Relation-Oriented Programming with Raloo
What Happens When ::ral Meets ::oo?

Abstract

Author: Andrew Mangogna
amangogna@modelrealization.com

Raloo is an objected-oriented extension to Tcl that combines the relational data structuring capability of TclRAL with an object-oriented programming style as provided by TclOO. This paper introduces Raloo and describes how the capabilities of Raloo may be used to capture the semantics of a software problem in a more declarative manner. Raloo supports problem decomposition into domains, with domains containing classes, relationships and domain functions. Both synchronous processing and asynchronous processing via state machines are provided. The relationship of Raloo to formal software methods is also discussed.

Relation-Oriented Programming with Raloo
What Happens When ::ral Meets ::oo?

Summary

Author: Andrew Mangogna
amangogna@modelrealization.com

Raloo is an objected-oriented extension to Tcl that combines the relational data structuring capability of TclRAL with an object-oriented programming style as provided by TclOO. Raloo serves as both a convenient relation-oriented programming scheme and as a framework for experimenting with programming approaches that emphasize declarative specification over procedural coding.

The Relational Model of Data provides a formal and comprehensive means to structure the semantics of any problem into data. As it is usually formulated, the Relational Algebra is presented as a conventional set of operations on values where processing and data are kept distinct. Frequently, the Relational Model is discussed only in the context of its foundation for Data Base Management Systems (DBMS), however the algebra of relations is independent of any particular usage or implementation and can form the basis for structuring data in more conventional programming approaches.

The advent of object-oriented programming techniques has demonstrated the utility of closely associating data with the processing intended for it. Although there are many formulations of object-oriented programming, the common features are:

(1) A close association of operations with the data to which they apply. These operations are often called methods and the methods provide a means of encapsulation for the data.

(2) Support for run-time resolved polymorphic methods. This is often associated with a concept of hierarchically-based inheritance of physical data structure that allows the various nodes of the hierarchy to specify different processing.

Raloo is a Tcl script-based extension that combines data structuring constructs from Relational Algebra with the constructs of object-oriented programming. Raloo is based on the TclRAL relational algebra extension using the object orientation of the TclOO extension. A Raloo solution to a software problem entails three distinct projections of the problem:

(1) A projection of the static nature of the problem as a set of relationally normalized classes and relationships for which the attributes of the classes constitute the data parameterization of the problem.

(2) A projection of the dynamic nature of the problem as a set of interacting state machines that capture the life-cycle of the active classes.

(3) An algorithmic projection consisting of Tcl code sequences that perform the necessary algorithmic computations.

In Raloo a problem is divided into distinct subject matters, known as domains. A domain consists of a set of classes, relationships and domain functions. Domains are the unit of encapsulation and the domain functions provide the procedural interface to the domain. A class is the programatic realization of a real-world entity of the subject matter. A relationship is the realization of the real-world associations among the classes. The classes and relationships have a direct correspondence to TclRAL relvars and constraints.

In addition to ordinary synchronous operations such as invoking methods and procedures, Raloo supports asynchronous computations by providing a class with the ability to have an associated Moore type state machine. State machines are driven by events that are generated to class instances and the Tcl event loop is used to coordinate the execution of the state machines. Events may carry parametric data which are delivered to the state machine actions as arguments. The actions of a state machine are specified in object-oriented Tcl code where specific methods are provided to manipulate the class schema and generate events.

As one state machine generates events to other state machines, the dynamics of the domain evolves as a thread of control. A thread of control is a tree that grows over time where the nodes represent class instances and an edge represents an event generated to a class instance. Raloo completes one thread of control before starting another thread of control, deferring new threads as necessary. Raloo also enforces a data transaction at the end of each thread of control. Referential integrity of the class schema is checked at each thread of control boundary and any operations that leave the class instances in a referentially inconsistent state result in an error and the data values are rolled back to where they were at the beginning of the thread of control. Raloo also supports monitoring and controlling the program dynamics by providing introspection on the thread of control information.

Raloo is based on the execution semantics of the Shlaer-Mellor Method, now know as Executable UML. This software methodology has been in continuous development since the late 1980's merging with the graphical conventions of UML to become a UML profile and encouraging UML's subsequent and ongoing attempts to embrace Model Driven Architecture. From an Executable UML view point, Raloo may be seen as a software architecture domain implemented in Tcl where there is a direct correspondence between Executable UML semantics and the constructs of Raloo. However, no Executable UML knowledge is required to use Raloo. From the

Tcl view point, Raloo may be seen as an object-oriented extension that emphasizes strong, consistent, constrained data structuring and asynchronous processing via state machines, the actions of which are coded in an object-oriented Tcl style. From either point of view, the goal of Raloo is to raise the level of abstraction for creating a program by providing a programming interface that is more declarative in nature. Classes, relationships and state machines, which constitute the structural basis for a domain, are specified to Raloo in a declarative manner with the Raloo package providing the common, factored code necessary to control program execution.

```
Abstract                    1                    May 4, 2008
```