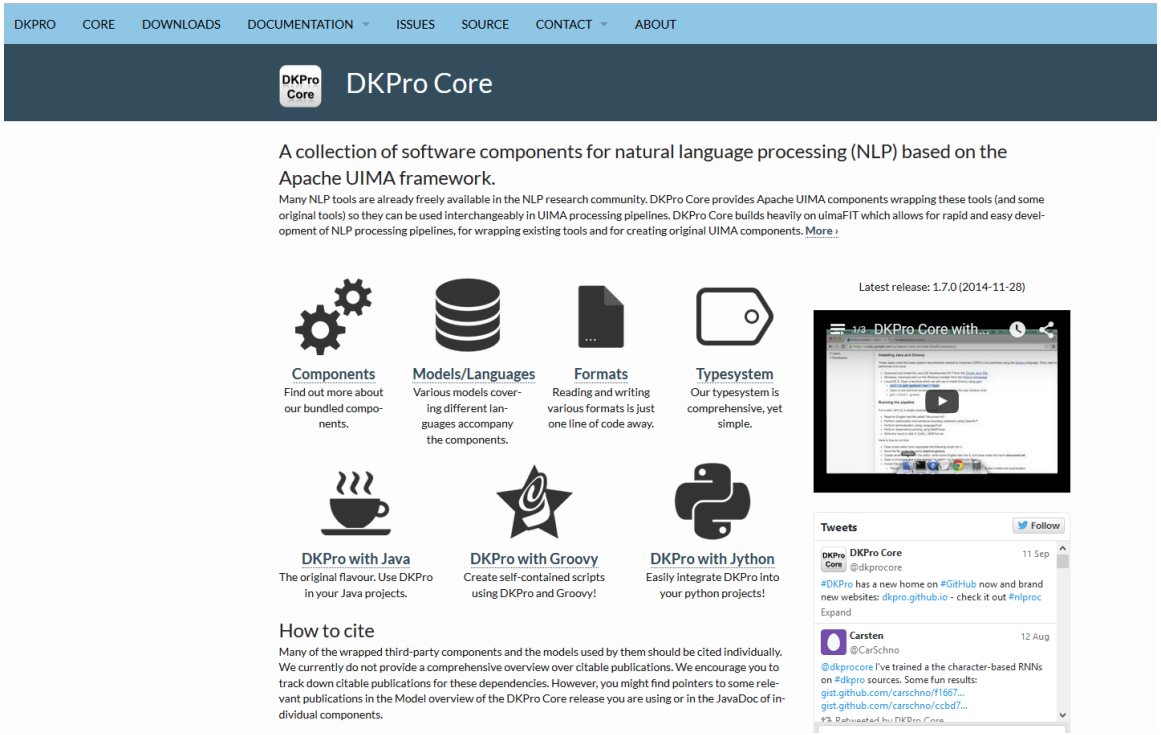
DKPro Core
Software components for NLP

M.Sc. Pedro Santos – Ubiquitous Knowledge Processing Lab



Agenda

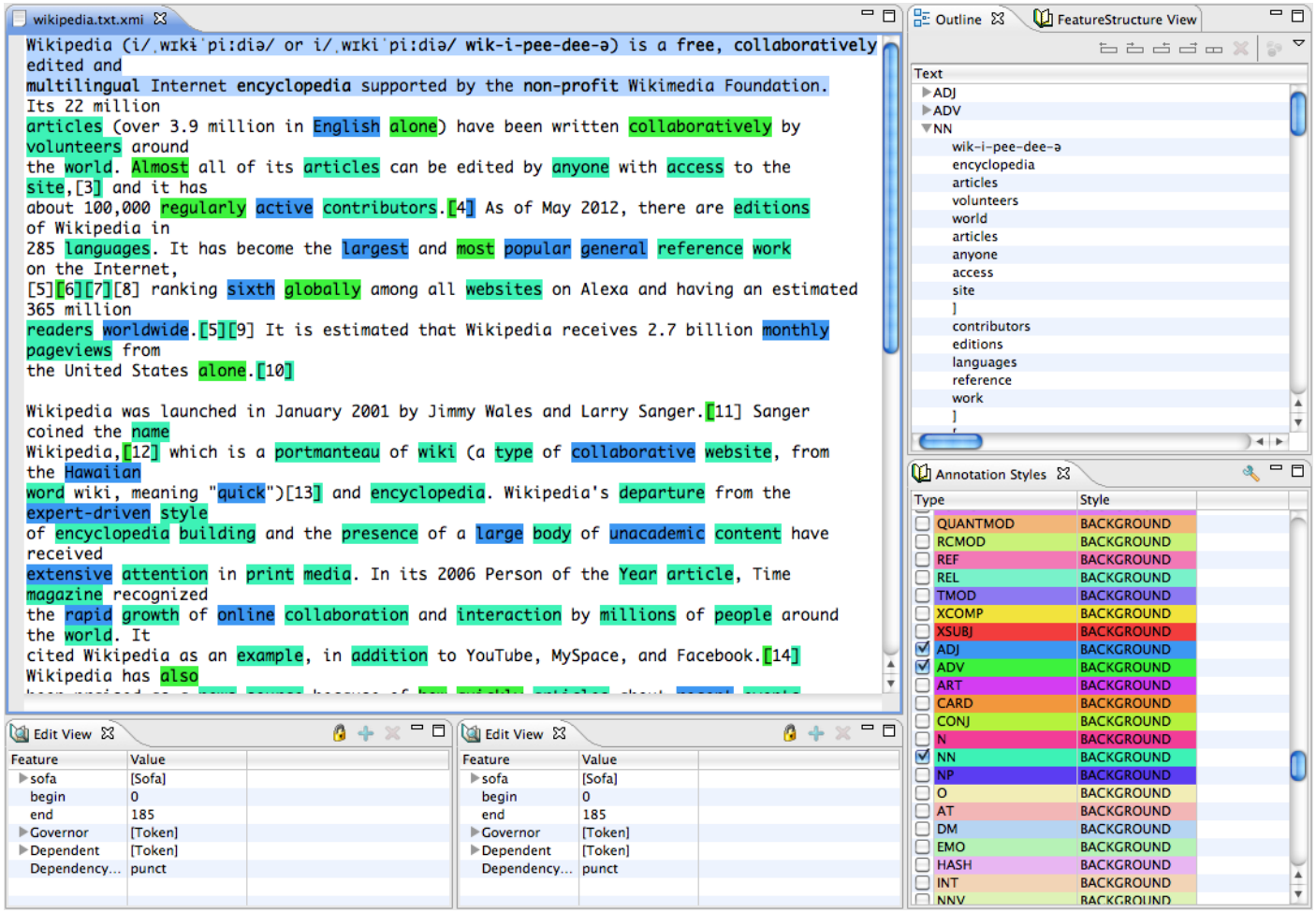
- What is a pipeline?
- Working with annotations
  - What is a type system?
  - What is the Common Analysis Structure (CAS)?
- Working with components
  - What is a reader?
  - What is an analysis engine?
  - What is a writer? (aka consumer)
- DKPro Core component collection

What is UIMA?

- Component-based architecture
  - Analysis of **unstructured data**
  - Structure from the unstructured data

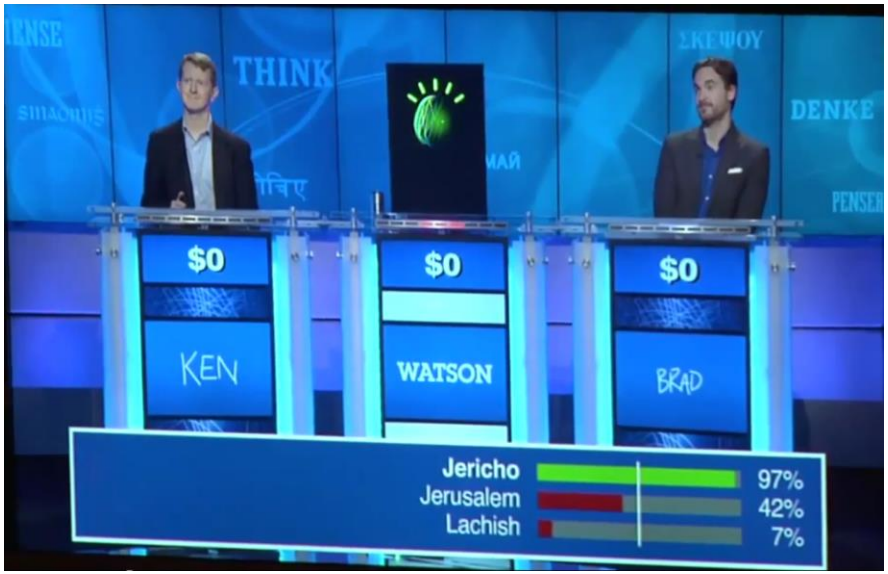
- How?
  - Like an assembly line...
  - Raw material
  - Refinement, step by step
  - Nice car in the end

Output Example (UIMA Annotation Editor)



Apache UIMA™ – Some history

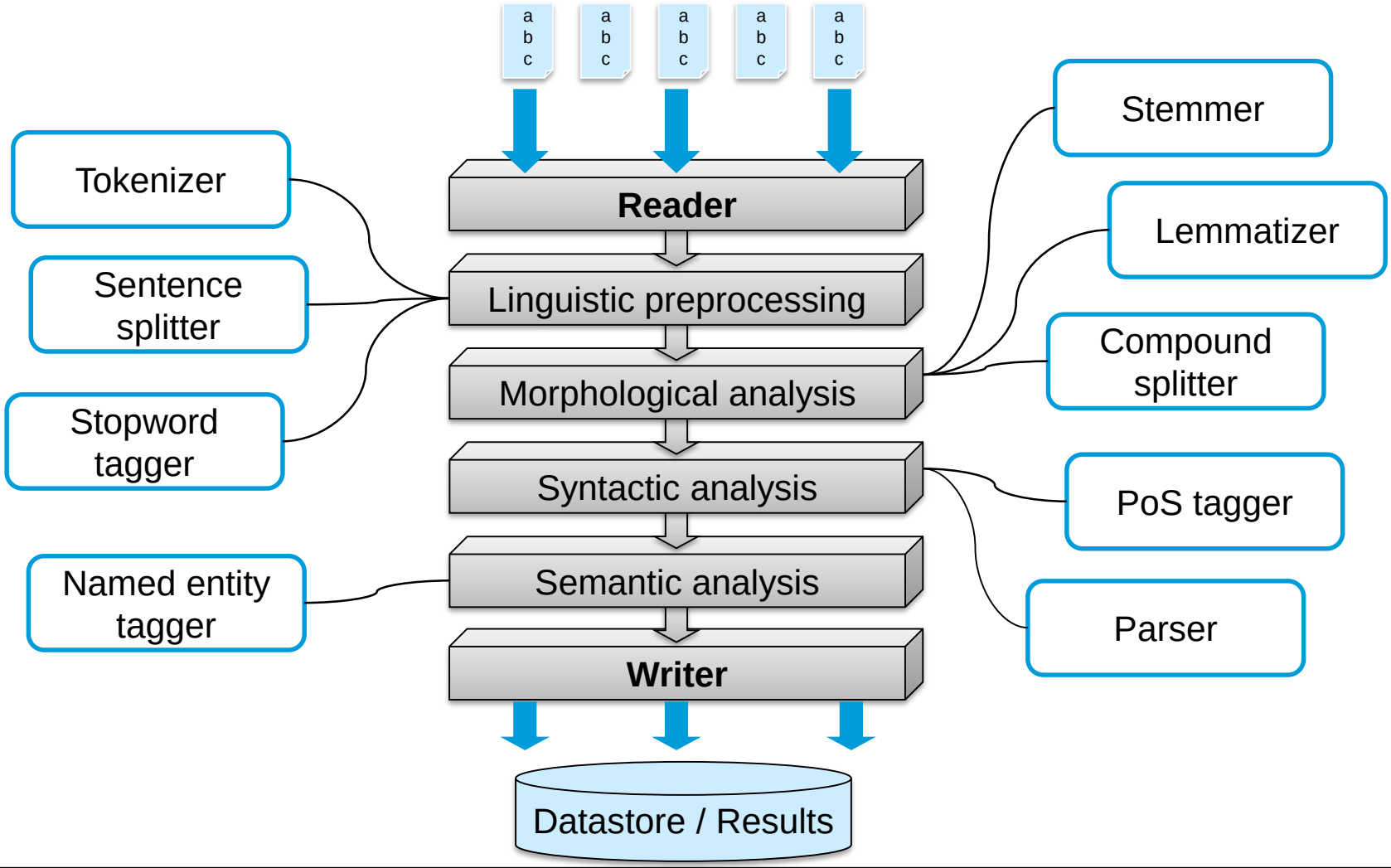
- 2003 – David Ferrucci and Adam Lally paper
  - *Accelerating corporate research in the development, application and deployment of human language technologies*
- 2004 – IBM alphaWorks project
  - IBM LanguageWare
- 2006 – Apache Incubator project
- 2009 – OASIS Standard
- 2010 – Full Apache project
- 2010 –IBM's *Watson* Jeopardy Challenge



http://uima.apache.org

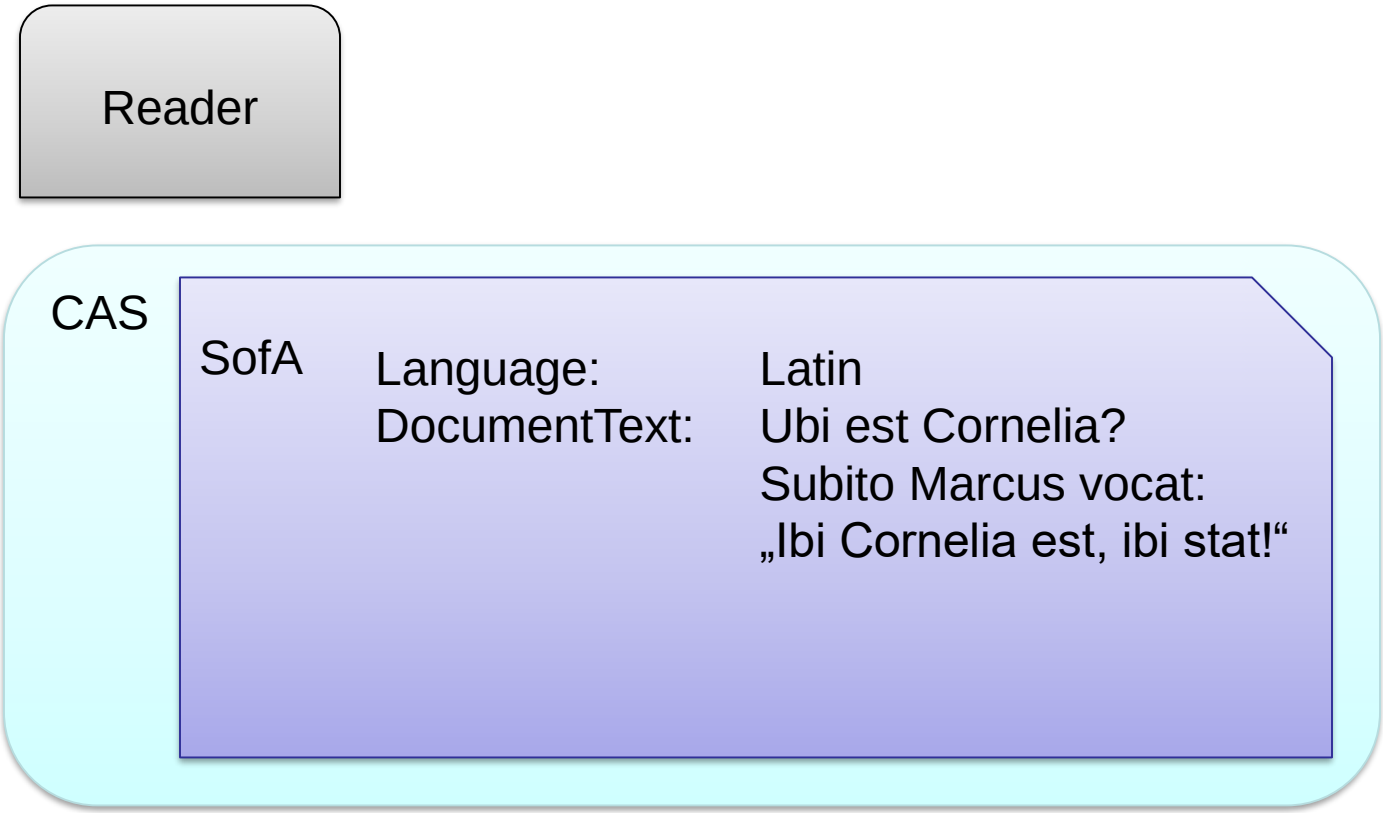
Pipelines

Pipeline Architecture



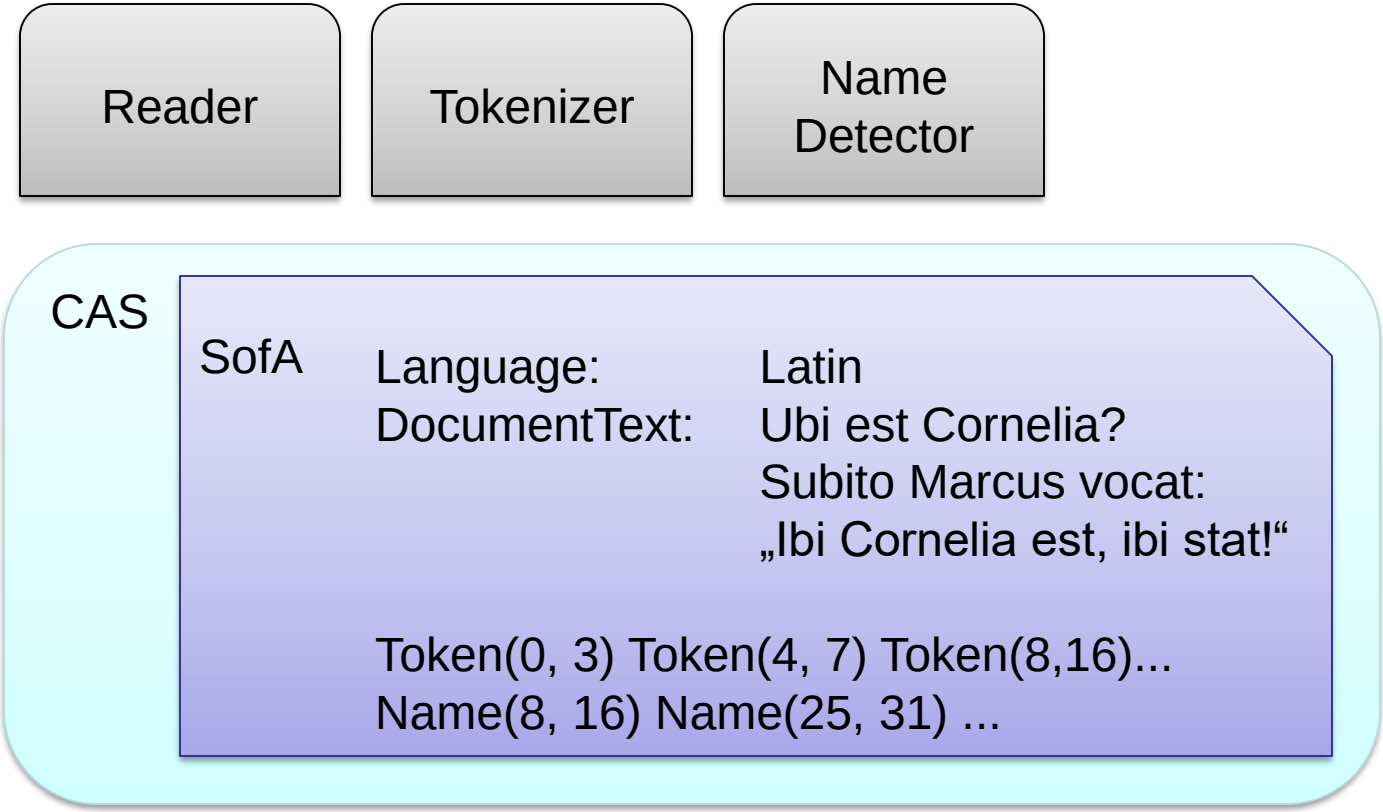
Component – Collection Reader

- Empty data structure (CAS) → Reader
- Reader → Text (SofA) and Meta-Data (e.g. language)



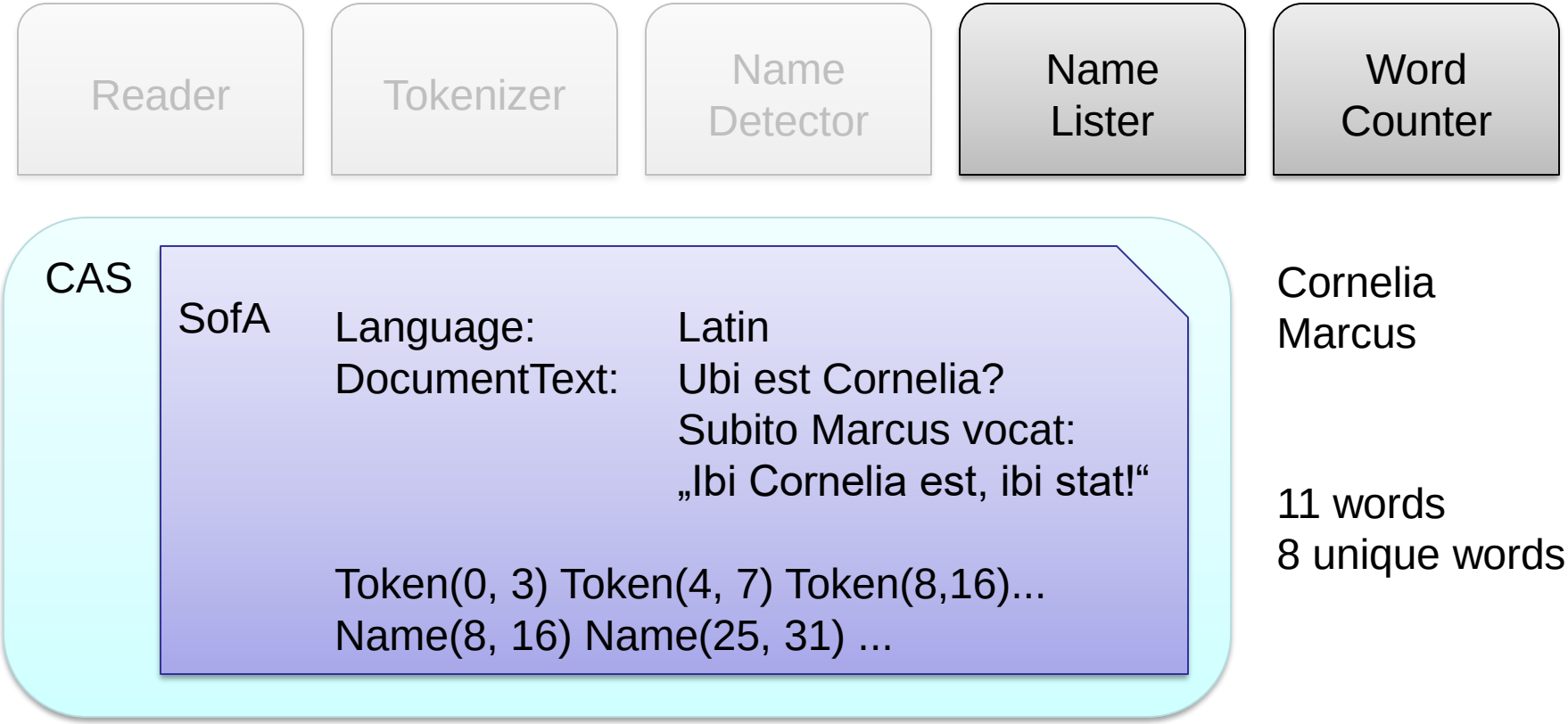
Component – Analysis Engine

- Structure → Analysis Engine (AE)
- Analysis Engine → Annotation



CAS Consumer

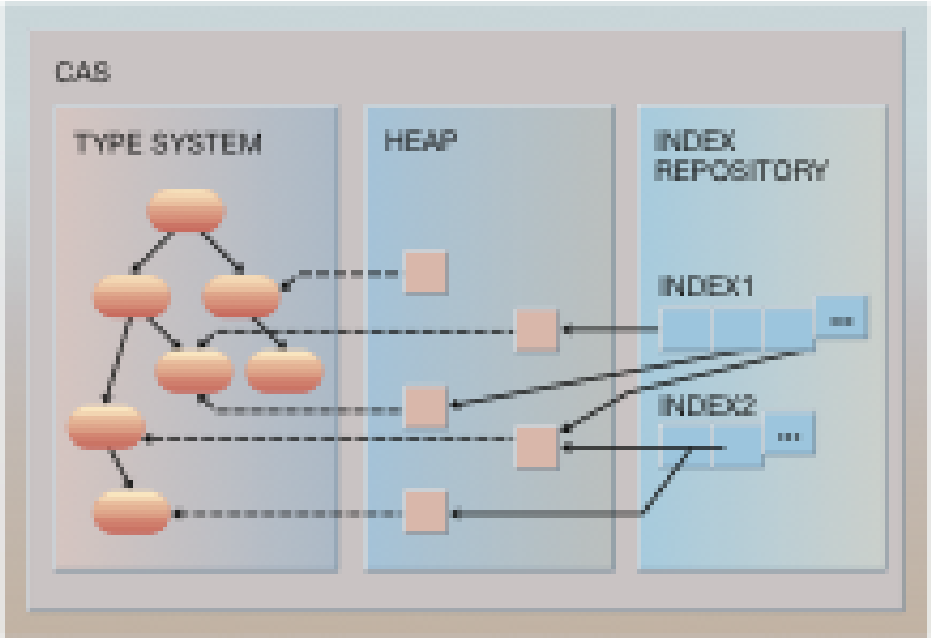
- Annotation -> CAS Consumer



UIMA Data Structures

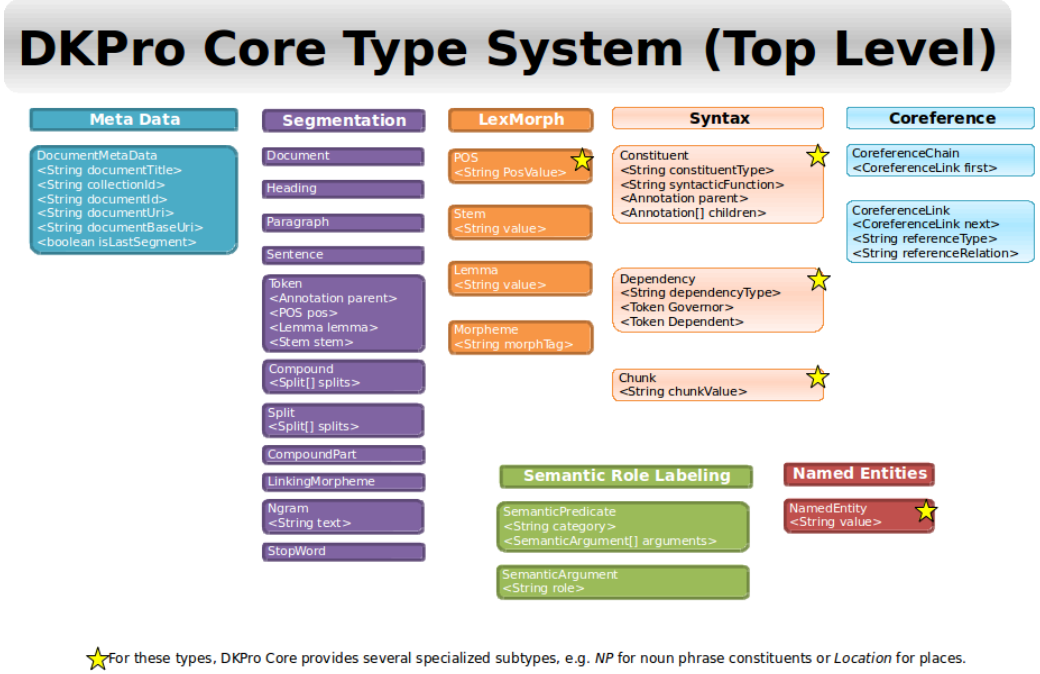
Common Analysis System (CAS)

- Access to primary data
- Secondary data storage
  - a.k.a. Annotations
- In-memory database
  - Annotation types = tables
  - Indexes



Type System

- Platform-independent specification
- Object-oriented type system:
  - Type → class
  - Feature → class member
  - Feature Structure -> instance
  - Single inheritance
  - Sub-type polymorphism
  - No methods or encapsulation
- Primitive types: integer, float, boolean, string
- Built-in complex types: arrays, lists, Annotation
- Communication contract

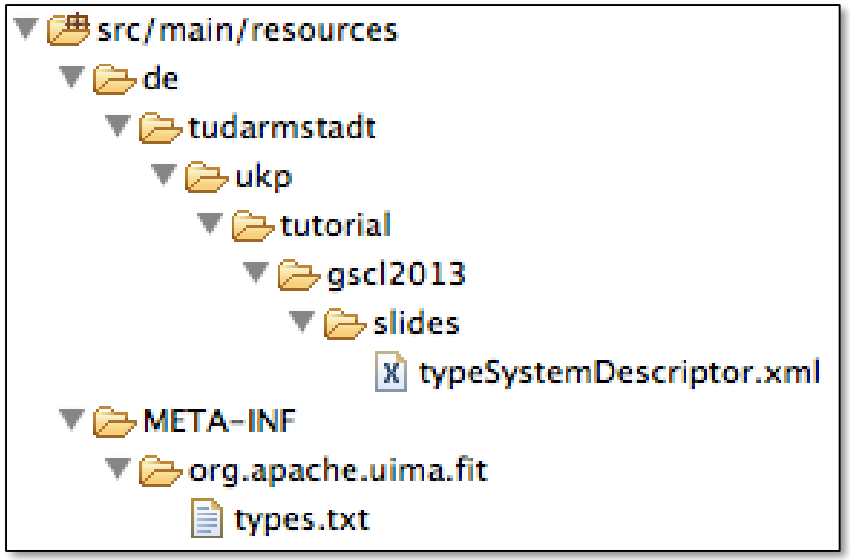


Java + CAS = JCas

- JCas = CAS types into the Java type system
- JCasGen
  - Java classes from XML type system descriptor
  - Token.java – feature structure wrapper with getters and setters
  - Token_type.java – type wrapper (cf. Java ‘Class’ class)
- JCas wrappers cannot be used stand-alone

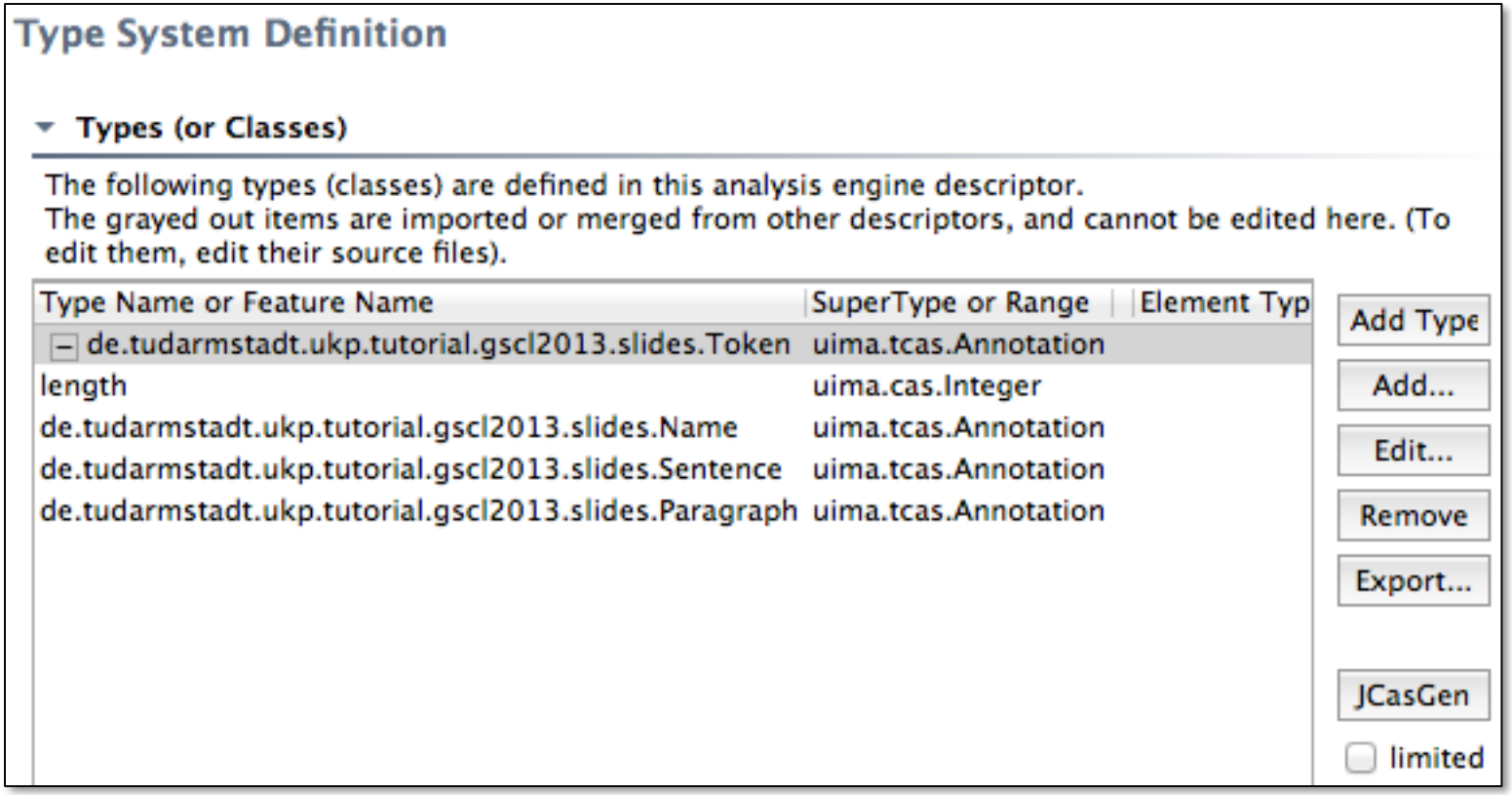
uimaFIT type system detection

- No explicit loading/creation of type system
- Type system detection mechanism
- Types defined in XML descriptor files
- Scaning of classpath for type system descriptor files



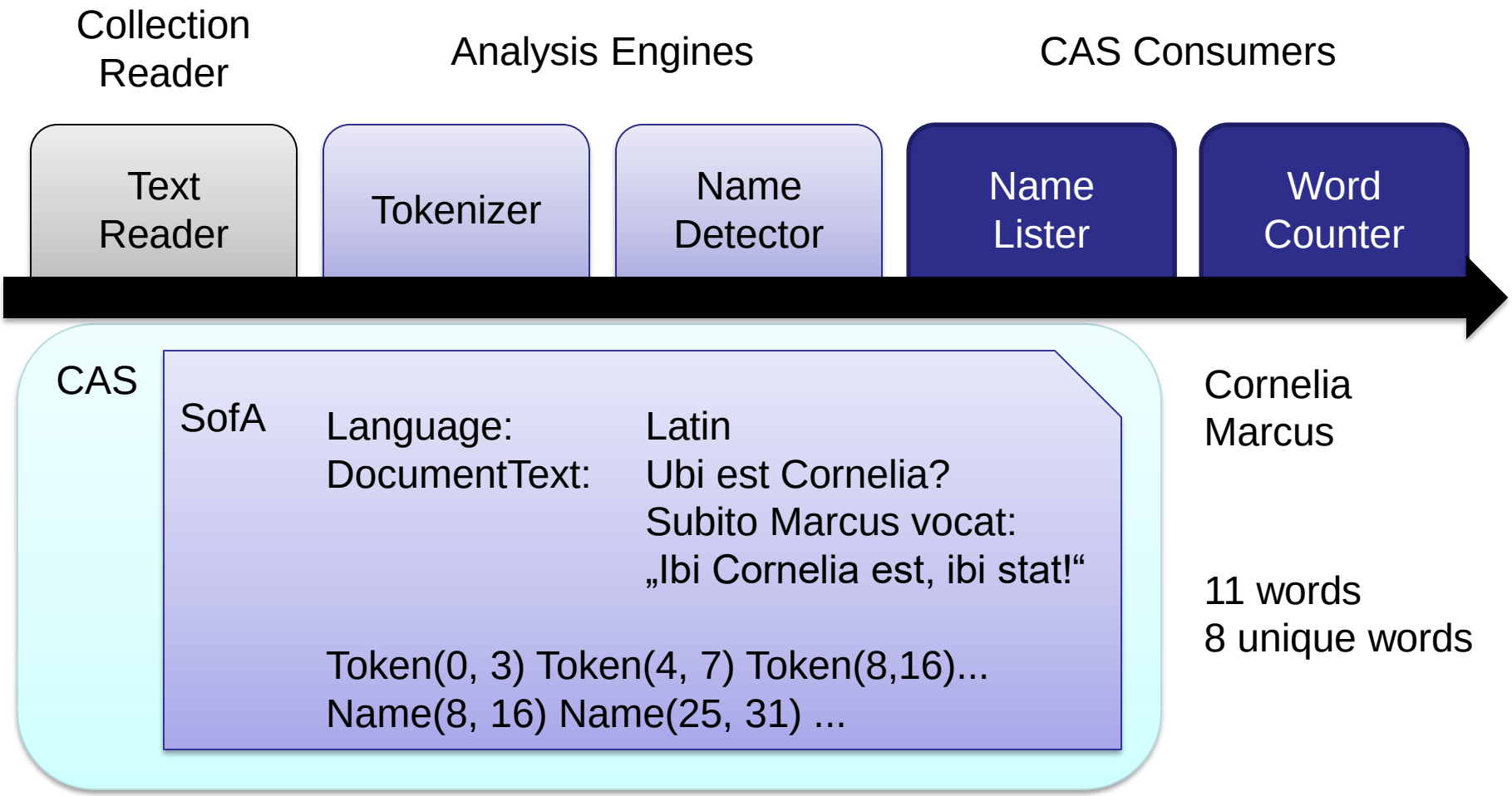
Type System Editor (Eclipse)

- JCasGen → UIMA types available as Java Classes



Components

Components

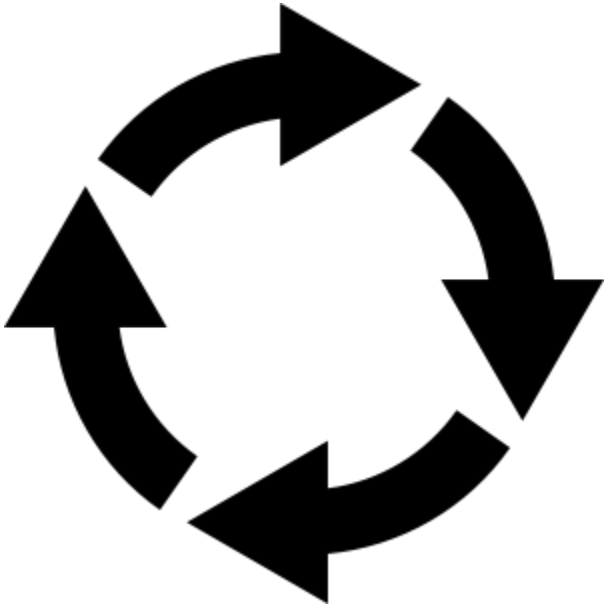

Collection Reader
Analysis Engines
CAS Consumers
Text Reader
Tokenizer
Name Detector
Name Lister
Word Counter
CAS
SofA
Language: Latin
DocumentText: Ubi est Cornelia?
Subito Marcus vocat:
„Ibi Cornelia est, ibi stat!“
Token(0, 3) Token(4, 7) Token(8,16)...
Name(8, 16) Name(25, 31) ...
Cornelia
Marcus
11 words
8 unique words

API – Life-Cycle Events

- Component life-cycle events
  - initialize()
  - reconfigure()
  - destroy()

- Processing life-cycle events
  - collectionProcessComplete()
  - batchProcessComplete()

- Other
  - typeSystemInit()



API – Processing Methods

- CollectionReader
  - hasNext()
  - getNext()
  - getProgress()
- AnalysisEngine
  - process()
- CasConsumer
  - process()

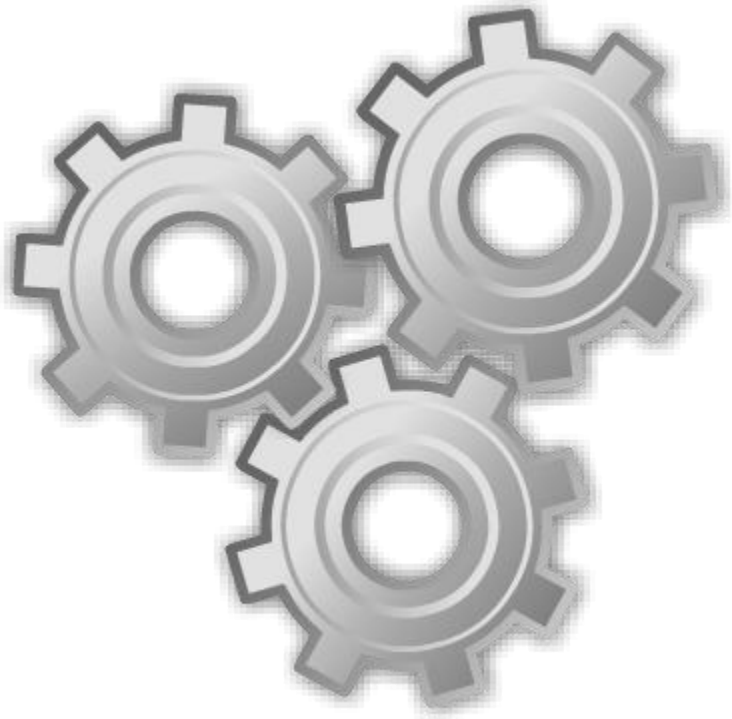
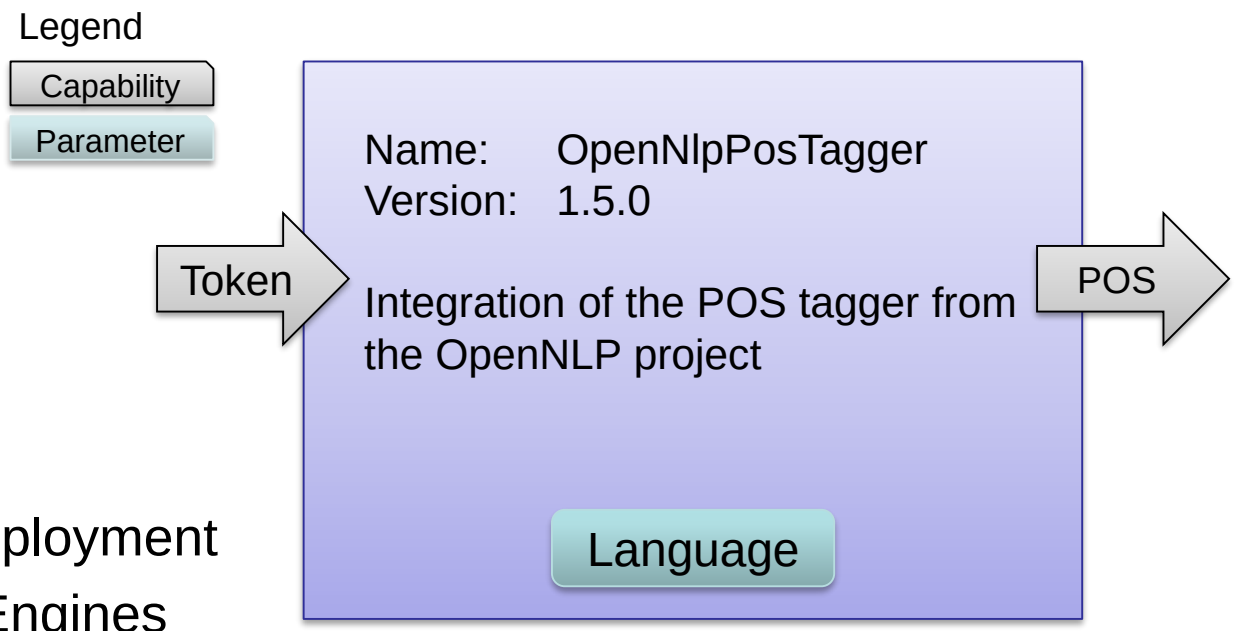


Figure: Analysis Engine Descriptor

- Name
- Version
- Vendor
- Type system
- Parameters
- Capabilities
- Indexes
- Resources
- Single- / multiple deployment
- Delegate Analysis Engines
- Flow control
- … a few more



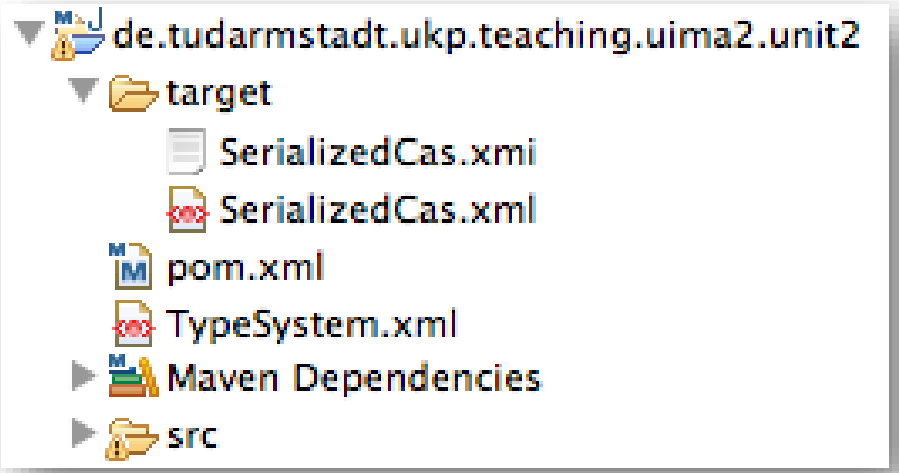
XML Descriptors – Pro & Contra

- Pro
  - “Officially preferred” form of configuration from UIMA components/resources
  - Widely supported by UIMA tooling
  - XML elements ↔ Java classes

- Contra
  - Mix declaration/documentation and configuration
  - Not included when refactoring code
  - No convenient API (remedy: uimaFIT factories)

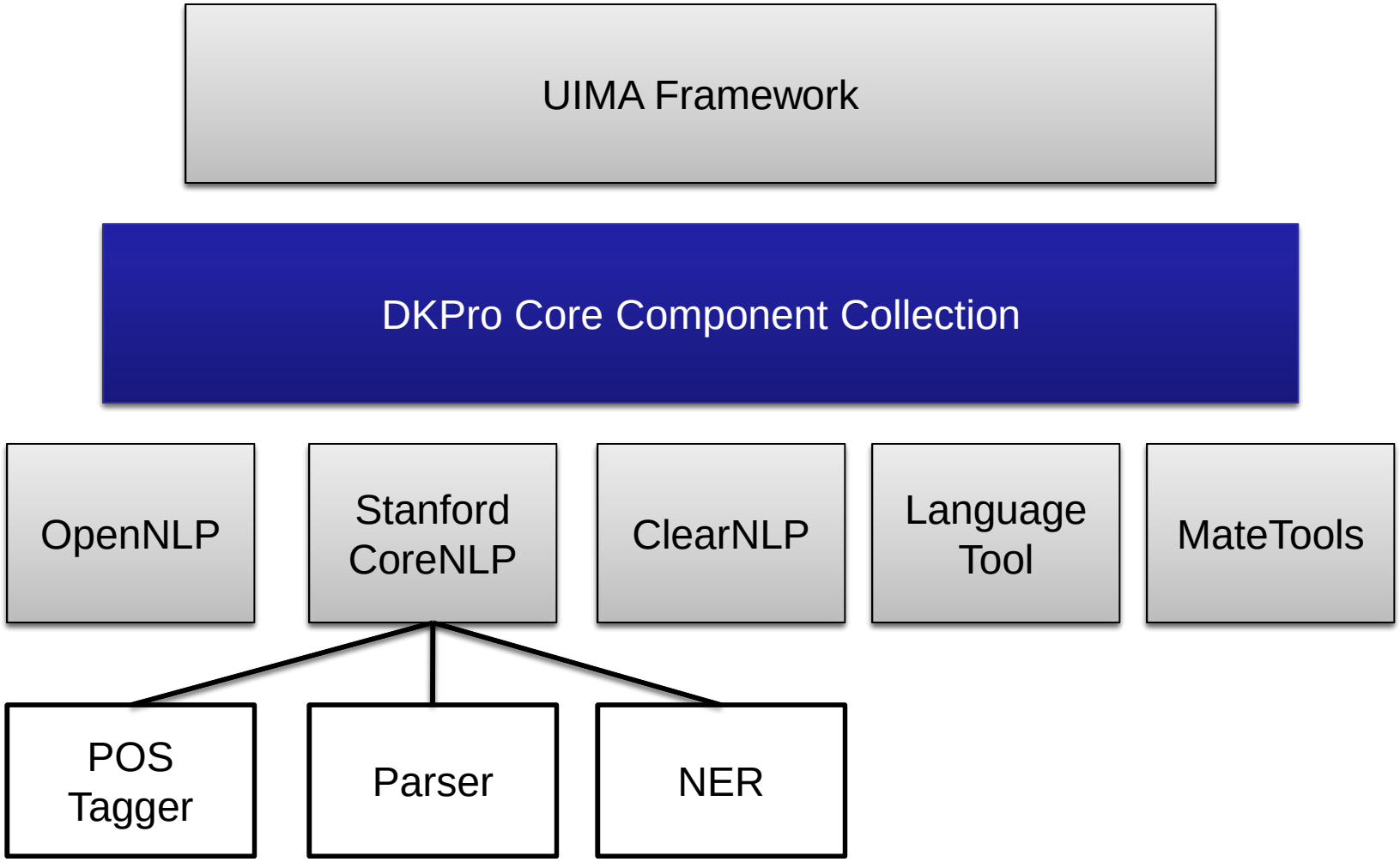
Persisting and loading a CAS

- Available serialization formats
  - XCAS
  - XML format
  - XMI
- Type-system definition not included!
- Tip
  - Persist type system as “TypeSystem.xml” at project root
  - Open and XMI file in that project with the CAS Editor



DKPro Core Component Collection

What‘s a component collection

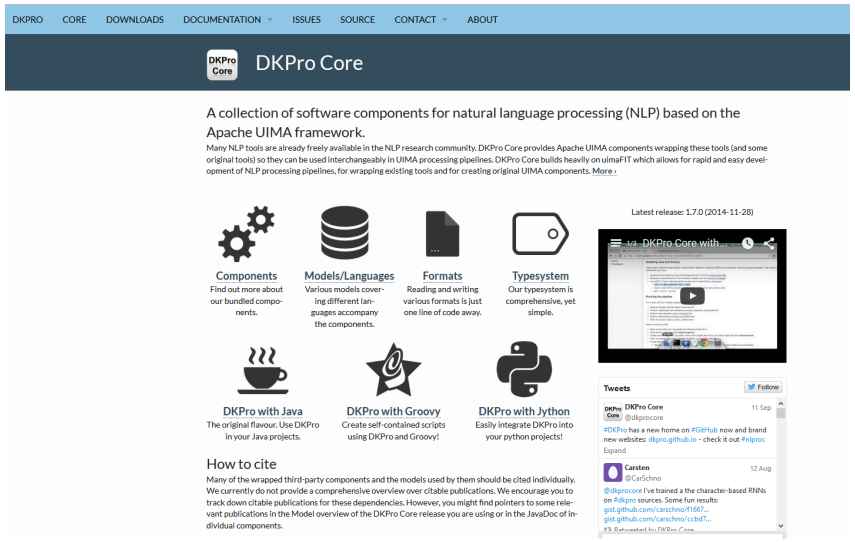


DKPro Core

- Integration framework
  - Processing: tools and models
  - Primary data: corpora
  - Auxiliary data: other language resources (e.g. lexical resources)

- Primarily integration of existing work, not original work
- Contribution: integration itself

- Open Source under Apache Software License & GNU Public License



https://dkpro.github.io/dkpro-core/

DKPro Core Philosophy

- **Simplicity**
  - Common data types
  - Common set of parameters
  - Sensible parameters defaults for minimal need for configuration
  - Convenient deployment of components and resources
  - Compose powerful pipelines with a few lines of code

- **Modularity**
  - Use only what you need

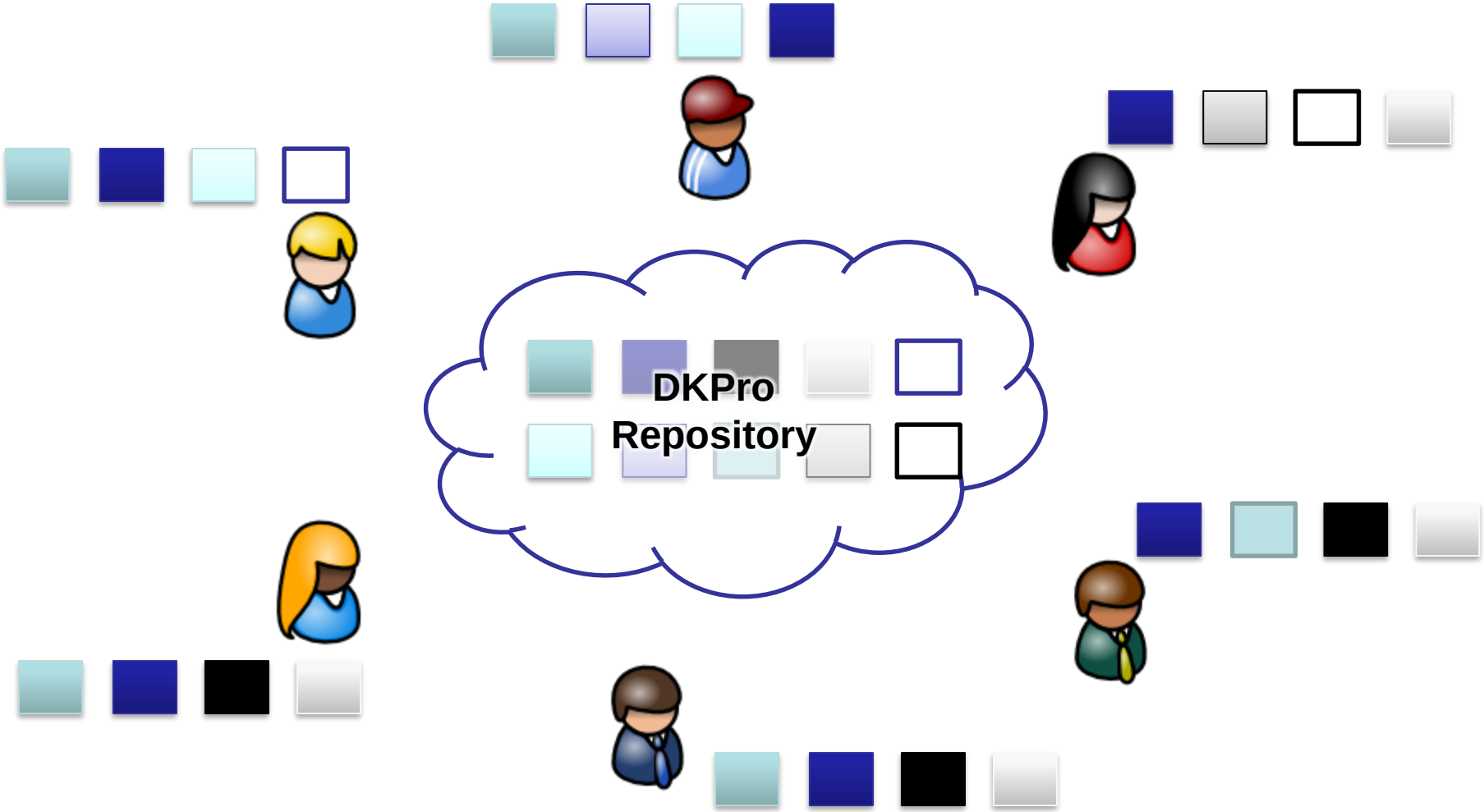
- ***Stuff has to "just work", everywhere.***

DKPro Core Philosophy

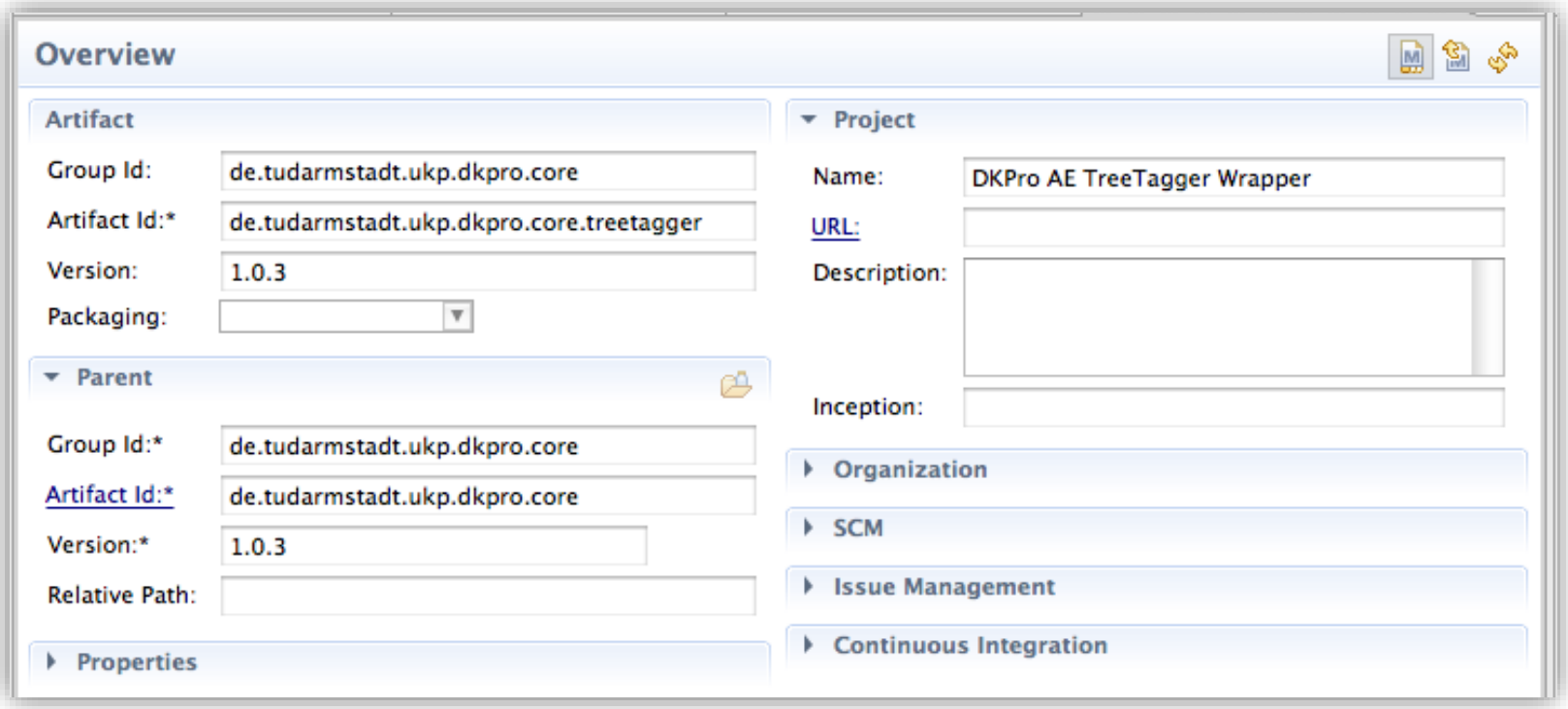
- **Flexibility**
  - Parameters override for fine-grained control
  - Data types extension with custom fields
  - Type mappings customization

- ***Stuff has to "just work", everywhere.***

Managing Deployment



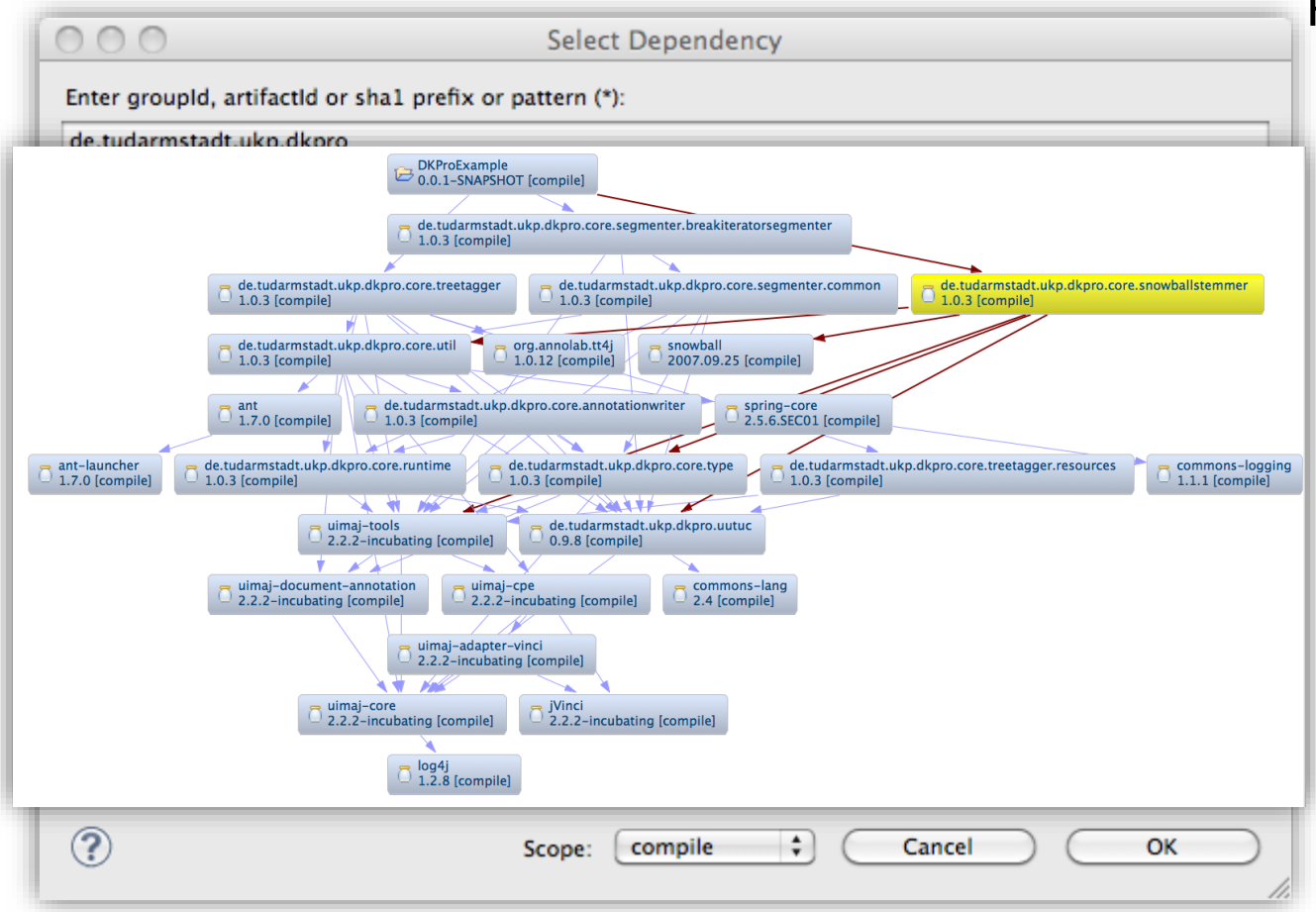
UKP OSS Component Repository
Publish component



UKP OSS Component Repository Retrieving components

TECHNISCHE UNIVERSITÄT DARMSTADT

Component Repository



Tools and formats

- **Integrated tools**
  - Stanford NLP
  - OpenNLP
  - Mate-Tools
  - ClearNLP
  - LanguageTool
  - TreeTagger
  - JWordSplitter
  - Snowball Stemmer
  - TextCat
  - MaltParser
  - MstParser
  - BerkeleyParser
  - …

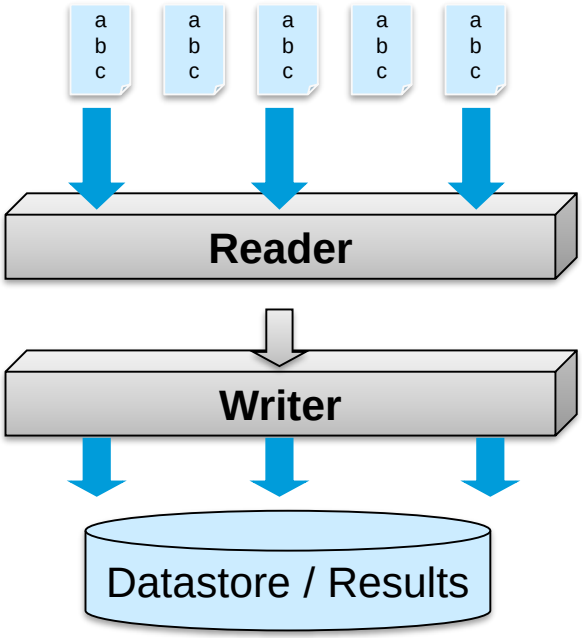
Tools and formats

- **Supported formats**
  - Text
  - PDF
  - TIGER XML
  - TEI XML
  - BNC XML
  - Negra Export
  - SQL Databases
  - Google web1t n-grams
  - …

Readers and Writers

- **Common parameters**
  - Source / target location
  - Source / target encoding
  - ANT-like patterns (for readers)
  - Language (for readers)

- **Common features**
  - Read data from file system, ZIP/JAR archives or classpath
  - Preserve directory structure on write for recursive reads



Some currently supported corpora/resources

- British National Corpus
- Wacky Corpora
- TüBa D/Z
- Tiger Corpus
- Digitale Bibliothek
- Brown Corpus
- ACL Anthology Reference Corpus
- …
- Google Web1T n-grams

Good range of pre-trained models

- Upstream models packaged for convenient deployment and use
- Additional model meta-data

- 90+ models
- 20+ tools
- 15+ languages

- Best supported
  - English (Penn Treebank Tagset, Stanford Dependencies)
  - German (STTS Tagset, Negra/Tiger)

Models/Languages

Various models covering different languages accompany the components.

DKPro Type System Overview

TECHNISCHE UNIVERSITÄT DARMSTADT

Meta Data

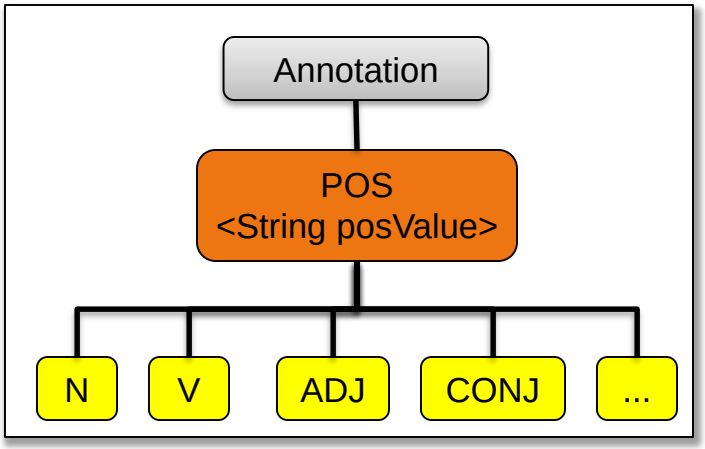
DocumentMetaData
<String documentTitle>
<String collectionId>
<String documentId>
<String documentUri>
<String documentBaseUri>

Segmentation

- Document
- Heading
- Paragraph
- Sentence
- Token
- Compound
- Split
- CompoundPart
- LinkingMorpheme
- Ngram
- StopWord

LexMorph

- POS
- Stem
- Lemma
- Morpheme



Syntax

- Constitutent
- Dependency
- Chunk

Coreference

- CoreferenceChain
- CoreferenceLink

Semantic Role Labeling

- SemanticPredicate
- SemanticArgument

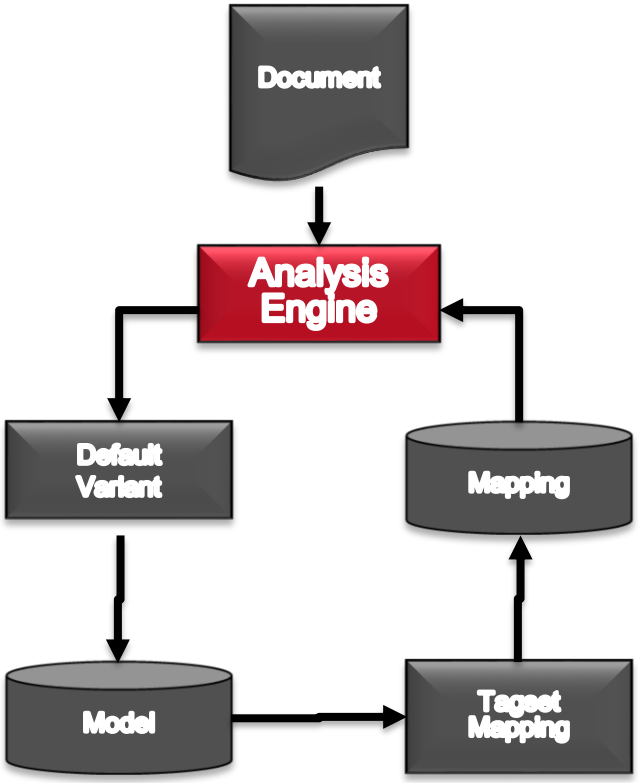
Named Entities

- NamedEntity
- Location
- Person
- ...etc...

Analysis Engines

- **Common parameters**
  - Model location
  - Model encoding
  - Model variant
  - Mapping location
  - Language

- **Common features**
  - Model loading based on document language
  - Print model tag set to log
  - Default variants



Hands-on