ViSta^{*}

Rodolfo Castelló, Rym Mili, and Ioannis G. Tollis

¹ Department of Computer Science, Institute of Technology and Higher Education of Monterey, Mexico

² Department of Computer Science, The University of Texas at Dallas, Richardson, TX 75083-0688, USA

1 Short Description

ViSta is a tool suite designed to support the requirements specification phase of reactive systems. It includes a template wizard, a graphical editor and a statechart visualization tool. The template wizard guides the user through the steps necessary for the extraction of relevant information from a textual description of requirements. This information is stored in a database that is used by the statechart visualization tool to automatically generate statechart layouts. The statechart visualization tool offers a framework that combines hierarchical drawing, labeling, and floorplanning techniques.

2 Areas of Application

Statecharts are widely used for the requirements specification of reactive systems. This notation captures the requirements attributes that are concerned with the behavioral features of a system, and models these features in terms of a hierarchy of diagrams and states. The usefulness of statecharts depends primarily on their readability, that is the capability of the drawing to convey the meaning quickly and clearly. Several visualization tools for the specification of reactive systems are available in the market [2,3]. Even though these tools are helpful in organizing designers' thoughts, they are mostly sophisticated small scale graphical editors, and therefore are severely inadequate for the modeling of complex reactive systems. Specifically, hand made diagrams quickly become unreadable when the specification complexity and size increase. Therefore computer assistance is of paramount importance for the graphical representation of complex reactive systems.

3 Layout Algorithms and Layout Features

In our approach, a statechart is treated as a graph. Nodes in the graph correspond to states, and arcs correspond to transitions between states. Our algorithmic framework [1] improves the readability of the drawings by focusing

^{*} Research supported in part by Sandia National Labs and by the Texas Advanced Research Program under grant number 009741-040.

on some of the most important aesthetic criteria for graph drawing, namely, *edge-crossing reduction*, *edge-bend reduction*, *labeling*, and *optimization of the drawing area*. It is based on several techniques that include hierarchical drawing, labeling, and floorplanning. The use of layout algorithms implies that even when a minor modification is performed on a drawing (e.g., addition of a node or a transition), the layout algorithms re-orders the nodes in such a way that the aesthetic criteria are met. Hence, the structure of the resulting drawing could be very different from the original one. ViSta offers a feature that can be used to preserve the *mental map* (i.e., the structure) of statecharts. This is important since the designer/specifier may find mistakes/omissions and hence may need to delete and/or insert states and/or transitions without losing the mental map.

4 Architecture

ViSta (version 2.3) was developed in JAVA 1.2 and uses JAVA's Swing API. It consists of four components: the *template wizard*, the *statechart graphical editor*, the *statechart visualization tool*, and the *database*.

The data stored in the database is summarized in a *decomposition tree*. This structure reflects the decomposition of superstates into substates. A node in the decomposition tree includes the following information: its name; its width and height; the coordinates of its origin point; a pointer to its parent; the list of its children; its decomposition type (e.g., *AND*, *OR* or *leaf*); the list of incoming arcs; the list of outgoing arcs; a list of attributes; and finally its aliases.

The template wizard is used when a textual description of requirements exists, and the user wants ViSta to generate the drawings. The user inputs a textual description by either opening an existing document or creating a new file. Then s/he selects information from the textual document and dynamically introduces it into a set of templates.

The user may decide to directly draw statecharts using the graphical editor. In the graphical editor, statechart elements are described using *item-forms* (e.g., state-form, transition-form). The information is immediately stored in the database, and is used by ViSta to generate (or update) the graphical, template and structured descriptions.

References

1. R. Castelló, R. Mili, and I. G. Tollis. An algorithmic framework for visualizing statecharts. In Joe Marks, editor, *Graph Drawing (Proceedings GD'00)*, pages 139–149. Springer-Verlag, 2001. Lecture Notes in Computer Science 1984.
2. D. Harel et al. Statemate: A working environment for the development of complex reactive systems. *IEEE Transactions on Software Engineering*, 16(4):403–414, May 1990.
3. Artisan Software Tools. Real-time studio: The rational alternative. Available from Artisan Software Tools over the Internet.
<http://www.artisansw.com/rtdialogue/pdfs/rational.pdf>. Accessed November 2000.