From Specification to Practice: An Empirical Study of Ethereum ERC Standard Adoption in Open Source Projects

Pamella Soares
Graduate Program in Computer Science
State University of Ceará
Fortaleza, CE, Brazil
pamella.soares@aluno.uece.br

Airlon Silva Filho
Center for Science and Technology and Center for Distance Education
Federal University of Cariri
Juazeiro do Norte, CE, Brazil
airlon.filho@aluno.ufca.edu.br

Raissa Rodrigues
Center for Science and Technology
Federal University of Cariri
Juazeiro do Norte, CE, Brazil
raissa.rodrigues@aluno.ufca.edu.br

Allysson Allex Araújo
Center for Science and Technology
Federal University of Cariri
Juazeiro do Norte, Brazil
allysson.araujo@ufca.edu.br

Jerffeson Souza
Graduate Program in Computer Science
State University of Ceará
Fortaleza, CE, Brazil
jerffeson.souza@uece.br

Abstract

Context. Ethereum Requests for Comments (ERCs) define technical specifications for smart contracts, but non-compliant and partially adopted implementations have been associated with vulnerabilities and financial losses. How developers adopt these standards in open-source practice remains empirically unexplored. **Objective.** This study investigates how ERC specifications translate into practice by characterizing developer discussions around these standards in open-source repositories. **Method.** We applied Mining Software Repository (MSR) to the Web3BlockSet dataset (391,596 issues/PRs, 381 official repositories and 4,500 community repositories, 2012-2026), identified ERC mentions via regular expressions (retaining 11,537 candidates, 3.0% of the corpus), validated discussions with Large Language Models (LLMs) to yield 9,005 substantive discussions (78.1%), and manually validated a stratified sample of 400 issues/PRs. **Results.** ERC adoption is concentrated in a few standards and repositories, follows ecosystem market cycles, and bug-related discussions predominate across nearly all ERCs. Co-occurrence patterns expose implicit inter-standard dependencies, lifecycle trajectories show progressive improvement in mature proposals in contrast to stagnation in recent ones, and 146 validated observed practices ground recurring implementation challenges in developer evidence. **Contributions.** We contribute *(i)* an empirical characterization of ERC adoption patterns, *(ii)* observed practices based on real-world developer discussions, *(iii)* a replicable methodology, and *(iv)* an annotated dataset of substantive ERC discussions.

Keywords

Ethereum, Smart Contracts, ERC Standards, Mining Software Repositories

1 Introduction

Blockchain stores transactions in cryptographically chained blocks, ensuring immutability and decentralization without trusted intermediaries [8]. Built on this infrastructure, smart contracts execute commands automatically when conditions are met [34]. This programmability is critical to Blockchain-Oriented Software Engineering (BOSE) [27], demanding specialized tools and practices to address the impossibility of updating contracts after deployment [16].

Although Ethereum [8] quickly became the leading platform for smart contracts, the lack of standards for contract interoperability became a problem as decentralized applications grew in complexity. In this context, *Ethereum Request for Comments* (ERCs) were designed as technical standards defining interfaces and expected behaviors, establishing how systems should interoperate. ERCs are proposed within the *Ethereum Improvement Proposals* (EIPs) process[1], a formal artifact comprising interface specifications, behavioral requirements, and security considerations whose lifecycle spans submission, public discussion, and iterative revision [5].

The evolution of these standards follows the quickly changing demands of decentralized applications. For instance, ERC-20 established the foundation for fungible tokens (FT), allowing the creation of interchangeable digital assets and driving phenomena such as Initial Coin Offerings (ICOs) [13, 37]. The limitation of this model for representing uniqueness led to ERC-721 [17], which enabled non-fungible tokens (NFTs) and made possible new digital markets, later complemented by mechanisms such as ERC-2981 for royalties [7]. Seeking flexibility, ERC-1155 addressed efficiency by managing multiple asset types within a single contract [28].

With increasing complexity, standards focused on interoperability and infrastructure emerged, such as ERC-165, which allows detection of supported interfaces [30]. In turn, ERC-1820 established a global registry of capabilities, facilitating composition among contracts [4]. Additionally, ERC-1967 and ERC-7201 addressed the challenge of evolving immutable contracts [19, 25].

As the Decentralized Finance (DeFi) ecosystem was becoming mature, other standardization efforts expanded toward higher-level financial primitives. ERC-4626, for example, formalized yield-bearing vaults, promoting consistency and interoperability across protocols [32]. More recent proposals, such as ERC-6909, reflect an ongoing effort to simplify and streamline token standards [31]. As

[1] https://eips.ethereum.org/

one can notice, each ERC usually arises in response to practical limitations, emerging application domains, or security challenges observed in deployments. However, despite this growing and increasingly structured body of standards, a gap remains between specification and practice. This gap is not simply anecdotal but has been observed in empirical analyses of real-world codebases. Large studies of open-source repositories show that smart contract specifications are frequently only partially implemented, inconsistently interpreted, or extended beyond their intended scope [9, 23]. Therefore, these deviations are not solely theoretical, they have been linked to concrete consequences, including user confusion, asset loss, and financial exploitation in deployed systems [15, 20].

Compounding this problem, developers routinely adapt the standards to gas optimization constraints and deployment environments the specifications did not foresee, although these adaptations remain undocumented [23]. Therefore, understanding where, how, and why this phenomenon manifests in practice is a critical open problem. For researchers, this signals the need for empirical methodologies that capture the distance between normative specifications and reality. For practitioners, it exposes the absence of evidence-based guidance where specifications are omitted or ambiguous.

To address this problem, we aim to conduct exploratory research into ERC adoption, the broader process of implementation, problem-solving, and interpretation of a standard. We investigate this through Mining Software Repositories (MSR), which is well-suited here because ERC adoption leaves direct traces in public repositories, where developers raise issues to report violations, open pull requests to propose fixes, and comment on edge cases the specification did not anticipate [11, 12]. In particular, we target the Ethereum open-source ecosystem, comprising both official repositories (e.g., `ethereum/EIPs`) and community-driven projects, using the Web3BlockSet dataset with 391,596 Issues and Pull Requests (PRs) from 381 official and 4,500 community repositories. To characterize how ERC specifications translate into practice, we address the following Research Questions (RQ):

- **RQ1:** *Which ERCs are most frequently discussed and how does the discussion evolve over time?* Rationale: Identify which standards concentrate developer attention and how discussion volume shifts over time across repository types.
- **RQ2:** *In what types of technical situations are ERCs discussed?* Rationale: Characterize the technical nature of ERC-related discussions and identify which standards generate the highest proportion of each category.
- **RQ3:** *Which ERCs are frequently co-discussed?* Rationale: Identify which standards appear together in discussions, exposing practical inter-standard dependencies.
- **RQ4:** *What implementation observed practices emerge from practical ERC discussions, and what evidence grounds them?* Rationale: Systematize recurring implementation challenges from developer discussions into actionable practices.
- **RQ5:** *How do ERC proposals evolve through their lifecycle in official repositories?* Rationale: Analyze ERC progression through official status transitions, identifying patterns of refinement and stagnation.

RQ1-RQ3 establish the empirical foundation by identifying where (RQ1), what types of problems (RQ2), and which dependencies (RQ3) attract developer attention, thereby providing context for the implementation observed practices extracted in RQ4. This study advances empirical understanding of ERC adoption in four dimensions: (i) large-scale analysis of discussion frequency, temporal trends, inter-standard dependencies, discussion category distributions per ERC, and lifecycle trajectories in official repositories; (ii) a validated set of 146 practices grounded in specification violations and emergent developer behaviors, organized by ERC and macro-topic; (iii) a replicable pipeline that integrates repository mining with LLM-based analysis to extract ERC adoption patterns at scale; and (iv) a curated set of substantive ERC discussions from open-source development, made available for future research.

The remainder of this paper is organized as follows: Section 2 reviews related work; Section 3 describes the methodology; Section 4 presents the results for RQ1-RQ5; Section 5 discusses the implications of the findings; Section 6 addresses threats to validity; and Section 7 concludes the paper.

2 Related Work

Prior work has approached Ethereum smart contracts through Mining Software Repositories (MSR), static analysis, and formal verification, but few studies examine how developers discuss through Issues/PRs ERC standards in practice. Norvill *et al.* [23] proposed an automatic approach to infer which ERC standards a contract implements, analyzing only the bytecode. The technique extracts function selectors (4 bytes) and applies clustering, identifying that many contracts implement only parts of the standards. However, static analysis does not capture developers' deviations from specifications nor the difficulties they face.

Chen *et al.* [9] focused on detecting inconsistent behaviors in already deployed ERC-20 tokens. Their tool, TokenScope, contrasts three sources of information considering manipulation of data structures that store balances, calls to standard methods (such as `transfer`), and emission of standard events (such as `Transfer`). Applied to the entire blockchain, TokenScope found 3.2 million transactions with inconsistencies, involving 7,472 tokens. Manual analysis of 2,353 open-source tokens identified 11 causes, including missing events, embedded fees, and integer overflow attacks. This work detects if there is deviation, but does not explore the developers' discourse on why these deviations occur.

Oliva, Hassan, and Jiang [24] mined deployed contracts (2015-2018) and found that 0.05% of contracts receive 80% of transactions, most of them token managers. This study maps deployment concentration but not how developers discuss standard adherence. Moon and Park [22] measured conformance gaps directly in the top-100 ERC-20 token contracts. Vaccargiu et al. [36] examined a decade of ecosystem events and their impact on contributor dynamics in Ethereum. Fekih et al. [18] reviewed 19 studies on formal verification of ERC-based contracts (2019-2023), identifying challenges such as state explosion and environment modeling. Although fundamental, formal verification starts from explicit specifications and does not address how standards are interpreted in practice.

These works establish three perspectives relevant to this study. First, ERC adoption is concentrated in a small set of token contracts, most of which implement standards only partially [23, 24]. Second, deployed contracts present measurable inconsistencies with respect

to ERC specifications, with identifiable recurring causes [9]. Third, formal verification approaches begin from explicit specifications and cannot capture interpretive gaps that emerge in practice [18]. None of these recover the developer rationale behind deviations, considering why practitioners deviate, how they adapt standards to deployment constraints, and what edge cases the specifications leave unaddressed. This study is the first large-scale empirical investigation of ERC adoption through developer discussions, examining 391,596 Issues/PRs to characterize the gap between specification and practice manifests in the Ethereum open-source ecosystem.

3 Methodological Procedures

To conduct this study, we applied a Mining Software Repository (MSR) approach on the Web3BlockSet dataset. We selected this dataset because it provides a large amount of discussions originating from developers in relevant projects of the blockchain ecosystem, with 391,596 issues/PRs collected from 381 official repositories (*Provider*) and 4,500 community repositories (*Community*), covering the period from 2012 to 2026. Our research design consists of five key steps: (1) Automatic Identification of ERCs, (2) Semantic Validation and Categorization, (3) Observed Practice Extraction, (4) ERC Lifecycle Data Collection, and (5) Analytical Procedures. Figure 1 presents an overview of the methodological process.

3.1 Automatic Identification of ERCs

We scraped the official ERC list[2] to obtain the respective `numbe r` and `title` (for example, 20 and "Token Standard") of all 133 ERCs. Next, we applied regular expressions to the `raw_text` field, capturing variations such as `ERC20` , `ERC20`, and `ERC 20`, extending to all 133 tokens obtained. In the Web3BlockSet dataset, `raw_text` concatenates each issue/PR's title, body, and comments.

After normalization and removal of duplicates per record, we created the columns `ercs_mentioned` (ERCs mentioned in the issue/PR), `ercs_titles` (titles of the mentioned ERCs), and `ercs_ count` (count of mentioned ERCs). Before extraction, we filtered records with fewer than 15 tokens and texts originating from bots (identified by keywords such as `snyk`, `copilot`, `graphite`), preserving only human contributions with minimum content. Of the 391,421 unique issues/PRs in the corpus, this step retained **11,537 records** (3.0%) that mentioned at least one valid ERC.

3.2 Semantic Validation and Categorization

Given the volume remaining after filtering, we employed Gemini 2.5 Flash Lite (Google Generative AI API) on the `raw_text` field for three tasks: **(i) substance verification:** determine whether the discussion effectively deals with the mentioned ERC; **(ii) technical categorization:** into BUG, FEATURE, QUESTION, DOCUMENTATION, or DISCARD, based on Colavito et al. [10] and here adapted for the blockchain domain; and **(iii) brief justification of the judgment:** explaining the reasons for the classification assigned to the respective Issue/PR. Gemini 2.5 Flash Lite was chosen for its low latency, long-context capacity, and classification quality.

From these tasks, the columns `llm_is_substantive`, `llm_ca tegory`, and `llm_justification` were generated. Of the 11,537 candidates, 9,005 were classified as substantive (78.1%), forming the basis for all subsequent analyses. The prompt (presented in Box 3.2) explicitly instructs the model to recognize variations of references to the standard, such as `standardX`, in order to avoid false negatives in semantic validation. The results were deduplicated by record identifier before final enrichment, preventing inflated counts from partial reprocessings. This approach yields an initial classification without fine-tuning. Prior work already demonstrated that LLMs with prompt engineering achieve agreement levels comparable to human annotators on similar labeling tasks [1, 10], and we followed established prompt construction patterns [38].

Prompt used for classification via LLM

```
You are an expert in Ethereum smart contracts and ERC standards.
Analyze the following text extracted from a GitHub issue/PR.
Your task is to determine:
1. SUBSTANCE: Does the discussion substantively address an ERC
standard   (e.g., ERC-20, ERC-721, ERC-1155, etc.)? Consider
all format variations such as "ERC-X", "EIP-X", "standard X".
Non-technical or tangential mentions are not substantive.
2. CATEGORY: Classify the discussion into exactly one of:
  - BUG: Reports an error, unexpected behavior, vulnerability,
or problem.
  - FEATURE: Requests new functionality, enhancement, or
improvement.
  - QUESTION: Asks for help, clarification, or information.
  - DOCUMENTATION: Discusses docs, examples, or specification
ambiguities.
  - DISCARD: Not relevant, mentions ERC only tangentially, or
insufficient.
3. JUSTIFICATION: Provide a one-sentence justification.
Return your answer in the following JSON format:
{ "is_substantive": true/false,
  "category": "bug|feature|question|documentation|discard",
  "justification": "..."
}
Text: {raw_text}
```

Manual Validation. We adopted stratified sampling by category, with 95% confidence and a margin of error of 5%, applied over the total of classified issues/PRs, following established guidelines for sampling in Software Engineering (SE) research [2]. Based on it, we selected 400 samples for manual analysis. Within each repository category, we limited the selection to a maximum of 3 samples per repository, ensuring diversity of contexts. The selection was made with fixed seed `random_state=42`, ensuring reproducibility.

When a repository pool was insufficient to fill the category quota, we drew from unselected repositories preserving target proportions. Each record received the columns `agree_with_ai` and `human_ju stification` to capture annotator decisions. We calculated inter-annotator agreement metrics to assess the quality of the automatic classification. Results of percentages of agreement between manual classification and the one generated by the LLM are: feature (93.4%), question (89.2%), documentation (81.1%), and bug (78.2%).

Two researchers each independently reviewed one half of the 400 annotated samples and then cross-checked the other half. Disagreements identified during cross-checking were resolved through discussion until consensus, which was then recorded as the final label, ensuring reliability of the annotated dataset. The resulting subdataset contains the fields `row_id`, `is_substantive`, `categor`

[2] https://eips.ethereum.org/erc

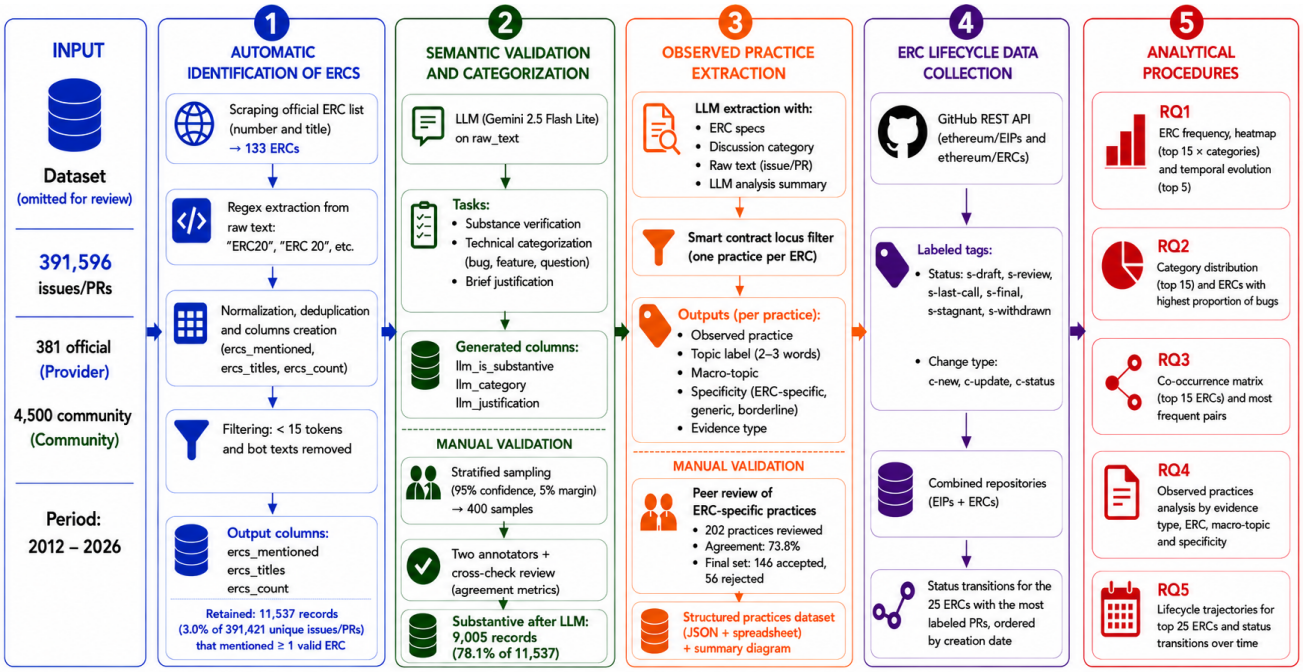

Figure 1: Overview of the five-step research design.

y, `justification`, `repository`, `repository_category`, `year`, `data_source`, `ercs_mentions`, `ercs_titles`, and `ercs_count`.

3.3 Observed Practice Extraction

We define an "*observed practice*" as an empirically grounded implementation directive derived from real-world developer discussions on GitHub. Hence, it captures either *(i)* a `spec_violation`: a behavior the ERC specification defines but that developers apply incorrectly in practice, or *(ii)* an `emergent_practice`: a behavior that arises in deployment contexts not anticipated by the specification. Unlike best practices inferred from specifications, observed practices are anchored in actual Issues and PRs, which makes them actionable for developers who implement or audit ERC-based contracts. Over the discussions validated in the previous step, we applied a structured LLM-based extraction also using the ge `mini-2.5-flash-lite` model. Unlike the first LLM application, which operates on raw discussion text alone, this step requires cross-referencing each discussion against the official ERC specification. For each discussion, the model received the discussion category, the raw text (issue/PR), the official specification of each mentioned ERC fetched directly from `eips.ethereum.org`, and an automated analysis generated in the semantic validation step.

The prompt included two annotated examples, one per evidence type, to guide output format and detail level. The full prompt is available in the replication package [33], omitted here due to space constraints. The prompt instructed the model to extract one observed practice per mentioned ERC, only when the discussion provided evidence of how ERC-defined behavior is broken or insufficient in real-world usage, in a way reading the specification alone would not make apparent. We integrated a *smart contract locus* filter into the prompt, requiring each accepted practice to modify the contract itself and excluding discussions resolved solely through client libraries, front-end applications, wallets, or off-chain services.

To allow for comparison and organization between different ERC standards in subsequent analyses, the prompt also assigned each observed practice a two- to three-word topic label, naming the ERC mechanism addressed (e.g., "operator authorization", "operator functionality"), and and then grouped these labels into broader macro-topic categories (e.g., "Account and Identity"), later reviewed by evaluators during peer validation. We then counted occurrences of each validated macro-topic per ERC (e.g., "Token Transfer Mechanics" spans 23 practices across ERC-20, ERC-721, and ERC-1155). Moreover, the prompt classified each practice by specificity (ERC-specific, generic, or borderline) to distinguish recommendations tied to ERC token semantics from general Solidity advice, ensuring the set of implementation observed practices remains useful for developers working with a specific standard.

Manual Validation. The 202 `erc_specific` practices were submitted to peer review. Unlike the classification validation in Section 3.2, where each item required only a binary substantive judgment, practice validation demanded cross-referencing the full issue/PR text against the official ERC specification, assessing evidentiary sufficiency, and evaluating the precision of the extracted directive. Given this complexity, Reviewer 1 and Reviewer 2 each assessed one half of the 202 entries. A third reviewer with over six years of smart contract experience assessed all 202 entries, serving as a common reference across both halves.

For each entry, reviewers examined the evidence columns (`evidence_type`, `macro_topic`, `spec_excerpt`, `issue_excerpt`), consulting the full issue/PR when the excerpt alone was insufficient,

and assessed whether the evidence supports the stated observed practice. We computed the agreement by pairing each primary reviewer with the expert on their respective half. Of the 202 practices, 138 were accepted by both reviewers (68.3%), 11 were rejected by both (5.4%), and 53 presented divergent judgments (26.2%), with an overall observed agreement of 73.8%. All 53 conflicts were resolved through asynchronous discussion until consensus, yielding a final set of 146 accepted and 56 rejected practices. The 89 `generic` and `borderline` practices were excluded before this review, as these practices lack direct grounding in ERC-specific mechanisms.

The output was consolidated into a JSON file indexed by ERC and exported as a spreadsheet, one row per practice. We produced a summary diagram (Figure 11) by counting occurrences of each validated macro-topic label per ERC, for instance, "Token Transfer Mechanics" accounts for 23 practices distributed across ERC-20, ERC-721, and ERC-1155, whereas "Allowance and Approval" concentrates 16 practices in ERC-20, with the remainder distributed between ERC-4626 and ERC-2612, yielding a cross-tabulation of practices across standards and topics. The full ERC-to-macro-topic mapping is available in the replication package [33].

3.4 ERC Lifecycle Data Collection

We used a different data collection method specifically to answer RQ5, since the Web3BlockSet dataset had few data originating from the official `ethereum/EIPs` and `ethereum/ERCs` repositories, as it had a rigorous filtering process. We therefore collected data directly from both repositories via the GitHub REST API, focusing on Issues/PRs labeled with standardized status tags (illustrated in Figure 2): `s-draft`, `s-review`, `s-last-call`, `s-final`, `s-stagnant`, `s-withdrawn`, and change type tags `c-new`, `c-update`, `c-status`.

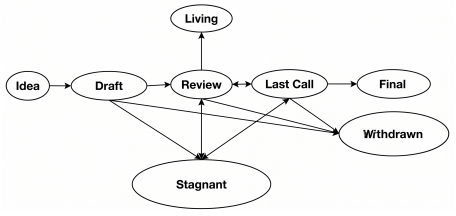

Figure 2: Standardization process for EIPs in all tracks [5].

We initially targeted `ethereum/ERCs` alone, but older ERCs such as ERC-20 migrated from `ethereum/EIPs` in 2023 and are rarely discussed in the newer repository. To fully represent lifecycle activity, we combined both repositories and reconstructed status-transition sequences for the 25 ERCs with the most labeled PRs.

3.5 Analytical Procedures

Table 1 maps each RQ to the corresponding analytical procedure.

4 Results

Results are organized according to the five RQ as follows.

Table 1: Mapping of RQs to analytical procedures.

RQ	Analytical Procedure
RQ1	We performed absolute frequency analysis of the mentioned ERCs, identified the top 15, and built a heatmap crossing the top 15 ERCs with the top 10 repository categories already defined by the Web3BlockSet dataset. Additionally, we traced the temporal evolution (2012-2026) of discussions for the top 5 ERCs identified.
RQ2	We analyzed the distribution of the discussion category (`llm_category`) for the top 15 ERCs and identified the ERCs with the highest proportion of bug reports.
RQ3	We focused on records mentioning multiple ERCs (`ercs_count` > 1) and built a co-occurrence matrix for the top 15 ERCs.
RQ4	We analyzed the set of implementation observed practices constructed in the previous step, examining the distribution of practices by evidence type, ERC, and macro-topic. We report the total number of practices, their breakdown by specificity classification (`erc_specific`, `generic`, `borderline`), and the cross-ERC distribution across macro-topics.
RQ5	For each of the 25 ERCs with the highest number of labeled PRs collected in the previous step, we reconstructed the sequence of status transitions ordered by creation date and resulting lifecycle trajectories.

4.1 RQ1: Which ERCs Are Most Frequently Discussed and How Does the Discussion Evolve over Time?

Figure 3 shows ERC-20 leading in *Metadata Repository* (97%) and *Smart Contract Library* (40%), according categories filtered from Web3BlockSet dataset. ERC-721 ranks second (717 in *Smart Contract Library*, 28%), followed by ERC-1155 (219, 9%). The remaining ERCs in the top 15 present frequencies below 100 discussions per category.

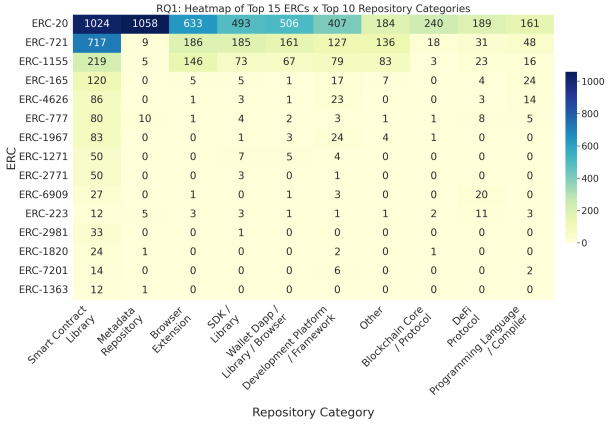

Figure 3: Freq. of the top 15 ERCs per repository category.

In terms of repositories, the 15 most active in ERC discussions include openzeppelin-contracts[3] (1,262 discussions, 14% of the total), assets[4] (around 900, 10%), and metamask-extension[5] (around 800, 9%). As we can see, ERC discussion is concentrated in a small core of influential projects. For practically all ERCs in the top 15, the *provider* origin (official repositories) accounts for more than 90% of

[3]https://github.com/openzeppelin/openzeppelin-contracts
[4]https://github.com/trustwallet/assets
[5]https://github.com/metamask/metamask-extension

discussions. The only exception is ERC-4626, with approximately 20% of discussions in *community* repositories, reflecting the DeFi community's interest in standardizing yield-bearing vaults.

Figure 4 presents the annual evolution of the number of discussions for the top 5 ERCs. ERC-20 grows consistently from 2016, reaches a peak in 2021 (1,193 discussions), and experiences a downward trend from 2022 onward. ERC-721 remains near zero until 2017, and experiences accelerated growth between 2020 and 2022, with a peak of 496 discussions in 2022, at the height of the NFT market. ERC-1155 follows a similar trajectory to ERC-721, but with lower volumes (peak of 233 in 2022). In turn, ERC-165 and ERC-4626 remain at low levels, but with gradual growth from 2021.

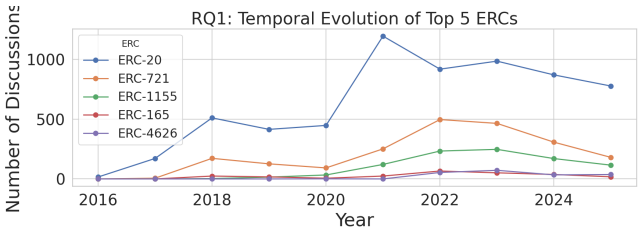

Figure 4: Temporal evolution of discussions (top 5 ERCs).

4.2 RQ2: In What Types of Technical Situations Are ERCs Discussed?

Figure 5 presents the proportional distribution of categories for the top 15 ERCs. The *bug* category is the most frequent in almost all standards, accounting for 50% to 62% of discussions. ERC-2771 has the highest *bug* proportion among the top 15 (62%). ERC-1363 is the only one with a high *documentation* proportion (48%), indicating its specification demands clarification more than it generates errors. ERC-6909 stands out for its high proportion of *feature* (48%), consistent with its character as an emerging standard still under extension.

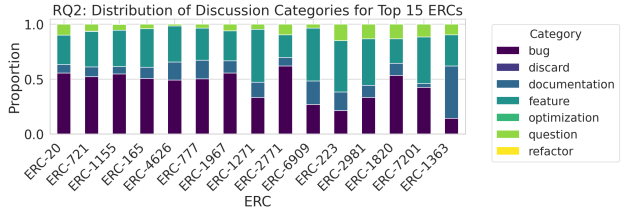

Figure 5: Distribution of discussion categories (top 15 ERCs).

Figure 6 presents the 15 ERCs with the highest proportion of discussions classified as *bug*, considering the full set of identified ERCs. ERC-820 leads with 84%, followed by ERC-2309 (80%), ERC-191 (71%), and ERC-6093 (70%). Among the standards with higher absolute volume, ERC-20 (56%) and ERC-1155 (55%) also appear on this list, indicating that both obscure standards and the most popular ones concentrate a high proportion of problems.

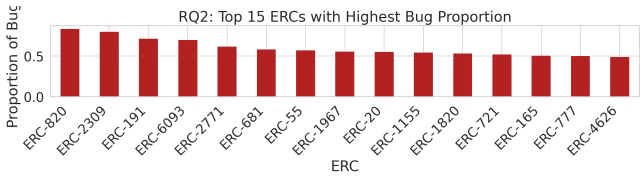

Figure 6: Highest proportion of discussions classified as *bug* (top 15 ERCs).

4.3 RQ3: Which ERCs Are Frequently Co-Discussed?

Figure 7 presents the co-occurrence matrix for the top 15 ERCs. The most frequent pair is ERC-20/ERC-721 (654 co-occurrences), reflecting the foundational relationship between FT and NFTs. The second most frequent pair is ERC-721/ERC-1155 (424), consistent with ERC-1155's role in unifying the models of both in a single contract. The third pair, ERC-20/ERC-1155 (267), completes a triangle of co-occurrences among the three most relevant token standards. ERC-165 frequently co-occurs with ERC-20 (85) and ERC-721 (106), evidencing its transversal role as an interface detection mechanism required by both. ERC-777 co-occurs with ERC-1820 (23), consistent with the formal dependency defined in ERC-777's specification.

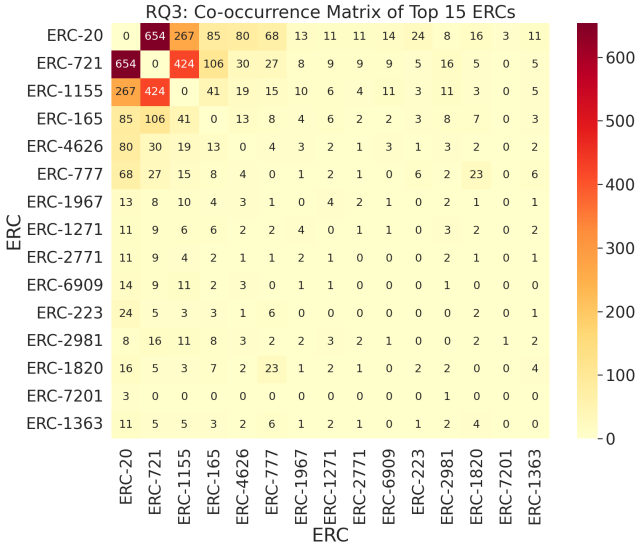

Figure 7: Co-occurrence matrix of the top 15 ERCs.

4.4 RQ4: What Implementation Observed Practices Emerge from Practical ERC Discussions, and What Evidence Grounds Them?

The extraction process initially yielded 291 candidate observed practices. Of these, 89 `generic` and `borderline` practices were excluded before peer review, as they fall outside the scope of ERC-specific implementation guidance (see Section 3). The remaining

202 `erc_specific` practices were submitted to peer review, 56 were rejected by consensus, yielding a final validated set of 146 observed practices drawn from 133 unique source issues. All analyses below and the guideline set made available in the replication package refer exclusively to these 146 validated practices.

Figure 8 shows the distribution of observed practices by discussion category and evidence type. Bug discussions are the largest source of observed practices (91 total, 44 `emergent_practice` and 47 `spec_violation`), followed by feature requests (37), questions (7), and documentation discussions (3). `spec_violation` practices are concentrated almost entirely in *bug* discussions, suggesting that violations surface as defects, not design discussions.

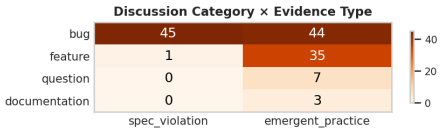

Figure 8: Distribution of extracted observed practices by discussion category and evidence type.

Figure 9 shows each ERC's contribution to shared macro-topics, ordered from most to least cross-cutting. *Contract Interaction* is the most cross-cutting macro-topic, spanning 6 ERCs, followed by *Advanced Mechanics* (5 ERCs), and *Signatures and Verification* and *Implementation Details* (4 ERCs each). *Token Transfer Mechanics*, the macro-topic with the highest absolute count (23 practices), spans only 3 ERCs (ERC-20, ERC-721, and ERC-1155), reflecting a high concentration of transfer-related issues in the basic token standards.

Aligned with previous results, ERC-20 dominates most macro-topics (75 of 146 practices, 51.4%), consistent with its role as the foundational token standard. ERC-721 (27 practices) concentrates in NFT-specific categories such as *Token Lifecycle Management* and *Token Transfer Mechanics*. ERC-1155 (7 practices) distributes across *Token Transfer Mechanics*, *Token Properties*, and *Contract Interaction*. ERC-4626 concentrates observed practices in *Advanced Mechanics* and *Allowance and Approval*, reflecting the particular risks of yield-bearing vault implementations. Particularly, *Signatures and Verification*, *Proxy Standards*, and *Account and Identity* are the only macrotopics in which ERC-20 is entirely absent.

The full set of 146 validated observed practices is available in the replication package as a filterable spreadsheet. Each entry includes the observed practice text (`observed_practice`), the evidence type, a specification excerpt (`spec_excerpt`), a discussion excerpt (`issue_excerpt`), the macro-topic, and a direct link to the issue/PR.

4.5 RQ5: How Do ERC Proposals Evolve through Their Lifecycle in Official Repositories?

Figure 10 shows lifecycle trajectories of the 25 most active ERCs, ordered by first labeled PR. Each point is a PR, colored by status and shaped by change type (*new proposal*, *content update*, or *status change*). The dashed line marks the creation of the `ethereum/ERCs` repository in January 2024, separating earlier activity recorded in `ethereum/EIPs` from subsequent activity in the repository.

The figure indicates two distinct behavioral patterns. Older ERCs (ERC-5507, ERC-5564, ERC-4337, ERC-721, ERC-20) display longer

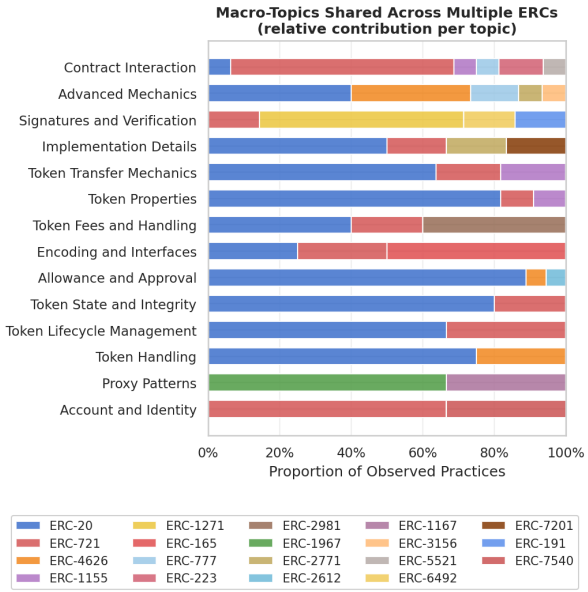

Figure 9: Relative contribution of each ERC to macro-topics shared across two or more standards.

activity spans, with repeated content-update PRs preceding and following status transitions, indicating that specification refinement is iterative and driven by sustained community engagement. Several of these proposals accumulate `s-draft` and `s-review` activity over years before advancing to `s-last-call` or `s-final`. In contrast, more recent proposals (ERC-7656, ERC-7208, ERC-7578, ERC-7730, ERC-7828, ERC-8109) concentrate activity in a narrow time window, mostly after the repository migration, and predominantly carry `s-draft` status, indicating that they are still in early stages of community review. A subset of proposals in the middle of the timeline (e.g., ERC-6059, ERC-5485, ERC-6551) show sparse activity followed by visible gaps, consistent with periods of stagnation. The migration to `ethereum/ERCs` introduced no observable discontinuity for proposals already in advanced stages.

Particularly, ERC-20 and ERC-721 enter the observation window already at `s-final`, with all subsequent activity consisting of content-update PRs. Additionally, 500 of the 1,432 recorded events (35%) are new-proposal submissions, reflecting continued expansion of the ERC ecosystem throughout the observation period.

5 Discussion

Adoption Patterns and Temporal Dynamics (RQ1). Our results confirm that ERC adoption is highly concentrated, as ERC-20, ERC-721, and ERC-1155 account for the majority of discussions, further concentrated in a few repositories such as openzeppelin-contracts, in line with prior work on the centralization of activity around token standards [24]. Although ERC-20 leads in absolute practice count (75 of 146), its practice density is comparable across the three standards, one per 84 discussions for ERC-20, one per 78 for ERC-721, one per 134 for ERC-1155, indicating that its dominance reflects volume, not a higher rate of implementation challenges.

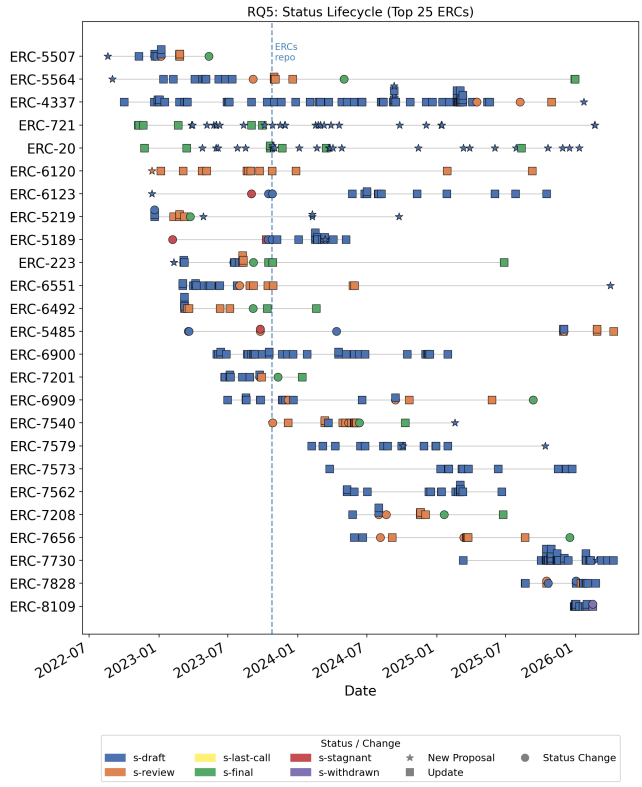

Figure 10: Status lifecycle trajectories of the 25 most active ERCs across official repositories.

The temporal evolution presents peaks aligned with *"DeFi Summer"* (2021) and the *"NFT boom"* (2022), indicating that standard adoption is reactive to market cycles rather than to specification maturity. The emergence of ERC-4626 appears as an exception to this concentration, since, motivated by the lack of standardization of tokenized vaults [32], the standard attracted substantial community engagement concurrent with its formalization process, as 27% of all ERC-4626 substantive discussions in this corpus (53 of 198) are dated to 2022, the year the standard reached `s-final` status. A pattern of community discussion outpacing formal stabilization that we also observe in RQ5 and that produced five of its seven extracted practices in Advanced Mechanics.

Answer to RQ1: *ERC-20 (1,024), ERC-721 (717), and ERC-1155 (219) dominate discussions, concentrated in openzeppelin-contracts (1,262, 14%). Volume peaks align with DeFi Summer (ERC-20, 1,193 in 2021) and the NFT boom (ERC-721, 496 in 2022). ERC-4626 is the sole exception, with 27% of its discussions preceding its* `s-final` *status.*

Technical Discussion Categories (RQ2) Standard implementation, i.e., any smart contract that implements or claims to implement a given ERC, whether a reference library (e.g., OpenZeppelin) or a community contract, proves highly error-prone, as evidenced by the predominance of *bug* discussions across nearly all top ERCs [40]. This pattern is also consistent with earlier detections of inconsistencies in token contracts [9]. In addition, lesser-known ERCs (ERC-820, ERC-2309, ERC-191) present the steepest *bug* rates, demonstrating that complex or poorly written specifications generate proportionally more problems than popularity. For example, ERC-820 [3] was deprecated after a Solidity update broke its compatibility with ERC-165, forcing early adopters to migrate to ERC-1820. This trajectory mirrors the stagnation pattern in RQ5, as without sustained community engagement, proposals leave implementers without stable guidance. ERC-2309 introduced a dual-event model for batch transfers that conflicts with standard event-based ownership tracking [26]. Finally, ERC-191 requires precise coordination of a version byte and validator address to prevent signature replay, creating recurring validation pitfalls across wallet implementations [35].

ERC-1363 [21], a payable token with a callback model, concentrates discussions on *documentation*, reflecting an insufficiently clear specification. In turn, ERC-6909 [31], a minimal multi-token interface, draws a high share of *feature* requests, consistent with a standard still being shaped by community extensions. Thus, category distributions serve as diagnostic signals, where high bug rates point to underspecified standards, high documentation rates to ambiguous prose, and high feature rates to incomplete coverage.

Answer to RQ2: *Bug discussions account for 50-62% of discussions across the top 15 ERCs. ERC-820 (83%), ERC-2309 (80%), and ERC-191 (71%) show the highest bug proportions overall. ERC-1363 stands out for documentation (48%) and ERC-6909 for feature requests (48%).*

Co-occurrence and Inter-standard Dependencies (RQ3). We have found that co-occurrence patterns expose practical dependencies undocumented in the EIPs themselves [23]. The ERC-20/ERC-721/ERC-1155 triangle reflects the foundational layering of the token ecosystem. ERC-165 [30] co-occurs with nearly all other standards, confirming its role as a prerequisite interface check. The ERC-777/ERC-1820 pair mirrors an explicit specification requirement, as ERC-777 consults ERC-1820 to verify recipient capabilities [4, 14], a case of formally coupled standards whose co-occurrence was therefore expected. These dependencies have direct security implications, as vulnerabilities propagate across contracts sharing interface assumptions. This risk is illustrated by the TreasureDAO exploit, where mixing ERC-1155 and ERC-721 within the same contract caused logical confusion and the theft of over 100 NFTs [6].

Answer to RQ3: *The most frequent pairs are ERC-20/ERC-721 (654), ERC-721/ERC-1155 (424), and ERC-20/ERC-1155 (267). ERC-165 co-occurs broadly with ERC-721 (106) and ERC-20 (85). ERC-777/ERC-1820 (23) reflects a formally specified dependency.*

Observed Practices (RQ4). The observed practices provide aspects that are not fully specified in the standards. Figure 11 summarizes the full set, grouping practices by ERC and macro-topic. ERC-20's thematic density contrasts with the narrower, security- and encoding-focused clusters of ERC-191 and ERC-712, reflecting how underspecification scope varies across standards. Among `spec_violation` practices, event emission requirements recur most often, particularly omission of the `Transfer` event in ERC-20 and ERC-721, confirming the pattern documented by Chen et al. [9].

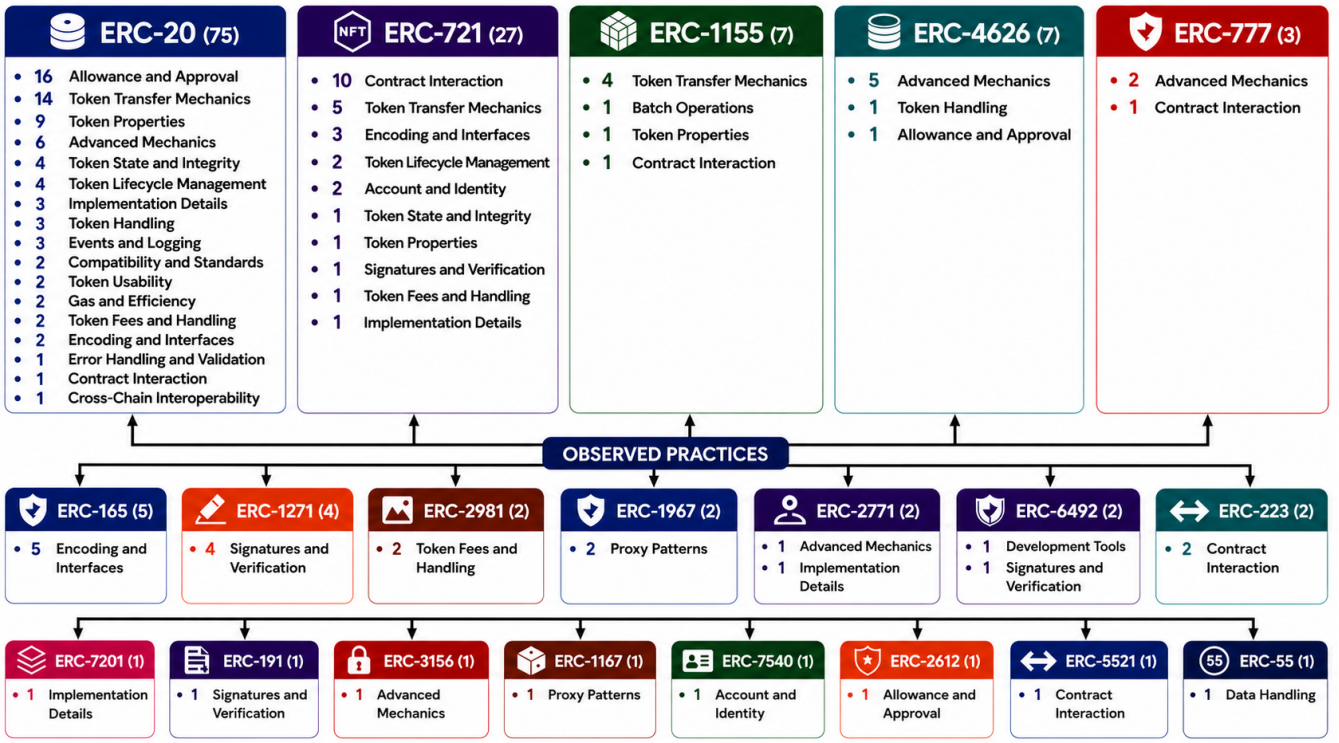

Figure 11: Representation of the set of observed practices.

The near-equal split between `spec_violation` and `emergent_practice` among practices extracted from bug discussions (49 vs. 44) carries a distinct implication for each evidence type. Spec violations, behaviors the ERC defines but developers apply incorrectly, suggest a failure of *specification clarity*, i.e, the requirement exists but is not operationalized. Emergent practices, behaviors arising from deployment contexts not anticipated by the specification, indicate a failure of *specification coverage*, as the gap itself is the problem. Given that 64% of all 146 validated observed practices are emergent, the primary risk in ERC adoption is not non-compliance with existing requirements but deployment in contexts the specification did not foresee, showing that real-world usage examples and known edge cases are as important as formal interface definitions.

Table 2 illustrates the two evidence types with concrete entries from the validated set. The first example is a `spec_violation`, the ERC-20 specification states that minting and burning operations SHOULD emit a `Transfer` event from or to the zero address, yet the `enzymefinance/protocol` repository contained functions that modified balances without triggering any event. A developer auditing a vault contract can use this practice as a checklist item, such as "*verify that every code path that changes a balance also emits the corresponding event*". The second example is an `emergent_practice`, where the ERC-20 specification only advises client interfaces to reset allowances to zero before setting a new value, it does not impose this on the contract itself. In practice, however, this gap surfaced repeatedly because tokens such as USDT revert when a non-zero allowance is overwritten with another non-zero value.

Table 2: Examples of observed practices.

Field	Value
type/ERC	Example 1: spec_violation (ERC-20)
macro_topic/topic	Events and Logging/event emission
practice	Emit `Transfer` events to and from the zero address for token creation and annihilation to ensure complete off-chain tracking of all balance changes.
spec_excerpt	*"A token contract which creates new tokens SHOULD trigger a Transfer event with the _from address set to 0x0 when tokens are created."*
issue_excerpt	*"The `createShares` and `annihilateShares` functions modify the balances of users. They should emit corresponding Transfer events to and from the 0x0 address to comply with ERC-20 and make transfers easier to track off-chain."*
type/ERC	Example 2: emergent_practice (ERC-20)
macro_topic/topic	Allowance and Approval/token allowance
practice	Ensure that token approvals first set the allowance to zero before setting a new non-zero value to prevent approval failures with specific tokens.
spec_excerpt	*"Clients SHOULD make sure to create user interfaces in such a way that they set the allowance first to 0 before setting it to another value for the same spender."*
issue_excerpt	*"Some tokens like USDT do not allow approving an amount $M > 0$ when an existing amount $N > 0$ is already approved [...] A simple fix would probably be to just clear any residual allowance at the end of a call."*

A developer would not derive this requirement from reading the standard alone, as the practice is anchored in failures reported.

Moreover, the absence of ERC-20 from *Signatures and Verification*, *Proxy Patterns*, and *Account and Identity* reflects architectural concerns orthogonal to FT semantics, as the first covers cryptographic signature validation (ERC-191, ERC-1271, ERC-6492), the second, upgradeable proxy contracts (ERC-1967, ERC-1167), and the third, identity and asynchronous deposit flows (ERC-721, ERC-7540). Specification gaps in these topics emerge only in standards specifically designed for such purposes, precisely the implementation concerns that the FT model leaves unspecified.

Answer to RQ4: *146 practices were validated from 133 source issues (observed agreement of 73.8%), 64%* `emergent_practice` *and 36%* `spec_violation`*. Bug discussions yielded the most practices (91). ERC-20 accounts for 75 practices (51.4%), with Token Transfer Mechanics as the densest macro-topic (23 practices).*

Lifecycle Trajectories (RQ5). Proposals that accumulate the most content-update PRs before `s-final` are also those with the most sustained community discussion. Mature ERCs (ERC-20, ERC-721, ERC-4337) show iterative refinement spread over years, whereas recent proposals concentrate activity in short bursts, reflecting the difference between standards shaped by sustained deployment experience and those still seeking community validation. The 2024 migration introduced no observable disruption for proposals already in advanced stages. Proposals with sparse activity followed by extended gaps represent a loss of momentum before reaching actionable status and leave developers without a stable reference.

Answer to RQ5: *Mature ERCs (ERC-20, ERC-721, ERC-4337) show iterative refinement across years before reaching* `s-final`*. Recent proposals (ERC-7656, ERC-7730, ERC-8109) concentrate activity in short bursts at* `s-draft`*. A middle group (ERC-6059, ERC-5485, ERC-6551) shows visible stagnation gaps.*

6 Threats to Validity

We discuss the main threats to the validity of this study following the classification proposed by Wohlin et al. [39] and the Repository Mining Empirical guidelines from Empirical Standards for SE [29].

Construct Validity. We identified mentions of the ERC through regular expressions, which may result in the omission of references in non-standardized formats. To reduce false negatives, the LLM step was instructed to accept variations such as "standard X" and "ERC-...". Future work will assess sensitivity to this choice via narrower/broader regex variants. Both LLM steps relied on a single model, which may introduce biases [10]. Additionally, filtering out records with fewer than 15 tokens reduced noise at the cost of potentially excluding short but substantively important comments (e.g., brief bug confirmations). To mitigate these, we manually validated a stratified sample of 400 issues/PRs, and the LLM classification achieved agreement rates above 78% for all categories.

Internal Validity. Discussions were truncated to 3,000 characters due to latency and cost constraints of the Gemini 2.5 Flash Lite API. The prompt prioritized the initial body of each issue/PR, where the core technical problem is typically stated. The peer review of observed practices achieved an overall observed agreement of 73.8%, with all 53 conflicts resolved through discussion. The divergence in acceptance rates between the two primary reviewers (91.2% vs. 71.6%) reflects genuine interpretive uncertainty about what constitutes sufficient evidence for an observed practice, and represents a threat to the internal validity of RQ4. Readers should treat the 146 validated practices as a conservative lower bound on the set of actionable behaviors present in the corpus. The 89 `generic` and `borderline` practices excluded before peer review represent a scope limitation. Their validity as potentially useful general guidance was not assessed, so findings on implementation challenges are bounded to ERC-specific practices.

External Validity. Our corpus covers only public GitHub issues and PRs, excluding private repositories, audit reports, and forums, potentially introducing selection bias toward publicly visible problems [20]. We therefore report all findings as characteristic of the open-source, public discussion landscape. Additionally, a small number of highly active repositories dominate the dataset. `openzeppelin-contracts` alone accounts for 1,262 of the 9,005 substantive ERC discussions (14%), and for 62% of all discussions within the Smart Contract Library category. A portion of the observed practices may therefore reflect OpenZeppelin's implementation conventions and contribution patterns rather than ERC-level challenges encountered by the broader developer population.

Conclusion Validity. We recognize associations between temporal peaks and external ecosystem milestones are observational. Accordingly, we avoid causal language and frame the observed alignments (DeFi Summer, NFT boom) as descriptive, visually identified patterns rather than statistically validated relationships.

7 Final Remarks

The gap between what ERC specifications prescribe and how developers implement them in practice poses a concrete risk to the Ethereum ecosystem, manifesting as vulnerabilities, financial losses, and undocumented adaptations. We investigated ERC adoption empirically across 391,596 Issues/PRs in the Ethereum open-source ecosystem. The main findings are: *(i)* ERC-20, ERC-721, and ERC-1155 dominate discussions in a few repositories; *(ii)* temporal peaks align with market cycles; *(iii)* bugs predominate, with lesser-known ERCs showing the highest rates; *(iv)* co-occurrence patterns expose implicit inter-standard dependencies; *(v)* 146 observed practices ground recurring implementation challenges in developer evidence; and *(vi)* mature standards require sustained engagement, while stagnating proposals leave developers without stable guidance.

From a research perspective, this work establishes that the ERC adoption gap is measurable and structured, providing a methodology and dataset extensible to other blockchain ecosystems or applied to study standard evolution longitudinally. From a practitioner perspective, the observed practices offer actionable directives for developers implementing and auditing ERC-based contracts.

Future work will scale the extraction pipeline to all 9,005 substantive discussions, examine resolution time and comment volume, compare findings with other blockchain ecosystems, and enrich each observed practice with compilable code examples to make the set more directly actionable for implementers and auditors.

Artifact Availability

To promote transparency and reproducibility, all study materials are openly available in our public repository [33].

References

[1] Gabriel Aracena, Kyle Luster, Fabio Santos, Igor Steinmacher, and Marco Aurelio Gerosa. 2024. Applying large language models to issue classification. In *Proceedings of the Third ACM/IEEE International Workshop on NL-based Software Engineering*. 57–60.

[2] Sebastian Baltes and Paul Ralph. 2022. Sampling in software engineering research: A critical review and guidelines. *Empirical Software Engineering* 27, 4 (2022), 94.

[3] Jordi Baylina and Jacques Dafflon. 2018. ERC-820: Pseudo-introspection Registry Contract. Ethereum Improvement Proposals, no. 820. https://eips.ethereum.org/EIPS/eip-820

[4] Jordi Baylina and Jacques Dafflon. 2019. ERC-1820: Pseudo-introspection Registry Contract. Ethereum Improvement Proposals. https://eips.ethereum.org/EIPS/eip-1820

[5] Martin Becze et al. 2015. EIP-1: EIP Purpose and Guidelines. Ethereum Improvement Proposals. https://eips.ethereum.org/EIPS/eip-1

[6] Beosin. 2022. Beosin's Analysis of the Arbitrum-based TreasureDAO Exploit. Beosin Blog. https://vaas.beosin.com/resources/beosin-analysis-of-the-arbitrum-based-treasuredao-exploit

[7] Zach Burks, James Morgan, Blaine Malone, and James Seibel. 2020. ERC-2981: NFT Royalty Standard. Ethereum Improvement Proposals. https://eips.ethereum.org/EIPS/eip-2981 Ethereum Improvement Proposals.

[8] Vitalik Buterin. 2014. A next-generation smart contract and decentralized application platform. Ethereum White Paper. https://ethereum.org/en/whitepaper/

[9] Ting Chen, Yufei Zhang, Zihao Li, Xiapu Luo, Ting Wang, Rong Cao, Xiuzhuo Xiao, and Xiaosong Zhang. 2019. Tokenscope: Automatically detecting inconsistent behaviors of cryptocurrency tokens in ethereum. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*. 1503–1520.

[10] Giuseppe Colavito, Filippo Lanubile, Nicole Novielli, and Luigi Quaranta. 2024. Leveraging gpt-like llms to automate issue labeling. In *Proceedings of the 21st International Conference on Mining Software Repositories*. 469–480.

[11] Valerio Cosentino, Javier L Cánovas Izquierdo, and Jordi Cabot. 2017. A systematic mapping study of software development with GitHub. *IEEE Access* 5 (2017), 7173–7192.

[12] Saori Costa, Matheus Paixao, Igor Steinmacher, Pamella Soares, Allysson Allex Araújo, and Jerffeson Souza. 2024. Sociotechnical dynamics in open source smart contract repositories: An exploratory data analysis of curated high market value projects. In *Proceedings of the 20th International Conference on Predictive Models and Data Analytics in Software Engineering*. 22–31.

[13] Paul Cuffe. 2018. The role of the erc-20 token standard in a financial revolution: the case of initial coin offerings. In *IEC-IEEE-KATS Academic Challenge*. IEC-IEEE-KATS.

[14] Jacques Dafflon, Jordi Baylina, and Thomas Shababi. 2017. ERC-777: Token Standard. Ethereum Improvement Proposals, no. 777. https://eips.ethereum.org/EIPS/eip-777

[15] Ajoy Das, Gias Uddin, and Guenther Ruhe. 2022. An empirical study of blockchain repositories in github. In *Proceedings of the 26th International Conference on Evaluation and Assessment in Software Engineering*. 211–220.

[16] Giuseppe Destefanis, Michele Marchesi, Marco Ortu, Roberto Tonelli, Andrea Bracciali, and Robert Hierons. 2018. Smart contracts vulnerabilities: a call for blockchain software engineering?. In *2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*. IEEE, 19–25.

[17] William Entriken, Dieter Shirley, Jacob Evans, and Nastassia Sachs. 2018. ERC-721: Non-Fungible Token Standard. Ethereum Improvement Proposals. https://eips.ethereum.org/EIPS/eip-721

[18] Rim Ben Fekih, Mariam Lahami, Salma Bradai, and Mohamed Jmaiel. 2025. Formal verification of erc-based smart contracts: A systematic literature review. *IEEE Access* 13 (2025), 11396–11422.

[19] Francisco Giordano, Hadrien Croubois, Ernesto García, and Eric Lau. 2023. ERC-7201: Namespaced Storage Layout. Ethereum Improvement Proposals. https://eips.ethereum.org/EIPS/eip-7201

[20] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M German, and Daniela Damian. 2014. The promises and perils of mining github. In *Proceedings of the 11th working conference on mining software repositories*. 92–101.

[21] Vittorio Minacori. 2018. ERC-1363: Payable Token. Ethereum Improvement Proposals, no. 1363. https://eips.ethereum.org/EIPS/eip-1363

[22] Hyeon-Ah Moon and Sooyong Park. 2022. Conformance evaluation of the top-100 Ethereum token smart contracts with Ethereum Request for Comment-20 functional specifications. *IET Software* 16, 2 (2022), 233–249.

[23] Robert Norvill, Beltran Fiz, Radu State, and Andrea Cullen. 2019. Standardising smart contracts: Automatically inferring ERC standards. In *2019 IEEE International conference on blockchain and cryptocurrency (ICBC)*. IEEE, 192–195.

[24] Gustavo A Oliva, Ahmed E Hassan, and Zhen Ming Jiang. 2020. An exploratory study of smart contracts in the Ethereum blockchain platform. *Empirical Software Engineering* 25, 3 (2020), 1864–1904.

[25] Santiago Palladino, Francisco Giordano, and Hadrien Croubois. 2019. ERC-1967: Proxy Storage Slots. Ethereum Improvement Proposals. https://eips.ethereum.org/EIPS/eip-1967

[26] Sean Papanikolas. 2019. ERC-2309: ERC-721 Consecutive Transfer Extension. Ethereum Improvement Proposals, no. 2309. https://eips.ethereum.org/EIPS/eip-2309

[27] Simone Porru, Andrea Pinna, Michele Marchesi, and Roberto Tonelli. 2017. Blockchain-oriented software engineering: challenges and new directions. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, 169–171.

[28] Witek Radomski, Andrew Cooke, Philippe Castonguay, James Therien, Eric Binet, and Ronan Sandford. 2018. ERC-1155: Multi Token Standard. Ethereum Improvement Proposals. https://eips.ethereum.org/EIPS/eip-1155

[29] Paul Ralph, Nauman bin Ali, Sebastian Baltes, Domenico Bianculli, Jessica Diaz, Yvonne Dittrich, Neil Ernst, Michael Felderer, Robert Feldt, Antonio Filieri, et al. 2020. Empirical standards for software engineering research. *arXiv preprint arXiv:2010.03525* (2020).

[30] Christian Reitwießner, Nick Johnson, Fabian Vogelsteller, Jordi Baylina, Konrad Feldmeier, and William Entriken. 2018. ERC-165: Standard Interface Detection. Ethereum Improvement Proposals. https://eips.ethereum.org/EIPS/eip-165

[31] JT Riley, Dillon, Sara Reynolds, Vectorized, and Neodaoist. 2023. ERC-6909: Minimal Multi-Token Interface. Ethereum Improvement Proposals. https://eips.ethereum.org/EIPS/eip-6909

[32] Joey Santoro, t11s, Jet Jadeja, Alberto Cuesta Cañada, and Señor Doggo. 2021. ERC-4626: Tokenized Vault Standard. Ethereum Improvement Proposals. https://eips.ethereum.org/EIPS/eip-4626

[33] Pamella Soares, Airlon Silva Filho, Raissa Rodrigues, Allysson Allex Araújo, and Jerffeson Souza. 2026. Repository of supplementary material for "From Specification to Practice: An Empirical Study of Ethereum ERC Standard Adoption in Open Source Projects". https://zenodo.org/records/19989812

[34] Miroslav Stefanović, Đore Pržulj, Sonja Ristić, Darko Stefanović, and Danilo Nikolić. 2022. Smart contract application for managing land administration system transactions. *IEEE Access* 10 (2022), 39154–39176.

[35] Martin Holst Swende and Nick Johnson. 2016. ERC-191: Signed Data Standard. Ethereum Improvement Proposals, no. 191. https://eips.ethereum.org/EIPS/eip-191

[36] Matteo Vaccargiu, Sabrina Aufiero, Cheick Ba, Silvia Bartolucci, Richard Clegg, Daniel Graziotin, Rumyana Neykova, Roberto Tonelli, and Giuseppe Destefanis. 2025. Mining a decade of event impacts on contributor dynamics in Ethereum: A longitudinal study. In *2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR)*. IEEE, 552–563.

[37] Fabian Vogelsteller and Vitalik Buterin. 2015. ERC-20: Token Standard. Ethereum Improvement Proposals. https://eips.ethereum.org/EIPS/eip-20

[38] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C Schmidt. 2023. A prompt pattern catalog to enhance prompt engineering with ChatGPT. arXiv.

[39] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, Anders Wesslén, et al. 2012. *Experimentation in software engineering*. Vol. 236. Springer.

[40] Weiqin Zou, David Lo, Pavneet Singh Kochhar, Xuan-Bach Dinh Le, Xin Xia, Yang Feng, Zhenyu Chen, and Baowen Xu. 2019. Smart contract development: Challenges and opportunities. *IEEE transactions on software engineering* 47, 10 (2019), 2084–2106.