

Towards Web3 Applications: Easing the Access and Transition

Guangsheng Yu
Data61, CSIRO
Sydney, NSW, Australia
Saber.Yu@data61.csiro.au

Xu Wang
GBDTC, UTS
Sydney, NSW, Australia
Xu.Wang-1@uts.edu.au

Qin Wang
Data61, CSIRO
Sydney, NSW, Australia
qinwangtech@gmail.com

Tingting Bi
Data61, CSIRO
Melbourne, VIC, Australia
Tingting.Bi@data61.csiro.au

Yifei Dong
GBDTC, UTS
Sydney, NSW, Australia
Yifei.Dong@uts.edu.au

Ren Ping Liu
GBDTC, UTS
Sydney, NSW, Australia
RenPing.Liu@uts.edu.au

Nektarios Georgalas
Applied Research, British Telecom
Martlesham, Woodbridge, UK
nektarios.georgalas@bt.com

Andrew Reeves
Applied Research, British Telecom
Martlesham, Woodbridge, UK
andrew.reeves@bt.com

ABSTRACT

Web3 is leading a wave of the next generation of web services that even many Web2 applications are keen to ride. However, the lack of Web3 background for Web2 developers hinders easy and effective access and transition. On the other hand, Web3 applications desire for encouragement and advertisement from conventional Web2 companies and projects due to their low market shares. In this paper, we propose a seamless transition framework that transits Web2 to Web3, named WEBTTCOM¹, after exploring the connotation of Web3 and the key differences between Web2 and Web3 applications. We also provide a full-stack implementation as a use case to support the proposed framework, followed by interviews with five participants that show four positive and one natural response. We confirm that the proposed framework WEBTTCOM addresses the defined research question, and the implementation well satisfies the framework WEBTTCOM in terms of strong *necessity*, *usability*, and *completeness* based on the interview results.

KEYWORDS

Web3, Web2, Dapp, Blockchain, Software Engineering

1 INTRODUCTION

Web3 has drawn intensive attention from communities and investors. As an umbrella term, Web3 covers a series of blockchain-based decentralized applications (Dapps), services and economics [1–5] that bring significant impacts on both traditional finance and cryptocurrency markets. To date (as of Sep, 2022), over 12, 143 Dapps² have been developed on-chain and 285, 152 smart contracts are deployed across 48 protocols. A total of 1.67M users are actively interacting with the smart contracts within 24h, as evidenced by their wallet addresses. In this sense, Web3 impresses users by providing such a *connect the wallet* button on the upper-right corner of each webpage. Users can use Dapps through embedded wallet entries by invoking specific functions that are deployed on blockchain-engined platforms (e.g., Ethereum [6]). The shift of back-end servers

from centralized clouds to decentralized chains has mostly distinguished Web3 and previous web styles.

Following its narrative connotation, we observe that Web3 users still occupy a pretty small percentage (around 0.03% of 4.95B³ Internet users) over the Internet. The constraints majorly come from its incompatibility: (i) a typical Web3 application cannot be smoothly applied in a traditional web context due to the absence of a blockchain engine; conversely, (ii) a traditional Web2 application can hardly be integrated with blockchain due to the lack of proper Application Programming Interfaces (APIs). Rational developers would start their work on wide-adoption applications, namely Web2 Apps, for higher user exposure and more potential revenues, rather than sparing efforts on Web3 applications that are full of uncertainty. Such concerns motivate this work:

How to ease the usage of Web3 applications and achieve a smooth transition of applications between the Web2 and Web3 space?

We investigate the bottlenecks of transition between Web2 and Web3 by digging into their distinctions in design and implementations. For most Web2 applications, user management, data manipulation, and private-preserving policies are constructed upon centralized databases in a closed manner. In contrast, Web3 Dapps usually apply asymmetric-encryption-based identity to establish more secure and robust user management in a decentralized manner among many parties without prior trustworthiness. Data manipulation differs from that of Web2 applications due to the immutability of data storage in Web3, and access control also requires new approaches to partition the visibility. In addition, existing Web3 Dapps lack flexible mechanisms to smooth the workflow of documenting Restful APIs and test-suites during the development phase. Therefore, the incompatible user management strategies, execution procedures for data manipulation, private-preserving policies, and the lack of Web3-compatible API tools have greatly retarded the smooth transition applications in different domains.

To fill the above gaps, in this work, we focus on proposing a practical solution to seamlessly integrate both Web3 and Web2 applications and related services. We deconstruct the Web2 architecture

¹WEBTTCOM stands for Web2 (Two)–Web3 (Three) Communicator.

²Data source: Dappradar <https://dappradar.com/industry-overview>.

³Data source: Digital 2022: Global Overview Report <https://datareportal.com/reports/digital-2022-global-overview-report>.

and extract three major components covering frontend APIs, backend servers, and supplementary databases. Aligning with the core principles of Web3, we accordingly modify these components to enable seamless integration with blockchain engines. Specifically, we design three types of adjustable components: a SaaS module for integrating Web2 software by providing generalized APIs, a backend interpreter that interprets and forwards requests from Web2 to Web3, a blockchain layer that configures self-governed policies via smart contracts as well as computes on-chain calculations through chain Software Development Kits (SDKs). Our proposed solutions can greatly promote the transition from classic Web2 applications to the Web3 space, without redundant development or complicated middleware. In short, we highlight the *contributions* as follows.

- We explore the connotation of Web3 by investigating plenty of in-the-wild Web3 projects. As a new concept, we compare existing Web2 solutions and so-called Web3 projects to figure out the root features of Web3 applications and their dependencies, extracting their differences compared with classic Web2 applications.
- Based on comprehensive analysis, we propose a seamless transition framework that transits Web2 to Web3, named WEBTTCOM. Our proposed solution establishes an interpreter to bridge the Web2 applications and the Web3 backend engines. In particular, the proposed framework WEBTTCOM can offer effective and reliable access control and user management across the decentralized Web3 and centralized Web2, and provide an approach to conduct transition with existing popular SaaS, and the framework. While operating, WEBTTCOM can also effectively improve production by automatically generating API documents for developers and communities.
- We provide a full-stack implementation ranged from the frontend and backend APIs, the structure design and smart contract programming, to the on-cloud dockerized deployment. Code size reaches up to 11,351 lines⁴. The system is applied to daily service management processes, and promotes the establishment of effective, flexible, reliable, and trustworthy Web3-driven applications. Notably, our prototype has been being inspected by the British Telecom (BT) in practice.
- We further conduct an evaluation from the developers' perspective by holding interviews with five skilled developers. The interview feedback shows that the research question is well satisfied by the proposed framework WEBTTCOM, and the presented full-stack implementation proves its strong capability of smoothing the transition in term of *necessity, usability, and completeness*.

The remainder of this paper is organized as follows. Section 2 provides the Web3 basics. Section 3 proposes the research methodology and presents our research question. Section 4 introduces a use case that implements the new pattern, followed by discussion about the implementation notices and limitations shown in Section 5. Section 6 provides existing studies related to this work. Section 7 concludes this paper and highlights our contributions.

⁴Specifically, we present the detailed size distribution. The frontend takes 4 MB with image resources, while 253 KB with code only (5367 lines). The backend is a 1.3 MB project and 249 KB for codes (4831 lines). Web3 takes 265KB (1153 lines).

2 APPROACHING WEB3: A PRELIMINARY

In this section, we show the Web3 basics by making a comparison with Web2 and providing a typical Web3 instance.

2.1 Differences between Web 1/2 and Web3

Traditional Internets, including so-called Web1 and Web2, have been developed for decades. Web1 is regarded as a suite of *read-only* protocols that contain static sites to present images, text, and videos. Users search for the targets by accessing web portals. Web2 changes the way of interaction by enabling user-generated content (UGC). Users can publish their original content, such as images, reviews, testimonials, or even podcasts on social media websites (e.g., Facebook, Twitter). In this sense, Web2 is regarded as *read-write*. Web3 differs from previous styles by adding features of *ownership* and *transfer*. Users will create self-controlled accounts, generally in the forms of *wallets*, to manage digital assets and virtual data. Rather than relying on centralized servers, Web3 users can freely transfer their assets under the governance of smart contracts, which brings the advantages of auto-execution, being accountable, and being globally verified. These smart contracts connect both upper-layer DApps and underlying blockchain platforms.

Table 1: Comparisons among Web1/Web2/Web3 and Our Work

	Functions	Architecture	Instance
Web1	read	client-server	Yahoo
Web2	read/write	client-server	Facebook, Google
Web3	read/write/own/transfer	client-SC-chain	Ethereum, BSC
Web2→3	read/write/own/transfer	client-Int.-SC-chain	WEBTTCOM

2.2 Typical Web3 Architecture

Tradition web architecture is based on a client-server model. The client is used for sending and receiving requests, while the server is used to process these requests and corresponding logic. The server side is also known as the backend, covering many fundamental aspects like operating systems (Windows, Linux), platforms (.Net, LAMP), and storage. API is to connect the application tier to servers. In contrast, the Web3 architecture replaces centralized backend servers with distributed ledgers. The backend contains two sectors, smart contracts (SC) for defining logic and rules, and blockchain platforms for processing transactions and achieving consensus. Web3 is more complex than traditional Web2 due to its complete decentralization which requires dealing with the consistency problem [7]. In this work, we aim to ease the transition between the Web2 application tier and blockchain-backend systems. We establish an interpreter (short for Int.) to seamless connect them.

3 RESEARCH DESIGN

In this section, we conduct an exploratory study [8], which proposes a new Web3 driven framework, named WEBTTCOM, for implementing applications; and we define the research question (RQ) for identifying and extracting the evidence (e.g., the benefits of Web3 applications) for evaluating our proposed Web3 framework.

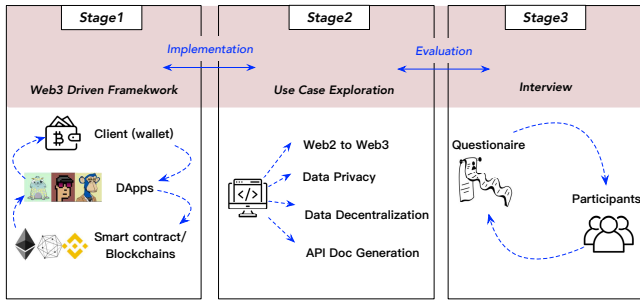


Figure 1: Overview of the study design

3.1 Research Question

This study aims to design and analyze whether our Web3 framework is effective and practical in terms of the quality attributes (i.e., smooth transition) and development productivity. Such characterization of Web3 will shed light on future Web3 applications and developments. Specifically, this work aims to address the Research Question (RQ) as:

Given the proposed Web3 framework (i.e., WEBTTCOM), is it effective, regarding the transition of Web2 to Web3 application development?

- **Web2 to Web3 Transition:** Given that many developers do not have relevant development background for Web3 applications, is the proposed framework practical and helpful for them to transit the Web2 to Web3 applications for meeting and implementing the specific requirements?
- **Data privacy and Governance:** In the evolution from Web2 to Web3, various issues related to data privacy and security, data privacy and governance in Web3 is one of the most discussed challenges for different stakeholders. As such, to address this concern, our RQ can explore whether the proposed Web3 framework (i.e., WEBTTCOM) ensures data privacy and governance.
- **Development Productivity:** If the proposed Web3 framework impacts developers’ productivity? For example, (1) practitioners’ daily development tasks and documentation. (2) problem-solving for Web3 application development.

3.2 Study Design Process

Our research methodology consists of three stages, as depicted in Fig. 1. In the first stage, we proposed a Web3 transition framework (i.e., WEBTTCOM), which includes an interpreter to help developers transit and bridge the Web2-based to Web3-based applications. We implemented a full stack project based on the proposed WEBTTCOM in the second stage. In the last stage, to evaluate the effectiveness of the proposed WEBTTCOM and the feasibility of the full-stack project, we interviewed five developers for their feedback and opinions.

Phase 1: Web3 Driven Framework. We proposed a Web3 framework (WEBTTCOM), which allows developers to implement effective and reliable applications. The framework guarantees:

- **Web2 to Web3 smooth transition:** our framework provides a trustworthy transition.

- **Blockchain backend:** the framework ensures the decentralization of Web3-driven applications.

The details of the framework are described in Section 4.1.

Phase 2: Web3 Application Implementation. In this phase, we implemented a full-stack application, which is based on the Web3 framework that we proposed. The details of the application are described in Section 4.2.

Phase 3: Interviews. As depicted in Fig. 1, in this phase, we invited five developers to participate in comprehensive interviews to confirm the effectiveness of our proposed Web3-driven framework and implementation. Each lasted 30 minutes.

- We invited five developers, who work at BT, to attend our interviews. These interviews were anonymous.
- We designed an interview questionnaire, which consists of three main questions (see Section 4.3.1) to get developers’ opinions on the effectiveness of our proposed framework and the implementation that we developed.
- We applied qualitative data analysis and consistent comparison to summarize the statements from developers, in terms of the productivity of using our framework and application.

Each interview consists of three parts of questions:

- **Part 1:** We asked demographic questions, such as the interviewees’ experience in Web3 development and management.
- **Part 2:** We asked open-ended questions to understand their opinions on Web3 design and development in practice.
- **Part 3:** We prepared candidate topics by carefully reading the contents of representative textbooks. We picked a list of topics that had not been explicitly mentioned in the open discussion and asked the participants to discuss those topics further. At the end of each interview, we thanked the participant and briefly informed them what we plan to do with his/her response.

4 RESEARCH RESULT

In this section, we propose a new framework, named WEBTTCOM, which not only smooths the transition between Web2 and Web3 but also achieves high data privacy and data governance, and improves development productivity. A full-stack implementation is presented as a use case to support the framework. We also conduct 30-minute interviews where five developers were invited to confirm the effectiveness of the proposed framework and implementation.

4.1 A New Framework: WEBTTCOM

A new framework, named WEBTTCOM (Web2 to Web3 Communicator) is proposed as a result. WEBTTCOM provides trusted and reliable service management across multiple organizations. The structure of WEBTTCOM consists of a blockchain layer, a backend interpreter, and a Web2 Software as a Service (SaaS), as in Fig. 2.

The blockchain layer saves service management data and supportive data and controls access to the service data. To this end, the blockchain layer includes a smart contract for data provisioning, a private data repository for service data, and a public data repository for data access policies and data hash checksum. In the smart

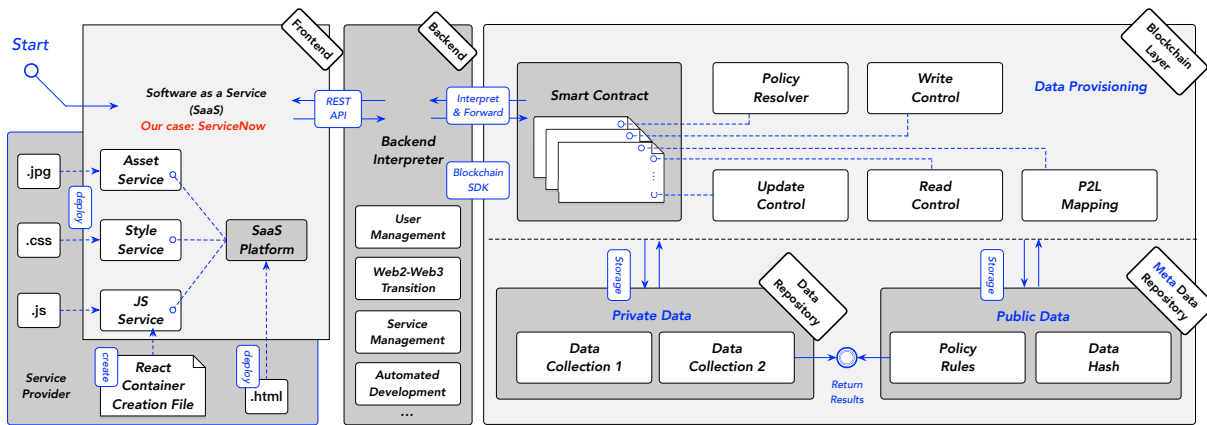


Figure 2: Overview of the instantiated service management platform.

contract, the policy resolver module extracts attributes of requests, such as requested data, request location, and user affiliation, and identifies effective access control policies. The write control module enforces data storage policies, such as data storage location and data expiration. The update control module takes charge of data updates, while the read control module enforces data access policies. The Physical-to-Logical (P2L) mapping module updates data presentation according to pre-defined physical data formats for data storage and logical data formats required by the SaaS.

The backend acts as a Web2-Web3 interpreter forwarding requests from the Web2 SaaS to the Web3 smart contract and formatting Web3 results for Web2 SaaS, as in the Web2-Web3 transition module. The backend interpreter meanwhile manages Web2 and Web3 users for the transition. On the Web2 side, the backend interpreter provides REST API [9] to the SaaS. The backend interpreter employs automated development technology for the automatic route and document generation. The backend interpreter communicates with the blockchain via the blockchain SDK.

The Web2 SaaS provides a service management interface in ServiceNow, enabling users to submit, view and process service tickets after logging in to the platform. The service management is realized in the form of ServiceNow React Container [10] so that the Web3-enhanced service management can be seamlessly integrated with popular service management platforms, i.e., ServiceNow in our case.

4.2 Implementing WEBTTCOM: A Use Case in a Service Management System

An implementation of the proposed framework WEBTTCOM is discussed in which a trusted and reliable service management across the University of Technology Sydney (UTS) and BT is established. The HyperLedger Fabric blockchain [11] is set up with one ordering service⁵ and two organizations (representing UTS and BT, respectively) in the same channel⁶ for the Web3 service. Each of the ordering service and the organizations has one Certificate Authority (CA) node and one peer node. Service ticket data are recorded

⁵The ordering service validates transactions and assembles valid transactions into ordered blocks.

⁶A Fabric channel is a private sub-network for permitted members.

in the Hyperledger Fabric in real-time and are certified by the data hash. With Web3 technology, the two organizations have a consistent view of service data. The platform is developed in TypeScript 4.4 using VSCode 1.69.1 and runs on the elastic servers of Amazon Web Service (AWS). MySQL 5.7 is selected to handle the Web2-based registries whereas the Web3-based registries rely on the native CouchDB of HyperLedger Fabric 1.4.

The implementation design of a service management system, as a use case of the proposed WEBTTCOM, is illustrated in Fig. 3 using the class diagram. *Blockchain* maintains a blockchain network with several registered *SmartContract*. *SmartContract* is inherited by three classes, i.e., *Web3UserRegistry*, *Web3ObjectRegistry*, and *Web3PolicyRegistry*. The *Web3UserRegistry* registers the user Web3 information in Hyperledger Fabric CAs and interacts with the corresponding customizable offchain-stored *Web2UserRegistry* in which the local user management is conducted. All objects including the product inventory, product order, service inventory, and trouble ticket service are onchain-stored in *Web3ObjectRegistry* with dedicated policies of access control stored in *Web3PolicyRegistry*. All registries are composed of *Record* in which the user-defined data record attributes are stored. Accessing the attribute values in *Record* relies on *Web3Config* in which the settings of *Blockchain* are defined. *AutoOpenAPIGenerator* auto-generates the OpenAPI standard [12] documentation for modules including *Web2UserRegistry*, *Web3UserRegistry*, *Web3ObjectRegistry*, and *Web3PolicyRegistry*.

4.2.1 Attribute-based Access Control in Web3. In the Hyperledger Fabric, each of the two organizations has one private data collection [13], which excludes the other organization, such that private business data are only saved and managed by the permitted organization. All blockchain members can access public onchain data. Web3 users are registered to CAs with their attributes, including organizations, departments and user types. The attributes are used for the attribute-based access control to the service data. Each Web3 user has a unique private key to sign Web3 transactions and access Web3 services.

Smart Contract: A service smart contract is developed to manage service tickets. The exposed entry points of the smart contract

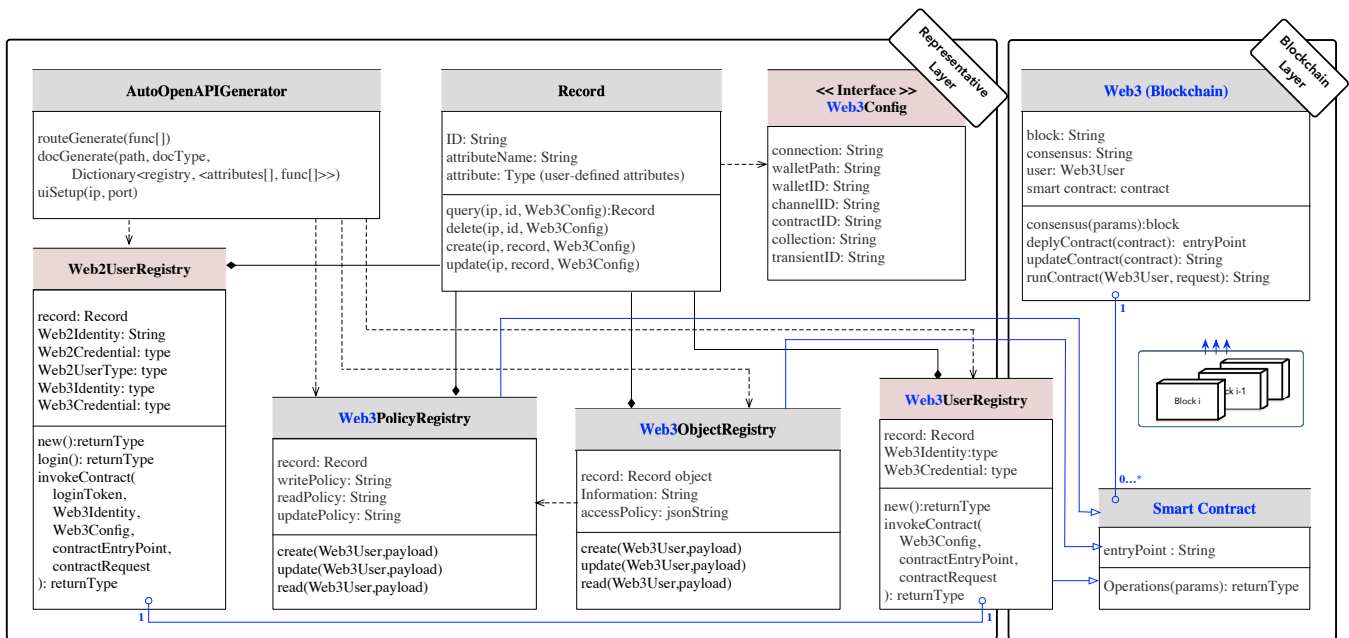


Figure 3: The class diagram of the instantiated service management platform.

are shown in Listing 1. The smart contract extends the Fabric *Contract* class and exports functions as APIs.

In the smart contract, create/update/read functions are developed for *Web3PolicyRegistry* and *Web3ObjectRegistry*, as shown in Fig. 3. *Web3PolicyRegistry* is implemented with the Fabric public data scheme where all blockchain peers have a copy of the public data, while *Web3ObjectRegistry* is realized with the Fabric private data scheme where only permitted peers have data copies. Variable *ctx* collects the context of the transaction calling the smart contract, including user attributes and timestamp for access control and transient service data⁷. Variable *attributes* provides Web2 attributes for access control, such as geolocation information. Variable *publicPayload* gives the data requested by the smart contract and recorded on the Fabric blockchain, such as access control policies.

Access Control: The smart contract enables attribute-based access control (ABAC) where access control policies are designed based on user attributes and environment attributes and are organized as the *Web3PolicyRegistry*. A policy can be either *writePolicy*, *readPolicy*, or *updatePolicy*, and each of them has unique fields describing access control requirements. All the policy types also have general metadata, including an updated time, an effective time and a priority level, for policy conflict resolver. All policies are created by organization admins and saved in the public data repository, such that policies can be accessed by all Web3 users and applied to all private data collections.

The policy resolving process is shown in Listing 2. The policy resolver first extracts the Web3 user attributes and environment attributes of the transaction, such as transaction timestamp, from the

ctx variable. Next, the policy resolver module retrieves all related policies according to the policy IDs embedded in the request data. Then, the policy resolver identifies effective policies by checking Web3 user roles, effective timestamps, policy priority, and geolocations. At last, the policy resolver returns effective policies for the request.

The write control, update control and read control modules then process the request according to the request data in the request transactions and effective policies. The request is rejected if no effective policy is returned from the policy resolver module. For a data write/update request, the write control and update control modules firstly identify the right private data collection from effective *writePolicy* and *updatePolicy*. The modules also check request attributes against the requirements specified in the effective policies and terminate the request for failed checks. The modules then extract transactional data and transient data from the request transaction and create a Web3 object with object information and related access policy IDs. Then, the modules can write/update the Web3 object to the data collection. For a data read request, the read control module gets the private data collection from the request transaction and reads the requested data out following the effective *readPolicy*. The read control module also implements the P2L mapping on the data keys following the mapping rules stated in effective *readPolicy*.

4.2.2 Web2-Web3 Transition. Each of organizations has an interpreter to convert Web2 requests to Web3 requests and forwards requests to specific Web3 nodes according to request attributes for Web3 processing. The interpreter then monitors Web3 processing results and creates returns for Web2 requests.

⁷Transient data is a type of data for Fabric private data collection, which can be read by smart contracts but does not be recorded in on-chain transactions.

Web2 APIs: The interpreter provides REST APIs supporting create, read, update operations in Web2 applications and implements Bearer Authentication scheme [14].

A Web2 request includes payloads describing create, read, and update operations on Web3 objects and policies. Besides that, the Web2 request contains a bearer token for authentication by the interpreter. The bearer token also allows the interpreter to identify the corresponding Web3 identity and private key for signing Web3 transactions. The Web2 request also gives details for the corresponding Web3 request, including the channel name, smart contract name and targeted private data collection, such that the interpreter can create correct Web3 request transactions and send them to the right Web3 node for processing.

Web3 Interaction: Interpreters manage user identities and interact with the service smart contract using the Hyperledger Fabric SDK [15], as shown in Listing 3. The channel and service smart contract are specified by variables *channelName* and *scName* in lines 10-11.

For every Web3 request, the interpreter needs to create a request transaction and sign the transactions with a valid Web3 user identity. To this end, the interpreter saves the Web3 user identities of all Web2 users in the organization, i.e., *walletPath* and *walletID* in Listing 3, and maintains the mapping between Web2 and Web3 identities. The interpreter is an organization admin user who can register and revoke users. During the user registration process, a user provides user information and credentials to the interpreter like general Web2 user registration. The interpreter firstly creates a new user record on itself and then sends the registration information to the organization CA node to create a Web3 identity. Then, the interpreter keeps the user's Web3 wallet including a private key and records the mapping between the Web2 identity and Web3 identity.

The interpreter needs to connect specific Web3 nodes that have requested private data collections. To this end, the interpreter maintains connection profiles⁸ to Web3 nodes from different organizations and records the mapping between private collections and the hosting Web3 nodes. When the interpreter receives a Web2 request, the interpreter identifies the data collection related to the request. The interpreter forwards the request to any Web3 node if the requested data is public. If the request points to some private data collection, the interpreter firstly identifies the hosting Web3 node and then forwards the request to the Web3 node using pre-configured connection profiles.

4.2.3 Transition with SaaS: ServiceNow-based Web Application. ServiceNow is a popular cloud-based workflow automation platform for enterprises by streamlining and automating routine work tasks. In many cases, companies are seeking a simple solution for connecting their existing ServiceNow applications to Web3 systems. As a result, we chose ServiceNow as our frontend system.

We intend to solve the following problems with our scheme:

- ServiceNow's existing system could be transformed in an efficient manner.
- Programmers' coding can be made easier by following a few simple steps.

⁸A connection profile describes the Web3 node to be connected, such as ID and IP/port.

In order to resolve these issues, we implement a ServiceNow container concept as shown in the left part of Fig. 2. We develop a web application using React and embedded it in ServiceNow. By using REST APIs, the embedded application can smoothly communicate with the backend and the interpreter.

The web application was developed using React, an open-source JavaScript library for creating user interfaces. There is a large community of programmers who use React. Therefore, most programmers are familiar with it and can easily obtain assistance from other members of the community.

A modern website deployment method - Webpack - is used for the deployment of this web application. Instead of deploying all resource files directly to the web server, Webpack binds multiple files into a single deployment file. Multiple HTML files are combined into one HTML file, and multiple Javascript files are combined into one Javascript file. This deployment method simplifies the resource access path, which is exactly what we require.

- The deployment utilizes two ServiceNow concepts: the UI page and the web service.
- The ServiceNow UI page allows developers to customize web pages using HTML or XML. In this case, we create one UI Page and insert the combined HTML code using Webpack.

The ServiceNow web services provide access to resources through REST APIs. Three web services have been developed: the Asset Service, the Style Service, and the JS Service. All images are accessible through the asset service. The style service provides all CSS access. The JS service provides access to JavaScript code.

There are three functions of the web application: user login, policy management, and ticket management. As part of implementing these functions, the embedded application communicates with the interpreter using HTTP REST APIs.

4.2.4 Backend and Automated Development Tools. Express framework is used to offer a flexible Node.js web application with MySQL being used for *Web2UserRegistry* and *Web3Config*. Sequelize is used to provide effective connections to the Web2 local registry in the manner of Object-relational Mapper (ORM). The platform is docker-based by default, and docker-compose is used to offer an orchestration service for distributed and flexible docker containers.

The system also offers automated generation and documentation of OpenAPI-compliant REST APIs [12] for *Web2UserRegistry*, *Web3UserRegistry*, *Web3ObjectRegistry*, and *Web3PolicyRegistry* via TSOA 3.11 [16]. The TSOA framework integrates OpenAPI compiler to construct Node.js serve-side application in type-safe TypeScript at runtime, as TypeScript is used for the programming of smart contracts at the Web3 side and backend/frontend at the Web2 side. The function, *routeGenerate*, can easily generate the API routes automatically with no pain based on the definition of controllers of all registries above; see the lower half of each registry in Fig. 3. At the same time, the YAML-based OpenAPI documentation and test suite are also generated to ease API testing and smooth the development of transition via *docGenerate*.

4.3 Interview Results

We conducted a set of interviews to assess the *Necessity*, *Usability*, and *Completeness* of the proposed framework WEBTTCom and the implementation.

4.3.1 *Interview Questions.* The technical background and Web3 experience were asked, followed by the following questions from the domain experts to ensure the *Necessity* of the new proposed framework WEBTTCom:

- What do you think of the pros and cons of Web2 and Web3 by now under your background?
- What do you think of the necessity of a smooth transition between Web2 and Web3 (both Web2 to Web3 and Web3 to Web2) from your organizational and personal perspectives?
- What are the possible challenges during the transition do you think is required to be resolved immediately?

Further, we asked following questions for feedback on WEBTTCom and overall *Usability* and *Completeness* of the implementation.

- To what extent the Service Management System developed by UTS and BT, the implementation of the proposed framework WEBTTCom, matches the principles of WEBTTCom and address the challenges above?
- What are your suggestions to enhance the suitability of the new framework WEBTTCom and its implementation?

4.3.2 *Key Findings.* We summarized key findings from the interviews that can support the proposed framework and implementation as a use case. The overall feedback of four developers was positive with the fifth one being neutral. Key findings are as:

1) ***Necessity - Enable Guided and Structured Design:*** Building Web3 technology into Web2 systems can solve critical security and trust issues in Web2 systems, especially for businesses across multi-organizations. Web2 systems are centralized and can hardly achieve trust across organizations, while the inherent consensus mechanism of Web3 technology can provide verified single-ground truth and thus build trust across parties. The second interviewee stated “A transition from Web2 to Web3 will bring the data trustworthiness and cybersecurity guarantee from Web3 to Web2 systems.”

It is of importance to have a smooth transition between Web2 and Web3. This is because Web2 is a mature technology, but developing Web3 applications could be challenging. As stated by the fourth interviewee with limited Web3 knowledge “The modification of the existing Web2 system should not be difficult. In order to avoid overloading the programmers with Web3 knowledge, they should not learn too much.”, and the fifth interviewee stated “Web2 and Web3 should be able to coexist and interact smoothly.” Detailed challenges during the transition between Web2 and Web3 include

- The transition of two different technologies, as stated by the fourth interviewee “Due to the differences in concept, technology and tools between Web2 and Web3, it is difficult to integrate the Web2 system with the Web3 system.”
- New access control schemes in Web3. The first interviewee stated “Web3 is transparent, how to apply flexible and feasible access control becomes important for cases where data privacy is considered”, and the third interviewee stated “DLTs are a promising solution due to the ability of smart contracts

to ensure that the required country-specific data management policies are agreed and enforced.”

- Heterogeneous user management across Web2 and Web3. The first interviewee stated “Approach to apply the user management in a shared ledger may be a challenge as Web2 parties tend to have separate user management systems at local.”
- High development and transition cost. The first and the fifth interviewees stated “Easing the transition by using automated tools is normal in Web2 applications and is also essential during the transition between Web2 and Web3.” and “Challenges include lack of experience developers, lack of available development resources, tools.”

2) ***Usability - Flexible Access Control:*** The developed flexible access control can implement access control policies as designed. The service provider can define access control policies, including data storage policies, write policies and read policies. The smart contract on the Hyperledger Fabric enforces all the policies and controls access to onchain data. The developed access control mechanism has been confirmed by all interviewees. The second interviewee stated “The system can enforce all the expected data governance and access control policies.”, and the third interviewee stated “In evaluating the solution we have demonstrated that a DLT Hyperledger layer can meet the required success criteria related to cross-country access control and user-management; as well as, connecting to a traditional SaaS workflow management layer.”

3) ***Usability - Compatible User Management:*** The developed user management compatibly manages Web2 and Web3 users. When a user submits a Web2 registration request, both Web2 and Web3 accounts are simultaneously created and then managed by the developed framework. When users log in and submit requests, the framework can perform authentication with the users’ Web2 credentials and process Web3 requests and responses in the representation of the users. All interviewees are satisfied with the developed user management. The second and fourth interviewees stated “A user can use one identity to access Web2 and Web3 services.” and “Web2 programmers are not required to touch too much.”

4) ***Usability - Easy Integration with Existing Platforms and Services:*** The developed system seamlessly integrates with existing service management on ServiceNow. All operations are conducted in ServiceNow, including login, access control policy management and service ticket management. The workflow keeps the same with the Web2 version. Nevertheless, all data are securely saved on the Hyperledger rather than disconnected databases and can be verified with the Web3-certified data hash. As the second interviewee stated “Users can access Web3-certified data services from the SaaS. The complicated Web3 details are transparent to users.”

5) ***Usability - Highly Automated Development Tools:*** Highly automated development tools are implemented for automated generation and documentation of OpenAPI-compliant REST APIs of both Web2 and Web3 registries. This significantly improves the efficiency of programming by sharing the same development workflows and schema formats between Web2 and Web3. As the 1st, 2nd, and 4th interviewees gave positive feedback by stating “Automated development tools significantly reduce the programmers’ workload and has almost become a must-use tool during the current workflow.”

6) **Completeness - Sufficient Decentralization with Strong Data Privacy and Governance:** The interviewees satisfy the framework WEBTTCOM and the use case in regard to achieving sufficient decentralization level while ensuring strong data privacy and governance; as the first interviewee stated “*The framework appears to be covering most perspectives including the access control, data privacy, data governance, user management, connections to existing SaaS, and development productivity.*”

7) **Completeness - Limited Cases:** The fourth and fifth interviewees stated “*More commercial Web2 systems should be integrated with Web3.*” and “*Need to find suitable business use cases to demonstrate the benefit of Web3.*”. Three interviewees noted that the proposed framework WEBTTCOM requires more business use cases to support by potentially integrating the existing commercial Web2 projects. While we admit this limitation at the time of writing, this work originates from solving the transition issue from Web2 to Web3 which is come across by our business partner proceeding with their latest strategy. This work is new and lacks sufficient existing use cases. Therefore, this work focuses on demonstrating how WEBTTCOM solves the research question, and we postpone extending use cases to improvements and future works.

8) **Potential Improvements:** Four interviewees stated that the framework and the given use case appear to be restricted to permissioned blockchain platforms such as HyperLedger Fabric while applying the same framework on other popular platforms such as Ethereum also requires further testing. One interviewee mentioned that the attempt to integrate with ServiceNow is a good start, but the connection with the existing services and functionalities provided by Platform as a Service (PaaS) or even Infrastructure as a Service (IaaS), such as AWS and Azure, also requires further testing.

5 LIMITATIONS AND VALIDITY

We apply the guidelines [17] in to discuss key threats to the validity (*construct, internal,external*) of this work.

Construct validity reflects what extent the research questions and methodology are appropriately used in a study. A threat in Stage 3 (i.e., the interviews) is whether or not our interviewees are representative. To reduce this threat, we invited five practitioners who come from the company that is applying our Web3 implementation. In addition, the areas that the participants have worked on cover a wide range of domains (e.g., blockchain and AI applications). However, they may not be representative of all practitioners. To mitigate this potential bias, we have carefully chosen questions and topics. Our interview respondents completed the interviews based on their opinions and perception. It is possible that they conflate the skills that are very important and the skills that are very relevant to their projects or industrial contexts.

Internal validity focuses on factors that may influence the validity of the results. The main threat in our study is whether the study process we designed and the Web3 framework and application that we proposed can answer our research questions. It is also possible that we draw the wrong conclusions about respondents’ perceptions from their comments. To minimize this threat, we read transcripts many times and checked the interview results and the corresponding comments several times.

The selection of statements produced at the end of the interviews may not be comprehensive and may be biased to the background of experts—who may not be able to articulate their own opinions. To mitigate this bias, we have taken the following steps:

- Aside from asking direct questions about their opinions about Web3 and applications that we designed and implemented, we also asked them to discuss general topics that they had not explicitly mentioned. The topics were selected from Web3 textbooks and online resources; they include concepts, comprehension, programming language, requirements, design implementation, testing, and tool usage.
- Three authors have performed data analysis to cross-check their answers by using card sorting [18], and we carefully examined and only included relevant statements.

External validity concerns the generality of our study results to other settings. Our results and summaries are based on the Web3 framework and application and interview participants’ opinions instead of a rigorous analysis of the claims participants make. It is possible that the opinions in terms of Web3 framework and application differ from participant and participant. To improve the generalizability of our results, we interview five respondents. Still, our findings may not generalize or represent the perception of all software engineers. For example, the respondents are from one company. It would be interesting to perform another study to investigate more software engineers to perceive the benefits and limitations of the Web3 framework and application in the future.

6 RELATED WORK

In this section, we give a quick overview of the notion of Web3 and then introduce ways of building Dapps on top of blockchains.

Notion of Web3. The concept of Web3 was first proposed by Wood [19] with an initial discussion focusing on blockchain-based digital infrastructure. Later, Weyl et al. [1] illustrate the ways of building a decentralized society by exploring Web3-related applications, requirements, opportunities, and challenges. Wang et al. [3] provide the discussion between Web3 and blockchain from the perspective of architecture design. The work has identified a total of 12 types of designs according to the data workflow of access, computation and storage. A series of reports from Consensus [20] investigates the economic performances created by decentralized networks. Web3 economy covers many on-chain protocols such as stablecoins, borrowing, lending, and leverage. These protocols are built on top of smart contracts. Liu et al. [21] explore three types of infrastructural enablers, including the single smart-contract powered chain, federated contracts, and interoperable blockchain platforms. However, Web3 sofar still confronts high-level controversies in terms of its definition and application. We, in this paper, extend our exploration by complying with its core *decentralization* nature guaranteed by underlying blockchain services.

Constructing Web3 Applications. Traditional Web1 and Web2 applications rely on centralized backend servers for computations and storage. Web3 applications [3] replace these servers with decentralized blockchain platforms [6][22]. Building a Web3 application requires three phases: (i) *embedding the wallet for access,*

(ii) *connecting frontend with blockchain platforms* and (iii) *operating executions on-chain*. A wallet [23][24] is backed by locally running lightweight nodes and helps Web3 users to create on-chain accounts (in the form of *address* [25]) as their identities. A user can initiate a request by sending a transaction from wallets to the transaction pool. Smoothly processing the received requests from the frontend requires a suite of standard protocols, including token standards (e.g., EIP [26], BEP [27]) and unified APIs. All the compiled bytecodes will be executed on-chain by iterative state transitions. The consensus mechanism is critical in maintaining state consistency and chain stability [28]. In some cases, external techniques are needed for supportive functionalities such as distributed storage [29], layer-two computations [30], cross-chain bridges [31] or on-chain oracles [32]. Our work achieves more than building a simple Dapp that is independent of operating blockchains. We, instead, develop a general interpreter to connect Web2 applications to the current leading Web3 platforms from the access layer to the chain layer. This gives an educational study for the community.

7 CONCLUSION

In this paper, we gave a research question by exploring the connotation of Web3 and the key differences between Web2 and Web3 applications. We proposed a new framework, named WEBTTCom, to enable a seamless transition from Web2 to Web3. In particular, WebttCom can smoothly connect traditional Web2 applications to mainstream Web3 blockchain platforms while additionally ensuring high data privacy and governance, and improving developer productivity. Our design innovative introduces an interpreter mechanism that can aggregate and deals with requests between Web2 and Web3 spaces. Accordingly, we implement a full-stack system with 11.351+ lines of code and launch a survey for evaluating the effectiveness of our framework. Corresponding interviews with five experienced participants confirmed WEBTTCom satisfies our research question with a few possible limitations of the framework and its related business cases. We further provide our recommended improvements, including the extension of the framework to involve more blockchain platforms and search for more business cases.

REFERENCES

- [1] E Glen Weyl, Puja Ohlhaber, and Vitalik Buterin. Decentralized society: Finding web3's soul. Available at SSRN 4105763, 2022.
- [2] Xu Wang, Xuan Zha, Wei Ni, Ren Ping Liu, et al. Survey on blockchain for internet of things. *Computer Communications*, 136:10–29, 2019.
- [3] Qin Wang, Rujia Li, Qi Wang, Shiping Chen, et al. Exploring web3 from the view of blockchain. *arXiv preprint arXiv:2206.08821*, 2022.
- [4] Guangsheng Yu, Xuan Zha, Xu Wang, Wei Ni, Kan Yu, Ping Yu, J. Andrew Zhang, Ren Ping Liu, and Y. Jay Guo. Enabling attribute revocation for fine-grained access control in blockchain-iot systems. *IEEE Transactions on Engineering Management*, 67(4):1213–1230, 2020.
- [5] Guangsheng Yu, Litianyi Zhang, Xu Wang, Kan Yu, Wei Ni, J. Andrew Zhang, and Ren Ping Liu. A novel dual-blockchained structure for contract-theoretic lora-based information systems. *Information Processing & Management*, 58(3):102492, 2021.
- [6] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.
- [7] Xu Wang, Wei Ni, Xuan Zha, Guangsheng Yu, et al. Capacity analysis of public blockchain. *Computer Communications*, 177:112–124, 2021.
- [8] Steve Easterbrook, Janice Singer, et al. Selecting empirical methods for software engineering research. In *Guide to advanced empirical software engineering*, pages 285–311. Springer, 2008.

- [9] Shaohua Wang, Iman Keivanloo, and Ying Zou. How do developers react to restful api evolution? In *International Conference on Service-Oriented Computing*, pages 245–259. Springer, 2014.
- [10] Servicenow. <https://www.servicenow.com/>.
- [11] Elli Androulaki, Artem Barger, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference (EuroSys)*, pages 1–15, 2018.
- [12] OpenAPI. The openapi specification. <https://github.com/OAI/OpenAPI-Specification>, 2022.
- [13] Hyperledger Fabric. Private Data. <https://hyperledger-fabric.readthedocs.io/en/release-2.2/private>, 2022.
- [14] SmartBear. Bearer Authentication. <https://swagger.io/docs/specification/authentication/bearer-auth>, 2022.
- [15] Hyperledger Fabric. Hyperledger Fabric SDK for node.js. <https://hyperledger.github.io/fabric-sdk-node/release-1.4/index.html>, 2022.
- [16] Lukeautry. Openapi-compliant rest apis using typescript and node. <https://github.com/lukeautry/tsoa>, 2022.
- [17] Claes Wohlin, Per Runeson, et al. *Experimentation in software engineering*. Springer Science & Business Media, 2012.
- [18] Sally Fincher and Josh Tenenber. Making sense of card sorting data. *Expert Systems*, 22(3):89–93, 2005.
- [19] Gavin Wood. Why we need web 3.0. <https://gavofyork.medium.com/why-we-need-web-3-0-5da4f2bf95ab>, 2021.
- [20] Consensus. Web3 report q3. <https://consensus.net/reports/web3-report-q3-2021/>, 2021.
- [21] Zhuotao Liu, Yangxi Xiang, et al. Make web3.0 connected. *IEEE Transactions on Dependable and Secure Computing*, 2021.
- [22] Binance smart chain. Accessible at <https://www.bnbchain.org/en/smartChain>, 2022.
- [23] Panagiotis Chatzigiannis, Foteini Baldimtsi, and Konstantinos Chalkias. Sok: Blockchain light clients. 2022.
- [24] Kostis Karantias. Sok: A taxonomy of cryptocurrency wallets. *Cryptology ePrint Archive*, 2020.
- [25] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A Kroll, and Edward W Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy (SP)*, pages 104–121. IEEE, 2015.
- [26] Ethereum improvement proposals. Accessible at <https://eips.ethereum.org/>, 2022.
- [27] Instance: Bep-20. Accessible at <https://academy.binance.com/en/glossary/bep-20>, 2022.
- [28] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 281–310. Springer, 2015.
- [29] Juan Benet. Ipfns-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561*, 2014.
- [30] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. Sok: Layer-two blockchain protocols. In *International Conference on Financial Cryptography and Data Security (FC)*, pages 201–226. Springer, 2020.
- [31] Alexei Zamyatin et al. Sok: Communication across distributed ledgers. In *International Conference on Financial Cryptography and Data Security (FC)*, pages 3–36. Springer, 2021.
- [32] Lorenz Breidenbach, Christian Cachin, et al. Chainlink 2.0: Next steps in the evolution of decentralized oracle networks. *Chainlink Labs*, accessible at <https://naorib.ir/white-paper/chinlink-whitepaper.pdf>, 2021.

APPENDIX

The entrance of the implemented service management contract is shown in Listing 1. The service management contract extends the Hyperledger Fabric *Contract* class. Users can invoke the functions by sending transactions containing the function names and parameters. Users can *create*, *update*, and *readWithFilter* objects saved on the Hyperledger Fabric. Functions are implemented in controller files.

```

1 import { Context, Contract, Info, Returns, Transaction } from 'fabric-
  contract-api';
2 import { Controller } from 'web3-controller';
3
4 export class ServiceContract extends Contract {
5     @Transaction(true)
6     @Returns('string')

```

```

7   public async create(ctx: Context, attributes: string, publicPayload:
string): Promise<string> {
8       return await new Controller(<'policy'|'object'|'user'>).
createRegistry(ctx, ctx.stub.getTxID(), attributes, publicPayload
);
9   }
10  @Transaction(true)
11  @Returns('string')
12  public async update(ctx: Context, dataId: string, attributes: string
, publicPayload:string): Promise<string> {
13      return await new Controller(<'policy'|'object'|'user'>).
updateRegistry(ctx, dataId, attributes, publicPayload);
14  }
15  @Transaction(false)
16  @Returns('string')
17  public async readWithFilter(ctx: Context, publicPayload:string):
Promise<string> {
18      return await new Controller(<'policy'|'object'|'user'>).
readRegistryWithFilter(ctx, publicPayload);
19  }
20 }

```

Listing 1: Entry Point of the Ticket Management Contract.

The read and update policy checking is shown in Listing 2. The function checks all policies linked to the corresponding objects, which are given by the parameter *policyIDs*. The function fetches the policies with the *policyIDs*, checks *user*, *location*, *validTime*, *updateTime*, and *priority*, and returns the effective policy.

```

1   public async getActiveReadAndUpdatePolicy(ctx: Context, location:
string, policyIDs: string[]): Promise<any> {
2       let effectivePolicy: any = {};
3       let user = lla.getUserInfo(ctx);
4       let ts = ctx.stub.getTxTimestamp().getSeconds() * 1000;
5       let latestTimestamp = 0;
6       let highestPriority = 0;
7       for (let policyID of policyIDs){
8           let policyObj: any = await new pc.policy().loadPolicyFromID(
ctx, policyID);
9           let updateTime = await lla.getLastUpdateTime(ctx,policyID);
10          let permittedUser = new userInfo(policyObj.privilege.
rowPolicy.userAffiliation)
11          if (user.match(permittedUser)) {
12              if (location == policyObj.privilege.rowPolicy.location.
countryName) {
13                  let startStamp = new Date(policyObj.validFor.
startDateTime).valueOf();
14                  let endStamp = new Date(policyObj.validFor.
endDateTime).valueOf();
15                  if (ts >= startStamp && ts <= endStamp) {
16                      if (policyObj.priority > highestPriority) {
17                          effectivePolicy = policyObj;
18                          latestTimestamp=updateTime;
19                          highestPriority=policyObj.priority;
20                      }
21                      if (policyObj.priority == highestPriority) {
22                          if (updateTime > latestTimestamp) {
23                              effectivePolicy = policyObj;
24                              latestTimestamp=updateTime;
25                          }
26                      }
27                  }
28              }
29          }
30          return effectivePolicy;
31      }
32  }

```

Listing 2: Read and Update Policy Checking Function.

The Web2-Web3 interpreter is shown in Listing 3. When receiving a Web2 request, the interpreter creates a Hyperledger Fabric connection, including the connection profile, Fabric user, channel name, and contract name, according to the payload of the Web2 request. The interpreter then forwards the Web2 request and parameters to the corresponding functions in the Web3 contract. When receiving Web3 results, the interpreter returns them to the Web2 request.

```

1   export async function Trans(
2       _fabric: ITransient, context?: Required<Pick<IUserManagement, "
userID" | "userPWD">>, fnc: string, ...args: string[]
3   ): Promise<string> {
4       let promise: Promise<string> = new Promise(async (resolve, reject)
=> {

```

```

5       try {
6           const fabricConCof: FabricConfiguration = new
FabricConfiguration(
7               _fabric.connection, /* The Fabric connection profile */
8               _fabric.walletPath, /* The path to Web3 credentials */
9               _fabric.walletID, /* The Web3 user */
10              _fabric.channelName, /* The Fabric channel name */
11              _fabric.scName, /* The smart contract name */
12              _fabric.listen /* Fabric event listener*/
13          );
14          const [gateway, connectionProfile, connectionOptions] = await
getFabricConfig(context, _fabric);
15          await gateway.connect(connectionProfile, connectionOptions);
16          let network = gateway.getNetwork(fabricConCof.channelName);
17          let contract = (await network).getContract(fabricConCof.scName
);
18          const channel = (await network).getChannel();
19          let result: Buffer | any;
20          result = await contract.submitTransaction(fnc, ...args);
21          await gateway.disconnect();
22          if (result) { resolve(result.toString()); }
23          else { resolve("Invoking ${fnc} succeeds."); }
24          } catch (error) { reject(error); }
25      });
26      return promise;
27  }

```

Listing 3: Web2-Web3 Interpreter.

APPENDIX-B

First response

1. What is your job in the organization? Technical or non-technical?

I am a software developer in my organization.

2. What is your Web3 background?

I have 4-year developing experience on Web3 (expert)

3. What do you think of the pros and cons of Web 2 and Web3 at present?

Web 2:

Pros: mature, design pattern, tools, workflows have been thoroughly explored and developed, and have been proven applicable for many huge and complicated tasks.
Cons: heavily relies on trustworthy centralized entities. Building trust among parties heavily relies on social engineering, which often causes interests dispute.

Web3:

Pros: achieve trustworthy relationship among parties without centralized entities in a decentralized manner.
Cons: immature, lack design pattern, tools, workflows. Popularity is not as good as Web 2.

4. What do you think of the necessity of smooth transition between Web 2 and Web3 from your organization and your personal perspectives? Web 2 to Web3 and Web3 to Web 2?

It is of importance to have smooth transition between Web 2 and Web3.

Web 2 project sometimes not only needs to focus on the trust issue, but also need to have an efficient incentive

mechanism to encourage the public to use the services. And Web3 fits the requirement by applying the decentralized ledger technology.

Many Web3 applications have proven the potential in many areas including finance, asset trading, gaming, law... However, Web3 is still a new to the market, and needs to increase its popularity by having collaboration with those giant companies and their mature products.

5. What are the possible challenges during the transition do you think is required to be resolved?

- (1) Web3 is transparent, how to apply flexible and feasible access control becomes important for cases where data privacy is considered.
- (2) Approach to apply the user management in a shared ledger may be a challenge because in Web 2 parties tend to have separate user management systems at local.
- (3) How to connect with existing mature products, such as SaaS, may be a challenge.
- (4) Easing the transition by using automated tools is normal in Web 2 applications and is also essential during the transition between Web 2 and Web3. Having such tools that can also be seamlessly integrated in existing Web 2 and Web3 frameworks may be a challenge.

6. To what extent the Service Management System developed by UTS and BT UK matches the framework and solve the questions?

a. Access control

Satisfied, the data-driven policy is smart.

b. User management

Satisfied

c. Connection with SaaS

Satisfied with the combination with ServiceNow which has been widely used in many existing products.

d. Automated development tools

TSOA is an interesting tool as Typescript is very popular and OpenAPI standard has almost been a must-use tool during the current workflow in Web 2 development.

7. What are your suggestions to enhance the suitability of the new framework WEBTTCOM and its implementation?

The framework appears to be covering most perspectives including the access control, data privacy, data provenance, user management, connections to existing SaaS, and development productivity. And the implementation also shows a good match to the framework. One point that could be improved is the extension of

Web3 platforms where other platforms can be involved other than HyperLedger. Then any possible changes of the implementation to match the framework should be considered.

Second response

1. What is your job in the organization? Technical or non-technical?

I am a researcher and software developer in my organization.

2. What is your Web3 background?

I have 4-year research and development experience on Web3.

3. What do you think of the pros and cons of Web 2 and Web3 at present?

Web 2:

Pros: Web 2 technology has been developed for decades. There are many mature solutions for various requirements and development tools for rapid development and simple management. Consumers have been well educated on the Web 2 service pattern.

Cons: The biggest challenge of Web 2 systems is the trust issue. For a single Web 2 service, consumers have to trust Web 2 service providers, and the service providers need to trust developers, infrastructure providers, etc. When multiple Web 2 systems are connecting to each other, they need to trust the services and data from others. In the case of untrusted relations or inconsistent data, it is challenging to solve disputes and provide services as a whole.

Web3:

Pros: Web3 can provide single ground truth to all participants in a trustless way. Different parties do not have to build trust with each other.

Cons: Web3 is a new technology. The architecture and process of Web3 services are very different from those of Web 2. Deploying and maintaining Web3 infrastructure and applications could be very time-consuming and risky. Consumers need to be educated about changes from Web3.

4. What do you think of the necessity of smooth transition between Web 2 and Web3 from your organization and your personal perspectives? Web 2 to Web3 and Web3 to Web 2?

We are aware of the benefit of Web3 and have a plan to employ Web3 technology in our business to provide trusted services and reduce business loss.

A transition from Web 2 to Web3 will bring the data trustworthiness and cybersecurity guarantee from Web3 to Web 2 systems. The Web3 token mechanism can support incentive schemes in Web 2 applications.

The smooth transition from Web3 to Web 2 applications will promote Web3 technology and applications. There have been many Web3 applications, like tokens and contracts. However, Web3 applications are only popular among the Web3 community because the requirements and access to Web3 applications are very different to widely-used Web 2 applications and can be hard for general users. A Web3 to Web 2 transition will simplify access and encourage more users to try Web3 technology and applications.

5. What are the possible challenges during the transition do you think is required to be resolved?

- 1) How to seamless integrate Web3 services into existing Web 2 systems
- 2) How to implement data governance and privacy policies in Web3
- 3) How to manage Web 2 and Web3 user identities simultaneously
- 4) How to reduce integration and development cost
- 5) How to efficiently manage Web3 infrastructure
- 6) Can Web3 technology support large-scale applications

6. To what extent the Service Management System developed by UTS and BT UK matches the framework and solve the questions?

a. Access control

Satisfied. The system can enforce all the expected data governance and access control policies.

b. User management

Satisfied. A user can use one identity to access Web 2 and Web3 services.

c. Connection with SaaS

Satisfied. Users can access Web3-certified data services from the SaaS. The complicated Web3 details are transparent to users.

d. Automated development tools

Satisfied. The automatic document generation technology can save 20

7. What are your suggestions to enhance the suitability of the new framework WEBTTCOM and its implementation?

The developed WebttCom is a good start for transiting Web 2 to Web3. I suggest the team develop more service functions to the framework, such as managing ticket files across multiple departments and organisations, such that the system usability could be improved. The framework

can also be integrated with other Web 2 businesses, such as workflow management. Besides new functions, comprehensive trials need to be conducted to verify the stability and capacity.

Third response

Hence, there are two options for the paper: 1) The survey section could provide an aggregated view from the project team's view on how well the prototype addresses the needs of Web3, without attributing views to individual organisations. Or 2) On the BT side we could write a section for the evaluation part of the paper.

To give you a flavour of what a BT authored section would look like: It would explain that Web3 is not well defined at the moment and is currently often linked to ideological aims around openness and decentralisation, compared to the current internet which is seen, by the Web3 community, as dominated by "big tech". The Web3 concept is a very nascent area at the moment where it will be interesting to see how it evolves and fits with the current Web 2 world which we live in.

Web3 needs to answer a number of core questions related to related to complexity, scalability, cost; and, critically, also cost-effectiveness and accountability, which are often more important to commercial organisations than an adherence to decentralisation principles. However, the lack of well-formed answers to these high-level generic considerations should not prevent us from identifying use cases and prototyping solutions to further our knowledge of the opportunities.

Looking at the Web3 technology portfolio it already includes some useful enabling technologies, including distributed ledger technologies (DLTs) which, with their inherently decentralised architecture, can be useful for specific applications. On their own they are unlikely to form the universal technology base for the next generation of the Internet; however, there are real use cases which exist today where DLTs may have an important role to play. The work with UTS is a example of this and examines a real-world use case related to IT service management across different country boundaries where different data privacy regulations apply.

In this context, DLTs are a promising solution due to the ability of smart contracts to ensure that the required country-specific data management policies are agreed and enforced. For the current prototyping project, it was assumed that the transition to a DLT based solution will require interworking with the more traditionally architected solutions which are already in common usage in commercial ecosystems. In evaluating the solution we have demonstrated that a DLT hyperledger layer can meet

the required success criteria related to cross-country access control and user-management; as well as, connecting to a traditional SaaS workflow management layer.

In meeting these criteria the prototype has demonstrated sufficient promise that we are examining the next stage in research and development, which is to run test scenarios, utilising realistic commercial data flows, through the prototype. This measured exploration and transition to DLT technologies shows the path which we believe many established commercial organisations will follow as they gain confidence that the decentralised approach can meet the required business requirements. The approach of monitoring the evolving Web3 concepts, and engaging with the core technology developments, to experiment and build early insight on use cases for these emerging capabilities (e.g. DLT, smart contracts) is a pragmatic approach for ensuring that value can be gained.

Fourth response

1. What is your job in the organization? Technical or non-technical?

I am a software developer in my organization. Technical.

2. What is your Web3 background?

I have a half year of Web3 background. My Web3 knowledge is limited.

3. What do you think of the pros and cons of Web 2 and Web3 at present?

Web 2:

Pros: Web 2 has a large development community including programmers, tools, and solutions. Web 2 has been widely applied to many organizations and it has been verified as a stable technology.

Cons: In the case of multiple organizations, it is not used for cross-validation and trust in the scenario.

4. What do you think of the necessity of smooth transition between Web 2 and Web3 from your organization and your personal perspectives? Web 2 to Web3 and Web3 to Web 2?

Since most systems are based on Web 2, we must focus on the transition from Web 2 to Web3.

The modification of the existing Web 2 system should not be difficult.

In order to avoid overloading the programmers with Web3 knowledge, they should not learn too much.

It is also possible to apply the existing Web 2 technology to Web3-based systems.

It is not necessary for normal users to recognize the transition.

5. What are the possible challenges during the transition do you think is required to be resolved?

1) Because of their limited understanding of Web3, programmers are reluctant to transfer their existing Web 2 systems to Web3.

2) Due to the differences in concept, technology and tools between Web 2 and Web3, it is difficult to integrate the Web 2 system with the Web3 system.

6. To what extent the Service Management System developed by UTS and BT UK matches the framework and solve the questions?

a. Access control

Solved. Web3 implements access control, which means that Web 2 programmers are not required to touch too much.

b. User management

Solved. It's similar to the above. Web3 implements access control, which means that Web 2 programmers are not required to touch too much.

c. Connection with SaaS

Solved, the modification of the existing Web 2 system is not too extensive. It is also possible to apply the existing Web 2 development method and architecture to the transition.

d. Automated development tools

Solved, automated development tools significantly reduce the programmers' workload.

7. What are your suggestions to enhance the suitability of the new framework WEBTTCOM and its implementation?

Attempts to integrate the existing commercial Web 2 system's functionality with Web3.

More commercial Web 2 systems should be integrated with Web3.

Provides more Web3 services.

Fifth response

1. What is your job in the organization? Technical or non-technical?

I am a developer, technical

2. What is your Web3 background?

3 years of exposure to Web3 development

3. What do you think of the pros and cons of Web 2 and Web3 at present?

Web 2: pros – mature technology, plenty of resources available; cons – being a centralised system, trust can be an issue
Web3: pros – has built-in trust mechanisms; cons – new tech, not much resources for developers

4. What do you think of the necessity of smooth transition between Web 2 and Web3 from your organization and your personal perspectives? Web 2 to Web3 and Web3 to Web 2?

Certain businesses require the trusted services provided by web3, but not all. Many existing services are fine with Web 2. Only services that deal with multi-organisations require web3. Web 2 and web3 should be able to coexist and interact smoothly.

5. What are the possible challenges during the transition do you think is required to be resolved?

Challenges include lack of experience developers, lack of available development resources, tools.

6. To what extent the Service Management System developed by UTS and BT UK matches the framework and solve the questions?

a. Access control

Completed.

b. User management

Completed.

c. Connection with SaaS

A good start in connection with ServiceNow

d. Automated development tools

Some tools developed.

7. What are your suggestions to enhance the suitability of the new framework WEBTTCom and its implementation?

Need to find suitable business use cases to demonstrate the benefit of web3.