Mining Features from the Object-Oriented Source Code of a Collection of Software Variants Using Formal Concept Analysis and Latent Semantic Indexing

R. AL-msie'deen*, **Abdelhak-Djamel Seriai***, M. Huchard*, C. Urtado**, S. Vauttier**, and H. Eyal Salman*

* LIRMM / CNRS & Montpellier 2 University, France
**Ecole des Mines d'Alès, Nîmes, France

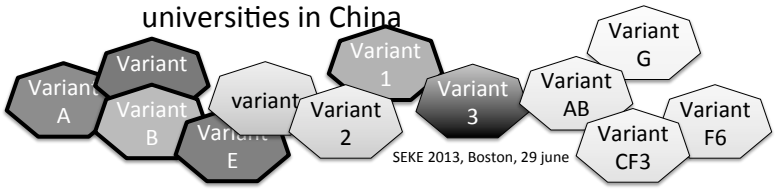
Seriai@lirmm.fr

Outline

- The context and the issue
- Our goal and the main hypotheses
- Our approach : The main ideas
- The process : step by step
- Experimentation and results
- Perspectives

The context (1/4)

- **Software variants**
  - Are similar software
    - Share some features, called common features, and differ in others, called optional features
    - Developed by ad-hoc reuse techniques such as clone-own
    - Usually, an existing product is copied and later modified to meet incremental demands of customers
  - Example
    - Wingsoft Financial Management System (WFMS)
    - Variants of the WFMS systems have been used in over 100 universities in China



The context (2/4)

- Software product Line
  - SPL supports efficient development of related software products
  - Manages common and optional features
    - A Feature is a system property relevant to some stakeholder used to capture commonalities or variations among systems in a family
  - Promotes systematic software reuse from SPL's core assets (such as features, code, documentation and etc.)

The context (3/4)

- Software Product Line
  - **Domain Engineering** : development for reuse
  - **Application Engineering** : development by reuse

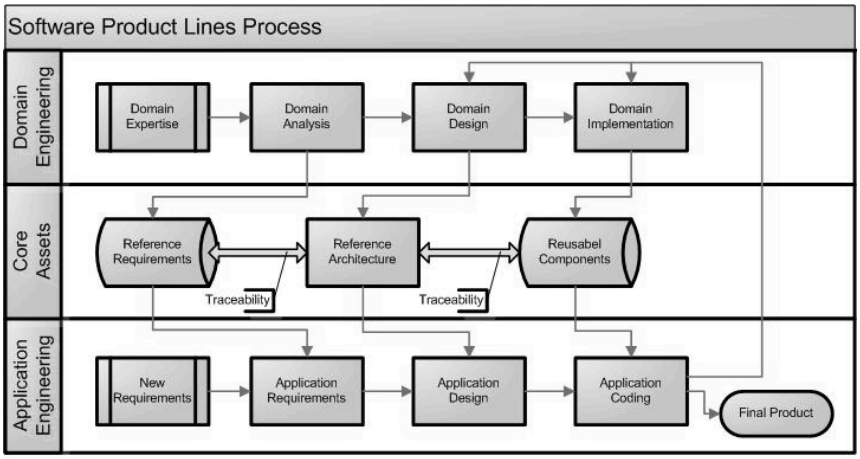
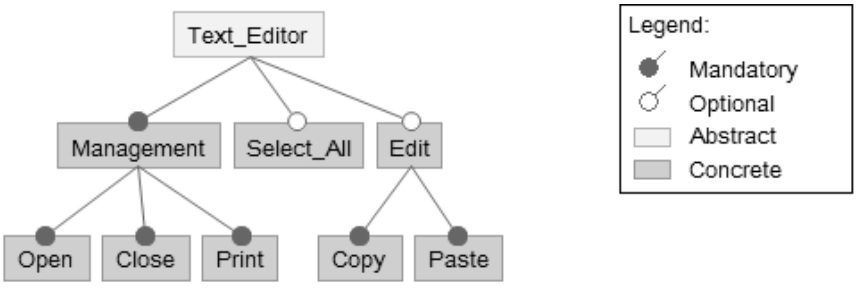


Image from : http://poltman.com/pm-en/img/TechnicalInformation/SoftwareModernization/ProductLines/ProductLines-01.jpg

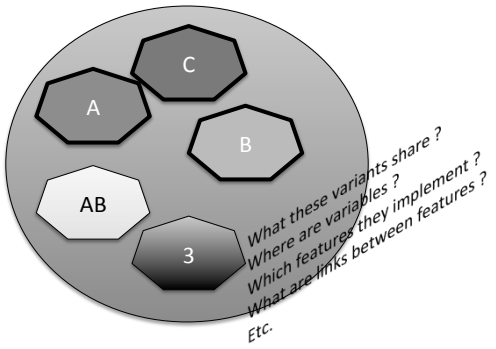
The context (4/4)

- Software Product Line
  - Feature model (FM)
    - Is a tree-like graph of features and relationships among them
    - Used to represent commonality and variability of SPL members at different levels of abstraction



Issue

- Software variants
  - Difficulties for :
    - Reuse
    - Maintenance
    - Comprehension
    - Impact analysis

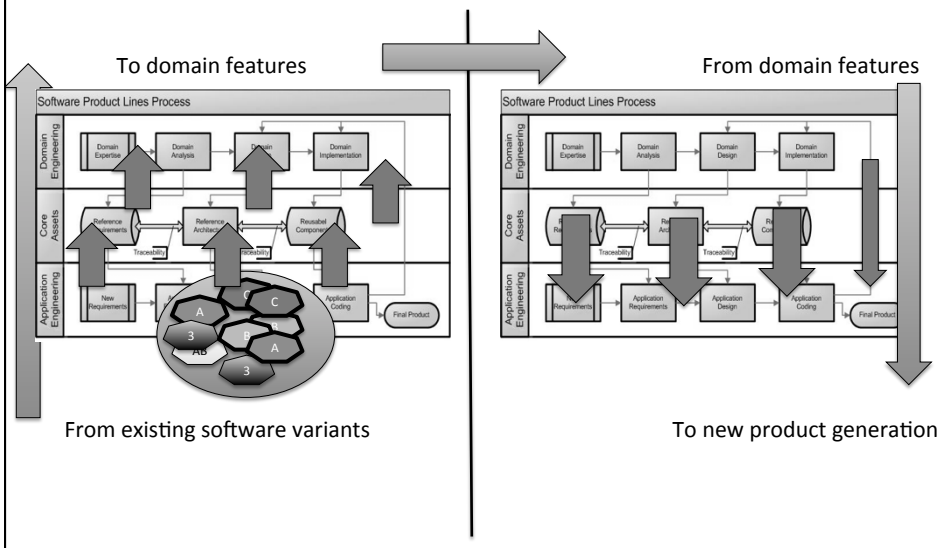


- Software Product Line
  - Design from scratch is a hard task (domain engineering)

Our Goal (1/2)

- Reengineering existing software variants into a software product line
  - Benefits
    - Software variants will be managed as a product line
    - Software product line will be engineered started from existing products (not from scratch)
  - Strategy
    - Feature model mining (reverse engineering step)
      - **Mining features**
      - Mining feature model structure (group of features)
      - Mining feature constraints
      - Mining feature relationships
    - Source code Framework generation (reengineering step)

Our Goal (2/2)



Our main hypotheses

1. Mining feature From object oriented source code
2. Focus on functional features
   - Functional features express the behavior or the way users may interact with a product
3. Focus on feature implemented at the programming level
   - The elements of the source code reflect these features
   - Feature are implemented as package, class, attribute, method, local variable, attribute access, method invocation, etc.
4. A Feature has the same implementation in all product variants where it is present (we not consider evolution)

The main ideas(1/4)

1. The initial search space is composed of all OO elements of software variants
2. Characterizing OO elements implementing features to cluster theme together
   - **Are similar elements : lexical similarity**
   - Are dependent elements : structural dependencies

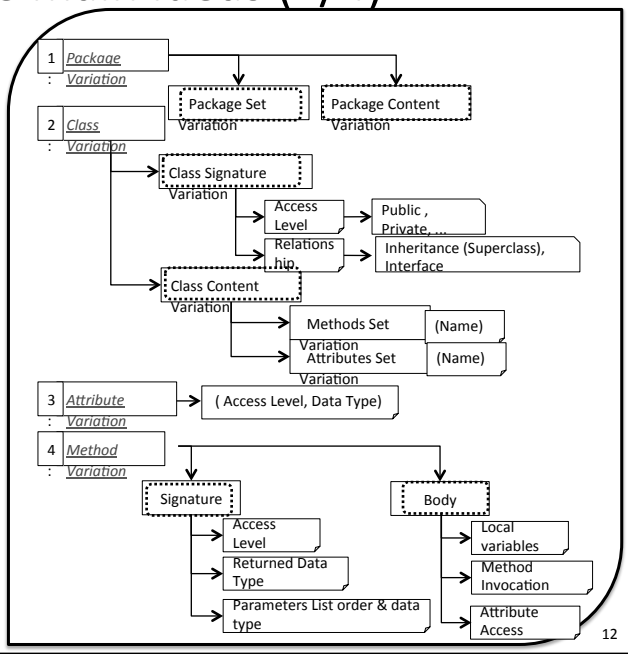
Identifying clusters composed of the most similar OO elements

based on LSI

The main ideas (2/4)

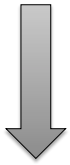
3. Optional features are implemented as variation in source code

Mining variations (= search space)



The main ideas(3/4)

4. Reducing the search space by Isolating groups of variations corresponding to some related features

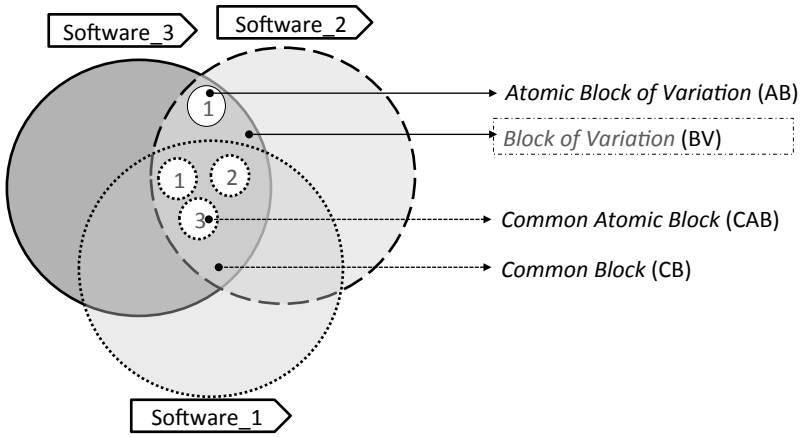


From one large search space to many sub search spaces

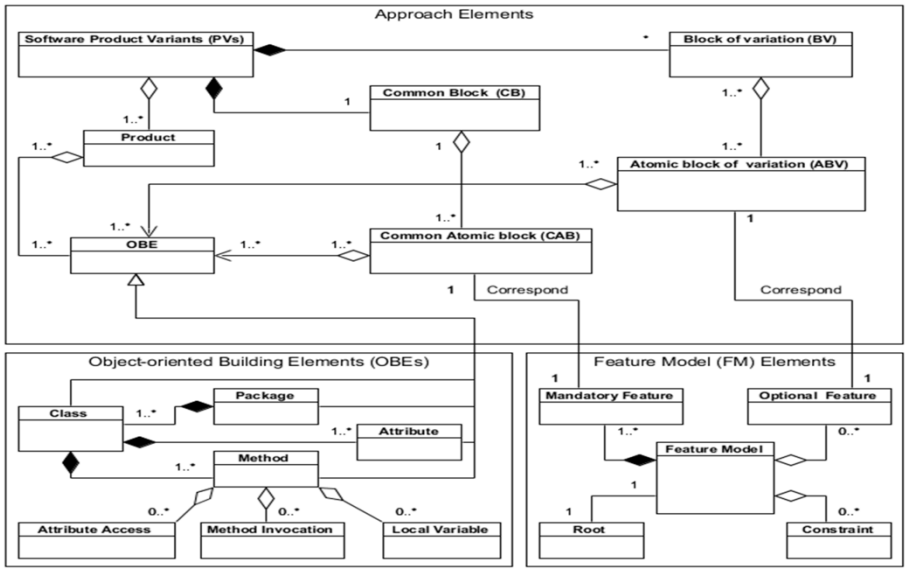
Identifying all groups of OO elements representing
differences or intersections between variants

based on FCA

The main ideas (4/4)



Object-to-feature mapping model

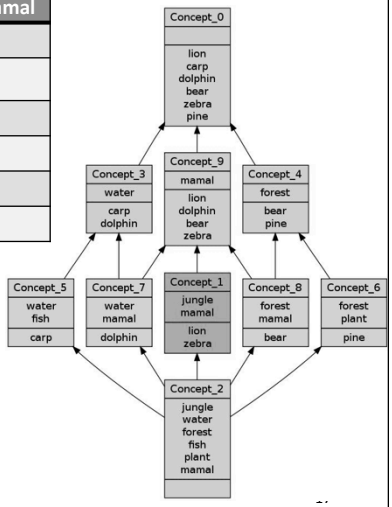


Used techniques : FCA and LSI (1/3)

- Formal Concept Analysis (FCA)
  - Is a technique for data analysis and knowledge representation based on lattice theory
  - It identifies meaningful groups of objects that share common attributes
  - It provides a theoretical model to analyze hierarchies of these groups
  - In order to apply FCA based on the definition of a formal context or incidence table of objects and their attributes

Used techniques : FCA and LSI (2/3)

	jungle	water	forest	fish	plant	mamal
lion	x					x
carp		x		x		
dolphin		x				x
bear			x			x
zebra	x					x
pine			x		x	

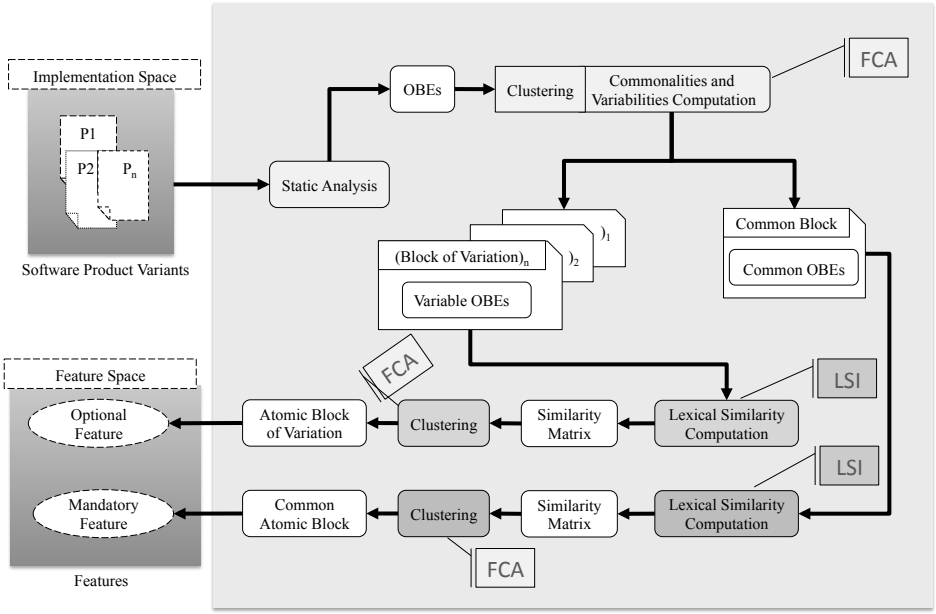


Extracted from : http://code.google.com/p/erca/wiki/FcaIntroduction

Used techniques : FCA and LSI (3/3)

- Latent Semantic Indexing (LSI)
  - Compute textual similarity among different documents
    - Based on the occurrences of terms in documents
      - If two documents share a large number of terms, those documents are considered to be similar
  - Three steps
    - A corpus of documents is built after pre-processing such as stop word removal and stemming performing
    - A term-by-document matrix is built, where each column represents a document and each row is a term. The values in the matrix indicate the frequency of the term occurring in the document
    - The similarity among documents is calculated using cosine similarity

The mining process : step by step



Identifying the Common Block and Blocks of Variation (1/3)

- Two steps
  1. A formal context, where objects are product variants and attributes are OBEs is defined
  2. Calculate corresponding AOC-poset
     - The intent of each concept represents OBEs common to two or more products
       - The intent of the most general (*i.e.*, top) concept gathers OBEs that are common to all products. They constitute the CB
       - The intents of all remaining concepts are BVs
         - » They gather sets of OBEs common to a subset of products and correspond to the implementation of one or more features
     - The extent of each of these concepts is the set of products having these OBEs in common

Identifying the Common Block and Blocks of Variation(2/3)

- Example

TEXT EDITOR VARIANTS DESCRIBED BY THEIR FEATURES

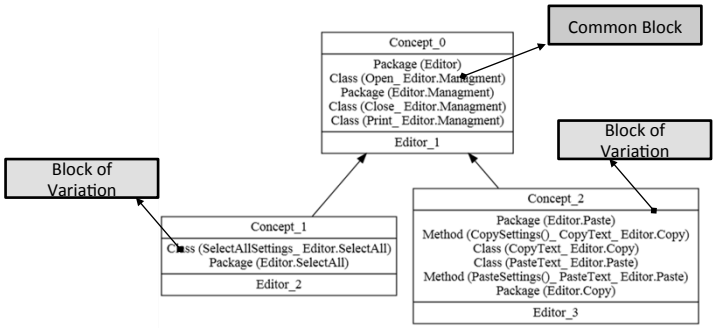
Variant Name	Features
Editor_1	Core (Open, Close, Print)
Editor_2	Core, Select_all
Editor_3	Core, Copy, Paste

The formel context

	Package(Editor.Managment)	Class (Close_Editor.Managment)	Class (Open_Editor.Managment)	Class (Print_Editor.Managment)	Package(Editor.Copy)	Class (CopyText_Editor.Copy)	Method (CopySettings_CopyText)	Package(Editor.SelectAll)	Class (SelectAllSettings_SelectAll)	Package(Editor.Paste)	Class (PasteText_Editor.Paste)	Method (PasteSettings_PasteText)
Editor_1	×	×	×	×								
Editor_2	×	×	×	×				×	×			
Editor_3	×	×	×	×	×	×	×			×	×	×

Identifying the Common Block and Blocks of Variation(3/3)

- Example
  - The AOC-poset



Identifying Atomic Blocks (1/5)

- Three steps
  - Exploring the BV's AOC-poset to Identify Atomic Blocks of Variation
  - Measuring OBEs' Similarity Based on LSI
  - Identifying Atomic Blocks Using FCA
- Exploring the BV's AOC-poset to Identify Atomic Blocks of Variation
  - Exploring the AOC-poset from the smallest (bottom) to the highest (top) block
  - If a group of OBEs is identified as an ABV, this group is considered as such when exploring the following BV
  - For Common Atomic Blocks (CAB), there is no such need to explore the AOC-poset as there is a unique CB.

Identifying Atomic Blocks (2/5)

- Measuring OBEs' Similarity Based on LSI
  - Building the LSI corpus
  - Building the term- document matrix and the term-query matrix for each BV and for the CB
  - Building the cosine similarity matrix

Identifying Atomic Blocks (3/5)

- Example of cosine similarity matrix

	Class CopyText Copy	Class PasteText Paste	Method CopySettings CopyText	Method PasteSetting PasteText	Package Copy	Package Paste
Class CopyText Copy	1	0.0556	0.9921	0.1715	0.9964	-0.0166
Class PasteText Paste	0.0556	1	0.1802	0.9931	-0.0285	0.9973
Method CopySettings CopyText	0.9921	0.1802	1	0.2935	0.9780	0.1086
Method PasteSetting PasteText	0.1715	0.9931	0.2935	1	0.0880	0.9821
Package Copy	0.9964	-0.0285	0.9780	0.0880	1	-0.1007
Package Paste	-0.0166	0.9973	0.1086	0.9821	-0.1007	1

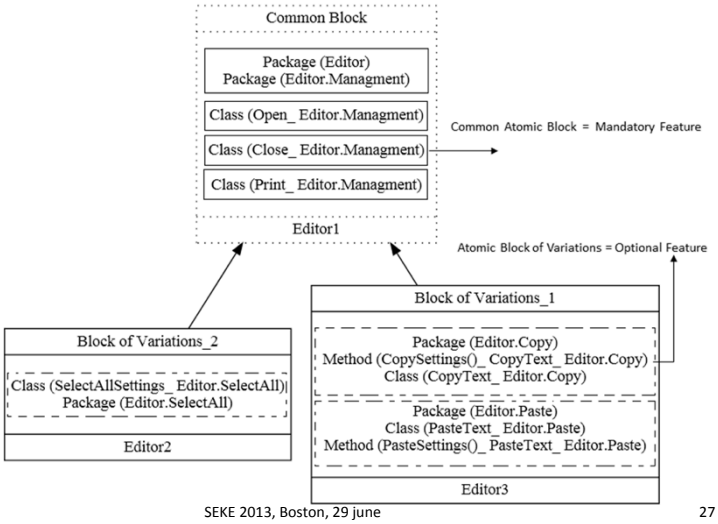
Identifying Atomic Blocks (4/5)

- Identifying Atomic Blocks Using FCA
  - Transforming the (numerical) similarity matrices of previous step into (binary) formal contexts
    - Only pairs of OBEs having a calculated similarity greater than or equal to 0.70 are considered similar
  - Example

	Class CopyText Copy	Class PasteText Paste	Method CopySettings CopyText	Method PasteSetting PasteText	Package Copy	Package Paste
Class CopyText Copy	X		X		X	
Class PasteText Paste		X		X		X
Method CopySettings CopyText	X		X		X	
Method PasteSetting PasteText		X		X		X
Package Copy	X		X		X	
Package Paste		X		X		X

Identifying Atomic Blocks (5/5)

- Identifying Atomic Blocks Using FCA



Experimentation and results (1/5)

- Case studies : Two Java open-source software: Mobile Media and ArgoUML

Product #	Mobile Media Product Description	LOC	NOP	NOC	NOOBE
P1	Mobile photo core	1,046	6	15	822
P2	Exception handling enabled	1,159	7	24	925
P3	Sorting and edit photo label enabled	1,314	7	25	1,040
P4	Favourites enabled	1,363	7	25	1,066
Product #	**ArgoUML Product Description**	**LOC**	**NOP**	**NOC**	**NOOBE**
P1	All Features disabled	82,924	55	1,243	74,444
P2	All Features enabled	120,348	81	1,666	100,420
P3	Only Logging disabled	118,189	81	1,666	98,988
P4	Only Cognitive disabled	104,029	73	1,451	89,273
P5	Only Sequence diagram disabled	114,969	77	1,608	96,492
P6	Only Use case diagram disabled	117,636	78	1,625	98,468
P7	Only Deployment diagram disabled	117,201	79	1,633	98,323
P8	Only Collaboration diagram disabled	118,769	79	1,647	99,358
P9	Only State diagram disabled	116,431	81	1,631	97,760
P10	Only Activity diagram disabled	118,066	79	1,648	98,777

Experimentation and results (2/5)

Case Study	Feature		Evaluation Metrics			
Mobile Media Features	Common	Optional	K	Precision	Recall	F-Measure
Album Management	×		0.05	83%	62%	70%
Splash Screen	×		0.05	71%	57%	63%
Create Album	×		0.05	81%	58%	67%
Delete Album	×		0.05	80%	62%	69%
Create Photo	×		0.05	81%	52%	63%
Delete Photo	×		0.05	78%	63%	69%
View Photo	×		0.05	87%	68%	76%
Exception handling		×	0.03	100%	70%	82%
Edit Photo Label		×	0.02	100%	77%	87%
Favourites		×	0.04	100%	80%	88%
Sorting		×	0.06	100%	78%	87%
ArgoUML Features	Common	Optional	K	Precision	Recall	F-Measure
Class Diagram	×		0.03	72%	56%	63%
Diagram		×	0.06	100%	80%	88%
Deployment Diagram		×	0.05	100%	74%	85%
Collaboration Diagram		×	0.06	100%	67%	80%
Use Case Diagram		×	0.03	100%	64%	78%
State Diagram		×	0.03	100%	69%	81%
Sequence Diagram		×	0.02	100%	67%	80%
Activity Diagram		×	0.06	100%	63%	77%
Cognitive Support		×	0.01	100%	70%	82%
Logging		×	0.02	100%	60%	75%

Experimentation and results (3/5)

- The effectiveness of IR methods is measured by their *RECALL*, *PRECISION* and *F-MEASURE*

$$Precision = \frac{\sum_i Correctly\ retrieved\ links}{\sum_i Total\ retrieved\ links}\%$$

$$Recall = \frac{\sum_i Correctly\ retrieved\ links}{\sum_i Total\ relevant\ links}\%$$

$$F - Measure = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}\%$$

- **Recall** is the percentage of correctly retrieved links (OBEs) to the total number of relevant links (OBEs) .
- **Precision** is the percentage of correctly retrieved links (OBEs) to the total number of retrieved links (OBEs) .
- **F-measure** is a balanced measure that takes into account both precision and recall.

Experimentation and results (4/5)

- Precision
  - For optional features appears to be high
    - This means that all mined OBEs grouped as features are relevant
    - Mainly due to search space reduction. In most cases, each BV corresponds to one and only one feature
  - For common features, precision is also quite high
    - Thanks to our clustering technique that identifies ABVs based on FCA and LSI
    - Is smaller than the one obtained for optional features
      - This deterioration can be explained by the fact that we do not perform search space reduction for the CB

Experimentation and results (5/5)

- Recall
  - Its average value is 66% for Mobile Media and 67% for ArgoUML
    - This means most OBEs that compose features are mined
    - Non-mined OBEs used different vocabularies compared to the mined ones
      - This is a known limitation of LSI which is based on lexical similarity

Perspectives

- Enhance the quality of the mining
  - Combine both textual and structural similarity measures
  - Identify junctions between features
  - More reducing of the search space
  - Etc.
- Feature model mining
  - **Mining features**
  - Mining feature model structure (group of features)
  - Mining features constraints
  - Mining feature relationships

Mining Features from the Object-Oriented Source Code of a Collection of Software Variants Using Formal Concept Analysis and Latent Semantic Indexing

R. AL-msie'deen*, **Abdelhak-Djamel Seriai***, M. Huchard*, C. Urtado**, S. Vauttier**, and H. Eyal Salman*

* LIRMM / CNRS & Montpellier 2 University, France
**Ecole des Mines d'Alès, Nîmes, France

Seriai@lirmm.fr