that must be intact for effective decision and work support. Fundamentally, this specification approach may result in a better integration of CE and Software Engineering. Techniques for specifying cognitive based HCIs will help CE based requirements become more mainstream and more easily integrated into software development and testing. Conversely, this allows system

developers to extend traditional software requirements and usability to include testing for effective cognitive work support.

## ACKNOWLEDGEMENTS

The work-centered research documented in this paper was sponsored by the Air Force Research Laboratory, Human Effectiveness Directorate, Wright-Patterson AFB, OH. Special thanks to Capt. Matt Forsythe for editorial comments and to Roger Robichaux of the Federated Software Group, Inc. for his insight into their software development process.

## REFERENCES

- Burns, C. and Hajdukiewicz, J. R. (2004). *Ecological Interface Design*. CRC Press.
- Carmel, E., Whitaker, R., and George, J. (1993). PD and Joint Application Design: A transatlantic comparison. *Communications of the ACM*, 36 (4), 40-48.
- Endsley, M.R., Bolte, B., and Jones, D.G. (2004) *Designing for Situation awareness: An Approach to User-Centered Design*. New York: Taylor and Francis Inc.
- Elm, W.C., Potter S.S., Gualtieri, J.W., Roth, E.M., & Easter, J. R. (2004). Applied Cognitive Work Analysis: A programmatic methodology for designing revolutionary cognitive affordances. In Hollnagel (Ed) *Handbook for Cognitive Task Design*. London: Lawrence Erlbaum Associates, Inc.
- Eggleson, R. G. (2003). Work-centered design: A cognitive engineering approach to system design. In *Proceedings of the Human Factors and Ergonomics Society 47th Annual Meeting*. Denver, CO: Human Factors and Ergonomics Society.
- Eggleson, R. G., Roth, E., Whitaker, R., and Scott, R. (2005). Conveying work-centered design specifications to the software designer: A retrospective case analysis. *Proceedings of the Human Factors and Ergonomics Society 49th Annual Meeting*. (pp.332-336). Santa Monica, CA: Human Factors and Ergonomics Society.
- Hoffer, Jeffrey A., George, Joey F., and Valacich, Joseph S. *Modern Systems Analysis & Design*, 2<sup>nd</sup> ed. Addison Wesley Longman, Inc., 1999. 485-498 pp.
- Roth E., Stilson, M., Scott, R., Whitaker, R., Kazmierczak, T, Thomas-Meyers, G., and Wampler, J. (2006). Work-Centered Design and Evaluation of a C2 Visualization Aid. *Proceedings of the Human Factors and Ergonomics Society 50th Annual Meeting*. (This Volume). Santa Monica, CA: Human Factors and Ergonomics Society.
- Wampler, J., Whitaker, R., Roth, E., Scott, R., Stilson, M. and Thomas-Meyers, G. (2005). Cognitive Work Aids for C2 Planning: Actionable Information to Support Operational Decision Making. In *Proceedings of the 10th International Command and Control Research and Technology Symposium* (June, 2005). Available online at: <http://www.dodccrp.org/events/2005/10th/CD/foreword.htm>
- Torres, R.J. (2002). *Practitioner's Handbook for User Interface Design and Development*. New Jersey: Prentice Hall.
- Wiegars, K., (2000). Karl Wiegars Describes 10 Requirements Traps to Avoid. *Software Testing & Quality Engineering*, 2(1), January/February.
- Woods, D. D. & Roth, E. M. (1988). Cognitive Engineering: Human Problem Solving with Tools. *Human Factors*, 30 (4), 415-430.
- Woods, D. D. (1995). Toward a theoretical base for representation design in the computer medium: Ecological perception and aiding human cognition. In J. Flach, P. Hancock, J. Caird, and K. Vicente (Eds.) *Global Perspectives on the Ecology of Human-Machine Systems*. Hillsdale, NJ: Lawrence Erlbaum Associates, 157-188.