Visualisation of Object Oriented Program Execution

John G. Hosking
Department of Computer Science University of Auckland
Private Bag 92019, Auckland, New Zealand
john@cs.auckland.ac.nz

Abstract

An environment for visualising object oriented program execution providing concrete and abstract views of program structure and behaviour is described. Of particular novelty are dynamic views using a road map metaphor.

Introduction

In prior work, we have developed SPE, an environment for visual and textual object-oriented programming [1] and Cerno, a companion system for visualising execution of programs built using SPE [1,2]. Here we describe an extension to Cerno providing abstract dynamic views based on a road map metaphor. The paper begins by reviewing Cerno's existing capabilities. The new map views are then described, followed by brief sections on implementation, related work, and current work.

Cerno

Fig. 1 shows Cerno visualising the execution of an OO program. Cerno can provide visualisations of:

- Object structure: icons represent objects and arrows represent inter-object references. Icon structure is under user control via an icon display specification language (DSL).
- Abstract structure: icons can represent aggregations of objects, such as all objects in a linked list.
- Dynamic behaviour, via timing diagrams, showing a history of method calls, and per-object call stacks associated with object icons in object structure diagrams.

While these visualisations provide much information, there is a need for more abstract dynamic representations to complement the abstract structural views. In the existing dynamic views it is easy to become lost in detailed calling patterns and miss an overview of the dynamic behaviour. Map metaphor views, motivated by a VL'95 keynote address [3], provide a way to correct this deficiency.

Map metaphor views

Fig 2 shows two "map" views of program execution state. These views include the following basic components:

- "Cities" represent "collections" of objects. City size and colour map to the number of objects in the collection. and the frequency of calls between objects in the aggregate.
- "Roads", with "lanes" in either direction, mapping lane thickness and colour to the frequency of calls from objects in one aggregate to those in another.

Using these views aspects of aggregate modularity such as the strength of coupling with other aggregates can be quickly assessed. Using multiple views, with differing aggregation structure, visualisations at various levels of abstraction and focus of attention can be constructed.

A key issue is specifying which objects belong to an aggregate, as this determines what abstractions can be represented. In the prototype described here, a tool allows users to interactively include objects or aggregates in another aggregate. The size and colour of and connections to the aggregate then change to reflect the increased number of objects, and inter- and intra- aggregate calls.

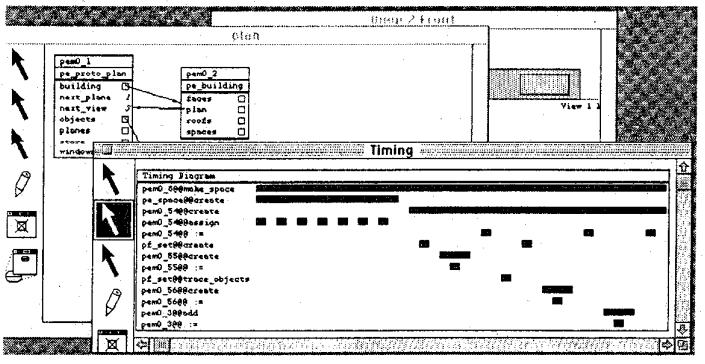

Figure 1: Cerno structure and timing diagram views

0-8186-7469-5/96 $05.00 © 1996 IEEE

For realistic use, a better way of specifying aggregate contents is required. A query language for this is being implemented. This permits users to specify, for example, that all objects in a given list are to be included. Such specifications can give rise to object aliasing, as one object can appear in several aggregates in a view. This is handled by icon annotation. A further important issue is naming of aggregates. In the prototype the default name is a list of object ids of objects in the aggregates. Users can optionally provide a more meaningful name for the aggregate.

We have found it useful to use non-linear (log) mappings to relate city and lane size/colour to aggregate number and call frequencies. This allows differentiation between small call frequencies, while covering an appropriate dynamic range for "busy"objects.

Class level abstraction

As described, map views are an object level abstraction of dynamics. They can, however, visualise class level dynamics by specifying aggregates consisting of all objects belonging to a class. Icon colour then indicates the call frequency between objects of a class while size indicates the number of objects created of that class. Further abstraction is obtained by constructing aggregates containing objects of multiple classes, such as a class and its subclasses, or all classes in a module. These views provide an execution time analogy to analysis/design level class/module interaction diagrams.

Implementation

Cerno is implemented in and provides visualisations of programs written in Snart, an object-oriented Prolog [1,2,4]. Snart provides facilities for monitoring the execution state of a Snart program. To adapt Cerno for use with other languages would require similar runtime system access, or a preprocessor to annotate the source code.

Related work

Much work has taken place in the field of algorithm animation, such as that of Stasko on Tango [5] and its derivatives, and more recent work such as Trip [6]. However, Cerno is aimed primarily at visual debugging, with the ability to construct and present abstract visualisations dynamically as a way of exploring the execution behaviour of a program. In this regard it is closer to the work of De Pauw et al [7]. This too uses spatial metaphors, such as in the inter-class call cluster view, but uses proximity to indicate interaction rather than the "road" approach presented here. Their methods are thus complementary to those presented here.

Current work

Current work aims at completing implementation of the aggregate query language. Longer term aims are to translate Cerno into a C++ implementation as part of an environment under development for C++ programming.

Acknowledgements

The author acknowledges the financial assistance of the Auckland University Research Committee.

References

[1] Grundy, J, Hosking, J, Fenwick, S, and Mugridge, W, Connecting the pieces, in *Visual Object-Oriented Programming*, Burnett, M, Golberg, A, Lewis, T (eds), Manning Publications, Greenwich, 229-254,1995.

[2] Fenwick, S., Hosking, J., and Mugridge, W.: A visualization system for object-oriented programs, *Proc TOOLS 15*, Prentice Hall, Sydney, 93-103, 1994.

[3] Kuhn, W, Pictures that show us the Way: Geographic Information Systems and Visual Languages, *Keynote address to VL'95*, Darmstadt, 1995.

[4] Hosking, J, Mugridge, W, and Blackmore, S, Objects and constraints: a constraint based approach to plan drawing, *Proc TOOLS 15*, Prentice Hall, Sydney, 9-19, 1994.

[5] Stasko, J., Tango: a framework and system for algorithm animation, *IEEE Computer*, 23(9) 27-39, 1990.

[6] Takahashi, S., Miyashita, K., Matsuoka, S., Yonezawa, A., A framework for constructing animations via declarative mapping rules", *Proc VL'94*, 314-322, 1994.

[7] De Pauw, W, Helm, R, Kimelman, D, Vlissides, J, Visualizing the behaviour of object-oriented systems, *Proc OOPSLA 93*, 326-337, 1993.

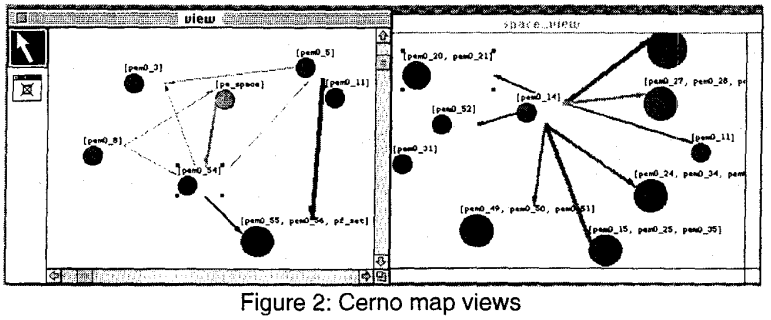

Figure 2: Cerno map views