Folklore in Software Engineering: A Definition and Conceptual Foundations

EDUARD PAUL ENOIU and JEAN MALM, Mälardalen University, Sweden
GREGORY GAY, Chalmers University of Technology and University of Gothenburg, Sweden

We explore the concept of folklore within software engineering, drawing from folklore studies to define and characterize narratives, myths, rituals, humor, and informal knowledge that circulate within software development communities. Using a literature review and thematic analysis, we curated exemplar folklore items (e.g., beliefs about where defects occur, the 10x developer legend, and technical debt). We analyzed their narrative form, symbolic meaning, occupational relevance, and links to knowledge areas in software engineering. To ground these concepts in practice, we conducted semi-structured interviews with 12 industrial practitioners in Sweden to explore how such narratives are recognized or transmitted within their daily work and how they affect it. Synthesizing these results, we propose a working definition of software engineering folklore as informally transmitted, traditional, and emergent narratives and heuristics enacted within occupational folk groups that shape identity, values, and collective knowledge. We argue that making the concept of software engineering folklore explicit provides a foundation for subsequent ethnography and folklore studies and for reflective practice that can preserve context-effective heuristics while challenging unhelpful folklore.

CCS Concepts: • **Software and its engineering** → **Software development process management**; **Software development methods**; **Programming teams**; • **Social and professional topics** → *History of computing*; **Computing profession**.

Additional Key Words and Phrases: Software Engineering, Folklore

1 Introduction

Software Engineering (SE) has generated a body of shared narratives, beliefs, jokes, customs, and myths, much like other established fields or communities [34]. From stories to narratives, these cultural artifacts seem to be passed among developers, testers, and managers. These narratives range from historical anecdotes and rituals to humor and persistent myths about productivity and development practices. Even if we often dismiss them as anecdotal or unscientific, such shared narratives appear to play a significant role in shaping how software professionals perceive their work, make decisions, and construct identities [6, 15, 40].

Folklore, in the academic sense, refers to the collective traditions, stories, and customs of a group of people. Folklorists, such as Alan Dundes [10, 13] and Simon J. Bronner [8], emphasize that folklore is alive and evolving, even in modern settings. Dundes defined "folk" as any group of people who share at least one common factor—be it an occupation, language, or any shared identity—and noted that any such group will have traditions it calls its own.

Every culture has origin stories and legends—software development is no different. Consider the "first computer bug", recorded in 1947 [3, 20], where engineers found a moth stuck in a Harvard Mark II relay and taped it in the logbook with the note "First actual case of bug being found". This story, often attributed to computing pioneer Grace Hopper[1], has been retold for decades as the origin of the term "debugging". The artifact (the moth in the logbook) is preserved by a museum and is an iconic piece of technology lore.

Just as folklore includes legends, it also contains myths and beliefs—things widely repeated that may be oversimplified, exaggerated, or outright false. SE has plenty of these "received wisdoms circulating among teams". For example, a

[1]See Smithsonian National Museum of American History, Log book with computer bug, https://n2t.net/ark:/65665/ng49ca746a3-b8b7-704b-e053-15f76fa0b4fa

Authors' Contact Information: Eduard Paul Enoiu, eduard.paul.enoiu@mdu.se; Jean Malm, jean.malm@mdu.se, Mälardalen University, Västerås, Sweden; Gregory Gay, greg@greggay.com, Chalmers University of Technology and University of Gothenburg, Gothenburg, Sweden.

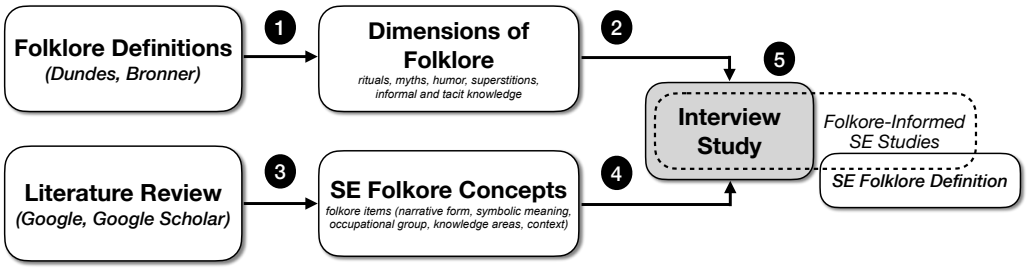

Fig. 1. Overview of the method used to explore SE folklore: theory grounding, folklore items, practitioner interviews, and synthesis into a working definition.

persistent belief is that some developers are 10 times more productive than others [6]. This notion stems from claims like "the best programmers are up to 28 times better than the worst programmers". Such claims, cited as "accepted truths" by authors like Robert Glass [16] and Laurent Bossavit [6], have gained almost legendary status.

In SE literature (e.g., [6, 15, 16]), there is a tradition of identifying and debunking software myths, essentially exposing beliefs that does not hold up to scrutiny. Laurent Bossavit's "The Leprechauns of Software Engineering" [6] explicitly calls out how folklore can turn into facts in the field through constant repetition and variation. Prior research has also examined how practitioners share experience through narratives such as war stories [22, 26] and how everyday work departs from formal processes [4, 18, 23], as also emphasized in ethnographic investigations (e.g., plans and situated actions [37]). Other lines of work in knowledge engineering and knowledge management have also focused on eliciting and externalizing expertise (e.g., [28]).

This study seeks to explore the concept of SE folklore by drawing from traditional folklore theory and applying it to the occupational practices within software development:

What constitutes SE folklore, and how can it be defined?

We aim to define what constitutes folklore in SE, identify its recurring dimensions, and examine how it manifests in different roles and knowledge areas. We examine existing implicit and explicit characterizations. We also seek to identify and categorize specific manifestations of folklore within SE practices. Ultimately, we define the concept of SE folklore and propose several areas of investigation within the context of SE. By synthesizing concepts from folklore studies with SE research, we cast a spotlight on a largely overlooked aspect of the field, and lay the foundation for future investigations into how cultural narratives shape SE.

2 Research Method

We aim to define and characterize SE folklore by identifying its dimensions and manifestations in SE contexts. The research process followed is illustrated in Figure 1.

2.1 Folklore Definitions and Dimensions

A comparative synthesis of folkloristics and folklore studies over time (e.g., [2, 17, 19, 39]) highlights different perspectives, including traditional knowledge, performance and event analysis, communication and genre approaches, and material culture traditions. With this intellectual legacy in mind, we focus on the modern reformulation of folklore [8, 9] that

emphasizes social interaction, a process-oriented conception of tradition, living folklore, and the roles of cognition and practice, as these lenses are the most immediately applicable to occupational folklore in SE settings.

We apply the definitions and categories from folklore studies—particularly from Dundes [10] and Bronner [8]—to explore folklore within an SE context (Step 1 in Figure 1). Dundes frames folklore as informally transmitted traditional knowledge shared among members of a folk group, emphasizing the roles of communication, repetition, and group identity. Bronner expands this by viewing folklore as embodied in everyday routines, rituals, artifacts, and symbols, underscoring its cultural and practical dimensions. Building on these foundations, we adapted dimensions such as rituals, myths, humor, superstitions, and informally transmitted knowledge as potential indicators of folklore (Step 2). These categories guide the development of a working definition of SE folklore.

2.2 Software Engineering Folklore Concepts

To explore how these folkloric dimensions manifest in practice (Step 3 in Figure 1), we manually collected and analyzed articles that either explicitly or implicitly engage with beliefs, informal knowledge, rituals, or cultural practices in software development.

One author conducted the search and initial screening; the resulting set of publications and items was then iteratively discussed with the other authors to refine the analysis. The initial classification was performed by the same author and reviewed and aligned through discussion with the author team. The selection process was informal and exploratory. We used Google Scholar and the Google search engine to identify literature that mentioned folklore in the context of SE or addressed relevant themes, using the phrase "*software engineering folklore*" and screening results by title. This process resulted in a set of items that should be interpreted as illustrative rather than comprehensive.

We then identified an example set of SE folklore concepts that characterize specific folklore items (Step 4 in Figure 1). To analyze these items, we used interpretive concepts (i.e., form, performance, meaning, transmission) informed by Bronner's practice-oriented description of folklore as cognitive and communicative praxis [9], as well as the *narrative forms* (e.g., myth, legend, anecdote, humor, ritual, artifact) drawn from Alan Dundes' typology [10]:

- **Myths** are stories believed to be true by a group.
- **Legends** are stories with a historical basis.
- **Anecdotes** are illustrative stories based on experience.
- **Rituals** are symbolic and repeated performed actions.
- **Artifacts** are material or symbolic objects.

We classify the narrative form of each item, as well as its *symbolic meaning*, related *occupational groups*, and relevant *knowledge areas*. The "*symbolic meaning*" refers to the cultural depth of that item, as outlined in Schein's model of organizational culture [29]. This concept captures whether the folklore functions as an artifact, an espoused value, or a basic underlying assumption[2]. Each symbolic layer represents a different degree of cultural visibility and resistance to change, and each interacts differently with folklore. The *occupational group* identifies the professional roles most closely associated with the folklore item, such as developers, testers, maintainers, architects, or project managers. Finally, each folklore item is mapped to one or more *SE knowledge areas* using the SWEBOK classification [3], thereby linking cultural expressions to specific technical domains, such as testing, maintenance, or design.

[2]Folklore items were interpreted as artifacts (visible behaviors, structures, processes), espoused beliefs and values (stated ideals, goals, justifications), or basic underlying assumptions (taken for granted beliefs shaping perception and action).
[3]For more details, we refer the reader to the SWEBOK guide: https://sebokwiki.org/wiki/An_Overview_of_the_SWEBOK_Guide

2.3 Interview Study

Drawing on SE concepts and dimensions of folklore, we designed an interview study to explore practitioner experiences (Step 5 in Figure 1). Through *iterative analysis and thematic coding*, we expanded the concept of SE folklore in practice and developed a working definition for future studies.

2.3.1 ***Data Collection.*** We conducted face-to-face and online *semi-structured interviews*, following the guidelines by Linåker et al. [21]. We follow Strandberg's guidance on ethical interviews [36]: consent was obtained, raw audio was stored with time and space limits, and transcripts were anonymized.

The *interview protocol* was planned and refined through piloting (in three initial interviews). Our analytic stance was semantic, emphasizing broad and data-driven coding while also attempting to fit the responses into the prior theoretical frame presented in Section 2.2. Before each interview, we explained the purpose, emphasized voluntary participation and the right to skip any question or withdraw, and described confidentiality practices. We requested permission to audio record solely to ensure accuracy; when granted, recordings were kept confidential. Interviews began with a brief icebreaker where participants described their current role, tenure in SE, and organizational background.

The *interview guide* included three main sections (see Table 1). Section 1 produced *general perceptions* by asking participants to define folklore in SE in their own context. Optional prompts included informally transmitted knowledge (stories, sayings, jokes, traditions) and recurring practices or rituals. Section 2 investigated four folklore types—myths and beliefs (including rules of thumb around productivity, quality, testing, programming, and management, and whether such beliefs are questioned or taken for granted); anecdotes and legends (repeated stories about SE projects, engineers, or memorable bugs, and the roles such stories play); rituals and practices (recurring SE routines such as stand-ups or events and any meanings beyond their practical purposes, e.g., identity or value signaling); and artifacts and humor (objects, memes, jokes, and what they imply about group identity, culture, or values in SE). Finally, Section 3 focused on transmission and impact, exploring how newcomers adopt folklore, how it endures over time, whether it has influenced decisions or conflicts, how its expression varies across organizations, and whether its effects are perceived as helpful, harmful, or mixed. Each interview closed with an invitation for additional examples and a debrief thanking the participant.

A pair of researchers conducted sessions—a primary interviewer and a second researcher who took notes and posed follow-up questions. Interviews typically lasted 45 to 60 minutes. Where recording was possible, we produced verbatim transcripts. When recording was not feasible, we created detailed summary transcripts that preserved the substance and meaning of the exchange and took contemporaneous or immediate post-hoc notes. All material was anonymized during transcription; personally identifying details and organizational names were replaced with neutral placeholders. Participants reviewed their transcripts for accuracy before analysis.

2.3.2 ***Population and Sampling.*** We used *purposive convenience sampling* to obtain variation in roles and organizational contexts among practitioners based in Sweden. For this exploratory phase, we aimed for 10–15 interviews to achieve depth and role diversity, monitor for thematic saturation, and to be able to add cases incrementally if new themes continued to emerge. We recruited practitioners through professional and research networks via direct invitations, and iteratively searched additional participants to address gaps in gender, role, and organizational context; recruitment stopped at $n = 12$ when successive interviews reiterated similar topics and no substantially new themes emerged.

Table 2 summarizes the 12 interviewees. Experience ranged from 9 to 41 years, with most over 15 years. Six held leadership roles, and others were senior engineers in requirements, development, testing, verification, and configuration

Table 1. Interview guide.

1. Could you briefly describe your current role and how long you have been working in SE? 2. How many different software dev organizations have you worked for?
3. In a general sense, how would you define the term folklore in software engineering? 4. When you think about folklore in the context of your work, what comes to mind?
5. Have you heard any myths and persistent beliefs, or any rules of thumb (e.g., about productivity, quality, testing, programming, management)? 6. Are there stories in your team, company, or community that people tend to repeat, for example, about past projects, engineers, or memorable bugs? 7. Can you describe any recurring practices or rituals you have encountered (for example, stand-ups, recurring events, or other routines)? 8. Are there objects, memes, or jokes that circulate within your team, company, or community?
9. Can you recall how this folklore was passed on to you when you joined a team or company? 10. Have you ever seen or experienced a piece of folklore influence a decision, shape an attitude, or contribute to a conflict at work? 11. How does the way folklore (such as beliefs, stories, rituals, or humor) is expressed or treated differ between the organizations you have worked in? 12. In your experience, do you see folklore as mostly helpful, mostly harmful, or a mix of both?
13. Before we wrap up, is there anything else you would like to share, any examples or experiences we have not touched on, that would be important for this study?

Table 2. Participant IDs and demographics.

ID	Demographics
P1	20 years experience, current role as project manager, prior roles as tester, developer, consultant.
P2	over 20 years experience, current role as requirements and engineer, prior role as developer.
P3	15-20 years experience in industry and academia, current role as automation lead, previously consultant and software and R&D engineer.
P4	9 years experience, current role as manager (development and verification), previously integrator, architect, requirements, safety engineer.
P5	41 years of experience, current role as academic, prior role as engineer and consultant.
P6	11 years of industrial experience, current role as academic, prior role as designer and requirements engineer.
P7	28 years of experience, current role in software engineering, previously across multiple companies.
P8	around 20 years of experience, current role as senior software engineer, previously at two companies.
P9	several years of experience, current role as developer and project lead, previously a verification engineer.
P10	24 years of experience, current role as manager for a software development team, previously 11 years as a developer, and 9 years as a test engineer.
P11	around 10 years of experience, current role as a software engineer, previously at three companies.
P12	21 years of experience, current role as technology leader, previously roles in several companies and universities.

management. There are both interviewees with a single-employer tenure and those with multiple-company careers. Three participants described long careers within one firm. At least eight worked across more than two companies, ranging from consulting across many sites, mixed industry-to-academia and consultancy paths, and small to large enterprise mobility. Two interviewees mentioned that during their careers, they interacted with over seven teams.

2.3.3 ***Data Analysis***. We analyzed the interviews following Braun and Clarke's guidelines for *thematic analysis* [7]. Initial coding focused on identifying recurring ideas related to the theoretical framework used (perceptions, folklore types, and transmission/impact) while remaining open to unanticipated categories. Coding then proceeded iteratively and collaboratively. After a calibration phase where all authors coded the same interview to establish agreement, all other interviews were independently double-coded by rotating pairs of authors. One author then merged the codings for each interview, resolving discrepancies through discussions.

We compared codes across interviews to surface common patterns, grouped related codes into candidate themes and subthemes in a digital whiteboard, and refined these through discussions. To aid transparency and traceability, we maintained a shared log of notes, steps, and coding. Coding was both deductive and inductive as well as semantic, allowing single excerpts to map to multiple codes and higher-order themes. We iteratively consolidated overlapping codes and themes. We monitored for thematic saturation by tracking the recurrence of themes across interviews and observed that the vast majority are reflected in multiple interviews.

The analysis produced a consolidated set of themes that describe how SE folklore is perceived, manifested, and propagated across organizational settings. The final outputs of the method were anonymized transcripts, a consolidated set of codes, a thematic map with named themes and subthemes, and curated quotations for each theme and subtheme.

Table 3. Selected folklore-related SE literature, sorted by date of publication.

Literature	Mentions Folklore	Contributions	Gaps and Opportunities
Neumann (1999) [24]	Yes	Applies folklore concepts to workplaces; emphasizes rituals and artifacts.	Suggests methods for studying workplace practices; can inform SE studies.
Basili & Shull (2005) [1]	Yes	Presents defect folklore as commonly held heuristics; examines their accuracy across studies.	Suggests ways to analyze belief formation and evaluate experience-based knowledge.
Passos et al. (2011) [25]	Yes	Uses term "technical folklore" to frame belief systems that shape SE practice.	An empirical attempt to connect belief origins, usage, and impact.
Shull (2012) [34]	No	Discusses the role of belief in SE; links beliefs to experience and calls for evidence-based validation.	Highlights the importance of understanding belief persistence; folklore theory could provide additional context.
Spínola et al. (2013) [35]	Yes	Defines technical debt folklore as community-based, experience-driven beliefs; surveys practitioners for consensus.	Indicates how shared opinions may influence SE practice; enables examination of belief adoption.
Bossavit (2017) [6]	Yes	Deconstructs several well-known SE claims ("leprechauns") as unverified beliefs passed as facts; supports empirical scrutiny.	Useful to trace belief folklore and myth-making in SE; foundational for defining and demythologizing SE folklore.
Zagalsky (2018) [40]	No	Investigates knowledge sharing in developer communities; analyzes media use and human factors.	Offers a foundation for applying folklore concepts to collaborative and knowledge-sharing of narrative aspects of SE.
Méndez Fernández & Passoth (2019) [15]	Yes	Directly connects SE to folklore; describes exemplar folklore in empirical SE; argues folklore is backed up by social mechanisms.	A good starting point in the analysis of folklore grounding; a critique of how conventional wisdom and weak claims persist in SE research and practice.
Shrikanth & Menzies (2020) [32]	No	Shows gaps between defect prediction beliefs and experiment results; shows most beliefs are context-dependent.	Opens possibilities for improving belief assessment through monitoring and context-aware adaptation.
Shrikanth et al. (2021) [33]	No	Evaluates long-standing SE beliefs; finds most have limited empirical support.	Highlights belief persistence despite lack of evidence; opportunity to interpret this persistence via folklore theory.
Ciancarini et al. (2023) [11]	No	Reviews literature on storytelling in SE; identifies functions of stories in collaboration and reasoning.	Basis for interpreting stories as folklore; potential for applying folklore theory to genres and archetypes.
Swillus et al. (2024) [38]	No	Explores developers' testing experiences; identifies lived experiences (i.e., rituals and social dimensions of testing).	Entry point to analyze software testing as a folkloric domain with habits, informal norms, and shared identity.
Romano et al. (2024) [27]	Yes	A confirmation that removing dead code improves the structure of code as well as resource usage in a specific context.	A single example of how validation can engage with folklore-generation and assumptions in SE.

3 Findings

Our findings are structured as follows: first, we summarize the contributions of selected studies to understanding SE folklore (Section 3.1); second, we present a thematic analysis of curated folklore items (Section 3.2); third, we present the thematic analysis of the interviews (Section 3.3); finally, we discuss how these findings align with folklore theories and identify conceptual and methodological gaps, proposing a definition of SE folklore (Section 3.4).

3.1 SE Folklore in the Literature

To set a foundation, we reviewed a manually curated sample of research literature that addresses informal knowledge, belief systems, narratives, and ritualized practices within software development. Table 3 presents a selection of thirteen publications that were analyzed according to their explicit discussion of folkloric concepts, their contributions to understanding the folklore of SE, and the specific conceptual or methodological gaps they expose.

While *the term "folklore" is rarely explicitly invoked in the SE literature*, several studies have addressed the persistence of various claims, myth-like narratives, and socially constructed norms in software development. Some studies explicitly use the term "folklore", such as Bossavit [6]—who critiques unempirical claims passed down as truths—and Spínola et al. [35], who label diverging practitioner opinions on technical debt as "technical debt folklore". Others, such as Neumann [24], provide theoretical grounding from library and information science, applying folklore studies to analyze how rituals and narratives shape professional identity. Several papers (e.g., Swillus et al. [38], Shrikanth et al. [33], and Ciancarini et al. [11]) reveal folklore (e.g., transmission of beliefs and practices or narratives about testing or project management) but do not frame them as folklore or analyze their symbolic and cultural significance.

In particular, Bossavit [6] examined the concept of folklore, arguing that many widely accepted "truths" in SE are myths perpetuated through misinterpretation, citation errors, and a lack of empirical verification. They show how anecdotal evidence and subjective opinions have been repeatedly cited as evidence, shaping industry beliefs without robust validation. Bossavit advocates for a more skeptical approach to SE knowledge, emphasizing the need for rigorous scrutiny and historical awareness to prevent the propagation of unfounded claims.

This selected body of work highlights a need for a definition of SE folklore that captures the informally transmitted knowledge and practices shared within occupational software development groups. Furthermore, it reveals several gaps

Table 4. Analysis of identified folklore items. Symbolic Meanings: A = Artifact, EV = Espoused Beliefs and Values, BUA = Basic Underlying Assumptions. Occupational Groups: DEV = Developers, TST = Testers, MNT = Maintainers, ARC = Architects, PM = Project Managers, ALL = All roles. Knowledge Areas: DSN = Design, CNST = Construction, TEST = Testing, MNT = Maintenance, MGMT = Management, PROC = Process, QUAL = Quality, ECON = Economics, PRO = Professional Practice.

ID	Folklore Item	Narrative Form	Symbolic Meaning	Occupational Groups	Knowledge Areas
F1	The vast majority of defects are interface defects [1].	Myth	BUA	DEV, TST	CNST, QUAL, DSN
F2	Object-oriented programming reduces errors and encourages reuse [33].	Myth	EV	DEV	DSN, CNST, QUAL
F3	Higher software quality results in better productivity [33].	Myth	EV	DEV, PM	PROC, QUAL
F4	Developer performance varies dramatically (e.g., 10x productivity) [33, 6].	Legend	BUA	DEV, PM	PRO
F5	Becoming an expert requires at least 5000 hours [33].	Myth	EV	DEV	PRO
F6	Technical debt is unavoidable in real-world projects [35].	Myth	EV	DEV, ARC	MNT, PROC
F7	Unintentional technical debt is much more problematic than intentional [35].	Myth	EV	DEV, MNT	MNT
F8	Technical debt originates from short-term optimization [35].	Myth	EV	DEV, PM	DSN, MNT
F9	Defect classes follow organization-specific patterns that can be detected within a given context [1].	Myth	BUA, EV	TST, DEV	QUAL, PROC
F10	Developers are more likely to test when they feel ownership over the code [38].	Anecdote	EV	DEV	TEST
F11	Testing is a tedious burden, far less exciting or rewarding than coding [38].	Anecdote	EV, BUA	DEV	TEST
F12	If you are not sure what to do, do something and fix it later [1].	Myth	BUA	DEV	CNST, MNT
F13	Personal artifacts and cluttered desks are part of work identity [24].	Artifact	A	DEV	PRO
F14	Jokes, photocopies, and office memes circulate informally in teams [24].	Humor	A	ALL	PRO
F15	Dead code removal improves performance and maintainability [27].	Myth	EV	DEV, MNT	CNST, MNT

and opportunities, including the absence of methods to study folklore elements such as rituals, storytelling, and belief formation in SE contexts. While some studies identify and validate beliefs (e.g., about defects, technical debt, or dead code), others reveal a persistence of unverified or outdated practices. Overall, few studies explicitly apply methods or theories from ethnographic studies to the software development domain [30]. Such methods could help explain how cultural narratives, myths, and shared practices influence both researchers and practitioners. This gap motivates the need to establish SE folklore as a formal concept with descriptive, critical, and cultural explanatory power.

3.2 Analysis of the Identified Folklore Items

Table 4 presents an analysis of folklore-related items (F1–F15) derived from the studies in Table 3. Our intention was not to provide an exhaustive account of folklore items, but to highlight exemplar cases that can be used to examine various dimensions and concepts of SE folklore. The folklore items span a range of topics—beliefs about where defects occur [1], the relation between quality, productivity, and experience [33], views on technical debt [35], perceptions such as the "super-programmer" [6], and the idea that removing dead code improves maintainability and performance [27].

Most items are framed as myths—generalized beliefs rooted in collective experience—particularly within software construction, maintenance, and quality. These myths typically express espoused values (e.g., F2–F3, F5–F8), but some reflect basic underlying assumptions (e.g., F1, F4, F12), meaning they seem to be taken for granted and are rarely questioned in practice.

For example, F1 disseminates the idea that the vast majority of defects are interface defects. This item seems to operate as a basic underlying assumption for developers and testers engaged in construction, design, and quality work. F5 advances the claim that becoming an expert requires at least 5,000 hours. As an espoused value, this item seems to deliver an encouraging message—sustained, focused effort is a route to deep SE competence. Shrikanth et al.'s reassessment indicates that this rule holds only in some cases [33]. Novices often match experts in both speed and quality, suggesting that experience adds little unless deliberate practice is in place.

Some entries are anecdotes (F10–F11)[4] or artifacts and humor (F14), yet these still signal cultural identity, emotion, or irony that guide team practices and shape social aspects of their work. Consider F11, the anecdotal belief that testing

[4]In our framing, an anecdote is a short, experience-based story used to illustrate a point. We tagged items as anecdotes when they are situated accounts. For example, F10 and F11 are commentaries made by interview participants in a specific case [38].

is a tedious burden. This item combines both espoused and underlying symbolic layers, indicating perceived tensions around the value of testing (relative to coding) that resonate in the testing area and across team cultures.

F13—the idea that personal artifacts and cluttered desks are part of identity—is categorized as an item related to "artifacts", as this belief reveals how material environments carry symbolic importance in practitioners' self-expression and professional practice. Meanwhile, F4 (the 10*x* developer legend) is built on decades of stories of mythical practitioners. This legend could potentially influence hiring, team composition, and performance expectations, especially among developers and project managers. These examples highlight how one can not only catalog folkloric content but also surface the socio-cultural operations that underpin SE.

Qualitative analysis of the folklore items reveals several themes. First, *many beliefs serve as heuristics that simplify reasoning in technically complex situations*. These include commonly repeated rules of thumb, such as the fact that most defects occur at module interfaces or that unintentional debt is worse than intentional debt, which seem to provide practical guidance, despite such guidance often lacking clear empirical grounding.

Other folklore items relate to *identity-related dimensions of SE work*. The perception that testing is tedious or not rewarding [38], or that a cluttered workspace is a sign of professional identity [24], reflects deeper assumptions about roles, hierarchies, and belonging. Similarly, the narrative of the 10*x* developer could reinforce individualist identity ideals within organizational culture. *Folklore related to technical debt* emerges as a particularly rich domain. Cultural beliefs about the inevitability of debt and its connection to short-term decision-making highlight organizational challenges.

Several folklore items appear to persist despite being challenged. These include beliefs in the universal benefits of specific design patterns and exaggerated claims about variations in developer performance. Their durability may suggest that factors such as cultural alignment, personal experience, intuition, and social reinforcement often outweigh empirical evidence in shaping beliefs. In addition, *ritual and informality* also hold significant folkloric meaning.

The "transmission" of folklore remains understudied. With few exceptions—i.e., via community interactions [40] and through shared project and learning practices [38]—there is little evidence or direct investigation into how these beliefs are passed from one practitioner to another, whether through mentorship, documentation, onboarding, or socialization. There is a need to trace transmission mechanisms to better understand how these practices persist and evolve across different contexts and through various media.

3.3 Interview Study Results

In this section, we present the interview results organized into practitioners' general perceptions of folklore, concrete folkloric forms (i.e., myths, anecdotes, rituals, artifacts, humor), and their transmission and impact on SE work. The emerging themes contribute directly to what constitutes SE folklore and how it can be defined by grounding the concepts in practitioners' accounts.

3.3.1 ***General Perceptions.*** Participants described SE folklore as a lived and practical phenomenon that sits between what practitioners *say* they do and what they *actually* do. Three emerging themes capture these views: *First Impressions of What Folklore Is*, *Practice versus Ideals*, and *Cultural Expressions*.

When asked what folklore meant, participants described it as unwritten, informal, and often second-hand know-how, typically shared through code reviews, onboarding, or hallway talk. Some framed it as myths and jokes, while others saw it as beliefs without direct evidence, overlapping with opinions and rules of thumb.

Because it functions implicitly, folklore is rarely named as folklore; it persists in practice and only becomes visible on reflection or when someone tries to spread it:

> "P7: Stories with monsters in the [forest], but in software engineering, these are beliefs… not supported by data or evidence."

Participants perceived much of what circulates as "*common wisdom*" driven more by opinion than by evidence, yet still shaping choices. They expressed that folklore captures the friction between ideals and practice, with cultures influencing which stories take hold and how they guide action:

> "P12: I think it is that perception and reality and myth and what is actually happening... and also which is the culture, because I believe that [culture] also matters. Even if you want to go against something, the culture forces you to come back and follow the guidelines."

Simultaneously, folklore was valued for its pragmatism, offering workarounds and context that formal processes may lack.

Folklore was also described as something you can see in how teams communicate. Participants pointed to national style differences in discourse and humor—for example, norms around constant counter-argument or the tolerance for rougher jokes—as shaping which stories get repeated:

> "P5: We are a multicultural company, but... overall company culture promotes how knowledge transfer happens... I did feel barriers because of peoples' skill set."

These views suggest that folklore is a constraint and an enabler.

3.3.2 ***Myths and Beliefs.*** Our analysis surfaced myths and beliefs. These beliefs act as shortcuts for coordination and justification, but they also disrupt planning, resourcing, and evaluation. We organize findings into seven themes, comprising recurring subthemes.

Testing Myths: Participants described testing as both marginalized and misunderstood. Testing was frequently framed as "*minority work*" delegated to the least empowered roles.

> "P1: As a tester, it is easy to be seen as a nitpicker... We will release the product as soon as the testers stop finding problems. So we need to test less... So the testers are the cause of the project delay."

At the same time, code coverage is pushed as a KPI, leading to performative activities to "*hit the number*". Across teams, testing was positioned as "*less than*" development—or "*anyone can do it*"—which was perceived as deteriorating test competencies in the organization. One participant mentioned that some practitioners caricatured exploratory testing as quick and freestyle, overlooking the structure and skill it requires.

Several participants noted automation myths, e.g., test automation being equated with progress. At the same time, automation's limits were understood as: "*automation only covers the easy parts*". Another participant mentioned the myth "*AI will replace testers*".

Process Myths: Participants repeatedly encountered "*silver bullet*"-type myths. For example, model-based SE (MBSE) was promoted as a savior, with promises of fit, ease, and immediate benefits. When benefits were delayed, blame shifted to immature teams rather than to poor fit to the context:

> "P7: Model-based development, or MBSE, holds great promise... But it has become a myth that it is suitable for all software."

Related to SE processes, the myth of a linear, sequential waterfall/V-model process, where activities are completed and never revisited, persisted even in organizations that practiced iterative work.

Hype and Status-Based Adoption: Some participants mentioned that some choices were guided by hype and status in the community rather than fit. One participant described half-baked research repackaged as a product and sold via consultancy narratives.

Participants also discussed the myth that "*newer means better*", frequently coupled with the idea that all organizations should imitate the practices of the largest tech companies, regardless of fit:

> "P7: ...We must use a new way; this means it is better. Folklore here is that everything new in SE is good. It is natural to have a sense that what is novel is good."

> "P8: ...this is how [big tech companies] do this thing. So that is why we should do it... do we really have their scaling problems? It does not matter, you know?... It says [big tech] on the box of this technology, so it must be 'super awesome'... so you [believe] the myth that it is good because it is from them."

Developer Myths: Participants reported beliefs related to the practice of development. For example, "*Code is your documentation*" rationalizes the use of code instead of other software artifacts (e.g., requirements, architectural designs) to document design decisions:

> "P3: So they had very sporadic, unfinished documentation, very, very cryptic... I heard the most famous quote from the previous team, which was like: Code is your documentation... if you want to know how something works, look into the code..."

Other myths relate to the idea that code—once working—should not be touched again, whether in the case of "*mysteriously working*" code or code written by "*perfect senior developers*":

> "P4: ...we had a system where the documentation included who wrote a certain code, and you could see the name. If the name was someone who had been there for fifteen years, a very senior developer, there was a myth that you do not touch it. It is basically perfect."

Myths Shaping Planning and Resourcing: We identified several planning and resourcing narratives that combined classic myths with organization-specific heuristics. The Mythical Man-Month myth was mentioned, where managers added people thinking it would make the project "*go faster*". Teams described "*multiply by pi*" estimation as a folk correction for optimism leading to poor estimation:

> "P1: Ask the developer ...How long? ...A week... Project manager says, Multiply by pi... I have seen people do that."

Testers causing delays was a recurring accusation near release gates, reinforcing some of the testing myths mentioned:

> "P1: We will release the product as soon as the testers stop finding problems. So we need to test less… So the testers are the cause of the project delay. [laugh] This is such an anti-pattern: the deadline is the 1st of December, development is delayed, so you squeeze the testing."

The belief that "*there is always a new release*" was mentioned by a participant, similarly "*it can be fixed in software*" justified shortcuts and late requirement changes. Other myths also shaped the perception of participants: the "*10-year rule*"—the idea that a technology must be proven for a decade to be adopted—and the "10*x programmer*" belief were mentioned as influencing hiring and allocation. Several participants mentioned the belief that their project will be "*bug-free if the process is followed*" to the letter:

> "P7: Tollgate meetings: after these, the software magically works... things fall into place once these tollgate meetings occur. Seems useful sometimes. Creates a false sense of security in project planning..."

Teams in different organizations seem to oscillate between method-freedom and heavy standards. In addition, the "*1-week handover myth*" was noted, suggesting that experience and knowledge can be fully documented and transferred on a schedule.

Agile Myths. Agile was reduced to rituals in some cases: "*agile means no requirements and documentation*". Daily stand-ups should be productive, but participants mentioned that they can turn into coordination and performative

meetings. Role misunderstandings seem to persist: "*Scrum eliminates project managers*". Also, agile adoption beliefs in software-intensive organizations often ignore contextual constraints.

AI in SE Myths. AI was framed as an inevitability. The fear of missing out in AI was perceived as accelerating adoption without addressing problems: "*AI will solve everything*" supported one-size-fits-all automation. At the same time, "*AI will replace experts*" fueled narratives around adoption.

3.3.3 ***Anecdotes and Legends.*** Our thematic analysis surfaced four emerging themes related to anecdotes and legends.

Control and Compliance Lore: Participants described a body of stories about compliance work, where legal and escrow tales circulate as warnings. For example, a customer asked for the escrow password, which exposed a leak and became a cautionary tale. Other stories of resistance to scaled processes, such as ad-hoc releases, were mentioned.

Memory and Cautionary Lore: Practitioners maintain a living memory. For example, some expressed nostalgia for "*the onsite good old days*" where software was made by small teams in a single location:

> "P3: Ah, the good old times, we would just go there, we would sit, and we would build the system and it would work because it was being made on the spot. But of course this does not scale."

Counteracting this, others discussed hype-cycle legends recounting technologies embraced and later rolled back. These stories are used to caution against trend-based adoption. In both cases, memory can act as a decision heuristic.

Legendary Bugs and Narrative Bias: Across interviews, several failure stories were repeatedly mentioned, including the "*Denver baggage failure*", "*Therac-25*", and other classic failures:

> "P5: There was an article in 1994 in Scientific American where they had this practical example of the luggage system at Denver Airport. It delayed the opening of the airport by six months or so and cost zillions of dollars... What they did not tell was that just a few years after, everything was up and running, and the luggage system paid back the investment and generated a profit."

Participants acknowledged the spread of narrative bias—failures are retold and amplified, whereas later successes receive little attention. Nevertheless, these legendary bugs are treated as domain-specific lessons that could influence practices.

Cautionary Tales about Testing and Design Debt: A fourth theme centers around how design and testing choices are transformed into folklore. Tales of designs that resist external demands reveal workarounds that satisfy short-term pressure but create hidden histories in artifacts—design decisions that future teams must decode. Hardware and software boundary tales persist as well—decoy circuits are recounted as clever but debt-creating design decisions. In contrast, participants circulated positive narratives about how architecture and design patterns enabled fast change.

Testing-related tales list routine development frictions—e.g., testing is never-ending, clashes appear when test code owned by territorial developers (*"my baby"* code) or *"sacred"* legacy modules are owned by experts.

3.3.4 ***Rituals and Practices.*** Across interviews, participants described rituals and practices that pass on know-how, coordinate work, and sometimes shape behavior, clustering into five themes.

Reuse Practices: Participants described blind reuse without validation ("*it worked there, reuse it here*") that later required costly rework, and repeated naming conventions that embed hidden histories in artifacts (e.g., cryptic column names or inherited signal labels). Some "*superstitions*" also persist, such as adding timing delays in test cases or reset habits, which originated in manual testing but are now used in automated tests.

Fig. 2. Illustrative examples of transmitted memes referenced in the interviews.

Mentorship Patterns: Informal mentorship often takes the form of withholding direct answers to develop further skill:

> "P3: ...you could get [ten] questions that are leading you to get the answer... But it was also a very unorthodox way of teaching someone. [...] But yeah, it was a drill to prepare you for the environment when you go to the site... a very unorthodox kind of mentor[ship]."

Also, weekly senior syncs offer a forum where experts share stories, heuristics, and surface patterns. Together, these forms create a channel to transmit knowledge.

Rituals as Social Glue and Morale Building: Teams use formal meetings as bonding opportunities. Stand-ups are used for social sessions; sprints are named after desserts with small rewards and recurring syncs as team-building. These rituals seem to be used to create belonging:

> "P3: ...for the sprint name, we would choose some kind of a name for a cookie or some dessert... and the reason was that if we managed to finish above [certain threshold] of the tasks in that sprint, then we would get it at the sprint retrospective or closing meeting."

Community of Practice: Outside formal roles, engineers sustain their community through voluntary coding meetups and discussions or education around quality:

> "P8: We have a few colleagues that meet up and do some programming online once a week... it is also not typically [a] work-related thing. For instance, let us try out some property-based testing, or something like that. A bit of fun, but also a bit of learning and doing it together."

These sessions provide places to explore new techniques. In addition, keeping "*quality*" as a standing topic (e.g., code review checklists, defect narratives) can be used to re-frame perceptions around joint responsibility.

Rituals for Control, Learning, and (Sometimes) Surveillance: Certain practices operate as a form of lightweight control over practitioners. Reviews and tollgates are treated as guarantees of quality and progress. Coffee or lunch breaks function as informal problem-solving sessions that seem to reduce silos and shared spaces were repeatedly cited as places where stories travel. Work-from-home mistrust was perceived as treating visibility as productivity, pushing in-person attendance rituals. Within SE practice, code review is narrated as a form of collective learning; checklists sometimes ensure rigor, but at times are used "*for visibility*". Participants also mentioned stand-ups as a form of surveillance ("*tell me you are working*") alongside standard planning practices that improved alignment and shortened feedback loops.

3.3.5 ***Artifacts and Humor.*** We identified two emerging themes.

Humor as Coping and Critique: Practitioners signal difficulty, emotion, and critique through jokes—e.g., the *"backlog equals graveyard"* or *"feature future"* metaphor shows skepticism about using prioritization mechanisms to delay or silently cancel a feature, *"drinking because testing"* frames testing as an ordeal, and "*trolls explain flaky systems*" offers a playful story that normalizes intermittent failures. Circulating texts, such as a printed quote sheet (i.e., catchphrases), provide a channel for critique and mentoring. Humor is also a form of resistance—e.g., using a joke to challenge unrealistic safety requirements or self-proclamation of oneself as a "*pain in the ass*" to show pride in verification and quality gatekeeping efforts. Newcomer socialization is evident in first-day jokes that motivate and tease, introducing newcomers to local expectations. Finally, cross-team banter that can go into blame cultures can be seen as a double-edged sword: it can coordinate developers and testers, for example, but when tensions rise, it can drift into stigma.

Meme Culture: Memes, generally in the form of images, are a specific form of humor often shared within teams. Memes resurface when certain events recur, allowing practitioners to learn from and remember past incidents. Memes are often created locally by practitioners for their specific projects or histories, and are often used to relieve hierarchical tensions (e.g., jokes about managers).

Several general memes were mentioned (some shown in Figure 2)—e.g., the "*swing*" meme (showing miscommunication about requirements), the "*silenced junior*" (two cats fighting, with the junior being silenced to avoid delaying a release), and *"backlog and fix it later"*. One participant also mentioned the "*this is fine*" (shared experience of living with chaos) and "*real men test in production*" (warning about the implications of minimizing testing) memes.

3.3.6 ***Transmission and Impact.*** Participants described SE folklore as both shaping choices and structuring how newcomers learn.

Perceived Impact of Folklore: Folklore affected decisions and the everyday environment. For example, model-based SE drove adoption decisions and participants emphasized that the effects were both harmful and helpful, depending on the context:

> "P2: MBSE will be a big game changer...[a] decision to introduce MBSE... created a lot of challenges... misunderstanding of when you get benefits."

In routine interactions, folklore served as a way of teaching or venting and was seen as generally helpful, yet this coexisted with an explicit, mostly harmful verdict on folklore due to perceived costs and for instilling non-evidenced beliefs. These doubts can affect which folklore items are introduced to newcomers and sustained.

On-boarding Transmission: The first transmission seems to occur during on-boarding. Memes used in on-boarding create shared references, often flowing from senior to junior alongside cautionary tales. In some teams, deliberately minimal or cryptic documentation also seems to push newcomers to lean on folklore. This early shaping seems to connect later to patterns.

Implicit and Informal Transmission. Once inside the team, quiet norms can repress questioning, while humor can influence the team climate. At the same time, small talk can help in building a sense of belonging. Folklore seems to be recognized upon reflection:

> "P4: ...this is something we have been unconsciously seeing and practicing and doing, but we have never formally looked at it or thought about it."

On-boarding stories continue to circulate, and some beliefs are transmitted in one-to-one discussions.

Transmission Mechanisms: These flows seem to be amplified by where and how practitioners meet. Rumors become known, especially as they are retold in common areas—e.g., coffee or lunch rooms—which become sites for sharing stories and lessons:

> "P9: People already assume that they know the whole picture... and they start spreading that as if it is the gospel."

Here, informal leaders and lived experience seem to set norms. However, language and cultural barriers can impede participation, shaping who speaks, who listens, and whose lessons are told.

Mediators: Verification leads are seen as practitioners who counter misconceptions and can change narratives, particularly where some organizations use experimentation. In other cases, some companies seem to allow beliefs to persist.

Scope and Impact: Just a few participants reported little to no SE-specific folklore within their organizations, expressing that many of these stories or practices are general to all workplaces. Another participant mentioned that "*negative folklore*" seems to fuel public mistrust of software and SE.

Organizational Culture: Folklore seems to vary by role and organization, and is influenced by company climate. One participant mentioned that team shuffling is a way of refreshing folklore by moving narratives across groups. The size of the organizations could matter too, with one participant mentioning that small firms they were part of are less ritualistic, while large firms accumulate such rituals more often.

Educational Channels: Computer science courses often leave students thinking "*code solves everything*" and that attitude seems to carry into practice. Some practitioners, especially juniors, also fixate on specific programming languages, a preference that tends to soften with experience.

Community Transmission: Practitioner schools-of-thought and skepticism of academic research can shape what folklore teams accept. External channels (i.e., talks by well-known figures, conferences) propagate beliefs, seeding narratives that reappear in practice.

Folklore Transfer: Participants mentioned evolving forms of transfer across careers. Early-career "*bucket filling*" involves practitioners accumulating narratives, they consolidate them mid-career, and veterans become set in their way of working ("*we tried that*"):

> "P11: They tried something, I do not know, 25 years ago, and then they keep saying 'we tried that once. It is not gonna work'... but because of this old story, they do not want to even try..."

3.4 A Software Engineering Folklore Definition

Software development in industrial settings, particularly within occupational groups, offers a ground for understanding how folklore manifests in SE environments [31]. By applying Alan Dundes' characterization—focused on folk groups, informal transmission, and tradition—and Simon J. Bronner's [8] practice-centered description—emphasizing praxis, knowledge in action, and phemic processes—we can explore how software development teams cultivate occupational folklore through practices, rituals, and traditions.

Based on Dundes' description, folklore is "traditional knowledge transmitted informally within folk groups" [12]. Knowledge is passed through experience, mentorship, or casual interaction rather than formal instruction, while practices and expressions are repeated with variations across teams and projects. The *occupational group* categorizations presented in Table 4 provide a basis for understanding how folklore manifests and functions within SE communities. For example, *software testers, designers, and developers can form distinct folk groups*. Folklore in software development can help create and reinforce group identity. Informal communication channels can serve as transmission vehicles for occupational traditions, shaping how teams work and interact. Our analysis uses a folklore lens (function, transmission, persistence, variation), but shows only a partial view. Without longitudinal data or cross-community circulation evidence, claims about folklore evolution remain interpretive.

Bronner represents folklore as "traditional knowledge put into, and drawing from, practice" [8]. In this explanation, folklore refers to how individuals enact traditions through their daily activities. Consistent with Bronner's practice lens, Enoiu's essay [14] illustrates how software testing preconceptions and routines circulate as lore, shaping testers' activities. Thus, actions and expressions can carry suggestive meanings, shaping cultural identity, and folklore emerges through the actual activity of testing software, not just through verbal communication. As shown in Table 3 and Table 4—as well as the themes surfaced in the interviews—folklore is deeply embedded in software development. Practices like testing, programming, reviews, deployments, and retrospectives often take the form of ritualized actions that carry symbolic meaning and help teams navigate complex work environments. Folklore lies in the praxis of these activities and how they are performed, adapted, and symbolized. In addition, following Bronner's notion of folklore as a communicative process integrated in social practices, our definition should emphasize not just what folklore items say, but how and when they are performed.

We synthesize concepts across (1) the folkloristic characterizations and dimensions, (2) the folklore items from literature, and (3) the thematic results from interviews. Building on the folkloristic, we consider the informality of transmission and praxis as foundational principles, both traditional and emergent. The literature review pinpoints the forms these take in SE as narratives and heuristics. Finally, the interviews show these elements in reality—myths, stories, memes, rituals, and practices that actively shape identity, values, and shared know-how across occupational folk groups and knowledge areas. We propose the following definition:

Software engineering folklore, comprising informally transmitted, traditional, and emergent narratives, heuristics, and artifacts enacted by practitioners, circulates within occupational folk groups (e.g., developers, testers, and managers) and shapes identity, values, and collective knowledge throughout the socio-technical ecosystem of software development.

This definition highlights the informal nature of knowledge and practices within occupational SE groups. Future research on this topic should further develop this definition and its underlying dimensions and forms, and validate the folklore concepts.

4 Implications and Applications of SE Folklore

To operationalize our definition of SE folklore, one would need to treat a folk item as a unit of analysis and evaluate it against the criteria and analytical dimensions (i.e., form, meaning, transmission, spread, practice, and relevance) to enable comparison and analysis.

For engineers, managers, and team leads, understanding folklore can support more thoughtful cultural and organizational interventions. Practically, this involves identifying persistent narratives and reflecting on their origins, usefulness, and limitations, recognizing when narratives indicate resistance to change or when they are needed to make implicit cultural norms visible. By viewing folklore not as problematic or negative but as cultural knowledge, teams can examine unhelpful stories or beliefs while preserving practices that are effective in their context and that reflect their values.

For researchers in empirical SE and socio-technical systems, folklore offers a lens for examining the lived realities of software teams. The identification of folklore in SE opens the door to empirical and interpretive studies that investigate the human, cultural, and narrative dimensions of tech work. Longitudinal, immersive fieldwork in software teams can surface informal beliefs, rituals, and artifacts that shape practices. Such ethnographic studies [41] are well-suited for

uncovering knowledge and group identity. Retrospective accounts and oral history can show how SE teams' stories become folklore and how beliefs shift over time [5].

5 Threats to Validity

We adapted folkloristics to SE, which may shape how folklore is identified and described; future work should refine these constructs with more practitioner input and comparative analysis among researchers. Our literature review also relied on a small, purposively selected sample and an informal search. Therefore, a more systematic protocol that includes grey literature is needed. Further, the interview study and the thematic analyses of the identified folklore items are subject to the authors' biases. To mitigate this threat, the coding was performed independently by all authors, then iteratively validated. Disagreements were discussed, resulting in only a few revisions. In addition, our convenience sampling and limited demographic diversity, particularly a skew toward more experienced participants, may bias the themes toward patterns typical of that group. This limitation may restrict generalizability across other experience levels, roles, and organizational contexts. These limitations define the scope of the current work but do not undermine its core contributions in exploring and defining SE folklore. Addressing them in future research will help build a more comprehensive and actionable understanding of SE folklore.

6 Conclusions

Drawing on foundational ideas from folklore studies, literature, and an interview study, we have proposed a working definition that characterizes SE folklore as informally shared, traditional, and evolving narratives integral to everyday SE practice. As in other fields, folklore can function not only as a source of informal knowledge but also as a means of creating identity group cohesion and managing uncertainty in complex social environments. Recognizing the role of SE folklore has both practical and theoretical value. For practitioners, it creates opportunities to reflect on norms, challenge myths, and maintain positive customs. For researchers, it opens up new directions for investigation, including ethnographic and folklore studies, comparative analysis across domains, and methodological tools.

Acknowledgments

Support was provided by the Software Center project 68 (TRACE) and MONA LISA (ITEA) project funded by Vinnova. Enoiu was also supported through the AI and Society Fellowship (AI@MDU). We thank Alex Cusmaru for his input and valuable discussions on the SE folklore concept and an earlier version of the manuscript.

References

[1] Victor Basili and Forrest Shull. 2005. Evolving defect 'folklore': a cross-study analysis of software defect behavior. In *Software Process Workshop*. Springer, 1–9.

[2] Dan Ben-Amos. 1971. Toward a definition of folklore in context. *The Journal of American Folklore*, 84, 331, 3–15.

[3] Kurt W Beyer. 2012. *Grace Hopper and the invention of the information age*. Mit Press.

[4] Alexander Boden, Gabriela Avram, Liam Bannon, and Volker Wulf. 2012. Knowledge sharing practices and the impact of cultural factors: reflections on two case studies of offshoring in sme. *Journal of software: Evolution and Process*, 24, 2, 139–152.

[5] Joanna Bornat et al. 2004. Oral history. *Qualitative Research Practice, SAGE*, 34–47.

[6] Laurent Bossavit. 2017. *The Leprechauns of Software Engineering: How folklore turns into fact and what to do about it*. Leanpub.

[7] Virginia Braun and Victoria Clarke. 2006-01-01. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3, 2, 77, 101.

[8] Simon J Bronner. 2016. *Folklore: the basics*. Routledge.

[9] Simon J Bronner. 2016. Toward a definition of folklore in practice. *Cultural Analysis*, 15, 1, 6–28.

[10] Simon J. Bronner, (Ed.) 2007. *Meaning of Folklore: The Analytical Essays of Alan Dundes*. University Press of Colorado. ISBN: 9780874216837. Retrieved July 24, 2025 from http://www.jstor.org/stable/j.ctt4cgrzn.

[11] Paolo Ciancarini, Mirko Farina, Ozioma Okonicha, Marina Smirnova, and Giancarlo Succi. 2023. Software as storytelling: a systematic literature review. *Computer Science Review*, 47, 100517.
[12] Alan Dundes. 1965. On computers and folk tales. *Western Folklore*, 24, 3, 185–189.
[13] Alan Dundes. 2019. Who are the folk? In *Frontiers of Folklore*. Routledge, 17–35.
[14] Eduard Paul Enoiu. 2025. An essay on the role of folklore in software engineering: preconceptions and their meaning in software testing and test automation. *Software Center Reporting Workshop*.
[15] Daniel Méndez Fernández and Jan-Hendrik Passoth. 2019. Empirical software engineering: from discipline to interdiscipline. *Journal of Systems and Software*, 148, 170–179.
[16] Robert L Glass. 2002. *Facts and fallacies of software engineering*. Addison-Wesley Professional.
[17] E Sidney Hartland. 1891. Report on folk-tale research in 1889-1890. *Folklore*, 2, 1, 99–119.
[18] Claire Ingram and Anders Drachen. 2020. How software practitioners use informal local meetups to share software engineering knowledge. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 161–173.
[19] Joseph Jacobs. 1893. The folk. *Folklore*, 4, 2, 233–238.
[20] Peggy Aldrich Kidwell. 2002. Stalking the elusive computer bug. *IEEE Annals of the History of Computing*, 20, 4, 5–9.
[21] Johan Linåker, Sardar Muhammad Sulaman, Rafael Maiani de Mello, and Martin Höst. 2015. Guidelines for conducting surveys in software engineering.
[22] Wayne G Lutters and Carolyn B Seaman. 2007. Revealing actual documentation usage in software maintenance through war stories. *Information and Software Technology*, 49, 6, 576–587.
[23] André N Meyer, Earl T Barr, Christian Bird, and Thomas Zimmermann. 2019. Today was a good day: the daily life of software developers. *IEEE Transactions on Software Engineering*, 47, 5, 863–880.
[24] Laura J. Neumann. 1999. Paper, piles, and computer files: folklore of information work environments. *Library trends*, 47, 3, 439–469.
[25] Carol Passos, Ana Paula Braun, Daniela S Cruzes, and Manoel Mendonca. 2011. Analyzing the impact of beliefs in software project practices. In *International Symposium on Empirical Software Engineering and Measurement*. IEEE, 444–452.
[26] Donald J Reifer. 2013. *Software war stories: Case studies in software management*. John Wiley & Sons.
[27] Simone Romano, Giovanni Toriello, Pietro Cassieri, Rita Francese, and Giuseppe Scanniello. 2024. A folklore confirmation on the removal of dead code. In *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering (EASE)*. ACM, 333–338.
[28] Sharon Ryan and Rory V O'Connor. 2013. Acquiring and sharing tacit knowledge in software development teams: an empirical study. *Information and software technology*, 55, 9, 1614–1624.
[29] Edgar H Schein. 2010. *Organizational culture and leadership*. Vol. 4. John Wiley & Sons.
[30] Helen Sharp, Yvonne Dittrich, and Cleidson RB De Souza. 2016. The role of ethnographic studies in empirical software engineering. *Transactions on Software Engineering, IEEE*, 42, 8, 786–804.
[31] Mary Shaw. 2002. Prospects for an engineering discipline of software. *IEEE Software*, 7, 6, 15–24.
[32] N. C. Shrikanth and Tim Menzies. 2020. Assessing practitioner beliefs about software defect prediction. In *International Conference on Software Engineering (ICSE): Software Engineering in Practice*. IEEE, 182–190.
[33] N. C. Shrikanth, William Nichols, Fahmid Morshed Fahid, and Tim Menzies. 2021. Assessing practitioner beliefs about software engineering: an empirical investigation. *Empirical Software Engineering*, 26, 4, 73.
[34] Forrest Shull. 2012. I believe! *IEEE Software*, 29, 1, 3–4. doi:10.1109/MS.2012.10.
[35] Rodrigo O Spínola, Antonio Vetrò, Nico Zazworka, Carolyn Seaman, and Forrest Shull. 2013. Investigating technical debt folklore: shedding some light on technical debt opinion. In *International Workshop on Managing Technical Debt (MTD)*. IEEE, 1–7.
[36] Per Erik Strandberg. 2019. Ethical interviews in software engineering. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 1–11.
[37] Lucille Alice Suchman. 1987. *Plans and situated actions: The problem of human-machine communication*. Cambridge university press.
[38] Mark Swillus, Rashina Hoda, and Andy Zaidman. 2024. Who cares about testing? co-creations of socio-technical software testing experiences. *arXiv preprint arXiv:2504.07208*.
[39] Stith Thompson. 1951. Folklore at midcentury. *Midwest Folklore*, 1, 1, 5–12.
[40] Alexey Zagalsky. 2018. *Knowledge Building in Software Developer Communities*. Ph.D. Dissertation. University of Victoria.
[41] He Zhang, Xin Huang, Xin Zhou, Huang Huang, and Muhammad Ali Babar. 2019. Ethnographic research in software engineering: a critical review and checklist. In *Proceedings of the Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ACM*, 659–670.