Mining Software Repositories for Traceability Links

Software Development Laboratory
<SDML>

Huzefa Kagdi
Jonathan I. Maletic
Bonita Sharif

KENT STATE UNIVERSITY

<SDML>

Research Goal

Uncover/discover traceability links between source code and other types of software artifacts using a Mining Software Repositories (MSR) type of approach

KENT STATE UNIVERSITY

<SDML>

Merits of a Change-Based Approach

- Actual changes to artifacts rather than estimations
- The practice of commits and commit messages embody part of the developer's knowledge and experience
- We feel that version history may contain domain-specific "hidden" links that program-analysis methods may fail to uncover

KENT STATE UNIVERSITY

<SDML>

A KDE Subversion Commit

Metadata

```
r472031 | bmeyer | 2005-10-19 11:14:47
    -0400 (Wed, 19 Oct 2005) | 1 line
Changed paths:
   M   /trunk/KDE/kdegames/ktron/
    ktron.cpp
   M    /trunk/KDE/kdegames/ktron/
    main.cpp

move scope
```

Differences

```
ktron.cpp r472030:r472031

--- ktron.cpp (revision 472030)
+++ ktron.cpp (revision 472031)
@@ -27,7 +27,6 @@
 #include <kmessagebox.h>
 #include <kaction.h>
 #include <kstdgameaction.h>
-#include <kapplication.h>
 #include <kstatusbar.h>

 // Settings
```

Change-set

{ktron.cpp, main.cpp}

KENT STATE UNIVERSITY

<SDML>

Traceability of Interest

- Uncovering traceability between source code and other artifacts*:*
  - User manuals, *e.g., HTML* and *XML/docbook*
  - Build management, *e.g., automake* and *makefile*
  - HowTo guides, *e.g., FAQs*
  - Software distribution, *e.g., ChangeLogs* and *README*
  - Progress monitoring, *e.g., TODO* and *STATUS*

- *Software Informalisms* in open source systems [Scacchi'02]

KENT STATE UNIVERSITY

<SDML>

Examples: Traceability Links

- *{kalzium.cpp, pse.cpp} → {index.docbook}*
- *{kalziumtip.cpp} → {detailinfodlg.cpp} → {Makefile.am} → {kalzium.cpp, kalzium.h}*
- *{TODO} → {pse.cpp}*
- Notations
  - {..,..,..} files in the same change-set/revision
  - → change-set commit order

KENT STATE UNIVERSITY

<SDML>

Refining the Problem

- Is the presence of different types of documents in a single change-set sufficient to infer traceability links between them?
- How do we account for related documents with potential traceability links committed in a series of change-sets?

KENT STATE UNIVERSITY

<SDML>

Hypothesis

If a group of specific artifacts (of different types) are co-changed together, *frequently*, then there is a high probability that they have a traceability link between them

KENT STATE UNIVERSITY

<SDML>

Our Approach

- Heuristics are used to group potentially related change-sets
- Sequential-pattern mining [Agrawal'92] analyzes related change-sets to mine for highly frequent co-occurring changed files - **change patterns**
- Change patterns are then analyzed to uncover patterns that contain source code files and other types of files - **traceability patterns**

KENT STATE UNIVERSITY

<SDML>

Heuristics based on Metadata

- Time Interval
  - Change-sets committed in a given time duration are placed in a single group
- Committer
  - Change-sets committed by a given committer are placed in a single group
- Committer + Time Interval
  - Change-sets committed by the same committer within the same time interval are placed in a single group

KENT STATE UNIVERSITY

<SDML>

Mining Ordered Patterns

- Related change-sets are grouped based on a grouping heuristic
- Ordering of change-sets in a group is based on their revision numbers
  - Change-sets with larger revision numbers occur after those with lesser revision numbers
- Sequential Pattern Discovery Algorithm (SPADE) [Zaki'02] is applied to mine frequent partial sequences of changed files from a set of groups [Kagdi'06]
  - Each group of related change-sets forms an transaction
  - Each change-set forms an event

KENT STATE UNIVERSITY

<SDML>

Examples: KDE Traceability Patterns

- Found in five days
  - *{kalzium.cpp, pse.cpp} → {index.docbook}*
- Repeated by five different developers
  - *{kalziumtip.cpp} → {detailinfodlg.cpp} → {Makefile.am} → {kalzium.cpp, kalzium.h}*
- Ten different developer-day combinations
  - *{TODO}→{pse.cpp}*

KENT STATE UNIVERSITY

<SDML>

Evaluation

- *KDE* (K Desktop Environment) is used as a subject system
- First mine a portion of the version history for traceability patterns - *training-set*
- Next mine a later part of the version history for traceability patterns - *evaluation-set*
- Assess how accurately the candidates generated from the training-set predict changes that occur in the evaluation-set

KENT STATE UNIVERSITY

<SDML>

Change and Traceability Patterns

Training-set

Heuristics	CP	TP	TP/CP%
Day	5,839	1,851	37.10
Committer	718	54	7.52
CommitterDay	2,372	277	11.68

Evaluation-set

Heuristics	CP	TP	TP/CP%
Day	1112	143	12.86
Committer	304	26	8.55
CommitterDay	835	8	0.9

KENT STATE UNIVERSITY

<SDML>

Assessment Metrics

- Coverage
  - The percentage of traceability patterns in the evaluation-set for which there is at least one candidate (correct or incorrect) recommended from the training-set
- Recall
  - The percentage of traceability patterns in the evaluation-set that are correctly recommended from the training-set
- Precision
  - The percentage of correct candidates suggested after a change in a change-set of a covered pattern
  - Minimum, maximum, and average for the evaluation-set

KENT STATE UNIVERSITY

<SDML>

Coverage, Recall, and Precision Results

Heuristic Queries	Coverage (%)	Recall (%)
Day	44	8
Committer	57	11
CommitterDay	25	25

Heuristic Queries	Precision (%)		
	Min	Max	Avg
Day	50	100	63
Committer	55	75	67
CommitterDay	100	100	100

KENT STATE UNIVERSITY

<SDML>

Related Work

- Traceability between artifacts in bug repositories and source code [Canfora'06, Cubranic'05, Sliwerski'05]
- Evolutionary couplings of source code [Gall'98, Zimmermann'04, Yang'04, German'04, Kagdi'06]
- Approaches typically analyzing a single software version for recovering traceability links e.g., Antoniol, Cleland-Huang, Egyed, Hayes, Marcus, Spanoudakis and Zisman

KENT STATE UNIVERSITY

<SDML>

Conclusions

- A heuristic based approach that uses frequent-pattern mining for mining traceability links was presented
- Showed on KDE that these traceability links can be uncovered and used with high precision
- This work compounded with existing approaches expands the area of traceability link recovery

KENT STATE UNIVERSITY

<SDML>

Future Work

- Additional heuristics for grouping related change-sets such as textual similarity of commit messages
- Fine-grained traceability link (e.g., class and method levels) using *srcML*
- Integrating our tools into Subversion

KENT STATE UNIVERSITY

<SDML>

MSR Survey

Kagdi, Collard, Maletic

JSME March/April 2007

MSR in the context of software maintenance – 56 pages