Leveraging Domain Vocabulary Across Artefacts

by
Tezcan Dilshener

tezcan@dilshener.de

Agenda

- Introduction
  - Research questions
  - Our Study
- Related work
  - Inspirations
- Data preparation
- Results
  - Vocabulary Correlation
  - Precision and recall
  - Comparing with another tool
- Conclusion

Introduction

- **Software require continued maintenance.**
  - **Original developers gone with the wind (left the project).**
  - **Documentation and other project artefacts tend to decay.**
- **Some of the challenges in maintenance.**
  - **Identification of high-level concepts in source code.**
  - **Understanding concept context and relations to its domain.**
- **Program comprehension overhead.**
  - **Effort required by developers without domain knowledge.**
  - **Role of domain-specific concepts' vocabulary.**
- **Our study: Comparing artefacts of two conceptually related applications addressing the Basel-II* Accord.**

✧ http://www.bis.org

Research Questions

- **Identify opportunities of using artefacts' vocabulary to reduce maintenance overhead**
  - **RQ1: What is the adherence of two conceptually related applications' vocabulary to the Basel II domain concepts?**
  - **RQ2: How can the vocabulary be leveraged when searching for concepts to find the relevant classes for implementing change requests?**
  - **RQ3: How much can our tool leverage from artefact and domain vocabulary compared to another state-of-the-art tool?**

Related Work

- **Evolution of Source Code Vocabulary by Abebe *et al.* [2]**
  - **type of vocabulary relation and what frequent terms refer to.**
  - **we compare vocabulary beyond code and include CRs.**

- **Recovery of traceability by Antoniol *et al.* [3]**
  - **from source code classes to functional requirements.**
  - **we attempt to recover between change requests and code.**

- **Bug localisation based on bug reports by Zhou *et al.* [4]**
  - **ranks source code files based on relevant bug reports.**
  - **we use the tool to rank our artefacts and compare the result.**

2. *Abebe et al. "Analyzing the Evolution of the Source Code Vocabulary", CSMR'09*
3. *Antoniol et al. "Recovering traceability links between code and documentation", TSE'02*
4. *Zhou et al. "Where Should the Bugs be Fixed?", ICSE'12*

ConCodeSe - data preparation

- **Extraction, search and analysis flow**

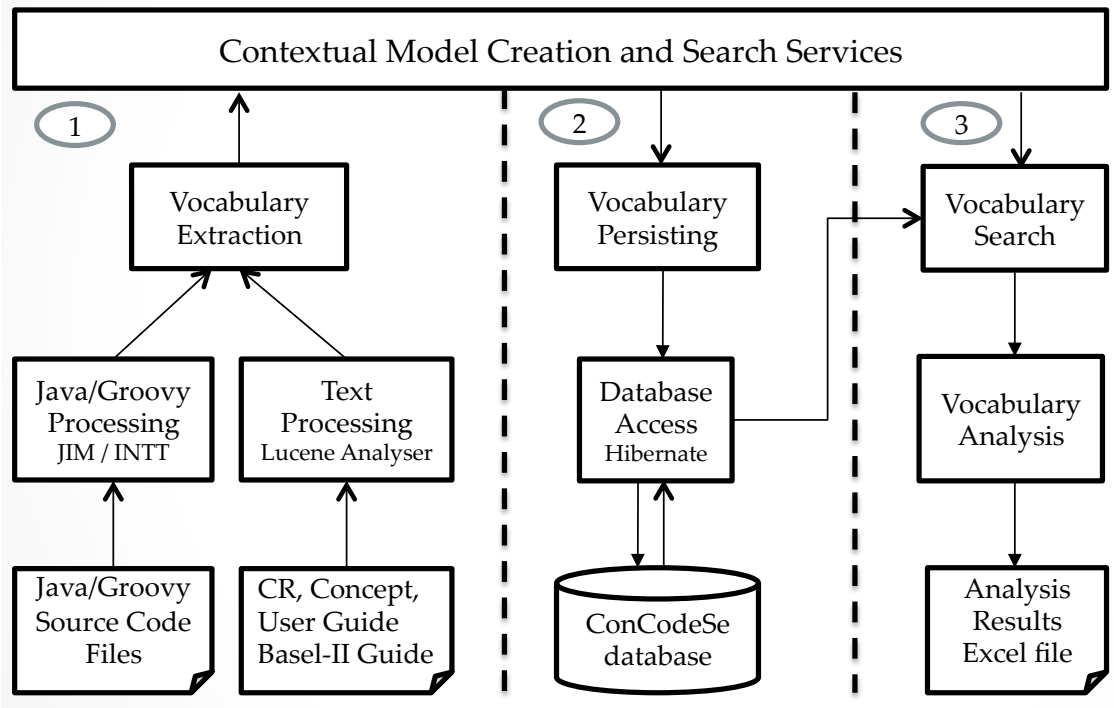

Fig. 1 ConCodeSe - Data extraction, storage and search

RQ1:Adherence to Basel II Accord

- **Searched occurrences of previously identified concepts.**
  - **87% occur in the official Basel II documentation.**
- **Each concept occurred in at least one project artefact.**
  - **concepts across all artefacts: 7 Pillar-One, 14 Pillar-Two.**
- **4 common concepts across both applications' artefacts.**
  - ***risk, index, value* and *time*.**

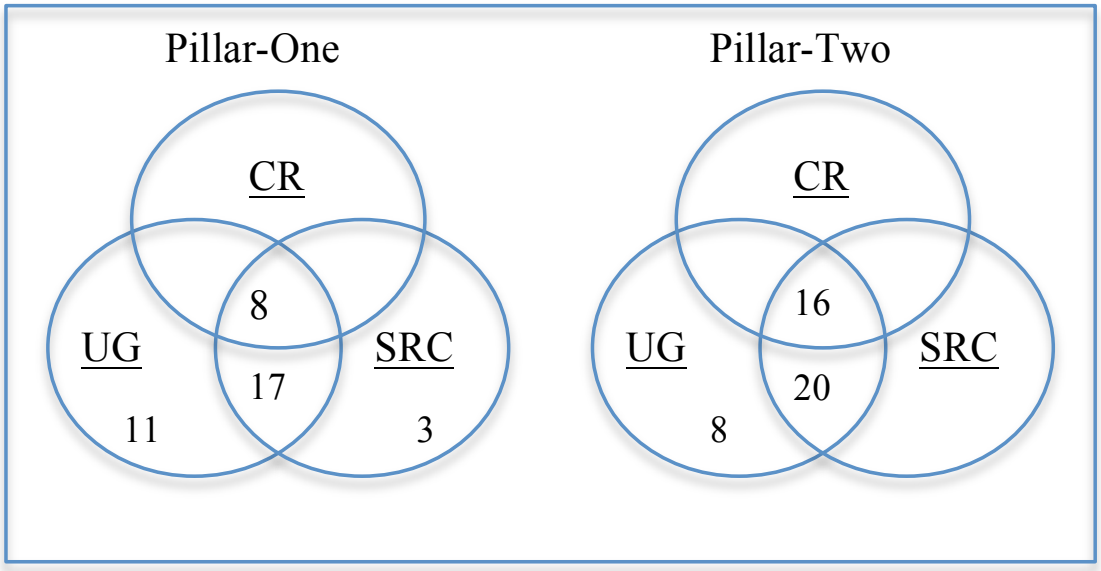

Fig. 2 Concept distribution among artefacts

RQ2:Searching for CR's classes

- **Searched each application for class names matching the concept words referred by the change requests (CRs).**
  - **For Pillar-One: 0% recall and precision.**
    - **Discarded frequently occurring concepts i.e. '*time*' & '*current*.'**
  - **For Pillar-Two: very high recall with very low precision.**
    - **Discarded project specific stop-words i.e. '*market*' & '*value*.'**
    - **Introduced a project specific mapping mechanism.**
      - **i.e. '*mask*' → '*helper*' based on project experience.**

Pillar-One		Pillar-Two	
recall (%)	precision (%)	recall (%)	precision (%)
33.33	50.00	100.00	20.27
10.00	40.00	100.00	7.59
50.00	9.96	83.33	6.17
46.15	25.00	71.43	6.17

RQ3:Our tool compared to another

- **Used a state-of-the-art traceability tool called BugLocator developed by Zhou *et al.* [5]**
- **BugLocator:**
  - **searches the corpus for the relevant classes using the terms found in a bug report.**
  - **ranks the effected files (listed in the bug report) using two different VSM similarity calculations.**
- **Performed the same search tasks for Pillar-One and Pillar-Two as of ConCodeSe**

	relevant classes	BugLocator	ConCodeSe
Pillar-One	131	15	21
Pillar-Two	46	10	16

Conclusions

- **An efficient approach to relate vocabulary of information sources for maintenance;**
  - **Basel II document, Concepts, CRs, user guide and code.**
  - **Vocabulary overlap between both application's code.**
- **Application of approach to industrial code that follows good naming conventions.**
  - **Alignment between guide and code could be improved.**
  - **Descriptive identifiers support high recall, but low precision.**
  - **Applied simple techniques and improved precision.**
- **In many cases our simple lexical text search approach outperformed BugLocator.**
  - **Illustrates how much it can be leveraged from the artefacts and domain vocabulary when they correlate,**
  - **Demonstrate that bug localisation improves when domain vocabulary is used.**

Further research

Our study showed that

- **Despite good naming conventions and vocabulary coverage;**
  - **Challenging to find the classes referred by a CR.**
  - **sophisticated approaches fall short when CRs are terse.**

In the next step of our research

- **combine domain ontologies with existing natural language and call-graph techniques.**
  - **navigate the call-graph to discover additional program elements.**
  - **utilise domain ontologies to evaluate their relevance.**

Our aim

- **Construct the contextual model in ConCodeSe to provide consistent set of clues and aid program comprehension during maintenance.**