The “Ellogon” Text Engineering Platform…

Developing Ellogon Components…

Georgios Petasis, Vangelis Karkaletsis

SKEL Laboratory

N.C.S.R. “Demokritos”

Overview

- Component Types
- How are Components Created?
- The Anatomy of a Component
- Writing a Wrapper Component
- Writing a Native Component

Component Types

- **Linguistic processing components.**
- **Visualisation components.**
- **Generic Tools.**
- **Components implementing Persistent Storage formats.**
- **Components implementing Collection creation handlers.**

Component Types

- **Linguistic processing components.**
- **Visualisation components.**
- **Generic Tools.**
- **Components implementing Persistent Storage formats.**
- **Components implementing Collection creation handlers.**

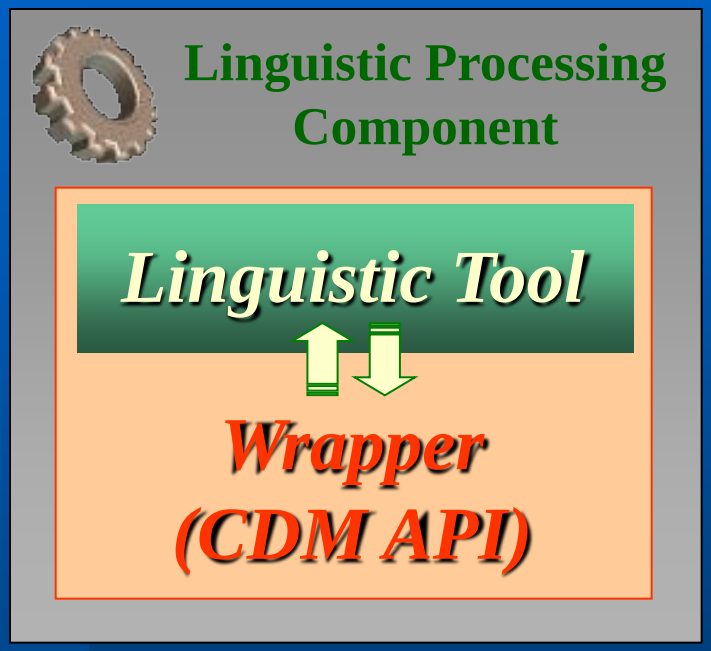
Linguistic Processing Components

- **Two Types:**
  - **Wrapper Components**
    - **For embedding existing tools into Ellogon.**
  - **Native Components**
    - **For creating new tools for use with Ellogon.**

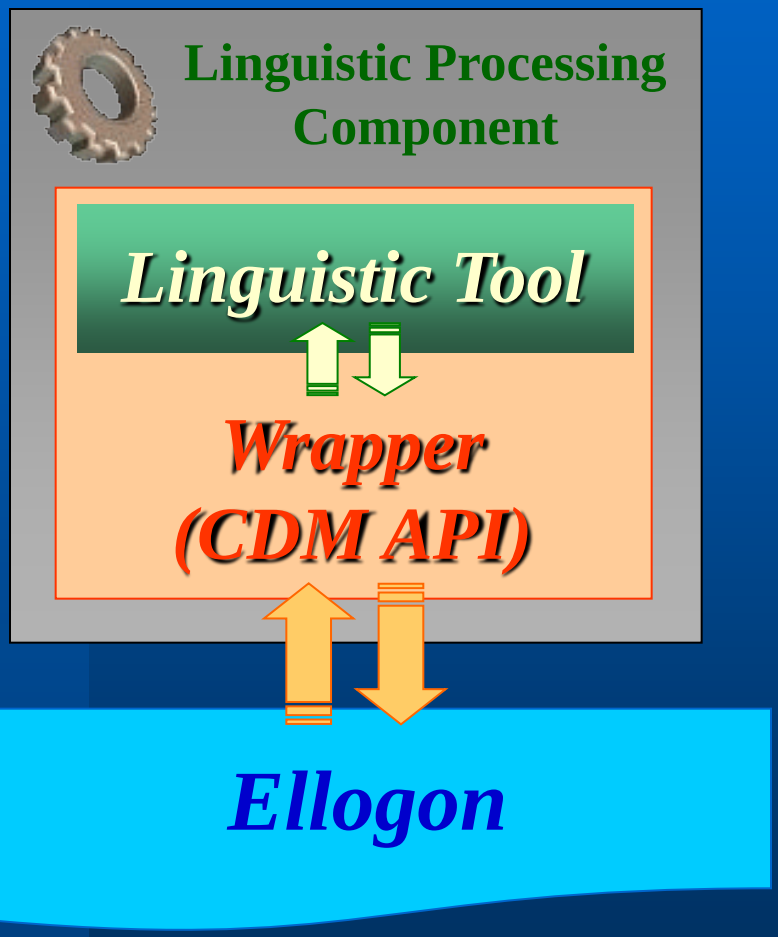
Wrapper Components

Linguistic Tool

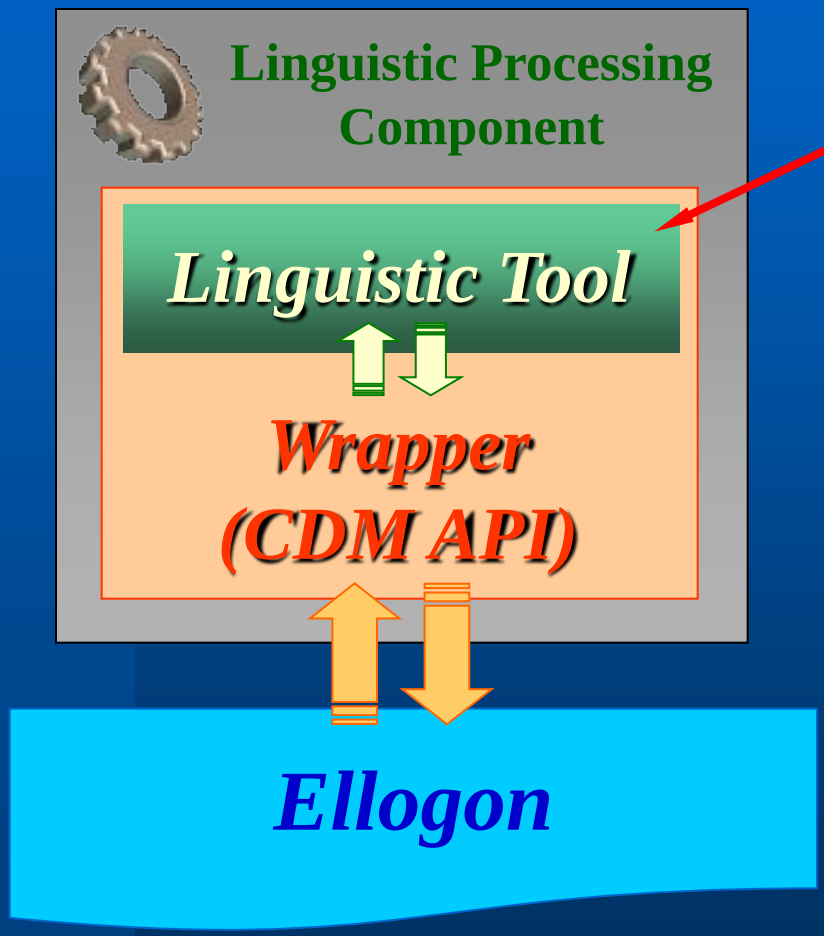
Wrapper Components



Wrapper Components



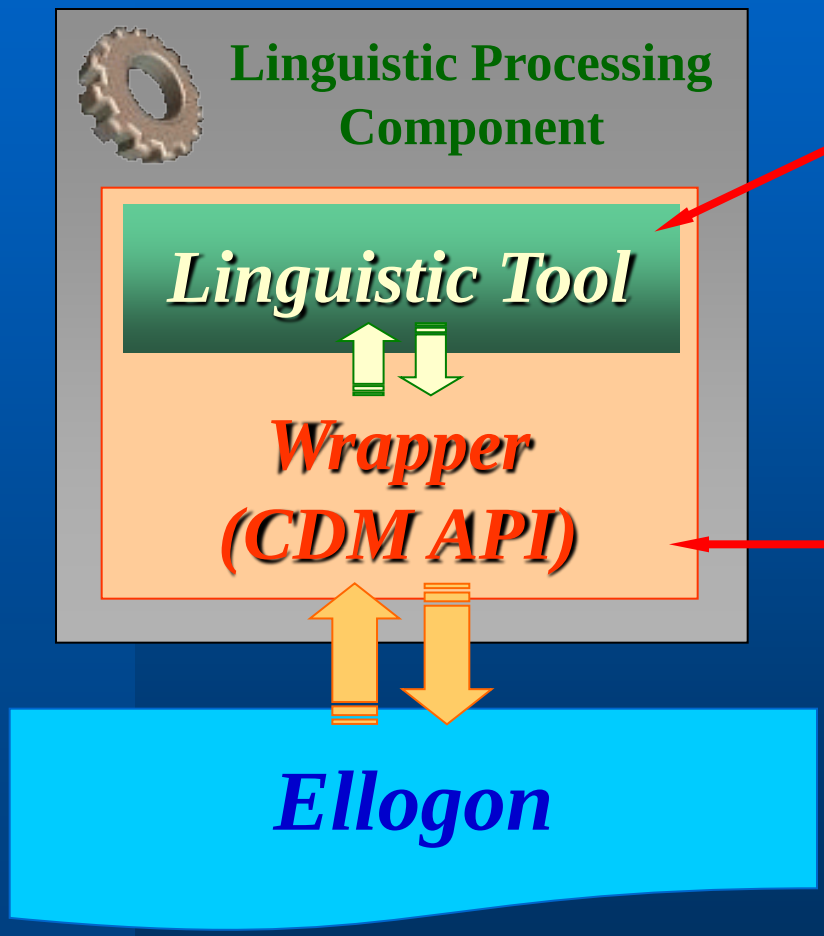
Wrapper Components



Linguistic Tool

This component part is responsible for the linguistic processing.

Wrapper Components



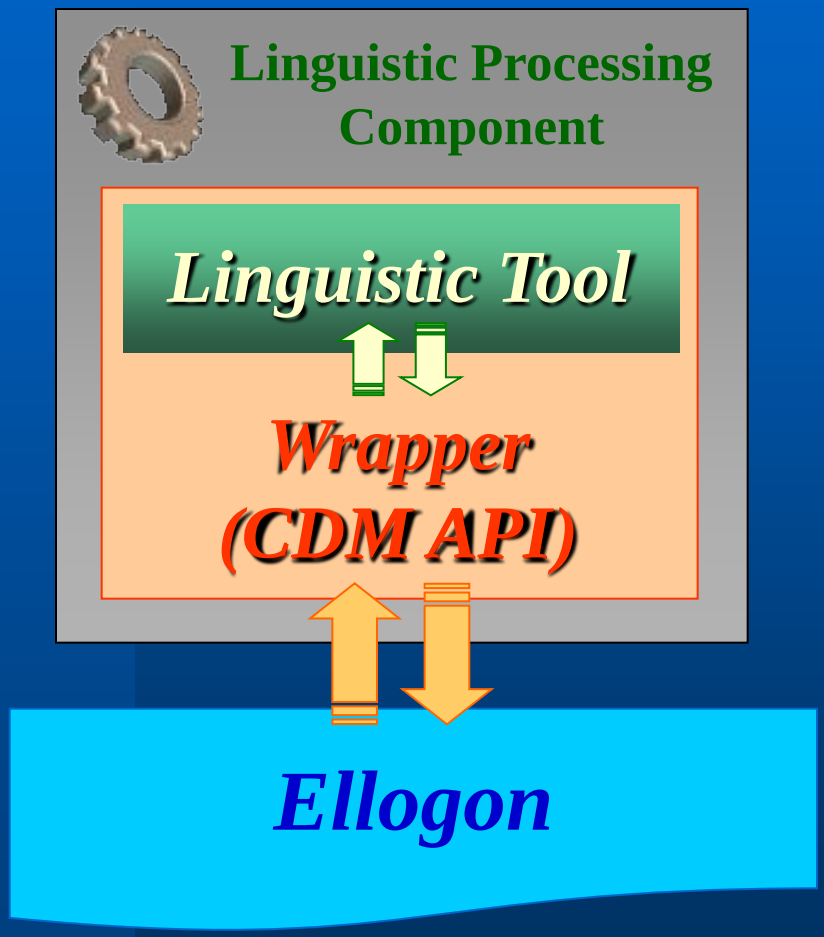
Linguistic Tool

This component part is responsible for the linguistic processing.

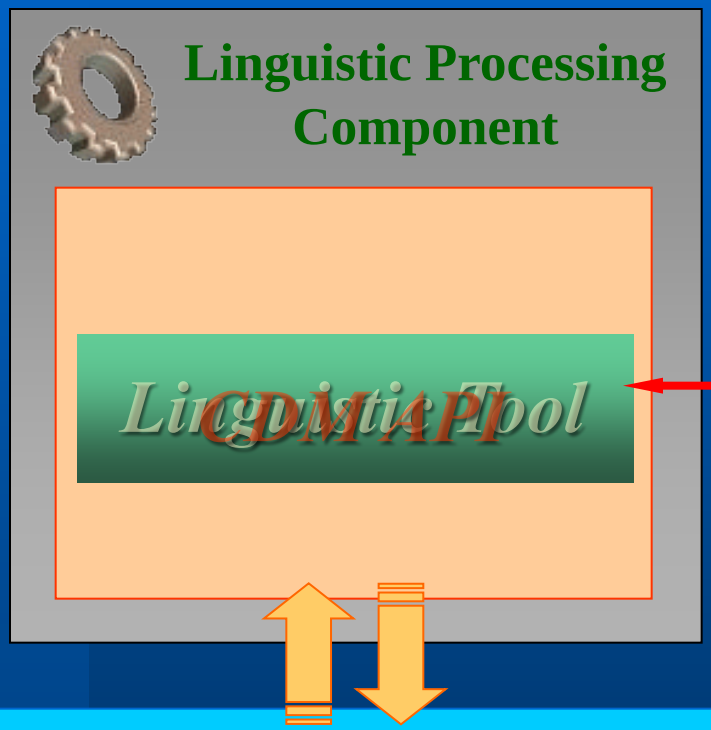
Wrapper

This component part is responsible for interfacing the linguistic processing with Ellogon.

Native Components



Native Components



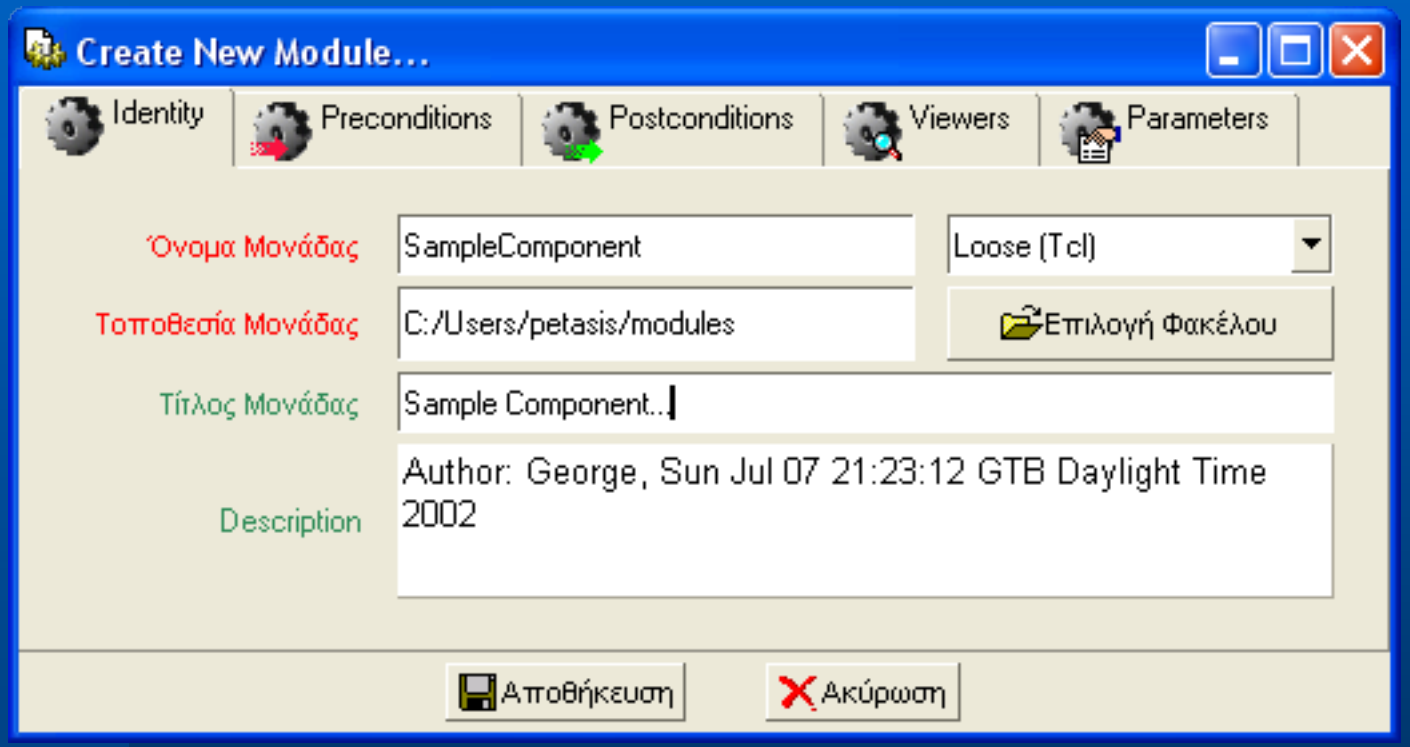
Native Component

The tool interfaces directly with Ellogon.

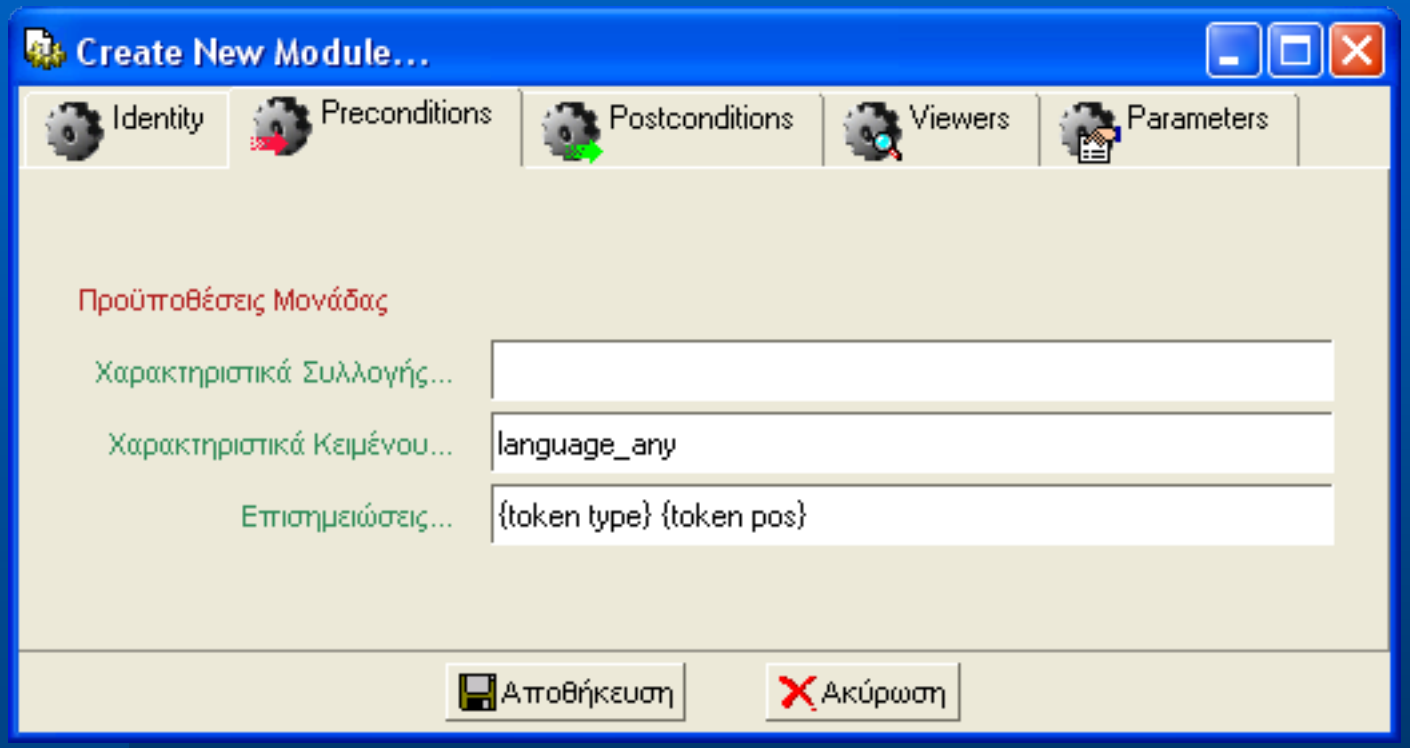
Overview

- Component Types
- How are Components Created?
- The Anatomy of a Component
- Writing a Wrapper Component
- Writing a Native Component

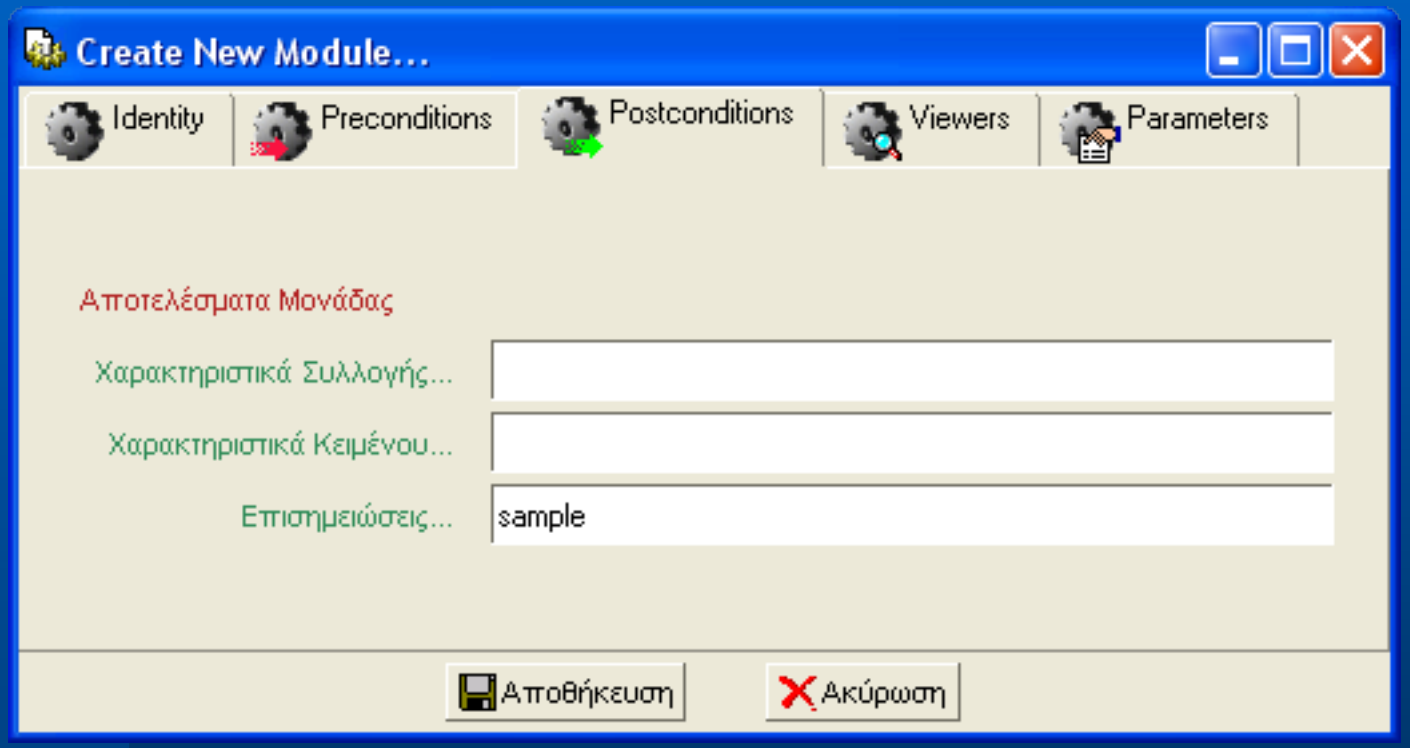
Creating Components



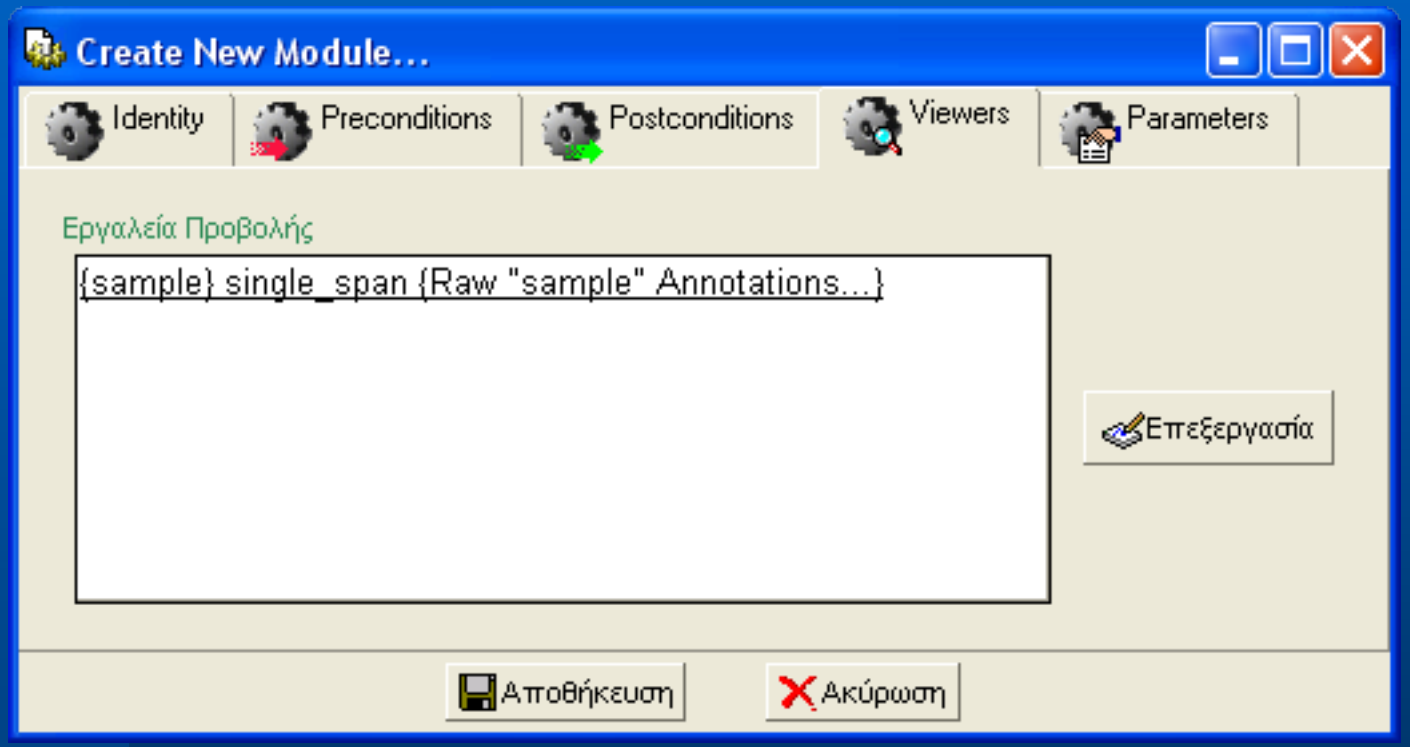
Creating Components



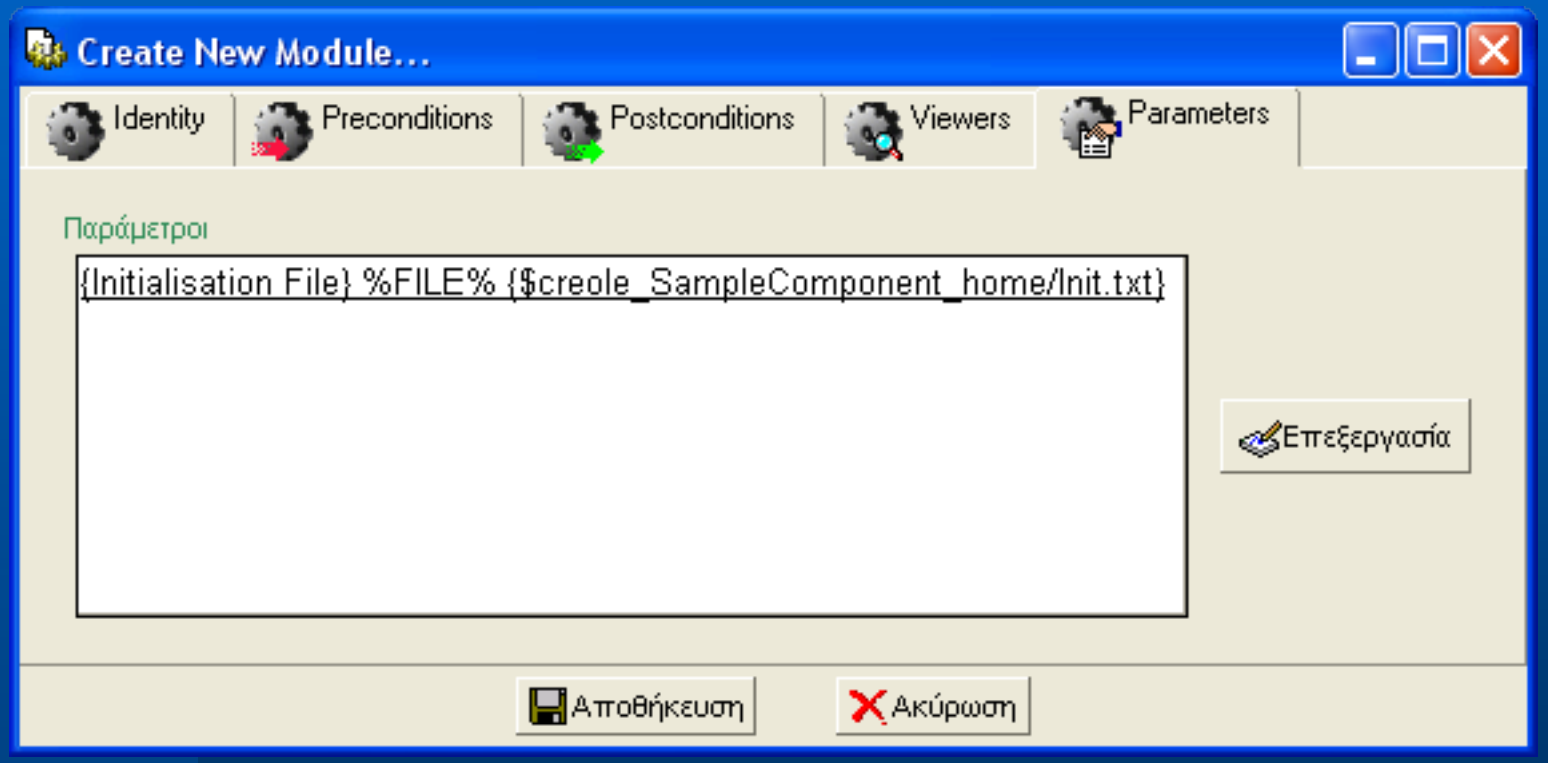
Creating Components



Creating Components



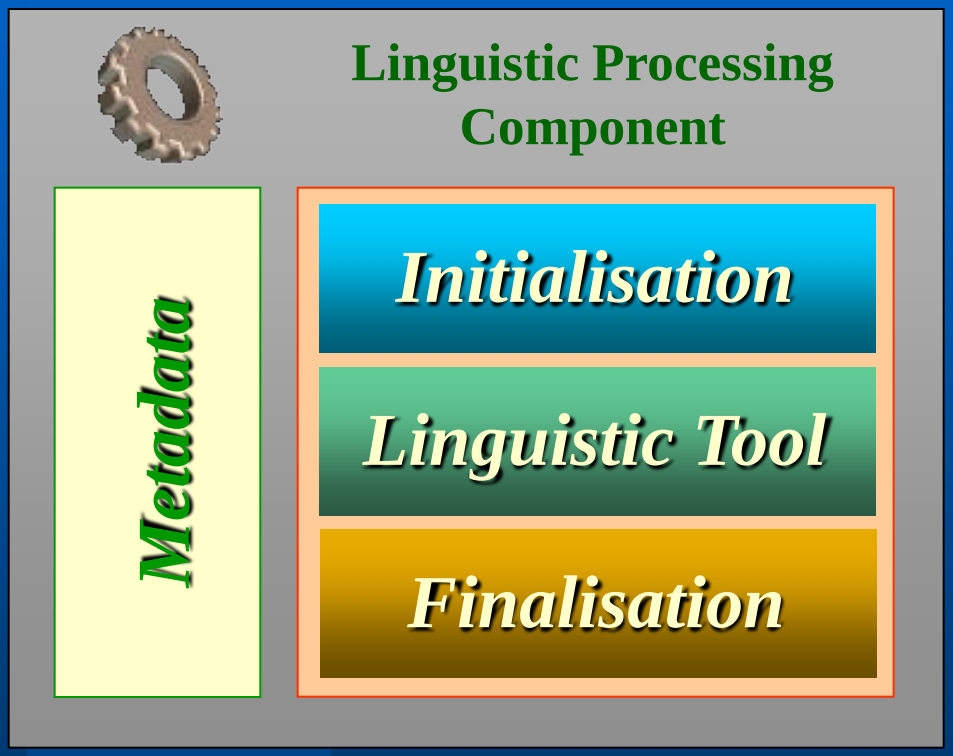
Creating Components



Overview

- Component Types
- How are Components Created?
- The Anatomy of a Component
- Writing a Wrapper Component
- Writing a Native Component

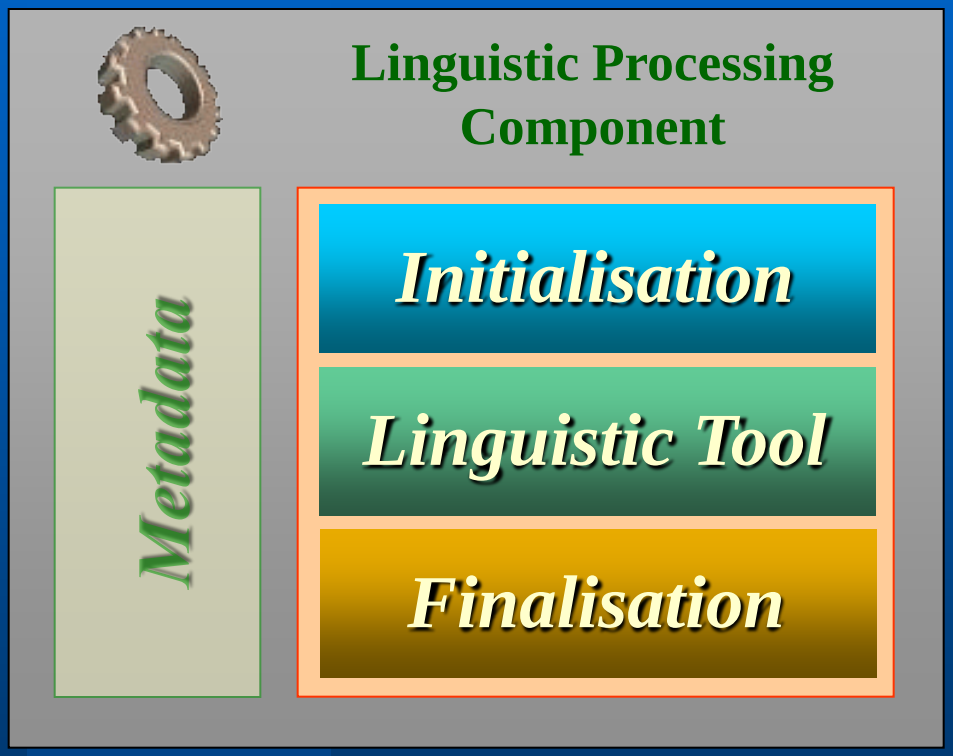
Anatomy of a Component



Anatomy of a Component


```
set creole_config(HBrill) {
  title {Hellenic POS Tagger}
  pre_conditions {
    collection_attributes      {}
    document_attributes        {language_hellenic}
    annotations                {token {sentence constituents}}
  }
  post_conditions {
    collection_attributes      {}
    document_attributes        {HBrill}
    annotations                {{token pos}}
  }
  viewers {
    {token pos}                          single_span {Brill POS Tags...}
    {token}                              raw         {Raw Token}
    $creole_HBrill_home/GreekTags.html text_file   {Greek POS Tag Definitions}
  }
  parameters {
    {Lexicon}          %FILE% "$creole_HBrill_home/greek/FinalLexicon"
    {Additional Wordlist}     {-w %FILE%}   {}
    {Process lines}           {-s %NUMBER%} {}
    {Start State Tagger only}  -S           {}
  }
  coupling dynamic
  description {This is the Greek Part-of-Speech Tagger}
  module_encoding iso8859-7
}
```

Anatomy of a Component



Anatomy of a Component


```
# Location of this module (SampleComponent)
global creole_SampleComponent_home

## creole_SampleComponent
proc creole_SampleComponent {doc args} {

  ## Put your code here...

  # record the fact that we ran and exit normally
  tip_PutAttribute $doc [tip_CreateAttribute SampleComponent \
                                [tip_CreateAttributeValue GDM_STRING {}]]
};# creole_SampleComponent

## Procedure: creole_SampleComponent_Initialize
proc creole_SampleComponent_Initialize {col doc args} {
  global creole_SampleComponent_home
  ## Do some initilisation here...
};# creole_SampleComponent_Initialize

## Procedure: creole_SampleComponent_Finish
proc creole_SampleComponent_Finish {col doc args} {
  global creole_SampleComponent_home
  ## Clean-Up...
};# creole_SampleComponent_Finish
```

Component Execution

Linguistic Processing Component

Initialisation

Linguistic Tool

Finalisation

Component Execution

Linguistic Processing Component

Initialisation

Linguistic Tool

Finalisation

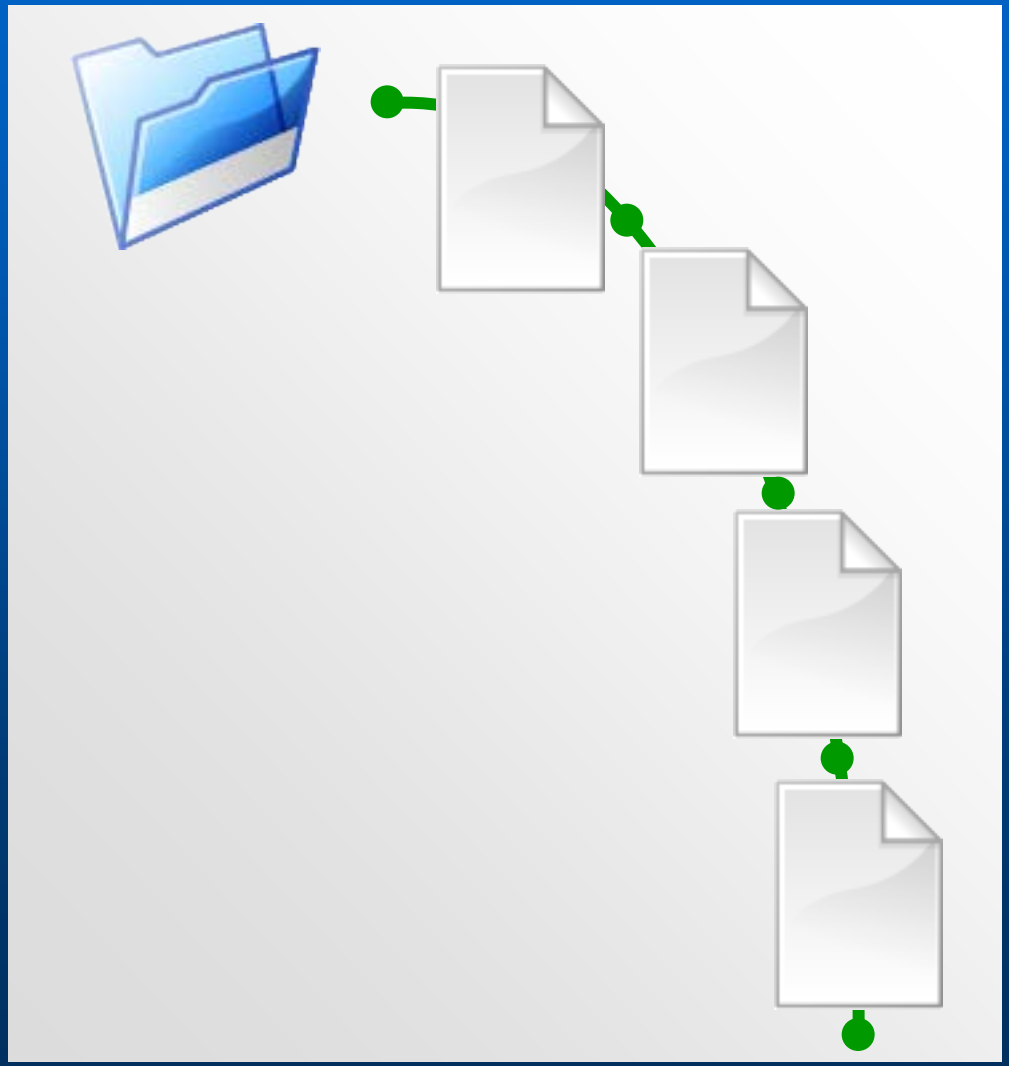
Component Execution

Linguistic Processing Component

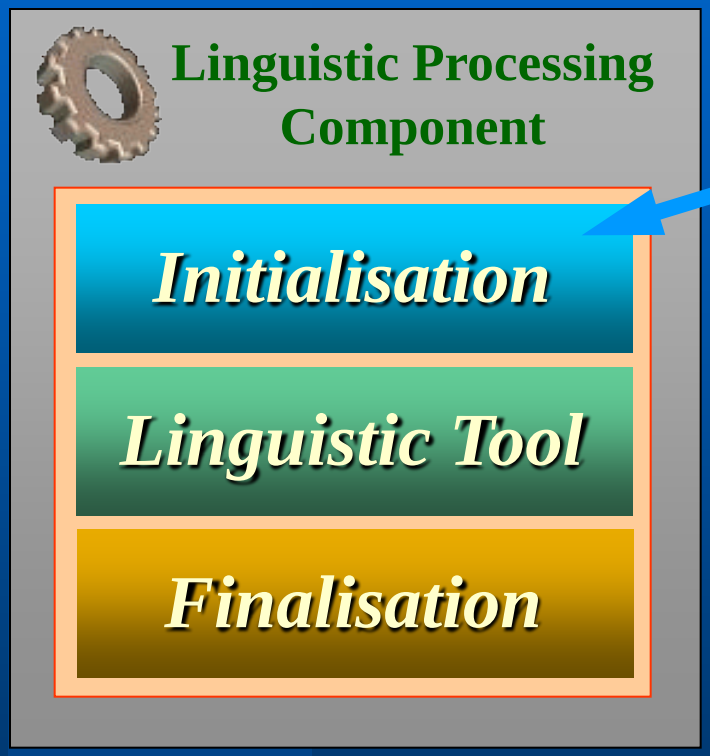
Initialisation

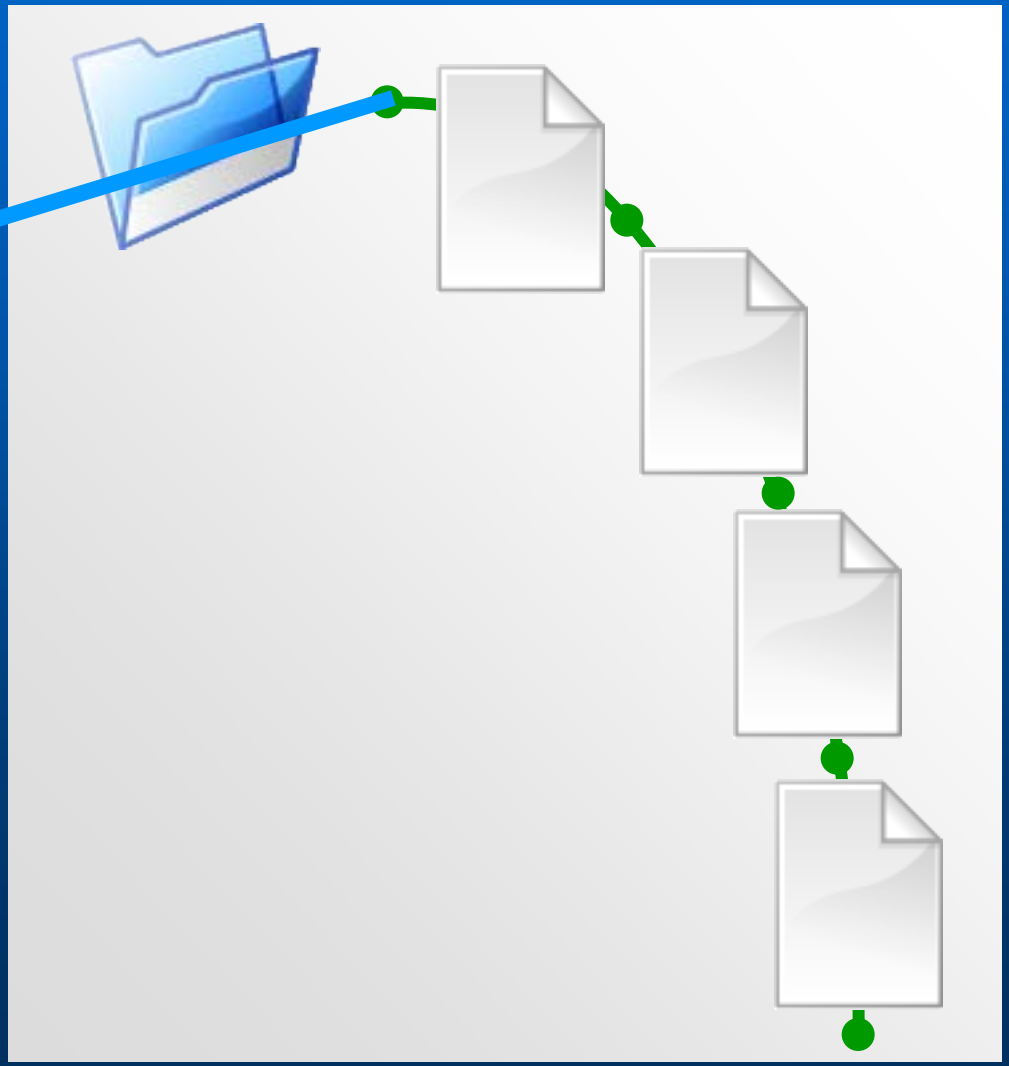
Linguistic Tool

Finalisation

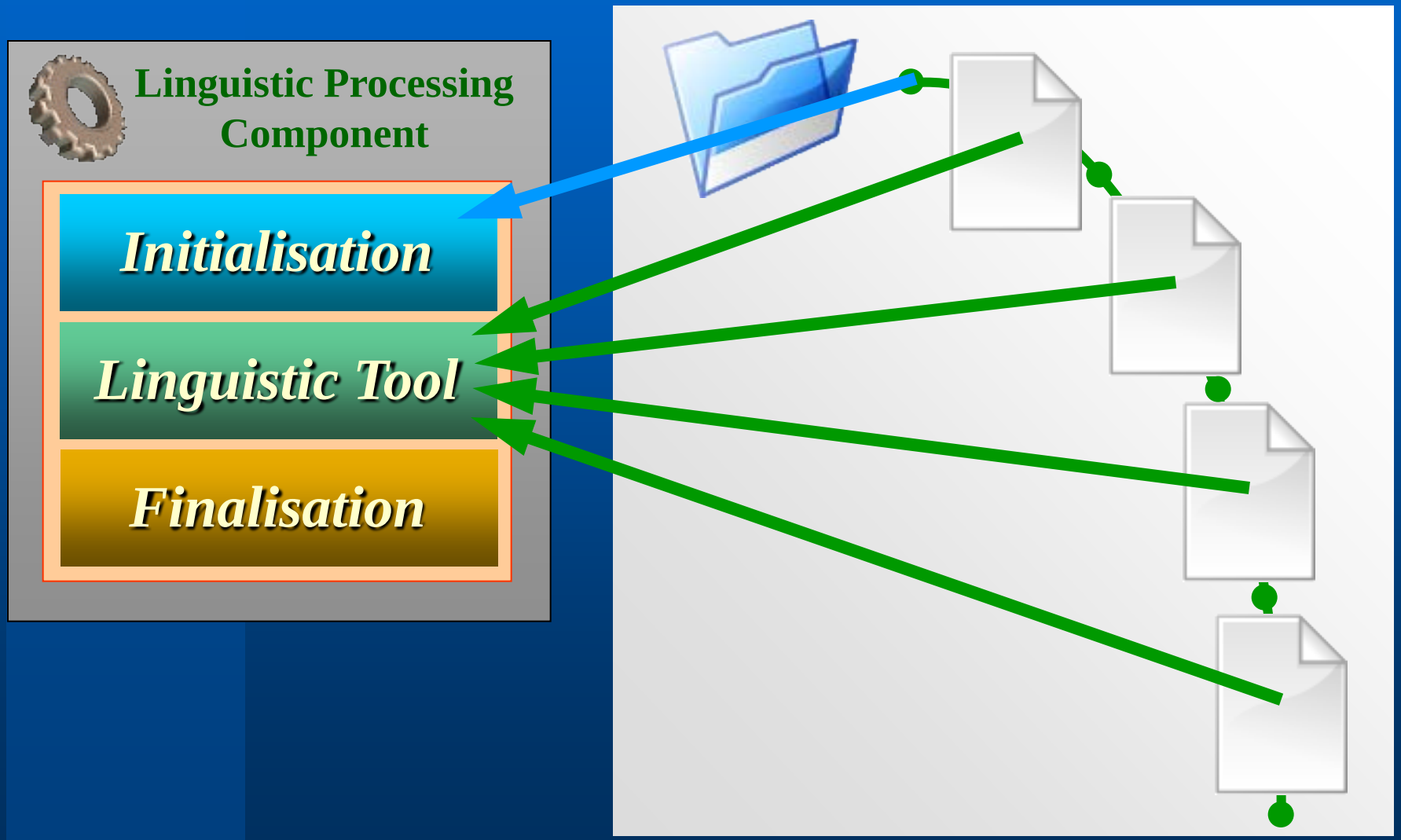


Component Execution

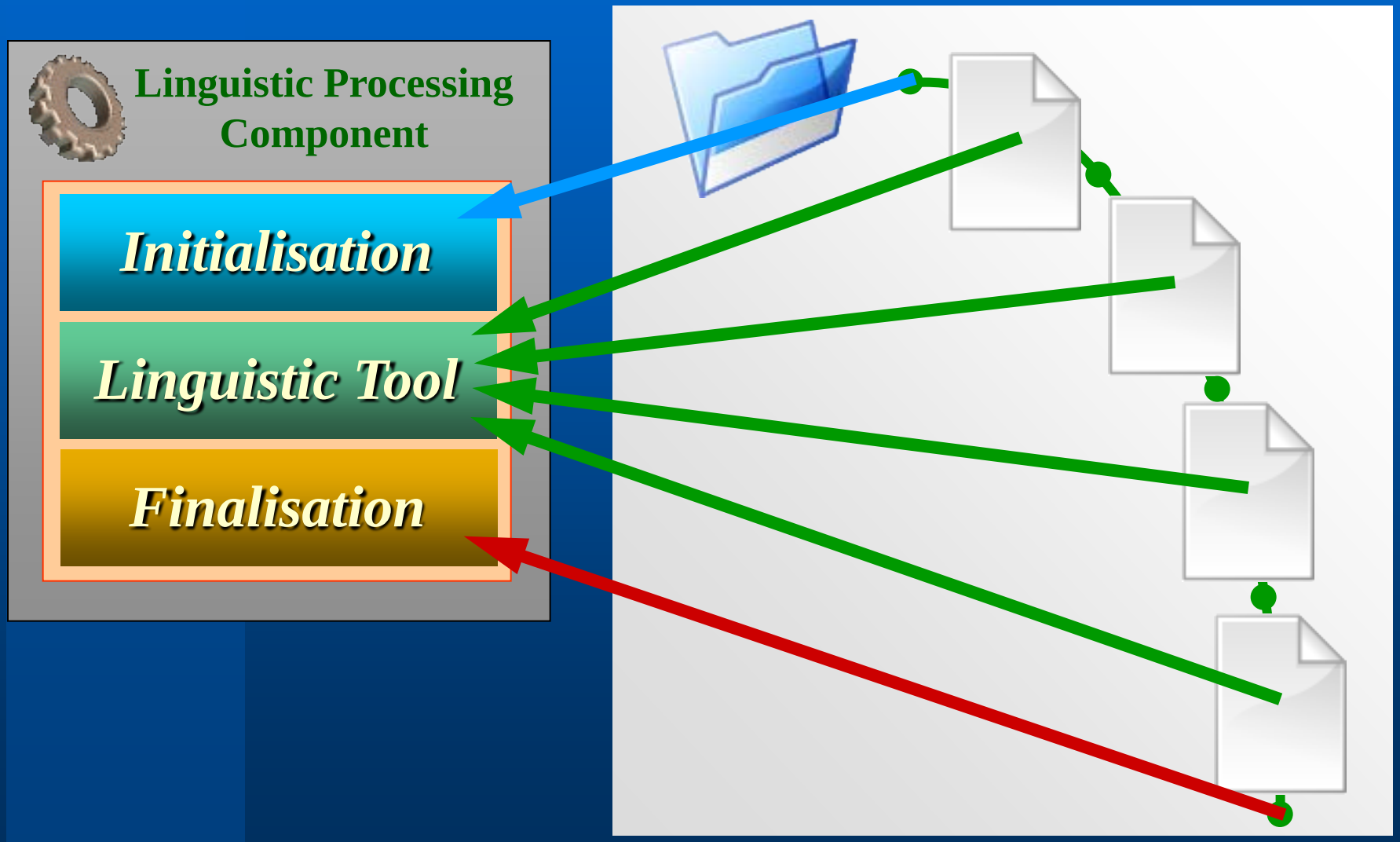




Component Execution



Component Execution



Overview

- Component Types
- How are Components Created?
- The Anatomy of a Component
- Writing a Wrapper Component
- Writing a Native Component

Wrapping a Tokeniser…

- **Tokeniser Input:**
  - **A file with the text to be processed.**

Input File Example

This a simple test. Wrapping an existing tokeniser as an Ellogon component.

Wrapping a Tokeniser…

- **Tokeniser Output:**
  - **A file with each token in a single line.**

Output File Example

This
is
a
simple
test
.
Wrapping
…

Preparing the Input File…

```
# Location of this module (Tokeniser)
global creole_Tokeniser_home

## creole_Tokeniser
proc creole_Tokeniser {doc args} {
  global CDM_TempDir

  # Open the input file:
  set input [open $CDM_TempDir/TokeniserInput[pid] w]
  # fconfigure $input –encoding utf-8
  puts $input [tip_GetRawData $doc]
  close $input

  # record the fact that we ran and exit normally
  tip_PutAttribute $doc [tip_CreateAttribute SampleComponent \
                              [tip_CreateAttributeValue GDM_STRING {}]]
};# creole_Tokeniser

## Procedure: creole_Tokeniser_Initialize
proc creole_Tokeniser_Initialize {col doc args} {
};# creole_Tokeniser_Initialize

## Procedure: creole_Tokeniser_Finish
proc creole_Tokeniser_Finish {col doc args} {
};# creole_Tokeniser_Finish
```

Executing the Tokeniser…

```
## creole_Tokeniser
proc creole_Tokeniser {doc args} {
  global CDM_TempDir creole_Tokeniser_home

  # Open the input file...
  set inputName  $CDM_TempDir/TokeniserInput[pid]
  set outputName $CDM_TempDir/TokeniserOutput[pid]
  set input [open $inputName w]
  puts $input [tip_GetRawData $doc]
  close $input

  # Executing tokeniser...
  exec $creole_Tokeniser_home/tokeniser –in $inputName –out $outputName

  # record the fact that we ran and exit normally
  tip_PutAttribute $doc [tip_CreateAttribute SampleComponent \
                              [tip_CreateAttributeValue GDM_STRING {}]]
};# creole_Tokeniser
```

Processing the Output File…

```
## creole_Tokeniser
proc creole_Tokeniser {doc args} {
  ...
  # Executing tokeniser...
  exec $creole_Tokeniser_home/tokeniser –in $inputName –out $outputName
  # Processing the output file...
  set output [open $outputName]
  set offset 0; set Text [tip_GetRawData $doc]
  while {[gets $output token] >= 0} {
    set start [string first $token $text $offset]
    set end   [expr {$start + [string length $token]}]
    set span  [tip_CreateSpan $start $end]
    set ann   [tip_CreateAnnotation token [list $span] {}]
    tip_AddAnnotation $doc $ann
    set offset $end
  }
  close $output
  file delete –force $inputName $outputName

  # record the fact that we ran and exit normally
  tip_PutAttribute $doc [tip_CreateAttribute SampleComponent \
                            [tip_CreateAttributeValue GDM_STRING {}]]
};# creole_Tokeniser
```

The Component Code…

```
proc creole_Tokeniser {doc args} {
  global CDM_TempDir creole_Tokeniser_home
  # Open the input file...
  set inputName  $CDM_TempDir/TokeniserInput[pid]
  set outputName $CDM_TempDir/TokeniserOutput[pid]
  set input [open $inputName w]
  puts $input [tip_GetRawData $doc]
  close $input
  # Executing tokeniser...
  exec $creole_Tokeniser_home/tokeniser –in $inputName –out $outputName
  # Processing the output file...
  set output [open $outputName]
  set offset 0; set Text [tip_GetRawData $doc]
  while {[gets $output token] >= 0} {
    set start [string first $token $text $offset]
    set end   [expr {$start + [string length $token]}]
    set span  [tip_CreateSpan $start $end]; set offset $end
    tip_AddAnnotation $doc [tip_CreateAnnotation token [list $span] {}]
  }
  close $output
  file delete –force $inputName $outputName
  # record the fact that we ran and exit normally
  tip_PutAttribute $doc [tip_CreateAttribute SampleComponent \
                            [tip_CreateAttributeValue GDM_STRING {}]]
};# creole_Tokeniser
```

Overview

- Component Types
- How are Components Created?
- The Anatomy of a Component
- Writing a Wrapper Component
- Writing a Native Component

A simple Gazetteer…

- **Objective:**
  - **Locate in text strings and classify them accordingly.**
- **Input:**
  - **A set of files, containing the strings/patterns to be located.**

A simple Gazetteer…

File: “Person.txt”

vangeli(s)?\s+karkaletsi(s)?
george\s+petasi(s)?

…

A simple Gazetteer…

File: “Person.txt”

vangeli(s)?\s+karkaletsi(s)?
george\s+petasi(s)?

…

George Petasis and Vangelis Karkaletsis are giving a lecture about Ellogon.

person

Initialising/Finalising Tokeniser…

```
## Procedure: creole_Gazetteer_Initialize
proc creole_Gazetteer_Initialize {col doc args} {
  global creole_Gazetteer_home creole_Gazetteer
  # Make sure no patterns already exist...
  unset -nocomplain creole_Gazetteer
  # Iterate over all files in the "Lists" directory...
  foreach filename [glob $creole_Gazetteer_home/Lists/*.txt] {
    set type [file root [file tail $filename]]
    set file [open $filename]
    # Add the patterns...
    while {[gets $file pattern] >= 0} {
      lappend creole_Gazetteer($pattern) $type
    }
    close $file
  }
};# creole_Gazetteer_Initialize

## Procedure: creole_Gazetteer_Finish
proc creole_Gazetteer_Finish {col doc args} {
 unset -nocomplain ::creole_Gazetteer
};# creole_Gazetteer_Finish
```

The Gazetteer Code…

```
## Procedure: creole_Gazetteer
proc creole_Gazetteer {doc args} {
  global creole_Gazetteer
  # Get a lower case version of the Document text...
  set Text [string tolower [tip_GetRawData $doc]]
  # Apply the patterns...
  foreach pattern [array names creole_Gazetteer] {
    foreach match [regexp –indices –all -- $pattern $Text] {
      foreach {start end} $match {break}
      set span [tip_CreateSpan $start $end]
      set attr [tip_CreateAttribute type \
        [tip_CreateAttributeValue GDM_STRING $creole_Gazetteer($pattern)]]
      # Create the Annotation...
      set ann [tip_CreateAnnotation lookup [list $span] [list $attr]]
      tip_AddAnnotation $doc $ann
    }
  }

  # record the fact that we ran and exit normally
  tip_PutAttribute $doc [tip_CreateAttribute Gazetteer \
                           [tip_CreateAttributeValue GDM_STRING {}]]
};# creole_Gazetteer
```