iTcl and TclOO
From the perspective of a simple user

Georgios Petasis
Software and Knowledge Engineering Laboratory,
Institute of Informatics and Telecommunications,
National Centre for Scientific Research “Demokritos”,
Athens, Greece
petasis@iit.demokritos.gr
Overview

- The shock of Tcl 8.6
- Porting existing code to TclOO
- Case study: the Ellogon NLP platform
-iTcl facilities in TclOO
- Organisation of classes in Ellogon
- Creating an Annotation Tool
- Concatenating Dialogs
- Conclusions
Assume an application that uses Itcl
- What happens if it is run under ActiveTcl 8.6 beta?
Ok, this happens as iTcl 3.4 is loaded in 8.6

- Lets compile Tcl from sources (CVS HEAD 27/Jul/10)
- iTcl now contains a new iTcl implementation (4.0b4)
Ok, iTcl 4.0 has a problem with a variable
- Lets “correct” this
iTcl and Tcl 8.6

- iTcl object variables not supported?
  - It seems no...
    ✓ but, object naming was internal
- iTcl 4.0 has been actively maintained!
  - Significant progress since last test (6-8 months ago)
  - Does not crash
  - A few “rough edges” remain
- But:
  - Support for iTcl object variables seems missing

- Status of iTcl next generation?
  - Unknown. Not working either in previous tests
Alternatives for running the application?

- Wait until iTcl 4.0 is ready
  - Will it support 3.4 object variable naming?

- Port the code from iTcl to TclOO
  - Hm, 41 classes? ~20,000 lines of code?
    ✓ Such a task needs to be automated

- Stick to Tcl 8.5 and iTcl 3.4
- But what happens with open source applications?
Overview

- The shock of Tcl 8.6
- Porting existing code to TclOO
- Case study: the Ellogon NLP platform
- iTcl facilities in TclOO
- Organisation of classes in Ellogon
- Creating an Annotation Tool
- Concatenating Dialogs
- Conclusions
Porting from iTcl to TclOO (1)

- A medium sized application: Ellogon
  - Open source (LGPL), http://www.ellogon.org
  - Sticking to Tcl 8.5 is not an option
  - But ~480 iTcl classes need to be ported!
    - Different “variable” syntax
    - The “my” keyword when calling methods
    - Different method exporting convention
    - ...
    - Where is TclOO documentation?
Porting from iTcl to TclOO (2)

- Largely a manual (and time consuming) effort
  - A helper Tcl script to perform “easy” substitutions
  - Several months were needed
  - But, a few portions could not be ported
    - TclOO has some limitations
      - Or do I have a bad programming style? 😊

- The task is now largely finished
  - And the helper Tcl script got quite complex

- And what about other applications?
  - How about turning the conversion script into an iTcl emulator?
Porting from iTcl to TclOO (3)

- The distance of a script that reformats code from an emulator is a simple "eval\”

- I have created a small package that emulates iTcl – 630 lines of code – Ignores less essential features (like protection)

- The goal is to get my applications running

11 Oct 2010

iTcl & TclOO: From the perspective of a simple user
Porting from iTcl to TclOO (4)

- A quick and simple approach actually
- Test application executes further than latest iTcl 4.0
- iTcl 3.x object variable references ("@itcl ...") are converted to TclOO equivalent

But:
- Not all code substitutions are performed
  - Adding the "my" keyword to existing code is tricky
  - 4 regular expressions are not enough to handle this
    - A package that "parses" Tcl is not available
  - Finally I gave up
    - no regular expressions for some cases
Porting: Differences (1)

Most notable differences between the two extensions:

- No configure/cget on TclOO objects
- No common variables across objects of the same class in TclOO
- No “static” class methods (methods that do not require an object to be called) in TclOO
- Different semantics for variables
- A specific method in the classes hierarchy of an object cannot be called in TclOO
- TclOO requires the keyword “my” while calling methods from inside of an object
Porting: Differences (2)

Most notable differences between the two extensions:

- TclOO automatically exports methods that start with a lowercase letter
- No facility for “local” to procedures objects (like itcl::local) in TclOO
## Porting: Similarities

<table>
<thead>
<tr>
<th>iTcl</th>
<th>TclOO</th>
</tr>
</thead>
<tbody>
<tr>
<td>method</td>
<td>my method</td>
</tr>
<tr>
<td>$this</td>
<td>[self]</td>
</tr>
<tr>
<td>chain</td>
<td>next</td>
</tr>
<tr>
<td>itcl::scope</td>
<td>my varname</td>
</tr>
<tr>
<td>inherit</td>
<td>superclass</td>
</tr>
<tr>
<td>itcl::body</td>
<td>oo::define body</td>
</tr>
</tbody>
</table>

### Interesting features of TclOO

- Everything subclasses oo::object
- "mixin"s
- "unknown" on objects
- The "my" keyword
Overview

- The shock of Tcl 8.6
- Porting existing code to TclOO
- Case study: the Ellogon NLP platform
- iTcl facilities in TclOO
- Organisation of classes in Ellogon
- Creating an Annotation Tool
- Concatenating Dialogs
- Conclusions
Case study: the Ellogon NLP platform

- Ellogon is an infrastructure for natural language processing
  - Provides facilities for managing corpora
  - Provides facilities for manually annotating corpora
  - Provides facilities for loading processing components, and apply them on corpora

- Development started in 1998
  - I think with Tcl/Tk 8.1
  - ~500,000 lines of C/C++/Tcl code
  - A lot of legacy code, especially in the GUI
    - No widespread use of tile/ttk
    - No OO (i.e. iTcl) in most parts of the code
Ellogon: plug-ins in many programming languages

CDM API

Direct Link
Java VM
Tcl VM
Perl VM
Python VM

C/C++ Components
Java Components
Tcl Components
Perl Components
Python Components

iTcl & TclOO: From the perspective of a simple user
The roadmap for Ellogon 2.0

The goals for Ellogon 2.0 are:

- Make Ellogon’s core thread safe (done)

- Make Ellogon multi-threaded (feasible?)
  - How Ellogon & the Tcl thread model can cooperate?

- Modernise GUI (using OO and ttk widgets)
  - ~30% completed
  - Initially written in iTcl, now ported to TclOO
  - Includes a complete rewrite of Annotation Tools of Ellogon
Annotation tools is a very demanding area

- A lot of tasks that need annotated corpora
- Each task, may have its own annotation scheme
- Each group, may pose different requirements for the tool

The first generation of tools was:

- coded in plain Tcl/Tk
- difficult to adapt/extend
This is a demo of Ellogon. It is a multi-lingual, cross-platform, general-purpose text engineering environment. Ellogon provides a powerful infrastructure for managing textual data. Embed text processing components and manage them with Ellogon. Some key features include full Unicode support, extensive multi-lingual graphical user interface and modular architecture.

iTcl & TclOO: From the perspective of a simple user

13 Oct 2010

22
First generation tools (2)

iTcl & TclOO: From the perspective of a simple user
First generation tools (3)

iTcl & TclOO: From the perspective of a simple user
Second generation tools (1)

KABUL, Afghanistan-Insurgents carrying rockets and grenades launched a brazen pre-dawn attack on a giant U.S.-run base north of Afghanistan’s capital on Wednesday, leaving at least 10 guerrillas dead and 7 foreign troops wounded.

The assault started when security personnel noticed one of the attackers wearing a suicide vest in a car outside the base, NBC News Correspondent Tom Aspell in Kabul reported.

"It looks like the attackers were trying to crash in through one of the main gates," he said.

Afghan troops killed seven of the estimated 20 militants, Aspell said, and the assault was over by midday.

The Bagram attack came one day after a suicide bomber struck a U.S. convoy in Kabul, killing 18 people. The Kabul dead included five American troops and a Canadian.
Second generation tools (2)

KABUL, Afghanistan - Insurgents carrying rockets and grenades launched a brazen pre-dawn attack on a giant U.S.-run base north of Afghanistan's capital on Wednesday, leaving at least 10 guerrillas dead and 7 foreign troops wounded.

The attack on Bagram air base, about an hour's drive north of Kabul, continued into daylight with sporadic fire of rockets and small arms outside. One rocket landed inside the base, causing minor damage, but no insurgents managed to get inside Bagram, according to NATO.

Helicopter gunships hovered above Bagram, the main base for the U.S.-led troops in Afghanistan with the largest airfield in the country. It was used by the former Soviet Union during its invasion of the country in the 1980s. advertisement I your ad here

The assault started when security personnel noticed one of the attackers wearing a suicide vest in a car outside the base, NBC News Correspondent Tom Aspell in Kabul reported.

"It looks like (the attackers) were trying to crash in through one of the main gates," he said.

Afghan troops killed seven of the estimated 20 militants, Aspell said, and the assault was over by midday.

Taliban offensive

The Bagram attack came one day after a suicide bomber struck a U.S. convoy in Kabul, killing 18 people. The Kabul dead included five American troops and a Canadian.

iTcl & TclOO: From the perspective of a simple user
KABUL, Afghanistan - Insurgents carrying rockets and grenades launched a brazen pre-dawn attack on a giant U.S.-run base north of Afghanistan’s capital on Wednesday, leaving at least 10 guerrillas dead and 7 foreign troops wounded.

The attack on Bagram air base, about an hour’s drive north of Kabul, continued into daylight with sporadic fire of rockets and small arms outside. One rocket landed inside the base, causing minor damage, but no insurgents managed to get inside Bagram, according to NATO.

Helicopter gunships hovered above Bagram, the main base for the U.S.-led troops in Afghanistan with the largest airfield in the country. It was used by the former Soviet Union during its invasion of the country in the 1980s.

The assault started when security personnel noticed one of the attackers wearing a suicide vest in a car outside the base, NBC News Correspondent Tom Aspell in Kabul reported.

"It looks like (the attackers) were trying to crash in through one of the main gates," he said.

Afghan troops killed seven of the estimated 20 militants, Aspell said, and the assault was over by midday.

Taliban offensive

The Bagram attack came one day after a suicide bomber struck a U.S. convoy in Kabul, killing 18 people. The Kabul dead included five American troops and a Canadian.

Overlapping areas: 0

Collection:(C:/Users/George/Collections/news articles collection_240610), Document(51345.txt)
Second generation tools (4)

KABUL, Afghanistan — Insurgents carrying rockets and grenades launched a brazen pre-dawn attack on a giant U.S.-run base north of Afghanistan’s capital on Wednesday, leaving at least 10 guerrillas dead and 7 foreign troops wounded.

The attack on Bagram air base, about an hour’s drive north of Kabul, continued into daylight with sporadic fire of rockets and small arms outside. One rocket landed inside the base, causing minor damage, but no insurgents managed to get inside Bagram, according to NATO.

Helicopter gunships hovered above Bagram, the main base for the U.S.-led troops in Afghanistan with the largest airfield in the country. It was used by the former Soviet Union during its invasion of the country in the 1980s. advertisement I your ad here

The assault started when security personnel noticed one of the attackers wearing a suicide vest in a car outside the base, NBC News correspondent Tom Aspell in Kabul reported.

"It looks like (the attackers) were trying to crash in through one of the main gates," he said.

Afghan troops killed seven of the estimated 20 militants, Aspell said, and the assault was over by midday.

Taliban offensive

The Bagram attack came one day after a suicide bomber struck a U.S. convoy in Kabul, killing 18 people. The Kabul dead included five American troops and a Canadian.

Overlapping areas: 0

Collection:(C:/Users/George/Collections/news articles collection_240610), Document:(51345.txt)

iTcl & TclOO: From the perspective of a simple user

13 Oct 2010 28
Second generation tools (5)

iTcl & TclOO: From the perspective of a simple user
Second generation tools (6)

iTcl & TclOO: From the perspective of a simple user
Second generation tools (7)

iTcl & TclOO: From the perspective of a simple user
iTcl & TclOO: From the perspective of a simple user
Second generation tools (9)

iTcl & TclOO: From the perspective of a simple user
iTcl & TclOO: From the perspective of a simple user
iTcl & TclOO: From the perspective of a simple user

13 Oct 2010
iTcl & TclOO: From the perspective of a simple user
Overview

- The shock of Tcl 8.6
- Porting existing code to TclOO
- Case study: the Ellogon NLP platform
- iTcl facilities in TclOO
- Organisation of classes in Ellogon
- Creating an Annotation Tool
- Concatenating Dialogs
- Conclusions
Ellogon TclOO classes: cget/configure

- **Add cget/configure on all classes**
  - No need for a complex implementation of configure/cget
  - I only use them to get/set variable values

- **Very easy to add new methods on all objects!**
  - Everything is a child of `oo::object`
  - Simple implementation

```tcl
oo::define oo::object method cget {_elep_oo_variable_name} {
  # Remove the preceding "-_" character...
  set _elep_oo_variable_name [string range $_elep_oo_variable_name 1 end]
  switch -glob -- $_elep_oo_variable_name {
    *(*) {
      # This is an array index!
      lassign [split $_elep_oo_variable_name ()] \
        _elep_oo_variable_array _elep_oo_variable_key
      my variable $_elep_oo_variable_array
    }
    default {
      my variable $_elep_oo_variable_name
    }
  }
  return [set $_elep_oo_variable_name]
};# cget
```

**iTcl & TclOO: From the perspective of a simple user**
Ellogon TclOO classes: common (1)

- TclOO has another trick:
  - Procedure oo::define::<name> extends oo::class
    ✓ Implementing ::oo::define::common allows to use the keyword “common” during class creation

```tcl
proc ::oo::define::common {varname args} {
    if {[llength $args] > 1} { ... }
    # Get the name of the current class
    set cls [lindex [info level -1] 1]
    oo::define $cls self export varname; # Export method varname
    # Initialise the variable
    if {[llength $args]} {
        set [$cls varname $varname] [lindex $args 0]
    }
};# ::oo::define::common
```

iTcl & TclOO: From the perspective of a simple user
But common also needs a method to be called from methods accessing common variables

```tcl
oo::define oo::object method common {args} {
    if {![llength $args]} return
    set callclass [lindex [self caller] 0]
    oo::define $callclass self export varname
    foreach vname $args {
        lappend pairs [$callclass varname $vname] $vname
    }
    uplevel 1 upvar {*$}pairs
    };# common
```

- Common and my cget/configure do not mix
## Define "classmethod"

```tcl
proc ::oo::define::classmethod {name {args ""} {body ''}} {
    # Code from: http://wiki.tcl.tk/21595#pagetoce30e53a1
    set argc [llength [info level 0]]
    if {$argc == 4} {
        uplevel 1 [list self method $name $args $body]
    } elseif {$argc == 3} {
        return -code error "..."
    }

    # Get the name of the current class
    set cls [lindex [info level -1] 1]
    # Get its private "my" command
    set my [info object namespace $cls]::my
    # Make the connection by forwarding
    tailcall forward $name $my $name
};
```

iTcl & TclOO: From the perspective of a simple user 13 Oct 2010
What about inheritance?

```tcl
oo::class create ELEP::Base::Utilities {
    classmethod userAppDir {} {...}
}
```

```tcl
oo::class create ELEP::System::System {
    superclass ELEP::Base::Utilities
    classmethod systemConfigurationDir {} {
        return [my userAppDir]/Systems/Config
    };
    # systemConfigurationDir
```

unknown method "userAppDir"
Overview

- The shock of Tcl 8.6
- Porting existing code to TclOO
- Case study: the Ellogon NLP platform
- iTcl facilities in TclOO
- Organisation of classes in Ellogon
- Creating an Annotation Tool
- Concatenating Dialogs
- Conclusions
Widget classes (1)

- In Ellogon, I don’t think in terms of Tk widgets
  - In fact, I totally ignore them
- Only 3 classes available, which represent widgets
  - Toplevel, Dialog, Widget, RibbonToplevel
    - RibbonToplevel has a Windows Ribbon instead of a menu
- Some common methods for all classes
  - getToplevel
  - getToplevelObject
  - getClientArea
- Automatic variables
  - win for toplevel/dialogs
  - widget for widgets
Widget classes (2)

- Widgets are destroyed when objects are deleted, and vise versa
  - In a way similar to iTk

- Toplevel/Dialogs generate widgets based on the object names

- Objects of the Widget class need the widget type and name
  - i.e. Widget ttk::button .button ?args?

iTcl & TclOO: From the perspective of a simple user
Ellogon building blocks

- Many building blocks that inherit Widget
  - Only the Tk widget that will contain the block is required (the “parent”)
  - i.e. ButtonAnnotator, 1-Click selector, TemplateFiller, TextViewer, HTMLViewer, AllignedTextViewer, etc.

- A generic class that represents an Annotation tool
  - Inherits from Toplevel
  - Splits client area into two columns, separated by a ttk::panedwindow

- All tools, subclass this class, add another layout if required, and create/place building block objects
Overview

- The shock of Tcl 8.6
- Porting existing code to TclOO
- Case study: the Ellogon NLP platform
- iTcl facilities in TclOO
- Organisation of classes in Ellogon
- Creating an Annotation Tool
- Concatenating Dialogs
- Conclusions
Creating an Annotation Tool

- Gluing building blocks is easy, but what about the user experience?

- Let's see an example, by creating an Annotation tool that annotates a document with a semantic model (i.e. an ontology)
  - For this task, the bits required are:
    1. An annotator to annotate “properties” found in the text
    2. An annotator to group properties into objects
    3. An annotator to group objects into other objects
Annotating properties: the button annotator (1)
Annotating properties: the button annotator (2)

- Cooperates with a viewer (text, HTML, Aligned text, Aligned HTML) and allows the user to annotate the selected text with one or more properties

- The annotation schema is dynamic
  - Method createSpecificationSelectorObject()
    ✓ Creates an AnnotationSpecificationSelector object
  - Method show()
    ✓ Calls AnnotationSpecificationSelector.show() and waits for an answer
  - Various schemas are read from an XML file, and presented to the user
  - Button annotator adapts to the selected schema
Grouping properties/objects

- The TemplateFiller annotator
- Again presents a dynamic schema
- Now method createSpecificationSelectorObject() - Creates an AnnotationAndTemplateSpecificationSelector object
- How easy is to mix the two annotators?
  - Easy, just create the two objects and place them on a single annotation tool
- Any disadvantages?
  - Yes. The user gets two dialogs for configuring a single tool!
The multiple dialog issue

How can this be resolved?

- A new class must be created, which is the concatenation of the two configuration dialogs
- The two objects must somehow create and use the same configuration object
Overview

- The shock of Tcl 8.6
- Porting existing code to TclOO
- Case study: the Ellogon NLP platform
- iTcl facilities in TclOO
- Organisation of classes in Ellogon
- Creating an Annotation Tool
- Concatenating Dialogs
- Conclusions
In iTcl was very easy:
- Create a new class that inherits the two configuration objects
- iTcl has the ability to call explicitly methods from the class hierarchy:
  ✓ Method populateDialogFrame() just creates two ttk::labelframe and calls populateDialogFrame() of the two inherited classes with the proper parent frame.

In TclOO the task is far more complex!
- You cannot simply inherit both classes
The best alternative?

- Create a new class that behaves as both configuration selectors, and drives instances of the two selectors internally
  - The new class must have all methods of the two objects
  - The new class must have all the public variables of both objects (so as cget/configure to work)
Exposing variables of contained objects

1. Declare all variables as “automatic”
   - i.e. in class, with the “variable” keyword
2. Use “upvar” to link variables between two objects

```tcl
#oo::class create ELEP::ViewerBase::AnnotationAndTemplateSpecificationSelector {
    variable dialog_window ann_selector templ_selector 
    language annotation attribute alternative groups values 
    template tables
    method init {} {
        set ann_selector [ELEP::ViewerBase::AnnotationSpecificationSelector 
                          create [::ELEP::Base::auto ::ELEP::ViewerBase::] 0]
        set templ_selector [ELEP::ViewerBase::TemplateSpecificationSelector 
                          create [::ELEP::Base::auto ::ELEP::ViewerBase::] 0]
        oo::objdefine $ann_selector export varname
        foreach var {language annotation attribute alternative groups values} {
            upvar [$ann_selector varname $var] [my varname $var]
        }
        oo::objdefine $templ_selector export varname
        foreach var {template tables} {
            upvar [$templ_selector varname $var] [my varname $var]
        }
        my createReader
        my createDialog
        my populateDialogFrame [$dialog_window getFrame]
        my restoreState [$dialog_window getFrame]
    };
    # init
```
Exposing methods of contained objects

- Methods from both contained objects must be exposed - through “unknown”

```tcl
method unknown {args} {
    ## Try to call the aggregated objects...
    if {![catch {ann_selector {*}$args} result]} {
        return $result
    }
    if {![catch {templ_selector {*}$args} result]} {
        return $result
    }
    next unknown {*}$args
};# unknown
```
So, are all problems solved?
- No

```tcl
method restoreState {frame} {
    $ann_selector restoreState $frame.annotation
    $templ_selector restoreState $frame.template
};# restoreState
```
What about efficiency?

- Is there a problem using unknown to “distribute” method calls to the proper object?
  - I don’t know, I haven’t measured
  - I assumed that there is a penalty, so I explored alternatives before implementing a similar approach for “merging” Button Annotator & Template Filler

- What I finally did, was to create a new class which
  - Inherits only ButtonAnnotator
  - The various methods of TemplateFiller are copied/extend methods of the new class
    - Thus “next” works, as there is only a linear hierarchy to follow
    - The configuration selector dialog object is single/common
Overview

- The shock of Tcl 8.6
- Porting existing code to TclOO
- Case study: the Ellogon NLP platform
- iTcl facilities in TclOO
- Organisation of classes in Ellogon
- Creating an Annotation Tool
- Concatenating Dialogs
- Conclusions
Mixins
- I have used “mixin”s a few times, but what are really “mixin”s?
  ✓ What happens with colliding method names, the constructor and inheritance?

Inheritance
- How do you inherit from classes whose constructors take different arguments?
- The same issue can occur with plain methods and “next”
- Is “next” limited, and an additional invocation method is required?
Conclusions (1)

- Both iTcl & TclOO have their strengths and weaknesses

- iTcl:
  - Lacked support for unknown
  - I had to use the “@itcl ...” variable naming for serialising objects
  - info method is error-prone
Conclusions (2)

- Both iTcl & TclOO have their strengths and weaknesses

- TclOO:
  - No support for calling a specific class method from the superclasses
  - Variables cannot be initialised without a constructor
  - Are traces supported?
    ✓ Can constructor arguments be recorded?

- Should things like classmethod & common be moved from the wiki to the Tcl core?
Thank you!