OASIS UIMA Technical Committee
Specification Overview

February 1, 2008
Outline

- Overview
- Status
- Design Goals
- Specification Elements
- Impact on Apache UIMA SDK
The UIMA Standard

• Platform-Independent Data Representations & Interfaces for Text & Multi-modal Analytics

• Enable Interoperability of Text & Multi-modal Analytics
  – Exchange Analysis Data
  – Exchange Analytic Metadata (Descriptions of what analytics do)
  – Interface with analysis applications at the Services Level
Analytics can detect a broad range of semantic types in text for example.
...in Different Languages...
...and in Different Domains...

DEFINITION OF PATIENT COHORTS

Data Discovery and Query Builder

Clinical Note (Document ID is 72432230)

Reason for Visit

Outpatient note. The patient returns to the Hospital. He is status post consolidation chemotherapy with high-dose ara-C for his
AML M4.

History of Present Illness

Please see detailed note from October. The patient continues to do well as an outpatient. He denies any mouth pain—no
nausea or vomiting. He has been eating and drinking without any difficulty. He denies any chest pain or shortness of
breath. The patient did complete his prednisone eye drops yesterday, and he denies any eye discomfort.

Current Medications

Voriconazole 200 mg twice daily, Lisinopril 10 mg once daily, Hydrochlorothiazide 12.5 mg once daily, Lopressor 75 mg
twice daily, Vitamin C 500 units once daily, Multivitamin

Physical Exam

Respiratory rate is: 18. General: Alert and oriented, in no acute distress. ENT: Oral mucosa is pink and moist, no lesions.
Hemat: O2-95. Rhythm is regular. Lungs have no rales or rhonchi. 3D usage: Cefuroxime Na 500

Highlighting Controls

- Free-text Query Hits
- Annotation Data Query Hits
- RxNorm Drugs
- MeSH Diagnosis
- MeSH Signs and Symptoms

Select All
Deselect All

Annotation Data (Held)

chest pain
Annotation Type = Signs and Symptoms
Annotations
MeSH Concept ID = D002637
Concept Name = Chest Pain
Certainty = -1
Confidence = 0.0
Debt Type: Senior Note
Issue Date: October 16, 2002
Maturity Date: October 2012
Dollar Amount: $300 million
Rate: 5.875%

- Find corporate debt buried within unstructured text of SEC filings
  - Identify all debt references, including types, dates, amounts, rates and other terms
- Highlight relevant concepts to simplify research
- Create index and table of contents of debt references for easier searching and navigation
- Create alerts to monitor for key indicators
...and in Different Modalities...
Need For A Standard

• Many Independently Developed Analytics
  – Parsers, Tokenizers, Entity Detectors, Topic Detectors, Summarizer, Classifiers, Speech Transcription, Video Analysis, Translation etc.

• Higher Level Applications need to mix and match
  – Business Intelligence, National Security, Healthcare, Customer Relationship Management, Web Self Service, Bioinformatics, Content Analytics etc.

• Support for interoperability is replicated over and over in industry, research and academia
The Apache UIMA SDK

• Broadly adopted Java Implementation
• Supports analytic development, composition and deployment
• Open Source at Apache
• Informed THIS Standard
• To support full compliance with Standard
Summary Impact on Apache UIMA
(assumes knowledge of Apache UIMA)

• Generalizing Analysis Data Representation
  – Regional References (how to point from meta-data into an artifact)
  – Behavioral Metadata (unambiguous, expressive description of analytic function)
  – Views (more general reference to partitions of analysis data)
  – Naming Conventions
  – Uniform treatment of data elements (flexibility in subject of analysis)
    – Details here

• Alignment with existing standards
  – XMI: for representing CAS as object graph in XML
  – UML: for describing type-system
  – WSDL: for Analytic Services
  – OCL: formal foundation for semantics of behavior metadata
Status Update

• The UIMA TC has met bi-weekly for 10 months and has completed a full review of the research report contributed by IBM as a initial proposal for a standard for interoperable text and multi-modal analytics based on UIMA.

• The UIMA TC will integrate all revisions gathered in review reports and meeting minutes into a formal specification draft by Feb 1, 2008.

• The UIMA TC will then conduct final votes on draft sections and any outstanding issues that remain.

• The UIMA TC will publish a final draft of the specification by end of March 2008.
Introduction to UIMA Standard
Example Use Case

### Analytics

**Relationship**

**Named Entity**

**Parser**

**Analysis Results**
*(i.e., Artifact Metadata)*

**Artifact (e.g., Document)*

Fred is the CEO of Center Micros.
Design Goals

• **Data Representation.** Support the common representation of *artifacts* and *artifact metadata* (analysis results) independently of *artifact modality* and *domain model*.

• **Data Modeling and Interchange.** Support the platform-independent interchange of *analysis data* in a form that facilitates a formal modeling approach and alignment with existing programming systems and standards.

• **Discovery, Reuse and Composition.** Support the discovery, reuse and composition of independently-developed *analytics*.

• **Service-Level Interoperability.** Support concrete interoperability of independently developed *analytics* based on a common service description and associated SOAP bindings.

Note: “Platform Independent Development” design goal in original spec draft was dropped as we have decided to focus on service-level interoperability only.
Specification Elements

1. Common Analysis Structure (CAS)
2. Type System Model
3. Base Type System
4. Abstract Interfaces
5. Behavioral Metadata
6. Processing Element Metadata
7. WSDL Service Descriptions
Common Analysis Structure (CAS)

- The common data structure **shared by all UIMA analytics**
- Supports interoperability by providing a common foundation for sharing data across analytics
- A CAS Represents the
  - **Artifact**: the content being analyzed AND
  - **Artifact Metadata**: the metadata produced by the analytics (e.g., Annotations)
- The CAS is an Object Graph where
  - Objects are instances of Classes
  - Classes are Types in a type system.
- Two fundamental types of objects in a CAS:
  - **Subject of analysis (Sofa)**, holds the artifact
  - **Annotation**, a type of artifact metadata that points to a region within a Sofa. Annotates or labels the designated region in the artifact. Example of a *stand-off* annotation approach.
Common Analysis Structure (CAS)

Relationship

CeoOf

Arg1:Person

Analysis Results
(i.e., Artifact Metadata)

Organization

Arg2:Org

Named Entity

Person

Parser

NP

VP

PP

Artifact (e.g., Document)

Fred is the CEO of Center Micros
CAS Model

- General Object Graph
- Expressive Representational Power
- Aligned with UML
CAS Model: UML Diagram

Schema
(see Type System Model)

1

TypeSystem
(from cas)

1..*

Class
name : String

0..*

Feature
name : String
lowerBound : int
upperBound : int

0..*

Reference

Attribute

DataType
name : String

+type

+instanceOf

+type

+instanceOf

Instances
(Object Graph)

0..*

CAS

Object

0..*

Slot

0..*

Value

0..*

PrimitiveValue

0..*

ReferenceValue
CAS Data Representation

- An interchange format for the CAS

- Specified Using *XML Metadata Interchange (XMI)*
  - An OMG standard for representing object graphs in XML.

- Motivation for Using XMI
  - Established standard
  - Aligned with object-graph representation of CAS
  - Aligned with UML and with object-oriented programming
  - Supported by tooling such as the Eclipse Modeling Framework (EMF)
```
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
   xmlns:org.example="http://docs.oasis-open.org/uima/org/example.ecore">

<org.example:Document xmi:id="1"
   text="Fred Center is the CEO of Center Micros."
   author="David Ferrucci"/>

<cas:LocalSofaReference xmi:id="2" sofaObject="1"
   sofaFeature="text"/>

<org.example:Person xmi:id="3" sofa="2" begin="0" end="11"/>
<org.example:Organization xmi:id="4" sofa="2" begin="26" end="39"/>
<org.example:CeoOf sofa="2" begin="0" end="31" arg0="3" arg1="4"/>

</xmi:XMI>
```
Specification Elements

1. Common Analysis Structure (CAS)
2. Type System Model
3. Base Type System
4. Abstract Interfaces
5. Behavioral Metadata
6. Processing Element Metadata
7. WSDL Service Descriptions
Type System Model

• Supports data modeling and interchange

• A CAS must conform to a user-defined schema

• A CAS schema a Type System

• Every object in a CAS must be associated with a Type defined by a Type System
Desired Features of Type System Model

- Object-Oriented
- Inheritance
- Optional and Required Features
- Single and Multi-Valued
- Range Constraints on Features
- Aligned with UML standard
- Supported by Tooling
Type System Representation

• Possible choices considered for the type system representation
  – *EMOF*: An OMG Standard well aligned with UML and Object Oriented Programming.
  – *Ecore*: The modeling language used by the Eclipse Modeling Framework (EMF). Provides equivalent modeling semantics to EMOF with minor syntactic differences.

• UIMA TC has chosen to adopt *Ecore* as its type system language, due to the availability of tooling provided by EMF.
Ecore “Kernel”

Excerpt from Budinsky et al., *Eclipse Modeling Framework*
Ecore as CAS Type System Model
Ecore as CAS Type System Model

Use standard Ecore

Schema

TypeSystem (from cas)

1

EClass

EClass

name : String

1

0..*

0..*

0..*

0..*

0..*

EStructuralFeature

name : String
lowerBound : int
upperBound : int

0..*

EReference

0..*

EAttribute

+eType

+eType

+eType

EDataType

name : String

+instanceOf

+instanceOf

+instanceOf

PrimitiveValue

ReferenceValue

Instances

(Object Graph)

CAS

1

Object

1

+referenceTo

+instanceOf

+definingFeature

n..*

Slot

Value
Example Ecore Type System

<ecore:EPackage xmi:version="2.0" ... name="org"
   nsURI="http://org.ecore" nsPrefix="org">
   <eSubpackages name="example" nsURI="http://org/example.ecore"
      nsPrefix="org.example">
      <eClassifiers xsi:type="ecore:EClass" name="NamedEntity">
         <eStructuralFeatures xsi:type="ecore:EAttribute" name="name"
            eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
      </eClassifiers>
      <eClassifiers xsi:type="ecore:EClass" name="Relation">
      </eClassifiers>
      <eClassifiers xsi:type="ecore:EClass" name="Person" eSuperTypes="#//example3/Entity
      #//example/NamedEntity">
         <eStructuralFeatures xsi:type="ecore:EAttribute" name="ssn"
            eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
         <eStructuralFeatures xsi:type="ecore:EAttribute" name="age"
            eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EIntegerObject"/>
      </eClassifiers>
      <eClassifiers xsi:type="ecore:EClass" name="CeoOf" eSuperTypes="#//example/Relation">
         <eStructuralFeatures xsi:type="ecore:EReference" name="arg0" lowerBound="1"
            eType="#//example/Person"/>
         <eStructuralFeatures xsi:type="ecore:EReference" name="arg1" lowerBound="1"
            eType="#//example/Organization"/>
      </eClassifiers>
      <eClassifiers xsi:type="ecore:EClass" name="Document">
         <eStructuralFeatures xsi:type="ecore:EAttribute" name="text"
            eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
      </eClassifiers>
   </eSubpackages>
</ecore:EPackage>
Specification Elements

1. Common Analysis Structure (CAS)
2. Type System Model
3. Base Type System
4. Abstract Interfaces
5. Behavioral Metadata
6. Processing Element Metadata
7. WSDL Service Descriptions
Base Type System

- Standard definition of commonly-used, domain-independent types
- Establishes basic level of interoperability among applications
- Includes
  - Primitive Types (defined by Ecore)
  - Annotation Types (Annotation representation and linkage to Sofas)
  - Views (Specific collections of objects in a CAS)
  - Other Commonly Used Types (e.g., Source Document Information)
- Annotation Types
  - Central to CAS
  - Discussed in detail in following slides.
- For details see Backup Slides.
• The **Annotation** class represents a type of object that is linked to a Subject of Analysis (Sofa).

• The **LocalSofaReference** class allows any field of any object in the CAS to be a Sofa.

• The **RemoteSofaReference** class allows the Sofa data to be located outside the CAS.

---

**Document**

- text = “Fred Center is the CEO of Center Micros…”
- author = “David Ferrucci”

**LocalSofaReference**

- sofaFeature = “text”
- sofaObject

**Annotation**

- Details next slide

**Person**

- sofa

**Organization**

- sofa

---

**SofaReference**

- sofaObject
- sofaFeature
- sofaUri

**Annotation**

- sofa
References to Regions of Sofas

- The Annotation class has subclasses for each artifact modality, which define how the Annotation refers to a region within the Sofa.

- The Standard defines subclasses for common modalities – Text and Temporal (audio or video segments).

- Users may define other subclasses.
Options for Extending Annotation Type System

Choice 1:
Subclasses of Annotation

- **TextAnnotation**
  - `beginChar : int`
  - `endChar : int`

- **TemporalAnnotation**
  - `beginTime : float`
  - `endTime : float`

- **PersonTextAnnotation**
- **OrganizationTextAnnotation**
- **PersonAudioAnnotation**
- **OrganizationAudioAnnotation**

Choice 2:
References to Annotations

- **Person**
- **Organization**

Choice is left up to designer of the application’s type system.

UIMA TC will elaborate on the pros/cons of each, beyond scope of this presentation.
Additional Annotation Metadata

- Annotation class has `metadata` feature for associating additional information that applies to all annotations.

- UIMA standard provides fields for confidence and provenance.

- Users may subclass `AnnotationMetadata` to add more information.
Example of Annotation Model Extension

- IBM’s Knowledge Level Types (KLT)[1]
  - Coreference modeled with reified HasOccurrence link
  - Provenance captured by componentId fields

- Possible redesign using standard
  - Standard definition of Entity and Annotation
  - Provenance captured by extension of the standard Provenance object.

Specification Elements

1. Common Analysis Structure (CAS)
2. Type System Model
3. Base Type System
4. Abstract Interfaces
5. Behavioral Metadata
6. Processing Element Metadata
7. WSDL Service Descriptions
Abstract Interfaces

Goal: Define standard component types and operations that UIMA developers can implement.
Types of UIMA Components

- **ProcessingElement**: Supertype of all components
  - **Analytic**: Performs analysis of CASes
  - **FlowController**: Determines the route CAS takes through multiple analytics.
  - **Analyzer**: Processes a CAS and possibly updates its contents
  - **CasMultiplier**: Processes a CAS and possibly creates new CASes
An Analytic is a component that performs an analysis operation on a CAS. Though it provides no methods (the specializations Analyzer and CasMultiplier do), this is a central concept in UIMA and deserves its own class.
Flow Controller UML Details

FlowController

- addAvailableAnalytics(analyticMetadataMap : AnalyticMetadataMap)
- removeAvailableAnalytics(analyticKeys : Keys)
- setAggregateMetadata(metadata : ProcessingElementMetadata)
- getNextDestinations(cas : CAS) : Step
- continueOnFailure(cas : CAS, failedAnalyticKey : String, failure : UimaException) : Boolean

AnalyticMetadataMap

- AnalyticMetadataMapEntry
  - key : String
  - ProcessingElementMetadata (from peMetadata)

Step

- SimpleStep
  - analyticKey : String
- MultiStep
  - parallel : boolean
- FinalStep

Keys

<<1..n>> key : String
Analyzer Interface Example

Client Application

CAS
TextDocument (Sofa)
Persons
Organizations

process(CAS, ID of TextDocument)

UI MA Analyzer

Returns CAS

CAS
TextDocument (Sofa)
Persons
Organizations
CeoOf
CAS Multiplier Interface Example

Client Application

- inputCas(CAS, ID of Video Stream URL)
- getNext()

CAS
Video Stream (Sofa)

- getCAS
  - Video Segment
  - getNext()
  - getNext()
  - retrieveInputCas()

UI MA CAS Multiplier

- Returns updated CAS
- Returns updated CAS
- Returns next output CAS

Segment Boundaries
Specification Elements

1. Common Analysis Structure (CAS)
2. Type System Model
3. Base Type System
4. Abstract Interfaces
5. Behavioral Metadata
6. Processing Element Metadata
7. WSDL Service Descriptions
Behavioral Metadata

• Formal declarative description of what a UIMA analytic does
  – What types of CASs it can process
  – What elements in a CAS it analyzes
  – What effects it may have on CAS contents as a result of its application.

• Supports
  – **Discovery**: Locate components that provide a particular function.
  – **Composition**: Help determine which components may be combined to produce a desired result
  – **Efficiency**: Efficient sharing of CAS content among cooperating analytics based on knowledge of analytic requirements.
Elements of Behavioral Metadata

- **Analyzes:** Types of objects (Sofas) that the analytic intends to produce annotations over.
- **Required Inputs:** Types of objects that must be present in the CAS for the analytic to operate.
- **Optional Inputs:** Types of objects that the analytic would consult if they were present in the CAS.
- **Creates:** Types of objects that the analytic may create.
- **Modifies:** Types of objects that the analytic may modify.
- **Deletes:** Types of objects that the analytic may delete.
<behavioralMetadata xmlns:org.example="http://docs.oasis-open.org/uima/org/example.ecore">

<analyzes>
  <type name="org.example:TextDocument"/>
</analyzes>

|requiredInputs>
  <type name="org.example:Person"/>
  <type name="org.example:Organization"/>
</requiredInputs>

<creates>
  <type name="org.example:CeoOf"/>
</creates>

</behavioralMetadata>
Using Views in Behavioral Metadata

- Analytics may specify that inputs & outputs are contained within the same View.
- May explicitly specify that the input & output annotations refer to the same Sofa

```xml
<behavioralMetadata xmlns:org.example="http://docs.oasis-open.org/uima/org/example.ecore">
  <requiredView sofaType="org.example:Document">
    <requiredInputs>
      <type name="org.example:Person"/>
      <type name="org.example:Organization"/>
    </requiredInputs>
    <creates>
      <type name="org.example:CeoOf"/>
    </creates>
  </requiredView>
</behavioralMetadata>
```

Where does this slide belong?
How Behavior Metadata Example Satisfies Requirements

- **Discovery:**
  - A component repository can be searched to locate an analytic that produces CeoOf annotations.

- **Composition:**
  - Person and Organization annotations are required inputs, so a user knows to combine a Person annotator and a Relation annotator with the CeoOf annotator to produce a valid composition.

- **Efficiency:**
  - If the CAS contains objects in the CAS that are not declared in the analyzes, required inputs, or optional inputs (e.g., Place annotations), then these do not need to be sent to the analytic.
Formal Semantics for Behavioral Metadata

All Behavioral Metadata elements may be mapped to THREE kinds of expressions in a formal language:

- **precondition:** Predicate that qualifies CASs considers valid input by the analytic

- **postcondition:** Predicate that is declared to be true of any CAS that has been processed by the analytic, assuming that the CAS satisfied the precondition when input to the analytic

- **projectionCondition:** Predicate that evaluates to the set of objects that the Analytic declares it will consider to perform its function (analyzes + required + optional). Anything not in the set would not have to be sent to the analytic.
Formal Semantics for Behavioral Metadata

- The semantics of Behavioral Metadata elements may be formalized using OCL (*Object Constraint Language*)
  - an OMG standard
  - A formal language for expressing pre, post and projection conditions
  - Used to ensure a standard interpretation of UIMA Behavior Metadata Elements

- Implementations are free to use other languages
  - For expressing and/or processing pre, post & projection conditions
  - There is no requirement for these expressions to be evaluated or enforced by a component or framework.
Example Mapping to OCL

\[
\begin{align*}
\text{<requiredInputs>} & \\
\quad & \text{<type name="org.example:Person"/>} \\
\quad & \text{<type name="org.example:Organization"/>} \\
\text{</requiredInputs>}
\end{align*}
\]

is equivalent to the OCL precondition that all valid input CASs must satisfy

\[
\begin{align*}
\exists (p \mid p.oclKindOf(\text{org::example::Person})) \land \\
\exists (o \mid o.oclKindOf(\text{org::example::Organization}))
\end{align*}
\]
Behavioral Metadata UML
Specification Elements

1. Common Analysis Structure (CAS)
2. Type System Model
3. Base Type System
4. Abstract Interfaces
5. Behavioral Metadata
6. Processing Element Metadata
7. WSDL Service Descriptions
Processing Element Metadata

- All UIMA Processing Elements (PEs) must publish *processing element metadata*
- Supports Discovery and Composition
Elements of PE Metadata

Processing Element Metadata

Identification Information
- Symbolic/unique Name
- Descriptive Name
- Description
- Vendor
- Version
- URL

Configuration Parameters
- Name and Description of Parameters
- Default Values

Type System Reference
- Defines types referenced from the behavioral specification.

Behavioral Metadata
- (described in previous section)

Extensions
- Allows the PE metadata to contain additional elements, the contents of which are not defined by the UIMA specification.
## PE Metadata Example

```xml
<pemd:ProcessingElementMetadata xmi:version="2.0"

  <identification
    symbolicName="org.oasis-open.uima.example.CeoRelationAnnotator"
    name="Ceo Relation Annotator"
    description="Detects CeoOf relationships in a text document."
    vendor="OASIS"
    version="1.0.0"/>

  <configurationParameter
    name="PatternFile"
    description="Location of external pattern file that finds CeoOf relations."
    type="ResourceURL">
    <defaultValue>myResources/ceoPatterns.dat</defaultValue>
  </configurationParameter>

  <typeSystemReference
    uri="http://docs.oasis-open.org/uima/types/exampleTypeSystem.ecore"/>

  <behavioralMetadata> ... </behavioralMetadata>

  <extension extenderId="org.apache.uima"> ... </extension>

</pemd:ProcessingElementMetadata>
```
Specification Elements

1. Common Analysis Structure (CAS)
2. Type System Model
3. Base Type System
4. Abstract Interfaces
5. Behavioral Metadata
6. Processing Element Metadata
7. WSDL Service Descriptions
WSDL Service Definitions

Implements the Abstract Interfaces defined in the UIMA specification

- Facilitates standard use of web-services as a CAS transport
- Specifies WSDL document for the UIMA Processing Element Service Interfaces
- Specifies a binding to the SOAP protocol.

This specification element facilitates interoperability by specifying a concrete SOAP interface that compliant frameworks/services must implement.
SOAP Service Example

UI MA Application

SOAP Message

CAS
TextDocument (Sofa)
Persons
Organizations

Web Server
With
SOAP Container

UI MA
CEO Annotator

CAS
TextDocument (Sofa)
Persons
Organizations
CeoOf

SOAP Message
+ SOAP Service Example

<soapenv:Envelope...>
  <soapenv:Body>
    <processCas xmlns="">
      <cas xmi:version="2.0" ... >
        <org.example:Document xmi:id="1">
          text="Fred Center is the CEO of Center Micros."
          author="David Ferrucci"/>
        </cas>
        <cas:SofaReference xmi:id="2" sofaObject="1" sofaFeature="text"/>
        <org.example:Person xmi:id="3" sofaObject="2" begin="0" end="11"/>
        <org.example:Organization xmi:id="4" sofaObject="2" begin="26" end="39"/>
      </processCas>
    </soapenv:Body>
  </soapenv:Envelope>

<soapenv:Envelope...>
  <soapenv:Body>
    <processCas xmlns="">
      <cas xmi:version="2.0" ... >
        <org.example:Document xmi:id="3">
          text="Fred Center is the CEO of Center Micros."
          author="David Ferrucci"/>
        </cas>
        <cas:SofaReference xmi:id="2" sofaObject="1" sofaFeature="text"/>
        <org.example:Person xmi:id="3" sofaObject="2" begin="0" end="11"/>
        <org.example:Organization xmi:id="4" sofaObject="2" begin="26" end="39"/>
      </processCas>
    </soapenv:Body>
  </soapenv:Envelope>
Backup Slides
Definitions

- **Artifact** refers to an application-level unit of information that is subject to analysis by some application. Examples include a text document, a segment of speech or video, a collection of documents and a stream of any of the above. Artifacts are physically encoded in one or more ways. For example, one way to encode a text document might be as a Unicode string.

- **Artifact Modality** refers to mode of communication the artifact represents, for example, text, video or voice.

- **Domain model** generally refers to a conceptualization of a system, often cast in a formal modeling language. In this report we use it to refer to any model which describes the structure of artifact metadata. A domain model provides a formal definition of the types of data elements that may constitute artifact metadata. For example, if some artifact metadata represents the organizations detected in a text document (the artifact) then the type *Organization* and its properties and relationship to other types may be defined in a domain model which the artifact metadata instantiates.

- **Artifact Metadata** refers to structured data elements recorded to describe entire artifacts or parts of artifacts. A piece of artifact metadata might indicate, for example, the part of the document that represents its title or the region of video that contains a human face. Another example of metadata might indicate the topic of a document while yet another may tag or annotate occurrences of person names in a document etc. Artifact metadata is logically distinct form the artifact, in that the artifact is the data being analyzed and the artifact metadata is the result of the analysis – it is data about the artifact.

- **Analysis Data** is used to refer to the logical union of an artifact and its metadata.

- An **Analytic** is a software object of network service that performs an Analysis Operation. They may be composed by some workflow of other *analytics*. 

Back to Design Goals
Base Type System Details
Primitive Types

- As defined by Ecore:
  - EString
  - EBoolean
  - EByte (8 bits)
  - EShort (16 bits)
  - EInt (32 bits)
  - ELong (64 bits)
  - EFloat (32 bits)
  - EDouble (64 bits)

To Review – added to address concern that Base Type System not completely covered.
Base Type System – Views

- **A View** is a collection of Objects.
- **An AnchoredView is a View** that is attached to a Sofa. All annotations that are members of the AnchoredView must refer to that Sofa.

```
Object (from cas)
+member 0..*

View
name : String 0..*

AnchoredView

SofaReference
+sofa

{members->forall(a | aoclTypeOf(uima.Annotation) implies a.sofa = self.sofa)}
```
Base Type System – Source Document Information

- Records information about the original source of the unstructured data contained in the CAS – a very common requirement.
- Still need to agree on features. The diagram shows the features implemented in Apache UIMA.
- Other suggestions: mimeType, fileName, fileTitle, security.