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 ...

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

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

```tcl
# 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, ...
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 XOTclCAlocMethod {
    {-argName "name" -required 1}
}
```

Generated Stub for Sample Method

```
static int
XOTclCAlocMethodStub(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[XOTclCAlocMethodIdx].paramDefs,
                      method_definitions[XOTclCAlocMethodIdx].nrParameters,
                      &pc) != TCL_OK) {
        return TCL_ERROR;
    } else {
        char *name = (char *)pc.clientData[0];
        parseContextRelease(&pc);
        return XOTclCAlocMethod(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

```tcl
# 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

<table>
<thead>
<tr>
<th>Object System</th>
<th>alloc</th>
<th>create empty obj</th>
<th>create with var</th>
<th>destroy</th>
</tr>
</thead>
<tbody>
<tr>
<td>XOTcl 1.6.0</td>
<td>4.23</td>
<td>7.89</td>
<td>13.59</td>
<td>3.57</td>
</tr>
<tr>
<td>XOTcl 2.0.0</td>
<td>3.69</td>
<td>5.00</td>
<td>5.90</td>
<td>2.95</td>
</tr>
<tr>
<td>TclOO 0.6</td>
<td>7.87</td>
<td>5.36</td>
<td>6.32</td>
<td>50.97</td>
</tr>
</tbody>
</table>

- 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

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

### 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

<table>
<thead>
<tr>
<th>Object system</th>
<th>Memory per object (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>XOTcl 1.6.0</td>
<td>450</td>
</tr>
<tr>
<td>XOTcl 2.0.0</td>
<td>514</td>
</tr>
<tr>
<td>TclOO 0.6</td>
<td>1473</td>
</tr>
</tbody>
</table>

### Memory Footprint per Object
Effects of resolvers

<table>
<thead>
<tr>
<th>Version</th>
<th>Time (sec)</th>
<th>Relative Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>XOTcl 1.6</td>
<td>4.16</td>
<td>100.00%</td>
</tr>
<tr>
<td>XOTcl 2.0.0</td>
<td>2.27</td>
<td>54.48%</td>
</tr>
<tr>
<td>XOTcl 2.0.0 (with resolver)</td>
<td>0.96</td>
<td>23.16%</td>
</tr>
</tbody>
</table>

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)