

---

# Bringing Procedural Knowledge to XLIFF

Prof. Dr. Klemens Waldhör  
TAUS Labs & FOM University of Applied  
Science

FEISGILTT  
16 October 2012  
Seattle, USA



# Presentation Overview

---



- ✓ Why?
- ✓ How?
- ✓ Examples
- ✓ Summary and advantages



# Beyond XLIFF 2.0

---

- ✓ Syntax
- ✓ Semantics
- ✓ Pragmatics



# XLIFF Limitations

---

- **XLIFF represents and communicates document related information**
  - Text
  - Attributes, properties
- **Semantic aspects are only partially supported**
  - Constraining values with XSD
  - XLIFF 1.x implementer specific interpretation
  - XLIFF 2.0 with modules beyond core
- **Complex tasks hard or impossible to realize**
  - Work flow aspects
    - Linport
  - Requires representing procedural knowledge



# The traditional approach

The screenshot displays a TMS interface with a multi-column table of translation segments. The table columns include 'Ausgangstext' (Source Text), 'Zieltext' (Target Text), 'Status', 'QM' (Quality), 'Matchrate', 'Autostatus', and 'letzter Bearbeiter' (Last Editor). The segments contain technical text about Apache configuration and multithreading. A context menu is open over the table, showing options like 'Aufsteigend sortieren', 'Absteigend sortieren', 'Spalten', and 'Filter'. On the right, the 'Segment-Metadaten' panel shows 'QM' (Quality) settings, 'Status' (Status 1, 2, 3), and 'Terminologie' (Terminology) with a 'recommended setup' and 'prefork' term.

The background window shows the 'beoLogisch Editor 0.4.0' displaying an XML document. The XML content includes a title 'Montage-, Einbau- und Betriebsanleitung' and a list of sections (sn 0-11) such as 'Radblocksystem DRS 250, 315, 8.800', 'Vorwort', and 'Weitere Unterlagen'. The XML uses tags like <sn>, <tn>, and <ph> to structure the document.

# Imagine...

---

- ▶ Customer wants to enhance existing XLIFF Editor or application with his own small “XLIFF apps”
    - ▶ Cost computations
    - ▶ Own similarity measures
    - ▶ ...
  - ▶ Translation service provider wants to add some additional operations through “XLIFF apps” as part of the XLIFF file
    - ▶ Constraint checks
    - ▶ Synchronizing TMs through web
    - ▶ Add a new MT system
  - ▶ ▶ ...
-

# Current Limitations...

---

- ▶ XLIFF does not support this
- ▶ Translation GUIs and application using XLIFF supply a basic functionality
  - ▶ Not extendible
  - ▶ Or proprietary approach for extension
- ▶ No common agreed way how running “apps” in translation applications



# Extending XLIFF

---

- **Use HTML approach**
  - From passive presentation to active content
  - Proven approach
  - JavaScript – ECMA Script - standard
- **Add Script support to XLIFF**
  - `<script...>` tag represents semantic knowledge
  - Invoked on element basis
- **Cross application functionality**
  - Vendor independent
  - Interoperability supported





# Advantages

---

- **XLIFFs cross platform goals supported**
  - Mappers between vendor specific XLIFF implementations
- **User is not limited to the functionality supplied by the application vendor**
  - Can extend functionality of editors
    - E.g. specialized word counting
    - Matching quality computation
    - Access external data sources like MT, term databases etc. using Web Service
  - May add his own work flow features



# Components - Questions

---

- ▶ **Functions and Procedures**

- ▶ How and where to define functions and procedures in XLIFF

- ▶ **Parameter Passing**

- ▶ How to handle parameter passing?

- ▶ **Return Value Passing**

- ▶ How to return values back to the application?

- ▶ **GUI Elements**

- ▶ Which GUI elements should be supported?

- ▶ **Security**

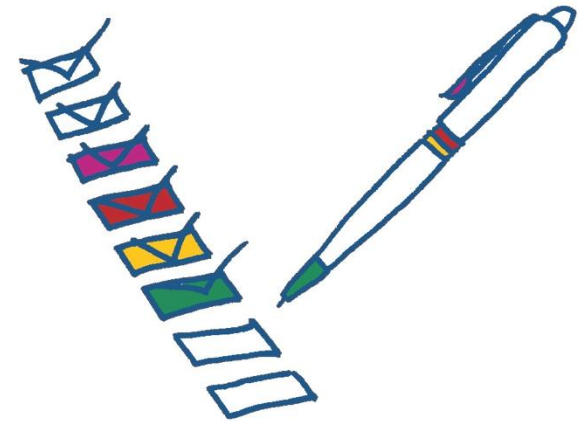
- ▶ How to avoid security leaks?



# Functions, Procedures

---

- Following a programming language approach
  - Like HTML
  - Include scripting code in XLIFF elements
    - Script tag
    - Properties
- Scripting language
  - ECMA Script (“Java Script”)
    - Wide spread
    - Easy to implement
    - Security concept
  - Other languages like PHP, Perl?
  - Access available in languages like Java etc.



# Types, Parameters, Return Values

---

## ➤ Types

- JSON

## ➤ Parameters

- Automatic passing of parameters
- JSON based representation of element as a parameter for script
- Additional parameters for script
  - Languages, file name, other elements...

## ➤ Return Values

- JSON supports complex objects
- Most modern programming languages support JSON



# Script Integration

---

## ➤ Using the HTML based approach

### ➤ Script tag

```
<script type="text/javascript"> ... some JavaScript code </script>  
<script src="javascript.js" type="text/javascript"></script>
```

## ➤ XLIFF

### ➤ Add to XLIFF Header

### ➤ Zero or more scripts allowed in header

```
<?xml version="1.0" encoding="UTF-8"?>  
<xliff version="1.0">  
<file datatype="XML" original="..." source-language="de" xml:space="default" target-  
language="en">  
<header>  
  <script type="text/javascript">  
    ... some JavaScript code  
  </script>  
  <script src="javascript.js" type="text/javascript"></script>  
  ...  
</header>  
  ...
```

# Script Invocation

---

## ➤ Differences to HTML

- Script tag only in header, not elsewhere in the document
- All functions etc. declared in header

## ➤ Script invocation in XLIFF as some kind of trigger

### ➤ Attribute in element

#### ➤ on-enter

- Run script when the element is entered by the application

#### ➤ on-exit

- Run script when the element is left by the application

### ➤ Additional triggers possible

- E.g. some kind of trigger which reacts on changes to the target element

### ➤ Implementation of trigger invocation up to application

- Editor, command line tools, ...



# Informal Script Invocation Logics

---

- **Application enters an element**
  - Check for on-enter attribute
    - if exists,
      - execute script
      - replace trans-unit with return JSON string converted to xml
- **Application leaves a element**
  - Check for on-exit attribute
    - if exists
      - execute script
      - replace trans-unit with return JSON string converted to xml
  - on-exit is applied to modified trans-unit of on-enter



# Default handling in Applications

---

## ➤ Problem

- XLIFF files will normally not contain any scripts
- Run scripts when application is used automatically

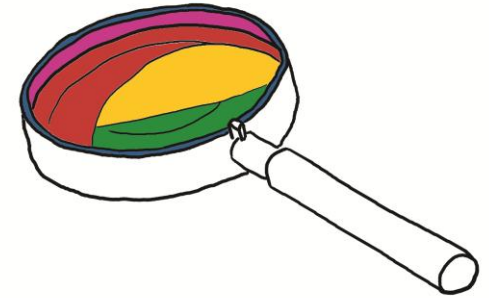
## ➤ Solution

- Support default handling for scripts
- Application tries to load default.js
  - Contains user written scripts
  - Read from directory of XLIFF file
- For each XLIFF element apply default functions
  - “on-enter-`<element-name>`” and “on-exit-`<element-name>`”
  - Example: on-enter-trans-unit
  - Function naming: Xpath?





# Script Example



## ➤ Main usage in trans-unit element

```
<trans-unit
  approved="no" id="3"
  translate="yes"
  on-enter="estimateTranslationTime"
  on-exit="validateTranslationTime"
>
  <source xml:lang="de"><g />haus</source>
  <target xml:lang="en"><g>house</g></target>
</trans-unit>
```

Call a JavaScript function **estimateTranslationTime** with trans-unit as JSON object when trans-unit entered

Call a JavaScript function **validate** with trans-unit as JSON object when trans-unit element is left

```
{
  "trans-unit": {
    "-approved": "no",
    "-id": "3",
    "-translate": "yes",
    "-on-enter": "estimateTranslationTime",
    "-on-exit": " validateTranslationTime ",
    "source": {
      "-xml:lang": "de",
      "#text": "haus"
    },
    "target": {
      "-xml:lang": "en",
      "g": "house"
    }
  }
}
```

# Script Example Return Value

```
<trans-unit
  approved="no" id="3"
  translate="yes"
  on-enter="estimateTranslationTime"
  on-exit="validateTranslationTime"
>
  <source xml:lang="de"><g />haus</source>
  <target xml:lang="en"><g>house</g></target>
</trans-unit>
```

## Application modifies trans-unit (not required)

```
<trans-unit
  approved="no" id="3"
  translate="yes"
  on-enter="estimateTranslationTime"
  on-exit="validateTranslationTime"
>
  <source xml:lang="de"><g />haus</source>
  <target xml:lang="en"><g>house</g></target>
  <prop-group>
    <prop prop-type="estimatedTime">00:00:10</prop>
    <prop prop-type="neededTime">00:00:12</prop>
  </prop-group>
</trans-unit>
```

## JSON Return Value for validate

```
{
  "trans-unit": {
    "-approved": "no",
    "-id": "3",
    "-translate": "yes",
    "-on-enter": "estimateTranslationTime",
    "-on-exit": "validateTranslationTime",
    "source": {
      "-xml:lang": "de",
      "#text": "haus"
    },
    "target": {
      "-xml:lang": "en",
      "g": "house"
    },
    "prop-group": {
      "prop": [
        {
          "-prop-type": "estimatedTime",
          "#text": "00:00:10"
        },
        {
          "-prop-type": "neededTime",
          "#text": "00:00:12"
        }
      ]
    }
  }
}
```

# GUI Elements, Forms

---

- **Script might require user interaction**
  - Extension of base application
  - Display of properties etc.
  - Input some data
- **Problem**
  - No GUI elements foreseen in scripting languages
- **Possible solution (Java approach)**
  - Support import of classes and access to classes / instances
  - Definition of some basic GUI element
    - Yes / No / Cancel / ok
    - Simple input dialogue
    - ...



# Example application

---

- Translation Change measure for MT matches
- Prototype
  - MT Translation for a segment
  - Copied to target
  - Similarity based on source segment comparison
  - Modification done by translator
  - In order to compute a fair price for the target changes compare translators changes with MT translation based on Levenshtein



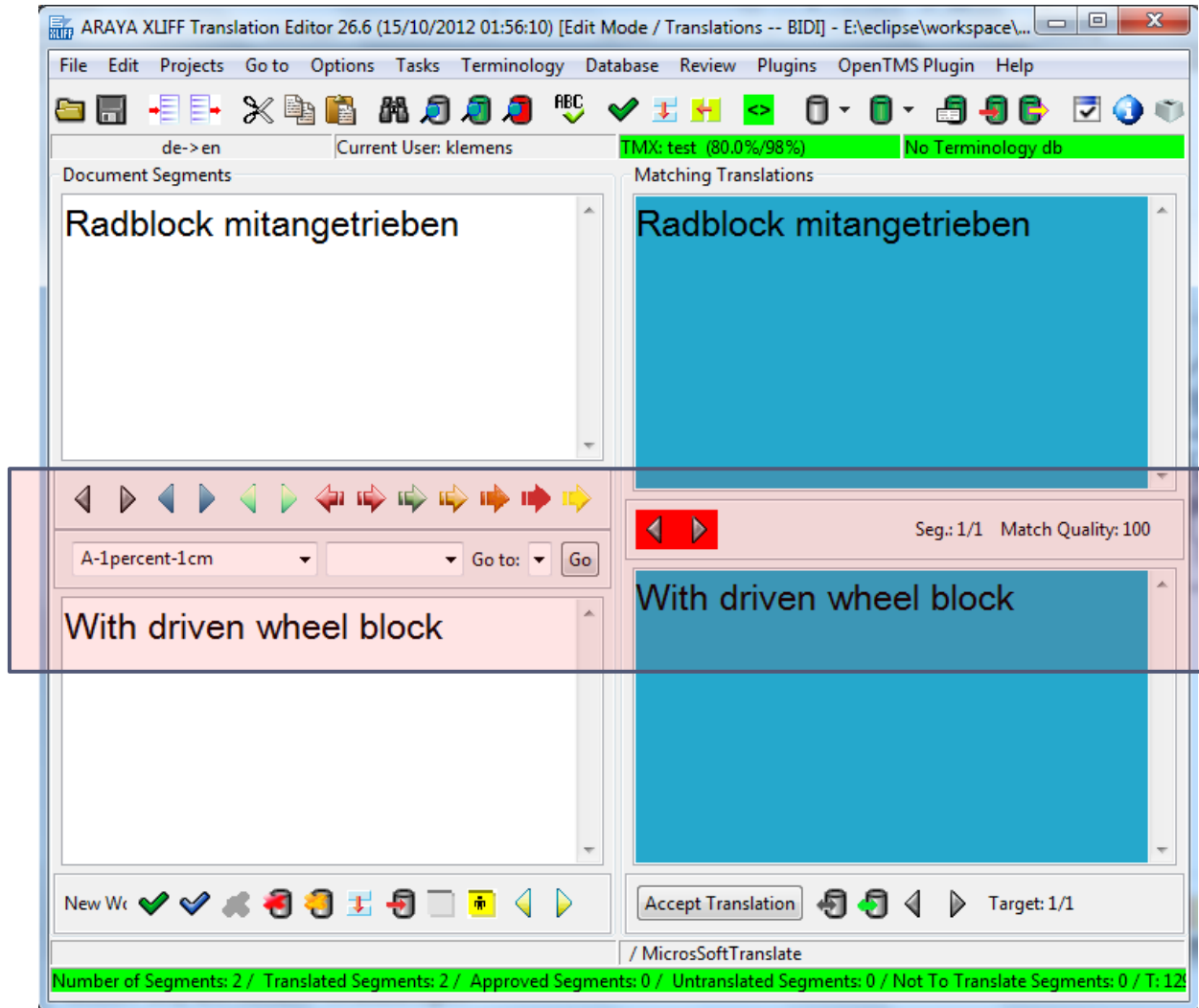
# Example XLIFF File

---

```
<?xml version="1.0" encoding="UTF-8"?>
<xliff version="1.0">
  <file datatype="XML" original="C:\Program Files\OpenTMS\test\tekom2009\sample.odt.content.xml" source-language="de"
xml:space="default" target-language="en">
  <header>
    <script src="file:///E:/eclipse/workspace/openTMS/test/ScriptTest/xml_for_script-
3.1/jsXMLParser/xmlDom.js" />
    <script src="file:///E:/eclipse/workspace/openTMS/test/ScriptTest/testscript.js" />
  </header>
  <body>
....
    <trans-unit approved="no" help-id="0" id="0" reformat="yes" translate="yes"
xml:space="preserve" on-enter="computeLevenOfTransunit(transunit);">
      <source xml:lang="de">Radblock mitangetrieben</source>
      <target xml:lang="en">With driven wheel block</target>
      <alt-trans match-quality="MT" id="054bdc3a-b361-4afc-b296-86ad2e819716"
xml:space="preserve" origin="MicrosoftTranslate">
        <source xml:lang="de">Radblock mitangetrieben</source>
        <target xml:lang="en">With driven wheel block</target></alt-trans>
    </trans-unit>
  </body>
</file>
</xliff>
```



# Example before changes by translator



# Example after changes by translator

ARAYA XLIFF Translation Editor 26.6 (15/10/2012 01:56:10) [Edit Mode / Translations -- BIDI] - E:\eclipse\workspace\...

File Edit Projects Go to Options Tasks Terminology Database Review Plugins OpenTMS Plugin Help

de->en Current User: klemens TMX: test (80.0%/98%) No Terminology db

Document Segments

Radblock mitangetrieben

Matching Translations

Radblock mitangetrieben

Seg.: 2/2 :ent-1cm Go to: Go

Wheel block also driven

With driven wheel block

Seg.: 1/1 Match Quality: 8

Accept Translation Target: 1/1

/ MicrosoftTranslate

Number of Segments: 2 / Translated Segments: 2 / Approved Segments: 0 / Untranslated Segments: 0 / Not To Translate Segments: 0 / T: 12

MT translation changed by translator  
Recomputed similarity based on comparing  
the MT alt-trans target with the new target

# Summary and Chances

---

- **Easy integration into existing applications**
- **Freedom for the user**
  - Make XLIFF based applications more versatile and powerful
- **XliffApp store**
  - Market for application/vendor independent modules
  - Additional revenue for vendor independent developers and translators
- **Next step**
  - Building a demonstration prototype
    - e.g. for openTMS





---

For more information, please contact:  
[klemens@translationautomation.com](mailto:klemens@translationautomation.com)

