Beyond Classical Software Families: Community-Driven Variability

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

Alexander Boll
University of Bern
Bern, Switzerland
alexander.boll@unibe.ch

Alexander Schultheiß
University of Bern
Bern, Switzerland
alexanderschultheiss@pm.me

Timo Kehrer
University of Bern
Bern, Switzerland
timo.kehrer@unibe.ch

Abstract

Both software engineering researchers and practitioners have increasingly shifted their focus from single software systems to software families, reflecting the need for software industrialization through systematic reuse of implementation artifacts. Interestingly, several vibrant ecosystems produce software families in a radically different way than classical variability-intensive systems, notably software product lines. The Bitcoin community, for instance, evolves its ecosystem through openly shared improvement proposals being continuously shaped and autonomously implemented by independent actors. While this novel paradigm of community-driven variability (CDV) has proven effective for driving flourishing technologies like Bitcoin and others, it also comes with unique challenges calling for novel solutions. In this paper, we define the key characteristics of ecosystems exposing CDV, highlight the novel problems they face, and outline our respective research vision.

CCS Concepts

• **Software and its engineering → Software creation and management**; *Software product lines*; *Interoperability*.

Keywords

software families, software variability, improvement proposals, implementation derivatives, interoperability, evolution

ACM Reference Format:
Roman Bögli, Alexander Boll, Alexander Schultheiß, and Timo Kehrer. 2025. Beyond Classical Software Families: Community-Driven Variability. In *Proceedings of Foundations of Software Engineering (FSE '25).* ACM, New York, NY, USA, 6 pages. https://doi.org/nnnnn.nnnnn

1 Introduction

Since Parnas' seminal work on program families in the 1970s [45], both software engineering researchers and practitioners have increasingly shifted their focus from developing single software systems to managing families of software variants sharing common functionality [47]. The most systematic class of approaches for developing such variability-intensive systems is summarized under the umbrella term of software product-line engineering [19, 47], which relies on an explicit model of variability in terms of features realized based on an integrated software platform [20, 28]. Recent literature also discusses more liberal approaches to managing software families, spanning a continuum that ranges from managing ad-hoc clone-and-own [35, 50, 53, 63] and feature toggling [41, 49] in distributed open-source communities to rigorous product-line engineering using a centrally managed integrated software platform [18, 52, 54]. Albeit at varying levels of systematic organization and pre-planning, it is the fundamental principle of reusing implementation artifacts that represents a common aspect across this continuum.

```
  BIP:             <BIP number, or "?" before being assigned>
* Layer:           <Consensus (soft fork) | Consensus (hard fork) |
                    Peer Services | API/RPC | Applications>
  Title:           <BIP title; maximum 44 characters>
  Author:          <list of authors' real names and email addrs>
* Discussions-To:  <email address>
  Status:          <Draft | Active | Proposed | Deferred | Rejected |
                    Withdrawn | Final | Replaced | Obsolete>
  Type:            <Standards Track | Informational | Process>
* Requires:        <BIP number(s)>
* Replaces:        <BIP number>
* Superseded-By:   <BIP number>
```

Figure 1: Excerpt of BIP preamble structure from BIP2 [29].

Interestingly, several vibrant ecosystems produce software families in a radically different way than classical variability-intensive systems. They are driven by factors other than software industrialization and mass customization, and exhibit variability that is not focused on reusing implementation artifacts. Instead, they focus on achieving interoperability within the software family through the ecosystem community's continuous effort to shape an open set of specification documents, referred to as *improvement proposals (IPs)*. Based on this set of IPs, developer groups within the community independently derive their own variants by selecting and implementing a desired subset of the specifications. This independent derivation fosters a broad range of software variability across multiple dimensions.

As an example for such an ecosystem, consider Bitcoin [43] with its various application types (e.g., core protocol, nodes, wallet applications, block explorers, side-chains) and actors (e.g., developers, users, analysts). The concepts that define Bitcoin, along with any potential features introduced to the ecosystem, are shaped by *Bitcoin Improvement Proposals (BIPs)*, a decentralized collection of open-source specification documents written by independent actors sharing mutual interests [2]. The BIP process itself is also defined in this manner, namely within BIP2 [29], an excerpt of which is shown in Fig. 1. Developers independently choose and implement subsets of BIPs in their applications, yielding a constantly growing set of software variants to which we refer as *implementation*

FSE '25, Trondheim, Norway

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/nnnnn.nnnnn

derivatives. Conceptually, the commonalities and differences among these derivatives can be partially described in terms of BIPs, but there is typically no reuse of development artifacts at the implementation level. Nevertheless, we see the ecosystem evolving with incredible dynamism, exposing multidimensional variability to which we refer as *community-driven variability (CDV)*.

However, Bitcoin is not the only example. Next to Bitcoin, this novel form of CDV is also observed in other ecosystems, each using a slightly different interpretation of improvement proposals (e.g., Ethereum Improvement Proposals (EIPs) [8], Bitcoin Lightning Improvement Proposals (bLIPs) [3], InterPlanetary Improvement Proposals (IPIPs) [9], The Onion Router Design Proposals (TORDPs) [13], or Nostr Implementation Possibilities (NIPs) [12]).

The paradigm of continuously shaping a de-facto standard and its implementation derivatives has proven to be an effective method for evolving open-source technologies with significant dynamism and traction. However, these ecosystems not only encounter challenges similar to those of classical variability-intensive systems, but also entirely new ones. Without an explicit variability model, managing the consistent evolution of improvement proposals becomes increasingly challenging and error-prone. Bitcoin's BIP2, for example, has recently (Sep. 18, 2024) received a revision request [32] motivated by several *"pain points"*. This indicates the need to improve the governance and management of the decentralized proposal process, addressing growing challenges wrt. maintaining overview, transparency, and consensus within the current proposal framework. Furthermore, derivatives may expose impaired derivative interoperability, which is usually not the case for classical software families where variants are meant to be standalone software products. For example, a Reddit post [17] raises awareness for incompatibility issues induced by *BIP32 HD Wallets*. Follow-up discussions on Bitcoin Stack Exchange [11] and a dedicated website on wallet recovery [14] underpin the severity of the problem.

While classical domains reporting successful SPL adoption and emerging technologies following the paradigm of CDV appear miles apart, we recognize the value in exploring this novel paradigm and the possibility of adapting concepts from one paradigm to the other. Since the use of features as a central domain abstraction in SPLs aligns well with IPs in CDV, adapting the idea of feature-oriented modeling and analysis seems promising for tackling CDV-induced problems, without necessitating the adoption of product-line development processes. Conversely, research on classical variability-intensive systems will gain new momentum through the unique problems posed by CDV, leading to advancements that will push the state-of-the-art and generate new insights that may ultimately influence other domains.

In this paper, we outline our research vision on entering the novel field of CDV, summarizing our contributions as follows:

- We introduce the concept of CDV and conduct an analysis of this emerging paradigm, presenting a set of defining characteristics (Sect. 2).
- Using this analysis, we examine key problems faced in ecosystems that exhibit CDV (Sect. 3).
- We derive concrete research goals aimed at addressing the identified problems and outline the next steps towards achieving these goals (Sect. 4).

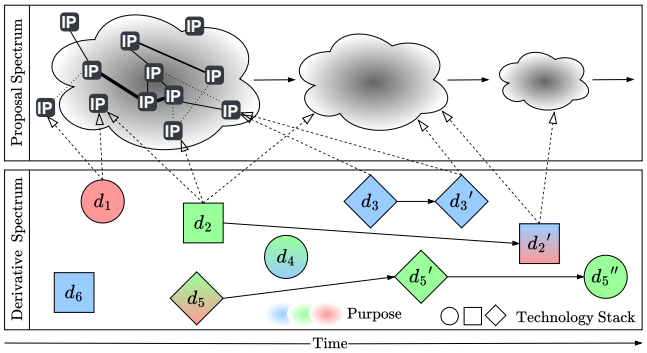

Figure 2: A schematic overview of the CDV landscape.

2 Characterization of a Novel Paradigm

To better understand the dynamics of CDV ecosystems, we thoroughly analyzed the Bitcoin ecosystem as a prominent representative. Our analysis is based on online resources, supplemented by interviews with a Bitcoin derivative developer and an advanced end user. We illustrate a summary of our results in Fig. 2. The proposal spectrum comprising the ecosystem's improvement proposals (IPs) is illustrated on top. The lower part illustrates the derivative spectrum comprising the ecosystem's applications, indicated as implementation derivatives $d_1 - d_6$ implementing varying sets of IPs. Both the proposal spectrum and the derivative spectrum evolve continuously, indicated by time progressing from left to right.

IPs are open-source specification documents written by independent actors sharing mutual interests. A substantial amount of IPs is closely related to the traditional notion of a feature, some IPs even become synonymous with feature names. For instance, the Bitcoin community speaks of *BIP32 HD Wallets* [64] or *BIP39 Mnemonic Seeds* [44], reflected in the user-interface terminology of wallet applications such as *Sparrow* [27].

Moreover, IPs have a dedicated status and may expose various kinds of interrelations (connection lines between IPs in Fig. 2). BIP2 (cf. Fig. 1), for instance, mentions status labels ranging from *draft* over *final* up to *replaced* or *obsolete*, and IP interrelations such as *requires*, *replaces*, or *superseded-by*. This indicates that IP statuses and interrelations are continuously reshaped, extended, overruled, or rejected. For example, BIP84 requires BIP173, while BIP173 has replaced BIP142 and itself is superseded by BIP350.

Applications constituting the derivative spectrum may be created at different points in time, each of them implementing an autonomously selected set of IPs (dashed arrows from d_n to IPs). While exposing variability in terms of conceptual features shaped by IPs, implementation derivatives can be built on various technology stacks and serve distinct or overlapping purposes. Fig. 2 illustrates this range of technology and purpose using different shapes and gradient-colored backgrounds, respectively. Some derivatives remain stable over time (d_1, d_6), while others may evolve.

From an organizational point of view, derivatives have full autonomy when composing their IP set. However, following the common goal of interoperability among derivatives, an ecosystem's community typically has a shared understanding of what is considered the de-facto standard at a specific point in time. We illustrate this evolving de-facto standard through a forward-moving IP cloud that

Characteristics Encouraging CDV

C1 – Crowdsourcing: There exists an open de-facto standard in the ecosystem that is continuously shaped by independent actors with distributed authority.

C2 – Improvement Proposals: This de-facto standard defines how the system shall operate using a set of improvement proposals (IPs) that can have dependencies, varying levels of importance, and undergo different states.

C3 – Independent Derivatives: Developers choose a set of IPs from which they implement independent derivatives using different technology stacks and targeting different use-cases.

C4 – Interoperability: The ecosystem's value and flourishing substantially depends on and encourages direct or indirect derivative interaction.

C5 – Decoupled Evolution: The de-facto standard, its feature specification, and the derivatives evolve autonomously and detached from each other while following their own life cycles.

Figure 3: Characteristics Encouraging CDV.

may morph into different shapes and sizes over time. Core IPs serving as active building blocks for other dependent IPs are likely to be implemented more frequently by derivatives than others, and thus contribute to the perception of a de-facto standard (central part of an IP cloud in Fig. 2). Outside this de-facto standard, there may be other IPs or informal proposals which are not (yet) officially approved but generally accepted by the community (outer part of an IP cloud). For example, other renowned sources augment the primary catalog of BIPs [2] such as, e.g., the SatoshiLabs Improvement Proposals (SLIPs) [51]. Likewise, IPs that are not yet finalized can still become de-facto standards if widely adopted. For example, BIP39, though officially holding "proposed" status until the end of 2024, has long been a standard feature among derivatives. Conversely, derivatives may counter established IPs using their own alternatives motivated by their own beliefs or technological goals. The derivative *Electrum* [6], for example, argues shortcomings in BIP39 and thus advocates its own alternative [7].

Based on our analysis, we define the constituting characteristics of ecosystems exhibiting CDV and present them in Fig. 3. To give an understanding on how these characteristics are present in different variability paradigms, we characterize a representative sample of such software ecosystems in Table 1. These include prominent CDV ecosystems and traditional variability-intensive systems, i.e., SPLs and Clone-and-Own systems frequently used as study subjects in prominent scientific publication outlet. CDV ecosystems fulfill ● most or all of the listed characteristics, though they may differ in details such as defining different sets of IP statuses, other kinds of IP interrelations, etc. In contrast, the remaining ecosystems fulfill the CDV characteristics only partially ◓ or not at all ○.

3 Emerging Problems

We identified several generalizable challenges faced by key actors in community-driven variability (CDV), including IP maintainers, derivative developers, and end users. We focus on those that transcend classical variability-intensive systems and were confirmed in our interviews with Bitcoin experts. Note that the listed problems are not confined to Bitcoin but also inherent to other ecosystems due to the fundamental characteristics of CDV.

P1 & P2 – Missing overview of proposal and derivative spectrum: Due to the dynamics imposed by characteristics C1-C5, communities typically lack an overview of the entire ecosystem and its evolution. Consequently, involved actors lack orientation

Table 1: CDV characteristics of selected ecosystems/projects.

Paradigm	Ecosystem/Project	C1	C2	C3	C4	C5
CDV	Bitcoin [2, 43]; Lightning [3, 48]	●	●	●	●	●
	Nostr [12]	●	●	●	●	●
	Ethereum [8]	◓	●	●	●	●
	Tor Protocol [13, 30]; IPFS [9, 21]	◓	●	●	●	●
SPL	Linux Kernel [15, 33]	◓	◓	◓	○	○
	Eclipse [25, 60]	◓	◓	◓	◓	○
	BusyBox [46, 62]	◓	◓	◓	○	○
Clone & Own	ApoGames [36, 42]	○	○	◓	○	◓
	Marlin Forks [37, 38]	○	○	◓	○	◓
	Health Watcher [56, 57]	○	○	◓	○	○

for guiding their decisions within the ecosystem. This missing overview is felt on both levels: the proposal spectrum (P1), and the derivative spectrum (P2). Realizing the need for an overview, the Bitcoin community already created a number of websites that monitor [10], compare [1, 24], or suggest [5] derivatives. We find these handcrafted ad-hoc monitoring efforts insufficient, but they underscore the richness of existing variability and, more importantly, the need to manage it effectively.

P3 – IP change impact assessment: The actors (C1) in the ecosystem face challenges during suggesting and updating IPs (C2), such as avoiding unforeseen side effects and change impact assessment (C4). For example, although on-boarding developer guidelines exist in Bitcoin [40], resources that document the interrelations between BIPs or their perceived feature impacts are missing.

P4 – Misalignment of proposal and derivative spectrum: There is a common interest to avoid a misalignment (C5) of derivatives and the proposal spectrum. However, developers (C3) lack the necessary guidance for alignment, while end users are unable to verify it, undermining trust in derivatives (C4) and into the ecosystem. This lack of guidance is exemplified in Electrum avoiding BIP39 [7], whereas Sparrow "tries wherever possible to adhere to commonly accepted standards in order to have as wide an interoperability as possible." [27]

P5 – Determining interoperability of derivatives: The shared interest in interoperability (C4) forces developers and end users to be aware of potential restrictions of derivative interactions. A lack of interoperability can lead to immense damage, such as permanent financial losses due to wallet recovery issues [14, 26] or incorrectly mined blocks [22]. Some communities already introduced partial solutions for this problem, e.g., *feature vectors* [4], a handshake, that tests what features the other derivative implements prior to actual interaction. However, users could place more trust into a more rigorous procedure, that is formally derived from and enforced through an ecosystem's variability model.

P6 – Ecosystem fork: The independent evolution of proposals and derivatives (C5) can lead to complex phenomena: As some IPs are embraced by the whole community, others may be rejected by a tight-knit part of the community (C3). This can lead to a split within the ecosystem into fractions or a complete detachment, as sub-communities drift further and further apart. Ultimately, such detachments provoke yet another variability source for both IPs (C2) and derivatives (C3), catalyzing the severity of P1-P5. In Bitcoin and related domains, for instance, this phenomenon is referred to as *fork* and has had occurred several times in the past (e.g., Bitcoin Cash, Gold, SV) [23].

4 Research Vision

Our research vision is to develop foundations for methods supporting the continuous evolution of ecosystems exposing CDV, tackling the problems identified in Sect. 3. We focus on understanding and auditing its multidimensional dynamics, and on providing means for constructive, organizational and analytic quality assurance. We present our research goals, promising starting points for technical solutions, and envisioned research methods and study subjects.

4.1 Research Goals

RG1 – Systematic treatment of CDV in proposal spectrum: Our first research goal is threefold. First, we aim to develop a variability modeling formalism and notation that can adequately capture CDV ecosystems and their evolution, providing a structured, explorable representation of the proposal spectrum amenable to analysis (P1). Second, we want to support the automated extraction of CDV models from various resources, with a focus on deriving variability models directly from IP collections. Third, analysis techniques shall be developed to reason about the structure and constraints of CDV models, spotting anomalous IPs and interrelations. This includes methods for differential analysis of CDV models representing different proposal spectrum snapshots, facilitating change impact analyses in the proposal spectrum (P3, P6).

Impact: Holistic modeling of a CDV ecosystem's topology fostering comprehensibility and auditability.

RG2 – Supporting cohesive evolution of proposal and derivative spectrum: Given the autonomous evolution of these two spectra, our goal is to better understand and measure their cohesion (P4). This includes providing configuration support through CDV model-guided IP selection and first cohesion assessments by, e.g., checking a given set of IPs against a CDV model. However, the major endeavor pursued with this research goal is to support tracing of IPs from the proposal to the derivative spectrum, providing a better understanding of the derivative spectrum (P2) and facilitate further change impact analyses (P3). Besides IP traceability, we aim at mining CDV models from existing derivatives, enabling comparisons with those extracted from the IP spectrum (P4) and analyzing potential drift between community forks (P6).

Impact: Streamline the evolution of ecosystems by increasing the efficiency and effectiveness of future development endeavors.

RG3 – Methodical handling of derivative interoperability impairment: We dedicate our final research goal to address the challenges related to impaired interoperability within the derivative spectrum (P5), which boils down to handling and detecting undesired inter-derivative IP interactions. Anticipated interactions shall be documented and articulated through the CDV model, amenable to automatically validating derivatives wrt. proposal spectrum alignment (P4). Unanticipated interactions impairing interoperability shall be detected through systematic IP interaction testing, which must be both effective and efficient to be accepted in practice.

Impact: Reduce the effort and complexity of proper inter-derivative feature testing, further maximizing interoperability and positive user experience.

4.2 Starting Points for Technical Solutions

In general, our technical solutions for achieving our research goals RG1-RG3 shall adopt existing variability mechanisms as far as possible, yet with radically different goals and assumptions, and without the need to adopt product-line development processes which hardly apply to the dynamics of community-driven ecosystems.

Inspired by classical approaches to variability modeling and problem space analysis [19], the first essential step towards RG1 is to develop a variability modeling formalism and notation that adequately captures CDV. We foresee a basic set of required concepts provided by the Universal Variability Language (UVL) and Hyper Feature Models (HFMs) [55]. The UVL already unifies the many existing variability modeling approaches used across various domains, and it may be extended by the means for describing the life cycle of IPs and their specific kinds of interrelations. HFMs extend traditional feature models in both space and time, a promising idea which may be adapted to reflect the multidimensional nature of CDV. As for advanced analyses of the proposal spectrum evolution, the idea of semantic feature model differencing [59] may be adapted to the differential analysis of CDV model snapshots.

The major task for realizing RG2 revolves around supporting IP traceability from the proposal to the derivative spectrum. We envision retroactive IP location techniques [34], as the dynamic nature of these ecosystems often hinders proactive IP tracing. Since we cannot assume the derivatives being created through traditional clone-and-own [35, 50], we may hardly adopt set-based techniques such as *Ecco* [39] for this task. However, it might be promising to evaluate the performance of feature location techniques being capable of working with single variants only [31]. Moreover, since derivatives may integrate reference libraries such as cryptographic primitives, extracting Software Bills of Materials (SBOMs) [65] for derivatives may inform the identification of their implemented IPs.

The most challenging part of RG3 is to support the detection of unanticipated IP interactions impairing interoperability. While pushing the boundaries from intra-derivative to inter-derivative interaction testing goes beyond software quality issues addressed by software product-line testing [16], it exposes similar challenges. As testing all the mutual IP interactions of implementation derivatives is infeasible, we strive for novel sampling methods that enable spotting the most harmful interactions effectively. To that end, we aim to lift existing combinatorial interaction testing strategies [61] to CDV models. This allows us to explore the sample space induced by different strategies and eventually making informed decisions in balancing efficiency and effectiveness.

4.3 Research Methods and Study Subjects

Given our technically focused research goals, we adopt a design science approach, implementing conceptual solutions as research prototypes for evaluation. We primarily strive for an evaluation strategy that favors maximizing internal validity over external validity [58]. We will first focus on the Bitcoin ecosystem for three reasons: (1) its large community and high degree of CDV, (2) the abundance of high-quality, openly available data, and (3) its long history, allowing for retrospective study and simulation of its evolutionary dynamics. Then, we increase the external validity of our results by studying other ecosystems sharing similar characteristics.

In parallel, we will conduct qualitative research through surveys and interviews with actors of representative ecosystems, for further validation and potential refinement of our problem analysis. Furthermore, we will explore potential impacts of our research on ecosystems that are closely related to CDV.

5 Conclusion

Community-driven variability (CDV) represents an emerging form of distributed software variability that transcends traditional centralized variability-intensive systems and is unexplored in current literature. This vibrant field offers a number of relevant challenges, which become more demanding as the communities grow and the ecosystems evolve. In our research vision presented in this paper, we outline how leveraging feature-oriented modeling and analysis concepts may be a promising starting point for effectively addressing CDV challenges without introducing product-line processes, fostering synergies and mutual impact.

References

[1] Bitcoin Wiki [2025-01-16]. *Bech32 Adoption.* Bitcoin Wiki. https://en.bitcoin.it/wiki/Bech32_adoption

[2] [2025-01-16]. *Bitcoin Improvement Proposals (BIPs).* https://github.com/bitcoin/bips

[3] [2025-01-16]. *Bitcoin Lightning Improvement Proposals (bLIPs).* https://github.com/lightning/blips

[4] [2025-01-16]. *BOLT9: Assigned Feature Flags.* https://github.com/lightning/bolts/blob/master/09-features.md

[5] Bitcoin.org [2025-01-16]. *Choose Your Wallet.* Bitcoin.org. https://bitcoin.org/en/choose-your-wallet

[6] [2025-01-16]. *Electrum Bitcoin Wallet.* https://electrum.org

[7] [2025-01-16]. *Electrum Seed Version System.* https://electrum.readthedocs.io/en/latest/seedphrase.html

[8] [2025-01-16]. *Ethereum Improvement Proposals (EIPs).* https://eips.ethereum.org

[9] [2025-01-16]. *InterPlanetary Improvement Proposals (IPIPs).* https://specs.ipfs.tech/ipips

[10] WalletScrutiny [2025-01-16]. *Know Your Wallet like You Built It.* WalletScrutiny. https://walletscrutiny.com

[11] [2025-01-16]. *Newest 'bip32-Hd-Wallets' Questions.* https://bitcoin.stackexchange.com/questions/tagged/bip32-hd-wallets

[12] [2025-01-16]. *Nostr Implementation Possibilities (NIPs).* https://github.com/nostr-protocol/nips

[13] [2025-01-16]. *Tor Design Proposals.* https://spec.torproject.org/proposals/index.html

[14] walletsrecovery.org [2025-01-16]. *Wallets Recovery [Beta].* walletsrecovery.org. https://walletsrecovery.org

[15] Iago Abal, Jean Melo, Stefan Stănciulescu, Claus Brabrand, Márcio Ribeiro, and Andrzej Wąsowski. 2018. Variability Bugs in Highly Configurable Systems: A Qualitative Analysis. *TOSEM* 26, 3, Article 10 (2018), 10:1–10:34 pages.

[16] Halimeh Agh, Aidin Azamnouri, and Stefan Wagner. 2024. Software product line testing: a systematic literature review. *Empirical Software Engineering* 29, 6 (2024), 146.

[17] Amichateur. [2025-01-16]. *[Awareness/Proposal] The Multitude of Different "Derivation Paths" in BIP32 Bitcoin Wallets Causes Incompatibility All around When It Comes to Wallet Seed Restore Operation for Non-Tech-Savy Users.* r/Bitcoin. https://www.reddit.com/r/Bitcoin/comments/qeu3j7/awarenessproposal_the_multitude_of_different/

[18] 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 *ICSE* (Hyderabad, India). ACM, New York, NY, USA, 532–535.

[19] Sven Apel, Don S. Batory, Christian Kästner, and Gunter Saake. 2013. *Feature-Oriented Software Product Lines - Concepts and Implementation.* Springer.

[20] Don S. Batory. 2005. Feature Models, Grammars, and Propositional Formulas. In *SPLC (LNCS, Vol. 3714).* Springer, 7–20.

[21] Juan Benet. 2014. IPFS - Content Addressed, Versioned, P2P File System. arXiv:1407.3561 [cs]

[22] Jonathan Bertrand. [2025-01-16]. *The Hidden World of Bitcoin Invalid Blocks: Insights and Implications.* D-Central. https://d-central.tech/the-hidden-world-of-bitcoin-invalid-blocks-insights-and-implications

[23] Jonathan Bier. 2021. *The Blocksize War: The Battle over Who Controls Bitcoin's Protocol Rules.* Independently published.

[24] Bitcoin Optech. [2025-01-16]. *Compatibility Matrix.* https://bitcoinops.org/en/compatibility

[25] Humberto Cervantes and Sonia Charleston-Villalobos. 2006. Using a Lightweight Workflow Engine in a Plugin-Based Product Line Architecture. In *CBSE (LNCS, Vol. 4063).* Springer, 198–205.

[26] Wai Kok Chan, Ji-Jian Chin, and Vik Tor Goh. 2020. Evolution of Bitcoin Addresses from Security Perspectives. In *ICITST*. 1–6.

[27] craigraw. [2025-01-16]. *Sparrow Bitcoin Wallet.* https://sparrowwallet.com/features

[28] Krzysztof Czarnecki and Ulrich W. Eisenecker. 2000. *Generative programming - methods, tools and applications.* Addison-Wesley.

[29] Luke Dashjr. [2025-01-16]. *BIP2: BIP Process, Revised.* https://bips.dev/2

[30] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. 2004. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*, Matt Blaze (Ed.). USENIX, 303–320.

[31] Bogdan Dit, Meghan Revelle, Malcom Gethers, and Denys Poshyvanyk. 2013. Feature location in source code: a taxonomy and survey. *JSEP* 25, 1 (2013), 53–95.

[32] Mark Erhardt. 2024. Re: [Bitcoindev] Time for an Update to BIP2? https://mailing-list.bitcoindevs.xyz/bitcoindev/82a37738-a17b-4a8c-9651-9e241118a363@murch.one Accessed: 2025-01-16.

[33] Patrick Franz, Thorsten Berger, Ibrahim Fayaz, Sarah Nadi, and Evgeny Groshev. 2021. ConfigFix: Interactive Configuration Conflict Resolution for the Linux Kernel. In *ICSE-SEIP*. IEEE, 91–100.

[34] Sandra Greiner, Alexander Schultheiß, Paul Maximilian Bittner, Thomas Thüm, and Timo Kehrer. 2024. Give an Inch and Take a Mile? Effects of Adding Reliable Knowledge to Heuristic Feature Tracing. In *SPLC*. 84–95.

[35] Timo Kehrer, Thomas Thüm, Alexander Schultheiß, and Paul Maximilian Bittner. 2021. Bridging the Gap Between Clone-and-Own and Software Product Lines. In *ICSE*. IEEE, Piscataway, NJ, USA, 21–25.

[36] Jacob Krüger, Wolfram Fenske, Thomas Thüm, Dirk Aporius, Gunter Saake, and Thomas Leich. 2018. Apo-Games: A Case Study for Reverse Engineering Variability From Cloned Java Variants. In *SPLC*. ACM, 251–256.

[37] Jacob Krüger, Wanzi Gu, Hui Shen, Mukelabai Mukelabai, Regina Hebig, and Thorsten Berger. 2018. Towards a Better Understanding of Software Features and Their Characteristics: A Case Study of Marlin. In *VaMoS*. ACM, 105–112.

[38] Max Lillack, Stefan Stanciulescu, Wilhelm Hedman, Thorsten Berger, and Andrzej Wasowski. 2019. Intention-Based Integration of Software Variants. In *ICSE*. IEEE / ACM, 831–842.

[39] Lukas Linsbauer, Felix Schwägerl, Thorsten Berger, and Paul Grünbacher. 2021. Concepts of variation control systems. *JSS* 171 (2021), 110796.

[40] Jameson Lopp. [2025-01-16]. *Bitcoin Technical Resources.* https://www.lopp.net/bitcoin-information/technical-resources.html

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

[42] Luciano Marchezan, Wesley K. G. Assunção, Gabriela Karoline Michelon, Edvin Herac, and Alexander Egyed. 2022. Code Smell Analysis in Cloned Java Variants: The Apo-Games Case Study. In *SPLC*. ACM, 250–254.

[43] Satoshi Nakamoto. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System.

[44] Marek Palatinus, Pavol Rusnak, Voisine Aaron, and Sean Bowe. [2025-01-16]. *BIP39: Mnemonic Code for Generating Deterministic Keys.* https://bips.dev/39

[45] David Lorge Parnas. 1976. On the design and development of program families. *TSE* 1 (1976), 1–9.

[46] Tobias Pett, Sebastian Krieter, Tobias Runge, Thomas Thüm, Malte Lochau, and Ina Schaefer. 2021. Stability of Product-Line Sampling in Continuous Integration. In *VaMoS*. ACM, Article 18, 9 pages.

[47] Klaus Pohl, Günter Böckle, and Frank J. van der Linden. 2005. *Software Product Line Engineering: Foundations, Principles and Techniques.* Springer.

[48] Joseph Poon and Thaddeus Dryja. 2016. The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments.

[49] Cosmin-Ioan Rosu and Mihai Togan. 2023. A Modern Paradigm for Effective Software Development: Feature Toggle Systems. In *15th International Conference on Electronics, Computers and Artificial Intelligence, ECAI 2023, Bucharest, Romania, June 29-30, 2023*. IEEE, 1–6. doi:10.1109/ECAI58194.2023.10193936

[50] Julia Rubin, Krzysztof Czarnecki, and Marsha Chechik. 2013. Managing Cloned Variants: A Framework and Experience. In *SPLC* (Tokyo, Japan). ACM, New York, NY, USA, 101–110.

[51] SatoshiLabs. [2025-01-16]. *SatoshiLabs Improvement Proposals (SLIPs).* https://github.com/satoshilabs/slips

[52] Ina Schaefer, Christoph Seidl, Loek Cleophas, and Bruce W. Watson. 2016. Tax-PLEASE - Towards Taxonomy-Based Software Product Line Engineering. In *ICSR* (Limassol, Cyprus). Springer, Berlin, Heidelberg, 63–70.

[53] Thomas Schmorleiz and Ralf Lämmel. 2014. Similarity Management via History Annotation. In *SATToSE*. Dipartimento di Informatica Università degli Studi dell'Aquila, L'Aquila, Italy, 45–48.

[54] Felix Schwägerl and Bernhard Westfechtel. 2016. SuperMod: Tool Support for Collaborative Filtered Model-Driven Software Product Line Engineering. In *ASE* (Singapore, Singapore). ACM, New York, NY, USA, 822–827.
[55] Christoph Seidl, Ina Schaefer, and Uwe Aßmann. 2014. Capturing variability in space and time with hyper feature models. In *VaMoS*. 1–8.
[56] Anas Shatnawi, Abdelhak Seriai, and Houari A. Sahraoui. 2015. Recovering Architectural Variability of a Family of Product Variants. In *ICSR (LNCS, Vol. 8919)*. Springer, 17–33.
[57] Anas Shatnawi, Abdelhak Seriai, Houari A. Sahraoui, Tewfik Ziadi, and Abderrahmane Seriai. 2020. ReSIde: Reusable Service Identification from Software Families. *JSS* 170 (2020), 110748.
[58] Janet Siegmund, Norbert Siegmund, and Sven Apel. 2015. Views on internal and external validity in empirical software engineering. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 1. IEEE, 9–19.
[59] Thomas Thüm, Don Batory, and Christian Kastner. 2009. Reasoning about edits to feature models. In *ICSE*. IEEE, 254–264.
[60] Jilles van Gurp, Christian Prehofer, and Jan Bosch. 2010. Comparing Practices for Reuse in Integration-Oriented Software Product Lines and Large Open Source Software Projects. *SPE* 40, 4 (2010), 285–312.
[61] Mahsa Varshosaz, Mustafa Al-Hajjaji, Thomas Thüm, Tobias Runge, Mohammad Reza Mousavi, and Ina Schaefer. 2018. A classification of product sampling for software product lines. In *SPLC*. 1–13.
[62] Alan Wang, Nick Feng, and Marsha Chechik. 2023. Code-Level Functional Equivalence Checking of Annotative Software Product Lines. In *SPLC*. ACM, 64–75.
[63] 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 *38th IEEE/ACM International Conference on Automated Software Engineering, ASE 2023, Luxembourg, September 11-15, 2023*. IEEE, 204–216. doi:10.1109/ASE56229.2023.00168
[64] Pieter Wuille. [2025-01-16]. *BIP32: Hierarchical Deterministic Wallets*. https://bips.dev/32
[65] Boming Xia, Tingting Bi, Zhenchang Xing, Qinghua Lu, and Liming Zhu. 2023. An Empirical Study on Software Bill of Materials: Where We Stand and the Road Ahead. In *ICSE*. 2630–2642.