With Great Humor Comes Great Developer Engagement

Deepika Tiwari
deepikat@kth.se
KTH Royal Institute of Technology
Stockholm, Sweden

Tim Toady
toady@eecs.kth.se
Programming Republic of Perl
Merise, Estonia

Martin Monperrus,
Benoit Baudry
{monperrus,baudry}@kth.se
KTH Royal Institute of Technology
Stockholm, Sweden

ABSTRACT

The worldwide collaborative effort for the creation of software is technically and socially demanding. The more engaged developers are, the more value they impart to the software they create. Engaged developers, such as Margaret Hamilton programming Apollo 11, can succeed in tackling the most difficult engineering tasks. In this paper, we dive deep into an original vector of engagement – humor – and study how it fuels developer engagement. First, we collect qualitative and quantitative data about the humorous elements present within three significant, real-world software projects: `faker`, which helps developers introduce humor within their tests; `lolcommits`, which captures a photograph after each contribution made by a developer; and `volkswagen`, an exercise in satire, which accidentally led to the invention of an impactful software tool. Second, through a developer survey, we receive unique insights from 125 developers, who share their real-life experiences with humor in software.

Our analysis of the three case studies highlights the prevalence of humor in software, and unveils the worldwide community of developers who are enthusiastic about both software and humor. We also learn about the caveats of humor in software through the valuable insights shared by our survey respondents. We report clear evidence that, when practiced responsibly, humor increases developer engagement and supports them in addressing hard engineering and cognitive tasks. The most actionable highlight of our work is that software tests and documentation are the best locations in code to practice humor.

CCS CONCEPTS

• **Software and its engineering** → *Programming teams*; • **Social and professional topics** → *Cultural characteristics*.

KEYWORDS

Humor, Developer engagement, Responsibility, Culture, Faking

LAY ABSTRACT

Modern software applications are built piecemeal. They are composed of a large pool of contributions made by individual software developers. These developers spend long hours brainstorming with colleagues, solving challenging engineering problems, and writing code. Software companies put different strategies in place to engage their developers and support them in coping with these hard tasks. In particular, some organizations and groups of developers decide to nurture the creative and expressive aspects of software development. In this work, we study a universal human trait that has been

key for creativity since the dawn of civilization: humor. Humorous software code is a powerful deterrent to boredom. It can uplift the mood of the developer who writes it, as well as a peer, separated by time and space, who chances upon this nugget in the codebase. For example, Leia Organa is as perfect as John Doe for a placeholder customer name while testing an e-commerce application. A gallery of photographs can serve as a cheerful journal of the Wednesdays spent hacking away with a colleague. Software may also act as a humorous, satirical piece on socio-political issues. In this paper, we discover the different ways in which software can be a source of developer amusement and engagement, and help foster a sense of community among them.
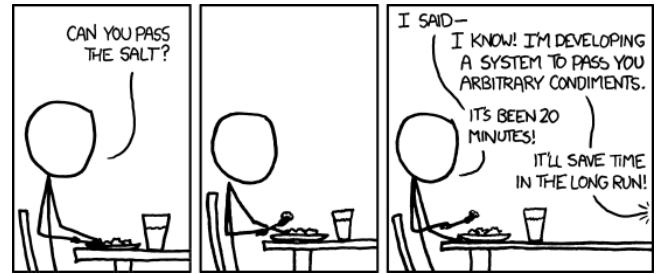
1 INTRODUCTION

Figure 1: Comic strips are classics in software engineering [1]. Credits: xkcd.

The development of high quality software is certainly no laughing matter. In addition to core technical problems, a major challenge of software engineering is its highly collaborative nature, which requires care and engagement. To achieve this, different organizations create a culture of engagement through various corporate strategies. For example, OpenBSD developers celebrate each major release through a new song [1]. Bluechip Silicon Valley company Stack Overflow choreographed the Dance Dance Authentication scheme [2] as company-wide software security training. Guidelines for organizing release parties exist for celebrating major milestones at Ubuntu [2], Fedora [3], and Debian [4]. Across companies, developer communities thrive within dedicated online channels [3] and through the gamification of the software engineering process [4].

---

[1] https://www.openbsd.org/lyrics.html
[2] https://wiki.ubuntu.com/BuildingCommunity/RunningReleaseParty/
[3] https://docs.fedoraproject.org/en-US/mindshare-committee/events/release-parties/
[4] https://wiki.debian.org/ReleaseParty

A crucial cultural phenomenon that comfortably predates software by at least 35, 000 years [5] is the practice of humor. Its purpose is to enchant and engage its audience [6, 7], and foster community-building. Humor is fundamental to the human experience [8]. Our enterprising species has incorporated it in most of its creative undertakings, i.e., our spoken and written word, our art, and even traditional engineering [9]. Yet, to the best of our knowledge, the interplay of humor and software engineering remains largely unexplored in the literature.

In this paper, we unveil the phenomenon of developers engaging with software engineering through humor. We start with case study research, and discuss three eminent examples of real-world humor in open-source software: the `faker` library to generate non-boring test data, a tool called `lolcommits` that captivates developers with selfies, and a parodic software tribute for making all tests pass in the CI, aptly named `volkswagen`. We highlight the humorous elements within each of these projects through quantitative and qualitative insights. Next, we conduct a survey, inviting developers to share their personal practice of, and experiences with, humor in the software they create.

Our results distinctly show the nuances of humor within software. The `faker` ecosystem is continually evolving through the new humorous material contributed by enthusiasts. `lolcommits` has been a handy tool for fostering collaboration, celebrating achievement, and reflecting on the passage of time. In addition to a joke, the software engineering community has also gained a useful tool as a side-effect of `volkswagen`. Overall, the community of developers who engage with the humorous elements within these projects is remarkable. Furthermore, 100 of the 125 respondents of our survey testified to experiencing humor within software. They share their valuable insights on practising humor responsibly [10], and we learn that humor is most likely to be found within code comments and test inputs.

The work most closely related to our original inquiry is about team communication [11, 12]. There are rare studies on software-related humor, such as on *Easter eggs* [13, 14], and *esolangs* [15]. Humor and creativity are closely associated, and this also holds for software engineering [16]. The novel aspect of this work is its focus on humor in programming and code, reconciling the rigorous engineering endeavour with our human inclination towards humor.

We summarize our contributions as follows.

- An exploration of the diverse facets of real-world humor in software projects.
- Actionable knowledge from developers on how to practice humor in their art, in order to create engaging developer communities.

In section 2, we dive deep into three illustrious examples of humor in software engineering. Next, section 3 presents the first ever survey targeted towards understanding humoristic developer practices. We discuss prior art on humor-related software in section 4, before concluding this piece in section 5.

2 CASE STUDIES OF SOFTWARE DEVELOPER ENGAGEMENT WITH HUMOR

On a typical workday, a developer writes code to implement a new feature, or to fix a bug. Additionally, a diligent developer writes tests to verify that the new code behaves as expected. The next step is to *commit* these changes to the project. Finally, our developer pushes this commit to the remote server where the project resides. After peer-review, all project-wide incoming commits trigger a sequence of events within the *Continuous Integration* (CI) pipeline. The most important event within this pipeline is to run the whole suite of tests in the project. Only if the test suite executes successfully are the new changes merged into the project. The process of coding, committing, testing, and integrating is perpetual. In this section, we investigate how humor has made its way into every step of this cycle.

We perform a case study based research [17] and analyze three notable, open-source development tools, which introduce humor in tests (subsection 2.1), commits (subsection 2.2), and the CI (subsection 2.3), respectively. For each of these projects, we report quantitative data about its codebase and community engagement, as well as qualitative insights from our manual analysis of the project. We also gather first-hand accounts of the significance of these humorous tools from interviews with developers who maintain or use them.

2.1 Testing: Faking It While Making It

Academics as well as industry practitioners unanimously agree that testing is crucial to developing high quality software [18]. Developers write tests to ensure that the features they have implemented within their system behave as expected, and that the introduction of new code does not introduce regressions. Each test contains components called test inputs that bring the system to a testable state. This involves initializing variables or objects, and setting up the resources required for the test, such as a database. Test inputs are required to trigger, and consequently verify, different software behaviors, yet constructing good test inputs is known to be challenging and time-consuming [19]. A solution to this problem is the use of data generators that synthesize realistic test inputs, such as a collection of customer contact details. These data generators are packaged in so-called *faking libraries*. `faker` is the gold standard for faking in Ruby [5]. It contains more than 20, 000 lines of Ruby code, and as reported in Table 1, has been starred by more than 10, 000 users on GitHub. As of this writing, `faker` provides 232 object generators. Each of these generators has meticulously been documented with examples.

`faker` provides a plethora of fundamental data generators, such as `address`, `barcode`, `date`, and `lorem`. In addition to conventional generators, the 845 contributors of `faker` have deemed it necessary to include generators that support developers with humorous data, such as a `lebowski` quote string generator, or a `funny_name` generator, to obtain `Ben Thair` or `Don Thatt` rather than the boring suspects, `Jane Doe` or `John Doe`. These extraordinary generators in `faker` refer to cultural elements such as movies, TV shows, music, video games, sports, and books, among others. For example, `Faker::TvShows::Seinfeld.business` yields one of the 23 Seinfeld-related business establishments, such as Vandelay Industries [6]. `faker` is meant to bring joy to test inputs.

In order to understand this humorous aspect of `faker`, as well as the technical and creative challenges of maintaining a project of

---

[5]https://rubygems.org/gems/faker
[6]https://github.com/faker-ruby/faker/blob/main/lib/locales/en/seinfeld.yml

Table 1: Humorous components within software projects can create developer engagement. We illustrate this phenomenon through three popular, open-source projects on GitHub: faker, lolcommits, and volkswagen. For each project, we highlight the number of contributors (#Contributors) and stargazers (#Stars), which capture quantitative evidence of developers' engagement with these projects on GitHub. In the Key Fact row, we report project-specific metrics reflecting its admiration among developers. The Quotes have been sourced from developer forums, social networks, and personal websites.

	faker	lolcommits	volkswagen
#Contributors	845	57	13
#Stars	10, 882	4, 606	12, 716
Key Fact	141 funny data generators contributed by 200 developers	Gem downloaded 266, 365 times, Multiple video montages and galleries, e.g., on YouTube	Featured 23 times as top trending on GitHub between 2015 and 2021
Quotes	*"I love the faker gem. It's so useful when you want to populate a dev db."* — xutopia on Reddit *"Love using Faker to populate demos with "real" content vs. just using [Lorem] Ipsum"* — Ecstatic-Leader485 on Reddit *"[Faker] is a work of art"* — @lylo on Twitter / X *"Part of your daily Rails testing diet"* — Longjumping_You_1786 on Reddit	*"I will enable this at home for my side-projects! This is really LOL.* — daGrevis on Hacker News *"I've been running lolcommits for 10 years [...]"* — Mihai, blog post *"I turned on lolcommits so I'd have a record of my working and kept hacking away. lolcommits is seriously underrated [...]"* — Evan, blog post *"This is a great idea - I wonder [why] I never thought of it"* — xmpir on Hacker News	*"I appreciate the author of this library to no end. There is something to be said about software being made in jest."* — birdiesanders on Hacker News *"My god, this is hilarious"* — clircle on Hacker News *"This is a work of genius. Best laugh I've had all week."* — diatu on Reddit *"This package is still one of my favorite things on the internet, ever."* — sensitive-ears on Hacker News

this stature, we interviewed Stefanni Brasil, a core maintainer of the project. Stefanni attributes the vibrancy and allure of faker to Matz, the creator of Ruby. *"Matz wants programmers to be happy."* [7] Speaking about generator contributions to faker, Stefanni remarked, *"I frequently review pull requests from people who want to add their favorite TV show,"* suggesting the emotional driver behind developer contributions. Indeed, through manual analysis, we determine that at least 141 of the 232 generators in faker may be considered unconventional and humorous. Over the 16 years that faker has been in active development, these generators have generously been contributed by 200 open-source enthusiasts. However, performance and engineering constraints must be considered before adding a new generator [8]. For this reason, every contribution to faker follows a template, is accompanied by relevant tests, documentation,

and locale support, and undergoes a stringent review process. Good fakes represent real engineering.

The continuous growth of the faker codebase has also been a subject of discussion among developers, who have previously been conflicted about moving a basic family of generators into a faker-lite gem [9]. They currently lack the data on how generators are used in the wild, which is necessary to ground the selection of a subset of generators for faker-lite. Stefanni notes, *"We only see the number of downloads, but we would really like to know how developers use the generators. Many of these uses may be in private projects, so there is no easy way to know."* Besides, as highlighted by a developer in the GitHub conversation, providing the full suite of generators by default facilitates an easier introduction of fun into software tests. In fact, such a split could complicate things for users.

---

[7] https://youtu.be/oEkJvvGEtB4?t=1795
[8] https://github.com/faker-ruby/faker/issues/2689

[9] https://github.com/faker-ruby/faker/issues/1539

```
1  seeder.create_if_none(Comment, num_comments) do
2    num_comments.times do
3      attributes = {
4        # hashtag hella art party.
5        body_markdown: Faker::Hipster.paragraph( sentence_count: 1),
6        ...
7      }
8      Comment.create!(attributes)
9    end
10 end
11 ...
12 seeder.create_if_none(Page) do
13   5.times do
14     Page.create!(
15       # We need to override the haptic JBOD pixel!
16       title: Faker::Hacker.say_something_smart,
17       ...
18       # The wise animal blends into its surroundings.
19       description: Faker::Books::Dune.quote,
20       ...
21     )
22   end
23 end
```

Listing 1: The developers of the Forem project seed their development database with some humor using faker.

Stefanni guarantees that the humorous generators within faker are safe for the foreseeable future.

We query GitHub to investigate the real-world usage of faker in open-source projects. One notable client of the faker library is a project called Forem [10], which powers a social networking platform for developers. It uses faker to populate its application and database for test execution, such as with the file seeds.rb. This file calls 10 distinct generators of faker. We show an excerpt of this file in Listing 1, where the database is being seeded with some fake Comments and Pages. Several attributes of the two entities, such as the body_markdown of each Comment (line 5), are produced using faker generators. The comments on lines 4, 15, and 18 are sample outputs from the three showcased generators. Note that the title of each Page (line 16) could easily be a Faker::Lorem.sentence. Instead, it is something smart a Hacker would say [11]. Working on software projects for her day job in a consultancy firm, Stefanni shared how she uses faker for testing. *"The good thing about testing is that we can be creative. I often use* cat *names. But my favorite is the* parks_and_recreation *generator. It is my favorite show."*

faker is only one member of a large and rich ecosystem of faking libraries. The PERL Data::Faker module [12] is an essential seed for this ecosystem, which ships with only 6 conventional generators. faker, inspired by Data::Faker, brought the idea of humorous generators, in addition to the conventional ones ported from PERL. This gem eventually triggered the development of other faking libraries, both in Ruby, such as ffaker [13], as well as in other programming languages, including Java, Python, PHP, JavaScript, Haskell, and even C++. Remixing the design of faker, many of these libraries offer diverse sets of vanilla as well as unconventional generators. For example, java-faker [14] has also introduced a new generator to produce quotes from the French comedy show Kaamelott. We

---

[10]https://github.com/forem/forem
[11]https://github.com/faker-ruby/faker/blob/main/lib/locales/en/hacker.yml
[12]https://metacpan.org/pod/Data::Faker
[13]https://github.com/ffaker/ffaker
[14]https://github.com/DiUS/java-faker

argue that the academic community can gain from paying more attention to faking libraries with respect to both testing and humor.

> *Summary of the* faker *case study*
>
> Hundreds of developers have made humorous contributions to faking libraries such as faker. Humor helps foster open-source collaboration beyond company walls. With humorous faking, software testers worldwide are treated with a nugget while debugging a failing test.

2.2 Commits: LOLs All the Way Down

A *commit* can be considered as the atomic unit of contribution to a software project managed by a version control system. A commit includes code changes, as well as a brief textual description of the changes. Each commit may be likened to the snapshot of the project at a point in time, and is uniquely identifiable through its hash. As an example, we can consider the Forem project discussed in subsection 2.1. With commit 944991a, a Forem developer has implemented a new feature, updated related tests, and described these changes with a commit message.

The number of commits made by developers globally on a workday is likely to be in the millions [20]. Given the significance of commits, there is an ecosystem of utilities that aid developers in committing. For example, commit messages may be linted to ensure that they abide by conventions with commitlint [15]. The git-secrets [16] tool scans commits to prevent credentials and secrets from accidentally making their way into the project codebase. A new way to commit is to let generative AI handle the message, such as with aicommits [17] or opencommit [18].

Admittedly, commits drive software development. However, they do not have to be mundane. gitmoji [19] serves as a handy tool for identifying the right emoji for a commit message, such as :egg: for a commit that adds or updates an Easter egg. Another crowd favorite is lolcommits, which has been downloaded upwards of 250, 000 times as of August 2023 [20]. lolcommits is a command-line utility that captures a portrait of a single developer or a group of developers, and attaches it to each commit made to a project. Table 1 presents some details on the lolcommits project. Nearly 60 developers have contributed to it, while the number of stargazers of lolcommits is more than 4, 600.

As noted on its GitHub README, lolcommits was originally a *"joke project"* showcased at a Hack && Tell meetup [21], by its creators Matthew R. and Matthew H. It has since matured, and can now be installed on computers running all major operating systems. When enabled for a project, lolcommits adds a post-commit hook to its .git folder. Then, each time a commit is made, lolcommits triggers a capture from the camera installed on the device. By default, each lolcommit includes the first 11 characters of the commit

---

[15]https://github.com/conventional-changelog/commitlint
[16]https://github.com/awslabs/git-secrets
[17]https://github.com/Nutlope/aicommits
[18]https://github.com/di-sukharev/opencommit
[19]https://github.com/carloscuesta/gitmoji
[20]https://rubygems.org/gems/lolcommits/versions/0.16.3
[21]https://hackandtell.org/

Figure 2: A crowd enjoying `lolcommits` in April, 2023

SHA, as well as the commit message, typed out in meme-like typography. Over the 12 years since its inception, the contributors to `lolcommits` have consistently provided new features, including different modes, typography, and additional plugins. `lolcommits` is committed to developers; it also supports the generation of videos and animated GIFs, which are features much appreciated by its user base.

`lolcommits` has a thriving community. There are creative video montages featuring the `lolcommits` of developers over the years [22]. Several developers also mention using `lolcommits` on their blogs and social pages. We contacted three such ardent users of `lolcommits`. Selena Small and Michael Milewski are colleagues who pair-program together, give talks in conferences across the world, and participate in hackathons. They often feature `lolcommits` in their projects and mentorship material. They also lolcommit live on stage, as part of their technical talks [23]. Evan Travers is a developer who has used `lolcommits` to document his progress on both professional and personal projects. Speaking about how she started using `lolcommits`, Selena recalled, *"As pair programmers, Michael and I constantly share our ideas and setup with each other. That is how I learned about `lolcommits`."* Michael on the other hand happened to be walking by when a colleague at his previous job was making a lolcommit. *"I accidentally found myself in a lolcommit, and thought it was really fun. It works so well!"* Evan discovered `lolcommits` together with his colleague, the developer of a Ruby gem for Urban Dictionary [24]. *We were subscribed to all newsletters about Ruby gems. One of them mentioned `lolcommits`.*

Attaching a photo to a commit may serve multiple purposes. Evan recounts a challenging day at work trying to solve a Heisenbug [21] which caused a broken build. *"It is nice to have a visual journal of how excited or frustrated you felt. While we were working on fixing that evasive bug, `lolcommits` captured photos of colleagues looking over my shoulder, pointing at my screen. We reflected on them over pizzas."* Michael and Selena believe `lolcommits` can help keep

things *"fun and lighthearted"* while working seriously. *"You can pose for a photo with your colleagues, be creative, even use props. It is an immediate celebration of your work!"* Figure 2 presents one of the `lolcommits` authored by Selena and Michael, starring many excited participants. Their own archive of `lolcommits` features several colleagues, amounts to 6.5 GB of disk space, and represents years of work fulfilment. They add that incorporating live `lolcommits` on stage during their talks *"adds an element of fun to our extravagant, technical presentations."* Selena and Michael also share their experience using `lolcommits` while on a Rails Camp [25], a 52-hour excursion in the woods to practice pair-programming. Their goal was to have at least one significant commit with each of the 38 participants during the camp. Selena found that `lolcommits` was instrumental in this context, *"We used it for onboarding new developers."* Towards the end of the camp, most people were excited about taking post-commit photographs [26]. *"Pair-programming is a good way to understand how people work, and `lolcommits` was the cherry on the cake!"*

> **Summary of the `lolcommits` case study**
>
> Developers make millions of commits everyday. Introducing humor through tools such as `lolcommits` is rewarding for developers, and contributes to a celebration of the work done. In addition to journaling with personal selfies, it fosters relationships with colleagues by encouraging in-person, group *usies*.

2.3 CI: But It Works on My CI

Every so often, a large corporation receives some bad press. Volkswagen made headlines in 2015, when it was discovered that they were not completely honest about their manufactured automobiles. The software embedded within their diesel vehicles reported lower emissions of environmental pollutants, specifically during test sessions in the laboratory [22]. This discovery had significant consequences, and key decisions were made such that car manufacturers undergo more scrutiny.

The Volkswagen story resonated with developers who earnestly test their projects within Continuous Integration (CI) pipelines. Some enterprising developers saw the Volkswagen headline as an opportunity to implement their own *defeat devices*. Such a device can tinker with test harnesses, tricking them into believing that all tests in the test suite pass, thus circumventing the CI to return the coveted green `:check_mark:`. The first software project to achieve this challenging technical feat was `phpunit-vw`, built for PHP projects [27]. Since then, such defeat devices for the CI have been implemented across multiple programming languages, including JavaScript, Rust, Ruby, Java, and even C++ [28]. To counter these, there also exists hoaxwagen, which detects if test results in the CI are being manipulated [29].

Developers Kenneth Auchenberg and Thomas Watson implemented `volkswagen` in JavaScript. During our interview with Thomas,

---

[22] https://youtu.be/mdzPI7Id840
[23] https://youtu.be/qCS1CYxOiEo?si=9kzesLgK3TNaCfi0
[24] https://rubygems.org/gems/urban/

[25] https://rails.camp/
[26] https://github.com/failure-driven/blog/blob/master/content/rails-camp/commit-31-given-when-then-steps.md#lolcommit
[27] https://github.com/hugues-m/phpunit-vw
[28] https://github.com/WyriHaximus/awesome-volkswagen
[29] https://github.com/CleanCode-Group/hoaxwagen

```
1  const isCI = require('is-ci');
2  // from arduino/arduino-ide
3  if (!isCI) {
4    console.log('Skipping notarization: not on CI');
5    return;
6  }
7  .........................................................
8  // from salesforce/observable-membrane
9  if (!isCI) {
10   console.error('This script is only meant to run in CI.');
11   process.exit(1);
12 };
13 .........................................................
14 // from videojs/video.js
15 const progress = () => {
16   if (isCI) {
17     return {};
18   }
19   return progressPlugin();
20 }
```

Listing 2: The development of `volkswagen` triggered the development of `is-ci`, which has since found legitimate use cases.

he recalled, *"We were inspired by the PHP version, and wanted to see if it could be done in JavaScript."* They leverage a technique called monkey patching, which dynamically modifies a function at runtime [23]. volkswagen hooks into Node and patches the `require` function. This modified `require` function links to a version of the test framework, which itself is modified to return a successful execution status for all tests. As a safeguard, volkswagen also checks that this monkey patching is activated only in the CI, and that the behavior of the tests is not changed if they are run locally. Thomas clarified, *"On the surface it might seem easy, but it is a challenge. You have to be aware of all testing frameworks, and also make it work with the whole, messy JavaScript ecosystem."* The first release of volkswagen was *"urgent,"* being rolled out within 24 hours after development started. Currently, volkswagen supports at least 18 different CI servers, and is capable of intercepting multiple JavaScript testing frameworks.

Thomas admits that volkswagen is a meant as joke, and does not envision a direct use case for it. Yet, volkswagen is a good candidate for a tool that attacks the supply chain by making tests pass, despite the introduction of malicious code [24]. While working on volkswagen, Thomas implemented a tool called is-ci, which identifies if the execution environment is in fact a CI server. He released is-ci as an independent package on the npm registry [30]. Contrary to volkswagen, is-ci is profoundly useful. We highlight three of the thousands of open-source usages of is-ci in Listing 2. The `arduino-ide` project from `arduino` (lines 2 to 6), and the `observable-membrane` project from `salesforce` (lines 8 to 12), rely on the output of is-ci to perform certain actions only during execution on the CI. Conversely, the `video.js` project (lines 14 to 20) uses is-ci to skip an operation if the execution environment is the CI.

volkswagen has had two worldwide effects. First, it became a project of Internet fame with its purely humorous nature, having been featured on the daily list of top-trending projects on GitHub 23

times between October 2015 and August 2021 [31], as highlighted in Table 1. *"It blew up! Kenneth, whose GitHub profile hosts* volkswagen, *became the top-ranked developer on GitHub while we were trending for the first time,"* Thomas recalled. Second, it triggered the implementation and release of the open-source package is-ci, which responds to an essential CI/CD use case, albeit through a complex web of monkey patching.

> **Summary of the `volkswagen` *case study*
>
> All components of software development can be the subject of humor. For example, thousands of developers have shared a joke about continuous integration. Software developed purely for fun may even lead to the inception of a truly impactful tool, such as is-ci.

2.4 Takeaways from the Case Studies

Our three case studies, faker, lolcommits, and volkswagen, highlight the intricacies of humor within software. Through faker, we see that humor can be incorporated within testing to engage developers and testers. The lolcommits project gives developers the opportunity to celebrate their development journey, and their ongoing collaboration with colleagues in a playful manner. volkswagen is a détournement of a societal issue, unexpectedly leading to a popular and useful open-source package. The community of developers who engage with humorous practices within software development is large, to an extent which is not known in the software engineering research community.

3 DEVELOPER SURVEY

We conduct the very first survey in the software engineering literature focusing on the practice of professional humor by developers. We have distributed a questionnaire through the most active online communities of developers, and collected unique quantitative and qualitative gems about the role of humor in software development.

3.1 Methodology

The developer survey is based on an online questionnaire. First, the questionnaire introduces the context with examples of real-world software humor. This is followed by questions asking the participant about their response to, and experiences with, humor in software. The final section asks for demographic information. We publish our questionnaire on multiple public channels for software developers: the *AskHN* forum on Hacker News [3], a developer chat group on Discord, two developer groups on LinkedIn, and 6 communities on Reddit, including r/programminghumour. Additionally, we share the questionnaire with the developers we interviewed for the case studies. The questionnaire was released online on August 21, 2023 and closed after five weeks. Our supplemental package includes the questionnaire, all responses, and further details [32].

We received 125 responses from participants whose experience with software development ranges from less than a year to up to 40 years. They work across various domains, such as software, finance,

---

[30]https://www.npmjs.com/package/is-ci

[31]https://github.com/larsbijl/trending_archive
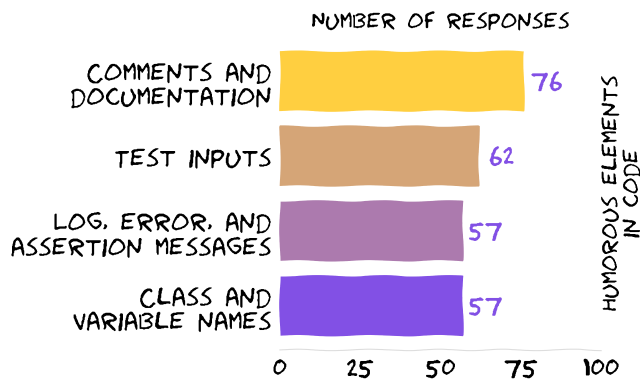[32]https://doi.org/10.5281/zenodo.8386612

Figure 3: According to the survey respondents, the practice of humor in software is appropriate within documentation, immediately followed by test inputs.

communication, and health. Of the 125 participants, 116 (nearly 92.8%) write or review code at least once a week.

The first question is about the reaction of the respondent to humor within software development. More than 93% of the respondents (117 of 125) say that they would react positively, while 8 would be neutral or negative. This fully confirms our core hypothesis that humor is an important channel for developer engagement.

3.2 Where Does Humor Belong in Code?

One important gap in the body of knowledge on software humor is about where it can be practiced. Our survey contains a dedicated section about this. The results are shown in Figure 3, which summarizes the top locations within software projects where developers engage with both code and humor. Of the 100 developers who reported having personally experienced software humor, 76 say that **comments and documentation** are the best locations to unleash it. Next comes a very special place in the heart of software humor practitioners: **test inputs**, with 62 participants finding it appealing for humor. This aligns with the `faker` case study of subsection 2.1, showing that cultural references and fun quotes are appropriate test data. Indeed, several participants mention `faker` as one of the libraries that provides this opportunity.

> *My humorous documentation is fully handcrafted.*

> *I use test data to run wild with humor since it does really feel like the place for it.*

Per Figure 3, two other locations for the introduction of humor are within log messages, as well as the names of code elements such as variable, classes, or functions. One developer shares their favorite example, the `phone_home` function within the `et` module of the Erlang standard library [33]. Additionally, several developers mentioned other ways, outside of code elements, for sharing a joke with a colleague: within code review comments or inside Slack channels.

[33]https://www.erlang.org/doc/man/et.html

> *Developers need to write* `et.phone_home` *to use that function, which is still funny to me.*

> *I incorporate humor within library names. One library ensured only one process of a specific type was alive in a cluster. It was named Highlander, after the movie in which only one of the immortals was supposed to stay alive... "There can be only one."*

The responses on our survey indicate that developers leverage humorous code to breathe life into their software projects. They favor tests and documentation for placing humor. From a social perspective, many developers admire humorous code authored by other developers.

Finally, the astute reader notices that we prepared the bar chart of Figure 3 in xkcd fashion using the dedicated `matplotlib.pyplot.xkcd` module. Evidently, xkcd transcends the humorous comic strips; it is a source of inspiration for developers to create humorous, reusable software components such as this library.

3.3 Humor, What is it is Good For?

The key hypothesis of our work is that humor is fundamentally good in software development, to engage teams involved in hard engineering work. Our survey respondents highlight several compelling reasons for including humor within software development.

First, humor is oftentimes enjoyed for its own sake. Developers solve technical challenges everyday, and humor can be be **a cheerful aid**. Humor in code *"helps keep it fun,"* shares one developer.

> *Humor should be there. It really makes my day and makes me smile. I love it and think fondly of people writing that part of the code or comment.*

> *I sometimes listen to an album from a friend which has some fun track names, like "Callback Hell" and "Merge Conflict." It makes my day lighter.*

Second, humor promotes **a sense of community**. For example, one participant recalls feeling connected to, and appreciative of, past and current colleagues who have authored humorous test inputs. One participant also mentions that humor can be an ice-breaker, especially for remote collaborators.

> *It makes a codebase feel more humanized, like it was created by a real person.*

> *My cat named Bob showed up in a call and a colleague asked: "oh, so that's the Bob Cat I've seen in the tests?" LOL.*

In addition to these positive effects, humor can have a practical impact on software development. For example, humor in test cases can facilitate **a common understanding** of the test intention. Funny stories that are invented and shared among colleagues create

bonds that are beneficial when teams face adversity at the software factory.

> *I incorporate something light (vs serious) to make things more fun and a bit easier to understand.*

> *Inventing personas within integration tests seems to have rubbed off in my team and we now reference them when discussing the testing approach for new features. It helps lighten the mood and make some boring tasks a bit fun.*

Overall, our survey respondents confirm that humor is a good way of increasing developer engagement. Humor makes the engineering process exciting, lightens the mood, promotes bonding, and may even contribute to more understandable code.

3.4 Responsible Humor?

With great humor, comes great responsibility. Developers balance the benefits of humor with other aspects of professional software development. Some respondents of our survey avoid incorporating humor within software, because they think that it may be detrimental to its quality.

> *I actively discourage my teams from getting cute because it often reduces readability and knowledge transfer.*

> *Adding extra dependencies for humor would be a big no for me.*

Meanwhile, many developers take a more nuanced approach to humor within software, and recommend exercising caution when practising it. For example, the location for the introduction of humor within code should be chosen wisely, considering its **impact on performance and quality**.

> *Code functionality should not be affected by humor, but documentation or comments are a good place for that whenever appropriate.*

> *I avoid writing humorous code that could be exposed to the end user, and ensure that my humorous tests are still straightforward and readable. No jokes just for the sake of jokes.*

For a responsible practice of humor, it is important to be mindful of the audience. Humor should not hinder **respectful collaboration**. Indeed, code review is important for risk management in humor, as highlighted by Stefanni (cf. subsection 2.1). *"I ensure that no contribution to* faker *is offensive, which can be hard to do with generators like* south_park."

> *Humor in software projects, especially pop-culture references, should only be incorporated if you are cautious and diligent. It should not create a toxic or unwelcoming culture.*

To sum up, our survey respondents offer actionable advice, and many aspects discussed here could readily be adapted to fit in the "Code of Conduct" or "Contribution guidelines" of developer communities. Clearly, humorous code should be respectful, in line with the global social responsibility theory [25].

3.5 Summary

Our three case studies and survey enable us to provide guidelines for developers and managers who want to introduce or encourage humor within their team:

- Code comments are perfect for expressing humanity and humor, with guarantees that humor does not impact the final product. For example, the comments in the Apollo 11 source code authored by Margaret Hamilton and her crew did not prevent a successful moonshot, as we see from Listing 3.
- Test inputs are appropriate for humorous creativity in code, and developers can rely on state of the art open source libraries to support this practice in different technical stacks.
- The more challenging the development task, the more tech leads should resort to humor as a facilitator of communication and collaboration, see Listing 3.

Summary of the developer survey

The developers surveyed across different continents and domains admit to expressing a bit of themselves through software humor. They highlight that a responsible practice of software humor does not adversely impact code quality and the community. The introduction of humor within software documentation and tests is an organic way to positively engage developers.

---

4 RELATED WORK ON HUMOR IN SOFTWARE

We now discuss related forms of software humor that enchant all members of society, and not just developers.

Software Easter Eggs. They are one of the oldest forms of humor in software [26]. This form of software humor consists of small pieces of code, hidden deep within commercial applications, meant to be fun and surprising. Over the decades, Easter eggs have been creative interludes for developers and true enchantment for users. Today, it is possible to find software Easter eggs in all sectors [14], from video games [27] to cars [28], and web browsers [13]. The smooth acceptance and the longevity of the egg require operational software engineering excellence [29]. In particular, testing Easter eggs is notoriously hard [30]. Cacciotto and colleagues suggest that Easter eggs in graphical interfaces encourage software testers to explore rare behaviors of the interface [31]. Ruus evaluated how the Konami Easter egg can be used to detect cross-browser compatibility bugs [32]. Overall, this literature shows that Easter eggs are positive intakes that favor the engagement of developers.

Humor and Software Development Education. Various studies have investigated the role that humor can play in education [33]. In particular, these studies look at humor to engage students with difficult topics [34], or subjects perceived as boring [35]. Jones documented the social, communication, and cognitive benefits of humor in education [35]. In relation with software, Vogler and colleagues have demonstrated the strong interactions between learning and humor

```
# BURN, BABY, BURN -- MASTER IGNITION ROUTINE
  BANK 36
  SETLOC P40S
  BANK
  EBANK= WHICH
  COUNT* $$/P40

# THE MASTER IGNITION ROUTINE IS DESIGNED FOR USE BY THE
# FOLLOWING LEM PROGRAMS: P12, P40, P42, P61, P63.
...
# HONI SOIT QUI MAL Y PENSE
...
```

Listing 3: Excerpt from the ignition routine in the Apollo 11 code that sent humans to the moon. When Margaret Hamilton and her crew developed the Apollo Guidance Computer, they were also fueled by humor [48]. Software humor predates the invention of "software engineering" as a concept at the 1968 NATO conference [49].

in online teaching [36]. Takbiri and colleagues succeeded in introducing a gamification element to engage with K-6 students and increase their motivation [37]. As for software engineering education, Prasetya and colleagues used various gamification techniques to teach formal methods [38]. Jwo made software engineering education a joyful learning experience using comics to convey the software development lifecycle [39]. Henry Gardner composed a series of humorous songs that reflect the different phases of software development, and has shown that these songs improve the appreciation of software engineering among his students [40]. _why_the_lucky_stiff wrote a book about Ruby, heavily relying on humor and cartoons, to make programming more appealing to adolescents [41].

Pataphysics, Esolangs, etc. The more we search for it, the more we find it. Raczinski explored pataphysics to develop a search engine that returns humorous and provocative search results instead of purely relevant ones [42]. Rick-rollette is a Chromium browser extension that Rickrolls its user with a 1% probability per click [34]. Esoteric languages, aka Esolangs [15], are programming languages with no direct utility except for humor and curiosity [43]. The authors and users of these languages have fun with the core design of programming languages [44]. In the past decade, emoji have become an essential element of conveying humor in software development, on GitHub [45] and Stack Overflow [46]. Clearly, humor is a piece in the grand puzzle of the pursuit of developer happiness, a key factor for code quality and software creativity [47].

5 CONCLUSION

This paper has explored the different ways in which humor is eating software. We have drawn attention to three software projects that are appreciated by developers because of their humorous nature. Libraries such as `faker` help eliminate boredom and invoke creativity within testing. `lolcommits` facilitates the celebration of each contribution made to a project by capturing developer photographs. Useful technology can arise from software made purely for fun, as we have seen with the `volkswagen` project. When handled with care and responsibility, humor is meritorious and inclusive. Real-world practitioners incorporate it within the software that powers all

sectors of modern society. Humor promotes a sense of community within colleagues, and serves as an outlet for self expression.

Our study barely scratches the surface of developer humor, and there is yet much to unearth. For example, research is needed to study the prevalence of memes in software engineering [50]. We have recently received a grant to investigate the dynamics of humor specialization per software stack, grounded on preliminary results on Rust [51] and past research on humor niches in French elite education [52].

ACKNOWLEDGMENTS

---

[34] https://github.com/DaviAMSilva/Rick-Rollette

REFERENCES

[1] Diomara Martins Reigato Barros, Luiz Ricardo Begosso, José Augusto Fabri, and Alexandre L'Erario. The use of comic strips in the teaching of software engineering. In *2017 IEEE Frontiers in Education Conference (FIE)*, pages 1–8. IEEE, 2017.

[2] StackOverflow. Stack overflow unveils the next steps in computer security. https://youtu.be/VgC4b9K-gYU, 2017.

[3] Maurício Aniche, Christoph Treude, Igor Steinmacher, Igor Wiese, Gustavo Pinto, Margaret-Anne Storey, and Marco Aurélio Gerosa. How modern news aggregators help development communities shape and share knowledge. In *Proceedings of the International conference on software engineering (ICSE)*, pages 499–510, 2018.

[4] Klaas-Jan Stol, Mario Schaarschmidt, and Shelly Goldblit. Gamification in software engineering: the mediating role of developer engagement and job satisfaction. *Empirical Software Engineering*, 27(2):35, 2022.

[5] Joseph Polimeni and Jeffrey P Reiss. The first joke: Exploring the evolutionary origins of humor. *Evolutionary psychology*, 4(1):147470490600400129, 2006.

[6] Phil Taylor and Peter Bain. 'Subterranean worksick blues': humour as subversion in two call centres. *Organization studies*, 24(9):1487–1509, 2003.

[7] Robert West and Eric Horvitz. Reverse-engineering Satire, or "Paper on Computational Humor Accepted Despite Making Serious Advances". In *Proceedings of the AAAI conference on Artificial Intelligence*, volume 33, pages 7265–7272, 2019.

[8] Nima Zargham, Vino Avanesi, Leon Reicherts, Ava Elizabeth Scott, Yvonne Rogers, and Rainer Malaka. "Funny How?" A Serious Look at Humor in Conversational Agents. In *Proceedings of the 5th International Conference on Conversational User Interfaces*, pages 1–7, 2023.

[9] Stephen Ogunlana, Chudate Niwawate, Truong Quang, and Le Chien Thang. Effect of humor usage by engineers at construction sites. *Journal of management in engineering*, 22(2):81–88, 2006.

[10] Jason T Peifer. Can we be funny? the social responsibility of political humor. *Journal of Mass Media Ethics*, 27(4):263–276, 2012.

[11] Christopher Mendez, Hema Susmita Padala, Zoe Steine-Hanson, Claudia Hilderbrand, Amber Horvath, Charles Hill, Logan Simpson, Nupoor Patil, Anita Sarma, and Margaret Burnett. Open source barriers to entry, revisited: A sociotechnical perspective. In *Proceedings of the 40th International conference on software engineering*, pages 1004–1015, 2018.

[12] Marco Hoffmann, Daniel Mendez, Fabian Fagerholm, and Anton Luckhardt. The human side of software engineering teams: an investigation of contemporary challenges. *IEEE Transactions on Software Engineering*, 49(1):211–225, 2022.

[13] Benoit Baudry, Tim Toady, and Martin Monperrus. Long live software easter eggs! they are as old as software. *Queue*, 20(2):31–42, 2022.

[14] Matthew Lakier and Daniel Vogel. More than just software surprises: purposes, processes, and directions for software application easter eggs. *Proceedings of the ACM on Human-Computer Interaction*, 6(CSCW1):1–26, 2022.

[15] Daniel Temkin. Language without code: intentionally unusable, uncomputable, or conceptual programming languages. *Journal of Science and Technology of the Arts*, 9(3):83–91, 2017.

[16] Wouter Groeneveld, Laurens Luyten, Joost Vennekens, and Kris Aerts. Exploring the role of creativity in software engineering. In *Proceedings of the Software Engineering in Society (SEIS) track at ICSE*, pages 1–9, 2021.

[17] Bent Flyvbjerg. Five misunderstandings about case-study research. *Qualitative inquiry*, 12(2):219–245, 2006.

[18] Nelson Elhage. Testing as communication. *Increment Magazine*, Testing(10), 2019.

[19] Shauvik Roy Choudhary, Alessandra Gorla, and Alessandro Orso. Automated test input generation for android: Are we there yet?(e). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 429–440. IEEE, 2015.

[20] Yangyang Zhao, Alexander Serebrenik, Yuming Zhou, Vladimir Filkov, and Bogdan Vasilescu. The impact of continuous integration on other software development practices: a large-scale empirical study. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 60–71. IEEE, 2017.

[21] Madanlal Musuvathi, Shaz Qadeer, Thomas Ball, Gerard Basler, Piramanayagam Arumuga Nainar, and Iulian Neamtiu. Finding and reproducing heisenbugs in concurrent programs. In *OSDI*, volume 8, 2008.

[22] Quirin Schiermeier. The science behind the Volkswagen emissions scandal. *Nature*, 9:24, 2015.

[23] Brian Pfretzschner and Lotfi ben Othmane. Identification of dependency-based attacks on node. js. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, pages 1–6, 2017.

[24] Santiago Torres-Arias, Hammad Afzali, Trishank Karthik Kuppusamy, Reza Curtmola, and Justin Cappos. in-toto: Providing farm-to-table guarantees for bits and bytes. In *Proc. of USENIX Security*, pages 1393–1410, 2019.

[25] Clifford Christians and Kaarle Nordenstreng. Social responsibility worldwide. *Journal of mass media ethics*, 19(1):3–28, 2004.

[26] Rodrigo B Salvador. History's first easter egg. *J Geek Stud*, 4(2):63–68, 2017.

[27] Battlefield V. Exploding whale easter egg. https://www.youtube.com/watch?v=JZPzT2AIR0o&t=101s, 2018.

[28] Emma Saldner and Ottilia Linde. How to design easter eggs for a car brand to enhance user experience. MSc thesis, Computer Science Department at Chalmers University of Technology, 2020.

[29] Thomas A Limoncelli. Operational excellence in april fools' pranks. *Communications of the ACM*, 61(3):54–57, 2018.

[30] Qing Xie and Atif M Memon. Using a pilot study to derive a gui model for automated testing. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 18(2):1–35, 2008.

[31] Filippo Cacciotto, Tommaso Fulcini, Riccardo Coppola, and Luca Ardito. A metric framework for the gamification of web and mobile gui testing. In *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 126–129, 2021.

[32] Risko Ruus. Hunting down easter eggs online by exploiting cross-browser compatibility issues: The konami code experiment. BSc thesis, Tartu University, 2011.

[33] Carl Christman. *Instructor Humor as a Tool to Increase Student Engagement.* PhD thesis, California State University, San Bernardino, 2018.

[34] Mary Chabeli, Jackie Malesela, and Monica Rasepae. Humour to facilitate meaningful learning in nursing education as experienced by learner nurses. *Journal of Education*, (59):89–114, 2015.

[35] George H Jones et al. Humor to the rescue: how to make introductory economics an appealing social science for non-majors. *American Journal of Business Education (AJBE)*, 7(2):151–156, 2014.

[36] Jane S Vogler, Sonya E Munsell, and Douglas Knutson. Lolsquared: When laughing-out-loud and learning-on-line intermingle in a computer-mediated classroom discussion. *Computers & Education*, 140:103597, 2019.

[37] Yazdan Takbiri, Azam Bastanfard, and Amineh Amini. A gamified approach for improving the learning performance of k-6 students using easter eggs. *Multimedia Tools and Applications*, pages 1–19, 2023.

[38] I. S. W. B. Prasetya, Craig Q. H. D. Leek, Orestis Melkonian, Joris ten Tusscher, Jan van Bergen, J. M. Everink, Thomas van der Klis, Rick Meijerink, Roan Oosenbrug, Jelle J. Oostveen, Tijmen van den Pol, and Wink M. van Zon. Having fun in learning formal specifications. In Sarah Beecham and Daniela E. Damian, editors, *Proceedings of the International Conference on Software Engineering, Software Engineering Education and Training track, ICSE-SEIT*, pages 192–196, 2019.

[39] Jung-Sing Jwo. Teaching software engineering through comics and micro-movie design projects. In *2015 10th International Conference on Computer Science & Education (ICCSE)*, pages 328–331. IEEE, 2015.

[40] Henry Gardner. Humorous songs to accompany the teaching of software engineering. 2021. http://users.cecs.anu.edu.au/~Henry.Gardner/Gardner_Humourous_Songs_Education_Preprint.pdf.

[41] _why the lucky stiff. *Why's Poignant Guide to Ruby, Tenderly written and illustrated by _why the lucky stiff*. 2005. https://poignant.guide/.

[42] Fania Raczinski, Hongji Yang, and Andrew Hugill. Creative search using pataphysics. In *Proceedings of the 9th ACM Conference on Creativity & Cognition*, pages 274–280, 2013.

[43] List of esoteric languages. https://esolangs.org/wiki/Category:Joke_languages. Accessed: October, 2023.

[44] Michael Mateas. Weird languages. In *Software Studies\A lexicon*, pages 267–276. MIT Press, 2008.

[45] Dong Wang, Tao Xiao, Teyon Son, Raula Gaikovina Kula, Takashi Ishio, Yasutaka Kamei, and Kenichi Matsumoto. More than react: Investigating the role of emoji reaction in github pull requests. *Empirical Software Engineering*, 28(5):123, 2023.

[46] Akhila Sri Manasa Venigalla and Sridhar Chimalakonda. StackEmo: Towards Enhancing User Experience by Augmenting Stack Overflow with Emojis. In *Proceedings of the European Software Engineering Conference / Foundations of Software Engineering (ESEC/FSE)*, pages 1550–1554, 2021.

[47] Daniel Graziotin, Fabian Fagerholm, Xiaofeng Wang, and Pekka Abrahamsson. What happens when software developers are (un) happy. *Journal of Systems and Software*, 140:32–47, 2018.

[48] Keith Collins. "BURN, BABY! BURN!": The code that took America to the moon was just published to GitHub, and it's like a 1960s time capsule. https://qz.com/726338/the-code-that-took-america-to-the-moon-was-just-published-to-github-and-its-like-a-1960s-time-capsule, 2016.

[49] Peter Naur and Brian Randell. Software engineering: Report of a conference sponsored by the nato science committee, garmisch, germany, 7th-11th october 1968. 1969.

[50] Constance Iloh. Do it for the Culture: The Case for Memes in Qualitative Research. *International Journal of Qualitative Methods*, 20:1 – 10, 2021.

[51] Raph Levien. Moving from Rust to C++. https://raphlinus.github.io/rust/2023/04/01/rust-to-cpp.html, 2023.

[52] Pierre Verschueren. Le rire de la vieille dame? humours, sciences et politiques rue d'ulm de 1945 à 1971. In *La Chose Publique*, pages 183 – 214. Champ Vallon, 2015.