Intel's Localization BUS Initiative
To XLIFF or not to XLIFF

Loïc Dufresne de Virel
Localization Strategist

Intel Information Technology

DISCLAIMERS

I’m presenting here what we are doing

Still a work in progress

I don’t pretend we have solved all our problems… and I don’t pretend I have the answers to all your questions ;-)

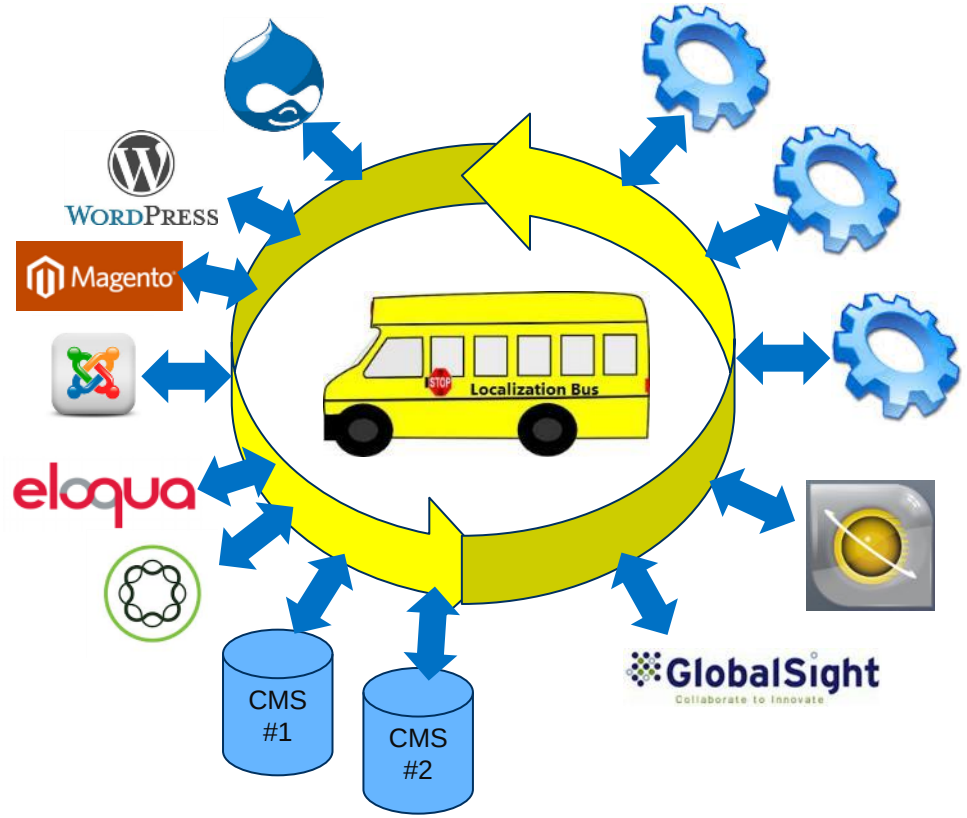
Some of our solutions are specific to our environment, but clearly we’re not alone in this journey

A mention of a 3rd party provider doesn’t constitute an endorsement from Intel

Where it all started…

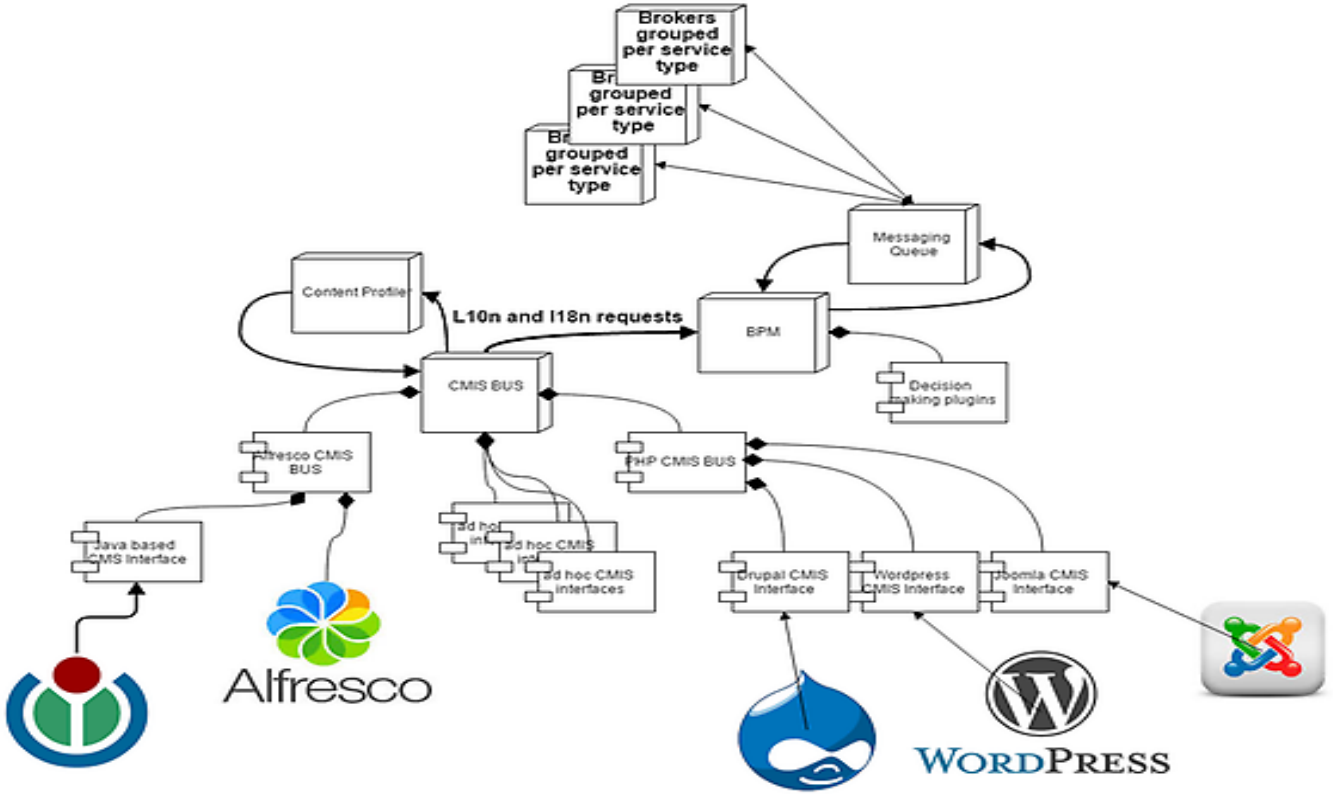
- Proliferation of point-to-point solutions
  - No input from localization team, purely reactive
- Duplication of efforts
- Lack of scalability and consistency
- Confusion among users (PM, translator, reviewer, etc.) and poor UX
- A maintenance nightmare

Our initial “vision”



- Localization Enterprise Bus
  - Connect many CMS and source control systems to many Localization tools
- Automate and Integrate for TTM and ROI
  - Set up recurring tasks to automatically detect, collect, and forward changed content to translation tools.
  - Web Portal + Documented API
  - Software Factory Integration
- Stakeholders / Beneficiaries:
  - All business units, regardless of their choice of content management system

An Early Rendering (Courtesy of CNGL ADAPT)



It’s All About Standards

CMIS 1.1

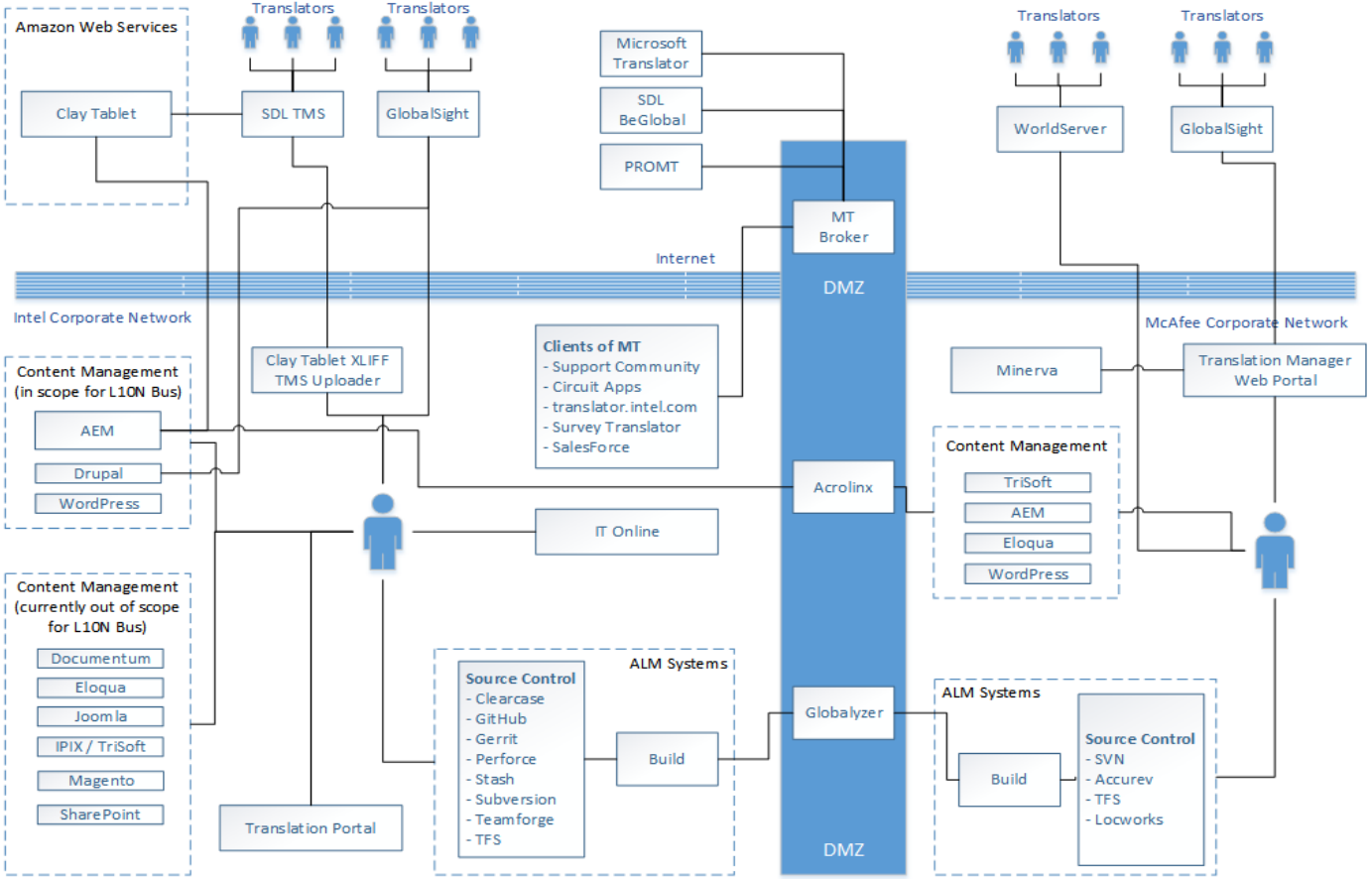
- Content Management Interoperability Services
- **CMIS servers and clients** for **compliance** and **abstraction**
- **+ ITS 2.0** on the CMS side

ITS 2.0 serves as the **bridge between** [monolingual] **content** and **bitext**

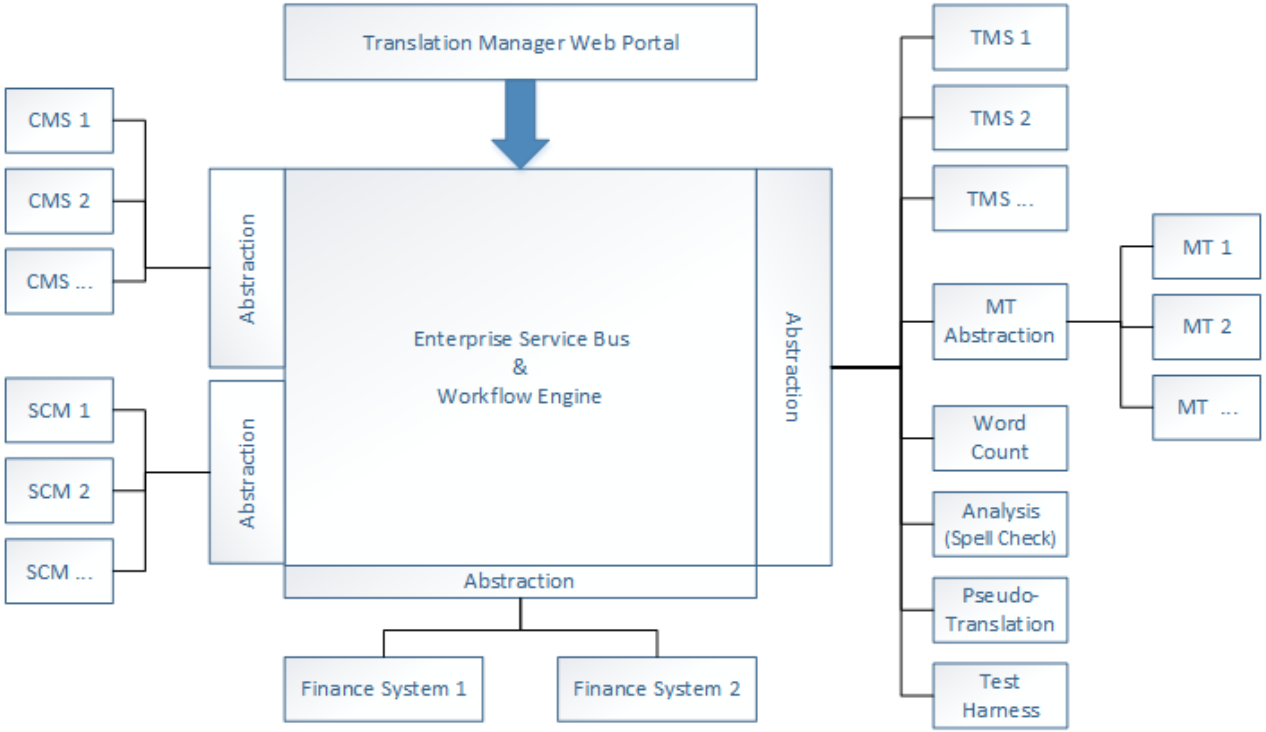
XLIFF 2.0, and [**XLIFF 2.0** + **ITS 2.0** =] **XLIFF 2.1**

And About Understanding Your Environment…

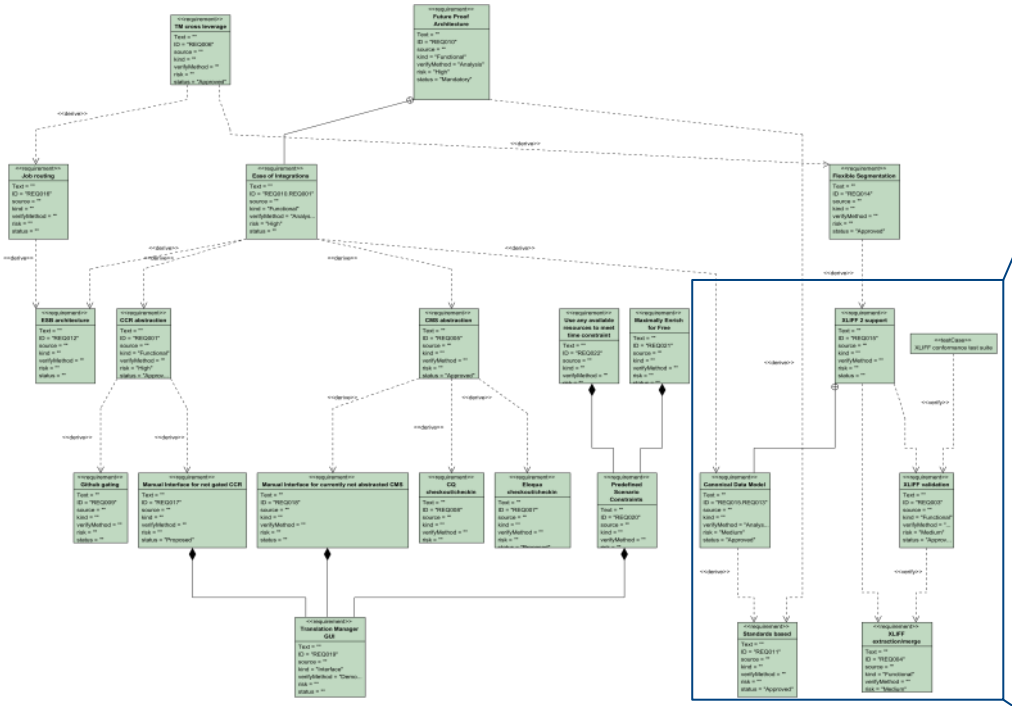
Our Existing Environment

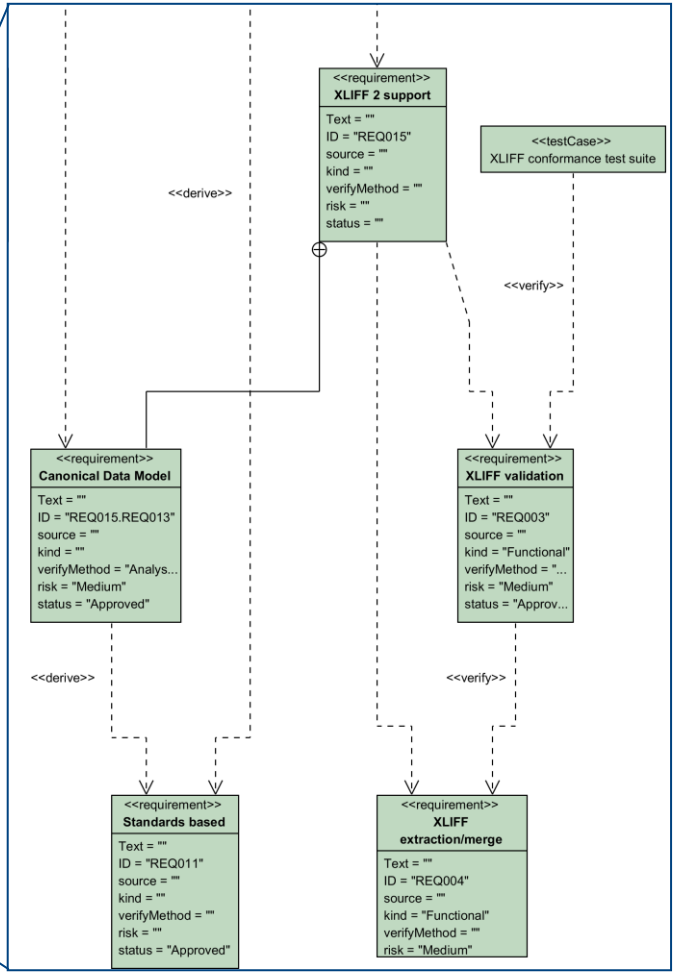


Our Proposed Solution



Where Does XLIFF Fit Into This?

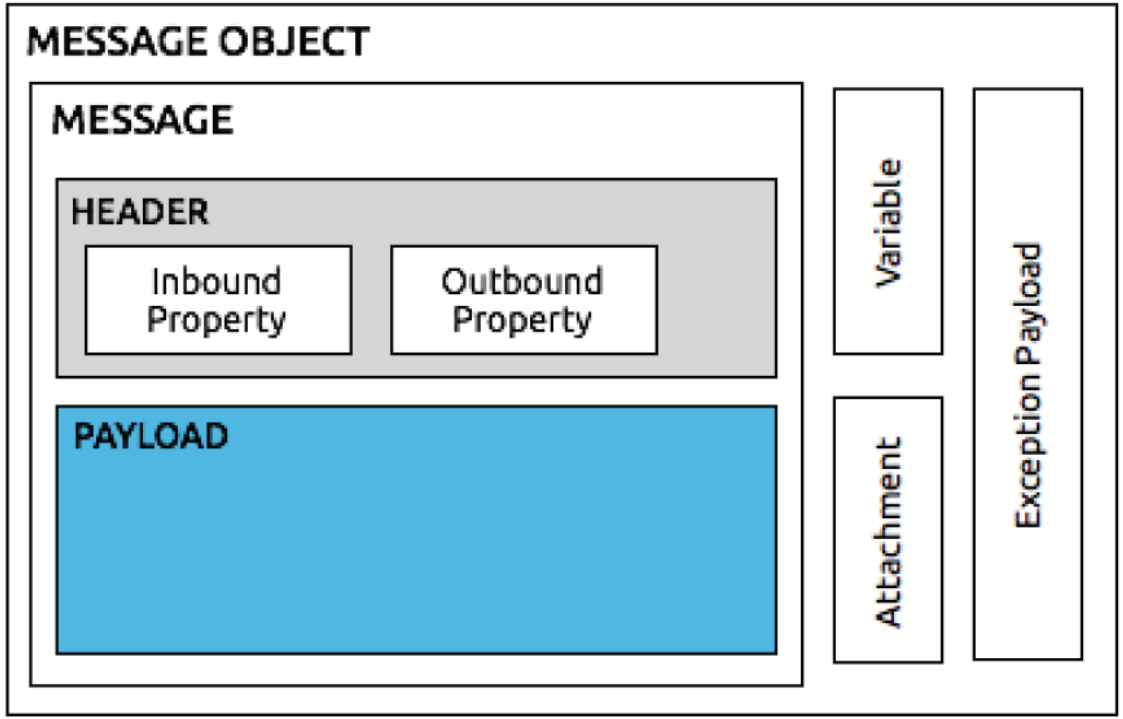




XLIFF As Canonical Data Model

- An XLIFF 2.x file is the basic payload token for a bilingual project
- This drives a few Architectural and Functional Requirements, such as:
  - BUS shall have XLIFF validation service
  - BUS shall have XLIFF extractor/merger service
  - Failure to validate XLIFF payload at any step shall be handled with canonical exception handling patterns
  - The data model shall contain a simplified bidirectional mapping between XLIFF 2.x and XLIFF 1.2

XLIFF and the ESB Message Structure



http://www.mulesoft.org/documentation/display/current/Mule+Message+Structure

Challenge #1: Converting all Incoming Content to XLIFF 2.x

If possible, leverage export capabilities of CMS / SCR to XLIFF

Convert known formats to XLIFF (commercial-grade converters)

Not required at that stage to properly segment the content

Conversion is equivalent to a text extraction – might be handled via default TMS as first L10n transformation

Difficult to accept “philosophically” but technically useful

Use Case #1: Supporting In-Context Preview

A very basic requirement that is often overlooked by tool vendors

Still need to iron this one out – conceptually simple, technically challenging

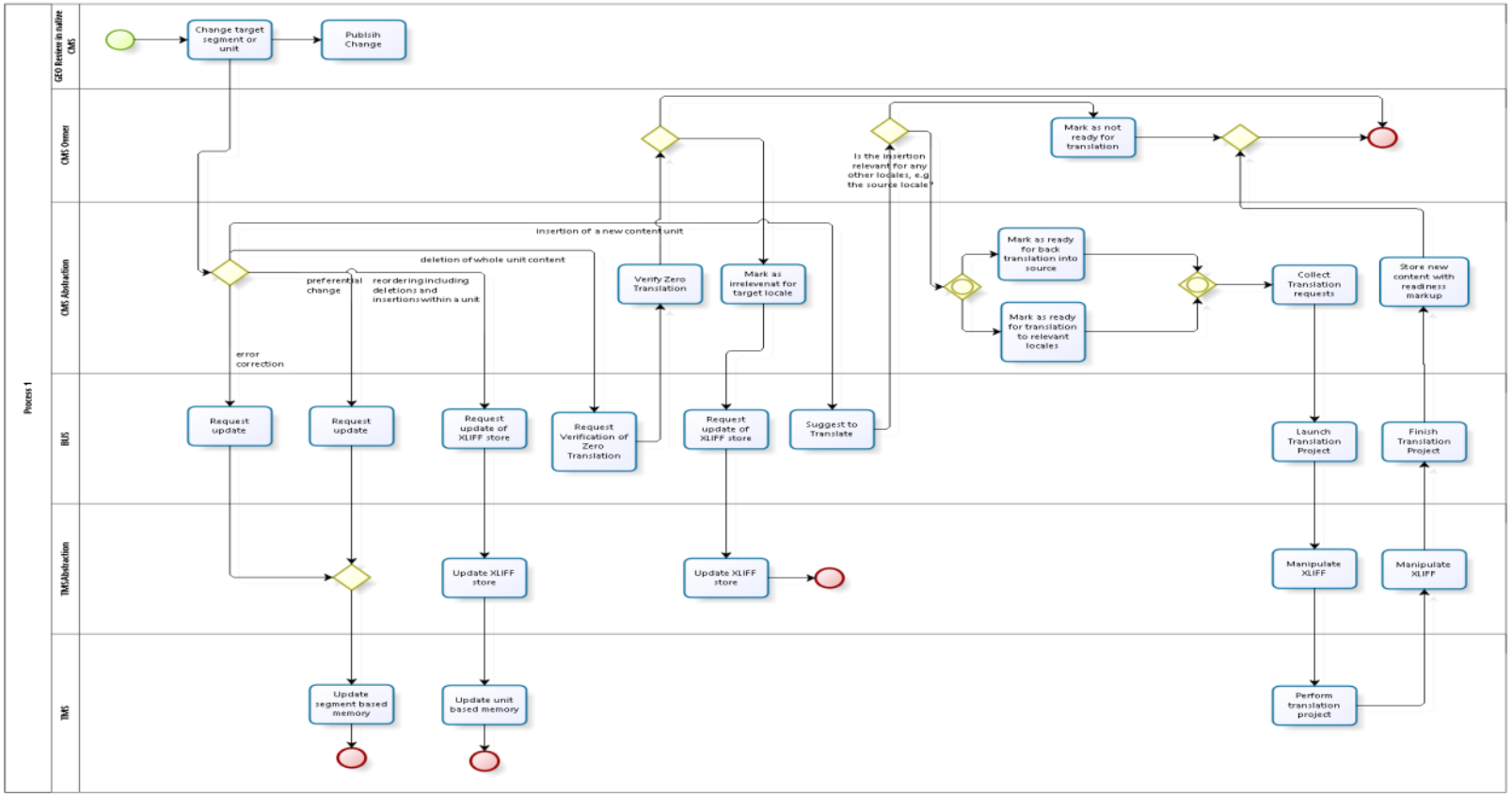
Maintain mapping between source format and XLIFF at all time, allowing for a call to a preview capability

Need to be able to rebuild the target file at multiple stages during the translation process

L10N BUS would support “branching out” from specific stages in current workflow to generate preview of translated page

Currently a major limitation on SDL TMS as CTA only supports one trip back to CMS

Use Case #2: Capturing Changes Made in CMS



Use Case #3: Sending Payload through Multiple L10N Transformations

Typical use case:

1. Run through first TMS, segmenting at paragraph level to extract in-context matches (high quality / high reliability)
2. Run through second TMS, segmenting at sentence level to increase TM leverage
3. Run new segments through MT
4. Run file through TMS for final review (post-edit)
5. Offer in context review to final in-country approver

Update status of individual XLIFF elements (TUs) as needed / relevant. Any change, no matter where made, gets captured in XLIFF store

Challenge #2: Build versus Buy

- ROI of such a complex solution is difficult to establish
- Focus is on eliminating major pain points for our stakeholders, time-to-market, and efficiency
- Need to evaluate potential solution from 3rd parties against our set of requirements / features
  - We might consider a solution that meets 80% of our requirements and collaborate with a 3rd party provider to develop the remaining 20%
  - Very few examples in the industry of companies having successfully built and deployed their own L10N BUS solution

In Conclusion

We recently embarked on a major project that [we hope] will greatly improve the provision of L10N services at Intel.

We are partnering with **CNGL/ADAPT** to design the **data model** and **architecture** for a **modular**, **extensible**, **vendor-agnostic**, and **future-proof I18n/L10n service bus**.

The proposed **data model** and the overall **bus architecture** benefit from the use of a **metadata-rich message** (**workflow token**) format that is largely informed by recent standards such as **CMIS 1.1**, **ITS 2.0**, **XLIFF 2.0**, and **XLIFF 2.1**.

Modularity of the above-mentioned **standards** offers a robust match for a **generalized and abstracted BPM bus** solution connecting a number of **messaging brokers**, grouping **Content Management Systems**, **Code Control Repositories**, and **I18n/L10n services** that cover **code scanning**, **pseudo-translation**, **machine translation**, **human translation**, **etc**.

Why Did I Share All This With You Today?

- I wanted to thank you for contributing to standards such as XLIFF
- I wanted to encourage tool providers to adopt / support XLIFF
- I obviously need all the help I can get ;-)