International Journal of Innovative Research in Engineering and Management (IJIREM)
ISSN (Online): 2350-0557, Volume 12, Issue 6, December 2025
DOI: https://doi.org/10.55524/ijirem.2025.12.6.11
Article ID IJIRE-1417, Pages 61-64
www.ijirem.org

Open-Source Software: The Invisible Engine of the Modern World

Pavan Kuntal[1], and Dr. Shweta Sinha[2]

[1] MCA Scholar, Amity Institute of Information Technology, Amity University Gurugram, Haryana, India
[2]Associate Professor, Department of Computer Science and Engineering, Amity University, Gurugram, Haryana, India

Correspondence should be addressed to Pavan Kuntal ; pavankuntal27@gmail.com

Received: 1 November 2025 Revised: 14 November 2025 Accepted: 28 November 2025

Copyright © 2025 Made Pavan Kuntal.et al. This is an open-access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

ABSTRACT- Open-source software once stayed hidden today, it quietly drives nearly everything digital around us. Think of Linux, Apache, Docker, or Kubernet es; they handle big parts of what happens online, from watching shows to moving money or buying stuff. As businesses move toward cloud setups and spread-out systems, using shared code isn't just useful anymore - it's a must for scaling up, keeping pace, or simply surviving in the race.
This research investigates how open-source code is built and kept up these days, revealing that when communities lead the work, concepts spread quicker, initial expenses drop significantly, while safety improves since anyone can check what's written. Yet problems remain - shaky parts in software pipelines, fuzzy leadership setups, volunteers stretched too thin managing projects, also legal messes from combining varied licensing rules.
In short, open-source quietly runs today's tech - boosting new ideas yet showing weak spots when tons rely on software looked after by just a few hands. This study mixes those upsides and flaws to show that what makes open-source work isn't only the code, but the humans behind it, how they team up, plus steady backing.

KEYWORDS: Open-Source Software, Cloud-Native Tools, Community-Driven Security, Software Supply Chain, Governance, Scalability, Sustainability

I.INTRODUCTION

Software makings come a long way. Back then, apps were kept private, built strictly within office limits. Today, open-source efforts allow people everywhere to adjust and swap designs freely. Right now, this non-cost tech goes beyond coding, it drives markets across continents. Someone building in Mumbai could shape tools running in a startup from San Francisco, proving worldwide teamwork keeps today's digital world moving [1].
Built on open-source tools, developers skip starting over, instead, they adjust ready-made setups. Yet widespread use brings problems: flaws in common code such as Log4j [2], maintainers working without pay [3], plus shaky licensing terms [4]. We check real-world usage, weigh benefits, then explore tensions between collective aims and profit needs.

II. LITERATURE REVIEW

The story of open source starts with Eric Raymond's 1999 take - the "bazaar," a messy yet free space, versus the "cathedral," controlled and top-down [5]. Back then folks questioned - what makes someone spend effort for no cash? Lately - say, between 2020 and 2025 - focus moved to keeping software supply chains secure: is it safe to use code made by people we don't know? [6]
Recent reports indicate a massive scale of adoption—Synopsys[7] highlights that nearly all modern applications rely on open-source components [7].Yet research by Vidanage et al. [8] shows how one broken tool can trigger chaos across countless systems.
Right now, plenty of groups prefer mixed methods. They create unique tools using public bases but hold tight control over some parts [9].

III. GLOBAL UBIQUITY AND DIGITAL UTILITY

Open-source software runs quietly behind the scenes in nearly all vital parts of today's tech systems - powering everything without being seen. Across different levels of technology, half of big companies worldwide run their operations on open-source OSs; close to that number depend on open tools for cloud setups and containers. Take cloud computing - it's built around shared rules and platforms such as Apache CloudStack or OpenStack, which handle core jobs like setting up virtual servers through Nova or managing block storage via Cinder. Because these pieces work together smoothly, organizations avoid getting trapped by one provider - a major plus when it comes to protecting national interests, according to defense experts. Beyond that, free-to-use code fuels fast progress in emerging fields set to change how we interact with machines
AI plus machine learning? Most core tools - like TensorFlow or PyTorch - are open source, which pushes progress way faster than closed alternatives usually can. What's more, public code means clearer insight into how systems work, making meeting rules easier, while keeping private data firmly in-house.
Decentralization: Blockchain runs on open code, sharing data publicly so no single point fails; it locks info in blocks that can't change - keeping records safe while spreading updates across many nodes instead of one central hub.

IV. WORKFLOW ARCHITECTURE OF THE OSS DEVELOPER MODEL

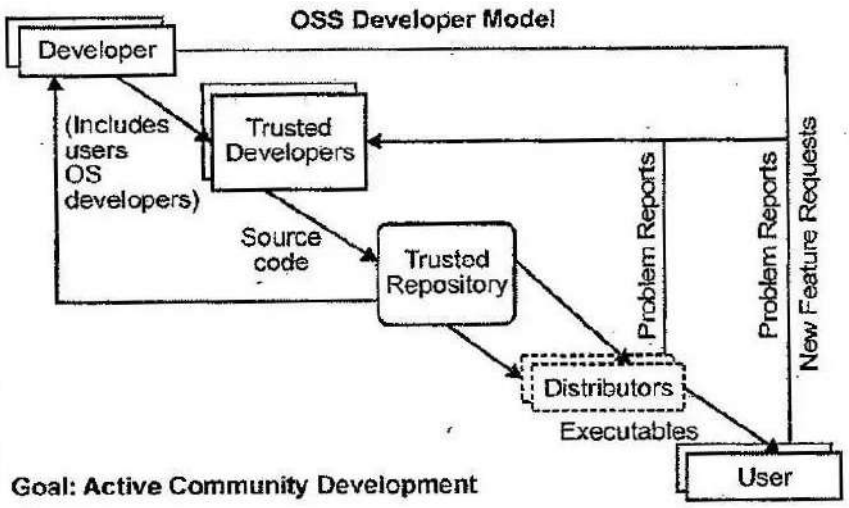

Figure 1: OSS Developer Model [4]

The Open-Source Software (OSS) Developer Model shows in Figure 1, how group-built software works in real-world projects. As noted by Owlgen (n.d.), it starts when coders - sometimes just regular users - submit improvements or fix errors in the code. Once submitted, experienced team members check each change for reliability, safety, and alignment with rules. When approved, these updates go into a secure central hub, becoming part of the main version. From there, packagers turn raw code into ready-to-run programs so everyday people can use them. Users share thoughts by reporting bugs or asking for new tools - this kicks off dev work again. Thanks to this ongoing exchange, projects stay clear, move fast, while people jump in regularly; that's how open-source thrives.

The model shows that duty spreads through the whole group instead of resting on one team. Each person - from coder to end-user - brings something useful, so the tool gets stronger features and works better as time goes by. Working together openly helps people learn from each other, guide newcomers, solve problems as a team, letting beginners improve while keeping the system flexible when new demands pop up. Also, since everyone can see how things are built, bugs get spotted faster and fixed sooner, which means flaws don't stick around hidden for long. Being transparent like this doesn't just gain user confidence - it pushes progress forward quicker, turning open-source into a lasting, fresh approach today's tech world uses.

V. COMPARATIVE ANALYSIS OF OPEN SOURCE AND PROPRIETARY MODELS

The growth of open-source software in companies mostly happens because it outperforms old closed systems - this splits tech into two separate directions. Looking at both reveals why open source became so widespread behind the scenes, while also showing issues linked to oversight and ongoing help. Day-to-day, firms frequently discover that free tools adapt more smoothly to their routines since they tweak the code to suit tasks rather than reshaping operations to fit what a seller offers. On the flip side, plenty of groups stick with private platforms thanks to steady assistance, simpler installation, and having just one contact if something breaks down. These contrasts add up to a real-world compromise: independence plus customization versus ease and clear direction. See the below Table 1.

Table 1: Comparative Analysis of Open Source and Proprietary Models

Feature	Free/Open Software	Closed/Proprietary Software
Cost	Usually free to use with no license fees; long-term costs stay low.	High upfront license fees plus ongoing maintenance costs.
Transparency& Control	Open access to view and modify the code; full control and better privacy through self-hosting	Code is hidden; works like a sealed system leading to lower visibility and trust gaps.
Flexibility & Customization	Highly adaptable; code can be tweaked for specific industries or user-level adjustments.	Limited customization: standard features may not meeta all user needs.
Security & Auditing	Open review by global experts- issues found and resolved quickly.	Often secure due to enterprise-grade encryption and compliance certifications.
Deployment & Support	Requires skilled in-house staff or community consultants; cost remains predictable.	Fast development with strong vendor support and lower setup complexity.

The biggest difference comes down to who's in charge. Because OSS is flexible and free to use, companies can tweak, adapt, or check the code whenever needed - super helpful when building advanced tech like AI. Instead of waiting for updates, teams move faster on their own terms. Proprietary systems give you customer service and ready-made rules, but tie you to one provider, slowing things if demands shift. Fixing issues means asking for permission instead of acting fast. On the flip side, open code lets everyone see inside which helps catch flaws quickly - but also hands hackers a roadmap to exploit weak spots. So, picking between open or closed isn't just technical it's about trading outside helps for more freedom and direct oversight.

VI. CHALLENGES: ETHICAL, LEGAL, AND SECURITY

Because the code's public, hackers can study it - which makes finding flaws simpler.

Maintainer burnout? It's no joke - loads of major projects depend on people volunteering, while dealing with insane pressure.

Licensing becomes tricky if you mix Copyleft with Proprietary - conflicts pop up, lawsuits tag along. The rules clash, they just don't fit.

Supply chain hacks occur if cybercriminals insert malicious scripts into popular open-source software usually by slipping them inside reliable upgrades or hidden components people trust without questioning.

Besides, plenty of businesses run on open-source software while totally clueless about the dangers or duties kind of like operating a high-speed vehicle without ever checking under the engine cover. Groups usually think another person's handling the code yet nobody might be in charge. When a glitch slams into something huge- just like Log4j it sparks chaos fast, showing how one small chunk of software can rattle the whole web.

Then again, there's the moral angle - also worth thinking about. Is it fair that just a few people who aren't paid handle upkeep on tools big companies profit from?

Wondering if those tackling the toughest safety jobs really get paid fairly? Many who handle risky tasks end up with almost nothing. The ones on the front lines usually see the smallest paychecks. Could there be a better way to reward their effort?

In lots of actual situations, businesses fix problems quickly yet skip lasting solutions - so the same flaws pop up repeatedly.

Fundamentally, the issue goes beyond programming - it involves humans, duty, well- being, also the joint frameworks keeping things running.

VII. FUTURE PROSPECTS

Folks are diving into homemade tech - with governments leading the charge toward open- source tools so they rely less on big international firms while dodging ties to outside tech ecosystems. It's not merely about saving money - it's tied up with keeping data under national control, holding power down the line, plus growing skilled workers who can manage essential software locally. At the same time, AI-made coding is tough to overlook now; as automated helpers churn out chunks, sections, or full programs, public code libraries will likely expand faster than ever before. Though it helps get more done, there's a downside - too many weak submissions, fuzzy ownership, or murky license rules. That's why better auto-tools will be key; they'll spot bugs, test if things fit together, highlight risks, plus confirm what's truly original - all to keep big, shared projects running smooth. We might also see hand-picked open-source hubs pop up, where known teams officially approve solid builds of widely used tools. Support could shift too, as businesses start paying upfront for software, they use daily, so devs don't run themselves into the ground. Put it all together, and open source down the line looks less wild, more guided, with clearer global direction.

VIII. CONCLUSION

Free software won the tech war - right now, it powers nearly every new setup. Because of this, expenses drop fast, plus creativity gets a boost. Still, things feel unstable lately. When corporations ignore the tools, they use daily, problems grow - think security leaks or burnt-out developers. Going forward needs a blend: open- source freedom combined with actual accountability. This combo? It'll shape how everything's built from today onward.

Simply put, today's tech relies more on groups of people than big companies - but those groups can't do everything by themselves. Firms using this software every day should step up, either through cash support, sharing fixes, or making systems safer.

Everyday folks? They'll notice it in their gadgets updates that run better, apps you can trust, plus features that keep up when things get busy. As for coders, what's coming feels way more team- driven: mixing free-flowing ideas with systems that hold up, so making new stuff won't leave them drained.

In short, what's coming into software isn't only about open or shut systems - instead, it's shaped by care, backed by teamwork, yet guided by duty.

CONFLICTS OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

[1] B. Fitzgerald, "The transformation of open-source software," *MIS Quarterly*, 2006. Available from: https://www.jstor.org/stable/25148740

[2] Z. Chen *et al.*, "A study on Log4j vulnerability," *IEEE Security & Privacy*, 2022. Available from: https://ieeexplore.ieee.org

[3] K. Crowston and J. Howison, "The organization of open-source projects," *First Monday*, 2005. Available from: https://firstmonday.org

[4] Owlgen, "How OSS developers work,". Available from: https://www.owlgen.com

[5] St. Laurent, *Understanding Open-Source Licensing*, O'Reilly, 2022. Available from: https://www.oreilly.com

[6] E. Raymond, *The Cathedral & the Bazaar*, O'Reilly, 1997. Available from: https://www.oreilly.com

[7] Synopsys, "2024 Open-Source Security and Risk Report," 2024. Available from: https://www.synopsys.com

[8] L. Vidanage *et al.*, "Dependency risks in OSS ecosystems," *IJSE Research*, 2023. Available from: https://www.ijser.org

[9] J. Feller and B. Fitzgerald, *Understanding Open-Source Software Development*, Addison-Wesley, 2002. Available from: https://www.pearson.com