

Using CogTool to Model Programming Tasks

Rachel Bellamy¹, Bonnie John², John Richards¹, John Thomas¹

¹Software Productivity Group
IBM T. J. Watson Research Center
Hawthorne, NY USA

²Human-Computer Interaction Institute
Carnegie Mellon University

¹[rachel, ajtr, jcthomas}@us.ibm.com](mailto:{rachel, ajtr, jcthomas}@us.ibm.com) ²bej@cs.cmu.edu

This work was supported by the Defense Advanced Research Projects Agency under its Agreement No. HR0011-07-9-0002.

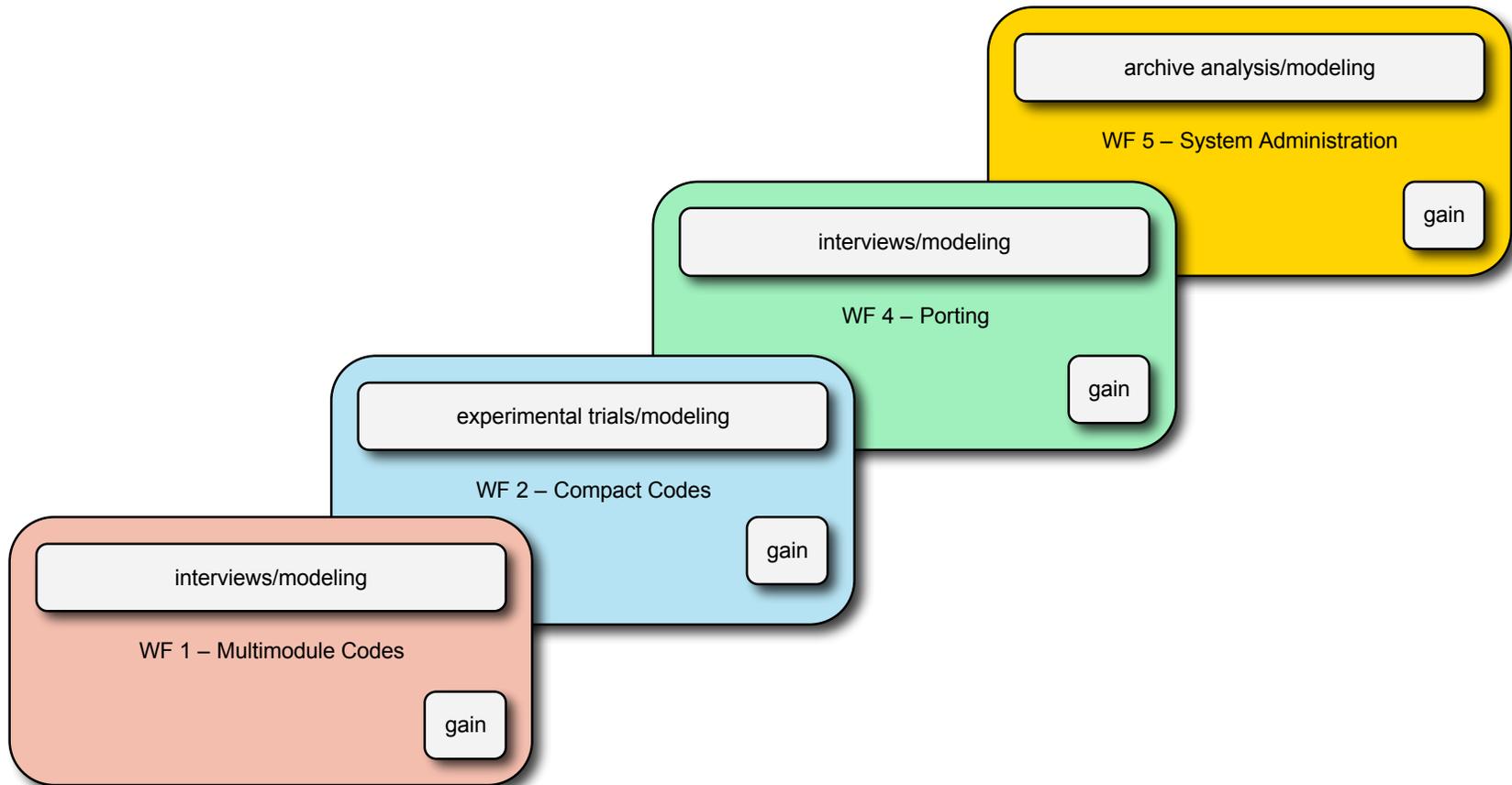
Additional support from IBM Research Division Open Collaborative Research Program.

Context: DARPA HPCS Initiative

- To spur creation of new generation of economically viable high productivity computing systems
- IBM PERCS System
 - General purpose sustained trans-petaflop* machine based on P7 Power architecture scaling to 500,000 processors
 - Capable of very high-speed random memory updates (GUPS)
- Introducing new parallel programming language (X10) and cross language support for PGAS programming model
- Including suite of new parallel development tools integrated within Eclipse IDE (parallel debugger, parallelism analyzers, optimization and refactoring aids, ...)
- Targeting 10X+ productivity gains, 2002 vs 2010, across multiple workflows

*1000 trillion floating point operations per second

Context : Productivity Assessment



$$\text{productivity gain}_{\text{per workflow}} = \frac{\text{totalTime}_{2002}}{\text{totalTime}_{2010}}$$

$$\text{totalTime}_{\text{per epoch}} = \sum \text{taskTime}$$

$$\text{taskTime} = (\text{executionTime} * \text{executionFrequency}) / \text{successRate}$$

Context : Assessment Methodologies

- Interviews and Analysis
 - Relatively low cost, broad coverage* (with large enough sample)
 - But, reduction to task times, and probability of completion not straightforward
 - Task and User Modeling
 - Intermediate cost, narrower coverage (relatively routine tasks, not problem solving)
 - Requires detailed specification of user tasks (drawing on methods above and below)
 - Can estimate *expert* task times with good accuracy based on prior empirical work
 - Empirical Measurement
 - Highest cost, narrowest coverage (problem solving tasks possible but sampling a problem)
 - Yields estimates of task times and success rates
 - Provides grounding for other work
 - Requires representative users performing representative tasks (hard to find and conduct)
- * for Workflows 1 and 4, ranges are 1-35 years parallel experience, 1 to 4000 kloc codes, 11 minutes to 2 years per task

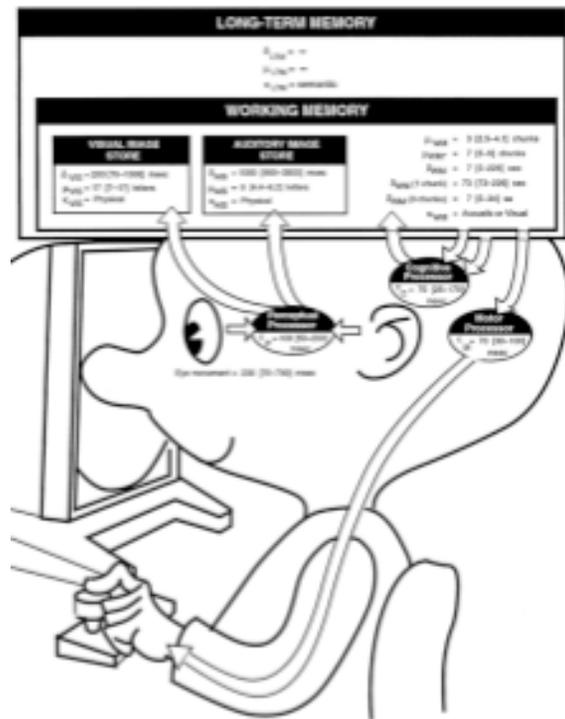
Background: Assessment Methodologies

Focus of this talk



- Interviews and Analysis
 - Relatively low cost, broad coverage* (with large enough sample)
 - But, reduction to task times, and probability of completion not straightforward
- **Task and User Modeling**
 - Intermediate cost, narrower coverage (relatively routine tasks, not problem solving)
 - Requires detailed specification of user tasks (drawing on methods above and below)
 - Can estimate *expert* task times with good accuracy based on prior empirical work
- Empirical Measurement
 - Highest cost, narrowest coverage (problem solving tasks possible but sampling a problem)
 - Yields estimates of task times and success rates
 - Provides grounding for other work
 - Requires representative users performing representative tasks (hard to find and conduct)

One Method: Keystroke-Level Modeling



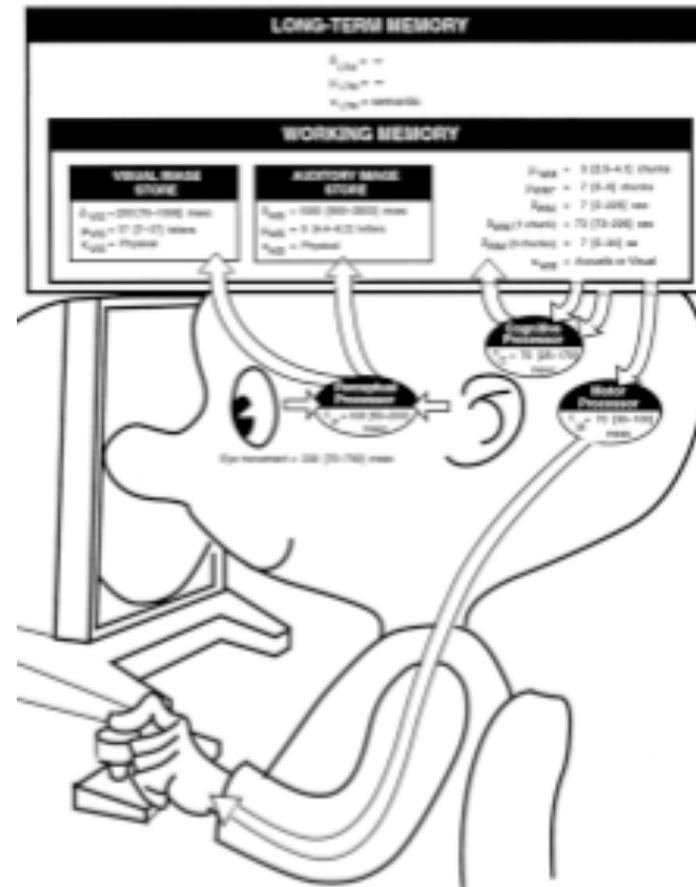
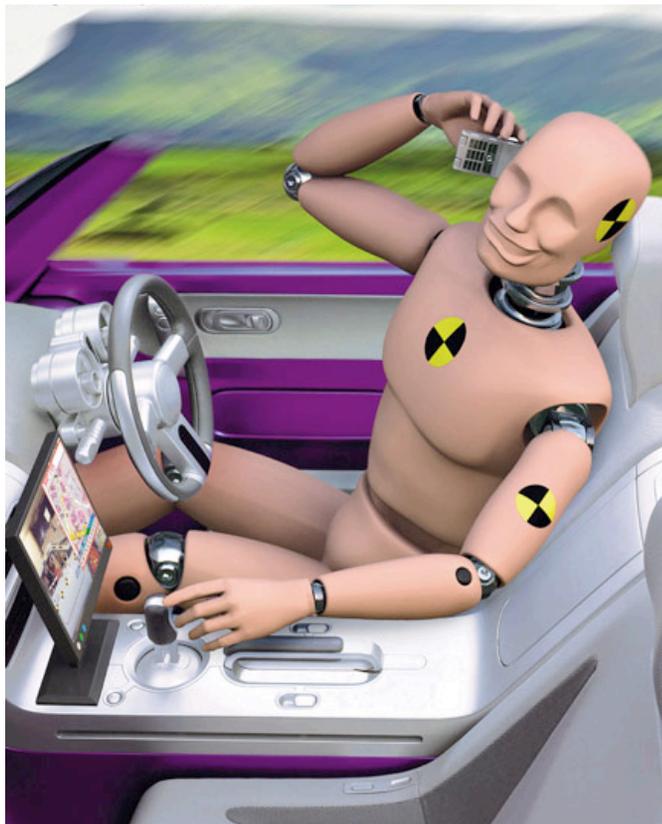
- Based on model human processor with perceptual, motor, and cognitive subsystems
- Accurate predictions of expert task times confirmed by over two decades of research

yields estimates of expert task time

A Brief History of KLM

- **The Psychology of Human Computer Interaction**, Card, Moran and Newell (1983)
- John, B. E. (1988) Contributions to engineering models of human-computer interaction. Doctoral dissertation, Carnegie Mellon University. (student of Newell)
- **The Keystroke Level Model (KLM) is applied to the complex tasks of telephone operators** predicting that a new workstation which saves keystrokes would actually *increase* response time. Empirical studies verified. Model allowed changes in procedures and interface.
- Result: **millions of dollars in capital saved and millions more in operating costs**. Gray, W. D., John, B. E., Stuart, R., Lawrence, D., & Atwood. M. E. (1990) GOMS meets the phone company: Analytic modeling applied to real-world problems. In D. Diaper et al. (eds), Human-Computer Interaction INTERACT '90. North-Holland: Elsevier Science Publishers. pp. 29-34.
- **KLM/GOMS is applied to other domains**. Altmann, E. M., & John, B. E. (1995) A preliminary computational model of **expert programming**. Carnegie Mellon University School of Computer Science Technical Report No. CMU-CS-95-172 and as the Human-Computer Interaction Institute Technical Report No. CMU-HCII-95-103.
- John, B. E., & Lallement, Y. (1997) Strategy use while learning to perform the Kanfer-Ackerman **Air Traffic Controller**® task. Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society, August 1997.
- **CogTool is developed** to make modeling more accurate and less time-consuming. Hudson, S. E., John, B. E., Knudsen, K., & Byrne, M. D. (1999). A tool for creating predictive performance models from user interface demonstrations. UIST'99: Proceedings of the ACM Symposium on User Interface Software and Technology.
- **Cognitive Modeling becomes standard method** taught at ACM's conference on Computer Human Interaction, CMU, Michigan, Georgia Tech and other major institutions focusing on Human Computer Interaction. (1995-Present).
- **CogTool applied to in-vehicle police IT**. Callander, M. and Zorman, L. 2007. Usability on patrol. In CHI '07 Extended Abstracts on Human Factors in Computing Systems (San Jose, CA, USA, April 28 - May 03, 2007). CHI '07. ACM, New York, NY, 1709-1714.
- **CogTool applied to tabs in Firefox**. Knight, A., Pyrzak, G., & Green, C. (2007). When two methods are better than one: Combining user study with cognitive modeling. In M.B. Rosson and D.J. Gilmore (Eds.): Extended Abstracts of the 2007 Conference on Human Factors in Computing Systems CHI 2007, (pp. 1783-1788). San Jose, CA: ACM.
- **Bonnie John sabbatical at Watson 2010**. Begin using CogTool to model programming behaviors relevant to PERCS.

Bonnie John: “Cognitive Crash Dummy”



CogTool* (Bonnie John et al)

The screenshot displays the CogTool interface with three main components:

- 1. Mock-up design in a storyboard:** A storyboard showing a sequence of UI screens connected by arrows, representing a user's workflow.
- 2. Demonstrate the tasks:** A window titled "Editor, Comment" showing a simulated user interaction with a "Comment" form. Below the form, a "Keyboard" button indicates that the task is being demonstrated using keyboard shortcuts.
- 3. Predictions appear in a spreadsheet:** A "Script Step List" window showing a table of predicted actions and their durations. The table includes columns for "Frame" and "Action".

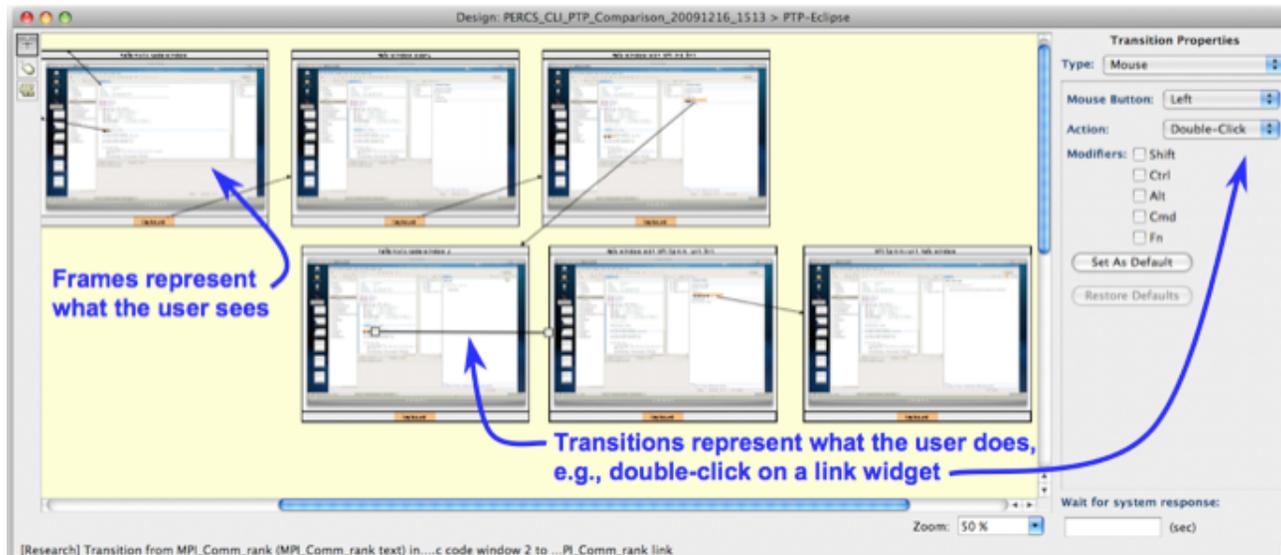
Frame	Action
Editor	Think for 1.2 s
Editor	Look at
Editor	Move Mouse
Editor	Left Click
Editor	Home Keyboard
Editor	Press 'O' (response to comment)
Editor_TitledOn	Home Mouse
Editor_TitledOn	Think for 1.2 s
Editor_TitledOn	Look at
Editor_TitledOn	Move Mouse
Editor_TitledOn	Left Click
Editor_TitledOn	Home Keyboard
Editor_TitledOn	Press 'O' (see recently get this)
Editor_TitledOn	Think for 1.2 s
Editor_TitledOn	Press 'E'
Editor_PreambleOn	Home Mouse
Editor_PreambleOn	Think for 1.2 s
Editor_PreambleOn	Look at
Editor_PreambleOn	Move Mouse
Editor_PreambleOn	Left Click
Editor_PreambleOn	Look at
Editor_PreambleOn	Move Mouse
Editor_PreambleOn	Left Click

Additional elements in the screenshot include a "Tasks" spreadsheet showing predicted times for "Edit Post With Mouse" (46.605 s) and "Edit Post with Keyboard Shortcuts" (42.372 s), and a "Human-Computer Interaction Institute" logo.

makes keystroke-level modeling practical

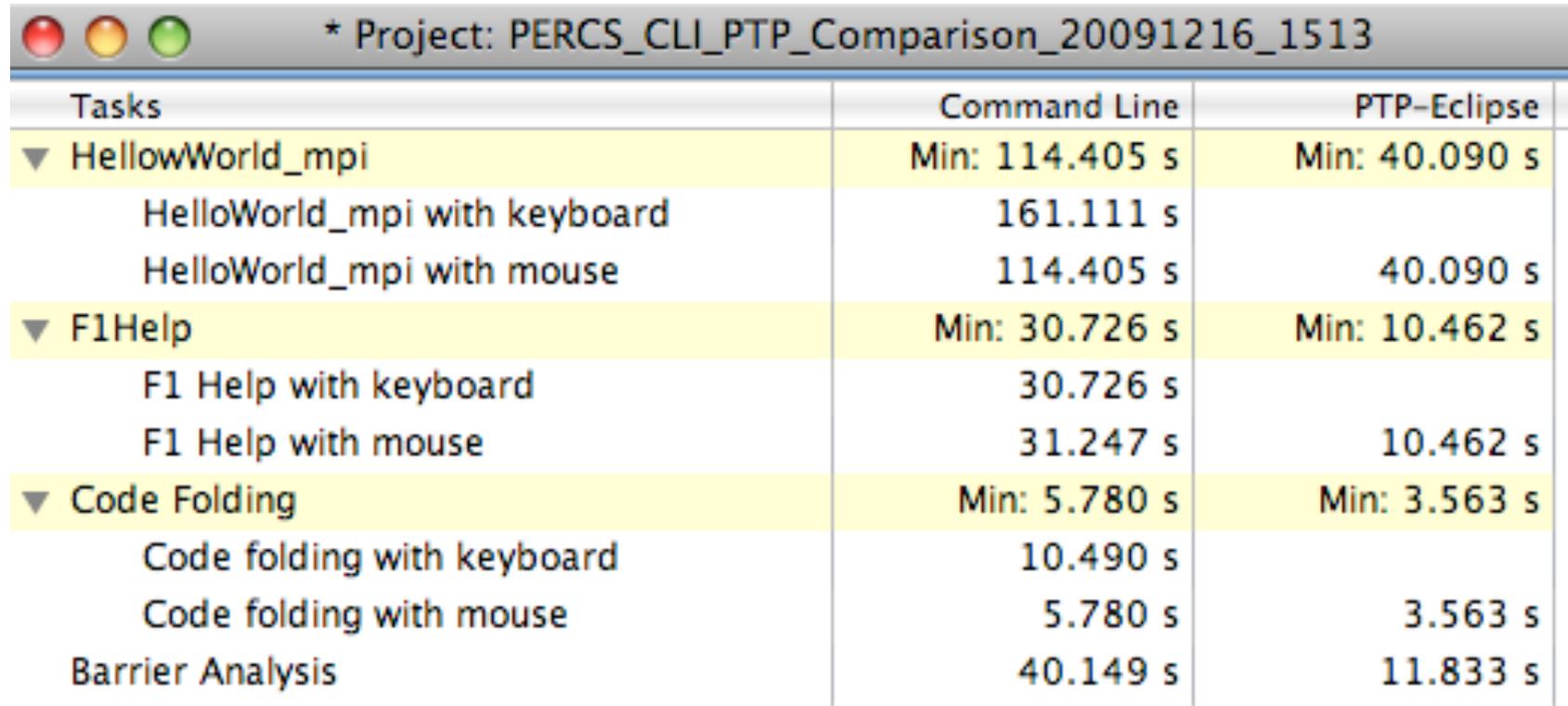
*<http://cogtool.hcii.cs.cmu.edu/>

Describe the Interface Design



A fragment of the PTP Eclipse storyboard for accessing help using the F1 key. The transition representing a double-click on a code object is selected (boxes at its ends), so its properties are shown in the properties pane on the right.

Produces Skilled Execution Time for the Task



The screenshot shows a window titled "* Project: PERCS_CLI_PTP_Comparison_20091216_1513". The window contains a table with three columns: "Tasks", "Command Line", and "PTP-Eclipse". The table lists several tasks, including "HellowWorld_mpi", "F1Help", and "Code Folding", each with sub-tasks for "with keyboard" and "with mouse". The "Command Line" column shows the time taken for each task, and the "PTP-Eclipse" column shows the estimated expert task time. The "HellowWorld_mpi" task has a minimum time of 114.405 s for the command line and 40.090 s for PTP-Eclipse. The "F1Help" task has a minimum time of 30.726 s for the command line and 10.462 s for PTP-Eclipse. The "Code Folding" task has a minimum time of 5.780 s for the command line and 3.563 s for PTP-Eclipse. A "Barrier Analysis" task is also listed with a time of 40.149 s for the command line and 11.833 s for PTP-Eclipse.

Tasks	Command Line	PTP-Eclipse
▼ HellowWorld_mpi	Min: 114.405 s	Min: 40.090 s
HellowWorld_mpi with keyboard	161.111 s	
HellowWorld_mpi with mouse	114.405 s	40.090 s
▼ F1Help	Min: 30.726 s	Min: 10.462 s
F1 Help with keyboard	30.726 s	
F1 Help with mouse	31.247 s	10.462 s
▼ Code Folding	Min: 5.780 s	Min: 3.563 s
Code folding with keyboard	10.490 s	
Code folding with mouse	5.780 s	3.563 s
Barrier Analysis	40.149 s	11.833 s

CogTool results window showing estimated expert task times.

KLM – 2002 CLI Help

Script: PERCS_CLI_PTP_Comparison_20091216_1513 > Command Line > F1 Help with keyboard

17 Find with MPI_Comm_rank

Prediction: 30.726 s

Show Visualization

cli-help.ogg

Script Step List

Frame	Action	Widget/Device
07 goggle find	Think for 1.200 s	
07 goggle find	Move Mouse	Find field
07 goggle find	Left Click	Find field
07 goggle find	Home Keyboard	
07 goggle find	Type 'M↑P↑↑↑↑-↑Init'	Keyboard
10 find with MPI_Init	Think for 1.200 s	
10 find with MPI_Init	Home Mouse	
10 find with MPI_Init	Move Mouse	MPI_Init-MPI_Init (MPI_Init link)
10 find with MPI_Init	Left Click	MPI_Init-MPI_Init (MPI_Init link)
11 MPI_Init help	Think for 1.200 s	
11 MPI_Init help	Move Mouse	Code window
11 MPI_Init help	Left Click	Code window
06 code window [2]	Look At	MPI_Comm_rank (MPI_Comm_rank text)
06 code window [2]	Think for 1.200 s	
06 code window [2]	Move Mouse	Google window
06 code window [2]	Left Click	Google window
...ith MPI_Comm_rank	Move Mouse	MPI_Init (MPI_Init text)
...ith MPI_Comm_rank	Left Double-Click	MPI_Init (MPI_Init text)
...ith MPI_Comm_rank	Home Keyboard	
...ith MPI_Comm_rank	Type 'M↑P↑↑↑↑-↑Comm↑-rank'	Keyboard
...ith MPI_Comm_rank	Think for 1.200 s	
...ith MPI_Comm_rank	Home Mouse	
...ith MPI_Comm_rank	Move Mouse	...MPI_Comm_rank (MPI_Comm_rank link)
...ith MPI_Comm_rank	Left Click	...MPI_Comm_rank (MPI_Comm_rank link)
10 Comm_rank help		

Keyboard

Zoom: 56.245 %

Look at Widget Think

[Research]

Mouse hand Right

Initial hand location Keyboard

Delete Step

Compute

KLM – 2010 Eclipse PTP FI Help

Script: PERCS_CLI_PTP_Comparison_20091216_1513 > PTP-Eclipse > F1 Help with mouse

MPI_Comm-rank help window

Prediction: 10.462 s

Show Visualization

Script Step List

Frame	Action	Widget/Device
hellompi.c code window	Look At	MPI_Init (Text MPI_Init)
hellompi.c code window	Think for 1.200 s	
hellompi.c code window	Move Mouse	MPI_Init (Text MPI_Init)
hellompi.c code window	Left Click	MPI_Init (Text MPI_Init)
hellompi.c code window	Think for 1.200 s	
hellompi.c code window	Type 'f1'	Keyboard
Help window opens	Type 'f1'	Keyboard
...with MPI_Init link	Think for 1.200 s	
...with MPI_Init link	Move Mouse	int MPI_Init(int*, char**) (MPI_Init help link)
...with MPI_Init link	Left Click	int MPI_Init(int*, char**) (MPI_Init help link)
Frame 1	Move Mouse	MPI_Comm_rank (MPI_Comm_rank text)
Frame 1	Left Double-Click	MPI_Comm_rank (MPI_Comm_rank text)
...th MPI_Comm_rank link	Think for 1.200 s	
...th MPI_Comm_rank link	Move Mouse	...MPI_Comm, int *) (MPI_Comm_rank text)
...th MPI_Comm_rank link	Left Click	...MPI_Comm, int *) (MPI_Comm_rank text)
...rank help window		

Keyboard

Zoom: 30.673 %

Look at Widget Think

Mouse hand Right

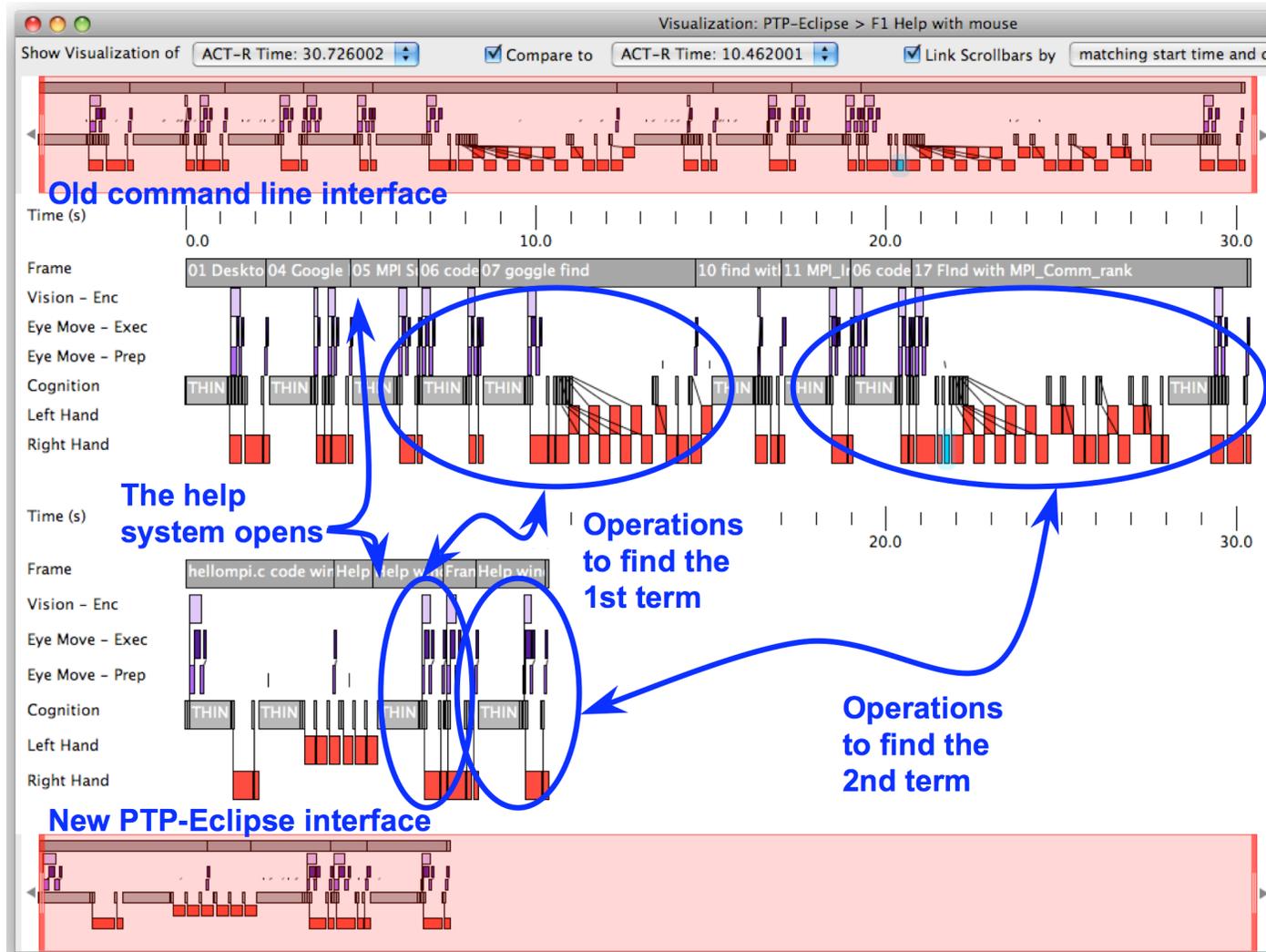
Initial hand location Mouse

Delete Last Step

Compute

Understanding Differences Between Systems

Task: Accessing Help



CogTool timeline visualization showing the differences between two systems.

KLM-2002 CLI MPI Project Setup from Template

Script: PERCS_CLI_PTP_Comparison_20091216_1513 > Command Line > HelloWorld_mpi with keyboard

bottom ./hwmpi

Prediction: 161.111 s

Show Visualization

Frame	Action
bottom line \$(MAKE)...	Type '↓↓↓↓↓↓↓↓'
bottom line empty 1	Think for 1.200 s
bottom line empty 1	Type 'dddddddddddddddddddd'
bottom line empty 1	Think for 1.200 s
bottom line empty 1	Type '↑;wq'
bottom line empty 1	Type 'f'
...pty line selected	Think for 1.200 s
...pty line selected	Type 'make'
...pty line selected	Type 'f'
bottom CL make	Think for 1.200 s
bottom CL make	Type 'ls'
bottom CL make	Type 'f'
bottom ls	Think for 1.200 s
bottom ls	Type './hwmpi'
bottom ls	Think for 1.200 s
bottom ls	Type 'f'
bottom ./hwmpi	

Keyboard

Zoom: 30.574 %

Look at Widget Think

Mouse hand Right

Initial hand location Keyboard

Delete Last Step

Compute

KLM-2010 Eclipse PTP MPI Project Setup From Template

Script: PERCS_CLI_PTP_Comparison_20091216_1513 > PTP-Eclipse > HelloWorld_mpi with mouse

Hello world project opened

Prediction: 40.090 s

Show Visualization

Script Step List

Frame	Action	Widget/Device
...Settings Wizard [2]	Home Keyboard	
...Settings Wizard [2]	Type '⌘⌘B⌘M⌘Corp 2007'	Keyboard
...Settings Wizard [2]	Think for 1.200 s	
...Settings Wizard [2]	Home Mouse	
...Settings Wizard [2]	Move Mouse	Greeting input
...Settings Wizard [2]	Left Click	Greeting input
...Settings Wizard [2]	Home Keyboard	
...Settings Wizard [2]	Type '⌘⌘1⌘1⌘1⌘Hello ⌘M⌘P⌘I⌘World⌘1⌘1⌘1'	Keyboard
...Settings Wizard [2]	Look At	SRC input
...Settings Wizard [2]	Think for 1.200 s	
...Settings Wizard [2]	Home Mouse	
...Settings Wizard [2]	Move Mouse	Next> (Next)
...Settings Wizard [2]	Left Click	Next> (Next)
MPI Project Settings	Left Click	Next> (Next)
Select Configurations	Think for 1.200 s	
Select Configurations	Move Mouse	Finish> (Finish)
Select Configurations	Left Click	Finish> (Finish)
...Id project opened		

Keyboard

Zoom: 38.016 %

Look at Widget Think

Mouse hand Right

Initial hand location Mouse

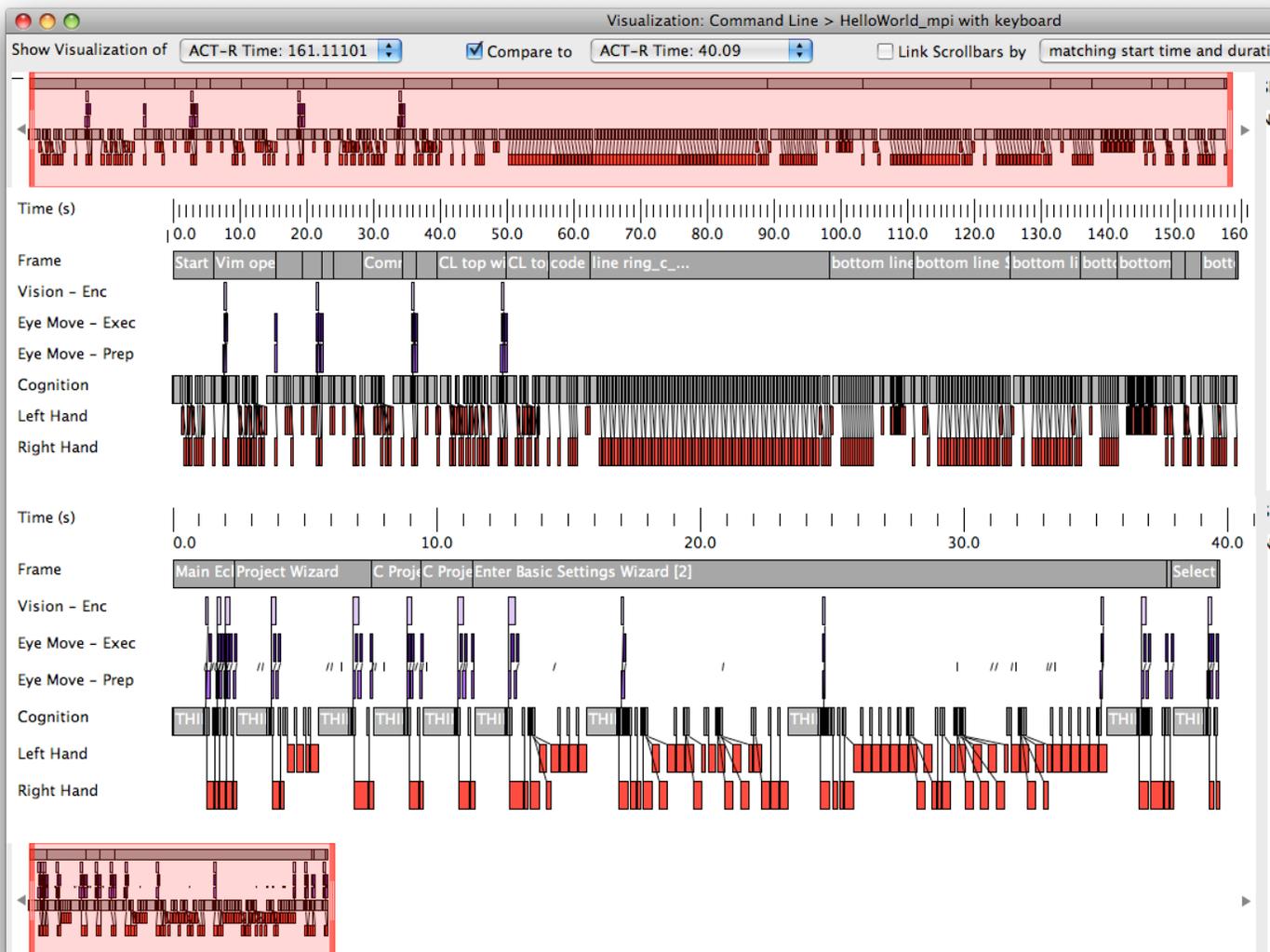
Delete Last Step

Compute

Understanding Differences Between Systems

Task: Project Setup From Template

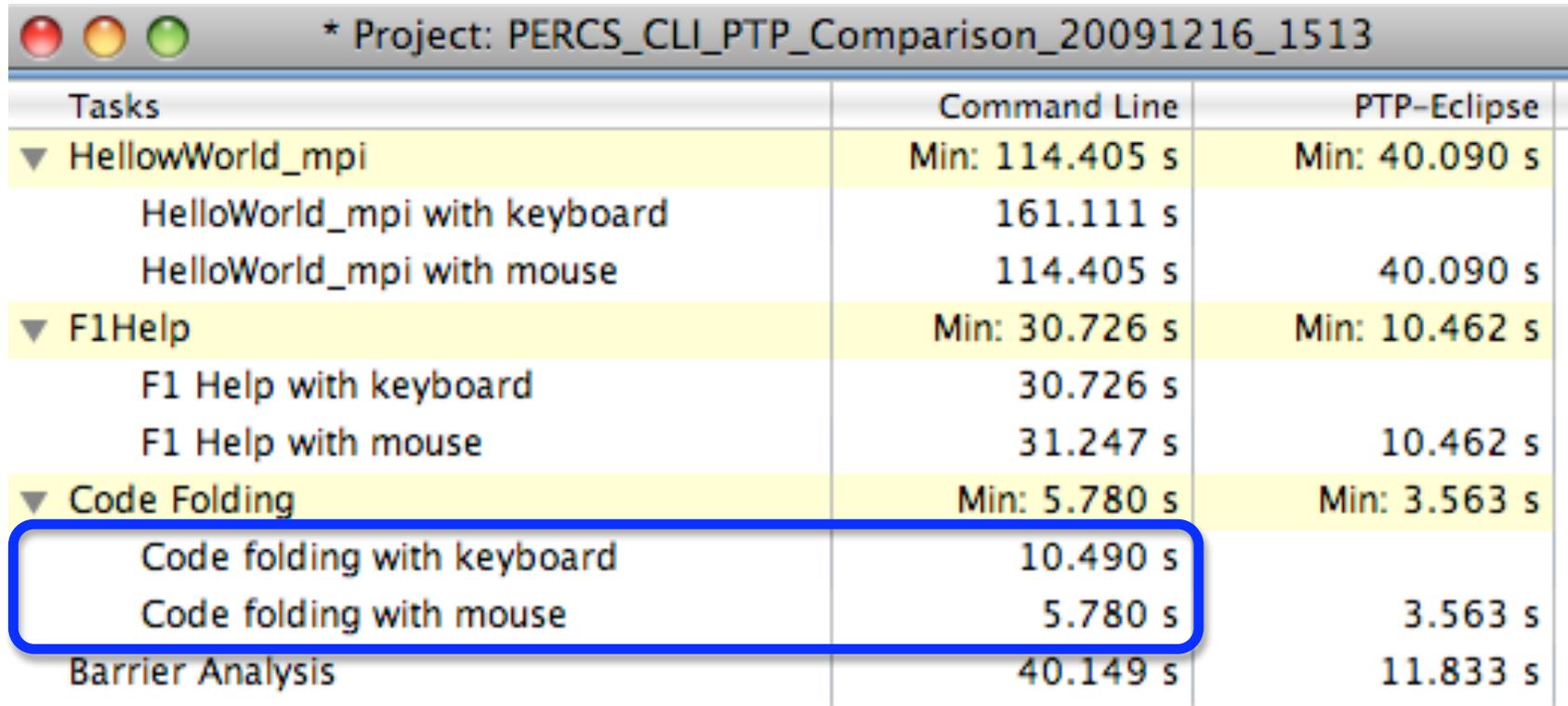
VIM - using
keyboard
when
possible



Eclipse PTP

Vim user has to do lots more actions than the Eclipse user. Details of the methods are sufficiently different to not warrant further comparison at the operator level.

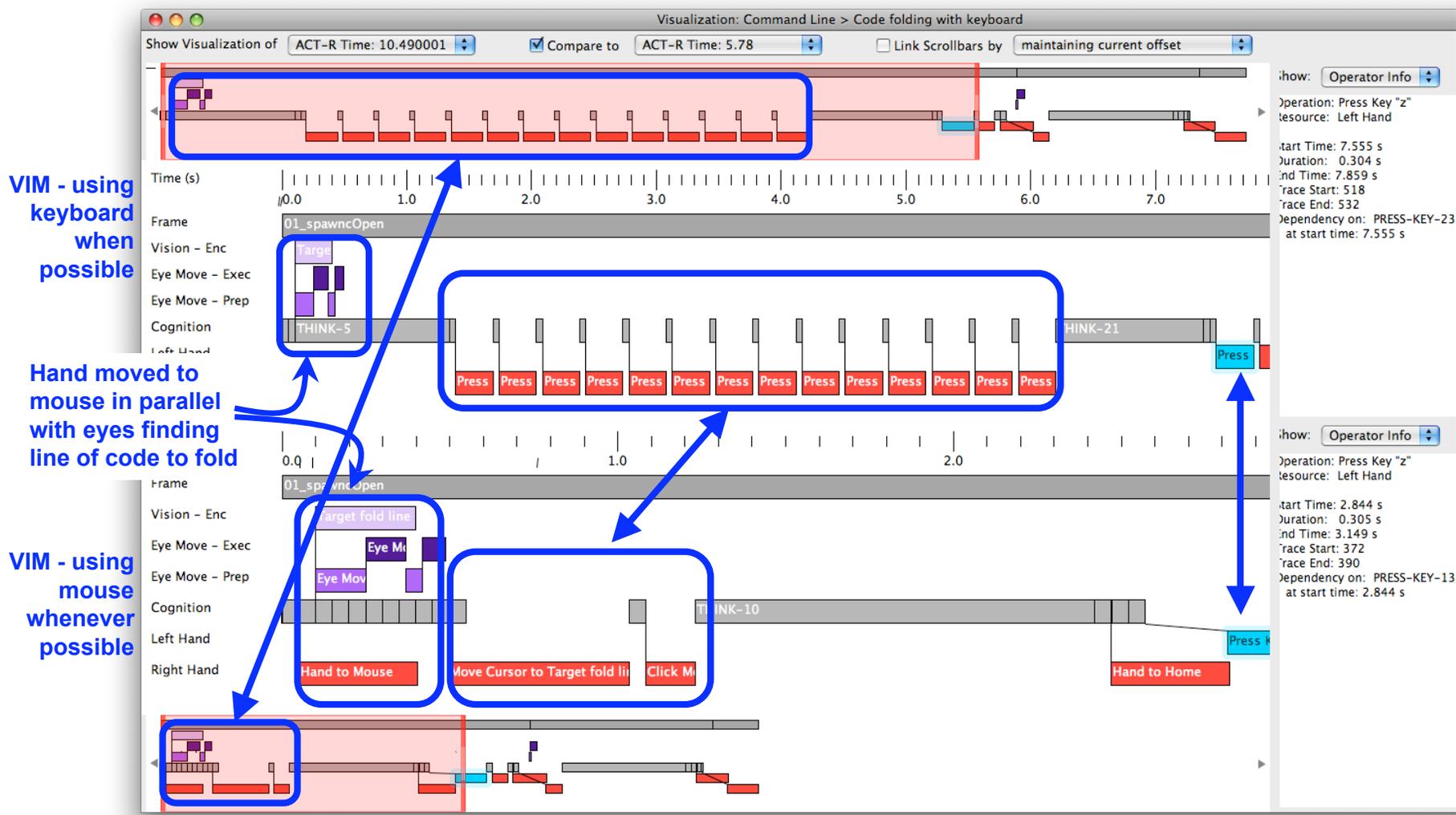
Running the Task on the Design



Tasks	Command Line	PTP-Eclipse
▼ HellowWorld_mpi	Min: 114.405 s	Min: 40.090 s
HelloWorld_mpi with keyboard	161.111 s	
HelloWorld_mpi with mouse	114.405 s	40.090 s
▼ F1Help	Min: 30.726 s	Min: 10.462 s
F1 Help with keyboard	30.726 s	
F1 Help with mouse	31.247 s	10.462 s
▼ Code Folding	Min: 5.780 s	Min: 3.563 s
Code folding with keyboard	10.490 s	
Code folding with mouse	5.780 s	3.563 s
Barrier Analysis	40.149 s	11.833 s

Understanding Differences Between Mouse and Keyboard VIM Methods

Task: Code Folding



Even though user preferring the mouse has to move their hand from the keyboard to the mouse, the user preferring the keyboard uses much more time to press all those navigation keys.

Some Reflections and Next Steps

- Difficult to understand tasks as neither modeler was a parallel programmer, nor the designer of the tools
 - Extract detailed task descriptions for all relatively high-frequency subtasks from empirical observations and interviews
 - Use manuals and documentation to recover how tasks were done in 2002
 - Rely on designers of 2010 tools to provide design storyboard
- Combine these estimates with other measures to produce overall productivity gain estimates for each workflow

CogTool Research Going Forward

- Augment CogTool using information foraging to model exploration. Allows models of:
 - Exploration of unfamiliar source code when debugging
 - Novice use of tools.
- Enhance tools to facilitate model creation
 - Import so designers can use their favorite tools to create storyboards and then easily import them into CogTool to obtain usability and accessibility metrics
 - Integrating model engine into design tools directly so usability and accessibility metrics are produced as side effect of ordinary development activities

Issues

- How much does tool time contribute to programmer productivity?
 - Drive tool-task ratio to zero
 - Intrusive tool-use interferes with creative tasks
- How much do programming languages and libraries contribute?
 - Ontology mapping (Blandford 1997) measures fit of language to problem domain, but doesn't provide task times.
- What about measuring the creative aspects of programming?

Thanks!

rachel@us.ibm.com

<https://researcher.ibm.com/researcher/view.php?person=us-rachel>