Appendix
What Could Possibly Go Wrong: Undesirable Patterns in Collective Development

December, 2023

Appendix A
A.1 Undesirable Patterns and Tools

In this section, we present a comprehensive list of the 42 identified undesirable patterns. Each pattern is accompanied by a descriptive name, definition, a typology based on the pattern's origins, and, if possible, matching smells presented by Caballero-Espinoza [1], Dogan et al. [2], Jerzyk et al. [3]. We list technical patterns first, followed by social patterns, and conclude with organizational patterns. We place patterns that fit into multiple categories accordingly between their related groups.

Some patterns are also accompanied with a list of consequences that were reported by at least 75% of the "Undesirable Patterns Overview" survey participants. Patterns that were chosen for their related tool and feature idea evaluation are also accompanied by a list of these ideas and scores that each answer option received in each influence group. Answers options with the highest scores within each group are highlighted. For each idea and each group we provide the overall idea score which was calculated with the following formula:

$$\text{Idea score} = 2 \cdot [\text{ratio of votes for "Solves the problem completely"}] +$$
$$+ [\text{ratio of votes "Solves a part of the problem"}] + 0.5 \cdot [\text{ratio of votes "Helps to manage the problem a little"}]$$

Codes for tool evaluation answers:
- $+$: Solves the problem completely
- $\pm$: Solves a part of the problem
- $\mp$: Helps to manage the problem a little
- $-$: Not beneficial at all

Codes for influence groups:
(A)uthority: I can decide that our team will use this tool from now on, and it is unlikely to be overruled by someone else
(I)nfluencer: I cannot decide that our team will adopt this tool, but I can strongly influence a discussion about whether or not this tool should be used (e.g. I'm one of three people who will make this decision together)
(S)uggester: I can suggest using this tool, but I have no impact on whether it will be adopted or not
(O)bserver: I have no say in these matters

For each idea, we gather all the ratings provided by survey participants and calculate their arithmetic mean for different groups of practitioners based on their influence in adopting new tools. For example, group *AIS* is a union of *Authority*, *Influencer*, and *Suggester* groups. We consider opinions from individuals with greater influence to be more valuable from an industrial perspective.

(1) High level code style disagreement
Categories: Technical code-related
A code style disagreement between colleagues which affects software design (e.g., OOP vs. functional, abstraction level, but NOT tabs vs. spaces). The problem often comes up in code reviews.

(2) Duplicated work
Categories: Technical code-related
Jerzyk: Duplicate code
The same feature or similar piece of code was developed several times independently by different developers.

Frequently reported consequences

- *86%* Wasting time: Fixing or redoing

(3) Low level code style disagreement
Categories: Technical code-related
Code style disagreement between colleagues on cosmetic issues, such as tabs vs. spaces, line length, or naming conventions (not the high-level abstractions such as OOP vs functional style).

(4) Negligent code reviews
Categories: Technical code-related
Dogan: LGTM reviews
Code reviewers do not put enough effort into reading the code and making comments on it, which might lead to bugs, improper code design, or an overcomplicated codebase.

Frequently reported consequences

- *82%* Wasting time: Fixing or redoing

Tool and feature ideas

NCR-1 A tool that tracks how much time has been spent on code review and alerts all stakeholders if it's insufficient.

	Total votes	+ (%)	± (%)	∓ (%)	− (%)	**Idea score**
AISO	448	3	22	29	**46**	0.425
AIS	445	3	22	29	**46**	0.425
AI	324	4	22	30	**44**	0.45
A	94	3	27	32	**38**	0.49

NCR-2 A tool that forces reviewers to write at least X words or spend at least Y minutes on the review.

	Total votes	+ (%)	± (%)	∓ (%)	− (%)	**Idea score**
AISO	448	3	14	26	**57**	0.33
AIS	445	3	14	26	**57**	0.33
AI	324	3	15	25	**57**	0.335
A	94	4	17	29	**50**	0.395

(5) Uncoordinated code changes

Categories: Technical code-related

Caballero-Espinoza: Class cognition

Several people update the same code base simultaneously, without discussing the changes they made. This could unintentionally introduce conflicts. This may be particularly significant if the code base is large – Git rebase and editing will not work because of the high speed of new changes.

Tool and feature ideas

UCC-1 A tool that tracks the commit times of various developers and analyzes the frequency of commits and merges affecting a particular part of the codebase. The tool warns engineers if they try to commit to a frequently changed part of the codebase regardless of the particular branch. This notifies them to proceed with extra caution.

	Total votes	+ (%)	± (%)	∓ (%)	− (%)	**Idea score**
AISO	355	6	32	**44**	18	0.66
AIS	354	6	32	**44**	18	0.66
AI	243	6	34	**43**	17	0.675
A	67	9	36	**45**	10	0.765

UCC-2 A tool that allows users to favorite some specific files or methods inside their Git-hosting platform, and alerts them if those files or methods were changed by someone else.

	Total votes	+ (%)	± (%)	∓ (%)	− (%)	**Idea score**
AISO	355	8	33	**35**	24	0.665
AIS	354	8	33	**35**	24	0.665
AI	243	9	33	**35**	23	0.685
A	67	12	39	**36**	13	0.81

UCC-3 An IDE tool that allows for a superficial temporary merge of another branch into the current one, and views the code of both branches together as if in a merge conflict resolver.

	Total votes	+ (%)	± (%)	∓ (%)	− (%)	**Idea score**
AISO	355	12	**52**	27	9	0.895
AIS	354	12	**52**	27	9	0.895
AI	243	12	**52**	26	10	0.89
A	67	20	**52**	22	6	1.03

(6) Slow code review process
Categories: Technical code-related
Dogan: Sleeping reviews
Code review and pull request merging processes take too long due to slow responses from assignees.

Frequently reported consequences

- *93%* Wasting time: Waiting

Tool and feature ideas

SCRP-1 A feature that allows review requesters to set a timer that sends a notification to assignees if the review has not been completed in time.

	Total votes	+ (%)	± (%)	∓ (%)	− (%)	**Idea score**
AISO	420	7	**41**	38	14	0.74
AIS	417	7	**40**	38	15	0.73
AI	297	8	**40**	37	15	0.745
A	76	8	**41**	39	12	0.765

SCRP-2 A feature that provides an extension to the usual merge process. First, when the reviewer and the committer agree on all but a few minor details, like function naming, the code is "pre-merged". This means it has been merged into the codebase with some kind of warning. Then, after all of the issues are resolved, the warning is removed and the review is closed. This would allow the team to add working and tested features to the product faster.

	Total votes	+ (%)	± (%)	∓ (%)	− (%)	**Idea score**
AISO	420	5	**36**	33	26	0.625
AIS	417	5	**36**	33	26	0.625
AI	297	5	**35**	32	28	0.61
A	76	3	**39**	34	24	0.62

SCRP-3 An IDE extension that allows for code reviews in an IDE. The reviewer checks out someone else's branch and opens the pending code review on that branch. The reviewer then makes notes and comments inside the IDE.

	Total votes	+ (%)	± (%)	∓ (%)	− (%)	**Idea score**
AISO	420	10	**54**	25	11	0.865
AIS	417	10	**54**	25	11	0.865
AI	297	10	**53**	25	12	0.855
A	76	11	**63**	17	9	0.935

(7) Breaking change
Categories: Technical code-related
An engineer tries to change code to meet their needs but inadvertently breaks some functionality used by other team members.

Frequently reported consequences

- *91%* Wasting time: Fixing or redoing

(8) Outdated tests

Categories: Technical code-related

Improper test maintenance. This happens when tests are not updated, extended, or removed after the code base update.

Tool and feature ideas

OT-1 An IDE feature that links the tests to the code and alerts the committer if the code was altered, but the tests were not updated, like when a method functionality was extended but the test was not updated to check the new functionality.

	Total votes	+ (%)	± (%)	∓ (%)	− (%)	**Idea score**
AISO	411	21	**51**	25	3	1.055
AIS	408	21	**51**	25	3	1.055
AI	276	21	**53**	24	2	1.07
A	77	18	**53**	24	5	1.01

OT-2 A tool that stores the testing results and checks whether the test statuses have changed after a new commit. It then produces a report that shows which of the tests started failing or working after the commit.

	Total votes	+ (%)	± (%)	∓ (%)	− (%)	**Idea score**
AISO	411	15	**51**	26	8	0.94
AIS	408	15	**51**	26	8	0.94
AI	276	13	**50**	29	8	0.905
A	77	16	**45**	32	7	0.93

(9) Tool incompatibility

Categories: Technical tool-related

Team members use different tools for the same goals. These tools might not work well together, causing workflow delay as engineers need to resolve compatibility issues. This problem may affect both system components (such as libraries) and auxiliary tools (such as IDEs).

(10) Multiple discussion channels
Categories: Technical tool-related, Technical information-related
Caballero-Espinoza: Black cloud
Information is spread across several discussion channels used by team members (messengers, issue and task trackers, email, meetings, etc.), so it can be difficult for team members to understand exactly what is going on.

Frequently reported consequences

- *89%* Wasting time: Communication

Tool and feature ideas

MDC-1 A tool that checks whether several meeting participants are assigned to the same ticket whose description or name matches the meeting's name. If this is true, the tool assumes that this ticket is relevant to the meeting and sends the meeting participants a suggestion to update the ticket if necessary.

	Total votes	+ (%)	± (%)	∓ (%)	− (%)	**Idea score**
AISO	394	3	28	**41**	28	0.545
AIS	391	3	28	**41**	28	0.545
AI	277	3	29	**39**	29	0.545
A	90	3	26	**42**	29	0.53

MDC-2 A feature that allows for the linking of chats and threads in a messenger, such as Slack, with specific issues in the issue tracker (e.g. a thread in Slack is linked to an issue on Jira).

	Total votes	+ (%)	± (%)	∓ (%)	− (%)	**Idea score**
AISO	394	8	**48**	35	9	0.815
AIS	391	8	**48**	35	9	0.815
AI	277	9	**49**	34	8	0.84
A	90	8	**50**	31	11	0.815

(11) Insufficient documentation
Categories: Technical information-related
Caballero-Espinoza: Invisible architecting
The existing documentation is not sufficient to understand the project because it is incomplete, badly written, very hard to understand, or was not updated for a long time.

Frequently reported consequences

- *84%* Wasting time: Communication

Tool and feature ideas

ID-1 A tool that gives you the ability to cross-link code, issues, chats, and documentation (e.g. a feature in Slack which enables linking of threads to a specific issue on Jira or YouTrack). The tool then extracts the topic from the ticket name and generates documentation by collecting data from all of the cross-links. Engineers can then update the documentation accordingly.

	Total votes	+ (%)	± (%)	∓ (%)	− (%)	**Idea score**
AISO	476	5	**46**	38	11	0.75
AIS	470	5	**46**	39	10	0.755
AI	330	6	**47**	37	10	0.775
A	83	5	**48**	37	10	0.765

ID-2 A tool that sends code authors alerts when they commit a new piece of code and forget to create documentation for it. It also sends notifications to code owners if the corresponding piece of documentation was last updated long before the last code update.

	Total votes	+ (%)	± (%)	∓ (%)	− (%)	**Idea score**
AISO	476	7	**47**	36	10	0.79
AIS	470	7	**47**	36	10	0.79
AI	330	8	**45**	37	10	0.795
A	83	6	**52**	34	8	0.81

ID-3 A feature that allows for the creation of documentation right in your IDE. You select a piece of code, press a button, and a new window will appear, in which you can write documentation for this code snippet. This documentation will be uploaded to the team's storage and automatically linked with the commit.

	Total votes	+ (%)	± (%)	∓ (%)	− (%)	**Idea score**
AISO	476	12	**53**	26	9	0.9
AIS	470	12	**53**	26	9	0.9
AI	330	12	**54**	24	10	0.9
A	83	16	**55**	23	6	0.985

(12) Improper confidential data management
Categories: Technical information-related
Caballero-Espinoza: Informality excess
Publishing passwords and private keys (e.g. by checking them in the repository).

(13) Mutable data sharing
Categories: Technical information-related
Jerzyk: Mutable data
Many team members use the same mutable data (e.g. the same database). So, when one person modifies the data, the other team members may be unaware of the change and, when they use it, they will assume the data was not changed.

Frequently reported consequences

- *77%* Wasting time: Fixing or redoing

(14) Faulty information sharing
Categories: Technical information-related, Social communication
Caballero-Espinoza: Sharing villainy
Colleagues not being able to answer questions clearly, with full detail or quickly enough. For example, when your colleague takes several days to answer a short question, or when they only give a short reply which is hard to understand without knowing all the related context.

Frequently reported consequences

- *77%* Wasting time: Waiting
- *75%* Wasting time: Communication

(15) Imperfect stakeholder communication
Categories: Social communication
Stakeholders do not provide clear and complete data because either they cannot, do not want to, or do not know how to provide such data.

(16) Long communication chain
Categories: Social communication
Caballero-Espinoza: Leftover techie
Teams and/or stakeholders involved in a project do not communicate directly, so things deteriorate into a "broken telephone" game. For example, a customer communicates with the development team via sales managers and business analysts.

Frequently reported consequences

- *82%* Wasting time: Communication

(17) Hard to share task context
Categories: Social communication
Caballero-Espinoza: Architecture by osmosis
Dogan: Missing context
Sometimes information cannot be easily transferred to another person. It might cause problems when the primary task assignee is planning to take time off work and needs to share all of their task related knowledge and data with other team members.

(18) Goal misunderstanding
Categories: Social communication
Caballero-Espinoza: Disengagement
Team members fail to understand the aim of a task, or a project, or a product, so the solutions do not align with the main goals. For example, a team working on a product prototype does not create a minimal viable prototype, as expected, but instead works on a highly-functional scaling solution with a lot of features.

Frequently reported consequences

- *79%* Project development impact: Delayed

(19) Lack of informal communication
Categories: Social communication
Team members have no means of communicating informally. As a result, the team relationships become too official and engineers are not integrated into the wider company.

(20) Passive engineer
Categories: Social varience
A team member does not do anything unless they are specifically told to.

Frequently reported consequences

- *83%* Wasting time: Waiting

(21) Cultural differences
Categories: Social varience
Caballero-Espinoza: Cognitive distance
Diversity of the team members' cultural backgrounds or languages which causes unintentional misunderstanding or miscommunication between team members.

(22) Dissensus of opinions and approaches
Categories: Social varience
Caballero-Espinoza: Dissensus, Organizational skirmish, Solution defiance
The team cannot agree on how to resolve or approach a non-technical problem due to a difference of opinions or approaches. For example, Alex wants to have three default reaction buttons for posts but Betty thinks it is better to have five buttons.

(23) Technical dissensus
Categories: Social varience
Caballero-Espinoza: Dissensus
Team members prefer different technical approaches and argue over which tool or language or framework to choose.

(24) No task progress report
Categories: Social reluctance
An engineer keeps working on a task without sharing their progress and issues with team members or managers. This might lead to bugs or incorrect task execution due to a misunderstanding on the engineer's part which, in turn, is also not communicated to their colleagues.

Frequently reported consequences

- *78%* Wasting time: Fixing or redoing

Tool and feature ideas

NTPR-1 A tool that allows you to create a short description of your current task, so that your colleagues are aware of what you are working on. Every day, users flag tasks they are working on, close them once completed, and then create new ones. The tool's simple notepad-like design allows you to quickly update everyone on the status of your tasks, and see how far your colleagues are progressing on your team's project.

	Total votes	+ (%)	± (%)	∓ (%)	− (%)	**Idea score**
AISO	435	12	30	**39**	19	0.735
AIS	433	12	30	**39**	19	0.735
AI	300	13	30	**38**	19	0.75
A	62	15	32	**40**	13	0.82

(25) Ignoring directions
Categories: Social reluctance
Caballero-Espinoza: Lone wolf
An engineer ignores the tasks or the directions given by the managers or technical leads. This might lead to bugs, improper software design, or incorrect task execution, so the end result does not fit the requirements.

Frequently reported consequences

- *82%* Wasting time: Fixing or redoing

(26) No acceptance criteria
Categories: Organizational planning
There are no clear and well-described acceptance criteria (e.g. a task is formulated as "We need to speed up the loading process a bit"), making it impossible to understand what should be the end result of the task.

(27) Insufficient planning
Categories: Organizational planning
Caballero-Espinoza: Dispersion, Time warp
The team does not plan projects in enough detail. For example, a frontend and a backend team do not plan out the API properly and design a product with incompatible parts.

Frequently reported consequences

- *91%* Wasting time: Fixing or redoing

(28) Unclear requirements
Categories: Organizational planning
No clear, well-thought-out, and formally defined issue requirements are given, so people interpret requirements individually.

Frequently reported consequences

- *81%* Wasting time: Fixing or redoing
- *79%* Wasting time: Communication
- *77%* Team effects: Frustration and low morale
- *75%* Project development impact: Delayed

Tool and feature ideas

UR-1 A feature that analyzes created tickets. The ticket should not be empty or too complex (e.g. long compound sentences). Its name should also match the content.

	Total votes	+ (%)	± (%)	∓ (%)	− (%)	**Idea score**
AISO	424	4	36	**42**	18	0.65
AIS	418	4	36	**42**	18	0.65
AI	297	3	35	**45**	17	0.635
A	77	4	38	**41**	17	0.665

UR-2 A tool that structures the writing of tickets. A particular implementation can be a chatbot which prompts users with specific questions and forms to complete (e.g. "What is the goal?" and "When is the deadline?"). It then produces a ticket with all of the submitted information.

	Total votes	+ (%)	± (%)	∓ (%)	− (%)	**Idea score**
AISO	424	7	**47**	36	10	0.79
AIS	418	7	**47**	36	10	0.79
AI	297	6	**48**	35	11	0.775
A	77	6	**47**	38	9	0.78

(29) Problem complexity mismanagement
Categories: Organizational planning
A complicated problem has not been broken down into smaller, more manageable chunks for a developer to work on.

(30) Disruption of plans
Categories: Organizational planning, Organizational management
Work priorities change unexpectedly, so engineers are often being forced to work on something unplanned.

(31) Poor onboarding process
Categories: Organizational management
Caballero-Espinoza: Newbie freeriding
New engineers do not get enough work-related information to start working efficiently. Often, the onboarding process contains only general information about office hours, benefits, vacation, etc. However, engineers learn nothing about the team's code style or code review rules and do not get introduced to the current projects.

Frequently reported consequences

- *82%* Wasting time: Communication

Tool and feature ideas

POP-1 A tool to keep all of the answers to frequent onboarding-related issues in one place. This would be similar to a separate Stack Overflow [4] workspace for the team or company.

	Total votes	+ (%)	± (%)	∓ (%)	− (%)	**Idea score**
AISO	411	9	**48**	33	10	0.825
AIS	409	9	**48**	34	9	0.83
AI	290	9	**46**	34	11	0.81
A	77	9	**44**	39	8	0.815

POP-2 A tool that tracks newcomer commit frequency over the first 6 months of work. This data can be used to analyze new engineers' development pace, and thus deduce the time spent on their onboarding.

	Total votes	+ (%)	± (%)	∓ (%)	− (%)	**Idea score**
AISO	411	3	26	32	**39**	0.48
AIS	409	3	26	32	**39**	0.48
AI	290	4	26	32	**38**	0.50
A	77	2	30	**38**	30	0.53

(32) Not enough human resources
Categories: Organizational management
The team's workload is too high, so the team has to either work overtime or delay task implementation.

(33) Inadequate leadership
Categories: Organizational management
The group lead does not follow the best management practices (does not listen to team members, gives unethical feedback, constantly makes significant planning errors, etc.).

Frequently reported consequences

- *75%* Wasting time: Fixing or redoing
- *75%* Wasting time: Communication
- *75%* Team effects: Frustration and low morale

(34) Toxic company culture
Categories: Organizational management
Company culture makes team members uncomfortable, e.g. due to microaggression, bullying, or general animosity.

Frequently reported consequences

- *91%* Team effects: Frustration and low morale
- *86%* Company effects: Toxic environment

(35) Underqualified colleagues
Categories: Organizational management
Caballero-Espinoza: Obfuscated architecting
Having team members who are unable to properly solve a task or effectively communicate a problem due to lack of experience or knowledge.

Frequently reported consequences

- *79%* Wasting time: Fixing or redoing

(36) Informational bottleneck
Categories: Organizational management
Caballero-Espinoza: Code red
Bus/truck factor issue. This happens when too few people know what is going on in some part of a project. This might lead to a project stalling when those with the most knowledge become unavailable, because progress cannot be made on the relevant part of the project.

Frequently reported consequences

- *83%* Wasting time: Waiting

(37) Insufficient financial resources
Categories: Organizational management
A shortage in the team budget preventing the team from buying proper equipment, taking required business trips or covering other necessary work-related expenses.

(38) Unreasonable tasks
Categories: Organizational management
The team receives tasks that are impossible to finish in the permitted time, tasks that contradict each other, or tasks that are otherwise unreasonable.

Frequently reported consequences

- *75%* Project development impact: Delayed

(39) Synchronizing between teams
Categories: Organizational synchronization
Caballero-Espinoza: Architecture hood, DevOps clash, Organizational silo
Different teams work on similar issues at the same time but cannot effectively synchronize their actions (meetings, statuses, rolling out into production, sharing information).

(40) Teamwork pipeline bottleneck
Categories: Organizational synchronization
A group of people cannot work efficiently together, because they cannot proceed without the results created by another group of people (e.g. frontend team cannot develop without a part of the API being implemented by the backend team first). The issue is not that one of the teams is too slow, but that the project development plan did not account for such bottlenecks.

Frequently reported consequences

- *91%* Wasting time: Waiting

(41) Mismatching working hours
Categories: Organizational synchronization
Different team members have different working hours and find it hard to collaborate due to little overlap in their working hours.

Frequently reported consequences

- *89%* Wasting time: Waiting

(42) Inefficient meetings system
Categories: Organizational synchronization
Caballero-Espinoza: Black cloud, Sharing villainy
Unreasonable meeting schedule (meetings are too long, too frequent, or too rare) which does not suit project needs.

Frequently reported consequences

- *79%* Wasting time: Communication

Tool and feature ideas

IMS-1 A bot that sends notifications X minutes before the meeting ends. The exact number is customizable by the meeting creator and the notifications are similar to those prior to the meeting.

	Total votes	+ (%)	± (%)	∓ (%)	− (%)	**Idea score**
AISO	401	3	21	**40**	36	0.47
AIS	397	3	21	**40**	36	0.47
AI	284	3	21	**39**	37	0.465
A	69	4	22	**38**	36	0.49

IMS-2 A tool that analyzes the data for the meeting's duration. It creates diagrams for employees regarding how much time they spend on meetings for each project and notifies the users if they spend a disproportionate amount of time on meetings for certain projects compared to others.

	Total votes	+ (%)	± (%)	∓ (%)	− (%)	**Idea score**
AISO	401	6	36	**39**	19	0.675
AIS	397	6	36	**38**	20	0.67
AI	284	6	36	**39**	19	0.675
A	69	6	37	**35**	22	0.665

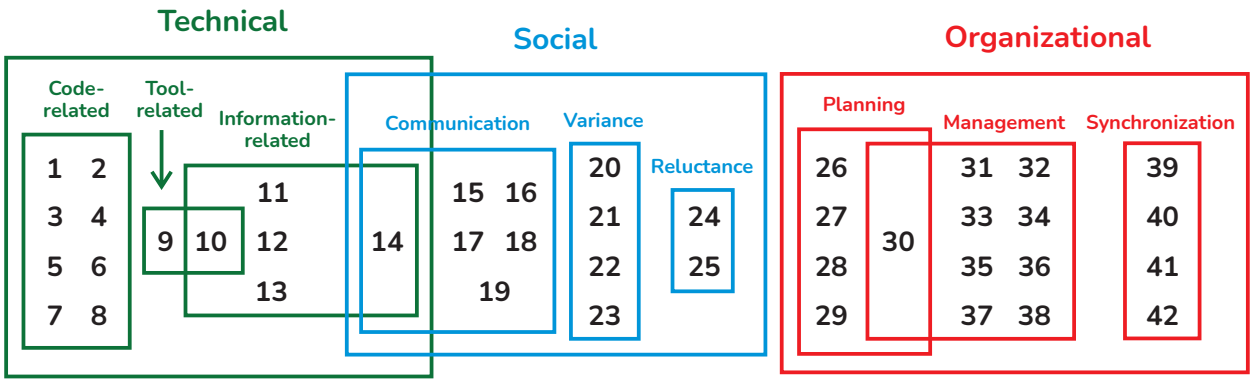

Figure 1: Overview of undesirable patterns categorized by their origins.

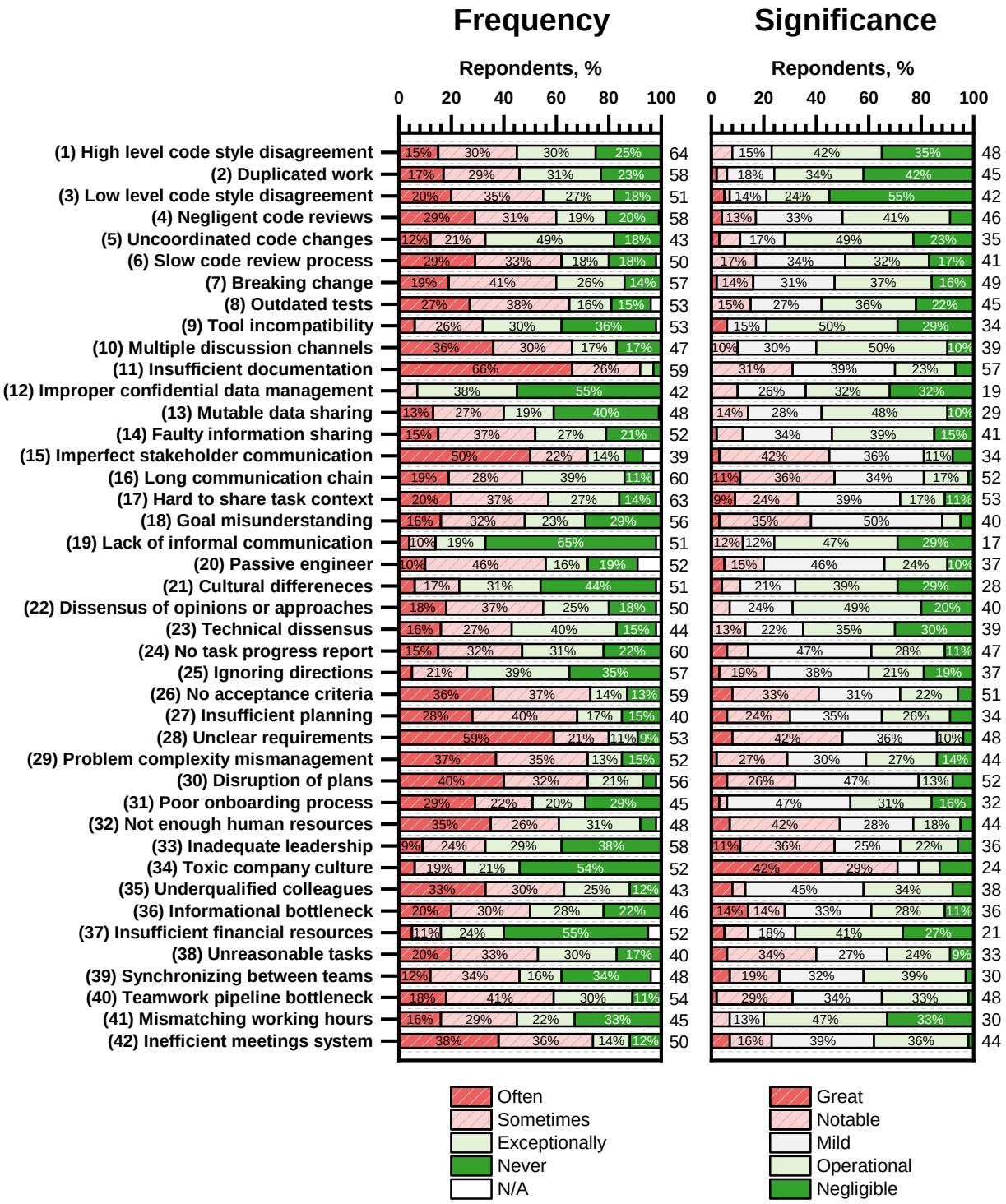

Figure 2: Frequency and significance of undesirable patterns reported in the "Undesirable Patterns Overview" survey. Numbers next to the coloured grid show the total number of votes.

A.2 Reported Frequency

This table presents percentages of the "Undesirable Patterns Overview" survey participants who voted for each frequency answer option of each pattern. The highest and lowest scores for each frequency option (except for *N/A*) are highlighted. The minimum and maximum of *Total votes* are also highlighted.

Pattern	Frequency (%)					Total
name	Often	Sometimes	Exceptionally	Never	N/A	votes
(1) High level code style disagreement	15	30	30	25	0	**64**
(2) Duplicated work	17	29	31	23	0	58
(3) Low level code style disagreement	20	35	27	18	0	51
(4) Negligent code reviews	29	31	19	20	1	58
(5) Uncoordinated code changes code	12	21	**49**	18	0	43
(6) Slow code review process	29	33	18	18	2	50
(7) Breaking change	19	41	26	14	0	57
(8) Outdated tests	27	38	16	15	4	53
(9) Tool incompatibility	6	26	30	36	2	53
(10) Multiple discussion channels	36	30	17	17	0	47
(11) Insufficient documentation	**66**	26	**5**	**3**	0	59
(12) Improper confidential data management	**0**	**7**	38	55	0	42
(13) Mutable data sharing	13	27	19	40	1	48
(14) Faulty information sharing	15	37	27	21	0	52
(15) Imperfect stakeholder communication	50	22	14	7	7	**39**
(16) Long communication chain	19	28	39	11	3	60
(17) Hard to share task context	20	37	27	14	2	63
(18) Goal misunderstanding	16	32	23	29	0	56
(19) Lack of informal communication	4	10	19	**65**	2	51
(20) Passive engineer	10	**46**	16	19	9	52
(21) Cultural differences	6	17	31	44	2	51
(22) Dissensus of opinions and approaches	18	37	25	18	2	50
(23) Technical dissensus	16	27	40	15	2	44
(24) No task progress report	15	32	31	22	0	60
(25) Ignoring directions	5	21	39	35	0	57
(26) No acceptance criteria	36	37	14	13	0	59
(27) Insufficient planning	28	40	17	15	0	40
(28) Unclear requirements	59	21	11	9	0	53
(29) Problem complexity mismanagement	37	35	13	15	0	52
(30) Disruption of plans	40	32	21	5	2	56
(31) Poor onboarding process	29	22	20	29	0	45
(32) Not enough human resources	35	26	31	6	2	48
(33) Inadequate leadership	9	24	29	38	0	58
(34) Toxic company culture	6	19	21	54	0	52
(35) Underqualified colleagues	33	30	25	12	0	43
(36) Informational bottleneck	20	30	28	22	0	46
(37) Insufficient financial resources	5	11	24	55	5	52
(38) Unreasonable tasks	20	33	30	17	0	40
(39) Synchronizing between teams	12	34	16	34	4	48
(40) Teamwork pipeline bottleneck	18	41	30	11	0	54
(41) Mismatching working hours	16	29	22	33	0	45
(42) Inefficient meetings system	38	36	14	12	0	50

A.3 Reported Significance

This table presents percentages of the "Undesirable Patterns Overview" survey participants who voted for each significance answer option of each pattern. The highest and lowest scores for each significance option are highlighted. The minimum and maximum of *Total votes* are also highlighted.

Pattern name	Significance (%)					Total votes
	Great	Notable	Mild	Operational	Negligible	
(1) High level code style disagreement	**0**	8	15	42	35	48
(2) Duplicated work	2	4	18	34	42	45
(3) Low level code style disagreement	5	2	14	24	**55**	42
(4) Negligent code reviews	4	13	33	41	9	46
(5) Uncoordinated code changes	3	8	17	49	23	35
(6) Slow code review process	**0**	17	34	32	17	41
(7) Breaking change	2	14	31	37	16	49
(8) Outdated tests	**0**	15	27	36	22	45
(9) Tool incompatibility	6	**0**	15	**50**	29	34
(10) Multiple discussion channels	**0**	10	30	**50**	10	39
(11) Insufficient documentation	**0**	31	39	23	7	**57**
(12) Improper confidential data management	**0**	10	26	32	32	19
(13) Mutable data sharing	**0**	14	28	48	10	29
(14) Faulty information sharing	2	10	34	39	15	41
(15) Imperfect stakeholder communication	3	**42**	36	11	8	34
(16) Long communication chain	11	36	34	17	**2**	52
(17) Hard to share task context	9	24	39	17	11	53
(18) Goal misunderstanding	3	35	**50**	**7**	5	40
(19) Lack of informal communication	**0**	12	12	47	29	**17**
(20) Passive engineer	5	15	46	24	10	37
(21) Cultural differences	4	7	21	39	29	28
(22) Dissensus of opinions and approaches	**0**	7	24	49	20	40
(23) Technical dissensus	**0**	13	22	35	30	37
(24) No task progress report	6	8	47	28	11	47
(25) Ignoring directions	3	19	38	21	19	37
(26) No acceptance criteria	8	33	31	22	6	51
(27) Insufficient planning	6	24	35	26	9	34
(28) Unclear requirements	8	**42**	36	10	4	48
(29) Problem complexity mismanagement	2	27	30	27	14	44
(30) Disruption of plans	6	26	47	13	8	52
(31) Poor onboarding process	3	3	47	31	16	32
(32) Not enough human resources	7	**42**	28	18	5	44
(33) Inadequate leadership	11	36	25	22	6	36
(34) Toxic company culture	**42**	29	**8**	8	13	24
(35) Underqualified colleagues	8	5	45	34	8	38
(36) Informational bottleneck	14	14	33	28	11	36
(37) Insufficient financial resources	5	9	18	41	27	21
(38) Unreasonable tasks	6	34	27	24	9	33
(39) Synchronizing between teams	7	19	32	39	3	30
(40) Teamwork pipeline bottleneck	2	29	34	33	**2**	48
(41) Mismatching working hours	**0**	7	13	47	33	30
(42) Inefficient meetings system	7	16	39	36	**2**	44

A.4 Reported Wasting Time Consequences

This table presents percentages of the "Undesirable Patterns Overview" survey participants who voted for each significance answer option of each pattern. The highest scores for each significance option are highlighted.

Pattern name	Wasting time consequences (%)			Total votes
	Waiting	Fixing or redoing	Communication	
(1) High level code style disagreement	32	68	54	48
(2) Duplicated work	23	**86**	34	45
(3) Low level code style disagreement	32	71	57	42
(4) Negligent code reviews	56	**82**	51	46
(5) Uncoordinated code changes	35	74	52	35
(6) Slow code review process	**93**	33	48	41
(7) Breaking change	47	91	37	49
(8) Outdated tests	33	74	38	45
(9) Tool incompatibility	46	58	46	34
(10) Multiple discussion channels	49	43	**89**	39
(11) Insufficient documentation	58	55	**84**	57
(12) Improper confidential data management	50	57	36	19
(13) Mutable data sharing	64	**77**	59	29
(14) Faulty information sharing	**77**	43	**75**	41
(15) Imperfect stakeholder communication	71	63	66	34
(16) Long communication chain	63	65	**82**	52
(17) Hard to share task context	70	48	66	53
(18) Goal misunderstanding	41	72	64	40
(19) Lack of informal communication	47	53	53	17
(20) Passive engineer	**83**	38	67	37
(21) Cultural differences	73	68	73	28
(22) Dissensus of opinions and approaches	43	69	69	40
(23) Technical dissensus	24	44	60	37
(24) No task progress report	60	**78**	51	47
(25) Ignoring directions	50	**82**	59	37
(26) No acceptance criteria	56	65	69	51
(27) Insufficient planning	55	**91**	46	34
(28) Unclear requirements	66	**81**	**79**	48
(29) Problem complexity mismanagement	55	61	47	44
(30) Disruption of plans	43	60	55	52
(31) Poor onboarding process	39	57	**82**	32
(32) Not enough human resources	43	43	43	44
(33) Inadequate leadership	56	**75**	**75**	36
(34) Toxic company culture	41	46	64	24
(35) Underqualified colleagues	71	**79**	50	38
(36) Informational bottleneck	**83**	42	56	36
(37) Insufficient financial resources	50	17	44	21
(38) Unreasonable tasks	64	55	64	33
(39) Synchronizing between teams	67	63	50	30
(40) Teamwork pipeline bottleneck	**91**	38	42	48
(41) Mismatching working hours	**89**	15	30	30
(42) Inefficient meetings system	42	28	**79**	44

A.5 Reported Project Development Impact Consequences

This table presents percentages of the "Undesirable Patterns Overview" survey participants who voted for each significance answer option of each pattern. The highest scores for each significance option are highlighted.

Pattern	Project development impact consequences (%)				Total
name	Cancelled	Insupportable	Delayed	Suboptimal	votes
(1) High level code style disagreement	0	7	17	15	48
(2) Duplicated work	3	11	23	31	45
(3) Low level code style disagreement	0	18	25	21	42
(4) Negligent code reviews	2	22	36	67	46
(5) Uncoordinated code changes	3	0	42	26	35
(6) Slow code review process	0	3	48	28	41
(7) Breaking change	2	9	40	44	49
(8) Outdated tests	0	15	59	51	45
(9) Tool incompatibility	0	8	23	11	34
(10) Multiple discussion channels	0	8	27	24	39
(11) Insufficient documentation	4	16	53	44	57
(12) Improper confidential data management	0	7	29	7	19
(13) Mutable data sharing	0	5	32	32	29
(14) Faulty information sharing	3	3	33	30	41
(15) Imperfect stakeholder communication	3	9	69	49	34
(16) Long communication chain	12	12	69	51	52
(17) Hard to share task context	2	8	72	44	53
(18) Goal misunderstanding	8	15	**79**	49	40
(19) Lack of informal communication	0	13	47	27	17
(20) Passive engineer	3	8	43	38	37
(21) Cultural differences	0	9	45	27	28
(22) Dissensus of opinions and approaches	0	0	37	26	40
(23) Technical dissensus	0	20	20	32	37
(24) No task progress report	7	9	51	31	47
(25) Ignoring directions	3	12	47	50	37
(26) No acceptance criteria	4	17	67	58	51
(27) Insufficient planning	6	12	61	52	34
(28) Unclear requirements	9	9	**75**	60	48
(29) Problem complexity mismanagement	0	8	61	39	44
(30) Disruption of plans	15	13	74	51	52
(31) Poor onboarding process	0	11	21	29	32
(32) Not enough human resources	21	36	59	57	44
(33) Inadequate leadership	14	19	67	56	36
(34) Toxic company culture	23	23	46	41	24
(35) Underqualified colleagues	6	12	62	47	38
(36) Informational bottleneck	3	6	64	33	36
(37) Insufficient financial resources	6	11	50	17	21
(38) Unreasonable tasks	11	14	**75**	54	33
(39) Synchronizing between teams	0	3	57	47	30
(40) Teamwork pipeline bottleneck	2	11	67	33	48
(41) Mismatching working hours	0	0	33	19	30
(42) Inefficient meetings system	2	9	47	37	44

A.6 Reported Team Effects Consequences

This table presents percentages of the "Undesirable Patterns Overview" survey participants who voted for each significance answer option of each pattern. The highest scores for each significance option are highlighted.

Pattern	Team effects consequences (%)			Total
name	Departure	Conflicts	Frustration	votes
(1) High level code style disagreement	10	32	32	48
(2) Duplicated work	9	17	31	45
(3) Low level code style disagreement	11	32	46	42
(4) Negligent code reviews	9	37	42	46
(5) Uncoordinated code changes	6	23	26	35
(6) Slow code review process	3	18	48	41
(7) Breaking change	2	19	26	49
(8) Outdated tests	13	21	33	45
(9) Tool incompatibility	4	35	38	34
(10) Multiple discussion channels	0	41	51	39
(11) Insufficient documentation	9	11	56	57
(12) Improper confidential data management	7	50	29	19
(13) Mutable data sharing	0	3	36	29
(14) Faulty information sharing	7	20	45	41
(15) Imperfect stakeholder communication	3	11	60	34
(16) Long communication chain	18	35	65	52
(17) Hard to share task context	18	16	46	53
(18) Goal misunderstanding	18	31	69	40
(19) Lack of informal communication	13	40	47	17
(20) Passive engineer	10	40	57	37
(21) Cultural differences	23	32	55	28
(22) Dissensus of opinions and approaches	9	43	43	40
(23) Technical dissensus	28	60	52	37
(24) No task progress report	7	47	47	47
(25) Ignoring directions	21	38	65	37
(26) No acceptance criteria	15	25	71	51
(27) Insufficient planning	9	36	46	34
(28) Unclear requirements	19	36	**77**	48
(29) Problem complexity mismanagement	3	13	55	44
(30) Disruption of plans	13	36	72	52
(31) Poor onboarding process	7	11	61	32
(32) Not enough human resources	39	25	66	44
(33) Inadequate leadership	33	50	**75**	36
(34) Toxic company culture	73	73	**91**	24
(35) Underqualified colleagues	15	26	53	38
(36) Informational bottleneck	11	17	56	36
(37) Insufficient financial resources	28	11	67	21
(38) Unreasonable tasks	29	39	71	33
(39) Synchronizing between teams	7	37	50	30
(40) Teamwork pipeline bottleneck	9	27	44	48
(41) Mismatching working hours	4	15	22	30
(42) Inefficient meetings system	21	21	70	44

A.7 Reported Company Effects Consequences

This table presents percentages of the "Undesirable Patterns Overview" survey participants who voted for each significance answer option of each pattern. The highest scores for each significance option are highlighted.

Pattern	Company effects consequences (%)		Total
name	Damaged reputation	Toxic environment	votes
(1) High level code style disagreement	0	10	48
(2) Duplicated work	9	20	45
(3) Low level code style disagreement	4	14	42
(4) Negligent code reviews	22	11	46
(5) Uncoordinated code changes	3	16	35
(6) Slow code review process	10	8	41
(7) Breaking change	16	12	49
(8) Outdated tests	21	12	45
(9) Tool incompatibility	8	11	34
(10) Multiple discussion channels	8	16	39
(11) Insufficient documentation	13	13	57
(12) Improper confidential data management	12	14	19
(13) Mutable data sharing	5	14	29
(14) Faulty information sharing	5	18	41
(15) Imperfect stakeholder communication	20	17	34
(16) Long communication chain	29	29	52
(17) Hard to share task context	18	18	53
(18) Goal misunderstanding	21	21	40
(19) Lack of informal communication	0	13	17
(20) Passive engineer	13	28	37
(21) Cultural differences	9	18	28
(22) Dissensus of opinions and approaches	6	17	40
(23) Technical dissensus	12	24	37
(24) No task progress report	5	20	47
(25) Ignoring directions	21	21	37
(26) No acceptance criteria	21	21	51
(27) Insufficient planning	24	15	34
(28) Unclear requirements	26	26	48
(29) Problem complexity mismanagement	13	13	44
(30) Disruption of plans	13	11	52
(31) Poor onboarding process	7	7	32
(32) Not enough human resources	21	36	44
(33) Inadequate leadership	25	39	36
(34) Toxic company culture	27	**86**	24
(35) Underqualified colleagues	6	15	38
(36) Informational bottleneck	11	14	36
(37) Insufficient financial resources	33	22	21
(38) Unreasonable tasks	14	32	33
(39) Synchronizing between teams	13	27	30
(40) Teamwork pipeline bottleneck	13	16	48
(41) Mismatching working hours	11	7	30
(42) Inefficient meetings system	12	16	44

Appendix B
B.1 Interview qualification survey

We are conducting a study on undesirable patterns in collective software development that hinder IT projects' progress. Please fill out this short 2-minute questionnaire to let us know about your experience with such matters. If your profile is a good match for this study, we will invite you to participate in an interview. The interview will be conducted remotely via Google Meet and will take around 90 minutes. Please note that the interview will be conducted in English.

As a thank you, everyone who completes the interview will be able to choose one of the following incentives: a $100 Amazon eGift Card or a 1-year license for the JetBrains All Products Pack, which includes all of the JetBrains IDEs.

Survey Terms and Conditions *
Interview Terms and Conditions *

What is your employment status?*
[Single-choice question]

- Fully employed by a company / organization
- Partially employed by a company / organization
- Self-employed (earning income directly from your own business, trade, or profession)
- Freelancer (pursuing a profession without a long-term commitment to any one employer)
- Working Student
- Student
- Retired
- Other, please specify:

Which of the following best describes your job role(s) regardless of your position level?*
[Multiple-choice question]

- Team Lead *(!)*
- DevOps Engineer, Infrastructure Developer, etc. *(!)*
- Developer Advocate
- DBA
- Instructor, Teacher, Tutor, etc.
- Product or Project Manager *(!)*
- Developer, Programmer, or Software Engineer *(!)*
- Data Analyst, Data Engineer, or Data Scientist *(!)*
- CIO, CEO, or CTO *(!)*
- Systems Analyst *(!)*
- Architect *(!)*
- UX/UI Designer
- Technical Support
- Tester or QA Engineer *(!)*
- Technical Writer
- Other, please specify:

[If no (!) position has been checked - END survey]

How many years of professional IT work experience do you have?*
[Single-choice question]

- Less than 1
- 1–2
- 3–5
- 6–10
- More than 10
- I don't have any work experience

What kind of development does your company do?*
[Multiple-choice question]

- Product development
- Outsourcing
- Custom-tailored software / hardware
- In-house development
- Internal deployment and maintenance of third-party tools
- Customer services development (websites, mobile apps, etc.)
- Open-source projects
- Other, please specify:

What is your country or region?*
[Choice of countries]

Please provide us with your contact details:
First name:
Email address:*

Thank you for completing our questionnaire. If this study is a good match for your profile, we will invite you to an interview via email.

B.2 Interview Script

Thank you for agreeing to take part in our research! My name is *[NAME]*, I am a researcher at JetBrains, and I am studying developers' user experience with the aid of surveys and interviews. Currently we are focused on studying communications in the developers teams.

Our interview will be 60-90 minutes long. I am going to ask you about your professional experience, about your experience with individual and team work. I will also ask you about possible complications and mistakes, where do they come from and what are their consequences.

Do you have any questions right now? If you are OK with that, I will record the interview for research purposes. To follow the GDPR law requirements, I am going to start recording and ask you if you agree to record the interview. If you do not mind, I would like to make notes while we are talking.

Cool, let's begin! First, I would like to learn a little bit about you and your job.

1. Tell me about your career in IT till today? For how many years in total have you worked?
2. What is your current position: developer, team lead, project manager, QA, etc.?
3. In what positions and roles have you worked in your career?
4. How many team projects did you approximately participate in?
5. What were the structure and the size of teams you worked with?
   - What is the current size and structure of your team?
   - What are the team member roles in the team you currently work in (e.g. data scientist, developer, devops, QA)?
   - What were the team member roles in the team you worked the longest?
6. We would like to know what the most common problems the developer teams face. Particularly important for us are the problems that arise due to bad coordination or bad coordination of interactions between the developers. The cases when work interaction procedures hamper reaching the results. When there are some lacks, weak points, incongruences or inefficient processes in the team. Could you please recall situations where the problems arose in a team due to the fact there were several participants and there were weak points in the interaction pipeline? Which would never happen for a single-person team.
7. Can you please formulate the **most typical** collective development problem?
8. Can you please formulate the **most significant** collective development problem that has a significant impact on the project?
9. How often in your practice do you encounter these problems? How many times per month/year does that happen?
10. Thank you. I would like to discuss a recent case in more detail. Please recall some teamwork related problem that was quite significant and happened in your work in the last half year.
    [A year, if nothing in half a year.]
    - How would you describe it?
    - What participants were involved in it?
    - What led to it, if you understood it, or figured it out from post mortems or discussions?
    - What were the consequences?
    - How did you solve the problem? Who participated in resolving the problem?
    - What did the experience teach you?
    - How much did it take to get the project in the state it was before the problem? You can estimate resources in person-hours (-days, -weeks, -months).
    - Did you change anything in how you or your team works after facing the problem? What exactly did you change?

11. Do you know any other examples of collective development difficulties in a team that you have not personally experienced? For example, you read about them or heard from your colleagues. Please list them and tell me how you learned about them. Do any of these difficulties seem significant to you?

12. Thank you. If I understand correctly, you have mentioned the following problems. *[Briefly list the problems covered by the respondent].* Let us imagine we have two scales for estimating problematic situations. Please assess the problems you have described according to these scales.
    Frequency scale:
    - 1 – I have never witnessed the problem;
    - 2 – I have encountered the problem on a couple of projects I have worked on;
    - 3 – I have encountered the problem on a significant part of the projects I have worked on;
    - 4 – I have encountered the problem on almost every project I worked on;
    - 5 – I have encountered the problem on almost every project I worked on more than once per project.

    Significance scale:
    - 1 – situation had minimal impact on the project;
    - 2 – situation required us to fix the consequences, but we managed pretty fast;
    - 3 – we had to move the deadlines somewhat to fix the consequences;
    - 4 – project was almost closed or we had to move the deadlines significantly;
    - 5 – the project was closed.

    [Send a copy of the scales to the meeting chat.]

13. Let us discuss more closely some of the teamwork related problems you have mentioned before. Please recall an example of a *[DESCRIPTION]* problem, preferably a recent one.
    [Select problems with interviewee's frequency and significance ratings' sum of at least 6. Go from the highest scored to the lowest scored problems on that list. Keep track of time to have enough time left for the questions about tools.]
    - How would you describe it?
    - What participants were involved in it?
    - What led to it, if you understood it, or figured it out from post mortems or discussions?
    - What were the consequences?
    - How did you solve the problem? Who participated in resolving the problem?
    - What did the experience teach you?
    - How much did it take to get the project in the state it was before the problem? You can estimate resources in person-hours (-days, -weeks, -months).
    - Did you change anything in how you or your team works after facing the problem? What exactly did you change?

14. Thank you for talking about the problems you encountered in your work. That was a really interesting and valuable conversation. Now, I would like discuss possible (and impossible) ways to address these problems. What do you do to prevent problems in the collective development procedures and the collective development itself? Can you give any tips or tricks for a newbie in the subject.

15. Do you know about existing analytic tools that help organizing team work? Do you use any of these tools? *[If the interviewee struggles to think of any, try asking about Microsoft or Oracle products. But do not suggest the answers.]*

16. *[For every problem that has a sum of the scores of at least 6].*
    Do you believe it is possible to create an analytical tool that would somehow address this problem? For example, by reducing its significance and
    or frequency. This tool might take any data on the work-related processes as an input (e.g. code, chats, code reviews) and produce descriptions of problems. It may also give recommendations on how to eliminate these problems.

17. Do you have any ideas of a tool for collective development that would simplify your work?

Thank you very much! That's all from me, I have no more questions.

Do you have any questions for me? Would you like to clarify anything?

Thank you for taking the time to attend the interview and sharing your valuable insights. We truly appreciate your contribution to our research. *[Ask about the prize. Ask if they want to receive the final paper.]*

Appendix C
The "Undesirable Patterns Overview" survey

Hello! Thank you for taking the time to help us by completing this survey. Not only are you helping us gather useful information, but you will also have the chance to win a prize: either a one year JetBrains All Products Pack subscription or a $100 Amazon gift card. Any information you provide will be confidential. Let's get started!

[TERMS AND CONDITIONS]
[MANDATORY QUESTIONS ARE HIGHLIGHTED WITH (*). END SURVEY SIGN MEANS THAT THE PARTICIPANT IS NOT ELIGIBLE TO TAKE THE MAIN PART OF THE SURVEY.]

What is you employment status?*
[Single-choice question]

- Fully employed by a company or organization
- Partially employed by a company or organization
- Self-employed (a person earning income directly from their own business, trade, or profession)
- Freelancer (a person pursuing a profession without a long-term commitment to any one employer) [END survey]
- Working student
- Student [END survey]
- Retired [END survey]
- Other, please specify: [END survey]

How many years have you been working in IT?*
[Single-choice question]

- I don't work in IT [END survey]
- Less than 2
- 2–4
- 4–7
- 7–15
- More than 15

What is your current job title or position?*
[Multiple-choice question]

- CIO, CEO, or CTO *(!)*
- Tester or QA Engineer *(!)*
- DBA
- Developer Advocate
- Instructor, Teacher, Tutor, etc.
- DevOps Engineer, Infrastructure Developer, etc. *(!)*
- Technical Support
- Team Lead *(!)*
- UX/UI Designer
- Developer, Programmer, or Software Engineer *(!)*
- Architect *(!)*
- Data Analyst, Data Engineer, or Data Scientist *(!)*

- Product or Project Manager *(!)*
- Technical Writer
- Systems Analyst *(!)*
- Other, please specify:

[If no (!) position has been checked - END survey]

How many team projects have you participated in as an engineer throughout your career?*
Note: Any project on which you have spent more than one month applies. It is considered to be a team project if you interacted with colleagues on the project at least once a week.
[Single-choice question]

- 0 [END survey]
- 1–4
- 5–9
- 10–19
- 20 or more

How many projects throughout your career have you led a group of people in?*
[Single-choice question]

- 0
- 1–2
- 3–5
- 6–9
- 10 or more

Which of the following tools do you regularly use?*
[Multiple-choice question]

- Source code collaboration tool (e.g. GitHub, GitLab, Bitbucket)
- Issue tracker (e.g. Jira, YouTrack)
- Team collaboration, task management, project or workflow management tools (e.g. Miro)
- Code review tool (e.g. Crucible, Upsource)
- Continuous Integration (CI) or Continuous Delivery (CD) tool (e.g. Jenkins, TeamCity)
- Service desk or helpdesk automation solutions (Zendesk)
- Static analysis tool (e.g. CodeClimate)
- Standalone IDE (e.g. Visual Studio, Eclipse, IntelliJ IDEA)
- Desktop Editor (e.g. Sublime Text, Atom, VS Code, Vim)
- In-cloud Editor or IDE
- None

Which of the following communication tools do you regularly use for work?*
[Multiple-choice question]

- Email (Microsoft Mail Server, Gmail, etc.)
- Instant messaging or video calling (Microsoft Teams, Slack, Skype, etc.)
- Video conferencing (Google Meet, Zoom, etc.)
- Calendars (Google Calendar, etc.)

- Corporate portal (MS Sharepoint, Pingboard, etc.)
- Service desk or Help desk (Zendesk, Jira Service Desk, etc.)
- Other, please specify:
- None

We will now ask you about problems occurring in the software development process. Let us broadly divide all such problems into two categories: technical and collective. A **technical** problem is a problem that can happen to any team, even a single-person one. A **collective development** problem is one that can happen only to a team of two or more members.

[PATTERNS SECTIONS. FIVE RANDOMLY SELECTED PATTERNS. EACH PATTERN – ONE SECTION.]

We will now present you with descriptions of five collective development problems. We would like you to assess whether you encounter such problems in your work, and answer several questions about them.

[Each section has the following structure.]

[PATTERN DESCRIPTION]

How often does this problem occur?*
[Single-choice question]
- This problem is outside my domain of responsibility [GO TO next section]
- I have never encountered this problem [GO TO next section]
- I have encountered this problem, but it was an unusual situation
- I encounter this problem occasionally, but not very often
- I often encounter this problem
- Other, please specify:

How significant is this problem?*
[Single-choice question]
- This problem hardly affects the workflow or does not affect it at all
- This problem affects the workflow, but does not impact work-related processes (release deadlines are not moved, no overtime work is required, no need to involve managers or other outside people to address the issue). Intermediate goals, however, may have to be changed
- This problem mildly impacts work-related processes (release deadlines are slightly moved, some features are postponed to be released later, managers participate in resolving the problem)
- This problem significantly impacts work-related processes (release deadlines are moved significantly as compared to the release cycle, some critical features are not shipped in time)
- This problem greatly impacts work-related processes (the project is closed or postponed indefinitely, some of the team members leave the team due to conflicts or general dissatisfaction)

Which of the following consequences has this problem ever led to? Please check all the options that describe a consequence that you have faced throughout your career.*
[Multiple-choice question]
- **Wasting time**
  - Waiting for other people to act
  - Redoing some task or fixing the code

  - Wasting time on the unnecessary communication
  - Other, please specify:
- **Project development impact**
  - The project was canceled
  - The project became insupportable
  - The project release was delayed
  - The product delivery was suboptimal or buggy
  - Other, please specify:
- **Team effects** consequences impact employees' work life.
  - Some of the team members left
  - There were more conflicts in the team
  - Team members were frustrated or their morale decreased
  - Other, please specify:
- **Company effects** consequences impact the company as a whole.
  - Damaged reputation
  - Toxic environment
  - Other, please specify:
- Other forms of impact (please specify)

Anything else you would like to share regarding the problem?
[Text Box]

[Last page]
Is there anything else you would like to share?
[Text Box]

Would you like us to email you with a further survey about tools for solving collective development problems?*
- Yes, please
- No, thanks

Would you like us to email you the resulting paper based on these research insights?*
- Yes, please
- No, thanks

[Prize choice. Local Amazon]
[Contacts]
[Contact and subscribe]

Thank you for sharing your experience with us!

Appendix D
The "Evaluation of Tool Suggestions" survey

Hello! Thank you for taking the time to help us by completing this survey. Not only are you helping us gather useful information, but you will also have the chance to win a prize: either a one year JetBrains All Products Pack subscription or a $100 Amazon gift card. Any information you provide will be confidential. Let's get started!

[TERMS AND CONDITIONS]
[MANDATORY QUESTIONS ARE HIGHLIGHTED WITH (*). END SURVEY SIGN MEANS THAT THE PARTICIPANT IS NOT ELIGIBLE TO TAKE THE MAIN PART OF THE SURVEY.]

What is you employment status?*
[Single-choice question]

- Fully employed by a company or organization
- Partially employed by a company or organization
- Self-employed (a person earning income directly from their own business, trade, or profession)
- Freelancer (a person pursuing a profession without a long-term commitment to any one employer) [END survey]
- Working student
- Student [END survey]
- Retired [END survey]
- Other, please specify: [END survey]

How many years have you been working in IT?*
[Single-choice question]

- I don't work in IT [END survey]
- Less than 2
- 2–4
- 4–7
- 7–15
- More than 15

What is your current job title or position?*
[Multiple-choice question]

- CIO, CEO, or CTO *(!)*
- Tester or QA Engineer *(!)*
- DBA
- Developer Advocate
- Instructor, Teacher, Tutor, etc.
- DevOps Engineer, Infrastructure Developer, etc. *(!)*
- Technical Support
- Team Lead *(!)*
- UX/UI Designer
- Developer, Programmer, or Software Engineer *(!)*
- Architect *(!)*
- Data Analyst, Data Engineer, or Data Scientist *(!)*

- Product or Project Manager *(!)*
- Technical Writer
- Systems Analyst *(!)*
- Other, please specify:

[If no (!) position has been checked - END survey]

How many team projects have you participated in as an engineer throughout your career?*
Note: Any project on which you have spent more than one month applies. It is considered to be a team project if you interacted with colleagues on the project at least once a week.
[Single-choice question]

- 0 [END survey]
- 1–4
- 5–9
- 10–19
- 20 or more

Let's assume that you have found a new tool that could help your current team better collaborate with each other and other teams, and you would like your team to use it. How much influence do you have in regards to adopting this tool?*
[Single-choice question]

- I can decide that our team will use this tool from now on, and it is unlikely to be overruled by someone else
- I cannot decide that our team will adopt this tool, but I can strongly influence a discussion about whether or not this tool should be used (e.g. I am one of 3 people who will make this decision together)
- I can suggest using this tool, but I have no impact on whether it will be adopted or not
- I have no say in these matters

Let's look at problems occurring in the software development process. These problems are broadly divided into 2 categories: technical and collective developement. A **technical** problem is a problem that can happen to any team, even a single-person one. A **collective development** problem is one that can happen only to a team of two or more members.

[PATTERNS AND TOOLS SECTIONS. FIVE RANDOMLY SELECTED PATTERNS. EACH PATTERN – ONE SECTION.]

The following are descriptions of five collective development problems. We would like you to assess whether you encounter such problems in your work, and answer several questions about tools that could possibly help identify and manage them.

[Each section has the following structure.]
[PATTERN DESCRIPTION]
How often does this problem occur?*
[Single-choice question]

- This problem is outside my domain of responsibility [GO TO next section]
- I have never encountered this problem [GO TO next section]
- I have encountered this problem, but it was an unusual situation
- I encounter this problem occasionally, but not very often
- I often encounter this problem

- Other, please specify:

Below are several descriptions of tools and features that could possibly help identify and manage this problem. Please assess how each tool may be beneficial to you.

[LIST OF (UP TO THREE) TOOL IDEAS.]
Each idea has the following structure:

[IDEA DESCRIPTION.]
[Single-choice question]
- Not beneficial at all
- Helps to manage the problem a little
- Solves a part of the problem
- Solves the problem completely

[After all the ideas, at the end of pattern section.]
If you have another idea for a tool that could help manage the problem mentioned above, please describe it here.
[Text Box]

Please add any comments you may have on the tool concepts presented above (e.g. possible modifications to the tool, or advantages and disadvantages of using it).
[Text Box]

Is there anything else you would like to share regarding the problem and any possible solutions?
[Text Box]

[Last page]
Is there anything else you would like to share?
[Text Box]

Would you like us to email you the resulting paper based on these research insights?*
- Yes, please
- No, thanks

[Prize choice. Local Amazon]
[Contacts]
[Contact and subscribe]

Thank you for sharing your experience with us!

References

[1] E. Caballero-Espinosa, J. C. Carver, and K. Stowers, "Community smells—the sources of social debt: A systematic literature review," *Information and Software Technology*, p. 107078, 2022.

[2] E. Doğan and E. Tüzün, "Towards a taxonomy of code review smells," *Information and Software Technology*, vol. 142, p. 106737, 2022.

[3] M. Jerzyk and L. Madeyski, "Code smells: A comprehensive online catalog and taxonomy," in *Developments in Information and Knowledge Management Systems for Business Applications: Volume 7*. Springer, 2023, pp. 543–576.

[4] "Stack overflow," 2008. [Online]. Available: https://stackoverflow.com/