A Knowledge Base Browser Using Hypermedia

Tony Pocklington
McDonnell Douglas Space Systems

Lui Wang
NASA/Johnson Space Center

Discussion of a hypermedia system we are developing to browse CLIPS knowledge bases. This system will be used to help train flight controllers for the Mission Control Center.

A Knowledge Base Browser Using Hypermedia

Tony Pocklington
MDSSC-Houston/MDCB2KI

Lui Wang
NASA-JSC/PT4

Abstract

Currently under development at the Johnson Space Center (JSC) for use in the Mission Control Center (MCC) are a group of expert system to assist the ONAV flight controllers. These expert systems serve two functions: to act as an assistant during real-time and simulated flight operations, and to act as trainers for controllers when flight operations are not in progress. The Knowledge Base Browser (KBB) is a tool also currently under development. The goal is to augment the expert system in its role as a trainer.

The KBB will take advantage of the structure of the rule base. The rules and relation patterns are the basic nodes and links of the hypermedia system. Meta links and collection nodes can also be used to further organize information for the loosely structured rule base. Browsing this knowledge base will be accomplished either by navigating through the various collection nodes that have already been defined, or through a query language.

1.0 Introduction

Currently under development in the Mission Control Center (MCC) at the Johnson Space Center (JSC) is a collection of expert systems to support the Onboard Navigation (ONAV) flight control position. The primary function of these expert systems is to act as an assistant to the ONAV flight controllers, who support the Ascent, Rendezvous and Deorbit/Landing Phases of a Shuttle mission.

Along with the expert systems are several other programs under development, including a data logger, a playback program, a program to display analog data, a plot program, and the Knowledge Base Browser (KBB) that is the focus of this paper.

2.0 Reasons For Browser

The KBB will serve two purposes: to assist in the verification of the rule bases for the various expert systems, and to augment the training of the flight controllers.

2.1 Verification

The verification of an expert system is a difficult task. In a rule base language, each rule is an independent entity. As the number of rules increases, the interaction of the rule base is more difficult to track and hence more difficult to verify. A typical knowledge base application, such as the Entry ONAV Expert System which has in excess of three hundred rules, needs a browsing mechanism that could simplify the process of inspecting these rules. One of the functions of the KBB will allow users to examine the causal relationship of the rules and facts in a rule base language.

2.2 Training

The KBB will also be an excellent training tool. A great deal of work goes into the development of a rule base, and implementing browse and query functions will allow these rule bases to be treated as experts. Novice flight controllers will be able to use the browser to read through all or

part of the rule base, while more experienced users will probably find the query capability useful to answer specific questions. For example, a relatively new controller might want to read all rules related to inertial measurement unit (IMU) failures, whereas an experienced user would be more interested in a rule regarding the reselection of an IMU which had had a prior communications fault.

3.0 CLIPS

The ONAV expert systems were developed using the 'C' Language Integrated Production System (CLIPS). CLIPS is a forward chaining rule base production system developed by the Software Technology Branch at NASA/Johnson Space Center. CLIPS has capabilities similar to OPS5 (Official Production System), and ART (Automated Reasoning Tool). The purpose of the tool was to address delivery of expert systems to conventional operational environment.

3.1 CLIPS Knowledge Base

In a knowledge base system like CLIPS, the notion of nodes and links are implicitly embedded in the rules. The fact patterns are viewed as links, and the rules are viewed as nodes. Each rule contains partial knowledge of the overall system and acts opportunistically based on the incoming data stream. Rules in general define how to transition between the different states. The antecedents of the rules capture the current events, and the consequents of the rules modify the system and take it to a new state. The rule nodes are one of the atomic units in the KBB. This is because each rule is syntactically an independent unit, but it only conveys partial information. The fact patterns, on the other hand, are the agents that link all the rules together. Therefore the fact patterns are the atomic links for the KBB.

3.2 CRSV

Some of the data structures for the KBB were taken from the Cross-Reference, Style and Verification (CRSV) utility. Some of the ideas for the verification part of the KBB came from CRSV as well.

CRSV is a tool that was also developed at JSC to help verify the CLIPS rule bases. The focus of this tool is to address the software engineering practice in rule base programming. For example, the tool detects and issues warning and error messages for "bad" programming style, syntax errors, and inconsistent data type. It also performs cross referencing among relations and variables. In addition, CRSV collects statistical information that may help developers to improve the system performance. Even though the target users and the purpose of the two tools, CRSV and KBB, are different, they share one major common function, which is the cross referencing or the browsing function. This function carries different meaning based on the users' perspective. The rule base programmers use the cross referencing function to verify facts assertions and retractions to the fact base. The domain experts use the browsing function to verify the completeness of the specification. Finally, the novice trainers use the browsing function to understand the causal effects of the system behavior.

4.0 Browsing

One of the most important features of the KBB will be the browsing function. The two means of browsing the knowledge base will be navigating through the various collections of rules and making queries of the rule base.

4.1 Navigation

The user will be given a list of collections, which may contain either rules or other collections of rules. Selecting a collection will cause a new list to be displayed, which will either be rule names or the names of more collections. A window displaying where the user is in the tree hierarchy of collections will also be available to keep users from getting lost.

4.2 Queries

The other means of traversing the rule base will be a query function.

4.2.1 Query Language

There will be a very basic query language available for the users. The query language will allow users to do simple searches on text strings and to combine these queries using logical operators.

4.2.2 Intelligence

It may be necessary to build some knowledge into the query language parser, in order to handle certain context problems. For example, suppose a user wants to view all rules pertaining to TACAN data being inhibited. A natural search string that he would then formulate would be "tacan inhibit". The actual string that must be searched for, however, might be "tacan-aif-pass inhibit". Since the word "inhibit" might also appear as the value of a variable, it might be necessary to search first on the words "tacan-aif-pass", and then apply rules to eliminate instances that did not either have the word "inhibit" or a variable whose value might be "inhibit". The rules to handle these searches will probably take the form of another CLIPS knowledge base, allowing users to quickly alter the rules without having to recompile and relink the system.

4.2.3 Nested Queries

The system must also have some means of handling queries that return either too much or too little data. One way to handle this is to allow nested queries, where the data returned from one query is used as the search space for the next query. It must also be possible for users to back up, either partially or all the way to the beginning level, where the entire rule base is used as the search space.

5.0 Scripting

The final part of the system is the scripting, or creation of the collections. The easiest way to create a collection is to save the results of a query. The user will be given the opportunity to supply a name for the new collection and save it to the system, either as one of the main collections at the root level of the system, or as a subcollection of one of the larger collections.

Also, it is possible that the results of a query the user wishes to save are too large for a single collection. The user will have the capability in such instances to select elements individually to be saved into subcollections.

6.0 Conclusion

When it is complete, the KBB will be a versatile tool for the verification and browsing of CLIPS rule bases. Some of the features of this hypermedia system will be the automatic creation of links based on the CLIPS rule structure, the ability to query the rules and save the result as a collection, and the ability to browse the rules, either sequentially or by using the links and collections.

Hypertext As a Model for the Representation of Computer Languages

Randal Davis
University of Colorado

Computer systems for operating the Space Station Freedom will include an object-oriented and English-like User Interface Language (UIL). We have proposed a representation of the Space Station UIL that is based on a hypertext model. We discuss the hypertext model of the Space Station UIL and show how this representation may be appropriate for other modern computer languages.

Hypertext As a Model for the Representation of Computer Programs

Experience with the Design of the Space Station Freedom User Interface Language

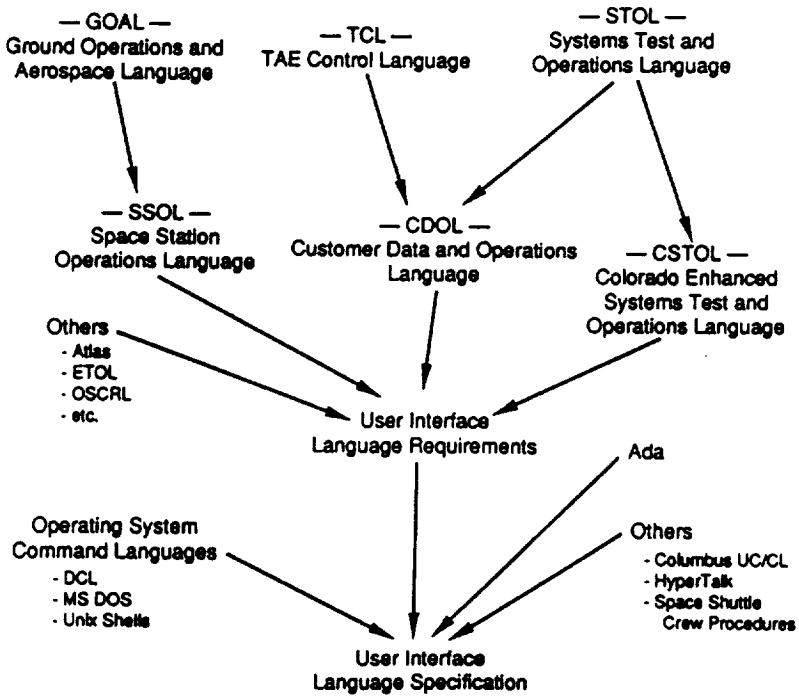
Randal L. Davis
Space Operations and Information Systems Division
Laboratory for Atmospheric and Space Physics
University of Colorado at Boulder

Presented at
Hypermedia '90
Aerospace Applications and Research Directions
Houston, Texas
5 December 1990

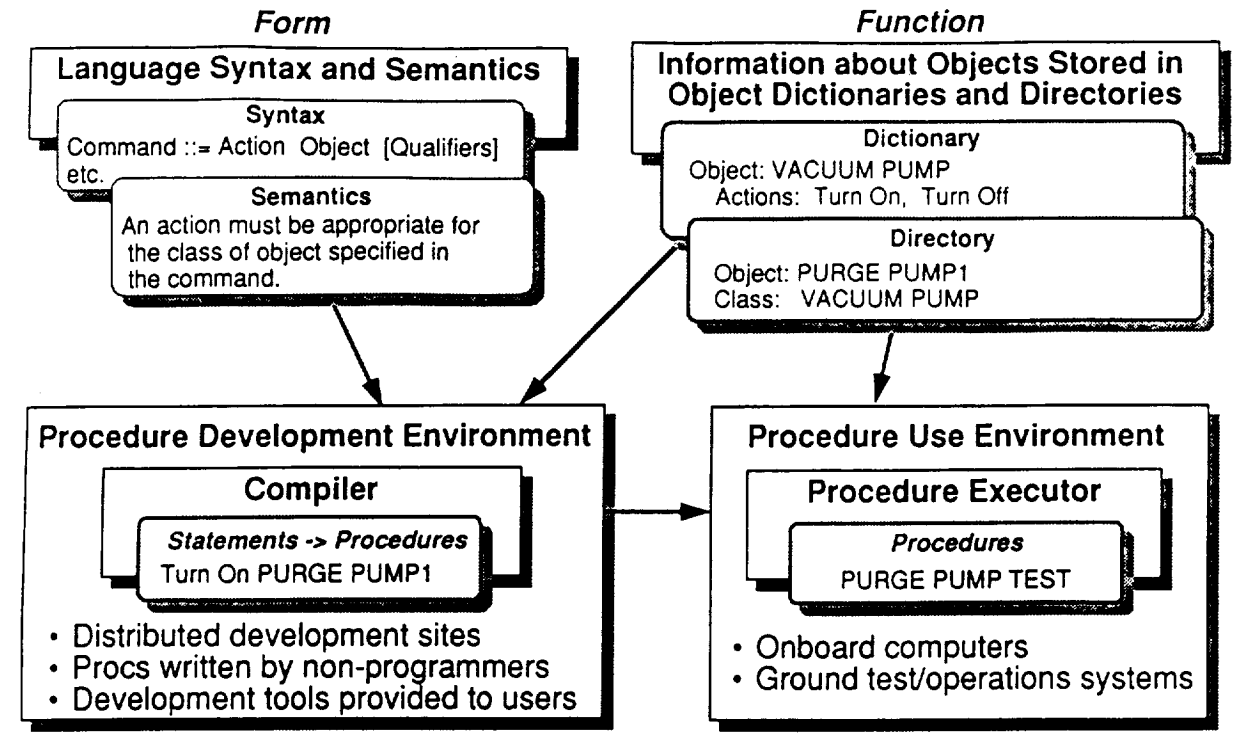
Introduction

- The Space Station Freedom User Interface Language (SSF UIL) is designed for use by the astronauts, ground controllers, scientific investigators, and hardware/software engineers who will test and operate the systems and payloads aboard the space station
  - Object-oriented
  - English-like
  - Will supplement the graphical user interface to systems and payloads by providing command line entry
  - Will be used to write test and operations procedures
- The SSF UIL design was influenced by the availability of new technologies, including hypertext
- We have found at least three places where hypertext is appropriate for use within the systems that will be used to create and run SSF UIL procedures

The SSF UIL Family Tree



Overview of SSF UIL Design and Usage



Hypertext Application #1: Linking Code and Annotation

- **Provide a link between units of code — statements, steps and procedures, and so on — and associated annotation and documentation**
  - **History of procedure's development [Development, Use]**
  - **Description of syntax and semantics [Development, Interactive Use]**
  - **Comments for programmers [Development]**
  - **Help for end users [Use]**
- **Why Hypertext?**
  - **Freeform comments aren't good for capturing specific information like program history**
  - **Intertwining code and commentary often makes a procedure more difficult for a skillful procedure developer to read**
  - **Comments don't survive parsing, so they aren't available to users who only have the object or executable representation of a procedure**
  - **Comments are formed from the character set used for the computer language, but it would often be desirable to allow graphics and other non-text information in comments**

Sample Procedure With Traditional Freeform Commentary

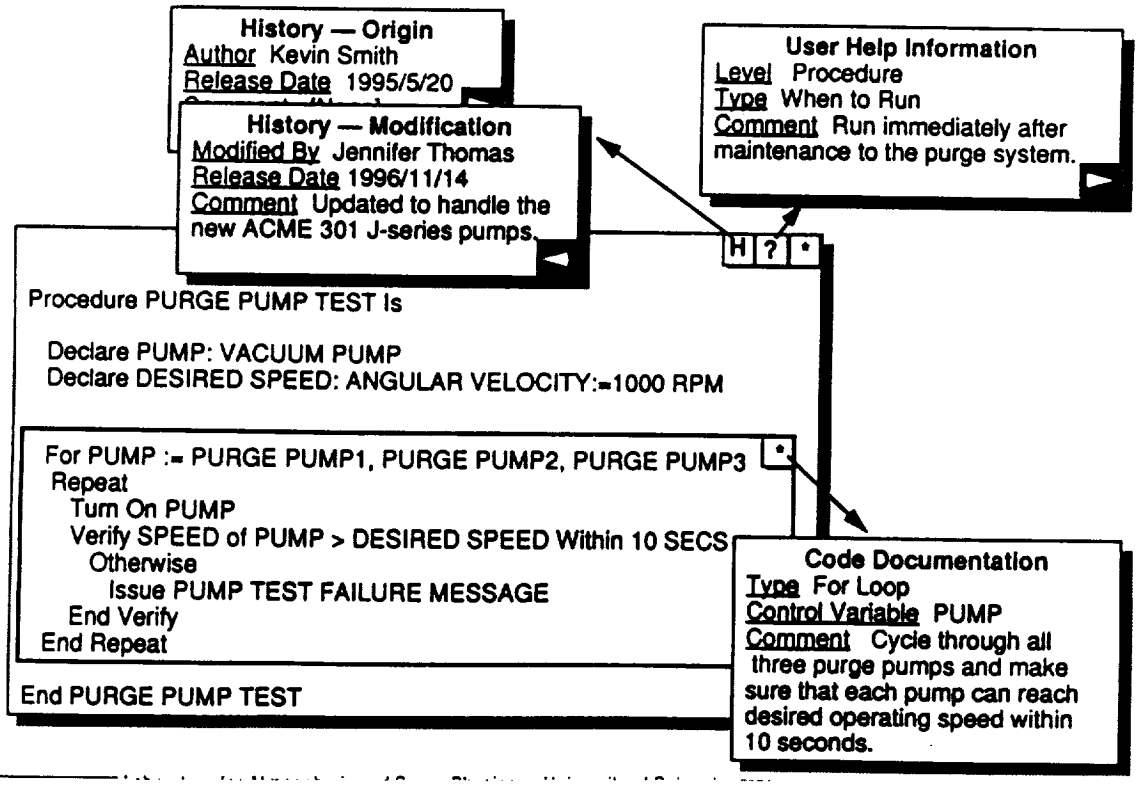
```
/*****************************************************************
Procedure PURGE PUMP TEST - This procedure performs a full
checkout of the ECLSS purge pumps. Run immediately after
maintenance to the purge system.

Written by Kevin Smith, 1995/5/20
Modified by Jennifer Thomas, 1996/11/14: Updated to handle
the new ACME 301 J-series pumps.
*****************************************************************/
Procedure PURGE PUMP TEST Is

  Declare PUMP: VACUUM PUMP /* Current pump under test */
  Declare DESIRED SPEED: ANGULAR VELOCITY:=1000 RPM

  /* Cycle through all three purge pumps and make sure that each
     pump can reach desired operating speed within 10 seconds */

  For PUMP := PURGE PUMP1, PURGE PUMP2, PURGE PUMP3
    Repeat
      Turn On PUMP
      Verify SPEED of PUMP > DESIRED SPEED Within 10 SECS
        Otherwise
          Issue PUMP TEST FAILURE MESSAGE
      End Verify
    End Repeat

End PURGE PUMP TEST
```

Code Documentation and User Instruction

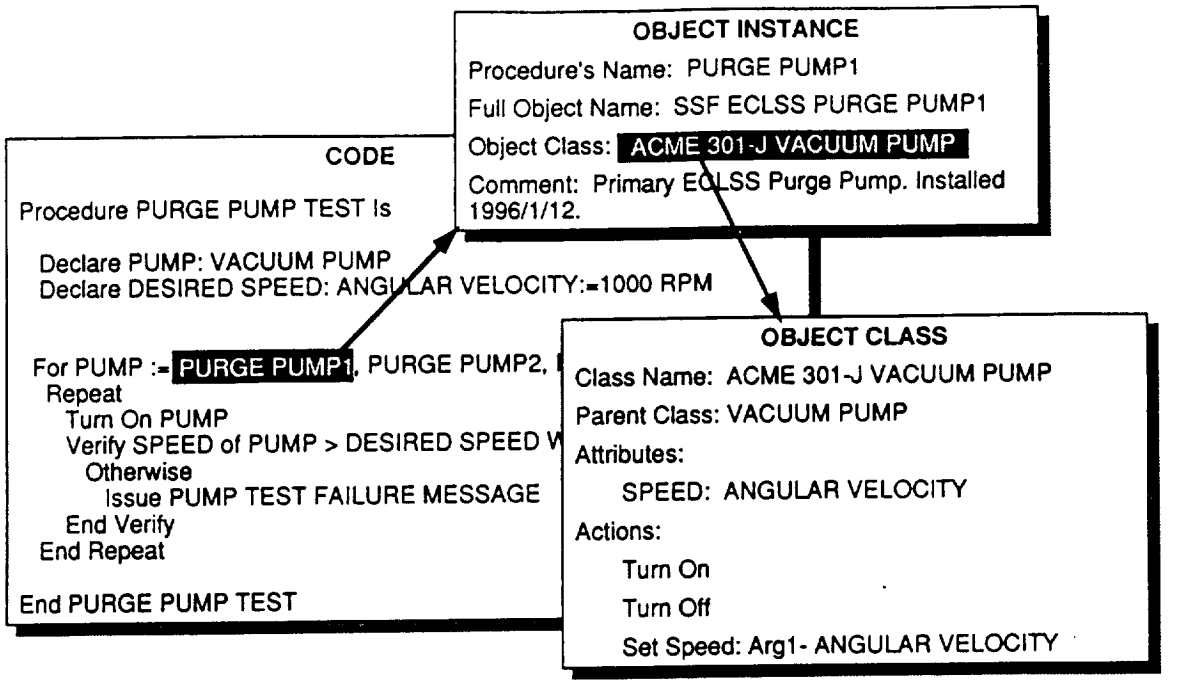
History

Code Documentation

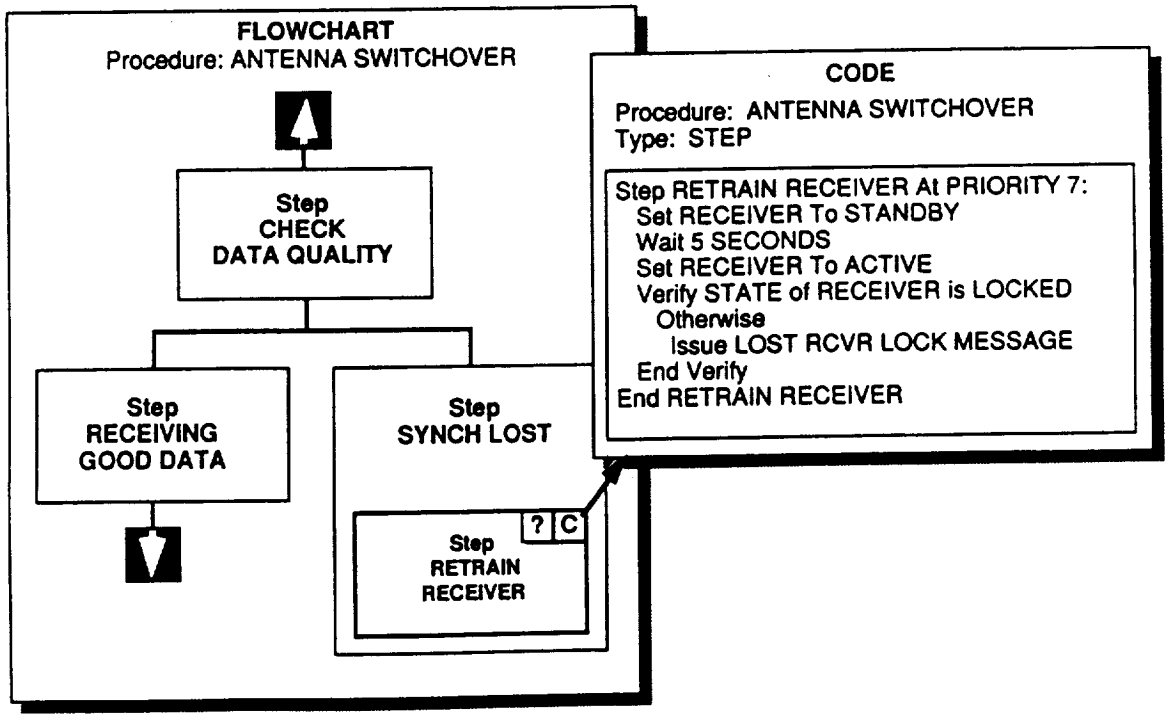
Procedure With Hypertext Annotation



Hypertext Application #2: Linking Code to Object Information



Hypertext Application #3: Linking Steps Within a Procedure



Conclusions

- Hypertext is appropriate and advantageous for the three uses we have examined:
  - Linking procedure code and annotation
  - Linking code to object instance and class information
  - Linking steps within a procedure
- The SSF UIL's object-oriented nature lends itself to representation through hypertext
- While the SSF UIL was specified from the outset with hypertext in mind, it has become clear that a traditional text-only representation for procedure code and annotation is desirable to promote portability
  - An annex is being added to the SSF UIL Specification to provide this
- Hypertext-based organization will be appropriate for other modern languages, particularly if they are designed from the outset to take advantage of new technology like workstations and personal computers

References

- **SSF UIL Documentation**
  - Space Station Freedom Program: *"User Interface Language Specification"*, Document No. USE 1001, Version 2.1, March 1990.
- **Use of Hypertext for Program Documentation**
  - Bigelow, J.: "Hypertext and Case", *IEEE Software*, March 1988, pp 23-27.
  - Wolfram, S.: *"Mathematica — A System for Doing Mathematics By Computer"*, Addison Wesley, 1988.
- **Concepts for Crew Procedures, Old and New**
  - Johns, G. L.: *"Flight Data FIle for the Space Station"*, 2 Volumes, MITRE Corporation, Document No. MTR10019, February 1987.
  - Johns, G. L.: *"Dynamic Display of Crew Procedures for Space Station"*, 2 Volumes, MITRE Corporation, Document No. MTR 88D0033, August 1988.
  - Johnson Space Center: *"Space Shuttle Flight Data File Preparation Standards"*, Document No. JSC-09958, May 1984.
  - Kelly, C. M.: *"Conceptual Definition for a Flight Data File Automated Control and Tracking System"*, MITRE Corporation, Document No. MTR-88D0017, July 1988.

Acknowledgement

This work was supported by NASA Goddard Space Flight Center under contract NAS5-29174. Design of the SSF User Interface Language was coordinated by the Space Station Freedom User Support Environment Working Group (USEWG), and many USEWG members have contributed to the language's development.

Session 6

Hypertext and Object Management

Chair: Bryan Fugate

AI GERM: A Logic Programming Front End for GERM

Safaa H. Hashim

HEAVENS System for Software Artifacts

Paul Matthews