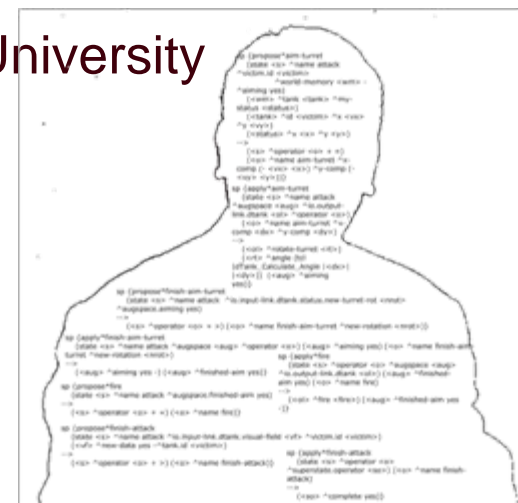


# Herbal and the Herbal Viewer

Frank E. Ritter<sup>2</sup>, Mark A. Cohen<sup>1,2</sup>, Steve Haynes<sup>2</sup>,  
Geoffrey P. Morgan<sup>2</sup>

1 BA CS & IT Department, Lock Haven University  
2 School of IST, Penn State



[acs.ist.psu.edu](http://acs.ist.psu.edu)

This project was supported by the US Office of Naval Research, award number N000140210021. We also thank Urmila Kukreja, Isaac Council, and the students of IST 402 the last two years for their work with and comments on Herbal.

The views expressed in this article do not necessarily reflect the positions or the policies of the U.S. Government, and no official endorsement should be inferred.

# The Problem






 (Ritter et al., 2003)

- Cognitive architectures appear like assembly code
- Explanation
- Reuse
- Ease of use
- ↙ A high-level language that maps more directly to the domain the user is familiar with


# Existing Solutions

- Visual Soar
- SoarDoc
- ViSoar
- G2A
- All graphical IDEs, e.g., JACK, iGen, Cogent

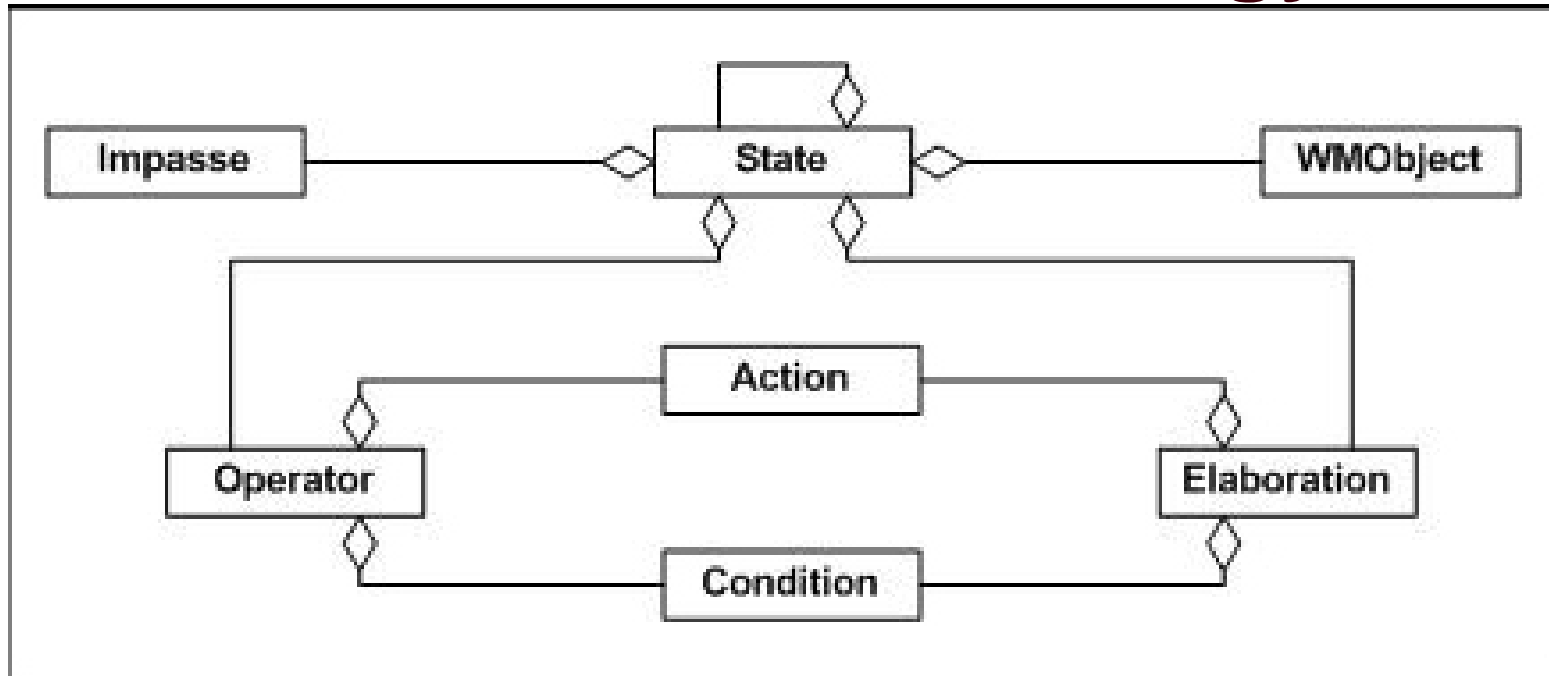
# Using a High Level Behavioral Representation Language

- Design rationale anchors model explanations  
( Haynes, 2001;  Haynes et al., 2004)
  - Design knowledge capture during development
- Model description needed for explanations ( Haynes, 2003)
  - Create model parts within IDE
- Responsibility-driven approach through a compiler
  - Organize knowledge and rules  
( Haynes et al., 2004;  St. Amant & Ritter, 2004, [www4.ncsu.edu/~stamant/G2A](http://www4.ncsu.edu/~stamant/G2A))

# Herbal Design

- Augment existing planning language with design rationale
  - ↪ Chose PSCM, RDF, and Protégé : tool availability, generality
  - ↪ Also studied direct translation ( St. Amant & Ritter, 2004)
  - ↪ Ontology also helps structure
- Explanation from declarative representation + rationale
- Compile into Soar rules (XSLT)
  - ⇒ (could also compile into ACT-R, JACK ?)
- Designed to leverage VISTA
  - ⇒ (declarative representation supports model tracing)  
[acs.ist.psu.edu/vista](http://acs.ist.psu.edu/vista) for our local training examples

# The Herbal Ontology



Programming in the Herbal high-level language involves instantiating objects using these ontological classes.

Programming a model is reduced to instantiating objects from a set of fixed classes, instead of coding the classes and structure implicitly in a large set of heterogeneous Soar productions.



# Herbal Integrated Development Environment (IDE) - Overview

The screenshot displays the Herbal IDE interface for a project named 'blocksWorld-0.4'. The window title is 'Protégé 2.1.1 (file:\C:\Documents%20and%20Settings\mcohen\My%20Documents\Development\PSU\HerbalIDE\workspace\Her...'. The interface includes a menu bar (Project, Edit, Window, Help) and a toolbar with icons for file operations and editing. Below the toolbar are tabs for 'Classes', 'Slots', 'Forms', 'Instances', 'Queries', and 'Model Attributes'. The 'Classes' tab is active, showing a class hierarchy on the left. The 'Display Slot' is set to 'herbal:Name', and the 'Direct Instances' list shows 'BlocksWorldState'. The main workspace is divided into several panels: 'Herbal:Name' (containing 'BlocksWorldState'), 'Herbal:Definition' (containing 'The Blocks World Problem Space'), 'Herbal:Operators' (containing 'MoveBlockToTable' and 'MoveBlockToBlock'), 'Herbal:Elaborations' (containing 'GoalStateReached'), 'Herbal:WorkingMemory' (containing a list of blocks: B, A, Table, COnTopOf, BOnTopOf, AOnTopOf, C), 'Herbal:How-it-works', and 'Herbal:Purpose'.

blocksWorld-0.4 Protégé 2.1.1 (file:\C:\Documents%20and%20Settings\mcohen\My%20Documents\Development\PSU\HerbalIDE\workspace\Her...  
Project Edit Window Help  
Classes Slots Forms Instances Queries Model Attributes  
Classes V  
:THING A  
:SYSTEM-CLASS A  
herbal:HerbalClass A  
herbal:Expression A  
herbal:Model (1)  
herbal:Production  
herbal:State A  
herbal:TopState (1)  
herbal:WMObject  
Display Slot  
herbal:Name  
Direct Instances V C  
BlocksWorldState  
BlocksWorldState (type=herbal:TopState, name=blocksWorld-0.4\_Instance\_1) C X  
Herbal:Name  
BlocksWorldState  
Herbal:Definition  
The Blocks World Problem Space  
Herbal:Operators V C + -  
MoveBlockToTable  
MoveBlockToBlock  
Herbal:Elaborations V C + -  
GoalStateReached  
Herbal:WorkingMemory V C + -  
B  
A  
Table  
COnTopOf  
BOnTopOf  
AOnTopOf  
C  
Herbal:How-it-works  
Herbal:Purpose

# Support in Herbal Viewer for Explanation



The screenshot shows the Herbal Viewer interface with the following components:

- Model Properties Graph:** A transition diagram with states: top-state, chase\*chase, wander\*wander, retreat\*retreat, and attack\*attack. Transitions are labeled "operator no-change".
- Model Tree:** A hierarchical tree of states and operators with their respective frequencies.
- State Trace:** A list of states and operators with their counts.
- Operator Trace:** A list of operators with their counts.
- Commentary:** A section for user notes.
- Vista Commands:** A section for system commands.

Yellow callout boxes provide explanations for various parts of the interface:

- Planned** (underlined): *How do I use it?* - model
- How does it work?** - Contrast classes
- Manual** (underlined): *How do I use it?*
- What** - structure
- What** - identity
- Definition** - structure + IDE
- When** - order - how often
- What** - relation

Arrows point from these callouts to specific elements in the software interface.



# Herbal Viewer and Soar 8.6

The viewer can now send commands to Soar directly, thanks to Soar 8.6

The screenshot displays the Herbal Viewer - Version 1.0 interface. The window title is "Herbal Viewer - Version 1.0". The menu bar includes "File" and "Window". The toolbar contains icons for "Show Graph Options", "Open Soar Model...", "Step", and "Stop". The "HERBAL" logo is visible in the top right corner.




The interface is divided into several sections:

- Model Structure:** A tree view showing the model hierarchy:
  - Model
    - BlocksWorldState (50%)
      - ResolveTie (50%)
        - PreferBOnC (100%)
        - PreferAOnB

- Visual Explanation:** Contains two tabs: "Visual Explanation" (selected) and "Explanation Properties".
- Transition Diagram:** A diagram showing a transition from "BlocksWorldState" to "ResolveTie". A green arrow labeled "operator tie" connects the two states. The "ResolveTie" state is represented by a box containing two small circles.
- Trace Console:** Contains four tabs: "State Trace", "Operator Trace", "Trace Commentary", and "Soar Console" (selected). The console displays the following text:
 

```
{O8 ^ss S1
  ^|herbal:Definition| |If there is an operator proposed t
  ^|herbal:How-it-works| ^|herbal:Name| |PreferBOnC|
  ^|herbal:Purpose| |to create a preference that favors pu
  ^|myOp| O6)
```
- Input and Output:** At the bottom, there is a text input field containing "p o8" and a "Run" button. Below the input field are "Clear" and "Save..." buttons.

# Herbal: Users to Use, Test, Expand Herbal & dTank

- IST 402: Models of human behaviour (12+3, and 38+3 students)
- BRIMS Tutorial (~25 tutees) (Cohen et al, 2005)
- Use of dTank Microworld to understand, create, and exercise adversarial Soar models
- Class projects
  - MacSoar 8.5 compilation  (see web site)
  - Architectural comparisons  (Sun et al, 2004)
  - Example reuse curves  (Morgan et al., 2005)
  - Errors and Soar (Ritter et al., in prep.)
- In general, the students encountered fewer problems programming in Soar when using Herbal

## Herbal Users: (cont.)

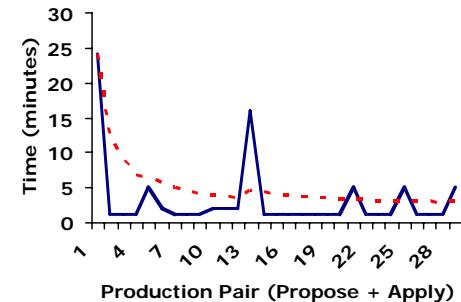
- 4/9 of the final group projects used Herbal to create their final projects (all groups used Herbal in their homework).
- 2/4 Herbal teams had a model that ran, and 2/4 using plain Soar had a model that ran.
- The most complex of the working models was an Herbal model that contained 16 pages of rules with 25 operators (>50 rules).
- All projects used Herbal's facility to include a header in their code that hooked up the models to dTank as they loaded.
- Students noted Herbal's ability to reuse conditions and actions across operators. Students also commented on the usefulness of Herbal's ability to automatically generate comments.
- During the course, students noted that Herbal did not support impasses, as one of the major disadvantages. Impasses added in v. 0.9.
- The other disadvantages noted by students were related to usability issues with Protégé. A new Protégé version (version 3.0) addresses many of these issues.

# Why Will Herbal Work?

- Principled design based on a theory of knowledge (PSCM, roughly and extended)
- Based on theory of explanations
- Based on data on explanations study
- New payoff - explanations
- Software engineering principles
  - Modularity
  - Software reuse
  - Design patterns
- No lost expressiveness (almost)
- Extendable by users / Common tools
- User base used for feedback (~12+2+36+30)
- Designed for usability by CS/HCI/Psy/IS team

# Herbal's Golden Nuggets

- Eases maintainability
- Accelerates programming
  - 3x Productivity (Yost with TAQL, 3 min./production)
  - 1x Productivity (IST 402 class, no complaints of Soar)
  - 1x-2x (Kukreja, first two programs @ 10 hours/prog.)
  - 3x productivity (Morgan et al., 2005)
- Promotes reuse (document and import models), Operators and elaborations can share conditions and actions; states can share operators, elaborations, and impasses (Morgan et al. 2005)
- Explanations may increase learning and use (not mentioned by our subjects)
- Ver. 1.0 (14/6/05) includes changes suggested at BRIMS and ONR (save before compile, ability to include code in generated code, explanations in the viewer, contrast classes, a Soar console interface built directly into the viewer, etc).



# Lumps of Coal

- Hand-code Action and Condition expressions in Herbal
- No Working Memory Browser in Herbal Viewer
- Does not compile to other architectures, yet











# Summary (mostly Gold)

- High level compiler for Soar, designed for other architectures
- Code reuse support
- Example meta-architecture

## Future

- More explanations
- More example models

# References [acs.ist.psu.edu/papers/](http://acs.ist.psu.edu/papers/)

-  Cohen, M. A., Ritter, F. E., & Haynes, S. R. (2005). Herbal: A high-level language and development environment for developing cognitive models in Soar. In *Proceedings of the 14th Conference on Behavior Representation in Modeling and Simulation*. 177-182. 05-BRIMS-044. Orlando, FL: U. of Central Florida.
-  Councill, I. G., Haynes, S. R., & Ritter, F. E. (2003). Explaining Soar: Analysis of existing tools and user information requirements. In *Proceedings of the Fifth International Conference on Cognitive Modeling*. 63-68. Bamberg, Germany: Universitats-Verlag Bamberg.
- Haynes, S. R. (2001). Explanations in information systems: A design rationale approach. Unpublished Ph.D. dissertation, Departments of Information Systems and Social Psychology, London School of Economics.
-  Haynes, S. R., Councill, I. G., & Ritter, F. E. (2004). Responsibility-driven explanation engineering for cognitive models. In *AAAI Workshop on Intelligent Agent Architectures: Combining the Strengths of Software Engineering and Cognitive Systems*.
-  Morgan, G. P., Ritter, F. E., Stevenson, W. E., Schenck, I. N., & Cohen, M. A. (2005). dTank: An environment for architectural comparisons of competitive agents. In L. Allender & T. Kelley (Eds.), *Proceedings of the 14th Conference on Behavior Representation in Modeling and Simulation*. Orlando, FL: U. of Central Florida. 05-BRIMS-043.
-  Morgan, G. P., Haynes, S., Ritter, F. E., & Cohen, M. (2005). Increasing efficiency of the development of user models. In *Proceedings of the 2005 Systems and Information Engineering Design Symposium*. E. J. Bass, (ed). IEEE and Department of Systems and Information Engineering, University of Virginia: Charlottesville, VA.
-  Ritter, F. E. (2004). Choosing and getting started with a cognitive architecture to test and use human-machine interfaces. *MMI-Interaktiv-Journal*, 7, 17-37. [useworld.net/mmij/musimms](http://useworld.net/mmij/musimms).
-  Ritter, F. E., Baxter, G. D., Jones, G., & Young, R. M. (2000). Supporting cognitive models as users. *ACM Transactions on Computer-Human Interaction*, 7(2), 141-173.
-  Ritter, F. E., Shadbolt, N. R., Elliman, D., Young, R., Gobet, F., & Baxter, G. D. (2003). *Techniques for modeling human performance in synthetic environments: A supplementary review*. Wright-Patterson Air Force Base, OH: Human Systems Information Analysis Center. [iac.dtic.mil/hsiac/S-docs/SOAR-Jun03.pdf](http://iac.dtic.mil/hsiac/S-docs/SOAR-Jun03.pdf)
-  Amant, R., & Ritter, F. E. (2004). Automated GOMS to ACT-R model generation. In *Proceedings of the International Conference on Cognitive Modeling*. 26-31. Mahwah, NJ: Lawrence Erlbaum.
-  Sun, S., Councill, I., Fan, X., Ritter, F. E., & Yen, J. (2004). Comparing teamwork modeling in an empirical approach. In *Proceedings of the Sixth International Conference on Cognitive Modeling*. 388-389. Mahwah, NJ: Lawrence Erlbaum.