Data Mining of SW Repositories – Why and What?

• To support decision making at all stages of the software development process

• To complement other sources of evidence
  – Surveys, Case Studies, Experiments

Acknowledgement

• The following persons contributed to the lecture slides:
  – Ezequiel Scott
  – Riivo Kikas
  – Didar Al-Alam
  – Faiz Shah

Context: Evidence-Based SE

• Knowledge in SE: Anecdotal vs. Evidence-based
• Evidence in Science -> Data
• Data Sources?
  – Surveys, Case Studies, Experiments, Project Repos, Dedicated collections:
    http://promise.site.uottawa.ca/SERepository/datasets-page.html
• Tip: Link to Lecture by Gregory Wilson:
  https://vimeo.com/9270320

About me

• Assoc. Prof. at UT (Software Engineering)
• Adjunct Prof. at University of Calgary, Canada (since 2005)
  – Senior Member of ACM & IEEE
  – Certified SCRUM Product Owner
• Group Leader & Department Head at Fraunhofer Inst. of Experimental SW Engineering (1996-2005)
• Siemens Corporate Research (1987-1996)

About me

• Assoc. Prof. at UT (Software Engineering)
• Adjunct Prof. at University of Calgary, Canada (since 2005)
  – Senior Member of ACM & IEEE
  – Certified SCRUM Product Owner
• Group Leader & Department Head at Fraunhofer Inst. of Experimental SW Engineering (1996-2005)
• Siemens Corporate Research (1987-1996)
Exploratory Questions

- **Existence questions** - Does X exist?
  - Example: Do issue reports actually exist?

- **Description and classification questions** - What is X like? / What are its properties? / How can it be categorized? / How can we measure it? / What is its purpose? / What are its components? / How do the components relate to each other?
  - Example: What are all the types of issue reports?

- **Descriptive comparative questions** - How does X differ from Y?
  - Example: How do issue report formats differ between open source and closed source development projects?

Knowledge and Design Questions

- **Knowledge Questions**: focusing on the way the world is
  - Questions about the normal pattern of occurrence of a phenomenon (Base-rate Questions)
  - Questions about relationships between two different phenomena (Relationship Questions)
  - Questions about causality between two phenomena (Causality Questions)

- **Design Questions**: concerned with how to do things better

Knowledge Questions (cont'd)

- **Base-rate**:
  - Frequency and Distribution Questions - How often does X occur? / What is an average amount of X?
    - Example: How many distinct issue reports per issue report type are created in large software development projects?
  - Descriptive-Process Questions - How does X normally work? / What is the process by which X happens? / In what sequence do the events of X occur?
    - Example: How do software developers use issue reports?

Knowledge Questions (cont’d)

- **Causality**:
  - Simple Causality Questions - Does X cause Y? / Does X prevent Y? / What causes Y? / What are all the factors that cause Y? / What effect does X have on Y?
    - Example: Does the use of GUI test tool X improve software quality?
  - Causality-Comparative Questions - Does X cause more Y than does Z? / Is X better at preventing Y than Z?
    - Example: Does the use of GUI test tool X improve software quality more than other GUI test tools?
  - Causality-Comparative Interaction Questions
Knowledge Questions (cont’d)

• Causality:
  ➔ Causality-Comparative Interaction Questions ➔ Does X or Z cause more Y under one condition but not others?

  Example: Does the use of GUI test tool X improve software quality more than GUI test tools in web application projects, but not in genuine mobile applications?

Design Questions

➔ "What is an effective way to achieve X?" / What strategies help to achieve X?

Examples:
- What is an effective way for teams to test mobile applications in order to improve quality without increasing cost?
- What is an effective way for teams to design mobile applications in order to improve energy efficiency?

The Wallace Model

Theories

- Theory Construction
- Logic (induction)

Empirical Generalizations (Laws)

Observations

Research Methods

- Data Analysis
- Parameter Estimation

Hypotheses (Research Questions)

Research Design

Data Collection & Research Methods

• Survey
  ➔ Questionnaire-based (primary study)
  ➔ Literature-based (secondary / tertiary study)

• Case Study
  ➔ Descriptive
  ➔ Exploratory
  ➔ Confirmatory

• Experiment
  ➔ Controlled Experiment
  ➔ Quasi-Experiment
  ➔ Longitudinal studies

• Many other ...
  ➔ Action Research
  ➔ Ethnography
  ➔ Design Science

Survey Research

• A survey is a data collection method or tool used to gather information about individuals in order to identify the characteristics of a broad population

• The defining characteristic is the selection of a representative sample from a well-defined population with the aim to generalise from the sample to the population.

• Usually conducted with questionnaires, but can also involve structured interviews or data logging techniques

• Example:
  ➔ Investigate to what extent, how, by which companies, and by whom within the companies, TDD is used.
Survey – Characterisation (cont’d)

When to use it?
- Either at start of research to get an understanding of the current situation...
- or at the end of a research phase to see the impact/acceptance/etc. of a new method/technique/tool

Issues:
- “Superficial” --> no explanation / no causality --> not suitable for hypothesis testing
- “Generalisability” of results depends on the choice of population and “response rate”, as well as validity and reliability of the data collection instrument

Survey – Example

What?

Research Questions:
- How is Agile practiced at Microsoft?
- i.e. What do engineers do?
- How do engineers feel about it?
- i.e. Do they like it?

Who, Where, and When?

Microsoft (worldwide, 2006)
Anonymous survey sent to 2821 engineers
• 10% random sampling of all developers, testers, program managers at Microsoft in October 2006
487 valid responses
• 44% developers, 28% testers, 17% program managers

Why?

Many agile approaches exist
– what’s in it for Microsoft?

Survey – Example (cont’d)

Team coding standards
Continuous integration of code
System metaphors
Simple design
Sustainable pace
User stories
Small iterations
Direct interaction with customers
Design improvement
Collective code ownership
Agile training
Whole team daily stand-up meetings
Test-driven development
Pair programming

Survey – Example (cont’d)

Agile practice penetration at Microsoft

425
400
375
350
325
300
275
250
225
200
175
150
125
100
75
50
25
0

Qantitative Results (Highlights)
• 33% of respondents (spread across divisions) report their team uses Agile methodologies.
• They mainly use Scrum (68%).
• Used for many legacy products.
• Agile usage does not appear to depend on team co-location.
• Test-driven development and pair programming are not very common.

Qualitative Results (Highlights)
• MS engineers who have used Agile like it for their local teams, but not necessarily for their organization.
• They worry about scale, overhead, and management buy-in.

Controlled Experiment – Simple Example

• Independent Variable: Tool used (Levels: X and Y)
• Dependent Variable: Design Quality
• Treatments: E = use the new Tool X / C = use the old Tool Y

Controlled Experiment – Characterisation

• An investigation of a testable hypothesis where one or more independent variables are manipulated to measure their effect on one or more dependent variables.
• In Software Engineering, typically, experiments require human subjects to perform some task.

NB: Design can be within-subject or between-subject
Controlled Experiment vs. Quasi-Experiment

Randomization is a prerequisite for a controlled experiment!

Randomized or True Experiment

Is random assignment used?

Yes

No

Is there a control group or multiple measures?

Yes

No

quasi-experiment

non-experiment

Experiment – Example

What?

Research Question:

What is best – Pair Programming or Solo Programming?

Why?

Many studies with contradicting results – mostly conducted with students (not with professional developers)

Who, Where, and When?

Norway, 2007

285 junior, intermediate and senior professional Java consultants from 29 companies were paid to participate (one work day)

99 individuals; 98 pairs

The pairs and individuals performed the same Java maintenance tasks on either:

• a “simple” system (centralized control style), or

• a “complex” system (delegated control style)

They measured:

• duration (elapsed time)

• effort (cost)

• quality (correctness) of their solutions

Why?

Many studies with contradicting results – mostly conducted with students (not with professional developers)

Experiment – Example (cont’d)

Experiment: Overall Effect of PP

Experiment: Effect of PP for Juniors

Experiment: Effect of PP for Seniors

Moderating Effect of System Complexity for Juniors

Duration: Overall Effect of PP

Experiment: Effect of PP for Juniors taking task complexity under consideration
Experiment: Effect of PP for Seniors taking task complexity under consideration

So, when should we use PP?

Case Study – Characterisation

Case Study – Guidelines

Case Study – Variants
Validity & Reliability of Empirical Studies

- Construct Validity
  - Concepts being studied are operationalized and measured correctly (do the measures used actually represent the concepts you want to measure?)
- Internal Validity
  - Establish a causal relationship and sort out spurious relationships (exclude confounding variables / by: random sampling, blocking, balancing)
- Conclusion Validity
  - Do proper statistical inference

- External Validity
  - Establish the domain to which a study’s findings can be generalized (precisely describe the population and experimental conditions)
- Reliability
  - The study can be repeated (i.e., by other researchers) and yields the same results
  - The measurement instrument is reliable (inter-rater agreement)

Data Mining of SW Repositories

- Why and What?
  - To support decision making at all stages of the software development process
  - To complement other sources of evidence
    - Surveys, Case Studies, Experiments

Data Mining in SW Engineering: Application Examples

- Journal: EMSE’16

- Conferences:
  - EASE’16: [http://ease2016.lero.ie](http://ease2016.lero.ie)

Data Mining in SW Engineering (2016)

- Journal: EMSE’16

- Conferences:
  - EASE’16: [http://ease2016.lero.ie](http://ease2016.lero.ie)

SE Data Repositories

- App stores (Google Play, etc.)
- Q/A web-pages (e.g., StackOverflow)
- Crash report repositories (e.g., Ubuntu’s repository)
- YouTube tutorials (e.g., tool tutorials)
- ELFF dataset at Brunel: [https://github.com/tjshippey/ESEM2016](https://github.com/tjshippey/ESEM2016)
- Data Showcases at MSR’16
- Industry data: ISBSG repository, Finnish dataset
- Issue Trackers -> e.g., JIRA
- Version Control Systems -> e.g., Git

GitHub & GHTorrent

- [https://en.wikipedia.org/wiki/GitHub](https://en.wikipedia.org/wiki/GitHub)
- API’s for Java, Ruby, Python, etc.

Articles:

- “The GHTorrent Dataset and Tool Suite” (2013)
- “Lean GHTorrent: GitHub data on demand” (2014)
GHTorrent – DB Dumps: Limitations

- Dumps contain only the first order dependencies
  - e.g., contributors to a repository and their followers, but not followers of these followers
- Creating the dumps can be a lengthy process, potentially requiring several days to complete
- No recovery actions in case of errors are currently implemented, potentially leading to incomplete dumps
  - e.g., if GitHub fails to answer an API request
- Requests to lean GHTorrent should not exceed 1000 repositories
  - This is to limit the load on GHTorrent servers

Getting data from GitHub repositories using the Java API
(by Ezequiel Scott and Didar Al-Alam)
Example: GitHubDataExtractor

We can use the GitHubDataExtractor project to retrieve data from Github repositories.

- The project relies on the Github API for Java
- You can download the GitHubDataExtractor from here
  - import the project into your favorite Java IDE (e.g. Eclipse) and then
  - add the required libraries to the build path

Links at: https://courses.cs.ut.ee/2017/dm/spring/Main/Links

What data can be extracted?

- Commits
- Pull requests
- Issues...

About the project

- There are two important classes:
  - RRCalc – just the main class
  - CommitDataCollection – the class in charge of collecting the commit data, it does the hard job
- In RRCalc, we set up important data such as the username, repository, the credentials, dates, etc.
- In CommitDataCollection, we use the Github API to connect with the Github services and obtain data from the repository

How does it work?

First, we have to create an object for the repository and set the credentials up:

```
RepositoryService repservice = new RepositoryService();
repservice.getClient().setCredentials(GitCredits[0], GitCredits[1]);
```

Username
Password

Then, we can use different services for retrieving the data from the repository.

There are three services available: Commit, Issue, and Pull. All of them require credentials.

- For downloading commits
  ```java
  CommitService commitService = new CommitService();
  commitService.getCommits(repo); // RRCalc parallels
  ```

- For downloading issues
  ```java
  IssueService issueService = new IssueService();
  issueService.getIssues(repo); // IssueService parallels
  ```

- For downloading pull requests
  ```java
  PullRequestService pullRequestService = new PullRequestService();
  pullRequestService.getPullRequests(repo); // PullRequestService parallels
  ```

Finally, we can retrieve all the data from each service and store it in List objects. It makes finding elements easier to do.

- For downloading commits
  ```java
  List<RepositoryCommit> commitList = commitService.getCommits(repo);
  ```

- For downloading issues
  ```java
  List<RepositoryIssue> issueList = issueService.getIssues(repo);
  ```

- For downloading pull requests
  ```java
  List<PullRequest> pullList = pullRequestService.getPullRequests(repo, "closed");
  ```
How does it work? (cont.)

Once we have obtained the lists with the data, we can retrieve all the info from the commit/issue/pull objects.

```java
// Getting the SHA key from the i-commit
String sha = commitList.get(i).getSha();
// Getting the author from the i-commit
String author = commitList.get(i).getCommit().getAuthor().getName();
// Getting the message from the i-commit
String message = commitList.get(i).getCommit().getMessage();
```
Predicting Issue Resolution Time – Why & How?

Why?
• Maintenance/Evolution is consuming a major share of the development effort
• Knowing the probable issue resolution time helps in the planning of resource allocation

How?
• Manually done by experts
• Automatically done by models

Motivation of Study
• Many attempts have been made to predict issue resolution time
  • Published work shows mixed results with regards to performance
• Availability of a case Company:
  • Expert estimates
  • Plan and actual data available
  • Question: Would automatic prediction outperform experts?

Related Work
• Little industry data available regarding expert estimates
• Several studies on automatic prediction (> 2006):
  • Usually using OSS data with actual IR times
  • Several methods used:
    • kNN, α-kNN, (simple) k-means clustering, Naïve Bayes Classifier, C4.5 Decision Tree, Naïve Bayes, and Logistic Regression
  • Different performance measures used:
    • MMRE, Pred.(25%) , classification accuracy, AUC
  • High variation in performance / Unclear whether experts are outperformed

Research Goals
(1) To compare the prediction quality of expert-based IRT prediction in a software company in Estonia with that of various fully automated IRT prediction approaches proposed/used by other researchers
  • including k-means clustering, k-nearest neighbor classification, Naïve Bayes classification, decision trees, random forest (RF) and ordered logistic regression (OLR)
(2) To improve the current IRT prediction quality in the company at hand

Approach
• Establish baseline (expert data in Company)
• Apply automatic prediction methods found in the literature to Company data
• Apply enhanced versions of the found prediction methods to Company data
• Compare results (using 4 performance measures)

Company Baseline
Dataset:
• IRs must be written in English
• IRs must be ‘closed’
• IRs must have both ‘estimated’ and ‘actual’ resolution times
2125 IRs in total
894 IRs used
Company Baseline

- Experts' performance: predicted versus actual

Number of issues in interval according to estimate (black)
Number of issues in interval actually (gray)

Intervals in days (8 hours):
[0, 0.5] - (0.5, 1] - (1, 3] - (3, 6] - (6, 11] - (20, 40] - (40, ...)

Automatic Prediction

- Using methods as published
- Using enhanced methods
  - Outlier removal
  - Advanced k-means

Automatic Prediction (as published)

- Using methods as published
- Using enhanced methods
  - Outlier removal
  - Advanced k-means

Automatic Prediction (enhanced)

- Using methods as published
- Using enhanced methods
  - Outlier removal
  - Advanced k-means

Comparison: Expert vs. Model

<table>
<thead>
<tr>
<th>Prediction Technique</th>
<th>Pred. abs. (0.5h)</th>
<th>Pred. abs. (1h)</th>
<th>Pred. rel. (10%)</th>
<th>Pred. rel. (25%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expert</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>0.150</td>
<td>0.171</td>
<td>0.121</td>
<td>0.138</td>
</tr>
<tr>
<td>C4.5 Decision Tree</td>
<td>0.405</td>
<td>0.378</td>
<td>0.459</td>
<td>0.459</td>
</tr>
<tr>
<td>Random Forest</td>
<td>0.333</td>
<td>0.443</td>
<td>0.311</td>
<td>0.311</td>
</tr>
<tr>
<td>RF (enhanced)</td>
<td>0.172</td>
<td>0.249</td>
<td>0.251</td>
<td>0.251</td>
</tr>
<tr>
<td>Ordered Logistic Regression</td>
<td>0.294</td>
<td>0.429</td>
<td>0.459</td>
<td>0.459</td>
</tr>
<tr>
<td>OLR (enhanced)</td>
<td>0.357</td>
<td>0.197</td>
<td>0.541</td>
<td>0.541</td>
</tr>
<tr>
<td>SVM (linear)</td>
<td>0.640</td>
<td>0.697</td>
<td>0.657</td>
<td>0.657</td>
</tr>
<tr>
<td>SVM (radial)</td>
<td>0.722</td>
<td>0.693</td>
<td>0.658</td>
<td>0.658</td>
</tr>
</tbody>
</table>
Results Summary

- RQ 1: Comparison Company vs. Published Models
  - Experts outperform published models

- RQ 2: Enhance Company’s Performance
  - Spherical k-means applied to Title only and with using only last 50 reported issues is for 3 out of 4 performance measures (slightly) better than experts

Discussion

The good news:

- Automatic prediction is roughly as good as experts and thus might be used instead of them

The interesting news:

- Experts and models might complement each other

Limitations – Threats to Validity

- External validity
  - Only one case with a relatively small data set

- Internal validity
  - The fact that the case company was recording plan/actual expert data might mean that they are relatively mature in this particular aspect (i.e., estimating IRT) and thus the comparison with automatic methods might be unfair

- Conclusion validity
  - Choice of performance measure

Application Example 4

- WAMA 2016
- Feature-Based Evaluation of Competing Apps
- Slides: Faiz Ali Shah

Motivation

- User feedback could help developers improve the quality of their app by comparing it with other similar apps

More precisely:

- To identify sets of app features loved by users in other apps but missing in company’s own app
- To identify app features which are perceived negatively by its users and need improvement

App Reviews Dataset

- We used app reviews dataset openly available on the website of Swinburne University of Technology

![Figure 1. Number of reviews in each app](link to app reviews)
**Approach**

Figure 2. Overview of the approach

**Tool Prototype: Show List of Apps and Select Base App**

**Tool Prototype: Present Extracted Features of Base App and Select Features of Interest**

Feature list of base app “Calorie Counter” with minimum support count = 22

**Tool Prototype: Present competing Apps**

Competing apps based on selected features of base app “Calorie Counter”

**Tool Prototype: Evaluation of Competing Apps**

Result 1: Feature-based comparison of the base app “Calorie Counter” with competing app “Map My Fitness”
Result 2:
Feature-based comparison of the base app “Calorie Counter” with competing app “Run Keeper”

\[
\text{Calorie Counter vs. Run Keeper} = \frac{\sum (\text{table cell count}[i] \times \text{sentiment distance})}{\text{feature count}}
\]

\[= \frac{1 \times 0 + 0 \times 1 + 0 \times 2 + 0 \times (-1) + 3 \times 0 + 0 \times 1 + 0 \times (-2) + 0 \times (-1) + 0 \times 0}{4}
\]

\[= 0\]

Result 3:
Feature-based comparison of the base app “Calorie Counter” with competing app “Strava Running and Cycling”

\[
\text{Calorie Counter vs. Strava Running and Cycling} = \frac{\sum (\text{table cell count}[i] \times \text{sentiment distance})}{\text{feature count}}
\]

\[= \frac{0 \times 0 + 1 \times 1 + 1 \times 2 + 1 \times (-1) + 1 \times 0 + 0 \times 1 + 0 \times (-2) + 0 \times (-1) + 0 \times 0}{4}
\]

\[= 0.5\]

Overall score of the base app compared to the competing app is positive.

Application Examples – Summary

- Ex1 – Release Readiness – RAISE 2016 (PhD)
- Ex2 – Issue RT (a) – MSR 2016 (PhD)
- Ex3 – Issue RT (b) – EASE 2016 (MSc)
- Ex4 – App Reviews – WAMA 2016 (MSc/PhD)
- More (ongoing PhDs):
  - Green Software
  - Open Innovation (RE)
  - …
- Many MSc thesis topics

Thank You!