Customisable Frameworks for Compositional Activity

Tim Anderson, Ross Kirk
Drake Music Project / Department of Electronics, University of York, UK
mta, prk@ohn.york.ac.uk

ABSTRACT: Composition using a computer system requires a user to perform many complex tasks or actions which manipulate musical data. These tasks may involve repetition and decision-making at each stage. This paper describes the design of a software overlay which allows a user to structure compositional tasks into coherent Activities within a computer-based composition system. Users can undertake compositional processes and make musical decisions at a high or low level by choosing an appropriate activity, within which options and paths of action may be selected.

Each composer has, and develops, ways of working - a palette of operations, strategies and structures - for the creation of a particular piece. He/she may start a new piece by first conceiving a set of concepts, data structures, rules or operations, then working within this framework to create the piece. Typically, several pieces may be produced within the same conceptual framework, and typically this framework is extended and modified as composition proceeds. In subsequent work, a composer will often reuse the compositional strategies and activities which were initially developed during the creation of previous work, as well as develop new ones.

A set of such activities - along with the musical considerations for making decisions about how and when to undertake an operation - can be termed a 'compositional framework'. State of the art music composition software packages, both experimental and commercial (eg Esquisses, Mixis, MODE, E-Scape, Logic) typically provide a complex set of possible operations which can be undertaken to create and manipulate musical data. A user of such a system needs to perform a large number of different data manipulation and control actions. The software overlay described here provides pathways through the maze of possible activities which such composition systems present. Composers can build up a library of such frameworks, and can see overtly how their compositional methodology is formulated. This can help composers describe and develop their compositional approaches and methods, and present them to others.

The framework is presented to the composer as a set of decisions about what operations to carry out (eg to enter or process material, or alter sounds). Operations may be carried out in a cyclic manner, or be context dependent. The presentation of the user interface in the form of such composition frameworks can also facilitate more efficient use of a system. Frameworks can thereby be useful to other kinds of user, such as disabled composers, for whom each individual operation or action may be tiring, thus discouraging experimentation. Composition students can work within a prepared didactic framework, progressing through a set of tutorial activities, within which they are allowed creative freedom, making musical decisions and choices. As students progress, the extent of the choices provided may be altered, and they may eventually work with advanced techniques within the frameworks of established composers.

For all users, undertaking compositional operations within a restricted set of options can be helpful. Composers are still allowed creative freedom; able to make musical evaluations, decisions and choices within the framework, and also modify a compositional framework themselves as required. The structuring of activities is deliberately underway as part of the discipline of composing within a framework.

Activities as structured Actions

The present software overlays the 'E-Scape' composition system (Anderson 90, 93) and structures compositional tasks within it into coherent Activities. Many Activities are provided as standard in the system, but a user can also construct additional Activities or modify existing ones. An 'Activity' consists of a number of Actions which may be undertaken in order, or navigated via a path (eg branches, repetitions, jumps) defined by user-selected choices. An Action may also consist of another Activity, enabling Activities of arbitrary complexity to be constructed. There are seven main classes of Action:

- SystemAction - a primitive named operation which the host system (eg 'E-Scape') can perform. Typical SystemActions in E-Scape involve selecting, playing and editing events.
- Activity - is another defined Activity.
* Jump - specifies a move to another Action in various ways.
* ValueSelector - allows the user to select a numeric value in a variety of ways.
* TextSelector - allows the user to enter text in a variety of ways.
* SwitchSelector - allows the user to directly select between two or more Actions.
* MenuSelector - allows the user to select between Actions displayed as a menu.

MenuSelectors and SwitchSelectors facilitate user choice of actions within an activity, allowing a user to determine the path taken through an Activity. All Selectors can be controlled by VirtualSwitch or VirtualController objects, which process incoming data from a variety of physical inputs, and may also take account of system variables previously set. Further detail is beyond the scope of this paper.

Example Activities

An overview of the structure of a simple low-level compositional Activity is shown below. Even such a relatively simple Activity is usually constructed from other lower-level Activities. For reasons of space, this example has no user choice (Selector) Actions, although its component Activity does. Such simple Activities will typically be incorporated within higher-level, more sophisticated ones.

Activity name: 'Copy block with conditions'

* Activity: Select (make current) a source block from which events are to be extracted.
* SystemAction: Load (make current) a ConditionFilter - a system object which facilitates the conditional selection of events using their parameter or time values.
* SystemAction: Copy events from the current block to the clipboard, using the current condition filter.
* Activity: Select a destination score and insert time, then paste clipboard events into it.

The latter Activity, for example, in turn consists of several lower-level actions:

- SystemAction: Select (make current) a destination score.
- SystemAction: Select (make current) a time within current score.
- SystemAction: Paste clipboard events into current score starting at current time.

Some other examples of Activities at various levels are listed below. Some Activities involve operations specific to a particular composer’s working methods, and have been given a short name whose meaning is not immediately apparent; other more verbose names attempt to give more indication of the nature of the Activity. Some Activities are relatively low-level and well-defined, achieving a specific aim with a degree of user choice provided. Other Activities are higher-level, encapsulating a large-scale set of musical processes, and incorporating many levels of choice and subordinate Activities.

- 'Create custom microtonal scales'
- 'Create scale from block'
- 'Audition hocket variations'
- 'Write piece in style 'new structure 95''
- 'Reduce block to tone row'
- 'Mutate tone row'
- 'Permute pitches randomly with weighting'
- 'Make Baseline'
- 'Create ostinato patterns from block'
- 'Expand block'
- 'Cycle pitches in block'
- 'Compress rhythm'
- 'Copy, invert pitch, and append'
- 'Transpose bar within a selected key'
- 'Copy, process duration, superimpose'
- 'Transform timbre trajectory'

Further work

Future AI development could allow the system to evaluate a user’s operations, and adapt its presentation of choices to the user, rather than being a fixed network of possible paths through activities. Activity frameworks could also be built up automatically by monitoring the kinds of operations the user carries out, and abstracting them into higher-level strategies, with user options provided at appropriate points.

References


• Jump - specifies a move to another Action in various ways.
• ValueSelector - allows the user to select a numeric value in a variety of ways.
• TextSelector - allows the user to enter text in a variety of ways.
• SwitchSelector - allows the user to directly select between two or more Actions.
• MenuSelector - allows the user to select between Actions displayed as a menu.
MenuSelectors and SwitchSelectors facilitate user choice of actions within an Activity, allowing a user to determine the path taken through an Activity. All Selectors can be controlled by VirtualSwitch or VirtualController objects, which process incoming data from a variety of physical inputs, and may also take account of system variables previously set. Further detail is beyond the scope of this paper.

Example Activities
An overview of the structure of a simple low-level compositional Activity is shown below. Even such a relatively simple Activity is usually constructed from other lower-level Activities. For reasons of space, this example has no user choice (Selector) Actions, although its component Activities do. Such simple Activities will typically be incorporated within higher-level, more sophisticated ones.

Activity name: ‘Copy block with conditions’ actions:
• Activity: Select (make current) a source block from which events are to be extracted.
• SystemAction: Load (make current) a ConditionFilter - a system object which facilitates the conditional selection of events using their parameter or time values.
• SystemAction: Copy events from the current block to the clipboard, using the current condition filter.
• Activity: Select a destination score and insert time, then paste clipboard events into it.

The latter Activity, for example, in turn consists of several lower-level actions:
• SystemAction: Select (make current) a destination score.
• SystemAction: Select (make current) a time within current score.
• SystemAction: Paste clipboard events into current score starting at current time.

Some other examples of Activities at various levels are listed below. Some Activities involve operations specific to a particular composer’s working methods, and have been given a short name whose meaning is not immediately apparent; other more verbose names attempt to give more indication of the nature of the Activity. Some Activities are relatively low-level and well-defined, achieving a specific aim with a degree of user choice provided. Other Activities are higher-level, encapsulating a large-scale set of musical processes, and incorporating many levels of choice and subordinate Activities.

‘Create custom micro tonal scales’ ‘Create scale from block’
‘Audition locket variations’ ‘Write piece in style ‘new structure 95’’
‘Reduce block to tone row’ ‘Mutate tone row’
‘Permutate pitches randomly with weighting’ ‘Make Baseline’
‘Create ostinato patterns from block’ ‘Expand block’
‘Cycle pitches in block’ ‘Compress rhythm’
‘Copy, invert pitch, and append’ ‘Transpose bar within a selected key’
‘Copy, process duration, superimpose’ ‘Transform timeline trajectory’

Further work
Future AI development could allow the system to evaluate a user’s operations, and adapt its presentation of choices to the user, rather than be a fixed network of possible paths through actions. Activity frameworks could also be built up automatically by monitoring the kinds of operations the user carries out, and abstracting them into higher-level strategies, with user options provided at appropriate points.

References