Towards Systematic Treatment of Community-Driven Variability

Roman Bögli*
University of Bern, Switzerland
roman.boegli@unibe.ch

Abstract

Modern decentralized software ecosystems such as Bitcoin evolve through crowdsourced improvement proposals (IPs) that are continuously shaped and autonomously implemented by independent actors. As a result, these ecosystems exhibit *Community-Driven Variability* (CDV), a novel paradigm that extends beyond traditional variability-intensive systems and introduces unsolved challenges and new opportunities. With this thesis, we approach these challenges and step forward to enable a holistic, systematic treatment of this domain. This includes (i) establishing a modeling formalism that adequately captures the versatility of CDV ecosystems, (ii) providing tool support for automated model extraction from the large corpus of open-source IP data, and (iii) integrating these mined models with analyses of concrete implementation derivatives.

Keywords

software families, software interoperability, improvement proposals

1 Introduction

Over the past decades, software engineering has steadily moved from building isolated systems to developing coordinated software families [34]. Methods, tooling, and organizational practices have since been reshaped to manage portfolios of related software variants that share core functionality while offering configuration points for diverse markets and operational contexts [35]. Software variability, generally defined as *"the ability to derive different products from a common set of artefacts"* [5], has become a central concern and is now treated as a first-class engineering capability in modern software development [14, 45].

The most systematic class of approaches for such variability-intensive systems falls under *Software Product-Line* (SPL) engineering [4, 5, 35, 40, 41], where explicit feature models describe reusable software capabilities and variation mechanisms then materialize concrete products based on specific configurations of those models [6, 45]. SPL success stories span embedded control [5] in automotive [16], aerospace [20], railway [1], telecommunications [30], and even the Linux kernel [43]. More liberal practices extend this spectrum from ad-hoc clone-and-own [22, 38, 51], through gradual SPL adoption [17, 21, 23, 24, 27, 29], to feature toggling [28, 37]. While diverging from centrally managed product-line engineering, all these approaches deal with software variability in one way or another.

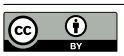
In our preliminary work, we broadened the field by introducing *Community-Driven Variability* (CDV) [12, 13] as a novel paradigm of software variability. Ecosystems that exhibit CDV produce software families through mechanisms fundamentally different from those of classical variability-intensive systems. Being decentralized and without a central authority, their evolution is driven through collectively curated *improvement proposals* (IPs) that serve as open specification documents. These IPs are maintained in distributed repositories across the internet by independent authors and collectively form what we term the *proposal spectrum*. Developers then select IP subsets to produce independent software derivatives, whose diversity spans simple feature variations to entirely different use cases or even technology stacks. Collectively, these variants form what we term the *derivative spectrum*. What unites these derivatives within a CDV ecosystem is their shared interest in interoperability, which substantially drives the ecosystem's usage value. Alongside the decoupled evolution of the crowdsourced proposal and derivative spectra, striving for interoperability is a defining key characteristic of CDV-exhibiting ecosystems [12, 13].

A prominent example of such a CDV-exhibiting ecosystem is Bitcoin [31] with its Bitcoin Improvement Proposals (BIPs) [9]. Numerous application types (wallets, nodes, watchtowers) [10, 11, 46] exist that support different BIP sets, yielding varying user experiences, security properties, and target platforms – yet all interoperate on the same underlying protocol. Similar dynamics exist in the Lightning Network [36] with its BOLTs [25] and bLIPs [26], the decentralized messaging relay system Nostr [32] with its Implementation Possibilities (NIPs) [33], the Tor protocol [15] with its Design Proposals [47], or the InterPlanetary ecosystem with its IPIPs [44]. Acknowledging that CDV attribution unfolds on a continuum, ecosystems such as the Linux kernel [48, 49], Eclipse [2], or Python [18] with its Enhancement Proposals (PEPs) [19] also exhibit some CDV traits, albeit to varying extent.

Software ecosystems that fulfill the constituting CDV characteristics share a set of problems, referred to as P1 through P6 in [12], that remain unaddressed. Five of them will be the subject of this thesis: **P1** and **P2** concern the missing overview within the proposal and derivative spectrum, respectively, manifesting in a lack of general orientation. **P3** captures the challenges actors face when proposing or updating IPs, such as anticipating change impact and potential side effects. **P4** addresses the detection of misalignment between the two spectra, i.e., declared versus actual IP adoption. Finally, **P5** describes the lack of rigorous methods to reveal and test (un)desired feature interactions among derivatives to evaluate interoperability.

This thesis contributes to the CDV domain by addressing these problems through formal variability modeling, automated model mining and investigation, and derivative-aware analysis. Therefore, we pursue three mutually reinforcing objectives outlined in Sec. 2.

*Advisor: Prof. Dr. Timo Kehrer. PhD year at time of symposium: Year 2 of expected 4.



ICSE-Companion '26, Rio de Janeiro, Brazil

ACM ISBN 979-8-4007-2296-7/2026/04
https://doi.org/10.1145/3774748.3787644

2 Research Scope and Objectives

This section outlines our research objectives, expected contributions, and corresponding evaluation plans. While our work initially focuses on the Bitcoin ecosystem, given its motivating role in defining CDV [12, 13], the planned contributions are designed to generalize across the wider continuum of CDV-exhibiting ecosystems.

2.1 Variability Modeling Formalism

The first objective is to devise a variability modeling formalism that adequately reflects how IPs accumulate and interrelate. The conceptual research challenge is to design a modeling notation that supports the superimposition of variability dimensions [3, 7] within the proposal spectrum, and serves as a basis for automated spatio-temporal analysis. As a starting point, we consider unifying existing modeling frameworks such as Hyper Feature Models (HFMs) [42] and Dynamic Feature Transition Systems (DFTS) [39]. HFMs add temporal evolution to feature models, albeit only for basic versioning, and DFTSs provide context-aware transitions – both promising for reflecting the multidimensional nature of CDV.

Expected Contribution. A variability modeling formalism and notation that can adequately capture the versatility of an ecosystem's proposal spectrum and its evolution. Therewith, we provide the foundation for structured representation of an ecosystem's proposal spectrum (P1) and make it amenable to systematic, automated exploration (Sec. 2.2). It further allows conducting differential analysis of CDV models representing different historic or hypothetical snapshots, facilitating IP change impact assessment (P3).

Evaluation Plan. Expressiveness and adequacy will be probed through expert walkthroughs in which practitioners encode known IP structures and validate the resulting models. Effectiveness will be evaluated through differential analyses of concrete model instances by contrasting (i) two Bitcoin snapshots from different points in time and (ii) a historically forked ecosystem that produced a community split [8]. Applying the formalism to additional ecosystems such as Nostr [32, 33] shall test the generalizability of our approach.

2.2 Automated Model Mining and Investigation

The second objective employs a data-driven strategy by mining CDV ecosystem models directly from their respective IP catalogs. As these catalogs are crowdsourced, they are distributed across the internet and document system behavior using unstructured text intended for human consumption. Our envisioned pipeline harvests these catalogs and periodically parses metadata (e.g., status, authorship, affected subsystems) and natural-language text (motivation, specification, test scenarios) to instantiate a holistic view on the ecosystem's proposal spectrum. Building on the modeling formalism (Sec. 2.1), we further envision a mining process that extracts model elements (features, constraints, versioning information) and interrelations (dependencies, conflicts, revisions) from these IPs.

Expected Contribution. A mining and analysis toolchain for structured, explorable representation of the proposal spectrum (P1), including techniques to reason about the structure and constraints of CDV models and to detect anomalous IPs and IP interrelations (P3). Acting as a guidance instrument for ecosystem actors (IP authors, derivative developers, end users), it enhances overall comprehension of the proposal spectrum. Enabled by the formalism, envisioned capabilities expand to temporal analyses, constraint querying, and systematic exploration of the configuration space.

Evaluation Plan. The toolchain shall systematically surface structural anomalies in the proposal spectrum that would be difficult to detect manually. As evolution of IP catalogs is usually supported by Git [50], these detections can be replayed on historical snapshots and compared against the ecosystem's actual developments, enabling us to assess whether predicted shifts (e.g., increasing centrality of an IP) align with observed community behavior. To assess mining accuracy, we will curate labeled IP subsets and evaluate extracted metadata and relations using standard information-retrieval metrics. In addition, we plan to conduct expert surveys to gather practitioner feedback on the tool's usefulness.

2.3 Bridging to Implementation Derivatives

With the third objective, we extend to the derivative spectrum by integrating the CDV model with concrete implementation derivatives. By linking our model formalism (Sec. 2.1) and mining toolchain (Sec. 2.2) to the derivative spectrum, we pave the way toward automatic validation of derivatives against the proposal spectrum.

Expected Contribution. By systematically linking CDV models with concrete implementation derivatives, this objective provides the missing derivative overview (P2). It further establishes a foundation for more rigorous inter-derivative testing, enabling us to systematically reveal interoperability impairments (P5). Automated conformance checks will flag misalignments between claimed and implemented IP subsets (P4), while change impact analyses will reveal which derivatives are affected by modifications in the proposal spectrum (P3).

Evaluation Plan. We assess the technical feasibility of conformance and impact analyses, as well as their practical utility for derivative developers. Effectiveness will be measured by applying our procedures to multiple Bitcoin derivatives across historical snapshots and known software forks, checking whether detected misalignments and inferred impact sets match documented incompatibilities or retrofit activities. Practical utility will be studied through expert feedback from derivative developers who apply our analyses to their own codebases, providing qualitative evidence on usability, precision, and relevance for ecosystem maintenance.

3 Conclusion

Community-Driven Variability (CDV) is an emerging form of distributed software variability that remains largely unexplored and extends beyond traditional variability-intensive systems. Building on our prior introduction of CDV [12], this paper outlines three research objectives that scope and concretize the agenda for this thesis. By delivering reusable modeling foundations, a mining and exploration toolchain, and derivative-aware analyses, the thesis aims to produce actionable guidance and provide a rigorous basis for studying CDV phenomena.

We expect that insights from classical variability research can be productively transferred to CDV settings and, in turn, that CDV's unique challenges will broaden the scope of variability research. In sum, this thesis opens new territory in software-variability research and forms a substantive step toward a systematic treatment of CDV-exhibiting software ecosystems.

References

[1] Muhammad Abbas, Robbert Jongeling, Claes Lindskog, Eduard Paul Enoiu, Mehrdad Saadatmand, and Daniel Sundmark. 2020. Product line adoption in industry: an experience report from the railway domain. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)* Article 3. Montreal, Quebec, Canada, 11 pages. ISBN: 978-1-4503-7569-6. doi:10.1145/3382025.3414953.

[2] Eclipse Foundation AISBL. 2025. Eclipse. Accessed: 2025-07-12. https://www.eclipse.org/home/whatis/.

[3] Sofia Ananieva et al. 2022. A conceptual model for unifying variability in space and time: Rationale, validation, and illustrative applications. *Empirical Software Engineering (EMSE)*, 27, 5, 101. doi:10.1007/S10664-021-10097-Z.

[4] Michał Antkiewicz, Wenbin Ji, Thorsten Berger, Krzysztof Czarnecki, Thomas Schmorleiz, Ralf Lämmel, Stefan Stănciulescu, Andrzej Wasowski, and Ina Schaefer. 2014. Flexible Product Line Engineering With a Virtual Platform. In *Companion Int'l Conf. on Software Engineering (ICSEC)*. ACM, New York, NY, USA, 532–535. doi:10.1145/2591062.2591126.

[5] Sven Apel, Don S. Batory, Christian Kästner, and Gunter Saake. 2013. *Feature-Oriented Software Product Lines - Concepts and Implementation*. Springer. doi:10.1007/978-3-642-37521-7.

[6] Don S. Batory. 2005. Feature Models, Grammars, and Propositional Formulas. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)*. Vol. 3714. Springer, 7–20. doi:10.1007/11554844_3.

[7] Thorsten Berger, Marsha Chechik, Timo Kehrer, and Manuel Wimmer. 2019. Software Evolution in Time and Space: Unifying Version and Variability Management (Dagstuhl Seminar 19191). *Dagstuhl Reports*, 9, 5, 1–30. doi:10.4230/DAGREP.9.5.1.

[8] Jonathan Bier. 2021. *The Blocksize War: The Battle over Who Controls Bitcoin's Protocol Rules*. Independently published, (Mar. 2021). ISBN: 979-8-721-89560-9.

[9] Bitcoin. 2025. Bitcoin Improvement Proposals (BIPs). Retrieved Jan. 16, 2025 from https://github.com/bitcoin/bips.

[10] Bitcoin Optech. 2025. Compatibility Matrix. Retrieved Nov. 13, 2025 from https://bitcoinops.org/en/matrix/.

[11] Roman Bögli. 2023. A Security Focused Outline on Bitcoin Wallets. (Mar. 2023). https://eprints.ost.ch/id/eprint/1103/.

[12] Roman Bögli, Alexander Boll, Alexander Schultheiß, and Timo Kehrer. 2025. Beyond Software Families: Community-Driven Variability. In *Companion Int'l Conf. on the Foundations of Software Engineering (FSE)*. ACM, 571–575. doi:10.1145/3696630.3728501.

[13] Roman Bögli, Alexander Boll, Alexander Schultheiß, and Timo Kehrer. 2026. Community-Driven Variability: Characterizing a new Software Variability Paradigm. *Automated Software Engineering*. Accepted for publication.

[14] Krzysztof Czarnecki and Ulrich W. Eisenecker. 2000. *Generative Programming: Methods, Tools and Applications*. Addison-Wesley. ISBN: 978-0-201-30977-5.

[15] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. 2004. Tor: The Second-Generation Onion Router. In *USENIX Security Symposium*, 303–320.

[16] M. Eggert, K. Günther, J. Maletschek, A. Maxiniuc, and A. Mann-Wahrenberg. 2022. In three steps to software product lines: a practical example from the automotive industry. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)*. ACM, 170–177. doi:10.1145/3546932.3547003.

[17] Wolfram Fenske, Jens Meinicke, Sandro Schulze, Steffen Schulze, and Gunter Saake. 2017. Variant-Preserving Refactorings for Migrating Cloned Products to a Product Line. In *Proc. Int'l Conf. on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 316–326.

[18] Python Software Foundation. 2025. Python. Retrieved Jan. 16, 2025 from https://python.org.

[19] Python Software Foundation. 2025. Python Enhancement Proposals (PEPs). Retrieved Jan. 16, 2025 from https://peps.python.org.

[20] I. Habli, T. Kelly, and I. Hopkins. 2007. Challenges of establishing a software product line for an aerospace engine monitoring system. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)*, 193–202. doi:10.1109/SPLINE.2007.37.

[21] Christian Kästner, Alexander Dreiling, and Klaus Ostermann. 2014. Variability Mining: Consistent Semiautomatic Detection of Product-Line Features. *IEEE Trans. on Software Engineering (TSE)*, 40, 1, 67–82. doi:10.1109/TSE.2013.45.

[22] Timo Kehrer, Thomas Thüm, Alexander Schultheiß, and Paul Maximilian Bittner. 2021. Bridging the Gap Between Clone-and-Own and Software Product Lines. In *Proc. Int'l Conf. on Software Engineering (ICSE)*. IEEE, 21–25. doi:10.1109/ICSE-NIER52604.2021.00013.

[23] Benjamin Klatt, Martin Küster, and Klaus Krogmann. 2013. A Graph-Based Analysis Concept to Derive a Variation Point Design From Product Copies. In *Proc. Int'l Workshop on Reverse Variability Engineering (REVE)*, 1–8.

[24] Jacob Krüger, Mukelabai Mukelabai, Wanzi Gu, Hui Shen, Regina Hebig, and Thorsten Berger. 2019. Where Is My Feature and What Is It About? A Case Study on Recovering Feature Facets. *J. Systems and Software (JSS)*, 152, 239–253. doi:10.1016/J.JSS.2019.01.057.

[25] Lightning Network. 2025. Basis of Lightning Technology (BOLT). Retrieved Jan. 16, 2025 from https://github.com/lightning/bolts.

[26] Lightning Network. 2025. Bitcoin Lightning Improvement Proposals (bLIPs). Retrieved Jan. 16, 2025 from https://github.com/lightning/blips.

[27] Lukas Linsbauer, Roberto Erick Lopez-Herrejon, and Alexander Egyed. 2017. Variability Extraction and Modeling for Product Variants. *Software and System Modeling (SoSyM)*, 16, 4, 1179–1199. doi:10.1007/S10270-015-0512-Y.

[28] Rezvan Mahdavi-Hezaveh, Jacob Dremann, and Laurie A. Williams. 2021. Software development with feature toggles: practices used by practitioners. *Empirical Software Engineering (EMSE)*, 26, 1, 1. doi:10.1007/S10664-020-09901-Z.

[29] Jabier Martinez, Tewfik Ziadi, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. 2015. Bottom-Up Adoption of Software Product Lines: A Generic and Extensible Approach. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)*. ACM, 101–110. doi:10.1145/2791060.2791086.

[30] Varvana Myllärniemi, Juha Savolainen, and Tomi Männistö. 2013. Performance variability in software product lines: a case study in the telecommunication domain. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)*. ACM, 32–41. doi:10.1145/2491627.2491631.

[31] Satoshi Nakamoto. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf.

[32] nostr-protocol. 2025. Nostr – Notes and Other Stuff Transmitted by Relays. Retrieved Mar. 9, 2025 from https://github.com/nostr-protocol/nostr.

[33] nostr-protocol. 2025. Nostr Implementation Possibilities (NIPs). Retrieved Mar. 9, 2025 from https://github.com/nostr-protocol/nips.

[34] David Lorge Parnas. 1976. On the design and development of program families. *IEEE Trans. on Software Engineering (TSE)*, 1, 1–9.

[35] Klaus Pohl, Günter Böckle, and Frank J. van der Linden. 2005. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer. doi:10.1007/3-540-28901-1.

[36] Joseph Poon and Thaddeus Dryja. 2016. *The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments*. ISBN: 979-8-865-99550-0.

[37] Cosmin-Ioan Rosu and Mihai Togan. 2023. A Modern Paradigm for Effective Software Development: Feature Toggle Systems. In *Proc. Int'l Conf. on Electronics, Computers and Artificial Intelligence (ECAI)*. IEEE, 1–6. doi:10.1109/ECAI58194.2023.10193936.

[38] Julia Rubin, Krzysztof Czarnecki, and Marsha Chechik. 2013. Managing Cloned Variants: A Framework and Experience. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)*, 101–110. doi:10.1145/2491627.2491644.

[39] Ismayle S. Santos, Lincoln S. Rocha, Pedro de A. Santos Neto, and Rossana M. C. Andrade. 2016. Model Verification of Dynamic Software Product Lines. In *Proc. Brazilian Symposium on Software Engineering (SBES)*. ACM, 113–122. doi:10.1145/2973839.2973852.

[40] Ina Schaefer, Christoph Seidl, Loek Cleophas, and Bruce W. Watson. 2016. Tax-PLEASE - Towards Taxonomy-Based Software Product Line Engineering. In *Proc. Int'l Conf. on Software Reuse (ICSR)*. Springer, 63–70. doi:10.1007/978-3-319-35122-3_5.

[41] Felix Schwägerl and Bernhard Westfechtel. 2016. SuperMod: Tool Support for Collaborative Filtered Model-Driven Software Product Line Engineering. In *Proc. Int'l Conf. on Automated Software Engineering (ASE)*. ACM, 822–827. doi:10.1145/2970276.2970288.

[42] Christoph Seidl, Ina Schaefer, and Uwe Aßmann. 2014. Capturing Variability in Space and Time with Hyper Feature Models. In *Proc. Int'l Workshop on Variability Modelling of Software-Intensive Systems (VaMoS)*. ACM, 6:1–6:8. doi:10.1145/2556624.2556625.

[43] Steven She, Rafael Lotufo, Thorsten Berger, Andrzej Wasowski, and Krzysztof Czarnecki. 2010. The Variability Model of The Linux Kernel. In *Proc. Int'l Workshop on Variability Modelling of Software-Intensive Systems (VaMoS)*. Vol. 37. Universität Duisburg-Essen, 45–51.

[44] IPFS Standards. 2025. InterPlanetary Improvement Proposals (IPIPs). Retrieved Jan. 16, 2025 from https://specs.ipfs.tech/ipips.

[45] Mikael Svahnberg, Jilles van Gurp, and Jan Bosch. 2005. A taxonomy of variability realization techniques. *Software: Practice and Experience*, 35, 8, 705–754. doi:10.1002/SPE.652.

[46] The Bitcoin Hole. 2025. Software Wallets: Comparing Bitcoin Software Wallets feature by feature. Retrieved Nov. 14, 2025 from https://thebitcoinhole.com/software-wallets.

[47] The Tor Project. 2025. Tor design proposals. Retrieved Jan. 16, 2025 from https://spec.torproject.org/proposals/index.html.

[48] Linus Torvalds et al. 2025. Linux Kernel Source Tree. (2025). Retrieved July 12, 2025 from https://github.com/torvalds/linux.

[49] Linus Torvalds et al. 2025. The Linux Kernel Archives. (2025). Retrieved July 12, 2025 from https://www.kernel.org/linux.html.

[50] [SW] Linus Torvalds and Git Development Community, Git version 2.52.0, 2025. URL: https://git-scm.com/.

[51] Liang Wang, Zhiwen Zheng, Xiangchen Wu, Baihui Sang, Jierui Zhang, and Xianping Tao. 2023. Fork Entropy: Assessing the Diversity of Open Source Software Projects' Forks. In *Proc. Int'l Conf. on Automated Software Engineering (ASE)*, 204–216. doi:10.1109/ASE56229.2023.00168.