



LREC 2010 - Web Services and Processing Pipelines in HLT: Tool  
Evaluation, LR Production and Validation

## A Graphical Interface for Computing and Distributing NLP Flows

UAIC - FIL

**Ionuț Cristian Pistol, Andrei Arusoaie, Andrei Vasiliu, Adrian Iftene**  
Faculty of Computer Science, University “Al. I. Cuza” of Iași  
{ipistol, andrei.arusoaie, andrei.vasiliu, adiftene}@info.uaic.ro





# Outline

---

## I

- What is and what does a LP meta-system
- Automated Linguistic Processing Hierarchy

## II

- Distributed computing and NLP
- General Environment for Cluster Computing

## III

- Merging ALPE and GECC: benefits, overhead and future developments





# LP meta-systems

- Make use of existing modules in building LP chains
- Make use of existing linguistic resources and allow the user to add/build new ones
- Allow the user to compare and choose between modules available
- Allow the user to build and run processing flows involving multiple modules
- GATE (<http://gate.ac.uk/>), the first one (1998) CUE (<http://www.dcs.shef.ac.uk/~hamish/dalr/granada/mason.final.html>), ATLAS (<http://www.nist.gov/speech/atlas/>), UIMA (<http://www.research.ibm.com/UIMA/>), LiveTree (<http://www.fxpal.com>)





# Linguistic Processing Chains

- **Linguistic Processing (Chains) Flows** – used by UIMA and ALPE
- Def: a series of sequential and/or parallel applications of processing modules on an initial input that produces a output. During the execution of the processing flow additional resources might be accessed (language models, lexicons, etc.).
- A processing flow can be executed locally, on a single computer or on a distributed network (GRID) or can be run remotely as applications of webservice and scripts.





# GATE

- GATE = General Architecture for Text Engineering
- Initial version : 1995
- Developed (and under continuous development) at the Sheffield NLP Group, under the supervision of prof.Dr. Hamish Cunningham
- Theoretical base: [Cunningham, 2000] Cunningham H. – „Software Architecture for Language Engineering”. PhD Thesis, University of Sheffield, 2000
- Open source, Java based





# Apache's UIMA

- UIMA: Unstructured Information Management Architecture
- Developed initially at the IBM's Research Center
- Since 2006 developed by the Apache Software Foundation
- Initial open source distributed version: june 2007 (UIMA SDK)
- Open Source, Java based
- Distributed as a SDK libraries package and an Eclipse plugin.





# What do they offer?

---

- A SDK for ease of integration
- Access to a consistent list of LP modules
- Integrated modules can be transferred between GATE and UIMA
- The user is able to add new modules
- The user is able to configure processing flows
- The user is able to run processing flows on any number of documents





# Why ALPE?

- Automatically detect the format and language of a document in order to select the appropriate modules and resources
- Include new formats in a processing flow without wrappers
- Offer simple choices between available modules based on clear parameters (speed, cost, performance)
- Automatically build processing flows
- Improve access to linguistic resources and tools to Humanities and Social Sciences researchers and students, maybe even to the general public





# The ALPE hierarchy of annotation schemas

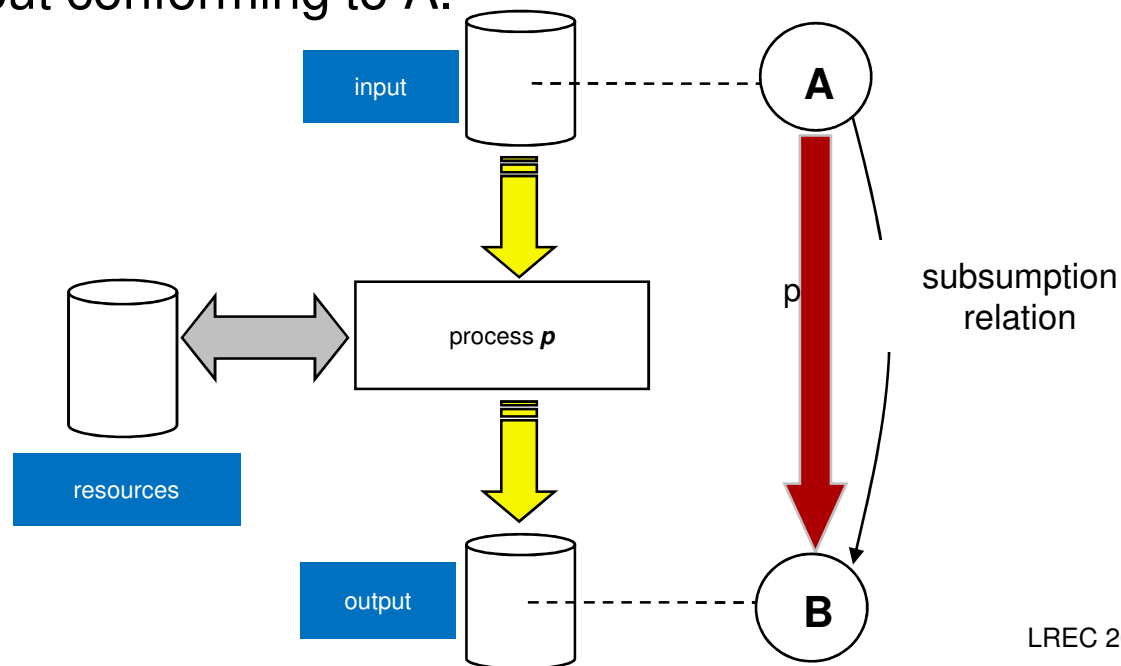
- The ALPE core hierarchy is a directed acyclic graph configuring the metadata of linguistic annotation in a hierarchy of XML schemas.
- Nodes in this graph are called **core nodes**. Each *core node* corresponds to a distinct XML annotation schema.
- Edges connecting core nodes are called **core edges**
- A *core edge* links a core node *A* with a core node *B* (say: *A* is **formally subsuming** *B*) iff:
  - any element (tag-name) of *A* is also in *B*;
  - any attribute in the list of attributes of a tag-name in *A* is also in the list of attributes of the same tag-name of *B*





# The augmented hierarchy (ALH)

- If node A subsumes node B we can attach a processing module to the edge between B and A. A module can be attached to the edge between B and A if running that module on a file conforming to the annotation schema corresponding to B obtains an output conforming to A.





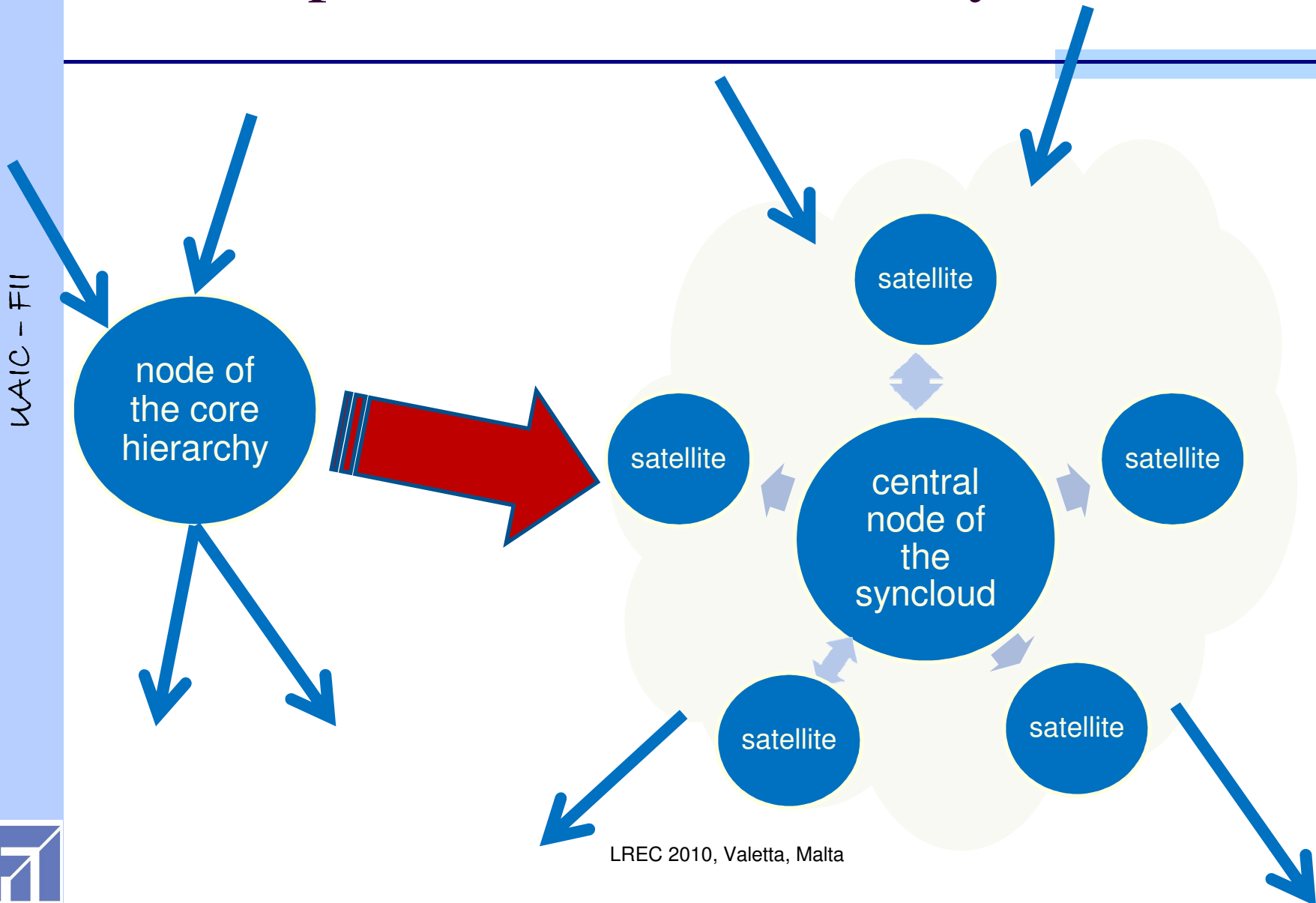
# Synclouds, central nodes, and conversion edges

- Each node of the core hierarchy has a clearly defined semantics
- Although not part of the core hierarchy, around each such node can co-exist also satellite nodes, carrying the same semantic information, but encoded differently: they form a **syncloud**
- The only node of the syncloud which is compulsory to belong to the core hierarchy is called the **central node**
- Central nodes are most likely to represent standards
- Satellite nodes of a syncloud can be connected to the central node with **conversion edges**: able to rewrite one format to another, without adding or removing information





# An expanded ALPE hierarchy node





# Building ALPE hierarchies

- The entire ALPE hierarchy building process involves interactions with one or more users who contribute :
  - annotated resources
  - annotation formats
  - processing modules
  - processing requests
- Details on the automated building procedure can be found in:

*Cristea, D., Forăscu, C., Pistol, I.C. (2006) "Requirements-Driven Automatic Configuration of Natural Language Applications". In Bernadette Sharp (Ed.): Proceedings of the 3rd International Workshop on Natural Language Understanding and Cognitive Science - NLUCS 2006, in conjunction with ICEIS 2006, Cyprus, Paphos. INSTICC Press, Portugal. ISBN: 972-8865-50-3.*

*Cristea, D., Pistol, I. (2008): "Managing Language Resources and Tools Using a Hierarchy of Annotation Schemas". Proceedings of the Workshop on Sustainability of Language Resources, LREC-2008, Marakesh.*





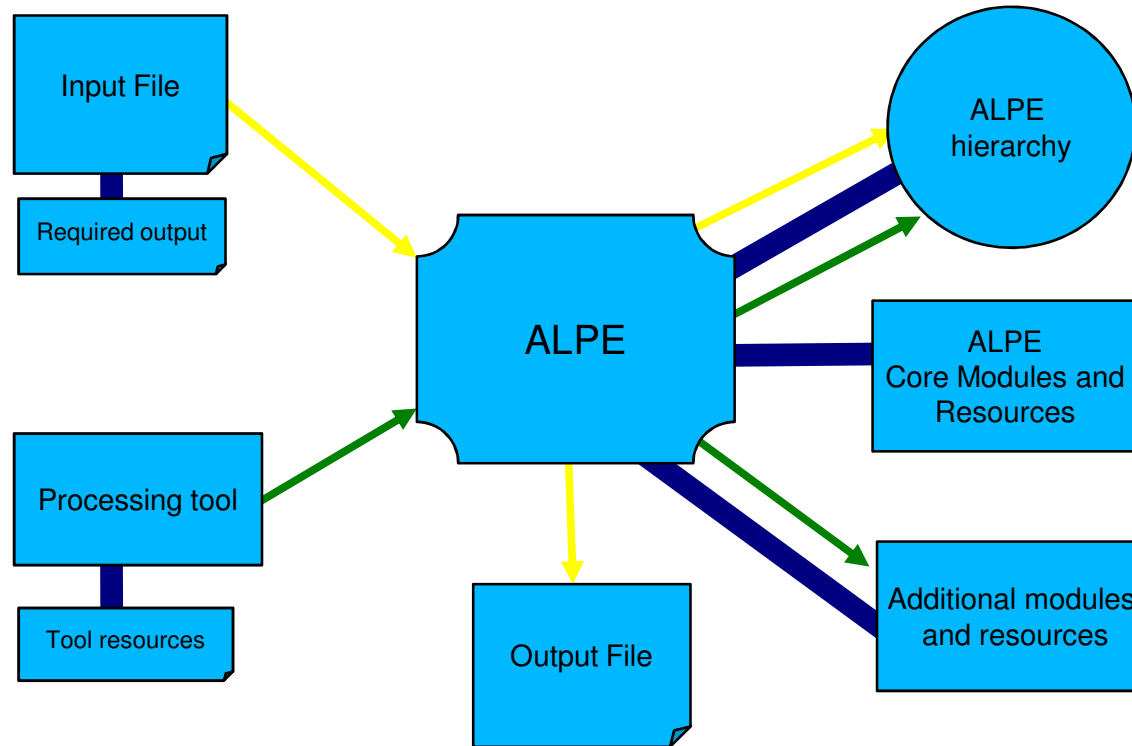
# ALPE processing flows

- ALPE flows are series of application of three operations on one or more processing module or annotated resource. The three operations are:
  - **Pipeline** ( $m_1, m_2, \dots, m_k$ ): adds the modules  $m_1, \dots, m_k$  to a processing pipeline
  - **Simplify** ( $sd_1, sd_2$ ): simplifies document from the standard  $sd_1$  to the subsumed standard  $sd_2$
  - **Merge** ( $flow_1, flow_2, \dots, flow_k$ ): produces a flow that merges the output of  $flow_1, flow_2, \dots, flow_k$
- The flow computation algorithm can produce more than one flow for a single query. The resulted flow have features such as cost, estimated speed, estimated quality of results, number of intermediate annotations, number of required modules, available languages.





# Using ALPE





# Distributed Computing and NLP

- GATE Cloud--a parallel distributed processing engine that combines GATE embedded with a heavily optimised service infrastructure running on supercomputer hardware (<http://gatecloud.net/>) - prototype
- UIMA Grid: Distributed Large-scale Text Analysis, UIMA flows as REST services
- Several efforts to develop NL tools as GRID services: Illsley et. Al., Hughes et. Al., Iftene et. Al.





# GECC

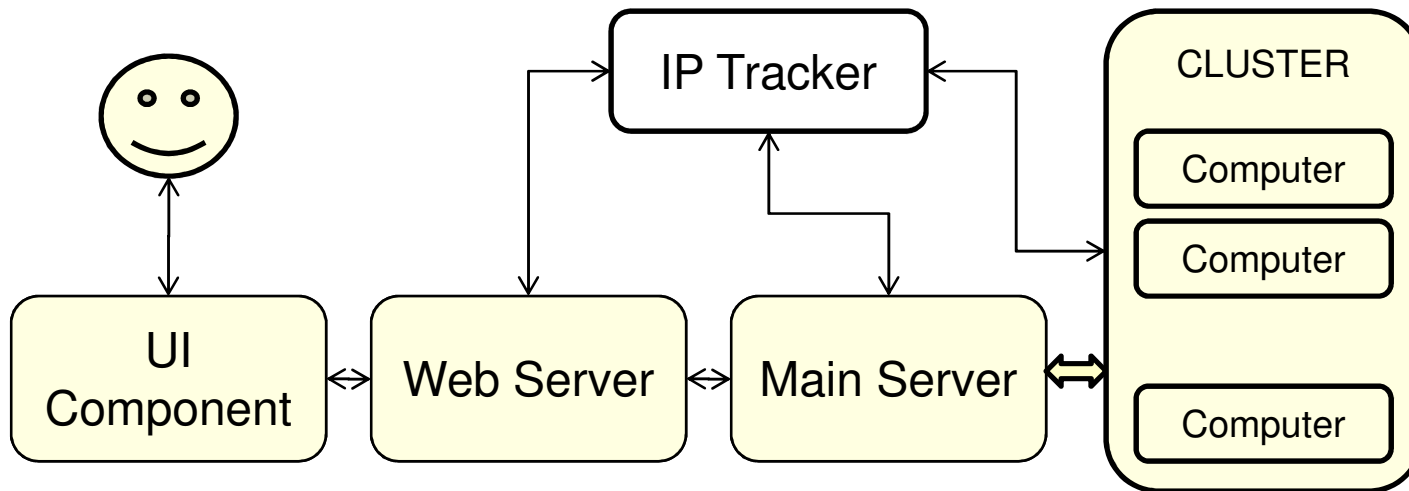
- What is GECC ?
- General Environment for Cluster Computing
- Offers a general way to run applications on a Cluster
- Provides a graphical interface to design workflows





# GECC

## ■ Architecture

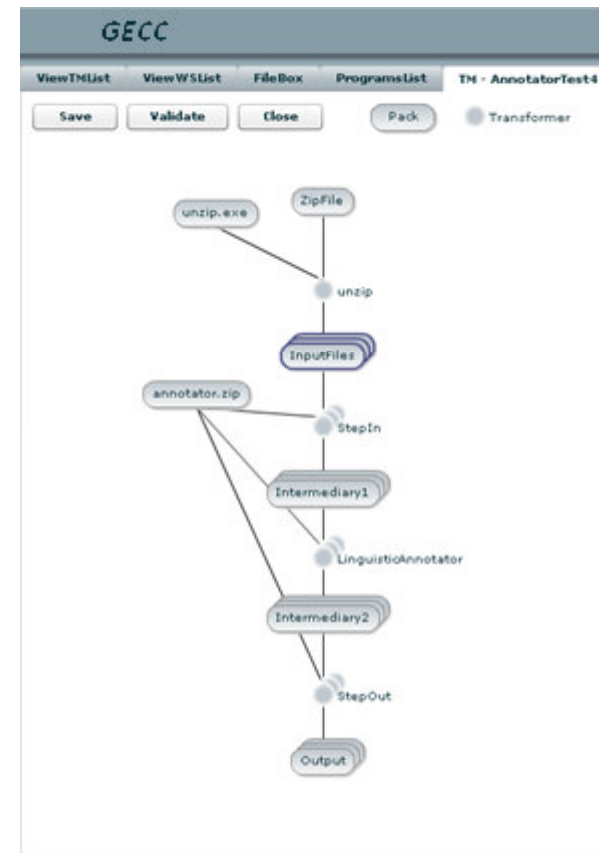




# GECC

## ■ User interface

- edit, save, validate
- *transformation model*
- *pack nodes*
- *work sessions*





# GECC

- WebServer
  - allows HTTP requests
  - start *worksessions* on Cluster
  - retrieves Cluster execution status
- MainServer
  - validates *transformation models*
  - cluster manager:
    - split transformation models into tasks
    - retrieve results from Cluster Computers





# GECC

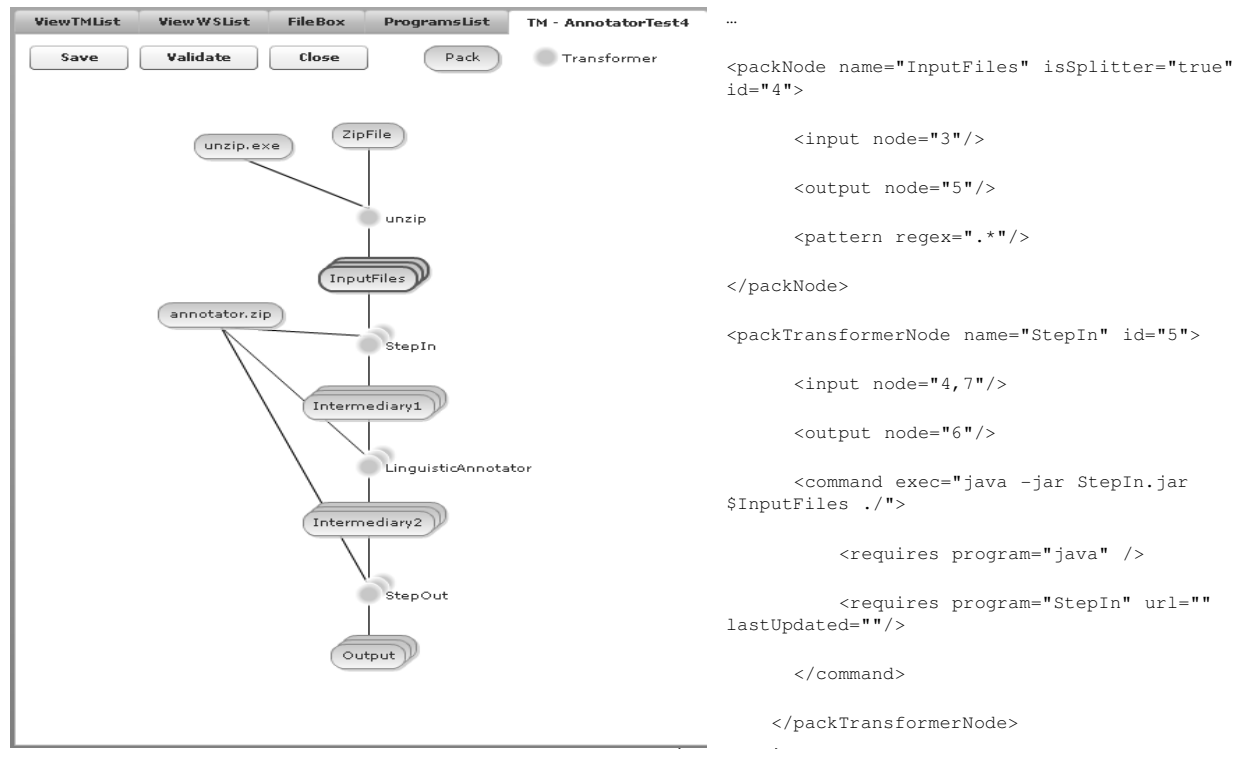
- IPTracker
  - permits zero configuration for all components
- Cluster
  - collection of Cluster Computers (CC)
  - executes complex tasks received from MainServer





# ALPE to GECC

- ALPE computed flow is saved as a GECC compatible XML configuration file





# Merging ALPE and GECC: benefits

- ALPE offers an user-friendly way to access NL resources and tools, which can be combined and run using GECC.
- The end-user has access to remote resources and tools, independent of software and hardware requirements.
- The processing flow, once loaded in GECC, can be edited via a graphical interface.
- Significant improvement in processing speed, especially for large corpora and/or complex flows. Experiments show an improvement of up to 2000%.
- GECC offers a flexible system of networked computers and requires only one stable central server.





# Merging ALPE and GECC: overhead

- Slight reduction in speed for smaller corpora and/or simple flows, due to file pack/transfer/unpack operations.
- Slight instability to module errors. ALPE handles individual module errors, but flows run via GECC are virtually blind at the moment.
- Web Services are used only as a single thread process at the moment.





# ALPE and GECC: future developments

- Compatibility between GATE and UIMA flows and ALPE flows, which will make GATE and UIMA flows executable using GECC.
- ALPE will be released as a Web Service allowing users to build and use ALPE hierarchies and tools available both on ALPE servers and locally.
- GECC will be released as an open-source platform.



# ALPE and GECC – developed at UAIC

---

UAIC – FII

# Thank you!

