SYSTEMS ENGINEERING RESEARCH CENTER

USC CSSE

Towards Better Understanding of Software Quality Evolution Through Commit-Impact Analysis

Sponsor: DASD(SE)

By
Mr. Pooyan Behnamghader

5th Annual SERC Doctoral Students Forum
November 7, 2017
FHI 360 CONFERENCE CENTER
1825 Connecticut Avenue NW
8th Floor
Washington, DC 20009

www.sercuarc.org

Outline

- Motivation
- Foundation
  - How to identify changes?
  - How to evaluate change in software quality?
  - How to scale and replicate?
  - How to explore the data?
- Empirical Study
  - Research questions
  - Data collection
  - Results
- Translating Research Into Practice

Motivation

- Better understanding of
  - —Software quality evolution.
  - —Conflicts and synergies among software quality attributes.
- To help
  - —Organizations determine which divisions and project types have better or worse quality; which quality attributes are being achieved poorly or well; and how do these correlate with customer satisfaction and total cost of ownership.
  - —Managers better understand which types of projects or personnel contribute most to quality problems or excellence, and which types of project events correlate with which types of quality increase or decrease.
  - —Developers continuously monitor software quality and improve software maintainability.

Motivation (cont.)

- The challenges involved in studying evolution
  - —Long development history
    - Multiple developers and many changes
  - —Heterogeneous evolution
    - Change in technology and structure
- Analyzing change among official releases!
  - — The important details between releases?
    - Impact of a single change
    - Impact of a developer over a period of time.
    - Impact of a series of tasks
  - —The data points where problems are introduced/solved?
    - Uncompilable revisions
    - Failed tests/validations
    - Security problems are introductions

Motivation (cont.)

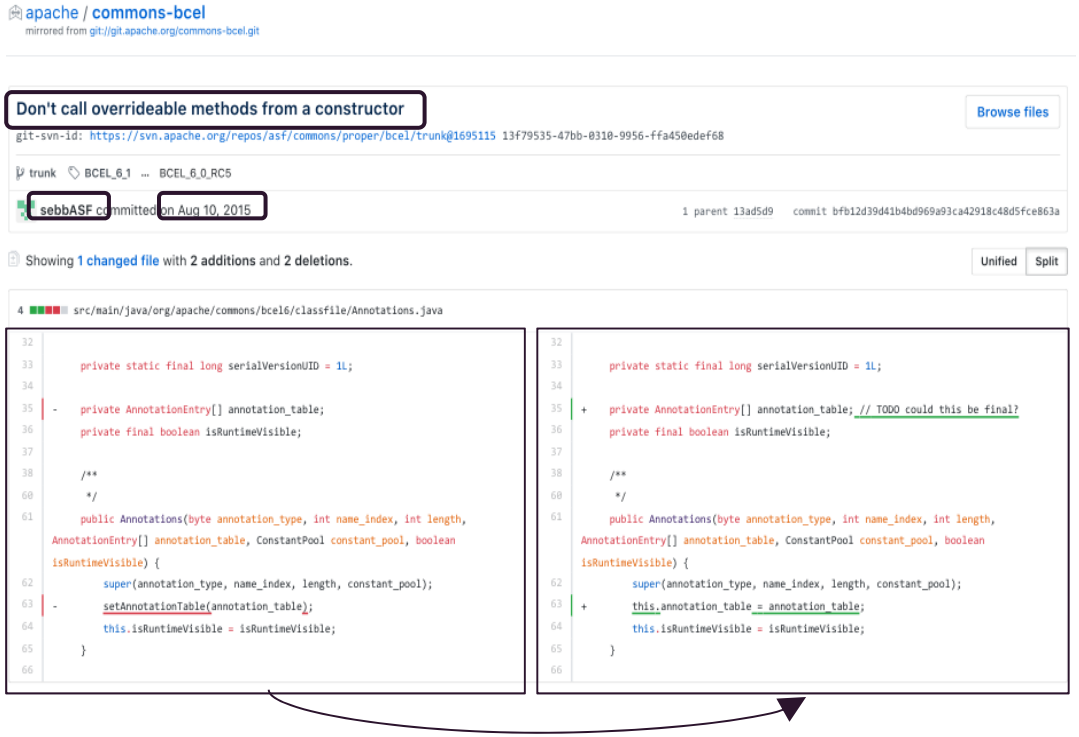
The Bottom Line:
Analyzing Software Quality Before and After Each Change!

Foundation

- How to identify changes?
  - —Version control system
  - —Commit-impact analysis

Version Control System

- Tracking change.
- Git
  - —Repository
    - o Full development history.
  - —Commit
    - o Details of every stage.
      - — Why.
      - — When.
      - — Who.
      - — What.

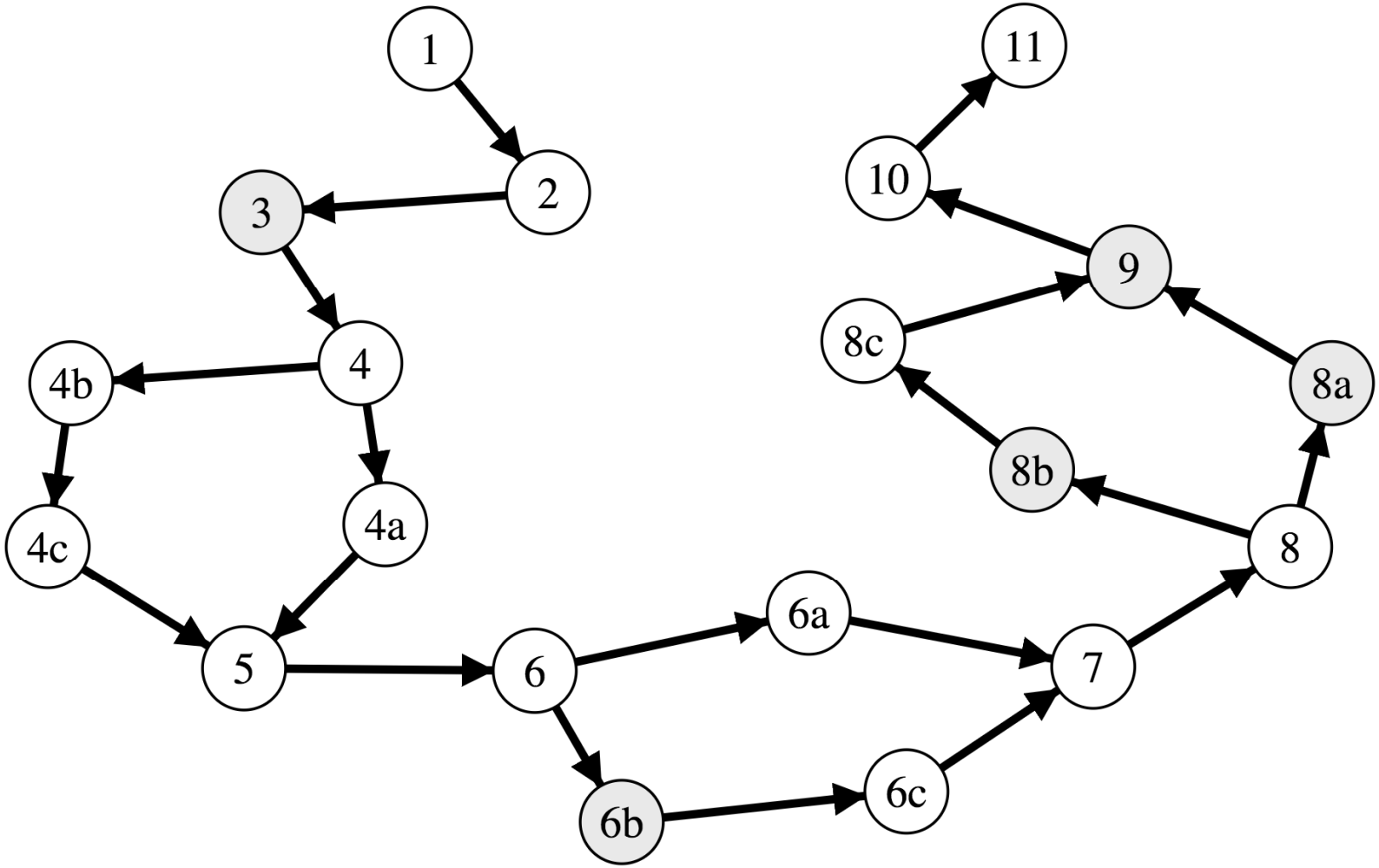


The details of a commit in Apache Commons Bcel (https://github.com/apache/commons-bcel/commit/25d3c6f1c061bb81bc384ac8cc05e72b57849cf4)

Commit-Impact Analysis

- Analyzing software quality before and after each change.
- Two key elements
  - Main module
    - The module containing most of the code.
    - Why limiting the scope to only one module?
      - Not all changes impact quality!
  - Impactful commit
    - Creates a new revision of the main module.
      - Must change at least one source file.
      - Can change other files and other modules.

Commit-Impact Analysis (cont.)



A Software System's Commit History.

- Impactful commits are denoted in gray.
- A higher number/letter means later in time.

Foundation (cont.)

- How to identify meaningful changes?
  - —Version control system
  - —Commit-impact analysis
- How to evaluate change in software quality?
  - —Programming analysis techniques
  - —Distance metrics

Programming Analysis Techniques

- Extracting quality attributes.
- Static
  - PMD, FindBugs, SonarQube, CheckStyle, UCC
  - Architecture recovery techniques (ARC, ACDC, PKG)
- Dynamic
  - Rendering HTML files.
  - Executing tests.

Distance Metrics

- Quantifying change.
- Different types of artifacts
  - —XML/Json/Excel Reports→ Model Parsers
    - Parsing the reports and storing the values in relational schemas
    - Subtracting two numbers/vectors (SQL queries)
  - —Graphs and Sets → Model Comparators
    - More advanced mathematical metrics
      - — E.g., Architecture-to-architecture comparison metric

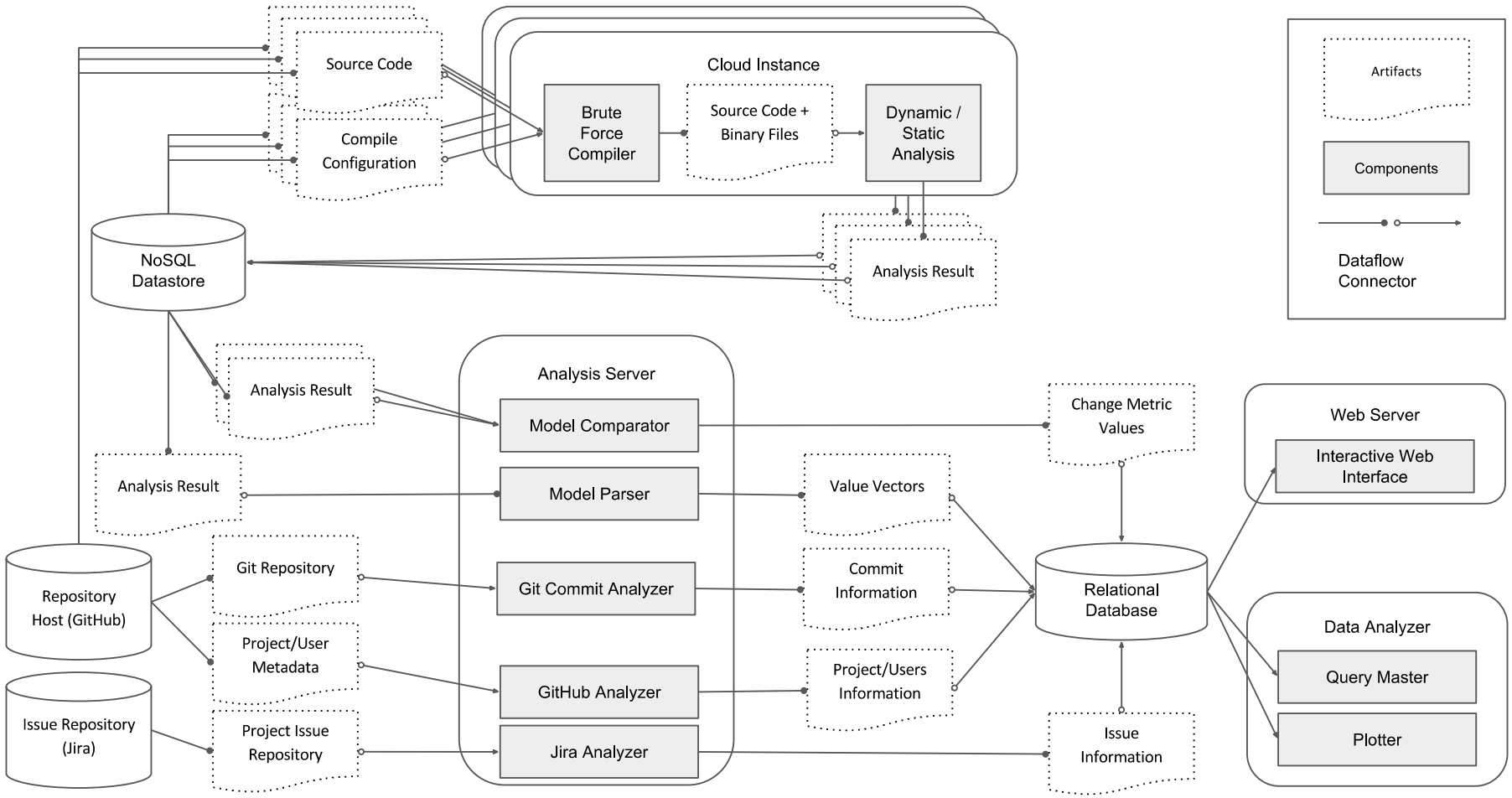
Foundation (cont.)

- How to identify meaningful changes?
  - —Version control system
  - —Commit-impact analysis
- How to evaluate change in software quality?
  - —Programming analysis techniques
  - —Distance metrics
- How to scale and replicate?
  - —Automated cloud-based infrastructure

Automated Cloud-Based Infrastructure

1. Retrieve a subject system's meta-data (e.g., number of contributors) as well as its commit history from GitHub.
2. Distribute hundreds of revisions (i.e. official releases and/or revisions created by commits) on multiple cloud instances.
3. Compile each revision and run static/dynamic programming analysis techniques on it.
4. Collect and parse the artifacts generated by programming analysis techniques to extract quality attributes.
5. Run various statistical analysis on software quality evolution.

Architecture



Foundation (cont.)

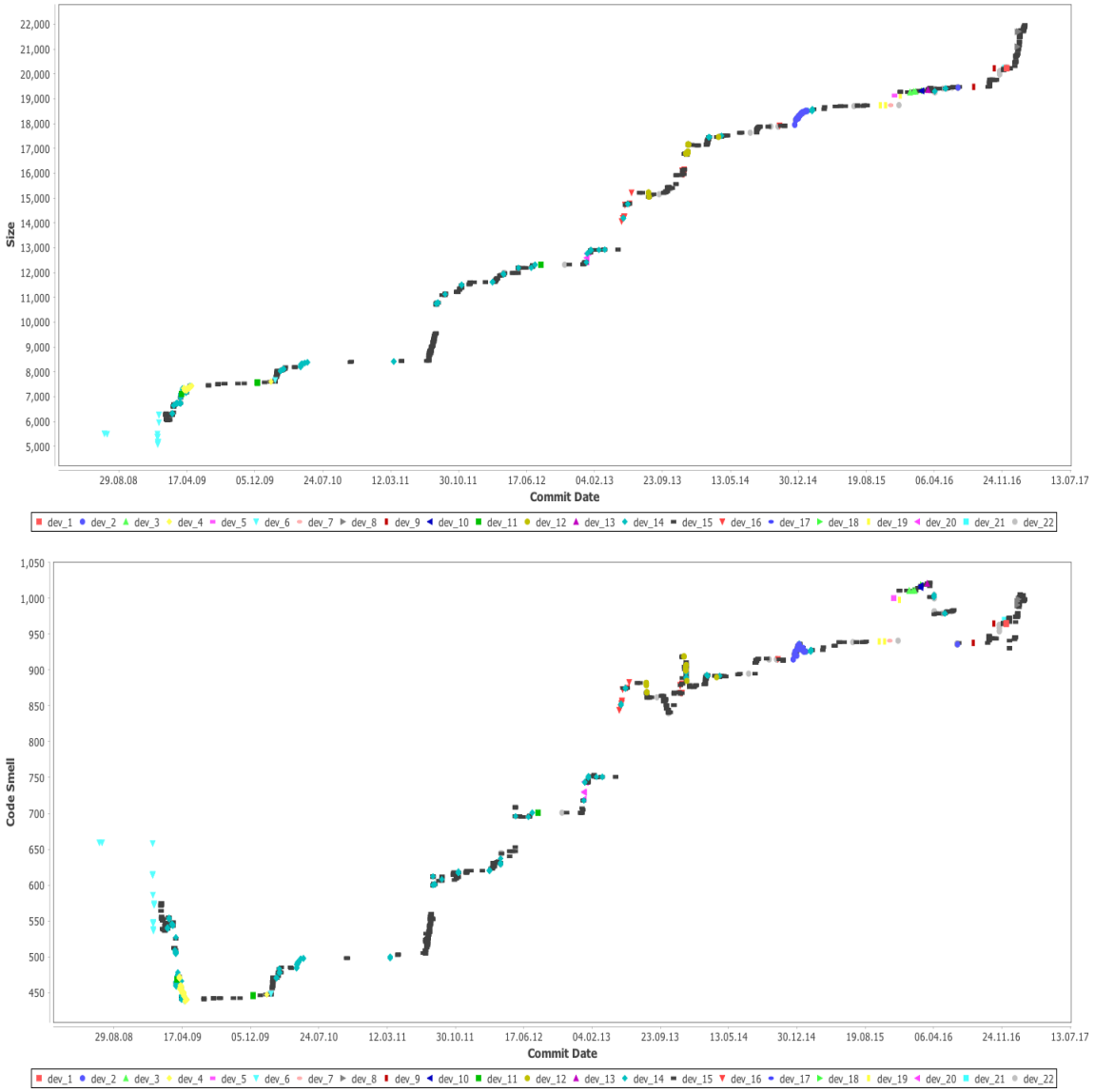
- How to identify meaningful changes?
  - —Version control system
  - —Commit-impact analysis
- How to evaluate change in software quality?
  - —Programming analysis techniques
  - —Distance metrics
- How to scale?
  - —Automated cloud-based infrastructure
- How to explore the data?
  - —Interactive desktop and web interface

Interactive Desktop and Web Interfaces

- Evolution trend of a metric.
- Impact of each developer.
- Coevolution of multiple metrics.
- Evolution graph of a metric around a data point.

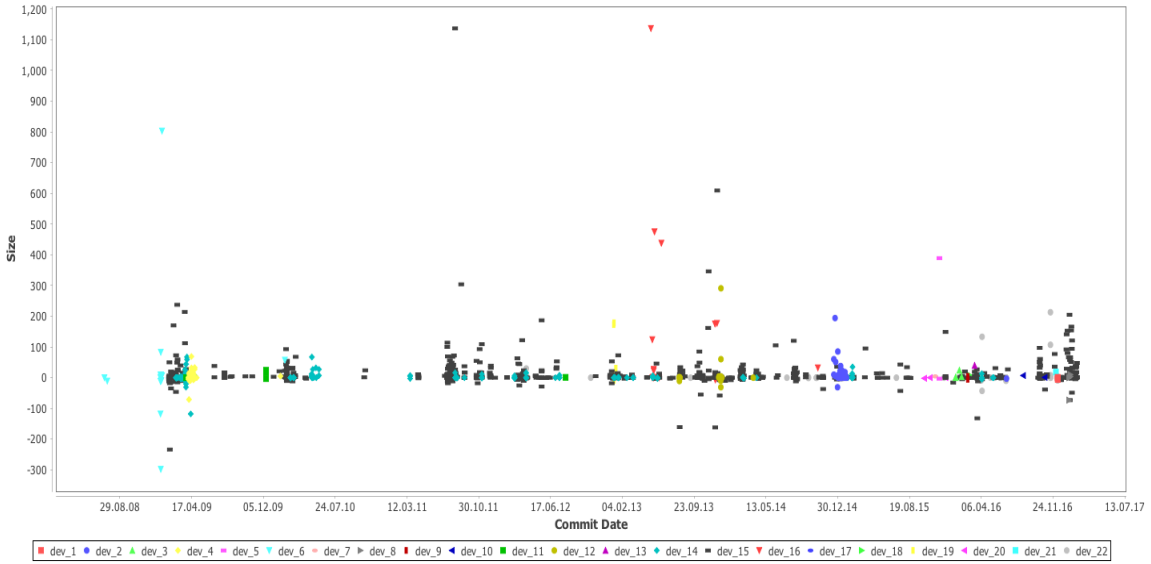
Evolution Trend of A Metric

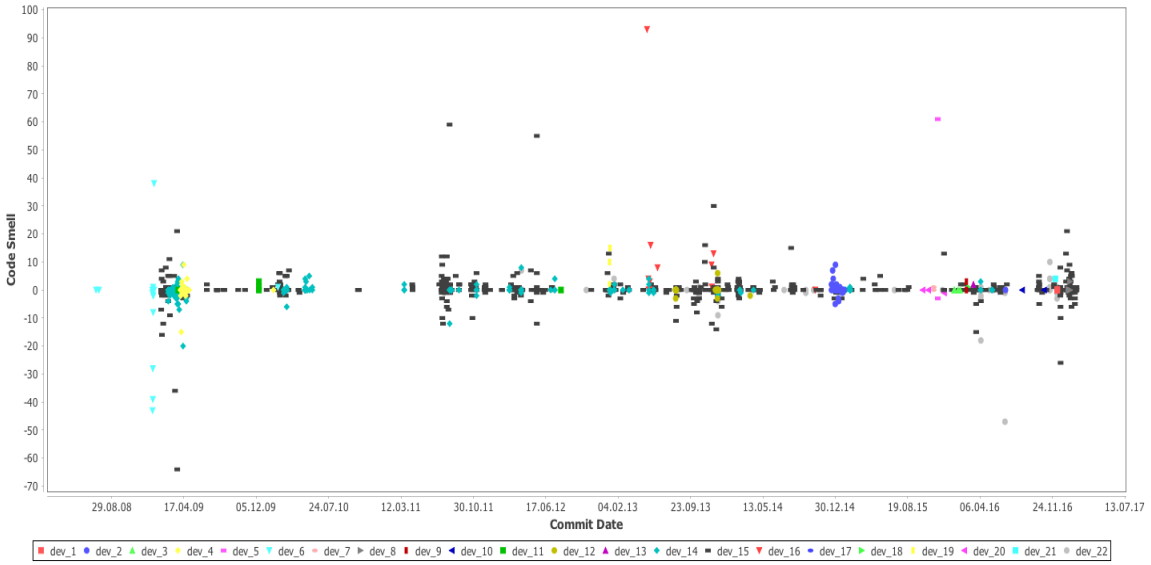
- How a single quality attribute evolves.
- Example:
  - —Two metric
    - o Size (top)
    - o Code Smells (bottom)
  - —One project
  - —A period of 9 years.
- Absolute value at any time.



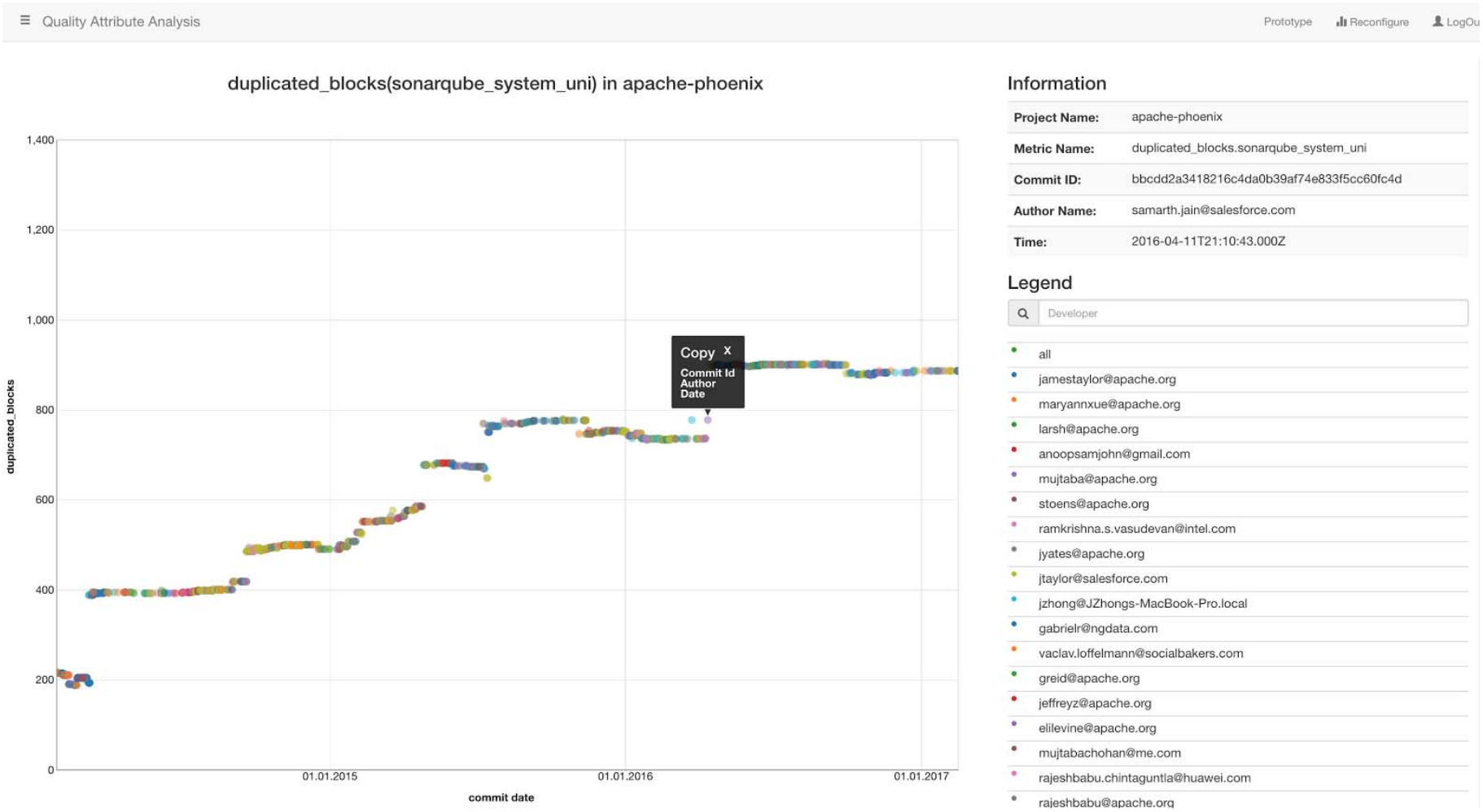
Impact of Each Developer

- How developers impact software quality.
- Example:
  - —Two metrics
    - o Size (top)
    - o Code Smells (bottom)
  - —One project
  - —A period of 9 years
- Impact of a commit.



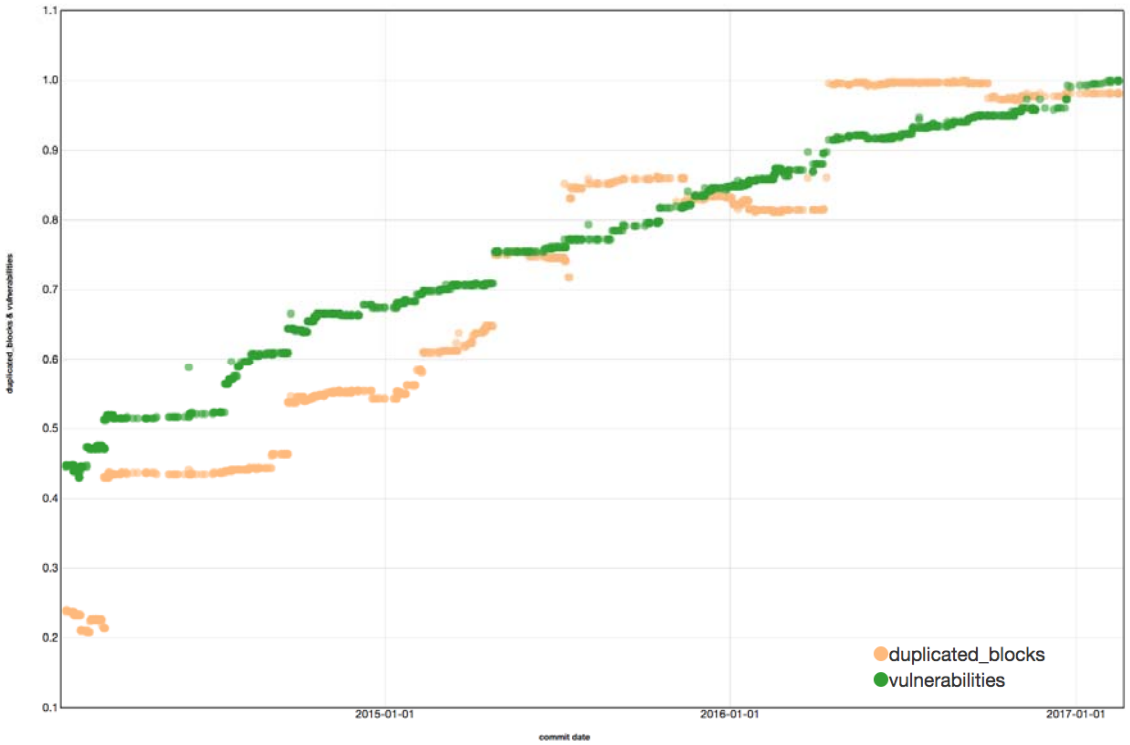


Interactive Web Interface (Evolution Trend)



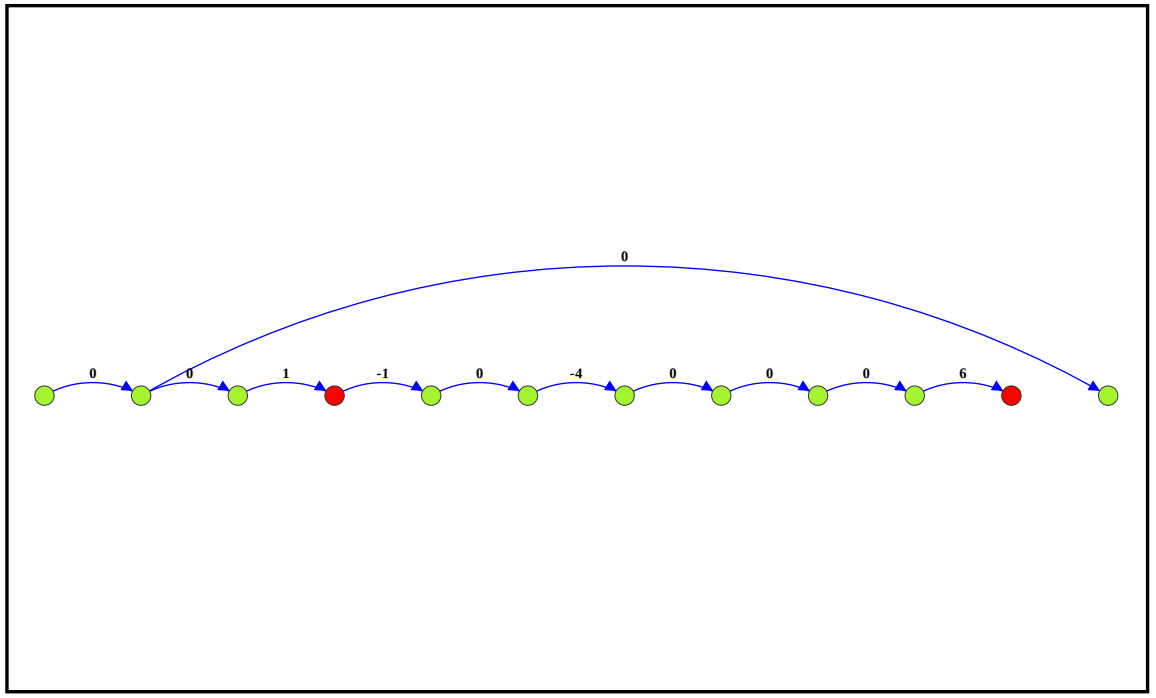
Coevolution of Multiple Metrics

- How multiple quality attributes coevolve.
- Example:
  - —Two metrics
    - o Duplicate Code Blocks
    - o Security Vulnerabilities
  - —Normalized values
  - —One project
  - —A period of 3 years



Evolution Graph of a Metric

- Evolution graph of a metric around a data point.
- Example:
  - —One metric
    - o Security Vulnerabilities
  - —One project
  - —A period of 1 week



- Legend
  - — Node represents commits
  - — Edges represent parent-child relationships between commits
  - — Values on Edges represent the impact of each commit
  - — Red commits are the ones that introduce new issues.

Empirical Study

- RQ1: To what extent do developers commit impactful changes?
- RQ2: To what extent and how do impactful commits break the compilability of the project?
- RQ3: To what extent do impactful commits affect software quality attributes?
- RQ4: Should developers rely on a single software quality metric as a change indicator?

Data Collection

- Collected the metadata of all Apache projects via GitHub API.
  - Name, # of commits, programming language, and last update date
- Selection criteria
  - Java
  - At least one update in 2017.
  - The main module exists.
  - No nontrivial prerequisite for compilation.
  - At least 100 compilable different revisions.
  - Less than 3K commits.

Data Collection (cont.)

- Process
  - —Impactful commits
    - o Identified.
  - —Revisions
    - o Downloaded, compiled, and analyzed.
  - —Models
    - o parsed and metrics extracted.
- Scale
  - —38 systems.
  - —19580 impactful commits and revisions.
  - —643 impactful developers.
  - —586 MSLOC.
  - —15 years timespan.

Empirical Study Setup

System	Domain	Rev.	Time span	MSLOC
Avro	Data Serialization	188	07/11-01/17	2.43
Calcite	Data Management	650	07/14-02/17	75.46
C-Bcel	Bytecode Eng.	576	06/06-12/16	16.5
C-Beanutils	Reflection Wrapper	139	07/07-11/16	1.71
C-Codec	Encoder/Decoder	368	09/11-09/16	2.32
C-Collections	Collections Ext.	565	03/12-10/16	13.87
C-Compress	Compress Lib.	1229	07/08-02/17	16.84
C-Configuration	Conf. Interface	333	04/14-01/17	8.83
C-CSV	CSV Library	598	11/11-02/17	0.68
C-Dbcp	DB Conn. Pooling	188	12/13-11/16	2.06
C-IO	IO Functionality	914	01/02-12/16	5.99
C-JCS	Java Caching	136	04/14-02/17	3.66
C-Jexl	Expression Lang.	295	08/09-10/16	2.26
C-Net	Clientside Protocol	877	08/06-02/17	14.96
C-Pool	Object Pooling	278	04/12-02/17	1.27
C-SCXML	State Chart XML	335	03/06-08/16	2.53
C-Validator	Data Verification	279	07/07-02/17	1.63
C-Vfs	Virtual File System	611	11/06-01/17	13.32
CXF-Fediz	Web Security	171	04/12-03/17	0.89
Drill	SQL Query Engine	616	04/15-02/17	58
Flume	Data Collection	342	08/11-11/16	2.14
Giraph	Graph Processing	387	11/12-01/17	15.12
Hama	BSP Computing	717	06/08-04/16	7.77
Helix	Cluster MNGMT	762	01/12-06/16	27.36
HTTPClient	Client-side HTTP	925	03/09-01/17	16.54
HTTPCore	HTTP Transport	565	03/09-02/17	7.38
Mina	Socket Library	263	11/09-04/15	2.1
Mina-SSHD	SSH Protocols	818	04/09-02/17	24.66
Nutch	Web Crawler	893	03/05-01/17	17.85
OpenNLP	NLP Toolkit	424	04/13-02/17	18.85
Parquet-MR	Storage Format	345	02/13-01/17	5.2
Phoenix	OLTP Analytics	1260	01/14-02/17	131.13
Qpid-JMS	Message Service	431	02/15-02/17	9.12
Ranger	Data Security	409	03/15-02/17	19.97
Santuario	XML Security	480	01/11-02/17	16.38
Shiro	Java Security	285	03/09-11/16	3.26
Tiles	Web Templating	318	07/06-07/16	1.71
Zookeeper	Distributed Comp.	610	06/08-02/17	14.98
Total		**19580**	**01/02-03/17**	**586.73**

Research Question 1

- To What Extent Do Developers Commit Impactful Changes?
- Data
  - All commits
  - All impactful commits
  - All developers
  - All impactful developers (at least one impactful commit)
- Ratio
  - Impactful commits to all commits
  - Impactful developers to all developer.

Centrality of the Main Module!

- Impactful
  - —48% of commits.
  - —69% of developers.
- What may significantly affect the ratios?
  - —Architecture of the system.
  - —The level of integration with other systems.
  - —Distribution of tasks during the development.

Impactful Commits and Developers Ratios

	RQ1					
System	Commit			Developer		
	All	Impactful	%	All	Impactful	%
Avro	757	188	25	39	19	49
Calcite	1201	673	56	121	95	79
C-BCEL	900	589	65	13	8	62
C-Beanu.	392	139	35	9	7	78
C-Codec	706	368	52	6	5	83
C-Collect.	890	570	64	16	10	63
C-Comp.	2037	1240	61	30	22	73
C-Config.	637	340	53	4	4	100
C-CSV	1032	606	59	11	7	64
C-DBCP	393	188	48	8	5	63
C-IO	1941	943	49	42	32	76
C-JCS	401	139	35	7	4	57
C-Jexl	675	317	47	8	4	50
C-Net	1588	902	57	11	7	64
C-Pool	546	278	51	11	8	73
C-SCXML	811	348	43	16	8	50
C-Validat.	639	287	45	16	10	63
C-VFS	1242	614	49	21	13	62
CXF-Fediz	1211	190	16	12	6	50
Drill	995	636	64	84	65	77
Flume	1194	347	29	45	26	58
Giraph	585	392	67	33	26	79
Hama	1582	732	46	23	16	70
Helix	1521	800	53	31	21	68
H-Client	1916	934	49	13	9	69
H-Core	1354	566	42	7	6	86
Mina	628	276	44	13	9	69
M-SSHD	1092	843	77	22	21	95
Nutch	2221	926	42	40	28	70
OpenNLP	665	438	66	14	14	100
P-MR	1647	349	21	119	41	34
Phoenix	1908	1290	68	78	62	79
Qpid-JMS	921	431	47	5	3	60
Ranger	1412	415	29	47	25	53
Santuario	743	496	67	3	3	100
Shiro	700	289	41	22	13	59
Tiles	1345	327	24	10	7	70
Zookeeper	1339	618	46	31	25	81
AVG			48			69
DEV			14			15

Research Question 2

- To What Extent and How Do Impactful Commits Break the Compilability of the System?
- Data
  - —Collected all possible compile commands.
  - —Targeted only the main module and its dependencies.
  - —Turned off running tests and validations.
  - —Fixed all missing dependencies over the history.
- Ratio
  - —Compilable impactful commits to all impactful commits.

How to Prevent Breaking Compilability?

- 2% of commits are not even compilable! How to prevent?
- Don't commit too early and too often!
- Compile the project in a new environment!
  - —Contributing alone.
  - —Changing build files.
  - —Adding new files.
  - —Doing maintenance and large refactoring.
- Avoid using snapshot versions of dependencies!

Compiled Impactful Commits

System	RQ2 Impactful Commit		
	All	Compiled	%
Avro	188	188	100
Calcite	673	650	97
C-BCEL	589	576	98
C-Beanu.	139	139	100
C-Codec	368	368	100
C-Collect.	570	565	99
C-Comp.	1240	1229	99
C-Config.	340	333	98
C-CSV	606	598	99
C-DBCP	188	188	100
C-IO	943	914	97
C-JCS	139	136	98
C-Jexl	317	295	93
C-Net	902	877	97
C-Pool	278	278	100
C-SCXML	348	335	96
C-Validat.	287	279	97
C-VFS	614	611	99
CXF-Fediz	190	171	90
Drill	636	616	97
Flume	347	342	99
Giraph	392	387	99
Hama	732	717	98
Helix	800	762	95
H-Client	934	925	99
H-Core	566	565	99
Mina	276	263	95
M-SSHD	843	818	97
Nutch	926	893	96
OpenNLP	438	424	97
P-MR	349	345	99
Phoenix	1290	1260	98
Qpid-JMS	431	431	100
Ranger	415	409	99
Santuario	496	480	97
Shiro	289	285	99
Tiles	327	318	97
Zookeeper	618	610	99
AVG			98
DEV			1

Research Question 3

- To What Extent Do Impactful Commits Affect Software Quality Attributes?
- Data
  - *all(s)*
    - All impactful commits in system s.
  - *change(s,m)*
    - All impactful commits in s that change metric m.
- Ratio
  - *change(s,m) to all(s)*

Software Metrics

- 3 Categories
- 9 Metrics

Group	Abbr.	Tool	Description
Basic	LC	SonarQube	Physical Lines excl. Whitespaces/Comments
	FN	SonarQube	Functions
	CS	FindBugs	Classes
Code Quality	CX	SonarQube	Complexity (Number of Paths)
	SM	SonarQube	Code Smells
	PD	PMD	Empty Code, Naming, Braces, Import Statements, Coupling, Unused Code, Unnecessary, Design, Optimization, String and StringBuffer, Code Size
Security	VL	SonarQube	Vulnerabilities
	SG	PMD	Security Guidelines
	FG	FindBugs	Malicious Code, Security

Impact on Basic

- Physical Lines (LC):
  - —70%
  - —Exceptionally low LC values belong to libraries.
- Functions (FN):
  - —36%
  - —Exceptionally low FN/LC values belong to libraries.
- Classes (CS):
  - —17%
  - —Exceptionally low CS/FN values belong to libraries.

Impact on Basic Metrics

	Basic		
	LC	FN	CS
Avro	80	44	21
Calcite	89	57	34
C-BCEL	54	16	5
C-Beanu.	47	17	7
C-Codec	42	20	6
C-Collect.	51	22	11
C-Comp.	64	32	10
C-Config.	65	34	9
C-CSV	39	16	3
C-DBCP	56	19	5
C-IO	55	29	8
C-JCS	79	47	17
C-Jexl	64	40	19
C-Net	59	21	6
C-Pool	44	18	8
C-SCXML	74	32	14
C-Validat.	56	12	5
C-VFS	51	21	6
CXF-Fediz	78	39	19
Drill	88	48	29
Flume	88	47	29
Giraph	85	58	39
Hama	79	44	25
Helix	87	47	24
H-Client	73	34	14
H-Core	69	40	17
Mina	79	40	15
M-SSHD	85	52	30
Nutch	81	33	17
OpenNLP	68	39	22
P-MR	76	45	29
Phoenix	84	43	20
Qpid-JMS	75	33	14
Ranger	88	44	14
Santuario	77	35	17
Shiro	70	43	21
Tiles	82	54	28
Zookeeper	83	39	15
AVG	70	36	17
DEV	15	13	9

Impact on Code Quality

- Complexity (CX):
  - —53%
  - —Run tests.
- Code Smells (SM):
  - —46%
  - —Take maintenance into account.
- PMD code quality (PD):
  - —60%
  - —Use integrated tools.

Impact on Code Quality Metrics

	Code Quality		
	CX	SM	PD
Avro	70	47	73
Calcite	75	58	83
C-BCEL	31	45	53
C-Beanu.	27	28	40
C-Codec	23	26	36
C-Collect.	30	29	38
C-Comp.	49	39	54
C-Config.	37	28	43
C-CSV	25	20	32
C-DBCP	35	40	46
C-IO	38	35	45
C-JCS	57	48	66
C-Jexl	56	46	59
C-Net	39	39	51
C-Pool	27	27	36
C-SCXML	52	49	63
C-Validat.	26	36	43
C-VFS	29	31	44
CXF-Fediz	62	50	68
Drill	74	58	80
Flume	66	59	72
Giraph	69	49	74
Hama	60	60	71
Helix	70	67	78
H-Client	54	40	56
H-Core	50	41	54
Mina	58	54	67
M-SSHD	75	58	75
Nutch	62	59	74
OpenNLP	54	56	61
P-MR	63	52	66
Phoenix	72	58	77
Qpid-JMS	60	43	60
Ranger	74	63	81
Santuario	66	54	73
Shiro	52	42	56
Tiles	67	56	68
Zookeeper	71	53	69
AVG	53	46	60
DEV	17	12	15

Impact on Security

- SonarQube (VL):
  - —5.7%
  - —No change in two cases.
- PMD (SG):
  - —1.9%
  - —No change in one case.
- FindBugs (FG):
  - —2.4%
  - —No change in two cases.

Impact on Security Metrics

	Security		
	VL	SG	FG
Avro	5.3	2.7	2.1
Calcite	4.1	0.9	0.9
C-BCEL	4.6	3.0	4.1
C-Beanu.	0.0	0.7	0.0
C-Codec	0.8	0.5	0.0
C-Collect.	0.9	1.6	1.6
C-Comp.	3.5	2.3	1.8
C-Config.	0.0	0.9	0.9
C-CSV	0.7	1.2	0.7
C-DBCP	2.7	1.6	2.1
C-IO	1.4	1.8	1.9
C-JCS	8.3	3.0	3.8
C-Jexl	4.2	1.7	1.7
C-Net	4.5	1.4	0.3
C-Pool	3.6	0.7	0.7
C-SCXML	4.2	0.0	0.6
C-Validat.	0.4	0.7	0.7
C-VFS	2.0	0.2	1.0
CXF-Fediz	9.3	1.2	3.1
Drill	8.3	1.5	3.2
Flume	7.4	1.2	1.8
Giraph	7.7	2.6	1.6
Hama	18.9	6.3	7.7
Helix	9.0	1.4	2.9
H-Client	2.9	2.0	1.6
H-Core	0.4	1.8	1.2
Mina	9.6	1.2	0.8
M-SSHD	17.0	2.6	3.7
Nutch	10.1	3.2	4.9
OpenNLP	6.8	4.6	4.1
P-MR	1.9	4.8	2.9
Phoenix	10.2	4.0	7.5
Qpid-JMS	5.8	1.9	2.3
Ranger	17.3	0.3	3.8
Santuario	6.7	3.0	4.3
Shiro	2.8	2.1	2.5
Tiles	7.1	0.6	2.9
Zookeeper	8.0	1.8	4.0
AVG	5.7	1.9	2.4
DEV	4.7	1.4	1.8

Research Question 4

- Should Developers Rely on a Single Software Quality Metric as a Change Indicator?
- Data
  - —*const(x)*
    - Impactful commits that do not change metric *x.*
  - —*const(x) ∩ change(y)*
    - Impactful commits in which metric *x* is constant while metric *y* changes.
- Ratio
  - —*const(x) ∩ change(y) to const(x)*

No Single Quality Metric Alone Suffices!

- Constant Physical Line
  - —All other attributes can change.
- Constant Functions and Classes
  - —There is a jump in VL and SG.
- Constant Code Quality Metrics
  - —Multiple metrics needs to be used.
  - —Security metrics don't change much.

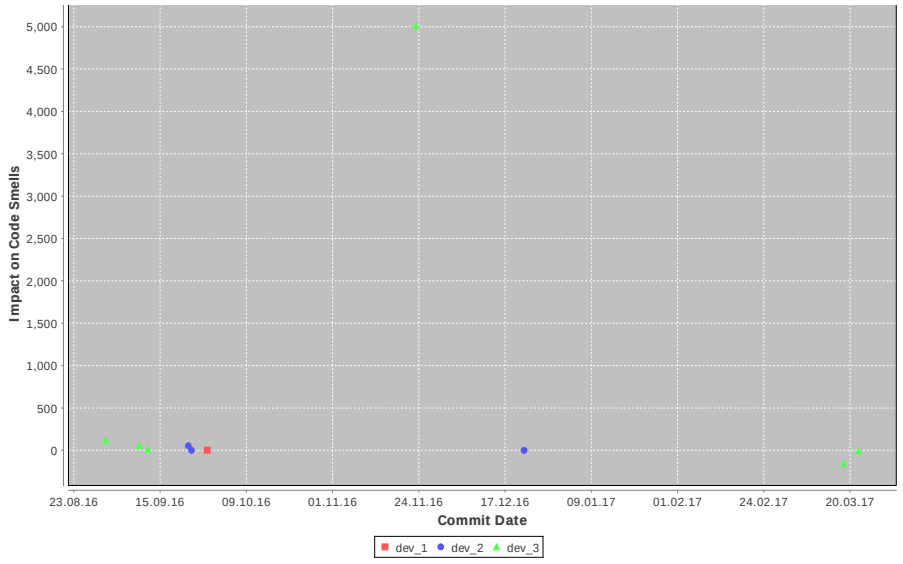
Const	Change								
	Basic			Code Quality			Security		
	LC	FN	CS	CX	SM	PD	VL	SG	FG
LC	-	1.2	0.4	5.5	13.3	17.0	0.7	0.2	0.6
FN	55.1	-	2.0	29.6	31.0	43.2	2.0	0.3	0.7
CS	65.1	24.5	-	45.7	38.4	54.0	3.1	0.9	1.2
CX	40.1	1.8	1.6	-	22.8	30.2	1.2	0.3	0.7
SM	52.6	17.1	3.9	33.5	-	38.8	1.3	0.5	0.6
PD	37.9	6.5	1.7	17.7	16.2	-	1.0	0.3	0.5
VL	69.1	32.8	13.8	51.5	43.7	58.8	-	1.6	1.4
SG	70.2	34.5	15.5	53.1	45.6	60.2	5.7	-	1.4
FG	70.1	34.4	15.2	53.0	45.3	60.0	4.9	0.7	-

- Constant Security
  - —Each metric shows a different aspect.

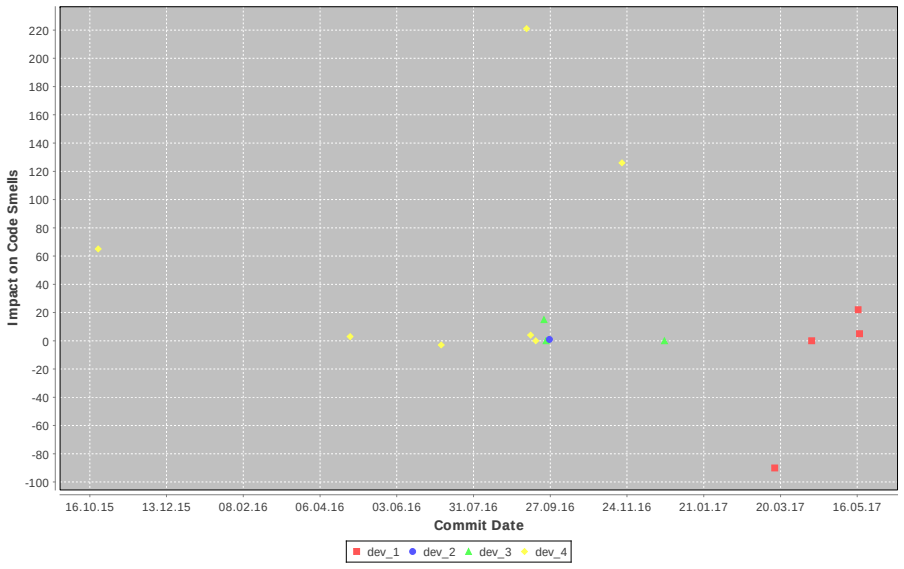
Translating Research Into Practice

- We recently delivered advanced tool assessments tutorials to front line acquisition engineers of a major governmental entity.
- This led to an in-depth analysis of the quality aspects of an open source software complex for decisions regarding quality, safety, and security "sniffs" and "taints" to assess an acquisition program of an unmanned system.

NASA-SYSTEM1



NASA-SYSTEM2



Conclusion

- Studying software quality before and after each commit (commit-impact analysis) can reveal a wealth of information about how the software evolves and how each change impacts its quality.
- Software developers who contribute to an open source software system have a high level of engagement in developing its the core module.
- An unexpectedly high ratio of commits are not even compliable in open source software systems.
- Different quality attribute can change even if the size does not change.
- Using one programming analysis technique does not suffice to evaluate change in software quality.

Future Work

- Difference between developers.
  - —Recently published a paper at IEEE STC.
- Increase and decrease in quality attributes.
- Dynamic analysis and regression tests.
- Effect and intent of the changes.
- Defect prediction models.