

Work-Centered Support System Technology: A New Interface Client Technology for the Battlespace Infosphere

Robert G. Eggleston,
Michael J. Young
Air Force Research Laboratory (AFRL/HE), WPAFB, Ohio 45433, USA,
&
Randell D. Whitaker
Logicon Technical Services, Inc., Dayton, Ohio

Abstract

Information superiority is a strategic goal of the US Air Force. To achieve this goal the Air Force aims to produce a battlespace infosphere that will provide an unprecedented degree of connectivity and availability of raw data and value-added information for warfighter use. The essential challenge of the infosphere is to be able to provide the right information, at the right time, in the right form to enable warfighters to take effective, coordinated action. Although the infosphere's core web and agent technologies are clearly able to provide a heterogeneous infosphere, improved interface technologies are also needed to address problems of information overload and how to provide support to specific end-users without the support tools themselves becoming an impediment to task performance. We have developed a prototype Work-Centered Support System software client as a means to address these interface issues. The WCSS approach achieves effective support in a software agent environment by blending direct manipulation, work field organization, and decision, collaborative, and product development aiding in a manner that is tailored to both formal and informal characteristics of user work. In this paper we describe the philosophy behind and characteristics of the WCSS technology. We illustrate the technology with a discussion of an interactive WCSS prototype designed to improve support to military airlift mission planners at the headquarter level.

Key Words: Collaborative Support, Decision Aiding, Direct Manipulation, Graphical User Interface, Human-Computer Interaction, Information Overload, Interface Agents, Network-Centered Interface Design, Visualization, Work Centered Support System

Paper to appear in Proceedings of the IEEE, NAECON, 2000, Dayton, OH, 10-12 October, 2000.

Work-Centered Support System Technology: A New Interface Client Technology for the Battlespace Infosphere

Robert G. Eggleston,
Michael J. Young
Air Force Research Laboratory (AFRL/HE), WPAFB, Ohio 45433, USA,
&
Randell D. Whitaker
Logicon Technical Services, Inc., Dayton, Ohio

Abstract

Information superiority is a strategic goal of the US Air Force. To achieve this goal the Air Force aims to produce a battlespace infosphere that will provide an unprecedented degree of connectivity and availability of raw data and value-added information for warfighter use. The essential challenge of the infosphere is to be able to provide the right information, at the right time, in the right form to enable warfighters to take effective, coordinated action. Although the infosphere's core web and agent technologies are clearly able to provide a heterogeneous infosphere, improved interface technologies are also needed to address problems of information overload and how to provide support to specific end-users without the support tools themselves becoming an impediment to task performance. We have developed a prototype Work-Centered Support System software client as a means to address these interface issues. The WCSS approach achieves effective support in a software agent environment by blending direct manipulation, work field organization, and decision, collaborative, and product development aiding in a manner that is tailored to both formal and informal characteristics of user work. In this paper we describe the philosophy behind and characteristics of the WCSS technology. We illustrate the technology with a discussion of an interactive WCSS prototype designed to improve support to military airlift mission planners at the headquarter level.

Key Words: Collaborative Support, Decision Aiding, Direct Manipulation, Graphical User Interface, Human-Computer Interaction, Information Overload, Interface Agents, Network-Centered Interface Design, Visualization, Work Centered Support System

1.0 Introduction

It is generally recognized that the ability to collect, integrate, synthesize, manage, and utilize information is critical to the success of military operations. Indeed, as greater parity is achieved through the world in terms of the implements of warfare, the ability to achieve information superiority becomes even more important to successful military operations. Like e-commerce in the commercial sector, the Air Force is investing in software agent and web-based technologies as a means to improve connectivity and enhance information availability and utilization in warfare. The vision is to exploit these and other technologies to produce a Joint Battlespace Infosphere (JBI) as a means to obtain information superiority (SAB Report, 1999).

While the JBI may be envisioned in many different ways, the central challenge of the infosphere is to aid

the warfighter at all echelons by providing the right information, at the right time, in the right form to enable coordinated, effective, and efficient military operations. Stated more globally, the goal of the JBI is to improve and shorten decision and execution cycles.

Clearly web and agent technologies, in principle, are able to smooth over connectivity issues associated with legacy "stove pipe" systems, make accessible an unprecedented amount of data, support application sharing, and mediate long-distance collaborations. It is equally clear these same benefits of a web-based enterprise system for military utilization introduce new challenges for the design of user interface clients as software components of the JBI.

The potential for information overload obviously may be exacerbated by the JBI unless each warfighter has an effective interface client that focuses

information on work at hand. Perhaps less obvious is that software agents operating in the service of a user may induce confusion and invite errors in understanding. This phenomenon has been called automation surprise and is a known problem with highly automated flight management systems (e.g. Sarter and Woods, 1997). Thus, on one hand software agents may reduce cognitive burden by fusing data into information and automating certain work tasks, but on the other hand they may increase cognitive burden and degrade decision making and execution by operating in a hidden and largely independent manner. We need interface technology that is able to harness the power of web-technologies like software agents while avoiding or at least mitigating the unintended consequences of their use in a JBI.

We believe interface clients of the JBI should be designed as tailored *Work-Centered Support Systems (WCSS)*. The user interface must be regarded as a fully integrated support system, and not just a set of displays and controls or a graphical user interface that provides access to utilities and discrete application tools. A unified WCSS is needed to minimize off-task cognitive demands placed on the worker that can result from interacting with a heterogeneous set of information sources and work artifacts, and to improve the speed and quality of task decision making. In this paper we describe the WCSS technology mainly from a technology design perspective. We identify defining characteristics of the technology and illustrate how three design principles are embedded in a WCSS interface client used to support airlift planing and replanning work.

2.0 Work-Centered Support System (WCSS)

Work-Centered Support System(WCSS) technology may be regarded as both a software interface client technology and a design technology. As an interface client technology it specifies characteristics of a software architecture that dictate features of the control structure, object model, and methods used to implement a specific interface client. As a design technology it specifies design principles and conceptual structures that are used to define the form, content, interaction and overall behavior of the interface client. Further, it includes strategies for reducing work complexity for the user. From the user's perspective, a WCSS appears as a graphical user interface with embedded support tools in a work-centered organizational structure. The goal of the technology is to provide an integrated and tailored support system that is sensitive to the current context state and offers support to work in a flexible

and adaptable manner. This paper discusses WCSS technology features mainly in terms of design principles and concepts and how they are used to aid an employee cope with work complexity.

Broadly speaking, work may be defined as “the professional responsibilities and activities of a worker” (Vicente, 1999). Typically, work is thought of functionally as a set of tasks to be performed to accomplish a desired goal. The latter view invites the notion that a WCSS should support individual tasks that are needed to produce the outcome artifact. This is certainly true, but it is an incomplete view. The problem with this view is that the worker normally, partially or completely, defines new aspects of a task during the course of working. If we are going to be able to handle these emergent aspects of work, then we must also be able to support work indirectly, because we do not know before hand some of the tasks/activities that might benefit from support. The WCSS addresses this problem by using structural (organizational) means to represent relevant features of the work domain in the user interface client. This approach enables the WCSS to be compatible with flexible and adaptive changes made by the user to meet unexpected or unplanned contingencies.

As an interface client technology a WCSS constitutes the layer of a software application that is devoted to the user interface. In a web-based enterprise environment, it constitutes a software client module that connects a user to the network environment for the purpose of supporting the range of work assigned to the user. In the network environment, a worker may use a large array of software applications and databases to accomplish work goals. The goals of the WCSS technology are to provide a homogeneous and unified work interface to this heterogeneous and diversified environment, and to assist the user in completing work tasks in an efficient and effective manner. We translated these desired goals into five objectives for each WCSS product:

- I. Maximize explicit reference to task domain elements in the on-screen information display.
- II. Maximize cross-reference among information displays and analysis tools.
- III. Minimize procedural costs for accessing and retrieving relevant information.
- IV. Maximize effective fusion of data from multiple databases that need to be consulted to complete work tasks.
- V. Minimize perceptual, cognitive, and motor burden associated with identifying, seeking,

or interpreting relevant information and producing work artifacts.

Different approaches may be taken to meet these design objectives; however, to support adaptive behavior in an efficient manner a WCSS must have at least three global characteristics.

- Integrated direct and indirect methods of aiding within each support tool.
- Tailored and context-sensitive support.
- Single organizing framework across the support tool set.

By direct support, we mean a WCSS uses methods that call attention to a situation or problem relevant to the obtainment of a work goal or that actually accomplishes some aspect(s) of work for the user (once given authority) without further user involvement. A WCSS also provides support by the way it presents and organizes information. An important method used to implement this type of indirect aiding involves the use of a kind of state space representation of the work domain. Representing the domain of work as a state space in work-centered terms, for example, can create a visualization that allows the user to detect a problem without the need to provide a direct alert or cue. Both direct and indirect aiding methods can be used to complement each other. In some instances a direct alert may be distracting or unable to provide sufficient information about the situation that is more easily obtained from indirect aiding. In other instances a direct alert may be needed to insure a critical incident is detected in a timely manner. A WCSS attempts to blend the use of both direct and indirect aiding in a way that exploits the benefit of each type while avoiding the potential cost each may impose. Further, within a WCSS, both types of support may be aimed at aiding different aspects of work, such as information finding, decision making, collaboration, or product development.

The second global characteristic of a WCSS addresses the form of aiding it employs. Aiding needs to be sensitive to business rules and other factors that serve to limit or constrain what and how a worker can resolve a problem. The form of the aid, therefore, must be sensitive to the prevailing work context and state. This means an aid needs to be tailored to *local conditions*. Thus, for example, decision support may be provided in one situation by presenting a list of options and in another by using a probe question. Like the intelligent employment of direct and indirect aiding types, the use of tailored and context sensitive forms of aiding also increases

the odds the aid itself will not impose a work cost during its employment. In other words, there is a lower likelihood the user will have to devote time and cognitive resources focused on how to understand the aiding or use the aiding facility rather than being able to naturally understand or deploy the tool immediately on the work problem. The major challenge in designing a WCSS is to figure out how to provide tailored, context-based support in a way that is responsive to flexible and adaptable behavior of the user.

The third characteristic addresses the issue of how to make an interface client consisting of individually tailored support tools congeal into a well-formed, single support system. The basic solution is to use a single organizing framework that spans all the support tools that comprise the WCSS system. The use of a desktop metaphor serves as an organizing framework for a basic human-computer interface. While this framework is tool centered (focused on the computer) a WCSS needs a framework that is work-centered (focused on the nature of work for a particular user). An organizing framework acts as a backdrop for all display representations used in the WCSS and thereby reduces its complexity for the user. Further, a single framework provides a referent that can be easily incorporated into the user's mental model, and this serves as another avenue by which work complexity can be reduced.

3.0 WCSS Design Principles

All of these global characteristics are intended to facilitate ease of use of the WCSS in a manner that minimizes work complexity as experienced by the user under dynamically varying task conditions. In an effort to achieve these characteristics, we have formulated and used three design principles that are manifested directly in a WCSS interface client. The three design principles are:

- ◆ Use a single ontology that expresses work in terms familiar to the user community.
- ◆ Represent the work *domain* to provide indirect aiding.
- ◆ Decompose complex work into smaller chunks by employing the Focus Method.

3.1 Work-centered ontology. A single ontology pervades the WCSS concept. An ontology is the set of terms, meanings and relations between terms that captures or represents some subject matter. A task or work domain ontology is a taxonomy of terms and associated meanings used by a worker to think about work tasks and problems and to produce work

products. An ontology acts as the organizing framework for the entire support system. It provides the constructs to be represented on the display surface of the interface as objects, icons, and terms used; hence the ontology is made visible in a WCSS.

It seems obvious that work support offered by an interface client is more readily understood and useful if it is expressed in terms the user currently uses to think about issues and problems and to produce work artifacts. It follows logically that a consistent, work-centered taxonomy with associated meaning (i.e. an ontology) will facilitate use of the support system. The user does not have to learn new, tool-centered terms, and he or she will not need to change referential or inferential focus during the course of working a problem and using the tool(s). But the significance of this design principle grows in importance when the WCSS is an interface client to a large, heterogeneous enterprise system. As we have indicated, typically each database and available software application will have its own unique vocabulary, syntax, and semantics, hence its own ontology, in an enterprise network system. A work-centered term expressed in the WCSS, therefore, may have different meanings in each of these different ontologies. This fact, plus the sheer magnitude of languages covered in databases that may need to be learned, can detract from user performance and complicate the use of a WCSS if a single, work-centered ontology is not employed and semantically mapped to information sources.

Another dimension to the problem is the issue of automation surprise. Because databases and applications internal to the enterprise system may employ machine intelligence in some form, there is ample opportunity for the automation to perform in ways that are not clear to the user. The resulting behavior diverts attention from the work to the tool, creates surprise, confusion, and may cause error in performance.

We have addressed these issues in the design of a WCSS by developing a work-centered ontology that is expressed in the interface representations and by exploiting a three-dimensional model of software agent interaction to attack the problems of semantic mapping (e.g. from databases to the interface client) and automation surprise. The three-dimensional model provides concepts for 1) how to arrange levels of agent autonomy for interface agents near the surface of the WCSS to those distributed within the infosphere, 2) how to address differential unity of software agents, and 3) how to provide a consistent work-centered ontology to coordinate user-agent

interaction. A more comprehensive discussion of a WCSS from a software agent perspective is provided in Young, Eggleston, and Whitaker (2000).

3.2 Work domain representation. The second design principle for a WCSS deals with the representation of the work domain. The focus of this form of representation is on the “field” where work takes place. It is a structural presentation of the interacting variables that influence the production of work products. These variables are spatially arrayed in a manner that supports visualization of a work state (see footnote) from a variety of perspectives. Variables may include abstract concepts as well as physical ones. The abstraction-decomposition work analysis method used by Rasmussen, Vicente, and others is an attempt to capture relevant variables that are used to define a work domain (cf., Eggleston, 1998; Rasmussen, 1986, 1994; and Vicente, 1999).

By presenting a visualization of the work domain, a user is able to use perceptual pattern detection and recognition to stimulate cognitive awareness of possible solutions that are sensitive to the constraints contained in the work field. It is in this way that a work domain representation serves to indirectly aid the user in solving difficult problems.

It is useful to note that whereas a work ontology is an organizing device that operates at the conceptual or knowledge level, the work domain state space operates at a perceptual level. By making visible the variables of the work field and not just the ones directly related to a specific issue, the user is able to see both a properly bounded solution space, and the available degrees of freedom to resolve the problem. This contrasts with an interface design approach that concentrates solely on the work product to be completed. When a product model is the only expression made visible in the interface, the product artifact is effectively lifted out of the analysis and decision making context.

3.3 Decompose work using the Focus Method. The final design principle addresses the issue of segmenting work into smaller units as a means of aiding the user to help cope with task complexity. The typical strategy is to decompose work into a set of functions that, if necessary, are further decomposed to subfunctions. The user interface is then often designed to support each function, essentially on a function by function basis. Several functional clusters may be presented on the same display surface, or they may be on separate surfaces. This approach may work fine when interactions among tasks are nonexistent or quite low. But when

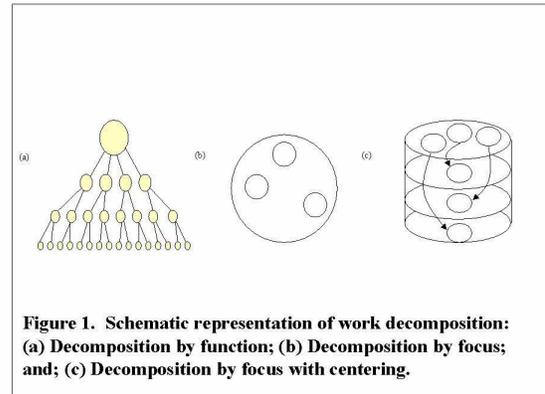
many factors interact, individual task decisions generally cross functional boundaries, and the user must handle this added complexity without any support from the user interface system. As a result, work performance suffers. The question, then, is: How can we gain the benefit of decomposing work into smaller units in a manner that better enables the solution of problems defined by multiple interactions?

WCSS technology is predicated on the use of a different type of work decomposition to better handle this problem. We call our technique the *Focus Method*. The basic principle incorporated in this method is to “decompose” work into work domain focus areas. A focus area is similar to a functional cluster, but it is different in that 1) the work domain as opposed to a function or task is expressed, and 2) the entire domain is included in the expression, not just a subset. The Focus Method may be thought of as moving the center of attention to a region of the work domain that lies at the heart of interest for certain problems or issues that must be resolved as part of the work process. Phrased another way, the goal is to highlight portions of the work domain as a whole instead of presenting it piecemeal. It is an emphasis area. In keeping with the second design principle, a more detailed visualization is used to present the focus area. The remainder of the work domain is expressed in the periphery of the WCSS display at a lower level of resolution. More resolution can be readily achieved when needed in this region of the work domain through the use of direct-access drill down and reach back to reveal more enriched forms of information and work tools.

It should be clear this approach to work decomposition is quite different from segmentation by function. With the deployment of the Focus Method a WCSS is able to enjoy the benefits of breaking work into smaller chunks without sacrificing direct engagement with the full relevant set of interacting variables contained in the work domain. The complexity of work is reduced without inadvertently and inappropriately reducing the problem complexity.

The distinction between these two approaches of work decomposition can also be seen conceptually with the use of a simple schematic (see Figure 1). A functional decomposition follows a whole-part strategy to parse work into a set of functions or tasks. It results in an inverted tree structure, with each node being a function. When many factors interact in work, the worker must traverse multiple branches of the tree to bring together all the relevant pieces,

which can place a heavy burden on cognitive capacity.



In contrast, the Focus Method operates on the work domain and uses an attentional strategy to achieve work decomposition. Attention is centered on a local aspect or region of the work domain. It is represented schematically in two ways (see Figure 1 b and c). A “compiled” form (Figure 1b) shows the focus areas (small circles) in a work domain represented as a circle in the plane. To better visualize the centering of attention, we can expand the work domain model to form a cylinder where slices through it represent the work domain from the unique perspective of a focus area that has been “centered” in terms of detail provided in the display visualization (see Figure 1c).

A work-centered ontology, work domain representation expressed as a state space, and the concept of focus areas constitute a nested set of methods that help to make work manageable and reduce cognitive complexity through organizational means. The power of using a comprehensive organizational strategy in the design of an interface client is often overlooked. To make these characteristics of a WCSS more concrete, we illustrate a WCSS as an interface client to an airlift planning and command and control enterprise environment. Selected portions of the WCSS are briefly presented and discussed in term of the three design principles.

4.0 A WCSS Prototype for Airlift Planning

A WCSS prototype has been developed to support military airlift planners responsible for authorized recurring missions. For this class of planner, known as Channel Planners, the basic unit of planning work is known as a route set. A route set is defined by a set of missions that complete a circuit. It includes all

aerial port stops to on-load and off-load cargo, passengers, or both, including the aircraft's point of origin. To plan or replan a route set or part of a route set (e.g. mission leg), the planner must consider many different factors. These range from issues about the cargo, available aircraft, aircrew, permissions and clearances, funding, etc. Channel route sets are formulated up to several months prior to execution. As a result, many events can happen during this time period that make it necessary to modify the route set. For example, a restriction can be placed on an airfield that conflicts with the planned date/time of the mission. During any duty day, the Channel planner must both plan new missions and detect events requiring replanning, such as a field closure.

For the purpose of illustration, we will consider a particular replanning problem that emerges when the number of aircraft that converge on an aerial port within a given time window exceeds that port's parking capacity. This is known as a Maximum on Ground (MOG) problem. We will look at the MOG problem in terms of work for a Channel Planner.

Figures 2, 3, 4, and 5 show selected panes from the WCSS. The full system includes additional interactive analysis tools, but these panes will suffice to make concrete important features of the WCSS technology. A complementary discussion of this interface client system is presented in Young et al. (2000).

A work-centered ontology serves as an organizing framework for the entire WCSS. This ontology can be seen in all of the analysis tools by the terms that are used. Each term is used in the work vocabulary of actual Channel Planners. To see the result of employing this and the other design principles, we will trace how a user may exploit the WCSS client for a "pop up" MOG event.

Let an airlift mission be posted to the enterprise system by another planner responsible for a different category of mission type. The WCSS MOG-alert agent (direct aid) detects that this newly posted mission induces a MOG problem at a specific aerial port for a route set (mission) that has been planned previously by our WCSS user. The alert is issued to the WCSS and expressed in the form of an icon on the Windows Task Bar. The planner may hover his/her mouse over the alert icon to learn the port involved and may click the icon to see the source of the problem. A mouse click opens the Conflict Summary Tool that shows the missions that result in a MOG problem at base x.

The Conflict Summary Tool (see Figure 2) is a context sensitive support aid that presents the problem to the mission planners involved in the conflict. The tool provides a limited but efficient and context sensitive way to support decision making and collaboration to resolve the problem. Without the tool, the problem is simply inherited by the planner with the lowest priority mission. With the WCSS tool, the planner whose action induced the impending problem finds out about the MOG condition in the context of his current planning activity. As a result, when the alert is detected, this planner has a current mental model of his planned mission and will often know where the plan can be modified without disturbing the route set. Because the information about this mission is fresh when the conflict is broadcast to him or her, little effort may be required to solve the potential problem and thereby eliminate conflict with other planners, who would have to become acclimated to the context before problem resolution could begin. The planner whose mission "caused" the problem can signal his or her intention by clicking the 'recut' box in the Conflict Summary Tool. Hence the WCSS in this situation provides a form of collaboration and decision support. More extensive forms of collaboration are provided by items in the periphery of the tool, and other information is also readily available to further aid decision making. All of these features are subtle, as they blend into the work state representation.

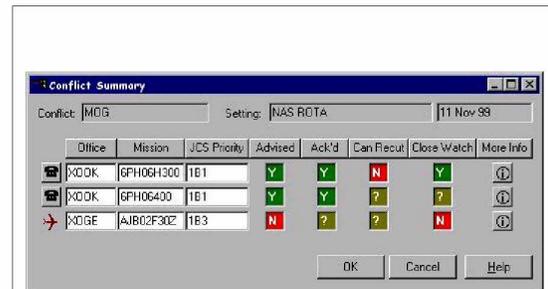


Figure 2. Conflict Summary Collaborative Decision Making Tool (meta work focus).

Assume that the lowest priority planner must resolve the MOG problem. In this situation, the planner will have to bring into focus the route set involved. (It is not likely he or she would be working on this route set at the time.) From the Conflict Summary Tool, this planner could focus on his/her mission's role in the conflict by selecting (via mouse click) the Port Analysis Tool (see Figure 3).

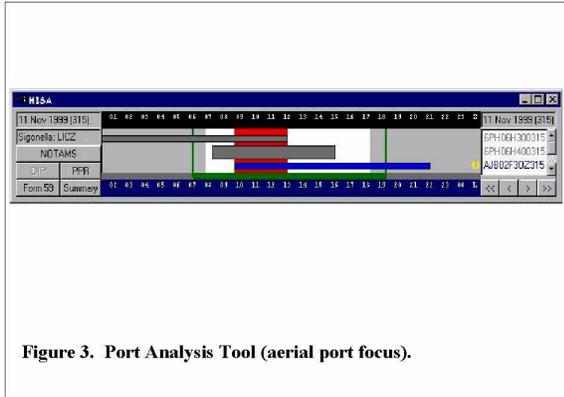


Figure 3. Port Analysis Tool (aerial port focus).

The aerial port focus (i.e. work Decomposition by Centering) is made clear by the visualization in the center of the tool. Horizontal bars of different thickness (to reflect narrow and wide bodies) portray ground-time profiles for aircraft at the port. The red highlighting, denoting the time period for which the number of aircraft exceeds parking spaces, represents the projected MOG condition. Given this visualization, the planner may see a simple solution. If not, data on other factors relevant to resolution can be readily invoked through the Port Analysis Tool’s peripheral buttons. These permit rapid drilldown to information on permission request requirements in effect at this port, diplomatic clearance request status, and detailed information about the aircraft and crew. Such information allows the planner to visualize what adjustment is feasible and where degrees of freedom are restricted. Course of action feasibility can be interactively tested, because additional alerts are generated if a selected alternative induces either upstream or downstream problems in the route set. To address the entire route set, the planner needs to center on a larger portion of the work space. He or she can do this by invoking the multi-port display (see Figure 4).

The Multi-Port Tool replicates the Port Analysis Tool for all ports involved in a route set, or a subset if selected by the planner. In this way interactions across ports can be easily visualized. As adjustments are made in the plan, a conflict alert agent provides direct aiding. Note that the “what if” analysis is carried out in the work-centered taxonomy, as was true for all other analyses and information collection activities.

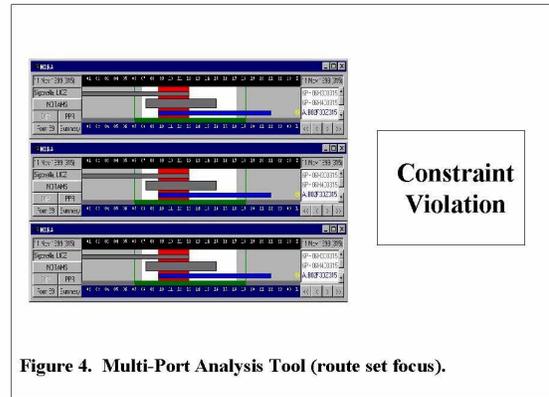


Figure 4. Multi-Port Analysis Tool (route set focus).

Now assume that while the planner was working this MOG problem, new alerts were issued to point out the possible need to replan other missions. The planner may briefly suspend work on the MOG problem to review these alerts and decide to leave them pending until the MOG problem is resolved. At this point, he calls up the Alert Summary pane that provides a queue of pending replanning problems (see Figure 5).

One feature of this tool is it allows the user to customize his “To Do” list. Some of the items may need immediate attention and others may not. Based on experience, the planner may anticipate some of the items will be overcome by events, making it advisable to defer working them. The Planner may set alarm schedules to indicate under what conditions and when and how often to be prompted about a particular mission that may need replanning. The Alert Summary tool, therefore, aids self-organizing meta work, in addition to providing direct information about pending replanning work. Self-organizing work represents another focus or center of the work domain for the Channel Planner.

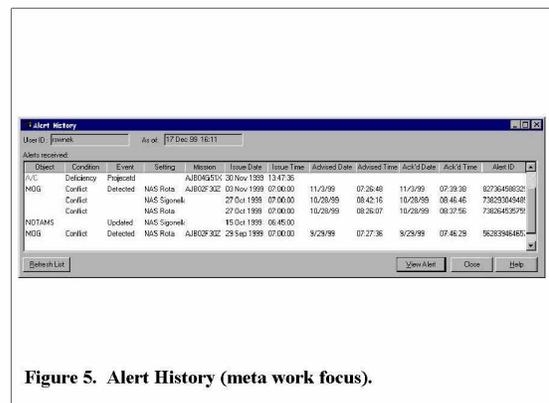


Figure 5. Alert History (meta work focus).

This brief walk through of a WCSS can only provide a flavor of the features and principles used in the development of WCSS technology. With the exception of the Multi-Port Tool, all of the tools have been implemented in the prototype system. The prototype has been demonstrated in a simulated enterprise environment that included agent-agent brokering and access to legacy databases. However, interaction between planners, and the software interface to legacy planning applications, remains to be implemented. Accordingly, aspects of the WCSS client that depend on the use of these assets have not been developed.

5.0 Summary and Conclusions

The interface client to an enterprise system needs to be regarded as a support system. For complex work, the type and form of aiding needs to be adaptable and flexible enough to meet emerging circumstances. WCSS technology explicitly considers the user interface as an integrated support system. It uses design principles and methods that are able to provide flexible and adaptive support in a manner that allows the user to stay focused on work, rather than having to periodically shift focus onto the tool itself. Thus, the complexity of work experienced by the user is reduced. The three principles of direct and indirect aiding, work domain representations using work ontology, and work division by the Focus Method all play together to make a well-formed, user-centered interface client.

6.0 References

Eggleston, R.G. , 1998. Cognitive engineering: The latest fad or a true step forward as an approach to complex multi-person system analysis and design? *In NATO Research and Technology Organization Meeting Proceedings 4, Collaborative Crew*

Performance in Complex Operational Systems, RTO-MP-4, December, 1998.

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

Rasmussen, J., Pejtersen, A.M., and Goodstien, L.P., (1994). *Cognitive systems engineering*. New York: John Wiley & Sons.

SAB, 1999. Building the Joint Battlespace Infosphere, Vol 1: Summary. *United States Air Force Scientific Advisory Board Report*, SAB-TR-99-02, December, 1999.

Sarter, N.B. and Woods, D.D., 1997. Team play with a powerful and independent agent: Operational experiences and automation surprise on the Airbus A-320. *Human Factors*, 39(4), 553-569.

Vicente, K.J., 1999. *Cognitive work analysis: Toward safe, productive, and healthy computer-based work*. New Jersey: Lawrence Erlbaum Associates.

Young, M.J., Eggleston, R.G., and Whitaker, R.D., 2000. Direct manipulation interface techniques for interaction with software agents. Paper presented at the NATO/RTO symposium on Usability of Information in Battle Management Operations, Oslo, Norway, April 2000.

Footnote. As used here, a domain state space presents a spatial layout of a work domain model that is expected to be necessary and sufficient to fully represent the field of practice. While related to a mathematical state space representation, a domain state space visualization is different. It does not include variables as coordinates to frame the space in the interface expression. Coordinates are used in the underlying analytical model.