Integrating Reuse into the Rapid, Continuous Software Engineering Cycle through Test-Driven Search

Position Paper @ RCoSE - 4th International Workshop on Rapid Continuous Software Engineering
Gothenburg, Sweden

Marcus Kessel and Colin Atkinson
E-Mail: kessel@informatik.uni-mannheim.de
## Software Reuse vs Agile Practices

**Goal:** Delivery of high-quality software in short time frames with minimal amount of effort

<table>
<thead>
<tr>
<th>Software Reuse</th>
<th>Agile Practices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exploits existing domain knowledge</td>
<td>Short, iterative and incremental development cycles</td>
</tr>
<tr>
<td><strong>Preplanned:</strong> like software product lines</td>
<td>support a wide range of development activities</td>
</tr>
<tr>
<td>→ high upfront costs</td>
<td><strong>Unanticipated:</strong> like ad hoc and/or pragmatic reuse</td>
</tr>
<tr>
<td><strong>Unanticipated:</strong> like ad hoc and/or pragmatic reuse</td>
<td>primarily focus on from-scratch development</td>
</tr>
<tr>
<td>→ high risk, low ROI</td>
<td></td>
</tr>
</tbody>
</table>

*Source:* [Univ of Mannheim Software Engineering Group](http://www.univ-mannheim.de)
**Software Reuse vs Agile Practices**

Goal: Delivery of high-quality software in short time frames with minimal amount of effort

**Software Reuse** vs **Agile Practices**

- **Reuse == Discontinuous?**
- **How can we integrate it into agile practices?**

**Software Reuse**

- Exploits existing domain knowledge
- Preplanned: like software product lines → high upfront costs
- Unanticipated: like ad hoc and/or pragmatic reuse → high risk, low ROI

**Agile Practices**

- Short, iterative and incremental development cycles
- Support a wide range of development activities
- Primarily focus on from-scratch development

**Goal:** Delivery of high-quality software in short time frames with minimal amount of effort
Rapid, Continuous Code Reuse

- CoSE has the potential to promote unanticipated reuse
  - domain knowledge contained in software repositories
  - artifacts produced by agile practices like Test-Driven Development (TDD) and Continuous Integration (CI)

- Rapid, continuous reuse of software components
  - exploit the increasing use of tests in automated CI
    → by test-driven search technology (TDS)

- Goal: Complement from-scratch development
  - increase efficiency in agile projects
  - (unanticipated) reuse and CI are synergistic and can effectively complement each other
Semantic Code Search: Test-Driven Search

1) Create Unit Tests

2) query

Test-Driven Search Engine

Component index

3) passing candidates

Ranking Engine

Developer (reuser)

Base64 Example

Signature

Base64(encode(char[]):char[]);

Test Cases (Input/Output)

e.g. RCoSE2018 → UkNvU0UyMDE4

4) explore, rank & select

"exec specification"
Enabler: Test-Driven Search Technology

1) Create Unit Tests

2) query

3) passing candidates

4) explore, rank & select

Developer (reuser)
Enabler: Test-Driven Search Technology

1) Create Unit Tests

2) query

3) passing candidates

Test-Driven Search Evaluation
- Retrieve textual candidates
- Run candidates against test class

4) explore, rank & select

Developer (reuser)

“exec specification“
Enabler: Test-Driven Search Technology

1) Create Unit Tests

2) query

"exec: specification"

3) passing candidates

Developer (reuser)

4) explore, rank & select

Ranking according to reuse utility
- Functional requirements
- Non-functional requirements
Enabler: Test-Driven Search Technology

1) Create Unit Tests

```
public class DemoTest {
    public void testFunction() throws Exception {
        // Test code
    }
}
```

2) Query

Test-Driven Search Engine

Component index

3) Passing candidates

Explore recommended reuse candidates

View
- source code
- Metrics (static/dynamic)
- Metadata ...

4) Explore, rank & select

Ranking Engine
Test-Driven Search: merobase prototype platform

1) Test Specification Editor

JUnit Test Class

Filters

2) Explore Passed Candidates

Test-Driven Search Engine (merobase web service)

Goal-oriented Ranking Criteria
Test-Driven Search Integrated into CI Cycle

1. Check in Changes
2. Fetch Changes (Trigger)
3. Build
4. Test Report Feedback
5. Fail or Success
6. Notify Success / Failure + Reuse Recommendations
7. Select, Adapt, Integrate Component Candidate

- Return Candidates (Ranked)
- Evaluate
- Database (Index)
- Index Code units
- (External) Software Repositories
- (Internal) Software Repositories
- Source Code Management (Version Control System)

Manager, Developer 1, Developer ...

Continuous Integration Platform
Search Plugin
Other

Test-Driven Search Platform

Software Engineering Group
Scenarios: Test-First Development

- **Reactive reuse**
  - unobtrusive/no change for developers

- **Proactive reuse**
  - optimize for test-driven reuse

```
  Add Test(s) ->
  Refactor Code
  Run all Tests / Fail? ->
  Trigger Search
  Continuous Integration & Test-Driven Search

  Reuse Successful? If not, revert to prior version
  Run Tests
  Write the Code
  Adapt, Integrate (or Learn)

  Inspect & Select Candidate
```

Software Engineering Group
1a) Dashboard Reporting (e.g. Jenkins CI)

1b) E-Mail Notifications

2) Explore & Select Reusable Candidates
Challenges of Rapid Reuse

- Software reuse
  - cover all reuse phases: select, adapt and integrate [Krueger 92]
  - specify both desired functional/non-functional requirements [Kessel 16]

- Agile teams: Principles & policies
  - violated?
  - see: „Simplicity--the art of **maximizing the amount of work not done**--is essential.” [Agile Manifesto]

- Triggering: Scalable and efficient strategies for initiating test-driven searches in CI; identify --
  - how developers practice TDD in CI environments
  - when and what has changed (on commit level)
  - what is being tested (CUTs) and how is it tested? (e.g. unit vs integration tests)
Thank you for your Attention!
Future Work: Next Steps

- Analysis of code version histories to identify testing strategies
  - Mining software repositories
    - TravisTorrent dataset [TravisTorrent]
  - Ask developers ...

- Start with unobtrusive reactive reuse
  - causes no change for developers
  - „replay“ commits from existing projects
  - let test-driven search engine run in the background
Literature