QTk – A Mixed Declarative/Procedural Approach for Designing Executable User Interfaces

Donatien Grolaux[1], Peter Van Roy[1], and Jean Vanderdonckt[1]

Université catholique de Louvain, B-1348 Louvain-la-Neuve, Belgium
{ned, pvr}@info.ucl.ac.be, vanderdonckt@qant.ucl.ac.be

When designing executable user interfaces, it is often advantageous to use declarative and procedural approaches together, each when most appropriate:

- A declarative approach can be used to define widget types, their initial states, their resize behavior, and how they are nested to form each window. All this information can be represented as a data structure. For example, widgets can be records and the window structure is then simply a nested record.
- A procedural approach can be used when its expressive power is needed, i.e., to define most of the UI's dynamic behavior. For example, UI events trigger calls to action procedures and the application can change widget state by invoking handler objects. Both action procedures and handler objects can be embedded in the data structures used by the declarative approach.

The QTk tool uses this mixed approach, tightly integrated with a programming language that has extensive support for records and first-class procedures. This permits *executable model-based UI design*: the UI models are executed at run-time without any compilation. To be precise, each UI model is a record that is transformed at run-time to its QTk specification, which is also a record.

We demonstrate the effectiveness of this approach by writing a context-sensitive clock utility, FlexClock, that changes its view at run-time whenever its window is resized. The utility is written in less than 400 lines. This includes full definitions of a calendar widget, an analog clock widget, and 16 views. Each view is defined as a record with three fields. All 16 views including formatting utilities are written in 80 lines total. The mechanism for creating a running UI from these definitions is written in 60 lines. Here is the definition of one view:

```
view(  desc: label(handle:H bg:white glue:nswe)
     update: proc {$ T} {H set(text:{FormatTime T})} end
       area: 40#10)
```

The desc field is declarative; it defines the view's structure as a record. Here it is a label widget with an embedded handler object referenced by H. The handler object is created by QTk when the widget is installed. The update field is procedural; it contains an embedded procedure that will be called once a second with a time argument T to set the displayed time. The area field gives the view's minimum width and height, used to select the best view at run-time.

QTk uses tcl/tk as its underlying graphics subsystem. It is part of the Mozart Programming System, which implements the Oz language. For full information see http://www.info.ucl.ac.be/people/ned/flexclock. This research is supported in Belgium by the PIRATES project, funded by the Walloon Region.