Interfacing ACT-R with External Simulations

Eric Biefeld, Brad Best, Christian Lebiere

Human-Computer Interaction Institute

Carnegie Mellon University

We Have Integrated ACT-R With Several External Simulations and Learned Some Key Lessons

- The applications
  - UT MOUT
    - A worst-case integration
  - AMBR
    - Better thought-out but still significant effort and speed problems
  - AMBR HLA
    - Interesting process but suboptimal result
  - IMPRINT/ACT-R (and CART/FRED)
    - Development of general-purpose layer
- The lessons
  - Most development time goes into infrastructure (Zachary)
  - General-purpose APIs can save orders of magnitude of integration time

We Can't Reimplement Large Simulations So We Must Build Links From the App to ACT-R

- Reimplementation cost in time/$ is prohibitive
- Can't modify: the simulation is always right
- Data must travel from and to both simulation and model
  - Communicating to the model the state of the simulation
  - Local cache of the state of the simulation may be needed
  - Communicating to the simulation the model actions
  - Each new communication link requires an API
- The developer also needs links to both the simulation and the model
  - Existing ACT-R Lisp API provides link to model
  - Debugging may be through either the model or the simulation
- Time synchronization is tricky
  - Real time vs. virtual time
  - Event-based vs. tick-based

Defining and Implementing Links to an External App Is Where the Time Goes

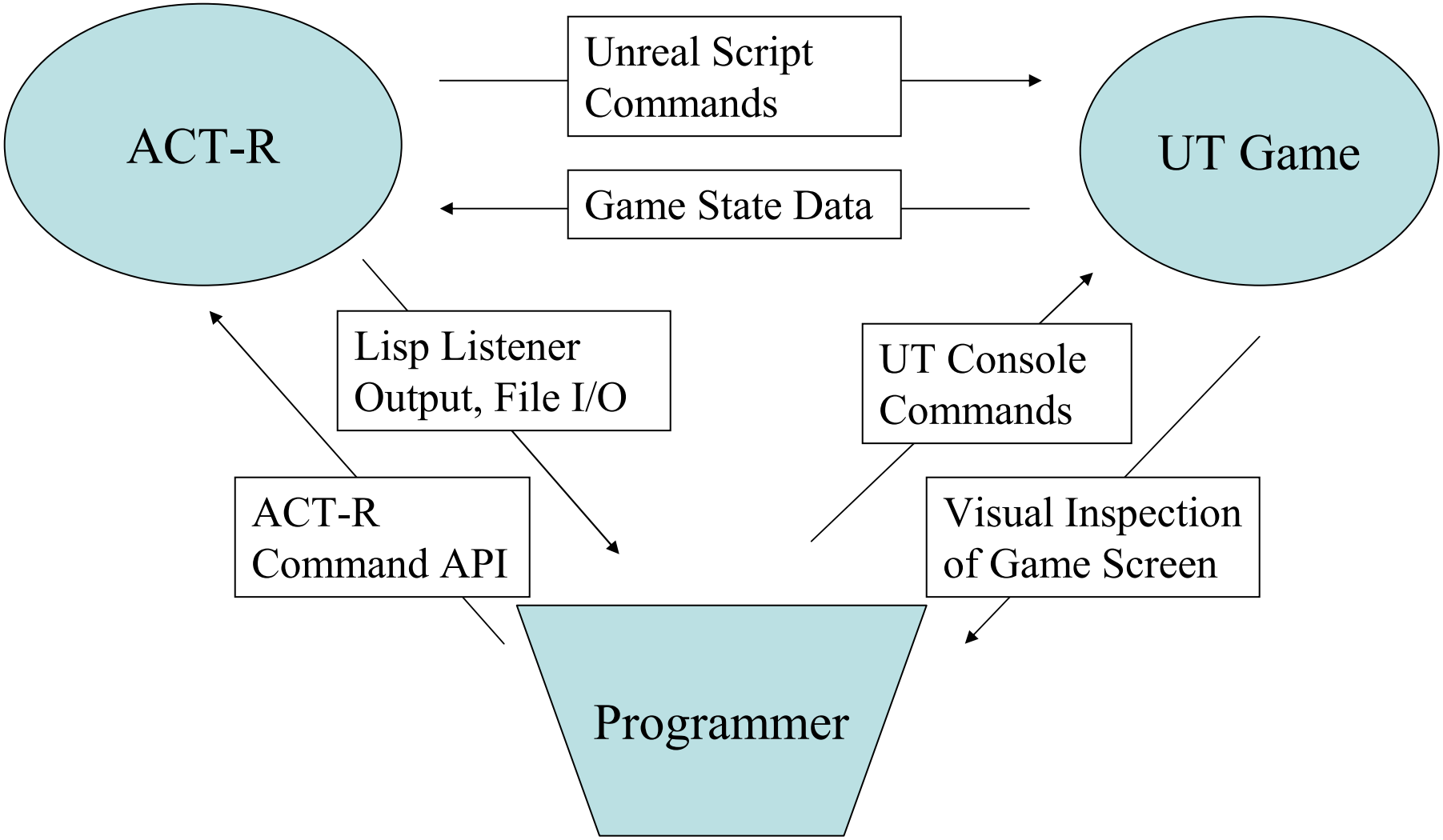
- Protocol
  - Data sharing: description of data structures (how?)
  - Time synchronization: advance time (when & who?)
- General-purpose solutions (save future costs)
  - HLA formalized the documentation and provides a run time infrastructure
  - General API for ACT-R?
- Special-purpose interface
  - Various implementation instances in following examples
  - Optimal but cost-intensive solution

Example 1: ACT-R Agents for Urban Combat Integrate with Unreal Tournament

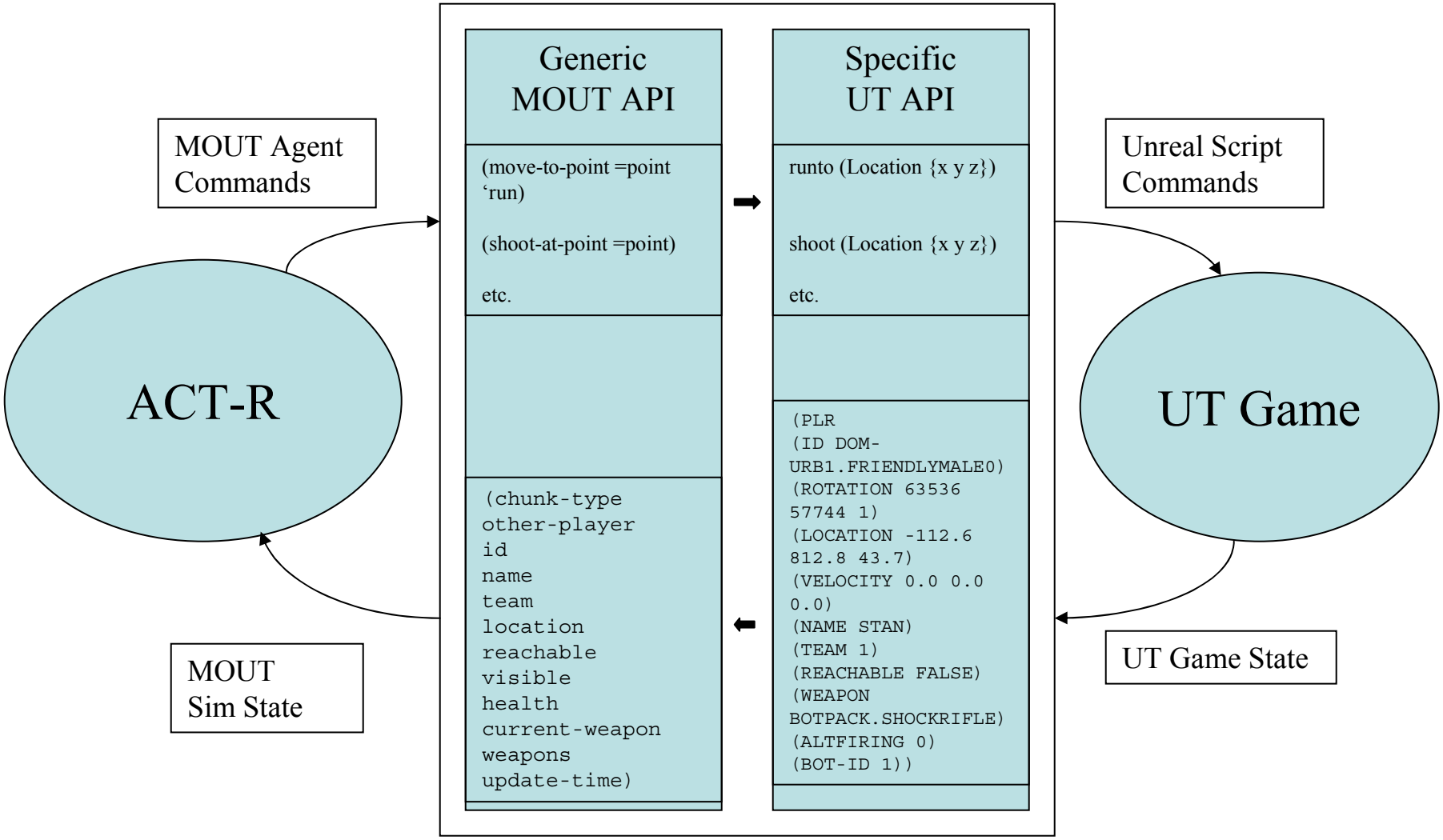
- Realtime App with realtime constraints
  - Processing and network bottlenecks
- Integration is more work than the modeling
  - Game does not provide primitives needed for cognitive modeling (no walls!)
- 2-layer API
  - Designed to accommodate change in simulation platform
  - May not get desired changes in next version

The Unreal Tournament game engine / simulation platform for MOUT

MOUT/UT Information Flow



ACT-R/UT Interface



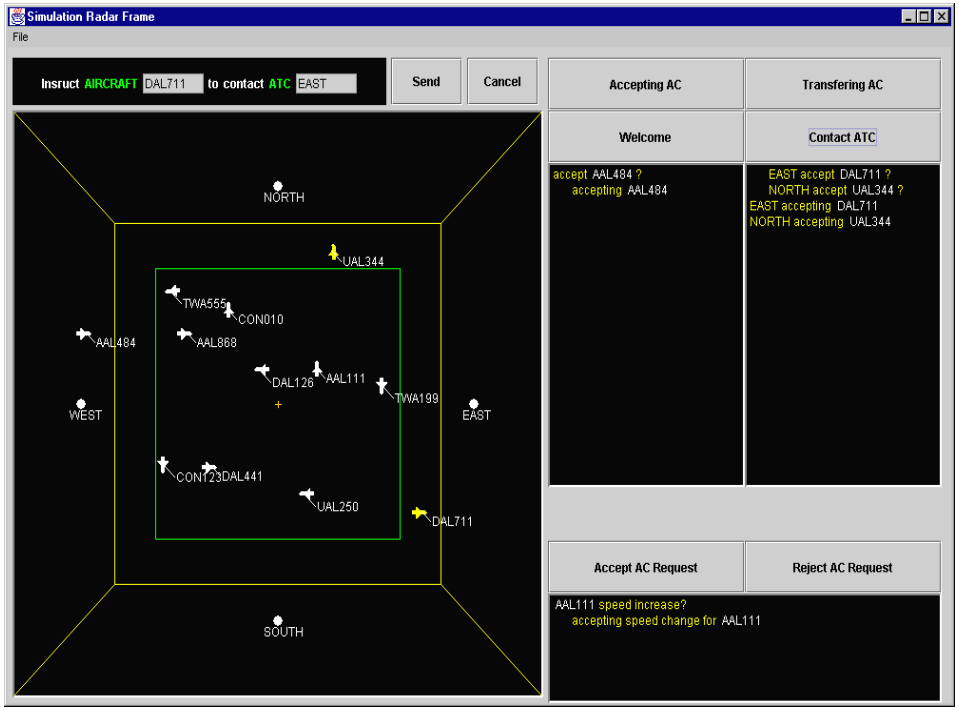
ACT-R UT/MOUT Time Sync

- Realtime App… sort of.
  - UT game time is the master clock
    - May slow down or speed up depending on CPU demands
  - UT sends game state updates at regular intervals
    - Upon update, ACT-R model runs cycles until ACT-R time is later than the UT time stamp in the update, then the update is applied
      - This prevents ACT-R perceiving events in the past: an update at time t will appear at ACT-R time t or slightly later
  - ACT-R must run ***much faster*** than realtime to keep up
    - The external app and other models may need to run on the same machine
    - Individual models must release control to allow other models to catch up – no thread hogging
  - Limits on how many agent models can be run in realtime

Example 2: AMBR

The Theory

- ATC simulation for cross-architecture modeling
- Communication protocol designed jointly for model and simulation
- Sockets: simple and easy



The Reality

- Too slow (250msec roundtrip for 50msec control loop: 5x RT)
- 2nd solution: run in RT as a human subject (funny side effects!)
- 3rd solution: run in same Lisp process as the simulation!

A sample protocol: Original AMBR API

Data communication between simulation (Core-OMAR) and agent (ATC Workplace)

Core-OMAR output to the ATC Workplace

(ATC::INIT-RADAR (string atc-name)) Initializes the radar screen for ATC model workplace and gives model name

(ATC::INIT-ATC (string atc-name) (double epos) (double npos)) Provides ATC name and screen position

(ATC::INIT-AC (string aircraft-name)) Identifies a new aircraft icon that is about to appear

(ATC::UPDATE-AC (string aircraft-name) (double epos) (double npos) (double evel) (double nvel) (double altitude) (string color)) Identifies a new location and color for an aircraft icon.

(ATC::REMOVE-AC (string aircraft-name)) Identifies an aircraft icon no longer seen on the radar screen.

(ATC::COMMAND-PROMPT (string prompt)) Indicates the appearance of a new prompt in the command line

(ATC::COMMUNICATION-MESSAGE (string speaker-name) (string message-content) (string panel-name)) Presents a new message at the bottom of the identified screen panel

Input to Core-OMAR by a Model at the ATC Workplace

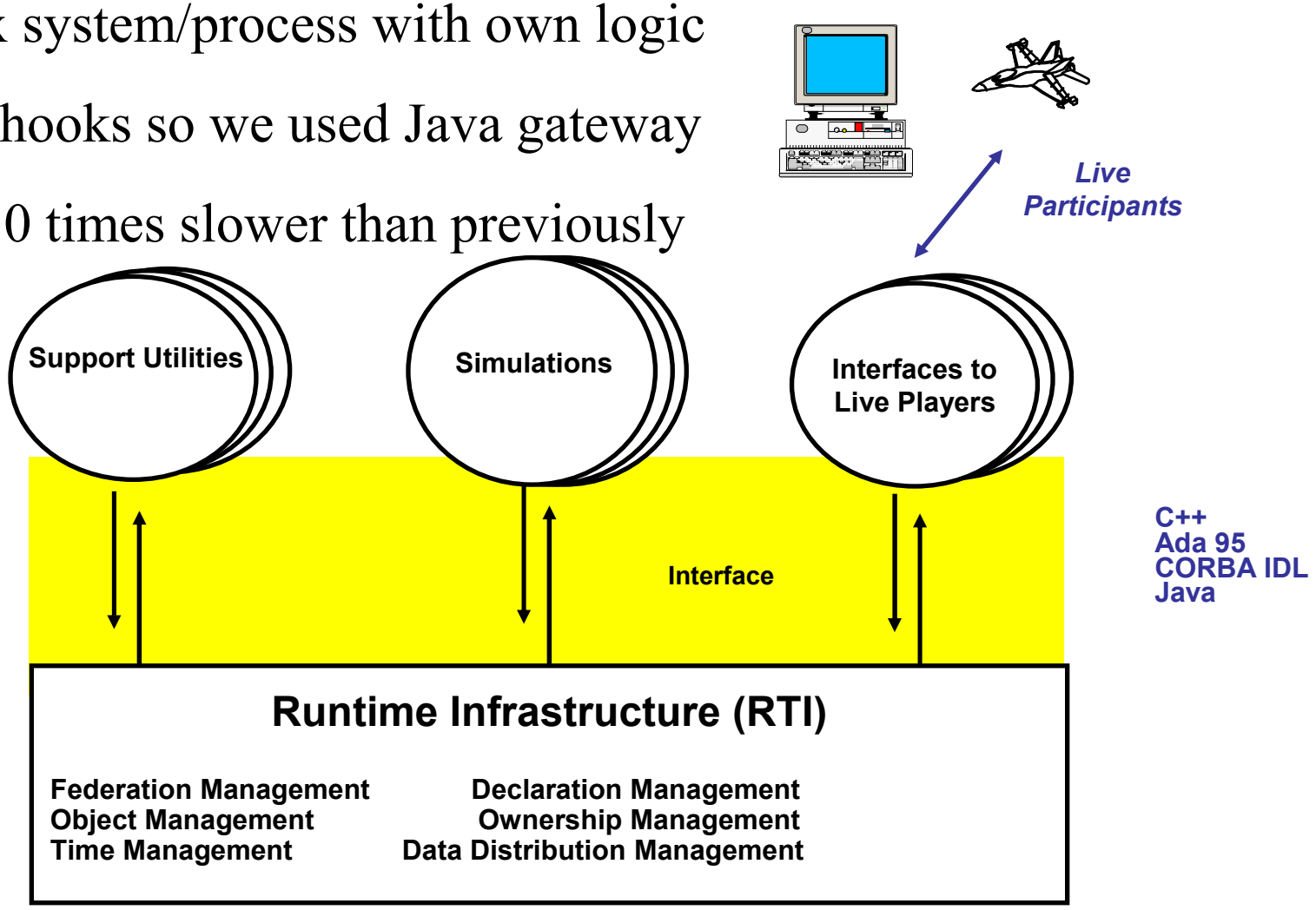
(ATC::GUI-BUTTON-PUSH (string label)) Indicates that a screen button that has been pushed

(ATC::GUI-OBJECT-SELECT (string icon-label)) Indicates that screen-icon for aircraft or ATC has been selecte

Time synchronization of simulator operation for multi-node distributed simulation

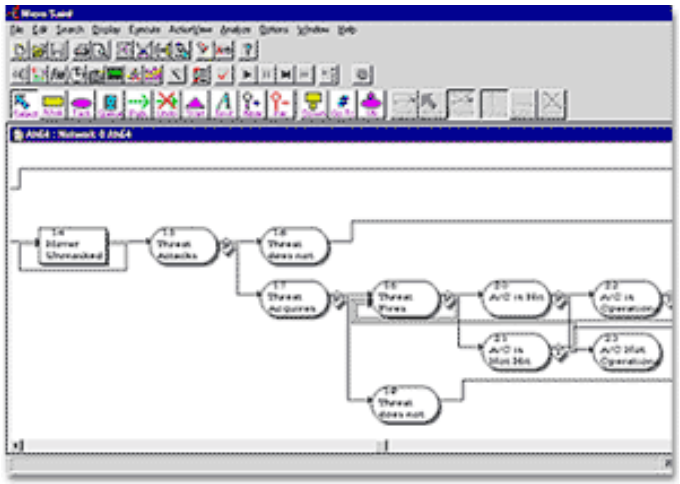
Example 3: HLA to the rescue

- Nice support for development process
  - Standard data exchange & time management schemes
- Complex system/process with own logic
- No Lisp hooks so we used Java gateway
- Result: 10 times slower than previously



Example 4: ACT-R/IMPRINT

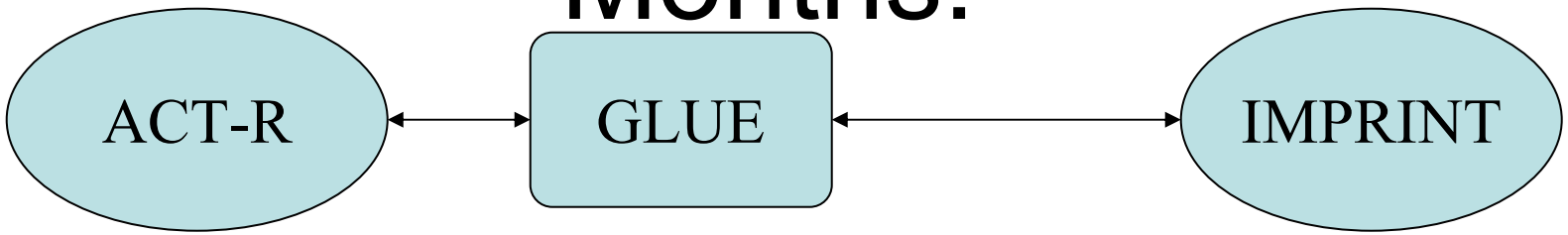
- IMPRINT
  - Discrete-event network simulation development tool
  - Designed specifically for modeling human performance
  - Funded by ARL built by MAAD
  - Can be used as a general-purpose simulation engine
- Done about 8 IMPRINT/ACT-R models
- Integration time brought down from 2 to 2months to days.
- Working to generalize to other simulations



NASA HEM: The first application

- Pilot taxi errors
- MAAD built IMPRINT model of aircraft taxiing
- CMU built ACT-R model of pilot decision-making

First IMPRINT/ACT-R: Months!

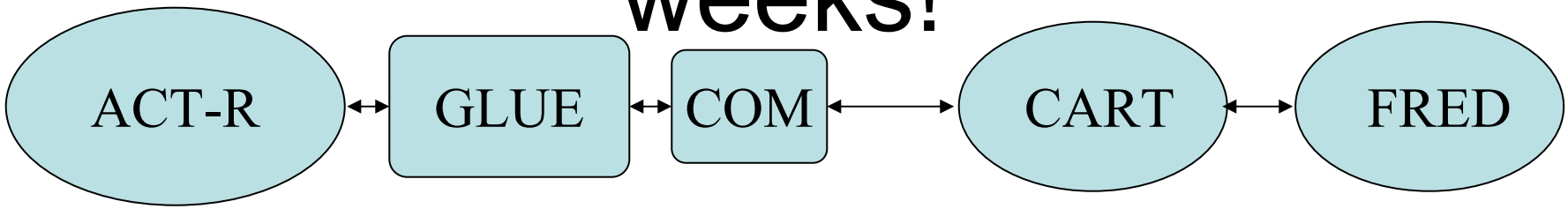


- IMPRINT has API based on COM
  - IMPRINT initiated ACT-R's decisions.
  - LISP glue had to single step IMPRINT and query IMPRINT's external variables.
- Months to finish integration
  - 300 lines IMPRINT's COM API
  - 1000 lines GLUE (one off)

CART: Next Application

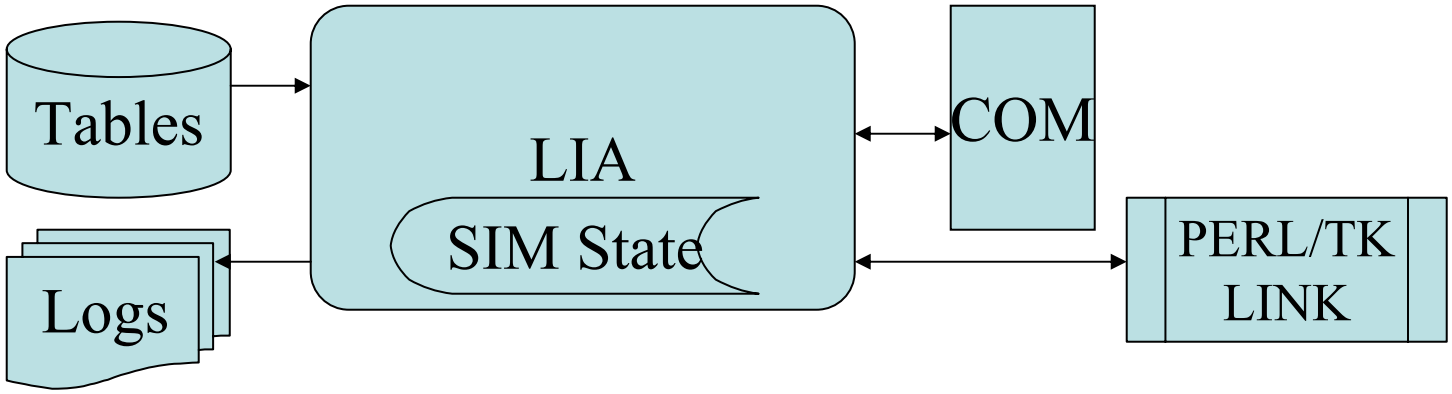
- Air Force version of IMPRINT
- Model of shootlist management for scud-hunting
- Runs on Silicon Graphics

CART / ACT-R: Down to weeks!



- FRED is a classified simulation of aircraft (JSF)
- Developed ACT-R HPM model from partial stub
- Used the External Macro Call (EMC) protocol
- Separated COM, EMC support from GLUE.
  - Needed only 4 of IMPRINTS API (just data)
  - Glue dropped to ~600 lines
  - COM & EMC support ~300 lines
  - Just weeks to build

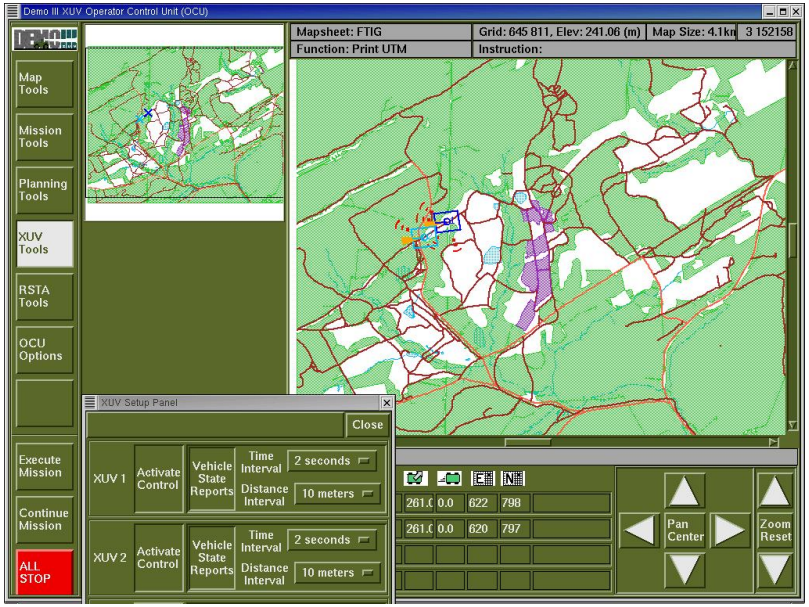
Write Once and Reuse: LIA

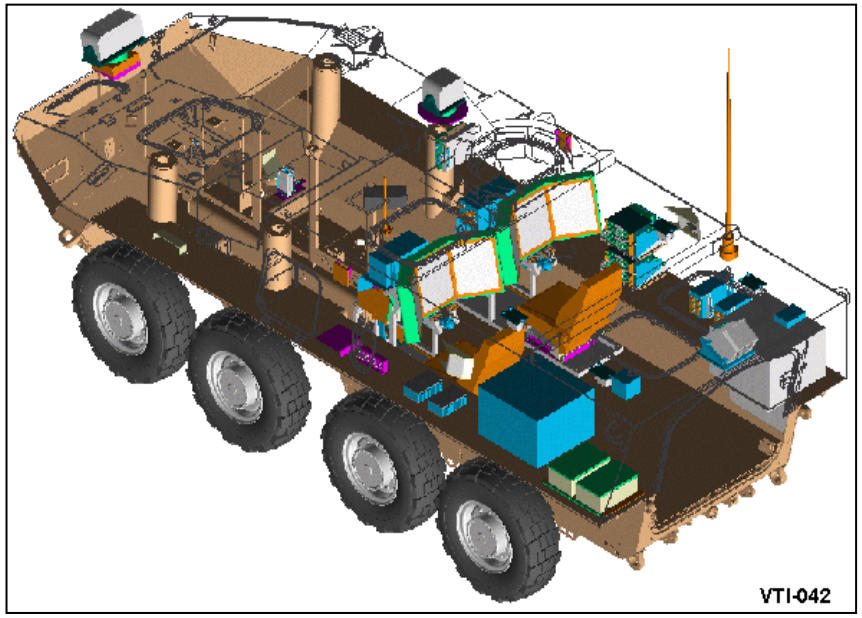


- Link IMPRINT ACT-R
  - Designed between projects.
  - Subset of ACT-R's API and Buffers
  - Uses Excel tables maintained by IMPRINT developer!
  - Defines a defEMC which handles the RPCs
  - About 400 lines with additional 200 for COM
  - Translates integers into symbols and simulation entities
  - Support for TK demo windows. (Perl for speed)

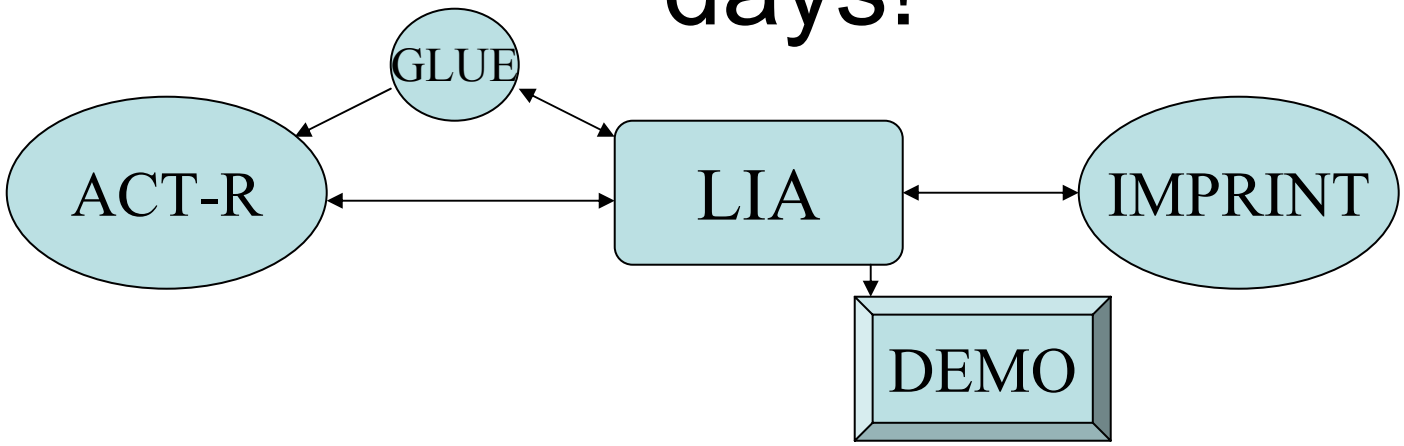
ARL ADA

- Build human performance model of the operation of robotic scout
- Modeled the navigation through complex displays
- Simplify the integration of IMPRINT and ACT-R





IMPRINT/LIA/ACTR: Done in days!



- Most complex IMPRINT model
- The basic integration done in days
  - Glue down to 200 lines
  - Debugged IMPRINT model from LIA's logs
- Built GUI to show ACT-R's decisions

Lessons Learned So Far Suggest a Future Approach

- Simulation is not going to change for us (we are not that big yet but we can/should do education of simulation community)
  - Insufficient primitives can make development costly
- Modeling is modeling
  - This aspect is no different in the context of external simulations
- Generate general-purpose API layer that solves the hard problems (language translation issues, timing)
- Need a thin special-purpose client that is fast and easy to develop (hours to days)
  - Which side/language of communication to put the thin client on? Simulation-specific?
- Cost/benefit of general/specific solutions (pay now, pay later)
  - The systems described took months, to weeks, to days of integration time

Further Questions on What to Provide for the ACT-R API?

- Should the simulation control cognitive processor?
  - ACT-R as cognitive server (or client)
  - Or Peer-to-Peer (Federation view like HLA)
- Do we need a general API for a cognitive system (popular enough to be a standard?)
- Is the ACT-R user API right for a simulation communication API?
- Reusable TCP/XML layer for ACT-R commands?
- ACT-R time rollback? (for TIME/WARP)