

# TclSim

A DSEL for Simulation

Matt Walker

# Overview

- Introduction to simulation
- Ideal language features
- Implementation issues

# Simulation

- Some systems resistant to formal description
  - Operating systems
  - Networks (argue with Dr. Misra)
  - Transportation
- How do we predict behavior?
  - Implementation is expensive

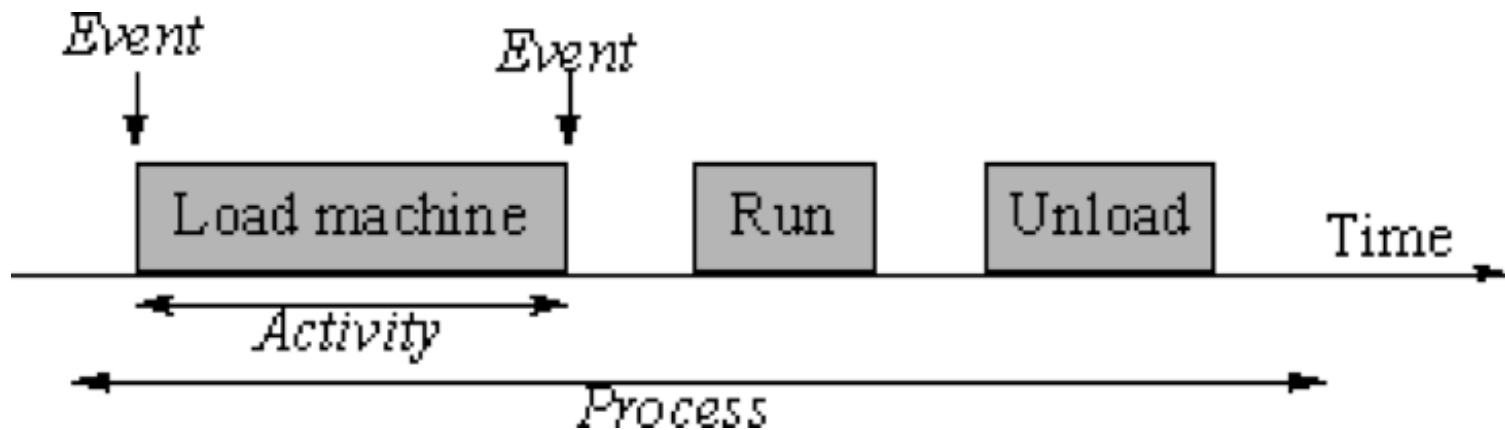
# Simulation

- State
- Time
- Behavior
- Statistics

# Discrete-Event Simulation

- Time of state changes discrete
- Events signal state change
  - Demarcate activities
  - Grouped together into processes
- Algorithm:
  - Maintain time-ordered queue of events
  - Perform earliest, inserting any new events into queue

# Process-Based Simulation



- Processes are sequences of events
  - Units of sequential behavior
  - Maintain local state
  - Advance simulation time by holding

# Other Aspects

- Resources
  - Finite capacity => competition
- Statistics
  - Measure resource use
  - Track system state change

# Language Features

- Procedural
  - Sequential descriptions
  - Side-effect system state
- Processes
  - Relative ordering of events and durations
  - Not absolute times
- Simula had it right!
  - Class = behavioral description
  - SimPy takes this approach

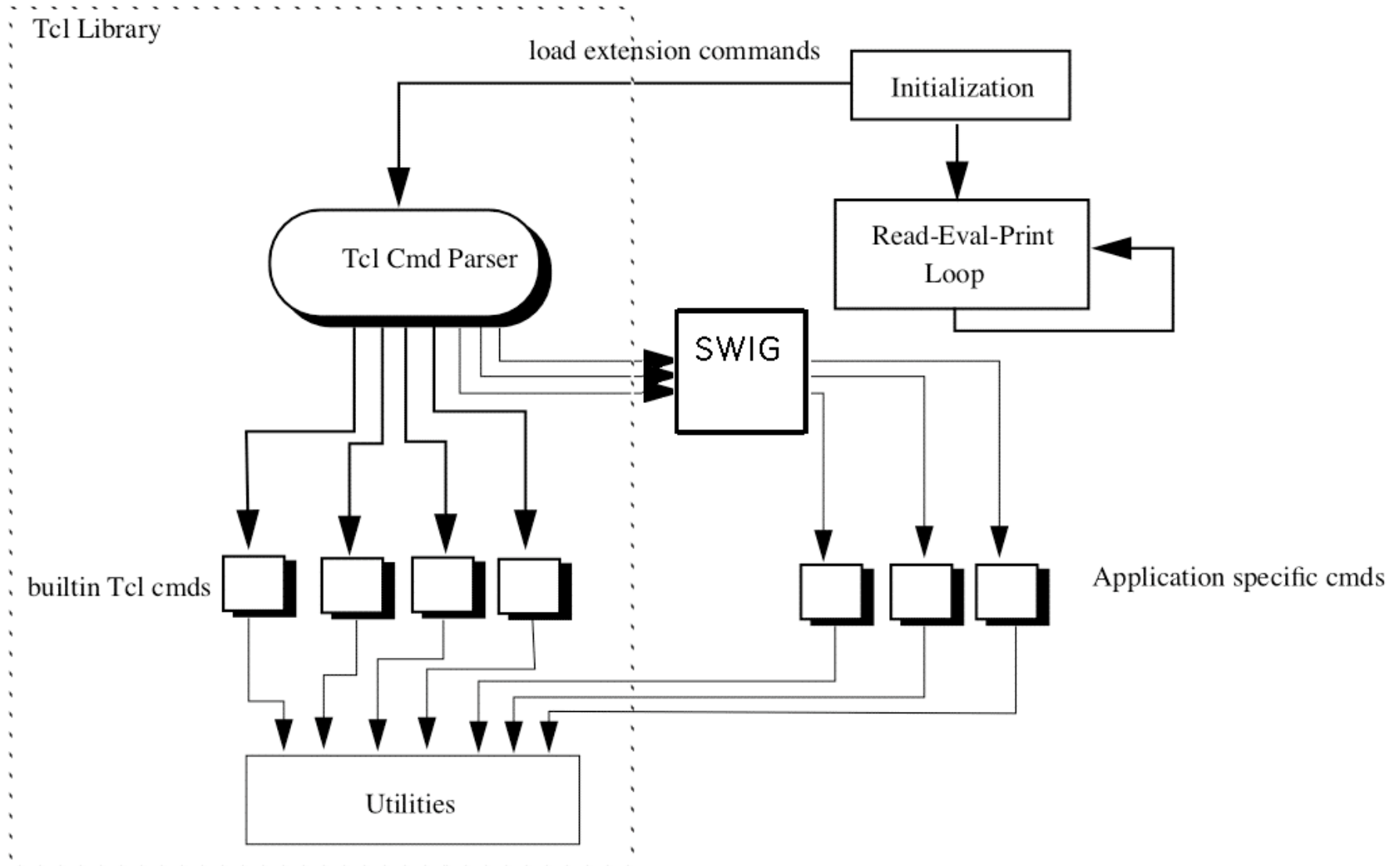
# Simulation Implementation

- Libraries are most common form
  - Provide full power of implementing language
  - Can be statically checked
  - Typically compiled; good for large simulations

# But...

- Simulation is experimental
  - Rapid prototyping valuable
  - Visualization desirable
- Idea: Tcl
  - Interpreted scripting language
  - Tk provides excellent visualization

# TclSim Implementation



# CSIM's Process Model

- Process behavior defined via “function” that calls `create()`
- Deadly sin of breaking programming abstractions
  - `create()` converts function to process
  - Copies activation record into heap
  - Subsequent `hold()` calls may re-activate “function”

# Glue: Process Behavior

- Wrap all calls to `create()` with code that interprets behavior defined in Tcl
- Each process makes its own interpreter
  - Interpreter requires state (environment, execution stack, etc.)
  - Cannot reuse same interpreter
  - How do processes communicate?

# Glue: Global State

- CSIM requires global state
  - Easy, efficient coordination
  - System statistics
- Tcl\_HashTable excellent tool
  - Maps names to values
  - “Lives” in surrounding C environment
- Tcl\_LinkVar()
  - Makes Tcl name an alias for C value

# SWIG

- Given CSIM headers, generates Tcl interface to library
  - 11.5k lines of code
  - Requires only minor modification
- Also used for glue code

# Conclusions

- SimPy stole my ideas!
  - Object-oriented
  - Uses Tk for visualization
  - Statistics can be automatically collected

# Future Work

- Resolve process model issues
  - Get source for CSIM
  - Choose another library
  - Write one in Tcl
- Tk visualization
- Automatically generate statistics code
- Leverage interpreter to output C for compilation of large simulations