Soar Lite:
Designed for Speed

Scott Wallace

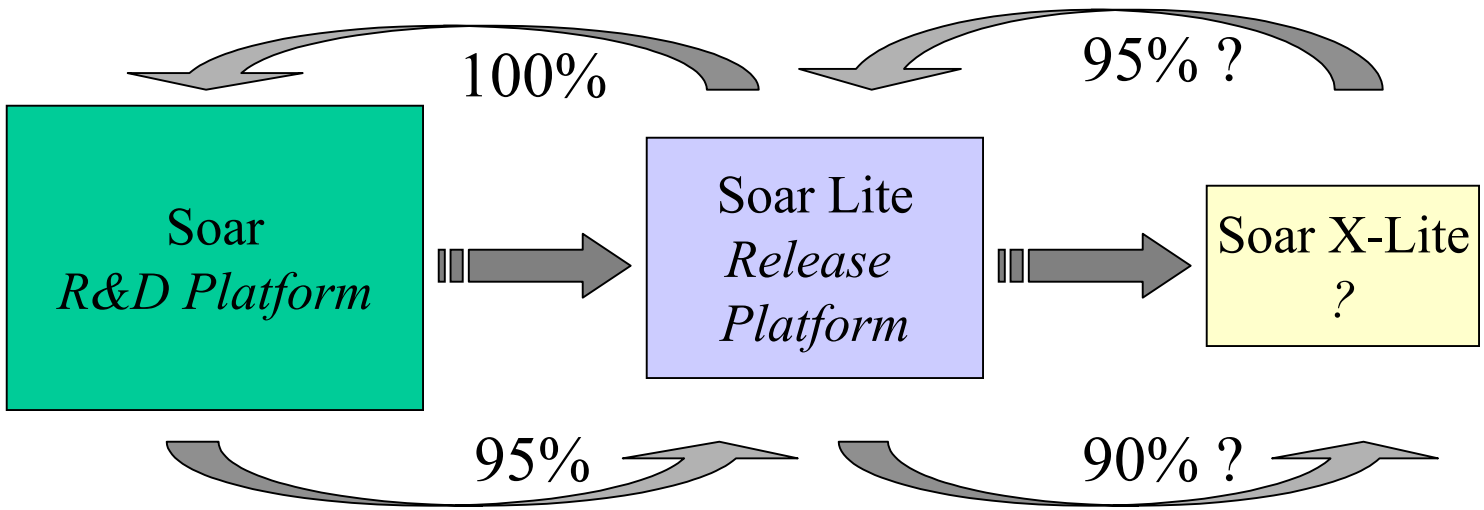
Karen Coulter

Motivation

- Meet performance requirements of some interesting domains
  - Video games
  - Mobile robots
- Want a embeddable, low-profile version
- Soar/CLIPS comparison revealed two things
  - simple modifications can greatly change performance
  - some architectural capabilities appear very costly

Original Design Goals

- Increase overall speed
- Maintain current production semantics
- Keep a core set of the architectures capabilities
- Create an API



Our Approach

❶ Modularize the Soar Architecture
- Determine a set of "core" functionality
- Define a set of modules
- Allow users to specify which modules should be included

❷ Benchmark the Architectural Variants

Soar's Core Functionality

- Loading a rete-network
- Allows execution, termination of agents
- Standard decision cycle
- Hierarchical goals
- I/O to and from an external environment

What was Modularized

- *Timing Facilities (Bob Wray)*
- High-cost Callbacks
- Learning/Justifications
- Backtracing/GDS Support

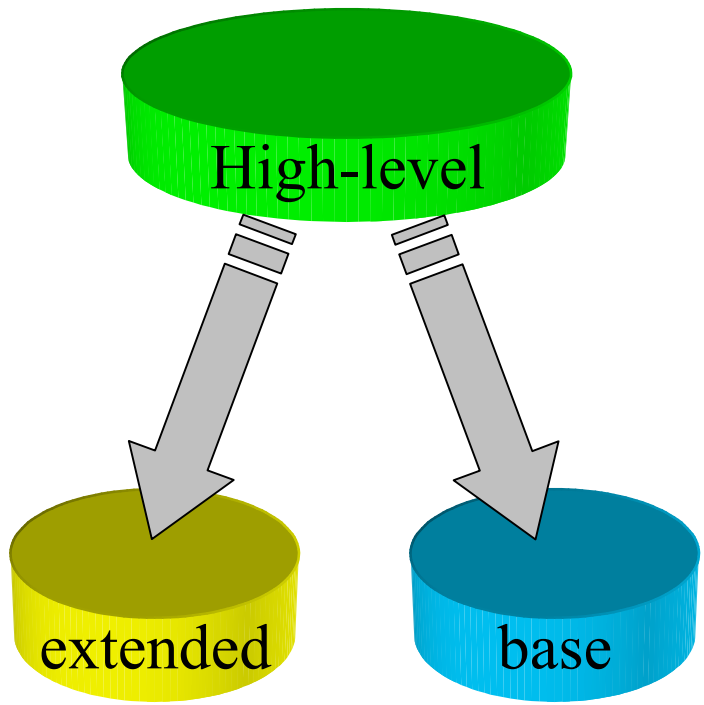
New Modules

- Debugging Facilities
  - Adds commands to examine Soar's internal memory pools
- Timer Verification
  - Ensures that timers are reporting non-zero value (i.e. timer resolution is adequately fine)
- Time Tracking by Decision Cycle
  - Creates an array storing the time required to complete a set of decision cycles
- Capture/Replay Input
  - Allows reproducible testing in non-deterministic (or otherwise complex) environments

Test Results

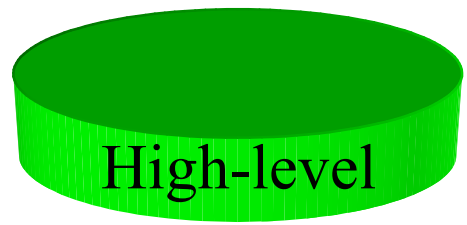
Multi-Tiered API

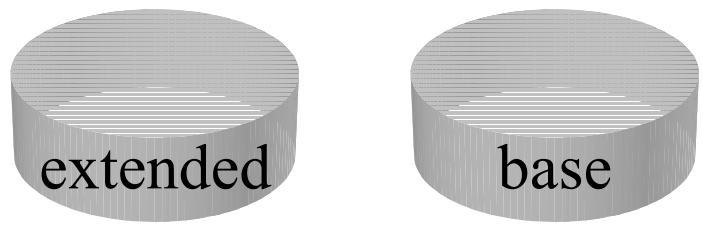
- Provide a consistent interface to Soar via Tcl and C function calls
  - simplicity of learning
  - increased accessability
  - ease of documentation



High Level API

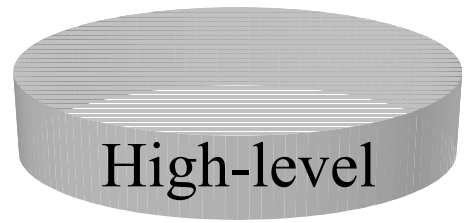
- Reproduce functionality of TSI interface without Tcl dependencies
  - Allows easy integration into other U.I.s
  - Uses a common function prototype
  - Takes care of argument context dependent parsing, error detection, etc.

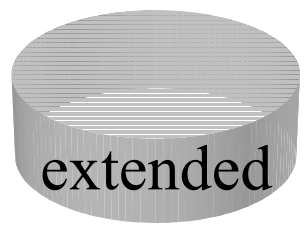


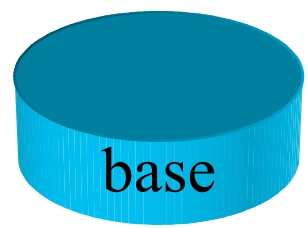


Base (Core) API

- Encapsulate minimal functionality to embed Soar within another application
  - Provides common ancestors for functions in the high-level API
  - Uses a typed arguments to eliminate unnecessary string parsing and aid integration into other apps.

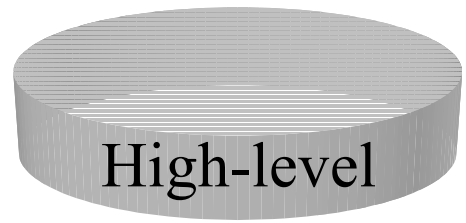


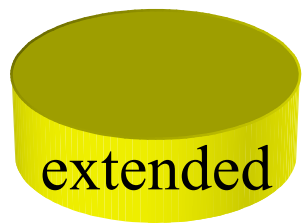


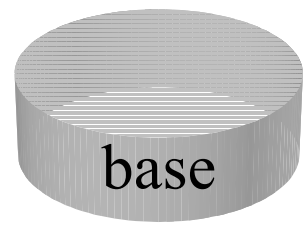


Extended API

- Extend the minimal functionality of the Base API to provide other commonly used features.
  - Provides common ancestors for functions in the high-level API
  - Uses typed arguments
  - Assumes interface printing capabilities







Next Steps

- Complete Remaining 10% of API functionality
- Complete Documentation
- Feedback
  - How should multi-agent support be handled?
  - What additional functionality would be useful
  - What level of abstraction is best for the I/O system

Nuggets and Coal

- Nuggets
  - Increases performance significantly in domains which make heavy use of subgoaling
  - Provides a programming interface which is consistent with the well documented, and familiar TSI interface
- Coal
  - Not MT-Safe
  - Some functions have been moved and/or renamed