

Shane Hoversten

Richard Polfreman, A task analysis of music composition and its application to the development of Modalyser

Polfreman's inquiry into the compositional process is motivated by a clear goal: understand the musical composition process and then, using this understanding, build tools to make it easier and better. This goal finds expression in Modalyser, a system built out of a set of graphical abstractions that interfaces with Modalys, a Lisp package for simulating acoustical objects. Polfreman's commendable idea is that most composers want to work at a higher level of abstraction than Modalys offers, in the analogous way that computer programmers want to think about problems in terms of their algorithmic constituents, not at the computer's level of bits and registers.

The compositional process is to be analyzed using a general task analysis framework, KAT/TKS. The result of this process is a general task model, which in this paper is a model of the musical compositional task. The task description produced by the analysis consists of three tasks, each of which is composed of a number of subtasks. A brief description of these tasks, and subtasks, follows.

Task 1: Design framework. Designing the framework sets down the initial constraints that the composition must meet, and a plan to meet them. The plan need not be complete, but should be explicit enough to set the rest of the compositional process in motion. The framework includes the generation systems, the musical structure, the performance systems, the overall format, and the tools. In other words: what sort of music you're going to make, what instruments you're going to make it with, how are you going to produce the music from those instruments, and how will you deliver the finished product.

Task 2: Research: This is a catch-all task regarding the research goals in support of the composition. This is a general notion and the author does not try to model it explicitly, which is good because a model that attempts to enumerate the enumerable must either fail or become general to the point of uselessness. (See other paper.)

Task3: Produce music: This is where the rubber meets the road, and the music is set down in some external form. This task is further structured by subtasks that mimic the structure of the music: write chunk, modify chunk, audition chunk, and format chunk. The flow between these subtasks reflects the iterative nature of creative development: put something down, listen to it, tweak it, add something to it, tweak it.

Polfreman spends some time discussing an important issue that the other paper completely neglected: "There is a common problem for task analysts which is to decide the grain of analysis, i.e., where should the analysis stop in terms of the breakdown of goals into more and more subgoals." After a discussion of some mediating factors he comes to where I think the answer must be found: the grain of analysis is sufficient when further decomposition "... would not be sufficiently relevant for user-interface design purposes." Polfreman realizes that a model is a tool; his goal is to develop a user interface. His model is useful only inasmuch as it facilitates such a task.

The task decomposition was then applied to the development of Modalyser, the front-end for Modalys, though “front end” implies a simple wrapper to shortcut translations from one medium to another, and Modalyser seems an entirely new abstraction. The paper describes the ways the existing Modalys functionality was packaged, and how it could be manipulated.

A more detailed example was shown, which translated the Modalys Lisp code to an architecture of interrelating graphical objects. I know very little about music in general, and even less about the acoustics of musical instruments in particular, and yet I had a reaction that I suspect would have pleased Polfreman: I imagined myself doodling with the various graphical components, moving them around, hooking them together, seeing what happened. It’s doubtful I would ever do such things with the Modalys-only incarnation of the system.

I am both a computer scientist and a Lisp aficionado. Modifying a Lisp “program” for an acoustic device would be very natural for me. The key point is that if I were to play with this system it would be because I wanted to make sounds; and while Lisp is a beautiful and powerful language it wouldn’t be the appropriate abstraction for my level of intent. It’s important to note that Modalyser is not inherently superior to Modalys. In fact, with regard to the design of anything, one thing is almost never superior to anything else. The cross product of the needs of the particular user for the particular task is the ultimate arbiter of “success” for a design. Modalyser seems a nice realization of this idea.