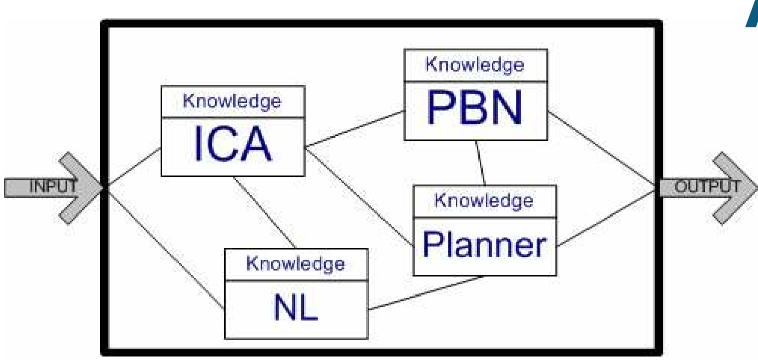
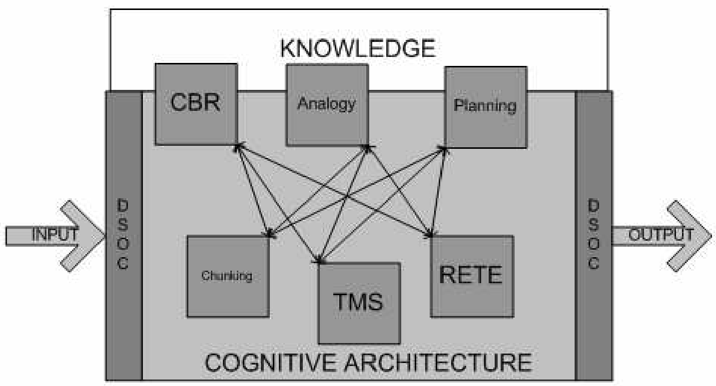
Modularity in Soar-based Applications: Practical Issues

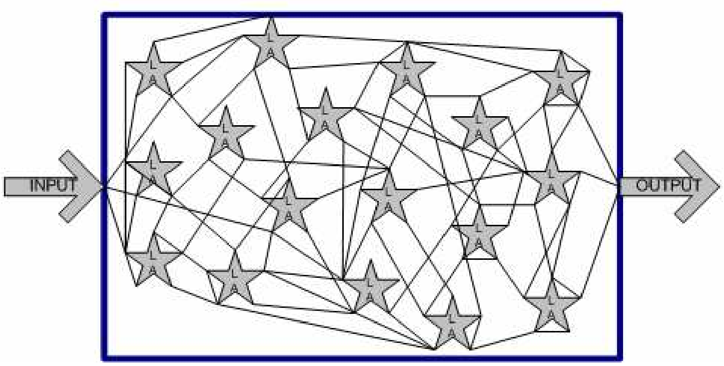
25 May 2006

Robert E. Wray
www.soartech.com

AI system engineering options



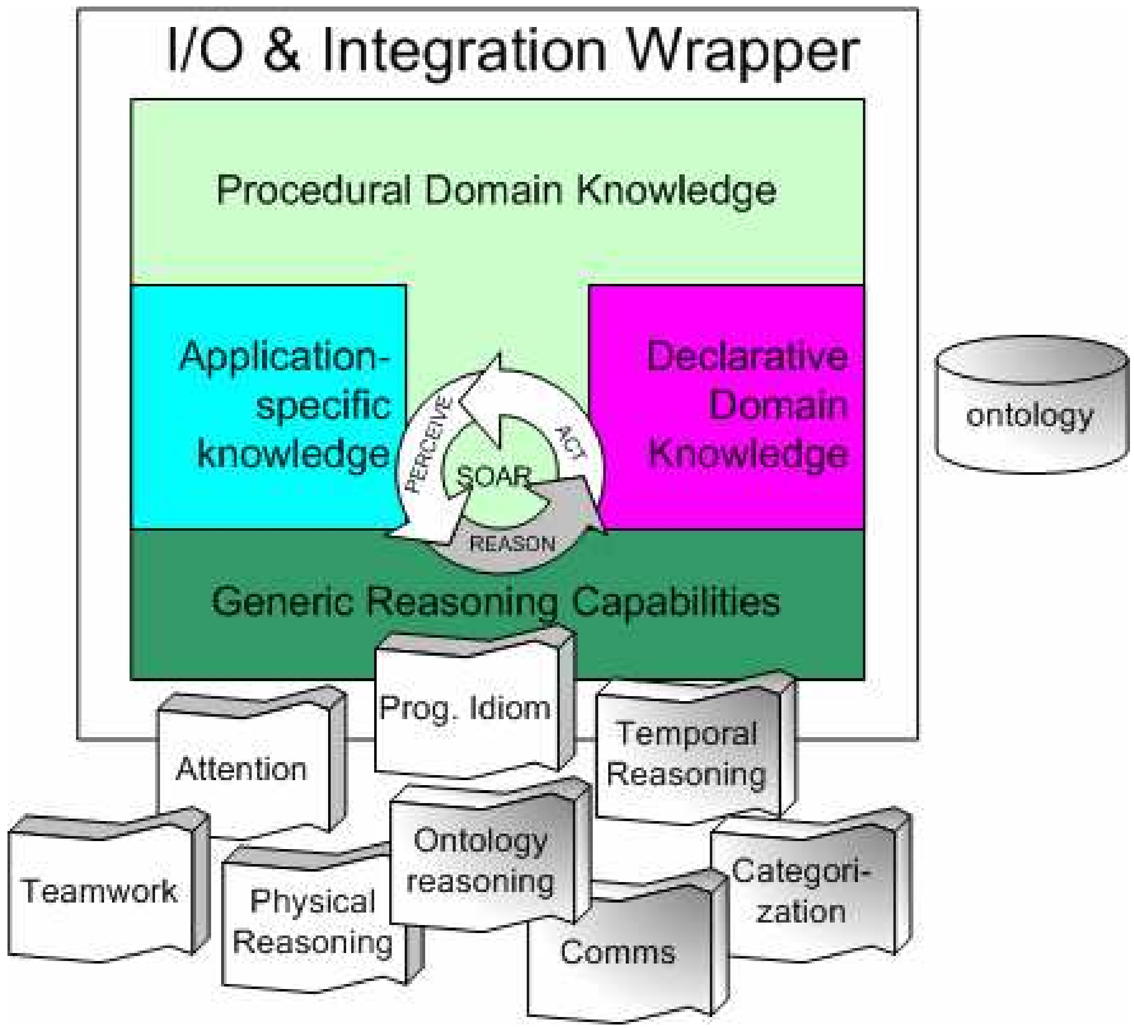




- Software Engineering (GOFAI):
  - Modular decomposition derived from functional decomposition
  - Fixed, engineered interaction & control
  - Knowledge matched to module
  - Black-box module operation
  - Includes “Engineered MAS” approaches
- Cognitive architecture approach
  - Uniformly encoded knowledge
  - White-box knowledge modules
  - Least-commitment control and knowledge integration
- DAI/Multiagent System approach
  - Opportunistic, unscripted interaction
  - Distributed (“no executive”) control
  - System behavior is “emergent”

Typical Soar-based Application

- Agent maps to human actor in a physical environment
- Agent should exhibit capabilities roughly comparable to the human agent in the environment
- Typical HBR/CM implementation is consistent with Soar theory:
  - Soar is the sole “intelligence” platform
  - All knowledge is dynamically integrated at run-time within Soar
  - Examples:
    - NASA-TD, TAS, RWA (STEAM), AMBR (SCA), etc.



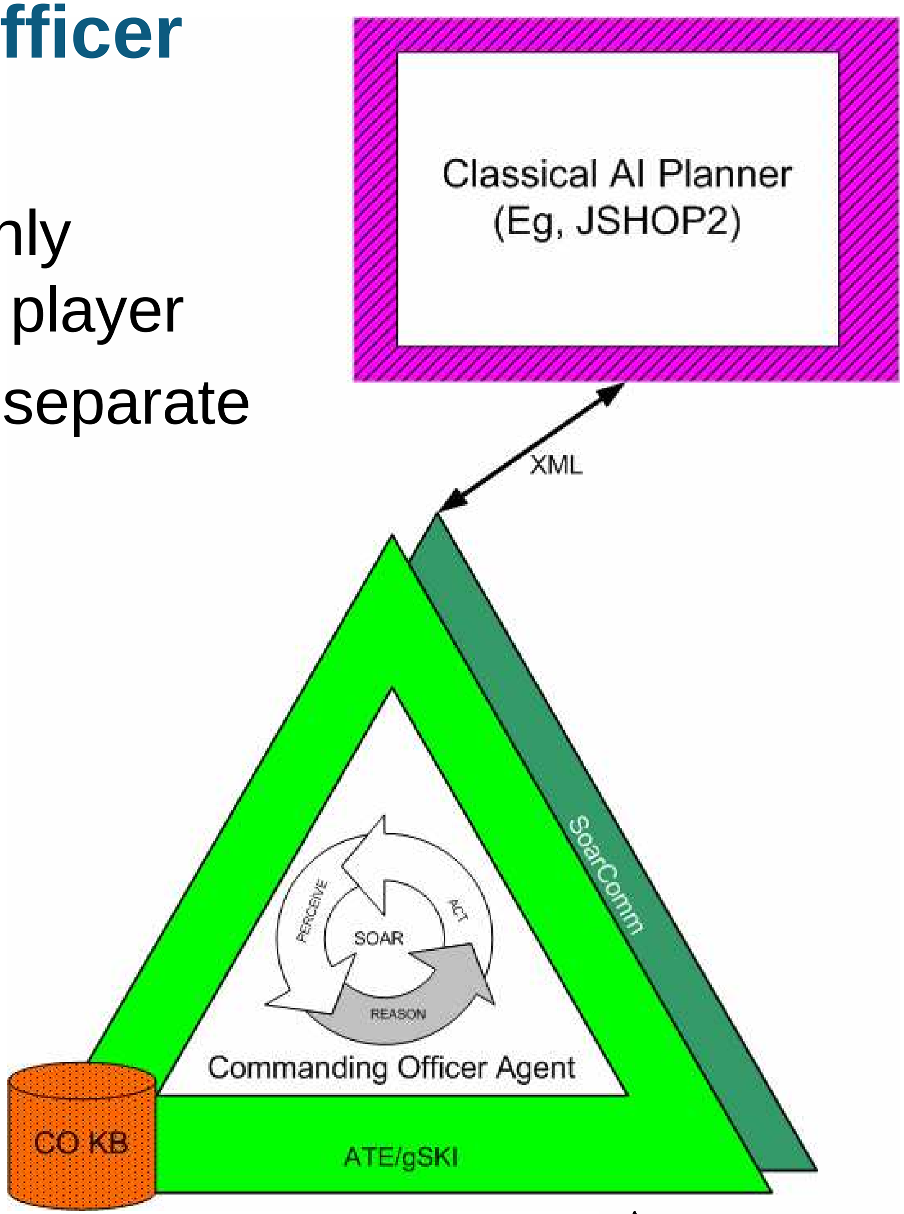
Soar Technology
Thinking *inside* the box.

Recent Soar Technology Application Architectures

- Current application development at Soar Tech demonstrates strong reaction to practical constraints of Soar
  - Result: System architectures beginning to look more like GOFAI systems than systems constrained by Soar theory

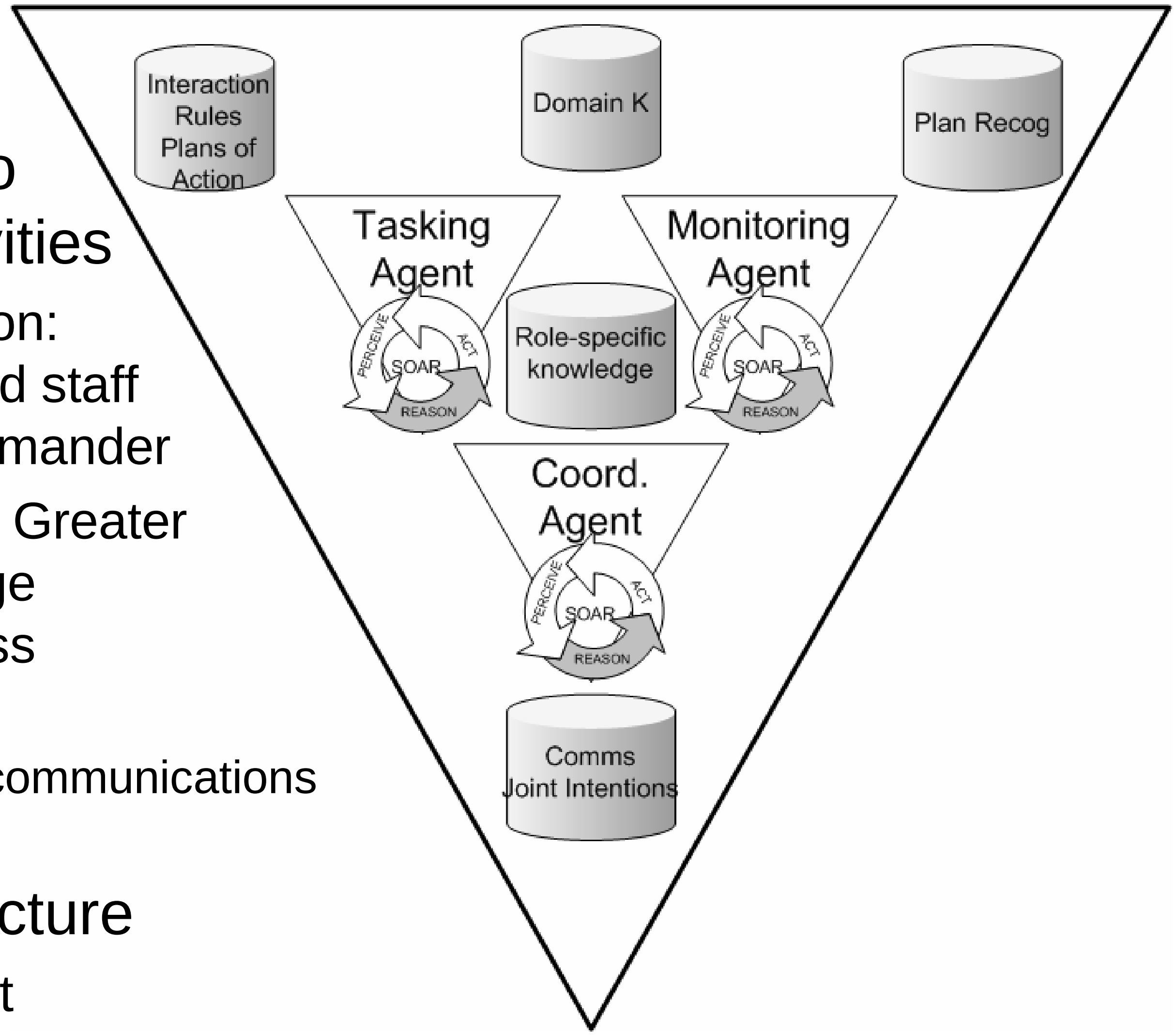
JFETS Commanding Officer

- "Command" function roughly comparable to RTS game player
- Realized as Soar agent + separate planning system



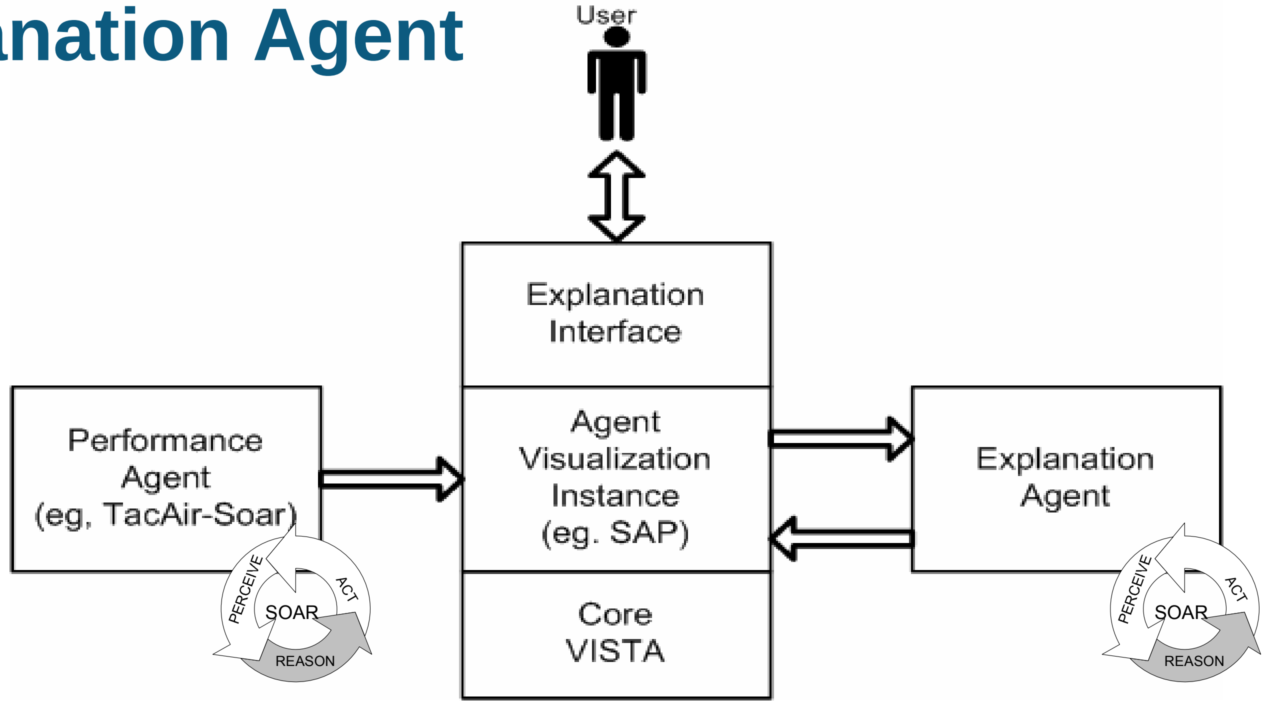
Commanding Officer Decomposition

- Decomposes command role into three distinct activities
  - Original assumption: model of command staff vs. individual commander
  - Partial motivation: Greater reuse of knowledge components across different domains
    - joint-intentions communications knowledge
- Extensible architecture
  - visualization agent



Soar Technology
Thinking *inside* the box.

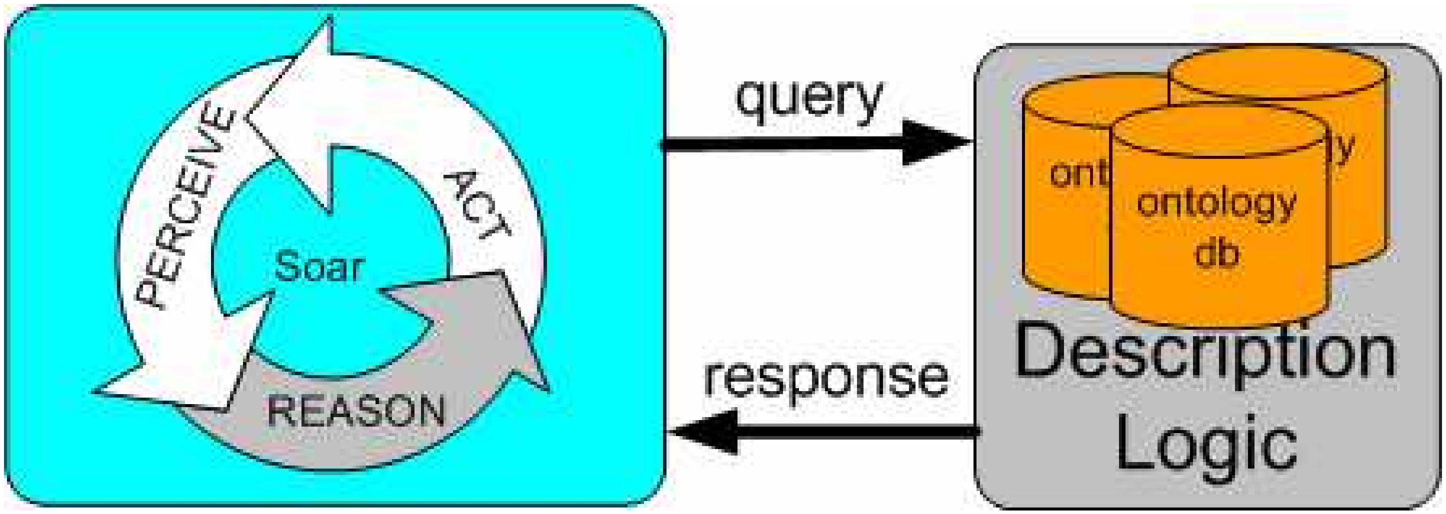
VISTA Explanation Agent



- Generation of explanations realized in a separate agent
- Architecture facilitates explanations from non-Soar agents

Soar Technology
Thinking *inside* the box.

Onto2Soar (proposed 2^{nd}-generation system arch)



- Move formal ontologies outside representation within Soar and use description logic tools to resolve ontology queries

Practical Issues in Soar Application Development

What is driving divergence from theoretical assumptions?

- Soar theory:
  - Completeness in functionality
  - Knowledge reuse across applications (especially knowledge for general capabilities)
  - Architectural assumptions enable run-time knowledge integration (interleaving and open, not encapsulated, knowledge dependencies)
- Application-development constraints:
  - Current release of Soar is not complete (research in progress)
  - Special-purpose mechanisms (e.g., planners) offer significant performance improvements
  - Knowledge reuse is the exception, not the rule
  - Knowledge development cost tends to scale super-linearly
  - Large knowledge bases do not necessarily provide adequate performance
  - No knowledge packaging methodology/tools
  - Soar is used for many non-HBR/CM applications
    - Are "Soar claims" specific only to human-inspired models, or to intelligent systems generally?

Soar Technology
Thinking *inside* the box.

Directions for application-development tools

- Research in knowledge packaging
  - Initial explorations/lessons: SCA (impasses), STEAM (annotations)
  - High utility for both traditional and GOFAI approaches
  - Preserves/enables “white box” modularity?
- Research/engineer “semantic interfaces”
  - Define good abstractions for Soar-Module information exchange
  - Example: Generic Soar-planner interface
- Research/engineer enabling technology for interfaces
  - Blackboards
    - Agent memory as blackboard (ala JESS)
    - Distinct shared memory component
      - Soar agent as blackboard (ala AIS)
  - Communication infrastructure
  - Understand and document trade offs!

Soar Technology
Thinking *inside* the box.

Conclusions

- Nugget
  - Soar theory offers compelling story for least-commitment control and dynamic knowledge integration
- Coal
  - Many practical limitations impede realization of theoretical benefits

- Unresolved questions
  - Which system engineering approaches are most appropriate for what kinds of applications?
  - What are “natural” units of agency for different kinds of applications?
  - Are Soar constraints in non-HBR systems useful?
  - What research and tools are needed to support the different directions for supporting application development?
  - How could Soar best be applied in DAI systems (if at all)?