Relation Oriented Programming with Raloo

What Happens When ::ral meets ::oo?

Andrew Mangogna

15th Annual Tcl/Tk Conference
October 20-24, 2008
Manassas, Virginia
Relation Oriented Programming

• Raloo is a Tcl script package that implements a form of Relation Oriented Programming.
• Raloo combines:
  – TclRAL ⇒ relation values, relvars, integrity constraints, relational algebra operations
  – TclOO ⇒ classes, objects, methods, OO building blocks
• Raloo emphasizes:
  – Strong data structuring via relations
  – Event driven state machines for sequencing processing
  – Tcl code for algorithmic processing
  – Domains for packaging subject matters
Raloo Combines TclRAL with TclOO

- Raloo Classes are TclOO classes with object data stored in a TclRAL relvar.
- Raloo objects reference tuples in the class relvar.
- Raloo relationships are TclRAL relvar constraints. Referential integrity is checked automatically.
- Raloo supports associating a state machine with a Class for asynchronous processing.
- Processing is accomplished by ordinary Tcl code.
Three Projections of a Raloo Solution

- Relationally normalized class model.
  - Classes
  - Relationships
  - Integrity constraints
- Finite state machine model of asynchronous processing.
  - Moore machine for active classes
  - State machine dispatch uses Tcl event loop
- Object oriented Tcl code for processing.
  - Methods for navigating the class model
  - Methods for generating state machine events
One Button Microwave

● One control button
  - Press button with door closed runs for 1 min.
  - Press button while running adds a minute.
  - Opening the door while running stops the oven and resets the time.

● Usual safeguards apply
  - Light must be on when the door is open or the microwave tube is on.
  - Microwave tube may only be on when the door is closed.
One Button Microwave

**Oven**
- *OvenId*
- *CookingTime*

**Lamp**
- *OvenId (R2)*

**Tube**
- *OvenId (R1)*

- lights
- 1
- R2
- 1
- R1
- 1

provides cooking energy for
One Button Microwave - Classes

Class Oven {
    Attribute {
        *OvenId int
        CookingTime int
    }
    Lifecycle {

        State initialCookingPeriod {} {
            # 1. Set time for 1 minute
            my writeAttr CookingTime 1
            my generateDelayed 60000 TimeExpired
            # 2. Generate: Turn on light
            set light [my selectRelated ~R2]
            $light generate TurnOn
            # 3. Generate: Energize power tube
            set tube [my selectRelated ~R1]
            $tube generate Energize
        }

        Transition initialCookingPeriod - TimeExpired -> \ cookingComplete
        Transition initialCookingPeriod - ButtonPushed -> \ cookingPeriodExtended
        Transition initialCookingPeriod - DoorOpened -> \ cookingInterrupted
    }

    ..........
One Button Microwave Demo

Oven User Interface Domain

Oven Management Domain
Move Along, Nothing New Here

- Ideas behind Raloo are not new or original.
- Three projections of the problem space.
  - Static structure encoded as a relation class model
  - Dynamics encoded as a state machine
  - Algorithms written in code
  - Capture program structure declaratively
- Raloo execution semantics match those of Executable UML.
- Raloo combines the foundations provided by TclRAL and TclOO.
  - TclRAL is a complete relational algebra
  - TclOO is a set of object oriented building blocks
Where to Get Raloo

- Raloo and TclRAL are both free software: [http://sourceforge.net/projects/tclral](http://sourceforge.net/projects/tclral)
- Requires TclOO (0.5.1).
- Requires Tcl 8.5 or better.
- Read the paper! Please. More examples, explanation and references there.