Project mining

Mining Projects from (Un)Structured Data

Saimir Bala, WU (Vienna University of Economics and Business)
Motivation

- You the manager of a software development company
- You are applying best practices, established project management guidelines and tools

But...

- What is really going on in your software development project?
- Why are deadlines not met? Why are the costs superior to the planned?
- Why is does the your software product require more maintenance than what you thought?
About software projects

- Software processes are carried out in a project-oriented fashion
- Goal is a release of a software product
- Artifact-centric processes
- Software development methodology (e.g., Scrum, Waterfall)
- Artifacts are tracked by means of Version Control Systems
- Should follow best practices (e.g. Principles of good modularization)

Q: How can we help the manager to gain transparency on the software project?
Project-Oriented Business Processes
## Project-Oriented vs. Classic Business Processes

<table>
<thead>
<tr>
<th>Project-Oriented</th>
<th>Classic Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plan (e.g. Gantt, PERT)</td>
<td>Process Model (e.g. Petri Net, BPMN)</td>
</tr>
<tr>
<td>One time (fixed goal and resources)</td>
<td>Recursive, Cyclic</td>
</tr>
<tr>
<td>Single instance</td>
<td>Many instances</td>
</tr>
<tr>
<td>Workpackages, Modules, Units</td>
<td>Activities</td>
</tr>
<tr>
<td>Subworkpackages, Submodules</td>
<td>Suprocesses</td>
</tr>
</tbody>
</table>
## Project-Oriented vs. Classic Business Processes

<table>
<thead>
<tr>
<th>Project-Oriented</th>
<th>Classic Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plan (e.g. Gantt, PERT)</td>
<td>Process Model (e.g. Petri Net, BPMN)</td>
</tr>
<tr>
<td>One time (fixed goal and resources)</td>
<td>Recursive, Cyclic</td>
</tr>
<tr>
<td>Single instance</td>
<td></td>
</tr>
<tr>
<td>Workpackages, Modules, Units</td>
<td></td>
</tr>
<tr>
<td>Subworkpackages, Submodules</td>
<td></td>
</tr>
</tbody>
</table>

**Process Mining**
Software projects are supported by a variety of tools

Examples
- Project management, Bug-tracking
- Development
- Dependency management
- Testing
- Continuous integration
- Documentation
- Version Control System
State of the Art: Activity Mining

Kindler et al. 2006
State of the Art: Dotted Chart

Song and van der Aalst. 2007
State of the Art: Evolution Storylines

Ogawa and Ma 2010
State of the Art: Visual Software Analytics

Wettel and Lanza 2007
Methodology: DSR

Identification of the problem → Objectives of the artifact → Artifact Development → Evaluation → Demonstration → Communication

Peffers 2007
Mining the Gantt Chart of a Project

VCS log

Project's GANTT chart

<table>
<thead>
<tr>
<th></th>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
</tr>
</thead>
<tbody>
<tr>
<td>WP 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WP 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WP 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WP 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example from the SHAPE project

Project structure

Activity inferred from single events
1. Threshold based
2. Activated when a tree node in the project structure is collapsed
3. Decomposed when node is expanded

Associated information:
1. User
2. Type of change
3. Comment

“example” work package
“pm” work package

single events within files
Indicators

- Data from the VCS
  - Authors
  - Files
  - Type of change, Etc
- Coverage, i.e. work-intensity
  the ratio between active working periods (i.e., the time spans of activities) and the total work package duration
- Expected active time between commits (**tc**)
  average work speed (commit frequency) during active times
Assumptions

The following assumptions are made:

1. Meaningful file structure
   - Project participants organize the files in a representative (e.g., spatially separating documentation from testing into different folders).

2. Regular commits
   - Project participants systematically commit their changes in the VCS

3. Descriptive comments.
   - Project participants write descriptive comments that allow others members to understand the changes made to the software
Resource Classification from Commit Messages
Resource Classification from Commit Messages
Resource Classification from Commit Messages: Developer vs. Tester

**Developer**

- backend: 10.5%
- data: 7.5%
- design: 16.0%
- documentation: 9.4%
- build: 2.0%
- toolManagement: 2.0%
- addition: 4.6%
- refactor: 4.9%
- vcsManagement: 13.4%
- test: 23.9%
- maintenance: 7.5%

**Tester**

- development: 43.3%
- documentation: 16.4%
- removal: 16.4%
- data: 4.5%
- vcsManagement: 3.0%
- web: 3.0%
- refactor: 1.5%
- test: 1.5%
- maintenance: 7.5%
Learning Decision Trees from Projects
Mining Hidden Work Dependencies
Artifact Evolution as Time Series

Are they similar?

Correlation!
Characterization of Projects wrt Dependencies

- operationcode
- mysqlpython
- monolingualwordaligner
- jgitcookbook
- graphql
- gantt
- facebookjavasdk
- caret
- camundaRD
- biglist

Legend:
- High Dist. High Dep.
- High Dist. Low Dep.
- Low Dist. High Dep.
- Low Dist. Low Dep.


Next steps

- Identification of the problem
- Objectives of the artifact
- Artifact Development
- Communication
- Evaluation
- Demonstration

Peffers 2007
Next steps

Identification of the problem → Objectives of the artifact → Artifact Development

Communication → Evaluation → Demonstration

Peffers 2007
Next steps

Identification of the problem -> Objectives of the artifact -> Artifact Development

Communication < Communication Evaluation < Demonstration

Peffers 2007
Questions?

DEPARTMENT OF INFORMATION SYSTEMS AND OPERATIONS
INSTITUTE FOR INFORMATION BUSINESS
Welthandelsplatz 1, D2/1.026
1020 Vienna, Austria

M.SC. SAIMIR BALA
T +43-1-313 36-5304
F +43-1-313 36-905304
saimir.bala@wu.ac.at
www.wu.ac.at
Backup slides
Data Model for SQL Querying VCS logs
Mining the Real Gantt Chart
Mining the Gantt Chart of a Project

<table>
<thead>
<tr>
<th></th>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
</tr>
</thead>
<tbody>
<tr>
<td>WP 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WP 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WP 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WP 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example from the SHAPE project

- **Project structure**

  - **“example” work package**
  - **“pm” work package**
  - **single events within files**

- **Associated information:**
  1. User
  2. Type of change
  3. Comment

- **Activity inferred from single events**
  1. Threshold based
  2. Activated when a tree node in the project structure is collapsed
  3. Decomposed when node is expanded
Indicators

- Data from the VCS
  - Authors
  - Files
  - Type of change, Etc
- Coverage, i.e. work-intensity
  the ratio between active working periods (i.e., the time spans of activities) and the total work package duration
- Expected active time between commits (tc)
  average work speed (commit frequency) during active times
The following assumptions are made:

1. Meaningful file structure
   - Project participants organize the files in a representative (e.g., spatially separating documentation from testing into different folders).

2. Regular commits
   - Project participants systematically commit their changes in the VCS

3. Descriptive comments.
   - Project participants write descriptive comments that allow other members to understand the changes made to the software.
## Some Real World Projects

<table>
<thead>
<tr>
<th>Project</th>
<th>Description</th>
<th>Commits</th>
<th>Users</th>
<th>Files</th>
<th>Duration</th>
<th>tc</th>
<th>coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opendata bot</td>
<td>Open Data AT Assistant: Data Pioneers Create Camp project</td>
<td>28</td>
<td>1</td>
<td>3507</td>
<td>16</td>
<td>0</td>
<td>100.00%</td>
</tr>
<tr>
<td>MiningVCS</td>
<td>Gantt chart visualization of projects</td>
<td>84</td>
<td>1</td>
<td>111</td>
<td>61</td>
<td>1.9</td>
<td>87.00%</td>
</tr>
<tr>
<td>MSR paper</td>
<td>Writing a conference paper</td>
<td>35</td>
<td>2</td>
<td>78</td>
<td>44</td>
<td>12.8</td>
<td>70.00%</td>
</tr>
<tr>
<td>Progit2</td>
<td>Pro Git 2nd Edition</td>
<td>1292</td>
<td>134</td>
<td>955</td>
<td>481</td>
<td>118.6</td>
<td>60.00%</td>
</tr>
<tr>
<td>GHDiscovery</td>
<td>GitHub Activities Discovery repository.</td>
<td>11</td>
<td>1</td>
<td>97</td>
<td>29</td>
<td>6.6</td>
<td>53.00%</td>
</tr>
<tr>
<td>SHAPE</td>
<td>Joint research project on railway automation</td>
<td>624</td>
<td>13</td>
<td>6470</td>
<td>1127</td>
<td>21.8</td>
<td>38.00%</td>
</tr>
<tr>
<td>papers from siemens</td>
<td>Repository form Siemens to keep track of paper writing processes</td>
<td>649</td>
<td>5</td>
<td>1791</td>
<td>1853</td>
<td>26.4</td>
<td>23.00%</td>
</tr>
<tr>
<td>Facebook-ads-java-sdk</td>
<td>Java SDK for Facebook Ads APIs</td>
<td>38</td>
<td>8</td>
<td>428</td>
<td>324</td>
<td>18.2</td>
<td>22.00%</td>
</tr>
<tr>
<td>Biglist-of-naughty-strings</td>
<td>Strings which have a high probability of causing issues when used as user-input data.</td>
<td>202</td>
<td>60</td>
<td>15</td>
<td>530</td>
<td>53.3</td>
<td>10.00%</td>
</tr>
</tbody>
</table>

We set the aggregation **threshold** to 7 days (i.e. two events belong to the same activity only if their temporal distance is one week or less)
Open-Data Helper Bot

- Open Data AT Assistant: Data Pioneers Create Camp project
  - Helps search for an open dataset
- 28 commits, 1 user
  3507 files, 16 days
- 0 tc, 100% coverage
Mining VCS Software

- This software project
- 84 commits, 1 user, 111 files, 62 days
- tc 1.9 hours, coverage 87%
MSR Paper

- Preparation of a conference paper
- 35 commits, 2 users, 78 files, 44 days
- 12.8 tc, 70% coverage
Book Writing Project

- Progit book 2nd edition
- 1292 commits, 134 users, 955 files, 481 days
- tc 118.6 hours, coverage 60%
Students Project: Discovering Github

Activities

- Student class project
- 1 user, 72 files, duration 29 days
- tc 8.8 hours, coverage 59%
SHAPE Project

- Joint research project on railway automation
- 6470 files, 13 users, duration 1127 days
- tc 21.8 hours, coverage = 38%

![SHAPE Project Diagram](image-url)
Writing Papers Project (from Industry)

- Repository for papers writing process taken from SHAPE project
- 649 commits, 5 users, 1791 files, 1853 days duration
- tc 26.4 hours, coverage 23%
Facebook ads java sdk

- Java development kit for Facebook ads
- 38 commits, 8 users, 428 files, 324 days
- 18.2 tc, 22% coverage
Big List of Naughty Strings

- An evolving list of strings which have a high probability of causing issues when used as user-input data.
- 202 commits, 15 files, 51 users, 531 days
- tc 53.3 hours, coverage 10%

No activities found in the subdirectories, i.e. no continuous work for in the same subdirectory within the given aggregation threshold.
Uncovering the Hidden Co-Evolution in the Work History of Software Projects
How can we use data generated from the software project to help gaining transparency on the status and work history?

R1 (Extract the work history)
- Discover the process of how artifacts evolve in the project as a labeled set of steps

R2 (Uncover Work-Related Dependencies)
- Identify that parts of the work are connected to other parts \( \rightarrow \text{co-evolution} \) of two artifacts?

R3 (Measure Dependencies)
- How strongly depend two artifacts on one another?
# State of the Art

<table>
<thead>
<tr>
<th>MSR</th>
<th>Process Mining</th>
<th>Visualization</th>
</tr>
</thead>
</table>
| • Mostly solving R2 (Uncover work-related dependencies and R3 (Measure dependencies))  
• Kindler et al. 2006, Goncalves et al. 2011, Poncin et al. 2011, Bala et al. 2015 | • No approach addressing R1, R2, and R3 simultaneously  
• Voinea and Telea 2006, Ripley et al. 2007, Greene and Fischer 2015 |
Approach
Challenges

- How to capture events?
- How to obtain the work history from the events?
- What are important informations we need to consider in order to identify dependencies?
- How to analyze the data?
- How to measure work-dependency?
The following assumptions are made:

1. Meaningful file structure
   - Project participants organize the files in a representative (e.g., spatially separating documentation from testing into different folders).

2. Regular commits
   - Project participants systematically commit their changes in the VCS

3. Descriptive comments.
   - Project participants write descriptive comments that allow others members to understand the changes made to the software
Concepts

Artifact evolution

• Changes made to an artifact during its lifetime, measured in Lines of Code

Dependency

• High similarity in the evolution of two software artifacts
Metrics

Degree of Co-Evolution

- **Strength of the connection.** A value in the interval \([0,1]\), where 1 is the highest degree of co-evolution.

File Distance

- **Distance between two files in the file tree.** Equal to the length of the path traversing the least common ancestor.
Generate events from software repository data into a log file

Preprocess log data

Compute artifact evolution

Compute Dependencies

Potential wok-dependencies

Select set of highly dependent & distant files

Compute Metrics
Computing Dependencies

Are they similar?

Correlation!
## Results

| Project       | Commits | Files | \( \chi^H \) | \( \chi^L \) \( (d^L, \chi^L) \) | \( d^L, \chi^H \) | \( d^H, \chi^L \) | \( d^H, \chi^H \) | \( |p_f| \) | \( \text{max}(|p_f|) \) | \( |A_{\text{err}}| \) | \( \text{max}(d) \) |
|---------------|---------|-------|---------------|-------------------------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| smsr          | 21      | 6     | 22            | 6                             | 0              | 9              | 6              | 13             | 2.71           | 5               | 1.82           | 1.43           | 6              |
| mwaligner     | 21      | 9     | 37            | 7                             | 6              | 30             | 1              | 7              | 1.11           | 2               | 2.40           | 0.94           | 3              |
| Biglist       | 202     | 15    | 22            | 90                            | 31             | 18             | 59             | 4              | 1.47           | 3               | 2.76           | 1.20           | 5              |
| camundaRD     | 11      | 15    | 74            | 26                            | 0              | 25             | 26             | 49             | 2.18           | 4               | 2.05           | 2.03           | 7              |
| graphql       | 256     | 30    | 89            | 357                           | 121            | 89             | 236            | 0              | 1.40           | 2               | 3.18           | 1.11           | 4              |
| jgitcookbook  | 135     | 89    | 773           | 2866                          | 505            | 289            | 2361           | 484            | 6.93           | 8               | 1.33           | 2.68           | 14             |
| mysqlpython   | 749     | 168   | 2288          | 11571                         | 742            | 591            | 10829          | 1697           | 2.59           | 7               | 1.65           | 2.52           | 11             |
| gantt         | 23      | 228   | 7006          | 14343                         | 386            | 3480           | 13957          | 3526           | 3.30           | 4               | 1.71           | 2.16           | 7              |
| facebookjavadoc | 38   | 293   | 16478         | 26092                         | 2017           | 16311          | 24075          | 167            | 6.21           | 8               | 4.78           | 5.58           | 13             |
| caret         | 864     | 432   | 15366         | 60874                         | 9538           | 14785          | 51336          | 581            | 3.01           | 4               | 3.15           | 1.60           | 7              |
| operationcode | 1114    | 1053  | 84024         | 444605                        | 2291           | 5537           | 442314         | 78487          | 4.27           | 8               | 2.01           | 4.85           | 15             |
Co-Evolution versus Distance

[Scatter plot showing the relationship between Degree of Co-Evolution and Distance.]
Characterization of Projects wrt Dependencies

- operationcode
- mysqlpython
- monolingualwordaligner
- jgitcookbook
- graphql
- gantt
- facebookjavasdk
- caret
- camundaRD
- biglist

Legend:
- High Dist. High Dep.
- High Dist. Low Dep.
- Low Dist. High Dep.
- Low Dist. Low Dep.
Zipf law on real projects: 100%
Zipf law on real projects: top 80%
Zipf law on real projects: top 50%

Camunda  Operationcode  Caret
Stories

Are they similar?
Conlusion

- Mining project-oriented business process is difficult
- Provide hints for the project manager
- Work dependencies not easy to be seen without analysing the work history
- Future work:
  - Improve method for comparing time series
  - Semantic analysis of process labels