Learning Design Guidelines by Theory Refinement

Jacob Eisenstein

Stanford University
549 Lasuen Mall
Stanford, California 94305
jacob@redwhale.com

The automation of design decisions can be seen as a problem of generating mappings between elements in an abstract specification of the object to be designed and the concrete parts of the object itself (Puerta and Eisenstein 1999). In some cases, it is difficult to discover a formalism that takes all relevant variables into account; human designers proceed by "intuition." Individual designers may have stylistic preferences that are purely idiosyncratic or are common only to one particular "school." By ignoring such preferences, automatic design forfeits the flexibility, creativity, and vitality of human design.

In short, automatic design algorithms suffer from a lack of flexibility. Adaptation is offered as a solution to this problem. By making automatic design algorithms adaptive, we can begin to do automation without a complete knowledge base—it can be developed and refined along the way. Stylistic preferences can likewise be accommodated if an automatic design algorithm can adapt to the user.

We add adaptation to an existing piece of automatic design software: the TIMM module of the MOBI-D user-interface design environment (Puerta 1997). MOBI-D maintains explicit, formal representations of the abstract and concrete sides of the interface. TIMM automates the mappings between the abstract domain objects and concrete presentation elements, using a decision tree. Although there exists a body of work on using decision trees to automate selection of interactors (Vanderdonckt and Bodart 1996), most theory-refinement algorithms involve neural nets (Maclin and Shavlik 1996). A theory-refinement algorithm for decision trees is presented below.

An error occurs when the user disagrees with one of TIMM's design decisions. The system then searches for a set of operations that can be performed on the decision tree so as to yield the least number of errors, when applied over the entire history of user interactions.

When multiple solutions produce the same number of errors, operations that are reversible and do not affect the structure of the decision tree are preferred. The most preferred operation is to change the output of the leaf that produced the error. Suppose that the decision tree dictates that boolean domain elements are best treated by a checkbox, but the user selects radio buttons instead. This stylistic preference can be satisfied by changing the output of the leaf for boolean domain variables to radio buttons.

An alteration of the boundary conditions is the next most preferred operation, because it can be reversed by altering the boundary conditions again. Suppose that the decision-tree selects radio buttons for variables with less than four allowed values, and list boxes when there are four or more allowed values. If the user rejects the selection of list boxes in cases where there are four allowed values, then a shift in the boundary condition will produce more agreeable output.

There is no facility for removing branches. Thus, the addition of a new branch is the least preferred operation, as it is irreversible. It is necessary to add a branch when there is a relevant piece of information that the decision tree did not consider. The designer's selection of certain interactors might be influenced by the size of their parent dialog window. Adding a new discriminant to take that information into account is the only way to properly correct the decision tree.

Preliminary testing has shown that all of these operations are useful. Local minima have not appeared, so the fact that our search is a greedy hill-climb does not pose a problem. Earlier, three advantages of adaptation were cited: acquisition of new design knowledge, accommodation of user preferences, and update of the automation algorithm in response to changing technology. Testing has shown that the theory-refinement algorithm described here can deliver all three of those advantages.

Acknowledgements

I thank my advisor, Angel Puerta, for the direction and support. The work is supported by RedWhale Software.

References

Maclin, R. and Shavlik, J. 1996. Creating Advice-Taking Reinforcement Learners. *Machine Learning*, 22:251-281.

Puerta, A.R. and Eisenstein, J. 1999. Towards a General Computational Framework for Model-Based Interface Development Systems. In Proc. of IUI 1999, pp. 171-178. Los Angeles: ACM Press.

Puerta, A.R. 1997. A Model-Based Interface Development Environment. *IEEE Software*, 14(4): 41-47.

Vanderdonckt, J. and Bodart, F. 1996. The "Corpus Ergonomicus": a Comprehensive and Unique Source for Human-Machine Interface Guidelines, in "Advances in Applied Ergonomics." In ICAE'96, p. 162-169. Istanbul – West Lafayette: USA Publishing.