A Data Set of Extracted Rationale from Linux Kernel Commit Messages

Mouna Dhaouadi
3rd year PhD Candidate
Université de Montréal, Canada
http://www.labs.iro.umontreal.ca/~dhaouadm/
Motivation

For a decision, software/system developer has rationale ‘Why’ reasoning for their decision

Rationale is useful information to:
understand the system,
learn from mistakes,
reuse solutions,
avoid conflicts
Rationale in Commits

“Fix: Added check to prevent null pointer exception.”

versus

“changes”

Challenges:
- Implicit or unrecorded
- Ambiguous language
  - Especially past/future tenses
  - Non-native writers
- Need technical understanding
Research Questions

How does rationale appear? What are its characteristics?

How can we structure, extract and manage rationale information from code commits?
Proposed Contributions

• An empirical contribution to better understand rationale in open-source software (*FSE SRC 2023, Manuscript under review 2023*)

• An end-to-end rationale extraction and management system (*ASE NIER 2022, MDE-Intelligence 2023*)
Running example: OOM Killer subsystem

**Linux kernel:**
- Development is through Git commits
- Culture for motivating/describing changes
- Valuable (highest-quality) source of rationale

**Out-of-Memory Killer subsystem:**
- When Linux kernel runs out of memory, OOM-Killer is called to avoid crashing
- Two broad steps:
  1. Select “best” task to kill
  2. Force task to release memory and exit
- Meaningful heuristics
Research Question 1

How does rationale appear? What are its characteristics?
Dataset

Manual labelling of OOM-Killer commit sentences.

Procedure:
- Collect 418 commits
- Remove merge commits, filter code sentences
  404 commits / 2234 sentences
- Three annotators label sentences
- Six piloting rounds -> codebook + protocol
- Resolve conflicts in discussion

An example commit
Codebook

- Obtained through piloting rounds and discussions
- **Multiple classifications per sentence**
- In disagreement, take classification union
- Fleiss kappa: Around 0.66 (fair to good agreement)
## Labelling example 1

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Labelling</th>
</tr>
</thead>
<tbody>
<tr>
<td>mm, oom: introduce independent oom killer ratelimit state</td>
<td>Decision</td>
</tr>
<tr>
<td>printk_ratelimit() uses the global ratelimit state for all printks</td>
<td>Supporting Facts</td>
</tr>
<tr>
<td>The oom killer should not be subjected to this state just because another subsystem or driver may be flooding the kernel log</td>
<td>Rationale</td>
</tr>
<tr>
<td>This patch introduces printk ratelimiting specifically for the oom killer</td>
<td>Decision</td>
</tr>
<tr>
<td>Sentence</td>
<td>Labelling</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>tlbt mmu_gather: Remove start/end arguments from tlbt_gather_mmu()</td>
<td>Decision</td>
</tr>
<tr>
<td>The 'start' and 'end' arguments to tlbt_gather_mmu() are no longer needed now that there is a separate function for 'fullmm' flushing</td>
<td>Rationale, Supporting Facts</td>
</tr>
<tr>
<td>Remove the unused arguments and update all callers.</td>
<td>Decision, Rationale</td>
</tr>
</tbody>
</table>
Labelling insights

Substantial overlap between the categories

Distribution of the sentences in the OOM dataset
Labelling insights

- 98.9% of commits contain rationale
- About 60% of sentences per commit contain rationale
Labelling insights

- Most the commits have fewer than 15 sentences.

- A lot of the short commits (fewer than 6 sentences) have a high rationale density (>0.6).

- As a commit becomes longer, it tends to have between 40% to 60% of its sentences containing rationale information.
Labelling insights

- Only five developers have written more than 16 commits. All the other developers have written fewer than 16 commits, and most of them fewer than 10 commits.

- More experienced developers' commits have a consistent rationale density near 60%.

Commits per author versus average rationale density
Labelling insights

- Common Structure:
  Decisions -> Supporting Facts -> Rationale -> Decisions

Distribution of the categories over the normalized positions of the sentences of the commit messages
Research Question 2

How can we structure, extract and manage rationale information from code commits?
OOM-Killer example

- Manually selected interesting commits from the Git history of OOM-Killer.
- Topic of "reclaiming used memory from the OOM victim".

Suren’s Challenge:
How does my decision impact previously established decisions? How to make sure I will not cause conflicts with existing rationales?

End-to-End Rationale Reconstruction, Mouna Dhaouadi, Bentley James Oakes, Michalis Famelis. The IEEE/ACM International Conference on Automated Software Engineering (ASE), Michigan, USA, 2022. NIER Track
Proposed solution

- A complete rationale extraction and management system

Automated rationale extraction

- Kantara Pipeline
- Information Inference Component
- Decisions and Rationale Graph
- Rationale Structuring
- Validation Mechanisms
- Analysis Techniques
- Retrieval System
- Tool
- Response Generation Component

Rationale Extraction
Kantara: Information Extraction pipeline

- Decisions extraction
- Rationale extraction
- Relatedness relationship
- Similar relationship
- Contradicts relationship
- History relationship

- Classification
- Semantic Role Labeling
- Text Similarity
- Natural Language Inference
- Heuristics

End-to-End Rationale Reconstruction, Mouna Dhaouadi, Bentley James Oakes, Michalis Famelis. The IEEE/ACM International Conference on Automated Software Engineering (ASE), Michigan, USA, 2022. NIER Track
Prototype to evaluate feasibility of Kantara

• Evaluated on the Out-Of-Memory Killer (OOM-Killer) example
Rationale structuring

Kantara Pipeline

Information Inference Component

Response Generation Component

Decisions and Rationale Graph

Validation Mechanisms

Analysis Techniques

Retrieval System

Tool
Knowledge Graph

- Nodes: decisions and rationales
- Relationships: relatedness, similar, contradicts, history

"reduces the probability of such a lockup"

Decision #D4
Introduce OOM reaper.
Date = March 25th, 2016

"memory is freed in a more controllable way with CPU affinity and priority of the caller"

Decision #D5
Introduce process_mrelease system call that releases memory of a dying process from the context of the caller.
Date = September 2nd, 2021

"so that it may quickly exit and free its memory"

Decision #D2
give current access to memory reserves if it's trying to die.
Date = March 23rd, 2011

"Eventually, kernel may hang up when oom kill occur"

Decision #D3
I and the original author Luis agreed to disable this logic.
Date = April 14th, 2011

"so that it can exit() soon, freeing memory"

Decision #D1
give the dying task a higher priority.
Date = August 9th, 2010

End-to-End Rationale Reconstruction, Mouna Dhaouadi, Bentley James Oakes, Michalis Famelis. The IEEE/ACM International Conference on Automated Software Engineering (ASE), Michigan, USA, 2022. NIER Track
Rationale management

Kantara Pipeline

Information Inference Component

Decisions and Rationale Graph

Validation Mechanisms

Analysis Techniques

Retrieval System

Tool

Rationale Extraction

Rationale Structuring

Response Generation Component
Look for potential conflicts when a new decision is proposed.

**Decision #D1**
- give the dying task a higher priority.
- Date = August 9th, 2010

**Decision #D2**
- give current access to memory reserves if it's trying to die.
- Date = March 23rd, 2011

**Decision #D3**
- I and the original author Luis agreed to disable this logic.
- Date = April 14th, 2011

**Decision #D5**
- Introduce process_mrelease system call that releases memory of a dying process from the context of the caller.
- Date = September 2nd, 2021

---

*Suren’s Challenge:*
How does my decision impact previously established decisions? How to make sure I will not cause conflicts with existing rationales?
Automatic rationale extraction

- Need ground truth for the automatic classification
Automatic sentence classification: Initial results

180 commits from the dataset we created (Linux kernel subsystem)

Binary classification: Logistic regression, decision tree, SVM
Multi-label classification: Random Forest, XGBoost, KNN

<table>
<thead>
<tr>
<th>Label</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision</td>
<td>0.76</td>
<td>0.69</td>
<td>0.72</td>
</tr>
<tr>
<td>Rationale</td>
<td>0.62</td>
<td>0.41</td>
<td>0.49</td>
</tr>
<tr>
<td>Supporting Facts</td>
<td>0.64</td>
<td>0.68</td>
<td>0.66</td>
</tr>
</tbody>
</table>
Conclusion and Future work

Summary:
• An empirical contribution to better understand rationale in-the-wild
• A rationale extraction and management framework
  Linux Kernel as object of the study

Future work:
• Improve automatic classification
• Finish implementing the framework
• Provide a tool (e.g., a pull request bot)

This work is partially supported by the ACM-W scholarship program and Google.