# Driving Knowledge Acquisition Via Expert Scripted Visualizations

**John Domingue**
Knowledge Media Institute,
The Open University, Walton Hall,
Milton Keynes, MK7 6AA, UK.
J.B.Domingue@open.ac.uk

**Abstract**

Software visualization is the use of filmcraft, cartoon animation, and graphic design techniques to display data structures, programs, and algorithms. It has successfully been used in a number of settings to aid expert and novice programmers debug their programs and to teach computer science students about algorithms. In this paper we describe a framework for using software visualization technology to drive the knowledge acquisition process.

## Introduction

This paper describes a framework, currently under development, for using Software Visualization (SV) technology to drive the knowledge acquisition process. Software visualization (Price et al. 1993) is the use of filmcraft, cartoon animation and graphic design techniques to display data structures, programs, and algorithms. It has successfully been used to validate Knowledge Based Systems (KBS) (Domingue 1995), aid expert (Eisenstadt & Brayshaw 1988) and novice programmers (Lieberman & Fry 1995) (Mulholland 1995), and to teach computer science students about algorithms (Brown 1988).

The framework proposed here is based around an Expert Scripted Visualization (ESV) which experts can create in a domain dependent visualization scripting environment. The scripting environment allows experts to set up and solve test cases by direct manipulation. The manipulations by the expert within the environment are stored as an ESV which can be replayed when required. The Knowledge Engineer (KE) uses the ESV as a focus for constructing a model within our Operational Conceptual Modelling Language (OCML) (Motta 1995). OCML contains constructs which allow a problem solving model to drive the domain visualization set up by the expert. Differences between the expert and model driven visualizations form the basis of future knowledge elicitation sessions.

Given the limited space in the rest of this paper I shall merely illustrate the framework using an imaginary scenario based on Sisyphus I (Linster 1992), a room allocation problem.

## A Scenario

The Sisyphus I problem is to assign a set of researchers from the YQT laboratory to a set of rooms within a new building. Various constraints are placed on the allocation of rooms. The constraints are:
i) the head of the group should be in a central office,
ii) the secretaries should be close to the head of the group,
iii) the project managers should have maximum access to the secretaries and the head of the group,
iv) smokers and non-smokers should not share an office,
v) as far as possible researchers sharing an office should be working on different projects.

In our approach the initial task for the expert is to define the concepts, properties and their associated graphical representations using a form based interface. The two tables below show part of the definition of the YQT -member concept.

A tool for creating scripts, a scripting environment, is generated by selecting a number of concepts. The generated tool has a well containing the iconic representations of the selected concepts, a scripting area and a menu bar. A scripting environment for the concepts YQT-member and room is shown in figure 1 above.

The expert can use the well on the left of figure 1 to create new instances of rooms and YQT-members. The attributes for each instance can be filled in using a menu derived from the concept definitions (part of the definition of the YQT-member concept was given in table 1). A menu for the YQT-member instance 'Angi' is shown in figure 2 below.

Once a set of icons has been created the expert then sets up a test problem which s/he solves by altering an icon's location, size, colour, type, or property values. The manipulations are recorded as an ESV, using a description language derived from our SV framework and tool Viz [Domingue et al. 1992]. Figure 3 below shows figure 1 after an expert has 'solved' the trivial allocation problem.

| Concept YQT-Member | | |
|---|---|---|
| *Property* | *Type* | *Graphical Representation* |
| Name | String | Label |
| Role | secretary, researcher, head of group, manager | Icon |
| Smoker | boolean | Icon |
| Hacker | boolean | Icon |
| Project | respect, kriton, tutor2000, babylon, eulisp, mlt, asserti, autonomous_system | Colour |

**Table 1. The definition of the YQT-Member concept.**

| Researcher Icon Table | *Non-Smoker* | *Smoker* |
|---|---|---|
| *Non-Hacker* |  |  |
| *Hacker* |  |  |

**Table 2. The mappings between the attributes smoker and hacker and their iconic representation for the researcher role.**

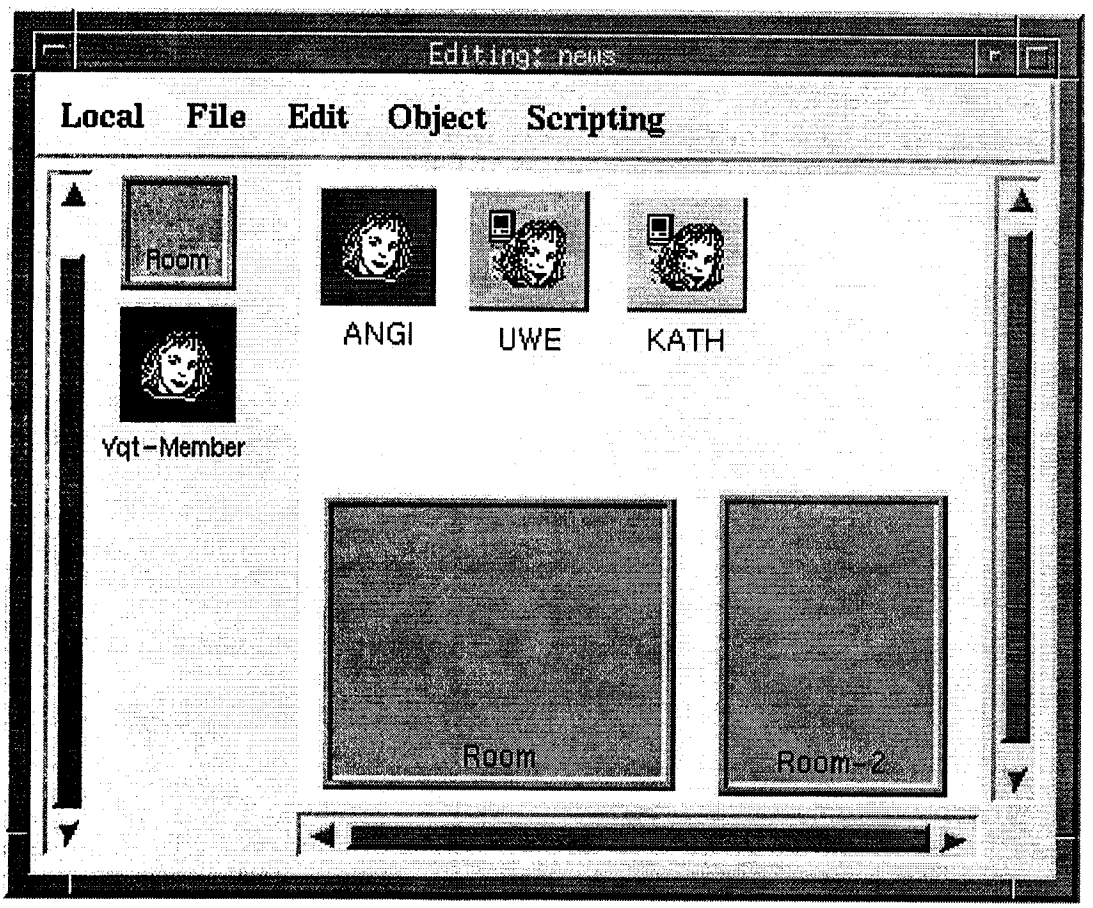


**Figure 1. The initial Sisyphus I scripting environment.**

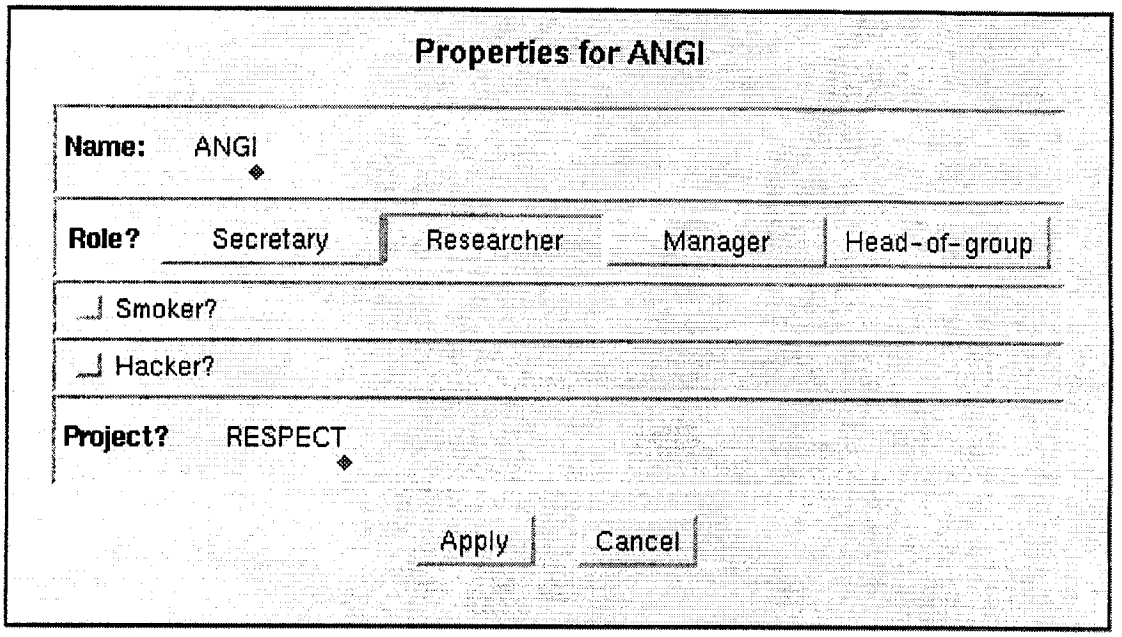


**Figure 2. A property menu for the YQT-member Angi.**

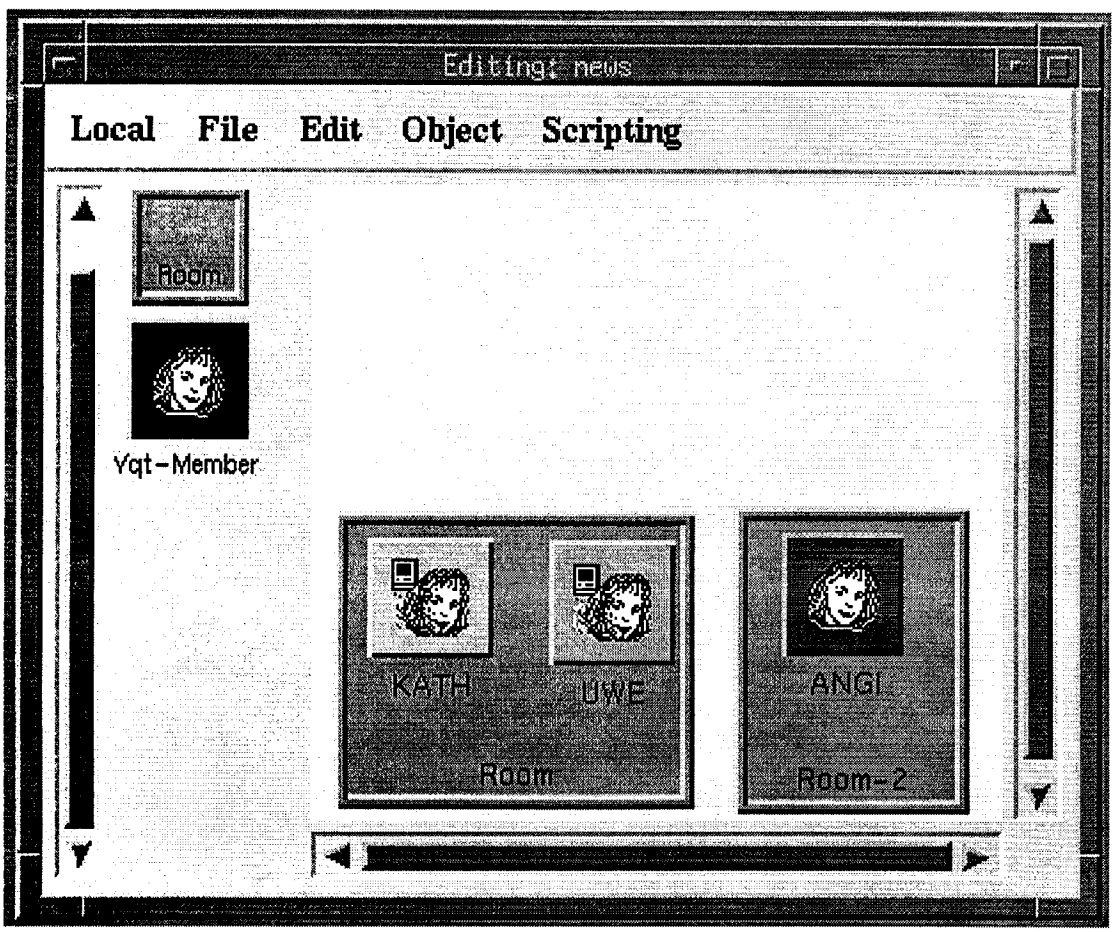


**Figure 3. The end state of an ESV for the three researchers.**

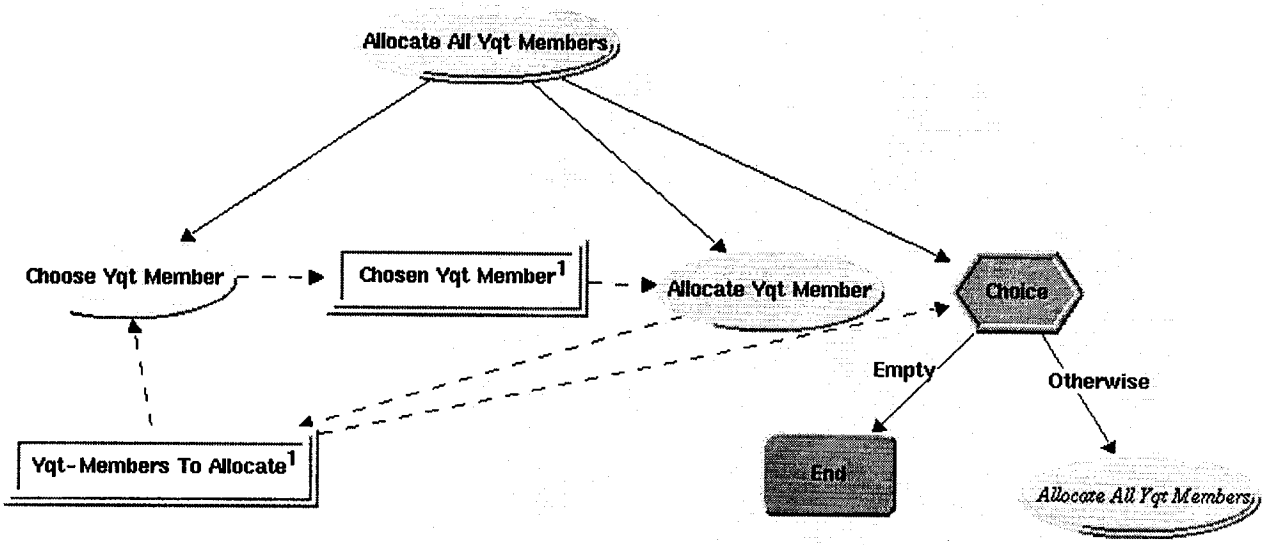


**Figure 4. An initial OCML model to emulate the first ESV.**

The next step in the acquisition process is that the KE attempts to replicate the ESV using OCML. An initial OCML model, derived from the ESV depicted in figures 1 and 3, is shown in figure 4 above. Within the OCML model above tasks are represented by ovals and rôles by rectangles (note that I distinguish between OCML rôles and Sisyphus roles with a circumflex). Tasks specify knowledge level activities. A rôle represents some segment of task input or output data (defined in a concept/instance or relation module). The cardinality of a rôle is indicated by a numeral in the top right corner of the rectangle. The solid lines in the diagram are control flow links and the dashed lines are data flow links. Duplications or aliases (two nodes representing the same task or role) are indicated by the use of an italic font. OCML models can also contain control constructs, two of which are present in figure 4. The hexagonal node is a choice control construct. Executing a choice construct passes control to one of the subtasks depending on the value of the input role. The rounded rectangle labelled 'End' is an end construct. The execution of a end construct passes control back to the root task (shown as a double oval).

The model in figure 4 specifies that to allocate all the members of YQT one first chooses a particular member who is then allocated. This continues until no YQT members are left. Terminal nodes contain simple rules describing, for example, how to allocate a single person to a room. The rules can contain expressions which drive a visualization, for example, the task allocate YQT member contains the expression (move-into ?chosen-yqt-member ?allocated-room). Once a model has been compiled it can be executed. As execution proceeds the currently executing tasks are highlighted in red and the domain visualization is animated.

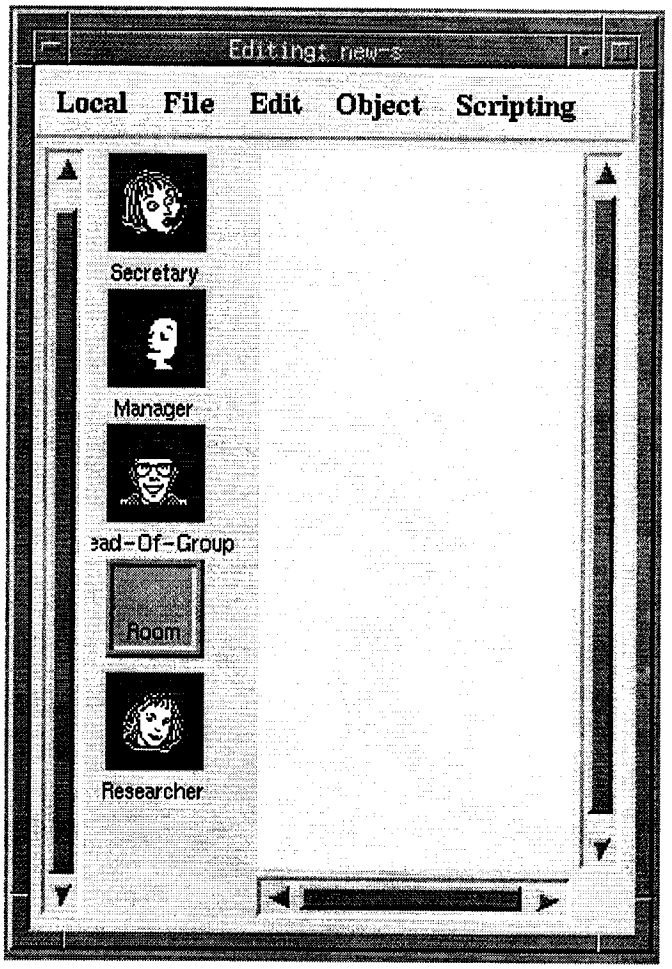


**Figure 5. A scripting environment associating roles within the YQT lab with concepts.**

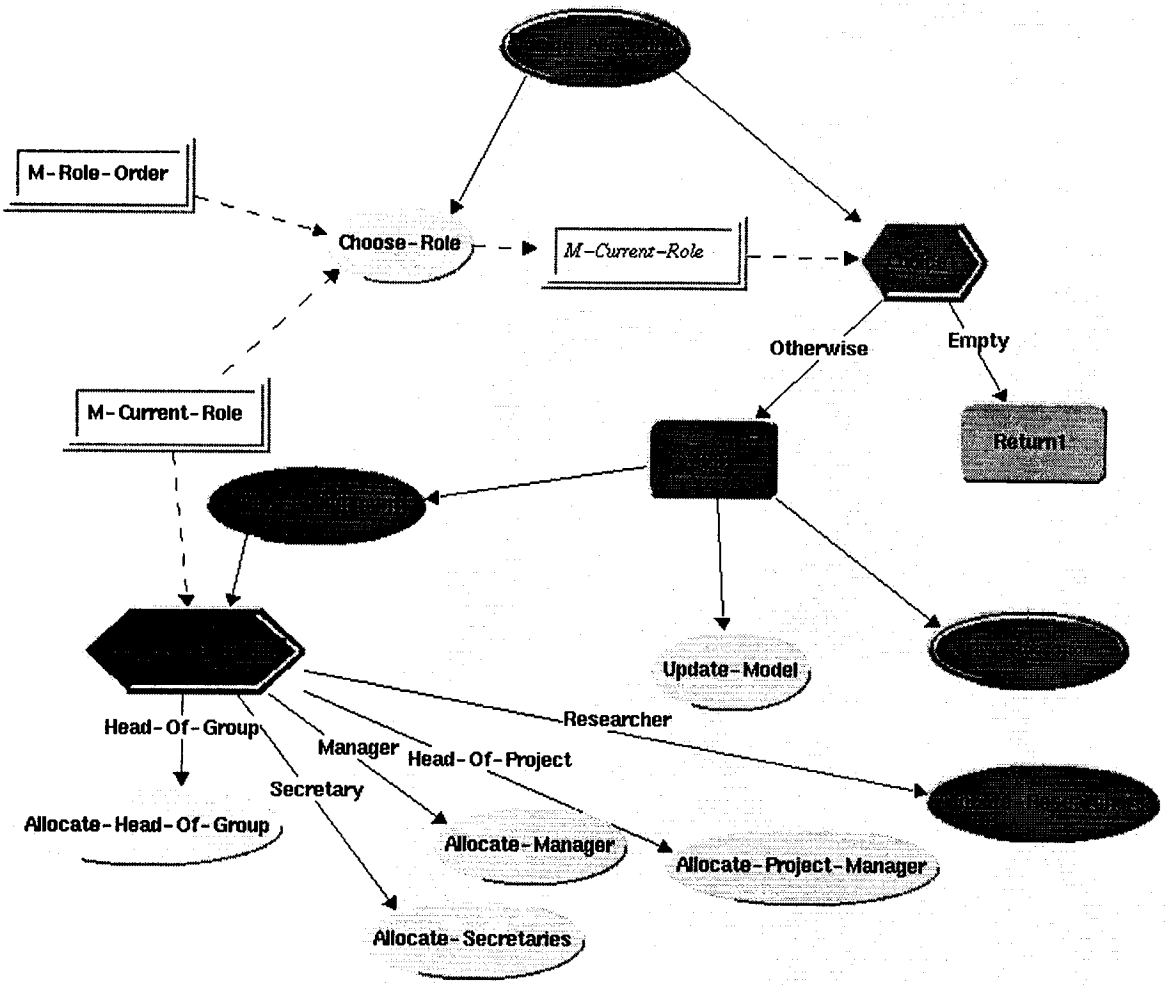


**Figure 6. Showing the execution of the final OCML control and data flow model for the Sisyphus I problem. The tasks design-allocation, select-assignments and allocate-researchers are currently being executed.**

The KE can validate the model by comparing the Model Driven Visualization (MDV) with the ESV, any discrepancies forming the focus for a future knowledge elicitation session. It should be noted that the OCML model visualization (highlighting the tasks in red) and MDV are synchronous allowing faults within the MDV can easily be tracked back to the offending task.

As the knowledge acquisition process continues the scripting environment may change. For example the KE may decide that role is an important criteria for ordering the allocation YQT-members, and therefore is best modelled as a set of concepts rather than a concept property. This decision would lead to a slightly altered scripting environment shown in figure 5.

The iterative process of extending the ESVs and OCML model continues until the expert and KE are satisfied that the OCML model has full coverage. The final model and full ESV for the Sisyphus I problem are shown in figures 6 and 7.

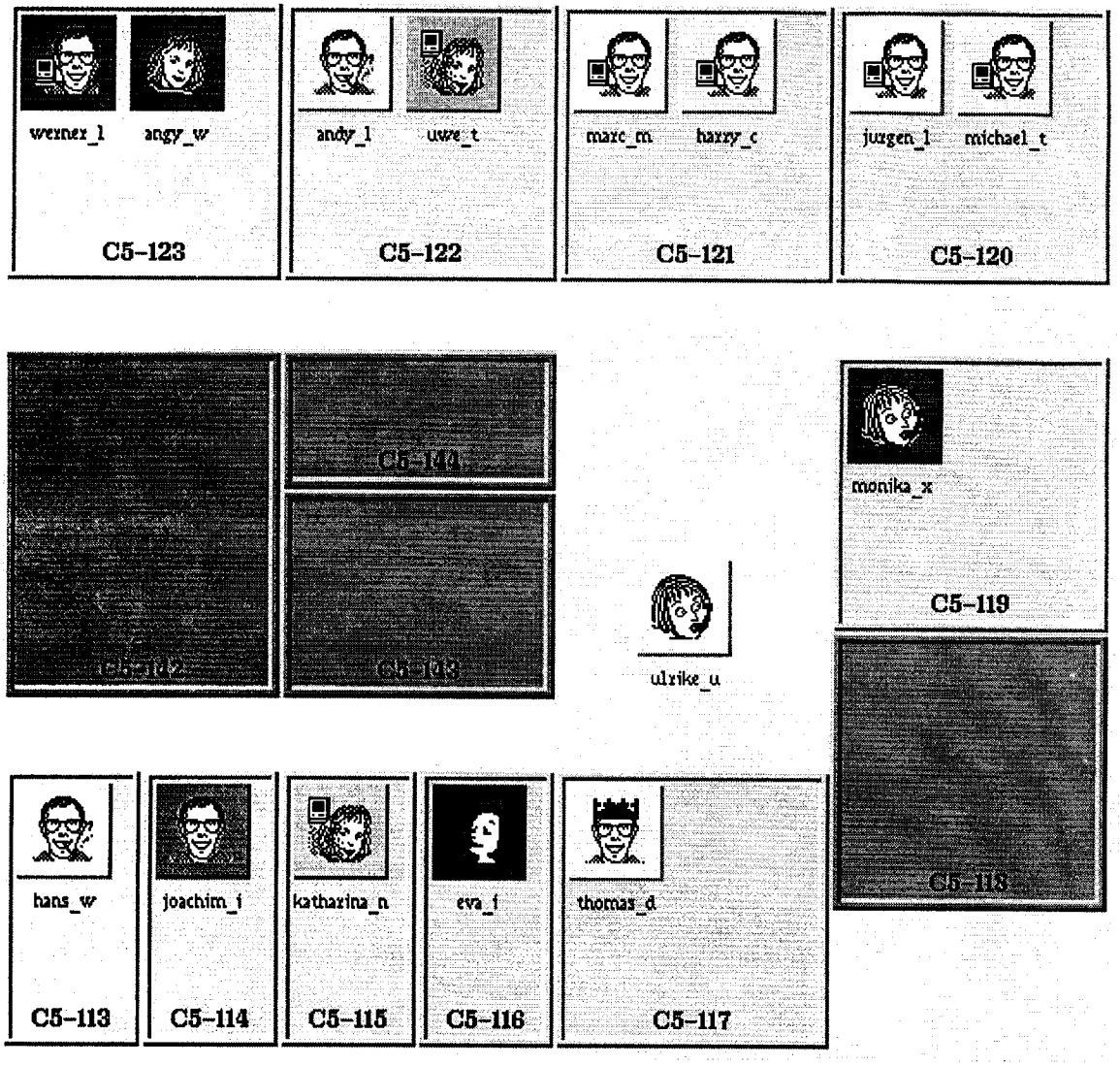


**Figure 7. The penultimate state of a ESV solving the full Sisyphus I problem.**

## Conclusions

Within this paper I have briefly described how we can use SV technology to drive the process of knowledge acquisition. This work, however, is still in its early stages and there remain questions of how general and scalable the methodology is. I hope to answer these by using this approach to solve larger problems (e.g. Sisyphus III (Shadbolt 1995)).

Future work will seek to integrate the framework with current research on KBS reuse. Specifically, investigating how visualization views (e.g. tree based views, table based views (see (Domingue 1992)) can be integrated into libraries of problem solving methods, and how domain dependent icons can be integrated into domain ontologies.

## References

Brown, M.H. 1988. *Algorithm Animation.* ACM Distinguished Dissertations, MIT Press, New York.

Domingue, J., Price, B. and Eisenstadt, M. 1992. Viz: a framework for describing and implementing software visualization systems. In Gilmore, D. and Winder, R. eds. User-Centred Requirements for Software Engineering Environments, Springer-Verlag.

Domingue, J. 1995. Using Software Visualization Technology in the Validation of Knowledge Based Systems. In Proceedings of Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff Canada.

Eisenstadt, M. and Brayshaw, M. 1988. The Transparent Prolog Machine (TPM): an execution model and graphical debugger for logic programming. *Journal of Logic Programming*, 5(4):277-342.

Lieberman, H. and Fry, C. 1995. Bridging the Gulf Between Code and Behaviour in Programming. In Proceedings of ACM SIGCHI Conference 1995.

Linster M. ed. 1992. Sisyphus '92: Models of problem solving. GMD report 630, GMD Sankt Augustin.

Motta, E. 1996. The Operational Conceptual Modelling Language Manual. Knowledge Media Institute Technical Report 22. The Open University, Walton Hall, Milton Keynes, MK7 6AA, England. Forthcoming.

Mulholland, P. 1995. Evaluating Software Visualization Systems: A case study in Prolog. Ph.D. diss., The Knowledge Media Institute, The Open University, Walton Hall, Milton Keynes, MK7 6AA, England.

Price, B. A., Baecker, R. M. and Small, I.S. 1993. A Principled Taxonomy of Software Visualization. *Journal of Visual Languages and Computing* 4 (3):211-266.

Shadbolt, N. 1995. The Sisyphus III proposal. Http://www.psych.nott.ac.uk/aigr/research/ka/SisIII/home-page.html