X3D-UML: 3D UML State Machine Diagrams

Paul McIntosh, Margaret Hamilton, and Ron van Schyndel
RMIT University, GPO Box 2476V, Melbourne, VIC 3000, Australia
paul.mcintosh@internetscooter.com,
{margaret.hamilton,ron.vanschyndel}@rmit.edu.au

Abstract. X3D-UML utilises X3D (eXtensible 3D) to enable standards-based advanced 3D UML visualisations. Using X3D-UML, 3D UML State Machine Diagrams have been evaluated against actual user tasks and data, using the Sequential Evaluation methodology. The results of User Task Analysis, Heuristic Evaluation and Formative Evaluation phases provide clear evidence that the use of UML extended with 3D is a practical solution for visualising complex system behaviour. RoseRT model metrics show between 56%-90% of state machine diagram work would benefit from such 3D UML extensions; hence the 3D improvement can deliver considerable benefit to organisations.

Keywords: X3D-UML, 3D UML, X3D, 3D Software Visualization, VRML.

1 Introduction

Computers in current day software development houses are now capable of rendering quality real time 3D visualisations. In the past, 3D representations for UML have been suggested [1-4] and some research has been undertaken on specific aspects of 3D UML and related visualisations [5, 6]. Despite this, there has been no definitive research into the overall benefit of completing common and critical software engineering tasks with UML extended using 3D visualisations. If measurable benefit in 3D UML abstraction can be obtained for one task of note, then this suggests an area of potential lies untapped and idle in the graphics cards of engineers’ computers.

Our research utilises the concept of X3D-UML, the presentation of UML in the Web3D ISO standard X3D (eXtensible 3D) [7], to test for benefit in the area of 3D UML state machine diagrams (also known as statechart diagrams). X3D-UML provides a standards-based approach to software visualisation, where the UML standard is presented through the X3D standard. This approach imposes constraints on the visualisation possibilities but those constraints enable the research results to be more broadly applicable to industry. X3D-UML is described further in section 2.

The reason that UML is chosen as the basis for 3D software visualisation research is that it is a current and accepted standard visual notation for software systems. Basing 3D software visualisation on an existing visual notation enables new visualisations to make use of the existing visual data such as layout, colour and other visual cues, which are already familiar to software engineers. Although in current implementations it is commonly thought of as a 2D notation only, it has been noted by Booch et al. that “the UML allows you to create three-dimensional diagrams.”[8]
The reason that X3D, the successor to VRML (Virtual Reality Modelling Language), is chosen as the visualisation medium is that it is able to capture complete virtual environments containing 3D, 2D, animation, spatialised audio and video, user interaction, user defined objects, scripting and physical simulations[7]. However, this rich set of features only represents the visualisation possibilities, more importantly X3D is capable of rendering the types of visualisation currently found in UML, being large quantities of lines and text [9], and this capability provides the critical link between current UML and advanced UML visualisation.

For this research we have used IBM Rational Rose RealTime [10] and referred to here by its common name RoseRT. RoseRT makes extensive use of state machine diagrams for describing system behaviour and these descriptions generate code (i.e. “the model is the code”). The advantage for researching state machine diagram visualisations is that software engineers work extensively with them and must define them precisely to get the system to behave as required. Also implemented is the concept of separate substate diagrams, which is recommended practice for large hierarchies [11]. The screenshot below (Fig 1) shows an example of the current 2D RoseRT state machine diagram implementation, with substates appearing on different tabs and also accessible through clicking on superstates.

Substate levels do not provide a complete state machine view but they do aid the engineer in managing the layout of diagrams and the understanding the system behaviour from different levels of abstraction. Our hypothesis is that engineers would benefit from a 3D state machine diagram giving them “the best of both worlds”, having advantages of separate substate diagrams but also the ability to view the state machine diagram as a whole.

![RoseRT example of a complete state machine with substates displayed in different window tabs. The ‘green’ state contains an icon indicating that substates exist.](image)