jACT-R : Java ACT-R

Why I’ve given up my free time
to reinvenvent the wheel

Anthony Harrison
anh23@pitt.edu
jACT-R : Why not use the Lisp?

- Partially driven by Christian’s quest for the “killer app”*
  - Niche languages are the antitheses of the “killer app”
- Many of the development tools that I want to see are extremely difficult to implement in Lisp (probably won’t be true when 6.0 comes out)
- Extend the potential range of phenomena and applications that ACT-R can be applied to

*Just wait until the spatial extension is complete. Then we’ll have ACT-R playing Quake.
What does Java bring to the table?

- “Write once, run anywhere” is no longer a cliché.
  (with the minor exception of OS 9.0)
- All Java applications are immediately viewable to jACT-R.
- “Model farming”, collaborative models, distributed modeling
- Wealth of standardized APIs
  - No more models of interaction with simulated devices - they can actually use the real devices.
  - Creation of new modeling tools relatively easy
**jACT-R : Modeler’s Perspective**

- Passed all compatibility tests* to date
- Interfaces Java applications transparently (and in real-time)
- Simple collaborative modeling
- Just as scriptable (although with Javascript not Lisp)
- Automatic bug reports bounced directly to me

* Tests based on Excel simulations of algorithms
**jACT-R: Modeler’s Perspective**

(cont)

- GUI that requires no funky commands or parameters (no *sgp* or *era*)
- Organized tree display of model makes large model management simpler
- “Intelligent guessing” of chunks, types, and slots reduces typing
- GUI permits saving of models at intermediate stages
jACT-R : Theory-Tweaker’s Perspective

- Completely modular in design (like 6.0 will be?)
  - Modules are separable
  - Separation of symbolic and subsymbolic logic and code
  - Separation of all equations
- Extensions (ala /PM) can be fully supported via extension APIs.
- All tweaks can be loaded dynamically without touching any of jACT-R’s code.
**jACT-R : Developer’s Perspective**

- Where the Lisp uses function hooks for customization, jACT-R uses events
  - Both synchronous and asynchronous
  - Examples: parameter changes, chunk creation, production firing, buffer content changes, logging events, etc.
- Want a CPM-GOMS style trace? Attach a ModelRunListener and process the events.
- Want a trace tool that permits rolling back the model to an arbitrary point in time?
  - Log all the events (each event has old & new values)
  - Apply or revert based on the time stamps
**jACT-R : Modeler’s Perspective**  
*(The Dark Side)*

- Production compilation does not exist (yet)
- PM extension is not complete
- No Lisp Import/Export
  - Models, while isomorphic, are syntactically very different
- Nowhere near as fast as it can be (and will be)
What’s Holding 1.0 Up?

- *jACT-R* core is unfunded. After work, before sleep is the only time it gets worked on (although the spatial extension is funded)
- There is only one developer
  - Good with designing
  - Bad with optimizing
- Development has been spread out
  - Core
  - Extensions
  - and Tools
More Info..

- **jACT-R Home**
  - [http://sourceforge.net/projects/jactr/](http://sourceforge.net/projects/jactr/)
  - All the latest releases, source, documentation, discussion groups, bug lists, mailing lists, CVS

- **Web-based download/install/run**
  - [http://simon.lrdc.pitt.edu/~harrison/jactr/jactr.jnlp](http://simon.lrdc.pitt.edu/~harrison/jactr/jactr.jnlp)
  - MacOSX, Windows, Solaris only right now. *nix coming soon.

- **eMail**
  - [jactr-announce@lists.sourceforge.net](mailto:jactr-announce@lists.sourceforge.net)
  - [jactr-developer@lists.sourceforge.net](mailto:jactr-developer@lists.sourceforge.net)
  - anh23@pitt.edu