COMP0104 Software Development Practice: Mining Software Repositories

Jens Krinke

Centre for Research on Evolution, Search & Testing
Software Systems Engineering Group
Department of Computer Science
University College London

Software Repositories

- Software repositories are record-keeping databases that store artifacts together with metadata about the artifacts.
- The artifacts in software repositories are created by software developers and other stakeholders during the development.
- Software repositories contain a wealth of valuable information about software projects.

Mining Software Repositories

- “The Mining Software Repositories (MSR) field analyzes the rich data available in software repositories to uncover **interesting** and **actionable** information about software systems and projects.”
- See msrconf.org
- Mining Software Repositories applies data mining to software data to extract useful information.

Aims

For research:

- Gain empirically-based understanding of software development.

For practitioners:

- Predict, plan, and understand various aspects of a project.
- Support future development and project management activities

Software Repositories

- **Historical Repositories** record information about the evolution and progress of a project.
- Examples: version control systems, issue trackers, mailing list archives, etc.
- **Run-Time Repositories** record information about the execution of a project, locally or deployed.
- Examples: crash logs, build logs, etc.

Software Repositories

- **Code Repositories** contain large number of independent software projects.
- Example: Github
- Other Software Repositories:
  - Development collaboration sites (StackOverflow)
  - App stores (apps and reviews)

Purpose of Mining Repositories (Examples)

- Understanding Software Systems

  Software repositories serve as a memory of software development projects.

- Predicting and Identifying Bugs

  Software repositories can relate source code properties to later discovered bugs.

- Uncover hidden dependencies

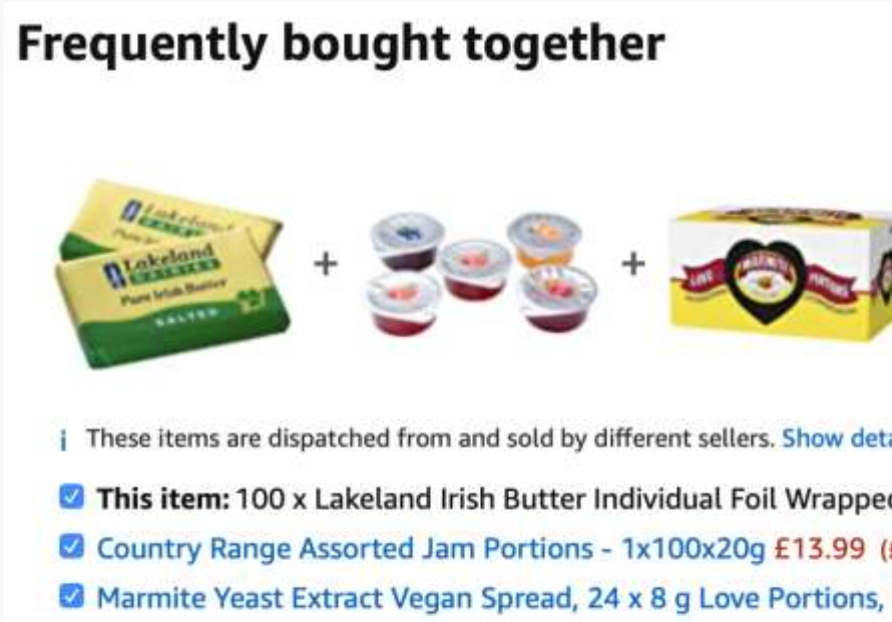
  Historical information can reveal relationships that are not visible in source code

Example: Co-Changes

Assumption: artefacts that have been changed together in the past will change together in the future.

Extract the changed artefacts of every commit and apply frequent itemset data mining:

- Butter is usually bought together with jam…
- AppTest.java is usually changed together with App.java


Frequently bought together

These items are dispatched from and sold by different sellers. Show deta

☑ **This item:** 100 x Lakeland Irish Butter Individual Foil Wrappe

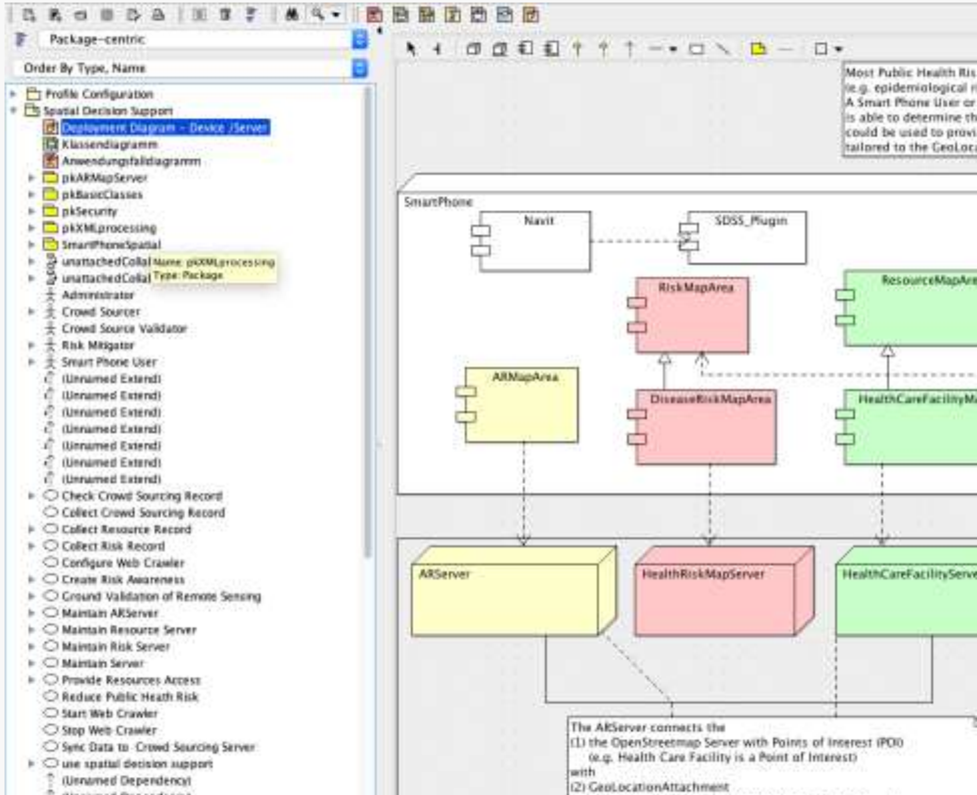
☑ Country Range Assorted Jam Portions - 1x100x20g £13.99 (

☑ Marmite Yeast Extract Vegan Spread, 24 x 8 g Love Portions,

Step 0: The System under Analysis

ArgoUML is a tool to create UML diagrams.

- https://github.com/argouml-tigris-org/argouml
- 17,833 commits
- Has been analysed in many research papers!



Most Public Health Ris
(e.g. epidemiological r
A Smart Phone User or
is able to determine th
could be used to provi
tailored to the GeoLoca

The ARServer connects the
(1) the OpenStreetmap Server with Points of Interest (POI)
(e.g. Health Care Facility is a Point of Interest)
with
(2) GeoLocationAttachment

Step 1: Identify all changed files

Easy!

- `git log --name-status --oneline --reverse`
- …

```
c1598f695e moved Compartment stuff to GEF, …
M ui/Actions.java
M uml/diagram/static_structure/ui/FigClass.java
D uml/diagram/ui/FigCompartment.java
M uml/diagram/ui/FigNodeModelElement.java
D uml/diagram/ui/FigNodeWithCompartments.java
…
```

Step 2: Transform the output

- A 50 lines Python script reads the git result line-by-line and writes a “transaction database” (17833 transactions):

```
…
7676
7677 7684 16500 16915
7677 7715 7716 16915 16916
…
```

- And a map from numbers to files:

```
…
7676: uml/ui/behavior/use_cases/PropPanelActor.java
…
```

Step 3: Mining

- Use a data mining tool to discover patterns, for example, SPMF by Fournier-Viger
- A typical approach does Frequent Itemset Mining via the Apriori algorithm.
- `java -jar spmf.jar run Apriori sets.txt output.txt 0.5%`
- Support:
  A pattern must be present in at least x% of all transactions.
  Here: at least 90 out of 17833 transactions.

Step 4: Analysis

Output

```
…
46138 #SUP: 141
31706 #SUP: 94
64849 #SUP: 100
48570 #SUP: 87
7465 7467  #SUP: 136
7465 7499  #SUP: 91
7496 7499  #SUP: 162
```

Map

…

7465: FigClass.java
7467: FigInterface.java
7496: FigEdgeModelElement.java
7499: FigNodeModelElement.java
…

Association Rules

- FigClass.java and FigInterface.java
  are changed together in 136 commits.
- FigClass.java is changed in 262 commits.
- FigInterface.java is changed in 181 commits.
- Association Rules:
  When FigInterface is changed, then FigClass is changed
  with 136/181 = 75% confidence.

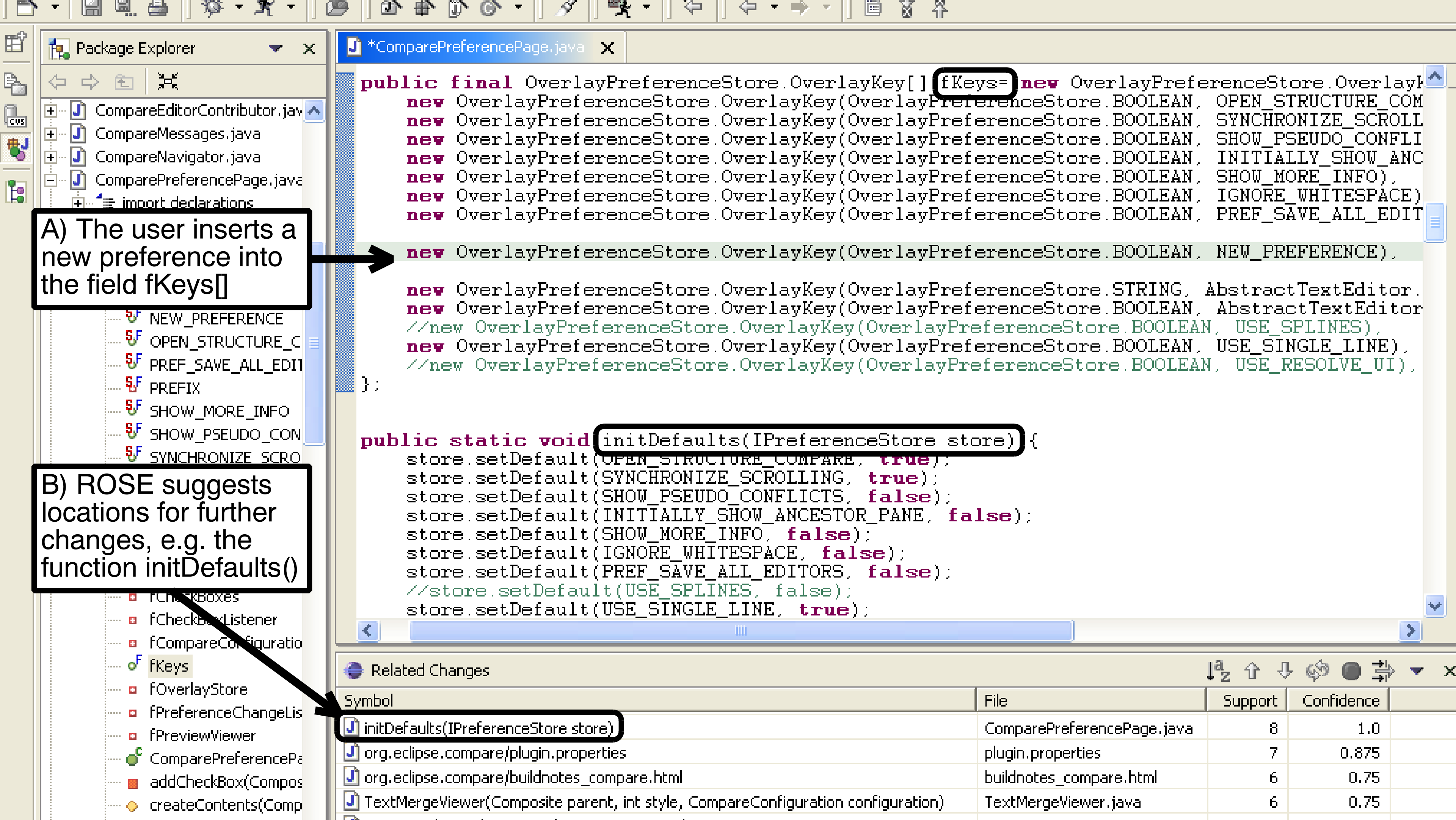
Association Rules

- Extract rules with at least 0.2% support and 95% confidence:
```
java -jar spmf.jar run FPGrowth_association_rules
sets.txt output.txt 0.2% 95%
```
- Result: 4803 rules (4096 with 100% confidence)
- Example:
  When PropPanelClassifierRole and PropPanelActor are changed together, then PropPanelMessage is changed, too.
- Usage:
  Warn if extracted rules are violated in a new commit.

Mining Version Histories to Guide Software Changes

T. Zimmermann, P. Weisgerber, S. Diehl, A. Zeller:
Mining Version Histories to Guide Software Changes.
International Conference on Software Engineering, 2014.

- Mined association rules for changes functions / methods.
- Their system ROSE aims to
  - suggest and predict likely further changes,
  - show item coupling, and
  - prevent errors due to incomplete changes.



A) The user inserts a new preference into the field fKeys[]

B) ROSE suggests locations for further changes, e.g. the function initDefaults()

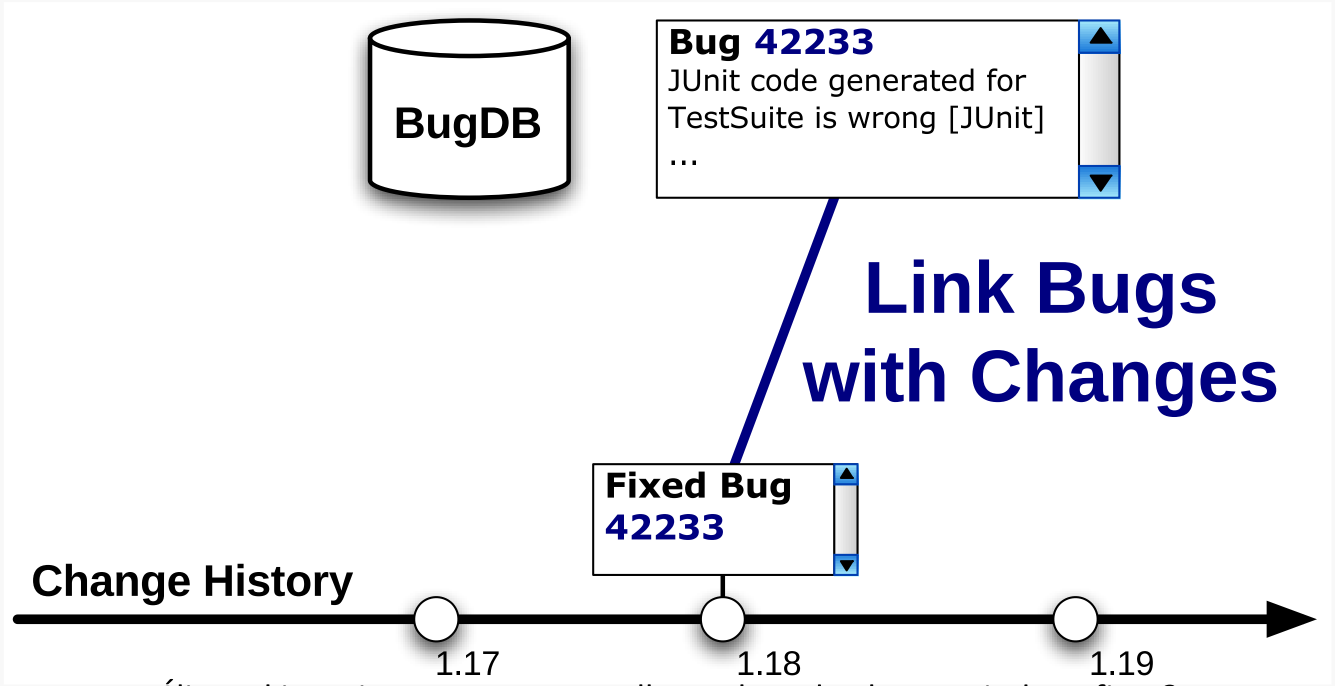
```
public final OverlayPreferenceStore.OverlayKey[] fKeys= new OverlayPreferenceStore.Overlayk
    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, OPEN_STRUCTURE_COM
    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, SYNCHRONIZE_SCROLL
    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, SHOW_PSEUDO_CONFLI
    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, INITIALLY_SHOW_ANC
    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, SHOW_MORE_INFO),
    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, IGNORE_WHITESPACE)
    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, PREF_SAVE_ALL_EDIT

    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, NEW_PREFERENCE),

    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.STRING, AbstractTextEditor.
    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, AbstractTextEditor
    //new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, USE_SPLINES),
    new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, USE_SINGLE_LINE),
    //new OverlayPreferenceStore.OverlayKey(OverlayPreferenceStore.BOOLEAN, USE_RESOLVE_UI),
};


public static void initDefaults(IPreferenceStore store) {
    store.setDefault(OPEN_STRUCTURE_COMPARE, true);
    store.setDefault(SYNCHRONIZE_SCROLLING, true);
    store.setDefault(SHOW_PSEUDO_CONFLICTS, false);
    store.setDefault(INITIALLY_SHOW_ANCESTOR_PANE, false);
    store.setDefault(SHOW_MORE_INFO, false);
    store.setDefault(IGNORE_WHITESPACE, false);
    store.setDefault(PREF_SAVE_ALL_EDITORS, false);
    //store.setDefault(USE_SPLINES, false);
    store.setDefault(USE_SINGLE_LINE, true);
```

Symbol	File	Support	Confidence
initDefaults(IPreferenceStore store)	ComparePreferencePage.java	8	1.0
org.eclipse.compare/plugin.properties	plugin.properties	7	0.875
org.eclipse.compare/buildnotes_compare.html	buildnotes_compare.html	6	0.75
TextMergeViewer(Composite parent, int style, CompareConfiguration configuration)	TextMergeViewer.java	6	0.75

Cross-Repository Mining

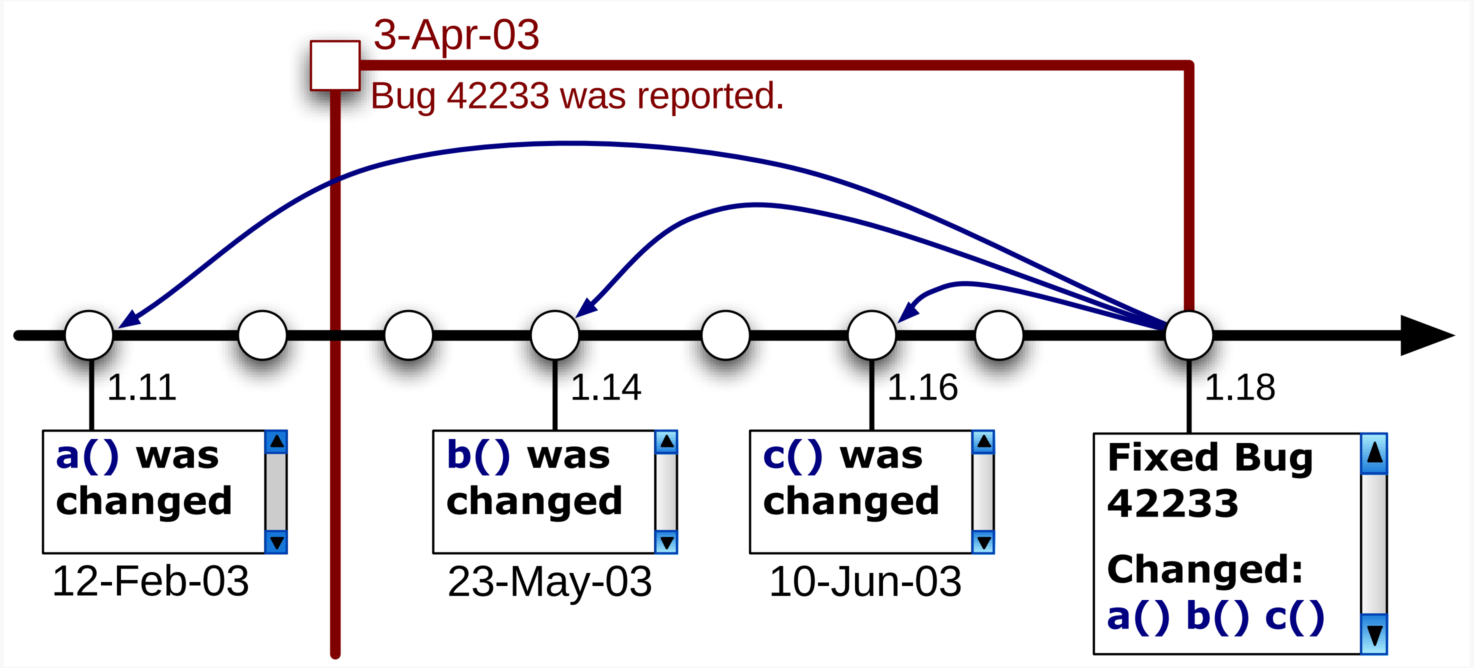
- Different repositories can be linked to identify relationships between artefacts of different types.
- Example:
  Link changes to bug reports to identify
  when bugs have been introduced.

When Do Changes Induce Fixes?



J. Śliwerski, T. Zimmermann, A. Zeller: When do changes induce fixes? International Workshop on Mining software repositories, 2005

When Do Changes Induce Fixes?



J. Śliwerski, T. Zimmermann, A. Zeller: When do changes induce fixes?
International Workshop on Mining software repositories, 2005

Do not program on Fridays!

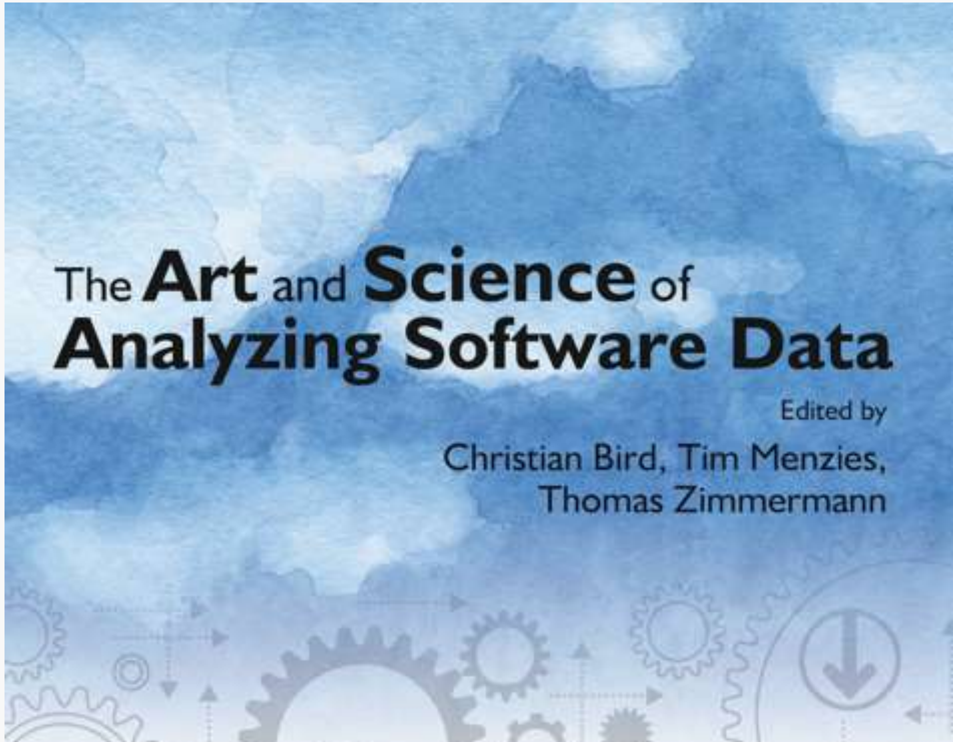
- An analysis of the Mozilla project showed that the likelihood of a change will induce a fix is highest on Fridays.
- For Mozilla, on average,
  - 41.5% of changes induce fixes
  - 48.5% of changes are fixes
  - 21.9% of changes are fixes that induce fixes
- For Eclipse, the numbers were much better…

Example Application: New Change

- Identify potential missing changes.
- Identify risky changes.
- Suggest reviewers.

Software Analytics

Mining Software Repositories is one major area of Software Analytics.


The Art and Science of Analyzing Software Data

Edited by

Christian Bird, Tim Menzies, Thomas Zimmermann

Concepts

- Software repositories are record-keeping databases that store artifacts together with metadata about the artifacts which are created by software developers and other stakeholders during the development.
- Mining Software Repositories aims at uncovering interesting and actionable information about software systems and projects by applying data mining to software data.