LocConnect: Orchestrating Interoperability in a Service-oriented Localisation Architecture

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, Ian.OKeeffe, Reinhard.Schaler}@ul.ie

Abstract
Interoperability is the key to the seamless integration of different entities. However, while it is one of the most challenging problems in localisation, interoperability has not been discussed widely in the relevant literature. This paper describes the design and implementation of a novel environment for the inter-connectivity of distributed localisation components, both open source and proprietary. 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*

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). As the definition suggests, localisation is a complex process. Localisation involves many steps: project management, translation, review, quality assurance etc. It also requires a considerable effort as it involves many languages, dealing with characteristics and challenges unique to each of these languages such as the handling of right-to-left scripts, collation, and locale specific issues. Time-frame is another parameter that affects the complexity of the localisation process. Localisation processes require dealing with 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, ranging from project management software to translation software. A large number of file formats are encountered during the localisation process. These file formats may consist of both open standard and proprietary file formats. Localisation processes involve different types of organisations (e.g. translation and localisation service providers) and different professions (e.g. translators, reviewers, and linguists). Localisation constantly has to deal with 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 quality (translations, user interfaces etc.) and quantity (number of locales, simships etc.) while minimising time and overall cost.

Interoperability is the key to the seamless integration of different technologies and components across the localisation process. The term interoperability has been defined in a number of different ways in the literature. For example, Lewis et al. (2008) define interoperability as: "The ability of a collection of communicating entities to (a) share specified information and (b) operate on that information according to an agreed operational semantics".

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 had much attention paid to it 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.

The remainder of the paper is organized as follows: Section 2 provides an overview of interoperability in general, and in localisation in particular, in the context of open localisation standards; Section 3 explains the experimental setup, introduces the LocConnect framework, and presents the localisation component interoperability environment developed as part of this research; Section 4 presents the architecture of LocConnect in detail; and section 5 discusses future work. The paper concludes with a summary of the present work and the contributions made by this study.

2. Background

Currently, 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) which does not specify message sequence or its constraints on parameters (Halle et al 2010). However, while it does describe the public interface to a web service, 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 the quality, quantity, time and cost aspects discussed earlier. These service attributes are much more difficult to define, as they cover the non-functional aspects of a service, e.g. how well it is performed. This contrasts with the more Boolean functional requirements (either it complies with the service support requirements, or it does not). 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 requiring the intervention of a skilled computer programmer to set up the call to the service, to validate the data sent in terms of string length, language pair, and so on, and then to handle the data that is returned. 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 into these workflows through the development of 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-based 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-TC 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-TC 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. The XLIFF standard is being continuously developed further by the OASIS XLIFF Technical Committee (2010).

2.2 Localisation Standards and Interoperability Issues

Although the adoption of localisation standards would very likely provide benefits relating to reusability, accessibility, interoperability, and reduced cost, software publishers often refrain from the full implementation of a 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 the 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), i.e. the need for the output of one tool to be transformed in order to compensate for another tool's non-conforming behaviour. This trait is present within the localisation software industry. Although the successful integration of different software brings enormous benefits, it is still a very arduous task.

Most current CAT tools, while accepting and delivering a range of file formats, maintain their own proprietary data formats within the boundary of the application. This makes sharing of data between tools from different software developers 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, rather than being used to the full extent of its abilities, and indeed many CAT tools seem to pay mere lip service to the XLIFF specification (Anastasiou and Morado-Vazquez 2010; 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. However, such an ideal component-based interoperability scenario in localisation is still far from reality. Therefore, in this research, we aim to model this ideal scenario by implementing a series of prototypes. As the initial step, an experimental setup has been designed containing the essential components.

The experimental setup includes 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 sent to a Workflow Recommender component. The Workflow Recommender implements the appropriate business process. By analysing source file content, resource files as well as parameters provided by the user, the Workflow Recommender offers an optimum workflow for this particular localisation project. Then, a Mapper component analyses this workflow and picks the most suitable components to carry out the tasks specified in the workflow. These components can be web services such as Machine Translation systems, Translation Memory Systems, Post Editing systems etc. The Mapper will establish links with the selected components. Then a data container will be circulated among the different components according to the workflow established earlier. As this data container moves through different components, the components modify 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.

Service Oriented Architecture 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 realise 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 into the aforementioned experimental setup. This experimental setup along with the orchestration engine provide an ideal framework for the investigation of interoperability issues among localisation components.

3.1 LocConnect

At the core of the experimental setup are the orchestration engine and the common data layer, which jointly provide the basis for the exploration of interoperability issues among components. This prototype environment is called LocConnect. The following sections introduce the features of LocConnect and describe its architecture.

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 of the LocConnect testing environment are summarized below.

- Common Data Layer and Application Programming Interface

LocConnect implements a common XLIFF-based datastore (see section 4.5) corresponding to individual localisation projects. The components can access this datastore through a simple API. Furthermore, the common datastore can also hold various supplementary resource files related to a localisation project (see section 4.4). Components can manipulate these resource files through the API.

- 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)

One of the important aspects of a distributed processing scenario is the ability to track progress along the different components. 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 that allows it to be easily localised into other languages.

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

In the present architecture, localisation project creation and completion happens within LocConnect. Therefore, an online XLIFF editor was developed and incorporated into LocConnect in order to facilitate post-editing of content.

- Component Simulator

In the current experimental setup, only a small number of components, most of them developed as part of the CNGL research at the University of Limerick and other participating research groups, have been connected up. The Workflow Recommender, Mapper, Leveraging Component and a Translating Rating component are among these components. A component simulator was, therefore, developed to allow for further testing of interoperability issues in an automated localisation workflow using the LocConnect framework.

A single-click installer and administrator configuration panel for LocConnect were developed as a part of this work to allow for easy installation and user-friendly 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 cloud-based storage for localisation projects. These and other key advantages of LocConnect from a business point of view are highlighted below.

- Cloud-based XLIFF and resource file storage

LocConnect can simply be used as a cloud-based XLIFF storage. Moreover, due to its ability to store resource files (e.g. TMX, SRX etc.), it can be used as a repository for localisation project files. As such, LocConnect offers a central localisation data repository which is easy to backup and maintain.

- Concurrent Versioning System (CVS)

During a project's life cycle, the associated XLIFF data container continuously changes as it travels through different localisation components. LocConnect keeps track of these changes and stores different versions of the XLIFF data container. Therefore, LocConnect acts as a CVS system for localisation projects. LocConnect provides the facility to view both data and metadata associated with the data container at different stages of a workflow.

- In-built Online XLIFF editor

Using the inbuilt online XLIFF editor, users can edit XLIFF content easily. The AJAX-based UI allows easy inline editing of content. Furthermore, the online editor 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 via the internet or an intranet. LocConnect can also act as a gateway application where LocConnect 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. Due to increased automation, we would expect 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.

A project manager logs into the LocConnect server and creates a LocConnect project (a.k.a. a job) by entering some parameters. Then the project manager uploads a source file. The LocConnect server will generate an XLIFF file and assign a unique ID to this job. Next, it will store the parameters captured through its interface in the XLIFF file and embed the uploaded file in the same XLIFF file as an internal file reference. The Workflow Recommender will then pick up the job from LocConnect (see the procedure described in section 4.2.1), retrieve the corresponding XLIFF file and analyse it. The Workflow Recommender will generate an optimum workflow to process the XLIFF file. The workflow describes the other components that this XLIFF file has to go through and the sequence of these components. The Workflow Recommender embeds this workflow information in the XLIFF file. Once the workflow information is attached, the file will be returned to the LocConnect server. When LocConnect receives the file from the Workflow Recommender, it decodes the workflow information found in the XLIFF file and initiates the rest of the activities in the workflow. Usually, the next activity will be to send the XLIFF file to a Mapper Component which is responsible for selecting the best web services, components etc. for processing the XLIFF file. LocConnect will establish communication with the other specified components according to the workflow and component descriptions. As such, the workflow will be enacted by the LocConnect workflow engine. Once the XLIFF file is fully processed, XLIFF content can be edited online using LocConnect's built-in editing component. During the project's lifecycle, the project manager can check the status of the components using LocConnect's live project tracking interface. Finally, the project manager can download the 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.

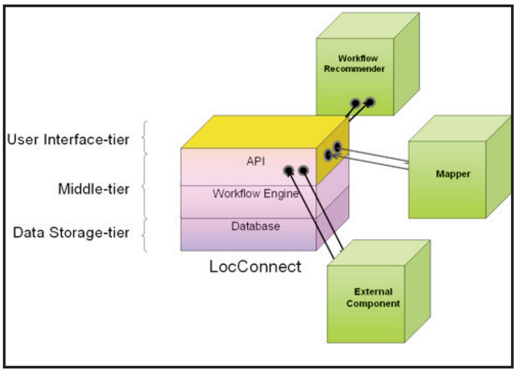


Figure 1. Three-tier architecture of LocConnect

User interface tier - a client-based graphical user interface that runs on a standard web browser. The user interface provides facilities for project management, administration and tracking.

Middle tier - contains most of the logic and facilitates communication between the tiers. The middle tier mainly consists of a workflow engine and provides an open API with a common set of rules that define the connectivity of components and their input output (IO) operations. The components simply deal with this interface in the middle tier.

Data Storage tier - uses a relational database for the storage and searching of XLIFF and other resource data. The same database is used to store information about individual projects.

The tiers are described below.

4.1 User Interface

Web-based graphical user interfaces were developed for:

1. Capturing project parameters during project creation;

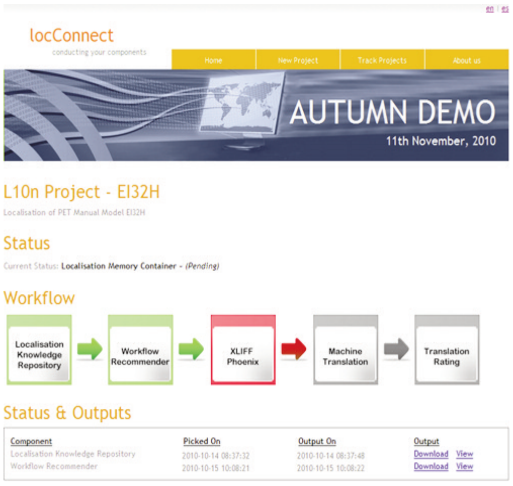
2. Tracking projects (i.e. to display the current status of projects);

3. Post-editing translations;

4. Configuring the server and localising the interface of LocConnect.

During project creation, a web-based form is presented to a user. This form contains fields that are required by the Workflow Recommender to generate a workflow. Parameters entered through this interface will be stored in the XLIFF file along with the uploaded source file (or source text) and resource files. The project is assigned a unique ID through this interface and this ID is used throughout the project's lifecycle.

The project-tracking interface reflects the project's workflow. It shows the current status of a project, i.e. pending, processing, or complete in relation to each component. It displays any feedback messages (such as errors, warnings etc.) from components. The current workflow is shown in a graphical representation. Another important feature is a log of activities for the project. Changes to the XLIFF file (i.e. changes of metadata) during different stages of the workflow can be tracked. The project-tracking interface uses AJAX technologies to dynamically update its content frequently (see figure 2).

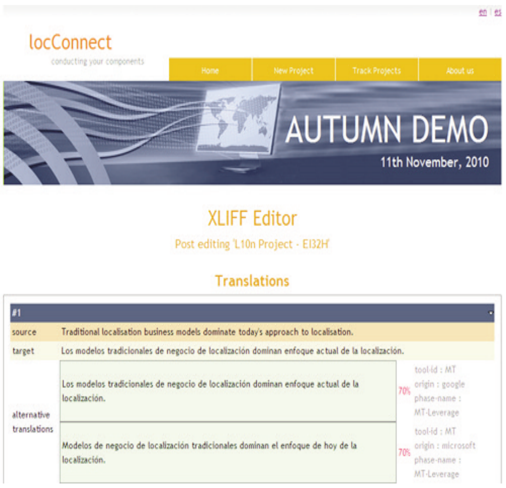


Current Status: Localisation Memory Container - *(Pending)*

Component	Picked On	Output On	Output
Localisation Knowledge Repository	2010-10-14 08:37:32	2010-10-14 08:37:48	Download View
Workflow Recommender	2010-10-15 10:08:21	2010-10-15 10:08:22	Download View

Figure 2. Project Tracking UI

At the end of a project's lifecycle, the user is given the option to post-edit its content using the built-in XLIFF post-editor interface. It displays source strings, translations, alternative translations and associated metadata. Translations can be edited through this interface. The Post-editing component also uses AJAX to update XLIFF files in the main datastore (see section 4.4). See figure 3 for a screen-shot of the post-editing interface. A preliminary target file preview mechanism has been developed and integrated into the same UI.



XLIFF Editor

Post editing 'L10n Project - EI32H'

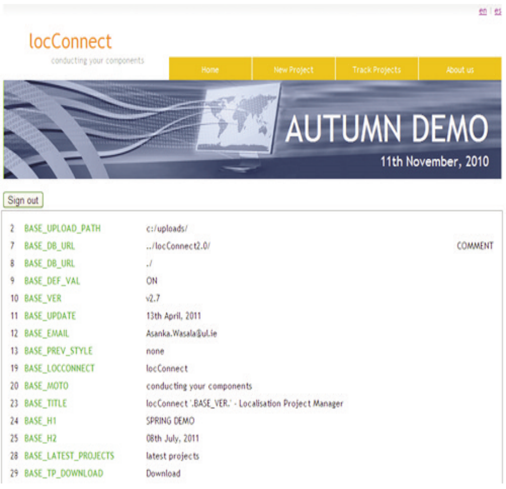
Translations

#1

source	Traditional localisation business models dominate today's approach to localisation.		
target	Los modelos tradicionales de negocio de localización dominan enfoque actual de la localización.		
alternative translations	Los modelos tradicionales de negocio de localización dominan enfoque actual de la localización.	70%	tool-id : MT origin : google phase-name : MT-Leverage
	Modelos de negocio de localización tradicionales dominan el enfoque de hoy de la localización.	70%	tool-id : MT origin : microsoft phase-name : MT-Leverage

Figure 3. Post-Editing Interface

A password-protected interface has been provided for the configuration of the LocConnect server. Through this interface various configuration options such as LocConnect database path, component descriptions etc. can be edited. The same interface can be used to localise the LocConnect server itself (see figure 4 for a screenshot of the administrator's interface).



2	BASE_UPLOAD_PATH	c:/uploads/	
7	BASE_DB_URL	../locConnect2.0/	COMMENT
8	BASE_DB_URL	./	
9	BASE_DEF_VAL	ON	
10	BASE_VER	v2.7	
11	BASE_UPDATE	13th April, 2011	
12	BASE_EMAIL	Asanka.Wasala@ul.ie	
13	BASE_PREV_STYLE	none	
19	BASE_LOCCONNECT	locConnect	
20	BASE_MOTO	conducting your components	
23	BASE_TITLE	locConnect '.BASE_VER.' - Localisation Project Manager	
24	BASE_H1	SPRING DEMO	
25	BASE_H2	08th July, 2011	
28	BASE_LATEST_PROJECTS	latest projects	
29	BASE_TP_DOWNLOAD	Download	

Figure 4. Administrator's Interface

The user interfaces were implemented in PHP, Javascript, XHTML and use 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. files such as TBX, TMX, SRX etc.);
- Resource Metadata (metadata to describe resource file content).

The LocConnect API provides functions for the following tasks:

1. Retrieving a list of jobs pending for a particular component (list_jobs method);
2. Retrieving an XLIFF file corresponding to a particular job (get_job method);
3. Setting the status of a job. The status can be one of the following: Pending, Processing, Complete (set_status method);
4. Sending a feedback message to the server (send_feedback method);
5. Sending processed XLIFF files to the sever (send_output method);
6. Sending a resource file (i.e. a non-XLIFF asset file) to the server (send_resource method);
7. Retrieving a resource file from the server (get_resource method);
8. Retrieving metadata associated with a resource file (get_metadata method).

A complete description of each REST-based function is provided below.

Obtaining available jobs: list_jobs method
This method takes a single argument: component ID. It will return an XML containing the IDs of jobs pending for any given component. The IDs are alphanumeric and consist of 10 characters. The component ID is a string (usually, a short form of a component's name, such as WFR for Workflow

Recommender).

This method 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
This method takes two arguments: component ID and job ID. It will return a file corresponding to the given job ID and component ID. Usually, the file is an XLIFF file, however it can be any text-based file. Therefore, the returned content is always enclosed within special XML mark-up: <content>..</content>. The XML declaration of the returned file will be omitted in the output (i.e. <?xml version="1.0" ..?> will be stripped off from the output).

This method uses the HTTP GET method to communicate with the LocConnect server.

```
<content><xliffversion='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
This method takes three arguments: component ID, job ID, status. The status can be 'pending', 'processing' or 'complete'. Initially, the status of a job is set to 'pending' by the LocConnect server to mark that a job is available for pick up by a certain component. Once the job is picked by the component, it will change the status of the job to 'processing'. This ensures that the same job will not be re-allocated to the component. Once the status of a job is set to 'complete', LocConnect will perform the next action specified in the workflow.

This method uses the HTTP GET method to communicate with the LocConnect server.

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 through this method. These messages will be 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 will be stored within the LocConnect server and it will appear in the activity log of the job. The messages are restricted to 256 words in length.

This method uses the HTTP GET method to communicate with the LocConnect server.

Sending a processed XLIFF file: send_output method
This method takes three arguments: component ID, job ID and content. The content is usually a processed XLIFF file. Once the content is received by LocConnect, it will be stored within the LocConnect datastore. LocConnect will wait for the component to set the status of the job to 'complete' and move on to the next step of the workflow.

This method uses the HTTP POST method to communicate with the LocConnect server.

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.

This method usew the HTTP POST method to communicate with the LocConnect server.

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.

This method uses the HTTP GET method to communicate with the LocConnect server.

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>
```

This method uses the HTTP GET method to communicate with the LocConnect server.

4.2.1 Component-Server Communication Process
A typical LocConnect component-server communication process includes the following phases.

Step 1: list_jobs

This component calls the list_jobs method to retrieve a list of available jobs for that component by specifying its ID.

Step 2: get_job

This component uses get_job to retrieve the XLIFF file corresponding to the given job ID and the component ID.

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

Step 3: set_status - Set status to processing

This component sets the status of the selected job to 'processing'.

Step 4: Process file

This component processes the retrieved XLIFF file. It may send feedback messages to the server while processing the XLIFF file. These feedback messages will be displayed in the job tracking interface of the LocConnect.

Step 5: send_output

This component sends the processed XLIFF file back to the LocConnect server using send_output method.

Step 6: set_status

This component sets the status of the selected job to 'complete'. This will trigger the LocConnect server to move to the next stage of the workflow.

4.3 Middle tier: Workflow Engine
A simple workflow engine has been developed and incorporated into the LocConnect server to allow for the management and monitoring of individual localisation jobs. The current workflow engine does not support parallel processes or branching. However, it allows the same component to be used several times in a workflow. The engine parses the workflow information found in the XLIFF data container (see section 4.5) and stores the workflow information in the project management datastore. The project management datastore is then used to keep track of individual projects. In the current setup, setting the status of a component to 'complete' will trigger the next action of the workflow.

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 projects and their status;
- Resource datastore holds data and metadata about other resource files;

The main datastore is used to store XLIFF files corresponding to different jobs. It stores different versions of the XLIFF file that correspond to a particular job. Therefore, the LocConnect server also acts as a Concurrent Versions System (CVS) for localisation projects.

The project management datastore is used for storing the information necessary to keep track of individual localisation jobs with respect to localisation workflows. Furthermore, it is used to store various time-stamps such as job pick-up time, job completion time etc by different components.

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

SQLite was chosen as the default database for implementing the logical data structure in this prototype, for a number of reasons. Firstly, it can be easily deployed. It is lightweight and virtually no administration required. Furthermore, it does not require any configuration.

4.5 XLIFF Data Container

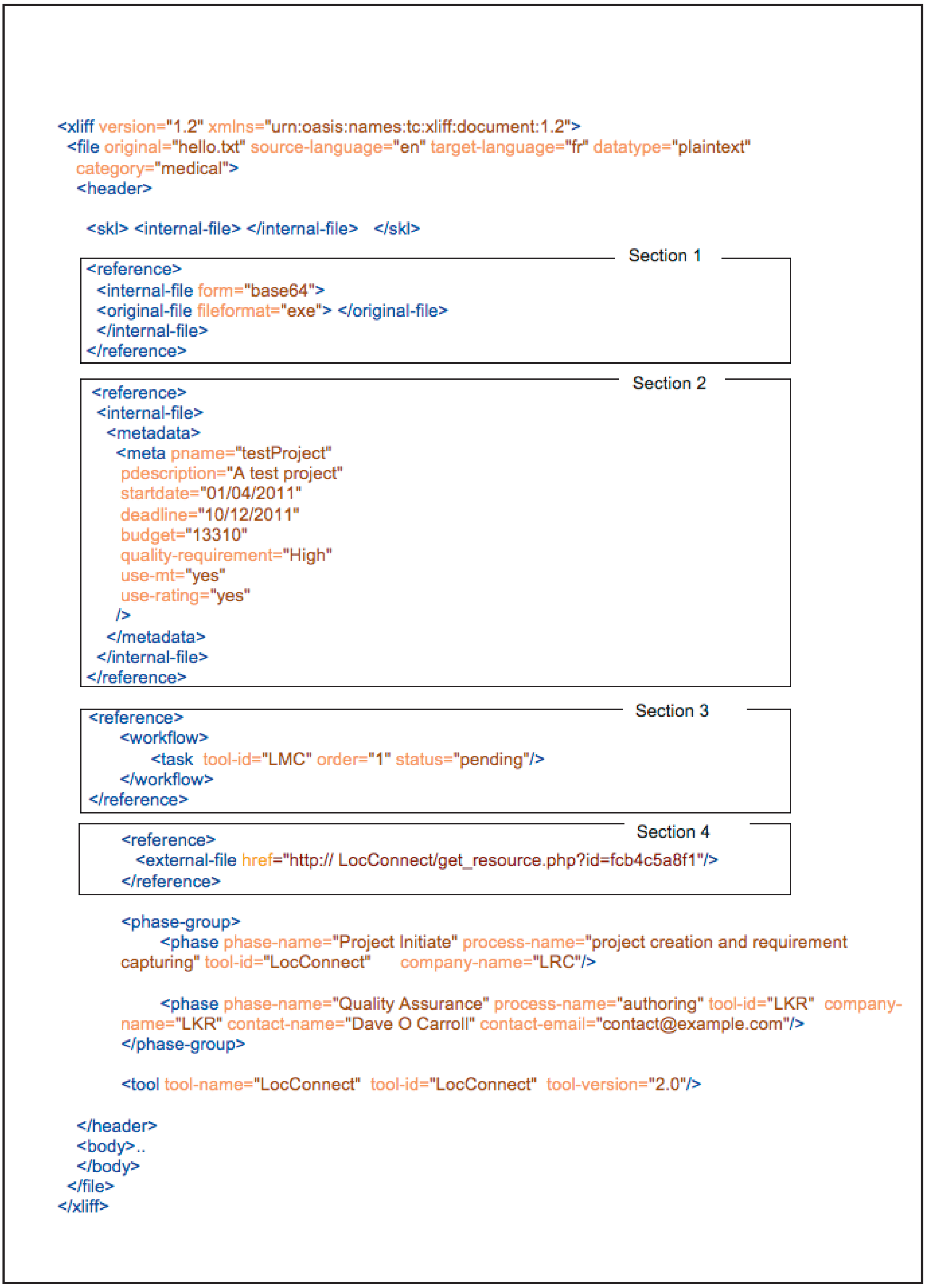
The core of this architecture is the XLIFF-based data container defined in this research. Maximum effort has been made to abstain from custom extensions in defining this data container. Different components will access and make changes to this data container as it travels through different components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

When a new project is created in LocConnect, it will append parameters captured via the project creation page into 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 4.4. 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 retrieval of the metadata associated with those resources.

After project creation, the data container generated (i.e. the XLIFF file) is sent to the Workflow Recommender component. It analyses the project metadata as well as the original file format to recommend the optimum workflow to process the given source file. If the original file is in a format other than XLIFF, the Workflow Recommender will suggest that the data container to be sent to a File Format Converter component. The file format converter will read the original file from the above internal-file reference and 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 data container with the converted file content is then reanalysed by the Workflow Recommender component in order to propose the rest of the workflow. The workflow information will be 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>. Furthermore, tools can include various 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 in that the 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.


Section 1
Section 2
Section 3
Section 4

```
<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>

  <reference>
    <internal-file form="base64">
    <original-file fileformat="exe"> </original-file>
    </internal-file>
  </reference>

   <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>

  <reference>
      <workflow>
          <task  tool-id="LMC" order="1" status="pending"/>
      </workflow>
  </reference>

      <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

5. Discussion and Future Work

Savourel (2007) highlights the importance of a "Translation Resource Access API" which facilitates localisation data exchange among different systems in a heterogeneous environment. Like Savrourel (2007) we also believe that access to a common data layer through an API would enable interoperability between different localisation components. The development of the prototype has revealed syntactical requirements of such an API as well as the common data layer. 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-TC 2008), XML elements cannot be included within an internal file reference. Doing so will result in an invalid XLIFF file. 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 where metadata should only be exposed to certain components. Therefore, some security and visibility mechanisms have to be implemented for the metadata. Moreover, there may be situations where components need to be granted specific permissions to access metadata, e.g. read or write. These problems can be overcome by separating the metadata from the XLIFF data container. That is, the metadata has to be stored in a separate datastore (as in the case of resource files). Then, specific API functions can be implemented to manipulate metadata (e.g. add, delete, modify, retrieve) by different components. This provides a secure mechanism to manage metadata.

The Resource Description Framework (RDF) is a framework for describing metadata (Anastasiou 2011). Therefore, it is worthwhile exploring the possibility of representing metadata using RDF. For example, API functions could be implemented to return the metadata required by a component in RDF syntax.

The current API lacks several important functions. Functions should be implemented for deleting projects (and associated XLIFF files), modifying projects, deleting resource files and modifying metadata associated with resource files etc. The current API calls set_output and set_status to 'complete' could be merged (i.e. sending the output by a component will automatically set its status to 'complete'). Furthermore, 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. This way, data security could be achieved to a certain extent. Furthermore, an API key should be introduced for the validation of components as another security measure. This way, components would have to specify the key whenever they use LocConnect API functions in order to access the LocConnect data.

The XLIFF data container could contain sensitive data (i.e. source content, translations or metadata) which some components should not be able to access. A mechanism could be implemented to secure the content and to grant permissions to components so that they would only be able to access relevant data from the XLIFF data container. There are three potential solutions to this problem. One would be to let the workflow recommender (or the Mapper) select only secure and reliable components. The second solution could be to encrypt content within the XLIFF data container. The third solution could be to implement API functions to access specific parts of the XLIFF data container. However, the latter mechanism will obviously increase the complexity of the overall communication process due to frequent API calls to the LocConnect server.

Because the XLIFF standard was originally defined as a localisation data exchange format, it has, so far, not been thoroughly assessed with regard to its suitability as a localisation data storage format or as a data container. A systematic evaluation has to be performed on the use of XLIFF as a data container in the context of a full localisation project life cycle, as facilitated by our prototype. For example, during the traversal, an XLIFF-based data container could become cumbersome causing performance difficulties. 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. The implications of such strategies would have to be evaluated, such as the need to equip the components with a module to extract and compress the data container.

While the current workflow engine provides 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). This would allow the LocConnect server to be easily connected to an existing business process, i.e. localisation could be included as a part of an existing workflow. In the current system, the workflow data is included as an internal file reference in the XLIFF data container (see section 3 of figure 5) which invalidates the XLIFF file due to the use of XML elements inside the internal file reference. In future versions, this problem can be easily addressed by simply storing the generated workflow as a separate resource file (e.g. using BPEL) and providing the link to the resource file in the XLIFF data container as an external file reference.

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 and powerful security measures such as 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. For example, components must keep checking for new jobs in the LocConnect server and fetch jobs from the server. 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 (e.g. the LocConnect server can push a job whenever a job is available for a component, rather than a component continuously checking the LocConnect server for jobs). The implementation of both 'PUSH' and 'PULL' based architectures would also help to establish the availability of the components prior to assigning a job, and help the LocConnect server to detect component failures. The current architecture lacks this capability of identifying communication failures associated with components. If the LocConnect server could detect communication failures, it could then select substitute components (instead of failed components) to enact a workflow. An architecture similar to internet protocol could be implemented with the help of a Mapper component. For example, whenever the LocConnect server detects a component failure, the data container could be automatically re-routed to another component that can undertake the same task so that the failure of a component will not affect the rest of the workflow.

The current resource datastore is only capable of storing textual data. Therefore, it could be enhanced to store binary data too. This would enable the storing of various file formats including windows executable files, dll files, video files, images etc. Once the resource datastore is improved to store binary data, the original file can be stored in the resource datastore and in XLIFF, and a reference to this resource can be included as an external file reference (see section 1 of figure 5).

In the present architecture, the information about components has to be manually registered with the LocConnect server using its administrator interface. However, the architecture 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. A sample XLIFF data container is introduced in figure 6.

This data container differs from the current data container (see figure 5) in the following aspects:

The new container:

- Does not represent additional metadata (i.e. metadata other than that defined in the XLIFF specification) within the data container itself. Instead, this metadata will be stored in a separate metadata store that can be accessed via corresponding API functions.

- Does not represent workflow metadata as an internal file reference. Instead, the workflow metadata will be stored separately in the resource datastore. A link to this workflow will then be included in the XLIFF data container as an external file reference (see section 2 of figure 6).

- Does not store the original file as an internal file reference. It will also be stored separately in the resource datastore. An external file reference will be included in the XLIFF file as shown in section 1 of figure 6.

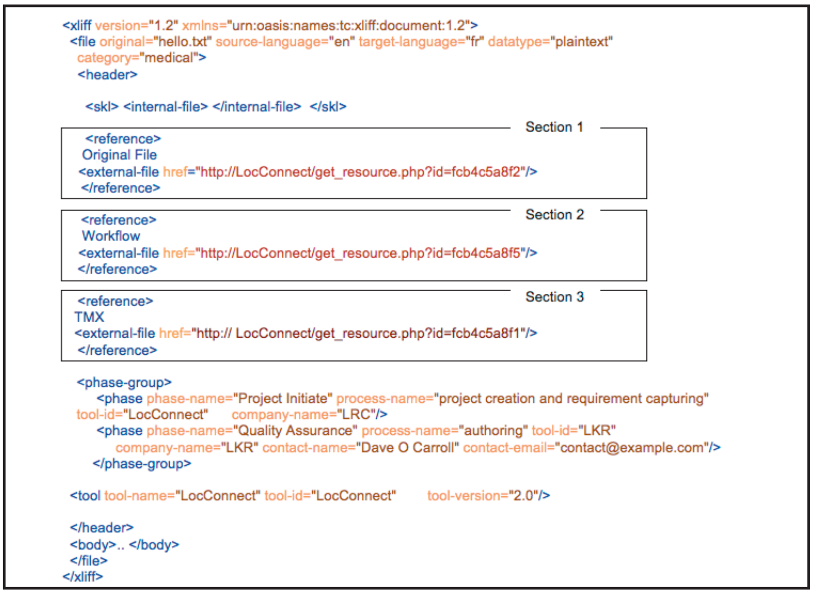
The new data container does not use any extensions to store additional metadata or data, nor does it use XML syntax within internal-file elements. Thus, the above architecture would provide a fully XLIFF compliant (i.e. XLIFF strict schema compatible) interoperability architecture. Due to the separation of the original file content, workflow information and metadata from the XLIFF data container, the container itself becomes lightweight and easy to manipulate. The development of a file format converter component based on this data container would also be uncomplicated.

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 development of the above prototype has mostly focused on addressing the syntactic interoperability issues in localisation processes. The future work will mainly focus on addressing the semantic interoperability issues of localisation processes by improving the proposed system. The LocConnect framework will serve as a platform for future research on interoperability issues in localisation.



```
<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>
   Original File
  <external-file href="http://LocConnect/get_resource.php?id=fcb4c5a8f2"/>
   </reference>
                                                            Section 2
   <reference>
   Workflow
  <external-file href="http://LocConnect/get_resource.php?id=fcb4c5a8f5"/>
  </reference>
                                                            Section 3
  <reference>
 TMX
 <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 6. Improved Data Container

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 the Localisation Research Centre (Department of Computer Science and Information Systems), 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-Vazquez, 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' [online], available: http://www.localisation.ie/xliff/resources/presentations/xliff_symposium_micahbly_20100922_clean.pdf [accessed 09 Jun 2011].

Corrigan, J. & Foster, T. (2003) 'XLIFF: An Aid to Localization' [online], available: http://developers.sun.com/dev/gadc/technicalpublications/articles/xliff.html [accessed 22 Jun 2009].

Fielding, R. (2000) 'Architectural Styles and the Design of Network-based Software Architectures' [PhD], University of California, Irvine.

Halle, S., Bultan, T., Hughes, G., Alkhalaf, M. and Villemaire, R. (2010) 'Runtime Verification of Web Service Interface Contracts', Computer, 43(3), 59-66.

Hayes, J. G., Peyrovian, E., Sarin, S., Schmidt, M. T., Swenson, K. D. and Weber, R. (2000) 'Workflow interoperability standards for the Internet', Internet Computing, IEEE, 4(3), 37-45.

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

Kindrick, J. D., Sauter, J. A. and Matthews, R. S. (1996) 'Improving conformance and interoperability testing', StandardView, 4(1), 61-68.

Lewis, G. A., Morris, E., Simanta, S. and Wrage, L. (2008) 'Why Standards Are Not Enough to Guarantee End-to-End Interoperability', in Proceedings of the Seventh International Conference on Composition-Based Software Systems (ICCBSS 2008), 1343630: IEEE Computer Society, 164-173.

Savourel, Y. (2007) 'CAT tools and standards: a brief summary', MultiLingual, September 2007, 37.

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.

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

XLIFF Technical Committee. (2008). 'XLIFF 1.2 Specification' [online]. http://docs.oasis-open.org/xliff/xliff-core/xliff-core.html [accessed 25 Jun 2009].

XLIFF Technical Committee. (2010) 'XLIFF2.0 / Feature Tracking' [online], http://wiki.oasis-open.org/xliff/XLIFF2.0/FeatureTracking, [accessed 23 Jul 2009].