A Work-centered Architecture and Design Method for Creating Joint Human-Computer Systems

Wayne Zachary, Ph.D.
CHI Systems

Robert Eggleston, Ph.D.
US Air Force Research Laboratory
Outline

♦ CHI Systems
♦ Preamble
♦ Theory-like stuff
♦ Practical-like stuff
Research, development, test, and transition products and processes to:

♦ Increase mission success, human performance, and system cost effectiveness

♦ Improve the relationship between people and technology

♦ Make complex technology more usable, intuitive, and effective
Areas we work in ...

♦ Decision-Support and Work-support Systems
  ♦ Command and Control
  ♦ Anti-terrorism and counter-terrorism
  ♦ Medical informatics and health record mobilization

♦ Training Systems
  ♦ Game-based and simulation-based trainers
    ♦ Serious games for serious times®
    ♦ Synthetic players and instructors

♦ Successful Transitions to Acquisition and Operational/Commercial Use
  ♦ USMC C2PC -- decision support toolbox component
  ♦ Future Force Warrior -- netted fires/effect C2 component
  ♦ iGEN™ -- cognitive agent software toolkit
The problem…

♦ Paralysis by Analysis
  ♦ ‘user centered’
  ♦ forward chained -- constrained by process not solution space

  produces

♦ Unusable Results
  ♦ cryptic analyses, Victorian prose, paper/storyboard prototypes, etc.
  ♦ consumers are S/W and H/W engineers (even Sys. Engs)
  ♦ organizationally separate, with own schedules and constraints

  instead of

♦ DESIGN DECISIONS
  ♦ specifications, justifications/constraints, ideally reach-back to domain data and constraints
Work within a well-defined design space
- abstractions of solutions that are known to work
- that simplify/structure S/W and H/W engineering efforts

Use characteristics/features of design space to drive data collection
- what is needed to make design decisions form the design space

And by the way
- link to software/system architecture, infrastructure, and reusable/tailorable component structure
Information System Design
Philosophies

♦ Traditional approaches
  technology centered
  problem-representation centered

♦ Result -- little match with actual work processes
  ♦ difficult to learn and operate
  ♦ requires human to change their work flow to ‘system-enforced’ terms, procedures, and work flows

♦ Alternative approach is work centered

♦ Goals of total system achieved through interactions among
  ♦ collaborative processes and teamwork
  ♦ work product development
  ♦ workflow management

♦ Cognition, in this view, is not ‘driver’ of work processes but is rather the consequence of work processes
Work Centered Support Systems (WCSS)

♦ Approach system design by focusing on the interactions among these three factors
♦ Work processes provides context for human cognitive processes

♦ Enable joint human–machine relationships that span continuum from
  ♦ *Passive* designs, intended to arrange information and functions in work–centered terms
  To
  ♦ *Active* designs, intended to add processing layer between user and underlying system/software that adapts information and functionality to end–user’s work
Human-Computer Support Relationships that ‘work’…

Support Functions

Active Support
- Work Management
- Work Structuring
- Infosphere Data Retrieval

Customized Automation
- Workflow Prioritization

Relational Support
- Explanation/elaboration
- What-if Analysis

Representational Support
- Work Environment Awareness
- Situational Awareness
## Work-Support Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity Management</td>
<td>helping the person decide what work activities need to be done at the current point in time, given the current work context</td>
</tr>
<tr>
<td>Work-flow Prioritization</td>
<td>helping the person decide which, among many work activities or work activity-instances that need to be done at a given time, have priority in the current work context and why</td>
</tr>
<tr>
<td>Situation Awareness</td>
<td>providing information, at multiple levels of abstraction, on infosphere systems or processes on which the user’s work is focused</td>
</tr>
<tr>
<td>Work-Environment Representation</td>
<td>constructing and displaying multiple abstracted representations of the structure and processes in the work domain or environment</td>
</tr>
<tr>
<td>Customized Performance Assistance</td>
<td>performing, at the request or approval of the person, a given work activity or sub-elements of it</td>
</tr>
<tr>
<td>Explanation/Elaboration</td>
<td>providing explanations of system, sensor, infosphere events or processes in terms of their impact or effect on the user’s own work and work needs</td>
</tr>
<tr>
<td>Infosphere data retrieval</td>
<td>going outside the WCSS into the infosphere (i.e., into data streams, databases, etc.), to collect and make pieces of information available for the user’s work process</td>
</tr>
<tr>
<td>What if</td>
<td>allowing the user to create a hypothetical work view in which possible future actions are defined and explored</td>
</tr>
<tr>
<td>Work Process Structuring</td>
<td>helping the person decide how a given work activity ought to be approached or broken down, given the current context of the work environment</td>
</tr>
</tbody>
</table>
WIL Example: NSFSA

Activity Management
Work structuring
Workflow Prioritization
Elaboration & Explanation

Infosphere Data Retrieval

Control of Customized Performance Assistance

Situation Awareness
Work-Environment Representation & Abstraction

Activity management

Customized Performance Assistance (proactive)

Customized Performance Assistance (negative)
ICW Functionality

- Customized Automation
- Workflow Prioritization
- Work Management
- Situational Awareness
- Work Environment Awareness

ICW Functionality

- Customized Automation
- Workflow Prioritization
- Work Management
- Situational Awareness
- Work Environment Awareness
ICW Functionality (Cont’d)

- **Situational Awareness**
- **Work Structuring**
- **Customized Automation**
- **What-if Analysis**

Analysis of events:
Payload measurands out-of-limits
Snapshot of event shows a possible laser dazzling from country G
Work-centered Interface Layer (WIL)

Generic, customizable WCSS architecture

Integrates
- active, representational, and relational work-support functions
- enabling processes
- interface to human workers and to underlying software & infosphere

in a reference architecture with implications for software infrastructure and reusable component structure
Work-centered Interface Layer

♦ Work support must be tailored to current work context
♦ Requires deeper-level enabling processes inside the computational component
  ♦ *Understanding* the work context
  ♦ *Assessing* what work needs to be done given the current work context
  ♦ *Tailoring support* to the work needs in current work context
WIL Conceptual Architecture

Worker/user

WIL Interface Widget
Work Domain Widget
Infosphere Widget

Domain & user transactions/events
Infosphere transactions/events

Context significance

Dynamic Context Model

User Action/Needs Prediction

Interaction Management/Broker

Tailored WCSS support function

Infosphere Transaction Agent/process

Infosphere Information Source
InfoSphere Information Source

Underlying system software and functionality
Infosphere Information Source

Front-plane
Midplane
Back-plane
Design Philosophy

♦ WIL reference architecture and set of general WCSS functions defines a *design space*.

♦ WIL software infrastructure and reusable component set defines an *implementation space* within the design space.

♦ Design method provides a means-end approach to:
  ♦ mapping the work domain into the design space
  ♦ mapping the WCSS design into the implementation space

♦ Constrains elicitation and implementation processes into paths that are known to be useful:
  ♦ focusing on efficiency rather than discovery
WIL and WCSS Development

♦ Basis for creating new WCSSs at two levels
  ♦ design level -- framework for analyzing and designing WCSSs
    ♦ requires a well-formed design method
  ♦ implementation level -- software components and architecture for rapidly implementing WCSSs
    ♦ requires a well-structured reference architecture and configurable, component software for generic functions

♦ Both levels enhanced by structured tools
  ♦ Embodying WIL design method
  ♦ Embodying WIL reference architecture and component software tools
Subject-matter expert (SME) - Cognitive engineer

**Design and Analysis tools**

- describe work domain
- determine functional requirements
- drill down to knowledge/data needed
- specify front-plane and back-plane interface requirements

**Work domain description**

WCSS functional requirements (WIL-based)

Knowledge needed to instantiate required WIL functions

Interface requirements
WIL-based WCSS Implementation

Work domain description
WCSS functional requirements (WIL-based)
Knowledge needed to instantiate required WIL functions
Interface requirements

Reusable:
WCSS infrastructure software components
knowledge templates
WIL-based WCSS code

Cognitive engineer
Software engineer

Collaborate

Integrated Software-Development Tools

Use requirements and specifications to:
• customize reusable components
• integrate domain-specific knowledge
• develop data interfaces
• integrate, debug, test
WCSS Application Tools (WAT)

- Design/Analysis Tools (D/AT)
- Integrated Development Environment (IDE)
- Design/Knowledge Repository
- Software Repository
- Work-Centered Support Software

WCSS Application Tools (WAT) boundary

SME
Cognitive engineer
Software engineer
Sequence of Steps
- paths of revision, re-thinking implicit only

Each step
- supported by conceptual building blocks (protocols, taxonomies, ontologies) or software building blocks (reusable code)
- produces an intermediate results (analytical or software), with linkages to previous intermediate results

Sequence of steps guides the process
Sequence of intermediate results provides a traceable design evolution
1. Describe the Work Domain
   - Structured domain description (SME-readable)

2. Determine the work support functions needed
   - WIL functions specified for the WCSS

3. Drill-down to relevant detailed knowledge & data
   - Work-domain knowledge and data needed to instantiate selected WIL functions for the WCSS

4. Build WCSS front-plane and back-plane
   - Customized GUI objects and Backplane interfaces

5. Assemble WCSS mid-plane from WIL components
   - Components needed from WIL Reference Arch.

6. Generate test, debug, refine prototype WCSS
   - Full WCSS (WIL front, middle, & backplane Code) for current domain

7. Customize (if/as needed)
   - Customized work-centered support-system code
Design Method Mapped onto WAT

1. Describe the Work Domain
   - Work-domain description protocol
   - Taxonomy of potential WIL functions and Selection guidelines
   - Knowledge acquisition forms/schemes for each Reference arch. component

2. Determine the work support functions needed
   - WIL functions specified for the WCSS
   - Work-domain knowledge and data needed to instantiate selected WIL functions for the WCSS

3. Drill-down to relevant detailed knowledge & data
   - Knowledge acquisition forms/schemes for each Reference arch. component

4. Build WCSS front-plane and back-plane
   - Library of customizable front-plane GUI objects for each WIL function or grouping
   - WIL reference architecture, infrastructure, and reusable components
   - Test, debug, and integration tools

5. Assemble WCSS mid-plane from WIL components

6. Generate test, debug, refine prototype WCSS
   - Source code for WCSS (i.e., WIL application)

7. Customize (if/as needed)
Abstract Software View

Cognitive Engineer

Software Engineer

D/A Tool

IDE

Repositories

Runtime Engine

User

Environment
What the DAT Does…

- Provide context-sensitive supporting information
- Track Data Entry Against Logical Dependencies
- Guide User Through WCSS Design Method
- DAT User Interactions
- Design/data repository
- Translate data to/from Repository format
- Coordinate Collaboration w/ SWEng User
- Provide Visual Representation of Process Status
- DAT Executive Process
- WAT IDE
More Detailed Software View

Design/Analysis Toolset

Elicitation Executive
WIL Function Selection Domain Knowledge Description Drill-Down

Context-Map Builder/Editor  Backplane (I/O) Specifier

Design Specification Repository
XML Design Specifications

Reference Architecture
Tailorable run-time engine w/Python backplane

Infrastructural Framework

Condition Builder/Editor  Frontplane GUI Configuration

Concept-map/Blackboard Editor

WIL Agent Knowledge Editor

Context-Maintenance Builder/Editor  Backplane Builder

Integrated Development Environment

Software Repository

WCSS (WIL) Application
Parting Thoughts…

♦ Overall process is focused on creating design decisions, embodied as traceable specifications, that support system development

♦ The purpose of the DAT to produce the XML specifications
  ♦ creating a well cognitively-engineered specification is a by product

♦ Tying the process to software components realistically constraints the design process and space
  ♦ A world composed of fully-custom GUIs is realistically not practical
Backups