Integration of GF into Be Informed

Jeroen van Grondelle
Herko ter Horst
Xander Ulterlinden

be informed

Project Context

- Implement Initial Integration of GF into Be Informed
  - Goal:
    - Show complete cycle of Ontology -> AST -> Linearization
    - In Be Informed Product using SDK
    - Based on Java GF Runtime
  - Risk reduction in project plan:
    - Proof integration early to identify tech issues
    - Without being functionally complete or have complete grammars
  - In DoW as **Milestone 12.1**
- After being trained in GF in Apeldoorn
- Performed independent of
  - Drafting of Requirement Document D12.1
  - Grammar Engineering for 4 Be Informed Domains

be informed

Approach

- Develop GF based explanation in parallel to our existing explanation service
  - Run in parallel for evaluation
  - Migrate when mature
  - Keep old implementation for unsupported languages
- Use the grammar modularization approach developed in WP12 as published in CNL 2012 paper
- Use the Tbox grammars as developed for the CNL paper ‘as is’, integration focuses on ABox
  - No complete coverage of our meta models yet
  - Some of its features rely on lemon markup, which is not used in this iteration

be informed

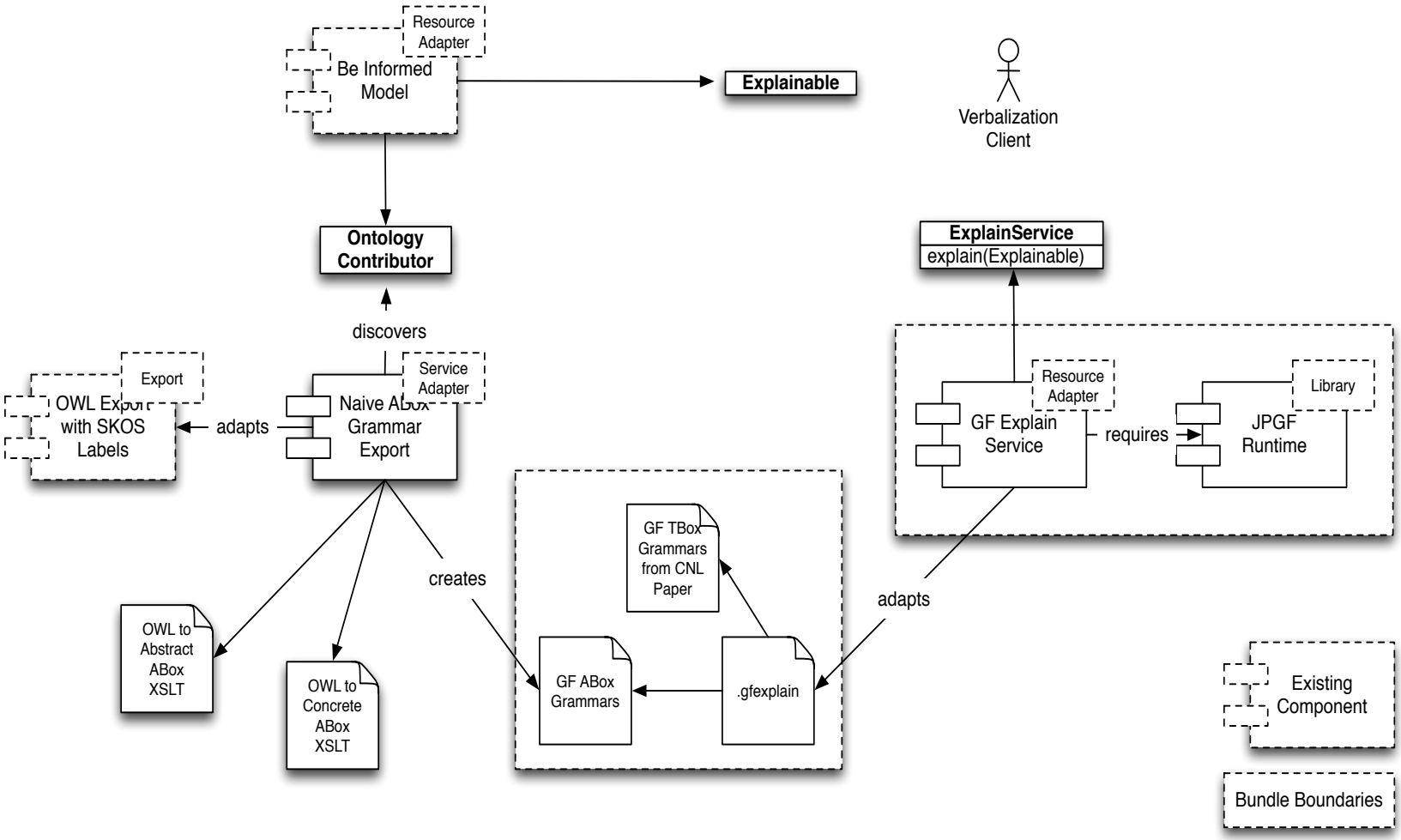
Components Developed

- Naive GF Abox Grammar Export
  - Create an Abox Grammar, without using lexical markup
  - Follows grammar patterns in CNL 2012 paper
  - Based on existing Be Informed Model to OWL/SKOS Export

- GF Based Explanation Engine
  - Embed GF Java Library in OSGi Bundle, implementing ExplainService interface in Be Informed's SDK
  - Construct AST's from Explainable interface
  - Serialize AST's using Java Runtime

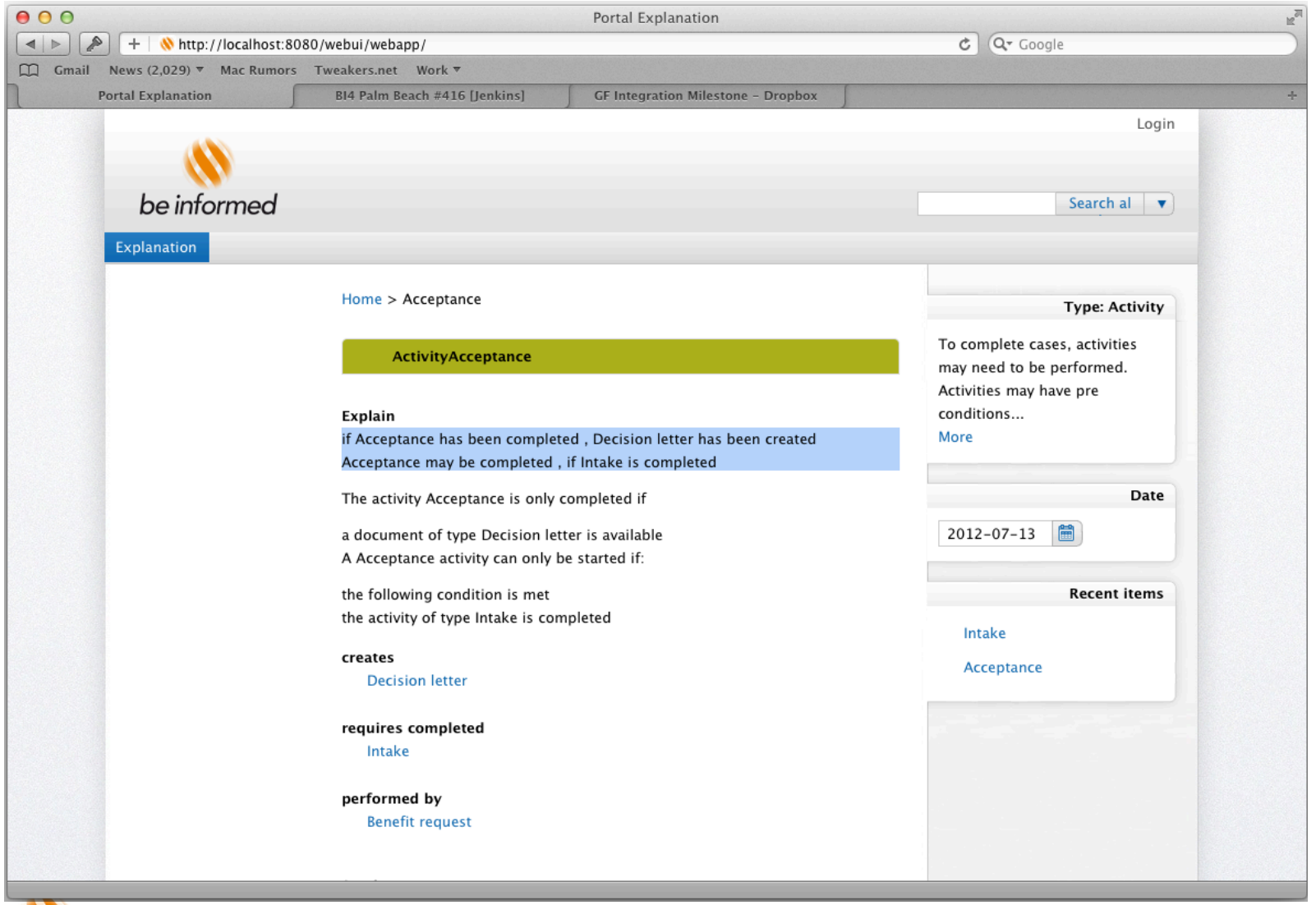
- Not all components using explanation anticipate alternative implementations being available in parallel
  - Patches needed here

Component Model



be informed

Result: GF linearization alongside existing



be informed

Conclusions

- Integration was succesful

- Issues encountered to follow up in the project
  - Java Runtime Interface for AST linearization
    - Crafting AST manually instead of by parsing is undocumented
    - Format used to encode AST's is not consistent with AST syntax in other GF tooling

  - In-tool GF compilation not supported by Java Runtime
    - Could we use Eclipse Plugin's infrastructure to allow in Studio GF compilation using GF Shell on model change?
      - Requires isolation of GF Abstraction in that plugin maybe?
    - Legal: Java Libarry is LGPL, GF Shell is GPL
      - Is in-process invocation of GF Shell allowed per license?