Towards Systematic Treatment of Community-Driven Variability

Roman Bögli, University of Bern, Switzerland

Introduction

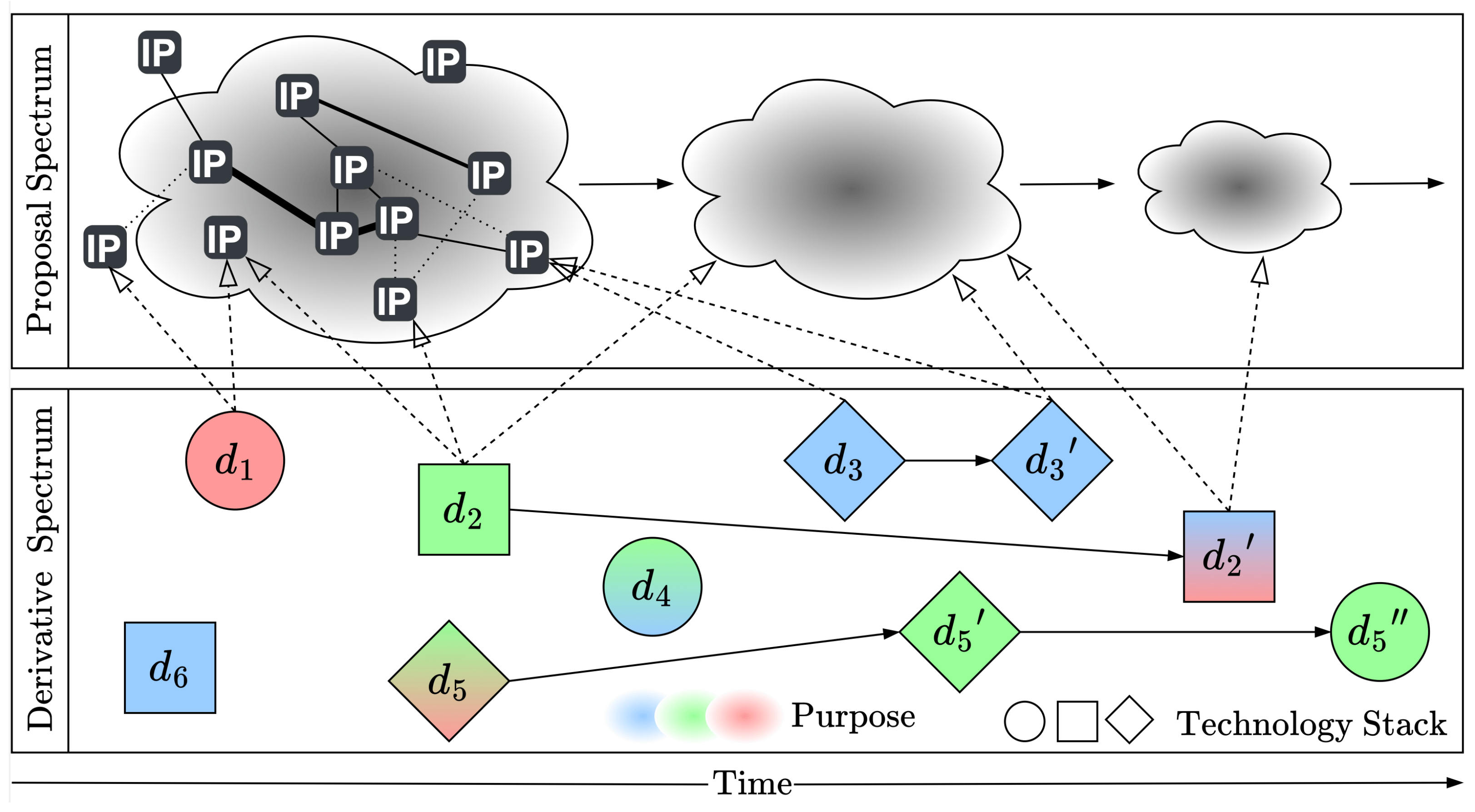
Modern decentralized software ecosystems such as **Bitcoin** [9], **Nostr** [33], **TOR** [47], or even **Python** [19] evolve through crowdsourced **improvement proposals (IPs)** that are continuously shaped and autonomously implemented by independent actors. As a result, these software ecosystems exhibit **Community-Driven Variability (CDV)** [12, 13], a novel paradigm that extends beyond traditional variability-intensive systems such as **software product lines (SPL)** [5, 35] and **clone-and-own** [22] approaches. With this thesis, we approach unsolved CDV-induced challenges and new opportunities and step towards a holistic, systematic treatment of this domain.

Community-Driven Variability (CDV)

Without a central authority, the evolution of CDV-exhibiting software ecosystems is driven through collectively curated IPs that serve as open specification documents for independently developed software applications.

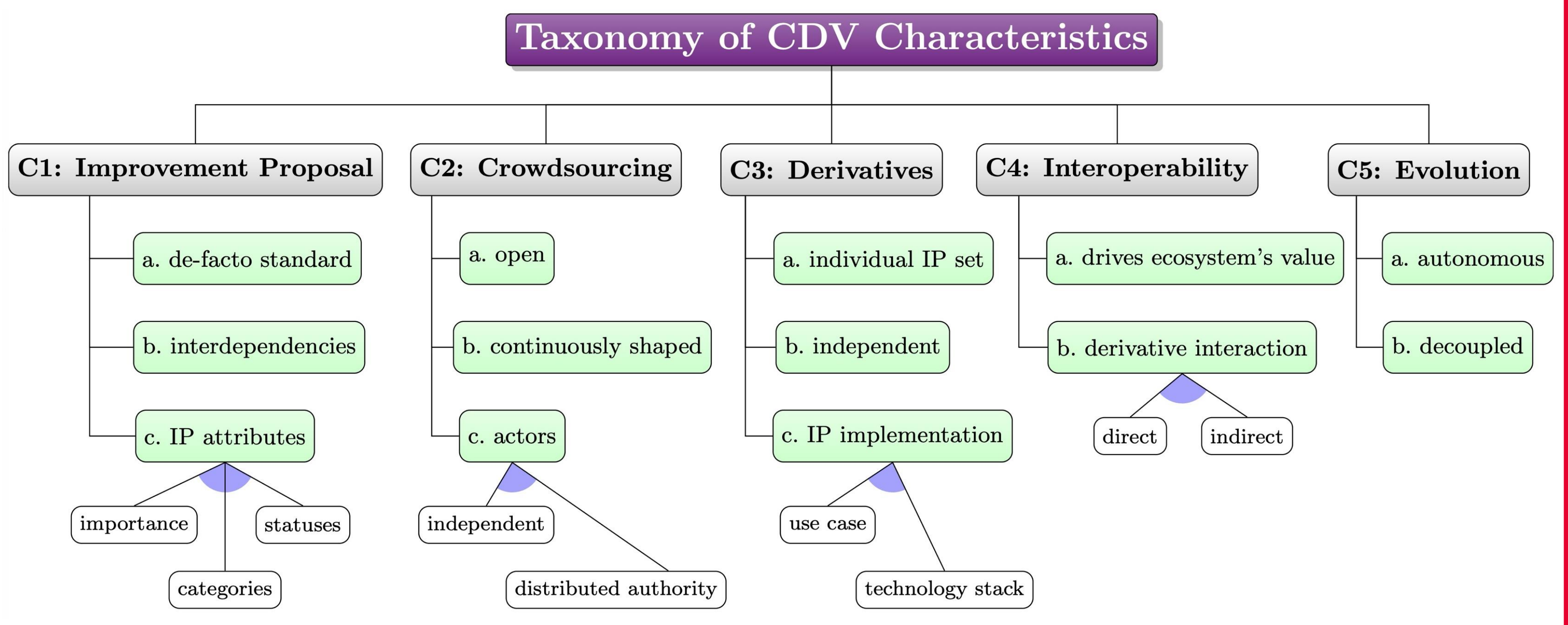
Proposal Spectrum: Represents the collection of IPs with different types and statuses, exposing various kinds of interrelations (IP connection lines). Core IPs serving as building blocks for other IPs are implemented more frequently by derivatives than others and thus contribute to the perception of a de-facto standard (central part of IP cloud).

Derivative Spectrum: Subsumes all concrete software derivatives ($d_1 – d_6$) that emerge at different points in time, each implementing an autonomously selected IP set (dashed arrows from d_n to IPs) while varying in technology stack and purpose (shapes and gradients).



Defining Characteristics

We defined five constituting characteristics that CDV-exhibiting ecosystems share, partitioned into 13 sub-characteristics (green boxes), which may be further refined by supplementary attributes (bottom uncolored boxes) [13].



Despite CDV-presence unfolding on a continuum, our derived taxonomy laid the groundwork to identify ecosystems that fall into this CDV paradigm [13].

Paradigm	Ecosystem/Project	C1			C2			C3			C4		C5	
		a	b	c	a	b	c	a	b	c	a	b	a	b
CDV	Bitcoin	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Nostr	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Tor Protocol	✓	✓	✓	✓	✓	✓	✓	-	✓	✓	✓	-	✓
SPL	Linux Kernel	✓	✓	-	✓	✓	✓	-	-	-	-	-	-	-
	Eclipse	-	✓	✓	✓	✓	✓	✓	✓	✓	-	-	-	-
CaO	Apo-Games	-	-	-	-	-	-	✓	✓	✓	-	-	-	✓
	Marlin Forks	-	-	-	✓	✓	✓	✓	✓	-	-	-	✓	✓
Other	Home Assistant	-	-	-	✓	✓	-	-	✓	✓	-	✓	-	✓
	Python	✓	✓	✓	✓	✓	-	✓	✓	✓	-	-	-	✓

Emerging Problems

P1 & P2 – Missing Overview in Proposal & Derivative Spectrum: missing overview within these spectra, manifesting in a lack of general orientation.

P3 – Change Impact Assessment: Challenges actors face when proposing or updating IPs, anticipating change impact and potential side effects.

P4 – Misalignment of Proposal & Derivative Spectrum: Addresses the detection and measurement of misalignment between the two spectra, i.e., declared versus actual IP relevance and adoption.

P5 – Derivative Interoperability Assessment: Lack of rigorous methods to reveal and test (un)desired feature interactions among derivatives to evaluate general derivative interoperability.

Research Objectives

While our work initially focuses on the Bitcoin ecosystem, given its visibility and motivating role in defining CDV [12, 13], we plan to generalize our contributions across the wider continuum of CDV-exhibiting ecosystems.

I. Variability Modeling Formalism

Objective: Devise a variability modeling formalism that adequately captures how IPs accumulate and interrelate. Further, it should support the superimposition of all variability dimensions [7] within the proposal spectrum to serve as a basis for automated spatio-temporal analysis.

Expected Contribution: A variability modeling formalism and notation that can adequately capture the versatility of an ecosystem's proposal spectrum and its evolution. We build on existing modeling frameworks such as Hyper Feature Models [42] and Dynamic Feature Transition Systems [39], proposing necessary unifications and extensions.

II. Automated Model Mining & Investigation

Objective: Employ a data-driven strategy by mining CDV ecosystem models directly from their respective IP catalogs. A pipeline should periodically harvest these distributed documents that define system behavior using unstructured text to instantiate a holistic view on the ecosystem's proposal spectrum.

Expected Contribution: A mining and analysis toolchain to systematically explore the proposal spectrum (cf. P1) and detect anomalous IPs as well as IP interrelations (cf. P3).

III. Bridging to Implementation Derivatives

Objective: Extend to derivative spectrum by integrating CDV models with concrete implementation variants to pave the way towards automatic validation of derivatives against the proposal spectrum.

Expected Contribution: Systematic integration of derivatives within existing CDV models to allow contextualized derivative overview (P2) and detect misalignments between claimed and implemented IPs (P4). It further establishes the foundation for inter-derivative testing to reveal IP interoperability impairments (P5) and to estimate the potential impact on the derivative spectrum by modification in the proposal spectrum (P3).

Bibliography

[5] S. Apel, et al. 2013. *Feature-Oriented Software Product Lines - Concepts and Implementation*. doi:10.1007/978-3-642-37521-7.

[7] T. Berger, et al. 2019. *Software Evolution in Time and Space: Unifying Version and Variability Management*. Dagstuhl Reports, 9, 5, 1–30. doi:10.4230/DAGREP.9.5.1.

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

[12] **R. Bögli**, et al. 2025. *Beyond Software Families: Community-Driven Variability*. FSE Companion. ACM, 571–575. doi:10.1145/3696630.3728501.

[13] **R. Bögli**, et al. 2026. *Community-Driven Variability: Characterizing a new Software Variability Paradigm*. Automated Software Engineering. (forthcoming)

[19] Python Software Foundation. 2025. *Python Enhancement Proposals (PEPs)*. peps.python.org.

[22] T. Kehrer, et al. 2021. *Bridging the Gap Between Clone-and-Own and Software Product Lines*. ICSE. IEEE, 21–25. doi:10.1109/ICSE-NIER52604.2021.00013.

[33] nostr-protocol. 2025. *Nostr Implementation Possibilities (NIPs)*. github.com/nostr-protocol/nips.

[35] K. Pohl, et al. 2005. *Software Product Line Engineering: Foundations, Principles and Techniques*. doi:10.1007/3-540-28901-1.

[39] I. S. Santos, et al. 2016. *Model Verification of Dynamic Software Product Lines*. SBES. ACM, 113–122. doi:10.1145/2973839.2973852.

[42] C. Seidl, et al. 2014. *Capturing Variability in Space and Time with Hyper Feature Models*. VaMoS. ACM, 6:1–6:8. doi:10.1145/2556624.2556625.

[47] The Tor Project. 2025. *Tor design proposals*. spec.torproject.org/proposals/index.html.

Roman Bögli started his PhD in April 2024 in the Software Engineering Group (SEG) led by Prof. **Timo Kehrer** at the University of Bern in Switzerland.

Poster presented at **ICSE'26** in Rio de Janeiro, April 2026, as accompanying material to the **Doctoral Symposium** paper of the same title (doi:10.1145/3774748.3787644).

roman.boegli@unibe.ch
romanboegli.ch seg.inf.unibe.ch

u^b

UNIVERSITY OF BERN