

# Developing and implementing a new tool into Claroline

Methodology and good practices to guarantee  
a successful collaboration between the “client”  
and the developer

Dominique FRANCOIS, UCL – ILV and Frédéric MINNE, UCL - IPM

# Overview

- Introduction
- Context
- Problems
- Solutions
- Conclusion

# Introduction

*Methodology and good practices to guarantee a successful collaboration between the “client” and the developer*

- Methodological objective
- Work in progress
- Generic but not universal

# Context

- Framework
- Actors
- Glossary

# Framework: FDP funding

- Fonds de développement pédagogique
  - Overview
  - Period and chronology

# Actors

- “Client”: Institut des Langues Vivantes (UCL -DF)
  - staff
  - 14 languages
  - public
  - Needs (Bologna)
- Developer : Institut de Pédagogie Universitaire et des Multimédias (UCL-FM)
  - function in Claroline

# Glossary: Needs (1)

- Function and relevance (generic and specific)
- Encoding entries: required fields
  - Word(s)
  - Definition, translation
  - Inflected forms *do – did – done – doing ...*
  - Register *formal, informal, ...*
  - Local variations *BrE colour, AmE color*
  - Sound/Image
  - Phonetic transcription
  - Dictionary(ies) *general, specific, topic, ...*

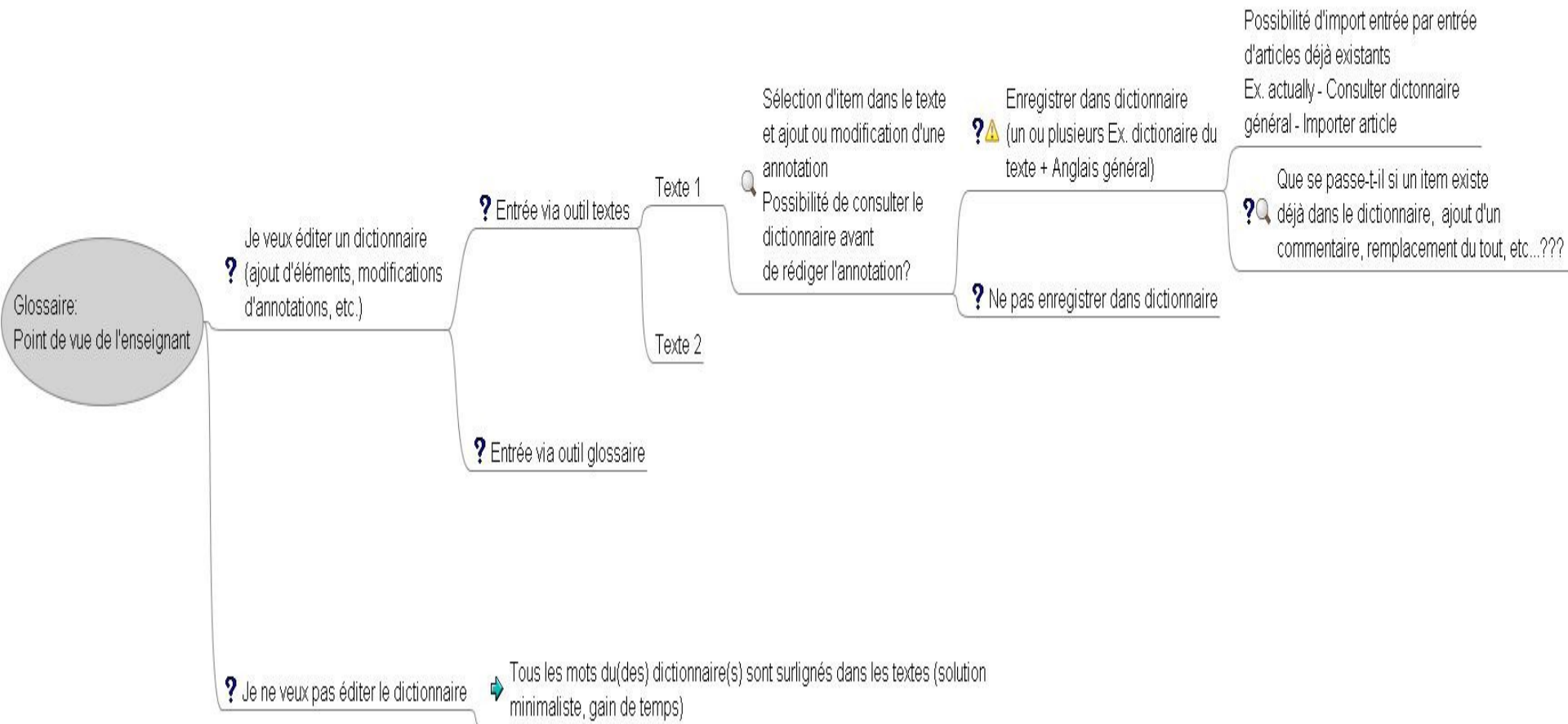
# Glossary: Needs (2)

- Options: preview, print, disable
- Various interlinked (overlapping?) levels:
  - Language *English*
  - General
  - Specific *Business English*
  - Course *BAC 3 IAG*
  - Unit *Telephoning*
  - Document (text, video, ...)

# Glossary: Needs (3)

- Inflected forms
  - Priority to full words (“now” not highlighted in “nowhere” >< plurals, “likely” vs. “unlikely”, separable verbs “vasthouden”, houdt ...vast)
  - Priority to polylexical units (“letter box”)
  - Case

## Glossary: Needs (4)



# Problems

- Developer and “Client”
- Institution versus Community
- Development process

# Developer and “Client”

Various representations related to 2 different  
fields and “worlds”  
(technology vs. pedagogy, needs vs.  
technical limitations)

# Developer and “Client”

- Different languages
- Different backgrounds and know-how
- Pedagogical needs vs. technical and technological limitations, time and financial restrictions, ...

# Institution versus Community

Institutional and community tensions  
due to different objectives

# Institution versus Community

- Generic versus Specific
  - Institutional level: multi-disciplinary
  - Community level : multi-institutional

# Institution versus Community

- Diverging philosophies
  - User-friendliness versus needs
  - Choices and technological restrictions versus innovation



# Institution versus Community

- Rhythm and deadlines
  - Claroline Release Calendar
  - Academic Calendar

# Development process

A software is an evolving entity, not only a final product released to the client

# Development process

- Need tests and feedback
- Maintenance : software lifecycle does not end with the release
- Sometimes difficult to see the development progress

# Solutions

- Developer and “Client”
- Institution versus Community
- Development process

# Developer and “Client”

- Reciprocal trust
  - (Systematic) coordination and exchange times and places
    - virtual (mail, wiki...)
    - Face-to-face (meetings...)
  - Role of the “client” in the development process

# Developer and “Client”

- Domain and requirements analysis
  - Common language
  - Domain Analysis
    - Current system : how does it work today ?
    - Future system : requirements
  - Based on use cases instead of solutions

# Institution versus Community

- Appointing a mediator
  - as a buffer
  - as a contact person
  - as a translator
- Making concessions
- Communicating !!!

# Institution versus Community

- Future plugin architecture
  - More flexibility
    - technologies at stake
    - complexity
    - timing and deadlines
    - generic versus specific

# Development process

- Agile Processes (XP, TDD, RAD...)
  - Incremental/iterative development processes
  - Release
    - soon
    - often
  - Prototypes
    - test, show
- Documentation



# Conclusion

- Communicate at any time and any level
- Choose a pragmatic way because :
  - no prefabricated solution
  - no universal solution
    - every project is different
    - every developer is different

# References

- Agile Alliance : <http://www.agilealliance.org/>
- Pragmatic Programmer :  
<http://www.pragmaticprogrammer.com/>
- eXtreme Programming :  
<http://www.extremeprogramming.org/>