Integrating UIMA with Alveo, a human communication science virtual laboratory

Dominique Estival
U. of Western Sydney
d.estival@uws.edu.au

Steve Cassidy
Macquarie University
steve.cassidy@mq.edu.au

Karin Verspoor
University of Melbourne
karin.verspoor@unimelb.edu.au

Andrew MacKinlay
RMIT
andrew.mackinlay@rmit.edu.au

Denis Burnham
U. of Western Sydney
d.burnham@uws.edu.au

Abstract

This paper describes two aspects of Alveo, a new virtual laboratory for human communication science (HCS). As a platform for HCS researchers, the integration of the Unstructured Information Management Architecture (UIMA) with Alveo was one of the aims during the development phase and we report on the choices that were made for the implementation. User acceptance testing (UAT) constituted an integral part of the development and evolution of Alveo and we present the distributed testing organisation, the test development process and the evolution of the tests. We conclude with some lessons learned regarding multi-site collaborative work on the development and deployment of HLT research infrastructure.

1 Introduction

The Alveo Virtual Laboratory provides a new platform for collaborative research in human communication science (HCS). [1] Funded by the Australian Government National eResearch Collaboration Tools and Resources (NeCTAR) program, it involves partners from 16 institutions in a range of disciplines: linguistics, natural language processing, speech science, psychology, as well as music and acoustic processing. The goal of the platform is to provide easy access to a variety of databases and a range of analysis tools, in order to foster inter-disciplinary research and facilitate the discovery of new methods for solving old problems or the application of known methods to new datasets (Estival, Cassidy, Sefton, & Burnham, 2013). The platform integrates a number of tools and enables non-technical users to process communication resources (including not only text and speech corpora but also music recordings and videos) using these tools in a straightforward manner. In this paper, we report on the recent integration of the Unstructured Information Management Architecture (UIMA) with Alveo. This integration is bi-directional, in that existing resources and annotations captured over those resources in Alveo can flow to a UIMA process, and new annotations produced by a UIMA process can be consumed and persisted by Alveo. We also introduce the general approach to user acceptance testing (UAT) of Alveo, focussing on the organisation and process acceptance adopted to meet the acceptance criteria required for the project and to ensure user uptake within the research community. Finally, we demonstrate the application of the testing process for acceptance of the UIMA integration.

Section 2 briefly describes Alveo and its components, in particular the tools and corpora already available on the platform and the workflow engine, then Section 3 describes the Alveo-UIMA integration. Section 4 describes the UAT requirement and the organisation of the testing among the Alveo project partners, outlines the actual testing process and gives examples of the tests, among them the UIMA tests, which were developed for the project. Section 5 discusses alternative strategies and we conclude with some lessons learned regarding multi-site collaborative work on the development and deployment of HLT research infrastructure.

2 The Alveo Virtual Laboratory

Alveo provides easy access to a range of databases relevant to human communication science disciplines, including speech, text, audio and video, some of which would previously have been difficult for researchers to access or even know about. The system implements a uniform and secure license management system for the diverse licensing and user agreement conditions required. Browsing, searching and dataset manipulation are also functionalities which are available in a consistent manner across the data collections through the web-based Discovery Interface. The first phase of the project (December 2012 – June 2014) saw the inclusion of the collections shown in Table 1.

1. PARADISEC (Pacific and Regional Archive for Digital Sources in Endangered Cultures): audio, video, text and image resources for Australian and Pacific Island languages (Thieberger, Barwick, Billington, & Vaughan, 2011)
2. AusTalk, audio-visual speech corpus of Australian English (Burnham et al., 2011)
3. The Australian National Corpus (S. Cassidy, Haugh, Peters, & Fallu, 2012) comprising: Australian Corpus of English (ACE); Australian Radio Talkback (ART); AustLit; Braided Channels; Corpus of Oz Early English (COOEE); Griffith Corpus of Spoken English (GCSAusE); International Corpus of English (ICE-AUS); Mitchell & Delbridge corpus; Monash Corpus of Spoken English (Musgrave & Haugh, 2009).
4. AVOZES, a visual speech corpus (Goecke & Millar, 2004)
5. UNSW Pixar Emotional Music Excerpts: Pixar movie theme music expressing different emotions
6. Sydney University Room Impulse Responses: environmental audio samples which, through convolution with speech or music, can create the effect of that speech or music in that acoustic environment
7. Macquarie University Battery of Emotional Prosody: sung sentences with different prosodic patterns
8. Colloquial Jakartan Indonesian corpus: audio and text, recorded in Jakarta in the early 1990's (ANU)
9. The ClueWeb dataset (http://lemurproject.org/clueweb12/).

Table 1: *Alveo* Data Collections

Through the web-based Discovery interface, the user can select items based on the results of faceted search across the collections and can organise selected data in Items Lists. Beyond browsing and searching, Alveo offers the possibility of analysing and processing the data with a range of tools. In the first phase of the project, the tools listed in Table 2 were integrated within Alveo.

1. EOPAS (PARADISEC tool) for text interlinear text and media analysis
2. NLTK (Natural Language Toolkit) for text analytics with linguistic data (Bird, Klein, & Loper, 2009)
3. EMU, for search, speech analysis and interactive labelling of spectrograms and waveforms (Steve Cassidy & Harrington, 2000)
4. AusNC Tools: KWIC, Concordance, Word Count, statistical summary and statistical analysis
5. Johnson-Charniak parser, to generate full parse trees for text sentences (Charniak & Johnson, 2005)
6. ParseEval, to evaluate the syllabic parse of consonant clusters (Shaw & Gafos, 2010)
7. HTK-modifications, a patch to HTK (Hidden Markov Model Toolkit, http://htk.eng.cam.ac.uk/) to enable missing data recognition
8. DeMoLib, for video analysis (http://staff.estem-uc.edu.au/roland/research/demolib-home/)
9. PsySound3, for physical and psycho-acoustical analysis of complex visual and auditory scenes (Cabrera, Ferguson, & Schubert, 2007)
10. ParGram, grammar for Indonesian (Arka, 2012)
11. INDRI, for information retrieval with large data sets (http://www.lemurproject.org/indri/)

Table 2: *Alveo* Tools

Most of these tools require significant expertise to set up and one of the Alveo project goals is to make this easier for non-technical researchers. The *Alveo* Workflow Engine is built around the Galaxy open source workflow management system (Goecks, Nekrutenko, Taylor, & Team, 2010), which was originally designed for use in the life sciences to support researchers in running pipelines of tools to manipulate data. Workflows in Galaxy can be stored, shared and published, and we hope this will also become a way for human communication science researchers to codify and exchange common analyses.

A number of the tools listed in Table 2 have been packaged as Python scripts, for instance NLTK based scripts to carry out part-of-speech tagging, stemming and parsing. Other tools are implemented in R, e.g. EMU/R and ParseEval. An API is provided to mediate access to data, ensuring that permissions are respected, and providing a way to access individual items, and 'mount' datasets for fast access (Steve Cassidy, Estival, Jones, Burnham, & Burghold, 2014). An instance of the Galaxy Workflow engine is run on a virtual machine in the NeCTAR Research Cloud, a secure platform for Australian research, funded by the same government program (https://www.nectar.org.au/research-cloud). Finally, a UIMA interface has been developed to enable the conversion of Alveo items, as well as their associated annotations, into UIMA CAS documents, for analysis in a conventional UIMA pipeline. Conversely annotations from a UIMA pipeline can be associated with a document in Alveo. Figure 1 gives an overview of the architecture.

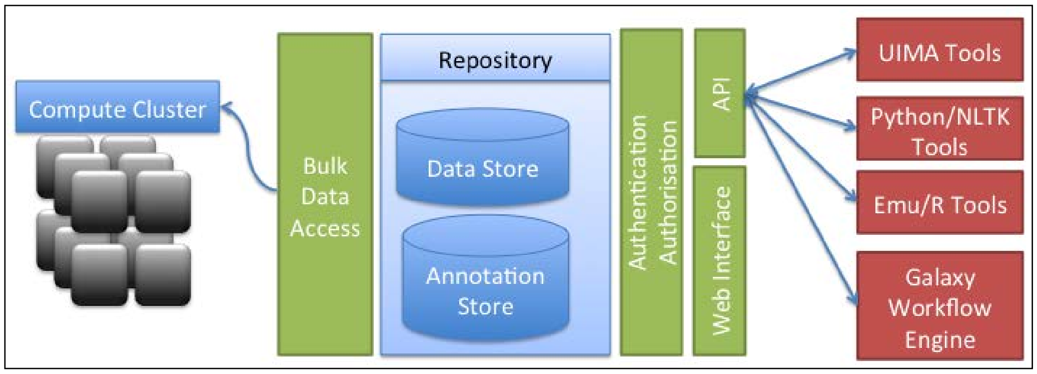

Figure 1: The architecture of the Alveo Virtual Laboratory

2.1 Annotations in Alveo

Annotations in Alveo are stored in a standoff format based on the model described in the ISO Linguistic Annotation Framework (ISO-LAF). Internally, annotations are represented as RDF using the DADA (Steve Cassidy, 2010). Each annotation is identified by a distinct URI and references into the source documents are stored as offsets either using character positions, times or frame counts for audio/video data. Annotations have an associated type attribute that denotes the kind of annotation (speaker turn, part of speech, phonetic segment) and a label that defines a simple string value associated with the annotation. Annotations may also have other properties defined as standard RDF properties and values.

The API exposes a direct interface to the annotation store in RDF via a SPARQL endpoint, but the normal mode of access is via the REST API where each item (document) has a corresponding URI that returns the collection of annotations on that item in a JSON-LD format. JSON-LD allows us to represent the full RDF namespaces of properties and values in a concise format that is easily processed using standard JSON tools. An example annotation delivered in this format is shown in Figure 2. The same JSON-LD format can be used to upload new annotations to be stored in Alveo.

```
{
 "@context": "https://app.alveo.edu.au/schema/json-ld",
 "commonProperties": {
    "alveo:annotates":"https://app…AusE08/document/GCSAusE07.mp3"
 },
 "alveo:annotations": [
 {
  "@id": "http://ns.ausnc.org.au/corpora/gcsause/annotation/535958",
  "type": "http://ns.ausnc.org.au/schemas/annotation/conversation/micropause",
  "@type": "dada:TextAnnotation",
  "end": "34",
  "start": "33"
 },
…}
```

Figure 2: An example of the Alveo JSON-LD annotation format

3 UIMA

3.1 Background

The Unstructured Information Management Architecture (UIMA) is an Apache open source project (http://uima.apache.org) (D. Ferrucci & Lally, 2004) that provides an architecture for developing applications involving analysis of large volumes of unstructured information. This framework allows development of modular pipelines for analysing the sorts of data available in Alveo, including speech, text and video. A number of groups around the world have adopted UIMA to enable easier interoperability and sharing of language technology components. This is true particularly in the biomedical natural language processing community; several groups have made tools available as UIMA modules. Most OpenNLP modules have been wrapped for use within UIMA, and the UIMA community more broadly has a range of language technology tools available in UIMA-compliant modules (David Ferrucci et al., 2010).

Each component in a UIMA application implements interfaces defined by the framework and provides self-describing metadata via XML descriptor files. The framework manages these components and the data flow between them. Since UIMA applications are defined in terms of descriptors that clearly specify both the component modules of the application and the configuration parameter settings for executing the application, they are "re-runnable, re-usable procedures" of the kind that Alveo aims to capture.

Given the objective of Alveo to facilitate access to analysis tools, and the UIMA objective of making such analysis tools interoperable, bringing the two frameworks together made sense. Thus the objective of the integration was to build a bidirectional translation layer between Alveo and any standard UIMA pipeline. In other words, the translation component was required to: 1) read corpus data including associated annotations stored in Alveo into a UIMA pipeline and 2) store annotations produced by a UIMA pipeline in Alveo.

3.2 Overview of the Conversion Layer Architecture

We opted for the most straightforward connection between the two frameworks, i.e. communicating directly with the Alveo REST API. The approach involves allowing annotations and documents to flow from Alveo, be processed externally to Alveo in a specially-configured UIMA pipeline, and then providing a mechanism for new annotations over the documents to be returned to Alveo for storage.
The Alveo REST API provides access to item metadata, documents and annotations using a JSON-LD based interchange format. The API supports most actions that are available via the web interface including meta-data queries and retrieval of documents either individually or in batches.

We built the UIMA-Alveo conversion layer, denoted *Alveo-UIMA*. It allows reading a group of documents from Alveo, and converting the documents along with their associated annotations into UIMA *Common Annotation Structure* (CAS) instances. It also allows annotations produced by a UIMA *Collection Processing Engine* (CPE) pipeline on a set of CASes to be uploaded to an Alveo server. *Alveo-UIMA* is built on top of a native Java wrapper for the Alveo REST API. It is implemented in Java and is distributed as an open-source package.[2] The conversion layer exposes the Alveo data as native Java data structures and is also available as a standalone package,[3] providing, as a side effect, a method to access the Alveo REST API without needing to invoke the UIMA machinery. Similar packages are also available for Python and R in the Alveo repository.

The first component of the UIMA interface, for reading existing items, is implemented as UIMA *Collection Reader*. This takes as parameters an Alveo item list ID, corresponding to a user-created list of documents, and server credentials. It converts the Alveo items from that item list, as well as their associated annotations, into UIMA CAS documents, which can then be used as part of a conventional UIMA pipeline. The UIMA processing pipeline can then take advantage of the annotations downloaded from Alveo (e.g. by using a speaker turn annotation to demarcate a sentence).

[2] https://github.com/Alveo; https://github.com/Alveo/alveo-uima

[3] https://github.com/Alveo/alveo-java-rest-api

The second component is for the opposite direction, i.e. taking annotations from a UIMA pipeline and associating them with the document in Alveo. There is no capability yet to add new documents that derive from outside Alveo, as this is not currently possible using Alveo's REST API. This means that documents for which we are uploading UIMA annotations must have originated in Alveo and have come from the *Collection Reader*, ensuring that each document has appropriate identifying metadata for Alveo.

Annotations need to be converted from Alveo to UIMA and vice versa, since the annotation formats are not identical. Some attributes, such as textual character offsets, are directly convertible, while others require more work. Every document retrieved from the Alveo server has metadata (e.g. data source, recording date and language) and annotations (e.g. POS tags) associated with it. Converting metadata from Alveo to UIMA is straightforward, as the expected metadata fields can be directly mapped to a customised UIMA type. Converting annotations from between the frameworks requires more work, due to the required use of a type system in UIMA. This is discussed further below.

3.3 Conversion from Alveo to UIMA

An annotation in Alveo consists of a beginning and ending offset, which can correspond to a character span for textual data or a time span for audio-visual data, a human-readable plain-text *label* indicating additional attributes of the data, and a *type* (a URI indicating the kind of entity to which the annotation corresponds, e.g. "speaker-turn", "intonation" or "part-of-speech"). Since UIMA also encodes text annotations as character spans, these can be straightforwardly converted into the UIMA CAS (audio-visual data can be similarly treated, but we have focused on text in the current work).

UIMA allows the definition of custom data types with specific fields for storing salient values. We add a generic Alveo annotation type (inheriting from the standard UIMA *Annotation*, which means it still has spans attached). The label of the annotation in Alveo is a non-decomposable string, so this top-level type has a field to store the label as a string.

Handling annotation types requires more care. Types in Alveo are encoded as fully-qualified URIs, while types in UIMA are more strictly defined. In particular, UIMA has a notion of a fully-specified *type system* associated with each pipeline specification, including a full inheritance hierarchy up to a root type and features corresponding to attributes of each type. There are also minor differences between the encoding of the type names (instead of a URI, UIMA uses a 'big-endian' qualified type name similar to a Java package, such as `com.example.nlp.Sentence`).

In addition, UIMA component specification requires specifying in advance the type system to which all annotations represented within UIMA must conform. All these requirements are handled in a type system generation phase triggered when the *Alveo-UIMA* collection reader is created.[4] During this phase, the reader requests an enumeration of all known type URIs from the Alveo server. Since we have no explicit additional information about type inheritance from the URIs, we make as few assumptions as possible by having all types inherit from a generic Alveo annotation parent type. The Alveo URIs are automatically converted to UIMA types names, essentially by reversing the components of the domain name, and replacing the '/' character in the path component of the URI with '.', with some extra handling of non-alphanumeric characters, giving conversions such as the following:

`http://example.com/nlp/sentence` → `com.example.nlp.Sentence`

In addition, the type URI is stored as an attribute of the UIMA annotation, providing an explicit record of the original Alveo type. Because it is far more natural to work with UIMA types than comparing string values when manipulating and filtering annotations in a UIMA pipeline, the automatically generated type system is very beneficial.

We note that there have been proposals to simplify the internal UIMA type system through the use of a generic `Referent` type which refers to an external source of domain semantics (D. Ferrucci, Lally, Verspoor, & Nyberg, 2009) This would be a good strategy to pursue here, so that the UIMA annotations could refer explicitly to the Alveo URIs as an external type system. However, it has been noted previously that this representational choice has consequences for the indexing and reasoning over the semantics of annotations in UIMA analysis engines (Verspoor, Baumgartner Jr, Roeder, &

[4] If the framework users are using UIMA canonically, where the type systems are described by pre-existing XML descriptors, they can explicitly request generation of the appropriate XML. The wrapper was primarily developed using UIMAFit (https://uimafit.apache.org/uimafit), which allows a more dynamic approach.

Hunter, 2009). The current version of UIMA does not provide direct support for this model and hence our strategy is in line with current technical practice. These proposals aim to not replicate the full external type structure within the UIMA type system definition, and this is the practice we follow here, although since Alveo does not currently have a strong notion of type hierarchy this was not a significant consideration.

3.4 Conversion from UIMA to Alveo

The annotation upload component is implemented as a UIMA CAS Consumer, i.e. a component which accepts CASes and performs some action with them. To upload annotations, as noted above, we expect the supplied CAS to derive originally from the Alveo server, with annotations added to that CAS by UIMA processing components. The original metadata from Alveo is used to determine the URL of the original item, and otherwise ignored.

The first step in annotation upload is to retrieve the original source document and remove from the set of annotations to be uploaded those annotations which already appear in the version found on the server. In addition, since processing pipelines may produce a wide variety of annotations which may not all be appropriate or relevant for uploading to Alveo, the annotation type must occur in a preconfigured whitelist. Converting each annotation from UIMA to Alveo is in some ways the inverse of the operation described in the previous section, although there are some intricacies to the process.

The character spans can be directly converted as before; again the type and label require more work. For type conversion, some sensible default behaviours are used. A configurable list of UIMA features are inspected on the UIMA annotations, and the first match found is used as the Alveo annotation type. This list of features naturally includes the default feature for storing the type URI, ensuring that annotations which derive from Alveo originally can be matched back to the original annotation. If no matches are found, a type URI is inferred from the fully-qualified UIMA annotation type name, using the inverse operation to that described above.

Alveo annotations also have labels, as noted in the previous section. As with the annotation types, there is a similar list of UIMA feature names which can be used to populate the label attribute on the Alveo annotation, defaulting to the empty string if no feature name is found. If these strategies do not produce the desired behaviour when uploading, it is possible to customise them by implementing a Java interface. Alternatively, it is also possible to insert a custom UIMA component into the pipeline to convert the added UIMA annotations so that the Alveo conversion works as desired.

4 Alveo User Acceptance Testing

As it was a requirement of the funding agency for the project to provide evidence of User Acceptance Testing (UAT) and acceptance of the results of these tests by the project governing body (the Steering Committee), the project was organised from the start around these requirements, with all the partners bidding for participation in tests of specific components or versions of the system. A testing schedule was developed to accompany the system development, with the aim of gathering feedback from the project partners during development to provide targeted input for improvement. A sub-committee of the Steering Committee was designated to oversee the tests distributed to the testers, examine the reports summarising the results of those tests and recommend acceptance.

Alveo was designed and implemented in partnership with Intersect, a commercial software development company specialised in the support of academic eResearch. This partnership afforded extensive professional support during development, using the Agile process (Beck & al, 2001) as well as thorough regression testing and debugging. In other projects of this type, Intersect provided UAT or managed the UAT process in-house. For the Alveo project, since user testing was the main way in which the academic partners were involved in the project, UAT was organised by the academic partners with technical support from Intersect. The central team at the lead institution oversaw the creation of the tests (see section 4.2), distributed the tests and monitored the results.

4.1 The Alveo UAT process

During development, Alveo was deployed incrementally on separate servers. While the Production Server remained stable between versions, the Staging 1 server was reserved for UAT and only updated when new functionalities were added; the Staging 2 Server was used for development and frequently updated. This rarely caused problems, even with the distributed nature of the testing process.

Each partner site engaged High Degree Researchers (HDRs), generally Masters and Doctoral students but also Post-Doctoral Fellows or project members, who had an interest in a particular domain or tool, or who could provide critical comments about the functionalities. Some Testers were Linguistics students with no computing background, some were Computer Science students with limited linguistic knowledge. At some sites, the Testers were Research Assistants who had worked on the tools or corpora contributed by their institutions, while others were the tool developers themselves. This variety of backgrounds and skills ensured coverage of the main domains and functionalities expected of the Alveo Virtual Lab. Some sites had undertaken to conduct large amounts of testing throughout the development, while other partners only chose to perform limited or more targeted testing, with commitments varying from 10 to 200 hours. Over 30 Testers participated at various times during of the project and a total of more than 300 hours has been spent on testing during Phase I.

4.2 Evolution of the tests

For each version of the system during development (Prototype, Version 1, 2, and 3) a series of tests were developed and posted on a Google Form. To record the results, the Testers filled out a Google form which was monitored by the central team. The first tests developed were very directive, giving very specific instructions as to what actions the user was asked to perform and what results were expected for each action, as shown in Figure 3, one of the tests for Version 1.

Test 2 - Browsing COOEE

1. Login to the main website.
2. In the list of facets on the left, click Corpus, this should show a list of corpus names.
3. From the list of corpus names click cooee, the page should update to show 1354 results, listing the first 10 matching items from the COOEE corpus.
4. In the list of facets on the left click Created, this should show a list of decades.
5. From the list of decades click 1870-1879, the page should update to show 61 results which are COOEE items from the 1870s.
6. From the list of matching items, click on the first item, the page should update to show the details of this item. Verify that the Created date is within the 1870s and that the isPartOf field shows cooee.
7. Scroll down to the bottom of the page where you should see links to the documents in this item. Click on the document marked Text(Original), you should see the text of the document including some markup at the start and end of the file.
8. Use the Back button in your browser to return to the item display page, click on the document marked Text(Plain), you should see the text of the document with no markup.
9. Use the Back button in your browser to return to the item display page.
10. When you are finished, click on the HCSvLab logo on the top left of the page to return to the home page and reset your search.

Figure 3: Test for Alveo Version 1

Gradually the tests became more open-ended, giving less guidance and gathering more informative feedback. The latest round of testing asked Testers to log in and to carry out a small research task, as shown in Figure 4, the instructions for the open form testing of Version 3.

Based on your own research interests and based on what you've seen of the HCS vLab platform, please try to make use of the virtual lab to carry out a small research task. Use the form below to tell us about what you tried to do: the collections and tools that you used, a description of your task, the outcomes and any problems that you faced.

Figure 4: Open form test for Alveo Version 3

Some of the early tests, such as the one shown in Figure 3, have become tutorials provided on the Alveo web page and are now available as help from within the Virtual Lab.

4.3 UIMA Testing

In order to test the Alveo-UIMA implementation and provide an example of how it can be used, we created a tutorial application[5] available from the Alveo github repository. This tutorial shows an example of instantiating a UIMA CPE pipeline which reads documents from an Alveo item list, augments it with part-of-speech annotations and uploads them to the Alveo server. A UIMA pipeline consists of a collection reader, and one or more CAS annotators. The UIMA tutorial pipeline includes the standard *Collection Reader* from Alveo-UIMA, a basic POS-tagging CAS annotator from DKPro-Core,[6] and the annotation uploading CAS annotator from Alveo-UIMA. An advanced version also demonstrates implementing an interface which remaps the POS tag types from those automatically-derived from DKPro.outputs.

5 Discussion

5.1 Related Work

There are several frameworks that have been developed to enable development and evaluation of text processing workflows, and UIMA has been used as the backbone for a few such frameworks due to its support for processing module interoperability. The Argo web service (Rak, Rowley, & Ananiadou, 2012; Rak, Rowley, Carter, & Ananiadou, 2013) is a recent web application that enables development of UIMA-based text processing workflows through an on-line graphical interface. In contrast to Alveo, documents are uploaded to the system within an individual user space, and resulting annotations are not persisted outside of the UIMA data structures; although they can be serialised and stored for subsequent re-use in processing pipelines, or exported as RDF, they are not directly accessible within the framework itself. The repository contains a wide range of NLP components, e.g., modules to perform sentence splitting, POS tagging, parsing, and a number of information extraction tasks targeted to biomedical text.

The U-Compare system (Kano et al., 2009; Kano, Dorado, McCrohon, Ananiadou, & Tsujii, 2010) also supports evaluation and performance comparison of UIMA-based automated annotation tools. It was designed with UIMA in mind from the ground up, enabling UIMA workflow creation and execution through a GUI. Therefore it assumes that all analysis of collections is performed with a set of UIMA components, and indeed provides a substantial number of such components in their repository, although other components can be added. The system is launched locally via Java Web Start; given recent changes to how browsers interact with Java, this no longer works reliably and off-line use (after downloading and installing) is likely necessary, although interaction with web service-based processing components is possible (Kontonasios, Korkontzelos, Kolluru, & Ananiadou, 2011).

A competing framework based on the GATE architecture (Cunningham, Maynard, Bontcheva, & Tablan, 2002) is the cloud-based AnnoMarket platform. This framework provides access to natural language processing (NLP) components, and a limited number of existing resources (one at the time of writing[7], with the facility to upload user-specific data) on a fee-for-service basis (passing along costs of using the Amazon cloud services). Results of NLP studies of this data can be downloaded, or indexed and made available for search. There are a wide array of annotation services and pre-configured pipelines available within the AnnoMarket that can be applied to a user's document collection, either directly through the on-line application or via a web service API.

5.2 Alternative strategies for UIMA integration with Alveo

There were several possible places where the UIMA-Alveo translation layer could have been inserted, and indeed several possible architectures were considered for integrating UIMA with Alveo.

Since Alveo was already working with the workflow engine Galaxy, one option was to create a compatibility layer to bridge UIMA with Galaxy, for instance to enable a pre-configured UIMA pipeline to be instantiated via a Galaxy wrapper. The technical details for accomplishing this were not immediately obvious, and it was decided that this approach would add substantial complexity to the conversion of annotations in the conversion layer.

[5] http://github.com/Alveo/alveo-uima-tutorial
[6] https://code.google.com/p/dkpro-core-asl/
[7] https://annomarket.com/dataSources, accessed 29 May 2014

Another option that was considered was to allow for dynamic construction of UIMA workflows from UIMA components directly through the Alveo web interface. UIMA is a workflow engine analogous to Galaxy, in that it enables dynamic configuration of pipelines from the available set of UIMA components set up in a given environment. In principle, therefore, it would be possible to enable specification, instantiation, and execution of UIMA pipelines from a set of UIMA components made available via Alveo. However, this would have required a substantial development effort specifically targeted towards hosting UIMA components and manipulating UIMA pipelines; it was decided that a more general approach to integrating a broader range of tools was more appropriate for Alveo. Given the recent availability of the Argo web application, an Alveo/Argo integration could be considered that would enable users to create UIMA workflows with Argo but execute them from Alveo, and on documents or corpora stored in Alveo (Rak et al., 2012; Rak et al., 2013). The current web service-based architecture of the *Alveo-UIMA* integration lends itself well to this possibility. This could be explored in future work.

The current implementation assumes that an Alveo user will have the knowledge to create and run UIMA pipelines externally to Alveo. A complementary strategy, possible now that the conversion layer is in place, would be to make complete, pre-configured UIMA pipelines available as tools that can be applied to Alveo corpora/data. A number of such services, e.g. services aimed at annotation of text with one of a set of biomedically-relevant entity types (diseases, genes, chemicals) have been built (MacKinlay & Verspoor, 2013). Each such service is run as a separate UIMA instance that is accessed via a web service. Text is passed in via the REST interface, handed over to the UIMA instance, processed, and annotations are returned. This basic model could be replicated for a number of UIMA pipelines that do standard text-related processing (e.g. split sentences, perform part of speech tagging and parsing, etc.) such that text extracted from Alveo could be processed by the UIMA-based service and annotations returned. This approach has been criticised for its inability to be extended or adapted (Tablan, Bontcheva, Roberts, Cunningham, & Dimitrov, 2013) although it is suitable where pre-packaged pipelines can be applied to accomplish tasks of broad interest.

6 Conclusions

The development of Alveo presented a number of challenges, some technical, such as the integration of UIMA with the platform, and others more logistic, such as the distributed nature of testing during development. In this paper, we described the solution and the choices we made for the implementation of UIMA pipelines, given the constraints regarding the organisation of items, documents and their associated annotations in Alveo. One of the conditions of success of such a project is that the platform be used by researchers for their own projects and on their own data. The organisation of the User Acceptance Testing, requiring partners to contribute during the development, and providing exposure to the tools and the datasets to a large group of diverse researchers is expected to lead to a much wider uptake of Alveo as a platform for HCS research in Australia. We plan to open it to users outside the original project partners during Phase II (2014-2016). We will also continue to explore further interactions with complementary frameworks, such that the data and annotation storage available in Alveo can be enhanced via processing and tools from external services to supplement the functionality that is currently directly integrated.

Acknowledgements

We gratefully acknowledge funding from the Australian Government National eResearch Collaboration Tools and Resources (NeCTAR) and thank all our collaborating partners in the Alveo Virtual Laboratory project: University of Western Sydney, Macquarie University, RMIT, University of Melbourne, Australian National University, University of Western Australia, University of Sydney, University of New England, University of Canberra, Flinders University, University of New South Wales, University of La Trobe, University of Tasmania, ASSTA, AusNC Inc., NICTA, and Intersect.

References

Arka, I. W. (2012). *Developing a Deep Grammar of Indonesian within the ParGram Framework: Theoretical and Implementational Challenges* Paper presented at the 26th Pacific Asia Conference on Language,Information and Computation.

Beck, K., et al. (2001). Manifesto for Agile Software Development. http://agilemanifesto.org/

Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python - Analyzing Text with the Natural Language Toolkit*: O'Reilly Media.

Burnham, D., Estival, D., Fazio, S., Cox, F., Dale, R., Viethen, J., . . . Wagner, M. (2011). *Building an audio-visual corpus of Australian English: large corpus collection with an economical portable and replicable Black Box*. Paper presented at the Interspeech 2011, Florence, Italy.

Cabrera, D., Ferguson, S., & Schubert, E. (2007). *'Psysound3': Software for Acoustical and Psychoacoustical Analysis of Sound Recordings*. Paper presented at the International Community on Auditory Display.

Cassidy, S. (2010). *An RDF Realisation of LAF in the DADA Annotation Server*. Paper presented at the ISA-5, Hong Kong.

Cassidy, S., Estival, D., Jones, T., Burnham, D., & Burghold, J. (2014). *The Alveo Virtual Laboratory: A Web Based Repository API*. Paper presented at the 9th Language Resources and Evaluation Conference (LREC 2014), Reykjavik, Iceland.

Cassidy, S., & Harrington, J. (2000). Multi-level Annotation in the Emu Speech Database Management System. *Speech Communication, 33*, 61–77.

Cassidy, S., Haugh, M., Peters, P., & Fallu, M. (2012). *The Australian National Corpus : national infrastructure for language resources*. Paper presented at the LREC.

Charniak, E., & Johnson, M. (2005). *Coarse-to-fine n-best parsing and MaxEnt discriminative reranking*. Paper presented at the 43rd Annual Meeting on Association for Computational Linguistics.

Cunningham, H., Maynard, D., Bontcheva, K., & Tablan, V. (2002). *GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications*. Paper presented at the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02), Philadelphia, USA.

Estival, D., Cassidy, S., Sefton, P., & Burnham, D. (2013). *The Human Communication Science Virtual Lab*. Paper presented at the 7th eResearch Australasia Conference, Brisbane, Australia.

Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A. A., . . . Welty, C. (2010). Building Watson: An Overview of the DeepQA Project. *AI Magazine, 31*(3), 59-79. doi: http://dx.doi.org/10.1609/aimag.v31i3.2303

Ferrucci, D., & Lally, A. (2004). UIMA: an architectural approach to unstructured information processing in the corporate research environme. *Natural Language Engineering, 10*(3-4), 327-348.

Ferrucci, D., Lally, A., Verspoor, K., & Nyberg, A. (2009). Unstructured Information Management Architecture (UIMA) Version 1.0 *Oasis Standard*.

Goecke, R., & Millar, J. B. (2004). *The Audio-Video Australian English Speech Data Corpus AVOZES*. Paper presented at the 8th International Conference on Spoken Language Processing (INTERSPEECH 2004 - ICSLP), Jeju, Korea.

Goecks, J., Nekrutenko, A., Taylor, J., & Team, T. G. (2010). Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology, 11*(8), R86.

Kano, Y., Baumgartner, W. A., McCrohon, L., Ananiadou, S., Cohen, K. B., Hunter, L., & Tsujii, J. I. (2009). U-Compare: share and compare text mining tools with UIMA. *Bioinformatics, 25*(15), 1997-1998.

Kano, Y., Dorado, R., McCrohon, L., Ananiadou, S., & Tsujii, J. (2010). *U-Compare: An Integrated Language Resource Evaluation Platform Including a Comprehensive UIMA Resource Library*. Paper presented at the LREC. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.180.8878&rep=rep1&type=pdf

Kontonasios, G., Korkontzelos, I., Kolluru, B., & Ananiadou, S. (2011). *Adding text mining workflows as web services to the BioCatalogue*. Paper presented at the Proceedings of the 4th International Workshop on Semantic Web Applications and Tools for the Life Sciences (SWAT4LS '11).

MacKinlay, A., & Verspoor, K. (2013). *A Web Service Annotation Framework for CTD Using the UIMA Concept Mapper*. Paper presented at the Fourth BioCreative Challenge Evaluation Workshop. http://www.biocreative.org/media/store/files/2013/bc4_v1_14.pdf

Musgrave, S., & Haugh, M. (2009). *The AusNC Project: Plans, Progress and Implications for Language Technology*. Paper presented at the ALTA 2009, Sydney.

Rak, R., Rowley, A., & Ananiadou, S. (2012). *Collaborative Development and Evaluation of Text-processing Workflows in a UIMA-supported Web-based Workbench*. Paper presented at the LREC. http://www.lrec-conf.org/proceedings/lrec2012/pdf/960_Paper.pdf

Rak, R., Rowley, A., Carter, J., & Ananiadou, S. (2013). *Development and Analysis of NLP Pipelines in Argo*. Paper presented at the ACL. http://aclweb.org/anthology//P/P13/P13-4020.pdf

Shaw, J. A., & Gafos, A. I. (2010). *Quantitative evaluation of competing syllable parses*. Paper presented at the 11th Meeting of the Association for Computational Linguistics. Special Interest Group on Computational Morphology and Phonology, Uppsala, Sweden.

Tablan, V., Bontcheva, K., Roberts, I., Cunningham, H., & Dimitrov, M. (2013). *AnnoMarket: An Open Cloud Platform for NLP*. Paper presented at the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013), Sofia, Bulgaria.

Thieberger, N., Barwick, L., Billington, R., & Vaughan, J. (Eds.). (2011). *Sustainable data from digital research: Humanities perspectives on digital scholarship. A PARDISEC Conference*: Custom Book Centre. http://ses.library.usyd.edu.au/handle/2123/7890.

Verspoor, K., Baumgartner Jr, W., Roeder, C., & Hunter, L. (2009). Abstracting the types away from a UIMA type system *From Form to Meaning: Processing Texts Automatically* (pp. 249-256).