XOTcl 2.0

A Ten-Year Retrospective and Outlook

Gustaf Neumann, Stefan Sobernig

Institute for Information Systems and New Media

WU Vienna, Austria

Agenda

Topics:

- Background
- Support for *Language Oriented Programming*
- Parameterization in XOTcl 2.0
- Refactoring and Code Development
- Evaluation

Not: Introduction, Feature comparison with TclOO, full serialization since many years

Background

- Developed in 1999, released around Tcl/Tk 2000, 20+ paper, 38 releases
- Used in products of several companies (e.g. Archiware, ..., Knowledge Markets)
- Part of e.g. OpenACS (Code Base 2 Mio LOC, built around Aolserver/Naviserver)
- In several e-learning systems (Learn@WU, Daimler, ...)

Research Goals behind XOTcl

- Language support for Design Patterns (filters, transitive mixin classes)
- Glueing on the OO level, Dynamic Object Systems

Language-Oriented Programming

Basic Ideas

- Little Languages (Jon Bentley, 1986)
- Discussion: Do we need a common OO Language or a OO Language framework for Tcl?
- Domain Specific Languages (Martin Fowler, 2005)

XOTcl 2.0 provides framework of TIP #279 + extensions

- TIP#279 based implementation of Incr Tcl in Nov 2006 (passed 90% of regression test)

Creating Derivative Object Systems

Basic idea: allow multiple object-systems in parallel ...

```
#
# Defining the Root Classes of XOTcl
#
::xotcl::createobjectsystem ::xotcl::Object ::xotcl::Class
```

... and "register" behavior under **arbitrary names** into every object system

```
#
# Register an XOTcl/C command: Class.alloc()
#
::xotcl::alias ::xotcl::Class alloc ::xotcl::cmd::Class::alloc
#
# a Tcl/C command: Object.append()
#
::xotcl::alias ::xotcl::Object append -objscope ::append
```

Implementation reuse, without OO inheritance, zero overhead

Parametrization

Parameterization in Tcl:

- *procs:* Definition of arguments (positional, defaults for "last arguments", varargs via "args"); introspection possible
- *cmds:* Every Tcl command is responsible for parsing its argument vector, responsible for semantics and error messages; no introspection possible

Parameterization in XOTcl 1.x

- *objects:* additionally parameterization of initial object state

Parameterization in XOTcl 2.0

Provide a **single mechanism** for proc, cmd and object parameterization, supporting

- positional and non-positional (named) arguments,
- required and non-required values,
- defaults handling,
- value constraints,
- consistent error messages, introspection, ...

Parameters for C-implemented Commands

Basic Ideas:

- Replace hand-coded, per Tcl cmd C code by an introspectable C structure
- Generate this C structure from a (Tcl-based) interface definition language
- Generated C code from this language (producing stubs)
- Use a single objv-parser for all C implemented commands
- Same mechanisms for Tcl-commands, C implemented XOTcl methods, ...

XOTcl 2.0: explicit signature definitions used for about 100 C implemented commands and methods

Signature Definition of C implemented Commands

Definition:

```
#
# XOTcl commands
#
xotclCmd alias XOTclAliasCmd {
  {-argName "object" -required 1 -type object}
  {-argName "methodName" -required 1}
  {-argName "-objscope"}
  {-argName "-per-object"}
  {-argName "-protected"}
  {-argName "cmdName" -required 1 -type tclobj}
}


#
# class methods
#
classMethod alloc XOTclCAllocMethod {
  {-argName "name" -required 1}
}
```


Generated Stub for Sample Method

```
static int
XOTclCAllocMethodStub(ClientData clientData, Tcl_Interp *interp, int objc,
                      Tcl_Obj *CONST objv[]) {
  parseContext pc;
  XOTclClass *cl = XOTclObjectToClass(clientData);
  if (!cl) return XOTclObjErrType(interp, objv[0], "Class");
  if (ArgumentParse(interp, objc, objv, (XOTclObject *) cl, objv[0],
                    method_definitions[XOTclCAllocMethodIdx].paramDefs,
                    method_definitions[XOTclCAllocMethodIdx].nrParameters,
                    &pc) != TCL_OK) {
    return TCL_ERROR;
  } else {
    char *name = (char *)pc.clientData[0];
    parseContextRelease(&pc);
    return XOTclCAllocMethod(interp, cl, name);
  }
}
```

Parameters for Tcl implemented Methods

Method Definition:

```
Class create C
C method foo {a {-trace false} b:integer c:optional} {...}
```

Description:

- Syntax for parameter definition influenced by OpenACS
- Non-positional parameter with default (2nd arg)
- Value-constraint on "b"
- Optional last argument (like for e.g. Tcl "set")
- Same functionality for all kind of parameterization

Object Parametrization in XOTcl 2.0

```
#
# Create a Class with parameters for its instances
#
Class create Foo -parameter {a:integer,required {trace:boolean false}}
...
# Computed parameter definition by
# Tcl-implemented method "objectparameter":
#
# -a:integer,required {-trace:boolean false} -mixin:relation -filter:relation
#       -class:relation args
...
#
# Create an object and parameterize it
#
Foo create f1 -a 123
```

General Improvements of the XOTcl 2.0 Code Base

- Usage of Tcl's client data in Stack frames
  - Get rid of XOTcl's own stack for Tcl versions before 8.5
- Simplified object deletion mechanisms
- Usage of variable and command resolvers
  - Used for variable and command names starting with a "."
  - Only on XOTcl method stack frames
- XOTcl 2.0 works with Tcl 8.5 and 8.6b1 (mostly with 8.4)

Code Example with XOTcl 2.0 Resolver

```
# Example from Language shoot-out
#
Class Toggle -parameter state
Toggle method value {} {
 set .state
}
Toggle method activate {} {
 set .state [expr {! ${.state}}]
 self
}

Class NthToggle -superclass Toggle -parameter max
NthToggle method init {} {
 next
 set .counter 0
}
NthToggle method activate {} {
 if {[incr .counter] >= ${.max}} {
   next
   set .counter 0
 }
 self
}
```

Evaluation

- **Code reduction** by more than 2500 lines of code through generalized parameterization
- Significant **speed improvements** relative to XOTcl 1.6 (creation with object-parameters more than twice as fast, dispatch with multiple arguments several times faster)
- Faster than TclOO 0.6
- On Shootout Benchmark:
  - Without resolvers: XOTcl 2.0 is about 2x faster than XOTcl 1.6
  - With resolvers: XOTcl 2.0 is about 4x faster than XOTcl 1.6

Time per Object Creation

Object System	alloc	create empty obj	create with var	destroy
XOTcl 1.6.0	4.23	7.89	13.59	3.57
XOTcl 2.0.0	3.69	5.00	5.90	2.95
TclOO 0.6	7.87	5.36	6.32	50.97

- Object Creation (bare create, with constructor, with single parameter) and destruction (in μs)
- XOTcl 2.0 is everywhere faster than XOTcl 1.6
- Best improvements for the most realistic cases

Number of Method Invocations per Second

Object system	args0	args3	np2	np2args3
XOTcl 1.6.0	771,968	693,450	336,866	184,997
XOTcl 2.0.0	1,437,876	1,268,713	1,175,979	882,530
TclOO 0.6	1,264,366	1,119,037	n.a.	n.a.

Method Dispatch Throughput per Second

- XOTcl 2.0 is everywhere significantly faster than XOTcl 1.6
- Dispatch on methods without arguments 2x faster
- Dispatch on methods with realistic arguments 5x faster

Memory Usage

Object system	Memory per object (bytes)
XOTcl 1.6.0	450
XOTcl 2.0.0	514
TclOO 0.6	1473

Memory Footprint per Object

Effects of resolvers

Version	Time (sec)	Relative Speed
XOTcl 1.6	4.16	100.00%
XOTcl 2.0.0	2.27	54.48%
XOTcl 2.0.0 (with resolver)	0.96	23.16%

Summary

- XOTcl 2.0 is significantly
  - more flexible,
  - powerful (e.g. extensible submethods) and
  - faster than XOTcl 1.6
- Runs on Tcl 8.5 and 8.6b1 (do we need 8.4?)
- Plan for Release: in 2009
- Still to do:
  - Documentation
  - Naming overhaul
  - Compatibility layer for XOTcl 1.x

- Testing with large code basis (i.e. OpenACS)