Using CogTool to Model Programming Tasks

Rachel Bellamy[1], Bonnie John[2], John Richards[1], John Thomas[1]

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

[2]Human-Computer Interaction Institute
Carnegie Mellon University

[1]{rachel, ajtr, jcthomas}@us.ibm.com [2]bej@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)

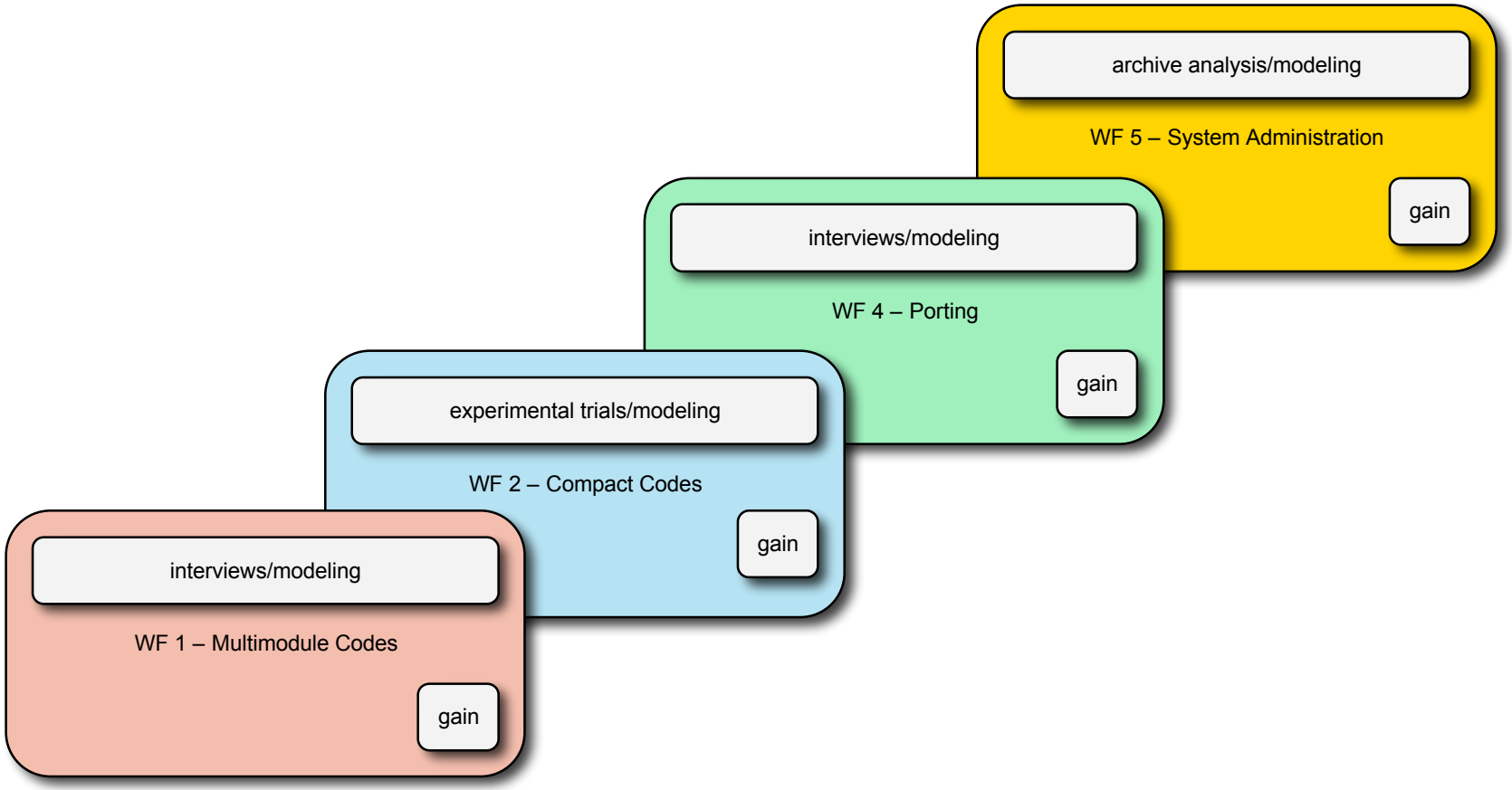
•Introducing new parallel programming language (X10) and cross language support for PGAS programming model

•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



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

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

taskTime = (executionTime * executionFrequency) / 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)

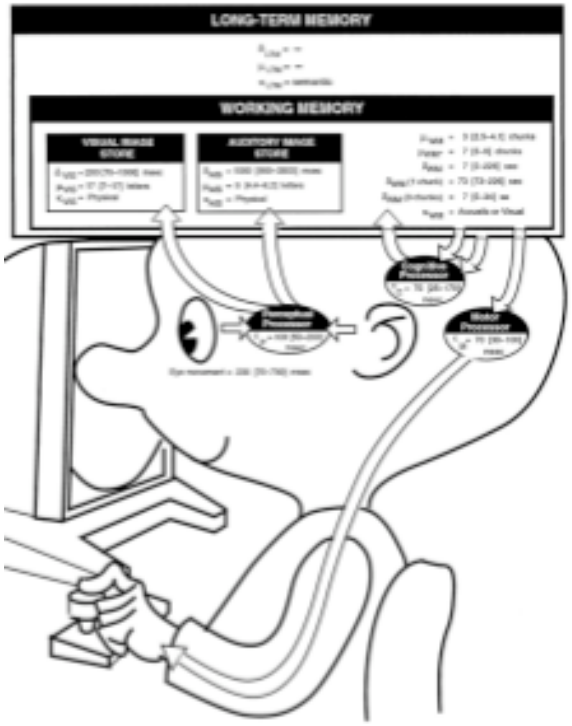
* for Workflows 1 and 4, ranges are 1-35 years parallel experience, 1 to 4000 kloc codes, 11 minutes to 2 years per task

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



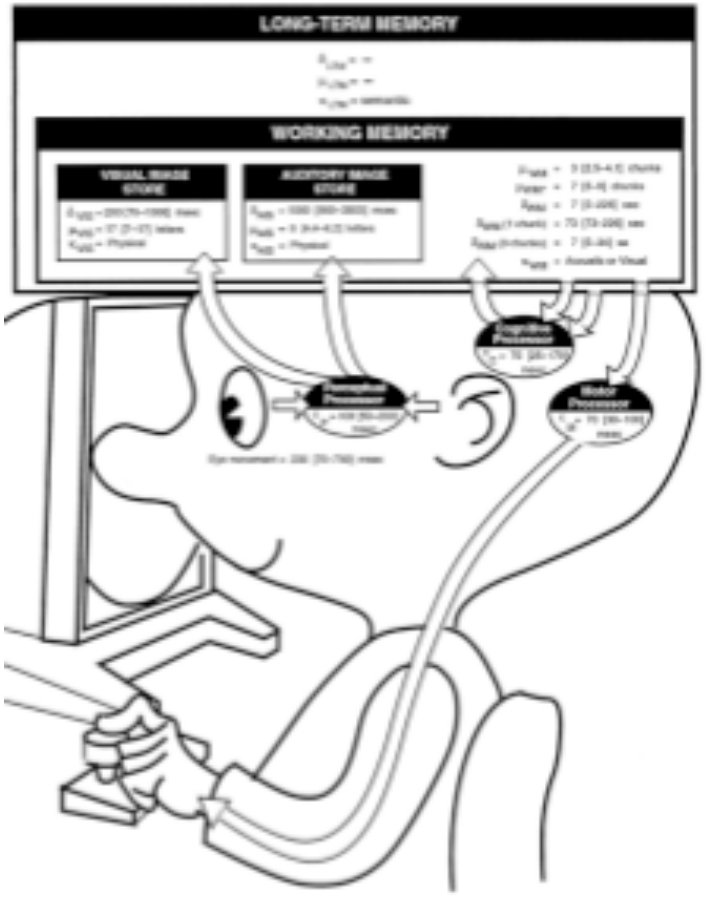
- Based on model human processor with perceptual, motor, and cognitive subsystems
- Accurate predictions of expert task times confirmed by over two decades of research

yields estimates of expert task time

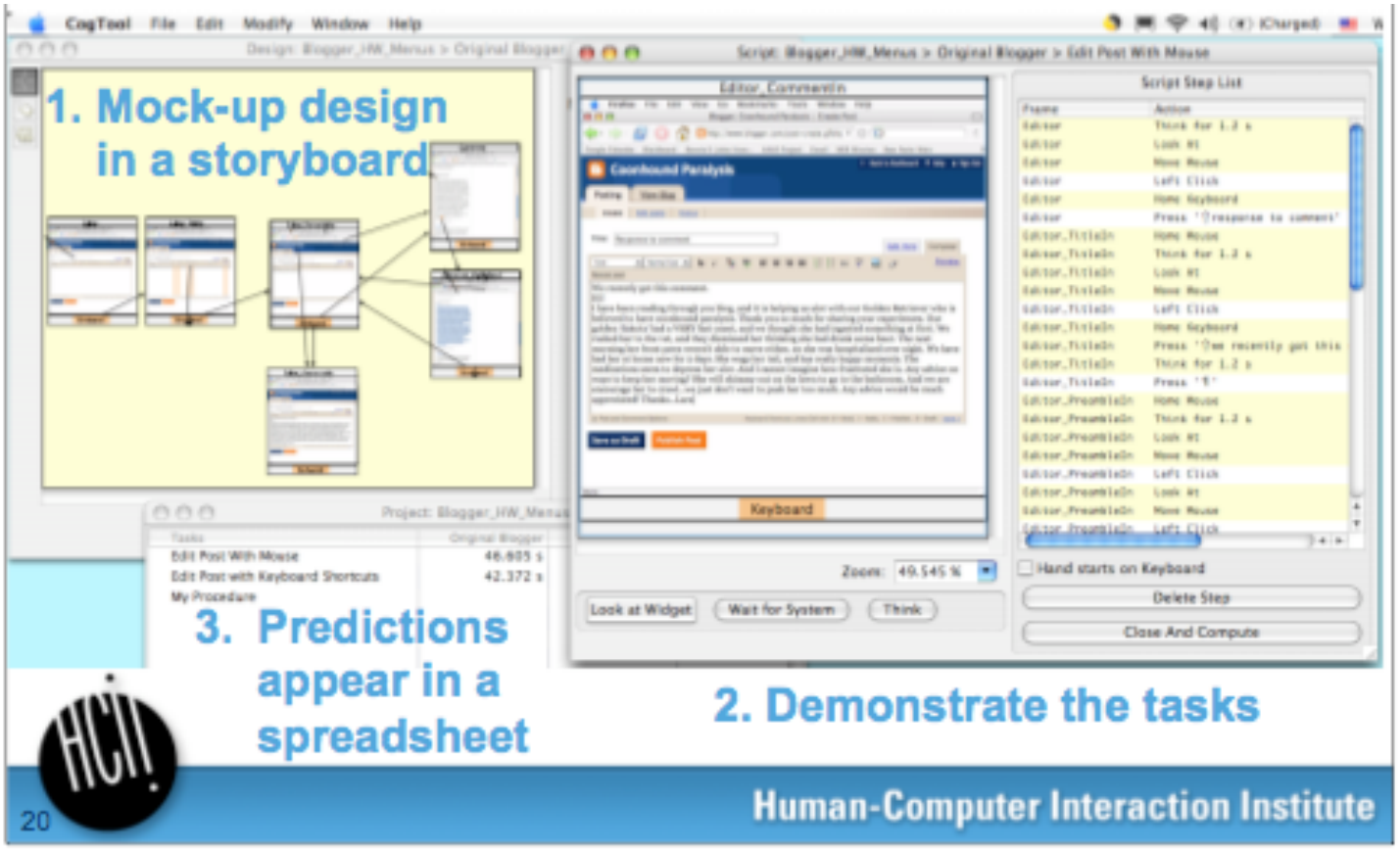
A Brief History of KLM

- **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”



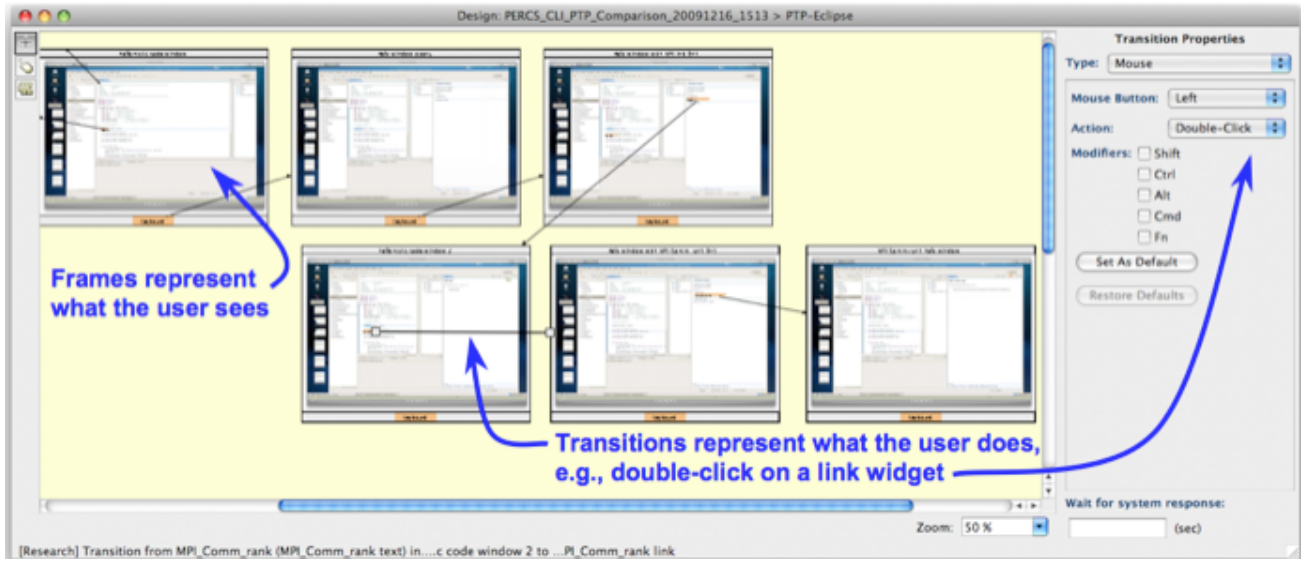
CogTool* (Bonnie John et al)



makes keystroke-level modeling practical

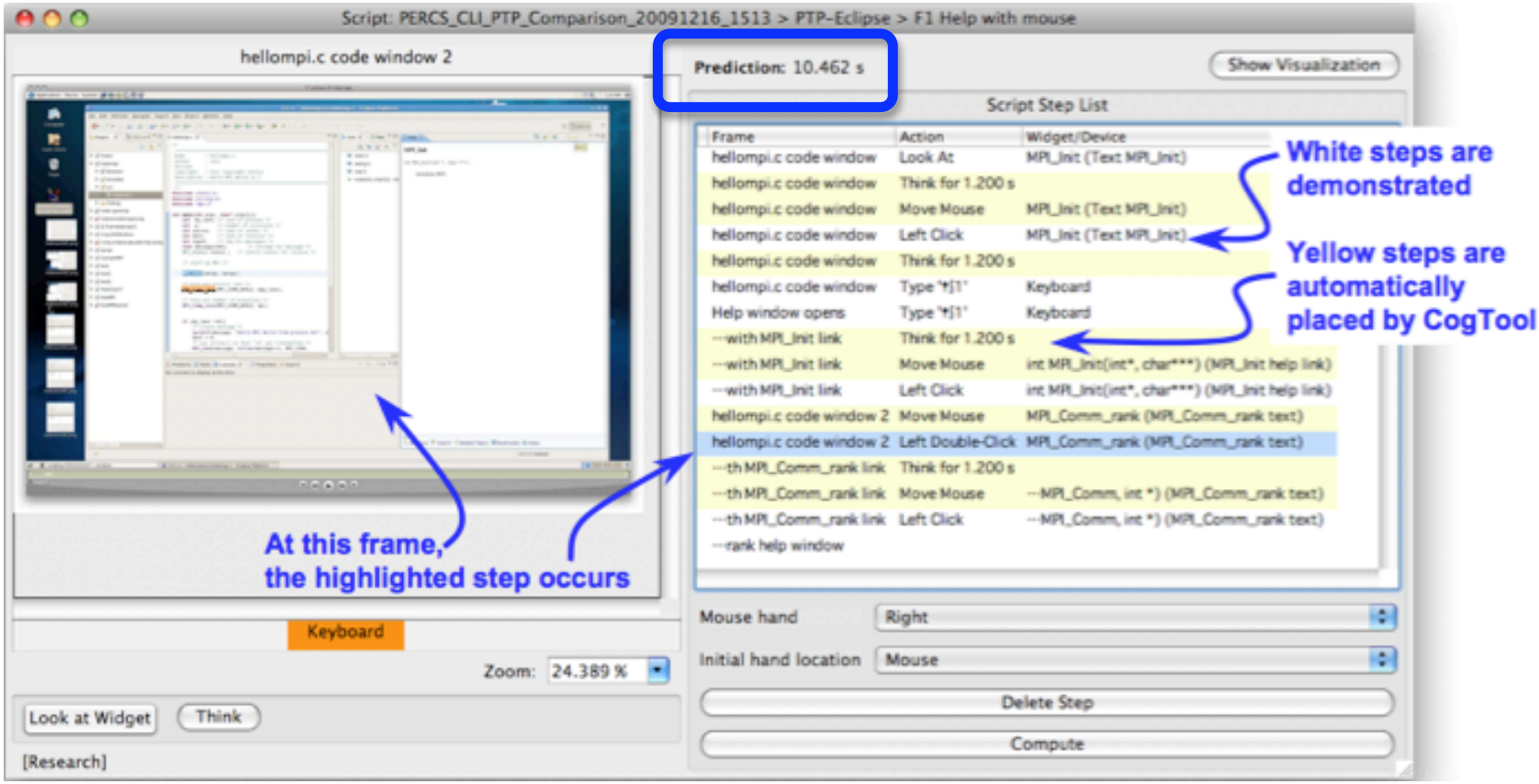
*http://cogtool.hcii.cs.cmu.edu/

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.

Demonstrate the Task



CogTool script window. The frame on the left shows the Eclipse window. The next step is to double click on a piece of code in that window (highlighted in the list of operators on the right).

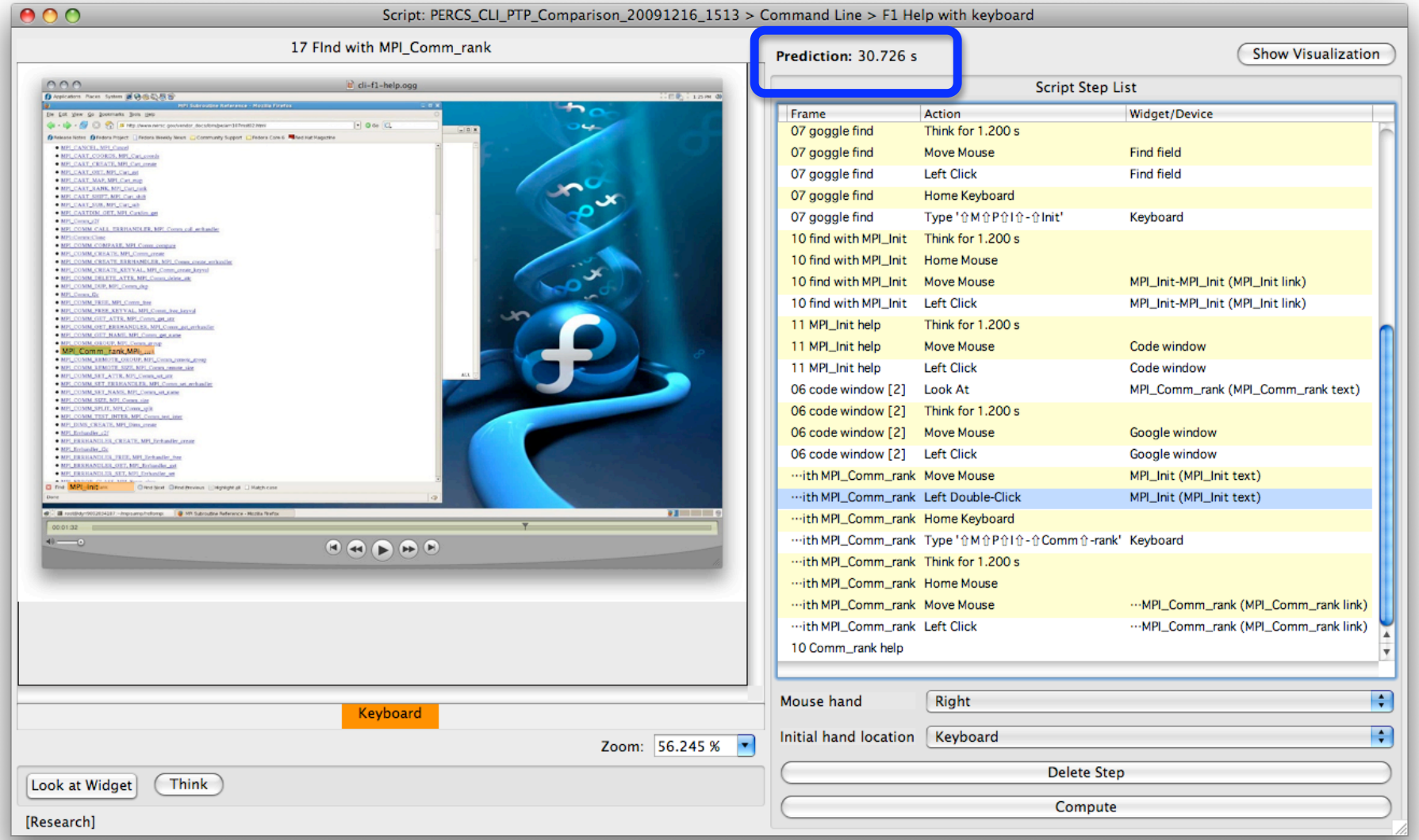
Produces Skilled Execution Time for the Task

* Project: PERCS_CLI_PTP_Comparison_20091216_1513

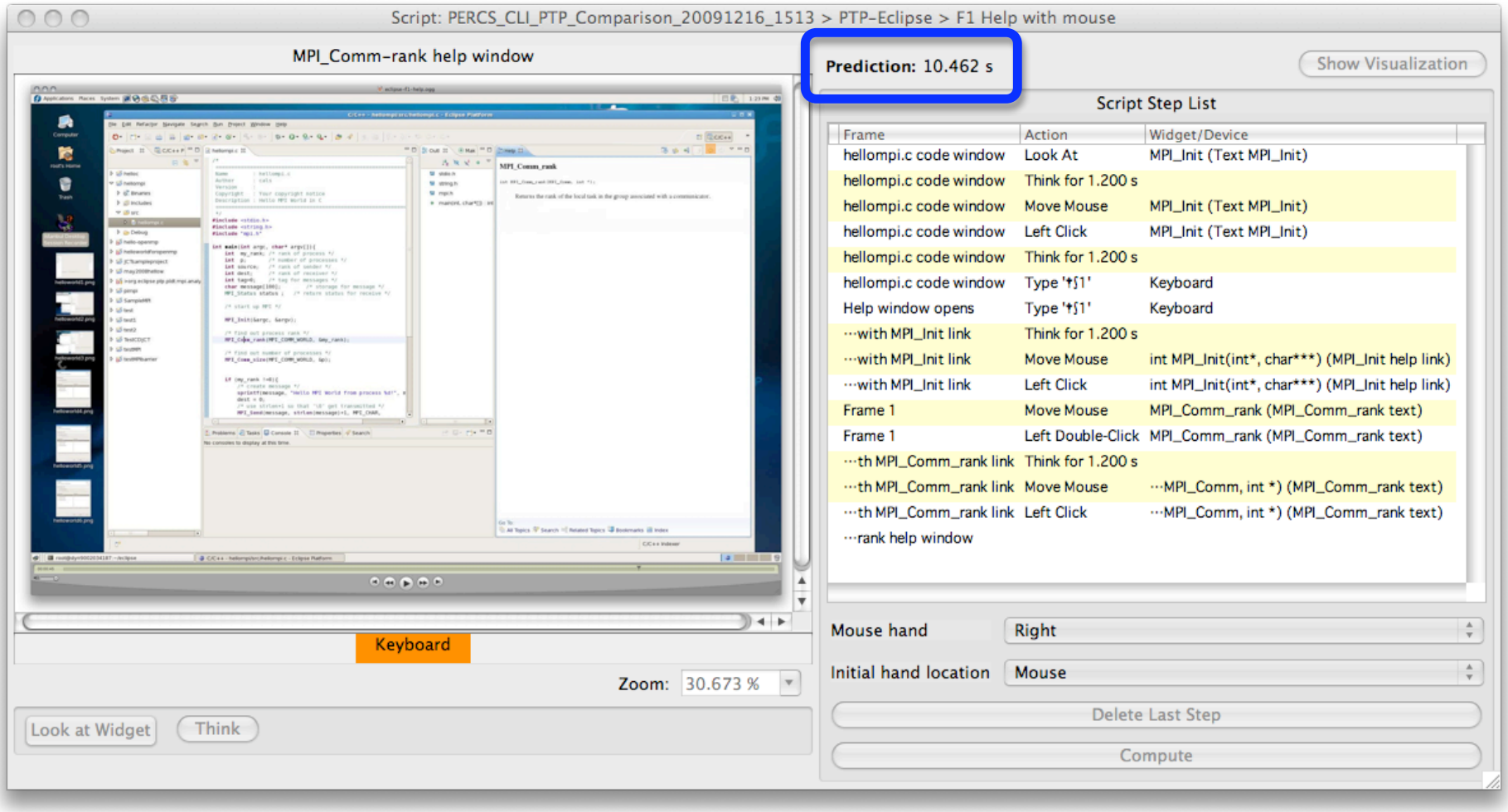
Tasks	Command Line	PTP-Eclipse
▼ HellowWorld_mpi	Min: 114.405 s	Min: 40.090 s
HelloWorld_mpi with keyboard	161.111 s	
HelloWorld_mpi with mouse	114.405 s	40.090 s
▼ F1Help	Min: 30.726 s	Min: 10.462 s
F1 Help with keyboard	30.726 s	
F1 Help with mouse	31.247 s	10.462 s
▼ Code Folding	Min: 5.780 s	Min: 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

CogTool results window showing estimated expert task times.

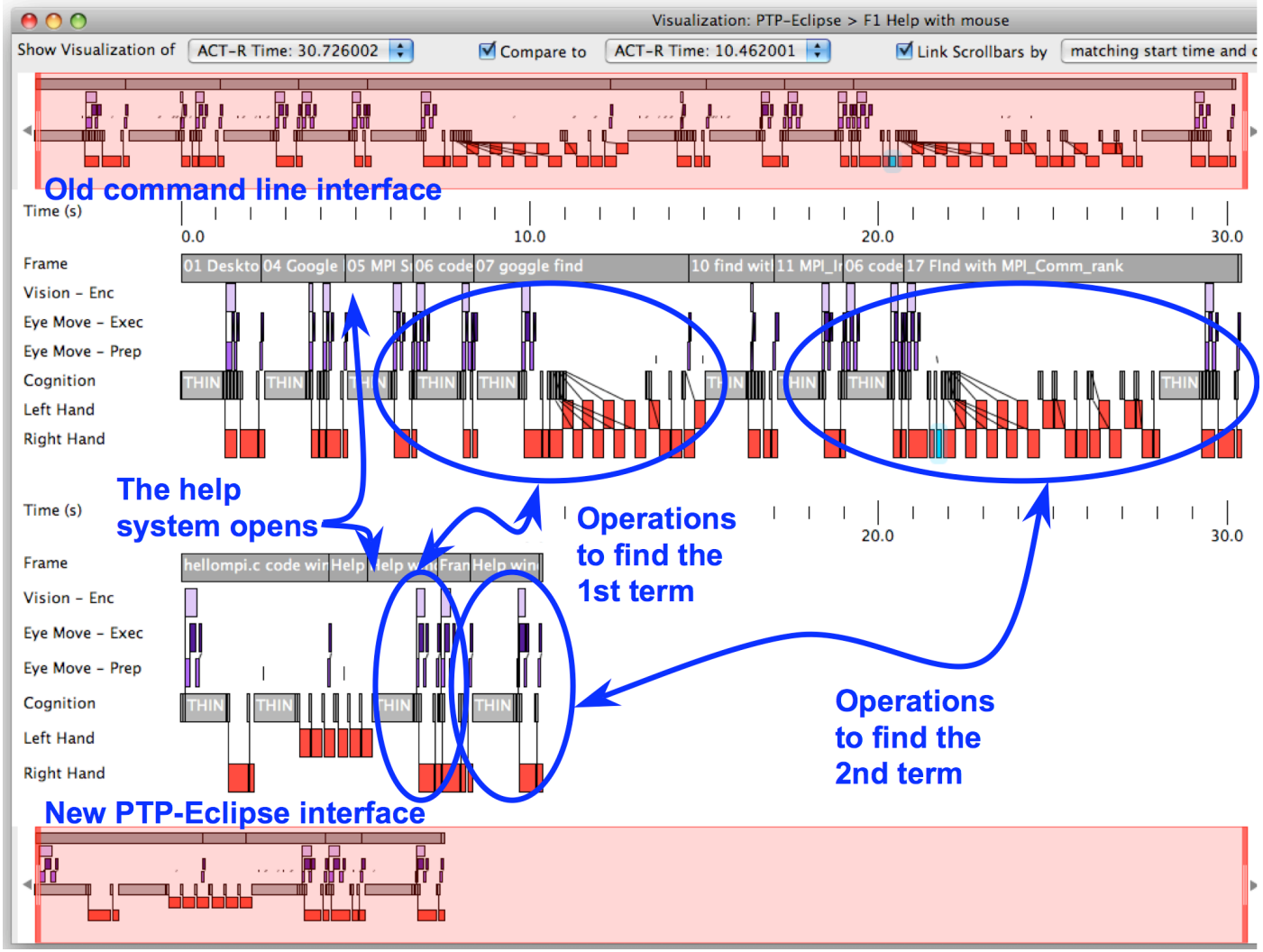
KLM – 2002 CLI Help



KLM – 2010 Eclipse PTP F1 Help

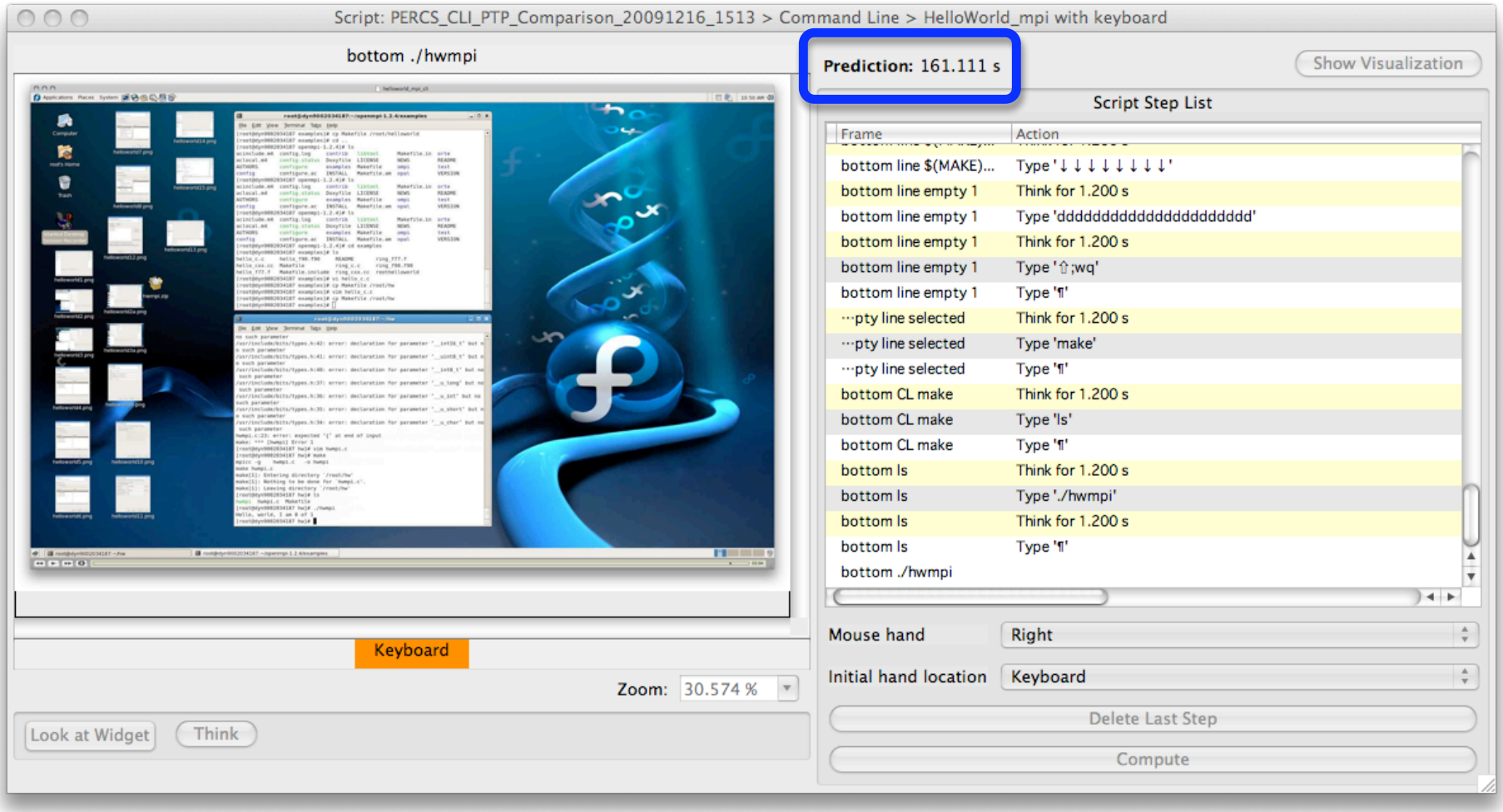


Understanding Differences Between Systems Task: Accessing Help

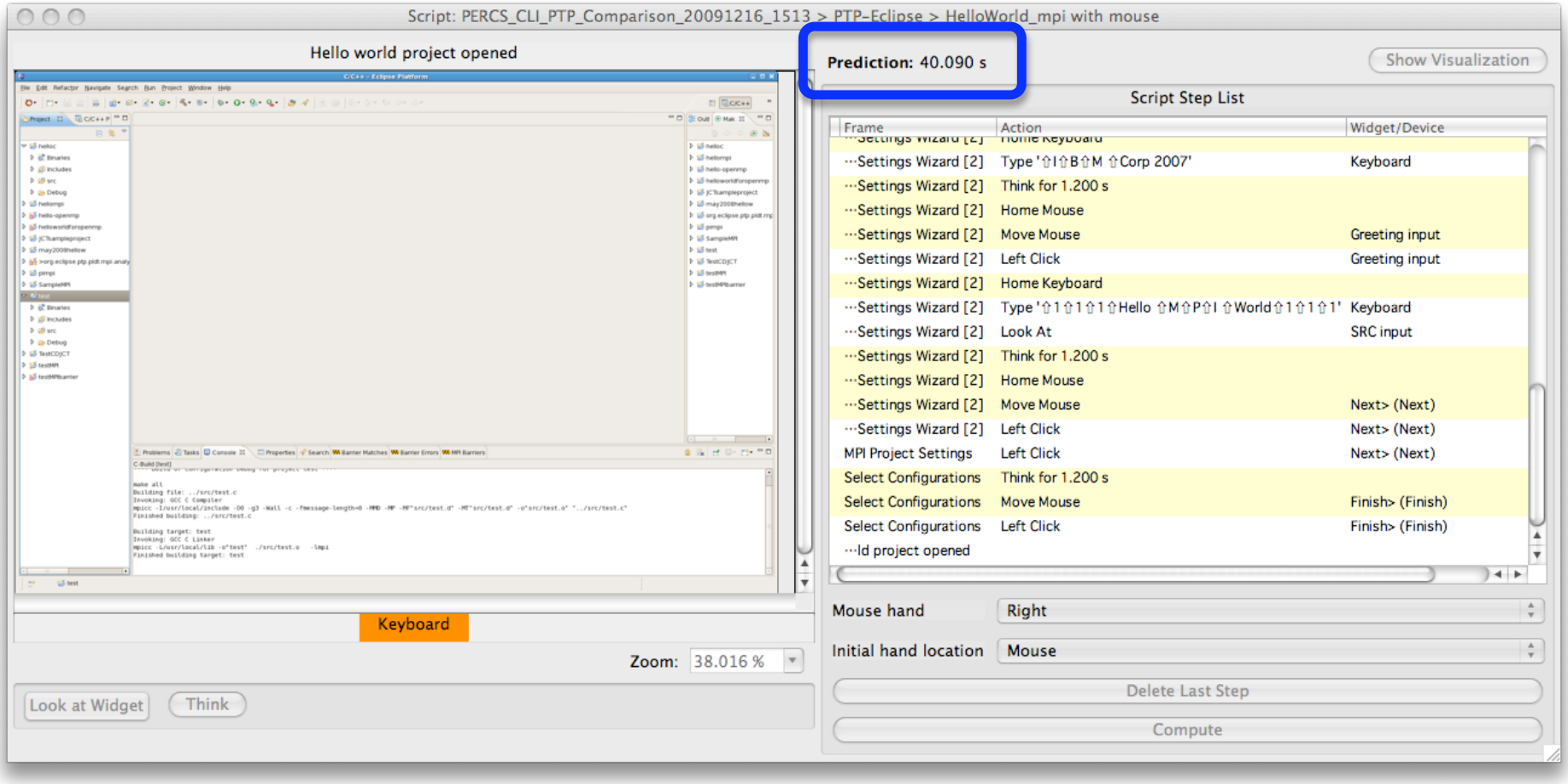


CogTool timeline visualization showing the differences between two systems.

KLM–2002 CLI MPI Project Setup from Template



KLM–2010 Eclipse PTP MPI Project Setup From Template



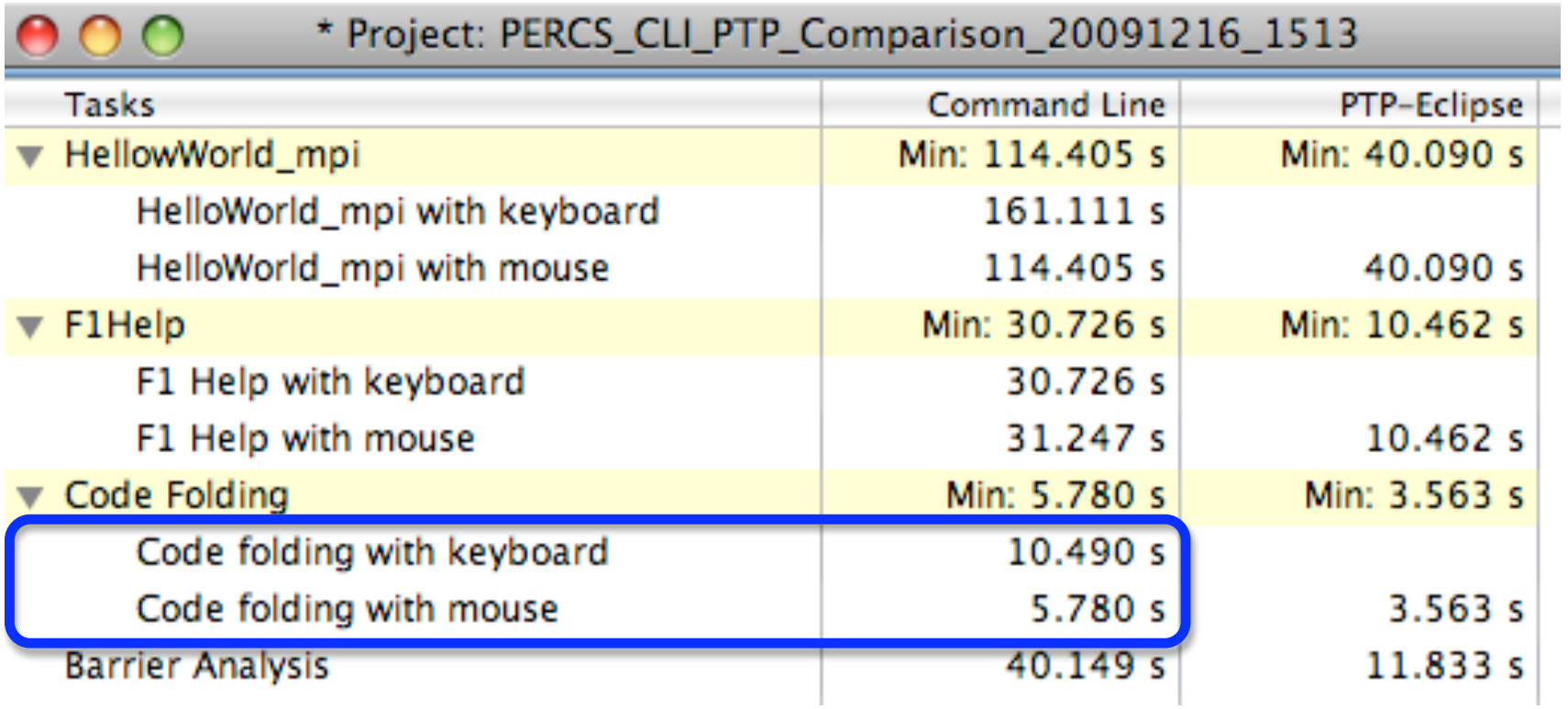
Understanding Differences Between Systems
Task: Project Setup From Template

VIM - using keyboard when possible

Eclipse PTP

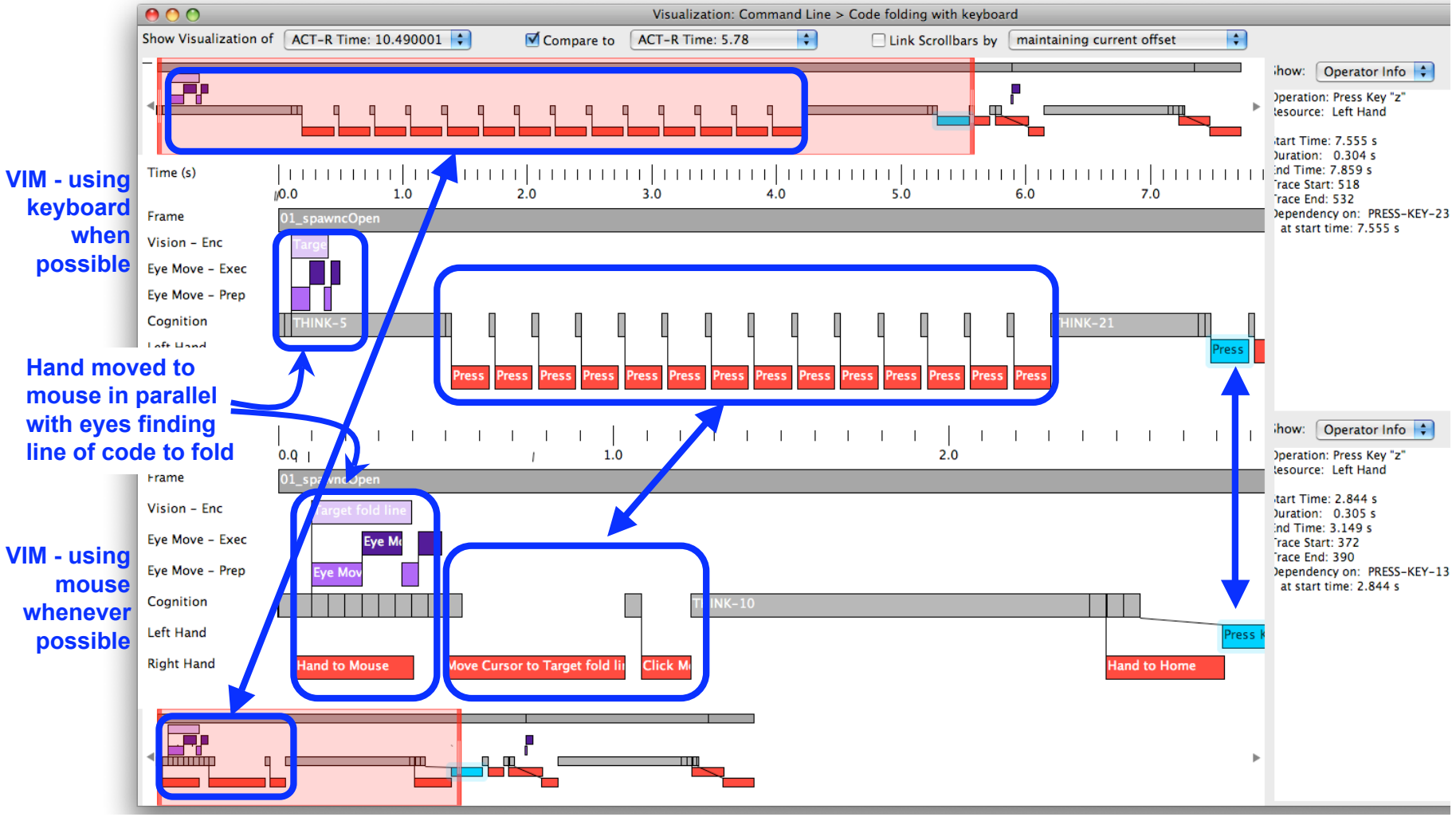
Vim user has to do lots more actions than the Eclipse user. Details of the methods are sufficiently different to not warrant further comparison at the operator level.

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 interfers with creative tasks
- How much do programming langauges 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