FEATURE MAINTENANCE WITH EMERGENT INTERFACES

Márcio Ribeiro(UFAL), Paulo Borba(UFPE) and Christian Kästner(Carnegie Mellon University)

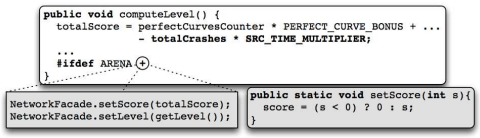
Introduction

- Developers often introduce errors into software systems when they fail to recognize module and feature dependencies;
- This problem is particularly critical for configurable systems, in such context, developers can easily miss cross-feature dependencies;
- In a prior study we found that cross-feature dependencies are frequent in practice;
- To reduce this feature-dependency problem, we propose a technique called emergent interface;

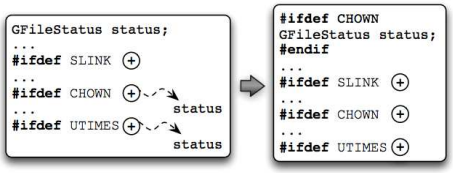
Maintaining Product Lines

- Maintaining software product lines are challenging because code fragments are configurable and may not be included in all product configurations;
- Developers need to reason about potentially different control and data flows in different configurations;
- Rapid feedback by testing is often too expensive or even not possible for all products;

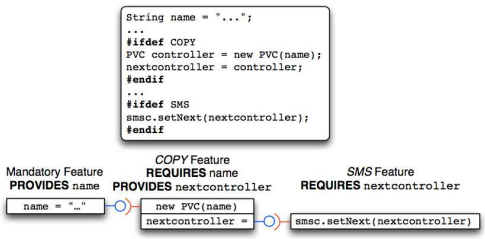
Scenario 1: Implementing a New Requirement



Scenario 2: Fixing an Unused Variable

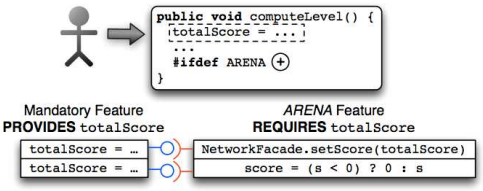


Emergent Interfaces (1/2)



*Inferring Emergent Interfaces on Demand

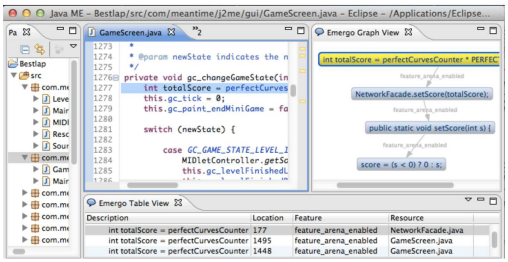
Emergent Interfaces (2/2)



*Abstracting Feature Code

*Avoiding Invalid Dependencies

Emergo



Experimental Design

- Goal
  - Compare maintenance of preprocessor-based product lines with and without emergent interfaces
- Questions
  - Do emergent interfaces reduce effort during code-change tasks involving feature-code dependencies in preprocessor-based systems?
  - Do emergent interfaces reduce the number of errors during code-change tasks involving feature-code dependencies in preprocessor-based systems?
- Metrics
  - Time
  - Incorrect solutions

Participants

- Pilot:
  - 6 graduate students
- Round 1:
  - 10 graduate students
- Round 2:
  - 14 undergraduate students
- ~50% were part-time students with professional experience in both rounds

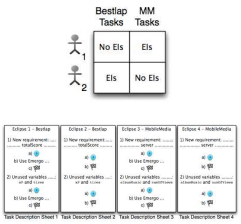
Material and Code-Change Tasks

- Two preprocessor-based product lines as experimental material:
  - Best Lap (15KLOC)
  - MobileMedia (3KLOC)
- Tasks:
  - New requirement for Best Lap;
  - New requirement for MobileMedia;
  - Unused variable in Best Lap;
  - Unused variable in MobileMedia.

Hypotheses

- H1(Effort) - With emergent interfaces, developers spend less time to complete code-change tasks involving feature dependencies in both product lines.
- H2(Error Introduction) - With emergent interfaces, developers commit less errors in both product lines when performing code-change tasks involving feature dependencies

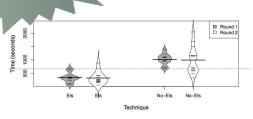
Design and Procedure

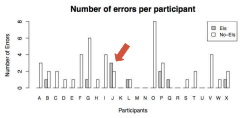


- Eclipse preparation with Emergo and Time Plugins
- Partially automated measuring number of errors
- For unused-variables they should write the response in the sheets

Results: New-Requirement Tasks

Statistically significant

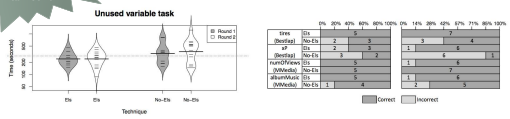




P-value: 2.237e-05
P-value: 5.343e-05

Results: Unused-Variable Tasks

Statistically significant at Round 2



P-value: 0.1306
P-value: 0.016

Threats to Validity

1. Limited to specific implementation technique and specific code-change scenarios
2. Tasks are too simple
3. Recruiting students instead of professionals
4. MobileMedia is a small product line
5. Emergent interfaces depend on data-flow analysis, which can be expensive to perform.
6. The time penalty added to wrong answers for the unused-variable

Conclusion

- The study presents:
  - Emergent Interfaces technique
  - Emergo, a tool capable of inferring interfaces from data-flow analysis on demand.
- Conducted a controlled experiment to evaluate what extent such tool support can help achieving better feature modularization
- The study also suggest that using this technique is possible to decrease effort and errors made in code-change.