Making Tcl "legacy" code object oriented

Artur Trzewik
mail@xdobry.de

http://www.xdobry.de/xotclIDE

5. European Tcl Workshop 2003 - Bergisch Gladbach

Motivation

- OO in Tcl still hot discussed
  - Posting „Do you want OOP?” in comp.lang.tcl 31 items
  - Wiki „Poll: do you want OOP?“ - 131 times edited
- Growing popularity of XOTcl (part of Active State Tcl Distribution)
- Still many questions and unclear answers?

Agenda

- Examples: OO-like code in pure Tcl
- Porting an Wiki example to XOTcl by using XOTclIDE
- XOTclIDE – what is new.
- Pro & Contr for OO with Tcl

Tk is object oriented

We can use window as object

```
button .tl.button
.tl.button confiugre -text „new text“
destroy .tl.button
```

How it works internal

```
proc ::tk::dialog::file::chooseDir::DblClick {w} {
     upvar ::tk::dialog::file::[winfo name $w] data
     set selection [tk::IconList_Curselection $data
(icons)]
      ...
}
```

Variant – global arrays

```
proc callOnObject {structureRef} {
   upvar #0 $structureRef myData
   set attr $myData(attr)

}
```

Most solution use arrays as primary structure.
Also global lists are possible.

Helper – lset, keyed lists (tclx), dict (Tcl5)

Variant – arrays in caller context

```
proc addNode {_g node args} {
     upvar 1 $_g g
     set id [llength $g(nodes)]
     set g($id) [concat $node $args]
     lappend g(nodes) $id
     set id

}
```

Bigger Example

- Package for tree processing.
- Source. Wiki (http://mini.net/tcl/1664)
  Simple tree layout by Richard Suchenwirth
- Clear pattern

Migration to XOTcl

```
# Original code
proc terminals _g {
    upvar 1 $_g g
    set res  {}
    foreach i [nodes g] {
        if {[sons g $i]==""} {lappend res $i}
    }
    set res
}
# migrated to XOTcl
Class OTree
OTree instproc terminals {} {
    my instvar g
    set res  {}
    foreach i [my nodes] {
        if {[my sons $i]==""} {lappend res $i}
    }
    set res
}
```

Tcl/XOTcl – client view

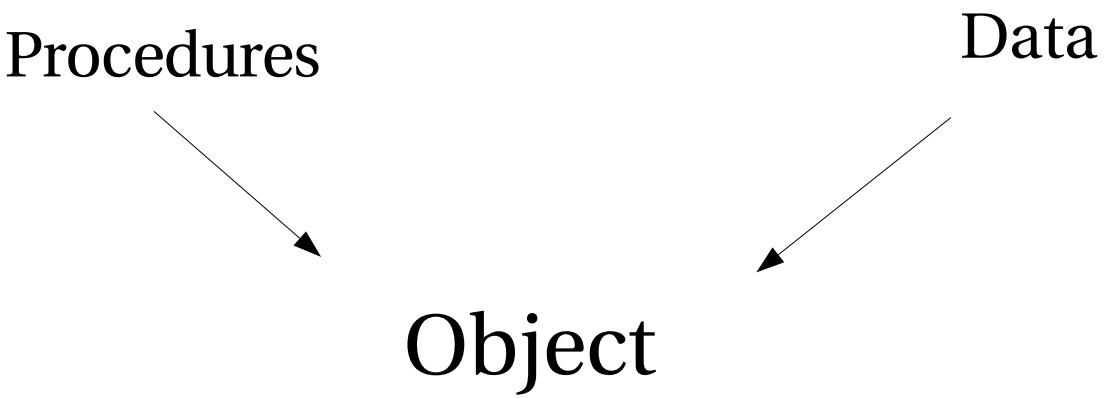
```
# Original code
graphInit g
treelist2graph {A {B C D} {E F G}} g
sons g B


# migrated to XOTcl
OTree create g
g treelist2graph {A {B C D} {E F G}}
g sons B
```

Also with pure Tcl is easy to use oo-like calls. Just define structure reference as procedure and forward commands (dispatching) . In this case reference management should be hidden from client.

```
proc $ref {name args} {
    eval $name $ref $args
}
```

Idea of OO



Data structures are hidden for Object user.

OO Programming vs. OO Programming Language

- OO Programming with not OO Language
  - Tk, Tix, GTK (even with heritage)
  - C++ was initially compiled to C
- Not OO Programming with OO Language
  - Anti pattern – Blob, Ghost, many C++ and Java projects.
  - DataSet in .NET, Math libraries (Java, .NET)
- Object orientation is the way to think about programming.

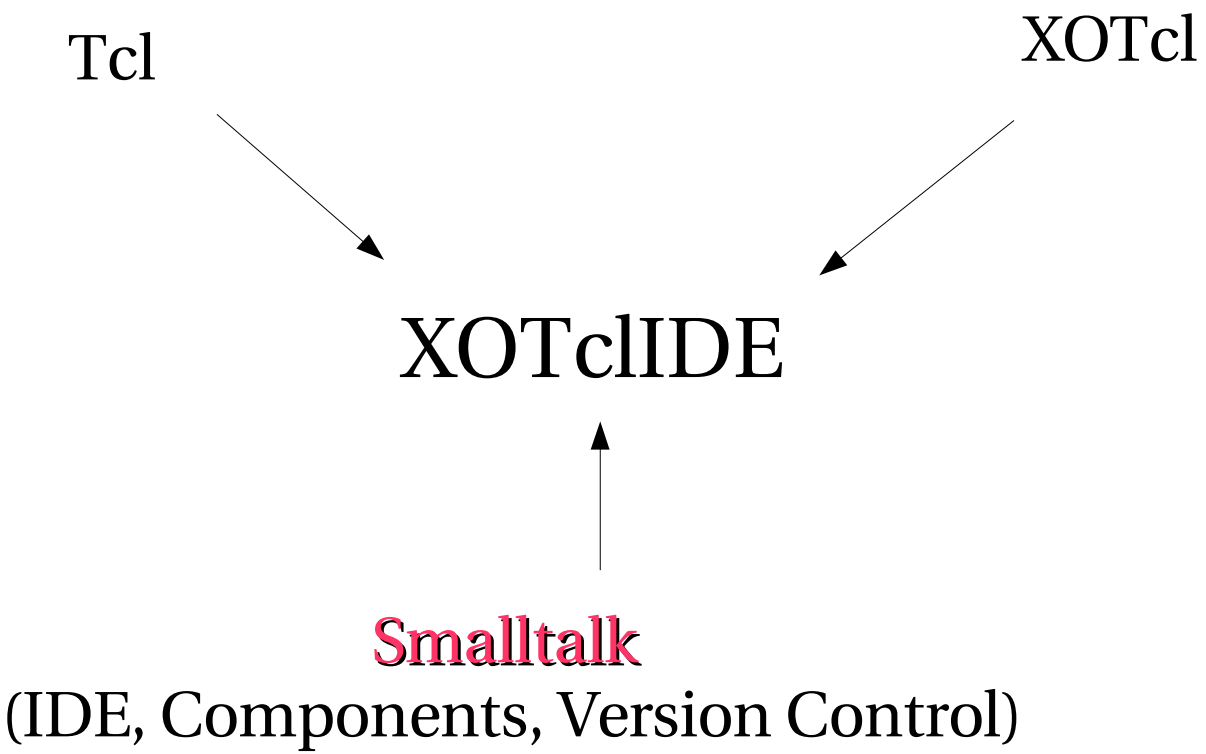
OO in Tcl vs. mainstream OO

- not static typed
- no information hidding (XOTcl)
- more dynamic (introspection=reflection)
- more consistent (class is object – XOTcl)
- better for Aspect Oriented Programming
- better for Pattern (with filters and mixins)
- more similar to Ruby, Smalltalk, Self, Lisp CLOS as to Java, C++, C#

Migrating Tcl – live

- Importing code
- Creating components
- Migrating to XOTcl
- Adaptations
- Tests

XOTclIDE


Tcl
XOTcl
XOTclIDE
Smalltalk
(IDE, Components, Version Control)

XOTclIDE - news

- Starkit support
- New databases (Oracle, Access, sqlite) for version control
- New Plugins
- Usability improvements (by Michael Heka and Fabrice Pardo)
- Supports new XOTcl features.

Advantages of OO

- ? Better and clear structure
- Lifetime support (no memory leaks)
- Good for complex structures (no need for references)
- ? More Productivity and Reuse
- ? Better for complex projects
- Additional Features XOTcl (mixins, filters, delegation, assertions, ...)
- No reinventing the wheel

Disadvantages of OO

- Additional Extension needed
- Less portability (jacl, hecl, jim, tclsharp)
- ? Performance
- Many extensions (XOTcl, ITcl, Snitt, Stoop, Classytcl)

That's all!

Questions ?
Fragen, Anregungen?
Demandoj?

http://www.xdobry.de/xotclIDE