ARCHITECTURE EROSION

SWE 4601

Understanding Architecture Erosion: The Practitioners' Perceptive, ICPC 2021 Conference

What is it?

- During the lifespan of a software system, its software architecture is constantly modified to satisfy new requirements and accommodate changes in the environment
- with the increasing complexity and changing requirements, the implementation may deviate from the architecture over time [2]. This divergence between the intended and the implemented architecture is often called Architecture Erosion (AEr).
- It is also known by
  - architectural decay
  - architecture degeneration
  - architecture degradation
  - design erosion

Impacts of AEr

- AEr can decrease software performance
- substantially increase evolutionary costs, and
- degrade software quality
- in an eroded architecture, code changes and refactorings could introduce new bugs and aggravate the brittleness of the system

Why the study stand out?

- It deals with the perception of developers.

- An in-depth exploration of the viewpoints of developers on the notion of AEr,
  - the causes and consequences of AEr,
  - the used practices and tools for detecting AEr, and
  - the measures to control and prevent AEr.

Study Design

- ❖ Collected data of developer discussions from six popular online developer communities related to architecture erosion.
- ❖ Conducted a survey with 10 participants and
- ❖ Held interviews with 4 participants.

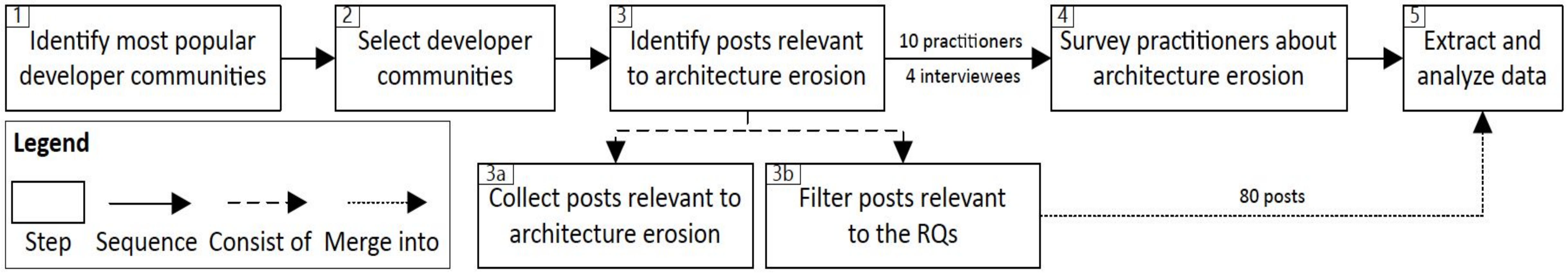
- ❖ Analyzed collected data sources (i.e., posts, questions and answers, surveys, interviews) using Constant Comparison.

analyze the perception of architecture erosion in practice **for the purpose of** understanding **with respect to** the notion, causes, consequences, detection and control of architecture erosion **from the point of view of** developers **in the context of** industrial software development.

Research Questions

- ❖ RQ1: How do developers describe architecture erosion in software development?
  - ➢ RQ1.1: Which **terms** do developers use to indicate architecture erosion?
  - ➢ RQ1.2: How does architecture erosion **manifest** according to developers?
- ➔ architectural decay, architecture degeneration, architecture degradation
- ❖ RQ2: What are the **causes and consequences** of architecture erosion from the perspective of developers?
- ➔ architecture smells, and violations of architecture decisions
- ➔ slowing down development or hampering maintenance.
- ❖ RQ3: What practices and tools are used to detect architecture erosion in software development?
- ❖ RQ4: What measures are employed to control architecture erosion in software development?

Overview of Research Approach


1 Identify most popular developer communities
2 Select developer communities
3 Identify posts relevant to architecture erosion
10 practitioners
4 interviewees
4 Survey practitioners about architecture erosion
5 Extract and analyze data
3a Collect posts relevant to architecture erosion
3b Filter posts relevant to the RQs
80 posts
Legend
Step
Sequence
Consist of
Merge into

Data Collection & Dataset

1. Identify most popular developer communities

➔ **Execute search queries**: Google search: "popular/ranking/top/best online developer communities/forums".

➔ **Identify websites**: the frequency of websites mentioned in the search results.

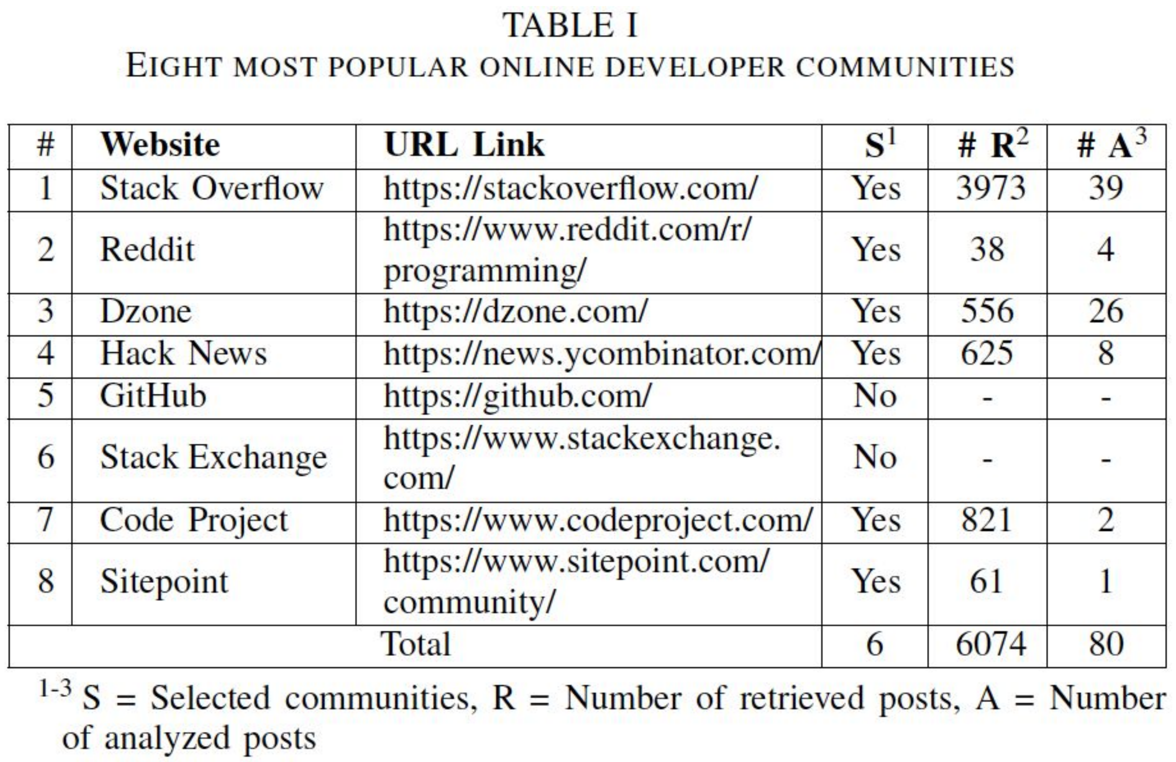


TABLE I

EIGHT MOST POPULAR ONLINE DEVELOPER COMMUNITIES

#	Website	URL Link	S[1]	# R[2]	# A[3]
1	Stack Overflow	https://stackoverflow.com/	Yes	3973	39
2	Reddit	https://www.reddit.com/r/programming/	Yes	38	4
3	Dzone	https://dzone.com/	Yes	556	26
4	Hack News	https://news.ycombinator.com/	Yes	625	8
5	GitHub	https://github.com/	No	-	-
6	Stack Exchange	https://www.stackexchange.com/	No	-	-
7	Code Project	https://www.codeproject.com/	Yes	821	2
8	Sitepoint	https://www.sitepoint.com/community/	Yes	61	1
		Total	6	6074	80

[1-3] S = Selected communities, R = Number of retrieved posts, A = Number of analyzed posts

Data Collection & Dataset

2. Select developer communities

→ pilot search using "architecture erosion", "architecture degradation", "architecture decay".

3. Identify posts relevant to architecture erosion

→ Collect posts relevant to architecture erosion
  - ((architecture OR architectural OR structure OR structural) AND (decay OR erode OR erosion OR degrade OR degradation OR deteriorate OR deterioration))

→ Filter posts relevant to the RQs
  - **Include**: The post (and its answers) discusses AEr and is relevant to answering the RQs.
  - **Exclude**: If two or more posts contain similar information,

Data Collection & Dataset

4. Survey practitioners about architecture erosion

→ Manually inspected the profiles of users, who were involved in the discussion of the 80 collected and filtered posts

→ Identified 38 valid,

→ searched linkedin and got info

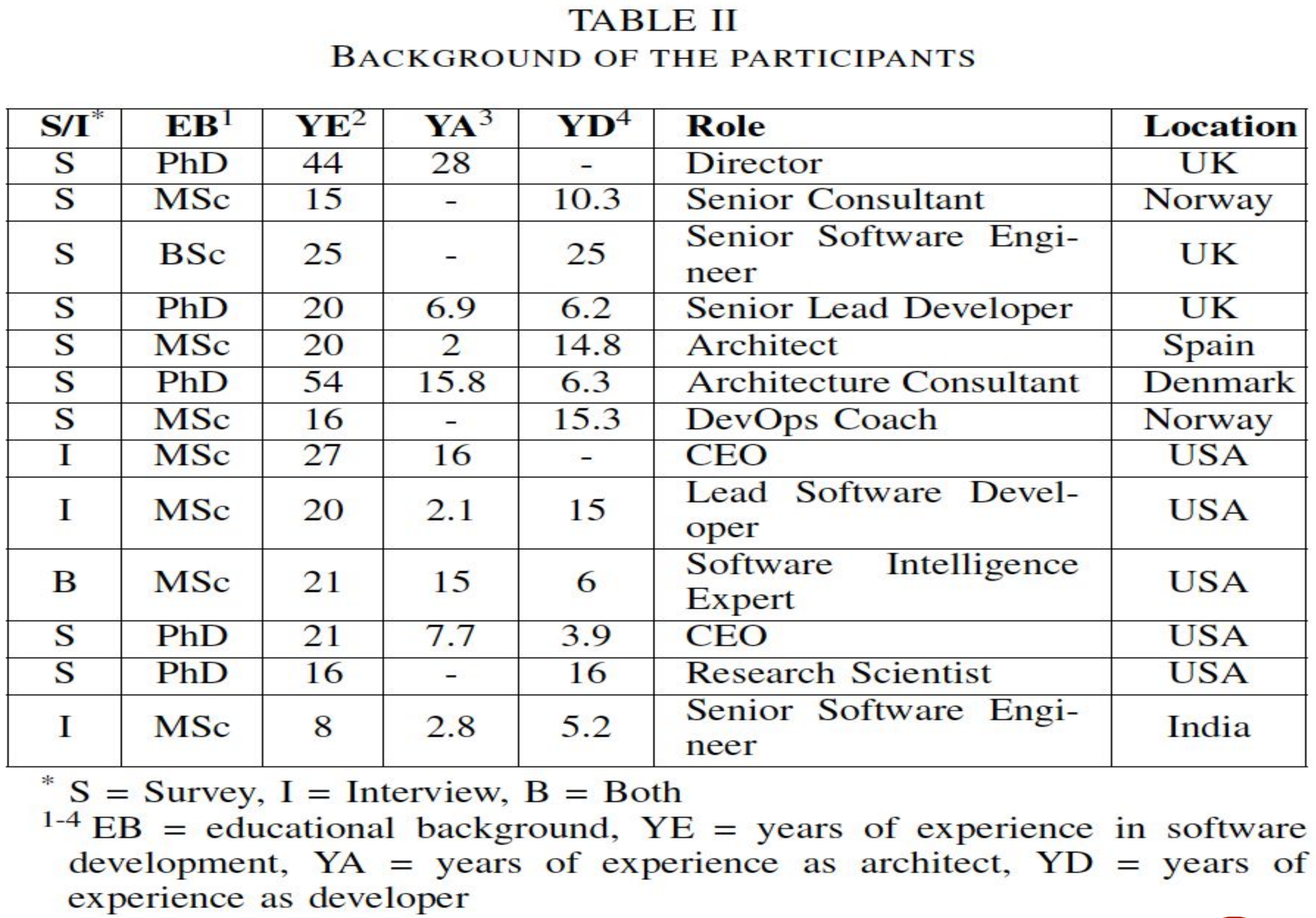


TABLE II
BACKGROUND OF THE PARTICIPANTS

S/I[*]	EB[1]	YE[2]	YA[3]	YD[4]	Role	Location
S	PhD	44	28	-	Director	UK
S	MSc	15	-	10.3	Senior Consultant	Norway
S	BSc	25	-	25	Senior Software Engineer	UK
S	PhD	20	6.9	6.2	Senior Lead Developer	UK
S	MSc	20	2	14.8	Architect	Spain
S	PhD	54	15.8	6.3	Architecture Consultant	Denmark
S	MSc	16	-	15.3	DevOps Coach	Norway
I	MSc	27	16	-	CEO	USA
I	MSc	20	2.1	15	Lead Software Developer	USA
B	MSc	21	15	6	Software Intelligence Expert	USA
S	PhD	21	7.7	3.9	CEO	USA
S	PhD	16	-	16	Research Scientist	USA
I	MSc	8	2.8	5.2	Senior Software Engineer	India

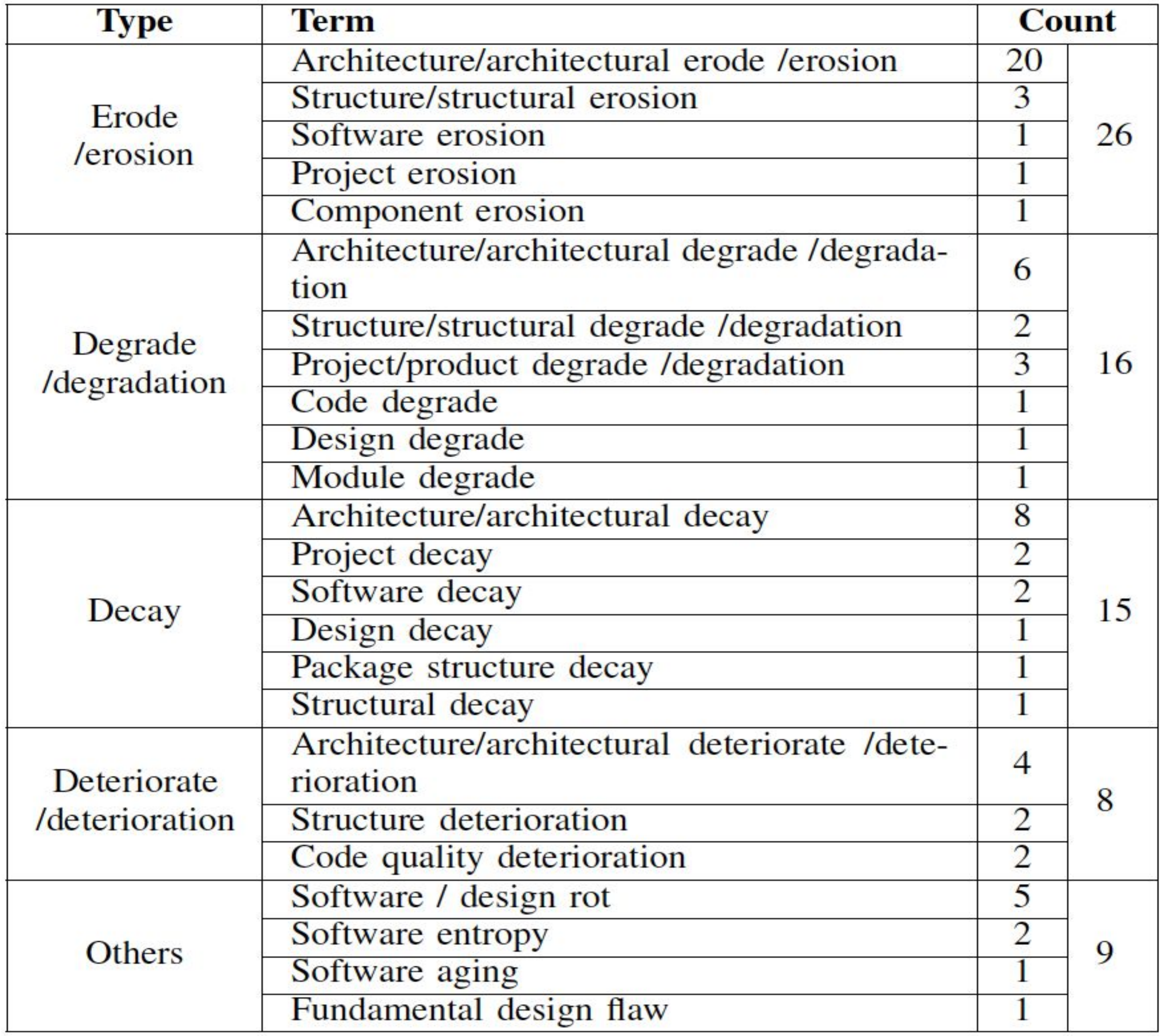
[*] S = Survey, I = Interview, B = Both
[1-4] EB = educational background, YE = years of experience in software development, YA = years of experience as architect, YD = years of experience as developer

Analysis on Extracted Data

- Use of Grounded theory method - Constant Comparison
- Open Coding, Axial Coding and Selective Coding

Results and Discussions

TABLE IV
TERMS THAT DEVELOPERS USED TO DESCRIBE ARCHITECTURE EROSION



Type	Term	Count	
Erode /erosion	Architecture/architectural erode /erosion	20	26
	Structure/structural erosion	3	
	Software erosion	1	
	Project erosion	1	
	Component erosion	1	
Degrade /degradation	Architecture/architectural degrade /degradation	6	16
	Structure/structural degrade /degradation	2	
	Project/product degrade /degradation	3	
	Code degrade	1	
	Design degrade	1	
	Module degrade	1	
Decay	Architecture/architectural decay	8	15
	Project decay	2	
	Software decay	2	
	Design decay	1	
	Package structure decay	1	
	Structural decay	1	
Deteriorate /deterioration	Architecture/architectural deteriorate /deterioration	4	8
	Structure deterioration	2	
	Code quality deterioration	2	
Others	Software / design rot	5	9
	Software entropy	2	
	Software aging	1	
	Fundamental design flaw	1	

Results and Discussions

- ❖ manifestation of AEr

  interviewee #1, “Architecture (erosion) is about the original architecture blueprint got lost when you can’t know architecture structure and boundaries anymore”.

- ★ From the structure perspective
  - ➢ For example, the violation of design rules about encapsulation that breaks implemented abstract layers
  - ➢ cyclic dependencies and increased coupling, both resulting in binding elements together that were intentionally separated by architects.
  - ➢ dead/overlapping code, and obsolete or incompatible third-party libraries.

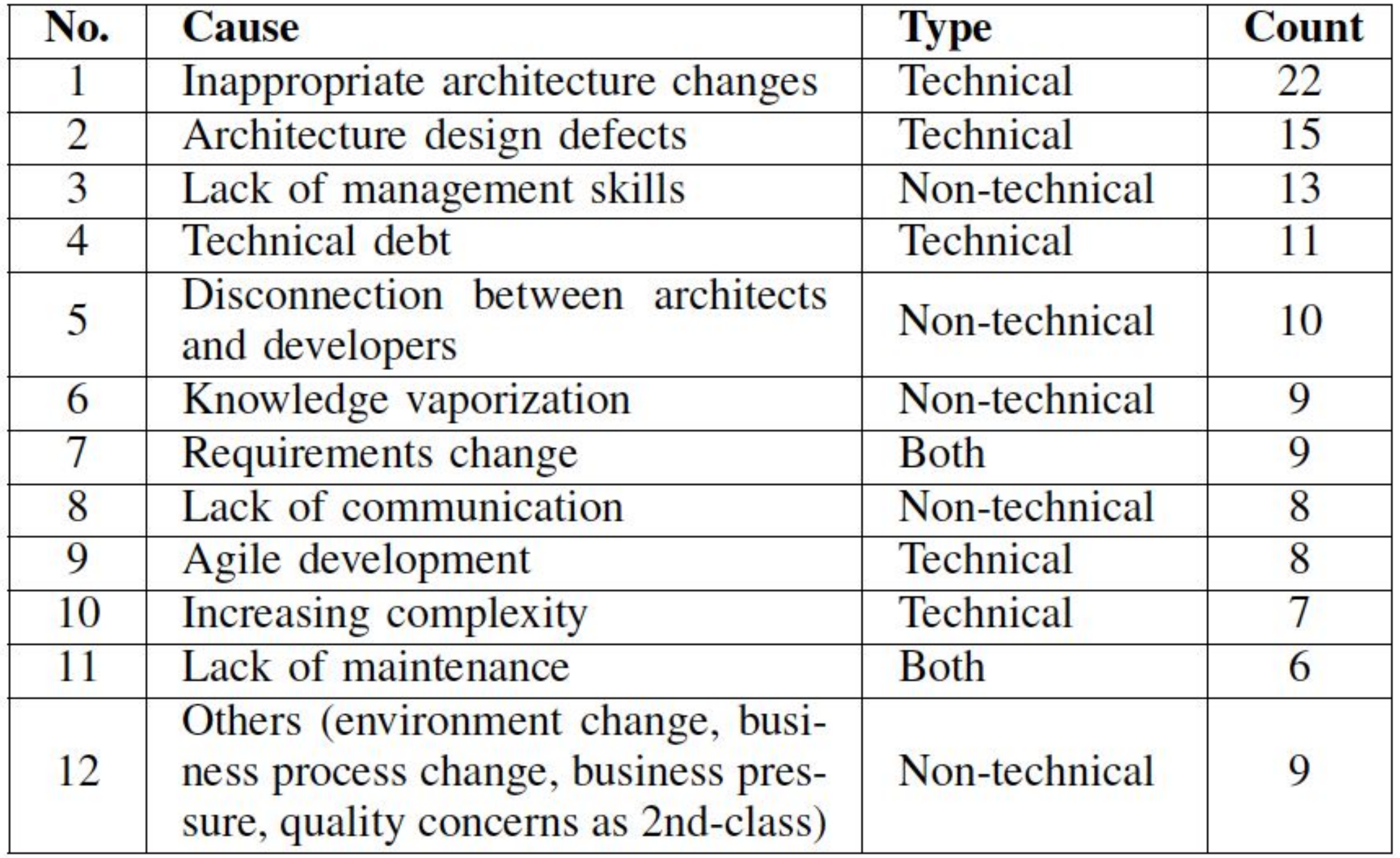
Results and Discussions

- ★ From the quality perspective: an eroded architecture may not meet the original or current non-functional requirements; performance, security, scalability, reliability.
- ★ From the maintenance perspective: an eroded architecture could be harder to understand, fix bugs, and refactor.
- ★ From the evolution perspective: an eroded architecture makes it hard or even impossible to plan the next evolution steps, e.g., which features to implement next or which technologies to adopt.
  - ➢ developer #3 pointed out “the unfortunate side effect of this [erosion] is that it becomes more and more difficult to add new visible features without breaking something”.

Results and Discussions

★ RQ2 - Causes and consequences of architecture erosion

TABLE V
CAUSES OF ARCHITECTURE EROSION

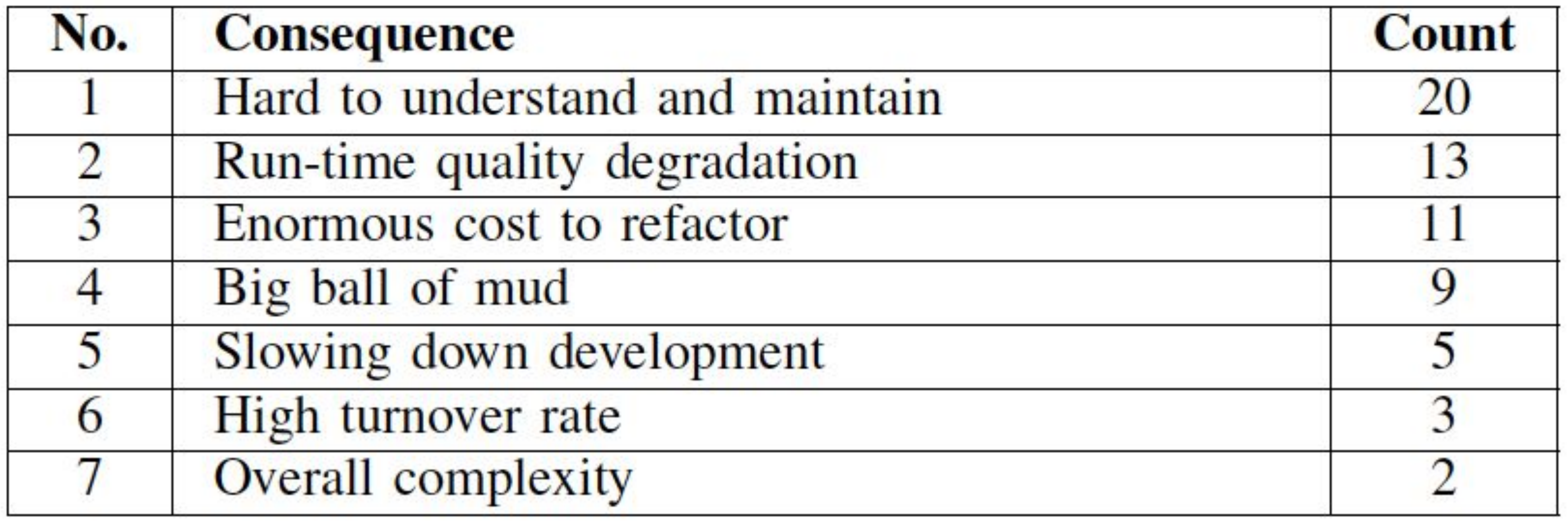


No.	Cause	Type	Count
1	Inappropriate architecture changes	Technical	22
2	Architecture design defects	Technical	15
3	Lack of management skills	Non-technical	13
4	Technical debt	Technical	11
5	Disconnection between architects and developers	Non-technical	10
6	Knowledge vaporization	Non-technical	9
7	Requirements change	Both	9
8	Lack of communication	Non-technical	8
9	Agile development	Technical	8
10	Increasing complexity	Technical	7
11	Lack of maintenance	Both	6
12	Others (environment change, business process change, business pressure, quality concerns as 2nd-class)	Non-technical	9

Results and Discussions

- RQ2 - Causes and consequences of architecture erosion

TABLE VI
CONSEQUENCES OF ARCHITECTURE EROSION



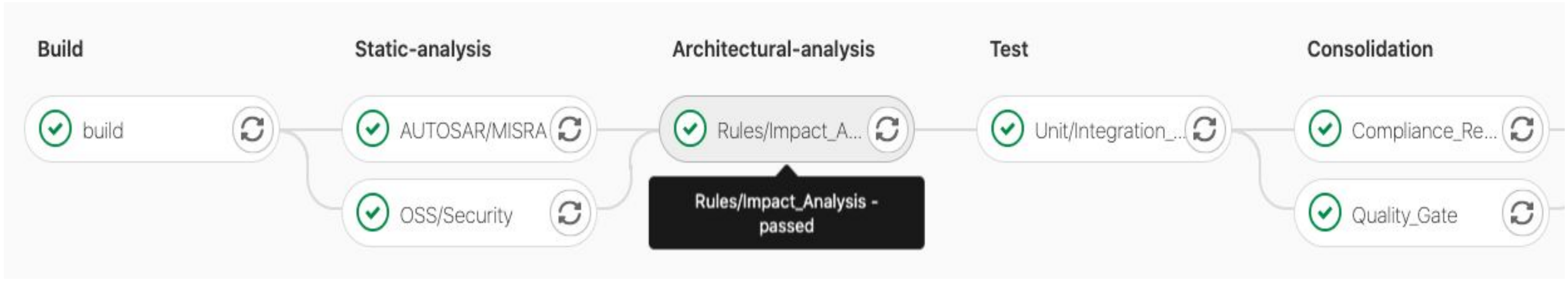
No.	Consequence	Count
1	Hard to understand and maintain	20
2	Run-time quality degradation	13
3	Enormous cost to refactor	11
4	Big ball of mud	9
5	Slowing down development	5
6	High turnover rate	3
7	Overall complexity	2

Results and Discussions

★ RQ3 - Identifying architecture erosion

➔ Tools

- Lattix - commercial tool → allows users to create dependency models to identify architecture issues;
- NDepend - commercial tool → can help users to find out architectural anomalies;
- Structure101 - commercial tool → offers views of code organization and helps practitioners to better understand the code structure and dependencies for checking architecture conformance.
- JDepend, Archie, Designite, SonarQube, SonarLint etc.



Results and Discussions

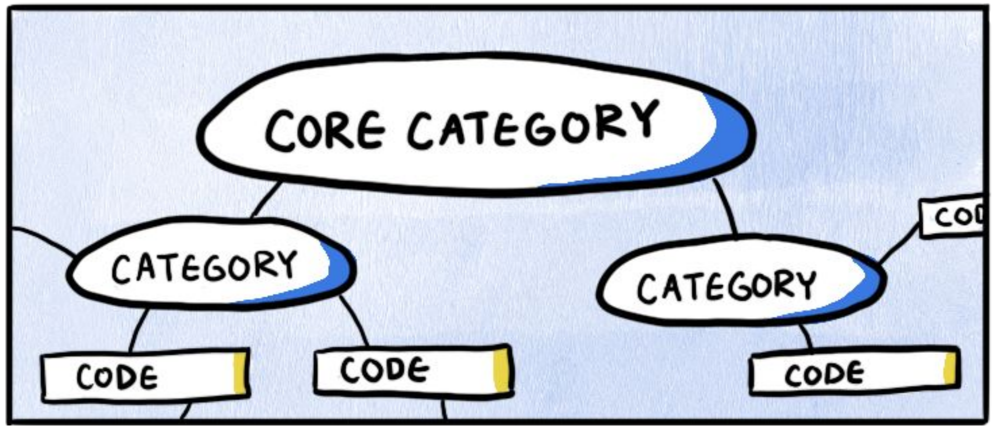
- RQ3 - Identifying architecture erosion
- Practices
  - **Dependency Structure Matrix (DSM)** can be used to visually represent dependency relationships between packages, classes of a system in the form of square matrix.
  - **Software Composition Analysis (SCA)** refers to the process that provides visibility of open source components in a system. T*ool should update the developers when a new open-source library becomes available.* SCA can especially help to determine latent obsolete components that can typically cause AEr. For example: WhiteSource
  - **Architecture Conformance Checking (ACC)** refers to the type of conformance between the implemented architecture and the intended architecture.
  - **Architecture monitoring** refers to using tools to monitor the health of the architecture by detecting architecture issues. such as coupling, size of files, hotspots.
  - **Code review** is a continuous and systematic process conducted by developers or architects to identify mistakes in code, such as violations of design patterns.
  - **Checking the change of architectural smell density** By observing the change rate of architectural smell density, developers can find out from which version, the architecture started to deteriorate.
  - **Architecture visualization** aims at representing architectural models and architectural design decisions using various visual notations.

Results and Discussions

- Q4 - Addressing architecture erosion
  - **Architecture assessment** refers to the assessment process throughout the life cycle during architecture design, implementation, evolution. developer #5 mentioned "Architecture erosion can happen in any software project where the architectural assessments are not part of the development process.
  - **Periodic maintenance** refers to regular activities (e.g., code refactoring, bug-fixing, testing) aimed at keeping a system "clean" and running smoothly.
  - **Architecture simplification**. When architectural complexity proliferates towards being uncontrollable, simplifying the architecture, and deliberately controlling the system size and complexity could be an option worthy of consideration. Refactoring
  - **Architecture restructuring** is a drastic, yet effective means to control AEr. Start by writing a set of tests and then redesign and re-architect the whole solution". However, it may require enormous time and effort.
  - **Organization optimization**. Hiring more capable team members might also be an good option to address AEr. Investment in people: training, study time, mentoring, etc".

Open, Axial and Selective Coding in Grounded Theory

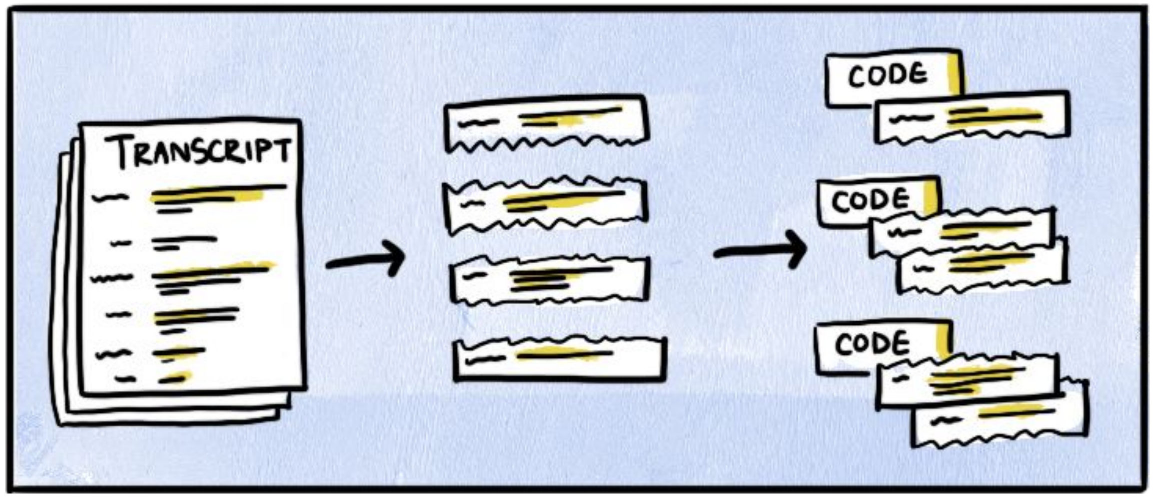
- With grounded theory, you derive new theories and concepts based on data, in contrast to other methods where you start with an existing theory and see whether or not your data applies to the theory
- Corbin and Strauss outline their approach to open coding, axial coding and selective coding in their 1990 paper, "Grounded theory research: Procedures, canons, and evaluative criteria."



https://delvetool.com/blog/openaxialselective

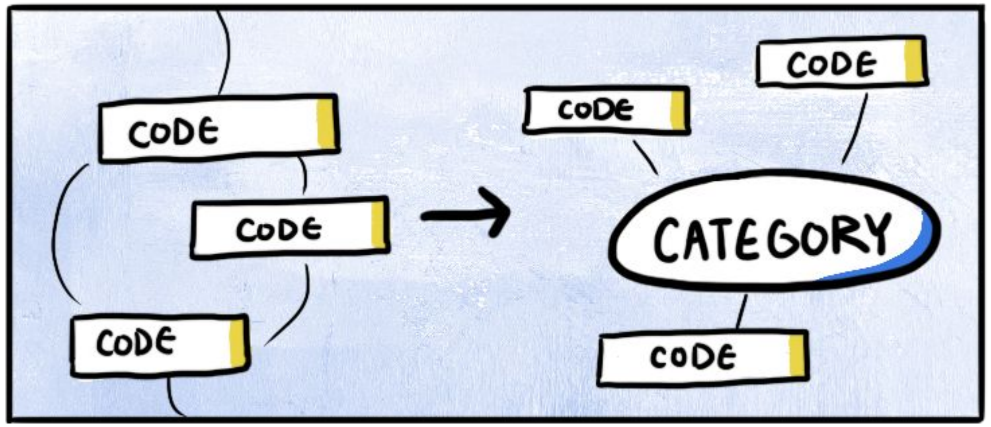
Open Coding

- ❑ Open coding is a common first step in the analysis of your qualitative research and is often used as the initial coding pass in Grounded Theory.
- ❑ You have started collecting qualitative data, such as transcriptions from interviews. With open coding, you break your data into discrete parts and create "codes" to label them.
- ❑ The purpose of breaking up your data and labeling them with codes is to enable you as the researcher to continuously compare and contrast similar events in your data
- ❑ This process forces you out of preconceived notions and biases about your own research.



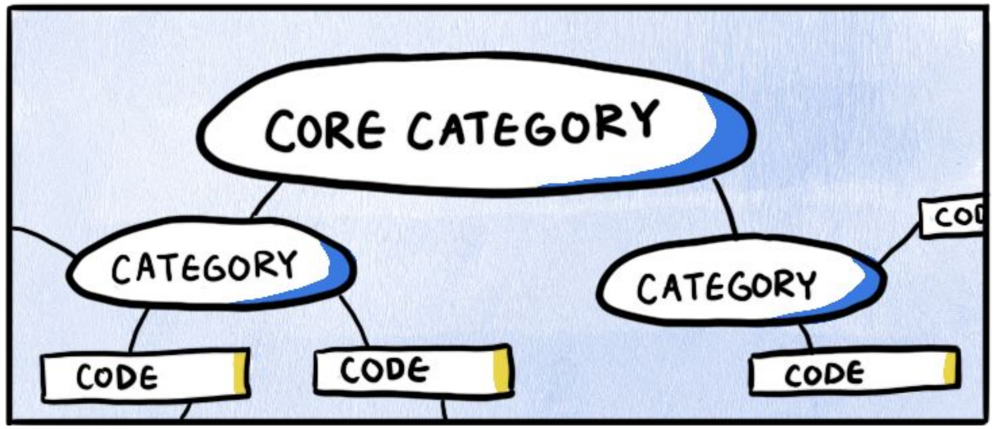
Axial Coding

- the second step of coding that follows open coding.
- In contrast to open coding where you break the data into discrete parts, with axial coding you begin to draw connections between codes.
- you read over your codes and the underlying data to find how your codes can be grouped into categories.
- A category could be created based on an existing code, or a new more abstract category can be developed that encompasses a number of different codes.



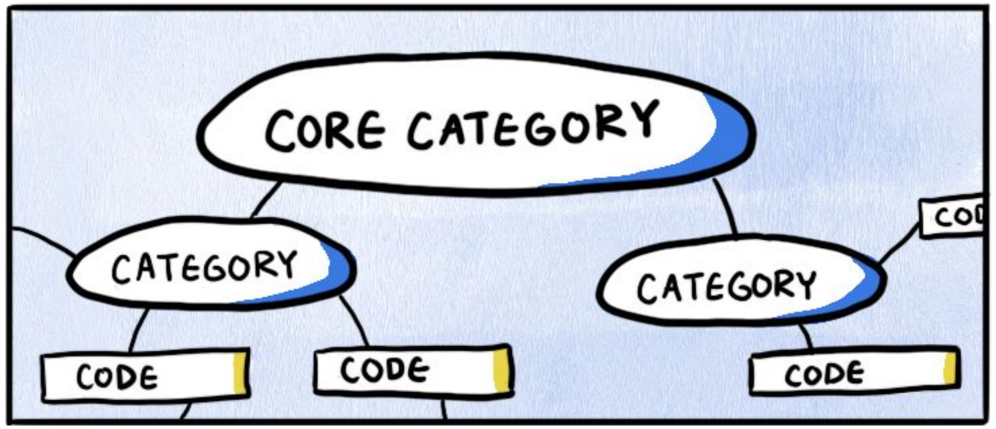
Selective Coding

- Selective coding is the last step in grounded theory , where you connect all your categories together around one core category.
- In doing so you define one unified theory around your research.
- Selective coding occurs later on in your research and connects categories you have developed from your qualitative data in previous coding cycles, such as axial coding.
- Selective coding is the culmination of the grounded theory process and its purpose is to either define a new theory or modify an existing theory based on your research.



When to use?

- You should consider these methods of coding when You want to follow a grounded theory method of qualitative analysis You want to derive new theories or concepts from your data You don't want preconceived theories to determine the outcome of your research You are conducting exploratory research where you want to generate new concepts and ideas



Steps

Open coding

1. **Turn your data into small, discrete components of data**
   Read through your qualitative data (such as transcripts from interviews) and analytically break it up into discrete, bite sized pieces of data.
2. **Code each discrete pieces of data with a descriptive label.**
   Interpret each piece of data and label it based off the properties of the data
   Ensure that any two pieces of data that relate to the same subject, should be labeled with the same codes

Axial coding

3. **Find connections and relationships between code**
   Now that you have a set of codes, identify connections between them. Look for causal conditions, context behind observations, and consequences of phenomena
4. **Aggregate and condense codes into broader categories**
   Determine broader categories that make connections between codes

Selective coding

5. **Bring it together with one overarching category**
   Select one core category that captures the essence of your research. It should be one big idea that captures a recurring trend in your data
6. **Identify the connections between this overarching category and the rest of your codes and data**
   How does your overarching category connect with the rest of your codes and data? Search for these connections in order to determine your final narrative
7. **Remove categories or codes that don’t have enough supporting data**
   Review all your other categories and codes and check if they have enough data to be robust. If they fall short, remove them
8. **Read the transcript again, and code according to this overarching category**
   Do another read of your transcripts and code according to this overarching category and code structure