

What's New in Soar 8.6.2

Douglas Pearson
+ Bob, Jon, Karen, Taylor, John

May 25, 2006

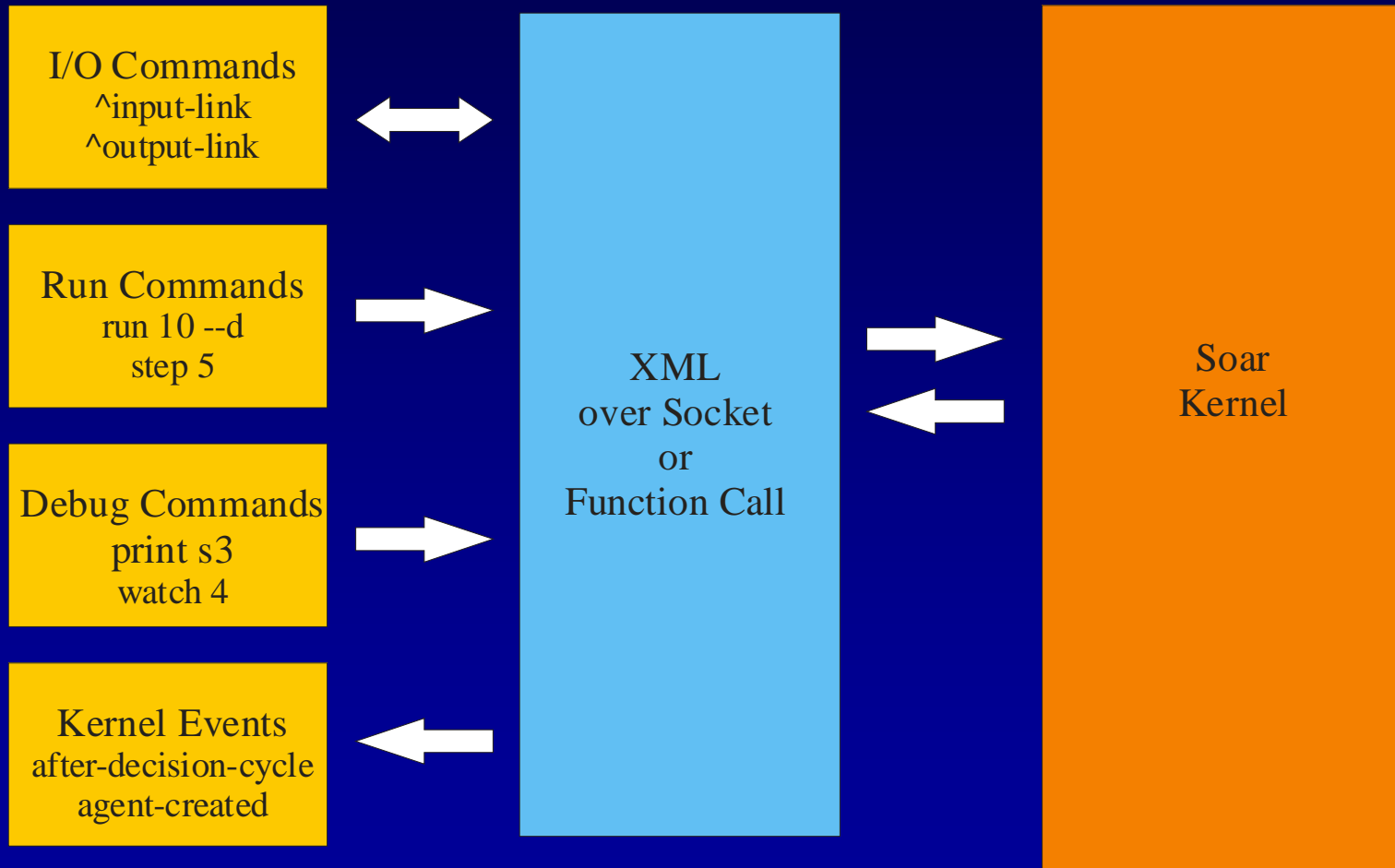
douglas.pearson@threepenny.net
soar-sml-list@lists.sourceforge.net



Quick Review of 8.6.0/8.6.1

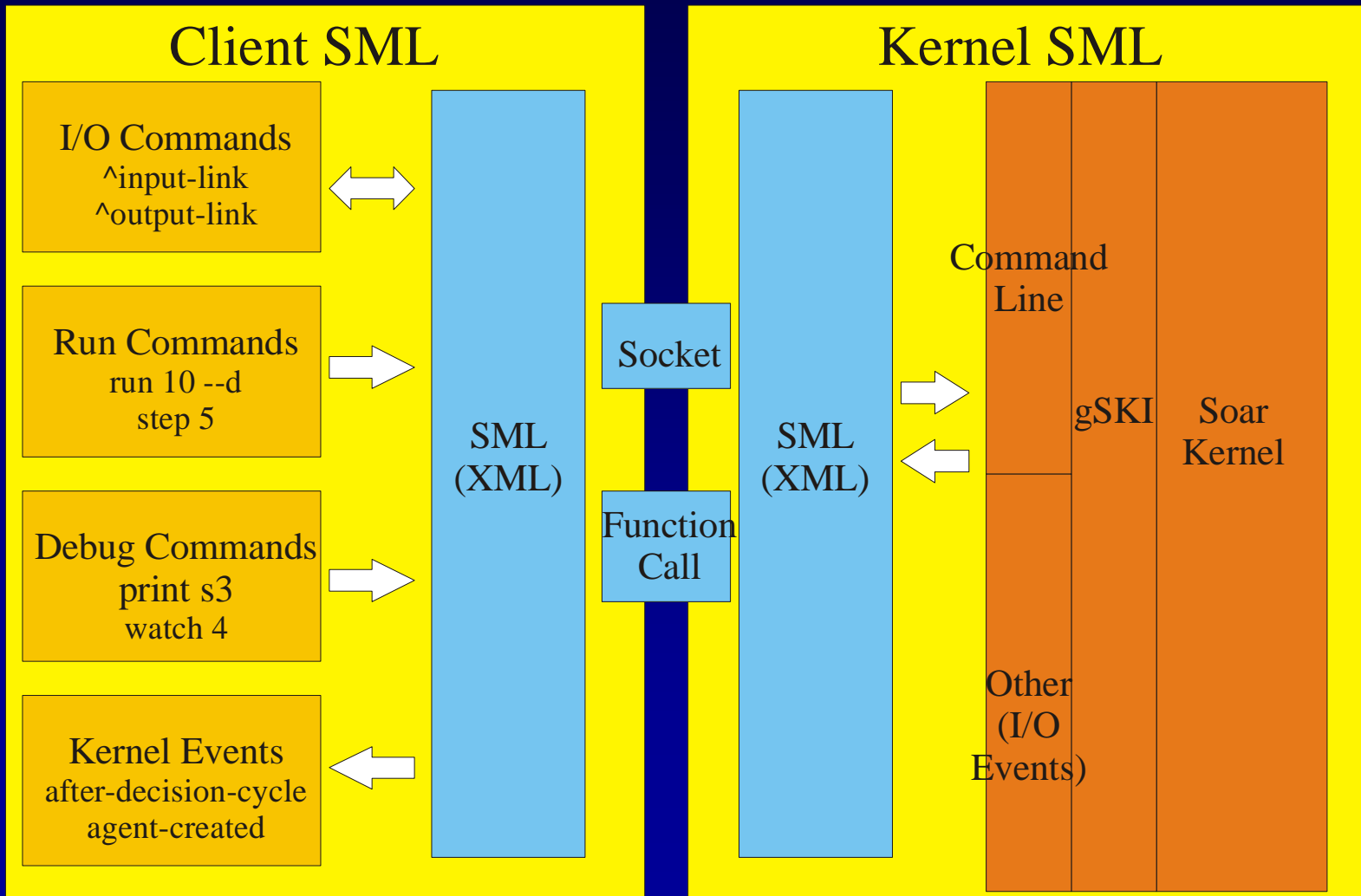
- Introduction of XML based interface to Soar
 - SML (Soar Markup Language)
- Opened the door to other languages
 - Java, C++, Tcl
 - New debugger in Java
- More flexible debugging
 - Embedding kernel into environment and debug remotely
 - Faster performance
 - Dynamic connection and disconnection of debugger
- No kernel level changes
 - Just new way to connect environments & tools to Soar
 - Cleaned up and rewritten command line interface

Connecting to Soar SML Style



Uniform for entire interface

Connecting to Soar SML Style



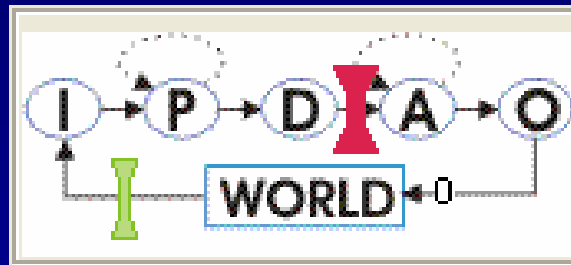
8.6.1 Status

- Downloads from Source Forge
 - ~3,000
- Many questions from unrecognized people and groups
- Anecdotally reasonably simple to interface too
 - Cross language capabilities working fine
 - Majority of tools and environments (that we know about) in Java
 - Some build and install issues on Unix/Linux
- But some remaining problems
 - Run til output broken
 - Print wmes on input link that aren't in working memory yet

Setting a “stop-point”



- In 8.5 and 8.6.1 “run n decisions” always stops after output
 - Hard to use “matches”
 - Want an easy way to control when to stop



- Green marker shows current phase
 - But only updates at end of a run
- Red marker shows where to stop (for “run n”)
 - Click in GUI in debugger to change
 - Or “set-stop-phase --before –apply” etc.

“Run 0” and “Run n”

- “run 0”
 - Runs all agents up to the current stop-point
 - Quick way to synchronize agents to a phase
- Decisions vs Decision Cycles
 - Decision cycle ends after output
 - Decision occurs when select operator / impasse
 - “run n” ?
 - Run for n **decisions** and then to the stop-point
 - 8.6.2. not 100% compliant to this yet but close

Flexibly Interleaving Agents

- 8.6.1
 - Agents always interleaved by phase
 - Tank Soar requires by output generation
 - Each tank “takes a turn” in the game
- 8.6.2
 - Can interleave by elaboration, phase, decision, output
 - E.g. “run -o 3 -interleave d”
 - Combined with other changes => total rewrite of scheduler

Different I/O Models

- Soar agent *always*
 - Receives input in the input phase
 - Generates output in the output phase
- Environments can vary
 - Asynchronous: Environment updates when each agent acts
 - Real world
 - Not all actors are Soar agents (or necessarily intelligent)
 - Synchronous: Environment updates after all agents act
 - Easier to debug
 - May be better for some research
 - Probably less interested in the environment / task and more in the agent

http://winter.eecs.umich.edu/soarwiki/Main_Page

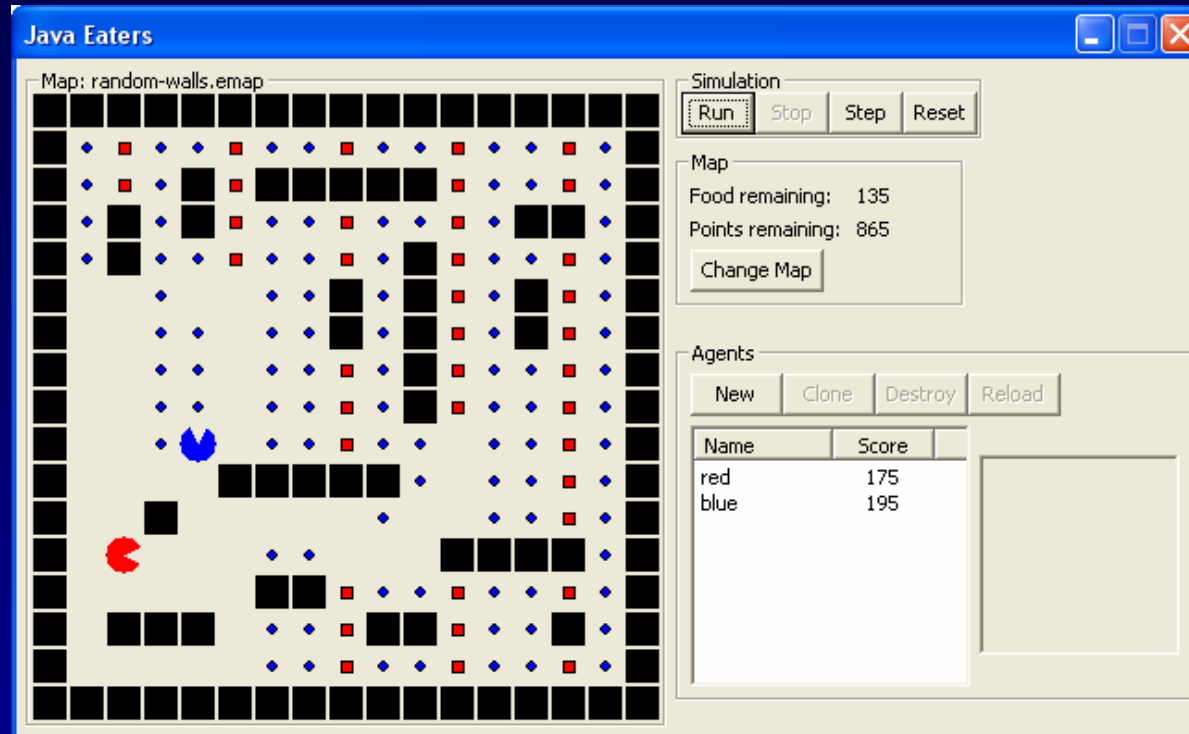
Output: Updating the World

- Option 1: Agent::Register(smIEVENT_AFTER_OUTPUT_PHASE)
 - Check for changes to the output link and change world
 - Good for asynchronous environment
 - Difficult for synchronous because agentA acts before agentB unless buffer actions in environment
 - Low performance – one event per agent per decision cycle. May not have acted.
- Option 2: Agent::AddOutputHandler(attribute, handler)
 - Called immediately after attribute is added to output link
 - Similar strengths and weaknesses to option 1
 - Better performance than option 1 but must know attribute names
- Option 3: Kernel::Register(smIEVENT_AFTER_ALL_OUTPUT_PHASES)
 - Called after all agents have completed output phase
 - Easier to produce synchronous interaction
 - Better performance – one event per execution cycle (for any number of agents)
- Option 4: Kernel::Register(smIEVENT_AFTER_ALL_GENERATED_OUTPUT)
 - Called after all agents have generated output
 - Turn based environments (e.g. Tank Soar)
 - Enforces completely synchronous behavior
 - An unusual choice

Push vs Pull for Input

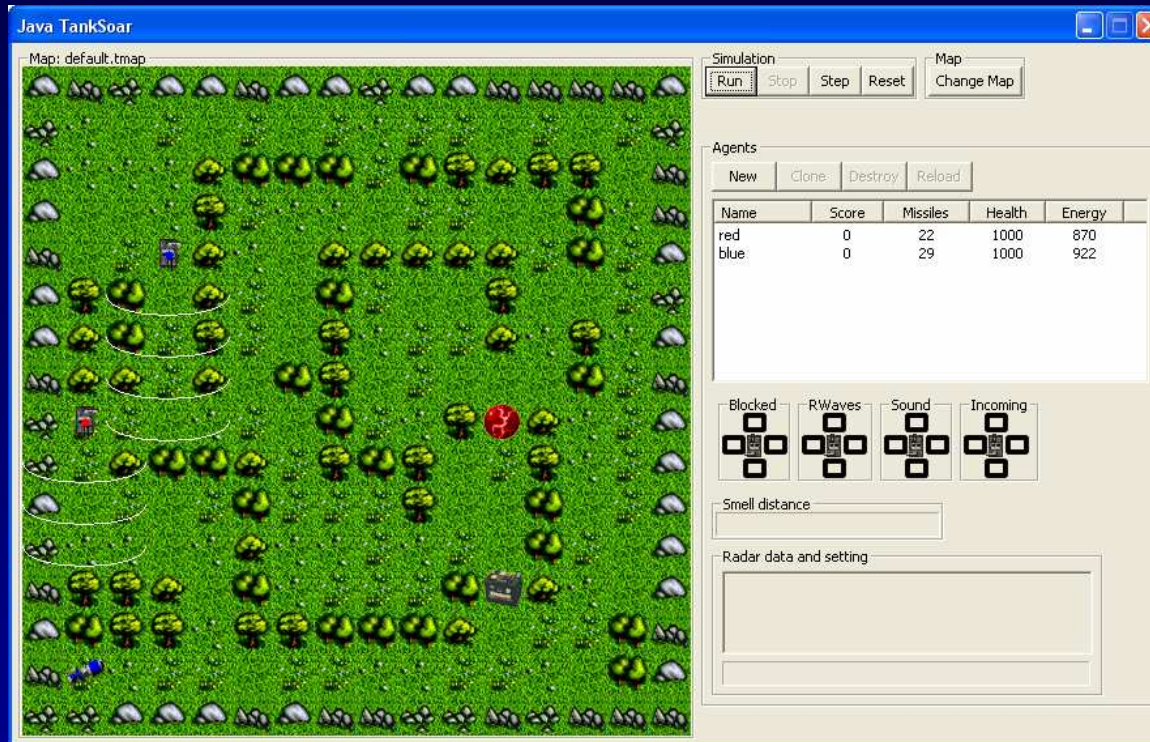
- Option 1: Push
 - When environment changes send new state to kernel
 - Ignores agent's phases
 - Requires SML/gSKI to buffer until each agent's next input phase
 - If environment changes faster than agent checks input, this option is lower performance
 - Implementation: Driven by output events (or external actors)
 - (Output) -> Change World -> Send Input
- Option 2: Pull
 - Agent calls over to environment each input phase to get current state
 - No buffering required
 - If environment changes slower than agent checks input, this option is lower performance
 - Implementation: Register(smIEVENT_BEFORE_INPUT_PHASE)
 - Send current state in event handler.
 - (Output) -> Change World. Don't send new input.
- Soar 8.6.2 supports all of these different input/output options
 - Please consider your task in selecting implementation
 - Pretty easy to switch back and forth
 - Unnecessary events can be expensive if they cross the client-kernel divide

Java Eaters



- All new implementation in 8.6.2
- Higher performance.
- Output – `smLEVENT_AFTER_ALL_OUTPUT_PHASES`
- Input – push model (output -> update -> send input)
- Run – `RunAllAgentsForever()`
- Quite common design for environments

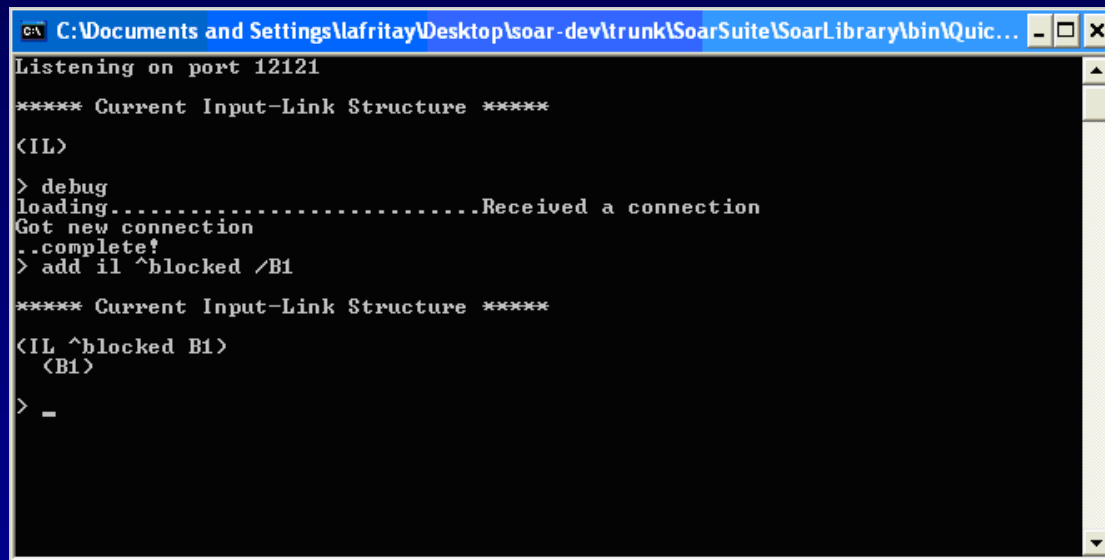
Java Tank Soar



- All new implementation in 8.6.2. Shares some code with Java Eaters.
- Higher performance.
- Can run without a UI
- Output – `smIEVENT_AFTER_ALL_GENERATED_OUTPUT`
- Input – push model (output -> update -> send input)
- Run – `RunAllAgentsForever(smI_INTERLEAVE_UNTIL_OUTPUT)`

New Tool: Quick Link

- Manually control the input link
- “Fake” an environment
 - Test specific situations



```
C:\Documents and Settings\lafritay\Desktop\soar-dev\trunk\SoarSuite\SoarLibrary\bin\Quic...
Listening on port 12121

***** Current Input-Link Structure *****

<IL>

> debug
loading.....Received a connection
Got new connection
..complete!
> add il ^blocked /B1

***** Current Input-Link Structure *****

<IL ^blocked B1>
  <B1>

> -
```

- Examine current input and output links
- Add input wmes
- Modify or delete existing input wmes
- Run Soar
- Store and load scripts of commands
- Not in 8.6.2 release but will follow shortly

New Tool: Soar Text IO

- Easy way to place text (individual words) onto the input link in a standard way
 - Providing problem sets to an agent
 - Providing guidance or instruction

```
^input-link
  ^text
    ^text-input-number <num>
    ^length <num-words>
    ^next
      ^value <word-one>
      ^next
        ^value <word-two>
        ^next
          ^value <word-three>
          ^next
            ^value nil
```

•Not in 8.6.2 release but will follow shortly

Better Logging

- How to log what Soar is doing?
 - Record trace as text file and parse it
 - Augment productions to output log information (Vista)
 - Modify kernel to generate logging data
- Alternative is a logging application (client)
 - Connects to Soar while it's running (no overhead when not logging)
 - Register for events you are interested in
 - Output log information in any format desired
 - Examples in C++ and Java included in 8.6.2

- E.g. To create a behavior trace in your format

```
MyXMLEventHandler(ClientXML* pTraceXML) {
```

```
    if (pTraceXML->IsTagState()) {
```

```
        std::string count = pTraceXML->GetDecisionCycleCount() ;
```

```
        std::string stateID = pTraceXML->GetStateID() ;
```

```
        std::string impasseObject = pTraceXML->GetImpasseObject() ;
```

```
        std::string impasseType = pTraceXML->GetImpasseType() ;
```

```
        // Write this out any way you want
```

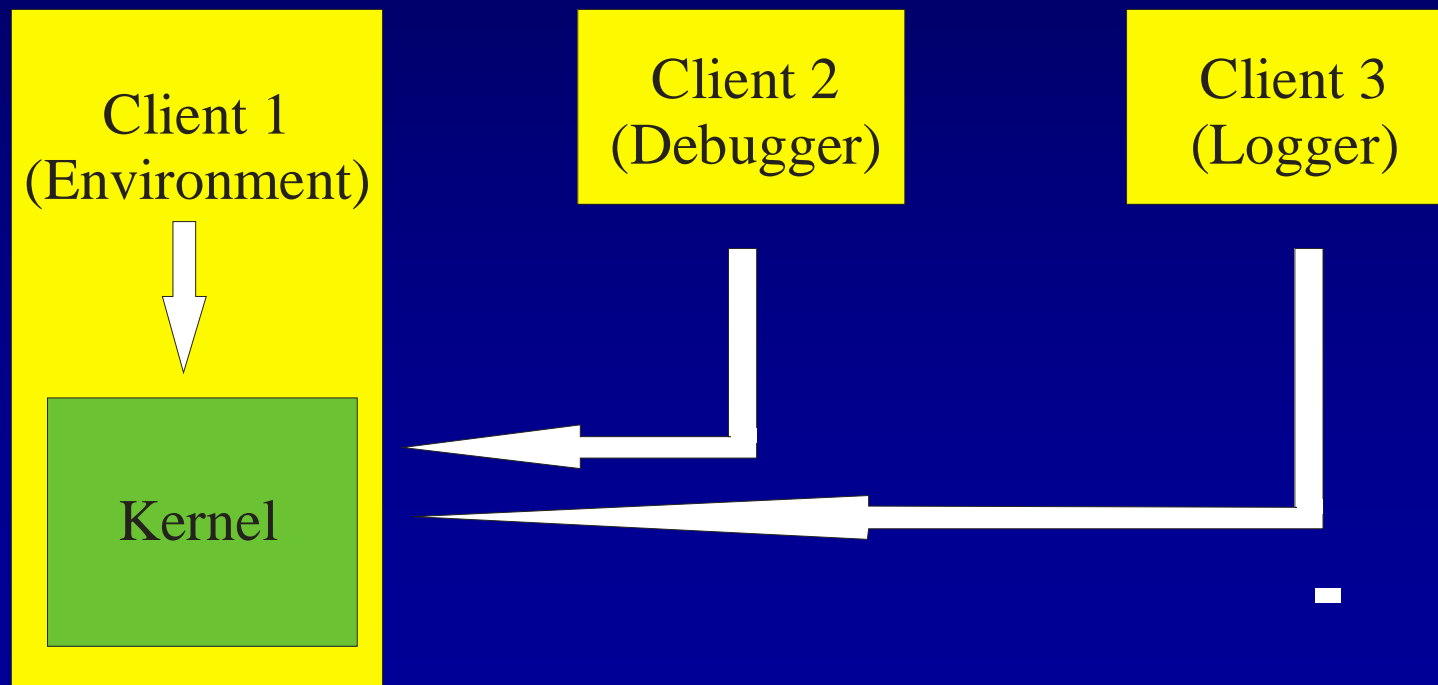
```
        fprintf(gOutputFile, "%s %s (%s %s)\n", count.c_str(), stateID.c_str(), impasseObject.c_str(),  
impasseType.c_str()) ;
```

```
    }
```

- The entire logging application can be ~40 lines of code

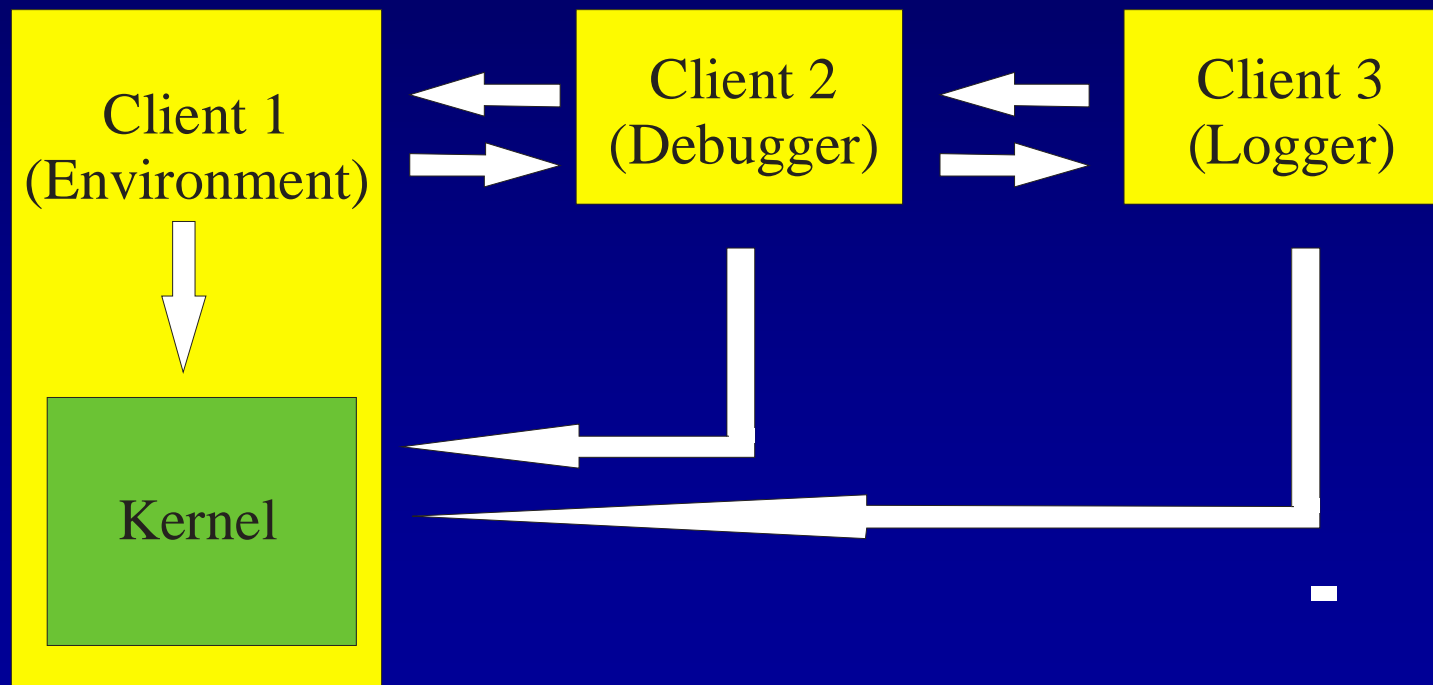
Client-to-Client Communication

- 8.6.1 – Sets of clients talking to kernel



Client-to-Client Communication

- 8.6.2 – Clients can talk to each other
 - E.g. Environment signally logger to start logging
 - E.g. Environment waiting for debugger to launch before proceeding



Client-to-Client Communication

- 8.6.2 – Clients can talk to each other
 - Actually messages routed through kernel
 - Implicitly synchronizes with kernel actions
 - Listen for messages
 - `Kernel::RegisterForClientMessageEvent("debugger-status", handler)`
 - Send messages:
 - `Kernel::SendClientMessage("debugger-status", "ready") ;`
 - Messages are strings
 - Can be simple text
 - Can be XML
 - Content entirely determined by client

Better Tcl Support

- Problem

- 8.5 Tcl was part of the kernel
- 8.6 Tcl removed from kernel, available for environments

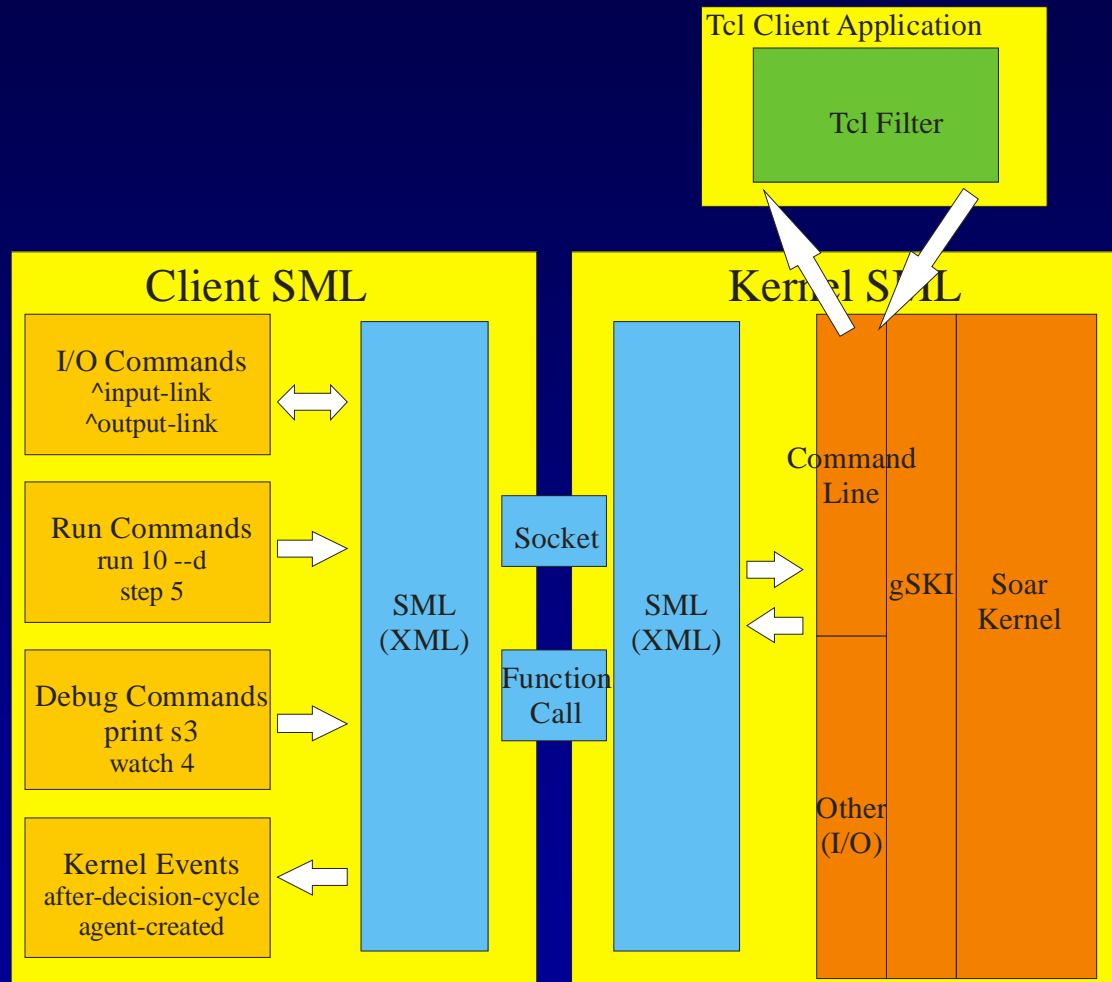
- But in 8.5 could use Tcl to generate/expand productions:

```
proc NGS_prefer-operator-x-over-y { op1 op2 {prefix
"operator-preference*"}} {
  sp "$prefix-$op1-over-$op2
    (state <s> ^operator <o1> + <o2> +)
    (<o1> ^name $op1)
    (<o2> ^name $op2)
  -->
  (<s> ^operator <o1> > <o2>)" }

NGS_prefer-operator-x-over-y generate-goal-subgoal-summary \
generate-goal-alternatives-summary
```

- Soar Tech uses this extensively
- How to support this in 8.6.2 when debugger in Java, kernel in C++ and Tcl not required?

Better Tcl Support



Better Tcl Support

- 1) Command typed in debugger
 - “printme s1”
- 2) Command is sent to KernelSML for execution
- 3) KernelSML recognizes a filter is present
 - Sends command to the filter
- 4) Tcl filter receives command
 - Selects an interpreter to use to execute the command
 - Executes the command as a Tcl command and captures the result
- 5) Passes the result back to KernelSML
 - Marks command as executed (i.e. eaten by filter)
- 6) KernelSML passes result back to debugger as result of command
 - Debugger prints “s1 ^io l1 ^superstate nil ...” etc.

Better Tcl Support

- Filter main loop:

```
proc MyFilter {id userData agent filterName commandXML} {
    set interpreter [getInterpreter [$agent GetAgentName] $agent]

    set xml [ElementXML_ParseXMLFromString $commandXML]
    set commandLine [$xml GetAttribute $sml_Names_kFilterCommand]

    # Evaluates the command within the child interpreter for this agent
    set error [catch {$interpreter eval $commandLine} result]

    # Return the result of the command as the output string for the command
    # which in turn will appear in the debugger
    $xml AddAttribute $sml_Names_kFilterOutput "$result"
}
```

- Soar commands routed back from Tcl to Soar

```
proc add-wme {args} {return [soar_agent ExecuteCommandLine "add-wme $args"]}
proc excise {args} {return [soar_agent ExecuteCommandLine "excise $args"]}
```

Properties of Filter Solution

- Full Tcl interpreter(s) in own process
 - No limitations on Tcl code
- Works for all clients not just Java debugger
 - E.g. Can load “Tcl productions” from standalone environment
- No modifications to clients
 - Debugger is completely unaware filtering is happening
- No impact on people not using Tcl
 - Debugger doesn’t include Tcl interpreter support
- Performance should be good
 - Only affects time to parse commands
 - Once Soar is running filter is never called
- Filter is modular and separate from rest of code
- Filter can be in any supported language
- Other filters are possible
 - E.g. Just listen for “source x.soar” and run a precompiler
 - Cleaner than modifying the command in the kernel code
- But Tcl filter’s not finished yet in 8.6.2

Commits are easier

- 8.6.1

```
Identifier* pSentence = pAgent->CreateIdWME(pAgent->GetInputLink(), "sentence") ;  
pAgent->CreateStringWME(pSentence, "newest", "yes") ;  
pAgent->CreateIntWME(pSentence, "num-words", 3) ;  
pAgent->Commit() ;
```

- Commit()

- Collects all input changes and sends them to kernel in one go
- Higher performance but error prone

- 8.6.2

```
Identifier* pSentence = pAgent->CreateIdWME(pAgent->GetInputLink(), "sentence") ;  
pAgent->CreateStringWME(pSentence, "newest", "yes") ;  
pAgent->CreateIntWME(pSentence, "num-words", 3) ;  
// Not needed: pAgent->Commit() ;
```

- AutoCommit on => slightly lower performance
- Kernel::SetAutoCommit(false) to revert to 8.6.1 behavior
- But kernel often inside environment now so less impact

- Init-soar

- Works (8.6.1 too) and resends input link to agents automatically

Wide Range of Events

- Run Events
 - [Before | After] Each Phase // As agent runs a phase
 - [Before | After] Decision Cycle // Completes decision cycle
 - [Before | After] Run Starts | Stops // Agent run / stop
 - [Before | After] Running // Each step of a run (decision, phase etc.)
 - Interrupt Check // Low bandwidth chance to stop
 - After Interrupt // Run was interrupted
- Update Events
 - After all output phases // Synchronous world
 - After all generated output // Turn based world
- Production Events
 - After Production Added // During loading
 - Before Production Removed // Excise
 - After Production Fired // Production firings
- System Events
 - After agent created // New agent created
 - Before agent destroyed // Agent about to be destroyed
 - [Before | After] agent reinitialized // init-soar
 - System Property changed // set [x] [y]
 - System Start | Stop // Agents are running / all stopped
- Trace Events
 - Print // Text output
 - Echo // Echo'd commands
 - Trace Output // XML output
 - Input Received // Listen for what environment is doing
- RHS and User Events
 - RHS Function handler // Implement RHS function in Java, C++, Tcl or C#
 - Command Line filter // Filter commands before kernel processes them
 - Client messages // From one client to another (not to/from kernel)
 - Edit production // Ask editor to locate production

Other 8.6.2 additions

- Added Java TOH as an SML tutorial explaining all steps
 - Line by line how to build a simple SML environment
 - Start here if building a new environment
- Added new phase specific events: before_input_phase etc.
 - More efficient than generic “phase” event
- Added log output in the debugger on a window by window basis
 - Necessary to separate out streams of output
- Added C# as a supported language
 - C++, Tcl, Java, C#
 - C++: Identifier* pInputLink = pAgent->GetInputLink() ;
 - Tcl: set inputLink [\$agent GetInputLink]
 - Java: Identifier inputLink = agent.GetInputLink() ;
 - C#: Identifier inputLink = agent.GetInputLink() ;
- Added Visual Studio 2005 support (as well as VS 2003)
- Added Java 5.0 support (as well as Java 1.4.2)
- Added new random number generator and srand() command
- Added synchronization option in debugger (so Soar doesn't run ahead)

Other 8.6.2 additions

- Improved Linux performance vastly (20x in some cases)
- Improved Windows performance further (> 30% faster in TOH)
- Fixed major top state memory leak (NO_TOP_REFS)
 - Memory usage no longer climbs when doing lots of top state work
- More efficient garbage collection when states (contexts) go away
- Lots of bugs fixed
 - <http://winter.eecs.umich.edu/soarwiki/>
- Loose coupling working
 - 8.6.2 debugger will load and run 8.6.1 kernel w/o modification
 - “New bits” (e.g. phase diagram) just do nothing