Towards a Shared Language for Problem-Solving in Design

Steven L. Tanimoto
University of Washington
tanimoto@cs.washington.edu

ABSTRACT

As tools for design get increasingly complex and powerful, it is increasingly important that their users have a solid grounding in what it is these tools do and how they do it. When these tools are software engineering tools, we can assume that their users have formal training in computing. However, when the tools are for designing art or multimedia objects, the artists using the tools typically don't have much background in the computing technology they are using. While interdisciplinary design teams can work around the shortcomings of individual team members, they might be more effective teams if they shared a common language and understanding of certain key aspects of the design process. At the University of Washington we have begun a project to study the use of the theory of problem solving as shared knowledge by design teams engaged in creating of multimedia games. The theory we used is derived from early work in artificial intelligence. The concepts of state space, search, operators and evaluation functions are key components of the shared knowledge. The theory is embodied in a software system called T-STAR (Transparent STate-spaces search ARchitecture) which supports a collaborative interface for problem-solving. A key question for the project is "to what extent will interdisciplinary design teams adopt and exploit the theory of problem solving when given an opportunity to do so?"

Categories and Subject Descriptors

D.2.14.a [**Human factors in software design**]: User interfaces; H.1.2.b [**Models and principles**]: Human-centered computing; H.5.2.f [**User interfaces**]: Graphical user interfaces; H.5.3.c [**Group and organization interfaces**]: Computer supported cooperative work; I.2.8.e [**Problem-solving, control methods, and search**]: Graph and tree search strategies; K.3.1.a [**Computer uses in education**]: Collaborative learning; K.3.m.b [**Miscellaneous**]: Computer literacy

SOD'07 Science of Design Symposium 2007, Humboldt State University, Arcata, California, USA

General Terms

Design, Human factors

Keywords

User interfaces, human-centered computing, design process methodology, collaborative learning, language of design, state-space search, tree visualization

1. INTRODUCTION

Many of the tools used for the design of computer-based objects are increasing in power and complexity. Computer programmers use software-development environments such as Eclipse and Microsoft Visual Studio. Artists use tools such as Adobe Illustrator, and some musicians use composing tools such as Sibelius, and Finale. Architects, engineers, screenwriters, and business planners use computer tools with more and more features and options. While great effort is usually directed at making these tools easy to use, it is often the case that designers don't fully understand their tools. Also, when members of an interdisciplinary design have to collaborate, there may be awkward miscommunications and inefficiencies, because they do not share a deep base of common knowledge about the theory of design.

A group of us at the University of Washington are beginning a study of the possible benefits of introducing a component of common knowledge to interdisciplinary design teams, where this common knowledge is based on the classical theory of problem solving described by H. Simon in *The Sciences of the Artificial.*

2. THEORY OF PROBLEM SOLVING

We have decided to adopt part of the classical theory of problem solving as the component of common knowledge for design teams in our work. This theory involves five key concepts: state, operator, space of states, evaluation function, and constraint. Although this theory was developed largely by computer scientists in the early days of artificial intelligence research, it is a widely applicable theory and offers the possibilities not only of working tools based on the theory but also of a lingua franca for aspects of the design process itself.

3. T-STAR FRAMEWORK

We have implemented a software system to form a foundation for our experiments. We call it T-STAR (Transparent STate-spaces search ARchitecture). It serves two pur-

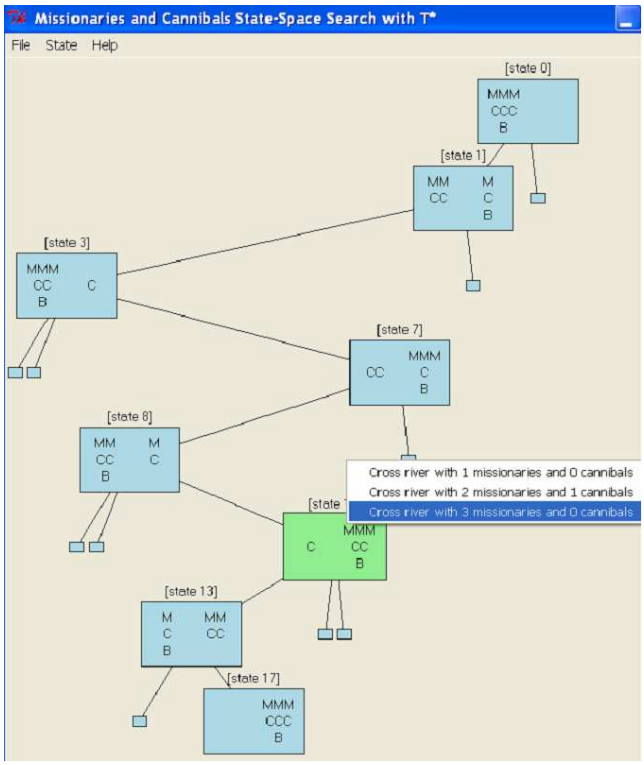

Figure 1: The Missionaries and Cannibals problem with the T-STAR system.

poses. First, it provides a graphical user interface to state-space search explorations. Second, it provides a software framework in which the various components of a problem-solving system can be embedded, with basic representations for states and their provenances, and operators with their preconditions. The actual software takes the form of a Python module that uses the Tkinter facility for graphical user interface development. The framework is typically applied by writing a new Python program that imports the T-Star module and subclasses its definitions for state, and node and provides a new set of operators.

An illustration of T-STAR applied to a classical puzzle (Missionaries and Cannibals) is shown in Figure 1. Here T-STAR allows the user to repeatedly select a state and an operator to apply to it. It can also automatically generate multiple levels of new states. By giving the user certain controls on the layout of the state-space graph, T-STAR supports visualizing the space of alternatives in problem solving (and design).

We have begun to implement an application of T-STAR to support a form of collaborative design. This program, called CO-STAR provides several additional features to T-STAR including (1) compound states that incorporate multiple media, (2) annotations of states to indicate authorship, comments and evaluations, and (3) facilities for merging trees of states generated by different team members.

CO-STAR is intended to provide a shared view of the state of a design process for an interdisciplinary design team. With a basic understanding of the theory of problem solving, we hypothesize that the team will be able to collaborate more effectively not only among themselves, but also with

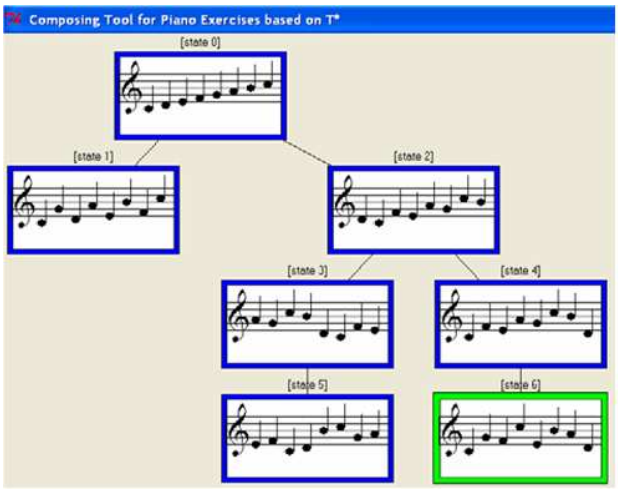

Figure 2: Musical motifs generated from a diatonic scale by applying permutation operators, in an application of T-STAR to composition.

automated services that help generate and screen candidate solutions to design problems.

4. IMPACT ON TEACHING OF DESIGN

In our project, we will have to teach users several new things: how to use CO-STAR; the underlying concepts of state, operator, state space, evaluation function, and constraint; and how to take advantage of this knowledge during collaborative design activities. We hope to develop a way of doing this that can transfer to other settings, and that other institutions might find useful.

One of the lessons we plan to teach is that the state-space search methodology can be applied to design in a variety of ways. One way is to guide the overall design process, so that each partial design in the process is represented by a state. A different way is to use a different state space and set of operators for each type of component that goes into the overall design, and generate material that would be integrated into the design at another level of the process.

5. IMPACT ON CREATIVITY

Systems such as T-STAR and CO-STAR, like many design tools, may seem both to encourage creativity and to limit creativity. We hope to reduce the limiting aspect, perhaps by allowing ways of using the software in which the users can "go outside" of the fixed operators and use the tree of states more as a version control mechanism and less as an expression of a narrow operator set. In any case, we hope that the "big picture" of the design process afforded by the state-space display helps designers to quickly push the limits of their imaginations within the spaces of possibilities in which they are working.

6. ACKNOWLEDGMENTS

The authors thanks B. Johnson, R. Karpen, S. Levialdi, and L. Shapiro for their participation in aspects of this work. This work was supported in part by the U.S. National Science Foundation under Grant 0613550.

7. REFERENCES

[1] N.Nilsson. *Problem-Solving Methods in Artificial Intelligence*. McGraw-Hill, 1971.

[2] H. A. Simon. *The Sciences of the Artificial*. MIT Press, third edition edition, 1996.

[3] S. L. Tanimoto. Transparent interfaces: Model and methods. In *Proceedings of the Workshop on Invisible and Transparent Interfaces*, May 2004.