The Tycho User Interface System

Christopher Hylands
Edward A. Lee
John Reekie

UC Berkeley
Dept. of EECS
Ptolemy: heterogeneous modeling

• Design methodology for reactive systems—signal processing, telecommunications, hardware-software codesign

• Heterogeneity—multiple, domain-specific, semantic models:
  • synchronous/reactive models
  • multi-rate dataflow
  • discrete-event simulation
  • modular hierarchical finite state machines
Ptolemy: interactive, high-level simulation

Author: Uwe Trautwein, Technical University of Ilmenau, Germany
**Tycho**

**Goal:** to extend the non-dogmatic nature of the Ptolemy kernel—multiple semantic models—to the user interface. Thus: multiple syntactic models.

**Authors:**

Edward A. Lee  
Christopher Hylands  
John Reekie  
Kevin Chang  
Cliff Cordeiro  
Wan-geh Chang  
Wei-Jen Huang  
Joel King  
Farhana Sheikh
Motivation

• Cross-platform portability
  ✔ [incr Tcl]/[incr Tk] — object-oriented scripting
  ✔ Java — portable, clean, system-level language
  ✗ C/C++ — ?

• New Ptolemy GUI
  ✔ Domain-specific graphical editors
  ✔ Design visualization and exploration
  ✗ Binary internal representations, complex databases

• Extensible, re-usable user-interface framework
  ✔ Sub-classable user-interface widgets
  ✔ Application framework
  ✗ Ad-hoc Tcl scripts
Frameworks: UI layout

- A displayer contains views
- Menu bar, tool bar, and status bar come “for free”

Theoreticians typically strive for simplicity, looking for the smallest set of primitives on which they can build a general methodology. In computer science theory, for
Frameworks: file management

- File load/save policies
- Access management
- Look-and-feel

Inherited policies
Frameworks: documentation

Multiple languages:
- ✔ [incr Tcl]
- ✔ Java

Class :tycho::WidgetPreferences

The WidgetPreferences class implements the Tycho preferences manager. It is a sub-class of Model, and uses the TIM file support and the

excludeoptions group widget args

Exclude one or more options of a widget from the subscription mechanism. Once excluded, requests
Component are customizable by:

- parameters
- inheritance

```::tycho::query "Do you like modal dialogs?" {
    {line address "Address of recipient" {johnr@eecs}}
    {lines contents "Contents of message" {No, not really} 4}
    {check auto "Send automatically?" 1}
}```
Model-view architecture

- Flexible data representations
- Unbounded history mechanism
- “Publish-and-subscribe”
Hierarchical graphics and interactors

An icon being connected with the mouse

IconFrame

Frame

Terminal

Stepper

Bounder

DragDropper

SmartLine
Custom simulation interfaces

- Ptolemy generates a Tcl-Tk package, which Tycho compiles
- User-interface uses Tycho infrastructure
- Tycho “schedules” real-time executions

Star parameters bound to custom control panel by name
Tcl ↔ Java

- WELD i/f prototype: Tcl U/I, Java network access
- Updating needed: Tcl8.0, Java 1.1.x, object references

::java::new VeryUseful foo
set bar [foo sqrt 42]

6.48074

TclJava0.4a
Tcl7.6
Java1.1.1
Status

• Version 0.2 released June 97
  
  http://ptolemy.eecs.berkeley.edu/tycho/

  ✔ Application framework
  ✔ Text editors and user interface components
  ✔ Automatic documentation
  ✈ Graphical and tree viewers, Tcl profiler

• Version 0.2.1 scheduled for Fall 97

  ✈ Itcl 3.0 (the byte-compiler version!)
  ✈ New Java interface
  ✔ “Network-aware”: HTTP, FTP, mailto, WELD
  ✈ Graphical editing support
Concluding remarks

- For user interface development, Tk is flexible and powerful.
- \[\text{[incr Tcl]}\] and \[\text{[incr Tk]}\] greatly improve program structure and clarity.
- \[\text{[Incr Tcl]}\]/\[\text{[incr Tk]}\] is portable: Tycho runs on UNIX and Windows.
- Development at this scale (85 kloc) is hampered by poor performance and lax parsing.
- Our experiments with Tcl + Java are very positive:
  - Java is easy to learn and use, and has extensive libraries.
  - Tcl/Tk simplifies user interface development and scripting.
  - An updated, platform-independent interface is essential.
- Hearty thanks to those who developed infrastructure we have used: Michael McLennan, John Ousterhout, Stephen Uhler, Mark L. Ulferts.