The Walking Packages: A Survival Analysis of ROS Repositories

Juliana Freitas[1], Elijah Phifer[1] and Felipe Fronchetti[1]

[1]Louisiana State University, Baton Rouge, Louisiana, USA

Abstract

The Robot Operating System (ROS) serves as a primary foundation for system development in the robotics domain [1]. Its long-term popularity is often correlated with the development of its packages, maintained by internal and external contributors in open-source software (OSS) repositories [2]. While ROS packages are considered a vital part of ROS, many cease activity for a still unexplored set of circumstances. Previous research on OSS development has shown that the lifecycle of open-source projects is often unstable, with many repositories transitioning through three states: active, zombie (where superficial contributions continue but do not meaningfully advance the software), and inactive [3]. In an attempt to better understand why ROS packages become inactive (or turn into walking packages), we present a work-in-progress survival analysis of 810 ROS2 package repositories hosted on GitHub, examining survival rates over time and repository characteristics associated with inactivity. Our findings show a 5-year survival rate of 77.1% for ROS2 package repositories. Using a Cox Proportional Hazards model (C-index = 0.808), we find that fork activity, newcomer influx, and organizational backing are the strongest protective factors against inactivity. Our preliminary findings also highlight that a substantial portion of the ROS2 ecosystem may already constitute "the walking packages"—projects that enter a *zombie* state, appearing active but lacking meaningful human maintenance.

Keywords

Survival Analysis, Software Repositories, Packages, ROS, OSS

1. Introduction

The Robot Operating System (ROS) has become a widely adopted middleware for modern robotics systems [4]. Its success is closely tied to the collaborative development model of the ROS ecosystem, where developers reuse existing packages to implement complex capabilities such as perception, navigation, and motion planning [5]. Through its modular publish–subscribe architecture and extensive open-source package ecosystem, ROS enables developers to distribute reusable functionality across a wide variety of robotic platforms [6].

However, this strong reliance on reusable packages also introduces sustainability risks. Like many open-source software (OSS) projects, ROS packages are largely maintained by voluntary contributors and external organizations, making them vulnerable to maintainer turnover and declining activity [7, 8]. When packages stop receiving updates, they can create maintenance gaps in robotics applications that depend on them. In robotics systems, such abandonment may lead to compatibility issues with newer ROS distributions, unexpected system behavior, and potential security vulnerabilities [9, 5].

In the broader OSS ecosystem, projects rarely become inactive abruptly. Instead, many transition through intermediate phases where activity persists but meaningful development slows down [3]. Within ROS, such partially maintained repositories may result in what we informally describe as *the walking packages*—projects that appear active but lack consistent human maintenance. These repositories can mislead developers who invest time integrating packages that may later prove unreliable or unmaintained.

Understanding the survival dynamics of ROS repositories is therefore essential for assessing the long-term sustainability of the robotics ecosystem. While prior studies have explored quality issues and collaboration patterns in ROS repositories [5, 9], little is known about their long-term survival behavior. In this work, we conduct a preliminary survival analysis of 810 ROS2 package repositories hosted on

RoSE'26: 8th International Workshop on Robotics Software Engineering, June 1st, 2026, Vienna, Austria.
jfreit4@lsu.edu (J. Freitas); ephife3@lsu.edu (E. Phifer); ffronchetti@lsu.edu (F. Fronchetti)

GitHub to investigate repository longevity and the factors associated with inactivity. Specifically, we address the following research questions:

RQ$_1$ *What is the survival rate of ROS package repositories?*
RQ$_2$ *How do the survival states of ROS package repositories change over time?*
RQ$_3$ *Which repository characteristics are associated with package inactivity?*

2. Related Work

Software Engineering in the ROS Ecosystem. Prior research has examined development practices and collaboration patterns in ROS repositories. Kolak et al. [10] show that widely used ROS packages are typically maintained by small, centralized groups of experts, revealing an organizational structure that differs from most OSS ecosystems. In a similar landscape, Pichler [9] shows that ROS packages often suffer from code quality issues and undocumented dependencies. Our goal with this preliminary investigation is to extend prior work on software engineering in ROS by providing an initial investigation of repository activity and the factors contributing to inactivity.

Survival Analysis in Open-Source Software: Survival analysis is a statistical technique used to study the time until an event is expected to occur [11]. Due to the nature of the data used in survival analysis, it is considered a special case of time-series analysis [3]. Survival analysis is used in OSS research to model project activity; existing literature suggests a significant number of projects become inactive within their first year [3]. The survival rate of open-source projects interests both the industries relying on OSS and the contributors deciding which projects to join. The survival of OSS projects has major implications for the sustainability and growth of these industries that rely on OSS [2, 12]. Our work extends previous research by investigating the survival rate of projects in the ROS ecosystem. Because ROS is seen as a *standard* [1] in robotics, the sustainability of this ecosystem is vital. Thus, it serves as a valuable research focus to understand the state of software for robotics.

3. Method

3.1. Repository Selection

To define a clear scope of ROS-related repositories to analyze in our study, we limited our data extraction process to active ROS2 package repositories hosted on GitHub. By *active*, we refer to packages from ROS2 distributions that have not yet reached their end-of-life (EOL), which at the time of data extraction[1] included the Kilted, Jazzy, and Humble distributions. Packages are the fundamental unit of organization, build, and release for software in the ROS ecosystem. They can be developed by anyone, from ROS maintainers to external organizations, and therefore provide an appropriate setting for analyzing software sustainability and survival dynamics.

To identify these packages, we used ROS Index,[2] the official search engine for ROS resources. Across the three active distributions, we extracted 6,242 package–distribution entries corresponding to 2,660 unique packages hosted in 874 GitHub repositories. Because a single repository may contain multiple packages, 287 repositories (35.4%) host more than one package. To ensure a minimum observation period and mitigate the well-known perils of mining GitHub and ensure that our dataset contains relevant engineered projects rather than toy repositories [13], we applied strict exclusion criteria. We removed repositories that were: (a) archived, (b) forks of other repositories, (c) less than six months old, or (d) lacking any detected programming language. This process removed 64 repositories (4 archived, 25 forks, 24 too recent, 15 with no languages), yielding a final dataset of **810 unique GitHub repositories** containing 2,250 ROS2 packages.

[1] Data extraction: March 2026.
[2] https://index.ros.org/

3.2. Survival Analysis

To support the survival analysis, we constructed structured event tables from GitHub activity logs. For each repository, we collected full activity histories via the GitHub REST API, capturing five event types: *commits*, *issues*, *pull requests*, *issue comments*, and *pull-request reviews*. Each event record includes the repository identifier, the event timestamp, and the account type of the author, allowing us to reconstruct the temporal activity history of each project. Account types are determined using the API's native `type` field: *Human* (`type = "User"`), *Bot* (`type = "Bot"`, e.g., Dependabot), or *Organization* (`type = "Organization"`). This classification is used throughout the analysis: in the state machine (RQ_2, Figure 2), only commits authored by *Human* accounts qualify a month as *Running*.

Inactivity definition. Following [3], we classify a repository as inactive when it shows no recorded activity for more than 180 consecutive days before the study end date. A repository therefore becomes inactive only when all forms of engagement cease. Repositories that remain active are treated as right-censored observations. The time-to-event is measured from repository creation to the date of the last recorded activity (for inactive repositories) or to the study end date (for active ones).

Survival rate. We estimate the survival function $S(t)$, defined as the probability that a repository remains active beyond time t, using the *Kaplan-Meier* (KM) estimator [11] implemented in the `scikit-survival` library. We compute $S(t)$ for the full dataset and separately for contributor tiers defined by the interquartile range (IQR) of contributor counts (Q1 = 3, Q3 = 18): Tier 1 (≤ 3 contributors), Tier 2 ($\in [4, 18]$ contributors), and Tier 3 (> 18 contributors). Statistical differences between groups are assessed using the log-rank test.

State evolution. To characterize how repositories cycle between activity levels over time, we model each repository as a three-state machine with states *Running* (at least one commit by a human author in the current month), *Zombie* (no human-authored commits in the current month, but at least one other recorded activity event—bot commits, issues, pull requests, comments, or pull-request reviews), and *Dead* (no recorded activity of any kind in that month). For each repository and calendar month we assign a state and compute transition probabilities across the dataset.

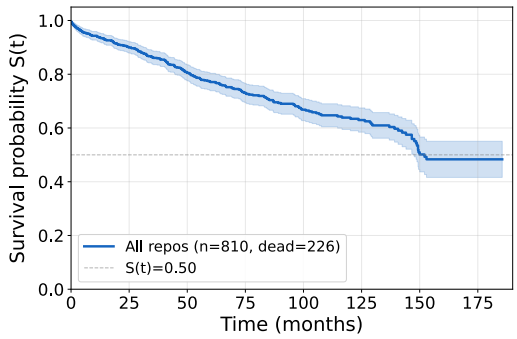
Inactivity factors. While the Kaplan-Meier estimator provides a visual representation of survival probabilities, we fit a *Cox Proportional Hazards* (CoxPH) model [11] using the `scikit-survival` library to quantify the simultaneous effects of repository characteristics on the risk of project abandonment. Covariates include activity rate features (commits per month, newcomers per month, forks per month), organization ownership, and governance artifact presence (README, CONTRIBUTING, PULL_REQUEST_TEMPLATE, and newcomer-friendly labels), each median-imputed and standardized prior to model fitting. A small L2 penalty ($\alpha = 0.01$) is applied to stabilize coefficient estimates. The direction and magnitude of each covariate's association with inactivity risk are summarized using Hazard Ratios (HR). In the survival analysis literature, the HR represents a relative risk of abandonment: $HR < 1$ indicates a protective effect corresponding to a decreased risk of inactivity, while $HR > 1$ indicates an increased hazard.

4. Preliminary Results

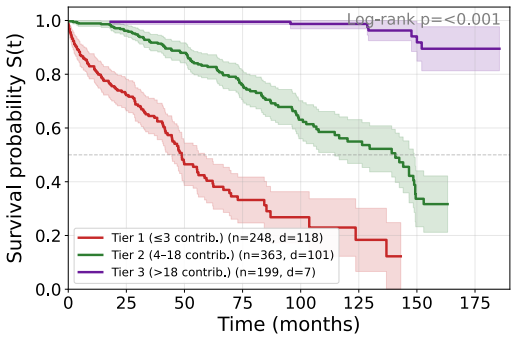
(RQ1) What is the survival rate of ROS package repositories?

In this research question, we analyze the survival probability over time for all ROS repositories in our study. Figure 1a illustrates this probability across the entire dataset. As shown, ROS repositories exhibit a gradual but continuous decline in activity over time. The overall survival rate is high: 77.1% of repositories remain active at the five-year mark, and only 226 (27.9%) became inactive by the study end date. The median survival time exceeds the observation window, meaning that most packages never fully transition to dead during the study period. These results compare favorably to broader OSS ecosystems—Ait et al. [3] reported a 5-year survival rate of 73% for general GitHub repositories—suggesting that ROS packages benefit from stronger institutional and community support.

Figure 1b extends this analysis by stratifying survival rates across the three predefined contributor



(a) Kaplan-Meier estimations of the survival curve considered over all repositories.



(b) Kaplan-Meier estimations grouped by the number of contributors per package repository.

Figure 1: The probability of survival over time for the package repositories analyzed.

tiers. Tier 1 projects (≤ 3 contributors) exhibit significantly shorter lifespans: their survival probability drops to 50.7% by month 48 and falls further to 40.4% by month 60, crossing below the 50% threshold within the observation window. In contrast, Tier 2 and Tier 3 projects demonstrate much stronger resilience, reaching $S(60\text{m}) = 0.827$ and $S(60\text{m}) = 0.995$, respectively, without ever dropping below 50%. These findings indicate that projects with more contributors survive significantly longer—a pattern consistently reported in OSS research [3, 14]. Package repositories that fail to attract new contributors tend to stagnate and become inactive, underscoring the critical role of newcomer onboarding and community support in ensuring long-term package sustainability [14, 15].

(RQ2, RQ3) How do the survival states of ROS package repositories change over time, and which repository characteristics are associated with package inactivity?

Figure 2: State machine representing the evolution paths[3] of repositories.

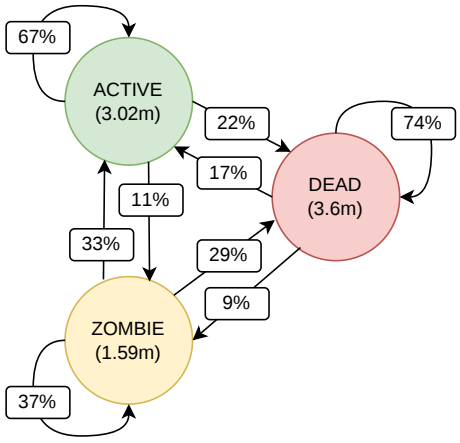


Table 1: CoxPH predictors of repository inactivity with bootstrap 95% CI.

Feature	HR	95% CI	p
Protective (HR < 1)			
Forks/month	0.02	0.00–0.17	0.026*
Newcomers/month	0.31	0.10–0.93	0.040*
Is Organization	0.50	0.34–0.72	< 0.001***
Commits/month	0.53	0.05–1.50	0.496
Risk (HR > 1)			
Has Newcomer Labels	4.98	3.69–7.57	< 0.001***

For the second research question, we model the activity evolution of ROS2 package repositories using the state machine shown in Figure 2, following the methodology of Ait et al. [3]. On average, repositories remain in the *Running* state for 3.02 months before transitioning. In the subsequent period, 67% of repositories remain running, while 11% move to the *Zombie* state and 22% transition directly to dead. The *Zombie* state is typically short-lived, lasting only 1.59 months on average. However, it represents an important transitional phase: 33% of zombie repositories return to the running state, while 29% transition to inactivity and 37% remain in the zombie state. Once a repository becomes inactive,

[3]State labels indicate the name of the status and the average time (in months). Transition labels indicate the probability to change status each month.

the state tends to persist. In fact, 74% of dead repository-months are followed by continued inactivity. Nevertheless, 17% of dead repositories eventually recover and become running again, indicating that inactivity is sometimes temporary rather than permanent. This observation aligns with prior work showing that open-source project abandonment is often gradual and occasionally reversible [16].

To understand the drivers behind these transitions in the ROS2 ecosystem, we analyze the repository characteristics associated with package inactivity ($\mathbf{RQ}_3$). Table 1 summarize the most influential factors identified by the Cox Proportional Hazards model. Our preliminary model achieved a C-index of 0.808 on the held-out test set, indicating a good ability to discriminate between long-lived and short-lived repositories. Among all features, fork rate emerges as the strongest protective factor ($\Delta C = 0.099$). Repositories with higher fork rates are significantly less likely to become inactive (HR = 0.016, $p = 0.026$), suggesting that active reuse and community engagement help sustain maintenance over time. Newcomer onboarding in ROS2 packages also shows a strong protective effect (HR = 0.309, $p = 0.040$): repositories that consistently attract new contributors are substantially more resilient to inactivity. This finding aligns with previous OSS research, demonstrating that contributor onboarding and community growth are critical for long-term sustainability [14, 15]. Organizational ownership also reduces inactivity risk (HR = 0.501, $p < 0.001$), consistent with prior evidence that institution-backed projects tend to have longer lifespans [3]. While commit rate does not reach statistical significance (HR = 0.530, $p = 0.496$), its protective direction and moderate hazard ratio suggest a potential association worthy of further investigation with larger sample sizes.

On the other side, we observe a counterintuitive effect for newcomer labels: their presence is associated with a markedly higher risk of inactivity (HR = 4.979, $p < 0.001$). We hypothesize this reflects reactive adoption: labels may be introduced after activity has already begun to decline, rather than proactively [17]. Further research on newcomer-oriented labels is necessary. Overall, our findings indicate that long-term sustainability of ROS2 package repositories is primarily driven by continuous newcomer recruitment, active ecosystem engagement through forks, and institutional backing [14, 15].

5. Conclusion and Future Steps

Our preliminary findings suggest that ROS2 repositories are relatively resilient compared to general OSS ecosystems, yet a non-trivial portion of them show signs of becoming “walking packages.” The data consistently highlights newcomer onboarding and community engagement as key differentiators: repositories that continuously attract new contributors and maintain active fork activity survive significantly longer and are less likely to enter "*zombie*" or inactive states. Organizational ownership further reinforces this pattern, while the presence of governance artifacts alone does not appear to protect packages from inactivity without an engaged community behind them.

For future work, we plan to deepen this analysis by extending it to other ROS-based repositories. For instance, we plan to investigate whether the ROS1-to-ROS2 transition has left behind a growing number of abandoned packages, and what other factors influence the survival of these projects. We also aim to explore dependency networks to better understand cascading inactivity risk, where the abandonment of a widely used package may threaten the sustainability of its dependents. Finally, we intend to complement these quantitative findings with qualitative interviews with ROS package maintainers, to better understand the human and organizational factors that ultimately determine whether a package survives or becomes just another walking package. The code and data used in this preliminary investigation are publicly available in our replication package[4].

Declaration on Generative AI

During the preparation of this work, the author(s) used Claude Sonnet 4.6 in order to: Grammar and spell check, Rephrase. After using these tools/services, the authors reviewed and edited the content as needed and take full responsibility for the publication’s content.

[4] https://github.com/riseatlsu/rose-2026-survival

References

[1] M. Albonico, M. Đorđević, E. Hamer, I. Malavolta, Software engineering research on the Robot Operating System: A systematic mapping study, Journal of Systems and Software 197 (2023).

[2] K. Crowston, J. Howison, Assessing the health of open source communities, Computer 39 (2006).

[3] A. Ait, J. L. C. Izquierdo, J. Cabot, An empirical study on the survival rate of github projects, in: Proceedings of the 19th International Conference on Mining Software Repositories, MSR '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 365–375.

[4] I. Malavolta, G. Lewis, B. Schmerl, P. Lago, D. Garlan, How do you architect your robots? state of the practice and guidelines for ros-based systems, in: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Practice, 2020.

[5] P. Estefo, J. Simmonds, R. Robbes, J. Fabry, The robot operating system: Package reuse and community dynamics, Journal of Systems and Software 151 (2019) 226–242.

[6] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, Sharing software with ros, IEEE Robotics Automation Magazine 17 (2009) 14–15.

[7] C. Miller, C. Kästner, B. Vasilescu, "we feel like we're winging it:" a study on navigating open-source dependency abandonment, in: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2023.

[8] I. Steinmacher, View Profile, T. U. Conte, View Profile, C. Treude, View Profile, M. A. Gerosa, View Profile, Overcoming open source project entry barriers with a portal for newcomers, in: Proceedings of the 38th International Conference on Software Engineering, ACM Conferences, 2016, pp. 273–284.

[9] M. Pichler, B. Dieber, M. Pinzger, Can i depend on you? mapping the dependency and quality landscape of ros packages, in: 2019 third IEEE international conference on robotic computing (IRC), IEEE, 2019, pp. 78–85.

[10] S. Kolak, A. Afzal, C. Le Goues, M. Hilton, C. S. Timperley, It takes a village to build a robot: An empirical study of the ros ecosystem, in: 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME), IEEE, 2020, pp. 430–440.

[11] D. G. Kleinbaum, M. Klein, Survival analysis a self-learning text, Springer, 1996.

[12] J. Marsan, M. Templier, P. Marois, B. Adams, K. Carillo, G. L. Mopenza, Toward solving social and technical problems in open source software ecosystems: using cause-and-effect analysis to disentangle the causes of complex problems, IEEE Software 36 (2018) 34–41.

[13] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, D. Damian, The promises and perils of mining github, in: Proceedings of the 11th Working Conference on Mining Software Repositories, MSR 2014, Association for Computing Machinery, New York, NY, USA, 2014.

[14] I. Steinmacher, M. A. Graciotto Silva, M. A. Gerosa, D. F. Redmiles, A systematic literature review on the barriers faced by newcomers to open source software projects, Information and Software Technology 59 (2015) 67–85.

[15] F. Fronchetti, I. Wiese, G. Pinto, I. Steinmacher, What Attracts Newcomers to Onboard on OSS Projects? TL;DR: Popularity, in: F. Bordeleau, A. Sillitti, P. Meirelles, V. Lenarduzzi (Eds.), Open Source Systems, Springer International Publishing, Cham, 2019, pp. 91–103.

[16] G. Avelino, E. Constantinou, M. T. Valente, A. Serebrenik, On the abandonment and survival of open source projects: An empirical investigation, in: 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), IEEE, 2019, pp. 1–12.

[17] I. Steinmacher, I. S. Wiese, T. Conte, M. A. Gerosa, D. Redmiles, The hard life of open source software project newcomers, in: Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2014, Association for Computing Machinery, New York, NY, USA, 2014, pp. 72–78.