A Framework for Automating Deployment and Evaluation of Blockchain Networks

Nguyen Khoi Tran\textsuperscript{a}, M. Ali Babar\textsuperscript{a,b}, Andrew Walters\textsuperscript{c}

\textsuperscript{a}The University of Adelaide, South Australia, Australia
\textsuperscript{b}Cyber Security Cooperative Research Centre, Australia
\textsuperscript{c}Defence Science and Technology Group, Australia

Abstract

A blockchain network is a distributed system established by mutually distrusting participants to operate a blockchain, enabling them to manage critical information such as account balances or asset ownership without a centralised third party. Blockchain network deployment and evaluation have become prevalent due to the emerging blockchain use cases by enterprises, governments, and Internet of Things (IoT) applications, which demand private blockchains rather than participating in public ones. A blockchain network architecture drives deployment and evaluation activities. Nevertheless, practitioners must learn and perform error-prone activities to transform architecture into a blockchain network and evaluate it. Therefore, it is beneficial to automate these activities so that practitioners can focus on the architecture design, a valuable and hard-to-automate activity. The key challenges of such an automation framework are keeping up with the advances in blockchain technologies and the increasing complexity of blockchain network architecture. This paper proposes NVAL, a software framework that implements a novel architecture-driven, community-supported approach to automate blockchain network deployment and evaluation. NVAL accepts blockchain network architecture as input. It supports complex multi-channel blockchain networks, an increasingly prevalent architecture for private blockchain. The framework keeps up with blockchain technologies by leveraging platform-specific automation programs developed by a practitioner community via runtime composition to handle new networks. We evaluated NVAL with a case study and showed that the framework requires only seven automation programs to deploy 65 blockchain networks with 12 diverse architectures and generate 295 evaluation datasets. Furthermore, it consumes only 95.5 ms to plan and orchestrate the deployment and evaluation, which is minuscule compared to the total time required for deploying and benchmarking a blockchain network.

Keywords: Blockchain, Distributed Ledger, Deployment, Evaluation, Automation, Framework

1. Introduction

A blockchain can be defined as a secure database and a replicated state machine operated by a distributed network of mutually distrusting participants \cite{1, 2, 3}. It helps participants to maintain and update shared state information, such as account balances or asset ownership, without relying on a centralised third party \cite{4}. The database contains transactions that describe state updates such as transferring currency between accounts. The state machine processes transactions to calculate a new state, such as new account balances after currency transfer \cite{3}. Both blockchain’s database and state machine are replicated across participants and synchronised by a fault tolerant consensus protocol \cite{5}, ensuring that participants can access the state information and detect divergence by others. By embedding data and logic into blockchain’s transactions, state variables, and state machines, participants conduct business processes without appointing a third party, reducing the failure risks and insider threats \cite{6, 7}. Blockchains have been leveraged successfully for financial use cases such as cryptocurrency (digital cash) \cite{8}, digital assets \cite{9}, and decentralised finance platforms \cite{10}. Blockchains have also been used as a secure timestamping service to prove the existence and authenticity of documents and a means for resisting information censorship \cite{8}. Most of the described use cases utilise global public blockchains such as Ethereum and Bitcoin. However, there are emerging use cases by enterprises, governments, edge computing, and the Internet of Things (IoT) \cite{11, 12} where public blockchains fail to provide the necessary performance, scalability, and confidentiality \cite{13}. Therefore, there has been an increasing demand for permissioned or private blockchains. For instance, when conducting the first blockchain-based Letter-of-Credit (LC) transaction between Vietnam and Korea on July 3, 2019, HSBC utilised a private blockchain based on R3’s Corda platform \cite{14} rather than a public one.

A distributed system established by mutually distrusting participants to run a blockchain is known as a blockchain network \cite{15}. It is made up of distributed computers
running a blockchain client, which is a software implementation of a blockchain protocol. Public blockchains run on public blockchain networks that emerge without coordination anonymous Internet users. Private blockchains are operated by private blockchain networks, which can be designed, deployed, and evaluated to ensure they fit an intended use case and hardware infrastructure [15, 2]. Let us elaborate on the design, deployment, and evaluation of blockchain networks with the following scenario.

Motivating Scenario: Multiple research labs from different universities wish to connect their robotic testbeds to form a joint experimentation platform to expand the scope of their experiments. For instance, researchers can use the platform to validate a robot coordination mechanism across multiple environments by assigning tasks to and receiving experiment data from edge agents such as uncrewed ground vehicles (UGV), uncrewed aerial vehicles (UAV), and uncrewed underwater vehicles (UUV) from other testbeds. A joint experimentation platform raises various security requirements: edge agents must verify the authenticity of incoming mission objectives [16], researchers must verify the integrity and provenance (history) of the experiment data [17], and infrastructure owners must track researchers’ requests for auditing. The information necessary for the above requirements constitutes the platform’s state (e.g., tasking status of a robot) and the state updates (e.g., a researcher assigning a task to a robot). Participants can maintain such information securely without hiring a third party or appointing a leader to operate a centralised service by leveraging a private blockchain. Doing so requires establishing a private blockchain network. Participants can control all aspects of the network, such as where to deploy its constituting blockchain clients and how to configure them. Participants can even split the network into separate blockchain channels that store different information and run different blockchain protocols. For instance, an inter-lab channel connecting powerful workstations can employ a secure but resource-intensive blockchain protocol whilst intra-lab channels between robots and base stations within a testbed can leverage a lightweight blockchain protocol (e.g., [18]). The described multi-channel heterogeneous blockchain network architecture is increasingly prevalent in edge computing and IoT blockchain use cases that involve edge computing and IoT [11, 2]. The heterogeneity of this architecture stands in stark contrast to the lack of support for multi-channel topology and multi-technology-per-network of most current tools and services.

Existing literature reports various approaches and tools for automating blockchain network deployment and evaluation. Practitioners can utilise generic IT automation tools such as Ansible[1] and Terraform[2] to perform repetitive activities such as provisioning and starting blockchain clients across computers to reduce time and implementation errors. However, generic tools fail to reduce complexity because practitioners must manually transform blockchain network architectures into tools’ instructions. A recent study on DevOps practices in blockchain-based software engineering found little evidence of real-world utilisation of generic automation tools [20]. Blockchain-specific automation frameworks such as PlaTIBART [30], Hyperledger Composer [31] and Hyperledger Bevel [32] (formerly known as the Blockchain Automation Framework) and MixBytes Tank [33] have been proposed to address the limitation of generic automation tools. Blockchain-as-a-Service (BaaS) platforms such as uBaaS [26], NutBaaS [19], AWS Managed Blockchain, Azure BaaS, and IBM Blockchain Platform are cloud services that also provide “one-click deployment” to provision blockchain networks on cloud-based virtual machines. While these tools and services are better than generic automation tools at abstracting the complexity, we found that most require low-level and technology-specific specifications. Furthermore, practitioners must manually transform blockchain network architectures into tools’ instructions. A recent study on the state of practice of blockchain-oriented software engineering found little evidence of real-world utilisation of generic automation tools [28]. In fact, “one-click deployment” is a highly demanded tool for development and testing according to a recent survey on the state of practice of blockchain-oriented software engineering [25].

Lack of support for heterogeneous multi-channel blockchain networks. Blockchain networks can be considered concrete implementations of blockchain protocols, providing software clients and utilities for deploying or participating in a blockchain network [11]. We found that most of the current tools only support a few prominent blockchain platforms. Hyperledger Bevel, the state-of-the-art blockchain platform, is a concrete implementation of the blockchain protocol. Public blockchains run on public blockchain networks that emerge without coordination anonymous Internet users. Private blockchains are operated by private blockchain networks, which can be designed, deployed, and evaluated to ensure they fit an intended use case and hardware infrastructure [15, 2]. Let us elaborate on the design, deployment, and evaluation of blockchain networks with the following scenario.

Motivating Scenario: Multiple research labs from different universities wish to connect their robotic testbeds to form a joint experimentation platform to expand the scope of their experiments. For instance, researchers can use the platform to validate a robot coordination mechanism across multiple environments by assigning tasks to and receiving experiment data from edge agents such as uncrewed ground vehicles (UGV), uncrewed aerial vehicles (UAV), and uncrewed underwater vehicles (UUV) from other testbeds. A joint experimentation platform raises various security requirements: edge agents must verify the authenticity of incoming mission objectives [16], researchers must verify the integrity and provenance (history) of the experiment data [17], and infrastructure owners must track researchers’ requests for auditing. The information necessary for the above requirements constitutes the platform’s state (e.g., tasking status of a robot) and the state updates (e.g., a researcher assigning a task to a robot). Participants can maintain such information securely without hiring a third party or appointing a leader to operate a centralised service by leveraging a private blockchain. Doing so requires establishing a private blockchain network. Participants can control all aspects of the network, such as where to deploy its constituting blockchain clients and how to configure them. Participants can even split the network into separate blockchain channels that store different information and run different blockchain protocols. For instance, an inter-lab channel connecting powerful workstations can employ a secure but resource-intensive blockchain protocol whilst intra-lab channels between robots and base stations within a testbed can leverage a lightweight blockchain protocol (e.g., [18]). The described multi-channel heterogeneous blockchain network architecture is increasingly prevalent in edge computing and IoT blockchain use cases that involve edge computing and IoT [11, 2]. The heterogeneity of this architecture stands in stark contrast to the lack of support for multi-channel topology and multi-technology-per-network of most current tools and services.

Limited scope of the supported blockchain platforms. Blockchain platforms can be considered concrete implementations of blockchain protocols, providing software clients and utilities for deploying or participating in a blockchain network [11]. We found that most of the current tools only support a few prominent blockchain platforms. Hyperledger Bevel, the state-of-the-

---

1. https://www.ansible.com
2. https://www.terraform.io
This paper proposes NVAL (Network Deployment and Evaluation framework), a software framework that implements a novel architecture-driven, community-supported approach to automate the deployment and evaluation of blockchain networks. The framework is driven by blockchain network architecture specifications that describe high-level design decisions (e.g., network topology, protocol choice) and support heterogeneous multi-channel networks. Instead of focusing on a few blockchain platforms and relying on the open-source contribution for extension, NVAL leverages the existing platform-specific automation programs developed by a practitioner community by combining them at runtime according to the given blockchain network architecture for deployment and evaluation. This approach was motivated by many open-source automation scripts for blockchain networks. As of February 2022, we found 52 open-source GitHub repositories on automated blockchain network deployment using Ansible, 67 repositories using Terraform, and 35 repositories using other scripting languages. Most blockchain platforms also provide deployment instructions in their documentation, which can be transformed into automation scripts. By leveraging the automation programs emerging from the community, we aim to help NVAL keep up with the advances in blockchain technology that could outpace any team working alone on the framework.

Let us elaborate on NVAL’s operation and utility via the motivating scenario. Figure 1 presents an overview of the framework’s users and workflow. NVAL benefits blockchain network architects and operators, who are members of the participating research labs appointed to design and deploy a private blockchain network. Architects gather requirements (e.g., available hardware, required latency and throughput, the required level of availability, threat models) and leverage the existing architectural knowledge (e.g., [15, 2]) to develop potential blockchain network architectures. Operators deploy and evaluate the given architectures with the help of NVAL. In order to leverage the framework, operators deploy an NVAL instance on a workstation that can access all computers participating in a blockchain network. Operators start a workflow by submitting a request that contains a blockchain network architecture and a list of evaluation metrics for measuring the required level of availability, threat models, and throughput. The framework applies actions according to the plan (Step 7) to deploy and evaluate a blockchain network (Step 8).
returns URLs to access the deployed blockchain network and some datasets containing the required evaluation metrics following a successful execution. If NVAL cannot resolve a plan, it declares the request unsatisfiable.

NVAL realises the stated approach with three novel elements. We propose a novel domain-specific meta-model called Blockchain Network Design (BND) to capture heterogeneous multi-channel blockchain network architectures. We propose the concept and architecture of composable actions to encapsulate and combine the existing automation programs to process new blockchain network architectures specified using BND. Finally, we propose a planning approach based on a state-space search algorithm to match individual blockchain channels from a given blockchain network architecture with composable actions and construct an execution plan for deployment and evaluation. We demonstrated and evaluated the feasibility of NVAL by conducting a case study on an architecture development project that utilised an NVAL proof-of-concept system to evaluate diverse blockchain network architectures empirically. The results show that NVAL successfully leverages seven automation programs to deploy 65 blockchain networks with 12 diverse architectures and generate 295 evaluation datasets. Furthermore, we found that NVAL requires only a tenth of a second (95.5 ms) to plan and orchestrate the deployment and evaluation. This figure is minuscule compared to the total time required for deploying and evaluating a blockchain network, which ranges from 7 to 14 minutes. The contributions of this paper are:

- We conceptualise and design the architecture of NVAL, a software framework for automating blockchain network deployment and evaluation. NVAL supports multi-channel heterogeneous blockchain networks and provides user-driven, community-supported extensibility to address the limitations of the current approaches.

- We propose Blockchain Network Design (BND), a meta-model for capturing blockchain network architecture as digital artefacts to guide the automated deployment.

- We conceptualise and design the architecture of Composable Actions, which encapsulates the existing automation programs into portable software components for runtime composition.

- We propose an automated planning approach for matching blockchain network architectures with composable actions and constructing execution plans.

- We demonstrated and evaluated NVAL via a case study, showing its applicability and efficiency.

2. Background

2.1. Blockchain Network

A blockchain is a share-write database and a secure computing engine operated by mutually-distrusting parties. Every participant hosts and controls some blockchain nodes, which are computers running a software implementation of a blockchain protocol. Blockchain nodes form a peer-to-peer system called a blockchain network to operate and secure a blockchain. Interactions between blockchain nodes follow a blockchain consensus protocol that dictates how they agree on the validity and order of blockchain transactions. The Proof-of-Work (PoW) protocol, also known as Nakamoto consensus [1], is one of the most popular blockchain consensus protocols for public blockchain networks. This protocol allows anonymous Internet users to operate a blockchain securely and resists Sybil attacks, where a participant creates multiple fake identities to influence the consensus process. As PoW-based protocols tend to require significant computing resources, private blockchain networks generally rely on alternative consensus protocols such as Proof-of-Authority (PoA), Raft [36], or Practical Byzantine Fault Tolerance (pBFT) [37]. Cachin and Vukolic [5] provide a survey of prominent consensus protocols.

Blockchain nodes within a blockchain network can assume different roles, which give them different degrees of autonomy and control over a blockchain and demand different amounts of computing resources [15]:

- **Full Nodes** hold complete blockchain replicas. They can verify transactions and execute software scripts embedded in transactions, also known as smart contracts.

- **Mining Nodes** participate in a consensus protocol to process and append transactions. Mining nodes are usually full nodes, but not the other way around.

- **Lightweight Nodes** hold only ledger metadata and must rely on trusted full nodes to verify transactions. They trade security and independence for reduced resource consumption.

- **Remote Nodes** are end-user clients responsible for crafting and signing transactions on behalf of human or machine users. They are also known as wallets.

- **Network Interfaces** offer APIs for remote nodes to interact with a blockchain.

Blockchain networks can be segmented into channels [15], which connect different subsets of blockchain nodes and maintain separate ledgers. Channels in the same network do not necessarily use the same blockchain consensus protocol. For instance, the motivating scenario in
Section 1 introduced a multi-channel heterogeneous blockchain network that contains an inter-lab channel running a resource intensive protocol and multiple intra-lab channels utilizing lightweight protocols.

2.2. Design Decisions and Architecture of a Blockchain Network

Private blockchain networks can be designed and deployed in a controlled manner to ensure their fitness for purpose. For instance, a blockchain network’s architect can decide to split the network into channels to create a separation between different user groups or business processes. An architect can also decide how mining nodes are assigned to participants, reflecting participants’ authority. By controlling the blockchain network’s topology, configuration, and deployment, an architect can fine-tune quality attributes of the network such as performance, resource consumption, availability, and security [15, 2].

Tran et al. defined a design space of blockchain network with 19 types of decisions [11], which can be organized into four types:

- **Topological design decisions** capture the overall shape of a blockchain network. They include how a network is segmented into channels and the number of blockchain nodes per channel.
- **Organizational design decisions** capture the assignment of blockchain nodes to participants. This assignment reflects participants’ governance power and authorization level in the network.
- **Consensus design decisions** capture the type and configuration of the consensus protocols used by blockchain channels in the network.
- **Deployment design decisions** capture the mapping between blockchain nodes and targeted hardware infrastructure.

The architecture of a blockchain network is the collection of its design decisions. Architecture specifications can capture blockchain network architectures as digital artefacts that computers can understand and implement. In this paper, we denote the language for writing such specifications as a meta-model of blockchain network architecture.

3. Approach

NVAL leverages a novel architecture-driven, community-supported approach to automate the deployment and evaluation of complex multi-channel blockchain networks. This approach consists of three phases presented in Figure 1.

The first phase is capturing the abstract and implicit blockchain network architectures as concrete artefacts that NVAL can understand and process. This phase requires a meta-model that defines concepts and relationships necessary for architects to describe blockchain network architectures. We found that the existing view models (4+1 [38]) and notations (Unified Modelling Language (UML)) lack a standardised way to describe blockchain network architectures, leading to ambiguity. Moreover, the existing view models tend to scatter the design decisions across multiple views, increasing the complexity of capturing and understanding design decisions and could lead to inconsistencies. Therefore, we propose a novel domain-specific meta-model for capturing blockchain network architectures called Blockchain Network Design (BND). The meta-model defines standardised concepts, relations, and architectural structures for modelling blockchain network architectures based on an existing design space [15]. Every blockchain network architecture specified using BND simultaneously describes organisational, logical, and physical aspects of a blockchain network, thus mitigating the problem of scattering design decisions across multiple views. Section 3 introduces BND in details.

The second phase is capturing the automation programs as composable software services that NVAL can combine to deploy and evaluate multi-channel heterogeneous blockchain networks. We denote the captured automation programs as composable actions. Operators can develop actions from scratch or refactor them from existing inner-source or open-source automation scripts. Realising the composable action concept raises two challenges: (1) defining action scopes such that composition is possible and (2) minimising effort in developing composable actions. Section 5 introduces our architectural solution for composable actions.

The third phase is planning and conducting the deployment and evaluation of incoming requests, which contain blockchain network architectures and lists of metrics to measure. We regard the action planning for a request as a search task on its implicit state-space with the goal state being the successful deployment and evaluation of a requested blockchain network. This decision stems from the following observations:

- **Every request has a state**, reflecting whether (1) it has been verified, (2) its channels have been deployed, and (3) its requested metrics have been measured.
- **Suitable function calls to actions transit requests between states**. These function calls receive fragments of requests, such as architectural specifications of blockchain channels, as inputs. The suitability of these calls depends on whether the functions can process the inputs and whether a request is at a state where such calls are required. For instance, if a requested blockchain network has not been deployed, it is too early to invoke evaluation actions.
- **The execution plan of a request is, therefore, a sequence of action calls that leads to a goal state.**
4. Blockchain Network Design Meta-model

The Blockchain Network Design (BND) meta-model describes blockchain network architectures as directed property graphs. This approach is motivated by the following observations about the design decisions underlying blockchain network architectures. Observations (1) to (3) show that blockchain network architecture can be modelled as a directed graph whose nodes and edges carry additional properties (a property graph). Observation (4) shows that a blockchain network architecture must also contain the organisational and physical structures around a blockchain network to provide a context for design decisions.

1. Some design decisions involve adding new connections between entities. For instance, the assignment of blockchain nodes to participants can be conceptualised as connections or links between them, specifying “assigned-to” relationships.

2. Some design decisions add both connections and entities. For instance, describing a blockchain network topology requires introducing entities such as blockchain nodes and channels. Describing a topology also requires introducing connections between nodes and channels, specifying “participate-in” relationships.

3. Some design decisions also involve assigning attributes to existing entities and connections. For instance, the type and configuration of consensus protocols can be captured as attributes of a “blockchain channel” entity. Similarly, node type (full, mining, lightweight, remote, interface) can be modelled as an attribute of a “blockchain node” entity.

4. Some design decisions require information beyond the control of an architect. For instance, the decision to deploy a blockchain node on a server requires information about the server, such as its specification and network address. Similarly, the decision to assign a blockchain node to a participant requires details about the participant and possibly its involved business processes. The architect controls neither the server nor the participant yet requires to add details to the architecture to provide context for design decisions.

Meta-model: The BND meta-model defines six vertex types and five edge types that architects can use to build property graphs that represent concrete blockchain network architectures. Figure 2 presents the types of vertices and edges making up the BND meta-model. The vertices describe the necessary entities for specifying design decisions, including blockchain nodes (BNode), blockchain channels, computing node (Node), computer network (Net), participants, and processes. The edges describe the relationships between entities. For example, an in-channel edge connecting a blockchain node and a blockchain channel shows that the node participates in the channel. Vertices and edges can carry additional properties such as computer hardware architecture, network bandwidth, blockchain node types, and consensus protocol configurations. Architects leverage vertices, edges, and properties to capture all the design decisions defined in the blockchain network design space and develop blockchain network architectures. In this paper, we denote the resulting property graphs representing blockchain network architectures as BND models. They can be serialised as JSON or XML documents for consumption by a software system like NVAL.

The resulting BND models describe three perspectives of blockchain network architectures simultaneously: organisational, physical, and logical (Table 2). The organisational structure of a blockchain network captures its organisational context and requirements, describing how participants collaborate and their authority level. The physical structure describes the underlying computing infrastructure available for running a blockchain network, capturing a network’s physical constraints. The organisational and physical structures shape the design decisions captured in the logical structure, which describes a blockchain network’s topology, the deployment of its nodes, and the assignment of its nodes to participants.

Example: Let us demonstrate the meta-model with an exemplary BND model of a two-channel blockchain network in the context of the motivating example (Figure 3). The blockchain network serves five participants within the same lab, including a researcher (denoted as Lab 1) and four robots (Robot 1 to Robot 4). These participants collaborate on a process (Provenance) that secures and updates the historical records (provenance) of mission objectives and corresponding data. The Lab 1 participant also participate in another process called Data Registry with other labs (not shown in the figure for brevity). The details regarding participants and processes are captured in the organisational structure of a BND model, forming the organisational context and requirements.

The blockchain network in Figure 3 operates on a hardware infrastructure consisting of a workstation.
The blockchain network described in Figure 3 contains two channels. The first channel, *Intra-lab*, serves the *Provenance* process. The channel contains five blockchain nodes (1-1 to 1-5) that are assigned to five participants and deployed on five computers. The second channel, *Inter-lab*, serves the *Data Registry* process. Its blockchain nodes are distributed across all participating labs. One of its nodes, 2-1, is assigned to Lab 1 participant and deployed on the *Workstation*. Details regarding the type of blockchain nodes and the configuration of the blockchain channels can be further specified in the properties of relevant entities, which we omit from the figure for brevity and clarity. The logical structure of a BND model captures these design decisions.

**Modelling Architecture with BND:** Before designing a blockchain network, architects must gather requirements regarding the organisational structure that a blockchain network serves. They also need the details regarding the underlying hardware infrastructure. Based on the gathered information, architects fill up a BND model’s organisational and physical structures. Afterwards, architects can capture their design decisions in the logical structure as follows:

- **Topological design decisions** describe how a blockchain network is segmented into channels and the number of blockchain nodes per channel. Operators specify these decisions by (1) adding *blockchain* nodes.
nodes, (2) adding channels, and (3) adding In-Channel edges between nodes and their channels.

- **Organisational design decisions** capture the assignment of blockchain nodes to participants. The type of the assigned blockchain node reflects a participant’s authorisation level. Operators specify these decisions by (1) adding Control-BNode edges between participants and blockchain nodes and (2) specifying the type of blockchain nodes by setting the NodeType property of blockchain nodes.

- **Consensus design decisions** capture the type and configuration of the consensus protocol used by blockchain channels in the network. Operators specify these decisions by setting the ConsensusProtocol property of channels.

- **Deployment design decisions** capture how blockchain nodes are deployed on targeted hardware infrastructure. Operators specify these decisions by adding Deploy-on-Node edges between blockchain nodes and computing nodes.

**Parsing Architecture with BND:** BND enables NVAL to process multi-channel blockchain networks in a channel-by-channel fashion. Let us demonstrate how NVAL parses the exemplary BND model presented in Figure 9 for deployment. The logical structure of the BND model reveals to NVAL that it needs to deploy two blockchain channels (Intra-lab and Inter-lab). Let us take a closer look at the Intra-lab channel. By following its In-Channel edges, NVAL can identify the blockchain nodes that it needs to establish (BNode 1-1 to BNode 1-5). By following the Deploy-on-Node edges of the blockchain nodes, NVAL learns where to deploy the blockchain nodes (Workstation and Onboard 1 to Onboard 4). By inspecting the properties of the Intra-lab channel, NVAL learns the type of required blockchain platform and configurations. By inspecting the properties of the computing nodes, NVAL learns that it needs to deploy blockchain clients on a powerful workstation running an X64-based processor and multiple resource-constrained onboard computers running ARM-v7 processors. The information regarding the required blockchain platform and the hardware architecture form the requirements used by NVAL to look for suitable automation programs. For instance, assuming that the Intra-lab channel utilises the Proof-of-Authority (PoA) variant of Ethereum, NVAL must look for automation programs that can deploy and configure PoA Ethereum clients on both X64-based and ARMv7-based computers. A similar parsing and processing can be performed for the other channel.

The ability to parse and comprehend blockchain network architectures also facilitates new architecture validation capabilities. For instance, verifiers can be developed to leverage heuristic or formal analyses to detect misconfigurations, such as deploying a resource-intensive mining node on a resource-constrained onboard computer.

**5. Composable Actions**

NVAL deploys and evaluates blockchain networks by reusing and combining existing automation programs, denoted as composable actions, according to the given blockchain network architectures. NVAL acquires composable actions via inner-source (codes from previous projects) or open-source channels (codes from the community). This section presents our proposed scope and architecture of composable actions to facilitate their development and reuse.

**Action Scope:** Balancing reusability and abstraction was our goal when defining the scope of composable actions. Based on our experience in developing and refactoring automation scripts for blockchain networks, we proposed to scope NVAL’s actions into three types corresponding to three types of activities conducted by NVAL. Table 1 presents the interface and description features of three action types that NVAL leverages to deploy and evaluate blockchain networks. NVAL requires the automation programs contributed by operators via inner-source or open-source channels to conform to these interfaces for interoperability.

- **Deployers** encapsulate the automation logic to deploy and configure one blockchain channel. They receive a fragment of a BND model describing a channel as input. NVAL distinguishes and chooses deployers based on the type of blockchain platform they support, the types of hardware architecture and operating systems they can work with, and whether they can work with multiple types of hardware architecture within one channel (e.g., X64-based and ARMv7-based hosts).

- **Evaluators** encapsulate the automation logic to measure specific evaluation metrics from one blockchain channel. They receive a list of required metrics and a BND fragment of the targeted blockchain channel as inputs. NVAL distinguishes and chooses evaluators based on the metrics they can measure and the types of blockchain platforms they can work with.

- **Verifiers** encapsulate reusable verifications of incoming requests, such as schema validation and consistency checking. Verifiers are optional actions. We introduced them to capture the potential analyses provided by BND models.

**Action Architecture:** How to ensure NVAL can understand and execute actions, regardless of their underlying technology, whilst minimising the complexity and implementation effort facing action contributors? We address these challenges by proposing a kernel-based microservice architecture for NVAL’s actions. From NVAL’s perspective, every action is a self-contained black box accessible only via a predefined service interface that conforms to ones specified in Table 4 ensuring that NVAL can understand and invoke actions. Every action is packaged and
The process of action is as follows:

1. Determining the type of action to develop based on the defined scope.
2. Refactoring or developing automation programs according to the defined scope.
3. Acquiring the kernel from NVAL.
4. Developing a bridging module by subclassing a suitable abstract module (DeplCore, EvaluatorCore, VerifierCore) and implementing its abstract functions (e.g., deployHandler(), teardownHandler()). At runtime, NVAL calls these functions through the service interface to reach the automation programs. By implementing these functions, contributors control how NVAL uses their automation programs.
5. Filling in the action metadata in the bridging module.
6. Packaging the service interface, the bridging module, the automation programs, and necessary utilities into a container. The kernel can provide a template for conducting this step.

Rationale and Counter-Example: The action scopes are informed by our experiences in developing automation scripts for blockchain networks. Let us consider the following a counter-example automation script we developed for the exemplary BND model above. The script received a list of computing hosts as input. Each list entry specifies a blockchain node’s network address and channel. The counter-example script instructs Ansible, an automation tool, to (1) deploy one Inter-lab channel using the PoW Ethereum platform, (2) deploy multiple Intra-lab channels with the PoA Ethereum platform, and (3) benchmark all channels using Hyperledger Caliper. From NVAL’s perspective, the described automation script is an overly large action that simultaneously performs multiple tasks. We made the following observations:

1. Overly large action scopes hamper reuse: For instance, we cannot reuse the counter-example script to deploy and evaluate PoA Ethereum channels in a different blockchain network directly because its ability to deploy PoA Ethereum channels is tightly coupled with deploying PoW Ethereum channels and running Caliper benchmarks.
2. Overly small scopes hamper abstraction: For instance, if we narrow down actions to specific technical steps such as building and deploying an Ethereum client, operators would need an in-depth understanding of Ethereum to use these actions. In other words, we lose the abstraction of implementation details, which is NVAL’s primary goal.
3. Deploying and evaluating channels are atomic activities. From an operator’s perspective, deployment and evaluation are either successful or failed. In other words, it does not matter to them whether the failure happens at the first compilation step or the last provisioning step. Therefore, the scope of actions must cover an entire deployment or evaluation step to preserve their abstraction.
4. Channels divide the work on a multi-channel blockchain network into independent units. For instance, we can break the deployment and evaluation of a three-channel blockchain network into three parts; each deploys and evaluates one channel. The same procedure can be used for all channels if they employ similar blockchain technology and request the same evaluation metrics.
5. Actions can be distinguished by what channels they...
can act on and how they do so. For instance, a deployment action might be able to build and deploy Ethereum nodes only on computers with an ARMv7 processor. This information helps NVAL match this action with blockchain network architectures that feature Ethereum protocol and low-power computers running ARMv7 chips.

The observations (1) to (3) show that deployment and evaluation of individual channels are atomic activities for handling heterogeneous multi-channel blockchain networks. Observation (4) shows that we need to reduce action scopes to individual blockchain network channels rather than an entire network to increase their reusability. Finally, observation (5) shows us how to describe actions to enable their reuse. The proposed action scopes in Table 1 were developed based on these five observations.

Composable Actions in Action Let us demonstrate how the counter-example automation script can be refactored into composable actions. Based on the scopes defined in Table 1, the script can be refactored into three following actions:

- **Deployer 1** specialises in PoW Ethereum. Assuming that it only works with computers running X64-based processors, we can denote this deployer as $\text{Depl}(\text{PoWth,X64},\text{True})$, meaning it supports PoW Ethereum, X64-based hosts and can work with infrastructure containing multiple host types.

- **Deployer 2** specialises in PoA Ethereum. Assuming that it can deploy on both ARMv7 and X64 hosts, we can denote this deployer as $\text{Depl}(\text{PoAeth,ARMv7\&X64},\text{True})$.

- **Evaluator 1** specialises in measuring performance metrics such as latency and throughput from Ethereum channels using the Caliper benchmark suite. We denote this evaluator as $\text{Eval}(\text{Perf,Eth})$.

NVAL can leverage these actions to handle the exemplary BND model presented in Figure 3. The Inter-lab channel matches with the Deployer 1 ($\text{Depl}(\text{PoWth,X64},\text{True})$) because the channel runs the PoW Ethereum platform and is deployed on a workstation running X64-based processors, thus requiring the capabilities that Deployer 1 provides. The Intra-lab channel matches with Deployer 2 ($\text{Depl}(\text{PoAeth,ARMv7\&X64},\text{True})$). The evaluator can be applied to both channels. The execution plan for the exemplary BND model is as follows:

1. $\text{Depl}(\text{PoWth,X64},\text{True})(\text{Inter – lab})$
2. $\text{Depl}(\text{PoAeth,ARMv7\&X64},\text{True})(\text{Intra – lab})$
3. $\text{Eval}(\text{Perf,Eth})(\text{Inter – lab})$
4. $\text{Eval}(\text{Perf,Eth})(\text{Intra – lab})$

6. Action Planning with State Space Search

Action planning denotes the process of finding a suitable sequence of action calls to deploy and evaluate a requested blockchain network architecture.

**Problem Formulation:** We regard the action planning for a request as a search task on its implicit state-space with the goal state being the successful deployment and evaluation of a requested blockchain network. We formalise the state-space search problem for action planning as follows.

**Definition 6.1** (Request). A request ($\mathcal{R}$) from an operator contains a blockchain network architecture for deployment and a list of metrics to measure from the deployed network. A request consists of multiple $n$ deployment and $m$ evaluation request components that can be tracked and resolved separately: $\mathcal{R} = (\mathcal{D}, \mathcal{E})$. A deployment request component is an individual blockchain channel to deploy. An evaluation request component is a list of metrics to measure from a channel. The state of a request is denoted as $\mathcal{S}_\mathcal{R}$.

**Definition 6.2** (Action). An action is a function that (1) receives a request component, (2) carries out a corresponding task, and (3) returns a new request state. NVAL has three action types: verifier (Eq. 2), deployers (Eq. 3), and

---

Figure 4: Architecture of a composable action for NVAL.
evaluators (Eq. 3). The set of actions that NVAL can access to plan and resolve a request is denoted as an action pool \( \mathcal{A} \) (Eq. 1).

\[
\mathcal{A} = \{ \text{Ver}, \text{Depl}, \text{Eval} \}
\]  

\[
\text{Ver} = \{ \text{ver} : \mathcal{R} \rightarrow \mathcal{B} \}
\]  

\[
\text{Depl} = \{ \text{depl} : \mathcal{D} \rightarrow \mathcal{S}_R \}
\]  

\[
\text{Eval} = \{ \text{eval} : \mathcal{E} \rightarrow \mathcal{S}_R \}
\]

**Definition 6.3 (Action Planning).** The action planning problem can be formalised as follows: given a request \( \mathcal{R} \), generate an action sequence \( \mathcal{A}_R = ((a, \text{input}) | a \in \mathcal{A}, \text{input} \in \mathcal{R}) \) such that the request’s state \( \mathcal{S}_R \) becomes a goal state after applying \( \mathcal{A}_R \).

**Approach:** We apply state-space search to solve the action planning problem. The main idea is employing a search algorithm (e.g., depth-first search, breadth-first search) to systematically explore a request’s state space to find a goal state. The path taken by the search algorithms to reach a goal state becomes the execution plan. The traversal of a request’s state space requires a state space model and a set of state transition functions that we define below.

**Request’s State Model:** Following [39], we model a request’s state space as a tree whose nodes represent the request’s states and edges represent action calls \((a, \text{input})\). A request’s state \( \mathcal{S}_R \) is a tuple with \( 3 + n + m \) elements (Eq. 5). The first three elements \((v, d, e)\) describe a request’s overall status, showing whether it has been verified, deployed and evaluated. The remaining \( n + m \) elements describe the status of individual request components, including failed (-1), in-progress (0), and successful (1). The initial state of every request is an all-zero tuple. Its goal state is an all-one tuple.

\[
\mathcal{S}_R = (v, d, e, \text{req})
\]  

\[
\text{req} \in \{-1, 0, 1\}^{n+m}
\]  

\[
v \in \mathcal{B}
\]  

\[
d, e \in \{-1, 0, 1\}
\]

**State Transition Functions:** The following state transition functions control how a search algorithm traverse a request’s state space.

\( \text{ACTIONS}(s) \) determines the possible out-going edges for each state. Each out-going edge represents an action call \((a, \text{input})\). The function returns action calls according to the following rules:

- If \( v \) is 0, return all verifiers \( \text{Ver} \)
- If \( d \) is 0, return all deployers \( \text{Depl} \) with deployment request components as their input. For instance, if a given blockchain network architecture contains two channels \((\text{ch1}, \text{ch2})\), and NVAL has two deployers \((\text{PoW()}, \text{PoA}())\), then \( \text{ACTION}(a) \) returns four actions: \( \text{PoW(ch1)}, \text{PoW(ch2)}, \text{PoA(ch1)}, \text{PoA(ch2)} \).
- If \( d \) is 1 and \( e \) is 0, return all evaluators \( \text{Eval} \) with evaluation request components as their input, similarly to the above example with deployers.
RESULT(s, a) determines the destination of an out-going edge. In other words, it returns a new state that captures the results of applying an action a to an initial state s. The results of this function are according to the following rules:

- If \( a \in Ver \), the verification state \( v \) is set to 1 to indicate that the verification has been planned.
- If \( a \in Depl \), the new state depends on whether the deployer in a can handle a given request component. This assessment is done by matching the deployer’s descriptive features with the channel’s characteristics (e.g., matching the type of supported blockchain platforms with the type of requested blockchain platform). If a deployer can handle a given deployment request component, then the request component’s status in \( S_E^p \) is set to 1, indicating that deployment has been planned. If all deployment request components have been planned, the overall deployment state \( d \) is updated to 1.
- If \( a \in Eval \), the new state depends on whether the evaluator in a can handle a given evaluation request component. If it can, then the status of the relevant request component is set to 1, indicating that evaluation has been planned. If all evaluation request components have been planned, the overall evaluation state \( e \) would also be updated to 1.

STEP-COST(s, a) returns the cost of applying an action a on a state s. This cost can represent an action’s resource consumption (CPU time, memory, network bandwidth), time delay, reliability, or financial costs. For simplicity, we set the STEP-COST uniformly to be one, which means the total cost of an execution plan is the number of actions taken. Future works could explore other STEP-COST to improve the efficiency of the state space search.

IS-GOAL(s) determines whether a given state \( s \) is the goal state. It returns true if all values of a state tuple \( S_E^v \) is 1. If the state space is exhausted before IS-GOAL(s) returns true, the request is deemed unsatisfiable.

Figure 3 depicts how a breadth-first search algorithm leverages the state transition functions to traverse the state space to search for a goal state. The node object represents a node in the state tree, which contains a state vector \( s \) and a set of actions leading to it. The POP() removes and returns the last item from an array.

State-space Search in Action: We describe an example that shows how NVAL plans to deploy and evaluate the exemplary BND model described in Section 4 (Figure 3). Recall that the exemplary BND model contains an inter-lab channel and an intra-lab channel, which we denote as ch1 and ch2. The first channel (ch1) uses a PoW Ethereum variant to connect workstations of the collaborating labs (X64-based computers). The other channel (ch2) is an intra-lab channel that uses a PoA Ethereum variant to connect robots (ARMv7-based computers) and workstations (X64-based computers) within a lab.

The state tuple of the sample request consists of eight components (three general states \( v, d, e \), two deployment requests, and three evaluation requests). The request is initially assigned with an all-zero state tuple \( (0, 0, 0, 0, 0, 0, 0, 0) \). The breadth-first search algorithm presented in Figure 6 was employed to traverse the state space. Let’s consider the expansion of the initial state as an example.

1. IS-GOAL(s) is applied on the initial state. It returns false, allowing the search to continue.
2. ACTIONS(s) is applied on the initial state to identify all available actions. Because \( v = 0 \), all verifiers are returned as possible actions.
3. RESULT(s, a) is applied on the starting state and the verifier action. It returns a new state \( (1, 0, 0, 0, 0, 0, 0, 0) \), reflecting that verification has been planned.
4. STEP-COST(s, a) assigns the cost of the verification to be 1, indicating that one action has been planned.

The traversal continues until reaching a goal state. All actions leading to that state becomes the experiment plan for the given request. Figure 6 depicts the state tree and the search process.

7. Evaluation

This section presents a case study that aims to evaluate and demonstrate the feasibility of NVAL in a targeted usage scenario. We validated the framework’s feasibility in two aspects. Firstly, we demonstrated that NVAL could combine a small set of composable actions to deploy and evaluate a larger set of complex blockchain network architectures. Secondly, we demonstrated that NVAL does not incur significant overheads to the deployment and evaluation process. We formulated these aspects of feasibility into the following research questions:

- RQ1: How effective is NVAL at orchestrating actions for deploying and evaluating diverse blockchain network architectures?
- RQ2: What is the degree to which NVAL can reuse actions across requests?
- RQ3: What is the processing time overheads incurred by NVAL?
We chose the case study method to evaluate and demonstrate NVAL’s feasibility because it is a powerful and flexible research method to investigate a contemporary phenomenon in its context [40, 41]. The studied phenomenon is the feasibility of NVAL. The context in which we study NVAL’s feasibility is an architecture development project that employs a proof-of-concept system of NVAL to experiment with various blockchain network architectures. For clarity, we use the term case to denote the architecture development project (context) and the term case study or study to denote the evaluation of NVAL via analysing the case. We collected data from the experiment results and NVAL’s logs by the end of the case as empirical evidence to answer the research questions. The following sections elaborate on the case, the methodology, and the results.

7.1. Case Description

Context: Recall that in the motivating scenario, multiple collaborating research labs wish to leverage a private blockchain to interconnect their robotic testbeds to form a joint experimentation platform. The labs must establish a private blockchain network that bridges various organisations’ clouds, workstations, and robots to leverage a private blockchain. Such a network has various potential architectures [2], which offer different trade-offs between quality attributes such as performance (transaction latency and throughput), resource consumption, availability, and security.

The case of our study is an architecture development project that emerged from this context. It aimed to identify an optimal blockchain network architecture for the joint experimentation platform by empirically evaluating potential architectures using a surrogate testbed infrastructure. The testbed resembles a scaled version of the joint experimentation platform. The project includes 13 experiments, each of which involves the deployment and benchmarking of a potential blockchain network architecture. The experiments were repeated five times, bringing the total number of deployed blockchain deployed channels to 150, constituting 65 blockchain networks. Most of the deployed blockchain channels produce two evalu-
ation datasets (performance and resource consumption), bringing the total number of datasets to 295. Due to the massive workload, the authors were involved in the project to employ NVAL to support the experiments.

**Surrogate Testbed Infrastructure**: The surrogate testbed infrastructure is a scaled version of the joint experimentation platform. The testbed is deployed and operated by the authors to conduct experiments in a controlled environment. Figure 7 presents the testbed, which contains two device clusters representing separate robotic testbeds from two research labs. Each cluster contains four edge devices, which are resource-constrained computers (Raspberry Pi 3B+ and Raspberry Pi 4) representing robots’ onboard controllers. Each cluster contains a powerful workstation acting as a cluster head or base station. The testbed also includes a workstation representing a remote cloud infrastructure that the research labs in the motivating scenario might utilise for data storage and processing. Upon this testbed infrastructure, experiments on blockchain network architectures were conducted.

**Experimented Architectures**: Twelve potential blockchain network architectures were deployed and evaluated by NVAL. Table 2 presents a detailed breakdown of these architectures in terms of their constituting blockchain channels, where channels’ full nodes are deployed (coverage), where channels’ mining nodes locate, and the type of blockchain consensus protocol that channels use. The experimented architectures are complex and heterogeneous, featuring both single-channel (D4 to D6) and multichannel topology (D1 to D3 and D7 to D12) and deploying blockchain nodes on both X64-based computers (base station) and ARMv7-based computers (edge agents).

Table 2: Blockchain network architectures deployed and evaluated by NVAL in the case study

| Arch ID | Channel ID | Coverage | Miner Location | Consensus |
|---------|------------|----------|----------------|-----------|
| D1      | Ch1, Ch2   | Within edge clusters | Edge devices | Ethereum PoA |
| D2      | Ch1, Ch2   | Within edge clusters | Fog nodes | Ethereum PoA |
| D3      | Ch1, Ch2   | Within edge clusters | Fog nodes | Ethereum PoW |
| D4      | Ch1        | Across device clusters | Fog nodes | Ethereum PoA |
| D5      | Ch1        | Across device clusters | Fog nodes | Ethereum PoW |
| D6      | Ch1        | Across device clusters | Fog nodes & Edge devices | Ethereum PoA |
| D7      | Ch1, Ch2   | Within edge clusters | Fog nodes | Ethereum PoA |
| Ch3     | Across device clusters | Fog nodes | Ethereum PoA |
| D8      | Ch1, Ch2   | Within edge clusters | Fog nodes | Ethereum PoW |
| Ch3     | Across device clusters | Fog nodes | Ethereum PoW |
| D9      | Ch1, Ch2   | Within edge clusters | Cloud | Ethereum PoA |
| Ch3     | Across device clusters | Cloud | Ethereum PoA |
| D10     | Ch1, Ch2   | Within edge clusters | Cloud | Ethereum PoW |
| Ch3     | Across device clusters | Cloud | Ethereum PoW |
| D11     | Ch1, Ch2   | Within edge clusters | Fog node | Ethereum PoA |
| Ch3     | Across device clusters | Fog node | Ethereum PoA |
| Ch4     | Across device clusters | Cloud | Ethereum PoA |
| D12     | Ch1, Ch2   | Within edge clusters | Fog node | Ethereum PoW |
| Ch3     | Across device clusters | Fog node | Ethereum PoW |
| Ch4     | Across device clusters | Cloud | Ethereum PoA |

**NVAL’s Action Pool**: The proof-of-concept system contained four deployers, two evaluators, and one verifier to deploy and evaluate the above blockchain network architectures. Table 3 presents the scopes of these actions. The authors constructed all the composable actions by refactoring existing automation programs previously developed for an IT automation tool called Ansible. **DeplPoA** was the first developed deployer, resembling the counter-example described in Section 5. The deployer was refined and extended to support more blockchain platforms (Ethereum Proof-of-Work) and hardware architecture (ARMv7), creating DeplPoA_x64_ARM and **DeplPoW_x64_ARM**. EvalPerformanceEthereum evaluator is capable of conducting performance benchmarks using the Hyperledger Caliper suite and measuring resource consumption of the underlying hardware using Prometheus. It supports both PoA and PoW variants of Ethereum. The **Verifier** action is capable of validating incoming deployment requests against a schema to detect syntactical issues.

The action pool of NVAL also contains **DeplBaseline** and **EvalBaseline**, which are special actions for determining the baseline resource consumption level of the testbed infrastructure. The baseline deployer deploys a unique blockchain channel called “baseline”, which does not contain any active blockchain node. The baseline evaluator only works with “baseline” channels. It measures the resource consumption metrics of nodes underlying these channels over a long period.

Table 3: Action Pool

| Deployer | Supporting Platform | Supporting Arch |
|----------|---------------------|-----------------|
| DeplPoA  | Ethereum PoA (Clique) | X64             |
| DeplPoA_x64_ARM | Ethereum PoA (Clique) | Mixed X64 & ARM |
| DeplPoW_x64_ARM | Ethereum PoW (Ethash) | Mixed X64 & ARM |
| DeplBaseline | Baseline | Mixed X64 & ARM |

| Evaluators | Supporting Platform | Supporting Metrics |
|------------|---------------------|---------------------|
| EvalBaseline | Ethereum | Resource Consumption & Performance |
| EvalPerformanceEthereum | Ethereum | Resource Consumption |

| Verifiers | Approach | Technique |
|-----------|----------|-----------|
| Verifier  | Structural verification | Schema validation |

7.2. Proof-of-Concept Implementation and Deployment

We developed and deployed a NVAL proof-of-concept system to conduct the case study. Figure 7 depicts the deployment structure of a NVAL instance:

- **NVAL Server** hosts the framework’s software components. It exposes functionality as Web services via a RESTful API. We used the Open API Specification version 3.0 to model the API and employed the Swagger Codegen utility to generate the service interface automatically from the specification. Javascript and NodeJS runtime environment were used to implement NVAL’s components.
• **Operator’s Computer** hosts the client-facing software for creating BND models, submitting requests to NVAL and monitoring the experiments (i.e., deployment and evaluation of a blockchain network). We developed the web application with ReactJS and [react-digraph](https://github.com/uber/react-digraph) libraries.

• **Target hardware infrastructures** consist of computers and networks used for deploying and evaluating blockchain networks (Figure 7). NVAL server can reach these computers via SSH.

Both the NVAL server and the client-facing software are deployed on a mobile workstation equipped with a quad-core processor and 16GB of memory.

7.3. Evaluation Metrics

We collected data to answer the research questions from experiment results and log files generated by NVAL’s proof-of-concept system. We calculate the following metrics based on the data to quantify the effectiveness and efficiency of NVAL.

**Execution Success Rate:** Success rate is the ratio between successful experiment runs and the total number of experiments. An experiment is successful if NVAL applies correct actions in a proper order to deploy a blockchain network correctly and retrieve all the requested performance metrics. The success rate is determined from the experiment results and used to answer RQ1. A high success rate indicates that NVAL is effective at deploying and evaluating blockchain networks.

**Per Action Utilisation Rate:** The utilisation rate of an action quantifies the degree to which NVAL can reuse it across multiple requests. This metric is used to answer RQ2. We calculated the utilisation rate from the experiment results in two following ways:

- **Ratio between the number of experiments in which an action participates and the total number of experiments**
- **Ratio between the number of times NVAL invokes an action, and the total number of action calls NVAL performs across the experiments**

**Processing Time Overheads:** This metric quantifies the amount of experiment time spent on NVAL-specific activities such as creating execution plans and invoking actions. We calculated this metric by analysing the NVAL’s log files that show us starting and ending times of every activity conducted within NVAL (Figure 8). This metric is used to answer RQ3. Lower overheads indicate a higher level of efficiency.
7.4. Results

RQ1 - The Effectiveness of NVAL: NVAL framework successfully planned and conducted five rounds of 13 experiments (twelve architecture and one baseline), leading to the deployment and evaluation of 150 blockchain channels, constituting 65 blockchain networks. All evaluation datasets were usable and free of anomalies. We also did not observe any false-negative from the planning mechanism as the action pool was adequate to satisfy all the given requests. Given these observations, we reached the following conclusion for RQ1:

| RQ1: | Given an adequate action pool, NVAL plan and conduct experiments accurately for 100% of the given experiment requests. |

RQ2 - Utilisation Rate of Composable Actions: NVAL’s action pool served 365 invocations across five rounds of 13 experiments. Table 9a presents the utilisation rate of each action across the experiments. Table 9b breaks down the utilisation of each action, showing the number of times it was called across the experiments.

The data shows that actions with broad functional scopes can support (i.e., being utilised by) more experiments. For instance, 92% of the experiments utilised EvalPerformanceEthereum because it can measure both performance and resource consumption, and it supports both protocol variants of Ethereum. Critical actions that are universally applicable, such as Verifier have almost perfect reusability with NVAL. On the other hand, actions with low demand and limited scopes, such as DeplBaseline and EvalBaseline, had a low utilisation rate.

An interesting observation from the data is that multiple actions with limited scopes can be combined to cover as many or even more experiments than larger actions. For instance, the DeplPoA deployer supports only 23% of the experiments by itself. However, the combination of three deployers working together was able to deploy all 12 blockchain network architectures. This observation suggested the feasibility of NVAL’s automation approach of reusing and combining existing automation programs to support future requests. We reached the following conclusion for RQ2:

| RQ2: | NVAL can reuse an action pool to deploy and evaluate new blockchain networks. It can orchestrate a small set of actions to serve a substantially larger set of requests. |

RQ3 - Processing Time Overheads: Figure 10 presents the durations of various activities conducted by NVAL. The time spent on all activities beside VERIFY, DEPLOY, and EVALUATION are considered overhead. The overhead can be classified into bootstrap overhead and experiment overhead. Bootstrap overhead is caused by the LOAD-MODULES activity that is performed once when NVAL boots. We found bootstrap overhead to be around half a second (441 ms).

Experiment overhead is caused by activities related to recording, planning, and orchestrating the deployment and evaluation. This overhead incurs whenever an experiment is conducted. We found the experiment overhead to be around a tenth of a second (95.5 ms), which is minuscule compared to the time required to deploy and benchmark a blockchain network, ranging from 7 to 14 minutes. Based on these findings, we reached the following conclusion for RQ3:

| RQ3: | NVAL’s overheads are negligible compared to the time required to deploy and evaluate blockchain networks. |

8. Discussions

8.1. Benefits and Usage Scenario

The NVAL framework provides practitioners and researchers with the ability to rapidly deploy and evaluate heterogeneous multi-channel blockchain networks. By leveraging BND meta-model, composable actions, and state-space search, the framework can deploy and evaluate blockchain network directly based on their architecture whilst abstracting procedural details and low-level configurations from users. This architecture-driven automation approach benefits practitioners and researchers by allowing them to focus on designing blockchain network architecture, a valuable and hard-to-automate activity. Automation of implementation and configuration steps also prevent errors such as missing steps and misconfigurations, reducing time and effort in blockchain network deployment and evaluation. Another benefit of NVAL is the capturing and sharing of procedural knowledge regarding deployment and evaluation activities. By capturing this knowledge explicitly as software artefacts (composable actions) rather than documents, practitioners and researchers can version control, test, refine, and employ the knowledge rapidly across multiple organisations. Finally, by enabling practitioners to rapidly deploy and evaluate their blockchain network architectures, NVAL can shorten the feedback loop and enable practitioners to conduct more design iterations, which ultimately lead to better architectures.

Unlike existing blockchain automation framework and services, NVAL offers practitioners complete control and transparency over the framework’s capability. NVAL is designed such that it can be deployed and operated by individual practitioners on consumer hardware, as demonstrated in the case study. These practitioners maintain a complete control over the capability of their NVAL instances as they can choose the actions making up the action pool of their instances. The kernel-based approach employed in these composable actions facilitate the development and sharing of new NVAL’s capabilities, without relying on permission or approval from any central authority such as service providers or project maintainers. These mechanisms together enable NVAL to keep up with the developments in blockchain technology and maintain its position as a leading research tool.
ever changing landscape of blockchain technology, increasing its transparency and longevity.

### 8.2. NVAL and Beyond

The case study has shown NVAL’s effectiveness in reusing and composing a small pool of automation programs to deploy and evaluate diverse and complex blockchain network architectures. However, our approach and the current implementation still have some drawbacks. This section analyses these drawbacks and presents some potential solutions, constituting future research.

Firstly, because NVAL reuses existing automation programs to deploy and evaluate blockchain networks, its capability is limited by the breadth of actions it can access. Therefore, the prime challenge of NVAL is ensuring a steady supply of composable actions for both existing and emerging blockchain technologies. The interfaces and the kernel-based architecture proposed in Section 5 lay a foundation for practitioners to capture and share their procedural knowledge as composable actions. Future research could focus on the automated transformation of the existing automation scripts and natural language instructions into composable actions. As of February 2022, we found 52 open-source repositories on automated blockchain network deployment with Ansible, 67 repositories with Terraform, and 35 repositories with other scripting languages on GitHub. The official documentation of many blockchain implementations also provide instructions for deploying private networks

> These automation scripts and tutorials represent raw materials for NVAL’s actions.

Secondly, the current version of NVAL does not feature extensive verifications of the contributed actions beyond manual inspections by practitioners. The rationale of this design is the assumption that actions primarily come from trusted inner-sources, and the community verifies open-source actions before they reach an end-user. This model might not scale when NVAL’s actions proliferate and increasingly become targets of malicious actors who want to leverage them as an attack vector to infrastructures. Future research could develop a resilient and decentralised registry for discovering and reporting malicious actions. The \( \text{STEP-COST()} \) function could leverage the ratings from this registry to adjust the planning procedure.

Thirdly, the current version of NVAL only employs Unicode strings and a substring detection mechanism to describe and match actions with architectural specifications written in BND. For instance, if a blockchain channel is deployed on X64-based computers and uses Ethereum PoW blockchain protocol, NVAL would match the channel with any action containing X64 and PoW in its meta-data. The case study demonstrated that the simple approach is effective in matching channels with actions based on their functionality. However, substring detection might not be adequate for sophisticated action matching that takes into account

---

4 https://geth.ethereum.org/docs/interface/

4\#private-network
**Table:**

| Activity                        | Avg. Duration (milisec.) | Max Duration (milisec.) | Std. Dev (milisec.) |
|---------------------------------|--------------------------|-------------------------|---------------------|
| STORE-EXP-RECORD                | 0.48                     | 3                       | 0.62                |
| VERIFY                          | 2.49                     | 18                      | 2.94                |
| ORCHESTRATION-OVERHEADS         | 5.66                     | 28                      | 4.99                |
| CREATE-EXP-RECORD               | 42                       | 78                      | 9.23                |
| PLANNING-OVERHEADS              | 47.37                    | 85                      | 19.38               |
| LOAD-MODULES                    | 441                      | 761                     | 80.68               |
| DEPLOY                          | 40377                    | 164084                  | 25187.3             |
| EVALUATION                      | 440864                   | 791532                  | 169850.88           |

(a) Durations of activities carried out by NVAL

| Activity                        | Avg. Duration (milisec.) | Max Duration (milisec.) | Std. Dev. (milisec.) |
|---------------------------------|--------------------------|-------------------------|---------------------|
| ACTUAL-DEPLOYMENT-EVALUATION    | 436675                   | 861352                  | 168265.33           |
| TOTAL-OVERHEADS                 | 95.5 (0.022%)            | 176                     | 29.9                |

(b) Total overheads of NVAL against the actual runtime

Figure 10: Durations of NVAL activities and the framework’s overheads.

---

consideration non-functional attributes of actions such as performance, reliability, and security. Future research could explore semantic descriptions and matching mechanisms. For instance, ontologies could replace the Unicode string to extend the description of actions to cover non-functional requirements. Ontology-based query can replace substring detection in matching channels with actions.

Finally, NVAL currently does not automatically feed the execution phase’s output to the planning phase. Practitioners must inspect the logs manually when a deployment fails and determine whether the failure is due to an exception or an action’s failure. If they detect that a composable action is the root cause, they need to remove it from the pool manually. Future research could explore techniques from self-adaptive software systems, such as MAPE-K feedback loops [42] to enable NVAL to observe and update its action pool automatically.

8.3. Threats to Validity

8.3.1. Construct validity

Construct validity concerns whether our case study design successfully assesses NVAL’s effectiveness and efficiency in real-world usage. We mitigated this threat by following an established guideline [11], which requires that case selection and data collection follow research objectives.

The selected case supports our research objectives because (1) it is a real-world use case of NVAL, (2) it requires the rapid deployment and evaluation of many blockchain networks, and (3) these blockchain networks have diverse and complex multi-channel architectures.

The data collection process was also driven by the objectives of assessing NVAL’s effectiveness and efficiency. We chose success and failure rates to quantify NVAL’s effectiveness because these metrics reflect the framework’s ability to correctly use an existing action pool to deploy and evaluate blockchain networks. We chose time overheads to quantify NVAL’s efficiency because time is the most tangible and significant resource that NVAL consumes, given that it is not computing-intensive.

8.3.2. External validity

External validity concerns whether the effectiveness and efficiency that NVAL shows in the case study can be generalized. Whilst the case study method does not aim to provide statistically significant conclusions [11], we attempted to broaden the generalization of our results by choosing a case that features diverse blockchain network architectures extracted from the literature. The variety of network topology and the inclusion of edge devices and fog nodes expand our generalization scope beyond enterprise blockchain use cases, covering the emerging use cases in cyber-physical systems and edge computing.

8.3.3. Reliability

Reliability concerns whether case study results depend on the researchers carrying out the study. We mitigated threats to reliability by the following strategies. Firstly, we avoided subjective data such as NVAL’s usability and focused on objective data gathered from the source code and runtime logs. Secondly, we only employed commodity hardware for NVAL and the experiment infrastructure to prevent skewing the efficiency results due to exclusive, high-performing computers.

9. Related Work

This paper addresses the problem of automating blockchain network deployment and evaluation, which contributes to the Blockchain Oriented Software Engineering (BOSE) research [13, 14, 15] and aligns with three research areas: blockchain network architecture, blockchain network deployment, and evaluation.
Utility for deploying business abstraction such as assets over an existing HL Fabric network. Deprecated in 08/2021.

An automation script for deploying and configuring an HL Sawtooth client on a computer.

Utility for deploying containerised blockchain network on any Kubernetes cluster.

One-click network deployment based on user-defined profiles.

A platform for deploying Ethereum clients wrapped by software agents that handle monitoring and off-chain communication.

A large-scale IP network simulator that supports deployment of distributed applications (blockchain networks).

A benchmark suite that can deploy blockchain nodes on private clouds in addition to running benchmarks.

A utility for deploying containerised blockchain network on any Kubernetes cluster.

A benchmark suite that can deploy blockchain nodes on private clouds in addition to running benchmarks.

A benchmark suite that can deploy blockchain nodes on private clouds in addition to running benchmarks.

A large-scale IP network simulator that supports deployment of distributed applications (blockchain networks).

A platform for deploying Ethereum clients wrapped by software agents that handle monitoring and off-chain communication.

A utility for deploying containerised blockchain network on any Kubernetes cluster.

9.1. Blockchain Network Deployment

Automated blockchain network deployment has been investigated by both academia and industry to abstract the time-consuming, and error-prone deployment process, allowing practitioners to focus on application development instead of learning networks and protocols. Automation has also been sought to make blockchain network deployment repeatable to support regression tests and continuous delivery.

Hyperledger Composer is amongst the first tools that recognise and address the complexity of establishing a private blockchain. It helps practitioners implement organisational design decisions by providing them with models and tools to deploy business abstractions such as assets and participants on an existing Hyperledger Fabric network. The Composer was deprecated in 2021.

The automation tools released after Composer have shifted their focus to blockchain networks’ logical and physical design decisions. Many of these tools feature some form of domain-specific languages to capture the topology and configurations of the requested blockchain networks. The PlaTIBART platform is among the first to tackle blockchain network deployment, focusing on Ethereum. It proposes a domain-specific language to describe Ethereum’s configurations, such as network ID and mining difficulty level. A Python-based network manager uses the configuration files to deploy Ethereum networks on connected computers. Hyperledger Bevel, formerly known as the Blockchain Automation Framework, is a state-of-the-art tool for rapidly deploying production-ready blockchain networks, supporting Fabric, Corda, Indy, and Quorum blockchain protocols. It uses Anisible, Helm, and Kubernetes to rapidly provision containerised blockchain networks on any Kubernetes cluster. Bevel’s inputs are low-level and technology-specific configurations such as consensus types, network addresses and ports. Gorski et al. proposed an approach similar to Bevel, but relying on UML deployment models rather than domain-specific models as inputs. Malik et al. by-passed blockchain network configuration files by proposing a shell script that practitioners manually place on targeted computers and invoke to deploy blockchain nodes. Alternatively, Li et al. proposed an algorithm automate the design and deployment of blockchain network based on the characteristics of sensor data and the underlying edge computing network.

Many Blockchain-as-a-Service (BaaS) platforms such as uBaaS, NutBaaS, RBaaS, AWS Managed Blockchain, Azure BaaS, and IBM Blockchain Platform also provide deployment automation as a prominent feature. They generally provide a “one-click deployment” utility for spawning a blockchain network across cloud-based virtual machines. This deployment utility is generally vendor-locked in terms of the supported blockchain technologies and computing nodes. Few offer any form of extensibility besides promised first-party extensions. A recent study on DevOps practices for blockchain-based software engineering found that while BaaS usage is becoming more common, vendor-specific limitations in terms of the supported blockchain platforms and hardware limit their adoption in blockchain-based software projects.
29. A few blockchain benchmark tools also support blockchain network deployment. For instance, MixBytes Tank [33] helps practitioners deploy a private Polkadot or Haya blockchain on private clouds before running a benchmark. MADT [17] is a Python-based network simulator that allows practitioners to deploy a private blockchain on the simulated network for development and evaluation.

Table 4 compares NVAL with the existing work. NVAL shares the core approach with PlaTIBART and Bevel, which involves capturing blockchain network architectures as digital artefacts and employing a centralised network manager to realise the requested blockchain networks. Unlike the existing system, NVAL’s meta-model (Section 4) enables architects to specify blockchain network architectures at a design-decision-level rather than a platform-configuration-level and allows them to capture multi-channel heterogeneous blockchain network architecture. The flexibility of blockchain network architecture input is facilitated by community-contributed automation programs (Section 5) and an automated planning approach (Section 6). As demonstrated in the case study, NVAL can combine multiple automation programs to process complex multi-channel heterogeneous blockchain networks at runtime. Moreover, end-users such as blockchain network operators can control and extend the framework’s capability by acquiring and adding actions to their NVAL instance rather than relying on a project owner or service provider. Besides deployment, NVAL also supports evaluation automation by introducing the evaluator action type to capture evaluation automation tasks and embedding evaluation requests and states into NVAL’s planning procedure. Finally, the case study demonstrates that NVAL only requires SSH connections to target computers to deploy and evaluate blockchain networks. Therefore, it can support any infrastructure rather than be constrained to a specific cloud.

9.2. Blockchain Network Architecture

This paper also contributes to the architectural research of blockchain applications and networks. This line of research started from the recognition of blockchain as a software connector type that bridges software components across organisations in a decentralised and trustless manner [7, 50]. Recent research on the software architecture aspects of blockchains has focused on architectural tactics [20] and patterns [52] to design on-blockchain smart contracts and the software systems surrounding them.

Whilst blockchain networks are also software systems, research on blockchain network architectures have received less focus. Xu et al. [13] proposed a taxonomy of blockchain networks to support practitioners in choosing a blockchain for their use cases. Tran et al. [2] was amongst the first to investigate the possible design decisions in establishing a blockchain network. Recent research has identified design patterns of blockchain networks for edge computing use cases, highlighting the prevalence of multi-channel heterogeneous blockchain networks [2].

This paper advances the research on blockchain network architecture by proposing a meta-model for capturing blockchain network architectures as computer-understandable artefacts. This meta-model is based on the design decisions identified by Tran et al. [2]. Future research, such as formal verification and recommendation of blockchain network architecture, can leverage the proposed meta-model.

9.3. Blockchain Network Evaluation

Blockchain network evaluation can be classified into benchmark and simulation. Benchmarks assess a blockchain network’s quality after deployment by subjecting the network to standardised workloads and measuring quality attributes of interest. Benchmarks can be conducted manually as demonstrated on the Quorum [53] and Red Belly [54], and Fabric [55] blockchain platforms. As proprietary benchmarks can be difficult to compare, researchers and industry have developed blockchain benchmark suites. BLOCK BENCH [20] is among the earliest benchmark automation suites. It provides workloads for benchmarking both end-to-end performance and individual layers such as consensus and processing. Pongnumkui et al. [56] extended the idea of BLOCK BENCH but emphasised the impact of workload size on the computing layer of blockchain nodes. Zheng et al. [57] proposed an alternative monitoring framework that analyses the logs generated by Ethereum clients to infer their performance in real-time. Beyond performance, researchers have also applied benchmarks to assess the correlation between CPU consumption and blockchain’s rewards [58], and bandwidth requirement for real-time use cases [59]. A survey of blockchain benchmark suites is available in [60].

Simulations assess the qualities of a blockchain network before deployment. They are helpful in scenarios where deploying a blockchain network prototype is impractical. The existing simulators [21, 22, 23, 24] vary broadly in simulation parameters and predicted indicators. However, their inputs are primarily reflections of the architectures of blockchain networks that they need to simulate.

NVAL does not offer new simulators or benchmarks. However, it offers a platform for integrating and employing simulators and benchmark suites. Specifically, simulators and benchmark suites can be encapsulated as composite actions and integrated into NVAL’s action pool. The bridging modules of these actions can transform architectural specifications (BND fragments) into the input expected by the enclosed simulators or benchmark suites.

10. Conclusion

This paper introduces NVAL, a software framework that automatically deploys and evaluates blockchain networks based on architectural specifications. This framework enables practitioners and researchers to focus on designing
blockchain networks and reduces the time and effort consumed by implementation mistakes and misconfigurations. NVAL operates by breaking down complex multi-channel blockchain networks into logical channels and leveraging the existing automation programs contributed by practitioners to deploy and evaluate these channels individually. We propose a novel meta-model to capture blockchain network architectures, a novel concept and architecture to capture automation programs, and a state-space search approach to match architectures with actions and construct execution plans automatically. A real-world case study with 65 blockchain networks featuring 12 diverse architectures shows that NVAL can accurately deploy and evaluate blockchain networks with negligible time overheads.

We plan to extend NVAL by introducing semantic descriptions to actions and BND models to enable more sophisticated semantic-based matching between actions and architectures. We also aim to explore AI-based techniques to build actions automatically from the existing open-source deployment scripts and natural language instructions. These solutions aim to further improve the longevity and effectiveness of NVAL as an all-in-one solution for researchers and practitioners to deploy and evaluate complex blockchain networks.

Acknowledgements

We would like to thank Chadni, Fattah, and Shaw from the Center for Research on Engineering Software Technologies (CREST) as well as anonymous reviewers for the insightful and constructive comments.

References

[1] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Chrisidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, et al., Hyperledger fabric: a distributed operating system for permissioned blockchains, in: Proceedings of the thirteenth EuroSys conference, 2018, pp. 1–15.

[2] N. K. Tran, M. A. Babar, Taxonomy of edge blockchain network designs, in: Software Architecture, Springer International Publishing, 2021, pp. 172–180. doi:10.1007/978-3-030-86044-8_10

[3] G. Wood, et al., Ethereum: A secure decentralised generalised transaction ledger, Ethereum project yellow paper 151 (2014) 2014–1–32.

[4] S. Nakamoto, Bitcoin whitepaper, URL: https://bitcoin.org/bitcoin.pdf (17.07.2019).

[5] C. Cachin, M. Vukolić, Blockchain consensus protocols in the wild, arXiv preprint arXiv:1707.01873.

[6] V. Buterin, The meaning of decentralization, Vitalik Buterin’s Blog, URL: https://medium.com/vitalikbuterin/the-meaning-of-decentralization-c0c92b76d274

[7] X. Xu, C. Pautasso, L. Zhu, V. Gramoli, A. Ponomarev, A. B. Tran, S. Chen, The blockchain as a software connector, in: 2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA), IEEE, 2016, pp. 182–191.

[8] M. Swan, Blockchain: Blueprint for a new economy, ” O’Reilly Media, Inc”, 2015.

[9] L. Kugler, Non-fungible tokens and the future of art, Communications of the ACM 64 (9) (2021) 19–20.

[10] E. Avgouleas, A. Kiayias, The architecture of decentralised finance platforms: A new open finance paradigm, SSRN Electronic Journal doi:10.2139/ssrn.3666029

[11] N. K. Tran, M. A. Babar, J. Boan, Integrating blockchain and internet of things systems: A systematic review on objectives and designs, Journal of Network and Computer Applications 173 (2021) 102844. doi:10.1016/j.jnca.2020.102844

[12] L. Pawczuk, J. Hollósky, R. Massey, B. Hansen, Deloitte’s 2020 global blockchain survey from promise to reality, Deloitte Insights (2020) 1–44.

[13] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, P. Rimba, A taxonomy of blockchain-based systems for architecture design, in: 2017 IEEE International Conference on Software Architecture (ICSA), IEEE, 2017. doi:10.1109/icsa.2017.33

[14] R. G. Brown, The corda platform: An introduction, Retrieved 27 (2018) 2018.

[15] N. K. Tran, M. A. Babar, Anatomy, concept, and design space of blockchain networks, in: 2020 IEEE International Conference on Software Architecture (ICSA), IEEE, 2020. doi:10.1109/icsa67534.2020.00020

[16] X. Liang, J. Zhao, S. Shetty, D. Li, Towards data assurance and resilience in IoT using blockchain, in: MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM), IEEE, 2017. doi:10.1109/milcom.2017.8170858

[17] A. Ramachandran, M. Kantarcioglu, SmartProvenance, in: Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy, ACM, 2018. doi:10.1145/3176258.3176333

[18] E. Bandara, D. Tosh, P. Foytik, S. Shetty, N. Ranasinghe, K. D. Zoysa, Tikiri—a lightweight blockchain for IoT, Future Generation Computer Systems 119 (2021) 154–165. doi:10.1016/j.future.2021.02.006

[19] W. Zheng, Z. Zheng, X. Chen, K. Dai, P. Li, R. Chen, NutBaAS: A blockchain-as-a-service platform, IEEE Access 7 (2019) 134422–134433. doi:10.1109/access.2019.2841905

[20] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, K.-L. Tan, BLOCKBENCH, in: Proceedings of the 2017 ACM International Conference on Management of Data, ACM, 2017. doi:10.1145/3035918.3064033

[21] R. Yasaweerasinghe, M. Staples, I. Weber, Predicting latency of blockchain-based systems using architectural modelling and simulation, in: 2017 IEEE International Conference on Software Architecture (ICSA), IEEE, 2017. doi:10.1109/icsa.2017.22

[22] B. Wang, S. Chen, L. Yao, B. Liu, X. Xu, L. Zhu, A simulation approach for studying behavior and quality of blockchain networks, in: Lecture Notes in Computer Science, Springer International Publishing, 2018, pp. 18–31. doi:10.1007/978-3-319-49478-42

[23] H. Zhang, C. Jin, H. Cui, A method to predict the performance and storage of executing contract for ethereum consortium-blockchain, in: Lecture Notes in Computer Science, Springer International Publishing, 2018, pp. 63–74. doi:10.1007/978-3-319-49478-45

[24] Y. Aoki, K. Otsuki, T. Kanoel, R. Banno, K. Shudo, SimBlock: A blockchain network simulator, in: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE, 2019. doi:10.1109/infocomw.2019.9085283

[25] L. Bass, P. Clements, R. Kazman, Software architecture in practice, Addison-Wesley Professional, 2003.

[26] Q. Lu, X. Xu, Y. Liu, I. Weber, L. Zhu, W. Zhang, uBaaS: A unified blockchain as a service platform, Future Generation Computer Systems 101 (2019) 564–575. doi:10.1016/j.future.2019.06.051

[27] A. A. Malik, D. K. Tosh, U. Ghosh, Non-intrusive deployment of blockchain in establishing cyber-infrastructure for smart city, in: 2019 16th Annual IEEE International Conference on Sensing, Communications, and Networking (SECON), IEEE, 2019. doi:10.1109/smecn.2019.8824921
[28] A. Bosu, A. Iqbal, R. Shahriyar, P. Chakraborty, Understanding the motivations, challenges and needs of blockchain software developers: a survey, Empirical Software Engineering 24 (4) (2019) 2636–2673. doi:10.1007/s10664-019-09708-7

[29] M. Wöhrer, U. Zilz, Devops for ethereum blockchain smart contracts, in: 2021 IEEE International Conference on Blockchain (Blockchain), 2021, pp. 244–251. doi:10.1109/Blockchain53865.2021.00040

[30] M. A. Walker, A. Dubey, A. Laska, D. C. Schmidt, PlaT-BART, in: Proceedings of the 4th Workshop on Middleware and Applications for the Internet of Things - M4IoT textquotesingle17, ACM Press, 2017. doi:10.1145/3152141.3152392

[31] Hyperledger, Hyperledger composer, https://github.com/hyperledger/composer (2019).

[32] MixBytes, Mixbytes tank, https://github.com/mixbytes/tank (2021).

[33] Hyperledger, Hyperledger bevel, https://github.com/hyperledger/bevel (2021).

[34] A. Das, G. Uddin, G. Ruhe, An empirical study of blockchain repositories in github (2021).

[35] G. Greenspan, Private blockchains are more than "just" shared databases, https://www.multichain.com/blog/2015/10/private-blockchains-shared-databases/ (Oct. 2015).

[36] P.-L. Aublin, S. B. Mokhtar, V. Quéma, Rbt: Redundant byzantine fault tolerance, in: 2013 IEEE 33rd International Conference on Distributed Computing Systems, IEEE, 2013, pp. 297–306.

[37] M. Castro, B. Liskov, et al., Practical byzantine fault tolerance, in: OsDi, Vol. 99, 1999, pp. 173–186.

[38] P. B. Kruchten, The 4+ 1 view model of architecture, IEEE software 12 (6) (1995) 42–50.

[39] S. J. Russell, Artificial intelligence: a modern approach, Pearson Education, Inc., 2010.

[40] R. K. Yin, Case study research: Design and methods, Vol. 5, sage, 2009.

[41] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empirical Software Engineering 14 (2) (2009) 131–164. doi:10.1007/s10664-008-9102-8

[42] IBM, An architectural blueprint for autonomic computing, IBM White Paper 31 (2006) (2006) 1–6.

[43] S. Porru, A. Pinna, M. Marchesi, R. Tonelli, Blockchain-oriented software engineering: Challenges and new directions, in: 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C), IEEE, 2017. doi:10.1109/icse-c.2017.142

[44] P. Chakraborty, R. Shahriyar, A. Iqbal, A. Bosu, Understanding the software development practices of blockchain projects, in: Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ACM, 2018. doi:10.1145/3239235.3240298

[45] M. Ortu, M. Orru, G. Destefanis, On comparing software quality metrics of traditional vs blockchain-oriented software: An empirical study, in: 2019 IEEE International Workshop on Blockchain Oriented Software Engineering (ICSE-B), IEEE, 2019. doi:10.1109/ICSE-B.2019.00076

[46] T. Gorski, Towards continuous deployment for blockchain, Applied Sciences 11 (24). doi:10.3390/app112411745 URL https://www.mdpi.com/2076-3417/11/24/11745

[47] MADT, Madt, https://github.com/dltcspbu/madt (2020).

[48] G. Li, Y. Dong, J. Li, X. Song, Strategy for dynamic blockchain construction and transmission in novel edge computing networks, Future Generation Computer Systems 130 (2022) 19–32. doi:https://doi.org/10.1016/j.future.2021.12.005 URL https://www.sciencedirect.com/science/article/pii/S0167739X21006856

[49] Z. Cai, G. Yang, S. Xu, C. Zang, J. Chen, P. Hang, B. Yang, Rbaas: A robust blockchain as a service paradigm in cloud-edge collaborative environments, IEEE Access 10 (2022) 35437–35444. doi:10.1109/ACCESS.2022.3161744

[50] X. Xu, I. Weber, M. Staples, Architecture for Blockchain Applications, Springer International Publishing, 2019. doi:10.1007/978-3-030-03035-3

[51] F. Wessling, V. Gruhn, Engineering software architectures of blockchain-oriented applications, in: 2018 IEEE International Conference on Software Architecture Companion (ICSA-C), IEEE, 2018. doi:10.1109/icsa-c.2018.00019

[52] X. Xu, C. Paulasso, L. Zhu, Q. Lu, I. Weber, A pattern collection of blockchain-based applications, in: Proceedings of the 23rd European Conference on Pattern Languages of Programs, ACM, 2018. doi:10.1145/3282308.3282312

[53] A. Baliga, I. Subhod, P. Kamat, S. Chatterjee, Performance evaluation of the quorum blockchain platform, arXiv preprint arXiv:1809.03421.

[54] T. Crain, C. Natoli, Y. Gramoli, Evaluating the red belly blockchain, arXiv preprint arXiv:1812.11747.

[55] C. Wickboldt, Benchmarking a blockchain-based certification storage system, Tech. rep. (2019).

[56] S. Pongnumkul, C. Siripanpornsana, S. Thajchayapong, Performance analysis of private blockchain platforms in varying workloads, in: 2017 26th International Conference on Computer Communication and Networks (ICCCN), IEEE, 2017, pp. 1–6.

[57] P. Zheng, Z. Zheng, X. Luo, X. Chen, X. Liu, A detailed and real-time performance monitoring framework for blockchain systems, in: 2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP), IEEE, 2018, pp. 134–143.

[58] A. Aldweesh, M. Alharby, E. Salaiman, A. van Moorssel, Performance benchmarking of smart contracts to assess miner incentives in ethereum, in: 2018 14th European Dependable Computing Conference (EDCC), IEEE, 2018, pp. 144–149.

[59] A. Meeuw, S. Schopfer, F. Wortmann, Experimental bandwidth benchmarking for p2p markets in blockchain managed microgrids, Energy Procedia 159 (2019) 370–375.

[60] R. Wang, K. Ye, C.-Z. Xu, Performance benchmarking and optimization for blockchain systems: A survey, in: International Conference on Blockchain, Springer, 2019, pp. 171–185.

[61] G. Li, Y. Dong, J. Li, X. Song, Strategy for dynamic blockchain construction and transmission in novel edge computing networks, Future Generation Computer Systems 130 (2022) 19–32. doi:https://doi.org/10.1016/j.future.2021.12.005 URL https://www.sciencedirect.com/science/article/pii/S0167739X21006856