Towards an Ontology-Based Approach for Deriving Product Architectures

Héctor A. Durán-Limón, Francisco E. Castillo-Barrera, Roberto E. Lopez-Herrejón

Variability in SPLs

- Ability of a system for being customised changed or extended
- Approaches are required for modelling and resolving variability
- Approaches for automating product derivation are needed

Product Derivation

- Most approaches focus on deriving product implementation
- A few approaches focus on deriving product architecture
  - Mainly based on propositional logic
    - Each variant element has to be accompanied with Boolean formulas
  - Others rely on MDD techniques
    - The domain design is specified in terms of transformation rules
- Specifying product derivations in such approaches can be complex and error prone

Our work – product architecture derivation

- Relies on ontology-based reasoning and model-driven techniques
- We argue that ontologies bring in more expressive power resulting in shorter and less complex descriptions

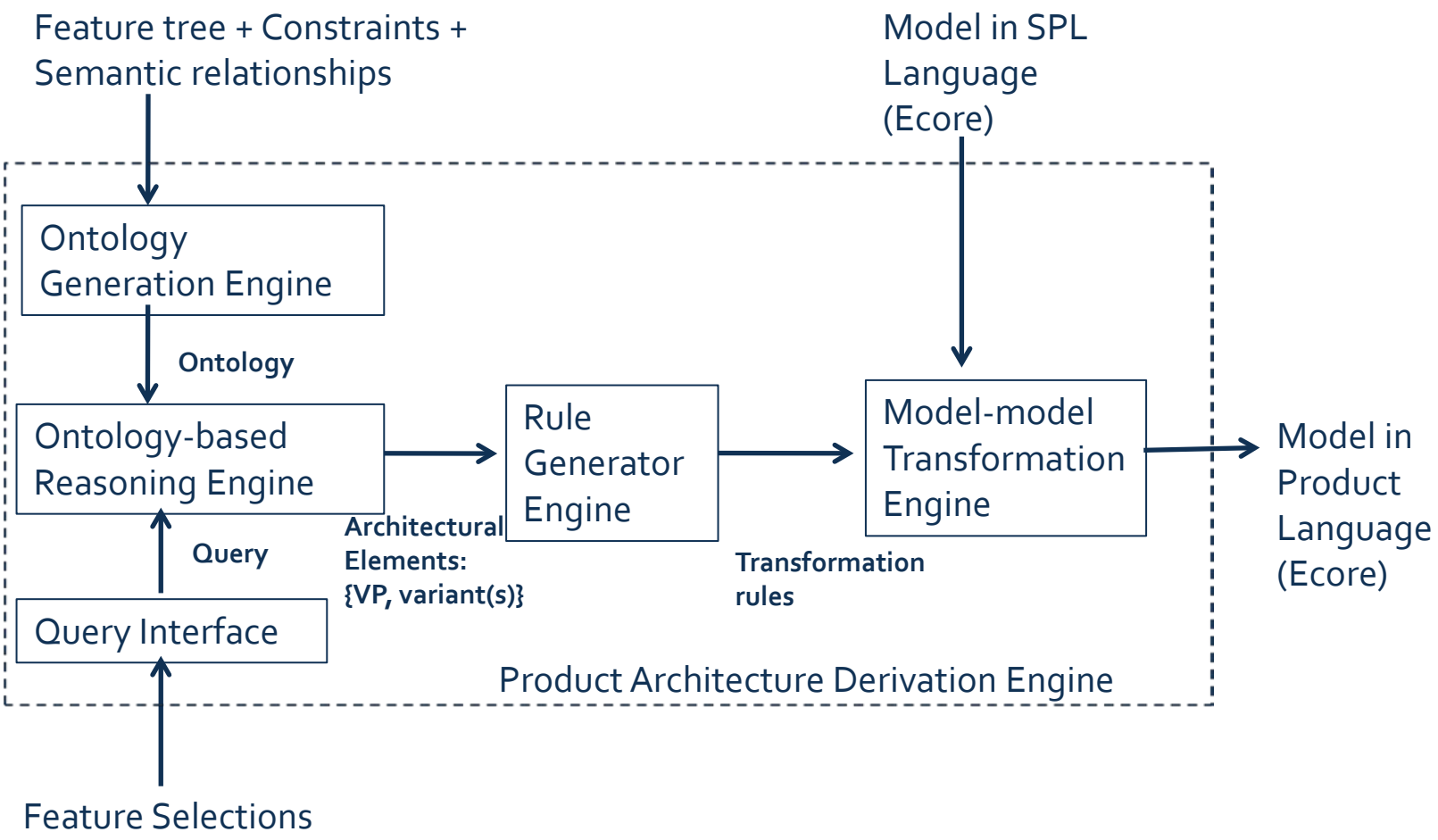
Ontologies

- Are an approach to represent knowledge in a hierarchical and structured way
- We use ontologies to capture knowledge of
  - Features
  - Constraints
  - Semantic relationships between features and architectural elements

Ontologies

- An ontology-based reasoning engine
  - Determine the architectural elements of a feature selection
  - Drive the generation of model transformation rules
    - These rules specify the transformation an SPL architecture to a specific product architecture
- Ontologies are represented in OWL
- OWL is based on description logic (DL), a family of logic-based knowledge representation formalism

Overall Framework Approach



Modelling Process – Two steps

1. Preparation
   - The metamodels of the SPL language and product language are specified. Only performed once or when new languages are used
2. Domain Modelling
   - Define feature model and feature constraints in a feature modelling tool
   - Define the SPL architecture in the SPL language using a visual editor
   - Define the architectural semantic relationships and the architectural constraints
   - Given a selection of features, the tool transforms the SPL architecture design into a product architecture design

Ontology Generation Engine

- Transforms feature tree, constraints, and the semantic relationships into Turtle
- This engine relies on:
  - an open source tool [1] as a feature tree editor
  - the approach of Wang et. al. [2] to represent a feature tree in an ontology
- Acceleo is used to transform the Ecore model into Turtle

[1] A. Gomez and I. Ramos, "Cardinality-Based Feature Modeling and Model-Driven Engineering: Fitting them Together," in VaMoS'10, 2010, pp. 61–68
[2] Hai H. Wang, et. al. Verifying feature models using OWL, Web Semantics: Science, Services and Agents on the World Wide Web, Volume 5, Issue 2, Software Engineering and the Semantic Web, June 2007, Pages 117-129

Query Interface

- Executes ontology queries based on feature selections
  - The queries obtain:
    - The architectural commonalities
    - The architectural variabilities, represented as a set of tuples:
      - <variation point, variants>
- Developed in Java and uses the OWL API library to perform the queries

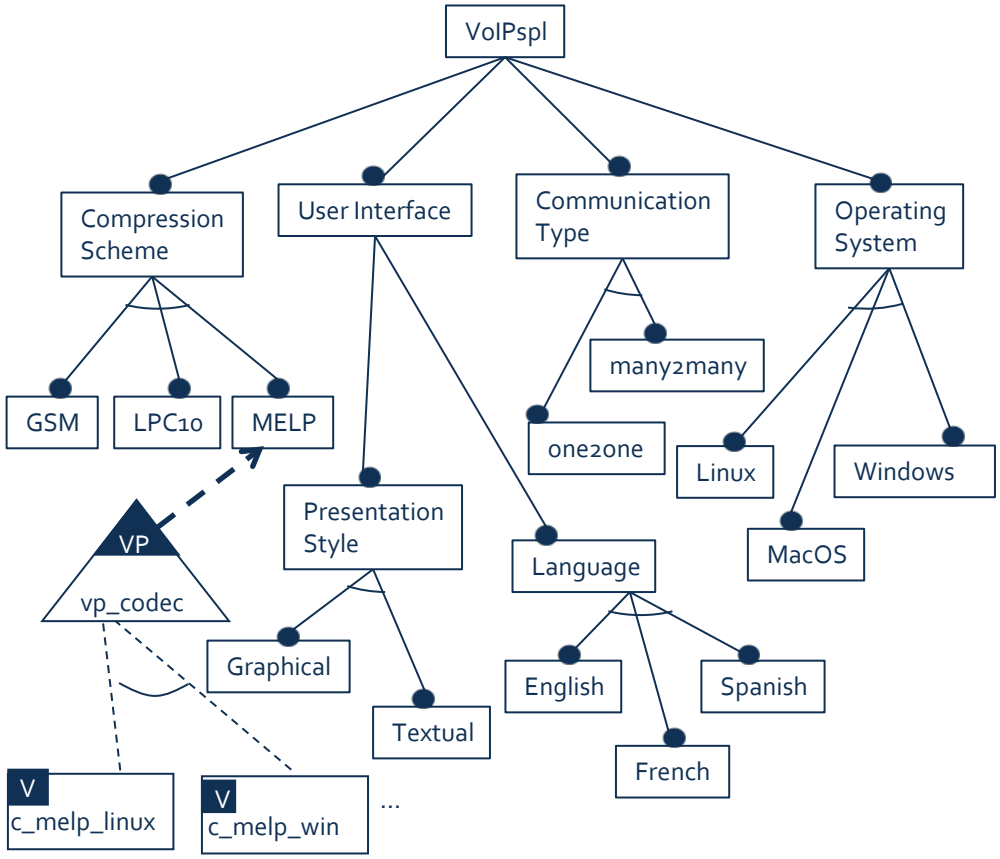
Query Interface

- Four type of queries :
  1. Obtain the commonalities: isCommonalityOf some $rootFeature_j$
  2. Obtain the architectural variation point associated with a particular feature *i*: isVariationPointOf some $feature_i$
  3. Obtain architectural variants of feature *m*: isVariantOf some $feature_m$
  4. Determines feature dependencies: isDependentOf some $feature_k$

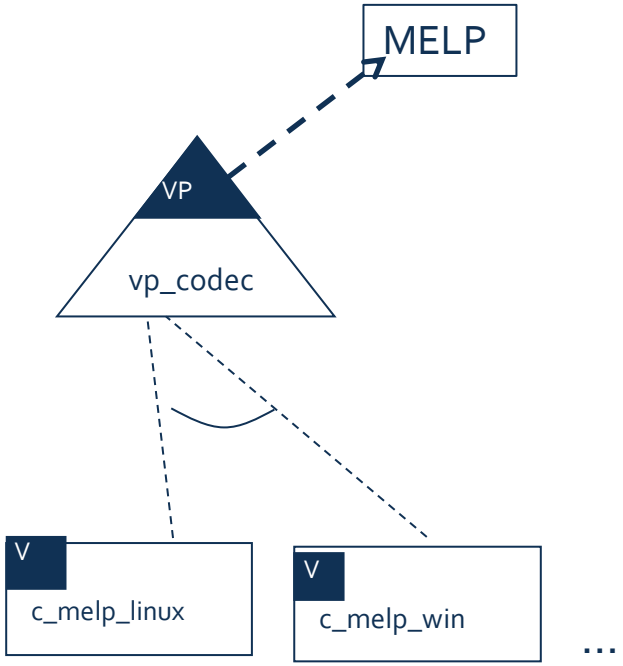
Rule Generator Engine

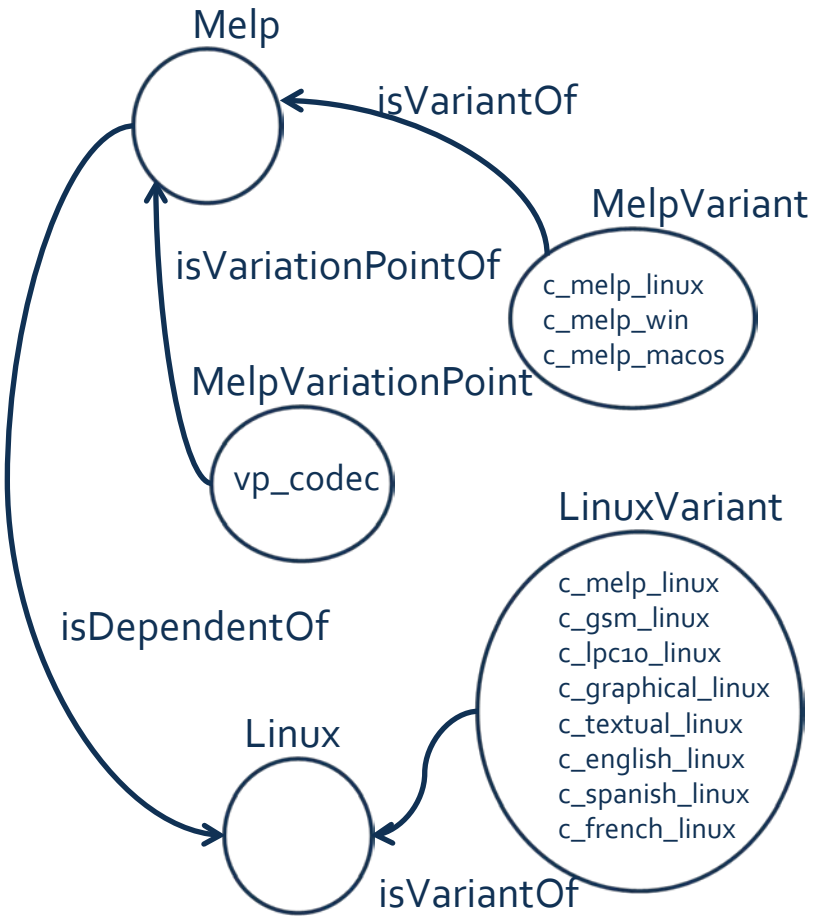
- This engine is in charge of:
  - Producing ATL rules
  - Invoking an ATL transformation engine to obtain the transformation from the SPL architecture to the product architecture
- Rules are generated based on the
  - Architectural commonalities
  - Architectural variabilities
- Development of this module is underway

Case Example: Feature Tree



Case Example: part of the generated ontology





Case Example: feature selection

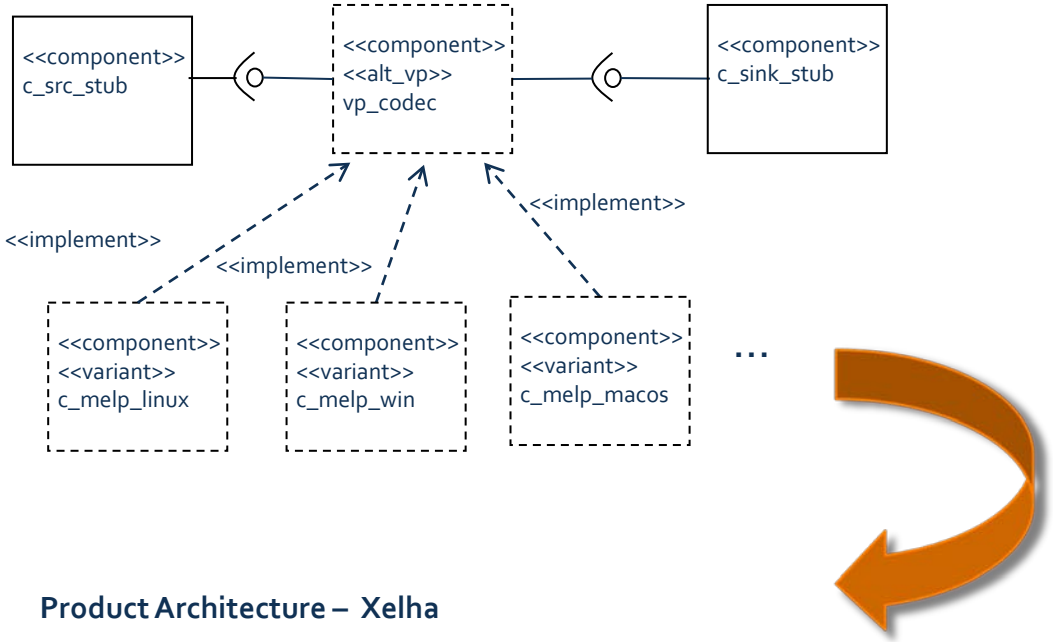
- Consider a number of feature selections are carried out including
  - Melp as the compression scheme
  - Linux as the operating system platform

Case Example: an ontology query

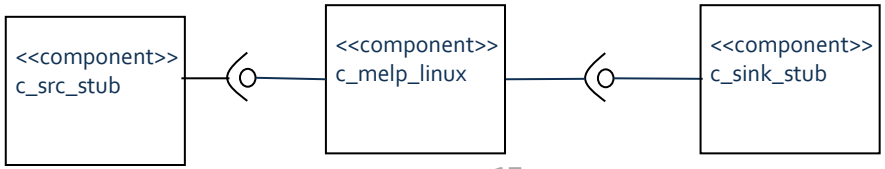
- Query to obtain the variants of Melp:
  - **isVariantOf some Melp and isVariantOf some Linux**
  - The result of this query is c_melp_linux
- In contrast, when using propositional logic:
  - **c_melp_linux requires (Melp and Linux) and (c_graphical_linux or c_textual_linux) and (c_one2one_linux or c_many2many_linux) and (c_english_linux or c_spanish_linux or c_french_linux)**
- The expressive power of DL is clearly more adequate than propositional logic

Case Example: deriving product architecture

SPL Architecture – PL-Xelha



Product Architecture – Xelha



Conclusions and Future Work

- We presented an approach to architecture product derivation based on ontologies and MDD
- Our driving motivation is to simplify the descriptions of architectural constraints
  - Since DL has more expressive power than propositional logic
- The implementation of the Rule Generator Engine and the OVM editor are underway
- An evaluation with large-scale and complex architectures is necessary

Acknowledgements

FWF Der Wissenschaftsfonds.