

**A DEVELOPMENT ENVIRONMENT AND METHODOLOGY FOR THE DESIGN  
OF WORK-CENTERED USER INTERFACE SYSTEMS**

Wayne Zachary

CHI Systems, Inc  
Lower Gwynedd, PA

Robert G. Eggleston

Human Effectiveness Directorate  
Air Force Research Laboratory

ABSTRACT

*Work Centered Support System design represents an approach to the development of user interface application as an integrated, multi-faceted active and passive aiding system. Several successful instances of WCSSs have been developed using largely labor intensive hand analysis and software coding methods. Here we describe a well-formed analysis, design, and implementation development environment, called the WIL Application Toolkit (WAT), as a work-centered development support aid for one type of WCSSs. The design principles and architectural properties of the WAT are discussed in the context of a design methodology. These aiding tools for interface system development is expected to improve WCSS design, shorten develop time, and improve sustainability of released interface products.*

## **Introduction: Work Centered Support Systems**

Traditionally, the user interface to machines and systems has been designed from the perspective of accessing machine/system functionality (rather than supporting human work). Human Factors professionals have repeatedly advocated a more user-centered approach to interface design, e.g., the use-centered approach of Flach & Dominguez (1995) or the work-centered philosophy of Rasmussen & Vicente (1990), Vicente & Rasmussen, (1992). In distinction to tool-centered design, which tends to define features of the interface in terms of tool properties, a work-centered approach seeks to define and organize the interface in terms of work factors. Work-centered design leads information representations and interface features that more directly relate to, and better support, the work the user is trying to perform, and ultimately enable intelligent, flexible, and adaptable work behavior. Such a philosophy seeks to represent properties of the work domain (as opposed to just task steps) as a way of opening the interface for the user to constructively deployment flexible and adaptive strategies to meet the inevitable uncertainties and disturbances that occur as work unfolds.

With this work-centered view of interface design, it is possible to treat the user interface as not just a (passive) window into the underlying system functionality but rather as a multi-faceted support system that is designed to aid the user in work performance (Zachary, 1985; Young et al., 2000). The Work-Centered Support System (WCSS) paradigm developed by Eggleston and colleagues (Eggleston et al., 2000) is an example design approach that integrates all a range of direct and indirect forms of work support (e.g. decision aiding, collaborative aiding, product development aiding, and work management aiding) within a common framework. It exploits a unified set of direct, indirect, passive, and active aiding methods, to provide context sensitive support for efficient, flexible and adaptive work performance.

Work-centered approaches, and more specifically the WCSS approach have resulted in more effective user interfaces, there is a critical need for methods and tools to make work-centered interface design and development more cost-effective and replicable. Initial progress toward this goal is reported in this paper.

## **Thesis: Creating General Design Methods from A Common Reference Model**

The approach taken to creating design methods and design/development tools for WCSSs was to work from an explicit reference model, consisting of a well-defined space of functionality and a well-defined underlying reference architecture. The explicit functional space bounds and defines the specific types of work-support that the WCSS can provide to its end-user. The reference architecture defines the ways in which these support functions can be implemented, using a common computational infrastructure supplemented by specific types of work-domain knowledge. A highly general reference model of this type was abstracted from a series of related WCSS applications that used a common architecture and design approach. Called the Work-centered Infomediary Layer or WIL, it defined a broad but well-bounded space of WCSS capabilities. A general WIL design method was created by abstracting and structuring the underlying WIL design approach. The WIL reference model and design methods are reported here. A key premise is that the WIL reference model and design method are sufficiently well-defined that they can be turned into tools to support the design and development of future WIL-type WCSSs.

## **WIL Reference Model**

WIL represents a class of work-centered support systems that incorporates instances of some or all of the work-support functions described in Table 1, using the generic computational architecture pictured in Figure 1. The central feature of the WIL concept is an *explicit representation of context*, which includes the overall dynamic work context, the underlying infosphere context, and even the WIL's own internal context (i.e., awareness of what it is doing, trying to do, has already done, etc.). Context is critical because work actions, decision-making, and information needs are strongly based on an internal representation of the current work context. This suggests that the work needs and goals of the user of WIL (or any interface) are frequently understandable only within the local work context in which they arise.

Will uses the principle of a shared relationship among the user, application resources, and task environment as the basis for establishing an understanding (by deploying reasoning mechanisms) of the work domain as a current work context. The context is then used to construct various forms of work-centered support, customized to the current situation.

From a processing perspective, the WIL reference architecture is divided into three layers. In the front-plane layer, the human user interacts through a series of direct interaction objects (i.e., widgets), whose look and feel are tied both to the work domain and the interface functionality they present.

Depending on the details of the specific work domain, the WIL application also needs to interact with various algorithmic components, databases, data streams, etc. This allows the WIL interface to avoid being the end-point to a 'stovepipe', connected to only a single sensor, database, or system. Rather, the back-plane contains properties that allow a WIL interface to interact with multiple components beyond its rear surface.

In between the front plane and back-plane of Figure 1 lies a set of processing capabilities that support end-user work in different combinations of ways, depending on the situational needs of the work domain addressed. It is this mid-plane layer that is the main locus of the work-centered processing within WIL as a WCSS architecture. Within the mid-plane, a WIL interface simultaneously and continuously performs three general tasks:

- understanding what work needs to be done (an understanding task);
- interpreting and determining how that work context relates to the user's (often implicit) work goals (the action/need assessment task); and
- determining how to support the human in achieving those needs and goals (the tailored support task).

The WIL architecture enables these functions through three infrastructural processes which automate understanding (by context representation), action/need (by context-driven interpretation techniques), and tailored support (by providing objects in the front surface of the interface that act as instances of general work functions as listed in Table 1.).

## WIL Design Method

Work centered interfaces, by their nature, require a work-centered analysis and design process. The prior WIL interfaces had made informal use of a design method which was formalized and refined to the point that it could be at least partially automated. The WIL design method is a structured process that breaks design into a series of well-defined steps, each of which systematically move the designer/analyst from features of the work domain toward specific system/interface features that will augment human performance in that domain. Each

step is supported with specific data-capture protocols and abstracted guidelines, taxonomies and/or rules that lead to the production of a well-defined output; this output is then used as an input to the next step of the process. By the end of the process, a detailed design has been produced. Importantly, the sequence of intermediate products also creates a traceable design history that supports downstream maintenance and evolution of the system, as the rationale and interrelations underlying the design are retained and are available for subsequent maintainer teams. The overall WIL design method created is shown in Figure 2, (for a more detailed discussion of each step, see Zachary et al, 2002).

## Findings

The WIL reference model was used to create several WCSSs (Zachary et al, 2002). Despite their successful work-centered design, all shared several pragmatic limitations, particularly regarding the way in which they were constructed. They were all hand-coded, and required extended interactions with work domain/subject-matter experts. They had development cycles that, while reasonable in a research context, were unacceptably long and costly from a commercial perspective.

A major goal of fully defining the WIL reference model and design method was to eliminate these limitations by creating software tools to support and streamline the process of creating WIL interfaces. These tools are termed the WIL Application Toolkit or WAT; in some sense, WAT represents a Work-centered Support System for the creation of WIL applications. The conceptual organization of WAT is summarized below.

Overall, the WAT supports the WIL design process through three activities:

- providing data and knowledge about the work domain, i.e., subject matter expertise (Steps 1 and 4 of Figure 2)
- making interface and software design/engineering decisions (Steps 2,3, and 5); and
- integrate and customizing the software code produced by the WAT (Steps 6 and 7).

This suggested three different types of WAT users. Subject matter experts (SMEs) provide work domain knowledge. A user interface engineer makes the interface and software engineering design decisions, a (pure) software engineer performs any needed code customization. Pragmatically, while it is possible for the SME to enter data directly into the WAT, it was deemed better for this entry to be mediated by the interface engineer.

The WIL design process also identified two different functions:

- analysis/design, which captures work domain information and functional design information, and
- software specification and development, which translates the analysis/design information into actual software.

These two functions are also differentiated by their immediate products or outputs.

Analysis/design functions all concern work-domain and functional design (i.e., non-executable) content, while the products of the software specification and development functions all involve executable (i.e., software) content. In addition, this grouping segregates the functions of WAT which potentially need to be accessible to the subject-matter-expert type of user (the analysis/design functions only), from those that need to be accessible to the interface engineer type of user (the software specification and development functions).

The resulting organization and top-level design for WAT is shown in Figure 3, with two component tools, an analysis/design tool (which captures its products in a design repository), and

a software specification and development tool (which captures its products into a software repository). This separation also created a potential problem, in that the two tools represent intermixed steps in the design process. Thus, they could not be used in a strictly sequential manner, but rather needed to be used in an interleaved manner to carry out the seven design process steps in the proper order. Thus, the user(s) will log in, enter data and otherwise interact with the tool, and log out, many times. As long as the two component tools are interrelated by shared data repository access, the content in the repository itself (i.e., what has been entered so far versus what has not) can be used to indicate to each of the tools whether it is possible to proceed to later steps in the process. Future steps in this on-going research effort are focusing on implementing the WAT and using it to develop a series of WIL applications in the Air Force.

## **Discussion**

WCSSs that emerge from this design methodology represent a context-based approach to user interface system design. A shared understanding of the context provides the foundation for providing a wide set of tailored aiding as properties of the interface system. Because local context, rather than task steps, is represented and reasoned about, the interface is poised to support unexpected events and other emerging contingencies. Hence, WCSSs are expected to be very robust in their ability to provide support.

The WAT development environment and methodology is crucial to our ability to meet reduced time lines and product sustainment requirements. A cost-benefit analysis identified several key benefits that WAT would bring to the WCSS development process to achieve this goal:

- reduce needs for SME/operational personnel time in design engineering;
- reduce the effort required for work-domain knowledge acquisition and engineering;
- shorten the time spent in design, development, and debugging of the software needed to implement the WIL application, because of the use of pre-validated high-level building blocks (from the WIL reference architecture), and the resulting reduction in the number of hand-coding components requiring more detailed validation and debugging.

Over the next several months, analysis and demonstrations of WAT components are planned.

## **References**

- Eggleston, R.G., Young, M.J., and Whitaker, R.D. (2000). Work-Centered Support System Technology: A New Interface Client Technology for the Battlespace Infosphere. *Proceedings of NEACON 2000, Dayton Oh, 10-12 October, 2000*, pp 499-506.
- Flach, J. M. and Dominquez, C. O. (1995). Use-centered design: Integrating the user, instrument, and goal. *Ergonomic and Design*, pp, 19-24.
- Rasmussen, J. and Vicente, K.J. (1990). Ecological interfaces: A technology imperative in high-tech systems? *International Journal of Human-Computer Interaction*, 2, 93-110.
- Vicente, K.J. and Rasmussen, J. (1992). Ecological interface design: Theoretical Foundations. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-22, 589-606.

Young, M., Eggleston, R., & Whitaker, R. (2000). Direct Manipulation Interface Techniques for Users Interacting with Software Agents. Proceedings NATO/TRO Symposium on Usability of Information in Battle Management Operations. Oslo, Norway.

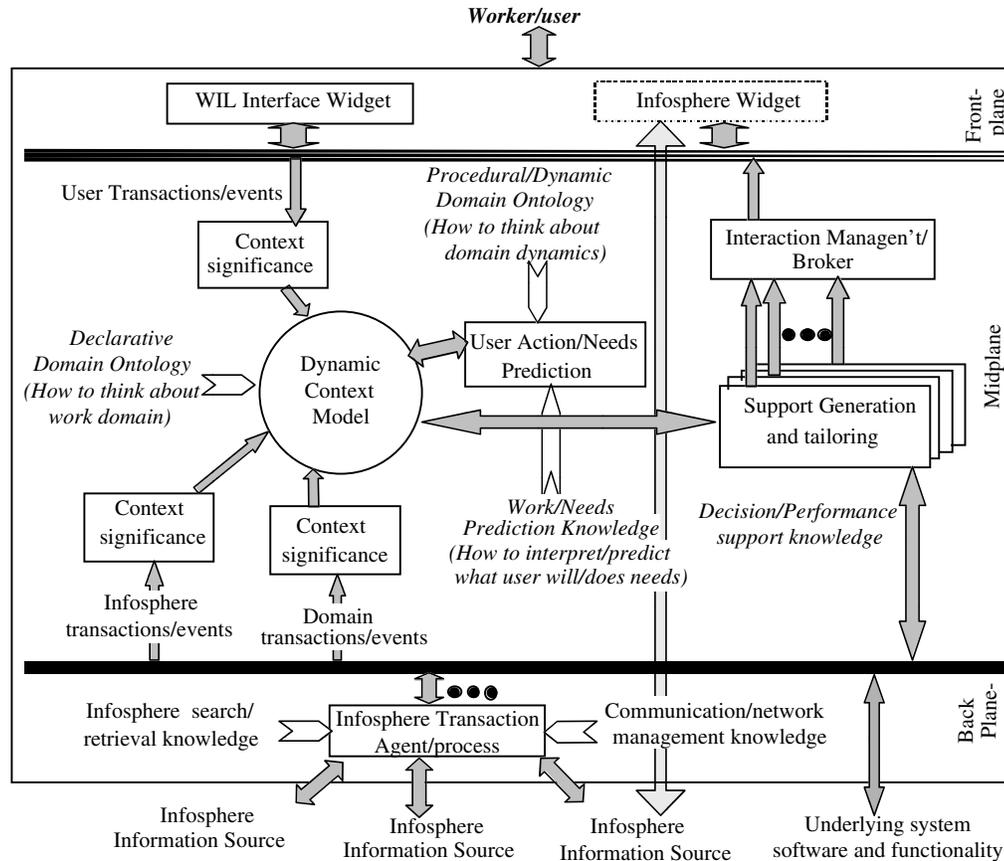
Zachary, W. (1985). Beyond user-friendly: Building decision-aid interfaces for expert end users. Cybernetics and Society Conference Proceedings, Tucson, Arizona. IEEE: New York. pp. 641-647

Zachary, W., Schremmer, S., and Donmoyer, J. (2002). Design Analysis for A Work-centered Infomediary Layer Application Toolkit (WAT). Technical Report 011214.01011. Springhouse, PA: CHI Systems, Inc.

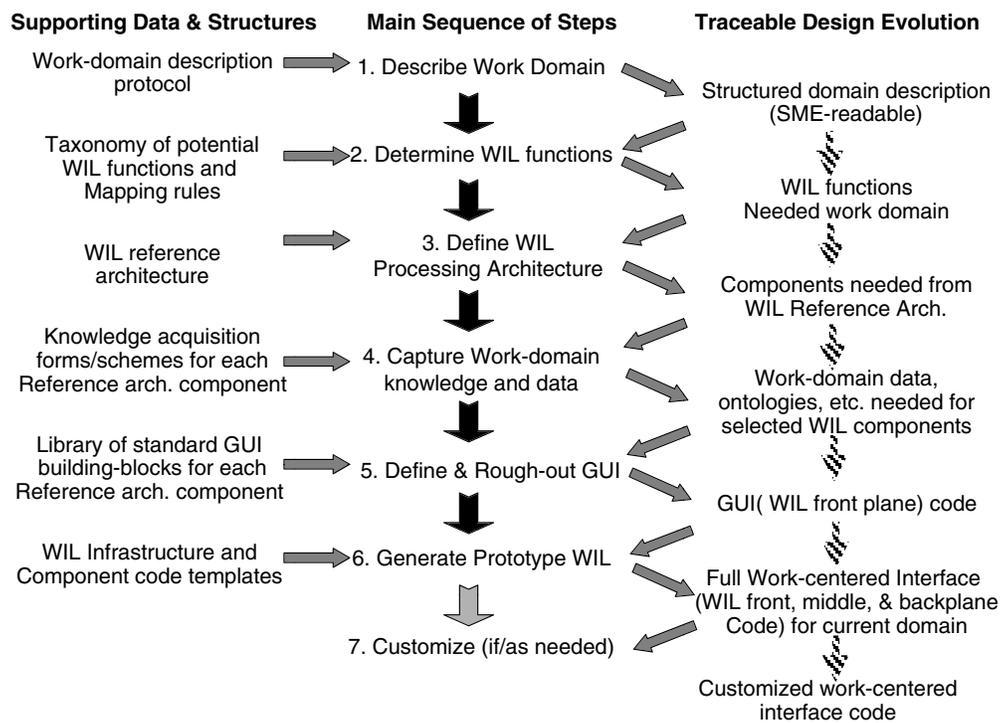
## **FIGURES AND TABLES**

**Table 1 WIL's General Work-Centered Support Functions**

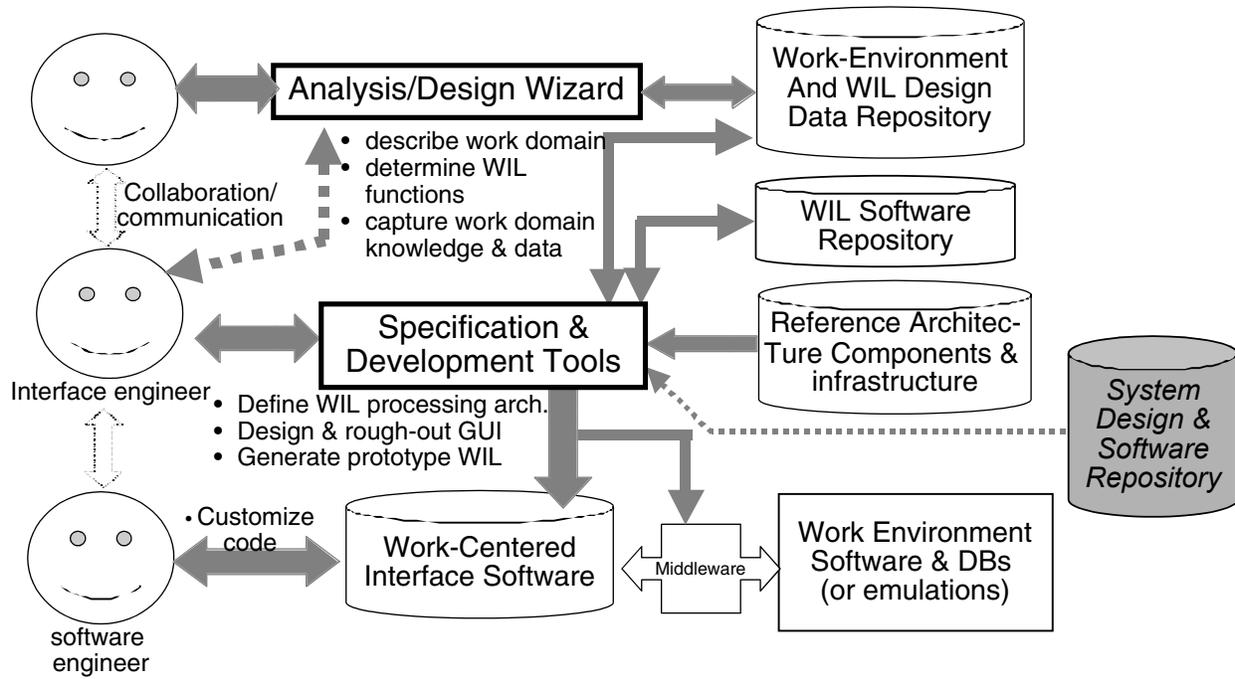
<b>Function</b>	<b>Description</b>
Activity Management	helping determine what work elements need to be done now, given the current work context
Work Process Structuring	helping determine how a given work element ought to be approached , given the current work context
Customized Performance Assistance	performing, at the request or approval of the user, a given work task or sub-element
Explanation/Elaboration	providing explanations of system, sensor, infosphere events in terms of their impact or effect on the user's work and work needs
Work-Environment Representation	constructing and displaying representations of the environment or work domain structure and processes
Infosphere data retrieval	actively collecting and making available pieces of information from the external environment (i.e., into data streams, databases, etc.)
What if	allowing the user to create hypothetical views in which possible future work actions are defined and explored
Situation Awareness	providing information at multiple levels of abstraction on the external system or process with which the user's work is involved
Work-flow Prioritization	helping determine which of the work elements that need to be done now have priority in the current work context (and why)



**Figure 1. WIL Conceptual Architecture**



**Figure 2. WIL Design Method**



**Figure 3. WAT Conceptual Organization**