Towards Better Understanding of Conflicts and Synergies Among Software Quality Attributes By Analysis of Abundant Data

Pooyan Behnamghader
USC Center for Systems and Software Engineering

Research Task / Overview

- Understanding software quality evolution, and conflicts and synergies among software quality attributes by analyzing impactful changes.
- Utilizing multiple programming analysis techniques to conduct multi-perspective analysis of software quality.
- Capitalizing on cloud services to analyze revision histories of large families of open-source software systems.
- Conducting large-scale replicable empirical studies.
- Providing interactive desktop and web interfaces to illustrate software quality evolution.

Goals & Objectives

- Helping 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.
- Helping project 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.
- Helping developers continuously monitor software quality and improve software maintainability.
- Helping acquisition programs evaluate system quality.

Methodology

How to Identify Impactful Changes?
Version Control System (i.e., Git)
- Tracks changes and contains fine-grained information, such as when change occurs, how, and by who!

Commit-Impact Analysis
- Analyzing software quality before and after commits that change the source code of the main module of software.

How to Evaluate Change in Quality?
Static Analysis Techniques
- Analyze software without running it.

Report Parsers
- Retrieve quality metrics from reports and store them in a unified relational schema.

How to Explore the Data?
Interactive Desktop and Web Interfaces.
- Evolution trend of a metric.
- Impact of each developer.
- Evolution graph of a metric.
- Co-evolution of multiple metrics.

Figure 1. A Software System's Evolution. (Impactful commits are denoted in gray.)

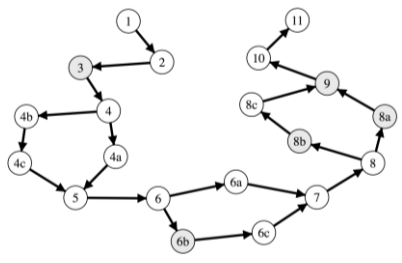


Table 1. Software Quality Metrics

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

Figure 2. Simple Analysis Workflow

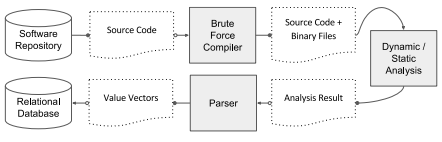


Table 2. Analysis Data Stored in Unified Relational Schemas

Application	Commit	LC	FN	CS	CX
apache-phoenix	92623a0ffa4d2132c3e1c2f0906924dec87c637	59096	5405	1048	15814
apache-phoenix	be8c2fb500b01774b83008a8d2152aea8a70905	59091	5404	1048	15814
apache-phoenix	0a39c5fd1b43462991851ff9184f4aa7f78d24348	59096	5404	1048	15820
apache-phoenix	334b83663c3149e49b9694c87ece6e0c97633e0e	59135	5404	1048	15830
apache-phoenix	8313a2a28c021c80e42b52e26525c5866eb5e987	59118	5404	1048	15825
apache-phoenix	473ca2a7c552209b17e6f0414 1cebdef2d323823	59122	5404	1048	15827
apache-phoenix	fc96ccb91050ebedc1315a6fb6e1c63671d4ed9	59111	5404	1048	15819
apache-phoenix	d40c5066edeb112fb59f7bebe6c78e2fb346d929	59120	5405	1048	15821

Figure 3. Evolution Trend of a Metric in a Subject System

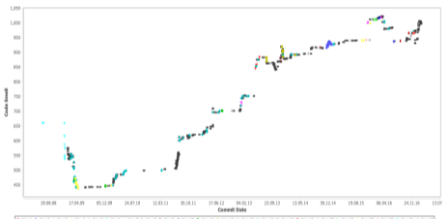


Figure 5. Evolution Graph of a Metric in a Subject System

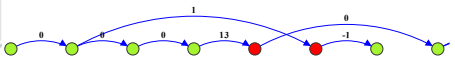


Figure 4. Impact of Developers on Software Quality

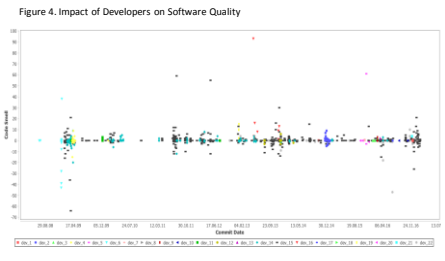
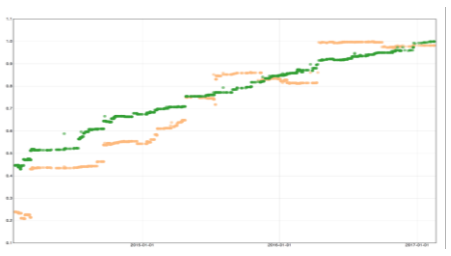


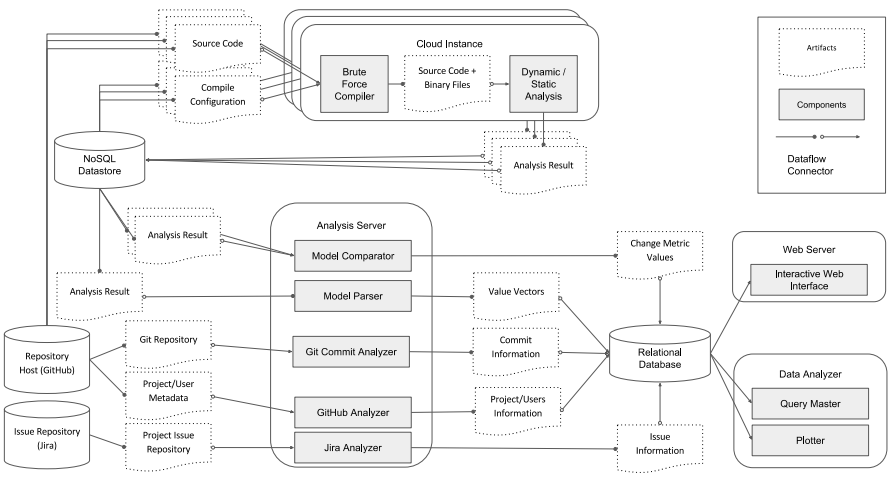
Figure 6. Coevolution of Two Metrics in a Subject System



How to Scale?
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. Run static/dynamic programming analysis techniques on each revision.
4. Collect and parse the artifacts generated by programming analysis techniques to extract quality attributes.
5. Run various statistical analysis on software quality evolution.

The entire analysis workflow is automated.

Figure 7. Workflow Architecture of Commit-Impact Analysis



Data & Analysis

Research Questions
- RQ1: To what extent do developers commit impactful changes?
- RQ2: To what extent and how do impactful commits break the compilability?
- RQ3: To what extent do impactful commits affect software quality attributes?
- RQ4: Should developers rely on a single 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

Subject system selection criteria:
- Java, updated in 2017, the main module exists, no nontrivial prerequisite for compilation, at least 100 compilable different revisions, and less than 3K commits.

Scale of the analysis
- 3 programming analysis techniques, 9 metrics, 38 systems, 19580 impactful commits and revisions, 643 impactful developers, 586 MSLOC, 15 years timespan.

Table 3. Empirical Study Setup and the Result of the Analysis for RQ1, RQ3, and RQ3

	Empirical Study Setup				RQ1						RQ2		RQ3								
System	Domain	Rev.	Time span	Size	Commit			Developer			Commit		Basic			Code Quality			Security		
		#	MM/YY	MSLOC	All #	Impactful #	Impactful %	All #	Impactful #	Impactful %	Compiled #	Compiled %	LC %	FN %	CS %	CX %	SM %	PD %	VL %	SG %	FG %
Avro	Data Serialization	188	07/11-01/17	2.43	757	188	25	39	19	49	188	100	80	44	21	70	47	73	5.3	2.7	2.1
Calcite	Data Management	650	07/14-02/17	75.46	1201	673	56	121	95	79	650	97	89	57	34	75	58	83	4.1	0.9	0.9
CJDCEL	Bytecode Eng.	576	06/06-12/16	16.5	900	589	65	13	8	62	576	98	54	16	5	31	45	53	4.6	3.0	4.1
C-Beanm.	Reflection Wrapper	139	07/07-11/16	1.71	392	139	35	9	7	78	139	100	47	17	7	27	28	40	0.0	0.7	0.0
C-Codec	Encoder/Decoder	368	09/11-09/16	2.32	706	368	52	6	5	83	368	100	42	20	6	23	26	36	0.8	0.5	0.0
C-Collect.	Collections Ext.	565	03/12-10/16	13.87	890	570	64	16	10	63	565	99	51	22	11	30	29	38	0.9	1.6	1.6
C-Comp.	Compress Lib.	1229	07/08-02/17	16.84	2037	1240	61	30	22	73	1229	99	64	32	10	49	39	54	3.5	2.3	1.8
C-Config.	Conf. Interface	333	04/14-01/17	8.83	637	340	53	4	4	100	333	98	65	34	9	37	28	43	0.0	0.9	0.9
C-CSV	CSV Library	598	11/11-02/17	0.68	1032	606	59	11	7	64	598	99	39	16	3	25	20	32	0.7	1.2	0.7
C-DBCP	DB Conn. Pooling	188	12/13-11/16	2.06	393	188	48	8	5	63	188	100	56	19	5	35	40	46	2.7	1.6	2.1
C-IO	IO Functionality	914	01/02-12/16	5.99	1941	943	49	42	32	76	914	97	55	29	8	38	35	45	1.4	1.8	1.9
C-JCS	Java Caching	136	04/14-02/17	3.66	401	139	35	7	4	57	136	98	79	47	17	57	48	66	8.3	3.0	3.8
C-Jexl	Expression Lang.	295	08/09-10/16	2.26	675	317	47	8	4	50	295	93	64	40	19	56	46	59	4.2	1.7	1.7
C-Net	Clientside Protocol	877	08/06-02/17	14.96	1588	902	57	11	7	64	877	97	59	21	6	39	39	51	4.5	1.4	0.3
C-Pool	Object Pooling	278	04/12-02/17	1.27	546	278	51	11	8	73	278	100	44	18	8	27	27	36	3.6	0.7	0.7
C-SCXML	State Chart XML	335	03/06-08/16	2.53	811	348	43	16	8	50	335	96	74	32	14	52	49	63	4.2	0.0	0.6
C-Validat.	Data Verification	279	07/07-02/17	1.63	639	287	45	16	10	63	279	97	56	12	5	26	36	43	0.4	0.7	0.7
C-VFS	Virtual File System	611	11/06-01/17	13.32	1242	614	49	21	13	62	611	99	51	21	6	29	31	44	2.0	0.2	1.0
CXF-Fediz	Web Security	171	04/12-03/17	0.89	1211	190	16	12	6	50	171	90	78	39	19	62	50	68	9.3	1.2	3.1
Drill	SQL Query Engine	616	04/15-02/17	58	995	636	64	84	65	77	616	97	88	48	29	74	58	80	8.3	1.5	3.2
Flume	Data Collection	342	08/11-11/16	2.14	1194	347	29	45	26	58	342	99	88	47	29	66	59	72	7.4	1.2	1.8
Giraph	Graph Processing	387	11/12-01/17	15.12	585	392	67	33	26	79	387	99	85	58	39	69	49	74	7.7	2.6	1.6
Hama	BSP Computing	717	06/08-04/16	7.77	1582	732	46	23	16	70	717	98	79	44	25	60	60	71	18.9	6.3	7.7
Helix	Cluster MNGMT	762	01/12-06/16	27.36	1521	800	53	31	21	68	762	95	87	47	24	70	67	78	9.0	1.4	2.9
H-Client	Client-side HTTP	925	03/09-01/17	16.54	1916	934	49	13	9	69	925	99	73	34	14	54	40	56	2.9	2.0	1.6
H-Core	HTTP Transport	565	03/09-02/17	7.38	1354	566	42	7	6	86	565	99	69	40	17	50	41	54	0.4	1.8	1.2
Mina	Socket Library	263	11/09-04/15	2.1	628	276	44	13	9	69	263	95	79	40	15	58	54	67	9.6	1.2	0.8
M-SSHD	SSH Protocols	818	04/09-02/17	24.66	1092	843	77	22	21	95	818	97	85	52	30	75	58	75	17.0	2.6	3.7
Nutch	Web Crawler	893	03/05-01/17	17.85	2221	926	42	40	28	70	893	96	81	33	17	62	59	74	10.1	3.2	4.9
OpenNLP	NLP Toolkit	424	04/13-02/17	18.85	665	438	66	14	14	100	424	97	68	39	22	54	56	61	6.8	4.6	4.1
P-MR	Storage Format	345	02/13-01/17	5.2	1647	349	21	119	41	34	345	99	76	45	29	63	52	66	1.9	4.8	2.9
Phoenix	OLTP Analytics	1260	01/14-02/17	131.13	1908	1290	68	78	62	79	1260	98	84	43	20	72	58	77	10.2	4.0	7.5
Qpid-JMS	Message Service	431	02/15-02/17	9.12	921	431	47	5	3	60	431	100	75	33	14	60	43	60	5.8	1.9	2.3
Ranger	Data Security	409	03/15-02/17	19.97	1412	415	29	47	25	53	409	99	88	44	14	74	63	81	17.3	0.3	3.8
Santuario	XML Security	480	01/11-02/17	16.38	743	496	67	3	3	100	480	97	77	35	17	66	54	73	6.7	3.0	4.3
Shiro	Java Security	285	03/09-11/16	3.26	700	289	41	22	13	59	285	99	70	43	21	52	42	56	2.8	2.1	2.5
Tiles	Web Templating	318	07/06-07/16	1.71	1345	327	24	10	7	70	318	97	82	54	28	67	56	68	7.1	0.6	2.9
Zookeeper	Distributed Comp.	610	06/08-02/17	14.98	1339	618	46	31	25	81	610	99	83	39	15	71	53	69	8.0	1.8	4.0
					AVG	48			69		98		70	36	17	53	46	60	5.7	1.9	2.4
ALL		19580	01/02-03/17	586.73	DEV	14			15		1		15	13	9	17	12	15	4.7	1.4	1.8

Results' Highlight
- On average, 48% of all commits impact the main module, and 69% of all developers contribute to the source code in that main module.
- On average, 2% of the impactful commits are not compilable.
- Different quality attributes may change even if the code count does not change.
- Only 5.7% impactful commits change VL, 1.9% change SG, and 2.4% change FG.
- Although security metrics change less frequently, it is crucial to utilize them as they can reveal the introduction of different kinds of security problems.

Tips to Avoid Introducing Compile Error When Contributing to Open Source Projects
Avoid using snapshot of dependencies!
Compile in a new environment:
- Contributing alone.
- Changing build files.
- Adding new files.
- Conducting large refactoring.

Don't commit too early and too often!

Table 4. Percentage of Const(X) \ Change(Y) to Const(X). Const(X) is the number of impactful commits that does not change a quality metric X, and Change(X) is the number of impactful commits that change X.

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	-

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.

Follow-up Study
"How Does Contributors' Involvement Influence Open Source Systems" [2]. In this paper, we investigated whether variations in contributors level of involvement shows variation in their contribution type and its quality with an emphasis on technical debt.

Figure 8: Code Smell Evolution in NASA-SYSTEM1

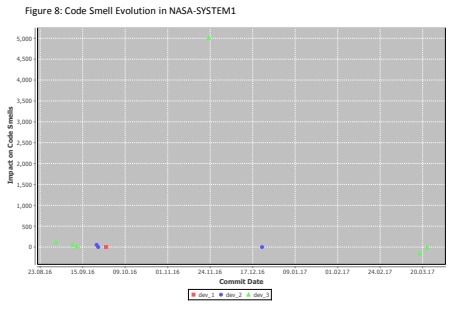
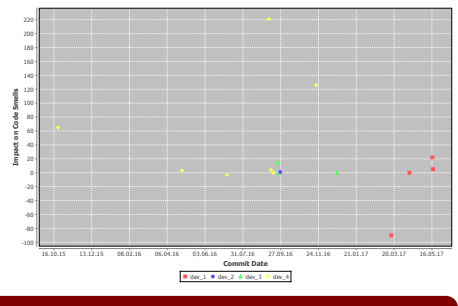


Figure 9: Code Smell Evolution in NASA-SYSTEM2



Contacts/References

References:
[1] P. Behnamghader, R. Alfayez, K. Srisopha, and B. Boehm. 2017. Towards Better Understanding of Software Quality Evolution through Commit-Impact Analysis. In 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS). 251–262.
[2] R.Alfayez, P. Behnamghader, K. Srisopha, and B. Boehm. 2017. How Does Contributors' Involvement Influence Open Source Systems. In 28th Annual IEEE Software Technology Conf.

Contact Info:
- 941 Bloom Walk, SAL 327. Los Angeles, CA 90089
- +1 213 400 7256
- pbehnamg@usc.edu

Future Research

- Difference between developers in terms of impact on software quality.
- Studies with dynamic analysis techniques and regression tests.
- Effect and intent of the changes on software quality.
- Software quality defect prediction models.