UIMA Presentation

Annie - Lukas - 2019.04.29

UIMA vs Apache UIMA

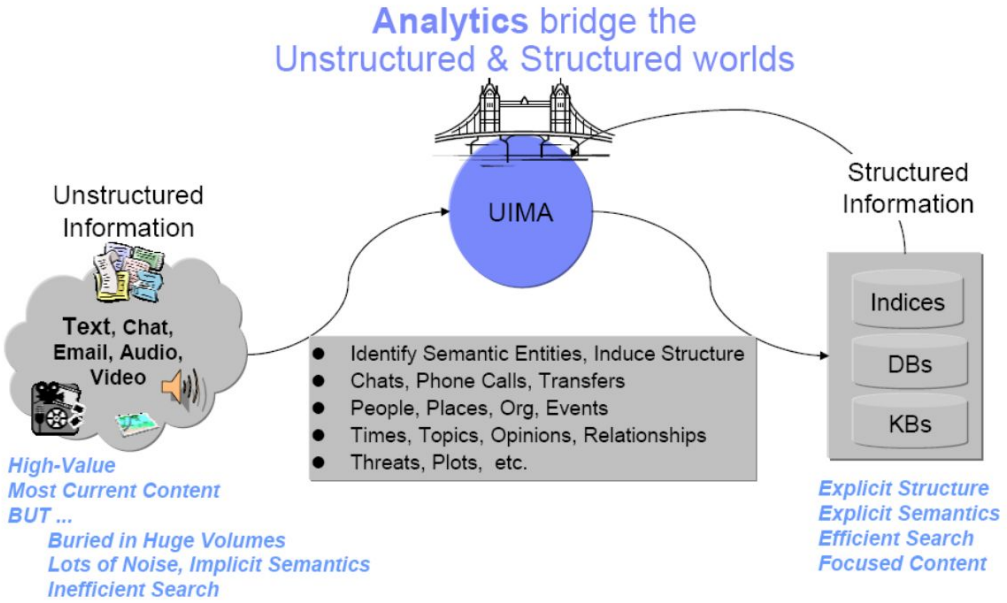
- UIMA is Unstructured Information Management Architecture is an architecture and software framework -> create, discover, compose, deploy multi-model analysis capabilities and integrate them with search technologies
- Apache UIMA framework is open source implementation of this architecture -> provide run time environment for developers to plug in and run their UIMA component implementations & with which they can build and deploy UIMA application

Apache UIMA

- Apache UIMA includes API & tools for creating analysis components, e.g: tokenizers, summarizers, categorizers, parsers, name-entity detectors
- Multi-modal support: text, audio, video
- Programming language support: mostly Java
  (also: a C++ enablement layer, Perl, Py, TCL)
- Different deployments: e.g tightly-coupled (same process) or loosely-coupled (separate process / different machines)

UIMA app

- UIMA app analyses unstructured information to discover, organize, deliver relevant knowledge



- UIMA helps to build the bridge between unstructured and structured worlds

Why there’s need an architecture?

- Apps can make use a variety of technologies (e.g statistical & rule based NLP, IR, ML, automated reasoning, WordNet, FrameNet …)
- These technologies are developed independently using different techniques, interfaces, platforms
- To make the best use of them, we should be able to exchange and combine them, but the integration is costly

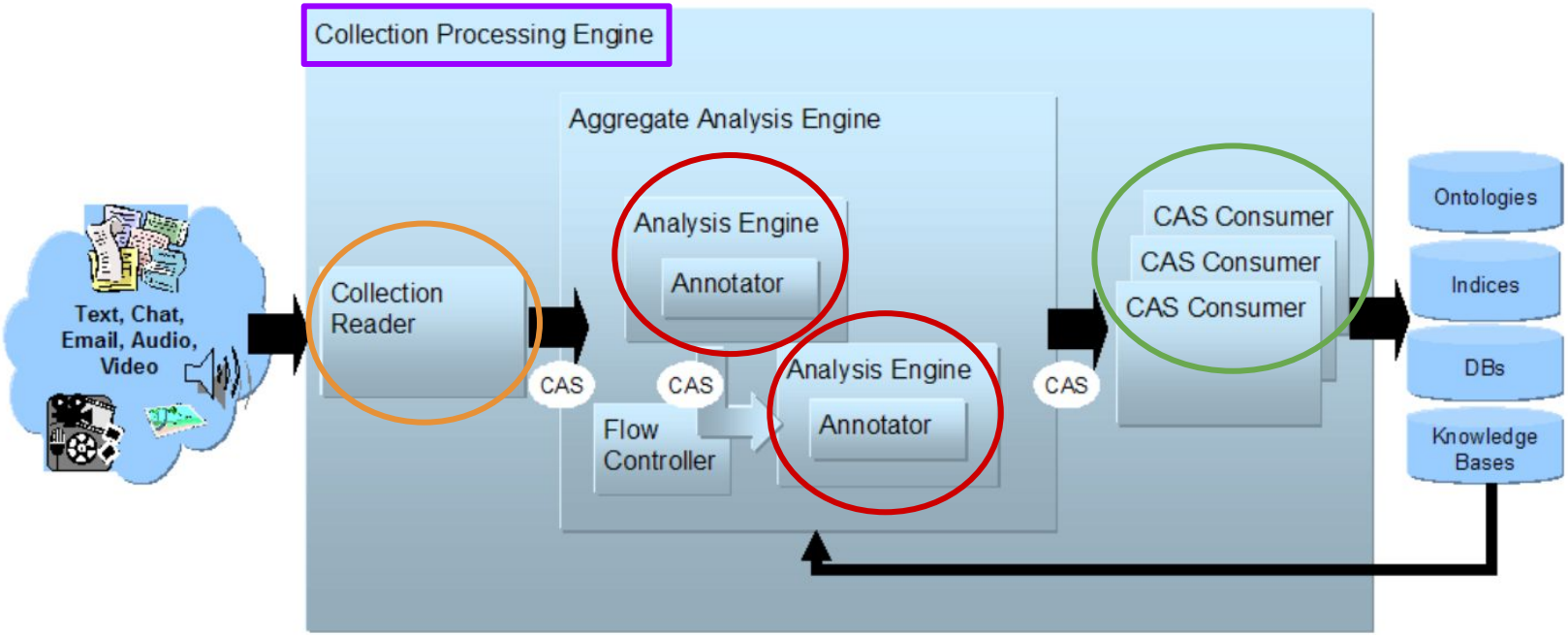
Some tasks and how to solve them

- Examples of UIMA app:
  - Process medical abstracts to discover critical drug interactions
  - Process documents to find key evidence indicating probable threats
- How?
  - Unstructured data is analyzed to interpret, detect and locate concept of interests, e.g: name entities (person, organization, …)
  - (more challenging) opinions, complaints, threats, facts, …
  - Relations e.g finances, supports, purchased, repairs, located in
  - >> each of the above steps can be an analytics component

How UIMA helps you?

- UIMA makes it easy to construct analytics components and combine them
- Helps to match suitable tools to part of solution, enable rapid integration, support all type of deployments
- Search, database, data mining engines can deliver newly discover content in response to request / queries

High-Level Component Architecture



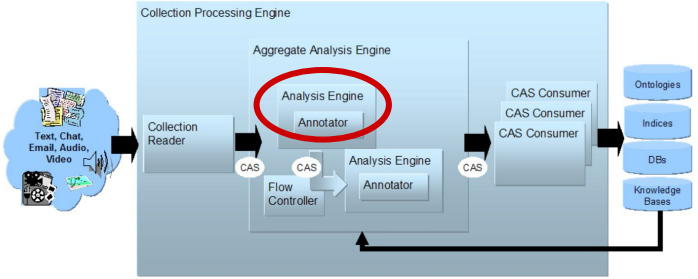
Collection Processing Architecture

In typical case, the application generally follows these steps:

1. Connect to a physical source
2. Acquire a document from the source
3. Initialise a CAS with the document to be analyzed
4. Send the CAS to a selected analysis engine
5. Process the resulting CAS → Analysis Engine
6. Go back to 2 until the collection is processed
7. Do any final processing required after all documents have been analyzed → CAS Consumer

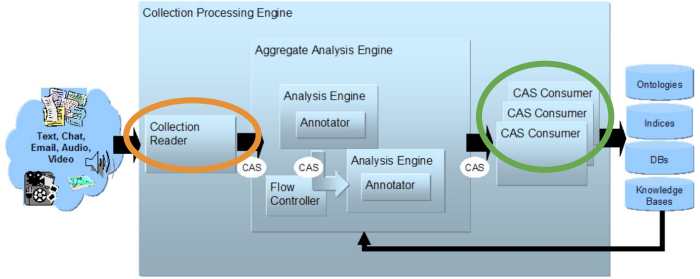
Collection Reader

Main Components



- **Analysis Engines (AEs)** are composed to analyze a document and infer and record descriptive attributes about document as a whole, and/or about regions therein. We can combine different AEs to get an **Aggregated Analysis Engine**
  - **Annotator**: some code that analyses the documents and outputs annotations on the content of document
  - > UIMA framework takes the annotator to produce AE
  - **Annotator Developer** is the one who codes the analysis algorithm

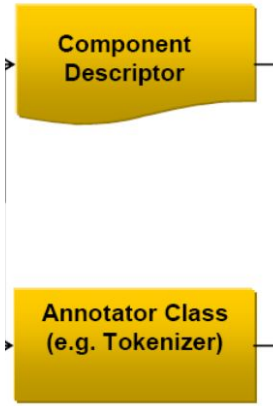
Main Components



- **Collection Reader** is to connect to and iterate through a source collection, acquiring documents and initializing CASes for analysis
- **CAS Consumers** function at the end of the flow to do the final CAS processing, e.g to index CAS contents in a search engine, extract elements of interest and populate a relational database or serialize and store analysis results to disk for subsequent and further analysis

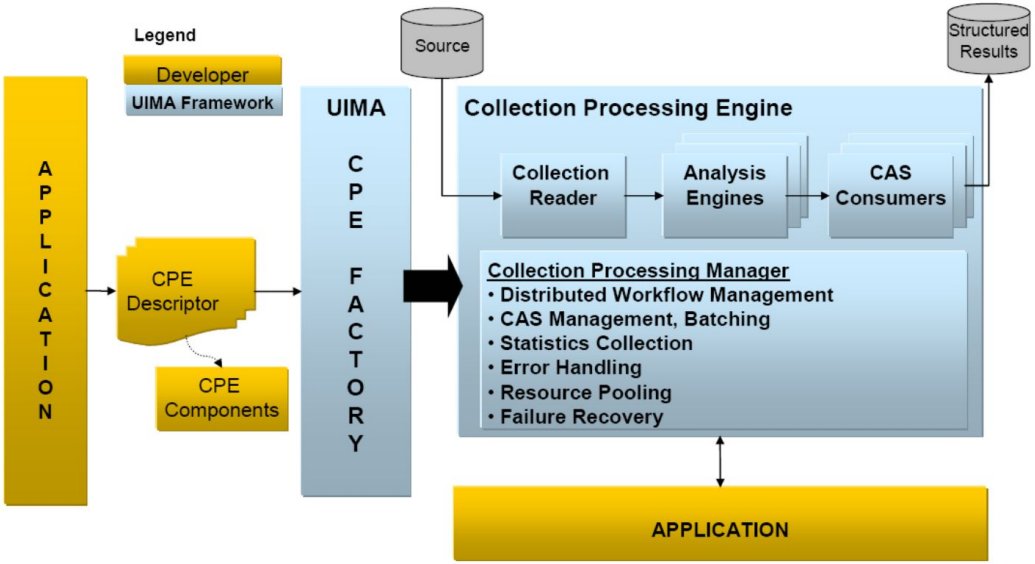
Component in UIMA

- For every component specified in UIMA, there are two parts required for its implementation:
  - Declarative part - **Component Descriptor** (in XML): contains metadata describing the component, e.g identity, structure, behavior. It also identifies type require for input and output CAS
  - **Code part - program in Java**; if you create an aggregate engine, no need to create new code; just point to other components where code was included



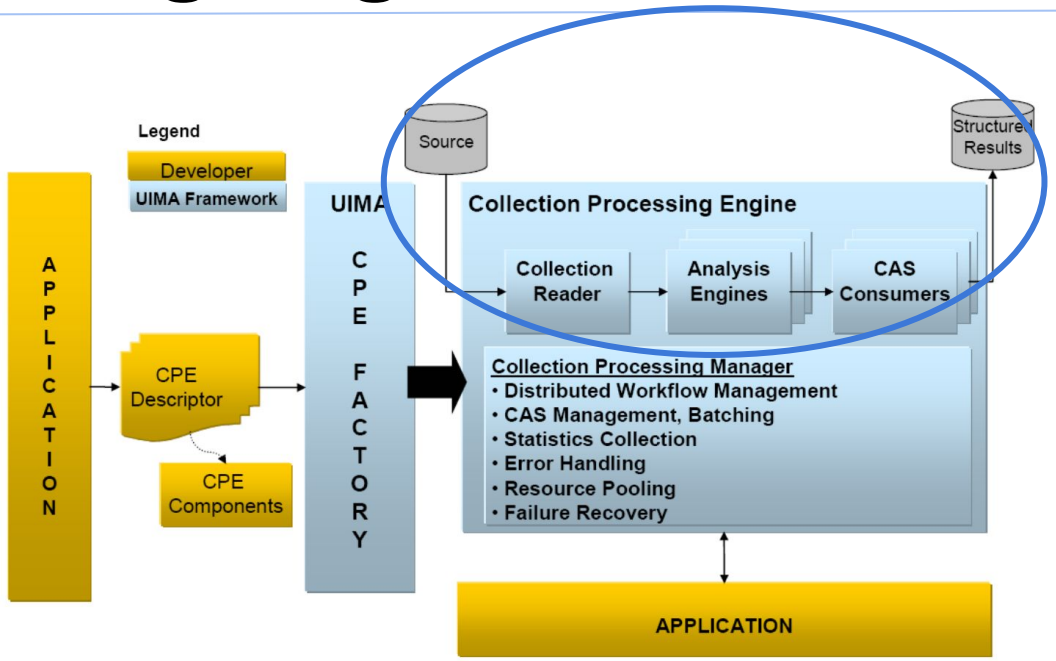
Collection Processing Engine

- Is an aggregate component that specifies a “source to sink” flow from a collection reader through a set of analysis engines and then to a set of CAS Consumers



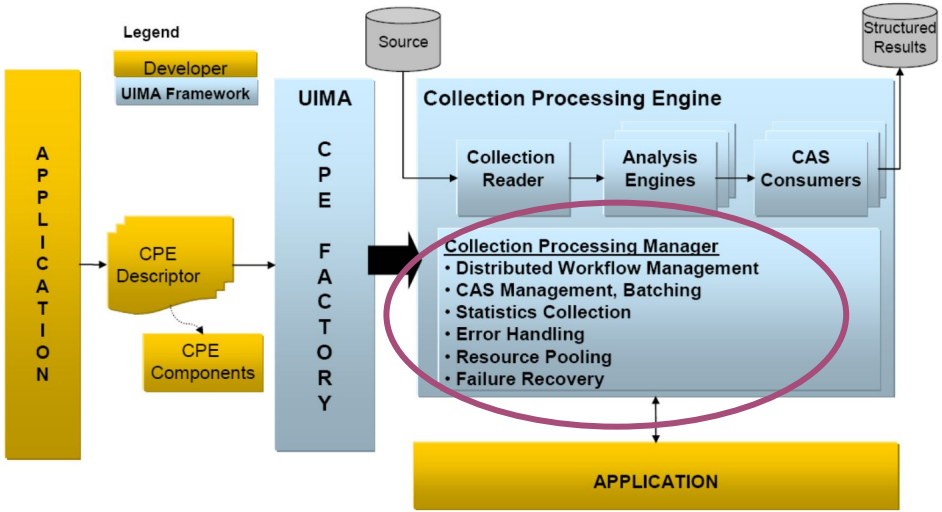
Collection Processing Engine

- CPEs are specified by XML files called CPE Descriptor, point to their contained component



Collection Processing Engine

- Collection Processing Manager (CPM) is capable of reading CPE descriptor, deploying and running the specified CPE. Key features are failure recovery, CAS management and scale-out, also faults log.

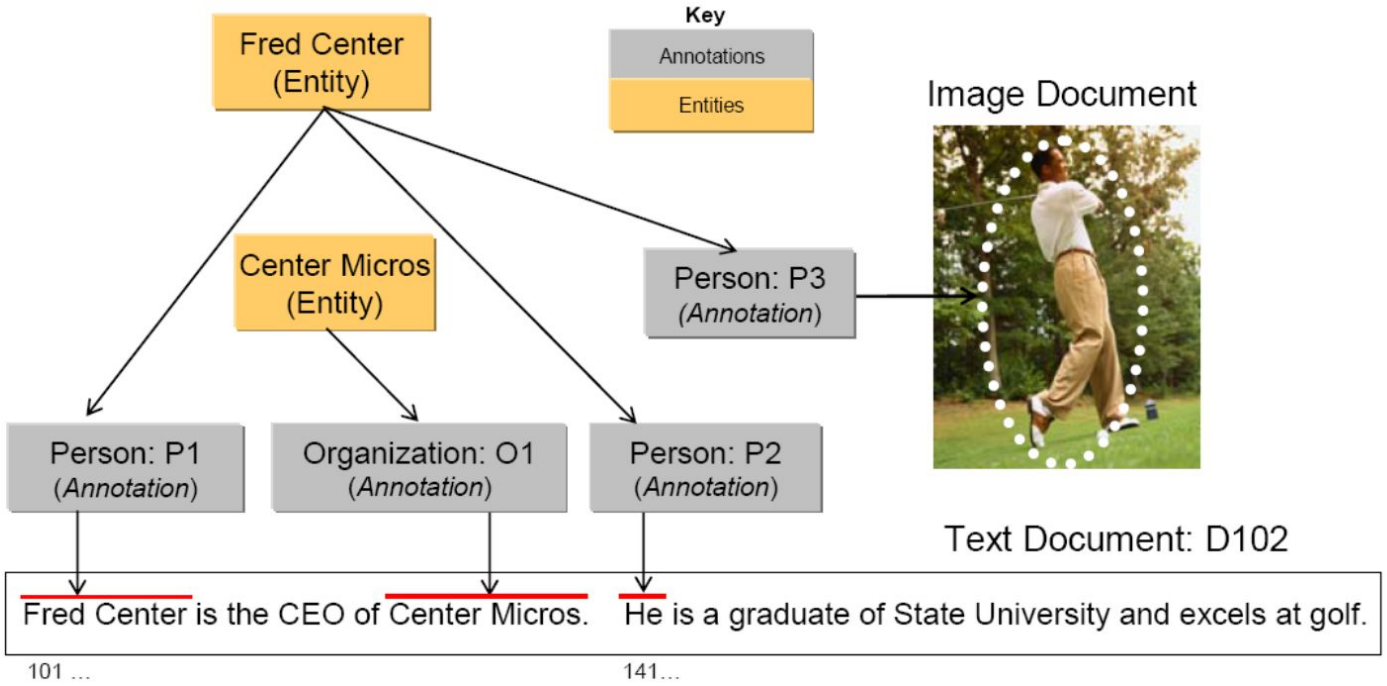


- CPM runs in a separate process or different machine from CPE components

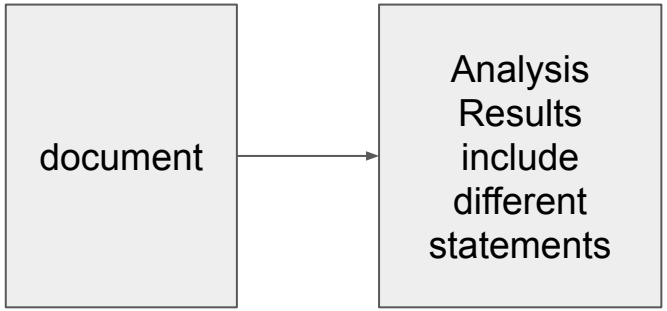
Roles and Interactions with UIMA

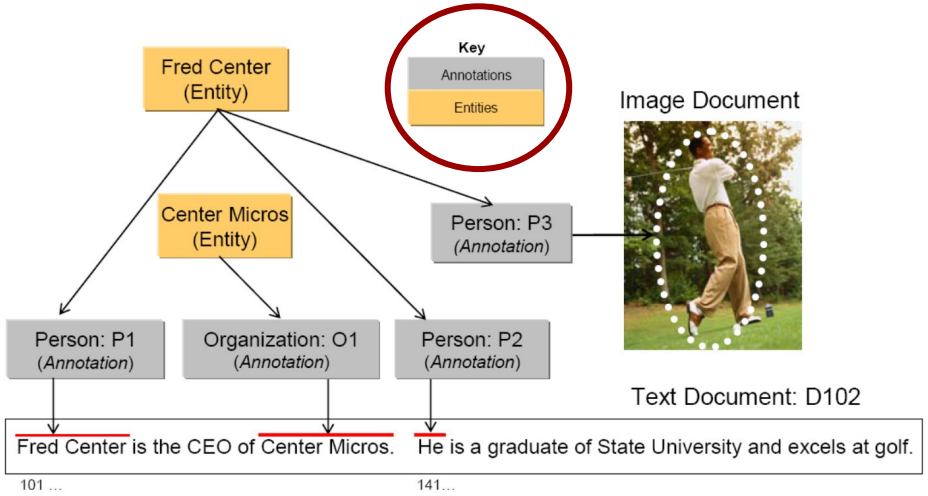
- UIMA component developer interacts with CAS, UIMA Context interfaces.
    - Through CAS interface, annotator developer interacts with document and read/writes the analysis results. He can also obtain indexed iterators to different objects in the CAS through a suite of access methods. Java annotator developers interacts with JCas -> interface to CAS objects, e.g Person type is rendered as Person class
    - Component developer can also access UIMA Context - the resource manager interface to ensure different annotators working together in an aggregate flow may share the same instance of a resource via

Analysis Engines and Annotators



Analysis Engine






```
(1) The Topic of document D102 is "CEOs and Golf".
```

```
(2) The span from position 101 to 112 in document D102 denotes a Person
```

```
(3) The Person denoted by span 101 to 112 and
  the Person denoted by span 141 to 143 in document D102
  refer to the same Entity.
```

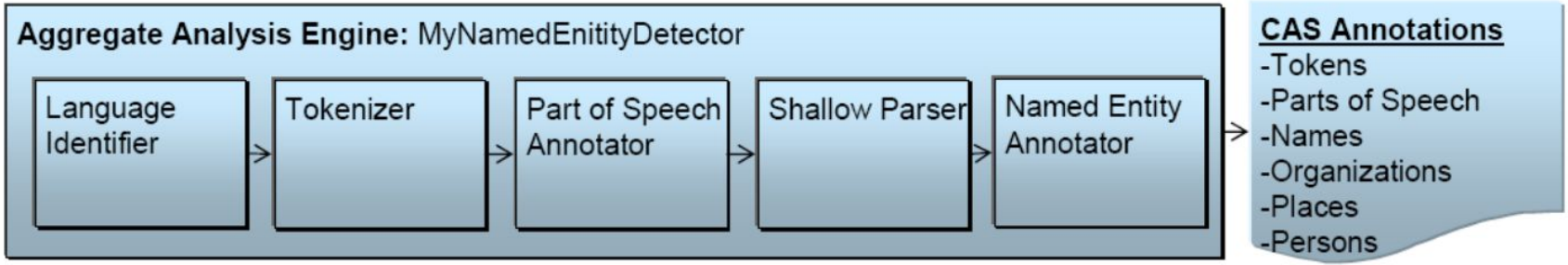
Terminologies regarding AE

- **Span** to refer to sequence of characters in the text document
- **Type** is predefined, e.g : Topic / Person
- AEs produce **analysis results**
- Analysis results can be represented in **CAS** (Common Analysis Structure), which is a **data structure**, **logically** contains the document being analyzed (it would create objects which link to corresponding span of text)

Terminologies regarding AE

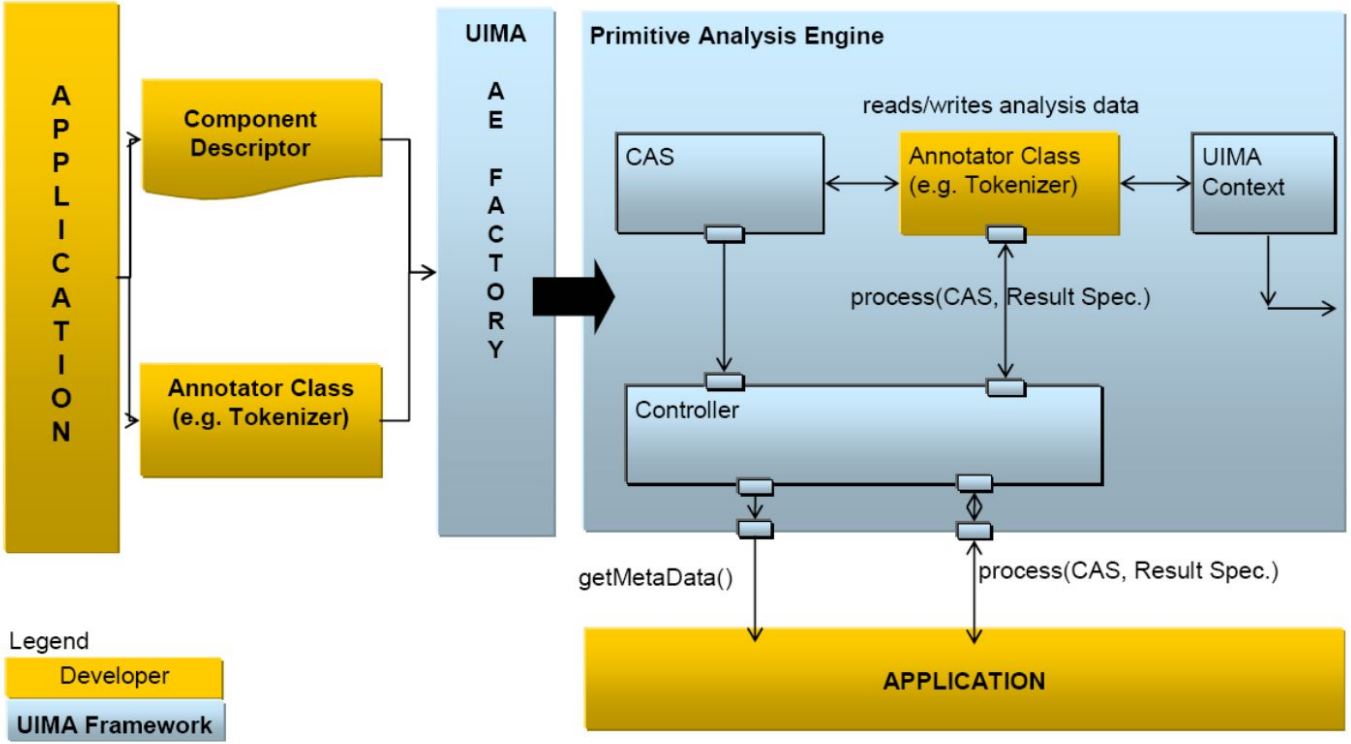
- A CAS is associated with a single artifact being analysed by a collection of UIMA analysis engines. It may have **independent views**, e.g different translations can be analysed independently, and possibly by different AEs. Each view contains a different **Subject of Analysis (Sofa)**

Aggregate Analysis Engines - Example



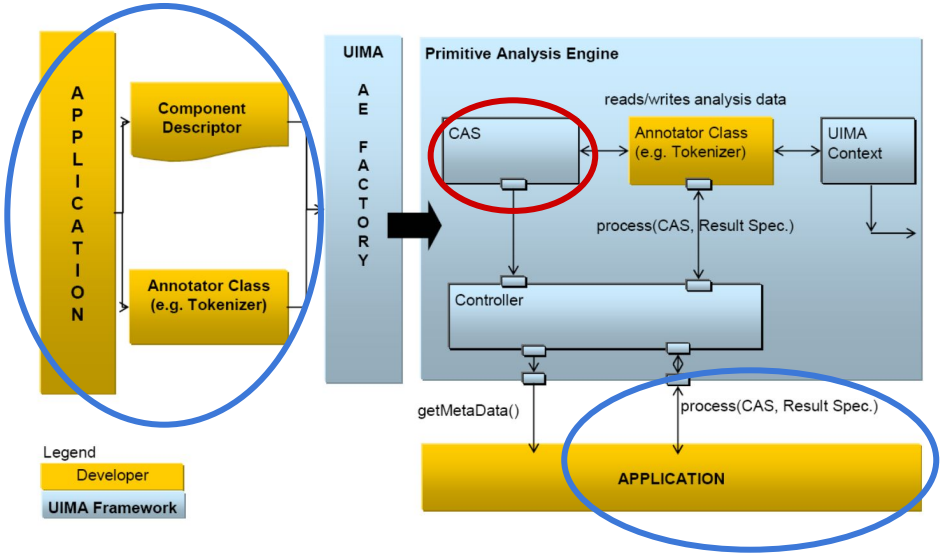
- This AE performs named entity detect, which includes pipeline of annotator starting with language detection, to tokenization, then part-of-speech detection, and deep grammatical parsing and finally named-entity annotator. Each step is required by the subsequent analysis. This is handled by **flow controller**

Create and Interact with an AE



Create and Interact with AE

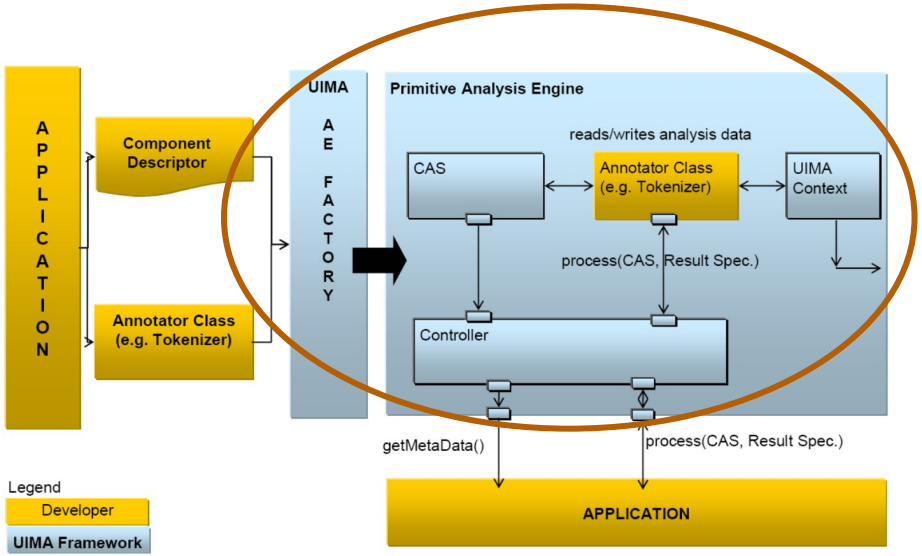
- Basic AE interface can be a simple CAS in / CAS out.
- The application is responsible for interacting with UIMA framework to instantiate an AE, create or acquire input CAS



- Also, initialize input CAS with a document and pass it to AE through process method

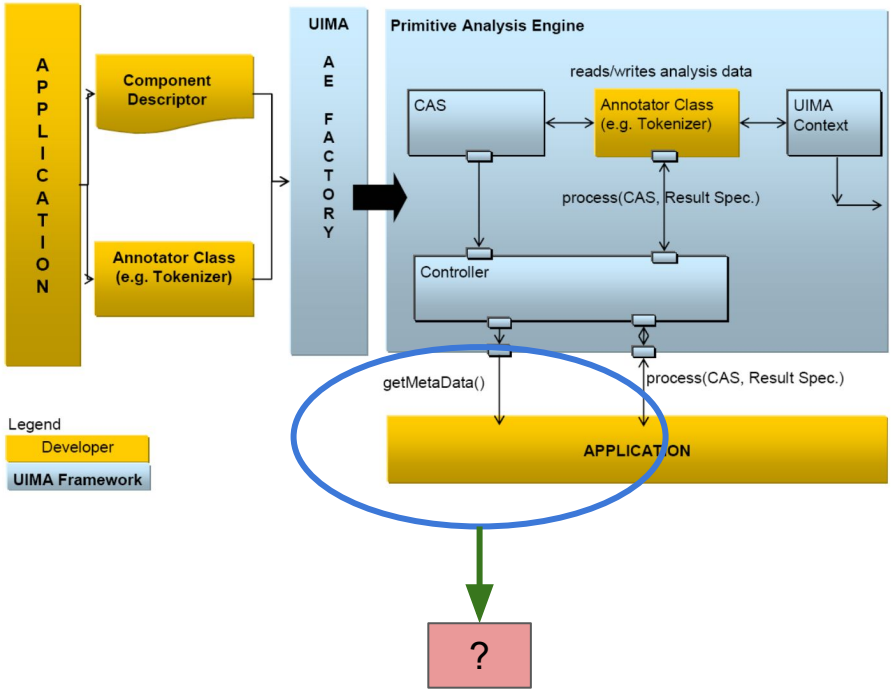
Create and Interact with an AE

- UIMA AE Factory takes the information from component descriptor and class files implementing annotator, instantiates the AE instance, setting up CAS and UIMA Context



Create and Interact with an AE

- The application then decides what to do with the returned CAS. Some possibilities are display the results, store the CAS to disk for post processing, extract and index analysis results as a part of search or database application



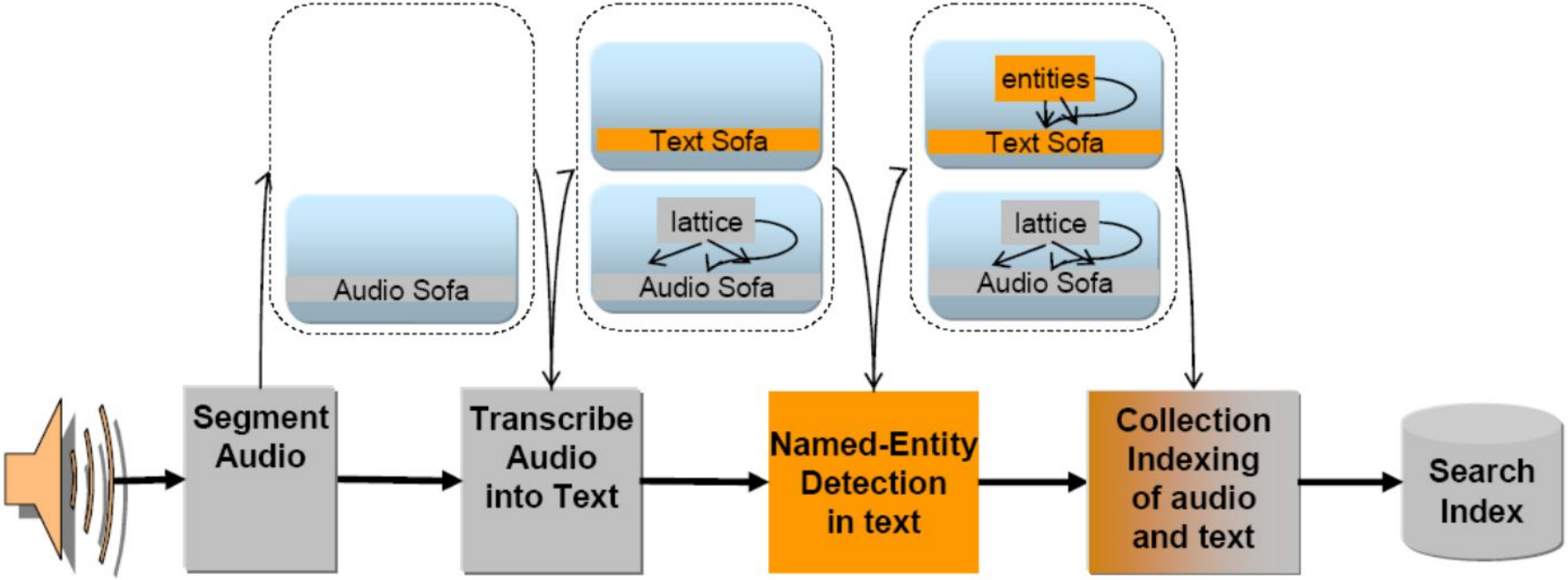
CAS Consumer - Example

- Semantic Search
  - Is a search paradigm that can exploit the additional metadata generated by analytics like a UIMA CPE
  - Can support query language called XML Fragments
  - Eg1: Produce documents contain “center” where it appears as part of a mention annotated as an organization
  - Eg2: produce documents contain a mention of an organization with “center” as part of its name where the organization is mentioned as part of a CEO-of relationship.

```
<organization> center </organization>
```

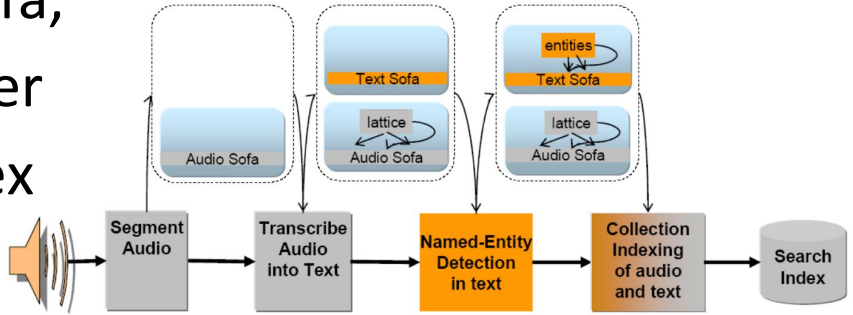
```
<ceo_of>
    <person> center </person>
    <organization> center </organization>
<ceo_of>
```

Multimodal Processing

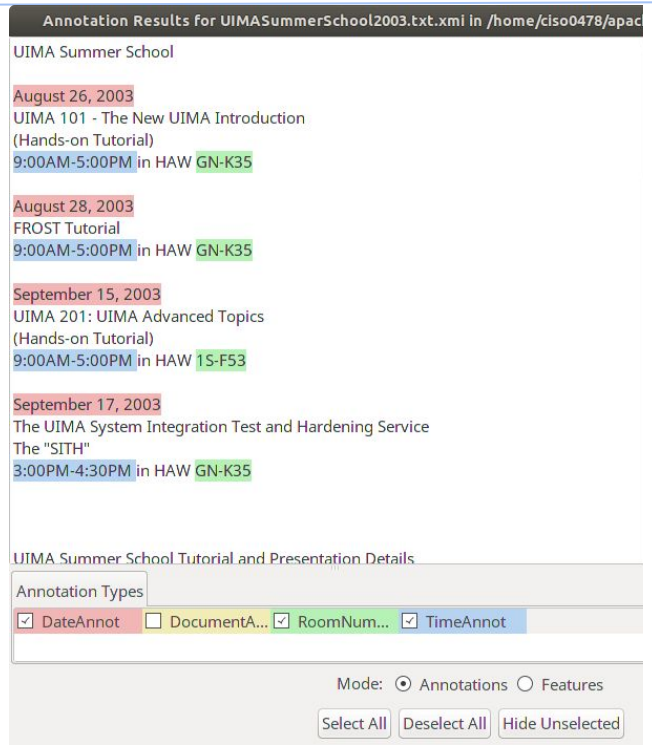


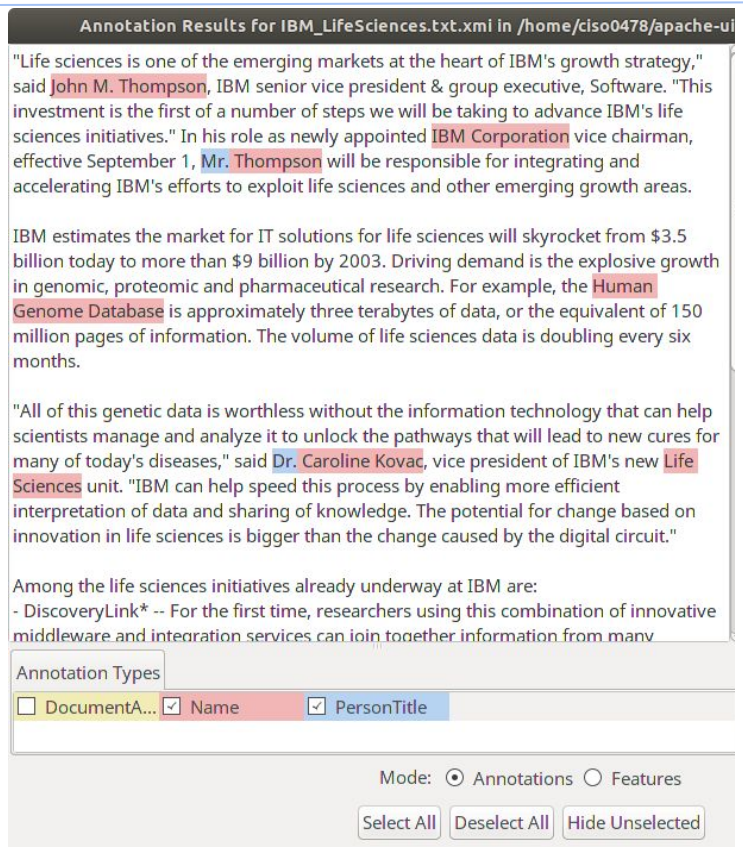
Multimodal Processing

- This processing pipeline starts with an audio recording of a conversation, transcribe the audio into text then extracts information from the text transcript.
- AE start at analyzing an audio Sofa, and later text Sofa. CAS Consumer likely want to build a search index from concepts found in text -> original audio segment covered by that concepts

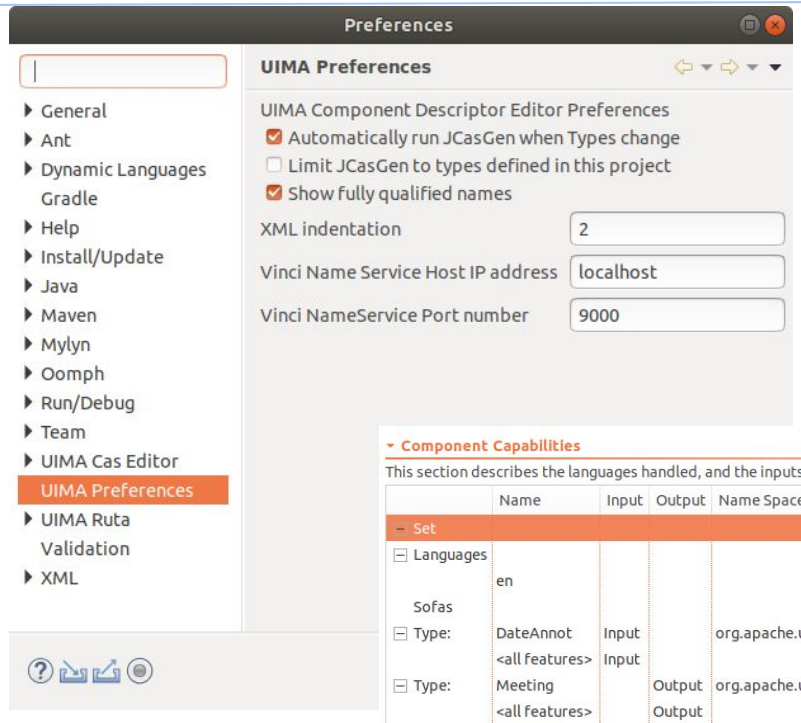


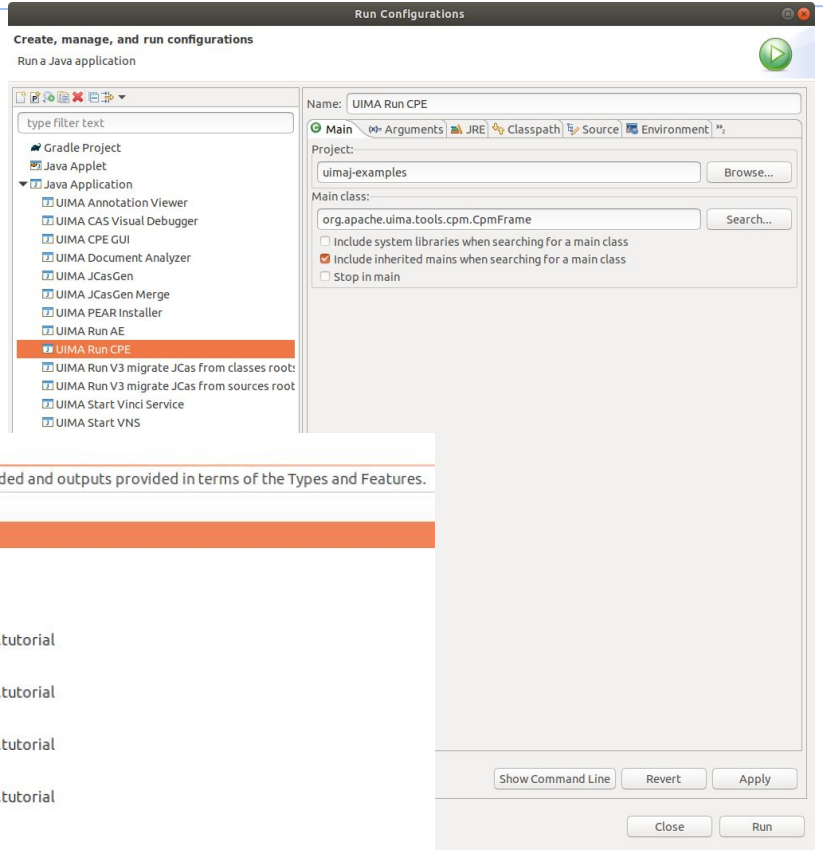
Screenshots - Annotation Results

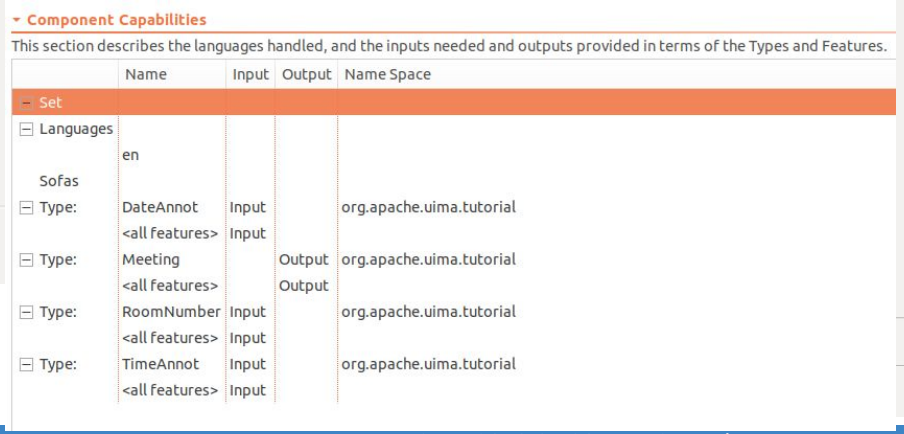




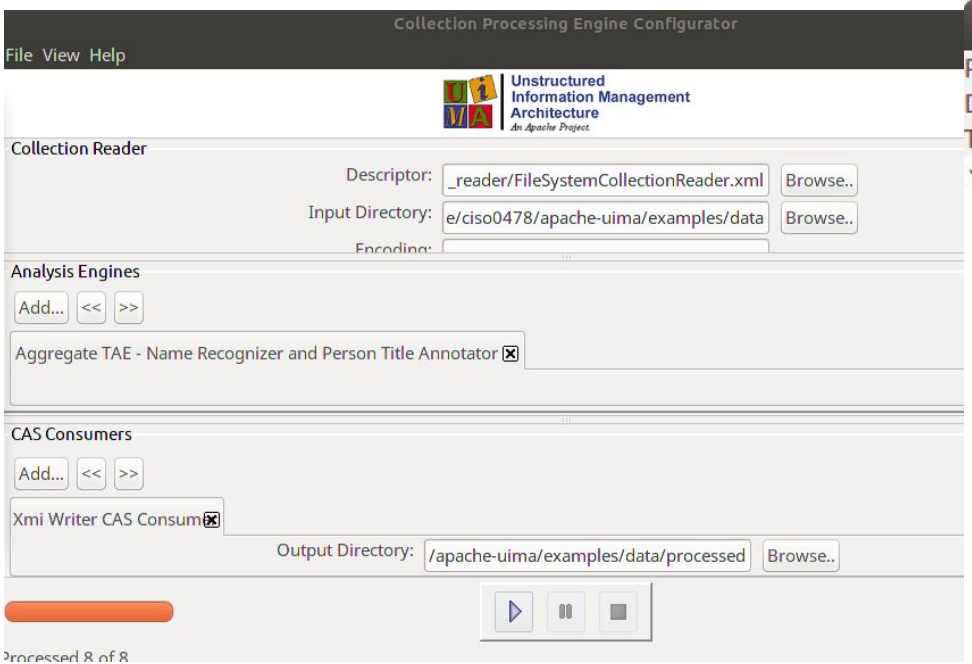
Screenshots - Windows







Screenshots - CPE



Screenshots - java code structure

- FileSystemCollectionReader.java
  - FileSystemCollectionReader
    - PARAM_ENCODING
    - PARAM_INPUTDIR
    - PARAM_LANGUAGE
    - PARAM_SUBDIR
    - mCurrentIndex
    - mEncoding
    - mFiles
    - mLanguage
    - mRecursive
    - addFilesFromDir(File) : void
    - close() : void
    - getNext(CAS) : void
    - getNumberOfDocuments() : int
    - getProgress() : Progress[]
    - hasNext() : boolean
    - initialize() : void

- XCasWriterCasConsumer.java
  - XCasWriterCasConsumer
    - PARAM_OUTPUTDIR
    - mDocNum
    - mOutputDir
    - initialize() : void
    - processCas(CAS) : void
    - writeXCas(CAS, File) : void
  - package.html

AITools UIMA Library

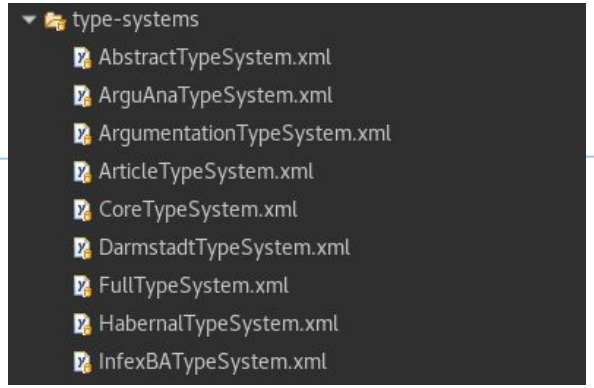
The AITools Uima Library offers:

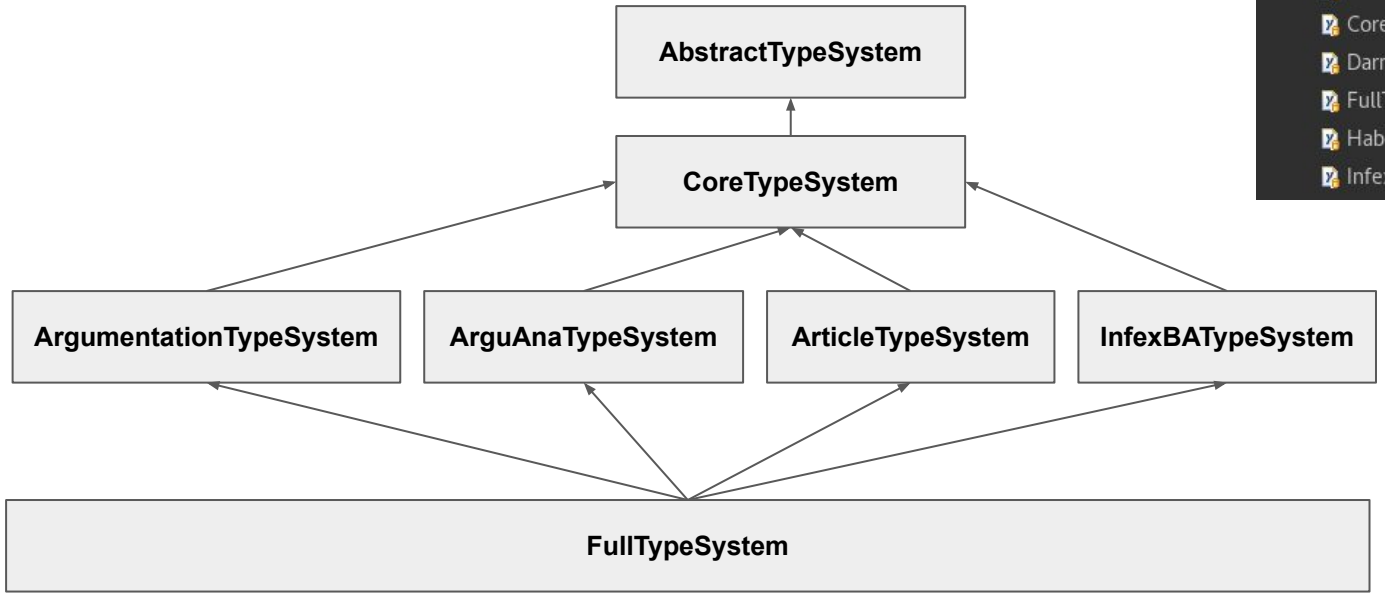
- Type Systems
- Collection Readers
- Analysis Engines (Primitives and Aggregates) for different Text Analysis Tasks
- Feature Extraction based on Analysis Results produced by UIMA

AITools UIMA Library

- Type Systems, Collection Readers, Analysis Engines directly use the UIMA Framework
- Feature Extraction does not use UIMA directly, but extracts Features based on the Annotation produced by UIMA

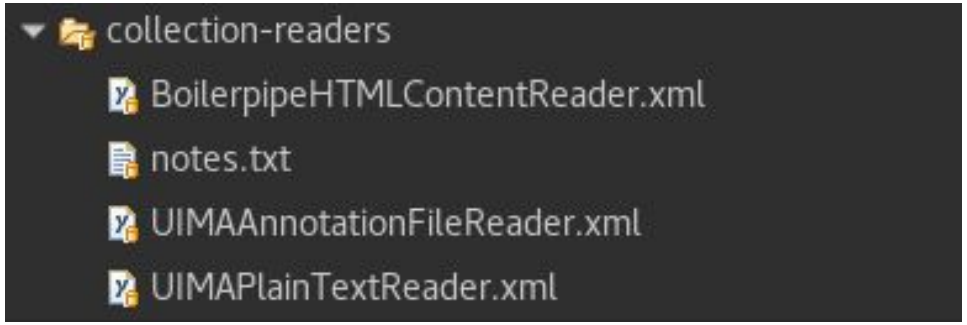
AITools - Type Systems



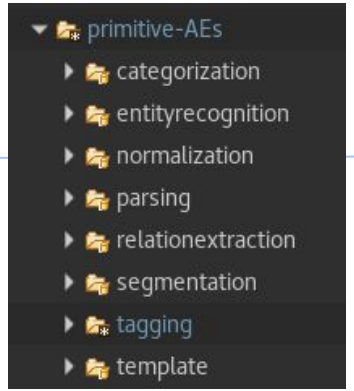


AITools - Collection Readers

- Collection Reader for most common file Types:
  - HTML-Files
  - UIMA-XMI-Files
  - TXT-Files



AITools - Analysis Engines



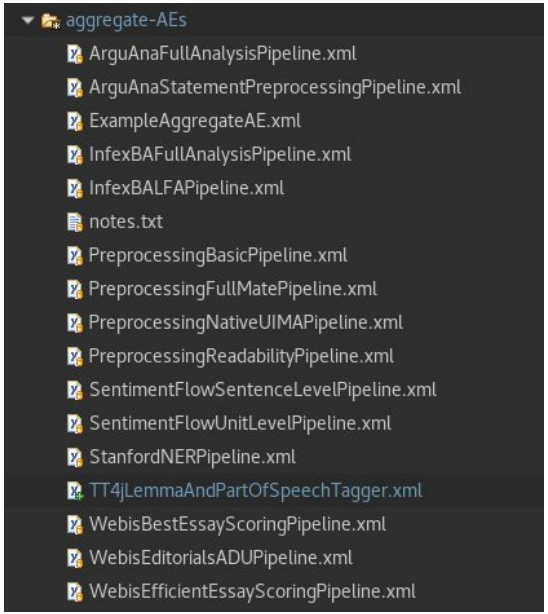
Primitive Analysis Engines:

- Different primitive Analysis Engines that implement Algorithms for different Text Analysis Tasks
- Some are general purpose and can be reused in different projects
- Some serve a task specific to a project and therefore can’t be reused for other projects

AITools - Analysis Engines

Aggregate Analysis Engines:

- Full Analysis Pipelines for different Tasks

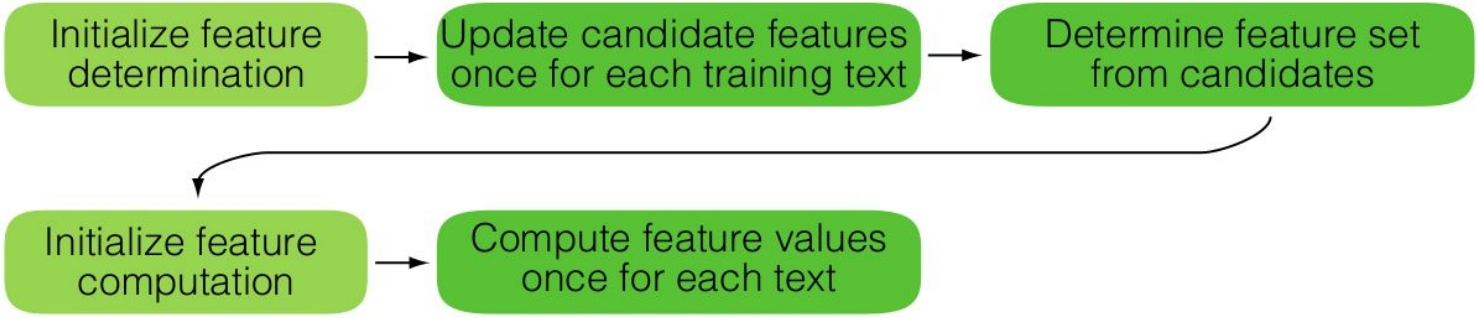


AITools - Feature Extraction

- Feature Extraction is not part of UIMA itself, but extracts Features based on the Annotations produced by the Analysis Engines
- Aggregate Feature Types can be constructed out of single Feature Types

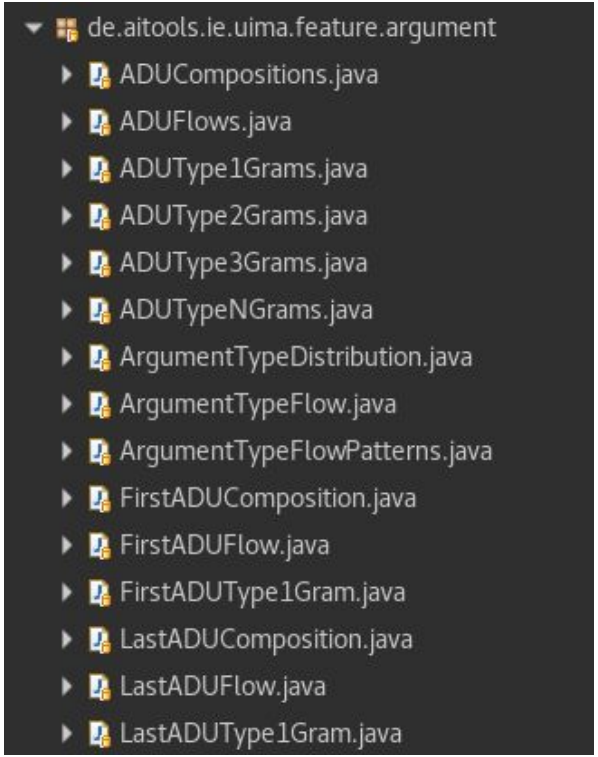
AITools - Feature Extraction

- Features can be added by using the Feature Interface that each Feature must implement



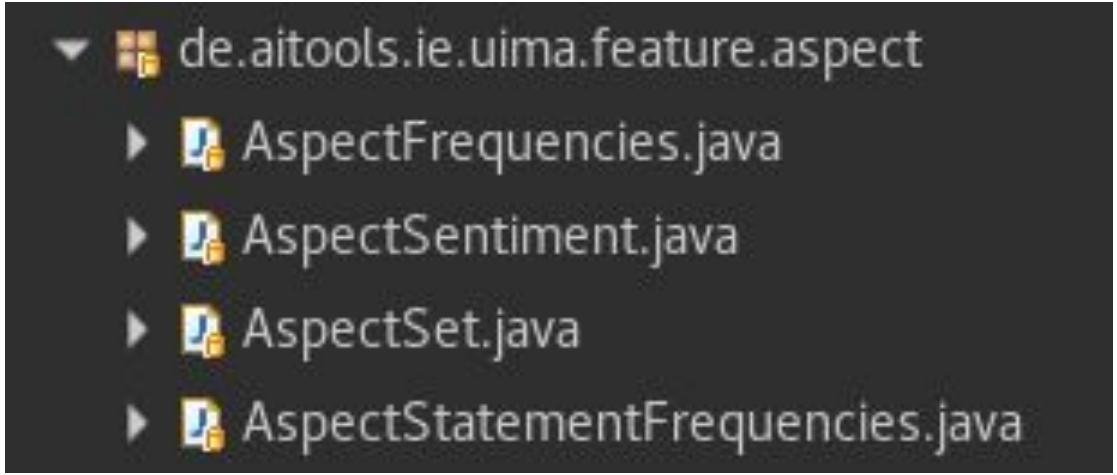
AITools - Feature Extraction

Argument-Features:



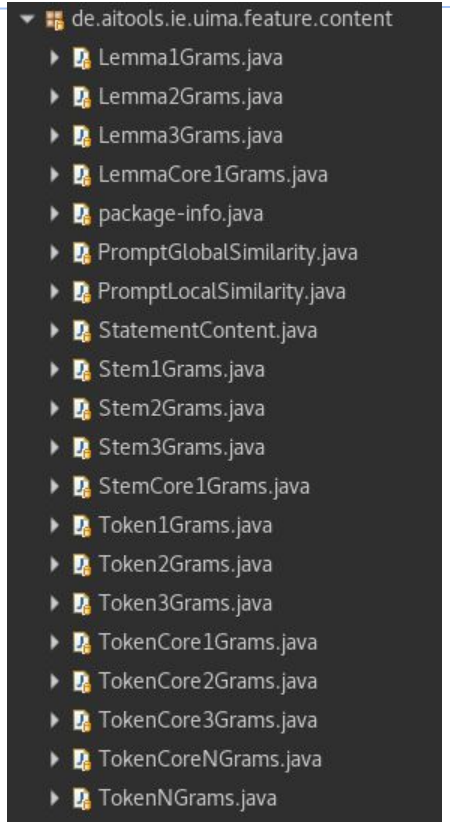
AITools - Feature Extraction

Aspect-Features:



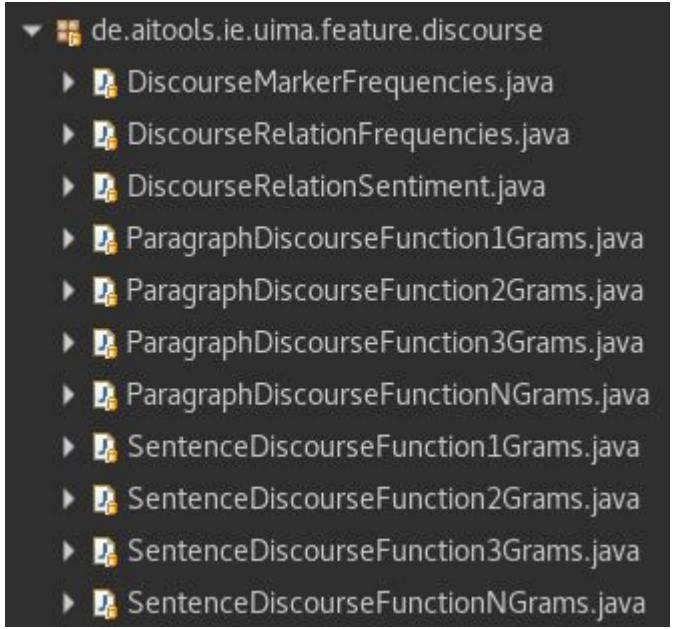
AITools - Feature Extraction

Content-Features:



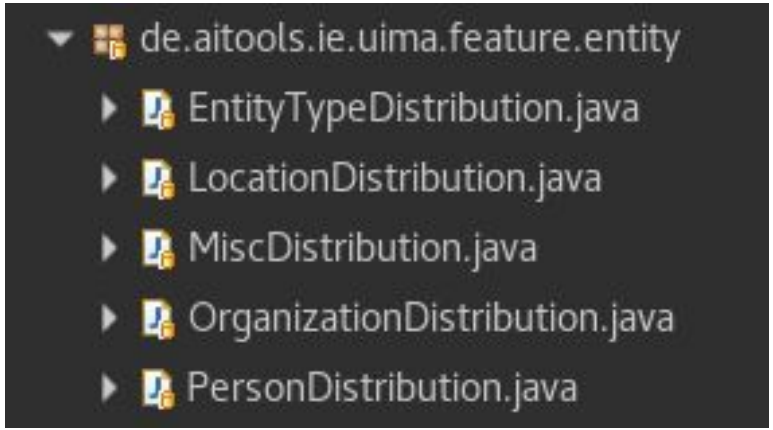
AITools - Feature Extraction

Discourse-Features:



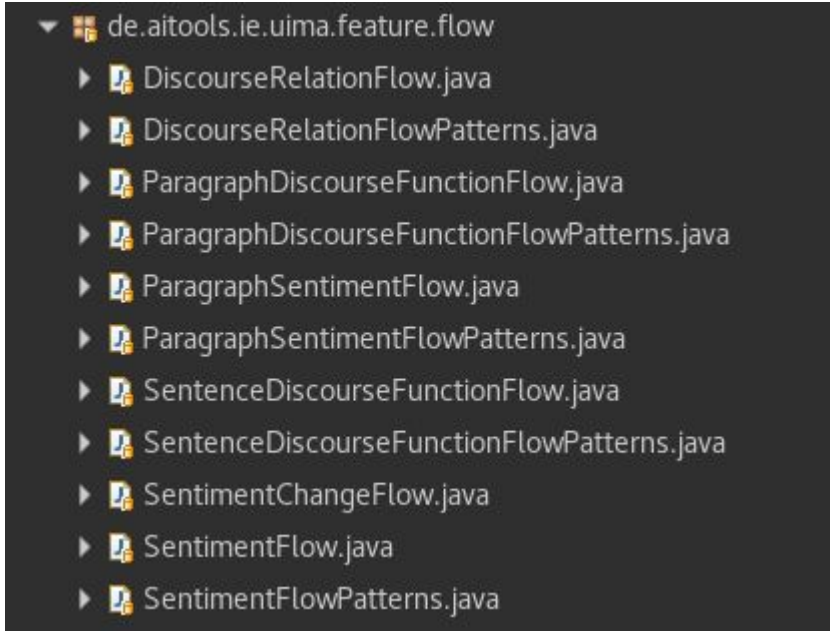
AITools - Feature Extraction

Entity-Features:



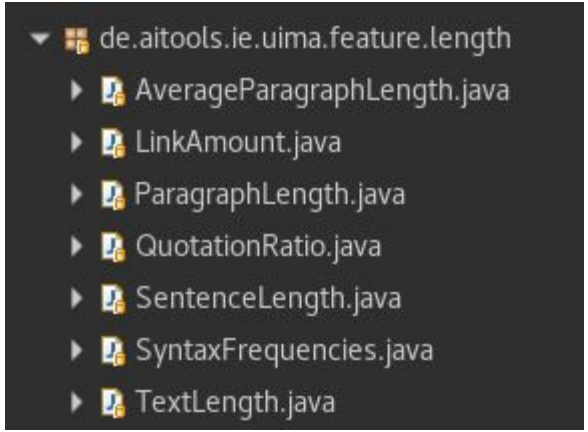
AITools - Feature Extraction

Flow-Features:



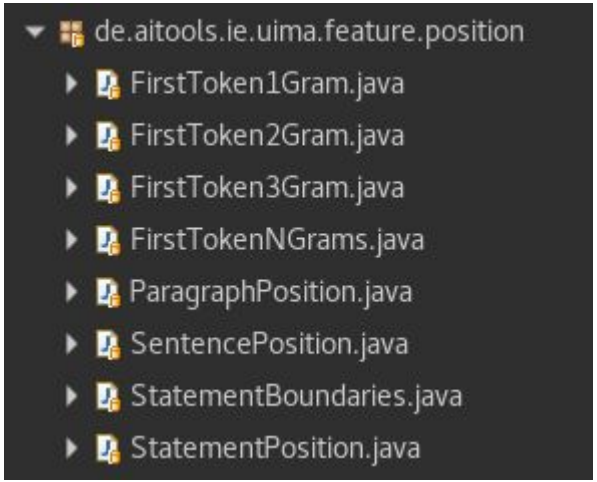
AITools - Feature Extraction

Length-Features:



AITools - Feature Extraction

Positional-Features:

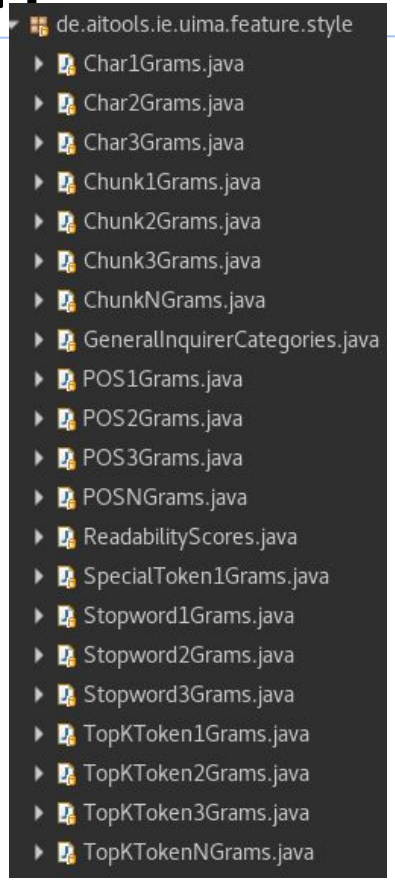


AITools - Feature Extraction

Sentiment-Features:

AITools - Feature Extraction

Style-Features:



Thank you for your Attention.