Ellogon and the challenge of threads

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 Ellogon NLP platform
- Ellogon architecture and data model
  - Collections and documents
  - Attributes and annotations
- The object cache
- Thread safety and multiple threads
- Conclusions

The Ellogon NLP platform (1)

- 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 applying them on corpora
- Development started in 1998
  - I think with Tcl/Tk 8.1 (beta?)
  - ~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

The Ellogon NLP platform (2)

- Ellogon was amongst the first platforms to offer complete multi-lingual support
  - Of course, it as using Tcl 8.1 ☺
- The first prototype was written entirely in Tcl/Tk
  - Performance was not good, but memory consumption was excellent!

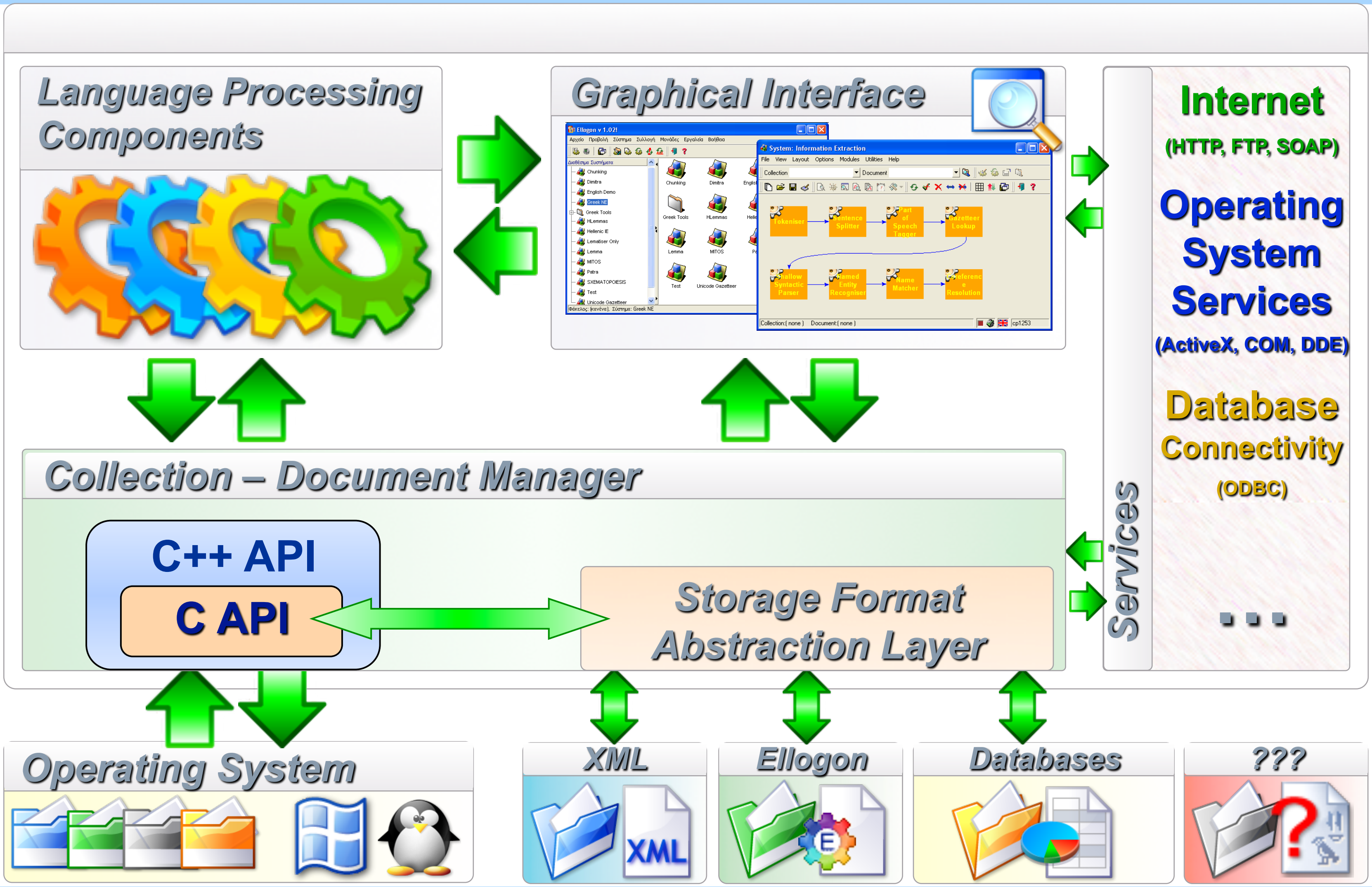
The Ellogon NLP platform (4)

- Too many Tcl objects required (> 10K)
- A solution from observing the data:
  - Objects tend to contain the same information

- Why not build a cache of objects?
  - Objects can be reused as appropriate
- Was it a good solution?
  - Yes, this approach worked well for many years

- But recent hardware brings a new challenge:
  - How can this data model meet multiple threads?

Ellogon Architecture



Language Processing Components

Graphical Interface

Internet

(HTTP, FTP, SOAP)

Operating System Services

(ActiveX, COM, DDE)

Database Connectivity

(ODBC)

...

Services

Collection – Document Manager

C++ API

C API

Storage Format Abstraction Layer

Operating System

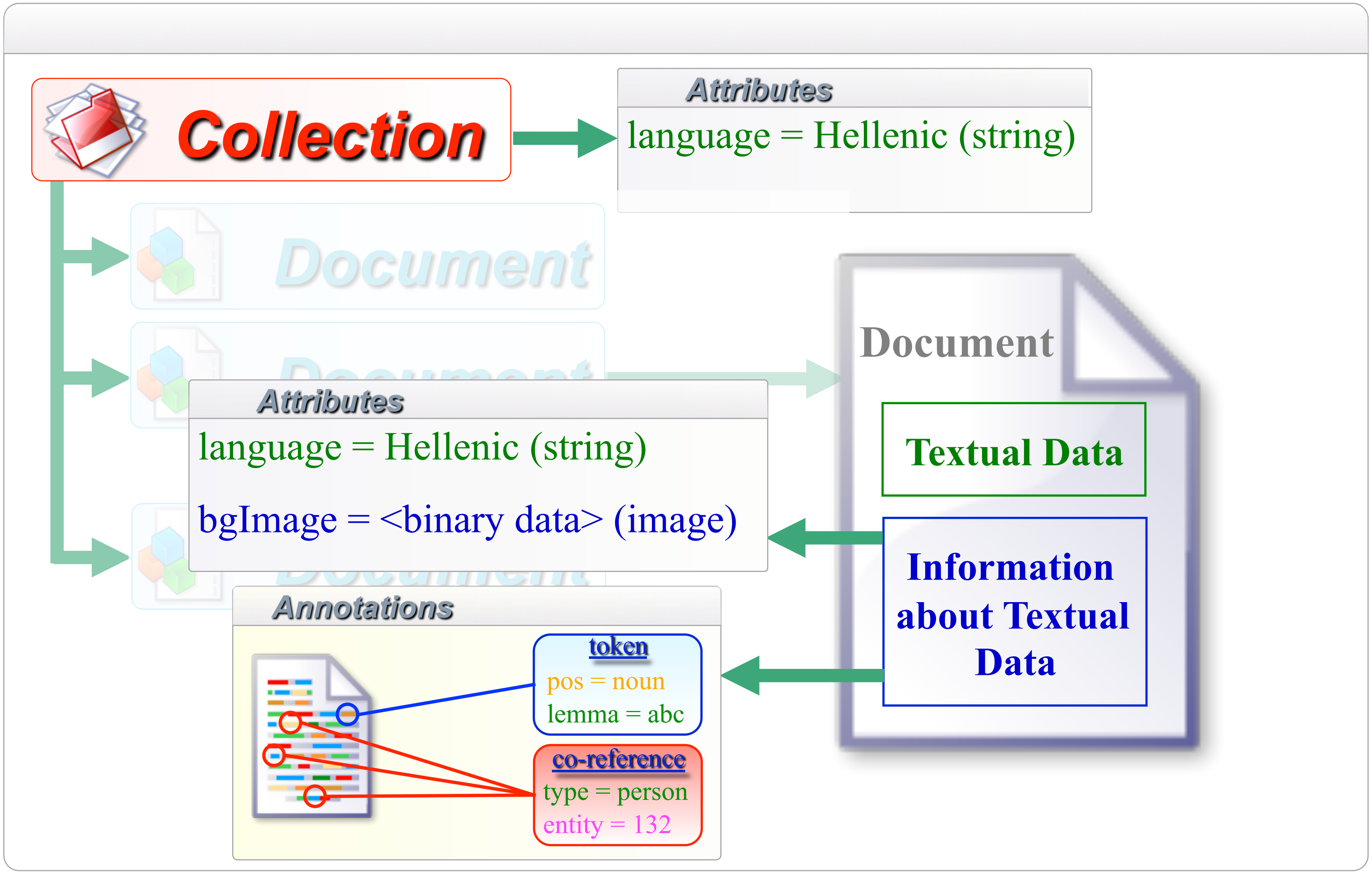
XML

Ellogon

Databases

???

Ellogon Data Model



Collection

Attributes

language = Hellenic (string)

Document

Document

Document

Document

Textual Data

Information about Textual Data

Attributes

language = Hellenic (string)

bgImage = <binary data> (image)

Annotations

token

pos = noun

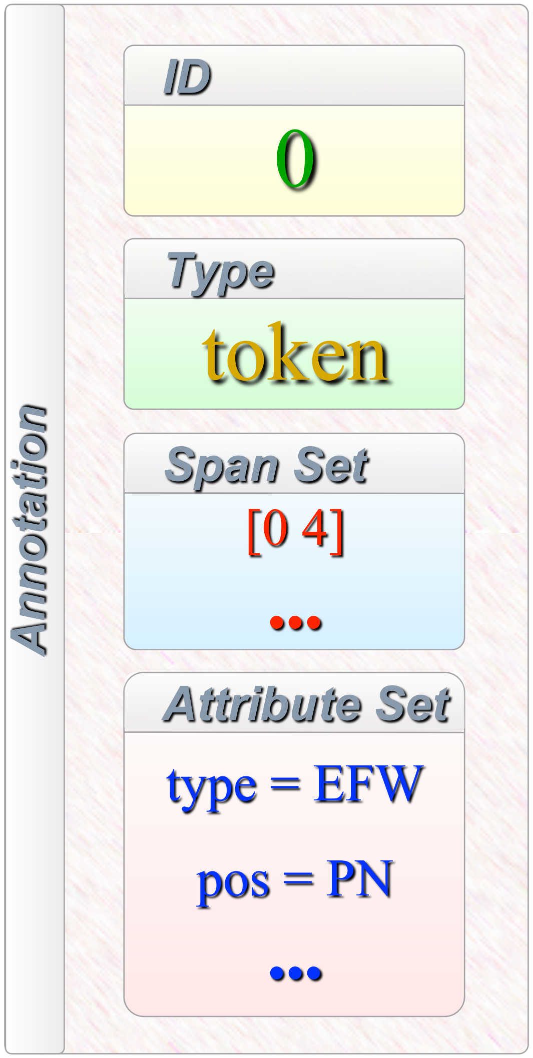
lemma = abc

co-reference

type = person

entity = 132

Annotations



Annotation ID

Unambiguously

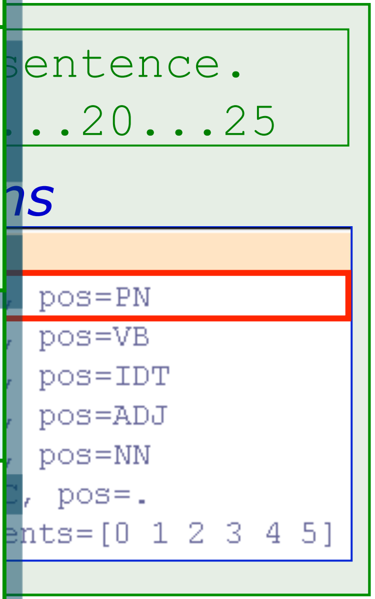
Annotation Type

Classifies annotations into categories

Denotes ranges of annotated textual data

Annotation Attribute Set

Contains linguistic information in the form of named values



The Collection

- A C structure, containing (among other elements):
  - A Tcl list object, containing the documents to be deleted (if any)
  - A Tcl command token, holding the Tcl command that represents the collection at the Tcl level
  - A Tcl Hash table that contains the attributes of the collection. Each attribute is a Tcl list object
  - Two Tcl objects that can hold arbitrary information, such as notes and associated information

The Document

- A C structure, containing (among other elements):
  - A Tcl command token, holding the Tcl command that represents the document at the Tcl level
  - A Tcl Hash table that contains the attributes of the document. Each attribute is a Tcl list object
  - A Tcl Hash table that contains the annotations of the document. Each annotation is either a Tcl list object, or an object of custom type

Attributes

- Each attribute is a Tcl list object, containing three elements:
  - The attribute name: the name can be an arbitrary string
  - The type of the attribute value: this can be an item from a predefined set of value types
  - The value of the attribute, which can be an arbitrary (even binary) string

Annotations

- An annotation is a Tcl object of custom type
- It can be roughly seen as a list of four elements:
  - The annotation id: an integer, which uniquely identifies the annotation inside a document
  - The annotation type: an arbitrary string that classifies the annotation into a category
  - A list of spans: each span is a Tcl list object, holding two integers, the start/end character offsets of the text annotated by the span
  - A list of attributes: a Tcl list object, whose elements are attributes

The object cache

- Ellogon implements a global memory cache for Tcl objects
  - Containing information from all opened collections and documents
- The cache is used when:
  - Creating an element (i.e. attribute, span, annotation, etc.)
  - An annotation/attribute is put in a document
  - A collection/document is loaded

Why is cache important?

- Linguistic information tents to repeat a lot
- Example: annotating a 10.000 word document with a part-of-speech tagger
  - 10.000 “token” annotations
  - Containing 10.000 “pos” attributes
- Assume a tag set of 10 part-of-speech categories
  - Each “pos” value has a potential repetition in the thousands
- Caching “token’ and “pos” makes sense
- Caching larger clusters/constructs of objects makes even more sense
- Sharing objects across document reduces memory consumption further

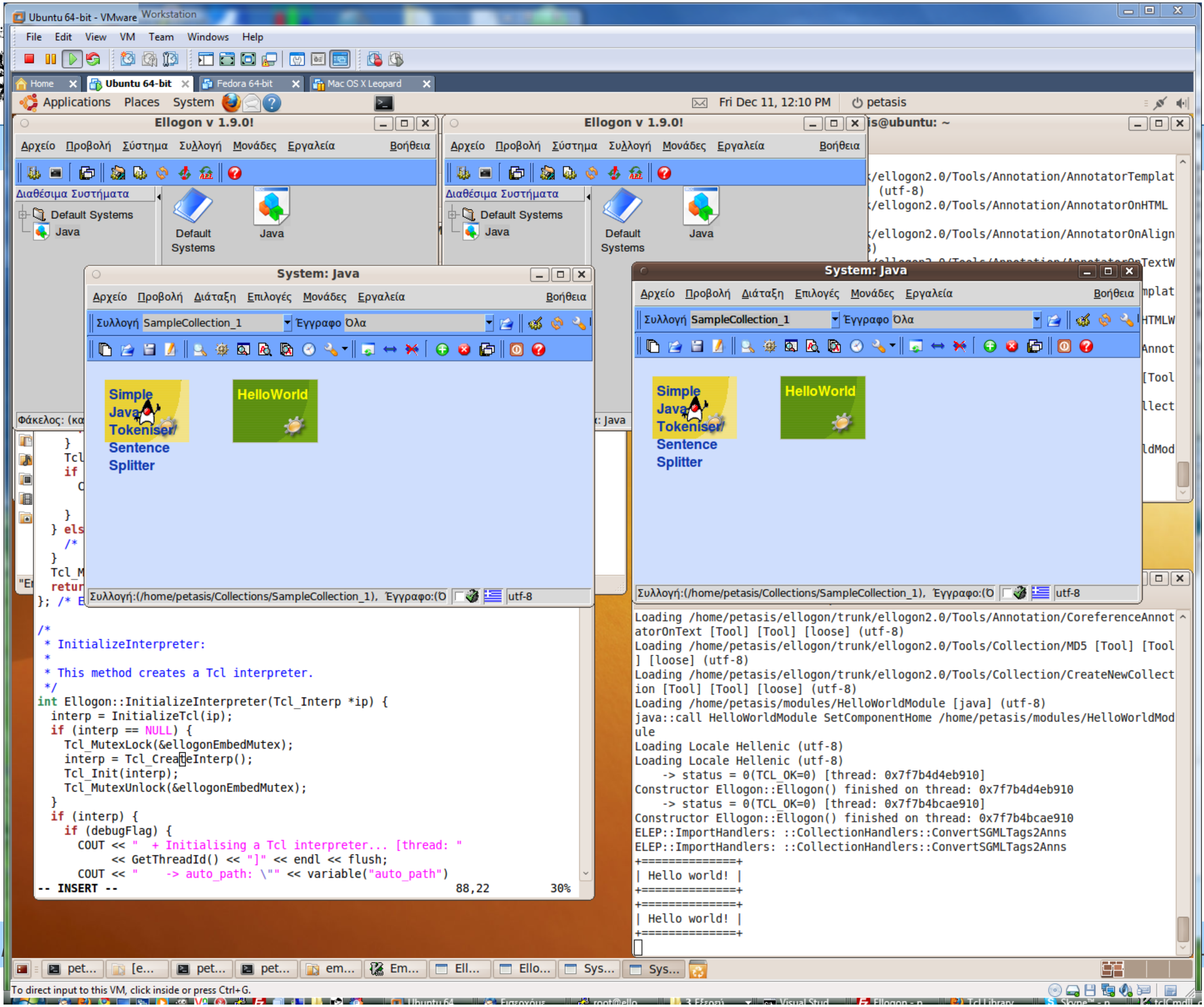
Thread safety (1)

- The object cache is thread “unfriendly”
  - Tcl objects cannot be shared among threads
- Parallel processing of documents is a highly desirable feature
  - But thread-safety is an open question for the Ellogon platform

Thread safety (2)

- The CDM implementing the data model (and the object cache) is already thread-safe:
  - The global variables/objects are few, and their access is protected by mutexes
  - The object cache is global, and protected again with a mutex
  - Ellogon plug-in components use thread-specific storage for storing their “global” variables
    - ✓ Through special pre-processor definitions for C/C++ components

- But thread-safety does not necessarily allow the usage of threads inside Ellogon


Ubuntu 64-bit - VMware Workstation
File Edit View VM Team Windows Help
Home
Ubuntu 64-bit
Fedora 64-bit
Mac OS X Leopard
Applications Places System
Fri Dec 11, 12:10 PM
petasis
Ellogon v 1.9.0!
Αρχείο Προβολή Σύστημα Συλλογή Μονάδες Εργαλεία Βοήθεια
Διαθέσιμα Συστήματα
Default Systems
Java
Default Systems
Java
System: Java
Αρχείο Προβολή Διάταξη Επιλογές Μονάδες Εργαλεία Βοήθεια
Συλλογή SampleCollection_1
Έγγραφο Όλα
Simple Java Tokeniser
Sentence Splitter
HelloWorld
Συλλογή:(/home/petasis/Collections/SampleCollection_1), Έγγραφο:(Ό
utf-8
Φάκελος: (κα
: Java
is@ubuntu: ~
pet...
[e...
pet...
pet...
em...
Em...
Ell...
Ello...
Sys...
Sys...
To direct input to this VM, click inside or press Ctrl+G.

```
    }
    Tcl
    if
      C

    }
  } els
    /*
  }
  Tcl_M
  retur
}; /* E

/*
 * InitializeInterpreter:
 *
 * This method creates a Tcl interpreter.
 */
int Ellogon::InitializeInterpreter(Tcl_Interp *ip) {
  interp = InitializeTcl(ip);
  if (interp == NULL) {
    Tcl_MutexLock(&ellogonEmbedMutex);
    interp = Tcl_CreateInterp();
    Tcl_Init(interp);
    Tcl_MutexUnlock(&ellogonEmbedMutex);
  }
  if (interp) {
    if (debugFlag) {
      COUT << "  + Initialising a Tcl interpreter... [thread: "
           << GetThreadId() << "]" << endl << flush;
      COUT << "    -> auto_path: \"" << variable("auto_path")
-- INSERT --                                              88,22         30%
```

```
k/ellogon2.0/Tools/Annotation/AnnotatorTemplat
 (utf-8)
k/ellogon2.0/Tools/Annotation/AnnotatorOnHTML

k/ellogon2.0/Tools/Annotation/AnnotatorOnAlign
3)
k/ellogon2.0/Tools/Annotation/AnnotatorOnTextW
```

```
Loading /home/petasis/ellogon/trunk/ellogon2.0/Tools/Annotation/CoreferenceAnnot
atorOnText [Tool] [Tool] [loose] (utf-8)
Loading /home/petasis/ellogon/trunk/ellogon2.0/Tools/Collection/MD5 [Tool] [Tool
] [loose] (utf-8)
Loading /home/petasis/ellogon/trunk/ellogon2.0/Tools/Collection/CreateNewCollect
ion [Tool] [Tool] [loose] (utf-8)
Loading /home/petasis/modules/HelloWorldModule [java] (utf-8)
java::call HelloWorldModule SetComponentHome /home/petasis/modules/HelloWorldMod
ule
Loading Locale Hellenic (utf-8)
Loading Locale Hellenic (utf-8)
    -> status = 0(TCL_OK=0) [thread: 0x7f7b4d4eb910]
Constructor Ellogon::Ellogon() finished on thread: 0x7f7b4d4eb910
    -> status = 0(TCL_OK=0) [thread: 0x7f7b4bcae910]
Constructor Ellogon::Ellogon() finished on thread: 0x7f7b4bcae910
ELEP::ImportHandlers: ::CollectionHandlers::ConvertSGMLTags2Anns
ELEP::ImportHandlers: ::CollectionHandlers::ConvertSGMLTags2Anns
+==============+
| Hello world! |
+==============+
+==============+
| Hello world! |
+==============+
```

Can Ellogon become multi-threaded?

- Difficult to be answered
- Requirements are:
  - The graphical user interface must not block during component execution
    - ✓It should be running in its own thread?
  - Each execution chain must run on its own thread
- The documents of a collections should be distributed into N threads
  - And processed in parallel
  - This is a highly desired feature ☺

Obstacles for multiple threads

- The object cache
  - Splitting it in multiple threads increases memory consumption
- The GUI is also a viewer for linguistic data
  - If running in a separate thread, deep copy of objects is required
- Plug-in components in Tcl
  - They frequently short-circuit the “API”, and tread API elements as Tcl lists
    - ✓ It is easier ☺

Conclusions

- Ellogon has been in active development and usage for more than an decade now
- Enhancements are required in order to exploit contemporary hardware better
- However, it is unclear whether threads can be introduced
  - Without a major re-organisation of the platform
  - Without breaking compatibility with plug-in components
- Any suggestions/ideas?

Thank you!