Architecture for Collaborative Translation Synchronization

Louis-Philippe Huberdeau

February 15, 2008

As part of the Cross Lingual Wiki Engine Project[1], an architecture to track changes across translations had to be created. The purpose of the architecture is to allow concurrent modification of any of the translated pages, track the changes and their propagation. The architecture does not attempt to control the content flow or to restrict it. However, it can be used to provide information about the state of the translation at any point in time.

The architecture is based on the fact that each language has its own dedicated page and that page histories are independent. This architecture was implemented as part of the multilingual support in TikiWiki. The mechanisms rely on human translation of the content. No machine translation is considered.

This document explains the theory behind the architecture as well as the implementation details. Explanations are built around a sample case. In explaining the implementation, many ways to express the theory are used.

1 Use case

In order to illustrate the theory, an example will be used through the entire document. The example proposes a single page translated in three different languages: English, French and Spanish. The actual implementation can work with an unlimited amount of translations. The limit set in the current use case is applied to reduce the complexity of the example.

A page is first created in English. Two modifications are made to the content. At some point in time, a first translation is made to French and Spanish. At that point, the content of all pages is equivalent. Later on, additional content is added to the French page. Changes made to the French page are then applied in the English version, making them synchronized. Additional content is then added to the Spanish page. Afterward, a Spanish translator updates his version of the page using the English version.

At this point in time, the English and French pages contain equivalent content. The content of the Spanish page is more advanced as it incorporated the changes from the French page through the English page and added content of its own. The system should indicate to both English and French readers that improved content exists in the Spanish translation.

In this example, English was used as a pivot language. All modifications made to pages were first translated to the English version before reaching other translations. This model is driven by social factors. The proposed architecture is not limited to this behavior. The decision should be driven by community culture or reinforced by additional tools. The architecture only keeps track of content.

The architecture does not restrict the order in which content propagation is made. Users are not forced to update their page from all other languages before adding content. Such a limitation would prevent some contributions from being made.

2 Use case as sets

From the use case, the state of the page at any point in time can be represented as a set of content elements. In figure 1, content elements have been given letters. A content element could be content addition,

[1]Project hosted on Wiki-Translation `http://wiki-translation.com`

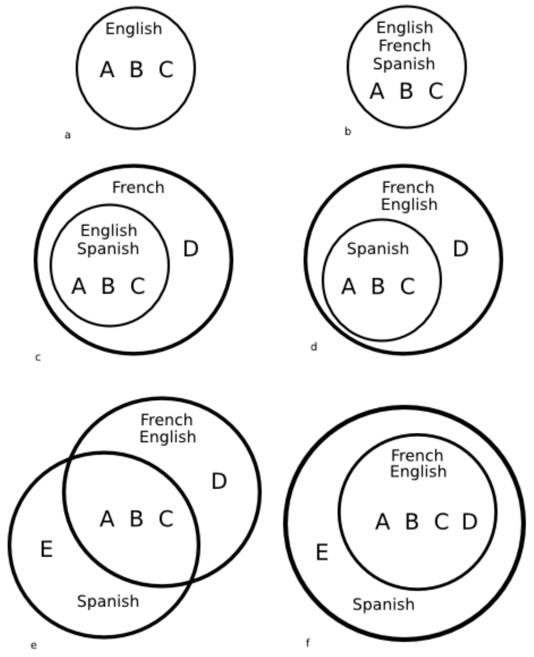

Figure 1: Use case represented as sets

removal or reorganization. It only matters that the change applied to the source page is original content and should be propagated to all translations.

The representation of translations as sets allows to ignore the synchronization points between languages. The current approach places the focus on the content of the pages themselves as modifications made by authors. Trying to identify the point in time when two pages are synchronized would usually require human validation. Instead, this model keeps track of which unique modifications the page has been exposed to. Pages that have been exposed to the same modifications are considered equivalent. Similarly, pages that have been exposed to more modifications can be considered superior. Because the architecture is based on human translation with no additional validation, relying simply on content exposure may not be reliable. Section 6 discusses these issues.

For the sake of simplicity, the graphic begins at the point right before the initial translations are produced. Figure 1a represents this state where only the English version exists and contains the content from its initial creation and the two subsequent edits.

Figure 1b presents the situation where both initial translations exist. At that point in time, the content of all three translations can be considered synchronized as the content was translated from English. All changes made to the English page in the past are represented in the translations.

When content is added to the French page, the content available in the other translations becomes only a subset of the French translation. As seen in figure 1c, English and Spanish remain untouched and are still equivalent, but an additional element was added to the French set.

In figure 1d, the English version incorporates the changes made to the French version. After this step, French and English become equivalent again. Only the Spanish version remains without the latest changes.

Afterward, new content is inserted in the the Spanish version without any attention taken to the other translations. A strong division in the content of the Spanish version in relation with the equivalent English and French can be observed in figure 1e.

In the final step, the Spanish version incorporated the changes from the English. As English previously incorporated the changes from the French translation, the Spanish version now contains all changes. As it can be seen in figure 1f, Spanish is now a superset. The English and French versions remain equivalents.

To make all translations equivalent, both English and French would need to update their content.

Consider α to be the source page and τ the translation target. Ω is the set of translations for a given page. The translation set is a set of related pages. θ is used to represent any page in the translation set. $C(\theta)$ is the set of content elements of page θ. Φ is the set of all content elements available in the translation set.

The objective of content synchronization is to reach a point where $C(\theta) = \Phi$ to make the page fully up to date.

In trying to reach this objective, content must be tracked during its evolution. Content tracking can be reduced to two distinct operations. The addition of new content can be seen as the inclusion of a new element in the set as seen in formula 4 and 5, in which δ is a new content element. The other operation is content propagation and is used when performing a translation from source to target as seen in formula 6. For a new translation, τ is initially defined as an empty set. Formulas 1 to 3 define validity constraints for the model.

$$\alpha, \tau, \theta \in \Omega \tag{1}$$

$$\forall \sigma \in \Omega \, (C(\sigma) \subseteq \Phi) \tag{2}$$

$$\forall \gamma \in \Phi \exists \sigma \in \Omega \, (\gamma \in C(\sigma)) \tag{3}$$

$$C(\alpha)' \quad \leftarrow \quad C(\alpha) \cup \{\delta\} \tag{4}$$

$$\Phi' \leftarrow \Phi \cup \{\delta\} \tag{5}$$

$$C(\tau)' \leftarrow C(\tau) \cup C(\alpha) \tag{6}$$

The representation of the content as a set allows to quickly observe where additional content can be found in other translations. From a snapshot of the content of the pages in their current version, it is possible to determine if translations need updates. Formula 7 presents the condition to validate to determine if an update is required from source to target. A target page needs update from the source page if the source page contains a content element not present in the target.

Formula 8 indicates if a page θ requires an update without targeting a specific source. The $NU(\theta)$ predicate indicates that θ *needs an update*. $NUF(\alpha, \tau)$ indicates that τ *needs an update from* α.

A page needs an update if if does not contain all the content elements available in the translation set.

$$\forall \alpha, \tau \in \Omega \, (\exists \sigma \in C(\alpha) \, (\sigma \notin C(\tau)) \Rightarrow NUF(\alpha, \tau)) \tag{7}$$

$$\forall \theta \in \Omega \, (C(\theta) \subset \Phi \Rightarrow NU(\theta)) \tag{8}$$

An other interesting representation of this concept is to see each change performed as a binary flag on the translation page. Each binary flag is a bit that must be translated to other languages for content to be synchronized. Table 1 presents the evolution of the use case using this representation.

Using such a representation, very simple bitwise operations can be performed to determine if a page is up to date. A bitwise *or* can be used to merge the translation bits during the propagation and an *exclusive or* can be used to highlight the missing bits.

3 Representation as a graph

The representation of the content as elements of a set provides a good way to observe the relationships between the languages and it allows to perform multiple common operations. However, as the view is based on snapshots, it provides very little information about the evolution of the content. Such an evolution is better represented as a graph.

As presented in figure 2, the use case above can be represented as a directional graph. More information than necessary was incorporated in the illustration to link to the story more closely and connect to the set representation.

In the graph, page evolutions are represented by solid edges. Page translations are represented by dashed edges. Each node represents a page version.

In the nodes, the page language and page version is indicated on the first row. The second row contains the content elements incorporated in the version. The content element letters match the ones used in the set representation.

The graph contains two types of nodes. The white nodes are page versions created from a page edit. With them, a new content element is added. The gray nodes are page versions created from a translation effort. They only incorporate content elements from their source version.

The illustrated graph can be defined like this:

$$\begin{aligned} V \quad = \quad \{ & \\ & en.1, en.2, en.3, en.4, \end{aligned}$$

Table 1: Evolution of the use case represented using translation bits

Step	English	French	Spanish
Page created in English	1	0	0
Modification made to the content	11	00	00
Modification made to the content	111	000	000
Original translation to French and Spanish	111	111	111
Modification made to the French translation	1110	1111	1110
Incorporation of changes from French to English	1111	1111	1110
Modification made to the Spanish translation	11110	11110	11101
Incorporation of changes from English to Spanish	11110	11110	11111

$$
\begin{aligned}
& fr.1, fr.2, \\
& es.1, es.2, es.3\} \\
E \quad = \quad & \{ \\
& (en.1, en.2), \\
& (en.2, en.3), \\
& (en.3, en.4), \\
& (fr.1, fr.2), \\
& (es.1, es.2), \\
& (es.2, es.3), \\
& (en.3, fr.1), \\
& (en.3, es.1), \\
& (fr.2, en.4), \\
& (en.4, es.3)\}
\end{aligned}
$$

As it can be seen, the graph representation presents the relationships between the different page versions to create paths in which the content elements will propagate. Content elements reach every node after which they have been introduced. The graph itself can be used to rebuild the set representation. Content elements get created when the vertex is not directly reached by an edge from an other translation.

Figure 2 presented a unified graph of the historical information and the translation paths. Smaller graphs can be represented for each content element by ignoring the version information as presented in table 2. These graphs can be combined into a weighted graph to identify the most common translation paths. In the same way, missing translation paths or paths that could use more frequent updates could be identified.

Table 2: Translation bit paths

Translation bit	V	E
A	$\{en, es, fr\}$	$\{(en, fr), (en, es)\}$
B	$\{en, es, fr\}$	$\{(en, fr), (en, es)\}$
C	$\{en, es, fr\}$	$\{(en, fr), (en, es)\}$
D	$\{en, es, fr\}$	$\{(fr, en), (en, es)\}$
E	$\{es\}$	$\{\}$

The graph representation highlights the paths in which the content flows. Sections where the content evolves independently and junction nodes where the content is being incorporated can be clearly seen. The time-line view allows to understand the dynamics around the translation process. Over a long period of time, the information gathered by the graph models could help identify the language clusters and find bottlenecks in the translation process. Such information will allow to orient future developments and orient the community efforts. On a weighted graph with weak links eliminated, the structure of the translation community could be observed.

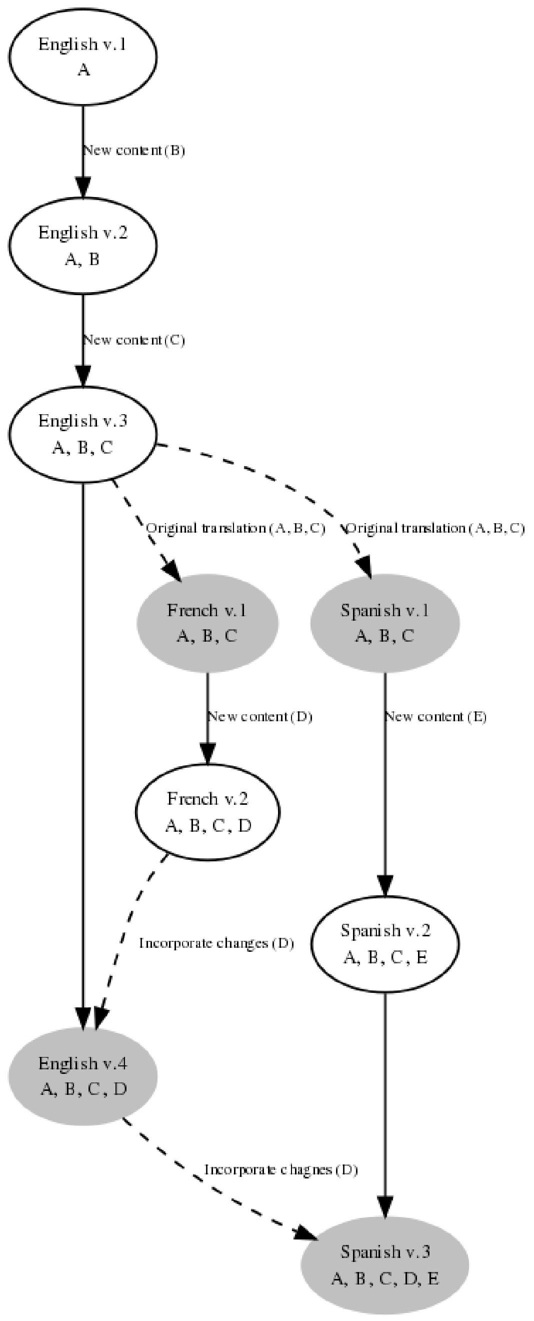

Figure 2: Use case represented as a graph

Unlike the set representation, the translation from which the content was propagated is known. The graph representation provides an audit trail that could be required in multiple usage scenarios. It could also be used to perform simple verifications of the quality of the translation.

Because the architecture is based on human translation and it inherently trusts translators to apply the modifications correctly, degradation may occur over time. Using the example with three languages, the longest path from one language to the other is of two translations. It is probably safe to believe that the content of D is very similar to the same content element in Spanish, especially since all languages are Latin-based. In a larger context with 18 concurrent languages, the content propagation paths between two languages could be much longer and result in significant changes. Long paths could be used to signal the need for review.

Alternatively, the translation community could structure the translation paths following a pivot or multi-pivot model to reduce the length of the translation paths.

4 Implementation

Because of the established test sites by the Cross Lingual Wiki Engine Project, the implementation had to be made as a part of TikiWiki. In the Open Source application, all the content is stored in an SQL database. A table was already available to define the translation sets. Only the actual tracking information had to be added.

Because all the content is already available in SQL, the additional information stored for tracking also had to be stored in the same way. Because understanding collaborative translation is an important part of the Cross Lingual Wiki Engine Project, the graph information had to be preserved. Simply storing the set information was not sufficient. However, because fast response times are required, only keeping the graph information was also not an option. Rebuilding the graph and running algorithms on every page load is simply not possible. Graphs will tend to grow in an unbounded way over time and decreasing performance would be observed.

The selected implementation stores the information as a graph in a table and uses a cache field in the table to reference to the first node of the graph for each translation bit. This strategy allows to keep all the graph information to study collaborative translation patterns and to execute the fast set operations required for common information display requirements. Information to relate to the page history and an additional field for extensions was added. Table 3 presents the table structure.

Table 3: Table structure

Field	Description
translation_bit_id	The primary key of the table. One entry for each bit created or propagated is created.
page_id	Foreign key to the page.
version	The version at which the translation bit appeared on the page.
source_translation_bit	The translation bit from which the page was propagated. This field allows to build directional graphs.
original_translation_bit	The cache translation bit pointing to the first translation bit created. This value could also be obtained by a recursive fetch on the source translation until a NULL value is found.
flags	Allows to set special flags on the translation bits to describe their purpose or to categorize them.

The table does not contain any direct input from the user other than the "flags" field. The data collection is fully automated and driven by the workflows in the user interface. When a normal edit is performed on a page, a translation bit is created. No translation bits are created for minor edits as those are only used for quick correction of typos or formatting mistakes.

Information extracted from the table itself allows to present links that lead to a translation interface. The translation interface presents the differences made to the page since the last update. Once the changes are saved, all translation bits from the source page until the translated version are propagated to the target page. The translation bits are completely transparent to the users as they only need to follow the natural workflow of translation and the system updates it's relationships correctly.

In the future, support for incomplete translation will need to be supported. In many cases, translators may choose to translate large texts in multiple edits, which may or may not be in the same session. An incomplete translation would not propagate any translation bits. Instead, all bits would be transferred during the final, complete, translation. It is not possible to manually select which bits have been translated as part of an edit as the only way to do so would be to refer to the original change in the original language. The translator may not understand the original language.

However, the "flags" field requires special user attention. Currently, the only flag supported by the implementation is the "critical" flag. It allows to mark a change as very important. Once the flag is set, the other translations display a warning indicating that the content of the page they are looking at is inaccurate and that an update is required. When the target translation is updated, the warning message disappears automatically.

This was a special requirement to handle cases where a page's content may lead to significant errors. In order to mark an edit as critical, the user must request it manually during the page edit.

A generic "flags" field was added rather than a specific field for this situation. Additional flags could be added to indicate lower priority changes or to add change classification information. Because the field is generic, flexible algorithms can be developed to suit unexpected user needs.

When a content element is created as part of a normal edit, both “source_translation_bit” and “original_translation_bit” are set to NULL. This allows to quickly identify the distinct translation bits available.

A single propagation operation can create multiple translation bits on the target page. During the propagation, “translation_bit_id”, “original_translation_bit” and “flags” from the source page that are not already included in the target version are collected. Translation bits are then added to the target page by specifying the “source_translation_bit” as the “translation_bit_id” matching the missing translation bit. The original translation bit is copied directly from the “original_translation_bit” of the source unless the source’s original translation bit is NULL, in which case “translation_bit_id” is used instead. The “flags” field is copied directly without modification.

Using these rules, table 4 presents the resulting data of the use case presented earlier. In the data set, “page_id” 1 is English, 2 is French and 3 is Spanish. To human readers, this content is hard to understand. However, it contains all the relationships required to rebuild the graph or the sets. Multiple queries can be used on the table to obtain useful information to present to the end user. Algorithm 1 is a query that can be used to generate a list of better pages. By this algorithm, a better page is a page containing translation bits not present in the current page. The query presented here is a simplified version from the one used in the real application. The complete query also fetches information about the page languages to adapt the output to user preferences as well as information about the translation history to create translation links.

A similar query to fetch worst pages has been written. A worst page is a page missing some of the translation bits contained in the current page. The list of worst pages is used to display the list of pages that could be improved based on the currently viewed content.

A query to fetch the missing translation bits on a page is also available. The query has an additional “flag” parameter to reduce the list to translation bits containing the given flag. With this parameter, the query is used to obtain the list of critical updates that must be applied. A separate query can then get the list of available sources for each translation bit. Together, can can indicate that the current page is inaccurate and provide a list of alternative translations to view.

Table 4: Table content using the sample use case

translation_bit_id	page_id	version	source_translation_bit	original_translation_bit
1	1	1		
2	1	2		
3	1	3		
4	2	1	1	1
5	2	1	2	2
6	2	1	3	3
7	3	1	1	1
8	3	1	2	2
9	3	1	3	3
10	2	2		
11	1	4	10	10
12	3	2		
13	3	3	11	10

Searching the table for the correct versions to perform the content differences on in order to present the content to the translators is a complex operation. It can be performed as a sub-query in most page listings as long as the source and target pages are known. The intuitive way to do so is to search for the last synchronization point between two pages. This tends to bring long text differences as target may have been

Algorithm 1 Obtain the list of pages with more translation bits

```
SELECT DISTINCT
 page.pageName page
FROM
 tiki_translated_objects a
 INNER JOIN tiki_translated_objects b
  ON a.traId = b.traId AND a.objId <> b.objId
 INNER JOIN tiki_pages page
  ON page.page_id = a.objId
 INNER JOIN tiki_pages_translation_bits candidate
  ON candidate.page_id = page.page_id
WHERE
 b.objId = ?
 AND IFNULL(
  candidate.original_translation_bit,
  candidate.translation_bit_id ) NOT IN(
   SELECT
    IFNULL(
     original_translation_bit,
     translation_bit_id )
   FROM tiki_pages_translation_bits
   WHERE page_id = b.objId
 )
```

used to update the source in between. Attempts to search for the last update point in both directions can be made to improve the situation. However, this may lead to content loss, even in a structured model like pivot.

Algorithm 2 presents a query that can extract the appropriate page version to perform the content diff from based solely on the presence of translation bits. The appropriate version is the version right before the lowest version in which a translation bit exists in the source but does not yet exist in the target. If no such version are found, meaning a complete translation was never performed, version 1 will be returned as forced by the default value. In some cases, this strategy may show some content that was already propagated to the target through other paths, but all changes required to propagate the translation bits will be present.

Using the contained translation bits as the only reference in the history look-up also has the advantage of having the content diff behaving in a much more logical way in environments with a large number of languages and irregular translation paths than searching for synchronization points between source and target. No matter where the propagation of the translation bits came from, the content diff will be based on the new translation bits added.

Algorithm 2 Obtain the source version of a content diff

```
SELECT
 IFNULL(
  IF(MIN(version) = 1, 2, MIN(version)),
  2 ) - 1
FROM tiki_pages_translation_bits
WHERE
 page_id = :source
 AND IFNULL( original_translation_bit, translation_bit_id )
  NOT IN(
   SELECT
    IFNULL( original_translation_bit, translation_bit_id )
   FROM tiki_pages_translation_bits
   WHERE page_id = :target
  )
```

5 Usage in a wiki with a staging phase

Some websites using a wiki engine to edit content add an additional staging phase to make sure the content presented to the end user is valid and accurate. In TikiWiki, the staging feature works by maintaining two separate set of pages. One of the set is hidden from non-editors and used to perform the edits on. The second set is the set pages containing the approved content. Only the approved set is presented to the non-editors. When the staging version of a page is approved, the revisions made to the staging page are copied over to the approved page. The approved version becomes a verbatim copy of the staging version, including all history details.

This process causes problems in the way the translation bits are created and propagated. Since edits and translations are only performed on the staging pages, the approved versions do not contain any translation tracking information and cannot provide valuable content to the end users.

Because the approved page is a verbatim copy of the staging version until a certain point in time, the translation bits from the staging version can be used to provide the information. However, the queries made to the table must be limited to the versions available the approved page. More content elements may be available in the staging pages and those will not be accessible to the non-editors.

With added complexity in the fetch queries, the copy of translation bits and trouble related to correct handling of the propagation can be avoided.

6 Handling subjectivity in translation

Human translation requires interpretation of the content. In most cases, translators will attempt to match the source content as closely as possible. However, in some cases, information may be altered in the process. For cultural reasons, some aspects may be left out of a given translation or different interpretations may be preferred. While this does not make much sense in a documentation effort, when dealing with political or religious aspects, it may be more frequent.

Deciding if such an interpretation of facts is desired during the translation effort is a community decision. It was established early on by the Wiki-Translation community that such interpretations should not be blocked in all situations. Instead, the tools should encourage synergy between translations and let them evolve independently.

Keeping the ethical aspects out of the question, there are other situations in which it may be desirable for content to be different between translations. An example of this is in a worldwide marketing firm. The business cases developed for the French market and the US market are very likely not to be the same. Some companies have high prestige in some areas of the world and are unknown in others. An attempt to perfectly align the translations would only create documentation poorly adapted to its market.

Interpretation during translation is perfectly valid. However, the current architecture is vulnerable to the telephone game[2]. From the moment a translation is made, the newly translated page becomes a possible source for further translations. There are no ways for translators to know if the source they are looking at is equivalent to the original change. Over long chains of translations the content could become entirely different. The problem could be avoided by not making the translation, but the page would then be considered permanently out of date.

[2]The telephone game is also known as Chinese whispers `http://en.wikipedia.org/wiki/Chinese_whispers`.

This could be solved by letting the engine know about subjective translations. When a translator makes a different interpretation of the facts, he could mark it as such. In this case, the bits propagated would be marked as subjective. This special flag on the bits would indicate that, while the translation is up to date, it may not be used as a source for those particular bits. Moreover, an additional content element should be created as part of the operation. As the translation is in fact adding new content to the article, other translations could also benefit from the interpretation made.

Making strong interpretations in the content will break the alignment of the translations. From the moment one interpretation is made, it is very likely that all subsequent translations will also be subjective. However, this technical reality is only a reflection of the nature of the content.

The mechanism still relies on trusting the translator to perform the job correctly and indicating deviations when they occur. In order to avoid spreading mistakes, it should be possible for a moderator to mark a translation as subjective after the fact. For that matter, a moderator should be able to cancel a translation in the case it was not performed correctly or not at all. Wiki pages allow to undo changes. The primary reason they work well in collaboration efforts is that the cost of correcting errors is very low. The translation architecture should propose functionality to bring this simplicity to handling of translation synchronization.

For any situation where the quality of the content and its translations is important, a review process should be in place. In a wiki, the review process is done naturally by the following visitors. However, in collaborative translation, a normal visitor will not get an overview of the translation situation as he does on a single page. Additional mechanisms for review will have to be used. These could be ad-hoc by having moderators watching translation sets or supported by tools.

7 Conclusion

The architecture presented above meets all the project's requirements in terms of data collection and also allows to present interesting information to the end users. The integration in the wiki itself will need improvements, but the architecture provides a good foundation for change tracking in collaborative translation.

Many special cases will require customization of the architecture usage. Page staging and subjective translation are two examples. The architecture was not created to be a governing solution to translation problems. It focuses on tracking changes between languages to simplify their synchronization. The architecture is only a part of the Cross Lingual Wiki Engine Project, which aims to solve those broader issues.

By collecting data, the architecture should help us understand what it means to translate collaboratively and find out what the real issues to solve are.