Towards a Model-Driven Approach for Hyperledger Fabric Network Orchestration

Mailson Teles-Borges[1], **Rafael Z. Frantz**[1], **Fabricia Roos-Frantz**[1]

[1]Unijuí University – Ijuí, RS – Brazil

mailson.borges@sou.unijui.edu.br

{rzfrantz, frfrantz}@unijui.edu.br

Abstract. *This paper presents a model-driven approach for specifying, validating, and deploying Hyperledger Fabric networks. We introduce a domain-specific metamodel that formalises network components–including organisations, channels, and ordering services–integrated with a tool that validates the configuration correctness and automates the production of cryptographic material and deployment files. Experimental evaluation using multi-organisation and multi-channel scenarios demonstrates support for heterogeneous taxonomies, version compatibility checking, and early detection of configuration errors.*

1. Introduction

The concept of blockchain emerged in 1991, when Stuart Haber and W. Scott Stornetta proposed a cryptographically secure chain of blocks to timestamp digital documents [Haber and Stornetta 1991]. In 2008, Satoshi Nakamoto introduced a new architecture that incorporated decentralised consensus through proof-of-work, enabling trustless peer-to-peer transactions without the need for intermediaries [Nakamoto 2008]. In 2014, Vitalik Buterin presented Ethereum which introduced the usage of smart contracts. Smart contracts are self-executing programmes stored on the blockchain that automatically enforce and execute predefined agreements once specified conditions are met [Buterin 2014]. In 2018, the Hyperledger Fabric project, initially led by IBM under the Linux Foundation, emerged as a modular and enterprise-oriented blockchain framework [Androulaki et al. 2018]. Unlike public blockchains, where users are anonymous and participation is permissionless [Yang et al. 2020], Hyperledger Fabric is a permissioned blockchain platform in which all network participants are identified and every transaction is authenticated. Due to the permissioned nature, the deploying and execution of smart contracts (known as chaincode in Fabric) are traceable and governed by access control policies. Since some of the main goals of Hyperleger Fabric is to bring privacy and governance for the blockchain ecosystem, building a Hyperledger Fabric blockchain requires that organisations (e.g., private companies, public institutions, universities, amongst others) define roles and permissions of each other.

Although Hyperledger Fabric provides the necessary modularity for enterprise use cases, its deployment remains still complex. For example, it is required to generate cryptographic material, set up channel configurations and deal with the certificate lifecycle. Unlike Ethereum, where a developer can simply use a local provider like Ganache, Fabric requires the manual orchestration of Certificate Authorities, Orderers, and Peers. Current abstractions, such as the Fabric Test Network [Hyperledger Foundation 2023] and Fablo [Hyperledger Labs 2026], limit customisation in favour of ease of use, require the installation of additional dependencies, or lack semantic validation of user-provided configurations. This creates a gap for developers who need a tailored environment without

the overhead of manual binary configuration, or for researchers who require a local development environment that closely mirrors the architectural constraints of a production deployment. Therefore, there is a technological and a research gap for proposing a tool capable of orchestrating all Hyperledger Fabric components without introducing technical complexity of manual management. The lack of a unified, declarative interface means that developers and researchers must set up, synchronise binaries, and environment variables, complicating the transition from experimental prototypes to production-ready architectures, and requiring deep expertise in underlying shell scripts.

This work addresses the complexity associated with configuring and deploying Hyperledger Fabric networks by proposing a model-driven approach for specifying, validating, and deploying such networks. Our approach is centred on a metamodel that formally specifies the structural and semantic relationships amongst network components. The metamodel is implemented in a tool that eliminates the need to install Hyperledger Fabric binaries or manage shell scripts, relying instead on containerised execution. The main contributions of this paper are: (i) the introduction of a declarative taxonomy metamodel that formally represents Hyperledger Fabric network components, including organisations, peers, orderers, certificate authorities, and channel configurations; (ii) the automation of artefact generation through a cross-platform command-line interface (CLI) tool; and (iii) the enforcement of network consistency by means of systematic validation of configurations prior to artefact generation, thereby preventing deployment errors such as inconsistent organisation definitions and incompatible version specifications. The remainder of this paper is organised as follows. Section 2 discusses existing tools that introduce abstraction mechanisms for managing Hyperledger Fabric networks. Section 3 presents the proposed metamodel. Section 4 describes how metamodel specifications are transformed into deployable artefacts. Section 5 presents a set of case studies used to evaluate the correctness and applicability of the proposed approach. Finally, Section 6 concludes the paper and outlines directions for future work.

2. Related Work

In this section, we analyse tools, approaches, or frameworks that have been proposed to simplify the configuration, deployment, and management of Hyperledger Fabric networks. Fabric Test Network is an official Hyperledger package for deploying simple local networks with two or three organisations [Hyperledger Foundation 2023]. FabNet uses a web-based tool to generate the scripts required to deploy local networks [Marcelletti and Re 2020]. Zikos *et. al.* present HFabD+M, which combines network configuration via a web-based interface with deployment through shell scripts generated from user-provided input [Zikos et al. 2022]. Fablo is a shell-based command-line tool that sets up and deploys local networks using a descriptive configuration written in JSON or YAML [Hyperledger Labs 2026]. Kong *et al.* propose SBC, a framework that relies on a web application to compute network configurations, which are then used by Ansible scripts to deploy Hyperledger Fabric networks [Kong et al. 2023]. In general, these approaches primarily rely on configuration templates, shell scripting, or external orchestration tools and do not provide a formal metamodel to represent structural relationships among network components prior to deployment. Most of them are shell-based [Hyperledger Labs 2026, Marcelletti and Re 2020, Zikos et al. 2022, Hyperledger Foundation 2023] and were created to simplify the deployment rather than ensuring the correctness of the network. Table 1 summarises and compares all related work.

Table 1. Comparison of Hyperledger Fabric Network Management Tools.

Tool	Specification Mechanism	Pre-deployment Validation	Requires Fabric CLI Binaries	Containerised	Cross Platform
Fabric Test Network	Script-based	No	Yes	Yes	No
FabNet	Web Interface	No	Yes	No	No
HFabD+M	Web Interface	Limited	Yes	No	No
SBC	Web Interface	Limited	Yes	No	No
Fablo	YAML/JSON	Limited	No	Yes	No
Our Tool	Model-driven DSL	Yes	No	Yes	Yes

**Limited* denotes basic syntactic or schema-level validation without formal semantic consistency.

3. Overview of the Metamodel

This section presents the proposed metamodel which serves as the foundation for automating the Hyperledger Fabric orchestration. The metamodel defines the structural and semantic elements required to describe a blockchain network. Figure 1 illustrates its hierarchy. The `Config` entity acts as the entry point, and contains `output` and `network` attributes. `Output` represents the target directory for generated files (e.g., `configtx.yml`, Docker Compose files, and cryptographic materials). `Network` is the name of the docker network. A Hyperledger Fabric network may comprise multiple `Channels`. While channels define the logical partitioning of the network, their configuration parameters are encapsulated in `Profile` entities. In alignment with the `configtx` structure, a `Profile` defines a reusable template for channel configuration. Consequently, multiple channels can be instantiated from the same profile.

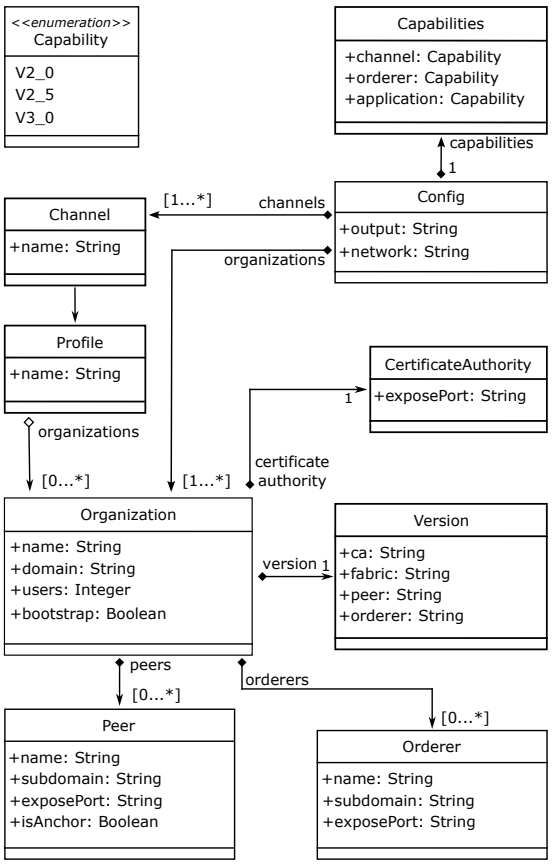

Figure 1. Metamodel.

```yaml
output: output

capabilities:
  channel: V2_0
  orderer: V2_0
  application: V2_5

organisations:
  - name: Org1
    domain: org1.com
    orderers:
      - name: Orderer
        subdomain: orderer
    peers:
      - name: Peer0
        subdomain: peer0
        isAnchor: true

  # Org2 omitted for brevity

profiles:
  - &DefaultProfile
    name: DefaultProfile
    organisations:
      - Org1
      - Org2

channels:
  - name: multichannel
    profile: *DefaultProfile
```

Listing 1. YAML encoding.

While Hyperledger Fabric allows for per-profile capability settings to support legacy nodes, our metamodel adopts a centralised strategy to simplify deployments and reduce configuration complexity. The `Capability` entity defines the versioning for the three layers of the network, which are: `Channel`, `Orderer`, and `Application`. A channel may contain many organisations that manage their own nodes. Organisations may run their nodes with versions different from each other. The capability definitions establish boundaries about functionality and version compatibility between the nodes that participate in the channel. Although default capabilities are defined globally, organisations can override binary versions for specific components such as Certificate Authorities, Peers, and Orderers. These overrides enable cross-version testing scenarios, provided that the selected versions remain compatible with the network definitions.

Each `organisation` is uniquely identified by its `name` and `domain` attributes. Organisations represent administrative entities responsible for managing network nodes and identities. The metamodel includes optional attributes such as `Users` and `Bootstrap`. The `Users` attribute defines the number of identities generated for the organisation. If not specified, only an administrative identity is created. The `Bootstrap` attribute indicates the organisation responsible for generating the genesis block during network initialisation. The metamodel enforces a constraint that allows only one organisation to be designated as the bootstrap authority. Each organisation contains an instance of `CertificateAuthority`, responsible for issuing digital certificates used for identity management. Organisations may define two types of nodes: `Orderers` and `Peers`. Both types of node require a `name` and `subdomain`. The Peer entity represents nodes that host ledgers and smart contract execution environments. Peers include an additional boolean attribute, `isAnchor`, which designates anchor peers responsible for cross-organisation gossip communication within channels. Since network nodes are deployed as containerised services, port exposure must be explicitly defined to enable external communication. The metamodel provides the `exposePort` attribute for this purpose, allowing deterministic port mapping between containerised services and the host environment.

4. DSL Parsing and Artefact Generation

To support Hyperledger Fabric artefact generation and network deployment, we developed a tool called Fabric Network Orchestrator (FNO). It is operating-system agnostic, requires no additional binary dependencies beyond its own executable, and bridges the gap between the abstraction provided by the metamodel and a fully deployable Hyperledger Fabric network. The tool is available on GitHub[1]. Figure 2 illustrates the FNO workflow: first, the metamodel is specified in one of the supported formats (currently YAML, JSON, or TOML)–see Listing 1. Once specified, and prior to artefact generation, the tool performs a comprehensive set of structural, semantic, and versioning validations to ensure that the configuration file is valid and deployable. Structural validations verify the uniqueness of organisation names and domains, the uniqueness of channel names, and that all channels reference valid profiles. Semantic constraints enforce that exactly one organisation is designated as the bootstrap organisation, that at most one peer per organisation is defined as the anchor peer, and that the network specifies at least one organisation. Versioning and compatibility checks ensure that declared capability versions are greater than or equal to `V2_0`, that all Fabric component versions satisfy the minimum supported version, and that component versions are compatible with the configured capabilities. Finally, deployment-related constraints verify that all declared ports are numeric

[1] [https://github.com/gca-research-group/fabric-network-orchestrator

values greater than zero. Once all validations are successfully completed, the tool generates the network artefacts (e.g., Docker Compose files and cryptographic materials). In the final step, the network is deployed.

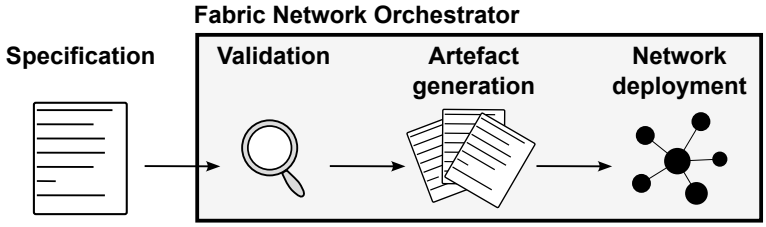

Figure 2. Deployment workflow.

5. Case Studies

We evaluated the proposed tool using eight configuration scenarios divided into valid and invalid configurations. The valid configurations are **(C1)** a minimal network with two organisations, one orderer, and one channel; **(C2)** a multi-organisation consortium with five organisations and two peers each; **(C3)** a multi-channel network with shared channel profiles; **(C4)** cross-version compatibility across Fabric components. The invalid configurations are **(C5)** a channel referencing an undefined organisation; **(C6)** organisations with incompatible binary versions; **(C7)** duplicate organisation names; and **(C8)** a channel without organisations. The corresponding configuration files are available on GitHub[2]. Each case study was evaluated by executing the `artefacts generate` command and, when applicable, deploying the network using the `network deploy` command. A case study is considered successful if (i) the configuration passes all validation checks, (ii) the tool generates a complete and consistent set of Hyperledger Fabric artefacts, and (iii) the resulting network can be deployed without manual intervention. For invalid configurations, success is defined by the detection and reporting of configuration errors prior to artefact generation. Table 2 summarises the execution results. The results indicate that the proposed tool successfully generated and deployed Hyperledger Fabric networks for all valid configurations without requiring manual adjustments. In the invalid configuration scenario, the tool correctly prevented artefact generation by detecting semantic inconsistencies, demonstrating the effectiveness of the validation mechanisms.

Table 2. Comparison of case study results.

Case Study	Artefacts Generated	Deployed	Error Message
C1	✓	✓	-
C2	✓	✓	-
C3	✓	✓	-
C4	✓	✓	-
C5	×	×	organization not defined: Org3
C6	×	×	peer version 1.4 is lower than required 2.0.0
C7	×	×	duplicate organization name: Org1
C8	×	×	profile DefaultProfile must include at least one organization

[2]https://github.com/gca-research-group/fabric-network-orchestrator/tree/main/samples

6. Conclusions and Future Work

This paper presented a model-driven approach for the specification, validation, and deployment of Hyperledger Fabric networks. By introducing a formal metamodel and an accompanying command-line tool, the approach bridges the gap between formal specification, artefact generation, and network deployment. The evaluation demonstrates that the approach supports different network taxonomies, although the tool was evaluated in a local environment with restricted scenarios. A more detailed comparison with existing tools for network configuration and deployment is still required. Future work includes supporting Kubernetes-based networks, integrating the CLI into code editor extensions such as Visual Studio Code, providing in-editor capabilities for metamodel specification, and mapping formal Object Constraint Language rules to enable formal verification.

Acknowledgments

This research is partially funded by the Co-ordination for the Brazilian Improvement of Higher Education Personnel (CAPES) and the Brazilian National Council for Scientific and Technological Development (CNPq) under the following project grants 311011/2022-5, 309425/2023-9, 402915/2023-2.

References

Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., Caro, A. D., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., Muralidharan, S., Murthy, C., Nguyen, B., Sethi, M., Singh, G., Smith, K., Sorniotti, A., Stathakopoulou, C., Vukolić, M., Wei, S., and Yellick, J. (2018). Hyperledger fabric: A distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, pages 1–15.

Buterin, V. (2014). A next-generation smart contract and decentralized application platform. https://ethereum.org/en/whitepaper/.

Haber, S. and Stornetta, W. S. (1991). How to time-stamp a digital document. *Journal of Cryptology*, 3:99–111.

Hyperledger Foundation (2023). Using the fabric test network. https://hyperledger-fabric.readthedocs.io/en/latest/test_network.html. Accessed: 2026-01-14.

Hyperledger Labs (2026). Fablo: A tool for rapid hyperledger fabric network setup. https://github.com/hyperledger-labs/fablo. Accessed: 2026-01-14.

Kong, V., Soeng, S., Lee, K.-H., and Cho, W.-S. (2023). Automated tool for building hyperledger fabric blockchain networks using ansible. *International Journal of Contents*, 19(2):15–27.

Marcelletti, A. and Re, B. (2020). Fabnet: an automatic hyperledger fabric network wizard. In *CEUR WORKSHOP PROCEEDINGS*, pages 59–67.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf.

Yang, R., Wakefield, R., Lyu, S., Jayasuriya, S., Han, F., Yi, X., Yang, X., Amarasinghe, G., and Chen, S. (2020). Public and private blockchain in construction business process and information integration. *Automation in construction*, 118:103276.

Zikos, I., Sendros, A., Drosatos, G., and Efraimidis, P. S. (2022). Hfabd+ m: A web-based platform for automated hyperledger fabric deployment and management. In *Global Emerging Technology Blockchain Forum: Blockchain & Beyond*, pages 1–6.