déjà vu all over again: Towards tools for faster, cheaper, better modeling & development

Bob Wray

Soar 24

10 Jun 2004

Introduction

- Lots of focus/interest in new tools
- What are the technical barriers to better, faster, cheaper?
  - Design methodology & tools
  - Long-lived code development & debugging tools
  - Production level reuse
- Those who forget the past are doomed to repeat it
  - Some lessons from the past

Software Life Cycle

- Design
- Code/Reuse
- Debug/Verify
- Maintain

} Document

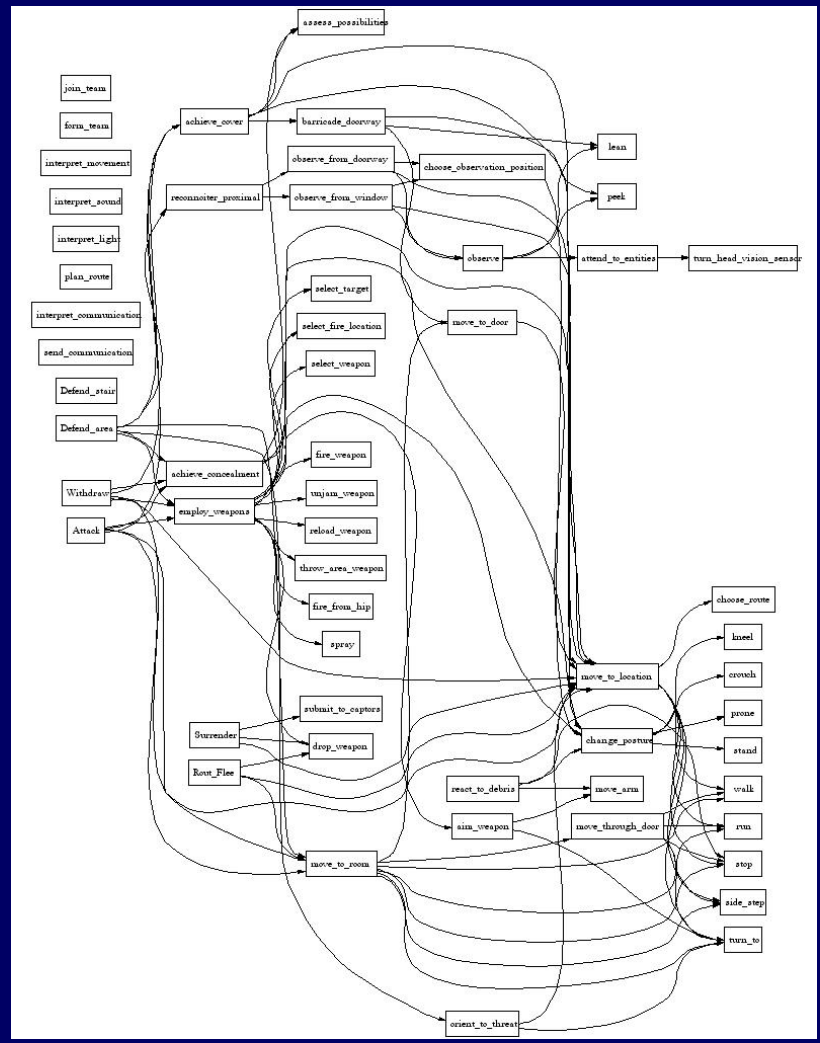
- Soar needs good tools for each of these elements
- Complete, interoperational tools across the life cycle are necessary for success
- Tools must be malleable (lots of requirements, sometimes conflicting)

Tensions

- Psychological plausibility vs. purely computational (AI) approaches
  - Low-level fidelity vs. (more immediate) high-level power
- Which users?
  - Highly trained knowledge developers vs. new Soar users vs. non-programmer domain experts
  - Research explorations vs. application-quality software
- Tractability (in development) vs. capability/scalability
  - Scripts -> FSAs -> Autonomous systems
- Programming solutions vs. providing capability (where does the intelligence really come from?)

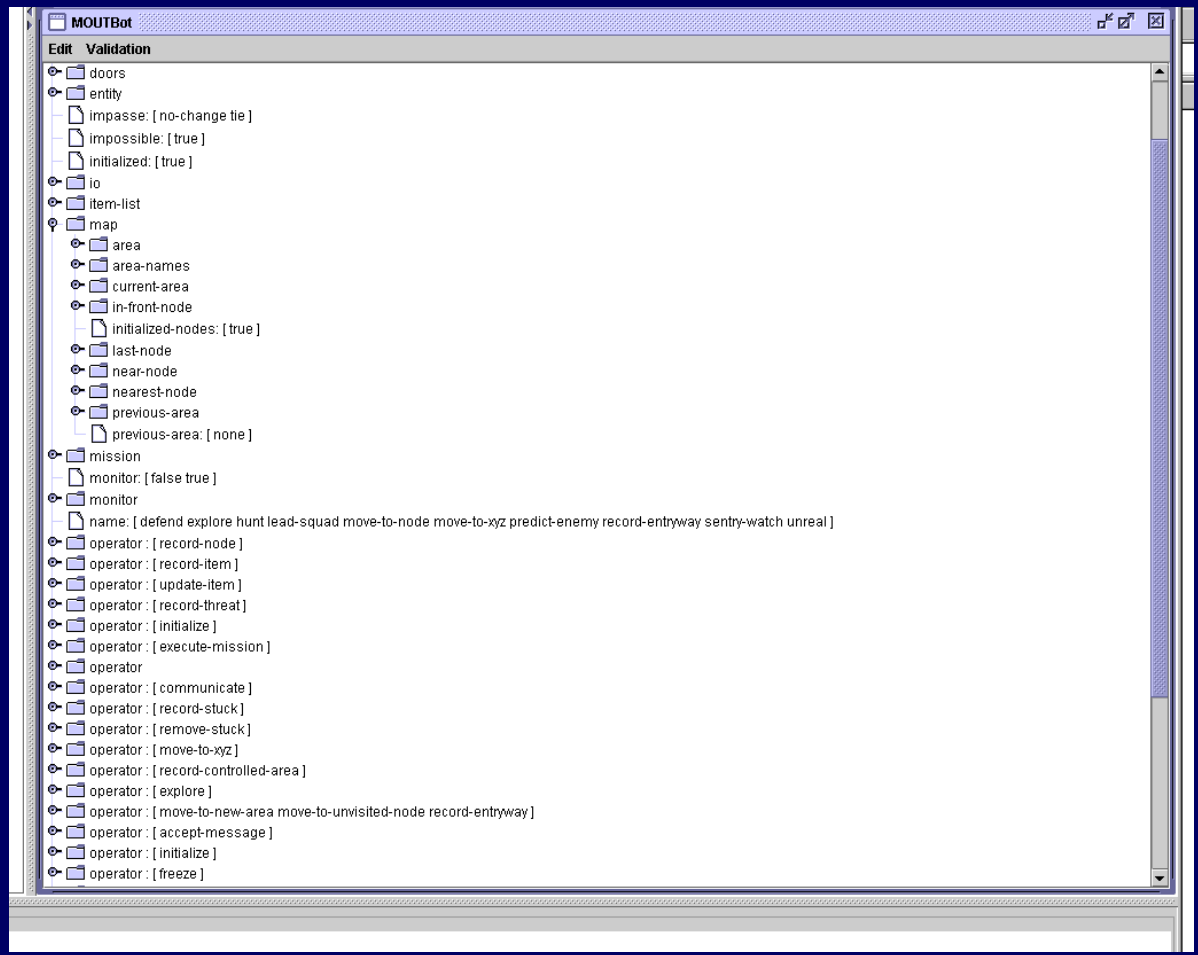
How do we design Soar systems?

- Operator hierarchies



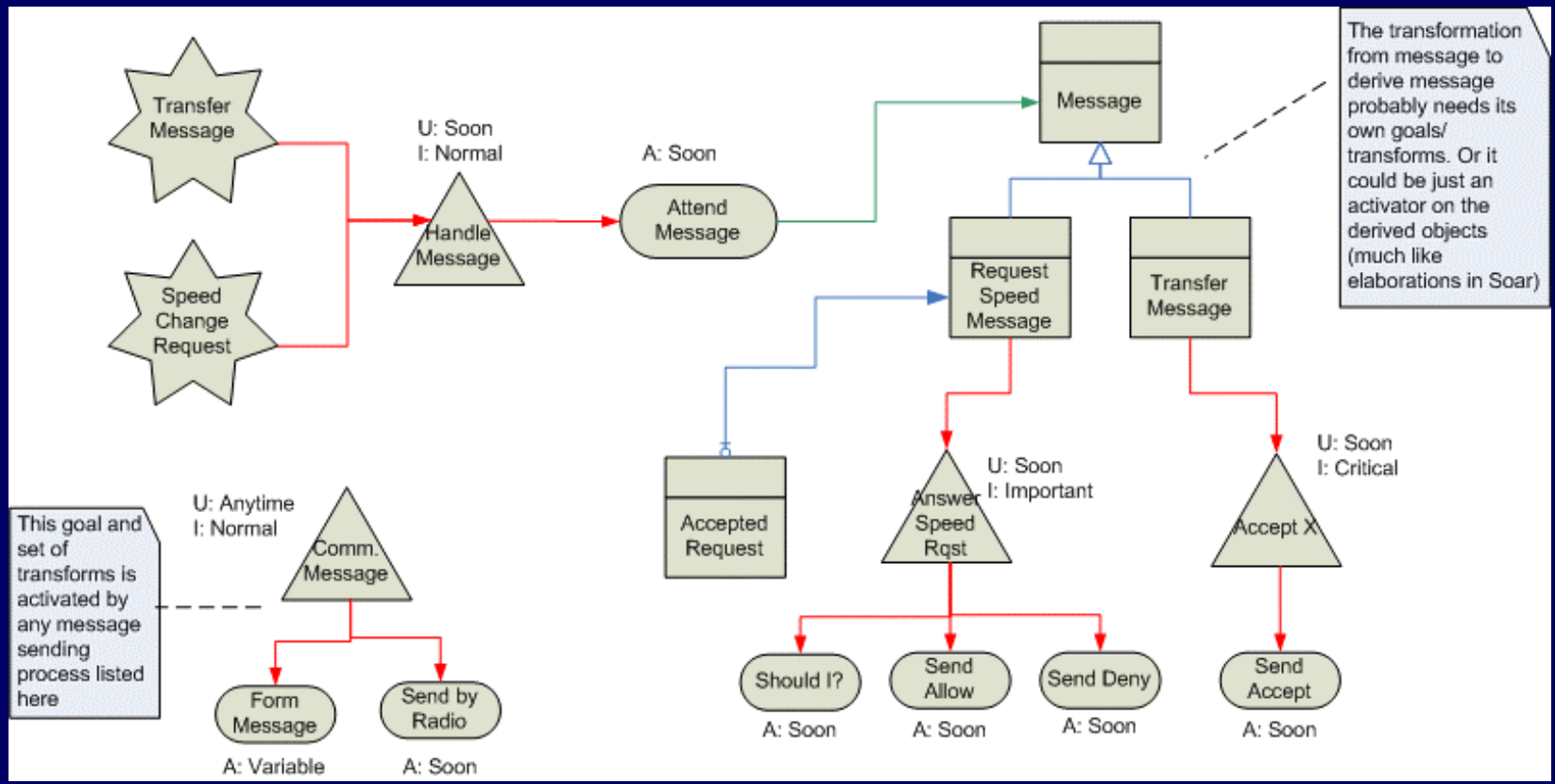
How do we design Soar systems?

- Operator hierarchies
- Data design



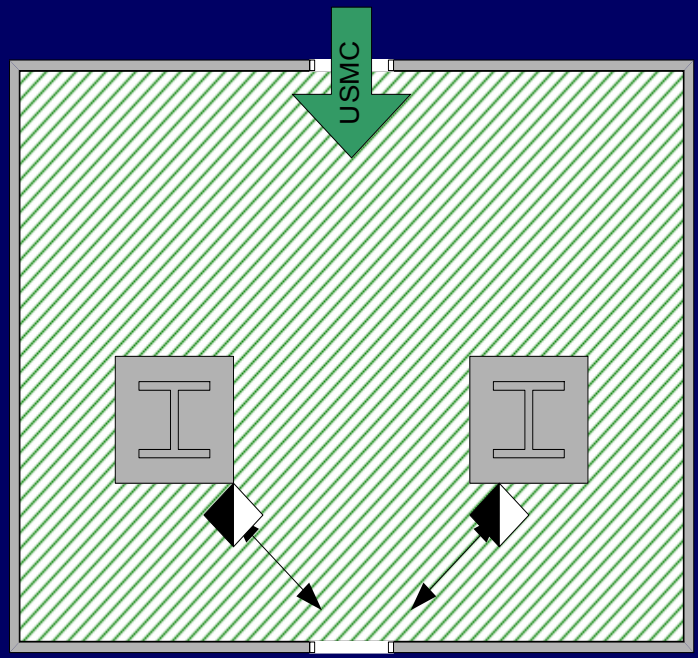
How do we design Soar systems?

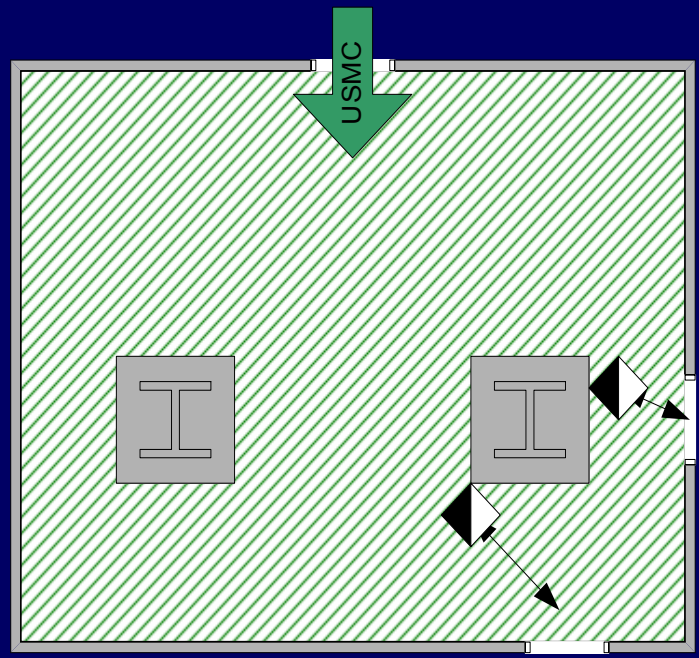
- Operator hierarchies
- Data design/BDI concepts



How do we design Soar systems?

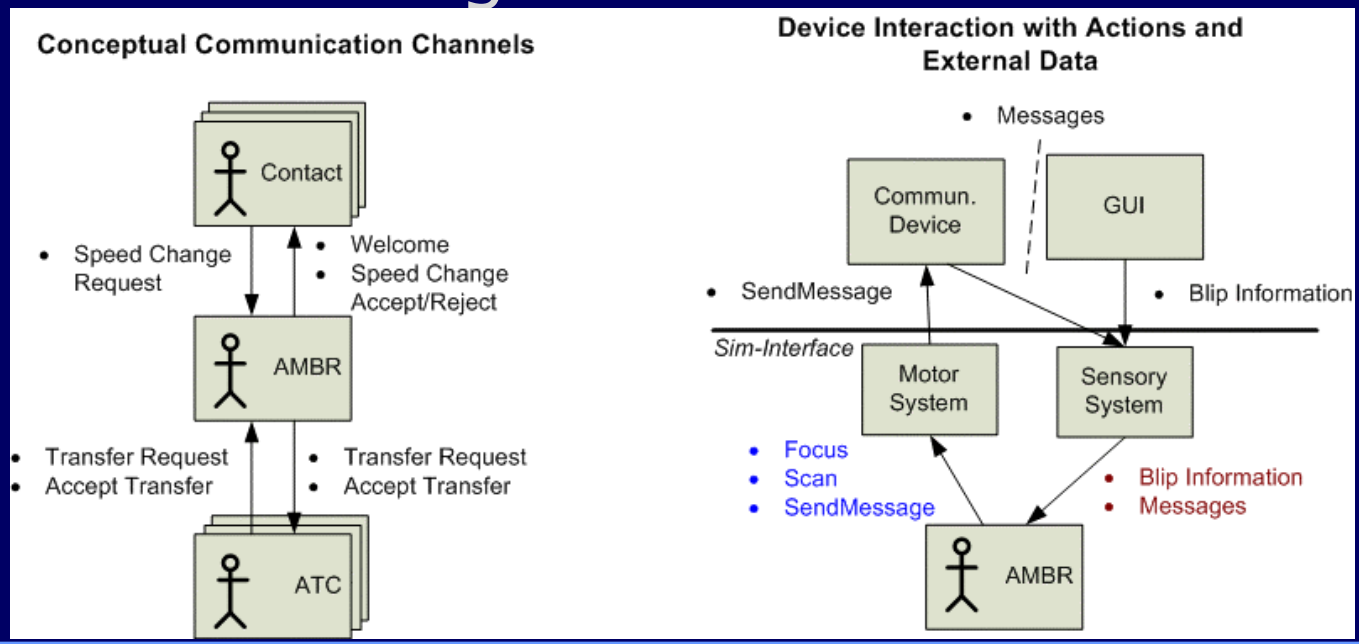
- Operator hierarchies
- Data design
- Use cases





How do we design Soar systems?

- Operator hierarchies
- Data design
- Use cases
- Information flow diagrams



Design Problems

- Little design support for new/novice user
  - No formalized design methodology
  - No examples of previous designs
  - Design decisions are not explicitly outlined/justified
- Designs are not reusable
  - Largely built with general, one-off tools (Visio)
  - Labor-intensive design iteration
  - Result: Design doesn't outlive initial implementation
    - (also true for traditional & OO software development?)
  - Design is inscrutable from the perspective of Soar productions
- Will BDPs address these issues?
- Does HLSR suggest a design methodology?
- Other ideas? (Soar Casebook)

Possible tools for design

- Agents community
  - UML/Agent UML
  - Agent-oriented programming
    - Prometheus
- Knowledge-based Systems
  - CommonKADS
    - Kalus et al report a KADS->Soar translator
  - Protégé forms (declarative knowledge capture)
- HCI/Cognitive Science
  - GOMS-level design/compile systems
    - ACT-Simple, St Amant & Ritter ICCM 04, etc.
  - TAQL (PSCM-level design tool)
- Redux?

Design ?s

- Should we be trying to codify the design process before building specific tools to support design?

Code & Debug

- Soar Development Environment (SDE)
  - Fully integrated Emacs-based Soar editor & run-time system. Circa ~'95
  - Powerful, well-designed, extensible, …

Code & Debug

- Soar Development Environment
  - Fully integrated Emacs-based Soar editor & run-time system. Circa ~'95
  - Powerful, well-designed, extensible, …
  - Not a success. Why?
    - Tied directly to a specific implementation of Soar
    - Soar changed, developer had other priorities, was never updated for newer versions of Soar

Code & Debug

- New Development tools:
  - Visual Soar/VS-Eclipse/HLRBL
    - What should we doing to ensure the tools can outlive/adapt with the next major architecture evolution?
  - Suggestions:
    - Define intermediate language (HLBRL's XML?)
    - Insulate tools from direct dependence on kernel
      - Independent models of the architecture (AsmL spec; ST,PSU Soar ontological models)
    - Extend Soar language for tool support
      - Built-in support for SDE/VS templates (allow/support user-defined templates?)
      - WME type hierarchy
    - Others?

High level languages

- Task Acquisition Language (TAQL)
  - Goal: Specify programs at the level of PSCM
  - Anticipated result:
    - Faster, cheaper model development
    - Abstract architecture details (remove barriers to entry)

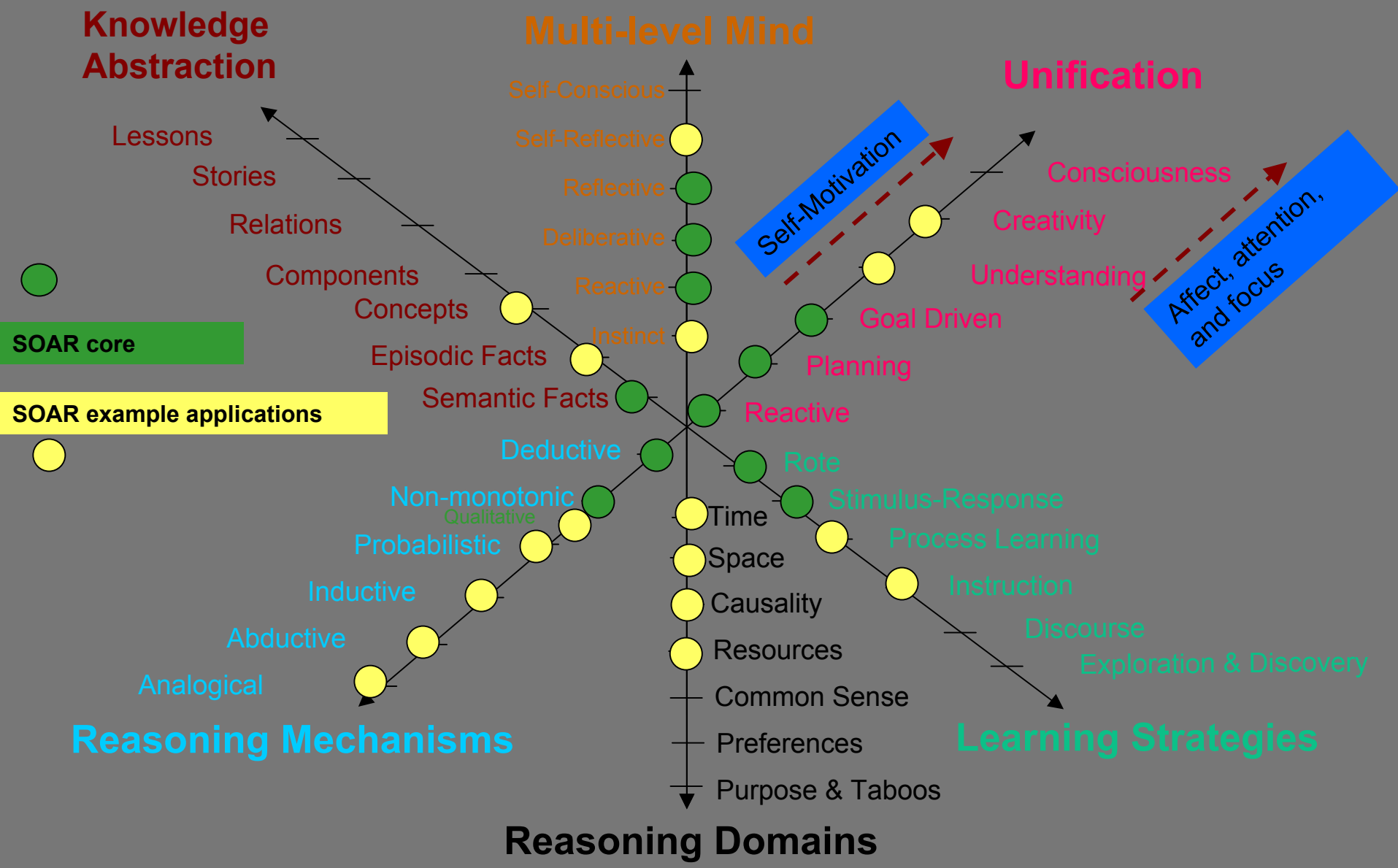
High level languages

- Task Acquisition Language (TAQL)
  - Goal: Specify programs at the level of PSCM
  - Anticipated result:
    - Faster, cheaper model development
    - Abstract architecture details (remove barriers to entry)
  - Actual result:
    - Short-lived/little penetration in Soar community
    - Mapping from TAQL/PSCM to Soar is incomplete
      - Users created TAQL programs but had to debug at the Soar level (even higher barrier to entry)
      - Not all learned knowledge mapped to PSCM

High level languages

- New approaches to high level languages
  - HLSR/HLRBL/ACT-Simple/St Amant & Ritter
    - What are the abstractions these languages support?
    - What do you lose when working at the higher level? (efficiency? fidelity? Expressive power?)
    - Do these solutions make it possible to work completely at the higher level (complete abstractions)?

A glass half empty/half full



Reuse

- Why is it so difficult to reuse production code?
  - "tangling" of domain, procedural, and architecture details
  - Architecture resource conflicts (goal stack)
  - Lack of design and documentation
  - Lack of intention/publication
    - ♦ additional work is generated by publication of a reusable component!

Goals for reuse

- Goals: Demonstrated production knowledge reuse
  - Soar component definition
    - How do we make it simple to define/understand "API" for production knowledge modules?
      - Aspects (e.g., STEAM)
      - Components (e.g., lookahead, SCA, NL-Soar)
    - Idioms for goal stack contentions (Soar 6?)
  - Monotonic production sets
    - Every new system includes all the knowledge of all previous systems
    - More practical compromises:
      - Extend default rules
      - Sourceforge behavior libraries (already exists?)

Discussion

- What are the technical barriers to better, faster, cheaper?
  - Design methodology & tools
  - Long-lived code development & debugging tools
  - Production level reuse
- What are non-technical barriers?
- Are there community goals we can pursue together?