The Future of the Ptolemy Project —
Design of Distributed Adaptive Signal Processing Systems

Edward A. Lee
Principal Investigator

UC Berkeley
Dept. of EECS

DARPA DASP Project

Phase 1 (11/96 — 5/98)

- Modular deployable design tools
- Domain-specific tools for distributed adaptive signal proc.
- Models for dynamically configured systems

Phase 2 (6/98 — 11/99)

- Process-level type system
- Formal analysis and debugging
- System-level visualization

Option (5/97 — 5/98)

- Array formalism for multidimensional signal processing

Adaptive Systems

Classical adaptive signal processing

- system identification
- interference nulling
- reversing distortion

Resource adaptive signal processing

- conserving power
- meeting changing latency and QOS requirements
- using available sensor data
- using network resources (memory, cycles, bandwidth)

Modular Deployable Design Tools

Past design software:

- Monolithic
- Huge
- Back-room use

Future design software:

- Modular
- Deployable
- In-the-field evolution

UNIVERSITY OF CALIFORNIA AT BERKELEY

Initial Strategy

Toolkit approach to design, creating an environment that is

- **safe (no core dumps)**
- **extensible**
- **distributable**
- **concurrent**
- **portable**

Deployed designs must minimize the use of

- **C, C++**
- **Thus, most of the existing Ptolemy kernel**

Initial Languages

In addition to satisfying all the above,

Tcl/Tk/Itcl

- **scripting language**
- **high-level, object-oriented**
- **universal, communicable data type (strings)**
- **extensive graphical user interface toolkits**

Java

- **faster (we have measured up to 8x)**
- **lower-level, object-oriented**
- **modularity built in**
- **concurrent (threads), although at a very low level**

Tycho

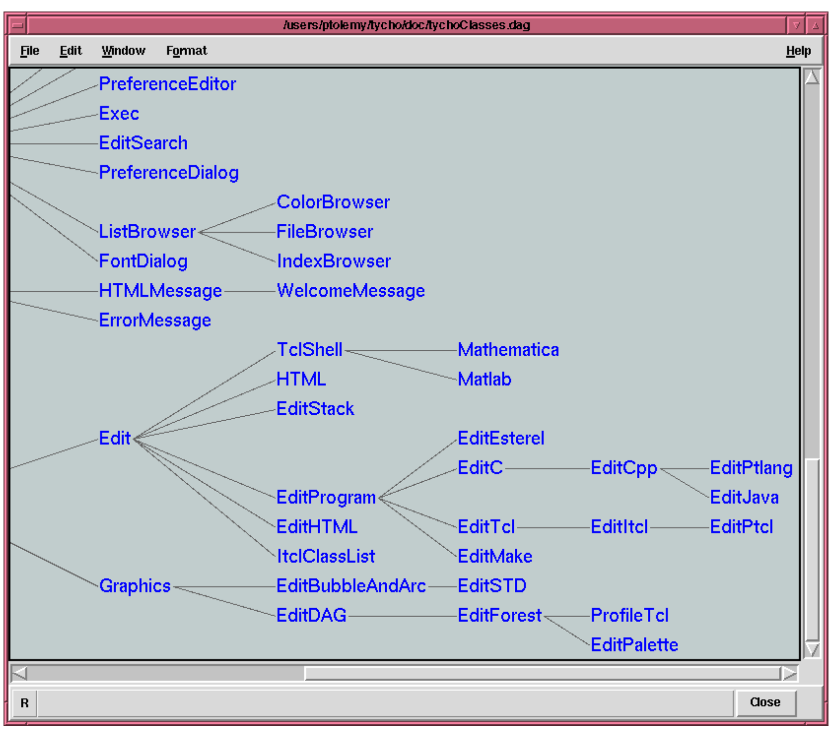
Modular Itcl class library

- system control
- configuration
- user interface

Current facilities:

- context-sensitive text editors
- scripting shells (Tcl, Matlab, Mathematica)
- graphics toolkit (the Tycho Slate)
- integrated, interactive, HTML documentation
- preferences manager, version control, widget library

A Portion of the Class Hierarchy (displayed in Tycho)



UNIVERSITY OF CALIFORNIA AT BERKELEY

The Tycho Slate

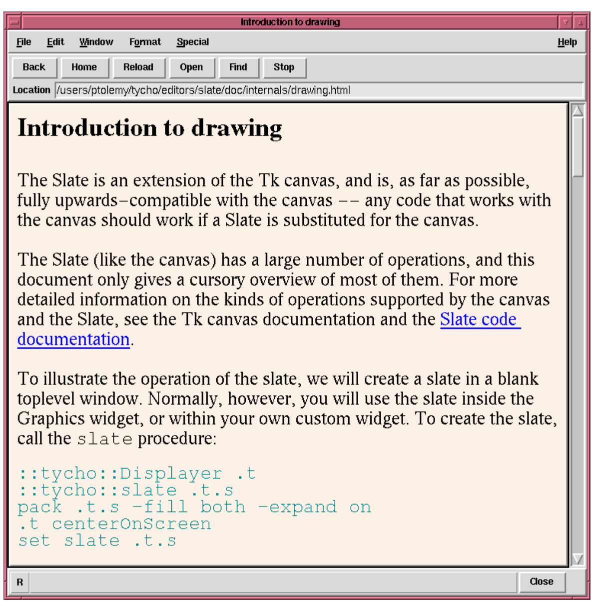
Extends the Tk canvas supporting

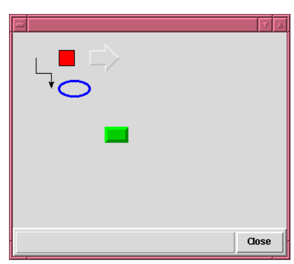
- **creating complex items,**
- **re-using common patterns of user interaction.**

There are two key uses of the Slate:

- **As a higher-level canvas for building graphical displays and editors. The Slate is used this way within the Graphics class and subclasses.**
- **As a toolbox for rapidly building custom widgets. The Slate is used this way to create some of the custom widgets used in Ptolemy C-code-generated systems.**

Integrated, Interactive Documentation





In the above example, clicking on the Tcl code at the bottom executes the code, creating the example slate on the right.

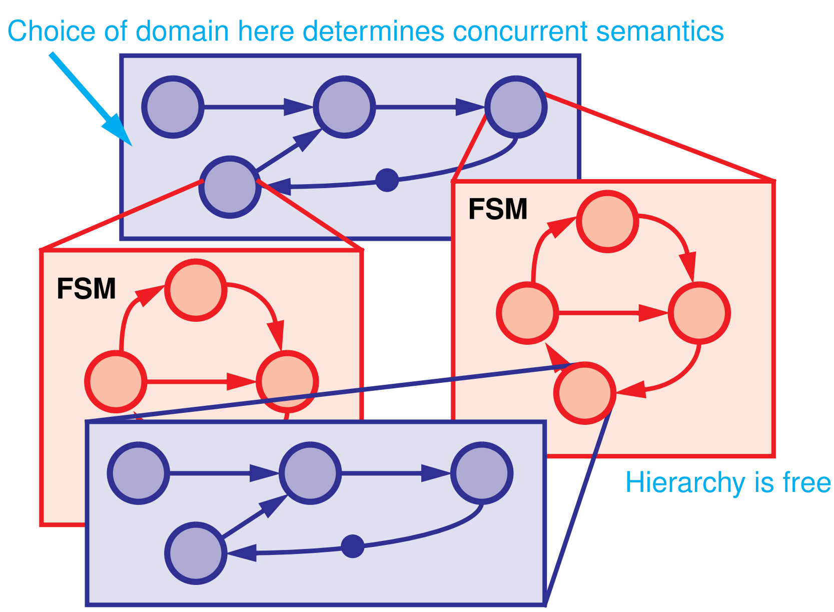
Progress To Date with Java

- implemented a Kahn process network model in Java
- outlined a dataflow/FSM hybrid model of computation
- installed Sun's tcljava interface
- 'tycho -java' starts up java with Tycho
- can reparent Java applet viewer within Tycho

Problems with Java/Tycho Interface

- Java documentation system is not compatible with Tycho's
- Need a communication architecture for Java/Tcl
- Need to evaluate effectiveness of just-in-time compilation

Sequential Control Mixed with Concurrency: *Charts



Determinate, Concurrent, Distributable Java Design

Based on a hierarchy of four increasingly expressive dataflow models:

- Synchronous dataflow
- Boolean dataflow
- Dynamic dataflow
- Kahn process networks

Combined hierarchically with

- Finite State Machines (FSMs)

Properties of the FSM/DF Hierarchical Nesting

- The combined models are more expressive than either DF or FSMs alone.
- In certain cases, the combination is still finite state, so critical questions remain decidable.
- Advanced visual syntaxes can be used for both models (but we need a heterogeneous syntax manager: Tycho).
- The formal properties of both models can be independently exploited to get high-quality synthesis of software or hardware.
- The combination achieves:
  - High performance number crunching (DF).
  - Control, mutability, and on-line decision-making (FSM).
  - Hierarchy (Ptolemy).
  - Concurrency (DF).

Alternative Concurrency Models

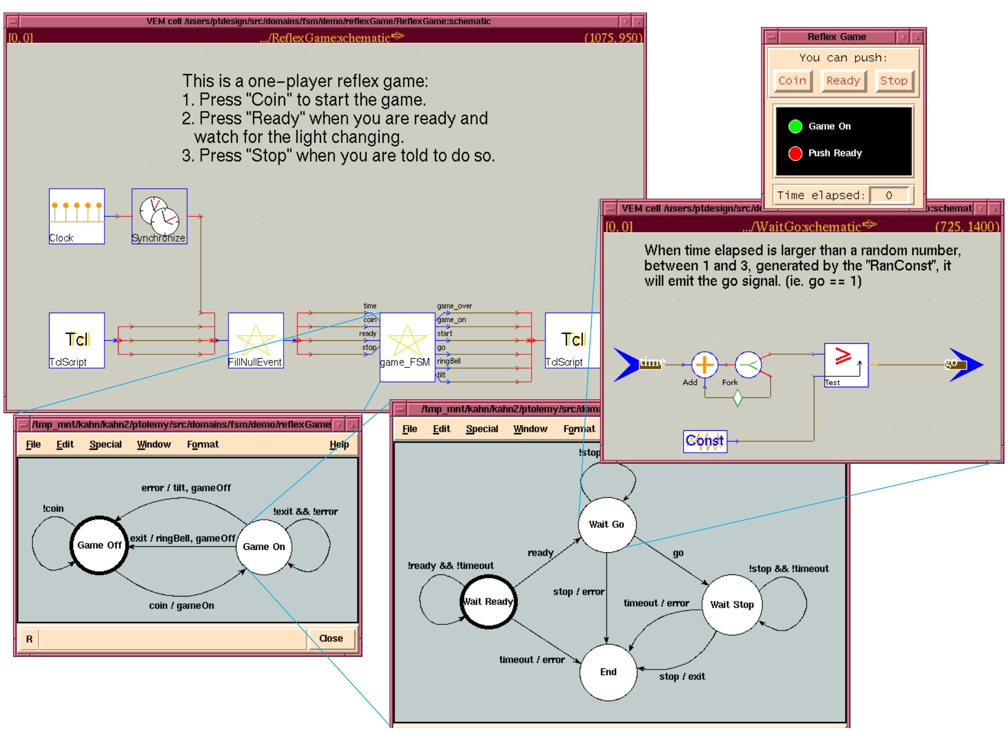
- **Synchronous/reactive systems (Esterel, Lustre, Signal, Argos) execute in a sequence of global *ticks*. Signals have values at each tick, and these values form a *fixed point* (x = f(x)).**

- **Discrete-event systems, also called *virtual time* systems (VHDL, Verilog, and some communicating FSMs), maintain a global notion of (simulated) time and execute chronologically.**

S/R systems are more tightly coordinated than dataflow systems, have strong formal properties, and can be implemented in software or hardware.

DE systems are expensive to implement in software but useful for modeling concurrent real-time systems.

Example: DE, Dataflow, and FSMs



Semantics — What does it Mean?

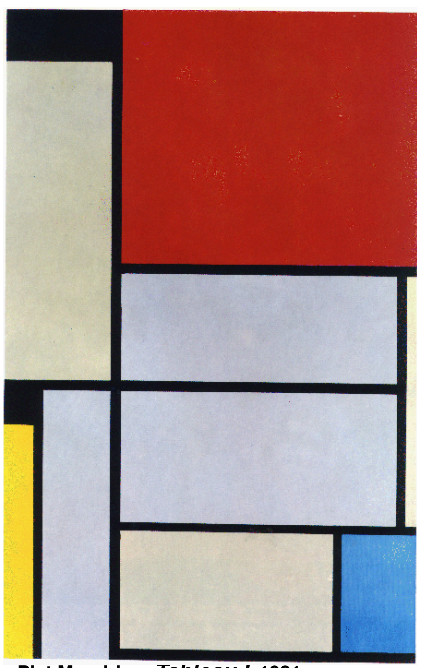
This well-established field addresses many of the core problems in the specification and modeling of concurrent systems:

Issues

- Concurrency
- Synchronization
- A model of time
- Turing completeness
- Determinacy
- Finite state
- Redundancy

Instascan MRI image of a brain responding to light stimulation.

Abstraction



Piet Mondrian, *Tableau I*, 1921

Abstraction is the act of pulling away or withdrawing from the physical properties of the implementation...

... closer to the problem domain,

... facilitate specification,

... avoid overspecification,

... hide irrelevant details,

... facilitate design re-use,

... facilitate design validation.