revista.tradumàtica
tecnologies de la traducció

84-100

Towards an Open Source Localisation Orchestration Framework

Asanka Wasala, Ian O'Keeffe and Reinhard Schäler
Centre for Next Generation Localisation / Localisation Research Centre / CSIS Dept., University of Limerick, Limerick, Ireland.

Asanka.Wasala@ul.ie
Ian.O'keeffe@ul.ie
Reinhard.Shaler@ul.ie

ABSTRACT

This paper describes the design and implementation of a novel environment for the inter-connectivity of distributed localisation components - both open source and proprietary - in response to the scarcity of relevant research in this area, as we view interoperability as being the key to the seamless integration of different entities. The proposed solution promotes interoperability through the adoption of a Service Oriented Architecture (SOA) framework based on established localisation standards. We describe a generic use scenario and the architecture of the environment that allows us to study interoperability issues in localisation processes. This environment was successfully demonstrated at the CNGL Public Showcase in Microsoft, Ireland, November 2010.

Keywords: interoperability, Localisation, SOA, XLIFF, Open Standards

RESUM *(Cap a una plataforma d'orquestració de processos per la localització de codi obert)*

Aquest article descriu el disseny i la implementació d'un nou entorn per a la interconnectivitat de components de localització distribuïts, tant de codi obert com a propietari, donant així resposta a l'escassetat d'estudis d'investigació rellevants en aquesta àrea, ja que considerem que la interoperabilitat és clau per aconseguir la integració completa de diferents entitats. La solució que proposem permet la interoperabilitat mitjançant un esquema d'arquitectura orientada a serveis (SOA) que es basa en els estàndards de localització habituals. Descriurem un escenari d'ús genèric i l'arquitectura de l'entorn que ens permetrà estudiar qüestions d'interoperabilitat en els processos de localització. Aquest entorn va ser presentat amb èxit en el CNGL Public Showcase que es va celebrar a la seu de Microsoft, Irlanda, el novembre de 2010.

Paraules clau: interopertibilitat, localització, SOA, XLIFF, estàndards oberts

RESUMEN *(Hacia una plataforma de orquestación de procesos para la localización de código abierto)*

Este artículo describe el diseño y la implementación de un nuevo entorno para la interconectividad de componentes de localización distribuidos, tanto de código abierto como propietario, con el propósito de paliar la escasez de estudios de investigación relevantes en esta área. A nuestro entender, la interoperabilidad es clave para conseguir la integración completa de diferentes entidades. La solución que proponemos permite la interoperabilidad mediante un esquema de arquitectura orientada a servicios (SOA) que se basa en los estándares de localización habituales. Describiremos un escenario de uso genérico y la arquitectura del entorno que nos permitirá estudiar cuestiones de interoperabilidad en los procesos de localización. Este entorno fue presentado con éxito en el CNGL Public Showcase que se celebró en la sede de Microsoft en Irlanda en noviembre de 2010.

Palabras clave: interopertibilidad, localización, SOA, XLIFF, estándares abiertos

1. Introduction

The term localisation has been defined as the "linguistic and cultural adaptation of digital content to the requirements and locale of a foreign market, and the provision of services and technologies for the management of multilingualism across the digital global information flow" (Schäler, 2009:57-67). Localisation is a complex process involving many tasks, many languages, *locale* specific issues, and time-frame issues such as frequent software updates, short software development life cycles and the simultaneous shipment of source and target language versions (simship). A broad spectrum of software is required to handle the process, and a large number of file formats are encountered. These file formats may be either open standard or proprietary. Localisation processes involve different types of organisations and different professions (e.g. translators, reviewers, and linguists). Localisation has to constantly react to new challenges such as those arising in the context of mobile device content or integration with content management systems. In this extremely complex process, the ultimate goal is to maximise the quality (translations, user interfaces etc.) and quantity (number of locales, simships etc.) while minimising the time and the overall cost.

Interoperability is the key to the seamless integration of different technologies and components across the localisation process and has been defined in a number of different ways in the literature. The most frequently used definition for the term "interoperability" is by the IEEE:

"Interoperability is the ability of two or more systems or components to exchange information and to use the information that has been exchanged." (IEEE, 1991).

However, interoperability, while presenting one of the most challenging problems in localisation, has not been paid much attention in the literature. We aim to address this deficit by presenting a novel approach to interoperability across localisation tools through the adoption of a Service Oriented Architecture (SOA) framework based on established localisation standards. We describe a generic use scenario and the architecture of the approach offering an environment for the study of interoperability issues in localisation process management. To our knowledge, this is the first demonstrator prototype based on SOA and open localisation standards developed as a test bed in order to explore interoperability issues in localisation.

2. Background

Nowadays, software applications are increasingly moving towards a distributed model. Standards are vital for the interoperability of these distributed software applications. However, one of the major problems preventing successful interoperability between and integration of distributed applications and processes is the lack of (standardised) interfaces between them.

In order to address these issues, workflow interoperability standards have been proposed (Hayes et al., 2000) to promote greater efficiency and to reduce cost. The Wf-XML message set defined by the Workflow Management Coalition (WfMC) and The Simple Workflow Access Protocol (SWAP) are examples of such internet-scale workflow standards (Hayes et al., 2000). Most of these standards only define the data and metadata structure while standards such as Hyper-Text Transfer Protocol (HTTP), Common Object Request Broker Architecture (CORBA), and the Internet Inter-ORB Protocol (IIOP) focus on the transportation of data structures (Hayes et al, 2000).

From a purely functional standpoint, we also have the Web Service Description Language (WSDL), the most recent version being WSDL 2.0 (W3C, 2007). WSDL is an XML-based language that defines services as a collection of network endpoints or ports. It is regarded as being a simple interface definition language (Bichler and Lin, 2006:99-101) which does not specify message sequence or its constraints on parameters (Hallé et al., 2010:59-66). However, it possesses limited descriptive ability and covers only the functional requirements in a machine-readable format. Where this becomes an issue is in defining a non-static workflow, as the interface does not provide enough information to allow a broker to make a value judgement in terms of other qualities that are of considerable interest in the localisation process, such as quality, quantity, and so on. Therefore, WSDL does not provide sufficient coverage to support our requirements for interoperability.

There are some notable examples of localisation and translation-centric web services, such as those currently offered by Google, Bing and Yahoo!. However, even here we run into interoperability issues as the interfaces provided do not follow any specific standard, and connecting to these services is still very much a manual process. Some localisation Translation Management Systems (TMS) purport to provide such flexibility, but they tend to be monolithic in their approach, using pre-defined workflows, and requiring dedicated developers to incorporate services from other vendors via bespoke APIs. What is needed is a unified approach for integrating components, so that any service can be called in any order in an automated manner.

2.1 The XLIFF Standard

The XML Localization Interchange File Format (XLIFF) is an open standard for exchanging localisation data and metadata. It has been developed to address various issues related to the exchange of localisation data.

The XLIFF standard was first developed in 2001 by a technical committee formed by representatives of a group of companies, including Oracle, Novell, IBM/Lotus, Sun, Alchemy Software, Berlitz, Moravia-IT, and ENLASO Corporation (formerly the RWS Group). In 2002, the XLIFF specification was formally published by the Organization for the Advancement of Structured Information Standards (OASIS) (XLIFF Technical Committee, 2008).

The purpose of XLIFF as described by OASIS is to "store localizable data and carry it from one step of the localization process to the other, while allowing interoperability between tools" (XLIFF Technical Committee, 2008). By using this standard, localisation data can be exchanged between different companies, organizations, individuals or tools. Various file formats such as plain text, MS Word, DocBook, HTML, XML etc. can be transformed into

XLIFF, enabling translators to isolate the text to be translated from the layout and formatting of the original file format.

The XLIFF standard aims to (Corrigan & Foster, 2003):

- Separate translatable text from layout and formatting data;
- Enable multiple tools to work on source strings;
- Store metadata that is helpful in the translation/localisation process.

The XLIFF standard is becoming the *de facto* standard for exchanging localisation data. It is accepted by almost all localisation service providers and is supported by the majority of localisation tools and CAT tools - you could read more about the XLIFF compliance in CAT tools in the paper by Morado and Wolff (2011) in this same volume. It is being continuously developed by the OASIS XLIFF Technical Committee (2010).

2.2 Localisation Standards and Interoperability Issues

Although the adoption of localisation standards would very likely provide many benefits, software publishers often refrain from the full implementation of the standard or do not carry out rigorous standard conformance testing. There is still a perceived lack of evidence for improved outcomes and an associated fear of high costs of standard implementation and maintenance. One of the biggest problems with regards to tools and technologies today is the pair-wise product drift (Kindrick et al., 1996:61-68), i.e. the need for the output of one tool to be transformed in order to compensate for the other tool's non-conforming behaviour. This trait is present within the localisation software industry.

Most current CAT tools maintain their own proprietary data formats within the boundary of the application. This makes sharing of data between tools very difficult, as conversion between formats often leads to data loss.

XLIFF, as mentioned above, intends to provide a solution to these problems, but true interoperability can only be achieved once the XLIFF standard is implemented in full by the majority of localisation tools providers. Currently, XLIFF compliance seems to be regarded as an addition to the function list of many localisation applications, and many CAT tools seem to pay mere lip service to the XLIFF specification (Anastasiou and Morado, 2010:255-276; Bly, 2010), outputting just a minor subset of the data contained in their proprietary formats as XLIFF to ensure conformance.

3. Experimental Setup

With advancements in technology, the localisation process of the future can be driven by a successful integration of distributed heterogeneous software components. In this scenario, the components are dynamically integrated and orchestrated depending on the available resources to provide the best possible solution for a given localisation project. To model this ideal scenario, we implemented a series of prototypes. As the initial step, an experimental setup has been designed containing multiple interacting components. Firstly, a user creates a localisation project by submitting a source file and supplying some parameters through a user interface component. Next, the data captured by this component is analysed by the Workflow Recommender component, which creates an optimum workflow for this particular localisation project. A Mapper component analyses this workflow and picks the most suitable components to carry out these tasks. These components are typically web services such as Machine Translation systems, Translation Memory Systems etc. A data container is circulated among the different components according to the workflow established earlier, with the components modifying the data. At the end of the project's life cycle, a Converter component transforms this data container to a translated or localised file which is returned to the user.

SOA is a key technology that has been widely adopted for integrating such highly dynamic distributed components. Our research revealed that the incorporation of an orchestration engine is essential to realize a successful SOA-based solution for coordinating localisation components. Furthermore, the necessity of a common data layer that will enable the communication between components became evident. Thus, in order to manage the processes as well as data, we incorporated an orchestration engine to the above experimental setup, providing an ideal framework for the investigation of interoperability issues among localisation components.

3.1 LocConnect

The prototype environment containing the orchestration engine and the common data layer is called LocConnect.

3.1.1 Features of LocConnect

LocConnect interconnects localisation components by providing access to an XLIFF-based data layer through an Application Programming Interface (API). By using this common data layer we allow for the traversal of XLIFF-based data between different localisation components. Key features:

- Common Data Layer and Application Programming Interface

LocConnect implements a common XLIFF-based data store (see section 4.5), accessible through an API, corresponding to individual localisation projects.

- Workflow Engine

The orchestration of components is achieved via an integrated workflow engine that executes a localisation workflow generated by another component.

- Live User Interface (UI)

An AJAX-powered UI has been developed to display the status of the components in real-time. LocConnect's UI has been developed in a manner so that it can be easily localized into other languages.

- Built-in post-editing component (XLIFF editor)

An online XLIFF editor was developed and incorporated into LocConnect in order to facilitate post-editing of content after project completion.

- Component Simulator

A component simulator was developed to allow for testing of interoperability issues using the LocConnect framework where other components were still under development or missing.

A single-click installer and administrator configuration panel for LocConnect were developed to allow for easy installation and administration.

3.1.2 Business Case

Cloud-based storage and applications are becoming increasingly popular. While the LocConnect environment supports the adhoc connection of localisation components, it can also serve as a cloud-based storage for localisation projects. Key advantages:

- Cloud-based XLIFF and resource file storage

LocConnect can be used for cloud-based XLIFF storage; it can also be used as a repository for localisation project files (e.g. TMX, SRX etc.), offering an easily-maintained central localisation data repository.

- Concurrent Versioning System (CVS)

LocConnect keeps track of changes to the XLIFF data container - both data and metadata - over a project's life cycle via versioning, thus acting as a CVS system for localisation projects.

- In-built Online XLIFF editor

Using the inbuilt online XLIFF editor, users can post-edit XLIFF content easily. It shows alternative translations as well as useful metadata associated with each translation unit.

- Access via internet or intranet

With its single click installer, it can easily be deployed through the internet or an intranet. LocConnect can also act as a gateway application where it is connected to the internet while the components can safely reside within an intranet.

- Enhanced revenues

The LocConnect-centric architecture increases data exchange efficiency as well as automation. This increased automation should produce lower localisation costs and increased productivity.

3.2 Description of Operation (Use Case)

The following scenario provides a typical use case for LocConnect in the above experimental setup.

- Project Manager (PM) logs in and creates a project (a.k.a. a job) by entering some parameters.
- PM uploads source file.
- LocConnect generates an XLIFF file and assigns a unique ID to this job.
- LocConnect stores the inputted parameters in the XLIFF file and embeds the uploaded file in the same XLIFF file as an internal file reference.
- Workflow Recommender picks up job from LocConnect and analyses it.
- Workflow Recommender generates optimum workflow and embeds workflow information in the XLIFF file, then it returns the file to LocConnect.
- LocConnect decodes the workflow information found in the XLIFF file and initiates the rest of the activities in the workflow.
- Most of the time, the XLIFF file is then sent to a Mapper Component which is responsible for selecting the optimum services for processing the XLIFF file.
- The workflow will then be enacted by the LocConnect workflow engine.
- Once processing completes, PM can post-edit the XLIFF content online using LocConnect's built-in post-editing component.
- PM can check status of components using LocConnect's live project tracking interface during project lifetime

- PM downloads processed XLIFF and the localised files.

4. Architecture

This section describes the LocConnect architecture in detail.

LocConnect is a web-based, client-server system. The design is based on a three-tier architecture as depicted in figure 1. The implementation of the system is based on PHP and AJAX technologies.

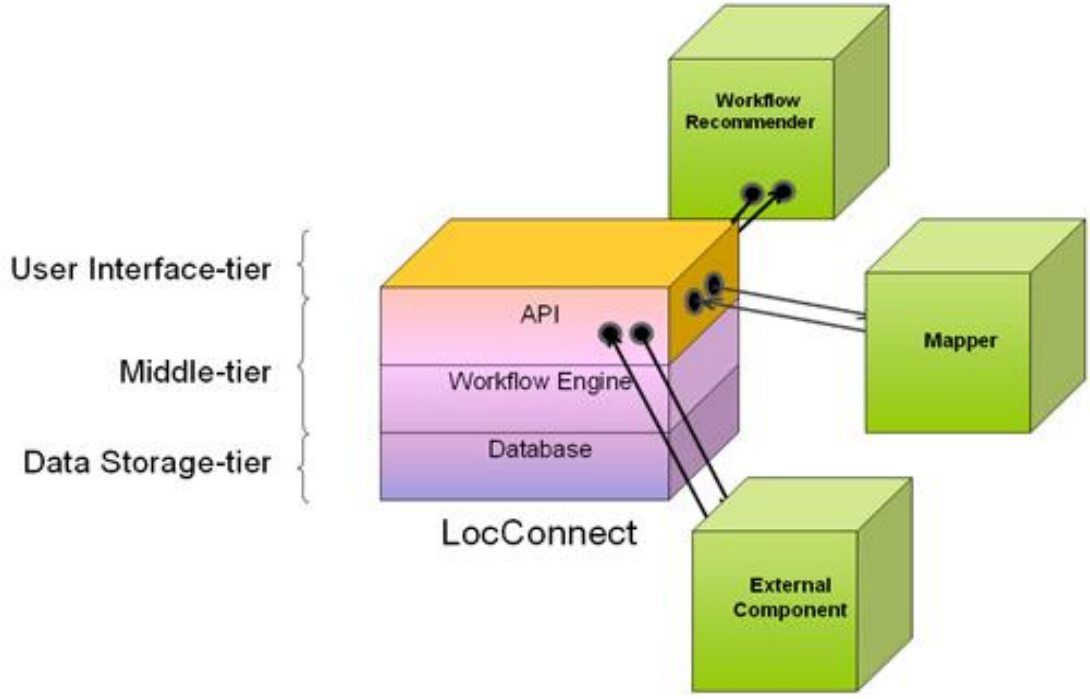

Figure 1. Three-tier architecture of LocConnect

User interface tier— client-based GUI runs on standard web browser, providing facilities for project management, administration and tracking.

Middle tier— contains most of the logic and facilitates communication between the tiers. It consists of a workflow engine and an open *API* that defines component connectivity and input output (IO) operations.

Data Storage tier— uses a relational database for data storage.

4.1 User Interface

Web-based graphical user interfaces were developed for:

1. Capturing project parameters during project creation;
2. Tracking projects (i.e. to display the current status of projects);
3. Post-editing translations;
4. Configuring the server and localising the interface of LocConnect.

The project creation interface is a web-based form used to create a project.

The project-tracking interface reflects the project's workflow. It shows the current status of a project in a graphical representation, i.e. pending, processing, or complete in relation to each component. It displays feedback messages (such as errors, warnings etc.) from components, and logs project activities (figure 2).

Figure 2. Project Tracking UI

The XLIFF post-editor interface (figure 3) displays source strings, translations, alternative translations and associated metadata, permitting translation editing. It uses AJAX to update XLIFF files in the main datastore (see section 4.4). A preliminary target file preview mechanism has also been developed and integrated.

Figure 3. Post-Editing Interface

A password-protected administration interface (figure 4) allows configuration of the LocConnect server. It also facilitates the localisation of LocConnect itself.

Figure 4. Administrator's Interface

The user interfaces were implemented in PHP, Javascript, XHTML and using the JQuery library for graphical effects and dynamic content updates.

4.2 Middle tier: Application Programming Interface (API)

The LocConnect server implements a Representational State Transfer (REST) - based interface (Fielding, 2000) to send and retrieve resources, localisation data and metadata, between components through HTTP-GET and HTTP-POST operations using proper Uniform Resource Identifiers (URI). These resources include:

- Localisation projects;
- XLIFF files;
- Resource files (i.e. non-Xliff files such as TMX, SRX etc.);
- Resource Metadata (metadata to describe resource file content).

The LocConnect API provides functions for the following tasks:

Obtaining available jobs: list_jobs method

Given a component ID, it returns an XML containing the IDs of jobs pending for any given component. It uses the HTTP GET method to communicate with the LocConnect server.

```
<jobs>
<job>16674f2698</job>
<job>633612fb37</job>
</jobs>
```

Retrieving the XLIFF file corresponding to a particular job: get_job method

Given component and job IDs, it returns the corresponding file. The returned content is always enclosed within special XML mark-up: <content>..</content> to permit non-XLIFF files. Uses HTTP GET method.

```
<content>
<xliff version='1.2'xmlns='urn:oasis:names:tc:xliff:document:1.2'>
<file original='hello.txt' source-language='en' target-language='fr' datatype='plaintext'>
<body>
<trans-unit id='hi'>
<source>Hello world</source>
<target>Bonjour le monde</target>
</trans-unit>
</body>
</file>
</xliff>
</content>
```

Setting current status: set_status method

Sets status given component ID, job ID & status. Status can be 'pending', 'processing' or 'complete'. Initially, it is set to 'pending' to mark that a job is available for pick up by a certain component. Once picked by the component, status changes to 'processing', ensuring the same job is not re-allocated to the component. When job is 'complete', LocConnect performs the next action specified in the workflow. Uses HTTP GET method.

Sending feedback message: send_feedback method

This method takes three arguments: component ID, job ID, feedback message. Components can send various messages (e.g. error messages, notifications etc.) to the server which are instantly displayed in the relevant job tracking page of the LocConnect interface. The last feedback message sent to the LocConnect server before sending the output file is stored in the activity log of the job. Messages are restricted to 256 words in length. Uses HTTP GET method.

Sending a processed XLIFF file: send_output method

This method takes three arguments: component ID, job ID and content. Once the content, usually XLIFF, is received by LocConnect, it is stored in the datastore. LocConnect waits for the component to set the status of the job to 'complete' and moves on to the next step of the workflow. Uses HTTP POST method.

Storing a resource file: send_resource method

This method takes one optional argument: resource ID and two mandatory arguments: resource file and metadata description. The resource file should be in text format. Metadata has to be specified using the following notation:

Metadata notation: 'key1:value1-key2:value2-key3:value3'

e.g. 'language:en-domain:health'

If the optional argument resource ID is not given, LocConnect will generate an ID and assign that ID to the resource file. If the resource ID is given, it will overwrite the current resource file and metadata with the new resource file and metadata. Uses HTTP POST method.

Retrieving a stored resource file: get_resource method

This method takes one argument: resource ID. Given the resource ID, the LocConnect server will return the resource associated with the given ID. Uses HTTP GET method.

Retrieving metadata associated with a resource file: get_metadata method

This method takes one argument: resource ID. The LocConnect server will return the metadata associated with the given resource ID as shown in the example below:

<metadata>

<meta key="language" value="en">

<meta key="domain" value="health">

</metadata>

Uses HTTP GET method.

4.2.1 Component-Server Communication Process

A typical communication process:

Step #1 : list_jobs

Calls list_jobs method to retrieve available jobs for that component ID.

Step #2 : get_job

Uses get_job to retrieve the XLIFF file corresponding to the given job and component IDs.

A component may either process one job at a time or many jobs at once. However, get_job method is only capable of returning a single XLIFF file at a time.

Step #3 : set_status – Set status to processing

Step #4 : Process file

Processes retrieved XLIFF file. Any feedback messages are displayed in the job tracking interface.

Step #5 : send_output

Sends processed XLIFF file back to the LocConnect server using send_output method.

Step #6 : set_status

Sets status of the selected job to 'complete', triggering LocConnect to move to the next stage of the workflow.

4.3 Middle tier: Workflow Engine

A simple workflow engine is incorporated into the LocConnect server. It does not support parallel processes or branching, but does permit multiple uses of a component. It parses the workflow information found in the XLIFF data container (see section 4.5) and stores workflow information in the project management datastore, which is used to keep track of individual projects.

4.4 LocConnect Datastore

The database design can be logically stratified in 3 layers:

- Main datastore holds XLIFF files;
- Project management datastore holds data about individual project status;
- Resource datastore holds data and metadata about other resource files;

The main datastore is used to store XLIFF files corresponding to different jobs, and supports versioning across jobs, thus acting as a Concurrent Versions System (CVS) for localisation projects.

The project management datastore is used for storing information needed for keeping track of individual localisation jobs with respect to localisation workflows, as well as time-stamps for job pick-up time, job completion time etc. by different components.

The resource datastore is used to store various text-based asset files (TMX, XLIFF, SRX, TBX, XML etc) associated with localisation projects. The components can store any intermediate files, temporary or backup files in this datastore. The files can be accessed at any stage during workflow execution. The resource files (i.e. asset files) can be described further using metadata. Metadata consists of key-value pairs associated with the resource files. The metadata can also be stored in the resource datastore.

SQLite was chosen as the default database, as it is easily deployed, lightweight and virtually administration-free.

4.5 XLIFF Data Container

The core of this architecture is the XLIFF-based data container defined in this research. Maximum effort has been made to abstain from custom extensions, as different components will access and make changes to this data container as it travels through different components and different phases of the workflow. The typical structure of the data container is given in figure 5.

```
<xliff version="1.2" xmlns="urn:oasis:names:tc:xliff:document:1.2">
 <file original="hello.txt" source-language="en" target-language="fr" datatype="plaintext"
  category="medical">
  <header>

   <skl> <internal-file> </internal-file>  </skl>
                                                              Section 1
   <reference>
    <internal-file form="base64">
    <original-file fileformat="exe"> </original-file>
    </internal-file>
   </reference>
                                                              Section 2
   <reference>
    <internal-file>
     <metadata>
      <meta pname="testProject"
       pdescription="A test project"
       startdate="01/04/2011"
       deadline="10/12/2011"
       budget="13310"
       quality-requirement="High"
       use-mt="yes"
       use-rating="yes"
      />
     </metadata>
    </internal-file>
   </reference>
                                                              Section 3
      <reference>
            <workflow>
                  <task tool-id="LMC" order="1" status="pending"/>
            </workflow>
      </reference>
                                                              Section 4
      <reference>
            <external-file href="http:// LocConnect/get_resource.php?id=fcb4c5a8f1"/>
      </reference>

      <phase-group>
         <phase phase-name="Project Initiate" process-name="project creation and
      requirement capturing" tool-id="LocConnect"     company-name="LRC"/>

         <phase phase-name="Quality Assurance" process-name="authoring" tool-
      id="LKR"  company-name="LKR" contact-name="Dave O Carroll" contact-
      email="contact@example.com"/>
      </phase-group>

      <tool tool-name="LocConnect"  tool-id="LocConnect"  tool-version="2.0"/>

  </header>
  <body>..
  </body>
 </file>
</xliff>
```

Figure 5. XLIFF-Based Data Container

When a new project is created in LocConnect, it will append parameters captured through the project creation page in the metadata section (see section 2) of the data container. The metadata is stored as key-value pairs. During the workflow execution process, various components may use, append or change the metadata. The source file uploaded by the user will be stored within the XLIFF data container as an internal file reference (see section 1). Any resource files uploaded during the project creation will also be stored as external-references as shown in section 4. The resource files attached to this data container can be identified by their unique IDs and can be retrieved at any stage during the process. Furthermore, the identifier will allow retrieving the metadata associated with those resources.

After the project creation, the Workflow Recommender component analyses the data container, and if the original file is in a format other than XLIFF, it will suggest using a File Format Converter component to convert the source file into XLIFF. The converted content will be stored in the same data container using the <body> section and the skeleton sections. The converted data container may be then re-analysed to propose the rest of the workflow. The workflow information is stored in section 3 of the data container. When the LocConnect server receives the data container back from the Workflow Recommender component, it will parse the workflow description and execute the rest of the sequence. Once the entire process is completed, the converter can use the data container to build the target file.

In this architecture, a single XLIFF-based data container is being used throughout the process. Different workflow phases and associated tools can be identified by the standard XLIFF elements such as <phase> and <tools>. Tools can include statistics (e.g. <count-groups>) in the same XLIFF file.

The XLIFF data container - based architecture resembles the Transmission Control Protocol and the Internet Protocol (TCP/IP) architecture, where data packet is routed based on its content. However, in this scenario, LocConnect plays several roles, including the role of a router, web server and a file server.

5. Discussion and Future Work

Whilst the prototype provides a test bed for the exploration of interoperability issues among localisation tools, it has a number of limitations.

In the present architecture, metadata is being stored as attribute-value pairs within an internal file reference of the XLIFF data container (see section 3 of figure 5). However, according to the current XLIFF specification (XLIFF Technical Committee, 2008), XML elements cannot be included within an internal file reference. While this could be interpreted as a limitation of the XLIFF standard itself, the current metadata representation mechanism also presents several problems. The metadata is exposed to all the components, yet there might be situations relating to security or visibility where access should be limited. These problems can be overcome by separating the metadata from the XLIFF data container. Then, specific API functions can be implemented to allow components manipulate metadata securely (e.g. add, delete, modify, retrieve).

Resource Description Framework (RDF) is a framework for describing metadata (Anastasiou, 2011:42-52) that is worthy of consideration. Perhaps API functions could be implemented to return metadata required by a component in RDF syntax.

The current API lacks several crucial functions, such as deleting projects (and associated XLIFF files), modifying projects, deleting resource files and modifying metadata associated with resource files. A mechanism could be implemented for granting proper permissions to components for using the above functions. User management is a significant aspect that we did not pay much attention to when developing the initial test bed. User roles could be designed and implemented so that users with different privileges can assign different permissions to components as well as different activities managed through the LocConnect

server, thus achieving some data security. Furthermore, an API key should be introduced for the validation of components as another security measure.

The XLIFF data container itself might contain sensitive data that requires a mechanism to secure the content. Some suggestions are: let the workflow recommender (or the Mapper) select only the secure components; encrypt the content; or implement API functions to access specific parts of the XLIFF data container.

As the XLIFF standard was originally defined as a localisation data exchange format, it has so far not been thoroughly assessed as a data container, but a systematic evaluation could be facilitated by our prototype. Different approaches to addressing likely performance issues could be explored, such as data container compression, support for parallel processing, or the use of multiple XLIFF-based data containers transmitted in a single compressed container.

While the current workflow engine provides the essential process management operations, it currently lacks more complex features such as parallel processes and branching. Therefore, incorporation of a fully-fledged workflow engine into the LocConnect server is desirable. Ideally, the workflow engine should support standard workflow description languages such as Business Process Execution Language (BPEL) or Yet Another Workflow Language (YAWL), possibly storing the generated workflow as a separate resource file and providing the link to the resource file in the XLIFF data container as an external file reference. This would allow LocConnect to be connected easily to an existing business process, i.e. localisation could be included as a part of an existing workflow.

LocConnect implements REST-based services for communication with external components. Therefore, it is essential to implement our own security measures in the REST-based API. Since there are no security measures implemented in the current LocConnect API, well-established powerful security measures (e.g.XML encryption, API keys) would need to be implemented in the API as well as in the data transmission channel (e.g. the use of Secure Socket Layer (SSL) tunnels for REST calls).

Currently, the LocConnect server implements a 'PULL' based architecture where components have to initiate the data communication process. The implementation of both 'PUSH' and 'PULL' based architectures would very likely yield more benefits. Such architecture would help to minimize communication overhead as well as resource consumption. It would also help to check the availability of the components prior to assigning a job, and help the LocConnect server to detect component failures, a capability the current architecture lacks. It would then be possible to automatically re-route the workflow whenever the LocConnect server detects a component failure.

The current resource datastore is only capable of storing textual data. If it could also store binary data then it could handle windows executable files, dll files, video files, images etc.

Finally, the information about components currently has to be manually registered with the LocConnect server using its administrator interface. This should be improved to discover and register ad-hoc components automatically.

5.1 Proposed improvements to the XLIFF based data container and new architecture

By addressing the issues related to the above XLIFF-based data container, a fully XLIFF compliant data container could be developed to evaluate its effect on improvements in interoperability. It would differ from the current data container (see figure 5) in the following aspects:

The new container:

- Would not represent additional metadata (i.e. metadata other than that defined in the XLIFF specification) within the data container itself. It would instead be stored in a separate metadata store.
- Would not represent workflow metadata as an internal file reference. Instead, the workflow metadata will be stored separately in the resource datastore, bound by a link to an external file reference.
- Would not store the original file as an internal file reference. It would be stored separately in the resource datastore, and linked via an external file reference.

Thus, the new data container would not use any extensions to store additional metadata or data, nor would it use XML syntax within internal-file elements, meaning that it would provide an XLIFF strict schema compatible interoperability architecture. Due to separation of the original file content, workflow information and metadata from the XLIFF data container, the container itself would become lightweight and easy to manipulate. File format converter components would also become less complex.

6. Conclusions

In this paper we presented and discussed a service-oriented framework that was developed and then applied to evaluate interoperability in localisation process management using the XLIFF standard. The use cases, architecture and issues of this approach were discussed. A prototype of the framework was successfully demonstrated at the CNGL Public Showcase in Microsoft, Ireland, in November 2010.

The framework has revealed the additional metadata and related infrastructure services required for linking distributed localisation tools and services. It has also been immensely helpful in identifying prominent issues that need to be addressed when developing a commercial application.

The prototype framework described in this paper is the first to use XLIFF as a data container to address interoperability issues among localisation tools. In our opinion, the successful implementation of this pilot prototype framework suggests the suitability of XLIFF as a full project life-cycle data container that can be used to achieve interoperability in localisation processes. The LocConnect framework will serve as a platform for future research on interoperability issues in localisation.

Acknowledgement

This research is supported by the Science Foundation Ireland (Grant 07/CE/I1142) as part of the Centre for Next Generation Localisation (www.cngl.ie) at Localisation Research Centre, CSIS Dept., University of Limerick, Limerick, Ireland. We would also like to acknowledge the vital contributions of our colleagues and fellow researchers from the CNGL project.

References

Anastasiou, D. and Morado Vázquez, L. (2010). "Localisation Standards and Metadata", *Proceedings Metadata and Semantic Research, 4th International Conference (MTSR 2010)*. Communications in Computer and Information Science, Springer. 255-276.

Anastasiou, D. (2011). "The Impact of Localisation on Semantic Web Standards", *in European Journal of ePractice, N. 12*, March/April 2011, ISSN 1988-625X. 42-52.

Bichler, M. and Lin, K. J. (2006). Service-oriented computing. IEEE Computer 39(3): 99–101.

Bly, M. (2010). "XLIFFs in Theory and in Reality". http://bit.ly/vPIHXR. Page consulted on: 09.06.11.

Corrigan, J. & Foster, T. (2003). XLIFF: An Aid to Localization. http://developers.sun.com/dev/gadc/technicalpublications/articles/xliff.html. Last updated: 16.02.10. Page consulted on: 22.06.09.

Fielding, R. (2000). "Architectural Styles and the Design of Network-based Software Architectures, PhD thesis", University of California, Irvine. http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

Hallé, S., Bultan, T., Hughes, G. and Alkhalaf, M. (2010). Runtime Verification of Web Service Interface Contracts, Computer, March. 59-66.

Hayes, J. G., Peyrovian, E., Sarin, S., Schmidt, M. T., Swenson, K. D., Weber, R. (2000). "Workflow interoperability standards for the Internet," *IEEE Internet Computing 4*, no. 3 (June 2000). 37-45.

IEEE. (1991). IEEE Standard Computer Dictionary. *A Compilation of IEEE Standard Computer Glossaries*. IEEE Std 610, p.1.

Kindrick, J. D., Sauter, J. A. and Matthews, R.S. (1996). "Improving conformance and interoperability testing", Standard View 4, no. 1 (3, 1996): 61-68.

Lewis, G., Morris, E., Simanta, S.,Wrage, L. (2008). "Why Standards Are Not Enough to Guarantee End-to-End Interoperability". Commercial-off-the-Shelf (COTS)-Based Software Systems, *Seventh International Conference on Composition-Based Software Systems (ICCBSS 2008).* 164-173.

Morado Vázquez, L. and Wolff, F. (2011). "Bringing industry standards to Open Source localisers: a case study of Virtaal". Tradumàtica, 9.

Schäler, R. (2009). Communication as a Key to Global Business. In: Hayhoe, G. *Connecting people with technology: issues in professional communication*. Amityville N.Y. Baywood Pub.. 57-67.

XLIFF Technical Committee. (2008). XLIFF 1.2 Specification. http://docs.oasis-open.org/xliff/xliff-core/xliff-core.html. Last updated: 01.02.08. Page consulted on: 25.06.09.

W3C. (2007). Web Service Description Language (WSDL) Version 2.0 Part 1:Core Language, W3C Recommendation*. In Chinnici, R., Moreau, J, J., Ryman, A. and Weerawarana, S. eds. W3C*. http://www.w3.org/TR/wsdl20.

XLIFF Technical Committee. (2010). XLIFF2.0 / Feature Tracking. http://wiki.oasis-open.org/xliff/XLIFF2.0/FeatureTracking. Last update: 14.03.11. Page consulted on: 23.07.09.