Blockchain Based Variability Management Solutions for Fog Native Open Source Software

Bjoern Holste
Blockchain Technology Research
Technology Institute
Frankfurt, Germany
bjoern@btri.de

Vlado Stankovski
Faculty of Civil and Geodetic Engineering
University of Ljubljana
Ljubljana, Slovenia
vlado.stankovski@fgg.uni-lj.si

Petar Kochovski
Faculty of Computer and Information Science
University of Ljubljana
Ljubljana, Slovenia
petar.kochovski@fri.uni-lj.si

Antonio Puliafito
Department of Engineering
University of Messina
Messina, Italy
apuliafito@unime.it

Philippe Massonet
Scientific Coordination
CETIC Research Center
Charleroi, Belgium
philippe.massonet@cetic.be

Abstract

The complexity of Cloud-Fog-Edge computing systems is high due to the high variability which can be expected to grow further in the future with more IoT devices, computing architecture and specific requirements from security, regulation and users. One way to manage this complexity is to consider the Cloud-Fog-Edge Continuum as a single infrastructure of devices and components to be used in a seamless way to deliver new advanced services. Such a framework would integrate the handling of all connected devices, requirements and manage their variability. This paper proposes a framework called VarOps which provides an intuitive Workbench, enabling the design of software considering variability, making possible the re-use of dockerized components to significantly increase efficiency in development of new solutions. VarOps exploits Blockchain as a distributed ledger technology, variability formal modelling approaches, Smart Contracts, Trustful and Trustless Smart Oracles and knowledge management solutions as the underlying technologies and presents some example use cases.

Index Terms

Software development, variability management, fog computing, Blockchain, Internet-of-Things

I. Introduction

There is an increasing demand for new Fog Native software solutions, such as smart applications and environments. Application areas include smart homes, smart cities and communities, industry 4.0, well-being, tourism, and smart construction. Such applications and environments are connecting the Internet of Things (IoT) with a plethora of algorithms and methods for Big Data processing (volume, variety, velocity, veracity), including the application of Deep Learning and other AI Methods applied on sensor and video streams with remarkable results [1].

The DevOps movement is focused on open source software engineering, using open source components, such as Docker containers. Container images are stored in repositories as Docker Hub, from where they are pulled to compose smart component based applications. Advanced software engineering tools are used in the process, such as Juju or Fabric8, allowing the composition of applications from software components written in various languages. However, these tools do not manage the constraints among the software components when composing applications. Various software components are usually packed into executable images, i.e. container or VM images. When applied in the IoT/smart environments and Big Data domains, this process requires fine-grained selection and tuning of functionalities and the optimisation of communication/timing of the processes, and the connectivity between the executing Edge-Fog-Cloud computing tiers of the applications. The heterogeneity is vast, particularly when applications need to be reused, repurposed, customised and deployed for similar use cases.

Applications may contain functionalities implemented as individual container images like video recognition, encoding and decoding, rendering, coordination, logistics algorithms or optimisation algorithms. Only a high level of optimisation can satisfy detailed Non-functional Requirements (NFRs) at both design, deployment and runtime. Optimising code requires knowledge of the target heterogeneous multiprocessor platform characteristics, which is another important issue that demands substantial engineering time.

However, existing methodologies implemented by software engineering tools do not address the sheer complexity and interdependency of Functional and Non-functional Requirements (FRs, NFRs), and do not support engineers to arrive to a high quality of service (QoS), low-cost, secure, privacy preserving and dependable smart applications.

The authors' perceived gap is the absence of mechanisms

- to manage the complexity of various needed functionalities when developing applications for similar, but not identical purposes and deployment scenarios, and

- to address high-level dependability requirements [2], including reliability, security, privacy, trust, QoS and other NFRs, when developing, deploying and operating Fog Native smart applications.

Central to our approach is the concept of *Fog Native Open Source Software Components*. We define a software component as a software artefact, such as software package, library, operating system (OS) kernel, object file, which forms a part of a disk image (e.g., an executable file, VM or container image). Software components contain some key functionalities (such as Microservices), and can be dynamically generated from an Integrated Development Environment (IDE), managed, combined together into executable application programs or disk images and stored in Open Source repositories. At runtime, such binary executables and images can be delivered, deployed, executed, terminated, and otherwise managed and orchestrated across the Edge-to-Cloud computing continuum, that is, across multiple computing tiers (such as different, geographically distributed IaaS providers).

Hypothesis - With devices, software and hardware becoming increasingly more distributed, heterogeneous and complex, a new set of decentralised variability management solutions are necessary for dependable and trustful Fog Native computing, capable of supporting the DevOps culture and practice.

Because smart software systems are challenged by the growing complexity of many and diverging product variants, it has been necessary to model the variability of software systems [3] [4]. The formalisation of variability and its use for variability management, is proven to improve efficiency. The variability refers to the ability of a product to be configured, customized, extended, or changed for use in a specific context [5], whereas variability management is the set of activities aimed to cover the creation and support of different product variants [6]. Variability Management encompasses the activities of explicitly representing variability in products throughout their life cycle, managing dependencies among different variabilities, and supporting the instantiations of those variabilities by making variability decisions [7]. Variability management has successfully been used in the context of embedded software solutions. Using variability management in Product Line Engineering (PLE) has led to substantial measured improvements. As mentioned by the Software Engineering Institute at Carnegie Mellon University in a 2015 report[1], the use of variability management in PLE has the potential to improve productivity up to ten times, increase quality up to ten times, reduce the cost up to 60%, and reduce manual tasks up to 87%. To illustrate these improvements, many real-world examples have been published by private companies. For example, the U.S. Army expects to save $584 million in development costs by procuring a family of live training systems as a product line rather than a series of separated acquisitions. Also, MarketMaker Software ltd. reported that PLE led to a four times reduction in time to market and a reduction in maintenance costs around 60%[2].

This paper proposes a framework and Workbench labeled *VarOps* to support the *development of new formal models that both formalize the decentralised software variants* (e.g., microservices, VM/container images) and *enable the creation of Fog Native software products* (smart applications) by selecting valid combinations of variants. Variability management will be integrated in the Workbench and its underlying platform. It is expected that the Workbench can radically improve the lifecycle of Fog Native Open Source software and contribute to achieving the required NFRs. The technical goals of VarOps are illustrated in Fig. 1.

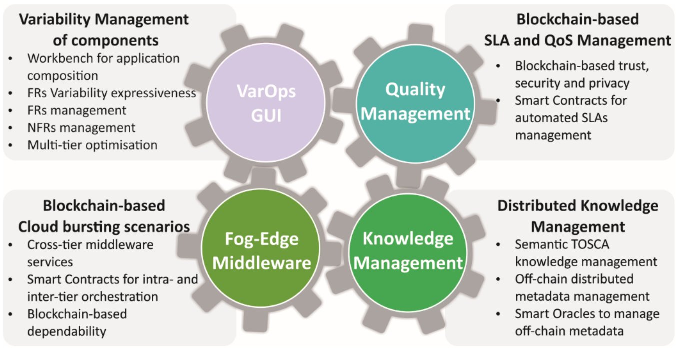
Variability Management of components
Workbench for application composition
FRs Variability expressiveness
FRs management
NFRs management
Multi-tier optimisation
VarOps GUI
Quality Management
Blockchain-based SLA and QoS Management
Blockchain-based trust, security and privacy
Smart Contracts for automated SLAs management
Blockchain-based Cloud bursting scenarios
Cross-tier middleware services
Smart Contracts for intra- and inter-tier orchestration
Blockchain-based dependability
Fog-Edge Middleware
Knowledge Management
Distributed Knowledge Management
Semantic TOSCA knowledge management
Off-chain distributed metadata management
Smart Oracles to manage off-chain metadata

Fig. 1. Solutions, models, Workbench and platform for decentralised variability management of Fog Native component-based applications which follow the DevOps culture and practice.

Because variability management solutions for the DevOps lifecycle are required, distributed ledgers and systems for management of distributed metadata, information and knowledge are a natural technology choice in the VarOps project. These form the basis for formal modelling [8], assurance, confidence, ranking and verification of decentralised variability solutions. Using Blockchain based variability management methods and solutions will enable the configuration during the development to be performed consistently and validly (e.g., taking into account constraints among variants), and automate the deployment of

[1]https://www.sei.cmu.edu/productlines/

[2]https://www.marketwatch.com/story/biglever-provides-advanced-product-line-engineering-capabilities-for-us-armys-consolidated-product-line-management-next-program-2015-05-11

such configurations. This reduces errors and configuration time compared to a manual approach. At the same time, Blockchain as a cryptographically sealed database provides high QoS level as it ensures that only valid variability configurations can be read from the Blockchain database and only authorized users can provide functional configurations.

II. Objectives of VarOps

Our expectation is that the proposed set of VarOps solutions will provide significant benefits to Fog Native computing, improving the overall software lifecycle. To provide guidelines in its development, we propose the following objectives:

Objective 1 - Develop *formal models and solutions* for decentralised variability management of Fog Native applications. The formal models and solutions will take into account the programmable, elastic, decentralised and virtualised environment, and the heterogeneity of the processing elements provided by data centres, ad hoc networking infrastructures, clouds, fog and edge providers and IoT devices. This new variability management approach (covering the entire lifecycle of multi-tier applications) has three important characteristics: 1) Full consideration of NFRs and FRs, 2) Customisable runtime environment encompassing Edge-to-Cloud tiers and 3) Automated Edge-Fog cloud bursting scenarios.

Objective 2 - Design a *DevOps lifecycle management environment* for Fog Native applications - a VarOps Workbench. This Workbench provides: 1) A frontend for user interaction throughout the VarOps lifecycle, 2) Intuitive interfaces for developers to design application logic, 3) Automated Service Level Agreements (SLAs) through the automated delivery of Smart Contract templates, deployment across administrative domains, and QoS monitoring, 4) Connectivity/networking management solutions for real time applications, 5) Visualisation, control and maintenance of production smart applications, and 6) Formal assurances and confidence parameters, ranking and verification methods for various stages of the applications' lifecycle.

Objective 3 - Design, develop and implement *trustful operation and monetization* possibilities for use in IoT, data, software, computing infrastructure and networks. Design and implement Blockchain based mechanisms for automated SLA management for smart application resources via the use of Smart Contracts and Smart Oracles that will 1) automate the resources negotiations, acquisition and application deployment process [9], and 2) improve the quality and trust in Fog Native software executing across different administrative domains.

Objective 4 - Design a *decentralised knowledge management technology* for semantic OASIS TOSCA[3] and the variability models, metadata and information. The VarOps Workbench will be complemented with a comprehensive knowledge management system, which will complement the formal variability models from Objective 1. The provided metadata will be used by trustless and trustful Smart Oracles to facilitate the automated operation of Smart Contracts in a trusted way.

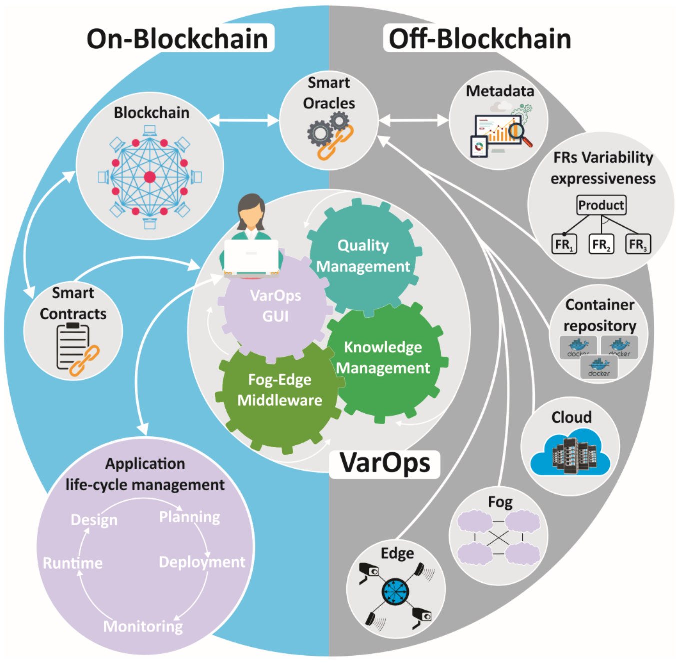

Fig. 2. VarOps runtime support for smart applications.

Objective 5 - Develop *Blockchain enabled Edge-Fog middleware technologies* for cloud bursting scenarios relying on Smart Contracts. It will use decentralised variability management models and semantically enhanced OASIS TOSCA descriptors for the management of QoS in the application runtime. It will be developed according to interoperability principles. Efforts will

[3]OASIS Topology and Orchestration Specification for Cloud Applications

be made to align with OpenFog[4] and CNCF[5] proposed open protocols and interfaces. This solution will include a scalable system for the monitoring of QoS to trigger Edge-to-Cloud bursting (auto-scaling) scenarios [10].

The VarOps Workbench methodology and tools will test and validate the developed solutions through a set of comprehensive smart pilot applications and scenarios (see Fig. 2). VarOps performance will be tested with a set of integrated Key Performance Indicators (KPIs) that are presented in Table I. The KPIs are designed to ensure that VarOps stays on track to reach its objectives. The KPIs will be compared against data from prior use productivity analysis of similar approaches. Furthermore the productivity increase from the BC implementation will be evaluated from SC's execution time and gas price.

TABLE I
KEY PERFORMANCE INDICATORS FOR VAROPS

Suggestion	**Target Measure**
Variability management modelling techniques and mechanisms to assist developers during the composition of functionalities in a Workbench, and to deploy a valid configuration of variants in Fog Native Software in a trustworthy way.	500% productivity increase by faster discovery and reuse of similar functionalities stored in container and VM images. Reduction of the complexity during the composition of functionalities. Dependable and trusted distributed architecture for software resources.
TOSCA based annotations for open source software components stored in container repositories	100% improved software component discovery, reuse and repurposing of software components. 100% improved utility of existing repositories.
Edge-Fog cloud bursting mechanisms via the use of trustful and trustless Smart Oracles, automated Smart Contract templates and Blockchain supported execution for trust and dependability.	Automated resource acquisition process and provide high degree of dependability and trust, not only to software, but also for the use of infrastructures, data and services.
Smart application configuration method based on selecting software components instead of copying and modifying code.	Reduction of development costs, at least 50%; and 400% increase in productivity and quality during the development of new smart application configurations.
Variability mechanisms for managing QoS and NFR constraints.	400% increase in QoS of new Fog Native application configurations.
Variability mechanisms for supporting efficiently a range of different smart application variants.	Reduction of manual tasks, at least 60%; and 100% valid selection of application variants of Fog Native Software.

III. CONCEPT AND REQUIREMENTS

A. The VarOps Concept

It is this paper's fundamental motivation to address FRs and NFRs in a balanced manner, thereby supporting Fog Native application design, development and deployment under inherent variability managed with the help of Blockchain technology. Following is an account of the ideas, the architecture and design considerations for the VarOps approach.

In a nutshell, VarOps will deliver a Workbench focused on the needs of the DevOps culture and practice. The Workbench will enable developers to specify and modify their FRs and NFRs, define their variability ranges, combine functionalities, as well as deploy, monitor and maintain them across the Edge-to-Fog computing continuum. The variability management approach will allow for discovery and use of various application components across distributed and decentralised repositories of VM and container images.

When building smart applications with the Workbench, it will be possible to record information related to the lifecycle of Fog Native software resources, so that various resources can be reused, repurposed and redeployed from one smart application to another, from one smart environment to another, from one Edge/Fog/Cloud provider to another. The VarOps Workbench will support the semantic annotation of resources used in the process, and their reuse in different contexts. The applications FRs and NFRs, including QoS hard and soft constraints, and various runtime rules would be included in a semantic specification of OASIS TOSCA prepared for use in a decentralised context and stored on a Blockchain.

A Blockchain-based practice for automation of the process, will be adopted throughout the platform. In addition, authentication and authorisation of software components (before running them) via existing community driven Blockchain

[4]https://www.openfogconsortium.org/
[5]https://www.cncf.io/

ledgers (such as Ethereum, Cardano, EOS, Insolar, BigchainDB or Tezos) is a feature, thus providing a "bullet proof" security and trust mechanism, where every building block and its QoS requirements must be ultimately trusted before a Smart Contract is concluded, automatically executed, and the smart application is deployed [11]. Smart Oracles, a novel approach, will provide for an ability to automatically generate Smart Contract templates that can be used in various runtime variability management scenarios, such as Edge-to-Fog bursting scenarios.

The VarOps architecture will provide the required back-end for the reliability, security, trust, charging and accounting functions of the variability-enabled services. In particular, the current trend in Blockchain technology suggests the usage of tokens both at the building block and at user levels to enable: 1) the provision of secured and accountable catalogues of variability-enabled modules, enabling suitable matching mechanisms to request, assign, deploy and dispatch modules, and 2) transactions for easy charging and deployment of services to the platform customers. These features will be key in the implementation of Smart Contracts for SLAs for the deployment of flexible and reconfigurable services without compromising security.

It is known that public permissionless blockchains, e.g., Ethereum, present scalability issues [12] that can prevent their use in VarOps scenarios, in which the number of users and the number of transactions required can be much higher than the ones supported by the blockchain network. Moreover, state-of-the-art smart contract programming languages are Turing complete but usually do not have useful constructs that facilitate the implementation of complex programming logic [13]. The Tendermint framework [14] allows to build consortium (permissioned) blockchains that are scalable based on a modified version of the Practical Byzantine Fault Tolerance (PBFT) consensus protocol. However, there is a trade-off between scalability and security. But if the number of nodes is high enough, the level of trust can acceptable without compromising scalability. Moreover, Tendermint allows for deterministic state machine applications in any programming language so already existing frameworks for the implementation of complex models can be used as they are.

B. Requirements for the development of the VarOps Platform

Several requirements for the development of the VarOps platform can be drawn from the above considerations:

1) Selection of suitable Blockchain technology with ability for permissioned architecture and API for seamless integration with the Workbench. This could be established with Ethereum or Tendermint-based BigchainDB. Final Evaluation is a part of the project's implementation.
2) Addressing multiple functional requirements: All applications need to manage variability of FRs for a new Fog deployment scenario.
3) Decentralisation, needs for multi-tier Edge-to-Cloud application design, deployable across different administrative domains. Reflects the need to automate the processes of data/software/infrastructure/network SLA management and the use for Blockchain to provide for variability management, dependability and trust.
4) System-level performance requirements: A certain performance must be achieved to deliver acceptable QoS Fog applications, and this requires significant variability management approaches, such as TOSCA model registration and middleware capable of providing low latency or high bandwidth (e.g., in Use Case 1), throughput and response time (e.g., in Use Case 2). Due to heterogeneity of the target infrastructures, every deployment configuration must be formally verified to assure smooth operation and high quality of experience.
5) Component (container, VM images) based application development: The Workbench must support the search, reuse, and repurposing of container/VM based software resources, and thus support various approaches to virtualisation.
6) Data intensive communication and I/O operations, the use of ad hoc networks: In IoT applications, there is also the need to stream data. High bandwidth communication channels must be dynamically negotiated and allocated to applications.
7) Verifiability: The VarOps Workbench must prove that the deployment configuration, and the Edge-Fog bursting scenario will achieve the desired QoS along with other NFRs. This will be achieved through the use of formal proofs [15].
8) Adaptability to changing infrastructure: Various dynamic Edge-to-Cloud offloading scenarios requiring the use of variability management principles will be made available.

These requirements cover the entire lifecycle of Fog Native applications. Such applications must be designed to run across the entire computing continuum, from the Edge to the Cloud.

C. Proposed Approach

The collected requirements motivate us to propose a variability management platform VarOps for Fog computing. VarOps provides novel support for defining, optimising and controlling requirements smart component based applications' life-cycle. Using a fully responsive Web based interface and back-end components, the VarOps Workbench can define dynamic application level mappings for the FRs and NFRs, rules and strategies to be employed on an application-by-application, component-by-component and tier-by-tier basis.

The VarOps Workbench will include variability management models and methods, particularly methods for variability specification, resolution, search, modification, annotation, maintenance and so on. Figure 3 illustrates the underlying VarOps design, which relies on knowledge management methods for the used building blocks.

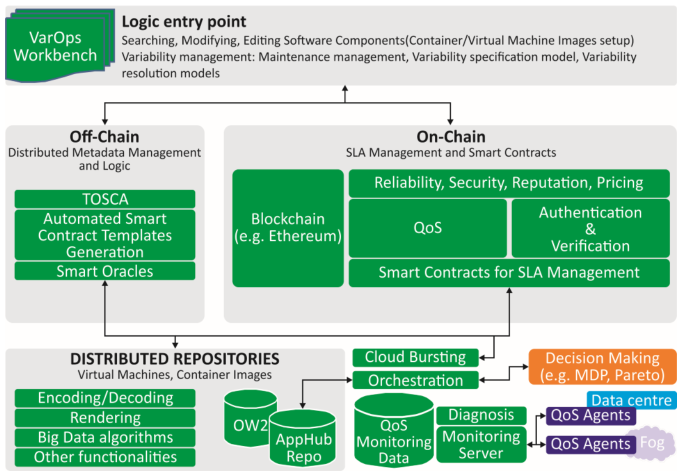
VarOps Workbench
Logic entry point
Searching, Modifying, Editing Software Components(Container/Virtual Machine Images setup)
Variability management: Maintenance management, Variability specification model, Variability resolution models
Off-Chain
Distributed Metadata Management and Logic
TOSCA
Automated Smart Contract Templates Generation
Smart Oracles
On-Chain
SLA Management and Smart Contracts
Blockchain (e.g. Ethereum)
Reliability, Security, Reputation, Pricing
QoS
Authentication & Verification
Smart Contracts for SLA Management
DISTRIBUTED REPOSITORIES
Virtual Machines, Container Images
Encoding/Decoding
Rendering
Big Data algorithms
Other functionalities
OW2
AppHub Repo
Cloud Bursting
Orchestration
Decision Making (e.g. MDP, Pareto)
QoS Monitoring Data
Diagnosis
Monitoring Server
Data centre
QoS Agents
QoS Agents
Fog

Fig. 3. Initial design of an integrated VarOps Workbench and the underlying Fog computing platform.

The Interactive Development Environment for Software Components (the IDE Workbench) provides an integrated environment for developers to create new Fog application configurations by selecting reusable components. Smart application configurations include similar (but different) software components and they are specified using Edge computing models (TOSCA models). The Workbench uses variability management techniques that specify the variable and reusable components of products as well as their constraints (e.g., if a component is mandatory or optional). The reusable components are shown in the Workbench as pieces that can be selected in a valid Fog application configuration by taking into account the formal variability specification. In addition to the reusable components, the Workbench shows the configurations that have been already prepared, which allows developers to check existing product configurations and to modify them. The variability management that Workbench includes aids the developer in creating new application configurations by reusing components and efficiently managing a range of possible configurations. Furthermore, the Workbench includes techniques for searching reusable components by providing a query in natural language as input. The Workbench exposes user interfaces for viewing and controlling the progress at runtime of the process parallelization, service deployment, and application execution.

The Smart Oracles and knowledge base (TOSCA and metadata management) provide formal reasoning and decision making mechanisms for the operation of Smart Contracts, including automated generation of Smart Contract templates. This addresses multiple NFRs, such as performance and availability.

The edge-fog cloud bursting subsystem provides Smart Contracts to horizontally scale the number of running container instances across administrative domains from Edge to the Fog. Apart from multi-level QoS monitoring services, it offers management services such as across-tiers orchestration, similar to capillary computing techniques.

The software repositories are an essential part of this loosely coupled decentralised VarOps architecture, facilitated via Blockchain based services, bearing in mind the hundreds of thousands of already existing software repositories, such as Docker Hubs.

IV. Illustrated use cases

Use case 1: Decentralized Smart Cameras Networks: Sensing, Processing, Implementation and Applications Decentralized smart cameras are real-time distributed embedded systems that perform computer vision using multiple observation points. They are networked to communicate and process collected data for applications such as surveillance, recognition, tracking, etc. They are becoming a fundamental part of our intelligent cities and homes, progressively gaining presence in our lives. This rapid development has been made possible due to several technologies. From advanced image sensors and video chips to embedded vision systems capable of efficient feature extraction, image encoding, video understanding and wireless transmission of the relevant visual content. Although decentralized camera networks have been rising in scale and adoption, effective methods for negotiation, and acquisition of cameras-as-sensors, as well as architecture-agnostic application deployment (and migration) processes are lacking. This is especially true when dealing with heterogeneous hardware, a common scenario for multi-tenant (e.g, Smart City) infrastructure, and with mobility requirements. The usage of VarOps, and in particular the Workbench, in such use cases can be exemplified in the following steps:

- register new (smart) edge nodes, e.g., IP- or GPU-enabled cameras, or actuator-bearing embedded systems, together with their privacy preferences, via Blockchain;
- search (e.g., proximal) nodes, e.g., networked smart cameras, by querying the Blockchain;
- register scene analysis (e.g., object recognition/classification/tracking) models, such as pre-trained deep neural nets, via Blockchain;

- search for scene analysis models, e.g., privacy-preserving, video streams processing by querying the Blockchain;
- once a set of available nodes and useful models is identified, corresponding Smart Contracts (e.g., tuned to the aforementioned privacy preferences) are made to assemble the (e.g., wide-area) target infrastructure and instantiate the SLA, and use the latter to wire up the (e.g., distributed tracking) application on top; and
- store (modified or new) analysis models, and their compositions, as artefacts/componentry in Blockchain-powered (e.g., auditable) repositories.

Use case 2: Sensor Based Cost Management on Blockchain The general idea is to create a system for automated cost allocation, to create accruals and automatically authorize the release of funds specifically tailored for a multitude of IoT device connections through a Blockchain. The location, type and number of sensors can change with every start of the application so that a high level of adaptability through very strong connection security and VarOps platform based variability management is required. This enables the user to add IoT devices to managed cost-spaces and impose rules for crowd-based accounting. Different machines and sensors can call these reserved funds to balance cost accounts. All data handling and transfer is done through the Blockchain and enables monetization of provided infrastructure for the owners of IoT devices. The VarOps platform is used to integrate smart contract SLA matching and variability management for the sensor IoT network. VarOps can provide a platform to configure flexible IoT networks that allow for exact cost allocation based on smart contract determined SLAs.

V. Summary and Outlook

In this paper we present VarOps, a framework to enable application developers to focus on functionalities that can be reused in multiple frameworks, thus generating substantial productivity gains, while simultaneously reducing complexity in management and maintenance. Some aspects of VarOps are aligned with related efforts in the US, such as at RDI2 (Rutgers University) on the need for a "Computing Continuum". The US vision identifies the need to create a nimble and programmable ecosystem that can adapt with changes in the underlying computational infrastructure and application requirements. The proposed VarOps framework will make a leap step towards achieving this vision by leveraging resources and automating Fog Native software specifications and software services/smart applications delivery at the logical extreme of the network, while unlocking potential for trading and fine grained monetisation of resources and services. This represents a comprehensive methodology supported by a set of new decentralised variability concepts, models, methods, services and tools. VarOps has the potential to deliver a trusted ecosystem of computational resources such as data, software components, repositories of components, IoT devices, computational infrastructures and networks, which can be used directly from the VarOps Workbench in a reliable, secure and dependable way by means of automated multi-party Smart Contracts serving a variety of business models, such as Subscription, Outcome Based, Asset-Sharing, Multi-Party Service Provisioning, "Razor Blade", (Raw) IoT Data Monetisation, Pay-Per-Usage and similar models.

References

[1] B. Holste and N. Schoeber, "Real-economy applications of blockchain technology," *Available at SSRN 3261612*, 2018.

[2] P. Kochovski and V. Stankovski, "Dependability of container-based data-centric systems," in *Security and Resilience in Intelligent Data-Centric Systems and Communication Networks*. Elsevier, 2018, pp. 7–27.

[3] M. Sinnema and S. Deelstra, "Classifying variability modeling techniques," *Information and Software Technology*, vol. 49, no. 7, pp. 717–739, 2007.

[4] S. Apel, F. Janda, S. Trujillo, and C. Kästner, "Model superimposition in software product lines," in *International Conference on Theory and Practice of Model Transformations*. Springer, 2009, pp. 4–19.

[5] L. Chen, M. Ali Babar, and N. Ali, "Variability management in software product lines: a systematic review," in *Proceedings of the 13th International Software Product Line Conference*. Carnegie Mellon University, 2009, pp. 81–90.

[6] M. Aiello, P. Bulanov, and H. Groefsema, "Requirements and tools for variability management," in *2010 IEEE 34th Annual Computer Software and Applications Conference Workshops*. IEEE, 2010, pp. 245–250.

[7] K. Schmid and I. John, "A customizable approach to full lifecycle variability management," *Science of Computer Programming*, vol. 53, no. 3, pp. 259–284, 2004.

[8] P. Kochovski, P. D. Drobintsev, and V. Stankovski, "Formal quality of service assurances, ranking and verification of cloud deployment options with a probabilistic model checking method," *Information and Software Technology*, vol. 109, pp. 14–25, 2019.

[9] P. Kochovski, S. Gec, V. Stankovski, M. Bajec, and P. D. Drobintsev, "Trust management in a blockchain based fog computing platform with trustless smart oracles," *Future Generation Computer Systems*, vol. 101, pp. 747–759, 2019.

[10] S. Taherizadeh, V. Stankovski, and M. Grobelnik, "A capillary computing architecture for dynamic internet of things: Orchestration of microservices from edge devices to fog and cloud providers," *Sensors*, vol. 18, no. 9, p. 2938, 2018.

[11] P. Kochovski and V. Stankovski, "Supporting smart construction with dependable edge computing infrastructures and applications," *Automation in Construction*, vol. 85, pp. 182–192, 2018.

[12] M. V. A. Chauhan, O. P. Malviya and T. S. Mor, "Blockchain and scalability," in *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. IEEE, 2017, pp. 122–128.

[13] M. Bartoletti and L. Pompianu, "An empirical analysis of smart contracts: platforms, applications, and design patterns," in *International Conference on Financial Cryptography and Data Security*. Springer, 2017, pp. 494–509.

[14] "Tendermint," in *https://tendermint.com/*, 2018.

[15] P. Kochovski, P. D. Drobintsev, and V. Stankovski, "Formal quality of service assurances, ranking and verification of cloud deployment options with a probabilistic model checking method," *Information and Software Technology*, vol. 109, pp. 14–25, 2019.