Run Time Reusability in Object-Oriented Schematic Capture

David Parsons, Tom Kazmierski

Southampton Institute, U.K.
{dave.parsons, tjk }@solent.ac.uk

This poster presents some important aspects of the architecture and functionality of an object-oriented schematic capture system for electronic circuit design and simulation. In particular it introduces the terms 'virtual polymorphism' and 'visual polymorphism' to describe techniques that provide run-time extensibility and flexible code generation within a visual environment.

Schematic capture systems convert a graphical representation of an electronic circuit into some other form for simulation or synthesis. This particular system generates code in VHSIC Hardware Description Language - Analogue and Mixed Signal (VHDL-AMS). This language, standardised by the IEEE in 1997, allows for hardware descriptions that include both digital and analogue components.

An important aspect of VHDL-AMS is that it allows for new types of component to be described using behavioural definitions rather than simply building larger aggregations out of sub-components that already exist in libraries. To enable a user to define new component types via the graphical interface, the system is built using an approach called 'virtual polymorphism'. This term is used to describe a situation where application level objects of different types (in this case electronic components) appear to have polymorphic behaviours even where they are represented by objects of the same class. This is achieved via a reflective architecture that allows component objects to be configured at run time by meta-data, enabling them to invoke various dynamically bound aggregations to provide their behaviour. For example, different component objects use objects of other classes to draw themselves using standard symbol sets and to generate code. By basing the system on this architecture, rather than using a traditional classification hierarchy of component classes, run time extensibility is provided by routines that dynamically add to the meta-data.

The second important aspect described by the poster is termed 'visual polymorphism'. This concept is based on a particular characteristic of VHDL-AMS code generation for mixed mode (digital and analogue) circuits, where we find that a single type of component may be represented by one of a number of different code models depending on the nature of its connectivity to other elements of a circuit. Visual polymorphism describes how a single visual image of a component encapsulates the automatic selection of the appropriate code model. A single gate component, for example, is able to select the appropriate models from possibilities that include digital, analogue or mixed mode input. It does this by giving digital component objects the ability to interrogate their external connections to find out whether they are joined to terminal nodes (analogue objects) or signal nodes (digital objects). From this information, each component is able to invoke a model with an appropriate type signature via its code generating objects.

This use of both virtual and visual polymorphism demonstrates that we can apply polymorphism as a conceptual approach, allowing objects to behave differently in different contexts, without necessarily using traditional implementation mechanisms. Systems can thus be made more flexible and easily extensible.