Project mining

Mining Projects from (Un)Structured Data

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

WU WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

EFMD EQUIS ACCREDITED · AACSB ACCREDITED · ASSOCIATION OF MBAS ACCREDITED

12 JUNE 2017

Motivation

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

But…

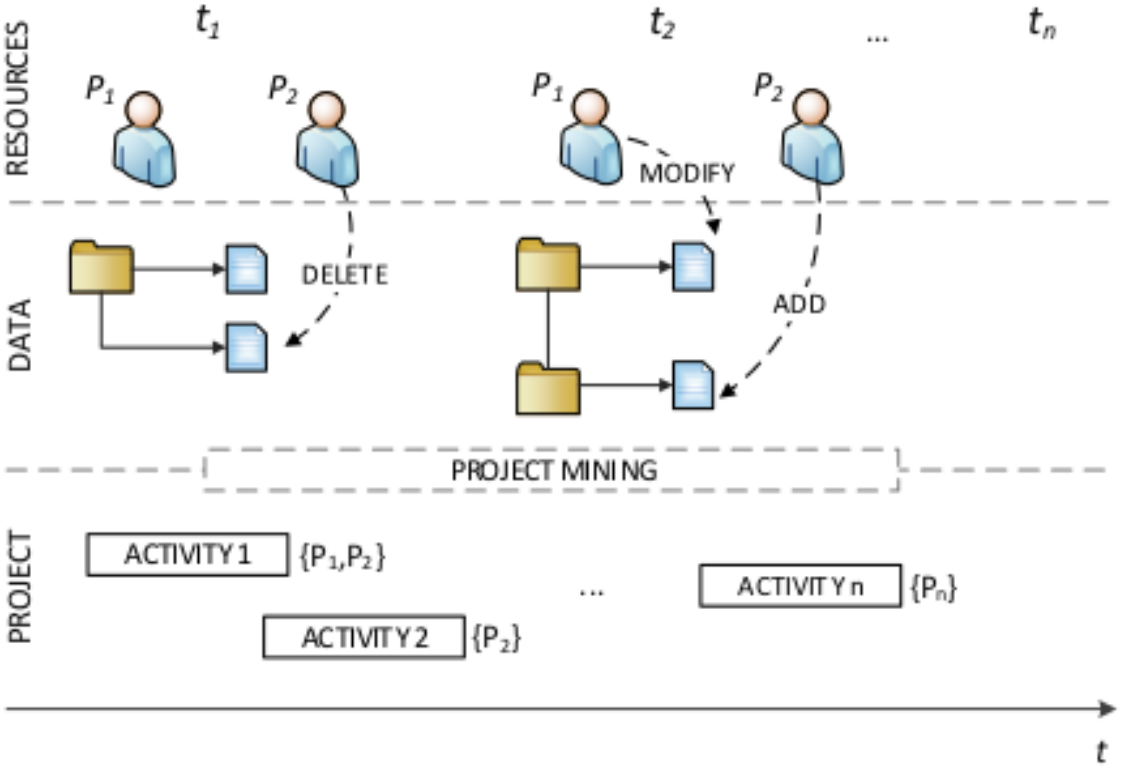
- What is really going on in your software development project?
- Why are deadlines not met? Why are the costs superior to the planned?
- Why is does the your software product require more maintenance than what you thought?

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



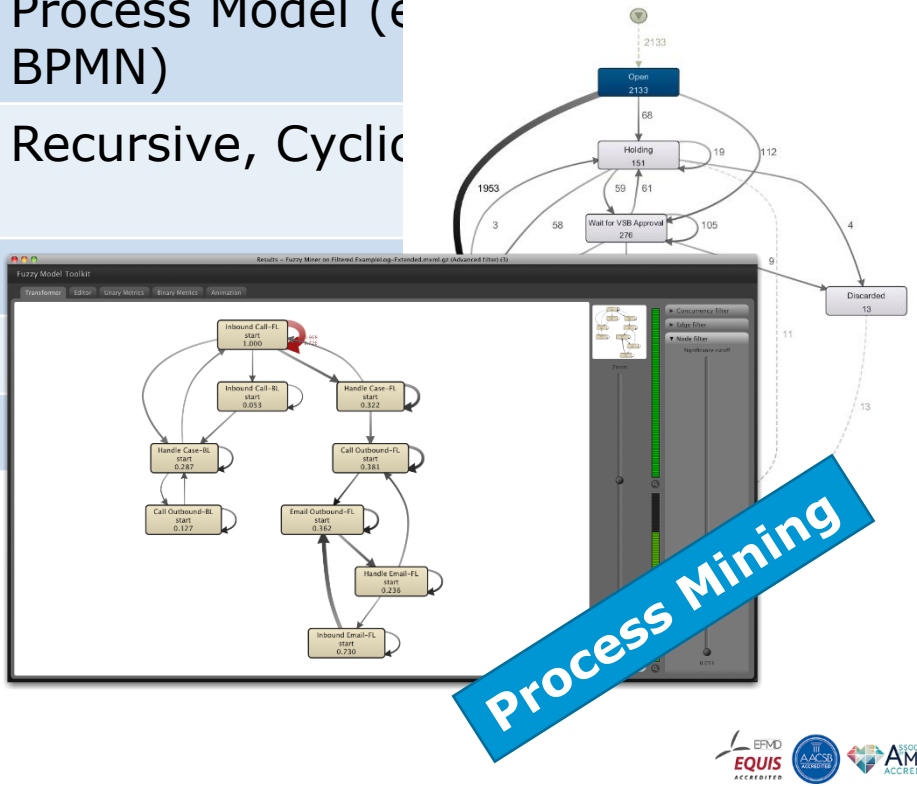
Project-Oriented vs. Classic Business Processes

Project-Oriented	Classic Processes
Plan (e.g. Gantt, PERT)	Process Model (e.g. Petri Net, BPMN)
One time (fixed goal and resources)	Recursive, Cyclic
Single instance	Many instances
Workpackages, Modules, Units	Activities
Subworkpackages, Submodules	Suprocesses

Project-Oriented vs. Classic Business Processes

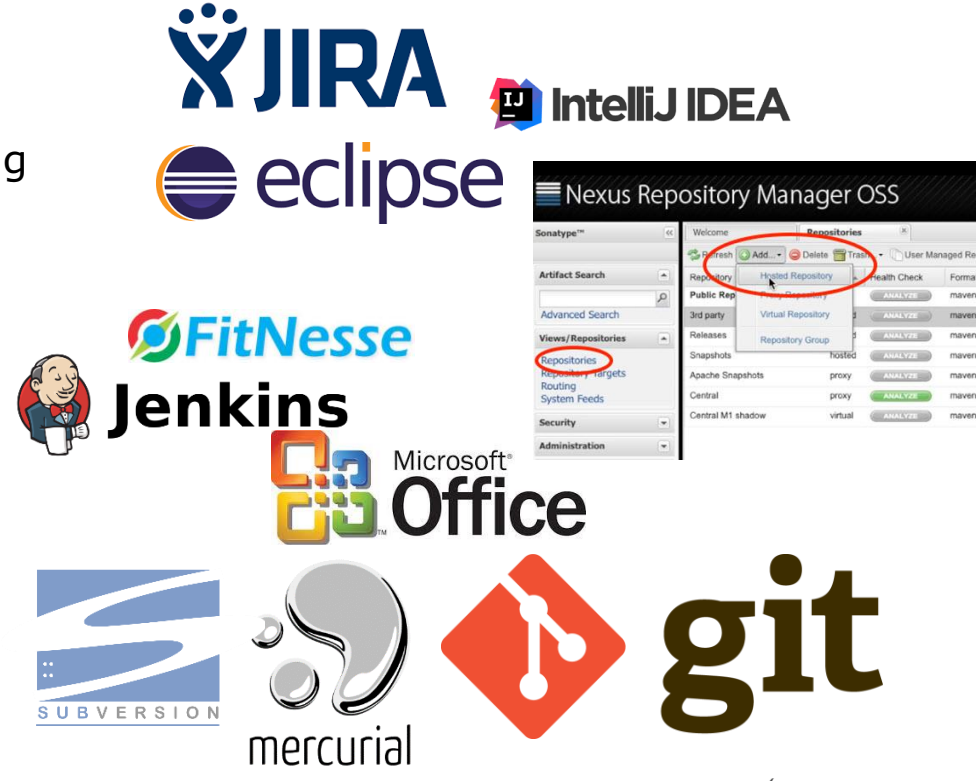
Project-Oriented	Classic Processes
Plan (e.g. Gantt, PERT)	Process Model (e… Petri Net, BPMN)
One time (fixed goal and resources)	Recursive, Cyclic
Single instance	
Workpackages, Modules, Units	
Subworkpackages, Submodules	

?

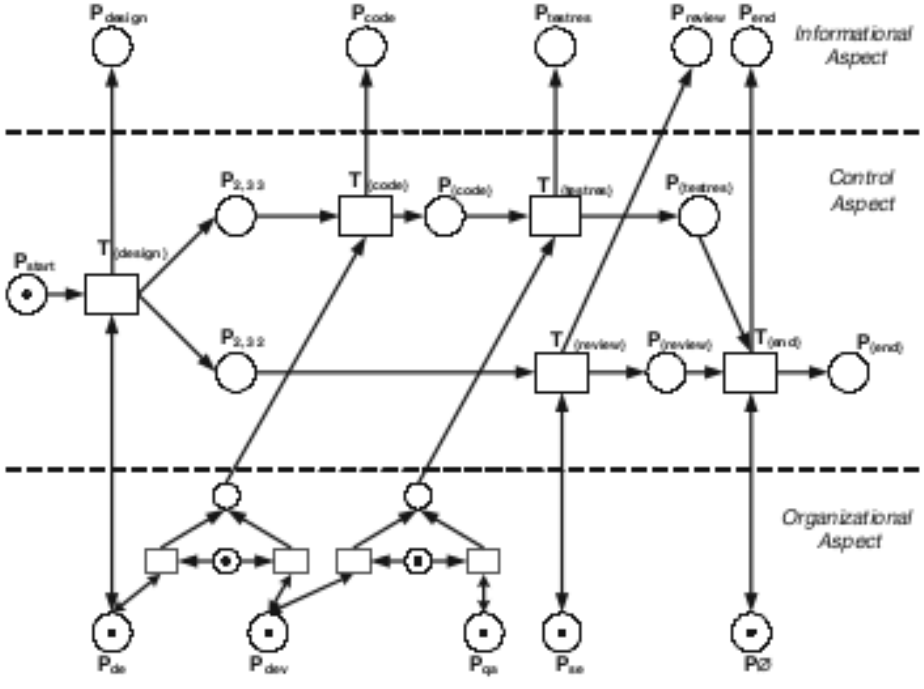


Software Projects Data

- Software projects are supported by a variety of tools
- Examples
  - Project management, Bug-tracking
  - Development
  - Dependency management
  - Testing
  - Continuous integration
  - Documentation
  - **Version Control System**

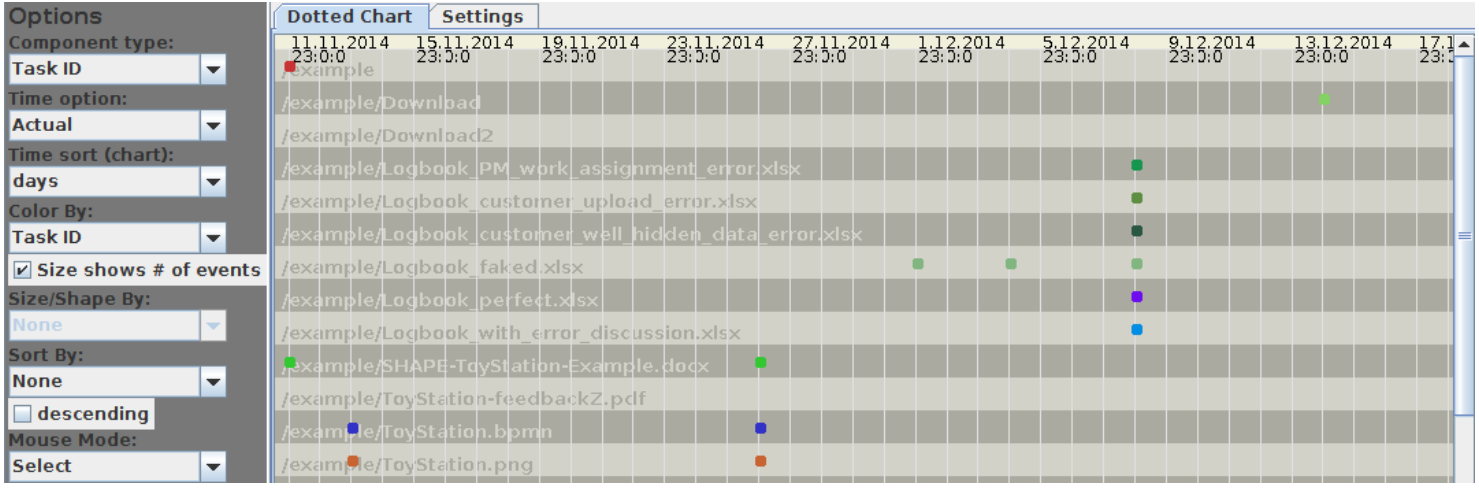


State of the Art: Activity Mining



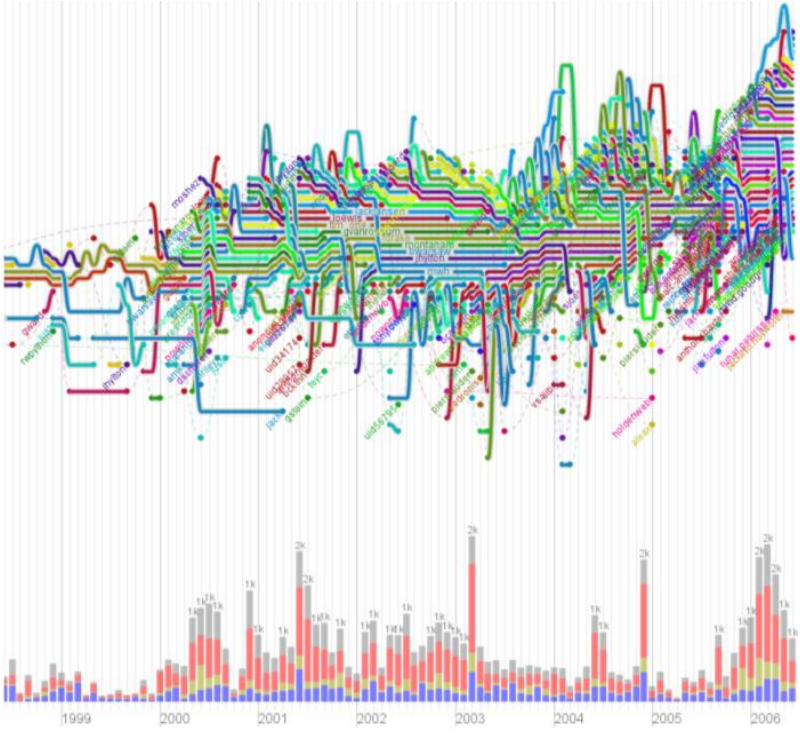
Kindler et al. 2006

State of the Art: Dotted Chart



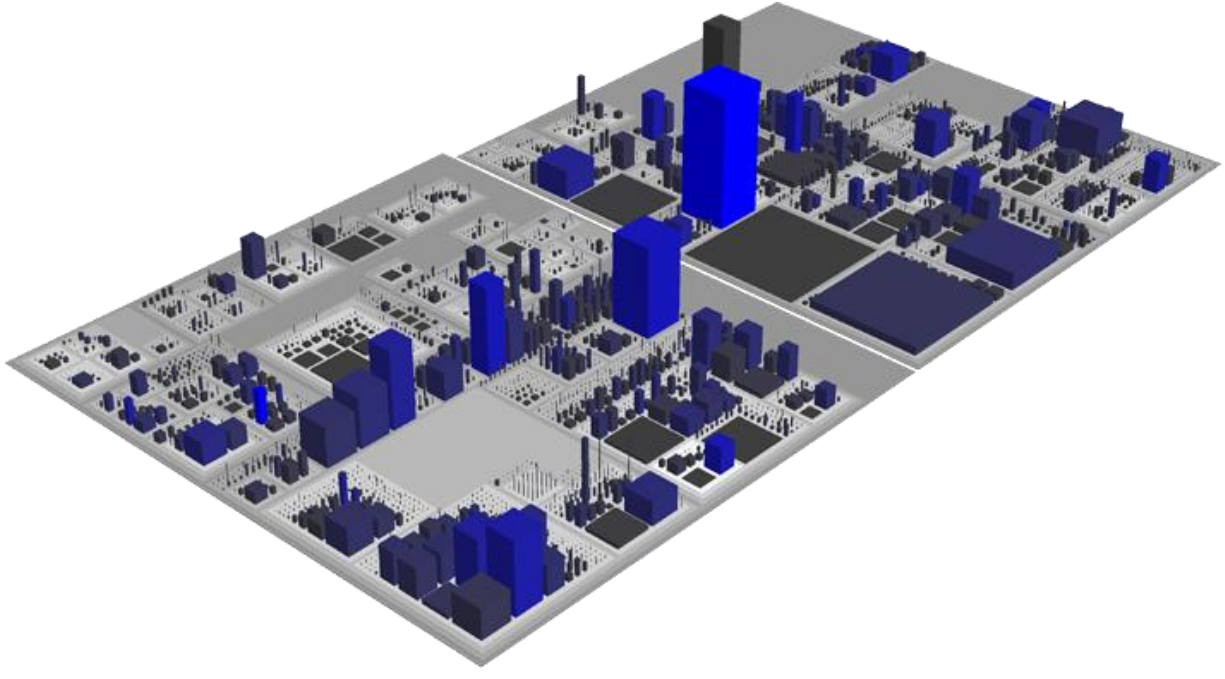
Song and van der Aalst. 2007

State of the Art: Evolution Storylines



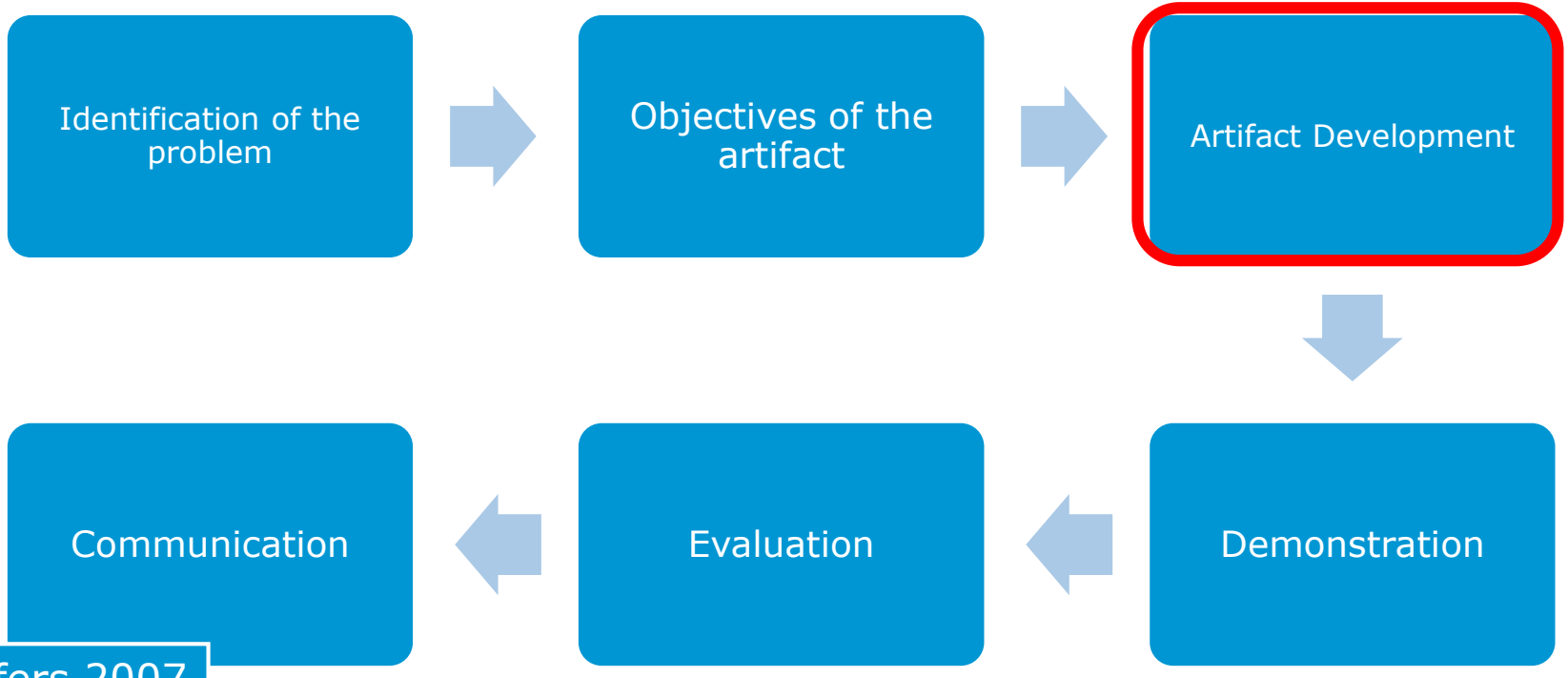
Ogawa and Ma 2010

State of the Art: Visual Software Analytics



Wettel and Lanza 2007

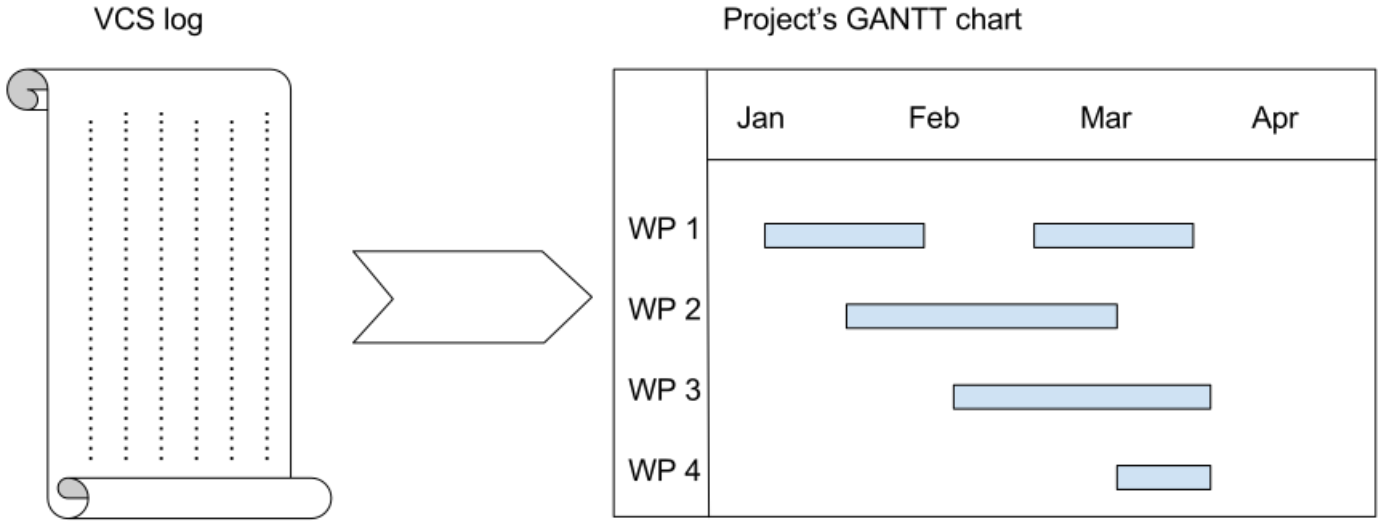
Methodology: DSR



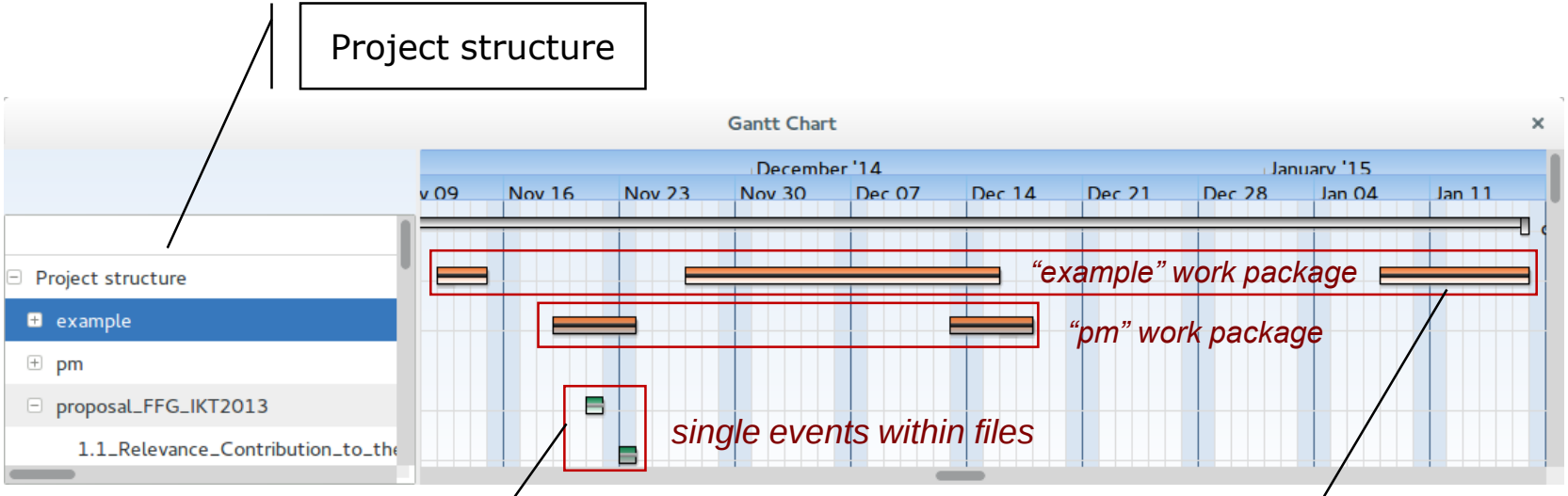
Peffers 2007

Mining the Real Gantt Chart

Mining the Gantt Chart of a Project



Example from the SHAPE project



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**)

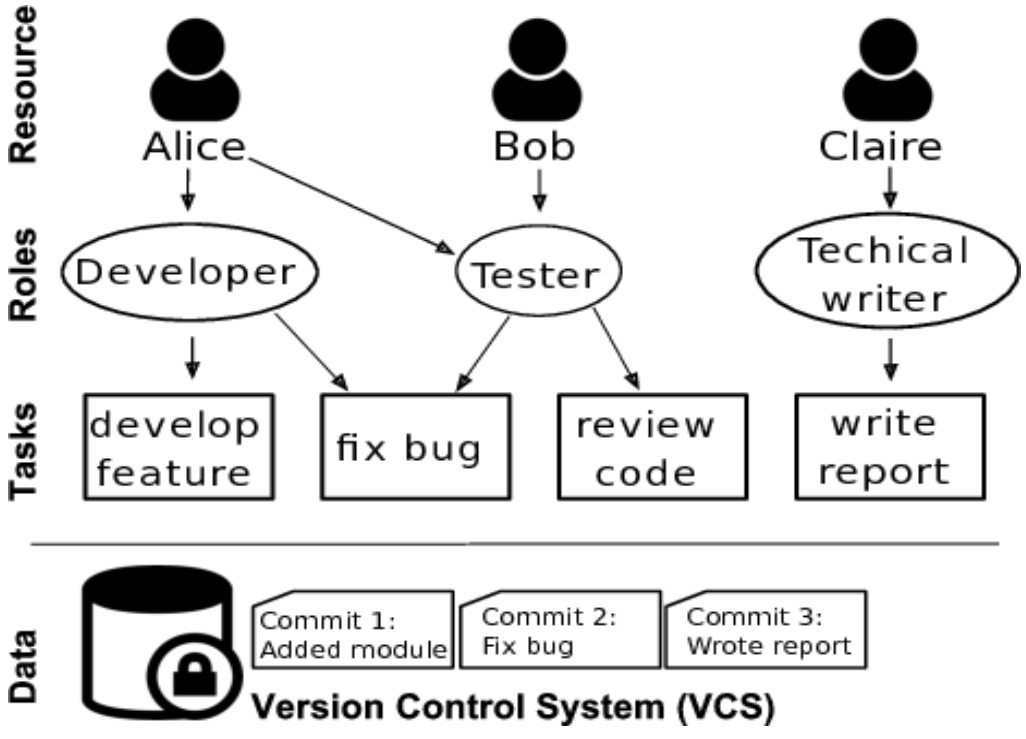
  average work speed (commit frequency) during active times

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

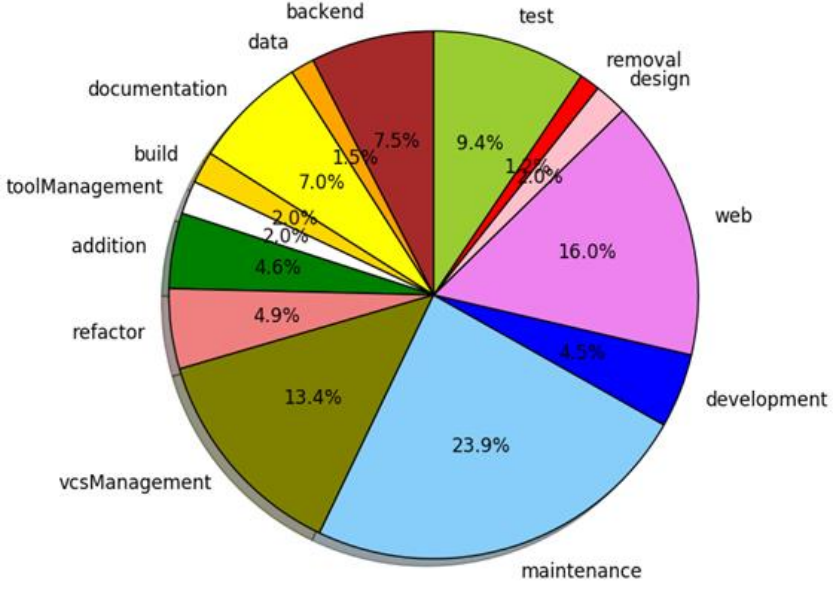
Resource Classification from Commit Messages

Resource Classification from Commit Messages

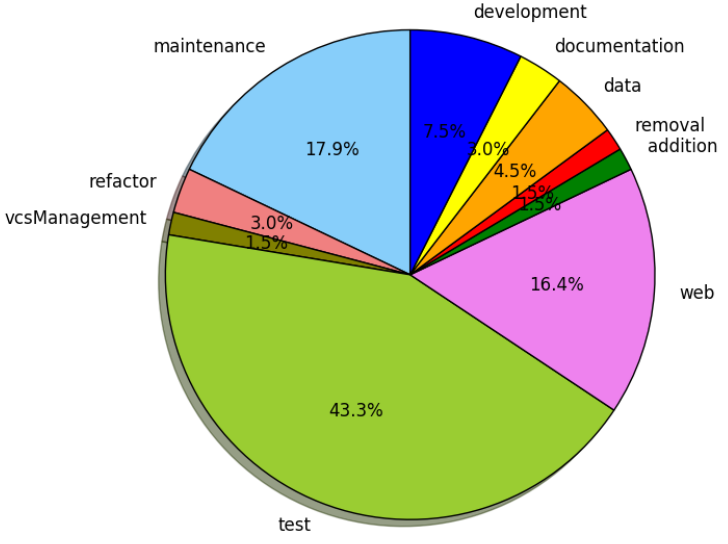


Resource Classification from Commit Messages: Developer vs. Tester

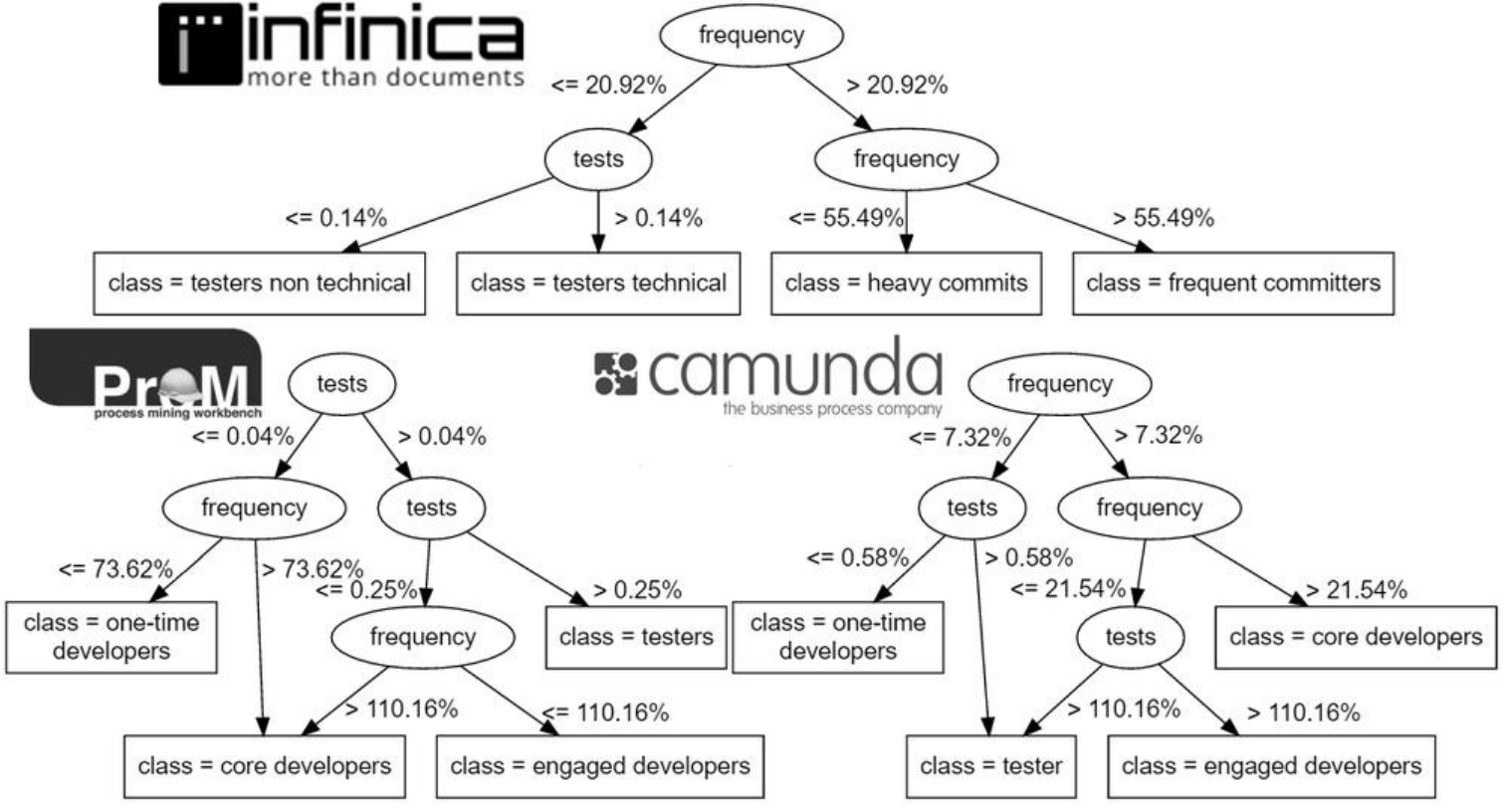
Developer



Tester



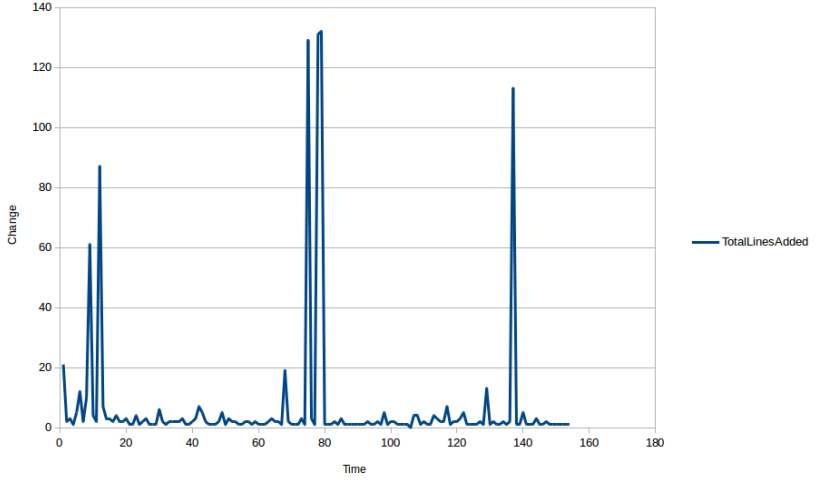
Learning Decision Trees from Projects

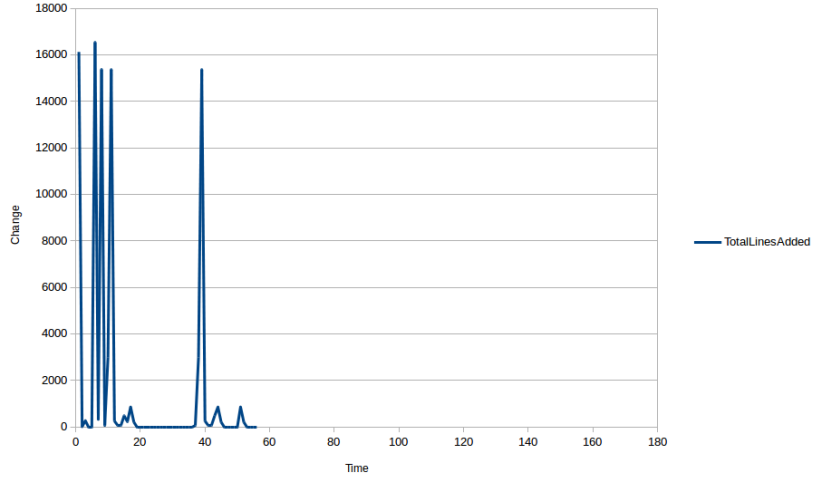


Mining Hidden Work Dependencies

Artifact Evolution as Time Series

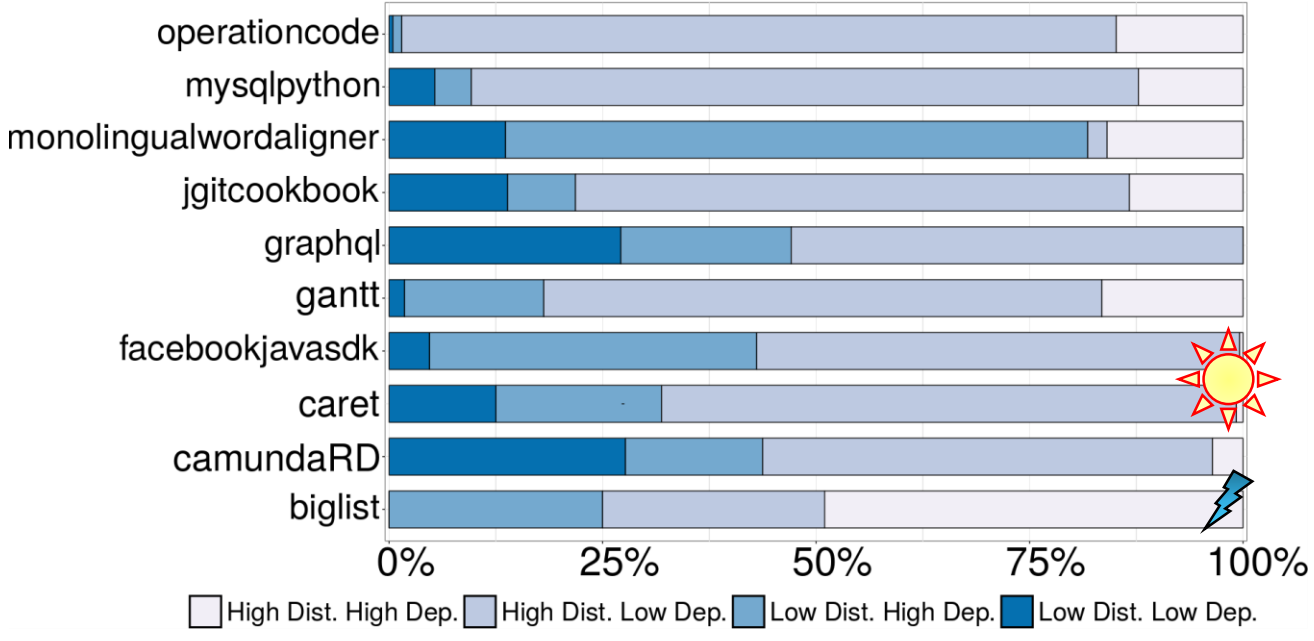
Are they similar?





Correlation!

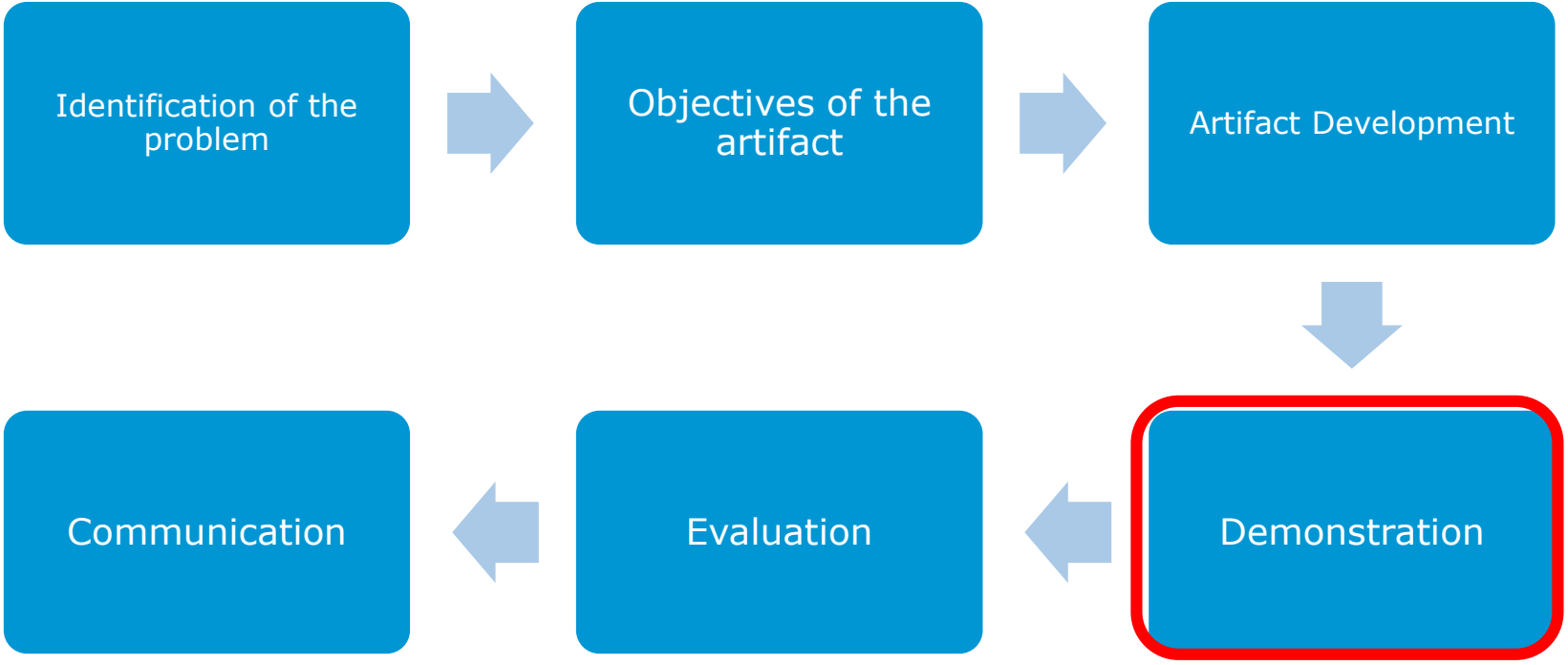
Characterization of Projects wrt Dependencies



Publications

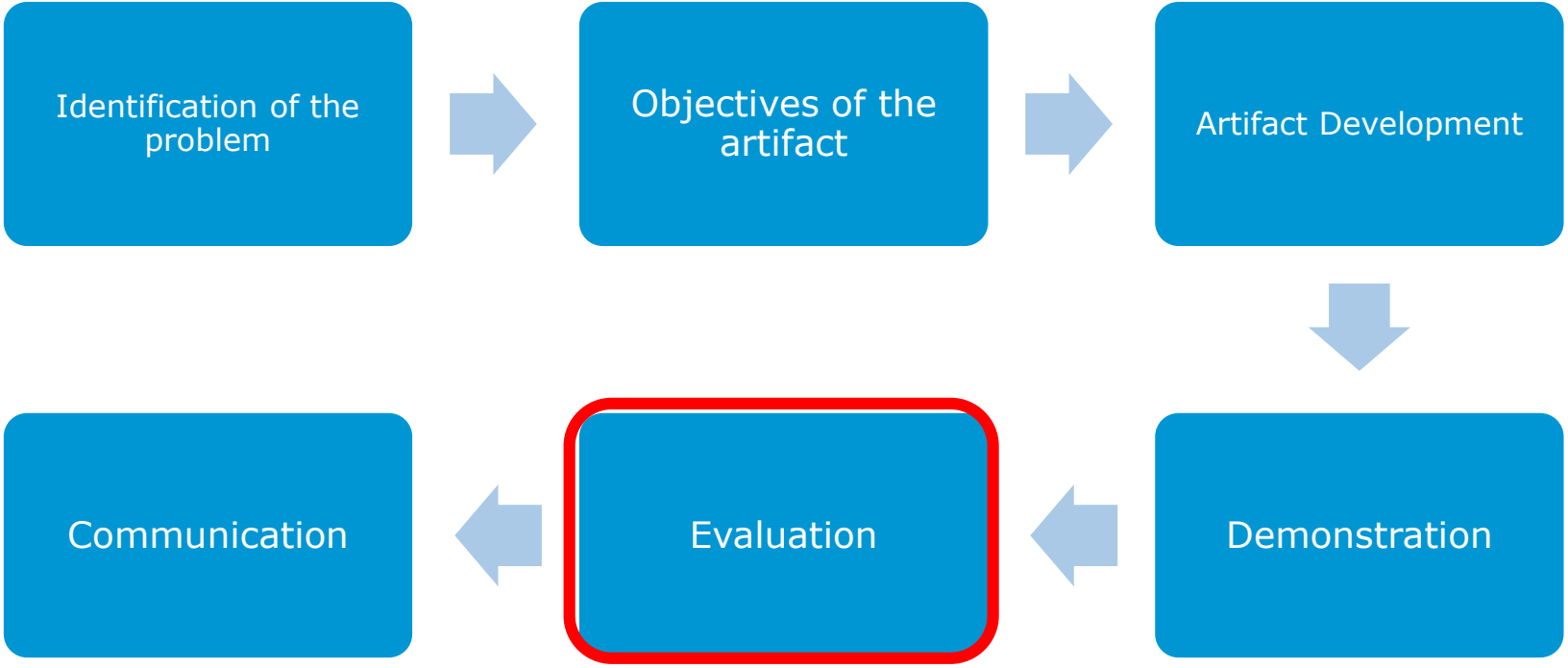
- Bala, S., Cabanillas, C., Mendling, J., Rogge-Solti, A., Polleres, A.: Mining Project-Oriented Business Processes. In: BPM. pp. 425–440 (2015).
- Agrawal, K., Aschauer, M., Thonhofer, T., Tomsich, N., Bala, S., Rogge-Solti, A.: Resource Classification from Version Control System Logs. In: EDOC Workshops (2016).
- Bala, S., Havur, G., Sperl, S., Steyskal, S., Haselböck, A., Mendling, J., Polleres, A.: SHAPEworks: A BPMS Extension for Complex Process Management. In: BPM (Demos). pp. 50–55. (2016).
- Bala, S., Revoredo, K., Mendling, J., Santoro, F.: Uncovering the Hidden Co-Evolution in the Work History of Software Projects. In: BPM. 2017 (*conditionally accepted*)

Next steps



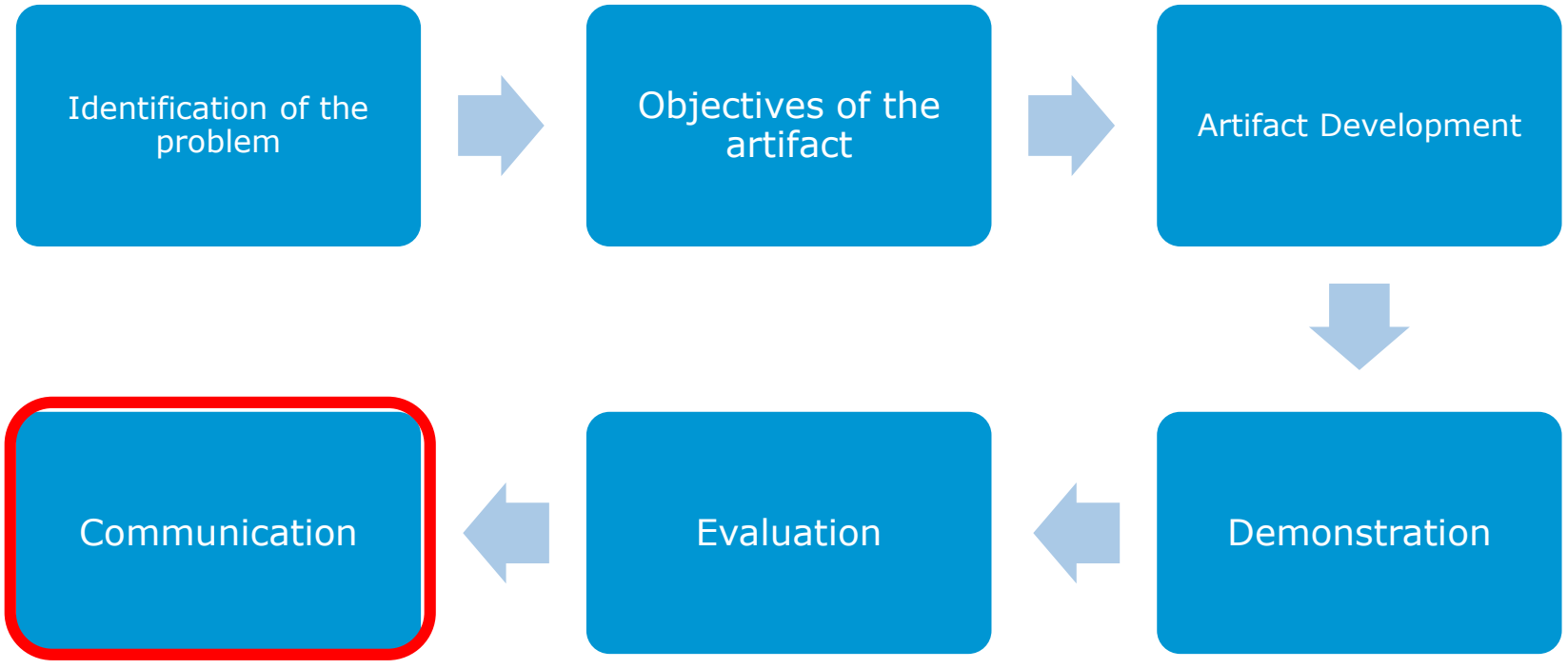
Peffers 2007

Next steps



Peffers 2007

Next steps



Peffers 2007

Questions?

VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

DEPARTMENT OF INFORMATION SYSTEMS AND OPERATIONS
INSTITUTE FOR INFORMATION BUSINESS

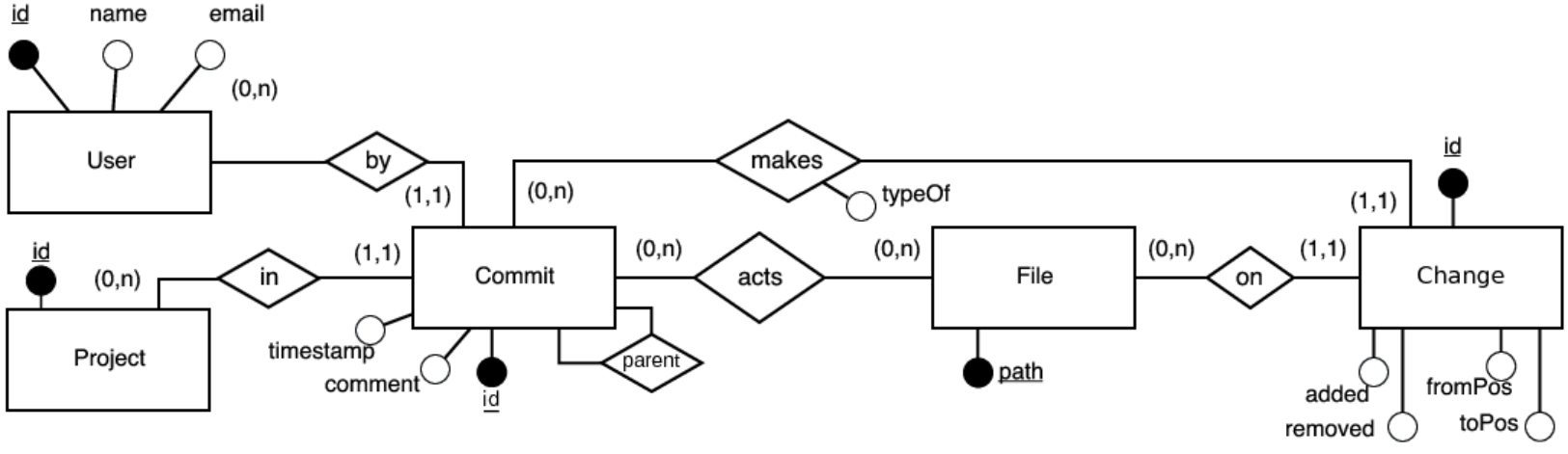
Welthandelsplatz 1, D2/1.026
1020 Vienna, Austria

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

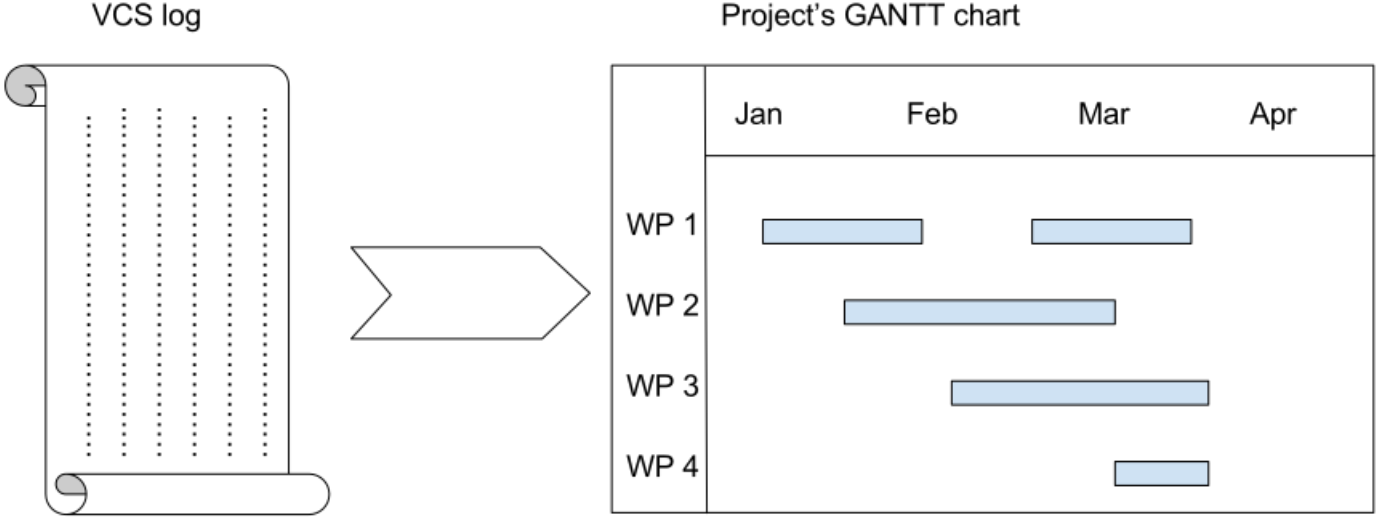
Backup slides

Data Model for SQL Querying VCS logs

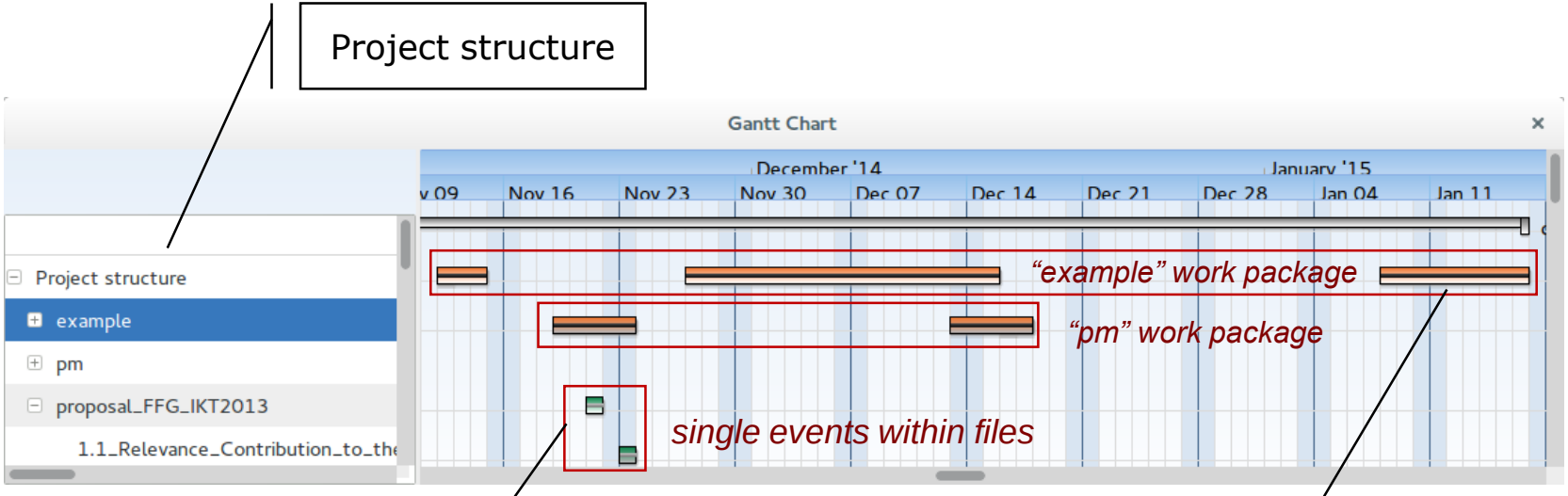


Mining the Real Gantt Chart

Mining the Gantt Chart of a Project



Example from the SHAPE project



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

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

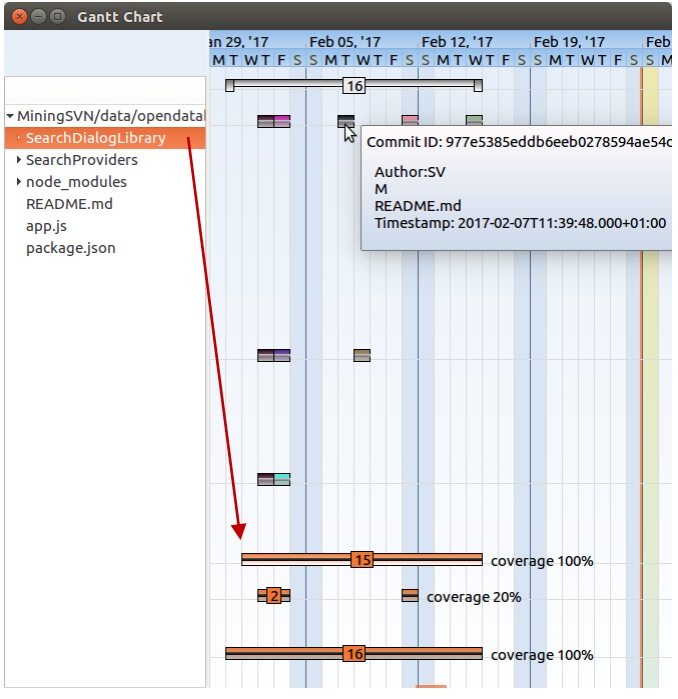
Some Real World Projects

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

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)

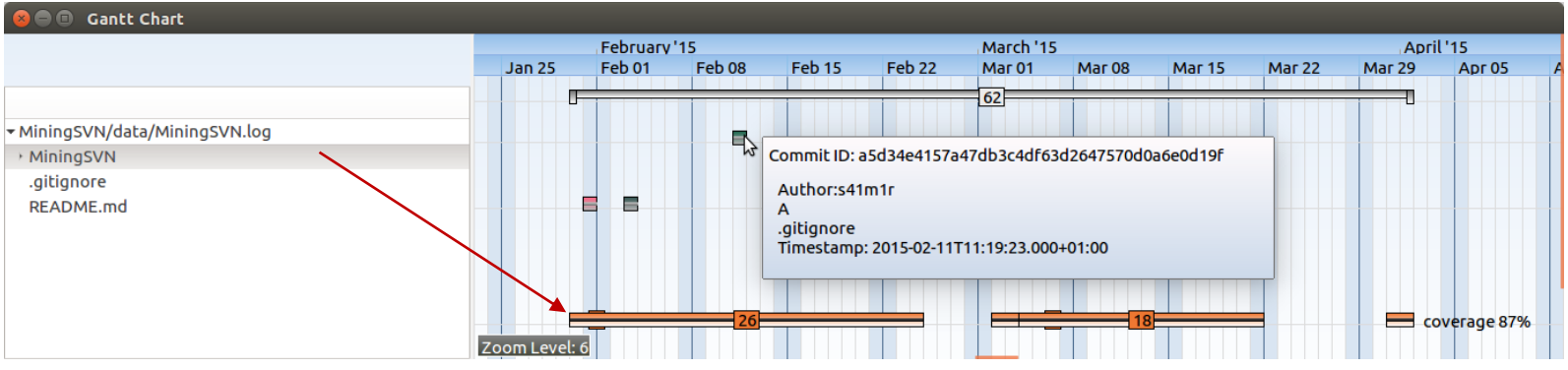
Open-Data Helper Bot

- 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



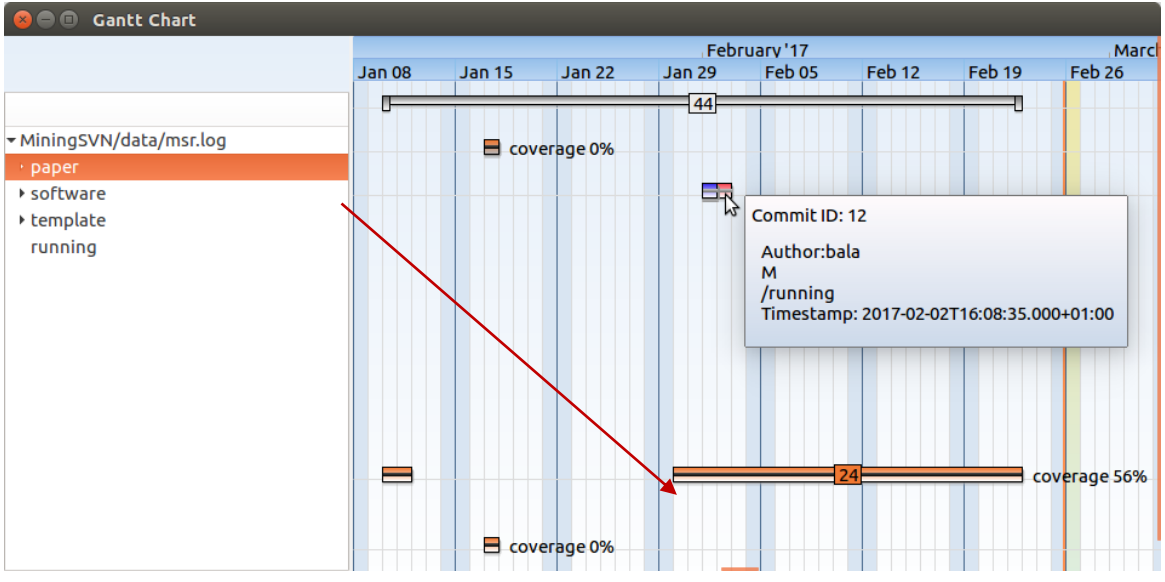
Mining VCS Software

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



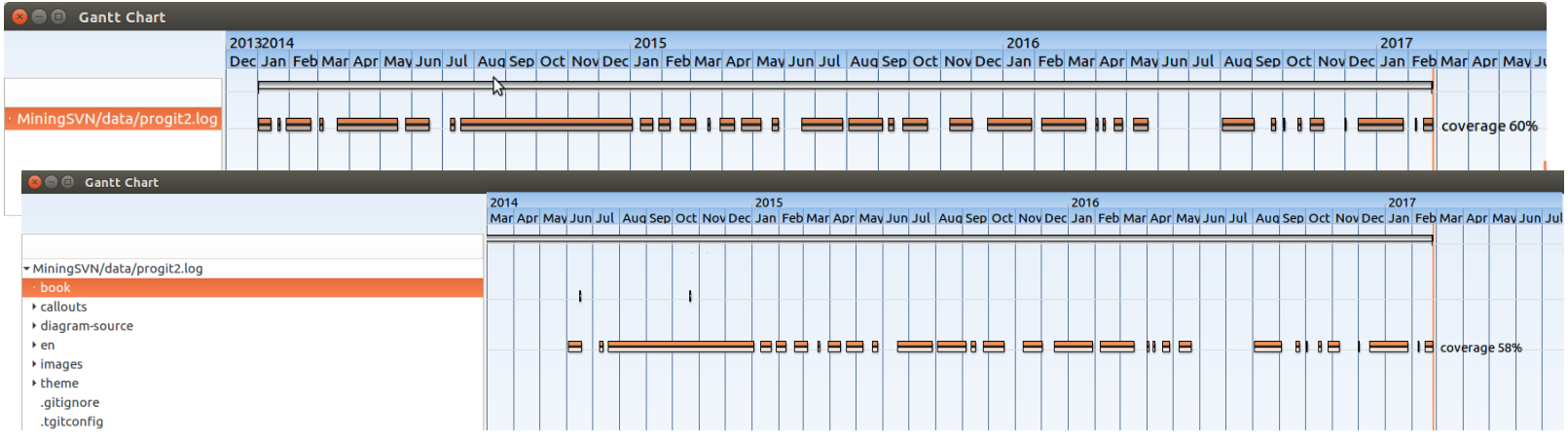
MSR Paper

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

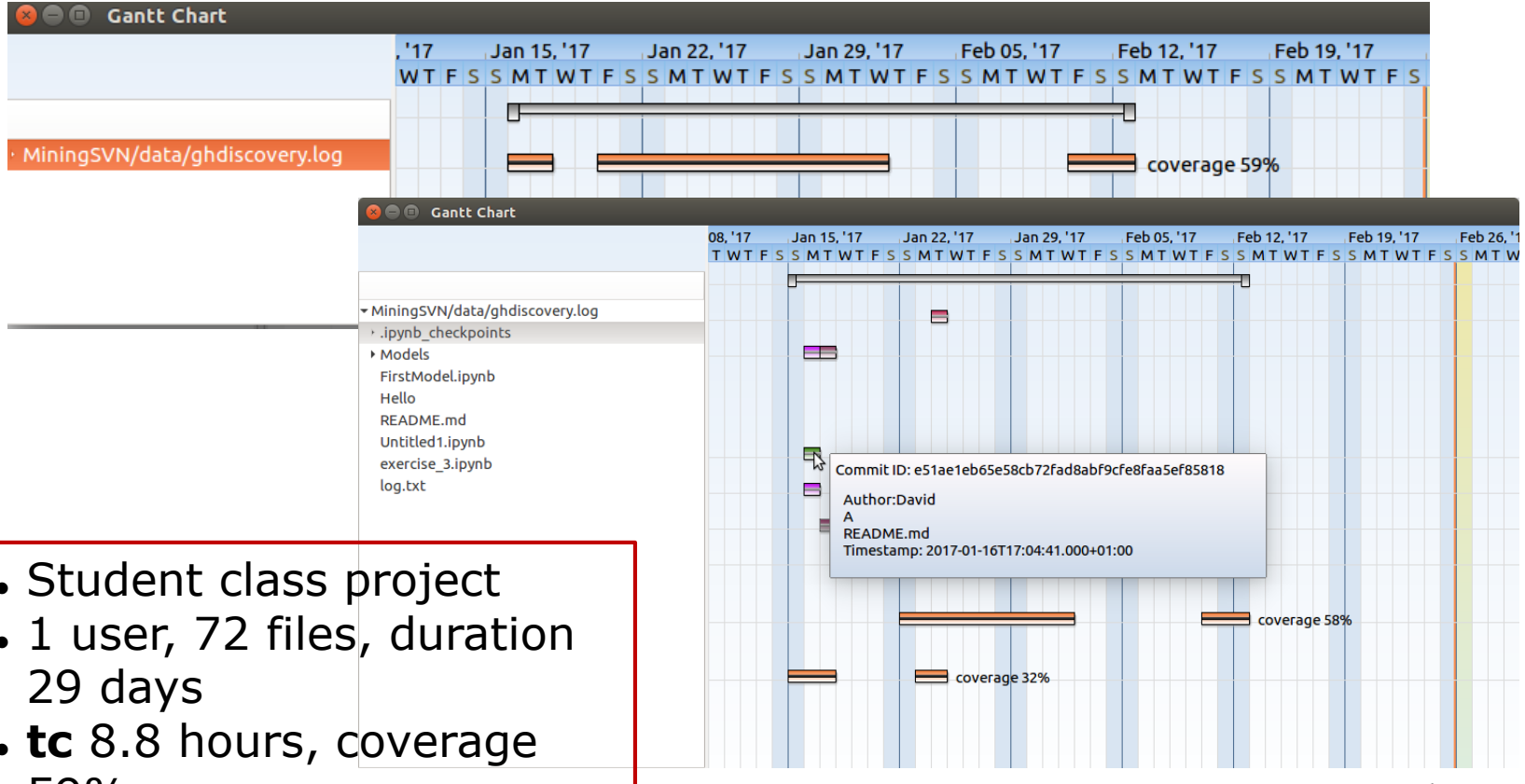


Book Writing Project

- Progit book 2nd edition
- 1292 commits, 134 users, 955 files, 481 days
- **tc** 118.6 hours, coverage 60%



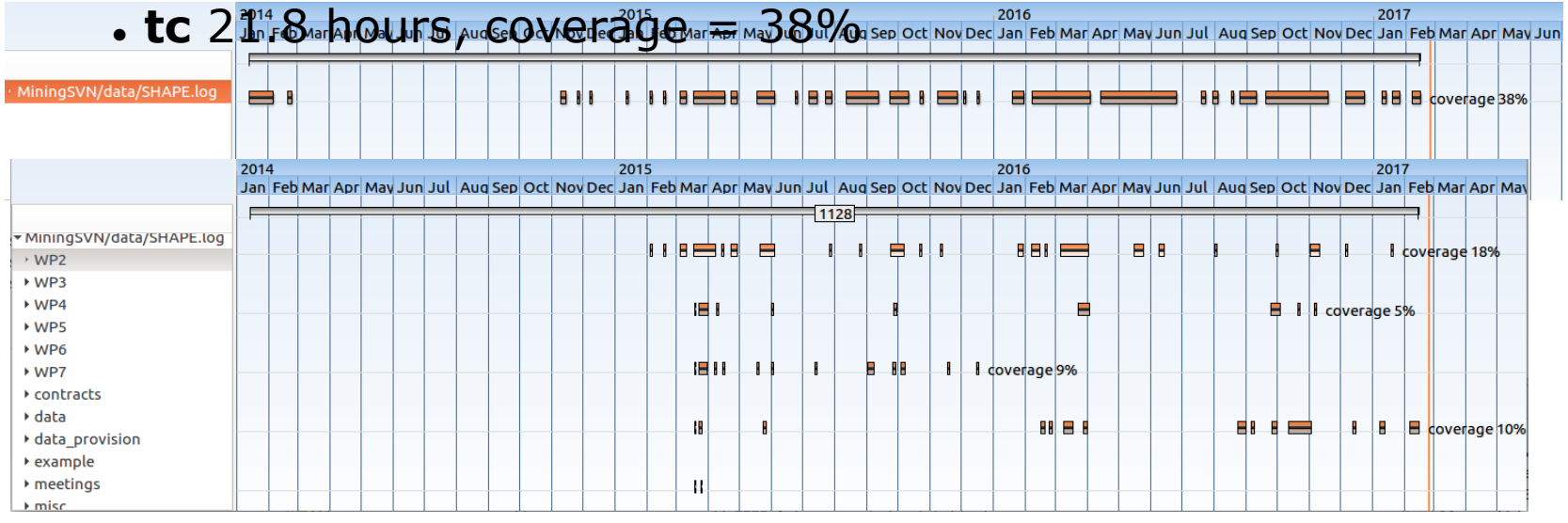
Students Project: Disovering Github Activities



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

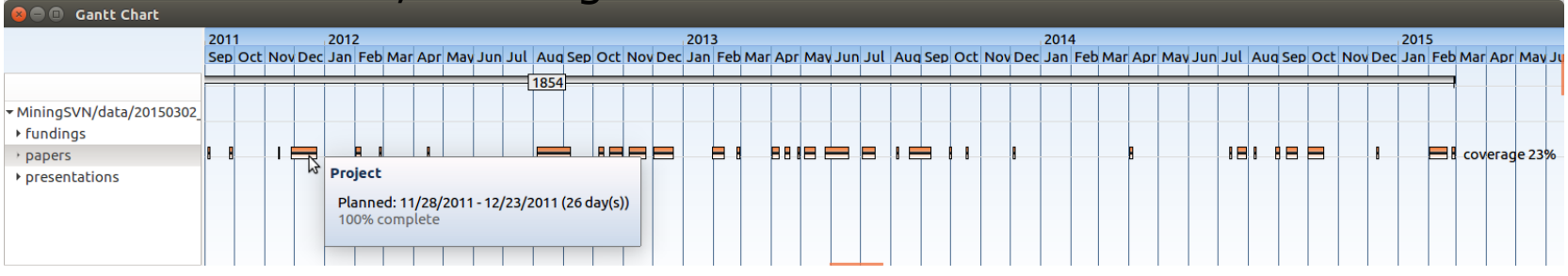
SHAPE Project

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



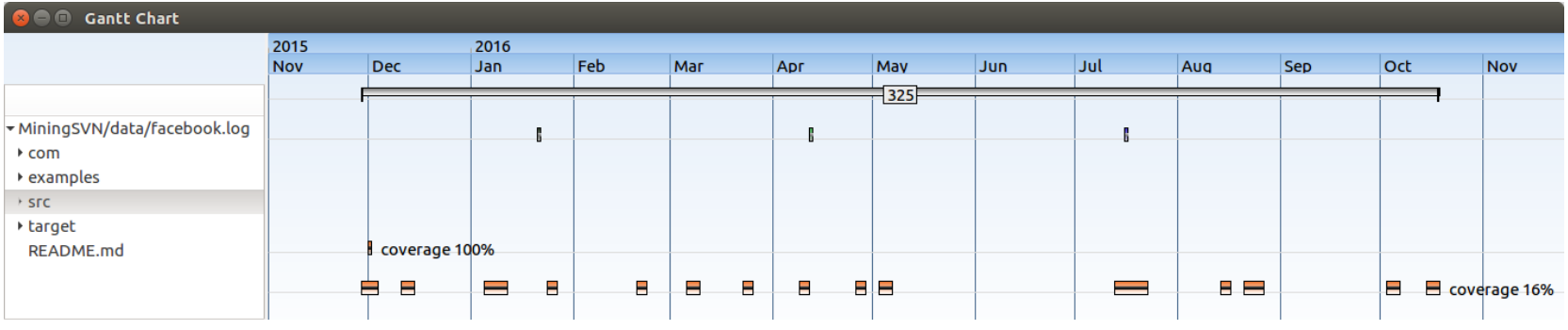
Writing Papers Project (from Industry)

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



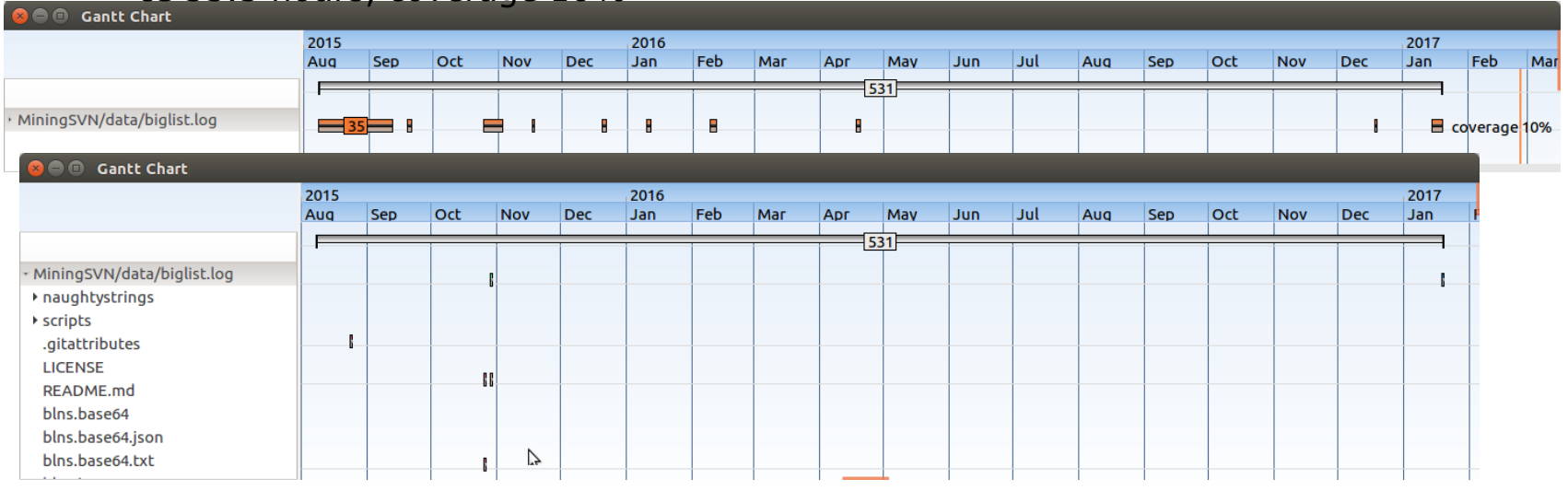
Facebook ads java sdk

- 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

WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

Requirements

- 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 → *co-evolution* of two artifacts?

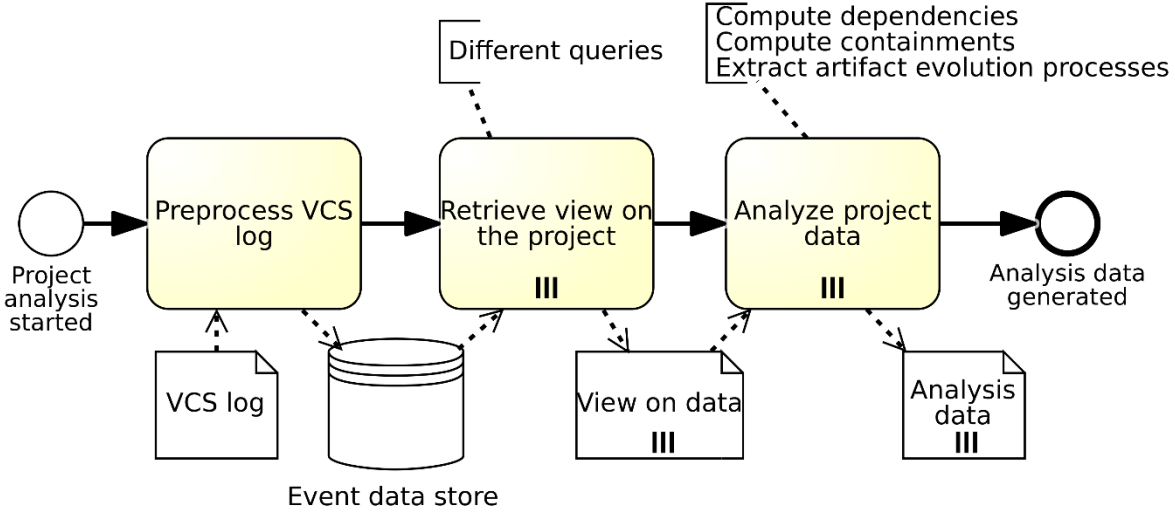
R3 (Measure Dependencies)

- How strongly depend two artifacts on one another?

State of the Art

MSR	Process Mining	Visualization
• Mostly solving R2 (Uncover work-related dependencies and R3 (Measure dependencies) • Zaidman et al. 2008, Zimmerman et al. 2008, D'Ambros et al. 2009, Lindeberg et al. 2016	• Mostly addressing R1 (Extract the work history) • 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


Different queries
Compute dependencies
Compute containments
Extract artifact evolution processes
Preprocess VCS log
Retrieve view on the project
Analyze project data
Project analysis started
Analysis data generated
VCS log
Event data store
View on data
Analysis data

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?

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

Concepts

Artifact evolution

- Changes made to an artifact during its lifetime, meaured in Lines of Code

Dependency

- High similarity in the evolution of two software artifacts

Metrics

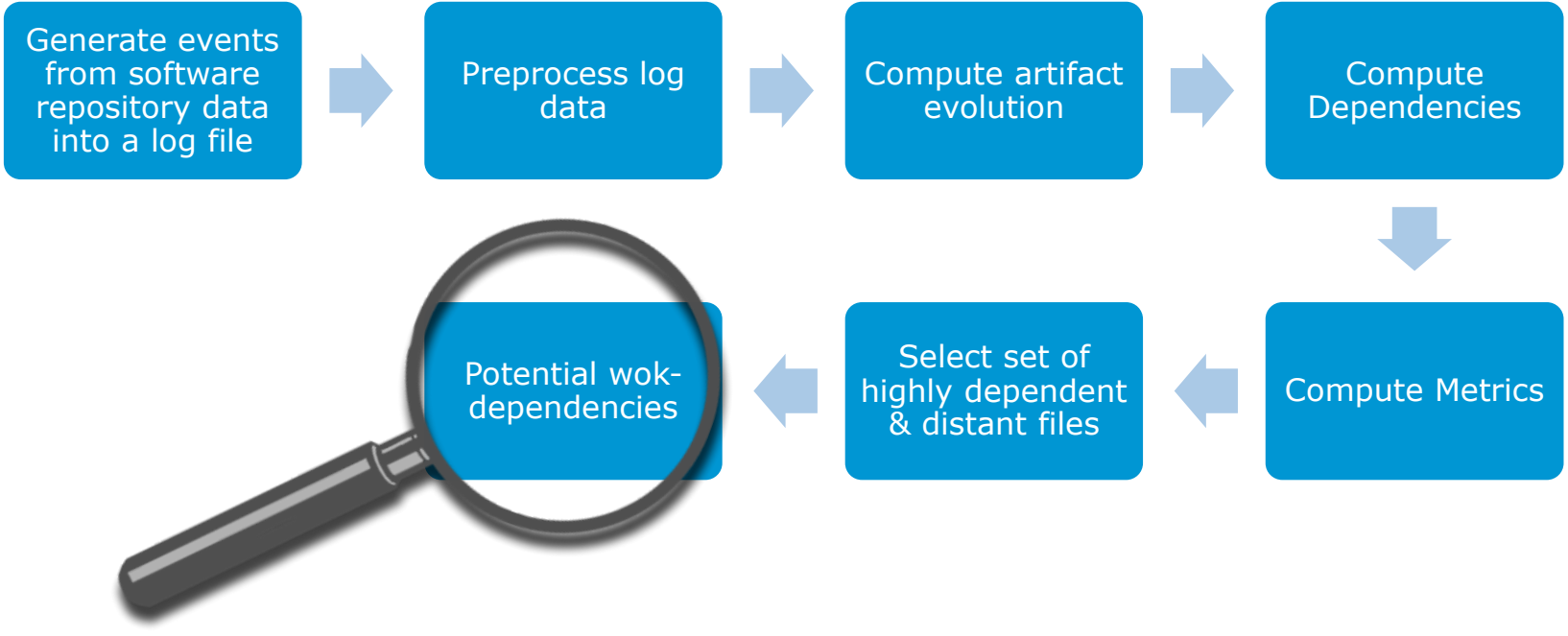
Degree of Co-Evolution

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

File Distance

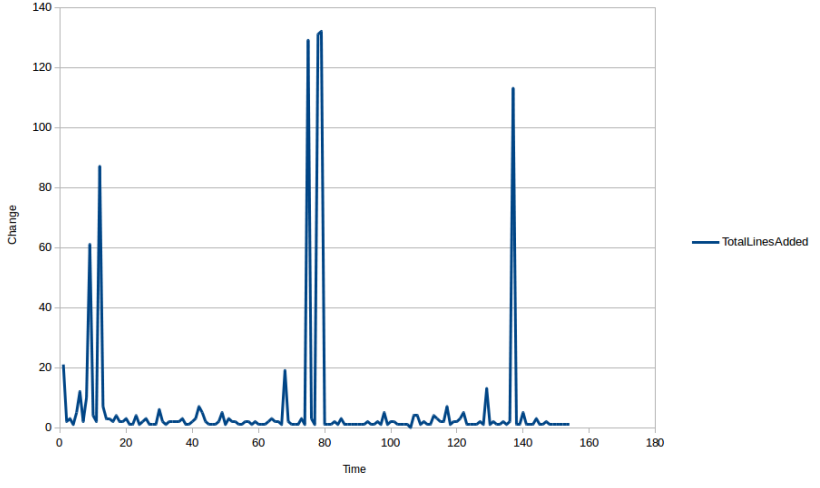
- Distance between two files in the file tree. Equal to the length of the path traversing the least common ancestor.

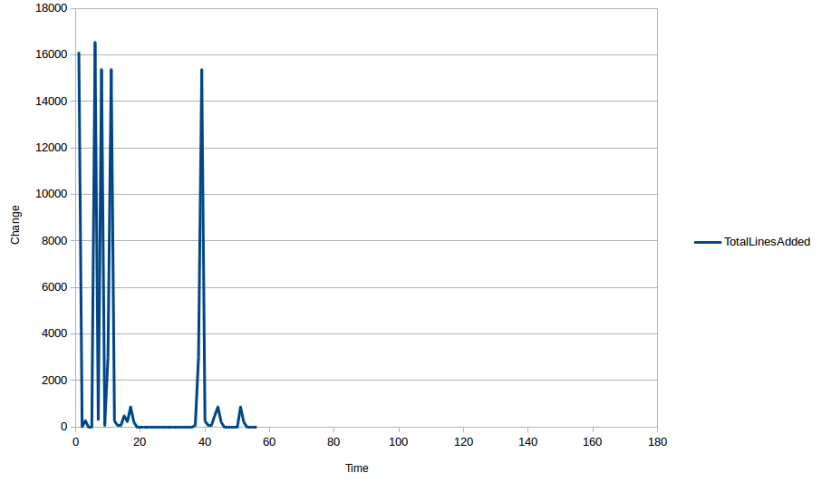
Method


Generate events from software repository data into a log file
Preprocess log data
Compute artifact evolution
Compute Dependencies
Compute Metrics
Select set of highly dependent & distant files
Potential wok-dependencies

Computing Dependencies

Are they similar?



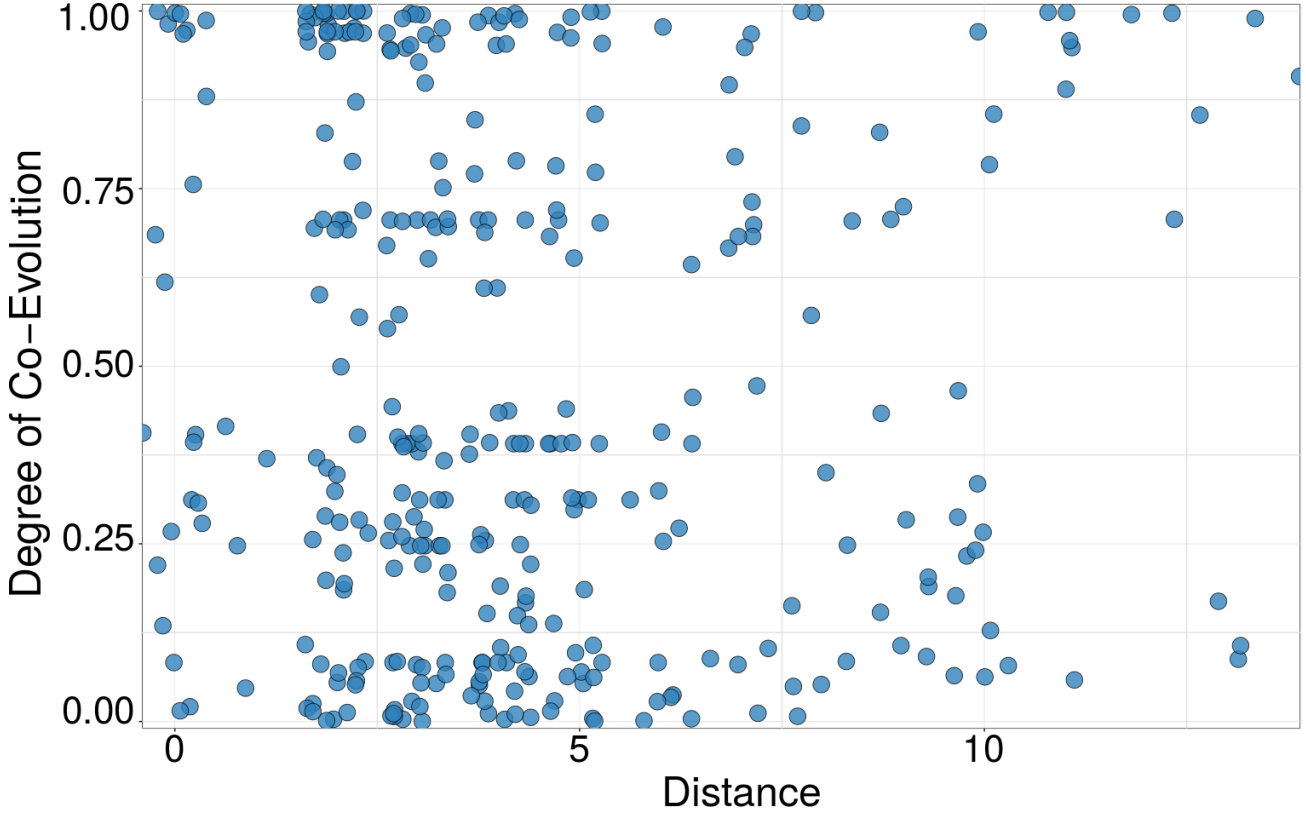


Correlation!

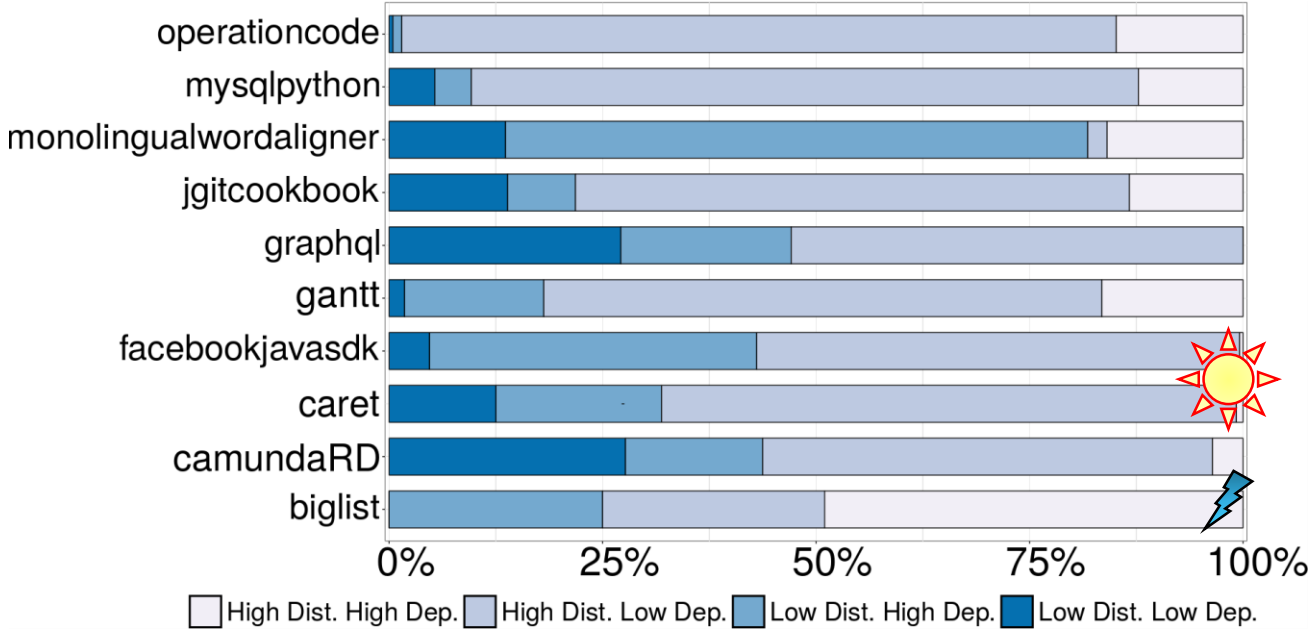
Results

Project	Commits	Files	χ^H	χ^L	(d^L, χ^L)	(d^L, χ^H)	$(d^H, \chi^L,)$	(d^H, χ^H)	$\overline{\lvert p_f \rvert}$	$max(\lvert p_f \rvert)$	$\lvert A_{evo} \rvert$	$\overline{d}$	$max(d)$
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
facebookjavasdk	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

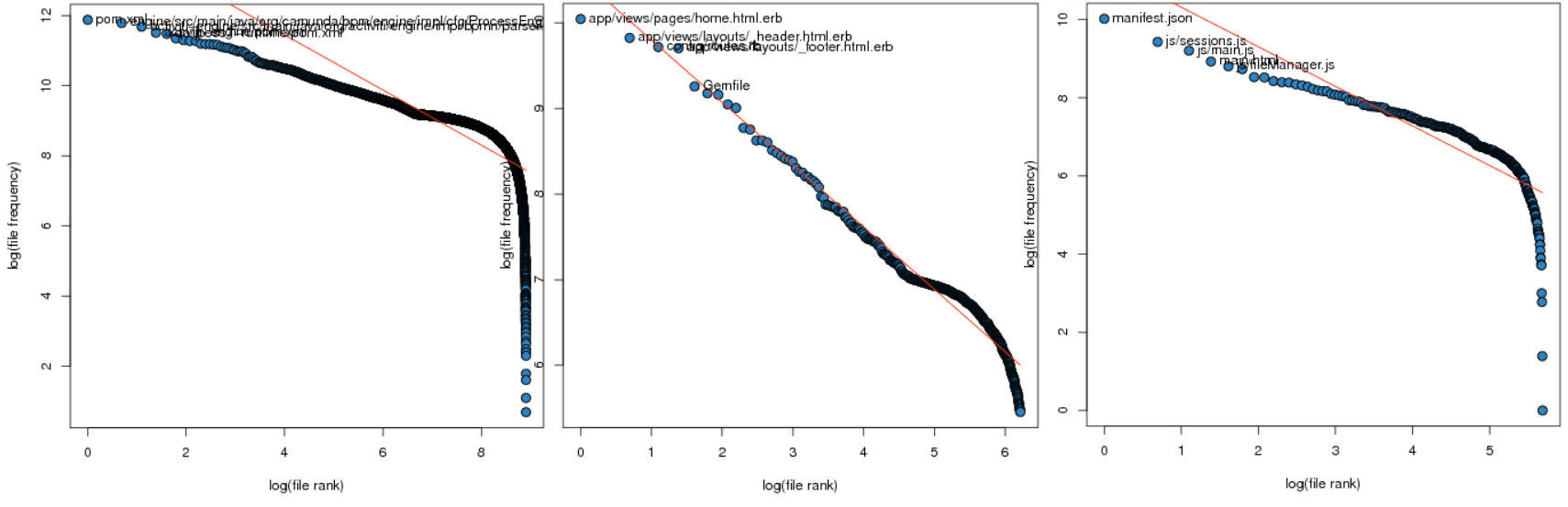
Co-Evolution versus Distance



Characterization of Projects wrt Dependencies



Zipf law on real projects: 100%

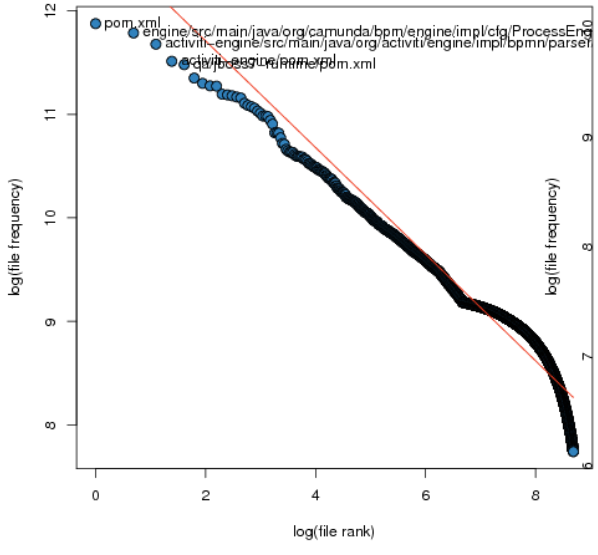


Camunda

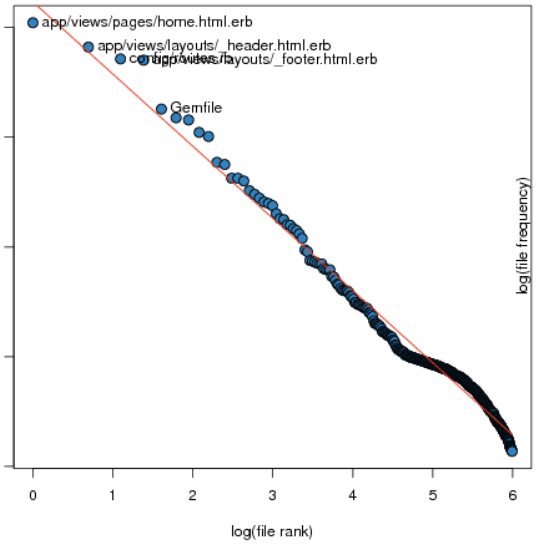
Operationcode

Caret

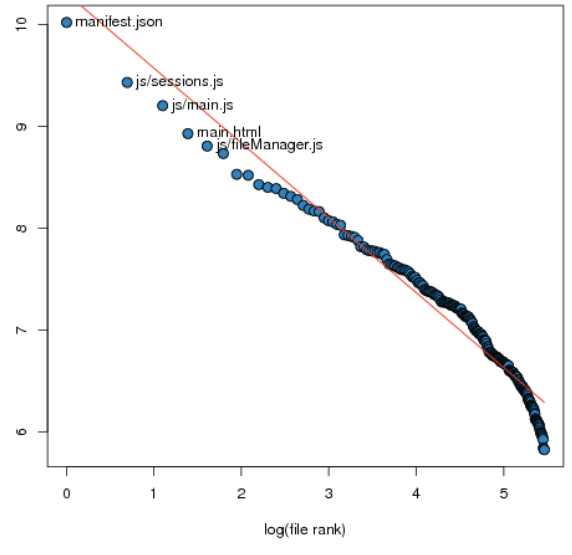
Zipf law on real projects: top 80%



Camunda

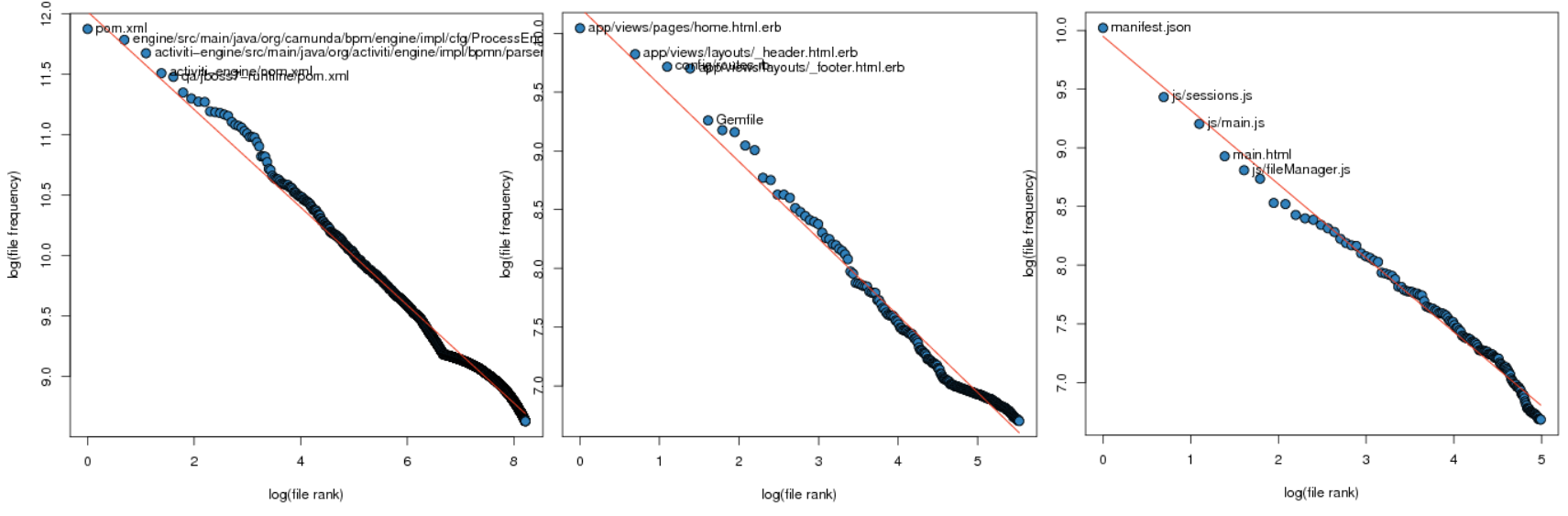


Operationcode



Caret

Zipf law on real projects: top 50%

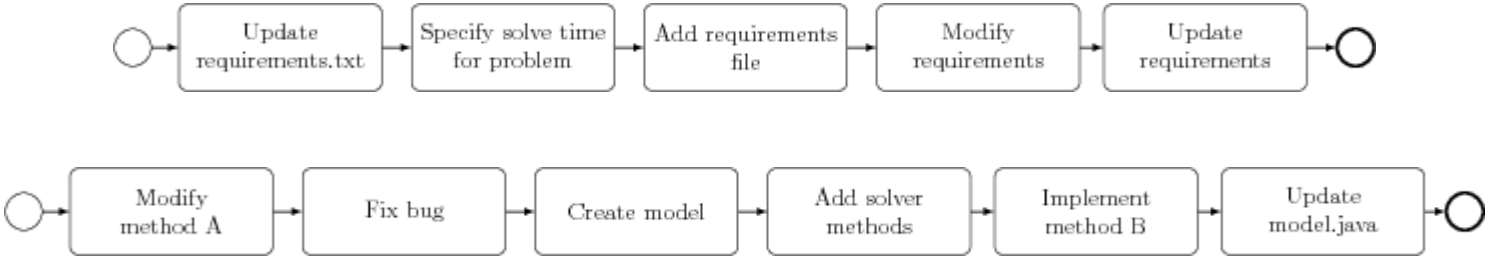


Camunda

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 comparting time series
  - Semantic analysis of process labels