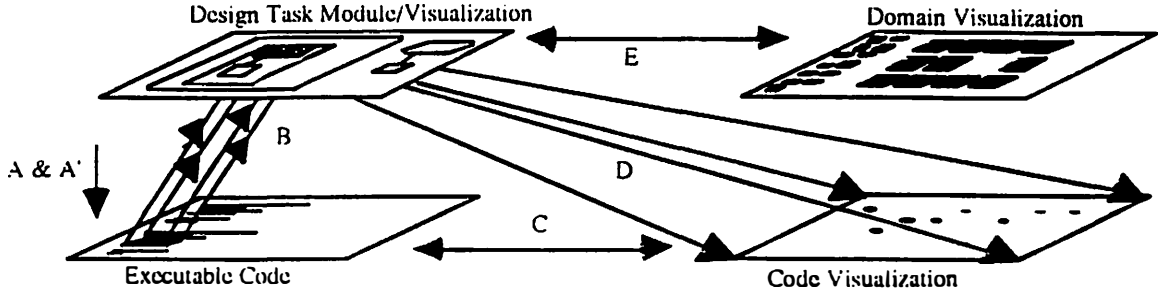
# Validating Knowledge Based Systems with Software Visualization Technology

John Domingue

Human Cognition Research Lab
The Open University
Milton Keynes, UK

The validation of a Knowledge Based System (KBS) involves comparisons between an external reference model and a system's component parts. In this paper I will describe how such comparisons can be aided by the application of Software Visualization (SV) technology. Software visualization is the use of filmcraft, cartoon animation and graphic design techniques to display data structures, programs, and algorithms. The described approach eases the task of mapping between the comparates by the use of dynamic code, design, and domain oriented visualizations of KBS execution.

The interactions between the visualizations are summarised in the figure below:




where the labels in figure above denote the following:

- *A* the transformation, carried out by the design compiler, of the design model into executable code,
- *A'* the ability to view or edit the source code representation for a design model component,
- *B* the ability to map back from segments of code to the associated design model component,
- *C* the bi-directional mapping between the static code representation and the code visualizations,
- *D* the ability to obtain a code level visualization for a particular execution of a design model fragment,
- *E* the synchronisation of the design task and domain visualizations.

Within the knowledge acquisition field, validation is of importance because a KBS cannot be totally formally specified. Current approaches provide static knowledge level visualizations with little or no link to code level execution. This places the knowledge engineer in the arduous situation of having to synthesise knowledge and code level connections whilst burdened with the difficulties of the validation task. The approach outlined in this paper alleviates this by integrating SV techniques into the design and implementation tools and by providing static and dynamic:

- domain views which easily map onto the domain ontology,
- design views showing how components of the design interacted, and
- code views showing the execution at the code level.