Using Interaction Logs for
Creation and Maintenance of Trace Links

Paul Hübner and Barbara Paech
Institute for Computer Science
Heidelberg University
Traceability – How to create Trace Links?

**Issue Tracker Context**

- Issue ID in commit messages are used to link code with issues
- only ~ 60% of all commits contain issue IDs
- No link maintenance

**Safety Critical & Verification Context**

- Many wrong links to get (almost) all correct link, manual vetting necessary
- Typical *recall* = “completeness” of over 80% and *precision* = “correctness” below 30%
- No link maintenance

IR Technique: calculate textual similarity and link similar artefacts
Motivation: Why Do We Need Another Trace Link Creation Approach?

• Our application context for trace links
  – Continuous and direct usage of trace links during development
  – Maintenance of links along with changes in linked artefacts
  – Unstructured requirements in an issue tracking system (ITS)
  – Source code in a version control system

• Problems of existing approaches and challenges for our approach
  – Use of trace links during the development requires good precision of trace links
  – Wrong links will prevent developers from using the links \(\rightarrow\) optimize precision
  – Existing approaches focus on recall optimization, bad precision is tolerated
  – Maintenance of links is often not considered
Interaction Recording for Trace Link Creation

- Record developer’s interactions with files in an IDE
- Assign interactions to a requirement
- Create trace links between the files from the interactions and the requirement

Developer’s IDE

Recorded Interactions:
01: 2018-02-04T15:33:42.270+02:00;Dev1;Action;$EditorEnter;ise;Hierarchie;/src/ise/InfoBoxManager.java
02: 2018-02-04T15:34:11.090+02:00;Dev1;Action;EditorBackSpace;ise;Editor;/src/ise/MainActivity.java

Interaction to Requirement assignment
Approach $IL_{Com}$ – Interaction Assignment

- $IL_{Com}$: use recorded interactions and issue IDs in commit messages
- No ID in commit: continue recording and assign to next commit with ID

Commit message with issue id:

commit b7deb13f94bf3b7b55f913aee675c704809ae8f8b
Author: Developer <dev@mail.org> 2018-02-04 21:54
Committer: Developer <dev@mail.org> 2018-02-04 21:54
Branches: master, origin/design, origin/develop

Remove focus whenever closing info box

Recorded Interactions:
01: 2018-02-04T15:33:42.270+02:00;Dev1;Action;$EditorEnter;ise;Hierarchie; /src/ise/InfoBoxManager.java
02: 2018-02-04T15:34:11.090+02:00;Dev1;Action;EditorBackSpace;ise;Editor; /src/ise/MainActivity.java

No ID in commit, interaction recording continues without assignment
**IL_{Com} Approach – Overview**

1. **Capture Interactions** during the Implementation of Requirements

   - **Requirements (ITS)**
   - **Interaction Log**
   - **Trace Links**
   - **Trace Links**

2. **Trace Link Creation**
   - **Event Types**
   - **Duration**
   - **Frequency**
   - **Interaction Log Aggregation**

3. **Trace Link Improvement**
   - **Source Code Structure Generation**
   - **Recall + Precision Improvement**

   - **Impl. Artifact (VCS)**
   - **Impl. as Plug-ins For Eclipse and IntelliJ**
   - **Implemented as Python NLTK based Tool**

**Assign interactions to requirement**

**Implemented as Python NLTK based Tool**

**GI FG-RE 2019**

Using Interaction Logs for Creation and Maintenance of Trace Links
**IL_{Com} Approach – Trace Link Creation**

**Interaction Log**

10:14:50.910;Edit;Editor;/git/Controller.js;  
10:1:  
10:1;  
10:1: **Recorded and assigned interaction log**  
10:14:44.100;Select;Navigator;/git/Controller.js;  
10:14:43.100;Edit;Editor;/git/Controller.js;  
10:14:42.100;Select;Navigator;/git/Controller.js;  
10:08:26.414;Task Activation;ISE2016-46;  

**Event Type-based aggregation**

**Edit (EVENT TYPE) + Editor + Controller.js**

10:14:50.910;Edit;Editor;/git/Controller.js;  

**Select (EVENT TYPE) + Navigator + Controller.js**

10:14:44.100;Select;Navigator;/git/Controller.js;  
10:14:42.100;Select;Navigator;/git/Controller.js;  

**Edit (EVENT TYPE) + Editor + Manager.js**

10:14:47.910;Edit;Editor;/git/Manager.js;  

**Select (EVENT TYPE) + Editor + Manager.js**

10:14:45.910;Select;Editor;/git/Manager.js;  

**Select (EVENT TYPE) + Navigator + Manager.js**

10:14:48.910;Select;Navigator;/git/Manager.js;  

**Trace Link Candidates**

**Controller.js (FILE)**

10:14:50.910;Edit;Editor;/git/Controller.js;  
10:1;  
10:1; **File-based aggregation**

10:14:42.100;Select;Navigator;/git/Controller.js;  
10:14:48.910;Select;Navigator;/git/Manager.js;  
10:14:47.910;Edit;Editor;/git/Manager.js;  
10:14:46.100;Edit;Editor;/git/Manager.js;  
10:14:45.910;Select;Editor;/git/Manager.js;  

**IDE Part-based aggregation**

**Editor (IDE PART) + Controller.js**

10:14:50.910;Edit;Editor;/git/Controller.js;  
10:1;  
10:1  

**Editor (IDE PART) + Controller.js**

10:14:44.100;Select;Navigator;/git/Controller.js;  
10:14:42.100;Select;Navigator;/git/Controller.js;  

**Editor (IDE PART) + Manager.js**

10:14:47.910;Edit;Editor;/git/Manager.js;  
10:14:46.100;Edit;Editor;/git/Manager.js;  
10:14:45.910;Select;Editor;/git/Manager.js;  

**Navigator (IDE PART) + Controller.js**

10:14:44.100;Select;Navigator;/git/Controller.js;  
10:14:42.100;Select;Navigator;/git/Controller.js;  

**Navigator (IDE PART) + Manager.js**

10:14:48.910;Select;Navigator;/git/Manager.js;  

**Select (EVENT TYPE) + Editor + Manager.js**

10:14:45.910;Select;Editor;/git/Manager.js;  

**Select (EVENT TYPE) + Editor + Controller.js**

10:14:47.910;Select;Editor;/git/Controller.js;  
10:14:46.100;Edit;Editor;/git/Controller.js;  
10:14:45.910;Select;Editor;/git/Manager.js;  

**Recorded and assigned interaction log**

10:08:26.414;Task Activation;ISE2016-46;
**IL\textsubscript{Com} Approach – Improvement Techniques**

<table>
<thead>
<tr>
<th>Trace Links – Duration</th>
<th>Trace Links – Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="ISE2016-46/git/Controller.js;3.000;3;Edit;Editor;" alt="Example" /></td>
<td><img src="ISE2016-46/git/Controller.js;3.000;3;Edit;Editor;" alt="Example" /></td>
</tr>
<tr>
<td><img src="ISE2016-46/git/Controller.js;1.190;2;Select;Navigator;" alt="Example" /></td>
<td><img src="ISE2016-46/git/Controller.js;1.190;2;Select;Navigator;" alt="Example" /></td>
</tr>
<tr>
<td><img src="ISE2016-46/git/Manager.js;2.000;2;Edit;Editor;" alt="Example" /></td>
<td><img src="ISE2016-46/git/Controller.js;1.190;2;Select;Navigator;" alt="Example" /></td>
</tr>
<tr>
<td><img src="ISE2016-46/git/Manager.js;1.810;1;Select;Editor;" alt="Example" /></td>
<td><img src="ISE2016-46/git/Manager.js;1.810;1;Select;Editor;" alt="Example" /></td>
</tr>
<tr>
<td><img src="ISE2016-46/git/Manager.js;1.000;1;Select;Navigator;" alt="Example" /></td>
<td><img src="ISE2016-46/git/Manager.js;1.000;1;Select;Navigator;" alt="Example" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Trace Links – Event Type</th>
<th>Trace Links – IDE Part</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="ISE2016-46/git/Controller.js;3.000;3;Edit;Editor;" alt="Example" /></td>
<td><img src="ISE2016-46/git/Controller.js;3.000;3;Edit;Editor;" alt="Example" /></td>
</tr>
<tr>
<td><img src="ISE2016-46/git/Controller.js;1.190;2;Select;Navigator;" alt="Example" /></td>
<td><img src="ISE2016-46/git/Controller.js;1.190;2;Select;Navigator;" alt="Example" /></td>
</tr>
<tr>
<td><img src="ISE2016-46/git/Manager.js;2.000;2;Edit;Editor;" alt="Example" /></td>
<td><img src="ISE2016-46/git/Manager.js;2.000;2;Edit;Editor;" alt="Example" /></td>
</tr>
<tr>
<td><img src="ISE2016-46/git/Manager.js;1.810;1;Select;Editor;" alt="Example" /></td>
<td><img src="ISE2016-46/git/Manager.js;1.810;1;Select;Editor;" alt="Example" /></td>
</tr>
<tr>
<td><img src="ISE2016-46/git/Manager.js;1.000;1;Select;Navigator;" alt="Example" /></td>
<td><img src="ISE2016-46/git/Manager.js;1.000;1;Select;Navigator;" alt="Example" /></td>
</tr>
</tbody>
</table>
IL\textsubscript{Com} Approach – Improvement Techniques

Source Code Structure-based

- **Precision**
  - Link R1 → C1 wrong?
  - Link R1 → C6 wrong?

- **Recall**
  - New correct links R1 → C8 & R1 → C9?
### IL\textsubscript{Com} Approach – Evaluation Studies Results

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Approach</th>
<th>Precision</th>
<th>Recall</th>
<th>#Links</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CE</td>
<td>TP</td>
<td>FP</td>
</tr>
<tr>
<td>M2007</td>
<td>IL\textsubscript{i}</td>
<td>1.000</td>
<td>0.929</td>
<td>2565</td>
</tr>
<tr>
<td></td>
<td>IR\textsubscript{i}</td>
<td>0.386</td>
<td>0.440</td>
<td>3143</td>
</tr>
<tr>
<td>M2012</td>
<td>IL\textsubscript{i}</td>
<td>1.000</td>
<td>0.800</td>
<td>1126</td>
</tr>
<tr>
<td></td>
<td>IR\textsubscript{i}</td>
<td>0.283</td>
<td>0.557</td>
<td>2766</td>
</tr>
<tr>
<td>S2017</td>
<td>IL\textsubscript{i}</td>
<td>0.682</td>
<td>0.465</td>
<td>148</td>
</tr>
<tr>
<td></td>
<td>IL\textsubscript{Com_i}</td>
<td>0.736</td>
<td>0.539</td>
<td>159</td>
</tr>
<tr>
<td></td>
<td>IR\textsubscript{i}</td>
<td>0.351</td>
<td>0.217</td>
<td>134</td>
</tr>
<tr>
<td>S2018</td>
<td>IL\textsubscript{Com_i}</td>
<td>0.900</td>
<td>0.790</td>
<td>271</td>
</tr>
<tr>
<td></td>
<td>IR\textsubscript{i}</td>
<td>0.369</td>
<td>0.557</td>
<td>466</td>
</tr>
<tr>
<td></td>
<td>ComL\textsubscript{i}</td>
<td>0.675</td>
<td>0.443</td>
<td>203</td>
</tr>
</tbody>
</table>

- **Perfect precision** → no wrong links
- **Almost perfect precision and good recall**
- **Even in the best case** 1/3 of the links wrong & low recall
$IL_{Com}$ Approach – Trace Link Maintenance

- Trace links have to be maintained along with changes in linked artefacts
  - Otherwise the links become useless
  - Maintenance of links is often not considered in link creation approaches
- Literature Review of existing Trace Link Maintenance (TM) Approaches
  - Generic TM process, for a standardized description of reviewed approaches
  - 16 TM approaches were reviewed
- Identification of TM approaches suitable for integration in $IL_{Com}$
  - Two approaches were selected since:
    - Same data sources (requirements, source code) are used
    - Fully automated $\rightarrow$ no interruption / additional effort for developers
Source Code Structure-based Trace Link Maintenance

[Ghabi 2012]

- Idea: Use the source code structure to check whether
  - Existing links are still valid and if
  - Links are missing
  - Similar to \( IL_{Com} \)'s source code structure (SCS)-based improvement techniques

- 16 rules (patterns) using existing links and source code
  - Judge the proximity of code classes and methods to requirements
  - Output a numerical value for each artefact pair
  - Used to judge existing/ missing links

- Integration in \( IL_{Com} \)
  - Supplement/ replace \( IL_{Com} \)'s SCS-based impr. Techniques
  - Partial integration: evaluate which rules perform best

Add new link \( R1 \rightarrow C2 \) since surrounding classes in call chain are already linked

Rule example “Call chain”:
Refactoring-based Trace Link Maintenance

[Rahimi 2018]

• Idea: Detect refactorings and perform refactoring specific link maintenance
  – Refactorings are detected using 2 consecutive versions of source code files/ req.
  – Refactoring specific rules maintain the existing links accordingly
  – E.g. for refactoring “extract method to class” existing links are moved to new class

• Two options for integration in $IL_{Com}$
  1. As in the original implementation:
     • Use 2 consecutive versions and text diff.
  2. Detect refactorings in recorded interactions by:
     • Directly used refactoring actions of IDEs
     • Sequences of low level interactions comprising a refactoring

2. option using interaction data is likely to perform better, since the original implementation relies on textual differences between two artefact versions
Conclusion and Outlook

• Interaction recordings of developers can be used to create trace links with high precision and good recall.
• Awareness for interaction logging and required effort for interaction assignment influence the resulting link quality.
• $IL_{Com}$ approach:
  – Removes additional efforts for developers and
  – Countervails insufficient precision and recall with improvement techniques
  – Integrates trace link maintenance capabilities due to:
    • The usage of source code structure and
    • Interaction recordings
• Next: Application of $IL_{Com}$ with link usage and maintenance.
Thank you for your attention!

Questions ... ???
More
References


IL\textsubscript{Com} Approach – Evaluation Studies Overview

<table>
<thead>
<tr>
<th>Study intent</th>
<th>1. Eval. Study</th>
<th>2. Eval. Study</th>
<th>3. Eval Study</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mylyn 2007</td>
<td>Mylyn 2012</td>
<td>Student 2017</td>
</tr>
<tr>
<td>Study intent</td>
<td>Initial Evaluation</td>
<td>Improvement techniques</td>
<td>Commit-based assignment</td>
</tr>
<tr>
<td>Project Organization</td>
<td>Open Source</td>
<td>Scrum</td>
<td>Scrum</td>
</tr>
<tr>
<td># developers</td>
<td>19</td>
<td>20</td>
<td>7</td>
</tr>
<tr>
<td>Requirements ITS</td>
<td>Bugzilla</td>
<td>Jira</td>
<td>Jira</td>
</tr>
<tr>
<td># / Format</td>
<td>50 Features</td>
<td>50 Features</td>
<td>19 User Stories</td>
</tr>
<tr>
<td>Source Code VCS</td>
<td>Git/ SVN (Java, XML)</td>
<td>Git (JavaScript)</td>
<td>Git (Java, XML)</td>
</tr>
<tr>
<td># Files</td>
<td>756</td>
<td>2119</td>
<td>91</td>
</tr>
<tr>
<td>Interaction rec. IDE</td>
<td>Eclipse</td>
<td>IntelliJ/ Webstorm</td>
<td>Eclipse/ ADT</td>
</tr>
<tr>
<td># / Asg. type</td>
<td>7687 (man.)</td>
<td>1660 (man.)</td>
<td>1171290 (man)</td>
</tr>
</tbody>
</table>
\( IL_{Com} \) Approach – Evaluations Experimental Design

Research Questions:

1. **Precision (P)** and **recall (R)** of \( IL_{Com} \) created trace links?
2. **P** and **R** of trace links created by **other techniques**?

---

**Gold Standard Creation** + improvement techniques

- IR (VSM, LSI), Commit only (ComL)
IL\textsubscript{Com} Approach – Generic Trace Link Maintenance Process

**Impact Detection**

1. Detection of Change
   - *Manual indication*
   - *Monitoring of artefact changes*

2. Detection of Impacted Links
   - *Manual indication*
   - *Monitoring with impact detection rules*

**Rule Data**

- *Model element type*
- *Interaction type & sequence*
- *Model element hierarchy*
- *2 artefact versions*
- *Term Pairs*
- *Software artefact type*
- *Source code structure*

**Change Execution**

3. Determination of Necessary Link Change
   - *Change type*
   - *Manual change of links*
   - *Tool-based link change*

4. Execution of Change
   - *Change type-specific rule execution*
   - *Manual use of output*

**Rule Data**

- *Model element type*
- *Model element hierarchy*
- *Linked artefact score threshold*
- *Software artefact type*