OASIS UIMA Technical Committee
Specification Overview

(DRAFT v0.4)

December 14, 2007
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

- Vencorazole 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.

Highlighting Controls

- Free-text Query Hits
- Annotation Data Query Hits
- RxNorm Drugs
- MeSH Disease
- 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 = 0.1
  - Confidence = 0.0
On October 16, 2002, we issued $300 million of 5.875% Senior Notes that mature in October 2012, with interest payable semi-annually on April 15 and October 15. A portion of the proceeds from these Senior Notes was used to repay a $115 million unsecured bank term loan due December 2003, to repay a portion of the Company’s outstanding commercial paper borrowings, and to settle interest rate hedges associated with the issuance and repayment of the related debt securities. On June 3, 2001, we issued $200 million of...
…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)

• 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

Question – should this go near end before details of impact on apache or here in Intro? If we leave out the detail for the Apache UIMA person it raises more questions that it answers and I think not worth having in intro.

To Review – Changed to NOT state alignment as an objective of the standard – this is the wrong intent an message. This is NOT just a standard for Apache UIMA
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
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.
Fred is the CEO of Center Micros
CAS Model

- General Object Graph
- Expressive Representational Power
- Aligned with UML

To Review
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)
CAS XMI Example

To Review - Aligned with our running text example
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.
Excerpt from Budinsky et al., *Eclipse Modeling Framework*
CAS Model: UML Diagram
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"
      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"
    xsi:type="ecore:EAttribute" name="name"
    xsi:type="ecore:EAttribute" name="age"
    xsi:type="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EIntegerObject"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="CeoOf"
    xsi:type="ecore:EAttribute" name="arg0"
    xsi:type="ecore:EAttribute" name="arg1"
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Document">
  </eClassifiers>
  <eSubpackages name="example3"/>
</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
- Use facilitates 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.

To Review – Acknowledged that use of the base type system is not strictly required. Added link to detail slides for Primitive Types, Views etc, in Back up slides.
Annotation / Sofa Reference

- 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.

To Review – Annotation Base Type
System discussion in next few slides has been revamped.

```
Document
text = "Fred Center is the CEO of Center Micros..."
author = "David Ferrucci"
```
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 Audio.

• Users may define other subclasses.

To Review– should we include a base type for video?
Ways to extend the Annotation Type System

Choice 1:
Subclasses of Annotation

Choice 2:
References to Annotations

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

UIMA Standard will explain pros/cons of each, beyond scope of this presentation.

To Review – significant change here. I think introducing the “Entity” type makes things a lot cleaner. Can include other ideas/options but stress not part of standard – just ways standard can be used to extend to application requirements or a richer standard.
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.

To Review – Is this what we agreed to do here?
To Review. Changes from the draft spec are the addition of the Entity type and the occurrence/occurrenceOf relationship with Annotation.
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 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 the route CAS takes through multiple analytics
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.

Separated flow-controller detail. Put note box on “Analytic” to rationalize purpose.
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
    +steps

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 updated CAS

CAS
TextDocument (Sofa)
Persons
Organizations
CeoOf

To review
CAS Multiplier Interface Example

Client Application

CAS Video Stream (Sofa)

inputCas(CAS, ID of Video Stream URL)

getNext()

CAS Video Segment

Returns next output CAS

getNext()

CAS Video Segment

Returns next output CAS

retrievieInputCas()

CAS Video Stream (Sofa)

Returns updated CAS

UI MA CAS Multiplier

To review
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.
Behavioral Metadata Example

<behavioralMetadata xmlns:org.example="http://docs.oasis-open.org/uima/org/example.ecore">

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

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

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

</behavioralMetadata>

To review – made consistent with running example and added explanations.
Behavior Metadata Example

• **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.

To review – shows how Behavioral Metadata supports its objectives.
OCL for Formal Semantics

• Behavioral Metadata has formal semantics
  – Based on OCL (Object Constraint Language), an OMG standard
  – Used to ensure standard interpretation of UIMA Behavior Metadata

• All Behavioral Metadata may be mapped to OCL expressions
  – 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)
Example Mapping to OCL

\[ \langle \text{requiredInputs} \rangle \\
\quad \langle \text{type name="org.example:Person"} / \rangle \\
\quad \langle \text{type name="org.example:Organization"} / \rangle \\
\langle /\text{requiredInputs} \rangle \]

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

\[
\text{exists}(p \mid p.\text{oclKindOf} \text{::} \text{org::example::Person}) \quad \text{and} \quad \\
\text{exists}(o \mid o.\text{oclKindOf} \text{::} \text{org::example::Organization})
\]
Using OCL Expressions Directly

• More expressive than simpler expressions
• Analytic metadata *could* include OCL pre, post and projection conditions directly
• Puts burden on frameworks to support OCL interpretation
  – May be distinct or optional compliance point
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

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

To review
Extracted from PE Metadata UML Diagram. Also added requiredView & optionalView, which were missing.
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*
- Support Discovery and Composition
- Includes
  - **Behavioral Metadata.** (Discussed above)
  - **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.
To review – new example
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

• 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.
To review – added this example. Do we need more WSDL/SOAP details?
Impact on Apache UIMA

November 9, 2007
Impact on Apache UIMA

• Well Aligned
  – CAS Data Representation: Apache UIMA already uses XMI
  – Abstract Interfaces: Very similar to the Apache UIMA interfaces

• Minor Differences
  – Type System Language: Apache UIMA supports Ecore import/export, but still uses its own “native” type system language, and there are a few minor mismatches in semantics. Many constraints expressible in Ecore are not enforced.
  – Processing Element Metadata: Apache UIMA has some things not in the proposed standard:
    • Indexes
    • Configuration Parameter Groups
  – SOAP Interfaces: Would need to be implemented as new Apache UIMA service adapters. Apache UIMA is designed to make this relatively easy to do.

• Major Differences
  – Type System Base Model: significant differences
    • Apache UIMA Annotation Base Model is different from proposed standard
    • Apache UIMA Views are 1-1 with Sofas
  – Behavioral Metadata: Apache UIMA has limited behavioral metadata and it lacks precise semantics (and therefore can’t be automatically converted to the proposed standard format).
Type System Base Model Differences

- **Naming differences**
  - Apache: uima.tcas.Annotation
  - OASIS: org.oasis-open.uima.TextAnnotation

- **Differences in begin, end offsets?**
  - Apache: UTF-16 code units (works well for Java)
  - OASIS: Unicode characters? (better for interoperability across platforms)

- **Differences in what can be a Sofa**
  - Apache: Annotation points to an object of type Sofa, which contains the data
  - OASIS: Annotation points to an object of type SofaReference, which then points to another object in the CAS that holds the actual Sofa data

- **OASIS allows separate Regional Reference object.**

- **Differences in Views**
  - Apache: Every Sofa has exactly one View, and every View must have a Sofa.
  - OASIS: Views are general collections of objects. A View *may* be linked to a Sofa, but this is not required. More than one View may be linked to the same Sofa.
Behavioral Metadata Differences

• Apache UIMA semantics are not well-defined
  – Allows specifying inputs and output but…
  – Not clear whether an input is required or optional
  – Not clear how input and output types relate to input & output Sofas

• Apache UIMA allows specifying multiple sets of capabilities
  – Allows specifying that different outputs may be produced depending on what inputs are received
  – Rarely Used
Service-Level Compliance

• Apache UIMA service adapters
  – Apache UIMA Analysis Engines can be deployed as UIMA-Standard-Compliant services.

• Implementing `getMetadata` operation
  – Type System converted to Ecore using existing converters (also need to convert base type system!)
  – PE Metadata can be serialized to standard-compliant format
    • Some things such as configuration groups would not be supported

  – Issue
    • How Can Apache UIMA Capabilities be published as OASIS UIMA Behavioral Metadata?
Service-Level Compliance

• Implementing *process* operation – need to map OASIS base type system. Not trivial but possible.
  – Convert Type Names
  – Convert character offsets
  – Create a Sofa object for everything that’s pointed to by a SofaReference
  – Separate RegionalReference objects might not be supported (I think the spec may not require them to be)
  – For non-anchored Views could create a “dummy” Sofa
  – **Not Clear:** What to do with an incoming CAS where more than one View points to the same Sofa??
Deeper Compliance

- Over time Apache UIMA could “internalize” more of the UIMA standard representations.
  - Type System could natively use Ecore, and provide additional enforcement of Ecore constraints.
  - Descriptor formats could support UIMA-standard metadata XML.
  - Behavioral Metadata could move to the OASIS standard and get away from the underspecified representation currently supported.
  - The standard Type System Base Model could be supported natively in the CAS.
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*. 
Representational Goals
(Addressing known Apache UIMA representational issues)

- Discovery and Composition
  - Robust, Unambiguous Metadata
  - Describing Analytics’ Input Requirements and Function
- Uniform treatment of data elements
  - Any item may be the subject of an analysis process
- More General “View” Concept
  - Views can contain any specific collection of data elements
- Improved Naming of Types
  - Better reflect their intent
- More Flexible way to connect meta-data to regions of artifact
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.

```
+member 0..*  
+member 0..*  
```

To Review
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.

```
SourceDocumentInformation
uri : String
offsetInSource : Integer
documentSize : Integer
lastSegment : Boolean
```
Out Takes
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.
Other Notes from 10/26 telecon

- Administrative API:
  - Pascal: Administrative API useful. Perhaps just a method on Abstract Interfaces that returns a log?
  - Adam: If there’s no standardization of log file content this may not be very useful.
  - We could try to define some standard kinds of log messages, but this would require more thought/discussions.

- Examples
Design Goal Overview

Provide a standard specification for text and multi-modal analysis that supports *data and service level interoperability* to facilitate the rapid combination and deployment of analytics in the development of UIM applications.

Redundant – Remove or Reword and not make as specific with regard to UIM applications

(REMOVED)
UIMA Framework Adoption

- Gartner Report Quote
- Universities
- Government
- Healthcare
- Business Intelligence
- 10’s of thousands of downloads
- Ported or Wrapped Analytics
  - OpenNLP
  - NetOWL
  - Julie
Better use of Existing Standards

- XMI for object graphs
- UML/Ecore
- WSDL
- OCL

Should delete.
Ways of Expressing Behavioral Metadata

• Type Names
  – Simplest Expressions

• OCL Expressions
  – Formal standard and semantic foundation
  – May be used 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.
Annotation Base Type System – Two Alternatives

- Offsets in the annotation object may result in an explosion of annotation types.
  - For a new concept an annotation type for EACH modality must be added
  - e.g. Organization ➔ OrgTextAnnotation + OrgAudioAnnotation
- Regional References introduce another level of indirection
  - Given a PersonAnnotation object, the modality must be known to retrieve the regional reference and then it’s covered text
  - Requires two objects per annotation INSTANCE instead of one
- For these reasons, the UIMA Annotation Base Model will define type systems for both alternatives, and let users choose.

Need to revisit this whole thing – want one standard here
Clean up text but maybe a pros, cons table its best.
Also reconsider regional reference
Entity Type (Person) <-- Annotation=RegionReferences ➔ Span
Are confidence and provenance part of base model or just to motivate example?

Need to look up or formally revisit if we can not find it.