

# *Applying Human Factors to the Design of Performance Tools*

Cherri Pancake

Northwest Alliance for Computational Science & Engineering

Oregon State University

[www.nacse.org](http://www.nacse.org)

Oregon  
State

## *The Paradox of Performance Tools*

Users are concerned about performance  
-- should be predisposed toward tools  
Instead, they complain about the lack of useful tools

- **Root of the problem: the nature of “tools”**
  - A tool isn't used or evaluated as a standalone element
  - Only valuable if it clearly helps user accomplish key tasks
- **A tool won't be accepted if**
  - It is prone to being misapplied
  - It doesn't support tasks that users want to perform
  - It doesn't support them in ways that are simple and natural

Oregon  
State

## *Where Human Factors Fit In*

- **Human factors**
  - Characteristics, capabilities and limitations of human beings
  - How these affect our use of technology
- **Goal of human factors engineering**
  - Improve system effectiveness and safety by addressing human factors explicitly
- **This presentation discusses**
  - Human factors issues involved in performance tuning
  - Why current tools don't address them well
  - What kinds of mechanisms are needed

Oregon  
State

## *How Users Approach Performance Tuning*

- During tuning, user poses questions (conscious or subconscious)
- Questions establish a **conceptual framework**
  - Series of subgoals that must be met
  - Dependencies determine sequencing needed

### **Example framework**

*Is there a problem? What are its symptoms? (Identification)*

*Where do things go wrong? What is the cause? (Localization)*

*What needs to be changed? (Repair)*

*Did it actually improve performance? (Verification)*

*Is there still a problem? (Validation)*



Oregon  
State

## Why Current Tools Aren't Enough

- Users need to compare different aspects of performance to determine which one is the problem
- Tools don't let users see interrelationships between metrics

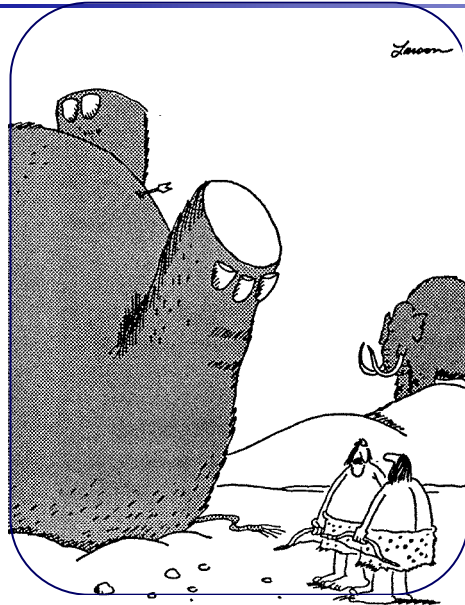
*"Don't tool developers know how awful it is to bring up a tool and have to scan through 100,000 communications or subroutine calls?"*

- Users want to narrow problem down to a particular code region
- Tools don't make it easy to pinpoint the corresponding source lines

*"It's not much help to know that it was a particular subroutine, unless I can tell it happened whenever X was invoked by Y, but not when X was invoked by Z"*

Oregon  
State

## Some "Fixes" Need to be Very Precise ...



"Maybe we should write that spot down."

Oregon  
State

## Why Current Tools Aren't Enough

- Users need to navigate through complex source hierarchies to find regions with similar behavior
- Tools don't help users "remember" locations they will want to re-visit

*"It's hard to find the funny behavior in the first place, but it's really frustrating when I can't find it again later..."*

- Users want guidance about what kind of changes are needed
- Tools only present behavior, but don't suggest how to change it

*"Tools don't tell me anything I can actually use to go back and fix my code"*

Oregon  
State

## What Users Really Want



Oregon  
State

## *Applying Human Factors Engineering to Tools*

- A good tool supports the activities users want to accomplish -- in the way that users want to do them
- User's conceptual framework establishes goals to be met
- User acts upon a task structure to meet each goal
  - *Localization goal: Where does problem occur? What symptoms?*
  - Example task structure:

- (1) Stabilize the problem (identify how to provoke it)
- (2) Determine which region(s) of code are suspect
- (3) Hypothesize why it's occurring there
- (4) Compare to see if "similar" regions have same problem  
...etc...

Oregon  
State

## *Case 1: Search Space Reduction Task*

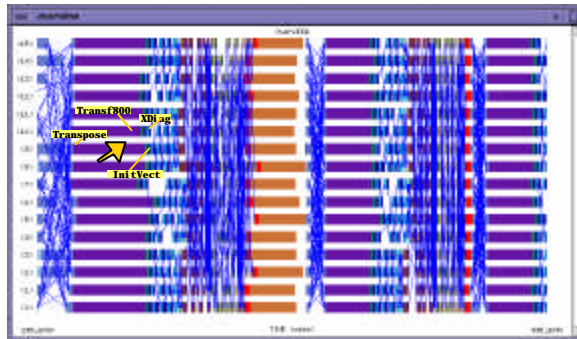
*What code region is "suspect"?*

- User's plan
  - ⇒ Make educated guess about nature of problem
  - ⇒ Test hypothesis through instrumentation/monitoring
  - ⇒ Narrow in on region of source code responsible
  - ⇒ Infer whether performance is "good" or "bad"
- Key HF problem: total search space (i.e., performance space) is very large
- Apply HF engineering
  - Support user's ability to explore performance space
  - Support user's ability to compare aspects of performance

Oregon  
State

## Apply Human Factors ... to Support Exploration

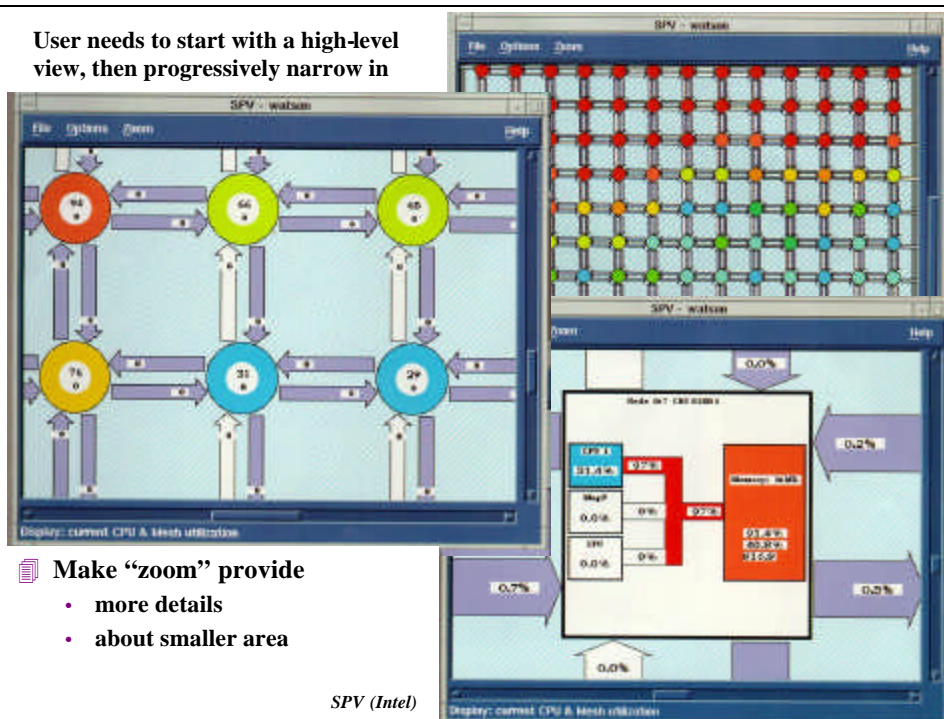
- Visualizations can be confusing
  - Textual labels help the user understand what's seen -- but they clutter the screen
- ☞ Make labels adapt to user's "focus of attention"
  - As cursor moves, surrounding labels appear, disappear



Enhancement of AIMS (Yan et al.)

Oregon State

User needs to start with a high-level view, then progressively narrow in



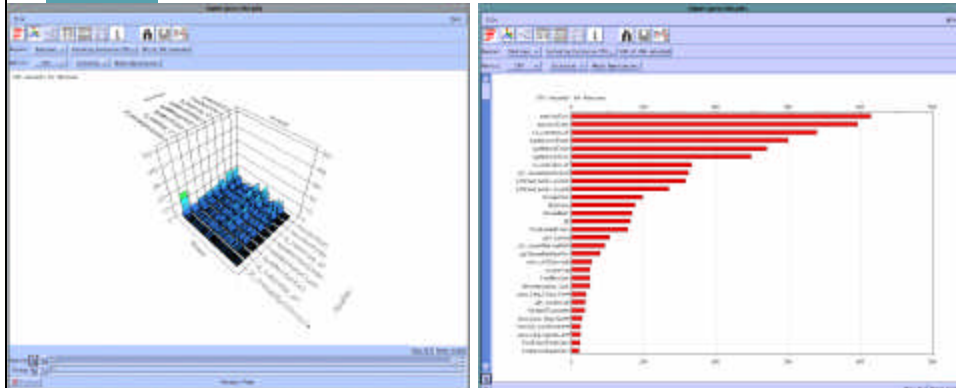
- ☞ Make "zoom" provide
  - more details
  - about smaller area

SPV (Intel)

## *Apply Human Factors ... to Support Comparison*

- User must compare data to determine “goodness” / “badness”
  - ☞ Allow user to “clone” displays
    - Duplicate window’s options can be changed (to compare two metrics, perspectives, etc.)

*Enhancement of CXperf (Convex/HP)*



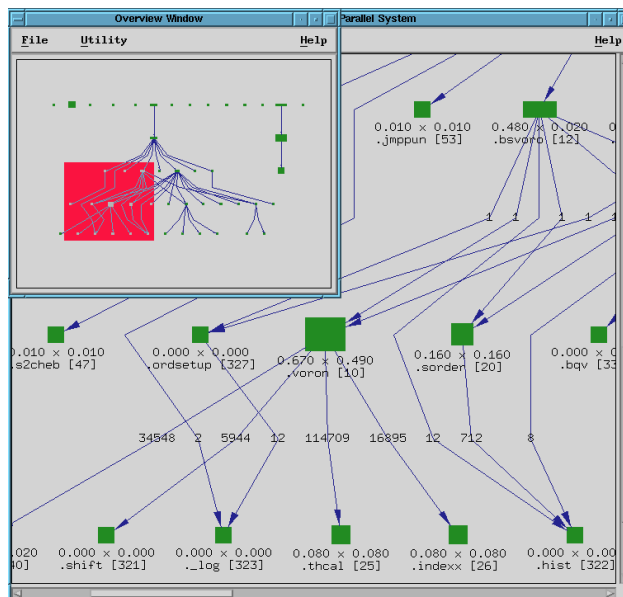
## *Case 2: Selective Modification Task*

- User’s plan
  - ⇒ Focus on one code location
  - ⇒ Determine cause of problem -- using intuition and judgement
  - ⇒ Modify code accordingly
  - ⇒ Move to next location
- Apply HF engineering
  - Support user’s ability to navigate through large displays
  - Support user’s ability to navigate through complex code space

## Apply Human Factors ... to Support Navigation

- Displays are so large it's easy to become "lost"
- User needs a sense of context

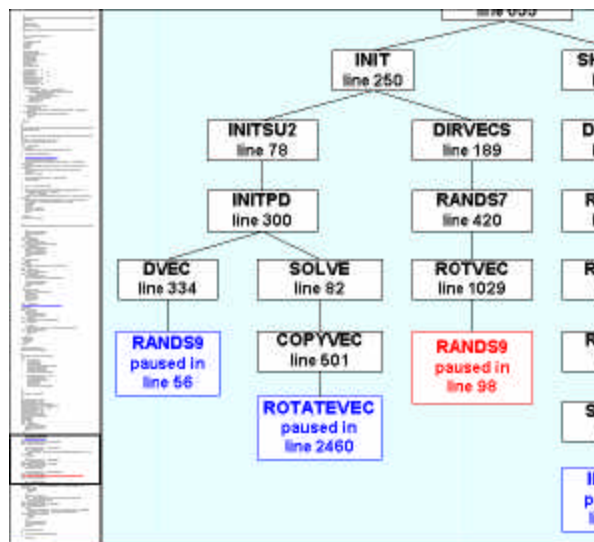
- ☞ Use thumbnails to
  - show relationships
  - make movement easier



## Apply Human Factors ... to Support Navigation

- Showing a source statement isn't enough
- User needs a sense of context

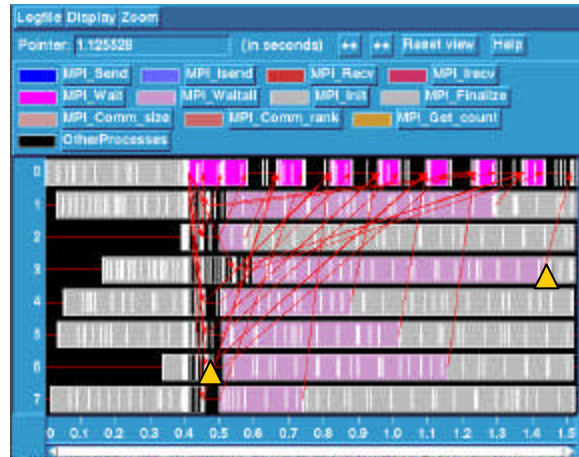
- ☞ Use miniaturizations to
  - show relationships
  - make movement easier



## Apply Human Factors ... to Support Navigation

- User repeatedly re-visits areas of the code or performance data
- Needs “landmarks” to find things quickly

- Add “user landmarks”
  - Make it possible to flag or highlight locations



Enhancement of Nupshot (Lusk)

## Applying Human Factors to Tool Visualizations

- Visualization should be the key to usable performance tools
  - Provides a way to manage large, complex data
  - Can capitalize on user's pattern recognition capabilities
  - Can assist the user to explore and “interpret” program behavior through alternate views
- But it's much harder than just using extra windows!
  - Tools show info without supporting navigation or comparison
  - Users need to approach performance data flexibly
    - » Multiple perspectives that can be focused independently
    - » Easy to experiment with different combinations
    - » Easy to maintain a *sense of context* and *control*

## *What's Really Wrong with Today's Tools?*

- Performance tools make it too hard to carry out common user tasks

*"Tools are supposed to make me more productive, but I can usually find the problem quicker with PRINT and calls to timers."*

*"I want to ask simple questions --  
why do tools make them so hard to answer?"*

Oregon  
State

## *Tools Need to Match User Needs and Habits*

A tool that doesn't consider human factors...

... may be worse than no tool at all!

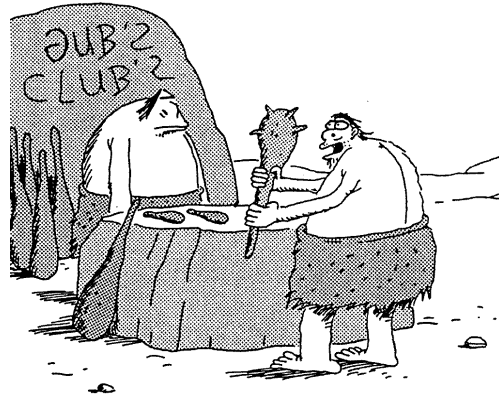


Mozart writing the digital version of his symphony No. 38 in D major

Oregon  
State

## *Remember...*

- Users want to explore the behavior of their code  
-- not the behavior of your tool !



"No, no... not this one. Too many bells and whistles"

Oregon  
State

*Produced with (involuntary) artistic assistance from:*  
**Gary Larson, creator of Far Side**  
**S. Harris, newspaper cartoonist**

Oregon  
State