# Branching Echoes: Exploring Branching Models as Sociotechnical Lenses for Community Smells

Diogo de Lima Menezes
Federal University of Mato Grosso do Sul
Campo Grande, MS, Brazil
diogo.menezes@ufms.br

Eos Xavier
Federal University of Mato Grosso do Sul
Campo Grande, MS, Brazil
eos.xavier@ufms.br

Rafael Tsutomu Jyoboji
Federal University of Mato Grosso do Sul
Campo Grande, MS, Brazil
rafael.tsutomu@ufms.br

Hudson Silva Borges
Federal University of Mato Grosso do Sul
Campo Grande, MS, Brazil
hudson.borges@ufms.br

Awdren de Lima Fontão
Federal University of Mato Grosso do Sul
Campo Grande, MS, Brazil
awdren.fontao@ufms.br

## Abstract

Branching models coordinate how software changes are isolated, integrated, validated, and promoted. In machine learning (ML)-based systems, they also mediate coordination across source code, data, models, experiments, and heterogeneous roles. Yet branching is typically analyzed as a technical configuration-management mechanism, leaving unclear whether its repository traces can reveal sociotechnical dysfunction. We report an exploratory single-case study of an early-stage ML project developing a system to extract structured information from judicial health-litigation documents. We analyzed five repositories and survey responses from all three members of the AI team, triangulating the branching model prescribed to the team, practitioners' perceptions, and enacted repository practices. Thematic analysis and repository mining revealed recurrent misalignments in issue traceability, pull-request discussion, experiment evaluation, and integration practices. These misalignments provided indicators consistent with the community smells *Black Cloud*, *Sharing Villainy*, and *Cookbook Development*; *Lone Wolf* emerged only from perception data and was not corroborated by repository evidence. We do not claim that branching practices cause or automatically detect community smells. Instead, our findings suggest that branching-model evidence can provide a low-cost sociotechnical proxy for locating coordination risks that merit further investigation. We derive implications for MLOps governance and identify conditions that future multi-team studies should evaluate.

## Keywords

Software Configuration Management, Branching Model, Machine Learning, Community Smells

## 1 Introduction

Branching models specify how software changes are isolated, integrated, validated, and promoted across development, testing, and production environments [7, 14]. Although commonly presented as technical configuration-management mechanisms, these models also define coordination boundaries: who may contribute to a branch, when work becomes available to others, which checks precede integration, and how responsibility for a change is transferred [18, 23, 28]. Prior studies associate branch structures and such policies with integration delays, merge conflicts, organizational alignment, and software-quality outcomes [6, 7, 23, 28]. Consequently, a prescribed branching model expresses how a team intends to coordinate its work, whereas its enacted use provides a repository-level record of how such coordination occurs.

This distinction becomes particularly relevant in machine learning-based software systems. In addition to source code, machine learning (ML) teams coordinate datasets, trained models, experiments, metrics, scripts, configuration files, and deployment artifacts [13]. These artifacts evolve at different rhythms and are commonly produced by professionals with different backgrounds, including software engineers, data engineers, and artificial intelligence (AI) engineers. Previous research reports recurring challenges involving versioning, reproducibility, documentation, handoffs, and collaboration across these roles [3, 5, 15, 17, 21, 22]. In this setting, a branching model can operate as part of the governance structure through which a team communicates integration readiness, records experimental decisions, and aligns contributions across technical roles.

Software engineering research has also shown that collaboration problems leave observable traces in development artifacts [1, 27]. Community smells describe recurring patterns of inadequate organization, communication, and coordination that may contribute to the accumulation of social debt [10, 26, 27]. Existing studies have investigated such smells through communication networks, issue trackers, commit histories, code comments, and practitioners' perceptions [4, 12, 27]. However, it remains unclear whether differences between how branching practices are prescribed, understood, and enacted can provide indirect evidence of conditions associated with community smells. We refer to such evidence as a *proxy*: it may locate sociotechnical conditions that merit investigation, but does not establish the presence or causes of a community smell or replace an investigation with the affected team.

We investigate this possibility through an exploratory single-case study of an early-stage ML-based software project conducted within a public innovation initiative. The project aimed to develop a system for extracting structured information from judicial health-litigation documents. We focus on its three-member AI team and analyze five repositories supporting the ML component. In the initial sprints, the research program and the project's leadership

prescribed a branching model to all teams. Our analytical mechanism contrasts three views of this model: the model as prescribed, the practices reported and understood by team members, and the practices evidenced by branches, commits, issues, and pull requests (PRs). Survey responses were examined through thematic analysis and subsequently contrasted with repository-mining evidence.

Our findings show recurrent misalignments concerning issue traceability, pull-request discussions, experiment evaluation, peer review, and branch integration. Evidence from both practitioners' perceptions and repository practices provided indicators consistent with *Black Cloud* and *Sharing Villainy*; *Cookbook Development* was more directly supported by the comparison between the prescribed process and its continued non-adoption, whereas *Lone Wolf* emerged only from perception data. These findings do not imply causality, but show how inconsistencies among prescribed, perceived, and enacted practices can make coordination risks visible. Accordingly, this paper provides an exploratory empirical account of branching-model evidence as a sociotechnical proxy, characterizes misalignment among these three views as the mechanism through which community-smell indicators become observable, and derives implications for machine learning operations (MLOps) governance. Given the single-case design, the contributions support analytical rather than statistical generalization and motivate broader multi-team studies.

## 2 Related Work

Prior research has formalized community smells [26] and investigated them through communication networks, version-control histories, issue assignments, open-source code samples, code comments, and practitioners' perceptions [1, 4, 9, 10, 12, 27]. This work follows the broader premise that software artifacts reflect how development activities are coordinated, as expressed by sociotechnical congruence [11]. Although such artifacts cannot fully represent team relationships, they provide non-intrusive evidence that can complement direct investigation with practitioners. A multivocal review of code samples in open-source ecosystems reinforces this premise: the condition of even auxiliary artifacts acts as a proxy for project health, with community smells often preceding or reinforcing technical degradation [9]. This study draws on four cataloged smells: *Black Cloud* (absence of structured communication), *Sharing Villainy* (knowledge not shared in forms the community retains), *Lone Wolf* (unilateral action disregarding peers), and *Cookbook Development* (attachment to routines, resisting adaptation) [10, 26].

Branching models similarly combine technical and organizational decisions. Prior studies associate branch structures with integration delays, merge conflicts, organizational alignment, and software-quality outcomes [6, 23]. Branches have also been conceptualized as goals pursued by virtual teams [7]. Other studies examine how branching choices depend on project context, influence integration at scale, and interact with human and technical aspects of merge work [14, 18, 28]. This literature establishes branching as a sociotechnical artifact, but primarily investigates integration and quality outcomes rather than community-smell indicators.

This perspective is especially relevant in ML-based systems, whose artifacts extend beyond source code. Studies report difficulties in versioning and reproducing these artifacts, despite their perceived importance [5, 22]. Heterogeneous and overlapping roles may also produce unclear responsibilities, limited documentation, and problematic handoffs between experimental and production work [3, 15, 17]. Recent research examines repository maturity and community smells in ML projects [4, 12, 16], process smells in industry [2], and the sociotechnical emergence of testing practices [25].

Overall, prior research establishes that community smells leave traces in development artifacts, branching models encode coordination decisions, and ML teams face distinctive configuration-management challenges. However, the intersection of these areas remains underexplored. In particular, previous studies have not explicitly examined whether misalignment among prescribed branching policies, practitioners' perceptions, and enacted repository practices can indicate named community smells. We address this intersection by analyzing branches, commits, issues, pull requests, and expected but absent integration records. Rather than proposing an automated detector, we investigate whether branching-model evidence can serve as a low-cost sociotechnical proxy for locating coordination conditions that merit examination.

## 3 Case Study Design

We conducted an exploratory case study following the guidelines proposed by Runeson and Höst [20]. This design enabled us to investigate branching practices and their sociotechnical implications within a contemporary ML-development project and its real-world organizational context. Because the study examines one project and one development team, its purpose is analytical exploration rather than statistical generalization.

### 3.1 Case and Unit of Analysis

The case is an early-stage project developing an ML-based system to extract structured information from judicial health-litigation documents. The project followed a Scrum-based process and involved 21 professionals across requirements engineering, front-end, back-end, data, and AI teams.

The unit of analysis is the AI team's use and understanding of the branching model across the five repositories supporting the ML component. The team comprised three practitioners with different levels of software-development and AI experience (Section 3.4.3).

### 3.2 Rationale

Versioning and integrating the heterogeneous artifacts of ML-based systems are recurring MLOps challenges [13], while poor configuration management may contribute to technical debt and limited reproducibility [21]. We focus on branching models because they define how contributions are isolated, reviewed, integrated, and promoted. Therefore, gaps between prescribed policies and enacted use may provide observable evidence of coordination problems.

Branching-related evidence is already available in software repositories and can be analyzed without introducing additional monitoring mechanisms. In this study, we contrast practitioners' reports with branches, commits, issues, and PRs to examine whether misalignments indicate conditions associated with community smells. We do not assess whether the prescribed model is intrinsically appropriate or claim that repository evidence confirms a smell.

## 3.3 Research Questions and Goal

This study evaluates branching-model evidence as a proxy for community smells in an ML-based software project. To identify sociotechnical dysfunction, we analyze, from the viewpoint of researchers and practitioners, the team's usage, shared perceptions, and agreements around branch structure and integration policies.

**RQ1: To what extent do the perceptions and shared understanding of team members around the branching model reveal the presence of community smells?** This question probes the perception side of the lens: whether the members' shared understanding is, by itself, informative about community smells. **Metric 1.1:** Community smells identified from team members' perceptions. **Metric 1.2:** Degree of consensus and divergence among members regarding branching-model integration policies.

**RQ2: To what extent does the comparison between the actual use of the branching model and the reported use by the team members indicate the presence of community smells?** This question probes the material side: whether the repository record corroborates or contradicts what the members report. **Metric 2.1:** Degree of alignment between the team's perceived branching policies and the actual use of branching-model integration policies on workflow artifacts (issues and PRs).

## 3.4 Data Collection & Execution

*3.4.1 Artifacts.* Two online forms supported the study. The first collected self-reported demographic data on the participants' professional background. The second, a survey form, was designed to gather qualitative data on participants' perceptions and understanding of the branching model, their current roles, and how they report its actual use. This form comprised 16 questions, combining closed-ended questions (a yes/no question and five-point Likert-type scales, from *never* to *always*) with open-ended questions eliciting the rationale behind each reported practice. The questions covered roles and tooling, the repository strategy, issue-based change management, evaluation metrics and policies in PRs, the registration of experiment discussions, and anticipated changes to configuration-management activities. Both forms were reviewed internally by the authors against the metrics they were intended to inform, but were not piloted with respondents before administration (Section 4). All relevant artifacts used in this study are publicly available, as described in the Artifact Availability section.

*3.4.2 Mining Software Repositories (MSR).* We established an MSR protocol designed to gather qualitative and quantitative data from GitHub repositories maintained by the AI team. The protocol was structured to extract all available data regarding branches, commits, issues, PRs, and documentation. The collected data were persisted in a local database (SQLite). Subsequently, the data underwent a transformation stage, generating a report with an intuitive visualization for further quantitative analysis – containing data such as commit, branch, issue, and PR counts, in addition to indicators such as the rates of PRs with comments and of PRs merged without peer review – and a human-readable file containing the extracted textual content in an organized format. We used Claude Opus 4.8 under supervision to implement the collection and report generation scripts and document the MSR process in a protocol, using PyDriller [24] and PyGithub. We also ran csDetector [1] on the five repositories. Its detection models are trained by its authors on substantially larger open-source projects, and on our five industrial repositories the tool reported near-uniform detections despite their very different sizes, which manual inspection of the artifacts did not corroborate; we therefore did not rely on its output. For confidentiality reasons, the analyzed data cannot be publicly shared; the collection scripts, the MSR protocol, and the survey instruments are included in the replication package.

*3.4.3 Participant Selection.* To strengthen the reliability of the collected perceptions of the prescribed branching model, given the naturally small population, all members of the AI team were selected as participants in the study. Each participant was anonymized and identified as PX, where X represents a randomly assigned number.

Table 1 summarizes the participants' self-reported experience in software development and in AI projects, the latter on a five-point Likert-type scale from *no experience* to *advanced experience*. By the eighth sprint, P1, P2, and P3 identified as developers and AI researchers, conducting experiments with diverse AI techniques to construct the ML component, while P2 served as the technical lead.

**Table 1: Participants' experience profile**

| P | Software Development | AI Development |
|---|---|---|
| P1 | University projects only | No experience |
| P2 | $>$ 5 years | Good experience (3.5 years) |
| P3 | $<$ 2 years | Good experience (3.3 years) |

*3.4.4 Study Execution.* The analyzed project was organized into three-week sprints; the study's main data collection took place in the eighth. In the initial sprints, all teams had been introduced to the prescribed branching model, adapted from Gitflow (`main`, `staging`, `developer`): feature or experimental branches merged into `developer`, which was promoted to `staging` and then `main`. Teams defined and documented their branch-integration rules and could adapt the model when changes were justified, discussed, and recorded. The initial model presented example processes for branch integration, requiring only that the team use PRs and issues to document change management. Teams defined the evaluation criteria for each change request, while the model suggested increasingly strict severity levels as changes neared the main branch.

The two forms (Section 3.4.1) were administered at different points: the demographic form, during the first sprint, gathered participants' experience and roles, while the survey form, administered during the eighth sprint, assessed their perceptions of the branching-model integration policies and updated their role information. As P2 joined the team after the first sprint, the corresponding demographic form was completed only at the end of the study execution. The perception responses were first submitted to thematic analysis (Section 3.4.5). We mined the five ML repositories across eight sprints (Section 3.4.2) and contrasted each participant's reported practices with commits, branches, PRs, and issues.

*3.4.5 Data Analysis.* To analyze the survey responses, this study used a thematic analysis approach, following the recommendations of Braun and Clarke [8], to identify interrelated themes relevant to the research questions.

To extract codes from the survey responses, inductive analysis iterations were conducted for each question. Three researchers

independently analyzed the answers, extracting relevant codes from text segments (quotations). To mitigate potential bias arising from shared coding preferences, they were instructed not to discuss or align their coding strategies during this phase, thereby preserving diverse individual perspectives. Still individually, each researcher then grouped the emerging codes into constructs of progressively higher abstraction. This grouping combined an inductive movement, deriving groupings from the data, with a deductive one, verifying whether the resulting constructs could be linked to community smells. The deductive step followed a shared glossary of smells compiled from the systematic review by Caballero-Espinosa et al. [10], consistent with the original formulation of community smells [26] and with their investigation in ML-based systems [4]. The linking of groupings to smells was interpretive, emerging from the researchers' analysis of the coded patterns in light of the glossary definitions. Every grouping link between a lower- and a higher-abstraction construct was justified with a recorded reasoning, enabling verification by peers.

After the individual phase, the researchers convened to reconcile their codes and groupings. Similar outputs were merged and divergent ones re-examined against the recorded quotations and justifications until consensus, producing a shared synthesis, retaining the quotations associated with each code.

The analysis of repository evidence was conducted inductively, using each survey question and the practitioners' answers to it as context. The questions were partitioned into three disjoint sets, each assigned to one researcher. For each question, the researcher independently examined the mined artifacts to determine: what could be concluded about the team's branching practice strictly from the material evidence; whether each participant's reported practice aligned with that evidence; and where a misalignment was identified and justifiable, whether the evidence supported indicating the presence of specific community smells, and which. The researchers recorded the artifacts underlying each conclusion and the justification for each smell indication.

After individual analysis, the three researchers met to review all questions. They examined the recorded artifacts and justifications and debated the interpretations. The resulting synthesis expressed the group's view on the degree of alignment between the practitioners' reported practice and the branching policies evidenced in the repository. Where misalignments were substantiated, the synthesis also recorded community smells indicated as potential sources of the observed dissonance between reported and evidenced practice.

## 3.5 Results and Discussion

*3.5.1 RQ1: To what extent do the perceptions and shared understanding of team members around the branching model reveal the presence of community smells?*

RQ1 Summary

*The practitioners' perceptions provided indicators consistent with four community smells:* Black Cloud, Sharing Villainy, *and, more tentatively,* Lone Wolf *and* Cookbook Development *(Metric 1.1). The strongest signal was the limited shared understanding of branching and integration practices: participants agreed on surface-level aspects of the process but reported different rationales, policies, and evaluation practices (Metric 1.2).*

The independent coding rounds produced 165 codes, of which 109 were retained after reconciliation. The resulting synthesis was dominated by themes describing divergence among participants rather than consistently shared practices.

The strongest theme, **divergent perceptions of the team's processes and policies**, appeared across all survey blocks. The participants reported three different rationales for the same multi-repository strategy: "to separate services from experiments" (P1), separating ML methodologies (P2), and supporting a future microservice architecture through a leadership decision (P3). Similar divergences concerned the purpose of issues, the use of evaluation metrics, and the need to adapt configuration-management practices. Moreover, no participant referred to a recurring mechanism through which these interpretations were discussed and reconciled. This pattern is consistent with *Black Cloud*, notably the limited structured communication about the team's process.

A second theme concerns **inadequate conditions for preserving and exchanging knowledge**. P1 reported experimental comparisons stored in spreadsheets attached to issue comments, while P3 stated that, for some experiments, only the final result was recorded and "the discussion occurred verbally in a later meeting." Documentation practices varied across members and experiments, without a shared protocol. These accounts provide indicators consistent with *Sharing Villainy*: relevant knowledge was produced but not systematically retained in a form accessible to the team.

Two narrower themes yielded more tentative indications. P3's perception that repository decisions were made unilaterally by the leadership, together with evaluation standards that only one participant could describe, suggests a possible concentration of decision-making and knowledge consistent with *Lone Wolf*. In addition, P1 and P3 anticipated no need to change configuration-management practices ("I don't foresee any need for change," P3), whereas P2 expected changes, providing a limited indication of *Cookbook Development*. These smells do not have equivalent evidential support: *Black Cloud* and *Sharing Villainy* recur across multiple survey topics, whereas *Lone Wolf* relies on two narrower observations and *Cookbook Development* on a single divergence combined with the absence of a formalized adaptation policy.

Regarding shared understanding (Metric 1.2), divergence was evident not only in isolated contradictory responses but also in the structure of the reconciled code set. Several higher-level themes captured divergent perceptions of pull-request policies, conflicting interpretations of issue usage, and dissonant expectations about adapting configuration-management practices. In one case, the synthesis retained the opposing groups "the team foresees changes" and "the team foresees no changes" because the responses could not be reconciled. Agreement was largely limited to the existence of practices, such as the use of multiple repositories, whereas their purposes and governing policies remained contested. Experiment evaluation represented the clearest coordination boundary, concentrating disagreements from all three participants about metrics, documentation, and the criteria used to assess and integrate experimental work. Thus, even before repository evidence was considered, the perception data indicated limited shared understanding of how the team coordinated and evaluated its work.

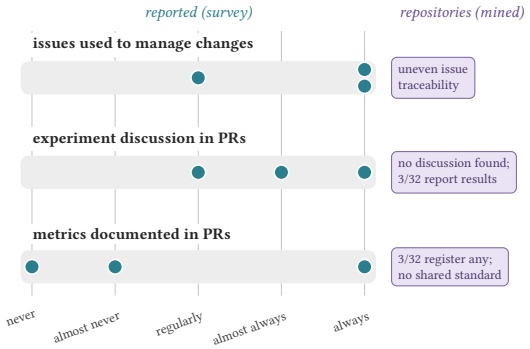


**Figure 1: Reported practice versus repository evidence; each dot is one practitioner's answer.**

*3.5.2 RQ2: To what extent does the comparison between the actual use of the branching model and the reported use by the team members indicate the presence of community smells?*

RQ2 Summary

*The comparison provided indicators consistent with* Black Cloud, Sharing Villainy, *and* Cookbook Development *(Metric 2.1). Reported and enacted practices diverged in issue traceability, experiment discussions, evaluation criteria, peer review, and branch integration (Figure 1). Repository evidence corroborated two indications from RQ1 and strengthened the tentative indication of* Cookbook Development*; however, it did not provide a repository correlate for* Lone Wolf.

We contrasted the practitioners' answers with the mined repository evidence in three question blocks: (i) the use of issues for change management, (ii) the registration of experiment discussions in pull requests, and (iii) the use and documentation of metrics and additional evaluation policies (Figure 1). For each block, we identified substantiated misalignments between reported and enacted practices and assessed whether they provided indicators consistent with specific community smells.

In the first block, P1 and P2 reported the team *always* used issues to manage changes and document development and experimentation, whereas P3 reported *regular* use and described issues as being used "as a consequence of creating tasks [...] not primarily for reporting bugs or requesting changes". Across all repositories, branch names often encoded issue identifiers (e.g., `feature/<issue>`), but adherence was uneven. One repository preserved full traceability, while others contained commits without issue references. In the repository with the largest history, sequences of product-code commits were made directly to `developer`, bypassing issue registration, peer review, and pull-request discussion. Thus, the evidence partially supports both accounts but reveals no consistently enacted standard for change management: experiment issues follow different structures, and issue templates are not systematically used.

In the second block, participants reported that experiment discussions were registered in pull requests *almost always*, *always*, or *regularly*. However, no discussion of experimental outcomes was identified in the analyzed pull requests or issues. The substantive exchanges that were found concerned implementation and configuration, while experimental results were occasionally reported by their authors without subsequent discussion. Only 3 of the 32 analyzed PRs recorded results, 43.8% contained any substantive comment, and 14 of the 22 merged PRs (63.6%) were integrated without a human comment. P3's report that some discussions occurred verbally helps explain this absence, while a PR comment conceding that "the current flow of the repository was set before the definition of the SCM [software configuration management] guidelines [...] we will still fix it" shows that the team was aware of the misalignment but deferred its correction.

The third block concerned metrics for comparing experiments and the documentation of additional evaluation policies. P1 and P2 reported that metrics were not used, citing different experimental scopes and the absence of standard datasets. P3 reported that metrics were used within the same experimental context and listed F1-score, precision, recall, and accuracy, while conceding that "we don't define the policy objectively". Reports about documentation frequency ranged from *always* to *never* (bottom track of Figure 1). Repository evidence was closer to the latter accounts: only 3 of the 32 pull requests recorded metrics or evaluation results, and experiments of the same type reported different measures. Some artifacts included unreported acceptance and reproducibility criteria, but these were applied inconsistently. Merges lacked approval, self-merges occurred, and branch protection and continuous integration were absent.

Together, the three blocks reveal more than simple noncompliance with a prescribed model. The repositories record inconsistent interpretations of change management, experiment evaluation, and integration, providing indicators consistent with *Black Cloud*. Experimental knowledge is retained through sparse, author-dependent records and verbal interactions rather than shared conventions, providing indicators consistent with *Sharing Villainy*. Finally, the acknowledged but unresolved divergence between the prescribed guidelines and the enacted workflow strengthens the indication of *Cookbook Development*. The strongest signal is the conflict among reported practices, observed practices, and missing expected records.

*3.5.3 Reading RQ1 and RQ2 Together.* The two readings converge on a core finding the design did not guarantee: the repositories could have agreed with the reported practice, and they did not. *Black Cloud* and *Sharing Villainy* were indicated by independent evidence on both sides; *Cookbook Development*, the most tentative indication in RQ1, is the one RQ2 substantiated most literally, in the team's acknowledgment that the prescribed guidelines are not followed, never accompanied by a recorded adaptation; only *Lone Wolf* does not survive the crossing, remaining a perception-side hypothesis for validation with the team. Because self-reports and repositories have distinct limitations, their convergence is informative.

This inverts the valuation reported for ML settings, where practitioners find versioning hard yet deem it essential, since datasets, models, and experiment metrics resist conventional control [5, 22]. Here the burden is met with silence: experimental discussion is retained nowhere, and experiment evaluation, the coordination point most specific to ML work, is where perceptions diverge most and the record is emptiest, echoing the thin documentation and broken handoffs reported for ML teams [3, 15, 17] and the environment in which *Sharing Villainy* takes hold, knowledge sharing not perceived as productive work [10]. These indications are early warnings of accumulating social debt [26], tied to technical degradation [10]

and, in ML projects, to self-admitted technical debt [12]. Repository bypasses may reflect community smells, revealing where social debt becomes integration risk.

The joint reading thus serves as preliminary evidence that branching-model analysis can act as a proxy for identifying community smell indicators in an ML team, and as a characterization of the mechanism: misalignment among the prescribed model, the practitioners' perceptions, and the enacted practice, as the unit of evidence through which coordination issues become visible. For MLOps governance, the implication is that branch usage deserves monitoring not only for version-control compliance, but as evidence of team alignment, integration readiness, and sociotechnical health: divergent rationales for one policy, empty discussion records, and unreviewed merges are signals available at low cost in artifacts the team already produces. The lens has limits: the sharpest indications appear where its two sides cross, as in experiment evaluation, yet a reading built on misalignment says little about a team in genuine agreement, a question this single case cannot answer.

## 4 Study Design Trade-offs & Threats to Validity

Following Robillard et al. [19], we discuss the main design decisions and their implications for interpreting the findings.

*Evidence Sources.* Community smells are latent social phenomena that neither surveys nor repositories capture directly. We combined an asynchronous survey with repository mining instead of relying only on interviews, observation, communication channels, or repository metrics. This choice enabled us to contrast the perceptions of all AI-team members with durable evidence from all five repositories. Surveys capture rationales but are exposed to ambiguous terminology, recall, and social-desirability effects, a risk heightened by the absence of a pilot; repositories provide observable traces but record only what reaches GitHub. We paired frequency-based questions with open-ended rationales and treated missing artifacts as informative only when a durable record was expected from the prescribed process or participants' accounts. Thus, the design supports triangulation but produces indicators rather than diagnoses of community smells.

*Depth versus Breadth.* We investigated one three-member AI team rather than multiple teams. This decision provided access to the complete team, all repositories supporting the ML component, and the prescribed branching model, enabling a detailed comparison among prescribed, perceived, and enacted practices. The trade-off is limited breadth: the project's early stage, small team, heterogeneous experience, and leadership structure provide rival explanations for the observed misalignments. Non-adherence may reflect process immaturity or an unsuitable branching model rather than persistent dysfunction. Accordingly, *Cookbook Development* reflected acknowledged non-adherence without documented adaptation, despite adaptation being allowed.

*Collection Timing.* We collected perceptions asynchronously during the eighth sprint rather than through repeated interviews or continuous observation. This reduced interference with project activities and allowed written rationales, but a single collection point cannot reveal how perceptions evolved. Responses may also reflect recall, social desirability, or exchanges among participants. Repository data from the first through the eighth sprint reduced dependence on retrospective reports, but cannot establish whether the identified patterns persisted or emerged in a specific period.

*Interpretation and Reliability.* We used inductive thematic analysis followed by interpretation against a community-smell glossary instead of relying solely on predefined codes or automated detection. This preserved contextual meaning but increased dependence on researcher judgment. Researchers first coded and grouped data independently, documented the rationale for each abstraction and smell link, and reconciled interpretations afterwards. Repository evidence was analyzed with the same written-justification procedure. These steps preserve a reviewable chain of evidence, but do not guarantee that another team would produce identical classifications.

## 5 Conclusion and Future Work

This exploratory single-case study investigated whether branching-model evidence can serve as a low-cost sociotechnical proxy for community smells in ML-based development. Contrasting the prescribed branching model, practitioners' perceptions, and enacted repository practices revealed recurrent misalignments: practitioners held conflicting views regarding repository strategies, the use of issues, and evaluation policies; no discussions concerning experiment results were found within PRs; only 3 out of 32 PRs had any documented metrics or results; and 14 of the 22 merged PRs were integrated without any human commentary. *Black Cloud*, *Sharing Villainy*, and *Cookbook Development* were indicated by independent evidence from both perceptions and repositories; *Lone Wolf* emerged only from perception data and found no repository correlate. Experiment evaluation combined the greatest divergence in perceptions with the emptiest repository record. The branching model did not cause these conditions, but it echoed them. For MLOps governance, divergent rationales for one policy, empty discussion records, and unreviewed merges are low-cost coordination-risk signals in artifacts the team already produces.

**Future work.** Multi-team and multi-project studies should replicate the analysis across teams of varying sizes, maturity levels, and, especially, different branching models, to establish when the proxy is informative beyond a single early-stage team; longitudinal and comparative designs could then examine whether particular structures and policies correlate with specific smells, moving from locating indicators toward supporting causal claims. Future work should also validate the indicators with the affected teams and examine teams in genuine agreement, to assess whether view alignment distinguishes healthy coordination.

## Artifact Availability

The artifacts of this study are available in an Open Science Framework (OSF) repository: https://doi.org/10.17605/OSF.IO/N6Z9S.

## Acknowledgements

This study was financed in part by the Fundação de Apoio ao Desenvolvimento do Ensino, Ciência e Tecnologia do Estado de Mato Grosso do Sul (FUNDECT, Call No. 42/2024), and by the Federal University of Mato Grosso do Sul (UFMS).

## References


[1] Nuri Almarimi, Ali Ouni, Moataz Chouchen, and Mohamed Wiem Mkaouer. 2021. csDetector: an open source tool for community smells detection. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*. ACM, 1560–1564. doi:10.1145/3468264.3473121

[2] Uğur Can Altun, Ismail Sergen Göçmen, Emre Sülün, Erdem Tuna, and Eray Tüzün. 2025. Process smells in practice: an evaluative case study. *Empirical Software Engineering* 30, 5 (2025), 115. doi:10.1007/s10664-025-10664-8

[3] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. 2019. Software Engineering for Machine Learning: A Case Study. In *Proceedings of the 41st IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 291–300. doi:10.1109/ICSE-SEIP.2019.00042

[4] Giusy Annunziata, Stefano Lambiase, Damian A. Tamburri, Willem-Jan van den Heuvel, Fabio Palomba, Gemma Catolino, Filomena Ferrucci, and Andrea De Lucia. 2025. Uncovering Community Smells in Machine Learning-Enabled Systems: Causes, Effects, and Mitigation Strategies. *ACM Transactions on Software Engineering and Methodology* 34, 6, Article 176 (2025). doi:10.1145/3712198

[5] Anders Arpteg, Björn Brinne, Luka Crnkovic-Friis, and Jan Bosch. 2018. Software Engineering Challenges of Deep Learning. In *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 50–59. doi:10.1109/SEAA.2018.00018

[6] Christian Bird and Thomas Zimmermann. 2012. Assessing the Value of Branches with What-if Analysis. In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering (FSE 2012)*. ACM, Article 45, 11 pages. doi:10.1145/2393596.2393648

[7] Christian Bird, Thomas Zimmermann, and Alex Teterev. 2011. A theory of branches as goals and virtual teams. In *Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering*. 53–56.

[8] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology* 3, 2 (2006), 77–101. doi:10.1191/1478088706qp063oa

[9] Arthur Bueno, Bruno B. P. Cafeo, Maria Istela Cagnin, and Awdren Fontão. 2025. Socio-Technical Smell Dynamics in Code Samples: A Multivocal Review of Their Emergence, Evolution, and Co-Occurrence. In *Anais do XXXIX Simpósio Brasileiro de Engenharia de Software (SBES)*. doi:10.5753/sbes.2025.9647

[10] Eduardo Caballero-Espinosa, Jeffrey C. Carver, and Kimberly Stowers. 2023. Community smells—The sources of social debt: A systematic literature review. *Information and Software Technology* 153 (2023), 107078. doi:10.1016/j.infsof.2022.107078

[11] Marcelo Cataldo, James D. Herbsleb, and Kathleen M. Carley. 2008. Socio-technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity. In *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 2–11. doi:10.1145/1414004.1414008

[12] Shamse Tasnim Cynthia, Nuri Almarimi, and Banani Roy. 2025. How Do Community Smells Influence Self-Admitted Technical Debt in Machine Learning Projects?. In *Proceedings of the 19th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. doi:10.1109/ESEM64174.2025.00072

[13] Beyza Eken, Samodha Pallewatta, Nguyen Khoi Tran, Ayse Tosun, and Muhammad Ali Babar. 2026. A Multivocal Review of MLOps Practices, Challenges and Open Issues. *Comput. Surveys* 58, 2, Article 39 (2026), 39:1–39:35 pages. doi:10.1145/3747346

[14] Pedro Lopes, Paola Accioly, Paulo Borba, and Vitor Menezes. 2025. Choosing the Right Git Workflow: A Comparative Analysis of Trunk-based vs. Branch-based Approaches. In *Anais do XXXIX Simpósio Brasileiro de Engenharia de Software (SBES)*. doi:10.5753/sbes.2025.9903

[15] Alina Mailach and Norbert Siegmund. 2023. Socio-Technical Anti-Patterns in Building ML-Enabled Software: Insights from Leaders on the Forefront. In *Proceedings of the 45th International Conference on Software Engineering (ICSE)*. 690–702. doi:10.1109/ICSE48619.2023.00067

[16] Nadia Nahar, Haoran Zhang, Grace Lewis, Shurui Zhou, and Christian Kästner. 2025. The Product Beyond the Model: An Empirical Study of Repositories of Open-Source ML Products. In *Proceedings of the 47th International Conference on Software Engineering (ICSE)*. doi:10.1109/ICSE55347.2025.00006

[17] Nadia Nahar, Shurui Zhou, Grace Lewis, and Christian Kästner. 2022. Collaboration Challenges in Building ML-Enabled Systems: Communication, Documentation, Engineering, and Process. In *Proceedings of the 44th International Conference on Software Engineering (ICSE)*. 413–425. doi:10.1145/3510003.3510209

[18] Nicholas Nelson, Caius Brindescu, Shane McKee, Anita Sarma, and Danny Dig. 2019. The life-cycle of merge conflicts: processes, barriers, and strategies. *Empirical Software Engineering* 24, 5 (2019), 2863–2906.

[19] Martin P Robillard, Deeksha M Arya, Neil A Ernst, Jin LC Guo, Maxime Lamothe, Mathieu Nassif, Nicole Novielli, Alexander Serebrenik, Igor Steinmacher, and Klaas-Jan Stol. 2024. Communicating study design trade-offs in software engineering. *ACM Transactions on Software Engineering and Methodology* 33, 5 (2024), 1–10.

[20] Per Runeson and Martin Höst. 2009. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering* 14, 2 (2009), 131–164. doi:10.1007/s10664-008-9102-8

[21] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, and Dan Dennison. 2015. Hidden Technical Debt in Machine Learning Systems. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*. Curran Associates, Inc., 2503–2511. https://proceedings.neurips.cc/paper/2015/hash/86df7dcfd896fcaf2674f757a2463eba-Abstract.html

[22] Alex Serban, Koen van der Blom, Holger Hoos, and Joost Visser. 2020. Adoption and Effects of Software Engineering Best Practices in Machine Learning. In *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. ACM, Article 3, 12 pages. doi:10.1145/3382494.3410681

[23] Emad Shihab, Christian Bird, and Thomas Zimmermann. 2012. The Effect of Branching Strategies on Software Quality. In *Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '12)*. ACM, 301–310. doi:10.1145/2372251.2372305

[24] Davide Spadini, Maurício Aniche, and Alberto Bacchelli. 2018. PyDriller: Python framework for mining software repositories. In *Proceedings of the 2018 26th ACM Joint meeting on european software engineering conference and symposium on the foundations of software engineering*. 908–911.

[25] Mark Swillus, Rashina Hoda, and Andy Zaidman. 2026. On the Emergence of Testing Strategies: A Socio-Technical Grounded Theory. *Empirical Software Engineering* 31, 5, Article 147 (2026). doi:10.1007/s10664-026-10828-0

[26] Damian A. Tamburri, Philippe Kruchten, Patricia Lago, and Hans van Vliet. 2015. Social debt in software engineering: insights from industry. *Journal of Internet Services and Applications* 6, 1, Article 10 (2015). doi:10.1186/s13174-015-0024-6

[27] Damian A. Tamburri, Fabio Palomba, and Rick Kazman. 2021. Exploring Community Smells in Open-Source: An Automated Approach. *IEEE Transactions on Software Engineering* 47, 3 (2021), 630–652. doi:10.1109/TSE.2019.2901490

[28] Weiqin Zou, Weiqiang Zhang, Xin Xia, Reid Holmes, and Zhenyu Chen. 2019. Branch Use in Practice: A Large-Scale Empirical Study of 2,923 Projects on GitHub. In *2019 19th IEEE International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 306–317. doi:10.1109/QRS.2019.00047