OASIS UIMA Technical Committee Specification Overview

(DRAFT)

October 26, 2007

Status

- 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.

Outline

- Design Goals
- Specification Elements

Design Goals Overview

The UIMA Specification design goals focus on providing a standard specification for text and multi-modal analysis that supports the *data and service levels of interoperability* to facilitate the rapid combination and deployment of analytics in the development of UIM applications.

Design Goals

- **Data Representation.** Support the 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 object oriented modeling and programming.
- **Discovery, Reuse and Composition.** Support the discovery, reuse and composition of independently-developed *analytics* to accelerate UIM applications development.
- **Out-of-the-Box Service-Level Interoperability.** Support the 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.

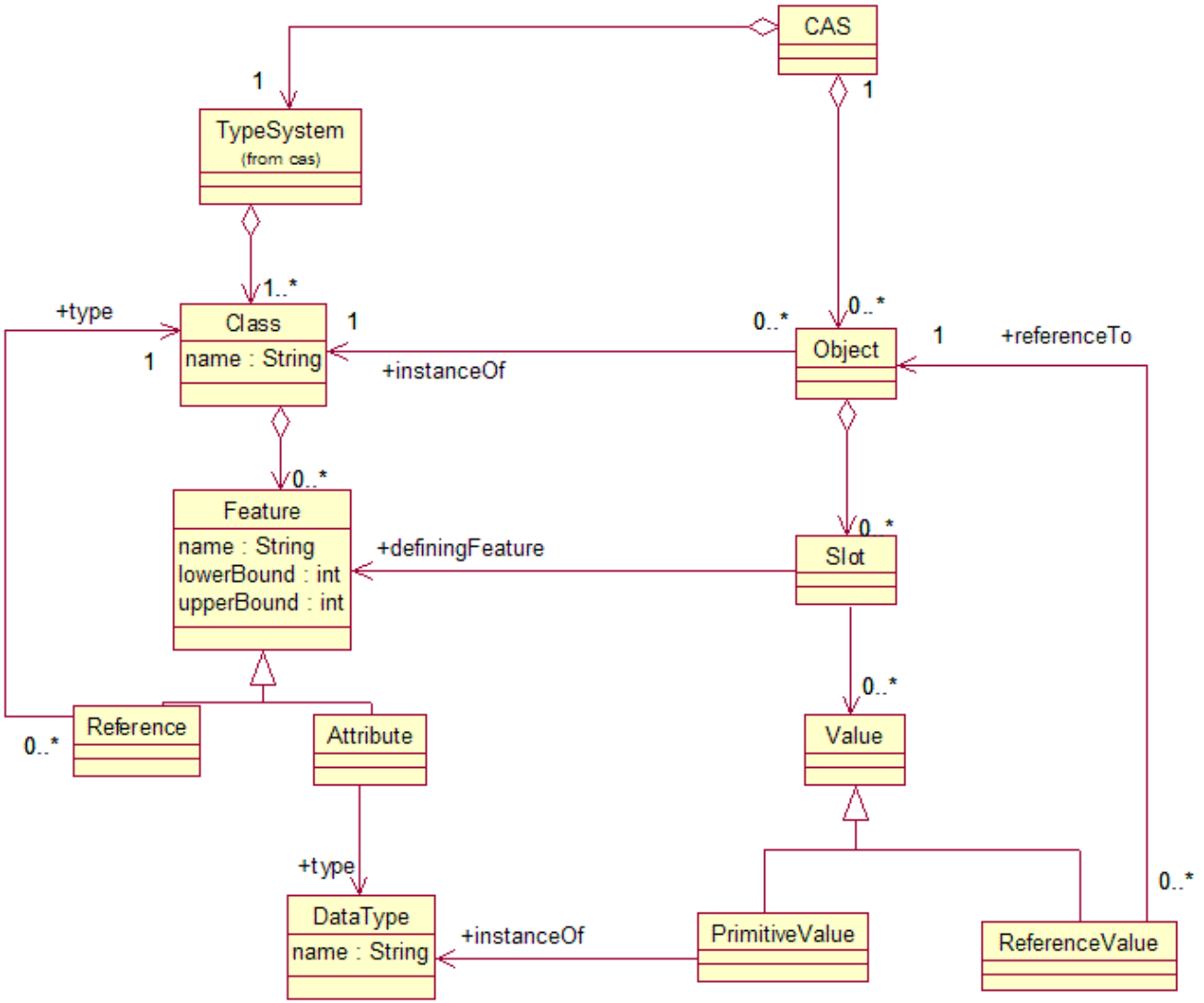
Specification Elements

1. Common Analysis Structure (CAS)
2. Type System Language
3. Type System Base Model
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**
- A CAS Represents the
  - *Artifact*: the unstructured information being analyzed AND
  - *Artifact Metadata*: the metadata produced by the analysis workflow (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 are:
  - *Subject of analysis (Sofa),* holds the artifact data to be analyzed
  - *Annotation,* a type of artifact metadata that points to a region within a Sofa. An annotation annotates or labels the designated region in the artifact.
- Provides a *stand-off* representation of annotations over the artifact.

CAS UML


CAS
1
1
TypeSystem
(from cas)
1..*
+type
Class
name : String
1
1
+instanceOf
0..*
0..*
Object
1
+referenceTo
0..*
Feature
name : String
lowerBound : int
upperBound : int
+definingFeature
0..*
Slot
0..*
Reference
0..*
Attribute
Value
+type
DataType
name : String
+instanceOf
PrimitiveValue
ReferenceValue
0..*

Common Analysis Structure (CAS)

To support data interchange a CAS is represented in XML using the *XML Metadata Interchange (XMI)* format, an OMG standard:

```
<xmi:XMI xmi:version="2.0" xmlns:xmi=http://www.omg.org/XMI
   xmlns:ex="http:///org/example.ecore">
  <ex:Quotation xmi:id="1"
    text="If we begin in certainties, we shall end with doubts; but if we
   begin with doubts and are patient with them, we shall end in
   certainties."
    author="Francis Bacon"/>

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

  <ex:Clause sofa="2" begin="0" end="30"/>
  <ex:Pronoun sofa="2" begin="3" end="5"/>
  <ex:Pronoun sofa="2" begin="29" end="31"/>
</xmi:XMI>
```

Type System Language

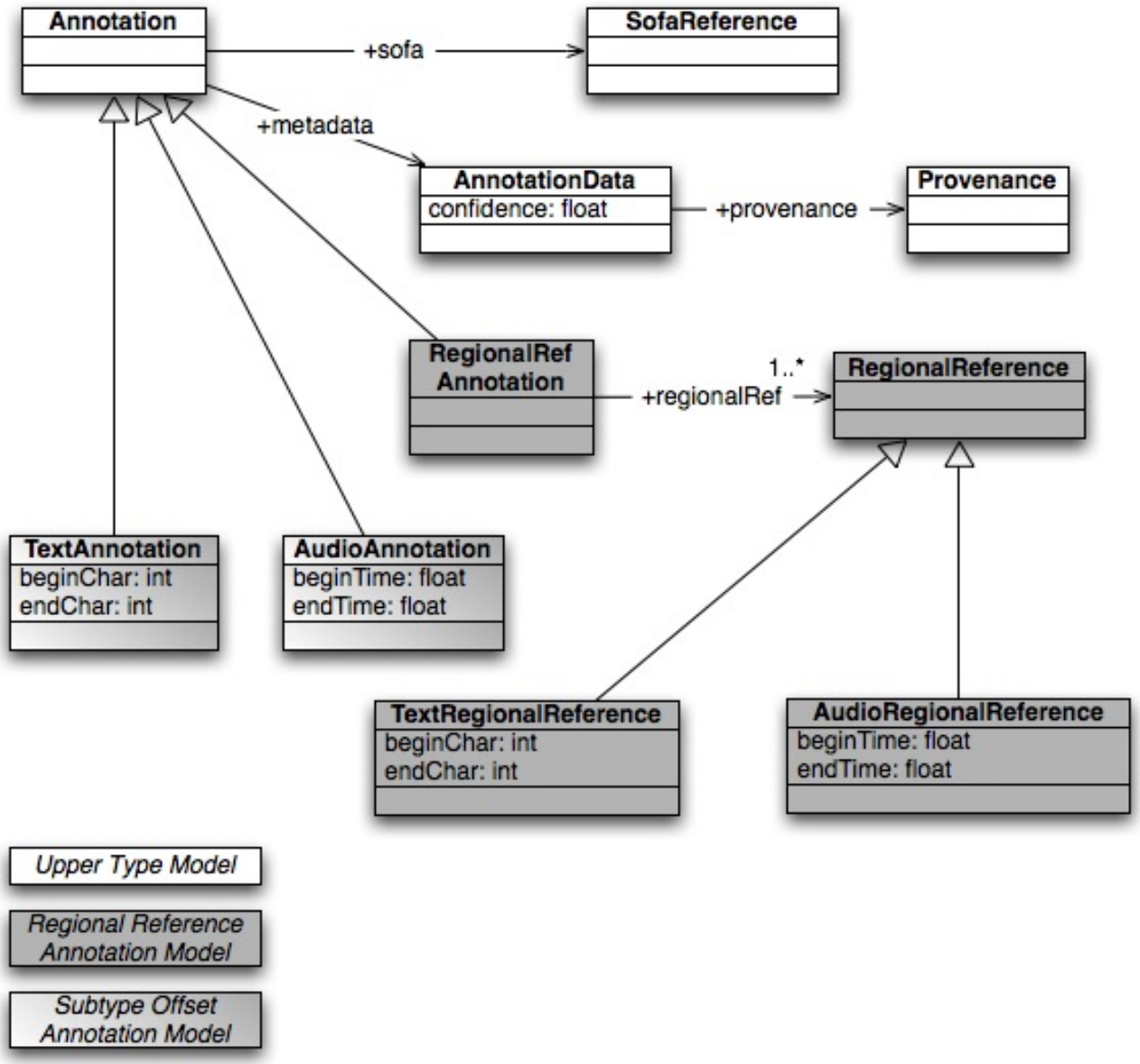
- Every object in a CAS must be associated with a *Type*.
- The UIMA Type-System language therefore is a declarative language for defining object models.
- UIMA Type Systems are represented in *Ecore*, which is based on the OMG standard, UML and is the modeling language used by the Eclipse Modeling Framework (EMF).

Type System Base Model

The Type System Base Model defines the set of predefined types that are assumed to be available to any UIMA-compliant analytic or system.

- Primitive Types
- Annotation Model (how annotations are represented and linked to regions of Sofas)
- Views (Specific collections of objects in a CAS. May be used to define specific interpretations or views of a *Sofa*.)
- Other Commonly Used Types (e.g., Source Document Information)

Annotation Base Model



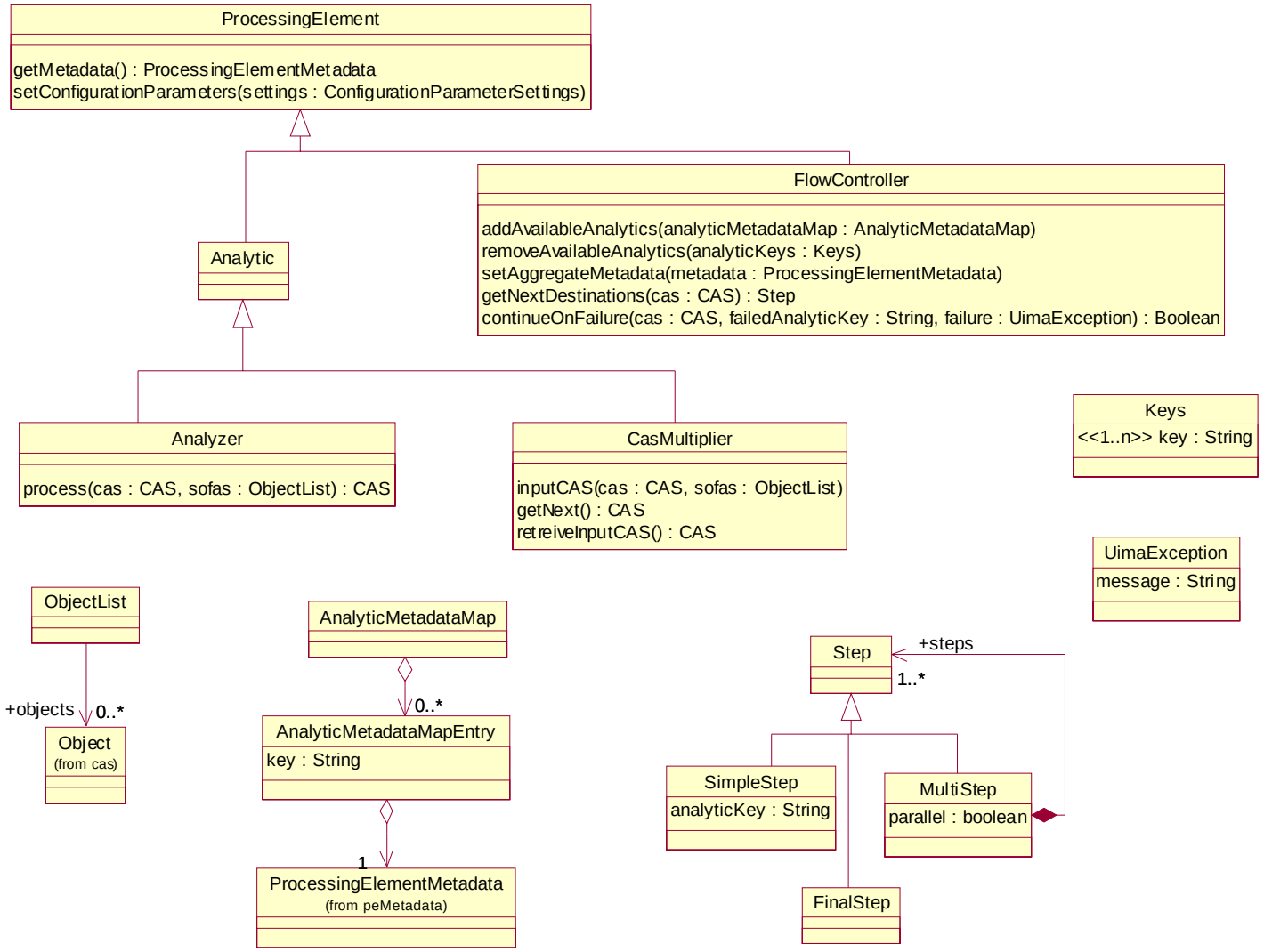
Abstract Interfaces

The goal of the Abstract Interfaces section is to provide a platform-independent model of the types of components that UIMA developers can implement and the operations supported by these components.

Types of Components

- ProcessingElement
  - The supertype of all components
- Analytic: performs analysis of CASes
  - *Analyzer*: Processes a CAS and possibly updates its contents
  - *CasMultiplier*: processes a CAS and possibly creates new CASes
- FlowController
  - Determines how CAS should be routed through multiple analytics

Abstract Interfaces UML



Behavioral Metadata

- Declaratively describes what a UIMA analytic does:
  - what types of CASs it can process
  - what elements in a CAS it analyzes
  - what sorts of 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 the analytics in a combination based on knowledge of analytic requirements.

Elements of Behavioral Metadata

- Supporting Discovery:
  - Analyzes (Sofas that the analytic intends to produce annotations over)
  - Required Inputs
  - Optional Inputs
  - Creates
  - Modifies
  - Deletes
- Supporting Composition:
  - Precondition: Predicate that qualifies CASs that the analytic considers valid input
  - Postcondition: Predicate that is declared to be true of any CAS after having been processed by the analytic, assuming that the CAS satisfied the precondition when it was input to the analytic
- Supporting Efficiency:
  - Projection Condition: Predicate that evaluates to the set of objects that the Analytic declares it will consider to perform its function.

Ways of Expressing Behavioral Metadata

- Type Names
  - Simplest Expressions
- OCL Expressions
  - Formal standard for more complex applications
  - Simple expressions can be captured as OCL
  - Other options possible
- Views
  - A convenient way to specify inputs and outputs that pertain to a particular *Sofa*.

Processing Element Metadata

- All UIMA Processing Elements (PEs) must publish ***processing element metadata***, which describes the analytic to support discovery and composition.
- Includes the Behavioral Metadata plus:
  - **Identification Information.** Identifies the PE. It includes for example a symbolic/unique name, a descriptive name, vendor and version information.
  - **Configuration Parameters**. Declares the names of parameters used by the PE to affect its behavior, as well as the parameters' default values.
  - **Reference to a Type System**. Defines types referenced from the behavioral specification.
  - **Extensions**. Allows the PE metadata to contain additional elements, , the contents of which are not defined by the UIMA specification. This can be used by framework implementations to extend the PE metadata with additional information that may be meaningful only to that framework.

WSDL Service Definitions

- Provides a WSDL document for the UIMA Processing Element Service Interfaces.
- Defines a binding to the SOAP protocol.
- This WSDL definition is an implementation of the Abstract Interfaces previously defined in the UIMA specification.
- This specification element intends to provide true out-of-the-box interoperability by specifying a concrete SOAP interface that compliant frameworks/services *must* implement.

Backup Slides

Dropped Design Goal

- Original Specification Draft contained the design goal:
  - **Platform Independent Development.** Facilitate the compliance of existing applications or the development of new applications on different platforms and in different programming languages.
- This seems out of place now since the specification is only defining services interfaces for UIMA. We do not address programming language bindings at all. APACHE UIMA defines Java Bindings.