Replication and Benchmarking in Software Analytics

Harald Gall

University of Zurich, Switzerland

http://seal.ifi.uzh.ch

@ BENEVOL, Mons, 17 Dec 2013

Panel @ ESEC/FSE 2013

Panel

Empirical answers to fundamental software engineering problems

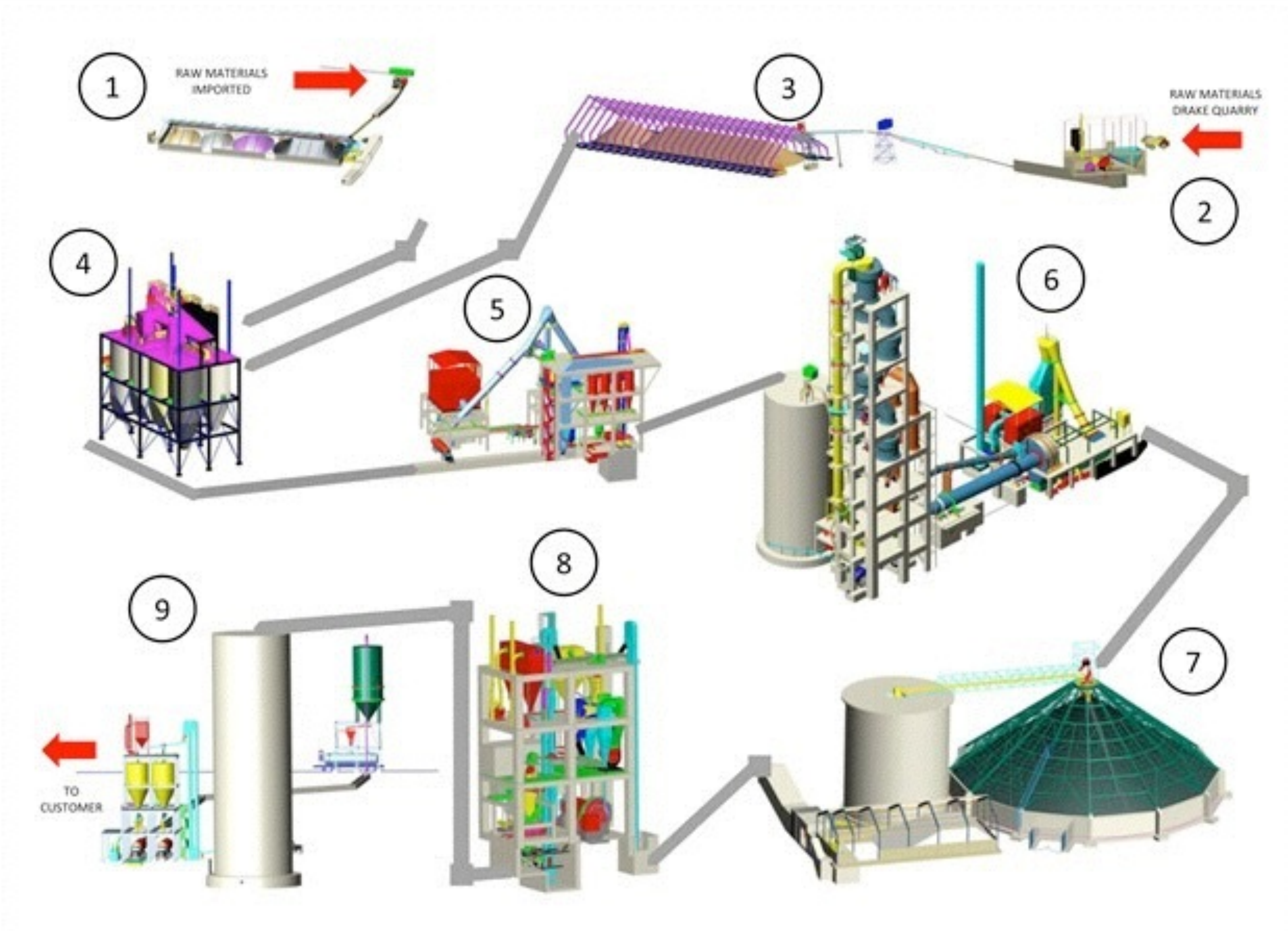
Wednesday, August 21
17:00-18:15, Column Hall

Chair
Bertrand Meyer (ETH Zurich, Switzerland, and ITMO, Russia)

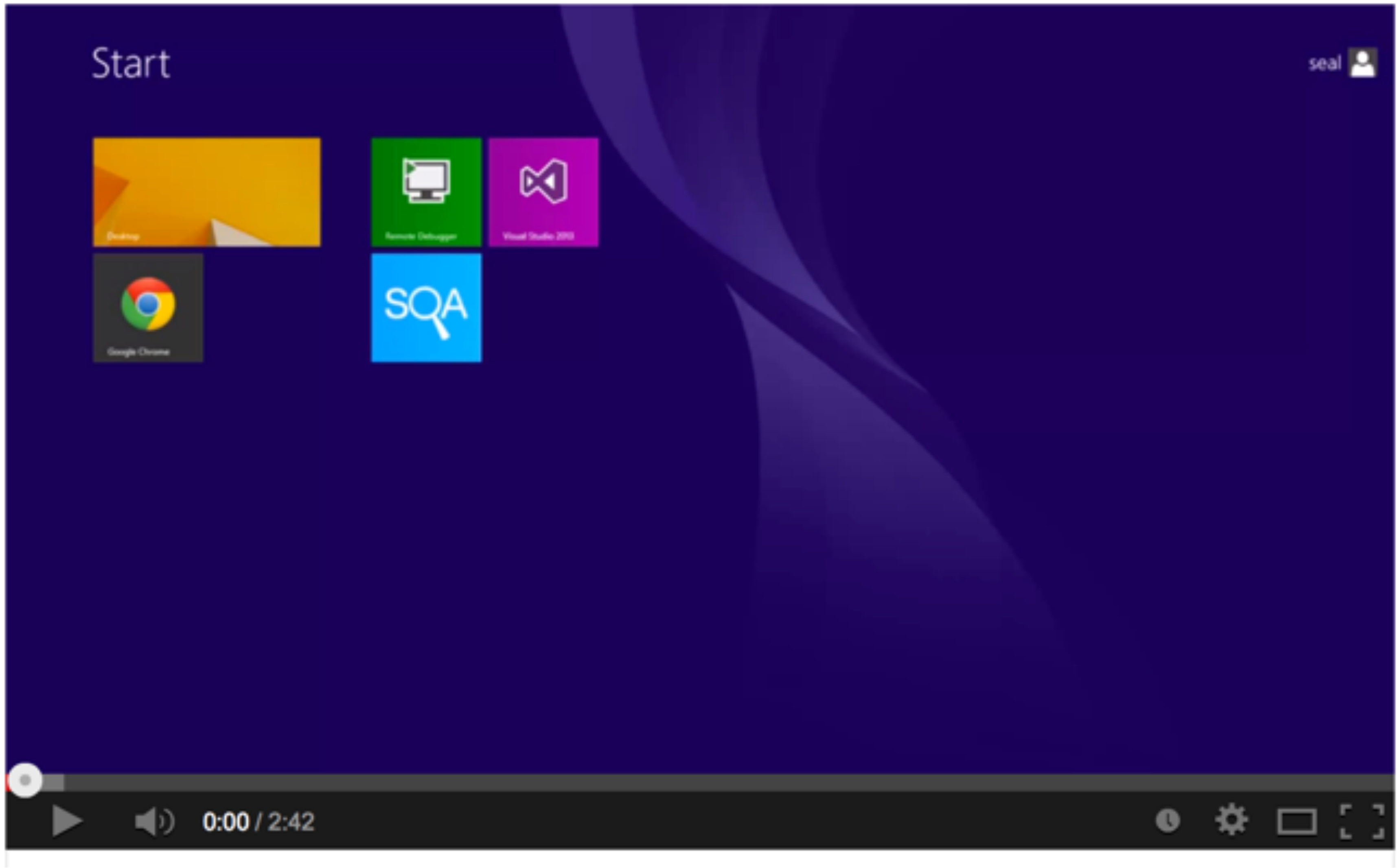
Panelists
Harald Gall (University of Zurich, Switzerland)
Mark Harman (University College London, UK)
Giancarlo Succi (Free University of Bolzano-Bozen, Italy)

Abstract
For all the books on software engineering, and the articles, and the conferences, a remarkable number of fundamental questions, so fundamental that just about software project runs into them, remain open. At best we have folksy rules, some possibly true, others doubtful, and others – such as "adding people to a software project delays it further" – wrong to the point of absurdity. Researchers in software engineering should, as their duty to the community of practicing software practitioners, try to help provide credible answers to such essential everyday questions. The purpose of this panel discussion is to assess what answers are already known through empirical software engineering, and to define what should be done to get more.

The Screening Plant of a SW Miner



SQA Mashup Teaser



Roadmap for the talk

- Challenges of Software Mining Studies
- Mining Studies: Where are we now?
- Software Analytics: Replication and Benchmarking
- An Infrastructure for Software Analytics

I. Challenges of Software Mining Studies

s.e.a.l.
software evolution & architecture lab

Which data sources?

- Evolution analysis data repositories à la **PROMISE**
  - Flossmole, Sourcerer, Ultimate Debian DB
  - Provide benchmark (raw) data
- Interactive online **web platforms** that provide various analyses
  - Boa, FOSSology, Alitheia core, Ohloh
  - Analyses offered by design
  - Data produced is best used within the system
- **Industrial** project data (not widely accessible)

What kind of studies?

- **Source code**
  - Which entities co-evolve/co-change?
  - How to identify code smells or design disharmonies?
- **Bugs and changes**
  - Who should / how long will it take to fix this bug?
  - When do changes induce fixes?
  - Predicting bugs and their components?
- **Project and process**
  - Do code and comments co-evolve?
  - Who are the experts of a piece of code?

Example: Bug Prediction

Using Code Churn vs.
Fine-Grained Changes

Predicting the
Types of Code Changes

Predicting the
Method

Using the Gini Coefficient
for Bug Prediction

Using developer networks
for Bug Prediction

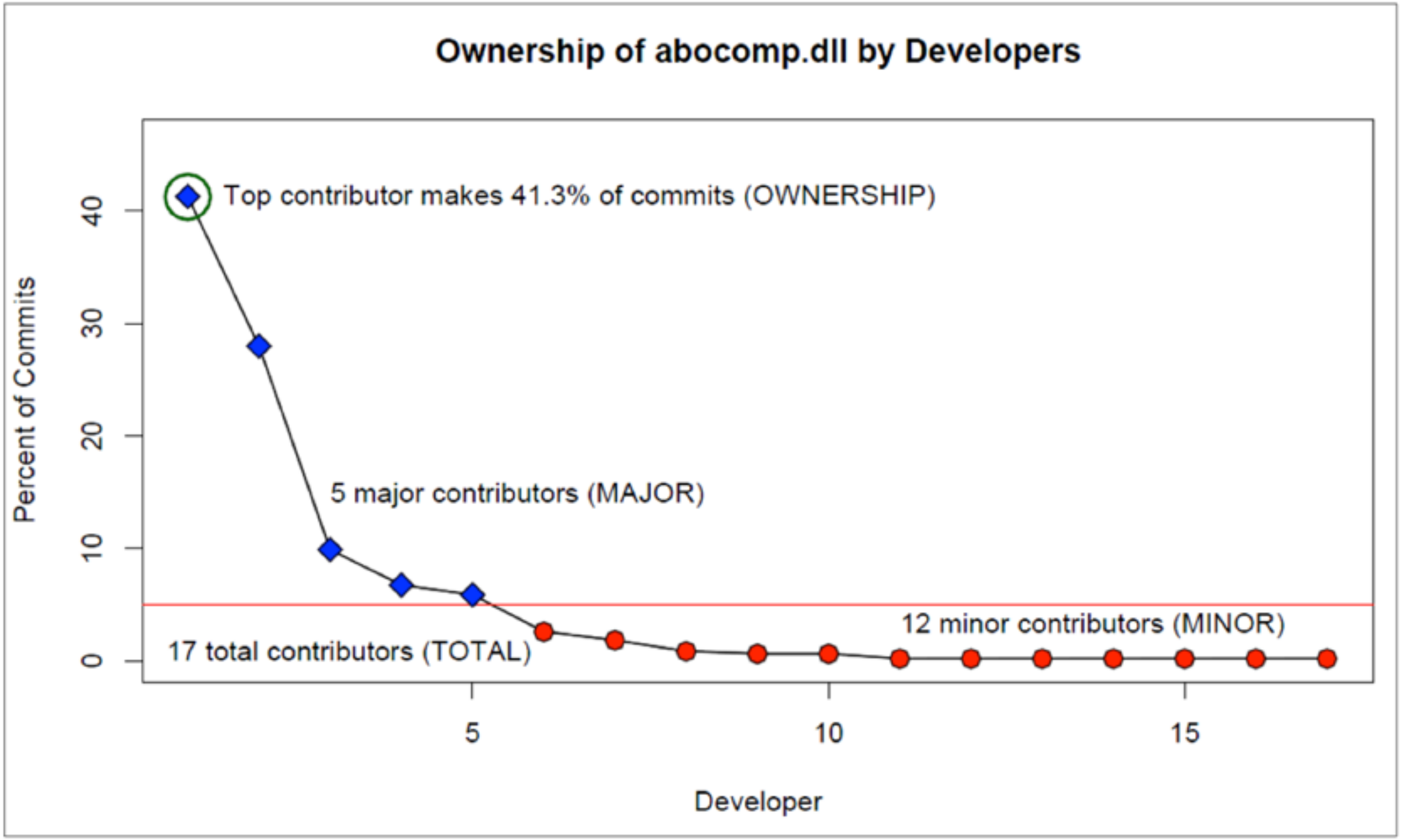
Performance of bug prediction

- Learn a prediction model from **historic data**
- Predict defects for the **same project**
- Hundreds of prediction models / learners exist
- Models work fairly well with precision and recall of up to 80%.

Predictor	Precision	Recall
Pre-Release Bugs	73.80%	62.90%
Test Coverage	83.80%	54.40%
Dependencies	74.40%	69.90%
Code Complexity	79.30%	66.00%
Code Churn	78.60%	79.90%
Org. Structure	86.20%	84.00%

From: N. Nagappan, B. Murphy, and V. Basili. The influence of organizational structure on software quality. ICSE 2008.

Example: Code Ownership



C. Bird, N. Nagappan, B. Murphy, H. Gall, P Devanbu, Don't touch my code! Examining the effects of ownership on software quality, ESEC/FSE '11

Actionable Findings

- "Changes made by minor contributors should be reviewed with more scrutiny."
- "Potential minor contributors should communicate desired changes to developers experienced with the respective binary."
- "Components with low ownership should be given priority by QA."

C. Bird, N. Nagappan, B. Murphy, H. Gall, P Devanbu, Don't touch my code! Examining the effects of ownership on software quality, ESEC/FSE '11

Studies and Issues

- Bug predictions do work, cross-project predictions do not really work
- Data sets (systems) need to be "harmonized"

- Open issues:

Replicability of studies

Benchmarks to be established

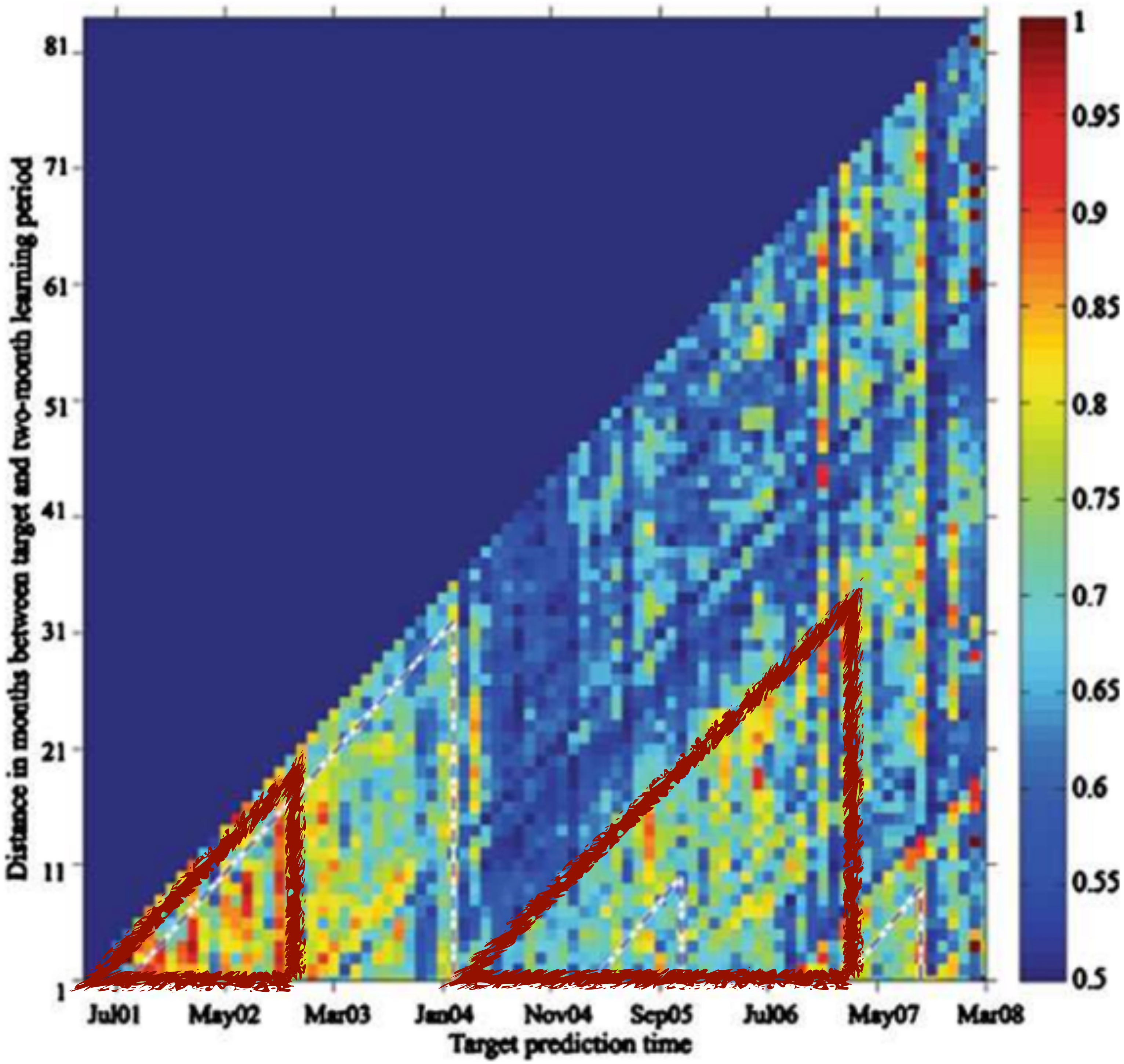
II. Software Mining Studies: Where are we now?

s.e.a.l.
software evolution & architecture lab

Nature of Studies

- Replication
  - **Less than 20% can be replicated,** from all the empirical studies published in MSR 2004-2009
    [G. Robles: Replicating MSR: A study of the potential replicability of papers published in the Mining Software Repositories proceedings. MSR 2010]
- Data availability
  - **Raw data for OSS** is easily available straight from publicly available sources
  - **(Pre)Processed data** is not yet widely available
- Data preparation / tailoring to stakeholders

Performance/Time variance



- Bug prediction performance varies over time
- OpenOffice 2001-2008
- monthly slices

- conceptual drift!
- phases of stability!

J. Ekanayake, J. Tappolet, H. Gall, A. Bernstein, Time variance and defect prediction in software projects, Empirical Software Engineering, Vol. 17 (4-5), 2012

III. Software Analytics: Where to go from here?

s.e.a.l.
software evolution & architecture lab

What is missing?

- Replication
- Large-scale comparative studies
- Preprocessing and Learners
- Calibration
- Benchmarking
- Line up of essential questions
- Adopting technologies from other fields

Replicability Evaluation

- Mining Studies of MSR 2004 - 2011
  - 84 (49%) experimental/empirical studies
  - 88 (51%) non-experimental studies (new methods, tools, case studies, visualizations, etc.)
- Studies classified into 6 categories and manually checked if they can be replicated with SOFAS:

  Version History Mining, History Mining,

  Change Analysis, Social Networks and People,

  Defect Analysis, Bug Prediction

MSR Replication with SOFAS

Research category	Number of papers	Fully replicable papers	Partially replicable papers	Non replicable papers
Version History Mining	8 (9%)	4	0	4
History Mining	17 (20%)	0	8	9
Change Analysis	13 (15%)	5	6	2
Social Networks and People	19 (22%)	6	5	8
Defect Analysis	19 (22%)	8	6	5
Bug Prediction	8 (9%)	2	2	4
	84 (100%)	25 (30%)	27 (32%)	32 (38%)

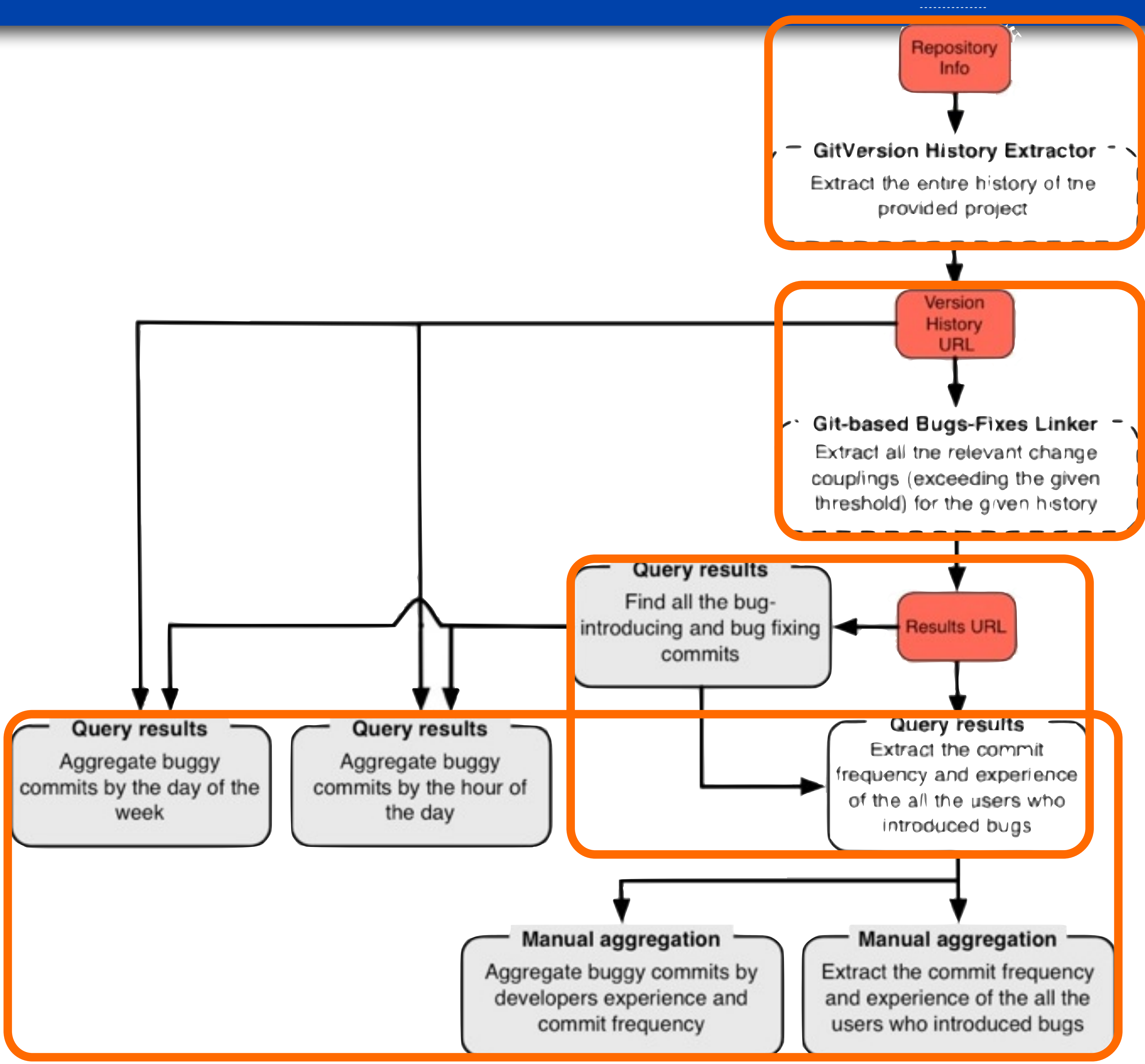
Replicability

- **Full replication:** 30% of the studies can be fully replicated out of the box
- **Partial replication:** 32% of the studies can be partially replicated
- As evaluation, we fully replicated "Do time of day and developer experience affect commit bugginess?" by J. Eyolfson, L. Tan and P. Lam, MSR 2011

The replication of the study

- We replicate the study to verify the 4 main findings
- We extend **the study** by testing the findings for **additional OSS projects**:
  - Apache HTTP, Subversion, and VLC
- We analyze the results
  - Do we achieve the same results?
  - Can the original conclusions also be drawn for the additionally investigated projects?

Analysis Workflow



Replication results /1

- **Percentage of buggy commits**
  - We confirmed the results of the original study with slight differences (different heuristic and date of analysis)
  - The additional projects exhibit similar values (22-28%)

		# commits	# bug-introducing commits	# bug-fixing commits
Original Study	**Linux**	268820	68010 (25%)	68450
	PostgreSQL	38978	9354 (24%)	8410
Extended Study	**Apache Http Server**	30701	8596 (28%)	7802
	Subversion	47724	12408 (26%)	10605
	VLC	47355	10418 (22%)	10608

Replication results /2

- **Influence of time of the day**
  - We confirmed the results of the original study
  - The amount of buggy commits are particularly high between midnight and 4 AM and tends to then drop below average (morning and/or early afternoon)

  - Windows of low bugginess greatly vary between projects
  - Commit bugginess follows very different patterns

Replication results /3

- **Influence of developer**
  - We confirmed the results of the original study
  - A drop in commit bugginess is evident with the increasing amount of time a developer has spent on a project
- **Influence of day of the week**
  - We confirmed the results of the original study
  - Different weekly patterns in the additional projects

Interpretation of results

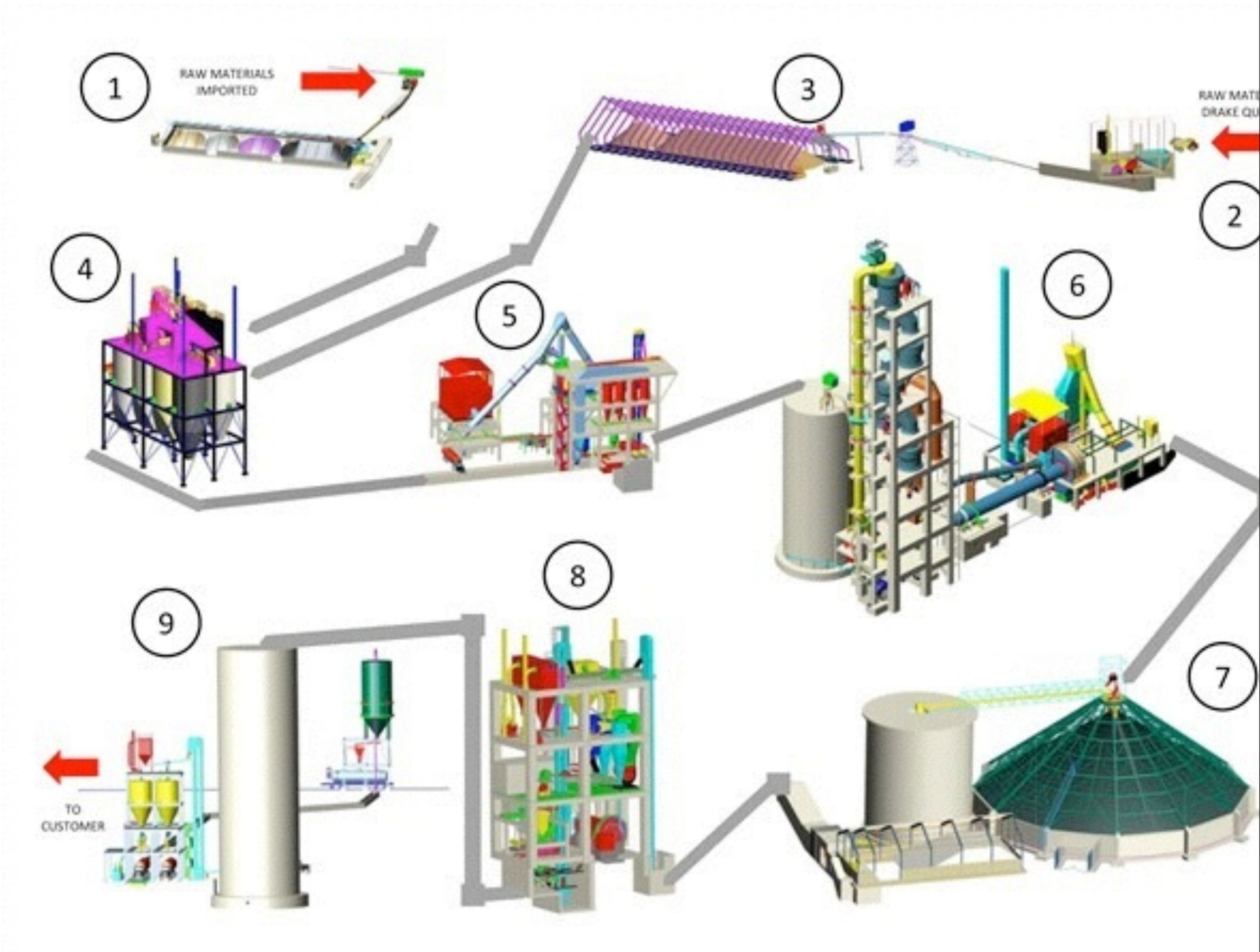
- **Feasibility**
  - We can **replicate 30%** of the analyzed studies and **compute the ground data** needed for another 32%
  - The studies we can replicate all use historical data extracted from different repositories
- **Scalability**
  - The approach can scale up to very many of projects
  - Once the **analysis workflow** is defined, it can be automatically run with different project repositories
  - Still, limitation is total execution time (Apache HTTP ~ 8 hrs)

Interpretation of results

- **Extensibility**
  - We only focused on the replication of existing studies
  - The results and ground data produced by SOFAS analyses can be fed to other services, used by third-party analyses and tools or combined with data from other sources.
  - Do time of day, developer experience **and file ownership** affect commit bugginess?
    - e.g. taking into account code ownership measured using the Gini coefficient [Giger, 2011]

To get to the next level ...

- Support for **replicability** & systematic analysis workflows
- **Calibration** of data preprocessing
- **Performance** measures & performance criteria for studies
- **Conclusion stability** of studies

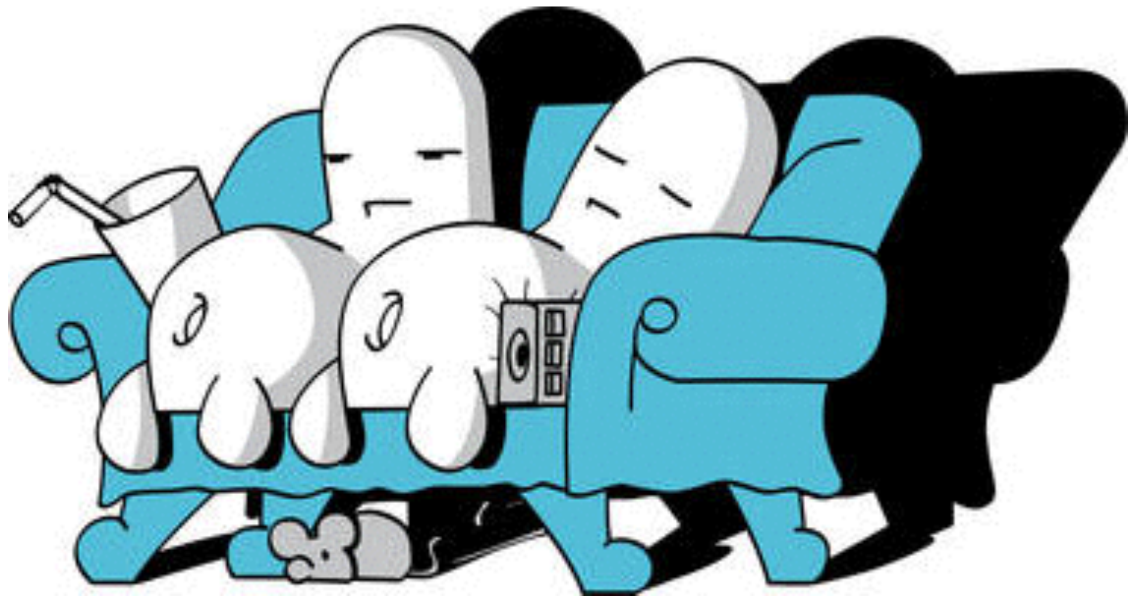


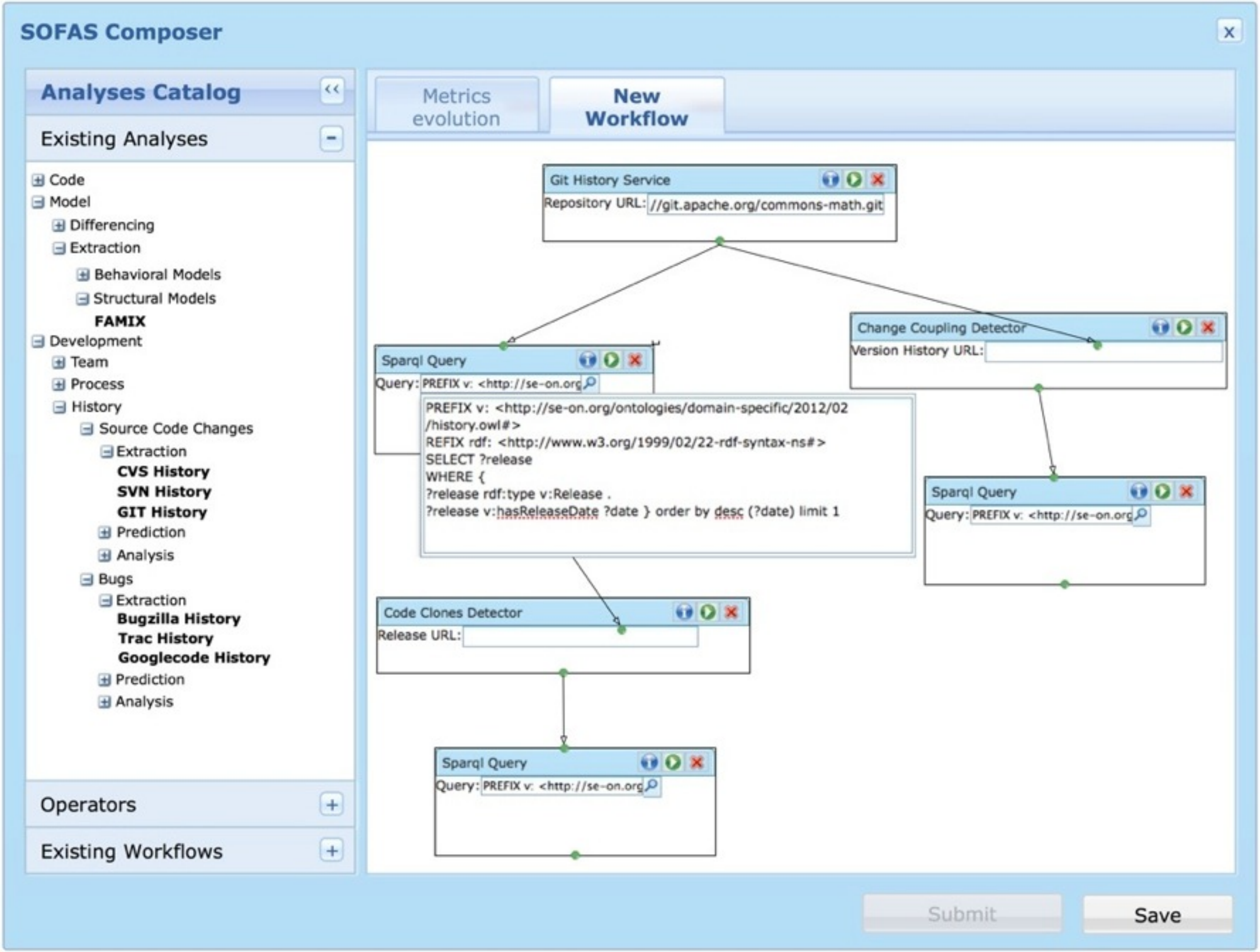
IV. Replicating Software Mining Studies with SOFAS

s.e.a.l. software evolution & architecture lab

SOFtware Analysis Services

- SOFAS = RESTful service platform by G. Ghezzi
- using software evolution ontologies
- enabling the composition of analysis workflows
- http://www.ifi.uzh.ch/seal/research/tools/sofas.html

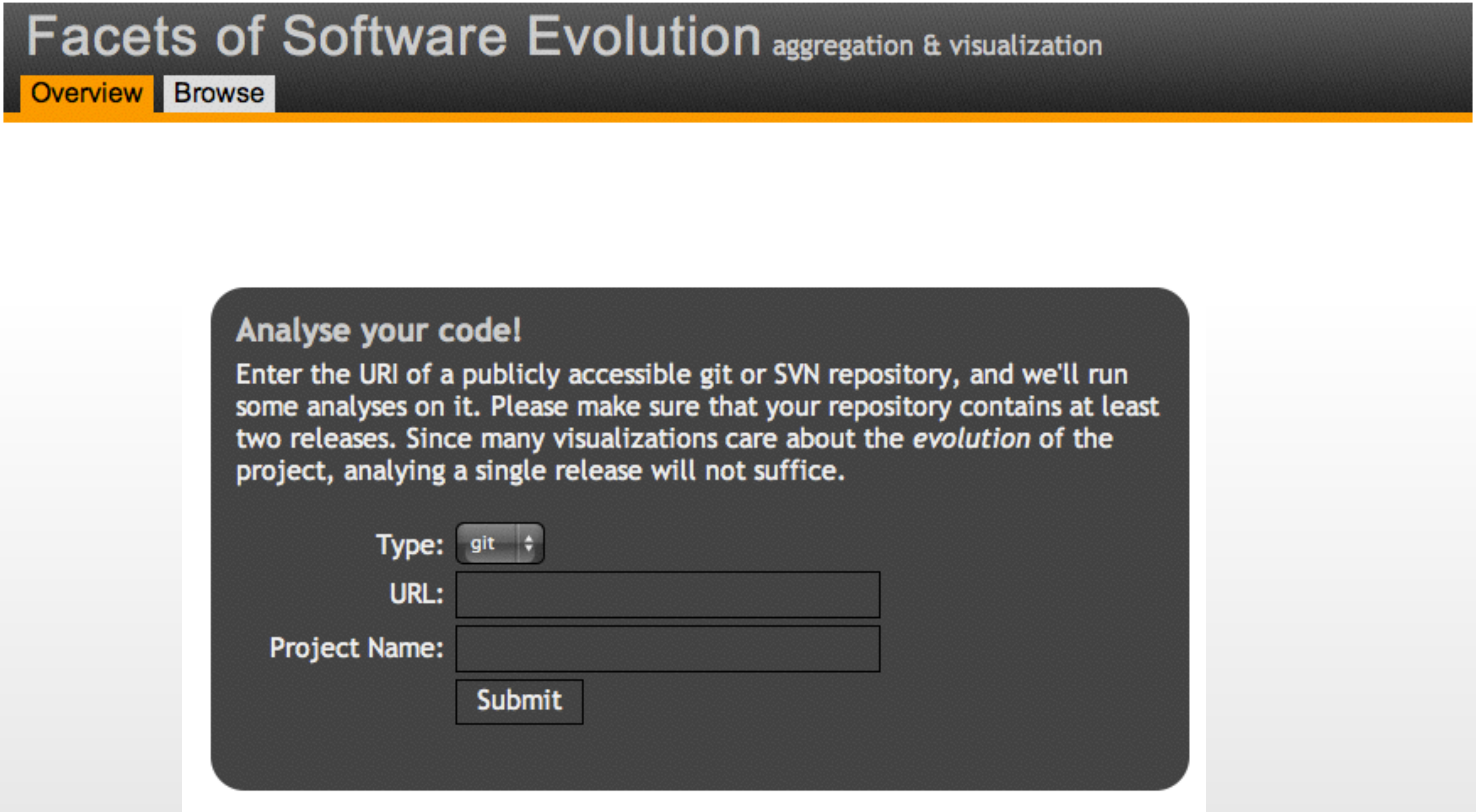




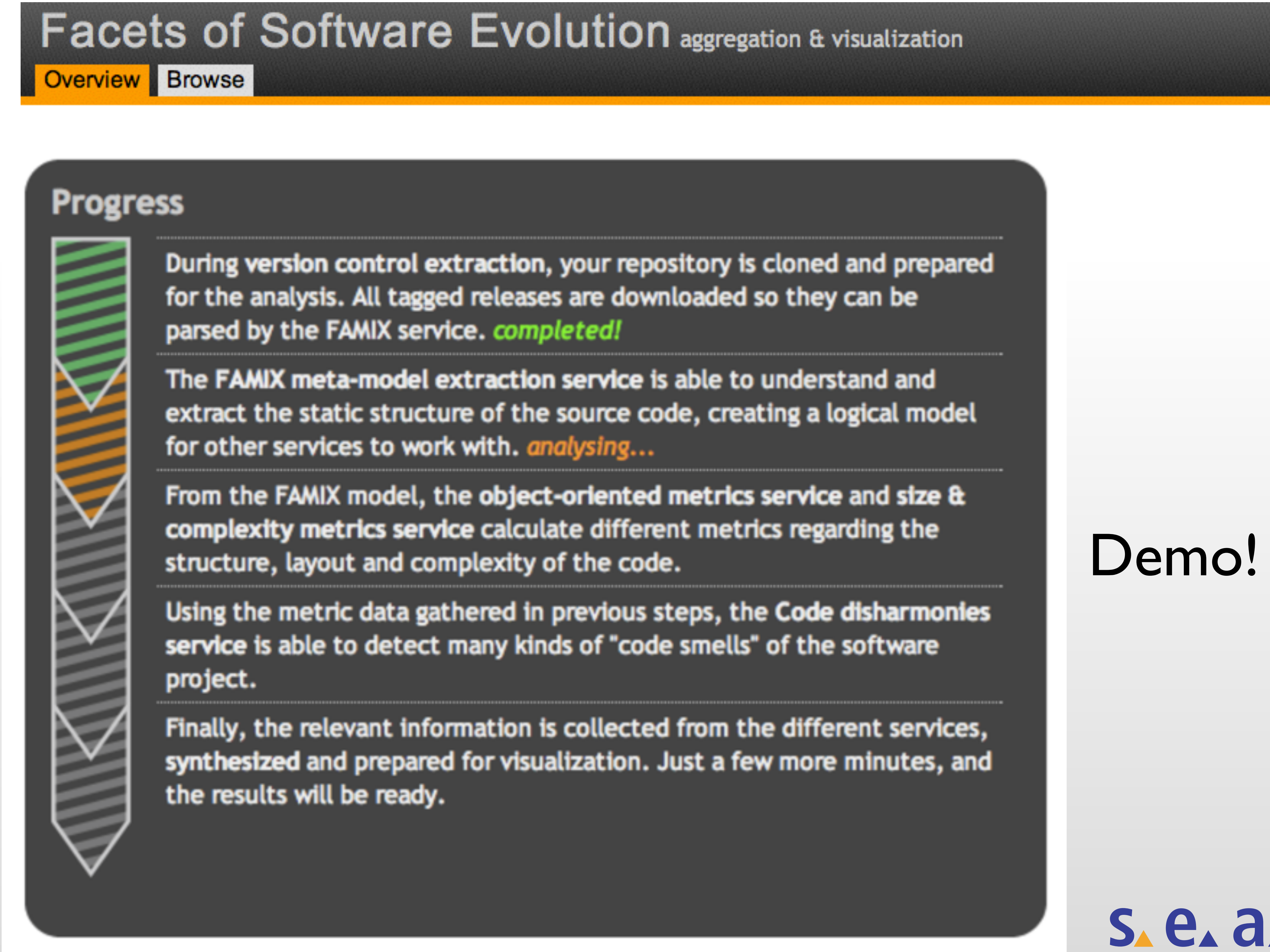
software evolution & architecture lab

Current SOFAS services

- **Data Gatherers**
  - Version history extractor for CVS, SVN, GIT, and Mercurial
  - Issue tracking history for Bugzilla, Trac, SourceForge, Jira
- **Basic Services**
  - Meta-model extractors for Java and C# (FAMIX)
  - Change coupling, change type analyses
  - Issue-revision linker
  - Metrics service
- **Composite services**
  - Evolutionary hot-spots
  - Highly changing Code Clones
  - and many more ...



s.e.a.l. software evolution & architecture lab


Facets of Software Evolution aggregation & visualization
Overview
Browse
Progress
During version control extraction, your repository is cloned and prepared for the analysis. All tagged releases are downloaded so they can be parsed by the FAMIX service. completed!
The FAMIX meta-model extraction service is able to understand and extract the static structure of the source code, creating a logical model for other services to work with. analysing...
From the FAMIX model, the object-oriented metrics service and size & complexity metrics service calculate different metrics regarding the structure, layout and complexity of the code.
Using the metric data gathered in previous steps, the Code disharmonies service is able to detect many kinds of "code smells" of the software project.
Finally, the relevant information is collected from the different services, synthesized and prepared for visualization. Just a few more minutes, and the results will be ready.

Demo!

V. Mashing Up Software Analytics Data for Stakeholders

s.e.a.l.
software evolution & architecture lab

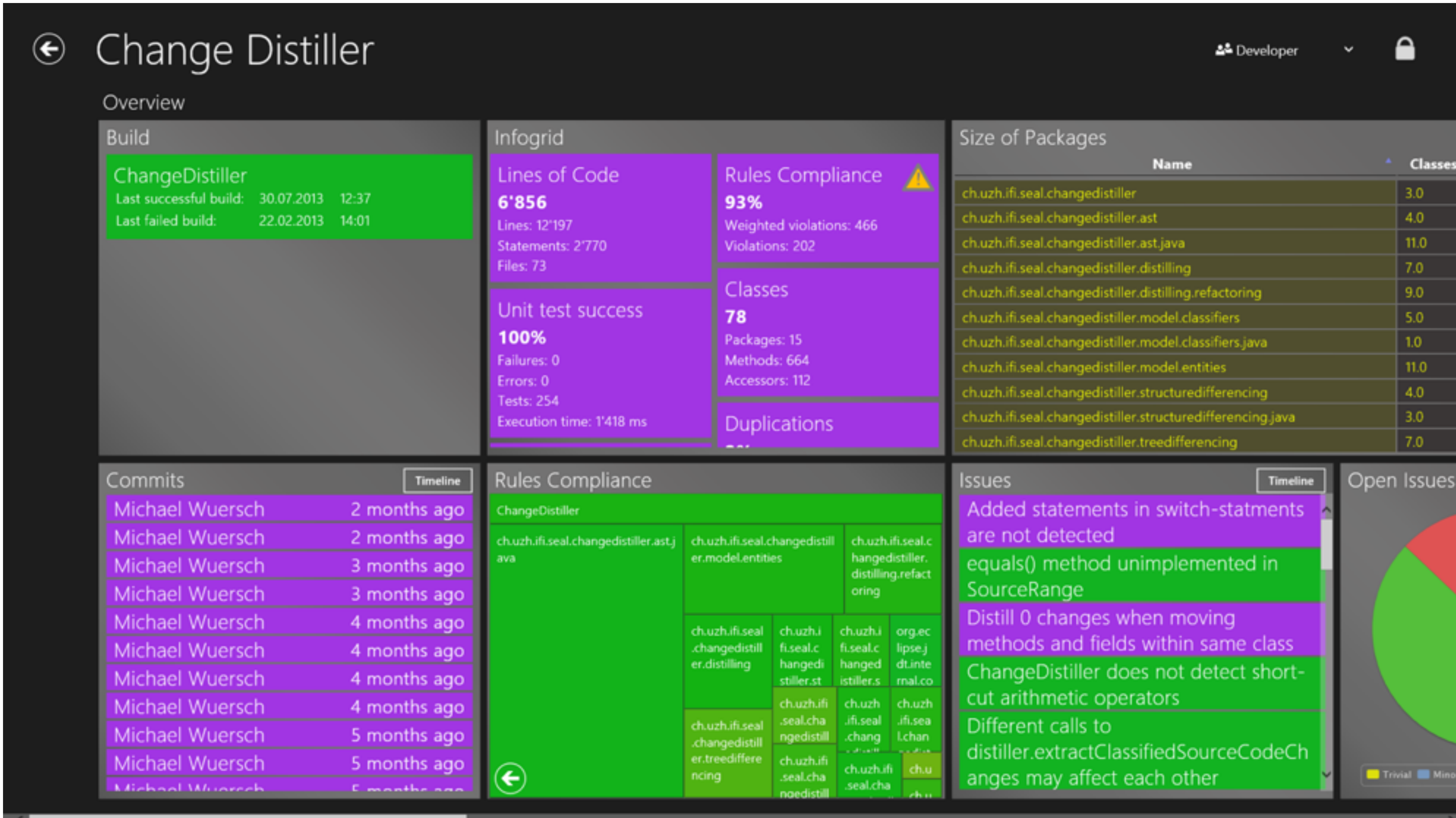
Multiple Stakeholder Mining

- We need to **tailor information** to the **information needs of stakeholders**, such as developers, testers, project managers, or quality analysts
  - study their needs beyond typical developer needs 'questions developers ask' by Sillito et al.)
  - devise prototypes to elicit that information needs, for example, SQA-Mashup for Integrating Quality Data

SQA-Mashup

- A Mashup of Software Project data
  - commit & issue & build & test data
  - all in mashups, integrated, easy to access
  - however, filtered to the information needs of stakeholders
- Most recent paper
  - **Martin Brandtner**, Emanuel Giger, Harald Gall, Supporting Continuous Integration by Mashing-Up Software Quality Information, CSMR-WCRE 2014, Antwerp, Belgium
- Available in Win 8 App Store
  - http://goo.gl/ZUWrvm

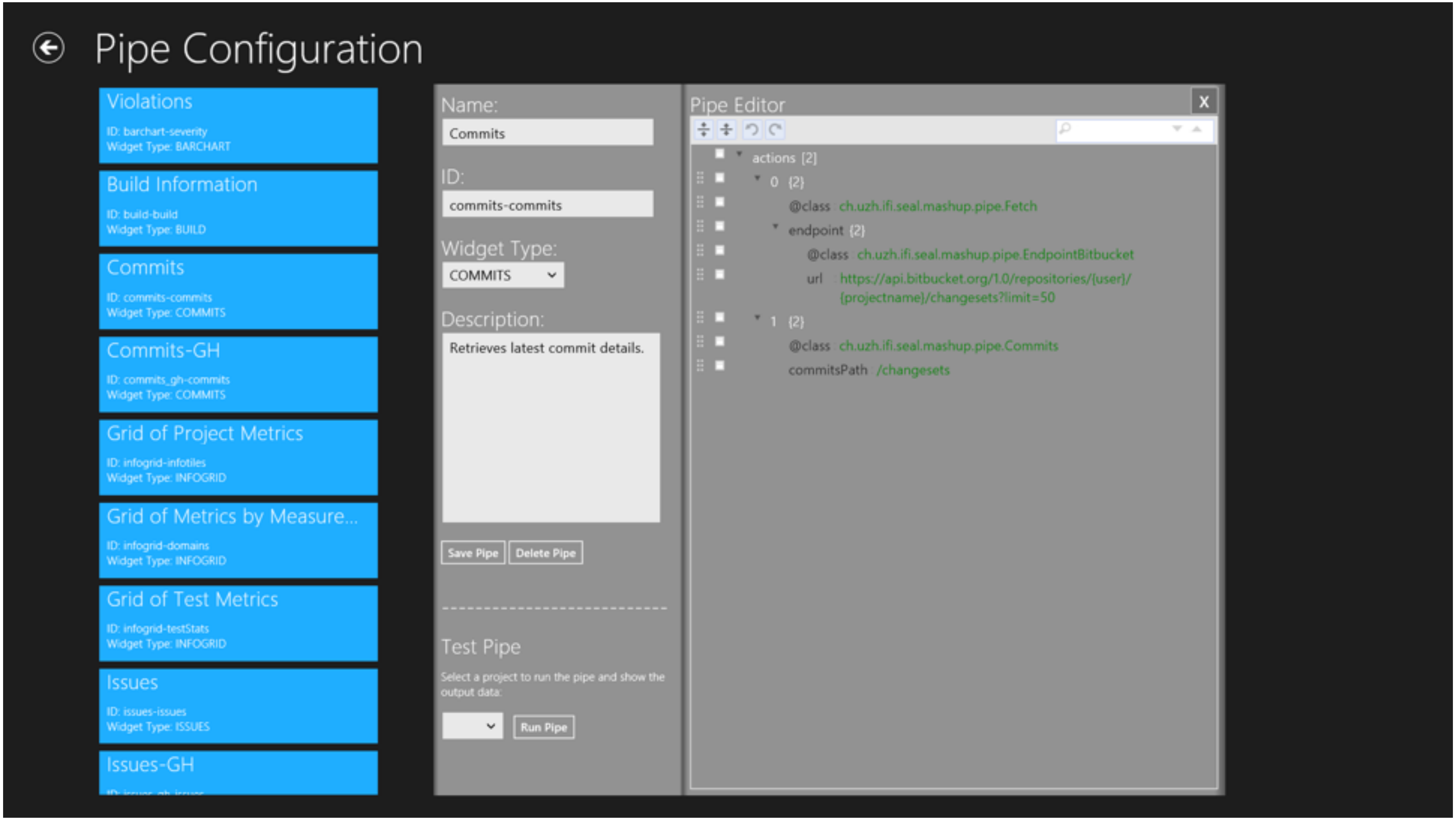
A Developer's view



A Tester's view

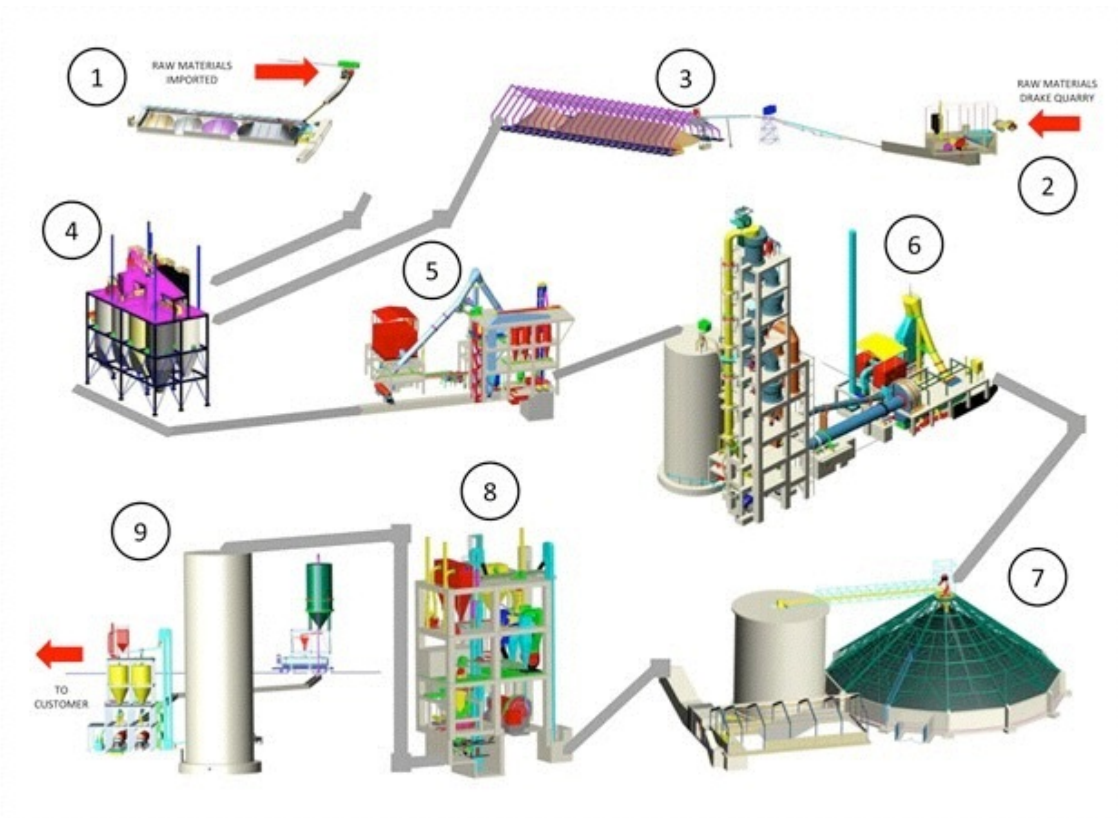
A project timeline

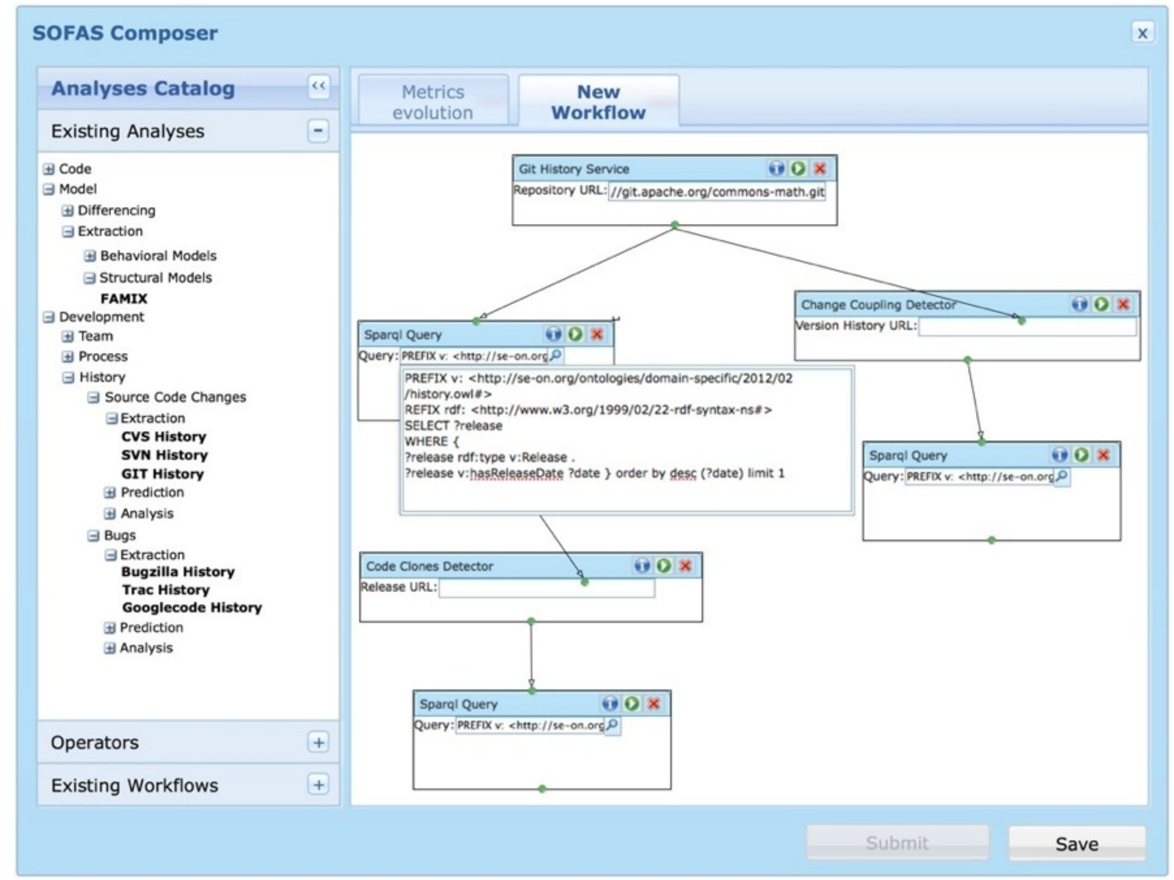
Mashup pipe configuration

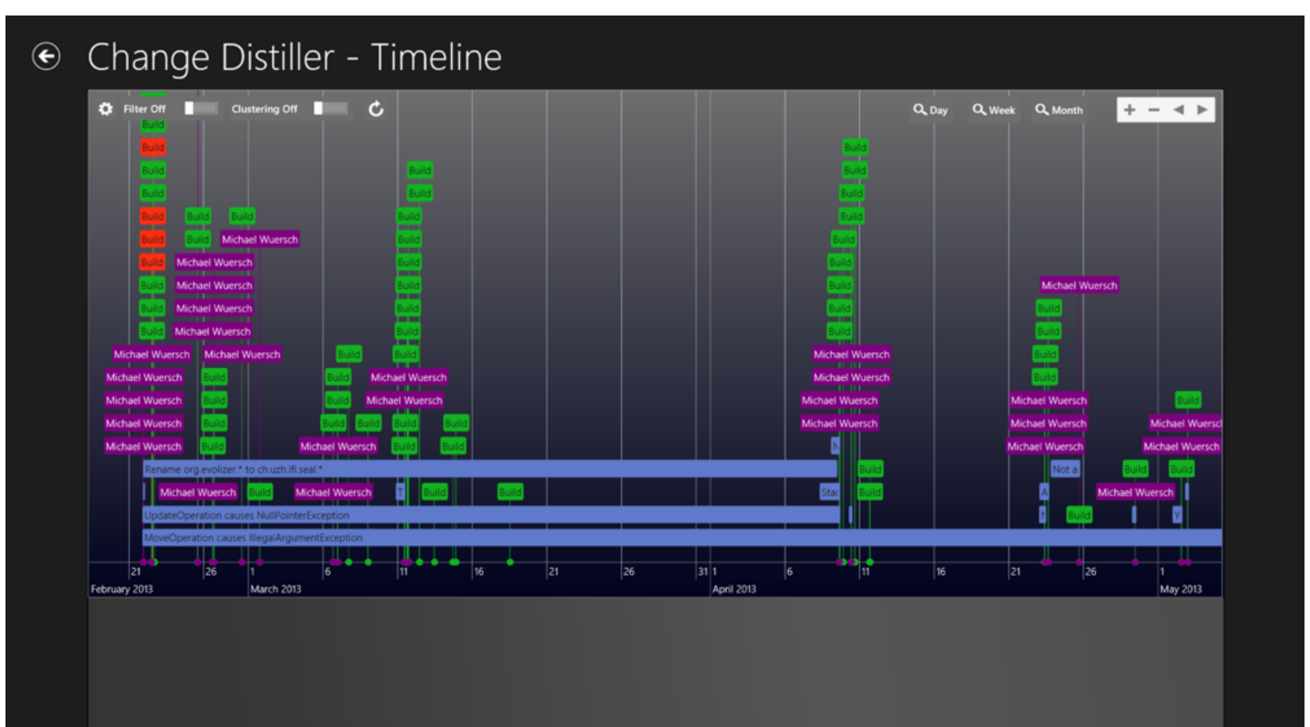


VI. Conclusions

Workflows & Mashups







Merry Christmas!