Interoperability Framework to Enhance DLT-based Systems Integration with Enterprise IT Systems

A thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy in Information Systems

Jitendra Vijaysingh Hushare

University of Canterbury

2021

Abstract

Distributed ledger technology (DLT) has generated tremendous interest due to its popular application to Bitcoin and other cryptocurrencies. Despite its enormous potential business benefits and even greater hype, DLT never attracted significant investment and its widespread implementation failed to occur. One of the most recognised reasons is the lack of an integration framework for integrating DLT-based systems with centralised or non-DLT information technology (IT) systems.

This research endeavours to fill this gap by designing a DLT interoperability framework (DIF). This framework is based on the interoperability principles derived from integrated DLT-based solutions and modern organisations' integration needs and practices. DIF enables organisations to design interoperability architecture and integrated solutions for enterprise implementation. Based on the DIF, this research also developed and instantiated a Hyperledger Fabric DLT solution prototype (HDSP) on Amazon Web Services (AWS) for the manuka honey supply chain (MHSC) use case.

The research utilised design science research (DSR) methodology to develop the DIF and HDSP. Iterative artefact evaluations were undertaken using formative (ex-ante), summative (ex-post), maturity model for enterprise interoperability (MMEI), IT professional evaluation, and artefact instantiation and demonstration techniques suggested in the DSR. The DIF, HDSP and their evaluation provide a pathway for organisations to design and implement integrated DLT-based solutions. The knowledge generated and utilised in this research provides a robust theoretical foundation for building and implementing such integrated solutions.

Acknowledgements

I am extremely thankful to the members of my supervisory team for their guidance, support, and timely advice. Steve kept me updated about research events and activities and introduced me to the right people. His frequent invitations to his home café for extended discussions and delicious meals served by his wife, Paula, provided enough food for thought to avoid possible research pitfalls, take an optimal approach, and make the best decisions. My primary supervisor, Constantine, provided critical feedback and gave me complete ownership of the research, so I was ultimately responsible for its outcomes. His relentless focus on improving my research communication, presentation, and attention to detail are lifelong learning lessons I will never forget. I appreciate these more than I can say.

Thank you also to our administration staff, especially Susan. Your ever helping attitude and smiling faces kept us moving; I hope you can always provide such great support.

My family is my source of inspiration. Thoughts of my son, Anish, and my daughter, Nandini, encourage me to do better every moment. My wife, Bharati, is a source of immense strength for me, and writing this Ph.D. would not have been possible without her. The courage she demonstrated during the COVID-19 pandemic was unparalleled, and she did exceptionally well in taking care of our children, family matters, and her own health. Also, despite her age, love, and attachment to me, my mother always gave me the wings and free will to do whatever I wanted. I am fortunate to have such great family support and inspiration in my life. Thank you, God!

Overall, the Ph.D. has been an enjoyable journey for me. After a long time, returning to academic life was like living one’s young student life again, and I think I lived it. Raja, Irina, Saman, Laura, Daniel, regular birthday parties, movie time in the department, frequent presentation sessions, barbeques, potlucks, and time spent with research colleagues took the Ph.D. pressure away. This research has made me more humble, respectful, valuing friends and people around me, and given me much more than I could have ever expected!

Table of Contents

Abstract ii

Acknowledgements iii

Table of Contents iv

List of Abbreviations vii

List of Figures x

List of Tables xi

Chapter 1: Research Introduction 1

1.1 Distributed Ledger Technology (DLT) Integration Challenges 1
1.2 Research Motivation 2
1.3 Research Goal 3
1.4 Research Questions 3
1.5 Research Objectives 3
1.6 Research Significance 3
1.7 Thesis Summary 4

Chapter 2: Literature Review 6

2.1 Interoperability and its Significance 7
2.2 Contemporary Interoperability Approach 7
2.2.1 SOA Integration Approach 8
2.2.2 API Gateway and Microservices based Integration 8
2.2.3 Event-Driven Integration Approach 9
2.3 Characteristics and Categorisation of DLT 9
2.4 Motivation to Improve DLT Interoperability 11
2.5 DLT Interoperability Challenges 12
2.6 Interoperability Solutions from Literature 12
2.7 Research Gap 15
2.8 Replicating and Extending the Interoperability Principles 16

Chapter 3: Design Science Research (DSR) Methodology 19

3.1 IS Research and Design Science 20

3.2 Design Science Research Contributions 22
3.3 Research Methodology...23
3.4 Guidelines for Design Science Research 25
3.5 DSR Knowledge Contribution and Consumption..28

Chapter 4: Distributed Ledger Technology Interoperability Framework (DIF)......................31

4.1 Interoperability Framework and its Components...31
4.1.1 Enterprise Participating DLT Network Nodes Component.....................................33
4.1.2 DLT Gateways Layer Component..34
4.1.3 DLT Metadata Component..35
4.1.4 In-Flow Broker Component...35
4.1.5 Message Service Layer Component 36
4.1.6 Out-Flow Broker Component..37
4.2 Foundational Principles for DIF...38
4.3 Integration Architectures based on DIF 40
4.3.1 Interoperability Architecture using API Gateways, ESB, and Service Mesh 40
4.3.2 Event-driven Distributed Architecture 43
4.3.3 Generic Interoperability Architecture...46

Chapter 5: Hyperledger Fabric DLT Solution Prototype (HDSP)...49

5.1 HDSP Architecture and Solution Design...50
5.1.1 Solution Architecture..50
5.1.2 Transaction and Data Flow...53
5.1.3 HDSP Solution Characteristics...55
5.2 Business Use Case and DLT Platform Selection 57
5.2.1 Supply Chain Use Case Selection 58
5.2.2 Distributed Ledger Technology Platform Selection...60
5.3 Hyperledger Fabric for HDSP Development 62
5.3.1 Hyperledger Fabric Components...62
5.3.2 HDSP Nodes Interactions...65
5.4 HDSP Based Knowledge Contribution...66
5.4.1 API Driven Architecture...67
5.4.2 Data Design 68
5.4.3 Smart Contract (Chaincode) Design..69
5.4.4 Deployment and Operational Aspects 71

Chapter 6: Framework and Solution Prototype Evaluation 73

6.1 Artefact Evaluation Methods 73

6.2 DIF Formative and Summative Evaluation..75

6.2.1 Summative Evaluation..76

6.2.2 Maturity Model for Enterprise Interoperability (MMEI) 77

6.3 DIF Evaluation by IT Professionals 80

6.4 DIF Validation by Instantiating Solution Prototype 81

6.5 Evaluation to Assess HDSP Design Options 82

6.6 HDSP Formative and Summative Evaluation...84

6.6.1 HDSP Formative Evaluation 85

6.6.2 HDSP Summative Evaluation 85

6.7 HDSP Demonstration and Test Cases Execution..87

6.8 How much Evaluation is Enough?..88

6.9 Summary of Artefact Evaluation..89

Chapter 7: Research Summary...92

7.1 Research Problem and its Significance 92

7.2 Research Contribution..94

7.3 DIF and HDSP Evaluation 95

7.4 Knowledge Contribution 96

7.5 Research Limitations..97

7.6 Suggestions for Future Research..98

7.6.1 Interoperability Framework Instantiation 98

7.6.2 Cross-DLT Interoperability 99

7.6.3 Decentralised Applications Interoperability 99

References...101

Appendix A 112

Appendix B 113

List of Abbreviations

AI	Artificial Intelligence
AK	Apache Kafka
AP	Availability Partition
API	Application Programming Interface
AWS	Amazon Web Services
BC	Blockchain
BFT	Byzantine Fault-Tolerant
CA	Consistency Availability
CAP	Consistency Availability Partition
CC	Chaincode
CFT	Crash Fault Tolerance
CI	Continuous Integration
CPU	Central Processing Unit
CRM	Customer Relationship Management
DAG	Direct Acyclic Graph
DApps	Distributed Applications
DEM	Design Evaluation Methods
DIF	Distributed ledger technology (DLT) Interoperability Framework
DLT	Distributed Ledger Technology
DSR	Design science research
DSRM	Design science research methodology
EAI	Enterprise Application Integration
EC2	Elastic Compute Cloud
ECM	Enterprise Content Management
EDA	Event-driven distributed architecture
ERP	Enterprise Resource Planning
ESB	Enterprise service bus
FSC	Food Supply Chain
GIA	Generic Interoperability Architecture
HDSP	Hyperledger Fabric DLT solution prototype

HF	Hyperledger Fabric
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secured
IAAES	Interoperability Architecture using API Gateways, ESB and Service Mesh
IBM	International Business Machine Corporation
IEEE	Institution of Electrical and Electronics Engineers
ID	Identity
IOT	Internet of Things
IP	Internet Protocol
IPC	Interprocess communication
IS	Information Systems
IT	Information Technology
JSON	Java Script Object Notation
MH	Manuka Honey
MHSC	Manuka Honey Supply Chain
MIS	Management Information System
MMEI	Maturity model for enterprise interoperability
MOM	Message-Oriented Middleware
MQ	Message Queue
MQTT	Message Queue Telemetry Transport
MSL	Messaging Service Layer
MSP	Membership Service Provider
NFC	Near Field Communication
PBFT	Practical Byzantine Fault Tolerance
PHP	Hypertext Preprocessor
RDBMS	Relational Database Management System
REST	Representational State Transfer
RFID	Radio Frequency Identification
RPC	Remote Procedure Call
SAML	Security Assertion Markup Language
SC	Smart Contract
SDK	Software Development Kit

SFTP	Secured File Transfer Protocol
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structure Query Language
TB	Tera bytes
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UI	User Interface
XML	Extensible Markup Language
YAML	Yet Another Markup Language

List of Figures

Figure 3.1. Organisational design and information systems design activities……………….21

Figure 3.2. Design science research contribution framework………………………………..22

Figure 3.3. Design science research methodology process model...…………………………24

Figure 3.4. An integrated perspective on knowledge production, contribution, and utilization to IS DSR……………………………………………………………………………….29

Figure 4.1. The distributed ledger technology interoperability framework (DIF) and its interacting components……………………………………………………………………32

Figure 4.2. Integration architecture, based on DIF, enabling interoperability among DLT and non-DLT systems using API gateways, ESB, and Service Mesh (Service Control Plane)……41

Figure 4.3. Apache Kafka DLT interoperability Architecture displays the integration of an organisation's multi-DLT network with its centralised IT systems……………………………...44

Figure 4.4. Generic interoperability architecture……..47

Figure 5.1. Solution Architecture displaying the end-to-end message flow of transactions…..50

Figure 5.2. Solution Architecture of the HDSP technical prototype instantiation…………….52

Figure 5.3. Transactions flow from the external devices or systems to HF Ledger, MongoDB database and customer user interface………………………………………………………..54

Figure 5.4. The major software components implemented in the HDSP……………………...55

Figure 5.5. The components of the Hyperledger Fabric 1.4 developed for the HDSP……......63

Figure 5.6. Order of transactions to achieve the consensus to commit the transaction to the ledger………………………………………………………………………………………65

Figure 6.1. The cyclic formative and summative evaluation process for the DIF……………..75

Figure 6.2. Iterative formative and summative assessment undertaken for the HDSP……... 85

List of Tables

Table 2.1. The table provides a comparative analysis of DLT interoperability solutions suggested in the literature……………………………………………………………………15

Table 5.1. The evaluation of five DLT platforms used to select a suitable one, Hyperledger Fabric, for the MHSC use case…………………………………………………………..61

Table 6.1. Mapping the maturity levels for the **EDA, IAAES,** and **GIA** interoperability solutions, respectively…………...78

Table 6.2. Mapping of the maturity levels of the Interoperability Framework by three professional participants, covering three significant interoperability barriers and four major concerns for each barrier……………………………………………………………………...81

Table 6.3. Sample RESTful input message from the producer, in JSON format……………...84

Table 6.4. Transaction execution time for the HF networks three peers, honey producer, distributor, and retailer nodes………………………………………………………………88

Chapter 1: Research Introduction

Distributed Ledger Technology (DLT) is a decentralised data and transaction management technology enabling companies and consumers to exchange and store data without traditional intermediaries. Distributed ledger technology runs on a set of machines with computational and storage resources called network peers, nodes, or participants. These nodes are not trusted individually but instead trusted as a group due to their diversity and numbers [1]. The distributed ledger technology research and implementation outcomes hint at their potential benefit for many economic, governmental, and service sectors. DLT demonstrated the ability to design and develop modern transparent, secure and reliable business capable systems [2]. The DLT-based solutions development and implementation have solved real-world problems, improved business processes, and eliminated inefficiencies and waste [3] [4] [5]. High profile co-operations, such as the collaboration between IBM and Walmart to improve supply chain process stimulated DLT usage among enterprises [6]. Solutions have been developed in almost all human aspects of life, such as health care, finance, education, supply chain management, governance, auditing and others [7]. Gartner predicted the mass adoption of this technology after climbing out of the initial phase of the trough of disillusionment [8].

This chapter presents the enterprises' DLT systems integration challenges, research motivation, goals, questions and objectives and its significance for organisations and future research.

1.1 Distributed Ledger Technology (DLT) Integration Challenges

The DLT systems implemented by the organisations are standalone and isolated, leading to the emergence of asset, process, and data silos, limiting the potential business benefits from these solutions [9]. There is a lack of accepted architecture, design patterns, and implementation models to integrate DLT system and associated business data with centralised (non-DLT) systems and vice-versa [10]. The absence of integration of these two types of systems is obstructing the value proposition of DLT for enterprises [11].

DLT interoperability solution approaches, suggested by Belchior et al. [7], fall into two major categories:

(1) interoperability among different BCs/DLTs (inter-DLT integration), including integration between DApps (distributed applications) based on DLT platforms; and

(2) interoperability between DLT systems and centralised enterprise IT (non-DLT) systems.

This research will focus on integrating DLT systems with centralised non-DLT systems. Researchers, universities and organisations are working to integrate different DLT platforms (inter-DLT) such as Hyperledger Fabric, Ethereum, Corda [7]. However, those solutions are not mature, and their integration benefits are not well established. Lafourcase et al. [13] suggested that it is not technically feasible to integrate multiple DLT/BC networks. Similarly, Macheel [12] and Lafourcase et al. [13] were sceptical about inter-DLT integration benefits, solution viability and implementation feasibility. For this reason, the integration of different DLT platforms is kept outside the scope of this research.

1.2 Research Motivation

This research is motivated to address the lack of an end-to-end solution for DLT interoperability problem explained in Section 1.1. This research is inspired to realise the maximum benefits of DLT solutions by following a novel approach to solve DLT interoperability challenges with the below considerations in mind:

1. **Enterprise Centric Solution**: Past research has been focused on technological aspects such as interoperability architecture, integration technology, protocols, and tools, with little emphasis on an enterprise requirements [13] [14]. From an organisational perspective, there has been a lack of focus on DLT systems integration with centralised non-DLT systems to augment the business capabilities of an enterprise. .
2. **End-to-end Solution**: Instead of a stand-alone, technical, or architectural level solution, enterprises expect a comprehensive interoperability solution, instantiable at the lowest technical level but also capable of enabling architecture, integration patterns, and design at the enterprise IT ecosystem level. This research develops a DLT interoperability framework (DIF) to enable enterprises to architect and design an integrated solution at the enterprise level.
3. **Implementable Solution**: This research focuses on technically feasible business solutions by architecting, designing, and building a viable solution to integrate centralised IT systems with DLT-based solutions.

1.3 Research Goal

This research aims to provide an integration framework to enable organisations to architect and design an interoperable IT ecosystem. The research proposes an enterprise framework, DIF, to design interoperable architecture to integrate DLT-based solutions with centralised systems.

1.4 Research Questions

Q1: What is the interoperability framework model to enable enterprise systems integration of DLT and non-DLT systems?

Q2: Does the design and implementation of the interoperability framework based DLT solution is practically viable and technically feasible?

1.5 Research Objectives

The objectives of this research were to:

- establish the interoperability principles;
- design the DLT interoperability framework (DIF) based on principles;
- validate the DIF by:
  - designing architecture and integration patterns for the prevalent integration design, architecture styles, and tools that modern enterprises use;
  - designing, developing, and instantiating the HF based solution prototype integrated with centralised (non-DLT) technology, tools, databases, and systems; and
- undertaking the iterative evaluation of the developed DIF, and the Hyperledger Fabric 1.4 (HF) DLT solution prototype (HDSP).

1.6 Research Significance

The artefacts developed in this research provide an end-to-end interoperability framework for enterprises to:

- design interoperability architecture for enterprise IT systems
- build and instantiate interoperable DLT-based solution for a business use case

Based on the interoperability framework, this research has developed interoperability architecture templates for enterprises' reference. The design and instantiation of DLT solution based on the framework provides a pathway for enterprises to develop and implement the integrated solution in their organisation.

1.7 Thesis Summary

The remainder of this is structured as follows:

- Chapter 2 presents the research's literature review and explains the significance and challenges of integrating DLT solutions. Interoperability is defined, and contemporary integration tools and design patterns popular in enterprises are discussed. Available interoperability solutions are discussed, their shortcomings and the research gap this study partially addresses are identified. The replicating and extension methods to develop interoperability principles based on the integrated DLT solution are also discussed.
- Chapter 3 briefly explains the implemented DSRM (DSR methodology) process model, DSR guidelines, and research suggestions followed to meet the research objective. The contribution of the research, both in terms of prescriptive and descriptive knowledge, is explained.
- Chapter 4 explains the Artefact-1 of this research, DLT interoperability framework (DIF), its components, integration layers, and the multiple DLT network nodes. The interoperability principles are explained, which are derived from the integrated solutions in the literature. Integration architectures are designed based on modern technology tools, integration patterns, and methods commonly used in enterprises.
- Chapter 5 explains the design, development, and instantiation of the Hyperledger Fabric 1.4 DLT solution prototype (HDSP), Artefact-2 of this research, to validate the DIF. This artefact, the HDSP, is developed using the Hyperledger Fabric 1.4 (HF) platform for the MHSC use case. The solution architecture, network node interactions, transaction handling, and message and data flow enabling the interoperable solution are all explained. As a contribution to knowledge, the software practices, data design, smart contract (SC), and deployment and operational practices learned during the development and instantiation of the HDSP are detailed.

- Chapter 6 explains the DSR evaluation strategy adopted and evaluation methods used to evaluate the DIF and the HDSP. Artefacts use formative and summative evaluation methods, artefact instantiation and maturity model for enterprise interoperability (MMEI) techniques for evaluation in an iterative manner. In addition, the HDSP uses solution demonstration, and DIF uses professional evaluations by using MMEI to evaluate the final research product, the DIF.
- Chapter 7 summarises the research contribution by designing the DIF based on interoperability principles. The Hyperledger Fabric DLT solution prototype (HDSP) is developed based on the DIF to demonstrate the integration of DLT solution with centralised systems. The design science research (DSR) methodology and the evaluation methods suggested in DSR are utilised to create the research artefacts and validate them. The knowledge developed during the research contributes to the DSR knowledge base and the research community. The future research suggestions based on the artefacts and knowledge generated in this research are discussed.

Chapter 2: Literature Review

This chapter undertakes the literature review to identify the research gap to address the DLT systems integration challenges. The literature review emphasise the significance of integrated IT systems for enterprises. The interoperability architecture, design patterns, and tools currently used by modern enterprises to integrate centralised IT systems are reviewed and analysed. Different types of DLTs are studied, and the enterprise challenge to establish the interoperability between DLT solutions with current IT systems is examined. The research gap is recognised by analysing the integrated DLT solutions identified in the literature review.

Internet made the most significant technological breakthrough connecting computers and servers, allowing shared services, computing, and increased storage capacity. This interconnectivity triggered a shift from single machine computing to distributed and interconnected processing [15]. The service oriented architecture (SOA) enabled the system components to break down into services, and integrate them to provide business processes and customer solutions [16]. The event-driven architecture (EDA), distributed computing, and Microservice architecture (MA) allowed the rapid development of small-sized (micro) and medium-sized (macro) functional services [17] [18]. However, integrating these services to create business solutions and processes is challenging [19]. This interoperability challenge of increased complexity, several services and disparate technologies and platforms compound with the introduction of new promising technologies, such as DLT, into the organisational IT landscape [11].

This chapter engages with the literature from research and practice to explain the overall interoperability landscape, relevant integration approaches, and research gap in enabling to address DLT integration problem. The following section, Section 2.1, defines interoperability and explains its significance for enterprises. Section 2.2 discussed organisations' contemporary interoperability tools, and integration patterns to address current integration needs. The evolution of DLT, and its technical capability and types, are discussed in Section 2.3. Interoperability has always been a challenge for organisations, and was augmented with the introduction of new technology such as DLT. The significance of DLT interoperability and its integration challenges are explained in Sections 2.4 and 2.5 respectively. Organisations, universities, and research institutes have developed solutions to address DLT integration challenges, as summarised in Section 2.6. The analysis of these discussed interoperability solutions helped identify the research gap in meeting DLT interoperability requirements of modern organisations, as explained in Section 2.7. Lastly, Section 2.8 describes the replication and extension methods used to establish the DIF interoperability principles based on discussed DLT-integrated solutions.

2.1 Interoperability and its Significance

The IEEE (Institute of Electrical and Electronics Engineering) defines *interoperability* as "the ability of two or more systems or their components to exchange information or to use the information that has been exchanged." Pillai et al. [20] suggested that cross-communication among the systems does not intend to make immediate state changes to another system and instead triggers functionalities or services on the other system expected to operate within its network to realise the state change. Over the last three or four decades, technology has become a key driver for enterprises seeking to help their business optimise and grow [21]. Technology systems are supposed to create an interoperable network enabling multiple systems or components to integrate and reuse the distributed business processing capabilities spread across multiple systems.

Enterprise systems are often heterogeneous and composed of diverse operating systems, devices, protocols, and technology platforms [22]. Modern enterprise architecture converges towards distributed architecture with heterogeneous systems spread across multiple geographical boundaries [22]. Despite platform, data, programming languages and execution differences of systems, interoperability synergises among these distributed capabilities to enhance and create new business processes [23]. Integrating these diverse distributed applications is a prerequisite for enterprises wanting to improve their business competitiveness [24]. Such interoperable systems must be governed, designed, and deployed to meet business processing requirements. IT system architecture, design, and infrastructure is supposed to be in line with an organisation's business vision, and designed, developed, and integrated according to the strategic view of Enterprise Architecture [25].

2.2 Contemporary Interoperability Approach

Interoperability is crucial for a business capable, collaborative, and integrated IT ecosystem that can produce efficient and risk-averse business outcomes [26]. Interoperability is a business case enabling lower business operational costs, better customer experience and service, and faster availability of information [27]. Enterprises invest significantly in solution integration by designing integration architecture and design patterns to integrate their distributed system capabilities [28] [29]. The following three subsections explain the contemporary integration methods and techniques widely used by modern organisations, which are utilised in this research to design the DIF based interoperability architectures.

2.2.1 SOA Integration Approach

Today's enterprise systems are distributed and built using service oriented architecture (SOA) [16]. This approach enables the composition of multiple fine-grained services into a coarse-grained one—the composing service results in more complex and functional ones than their standalone constituent services. A popular architectural style is to string together the services to design and develop business processes, IT services, and customer solutions [30].

Service oriented architecture based distributed services and solutions are loosely coupled, and implement the "separation of concern" principle [31]. Message queuing techniques enable service capabilities to integrate explicitly, using products such as message-oriented middleware (MOM) and enterprise service bus (ESB), as extensions [25]. Enterprise service bus (ESB) applies modern operating system design concepts to integrate independent running services on separate, distributed, and disparate networks [32]. ESB has enabled commodity services such as data transformation, message transformation, intelligent routing, and database access to integrate with business services. Enterprise service bus has been established as a common purpose, flexible to adopt in projects and scalable beyond the limitations of hub-and-spoke broker enterprise integrators [32].

2.2.2 API Gateway and Microservices based Integration

As distributed systems are becoming the norm, there is a proliferation of microservices being built and deployed in IT systems and applications [33]. Microservices offer cross-platform compatibility by exposing their business capabilities via API RESTful or SOAP-based web services. The API Gateway and Service Mesh enables interoperability among these services [34]. API gateway is a single-entry point for all the system services and addresses some significant common concerns such as security, caching, monitoring, and message throttling [35].

Microservices rapidly emerge as a trendy service implementation design and development pattern in Cloud and large IT environments to improve system integration and scalability [36]. However, this increases the complexity of service communications, library management, and operational aspects of the network function logic (e.g. service routing) with the application code [35]. Service mesh and tools such as Istio are considered acceptable solutions to address these concerns, reducing the complexity of managing services and operational challenges such as traffic routing, service discovery, visibility, and failure handling [37].

2.2.3 Event-Driven Integration Approach

An event can be a user action, a trigger within a system, or an action caused by a system interface. In event-driven integration, the event will transfer the event-state and deliver it to the systems interested or subscribed to the event, and consumed by various systems, and applications [18]. An event is an asynchronous process that delivers the event-state message to the receiver, triggering event processing by the receiving service. Event-driven integration can work independently or coexist with API-driven integration to enable more agile, capable, and faster integration with distributed systems [38].

Event-driven architecture (EDA) is a model and software architecture for application and system design. Blue-chip organisations (established, stable and well-recognised organisations) have widely adopted EDA, as it offers minimal coupling of services and is a good option for modern distributed application architecture [39]. Apache Kafka (AK) is the most widely implemented EDA tool for implementing event-driven architecture in enterprises, and provides an asynchronous protocol to connect system components, sub-components and programs [40]. Apache Kafka is based on a distributed architecture, providing storage, availability, and linear scale-out [40], and can run hundreds of terabytes of production topics, against which the users can define queries and execute them using an SQL interface [41].

2.3 Characteristics and Categorisation of DLT

In 2008, an author assuming the name of Satoshi Nakamoto proposed innovative distributed ledger technology (Blockchain is a type of DLT) [42]. This technology frees the book and record-keeping from the centralised, authoritative setup, to a distributed, shared setup, prompting a significant mind-shift in business processing [43]. A distributed ledger is a digital database, updated and held by all the participants of a large or medium network, independent of each other [1]. This ledger functions without a central authority, and broadcasts and manages transactions, events, and data. Every node or participant on the network agrees to abide by a process to reach a transaction consensus. The distributed ledger is updated, and the network nodes update the copy of their ledger [5]. Thus, the distributed ledger liberates the network nodes or unknown participants from the lack of individual trust issues by establishing collective trust and building transparent systems.

Principally, DLT is based on globally distributed ledgers and connected through a peer-to-peer network to verify, approve, and store transactions. The technology that controls the distributed ledger manages the network's transactions chronologically, signs them cryptographically, and makes transactions immutable and accessible to all the network participants [44]. The ledger copies are distributed across the network, and transactions or events cannot be entered into the ledger unless the consensus criteria are met. However, data are immutable once entered and cannot be changed or deleted [45]. Distributed ledger and data immutability improve trust amongst non-trusted network participants; the cryptographic signature of records ensures transaction and data security.

Distributed systems must be Byzantine fault-tolerant (BFT) to protect networks from malicious nodes [46]. DLT executes a consensus algorithm to generate agreement on a ledger state in the presence of Byzantine faults. Consensus algorithms are a critical aspect of DLT, because they define the behaviour of network nodes, the security assumptions, and the interaction of each node [47]. Therefore, they affect how DLT peers interact and function; for example, in Bitcoin’s PoW (proof-of-work), nodes have to calculate a cryptographic challenge, competing with each other to validate transactions [48]. Another DLT, Tendermint, performs BFT state machine replication for deterministic state machines, supporting fewer than one-third of faulty participants [49]. In the widely used Hyperledger Fabric, a private DLT network platform, a BFT consensus algorithm allows higher transaction throughput than PoW [50]. This capability enables a subset of nodes (called “endorser peers”) to endorse and execute transactions using a weaker consensus mechanism [51].

Apart from differences in the consensus mechanism, DLT networks can be considered public (permissionless) or private (permissioned). A permissionless network does not require trusted and verified participants to access the network ledger [52]; Ethereum and Bitcoins are examples of popular public DLT networks. Permissioned DLTs use a technical platform on which users are verified and trusted and thus can be held responsible or accountable according to a DLT network governance model, and are therefore suitable for governmental and enterprise needs [53]. Hyperledger Fabric, Tendermint, Quorum, Multichain, and Corda, are instances of permissioned DLTs. Most enterprises will participate in a permissioned DLT network [2]. This research is centred around enterprise requirements and focuses on permissioned types of networks.

2.4 Motivation to Improve DLT Interoperability

Modern organisations use every opportunity to leverage IT systems to improve their business performance and reduce costs by managing operational risks. Before investing in and implementing new technology such as DLT, enterprises consider the architecture, solution design, implementation instances, and operational risks [54]. In addition, organisations also evaluate the integration capability of the new technology (e.g. DLT) with the prevalent IT technologies to assess its ability to synergise their overall IT capabilities [55].

Enterprises face numerous challenges in designing, developing, and instantiating DLT solutions, including, but not limited to, interoperability, security, scalability, legal issues, and data privacy [56]. Research work by Kumar et al. [2], Andoni et al. [57], Rodríguez-Espíndola et al. [58], and Patki et al. [59] suggested that the lack of standard DLT interoperability solutions is one of the main reasons for the lack of broader enterprise adoption of DLT. The authors' reasons were that DLT will neither run in isolation, nor replace operationalised solutions in the foreseeable future. DLT systems have to be integrated with centralised IT (non-DLT) systems to synergise the capabilities of these two types of systems. This integration will augment an organisation's IT capability, empowering it to improve and build new business processes and capabilities. This requirement makes interoperability a prerequisite for any enterprise considering implementing a DLT solution.

The main focus of DLT adoption is to include the business-critical core functionality in DLT, which directly helps enable trust and transparency amongst network participants [9]. The enterprise trend is to create a minimal DLT functional ecosystem with few relevant participants [60]. This tendency and the large variety of DLT platforms have resulted in niche, heterogeneous DLT networks with different consensus management and data processing methods, resulting in siloed business capability, along with prevalent asset and data silos [61]. Today’s enterprises operate in a highly competitive environment where business efficiency and process optimisation are critical aspects, adversely impacted by systems and applications that work in silos. A business will invest in new technologies only if stakeholders see the synergy between the impending technology and their implemented IT capabilities. To that end, seamless integration among these systems and applications is an essential capability sought by enterprises [62]. This interoperability will empower enterprises to leverage new, diverse, and disparate technologies, distributed data, and enable infrastructure ability for business benefits [63].

2.5 DLT Interoperability Challenges

Enterprises are interested in designing and implementing production-grade, practical, and manageable DLT solutions [64]. The variety of DLT technologies makes it challenging to categorize DLTs and their interoperability solutions, as there are no accepted DLT interoperability standards [65]. The possible benefits and operational feasibility of interoperability amongst various DLT platforms (inter-DLT) are not well established and not discussed in this research. Organisations are expected to integrate DLT solutions with centralised IT systems to acquire the knowledge, experience in designing, developing, and implementing the integrated DLT systems. Even if an enterprise is part of numerous DLT networks (and most likely they will be), they will visualise the benefits of integrating the business capability and data from these multiple DLT networks with their prevalent IT capability. Providing this interoperability capability can be a steppingstone to encourage organisations to adopt multi-DLT solutions; this research is focused on this crucial aspect.

Based on Belchior et al.’s research [7], two significant levels of interoperability can be established: (1) among different BCs/DLTs (including between the DApps (distributed applications using DLT platforms); and (2) between DLT systems and centralised enterprise IT (non-DLT) systems. Modern enterprises predominantly use centralised IT systems [8]. Therefore, this research focuses on interoperability from an enterprise perspective and is dedicated to integrating DLT systems with centralised (non-DLT) systems.

2.6 Interoperability Solutions from Literature

The interoperability solution of BC with contemporary technologies (artificial intelligence [AI] and 3D printing) was suggested by Rodriguez-Espindola et al. [66] for a humanitarian supply chain. Their research investigated the challenges faced in the 2007 flood disaster in Mexico to identify a potential solution for effectively supplying relief food, and designed a framework to achieve workable integrated solutions. The authors argued that combining these three technologies using the integration framework can positively affect a humanitarian supply chain. The integration of BC with robotics and AI was discussed by Lopes et al. [67], but without elaborating on the proposed solutions, methods, or framework. Chavali et al. [68] discussed AI and BC integration and explained a decentralised AI framework and AI service collaboration model. The flow of secure, extensive, and accurate data from BC drives the AI systems providing transformational value to enterprises.

The integration of BC with IoT (internet of things) was discussed in detail by Reyna et al. [58] and Panarello et al. [69], and predicted that the integration between these two modern technologies would revolutionise IT solutions by integrating the IoT devices with current systems and BC/DLT solutions. Blockchain may enhance the IoT solution by providing trusted and secured shared services and making information traceable and reliable. Transparent and open data sources will remain immutable over time, enhancing IoT security. In areas such as smart cars and smart cities, disseminating trusted and reliable data will encourage new users and participants to join the ecosystem. Three integration approaches were discussed by Reyna et al. [58]:

(1) IoT device to IoT device;
(2) IoT to BC; and
(3) a hybrid approach, in which only partial data and interaction were executed in BC, while the remainder were shared directly among the IoT devices.

Their discussion focused on a literature review of IoT and BC interoperability solutions and the possible benefits and risks of integration. However, they did not provide the technology solutions, frameworks, methods or similar artefacts needed to address the core interoperability challenges.

The integration of BC with a MIS (management information system) to integrate Web 2.0 (client-server based application) with Web 3.0 (decentralised BC-based systems) by using Ethereum was explained by Chan et al. [70]. Consortium or private BC nodes were set up on the Ethereum platform utilising the Web3.js (a library that defines the format to exchange data with Ethereum) API, and Geth (Go Ethereum client). The BC data accessible from web applications help store the key records on the BC, offering data transparency and immutability, and functioning as the single source of truth. A framework to integrate traditional Web 2.0

based applications with BC-based Web 3.0 based systems was also illustrated. The web application was programmed to interact and communicate with an Ethereum node using interprocess communication (IPC) or HTTP. There was no direct interaction between BC and the application; the exchange executed with an Ethereum node, which interacted with the other nodes in BC. The integration solution suggested, however, had significant shortcomings:

- it did not suggest the technical or architecture design of the solution implemented;
- the Ethereum client Geth needed to be installed on all the BC nodes;
- the integration framework was based on Ethereum, so for other BC technologies, the framework would be irrelevant;
- framework principles, fundamental rules and constraints that would help decide the business situations in which the framework could be applied were not suggested; and
- the solution focused on web integration; however, BC would add significant value in integrating the back-end processes that support the core business capability [71].

The interoperability of BC with a healthcare application was explained by Zhang et al. [10]. Their research focused on maintaining ongoing changes while managing integration complexity, minimising data storage, addressing security concerns, and tracking health variations among a large population. This research used a health DApps (distributed application) to address the interoperability challenges when the DApps is changed and functionality is extended. The solution used four software design patterns to decouple the interaction and improve application scalability, sharing resources more efficiently to help improve the overall design and functionality, and make applications more modular and easier to maintain. The study used public BC to create an interoperable environment using BC's immutable and verifiable characteristics, which cannot be achieved with centralised systems [10]. Their research is specific to the health care domain, and no framework or methods were suggested as applicable among other domains and BC technologies. Furthermore, the solution offered was for the public BC, which is unrelated to the enterprise environment.

The summary of the comparative analysis of the solutions discussed in this section, indicating their strength and weakness, are summarised in Table 2.1.

Interoperability Solution	Technical implementation	Solution Design	Enterprise Interoperability
Solution by Rodriguez-Espindola et al. [66]	No	No	Yes
Solution by Lopes et al. [67]	Yes	No	No
Solution by Chavali et al. [68]	No	Yes	Yes
Solution by Reyna et al. [58]	No	Yes	No
Solution by Panarello et al. [69]	No	Yes	No
Solution by Chan et al. [70]	No	No	Yes
Solution by Zhang et al. [10]	Yes	No	No

Table 2.1. The table provides a comparative analysis of DLT interoperability solutions suggested in the literature. Each row in the table indicates the strength of the solution from technical implementation, Solution Design and Enterprise Interoperability perspectives.

2.7 Research Gap

Enterprises are cautious when considering implementing new technologies due to the risks these present to their day-to-day business operations discussed in Section 2.4. Therefore, any new technology, such as DLT, AI, Machine Learning, IoT, Cloud etc., will be monitored for a few years before being tested for proof of concept or solution feasibility in a limited and risk-averse business context. During this phase, apart from factors such as security, scalability, and legal issues, the implementation will be closely monitored for its interoperability with prevalent systems. As discussed in Section 2.5, this integration is expected to augment the enterprise's current IT solutions' ability to create or significantly enhance business capability and processes. The success of such limited implementation will encourage enterprises to invest in broader implementation in other critical business areas of the organisation. As technology usage grows, enterprises start focussing on resolving other implementation challenges simultaneously, seeing the utility of technology for their business.

Academic and professional literature has proposed inadequate solutions to address BC or DLT interoperability challenges, and many are analysed in terms of their interoperability capability in Section 2.6. Few research focuses on the BC solution's technical implementation without elaborating on how it can scale to a complex and highly heterogeneous enterprise systems landscape. Furthermore, these solutions do not consider that enterprises are most likely to participate in multiple BC/DLT networks that limit the usability of their single-DLT based solutions. Some solutions are limited to a single DLT network [70], and some lack focus due to the lack of design rules or principles [67] or the lack of insight into providing the implementation architecture of these solutions for the prevalent enterprise interoperability patterns [58] [69]. Although a few solutions offer an interoperability framework or architecture [66], most do not provide the association and synergy between a low-level technical solution and the suggested IT system architecture or framework [10]. This inadequacy results in a lack of coherent interoperability solutions for enterprises to seriously consider for experimentation and further implementation.

There is a research gap relating to the pressing need for DLT solutions that are integrated, practical, and relevant to the enterprise's business environment. Such solutions are supposed to demonstrate the feasibility of DLT-based systems at the technical level by integrating DLT solutions with modern technologies. This specific technical solution is expected to be based on an interoperability framework or integration model or architecture suitable to an enterprise's complex and diverse IT ecosystem and technology landscape. The framework or model have to be linked to the organisation's prevalent interoperability architecture, design patterns, tools, and implemented solutions. Such a framework would assist in providing the architecture and design visualisation based on modern enterprise tools, technology, architecture, and design patterns. The solution is supposed to be validated against implemented solutions to establish the framework's usefulness for enterprises considering DLT implementation in their specific business context. This research aims to fill this gap by designing, developing, instantiating, and evaluating such a solution.

2.8 Replicating and Extending the Interoperability Principles

This research conceptualises and designs a DLT interoperability framework (DIF). The DIF is based on interoperability principles developed from the current research. These principles are replicated, extended, and derived from the integration principles and rules suggested explicitly or implicitly, based on the analysed DLT integrated solutions overviewed in Section 2.6 by [66], [67], [68], [58], [69], [70] and [10]. No single solution currently meets enterprises' comprehensive interoperability requirements, and the extant solutions have limited strengths and practical benefits. A few solutions have proposed integration design at technical and programme levels [67], [70], and a few at design pattern and system component levels [68], [70]. Other solutions have suggested interoperability architecture at the IT ecosystem level [58], [10]. The solutions proposed in Section 2.6 have been implemented in many business domains due to their suitability to a business context, requirements, and unique challenges. In addition to currently available DLT platforms (e.g., HF, Ethereum, R3 Corda, Multichain), new platforms will eventually emerge to support future business needs. Such multi DLT-platform ecosystem sets the expectations that the interoperability framework have to be relevant and applicable to current and future multiple DLT technologies. Considering this requirement, the following paragraphs explain the techniques and methods used to extend and replicate the principles from the discussed DLT interoperability solutions outlined in Section 2.6.

Tsang et al. [72] discussed six types of replications and extensions. The current research has utilised Conceptual Extension, in which the constructs of the solution design, architecture, and framework principles from the solutions suggested by Rodriguez-Espindola et al. [66] and Lopes et al. [67], are used to create an enterprise-centric principle. These solutions also demonstrate that only real-time data valuable to coordinate supply chain logistics with the decision-makers have to be included in DLT; centralised systems can process the rest. The solution suggested by Lopes et al. [67] advocated that the API connects with the external algorithm capabilities, replicated in the DIF. The solution by Chavali et al. [68] indicated the integration of BC and other systems using high-level APIs. The importance of the API layer in event triggering and data exchange in designing an interoperable solution was discussed in detail by Hewett et al. [22], He et al. [24], and Svetashova et al. [73].

Beck [74] discussed the systematic extension, including the extension of solutions explained in previous studies. The strategy to extend the scope of earlier studies and replace the original study’s construct with one of the many constructs in the targeted research is used in the current research to increase the generalisation by modifying and extrapolating the original design. Research by Gadge et al. [34] discussed the principle of data management by embedded or IoT devices and cost-efficient operations. These principles are extended in the DIF to select the limited business data included in the ledger and IoT-enabled design. When documents or massive data are involved, only the digital signature is supposed to be stored in the ledger; the actual data can reside on centralised IT systems.

Brown et al. [75] and Fuess [76] emphasised replication by extending the work of previous studies, and making changes in the attribute or active variables to increase the external validity of the research effort (e.g., generalisability). Panarello et al. [69] and Chan et al. [70] covered the integration based on Cloud integration, APIs, and efficient data structures in BC. These are replicated and extended in the current research to increase the relevance and widespread use of framework principles for modern enterprises. The replication of web (front-end) integration of DLT is avoided so enterprises can focus on integrating core business logic to the back-end, and continue using the current front-end technology stack for their customer user experience. Hyman et al. [77] separated three types of research operations subsumed under replication. The DIF used the first type of replication that builds upon, extends, and validates under changing circumstances, the suggestions and findings of previous studies. The interoperability principles of the DIF have avoided the implementation challenges (i.e., scalability, tight coupling, and duplicated resources) faced by Zhang et al. [10] to implement the DASH (Distributed App for Smart Health) system. The principle of storing hashing or signatures, and selective business data in BC ledgers avoids the storage capacity challenge faced by Reyna et al. [58]. The significance of providing Cloud-enabled interoperable solutions is explained in research by Hewett et al. [22], Belchior et al. [7], Nguyen et al. [78] and Besancon et al. [79].

Chapter 3: Design Science Research (DSR) Methodology

This chapter explains the research activities undertaken based on the design science research methodology (DSRM) and the criteria used to categorise the research that falls under design science research (DSR). This research followed the seven core guidelines and fundamental principles of DSR suggested by Hevner et al. [80]. A knowledge contribution framework is utilised to demonstrate the usage of the knowledge base and this research’s contribution to the knowledge base.

Enterprises design, develop and operationalise information systems (IS) to establish business capabilities and processes by developing business solutions, work systems, and business processes. Information systems researchers are responsible for acquiring and extending knowledge to help enterprises develop and enhance IT systems for business growth and cost-effective operations. Behavioural science and design science are two distinct but complementary paradigms used to acquire and further such knowledge in IS [80]. The behavioural science paradigm is rooted in natural science research methods and endeavours to explain and predict human and organisational phenomena. This paradigm enables practitioners and researchers to understand the people, organisational, and their technological interactions to enhance business operations and management by improving the efficiency and effectiveness of the organisation [81].

The design science paradigm is a problem-solving paradigm rooted in engineering and the science of artificial (built by humans, instead of nature) [82]. It pursues innovations to define and establish practices, ideas, products, solutions, and technology capabilities, to promote design and analysis, implement and utilise an IS in an organisational context. The artefacts created by design science follow natural laws and behavioural science theories and rely on the extant predominant kernel theories [83]. Ever-changing and demanding business capability requirements expect creative advances in solution domains when the available approach is insufficient or needs enhancement to meet the needs of a new or changed business and technology context. As new technology capabilities such as DLT emerge, design science is applied to current and new business areas to improve business processes and operations in ways previously beyond the IT ambit's imagination [84].

This chapter explains the research methodology, guidelines, and steps followed to meet the research objectives (Section 1.5). The following section describes the importance of information system (IS) research for organisations and the contribution of DSR towards that. Section 3.2 classifies the study into four quadrants and categorises the studies that fall under DSR. The following section explains the DSR methodology and the steps this research followed based on that methodology. Section 3.4 sets out the seven DSR guidelines followed in this research. The last section discusses the knowledge contribution this research has made to the DSR knowledge base.

3.1 IS Research and Design Science

Organisations are large, complex, purposeful, and artificially composed of structures, people, work systems, and technologies working together to pull enterprises in the desired direction [85]. These components require a close alignment between the business strategy, goals, and organisation processes with the IS strategy, tools, technology, and IT infrastructure [80]. Figure 3.1 presents the required alignment and illustrates the transition of business strategy to organisation infrastructure and IT strategy to create effective IS solutions and support infrastructure. This interplay between business strategy, organisation infrastructure, IT strategy, and IS infrastructure must be addressed in IS research [80]. As the role and influence of technology in enabling business strategy and organisations' infrastructure will grow, this interplay is becoming particularly critical. Therefore, IS researchers are expected to endeavour to allow IT strategies to experiment and invest in new technology solutions and IT infrastructure. This will enable enterprises to use cutting-edge technologies such as DLT, IoT, AI and similar to explore, engage in, and design new structures to align the IT roadmap with their business strategies.

Enabling an organisation to implement its business strategy by solving real-world problems is the crux of design science and is achieved by producing artefacts or final DSR products [86]. In addition to the final product or artefact, DSR is also concerned with the research process, methodology, and activities followed to reach the outcome [87]. Design science research undertakes an artefact validation and evaluation to justify the artefact's ability to address a stated problem efficiently and effectively [88]. The artefact or product is evaluated iteratively to provide feedback to understand the problem better and improve the artefact and design process to help solve the problem efficiently [89]. The design science researcher runs this build and evaluate cycle multiple times in a loop to evolve and improve on the design artefact and design process. Two design processes:

- build (including design) artefact, and
- evaluate artefact

and four design artefacts of:

- constructs (symbols and vocabulary),
- models (representations and abstractions, e.g., DIF),
- methods (practises and algorithms), and
- instantiations (prototype and implemented systems, e.g., HDSP)

are an integral part of DSR [90].

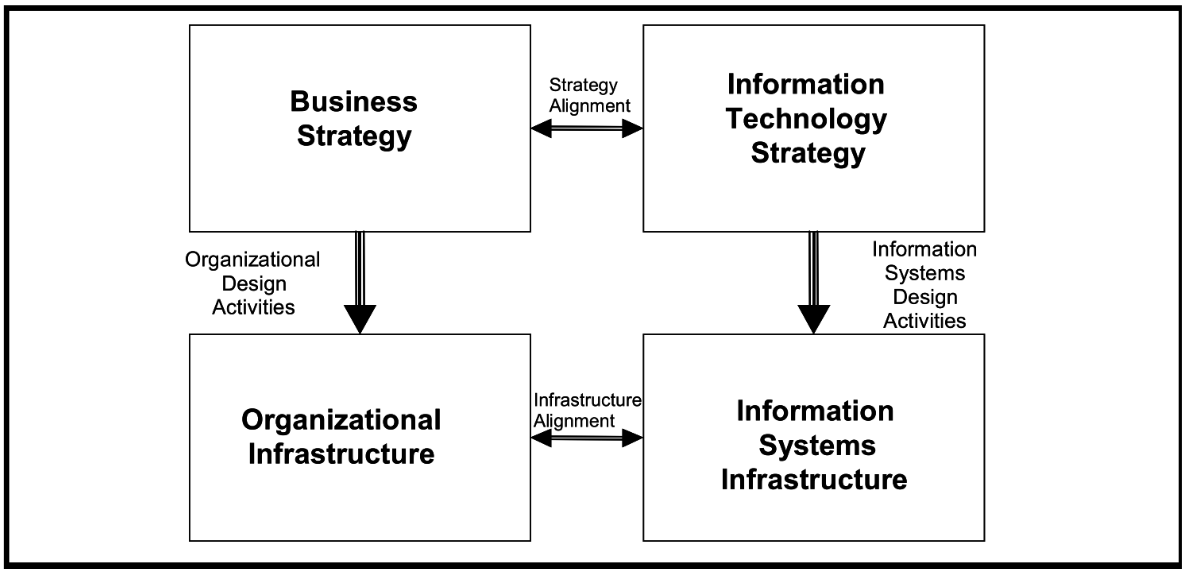
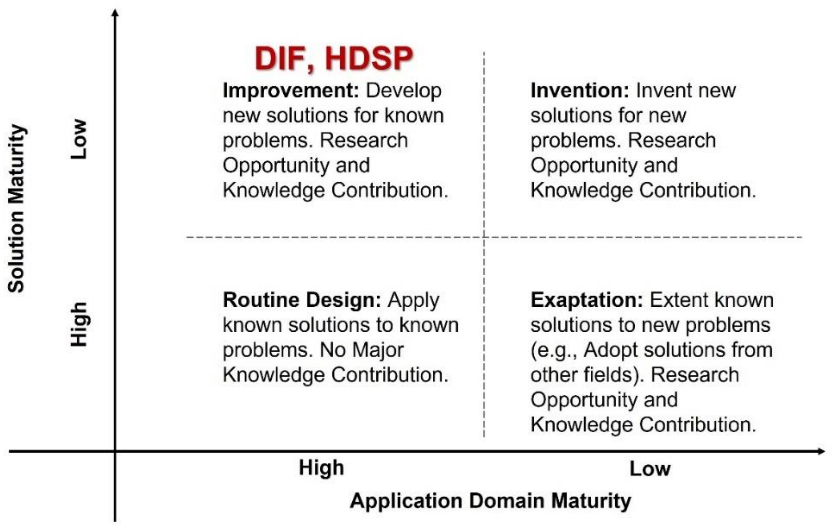

Business Strategy
Strategy Alignment
Information Technology Strategy
Organizational Design Activities
Information Systems Design Activities
Organizational Infrastructure
Infrastructure Alignment
Information Systems Infrastructure

Figure 3.1. Organisational design and information systems design activities. It illustrates the alignments between business and IT strategy and organisational and IS infrastructure. Reprinted from [80].

This research modelled a DLT interoperability framework (DIF), representing the integration framework, and instantiated the HDSP to validate the DIF. The DIF helped understand the problem and suggested a solution by representing the correlation between critical aspects of a problem and its solution components. The instantiation of the HDSP demonstrated that the DIF can be practically implemented in a working IT system to address the interoperability challenge. Based on the DIF, this instantiation of the HDSP shows the feasibility of both the design process and designed artefacts to solve the problem. The design science research also demonstrates the solutions' suitability by undertaking a concrete assessment to validate that the DIF meets its intended objectives.

3.2 Design Science Research Contributions

Not all studies and activities producing artefacts or products can be classified as DSR contributions. Gregor et al. [91] created a 2x2 matrix and defined rules to categorise the studies; see Figure 3.2 presents the research activity contexts and three types of DSR contributions. From high to low, the horizontal axis indicates the maturity of the problem context. The vertical axis, from high to low, represents the present maturity of artefacts as a possible solution starting point for research.



DIF, HDSP
Improvement: Develop new solutions for known problems. Research Opportunity and Knowledge Contribution.
Invention: Invent new solutions for new problems. Research Opportunity and Knowledge Contribution.
Routine Design: Apply known solutions to known problems. No Major Knowledge Contribution.
Exaptation: Extent known solutions to new problems (e.g., Adopt solutions from other fields). Research Opportunity and Knowledge Contribution.
Solution Maturity
Low
High
High
Low
Application Domain Maturity

Figure 3.2. Design science research contribution framework. It represents the matrix of potential DSR contributions and research activities context. The artefacts of this research, DIF and HDSP, fall under the Improvement category. Adapted from [91].

The 2x2 matrix in Figure 3.2 focuses on the research project's knowledge starting point (i.e., maturity) to better understand the project’s objectives and expected research contribution. The research that falls into Improvement, Invention and Exaptation is categorised as a DSR study. These three quadrants, and the Routine Design (non-DSR study) quadrant, are explained next.

Improvement: This quadrant accommodates a solution resulting in more efficient and effective processes, products, technologies, services, or ideas [91]. Both the artefacts of this research fall into this quadrant, which offers a DLT interoperability framework (DIF) to enable the design and development of an integrated DLT solution with centralised IT systems (HDSP).

Invention: This quadrant accepts radical breakthrough solutions, a distinct mind-shift from the accepted way of thinking [91]. Solutions in this quadrant are scarce because innovations are scarce, and inventors are even scarcer.

Exaptation: Individuals with expertise in multiple disciplines often develop original thoughts and ideas. This mindset allows insights and interconnections within a field to result in the exaptation of artifacts to another field [91].

Routine Design: In this quadrant, the present expertise and knowledge about the problem area are well understood, and use prevailing solutions and artifacts to tackle and solve the opportunity or problem. The critical difference between routine design and DSR is that DSR solves interesting, unsolved problems in innovative, more efficient, and effective ways. The critical differentiator is the clear and visible identification of contribution to DSR knowledge based on research artefact(s) or methodologies or both [80].

3.3 Research Methodology

The design science research methodology (DSRM) is a widely accepted research framework for researching Information Systems based on DSR principles [84]. A design science research methodology offers methods, procedures, and practises to execute DSR, and provides a model for undertaking, evaluating and communicating research in IS [92]. Design science creates, evaluates, and validates IT artefacts meant to solve organisational wicked problems [93]. Design research expects to diverge from interpretative analysis or theory testing by relying on the process model to provide guidance (as consumers, editors, and reviewers) in setting the expectations from design research outputs [94]. The design science research methodology includes three elements: conceptual principles to define DSR, practice rules, and processes for carrying out and presenting the study [84]. A DSRM process model and its six sequential activities followed by this research are presented in Figure 3.3.

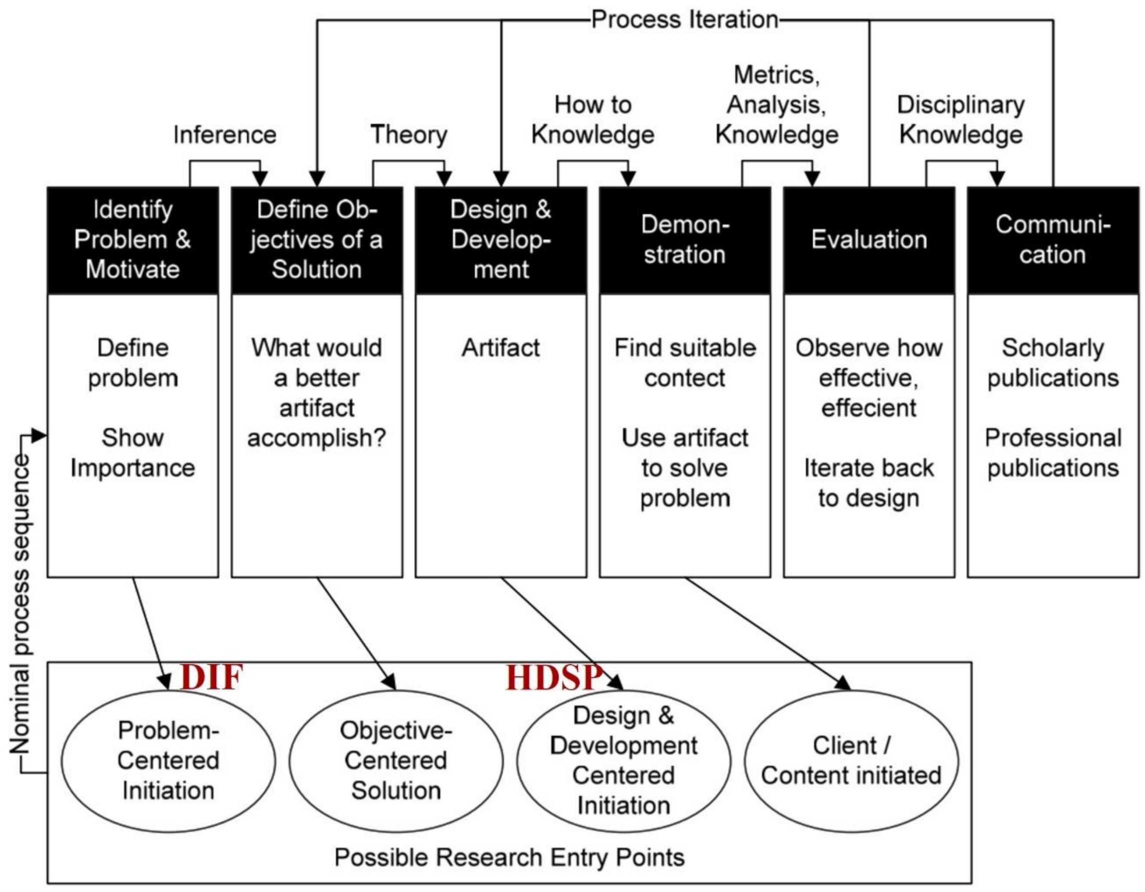

Process Iteration
Inference
Theory
How to Knowledge
Metrics, Analysis, Knowledge
Disciplinary Knowledge
Identify Problem & Motivate
Define Objectives of a Solution
Design & Development
Demonstration
Evaluation
Communication
Define problem
Show Importance
What would a better artifact accomplish?
Artifact
Find suitable contect
Use artifact to solve problem
Observe how effective, effecient
Iterate back to design
Scholarly publications
Professional publications
Nominal process sequence
DIF
HDSP
Problem-Centered Initiation
Objective-Centered Solution
Design & Development Centered Initiation
Client / Content initiated
Possible Research Entry Points

Figure 3.3. Design science research methodology process model. It represents the six sequential steps followed in this research. The artefact DIF followed the problem-centred initiation, and the HDSP followed the design and development centred initiation research entry point. Adapted from [88].

This research followed the DSRM process model and executed the recommended six steps in the sequence presented in Figure 3.3. The problem-centred initiation is the research entry point and trigger for the DIF. Evaluation and communication steps of the DSRM process model provided feedback and insights in an iterative manner that incorporated recalibrating the solution objectives and improving the design, development, and output of the artefacts. The steps followed by this research were as follows:

1. **Identify Problem and Motivation**: A literature review was undertaken using academic journals and professional publications to identify the significant challenges DLT faces in its widespread enterprise usage. Enabling the interoperability of DLT with centralised IT systems was one of the most relevant research areas identified that would motivate organisations to invest in and implement DLT-based solutions.

2. **Define the objective of a solution**: Apart from the business benefits of DLT, enterprises want to know the interoperability capabilities of DLT solutions. Enterprises are keen to utilise interoperability architecture, framework, and DLT solution patterns to deploy in their IT ecosystems, accordingly, the objectives of this research are defined in Section 1.5.
3. **Design and Development**: This research designed and developed two artefacts:
   a. It conceptualised and modelled the DLT interoperability framework (DIF) to enable interoperable architecture and integration design patterns.
   b. To validate the DIF, the research designed, developed and instantiated the HDSP. It also followed the design and development centred initiation process (Figure 3.3) to establish the interoperability of HDSP, based on the principles and solution objectives established in the DIF.
4. **Demonstration**: The usefulness and utility of the DIF were demonstrated by designing the interoperability architecture and integration design using the DIF for the popular architecture models, integration patterns, and tools used by modern enterprises. The working functionality of the instantiation of the HDSP, based on the DIF and its integration principles, was demonstrated to departmental academic staff, fellow researchers, and practitioners.
5. **Evaluation**: The HDSP was instantiated to evaluate and validate the utility and usefulness of the DIF. In addition, the DIF and the HDSP were evaluated based on the evaluation techniques and methods suggested in DSR and explained in Chapter 6.
6. **Publication**: Publication efforts were made with top tier peer-reviewed journals:
   a. The research paper, "Integrating Distributed Ledger Technology into the Distributed Enterprise Architecture with Apache Kafka", was sent to the IEEE Software journal two times.
   b. The research paper, "Recommended Software Engineering practices for Distributed Ledger Technology Solutions", is prepared.

3.4 Guidelines for Design Science Research

This research followed the seven core guidelines and fundamental principles of DSR suggested by Hevner et al. [80], as follows:

1. **Problem Relevance**: The literature review of academic and industry publications identified enterprises' problems in implementing DLT solutions in Section 2.5. Organisations are looking for interoperability architecture and design patterns to build and implement integrated DLT-based solutions with their centralised IT ecosystems.
2. **Design as an artefact**: Two artefacts were designed to address the interoperability challenge. The first artefact was the DLT interoperability framework (DIF) used to enable organisations to build integrated DLT-based solutions with centralised IT systems. The second artefact is the interoperable HDSP, which was designed, developed, and instantiated according to the DIF and its interoperability principles.
3. **Research Contribution**: The DIF was produced to enable enterprises to develop integration architecture for their IT ecosystems. Subsequently, the DIF was utilised to design the standard interoperability architectures based on popular industry interoperability architecture and integration patterns. Another contribution of this research is in designing, developing, and instantiating an interoperable HDSP to validate the DIF and its principles. The knowledge generated based on the development of the two artefacts is the contribution of this research to the DSR project knowledge base.
4. **Design as a Search Process**: The interoperability complexity increases as new technologies like DLT enter the IT landscape. Such complex and large-scale interoperability problems often need to be abstracted by breaking into simpler domains and representing only a subset of the relevant parameters and components [80]. The problem was divided into two levels:
    a. The DIF was conceptualised and modelled according to interoperability principles to address the enterprise-scale interoperability challenge for diverse and large systems, abstracting the code complexity and low-level technical design.
    b. The HDSP was designed and instantiated by implementing low-level technical design and code functioning based on the DIF and interoperability principles.

    It is neither feasible nor practical to model an integration framework applicable to all organisations' diverse and unique interoperability needs. This research designed DIF-based implementation architecture using enterprise service bus (ESB), API gateway and Microservices implementation, event-driven architecture-based implementation, and generic integration implementation.

5. **Research Rigor**: Research by Hevner et al. [80] suggested that the assessment of rigour must be in the context of generalisability and applicability of artefacts; relevance will be compromised if there is an over-emphasis on rigour. The rigour of the DIF in this research was based on the framework's practical usability in terms of helping enterprises design integration architecture. This ability was demonstrated by developing practical integration architecture for popular enterprise interoperability patterns based on the DIF. Moreover, the development of the HDSP established the rigour and robustness of the DIF.
6. **Artefact Evaluation**: The artefacts produced in this research went through formative evaluation (synonym for ex-ante evaluation), summative evaluation (synonym for ex-post assessment), the maturity model for enterprise interoperability (MMEI) framework, and artefact instantiation [95]. The DIF was validated by successfully instantiating the HDSP. The natural and iterative evaluation of the DIF was undertaken by developing interoperability architecture and integration design patterns based on the DIF. The HDSP was developed based on the DIF; even though its evaluation was predominantly artificial (technical evaluation), a natural evaluation aspect was added by including the contemporary technology, tools, architecture, and design patterns popular in modern enterprises.
7. **Communication of Research**: The following opportunities were used to communicate the research process and its outcome to research, academic, and practitioner communities:
    a. Participation in the New Zealand Information Systems Doctoral Consortium in June 2018, July 2019, and July 2021. The conference paper, "Customer-Centric Traceability Framework for Food Sustainability and Quality Parameters to Improve Transparency in Food Supply Chain (FSC)", was presented.
    b. The "Best presentation" award was received at the International Conference on Design Science Research in Information Systems and Technology, January 2020, Bangkok, on "Cloud-Based Distributed Ledger Technology for manuka honey Traceability System".
    c. The research was presented at a design science course (worth 180 working hours), conducted by Dr Jan vom Brocke and Dr Robert Winter, and attended by Ph.D. students from around the globe.

d. The research artefacts, DIF and HDSP, were explained and discussed with academic departmental academic staff, IT professionals, and research colleagues, to solicit feedback to help improve the final research output.
e. The research was presented at the Accounting and Information Systems Department Conference, University of Canterbury, in November 2020. Feedback received from participants was incorporated into the research output.
f. A research presentation was made to the Centre for Inclusive Digital Enterprise (CeIDE) research group in October 2020; feedback and constructive comments were incorporated into the research process and output.
g. The Ph.D. research proposal was presented to the Christchurch Institute of IT Professionals, and their feedback and research direction comments were incorporated and used to enhance the research progress.
h. The research was presented at the Accounting and Information Systems Department Conference, University of Canterbury, in October 2021. Feedback received from participants was incorporated into the research output.

3.5 DSR Knowledge Contribution and Consumption

Drechsler et al. [96] proposed a conceptual framework covering the two DSR knowledge bases of descriptive knowledge (Ω-knowledge) and prescriptive knowledge (λ-knowledge). Solution design knowledge is actionable and technological, and falls into the prescriptive knowledge domain. Descriptive knowledge involves explaining artificial, natural, and human-related activities, composed of measurements, classifications, observations, and logging these explanations into accessible forms [91]. Both knowledge bases illustrate six directions to produce, utilise, and contribute knowledge as modes of design theorising. Figure 3.4 presents an enhancement of the framework suggested by Gregor and Hevner [91] and separates the two knowledge bases from project design knowledge. Design science research projects utilise available knowledge and produce conjectural, untested, and transitory knowledge and entities, most likely in an unstructured, heuristic, and creative manner [97]. This knowledge is shared only amongst project team members. Only selected knowledge suitable for the broader research community is contributed to the DSR knowledge base.

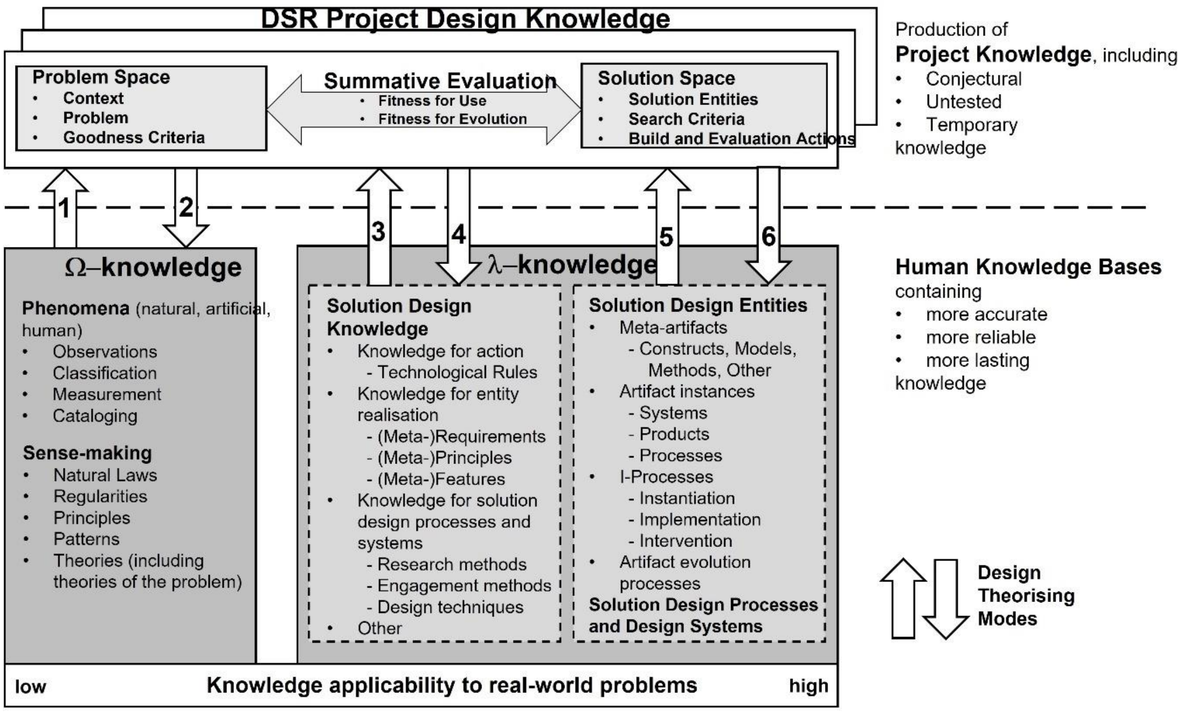


Figure 3.4. An integrated perspective on knowledge production, contribution, and utilization to IS DSR. Reprinted from [96].

This research utilised the framework in Figure 3.4 to use available knowledge and contribute to the DSR knowledge base, as follows:

1. **Mode-1**: Relevant knowledge was utilised to acquire an in-depth understanding of the problem, its context, and the possible diagnosis of the enterprise interoperability challenge. A clear goal statement was identified based on understanding a real-world problem. The research defined solution requirements and goodness criteria, against which the two developed artefacts, the DIF and HDSP, were evaluated. The second source of knowledge was the practical experience, intuition, tacit and non-scientific knowledge, insights, and skills of the researcher, professors, supervisors, IT professionals, and colleagues involved in the research.
2. **Mode-2**: This research enhances understanding of the context and problem by emphasising the need for and importance of DLT solution's integration with centralised IT systems. It explains the background and significance of both artefacts from a business capability, competency, and processes enhancement perspective. The research also provides indicators for possible changes to the behaviour of organisations, DLT network participants, and others, due to the trust and transparency enhanced amongst DLT network peers.

3. **Mode-3**: Knowledge and experience of Enterprise Architecture, ESB (enterprise service bus), layered architecture, API gateways, Service Mesh, Microservices, Apache Kafka, SOA, event-driven architecture, interoperability patterns, and message management, networking and oracles were used to design the DIF. System architecture, design knowledge and tacit experiences were utilised to build the HDSP. The expertise in RESTful API-driven system design, database knowledge, user interface, data modelling, system design principles and rules, and layered architecture patterns were utilised to design the system components. To develop and instantiate the HDSP, programming knowledge, database processing, Docker, Cloud technology, DLT, and testing knowledge were all utilised.
4. **Mode-4**: The DIF validates the developed interoperability principles derived from available integrated DLT solutions. The DIF also advocates the system architecture and design for interoperability architecture using API Gateways, ESB and Service Mesh, event-driven distributed architecture, and generic interoperability architecture. The HDSP design, development, and instantiation followed the DIF and its principles, which can be used in similar business solutions. The research also recommends software development practices, API driven architecture, data design rules, Smart Contract design and development practice, and operational aspects of DLT-based solutions, that enterprises and the research community can utilise.
5. **Mode-5**: The DIF utilised the layered architecture model, SOA principles, metadata model, external data feed, and canonical messaging principles to develop and refine the framework. The Hyperledger Fabric DLT solution prototype (HDSP) utilised the available solution entities to design, develop and instantiate the technical prototype. It also used DLT platform Hyperledger Fabric, MongoDB database, AngularJS framework, Node.js libraries and JavaScript Object Notation (JSON) data structures to implement the HDSP. Available knowledge of DSR artefacts evaluation was utilised to evaluate both artefacts.
6. **Mode-6**: The DIF and architecture models developed based on the DIF (see Section 4.3) are its solution design contribution. The HDSP's technical instantiation, the components (API layer, Hyperledger network, DLT gateways and utilisation of MongoDB), the transaction process flow, and DLT network nodes' interaction in the MHSC business context are the knowledge contribution of the HDSP.

Chapter 4: Distributed Ledger Technology Interoperability Framework (DIF)

This chapter elaborates on the DLT interoperability framework (DIF), a technology agnostic framework based on modern systems interoperability principles. It is a generic integration framework relevant to many business domains, and applicable to all DLT/BC technology platforms. The DIF can be applied to many architecture patterns popular in enterprise IT systems, including event-driven architecture, API gateway, ESB, distributed system architecture or Service mesh for Microservices architecture. Based on SOA principles, the API-driven design of the DIF enables the pluggable and loosely coupled integration of systems and their components. This service-focused design also facilitates integration with the IoT devices capable of handling the APIs or request and response messaging patterns, including synchronous and asynchronous communication.

The following section describes the DLT interoperability framework (DIF), its components and their interrelationship, and its suitability to the enterprise requirements. Section 4.2 elaborates on the interoperability principles based on which the DIF was conceptualised and modelled. Those multi-platform and technology agnostic principles and design guidelines were practically instantiated by developing the HDSP, as explained in Chapter 5. Finally, Section 4.3 suggests three interoperability architectures, based on the DIF, for the popular interoperability patterns, integration design and tools used by modern enterprises.

4.1 Interoperability Framework and its Components

This research developed a DLT interoperability framework (DIF), represented in Figure 4.1. The model and its components are based on the following:

- The interoperable solutions discussed in Section 2.6,
- The contemporary and popular integration approach adopted by organisations, is explained in Section 2.2, and
- interoperability principles are explained in Section 4.2.

The interoperability model and its components are explained in detail in this section. The DIF enables enterprises to integrate DLT systems with centralised IT systems. This framework improves the flow of data and information, message handling and technology agnostic interaction among diverse IT systems and their components. The framework supports the distributed environment, and its component design and interaction enable the integration of modern technologies such as DLT, IoT, Cloud computing, Big Data. This integration framework will allow the strength and capabilities of DLT (e.g., transparency, decentralisation, immutability) to be integrated with centralised systems. The DIF offers a blueprint for developing integrated application design, and enhancing the inter and intracompany business process. Its components, their interaction, and interrelationship are summarised in Figure 4.1 below.

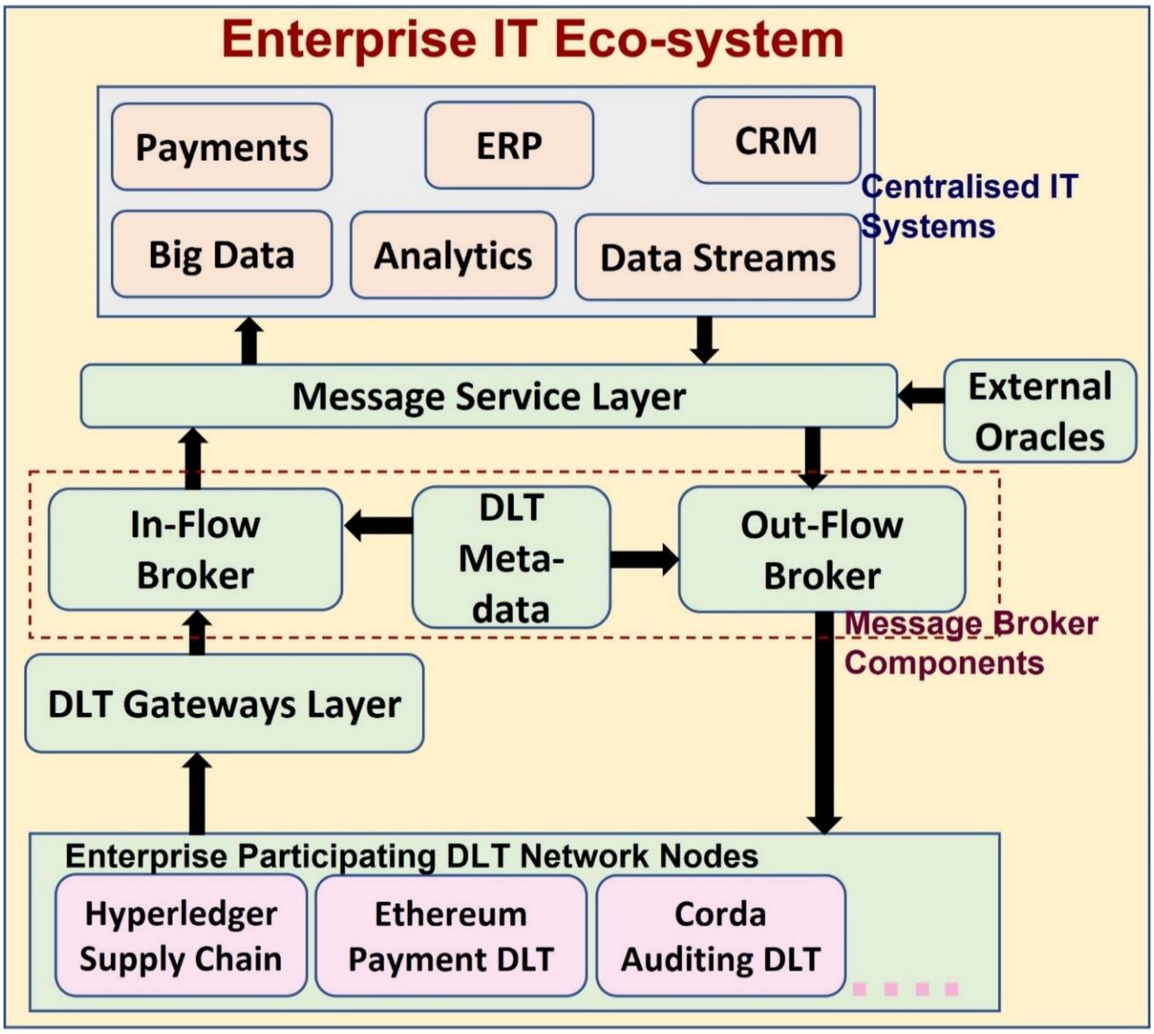


Figure 4.1. The distributed ledger technology interoperability framework (DIF) and its interacting components. The API-centric design of the framework enables the DIF component interactions via message flow; these components are represented in light green colour.

4.1.1 Enterprise Participating DLT Network Nodes Component

Purpose: This component aims to consolidate all the ledgers of different DLT networks into a single layer. This layer will enable a consistent interaction of all DLT ledgers with the other components of the framework.

Input: The transactions/events to be processed by DLT network ledgers.

Output: Transactions and events in DLT network and relevant events triggered internally by DLT network will be pushed to DLT Gateways.

Explanation: In the foreseeable future, an enterprise in a practical situation may participate in multiple DLT networks [63]. For example, a supply chain retail enterprise can be part of the Hyperledger supply chain network, participate in the Ethereum Auditing network, have an active node in the Corda Payment DLT network, and so on. Each DLT network would have multiple peers in the organisation. The bottom component of the DIF in Figure 4.1 displays the enterprise participating nodes in different DLT networks. DLT ledger of these networks will be stored on enterprise servers and managed by respective organisations, either in-house or on a Cloud. Although these DLT networks can function independently or in collaboration, the transactional data stored on the ledger is accessible and controlled by the organisation's IT infrastructure. Their ledger data can be exchanged and used by other systems and applications within the same enterprise.

The Enterprise Participating DLT Network Nodes component of DIF ensures that the architecture and design of multiple DLT networks in an enterprise remain simple. The data on DLT networks are passed to the enterprise’s centralised IT systems via DLT gateways (explained in Sub-section 4.1.2). The data from these diverse DLT networks are similar to any other organisational data and can be used according to the enterprise's business needs. Furthermore, a DLT network can trigger a business capability within another DLT network. For example, an HF supply chain network event triggers a message via DLT gateways and passes to the Messaging Service Layer. This layer can route the message to another DLT network via the Out-Flow Broker component (by reformatting the message) to invoke business capability, e.g., in Ethereum Payment DLT.

4.1.2 DLT Gateways Layer Component

Purpose: This layer comprises programs or modules to capture and process events triggered by DLT networks' Smart Contract (SC).

Input: The event messages pushed by DLT network SC.

Output: The event details are pushed to the In-Flow Broker for message transformation.

Explanation: In DIF, the Enterprise Participating DLT Network Nodes store the events or transaction data that DLT processes in the network's respective ledgers. When a record is stored in the ledger, an event is triggered and processed by DLT Gateway (the invoice-transaction.js component in HDSP, as explained in Chapter 5), allowing non-DLT centralised systems to access DLT transactions, events, and data. The DLT Gateways component can be considered an architecture layer of one or many interacting components instead of having a separate module for each DLT platform or network (in the HDSP, it is a single program that interacts with the HF ledger, as explained in Chapter 5). The API message format pushes the committed DLT data to the In-Flow Broker component, which converts the messages of different formats from various DLT networks into the desired format convenient for consumption by centralised IT systems.

The DLT Gateways is an essential layer of the DIF. Rezaei et al.'s [98] research have suggested four interoperability types: technical, syntactic, semantic, and organisational interoperability. The DLT Gateways component, Message Broker components and the Message Service Layer (MSL) facilitate these four types of interoperability. The gateway also enables the exchange of DLT data with the non-DLT systems, which can then be reformatted, massaged, or modified to be used by the centralised systems. This component can trigger the capability or business process of another DLT network via the Message Server Layer (explained in Sub-section 4.1.5) by using the API messages. The number of programs or modules within DLT Gateways component is determined by the number of DLT networks the enterprise is dealing with, and the technology platforms supporting those networks. This component can also capture performance values, metrics, and data volume exchanged among DLT and non-DLT systems based on the organization's needs.

4.1.3 DLT Metadata Component

Purpose: The DLT Metadata component holds DLT network static metadata to parameterise in-flow and out-flow message processing.

Input: Input is static DLT metadata such as Smart Contract, DLT network parameters, Routing parameters, Language supported, Consensus mechanism, and centralised IT systems parameters.

Output: This passes the parameters and meta-data to In-Flow and Out-Flow Broker components.

Explanation: This component can parameterise the process of message brokering. The message conversion or formatting logic can be triggered for the In-Flow Broker components based on the source DLT network of the message and message destination. Similarly, for Out-Flow Broker components, based on the destination DLT network and the source of the message, the appropriate message reformatting algorithm can be activated.

4.1.4 In-Flow Broker Component

Purpose: The In-Flow Broker component transforms messages destined for centralised IT systems.

Input: Input is the parameters from DLT Metadata and the incoming message from DLT Gateways.

Output: The output is the transformed message in standard or customised format, which is pushed to the Message Service Layer.

Explanation: The In-Flow Broker ("In-Flow" because the messages are flowing into centralised IT systems) component manages the messages and their transformation, significantly simplifying the architectural design of the integrated solution. It mitigates the risk of dependency or tight-coupling of DLT with the Web Server, business orchestration or front end components [70]. This component enables semantic and syntactic interoperability, as an essential capability suggested in Rezaei et al.'s research [98], in collaboration with DLT Gateways and the Message Service Layer component. In the HDSP, this is a dummy program; however, in a multi-DLT ecosystem, this layer can comprise one or multiple components. Although the input messages to the In-Flow Broker will be in different formats, this layer's features convert the message into the desired standard format. This transformation enables message routing to the destination system in a uniform way. It can also transform the message in a standard way to be routed to another DLT network (by the Message Service Layer) via the Out-Flow Broker (see Sub-section 4.1.6) to trigger an event or transaction in another DLT network. Message transformation tools such as Apache Camel can reformat the messages.

This layer can help keep the Messaging Service Layer (see Sub-section 4.1.5) manageable, lean, and maintain its performance by keeping the messaging routing, auditing, and logging separate from message transformation. The message brokering process can become heavy and CPU (central processing unit) intensive, and can become a bottleneck, impacting the other critical aspect of message integration. If the enterprise adopts a new DLT network, the majority of the impact would be absorbed by this layer, with minimal impact on DLT Gateways Layer and the Message Service Layer.

4.1.5 Message Service Layer Component

Purpose: This component is responsible for message routing, protocol transformation, message orchestration, encryption, and metric collection.

Input: Input is DLT messages from the In-Flow Broker and centralised IT systems.

Output: DLT message is sent to centralised systems, and messages from traditional systems are sent to the Out-Flow Broker component.

Explanation: The Message Service Layer can implement a parameterised routing logic to decide on the message destination system, enabling decoupled systems, as suggested by Zhang et al.'s research [10]. This component, along with the functionality of In-Flow and Out-Flow Broker components, helps manage the application and system contract as indicated in Abebe et al.'s research [9] architecture and message flow. The destination system, routing mechanism, and routing protocol can be parameterised in the database, XML, or JSON-based data to map the correct source of DLT network to the destination (centralised) IT system. This layer can support communication through different protocols, reasonable messaging styles, and storing and forwarding messages to multiple systems. The parameterised encryption logic can be implemented if the message has to be routed to the internal and external partner or third party system, or across an organisation firewall.

The component (inside the Message Service Layer) to control the message flow from the centralised system to the Out-Flow Broker (see Sub-section 4.1.6) is expected to be comparatively simple. It does not need to route the message, but just adds the routing parameters to delegate the routing responsibility to the Out-Flow Broker component. This component can transform the message protocol format based on the destination DLT network the message is supposed to be routed to. This layer can also provide correct authentication to the message based on the access requirement of the destination DLT network. Message aggregation and message splitting can be executed to merge multiple messages or split them into multiple messages before being routed to the destination DLT network. Based on the architecture style, it can function in additional roles; for example, in event-driven architecture, it can act as an Event Listener.

This component can manage the quality of service requirements such as performance criteria, transaction management, and exception handling. It can record, capture, monitor, and track service invocation and message routing activities. The security requirements of messages can also be managed in this layer. External Oracles can feed the live external data (e.g., currency exchange rates, commodity prices) to the incoming and outgoing messages, including the transactions from DLT or messages passed to DLT networks.

4.1.6 Out-Flow Broker Component

Purpose: To transform the centralised systems message and route it to the destination DLT network.

Input: Input is the parameters from DLT Metadata and incoming messages from the centralised system in a standard format.

Output: The transformed message is routed to the destination DLT network.

Explanation: The Out-Flow Broker ("Out-Flow" because the messages flow out from the centralised IT systems) transforms the messages from the centralised IT systems into the format required for the destination DLT network. This mitigates the risk of dependency or tight coupling of DLT with the Web Server, business orchestration, or front-end components [70]. This component supports semantic and syntactic interoperability, as an essential capability suggested by Rezaei et al.'s research [98], in collaboration with DLT Gateways and the Message Service Layer (MSL) component. This layer conducts data positional mapping or semantic mapping, and routes the message to the destination DLT network based on the header or configuration set-up by the Message Service Layer (see Sub-section 4.1.5). It also supports service virtualisation so that changes to the end-points (DLT network) can occur without impacting the service providers and service consumers.

The three components in Figure 4.1, In-Flow and Out-Flow Broker and DLT Metadata, comprise the Message Broker Components. The In-Flow Broker, as explained in Section 4.1.4, supports the Gateways Layer to format the incoming messages from the source DLT platform in the standard format for consumption by any API or message enabled device or system/application. The Out-Flow Broker component uses DLT Metadata to convert the messages from centralised IT systems or devices into the specific format for the message destination DLT platform. In many organisations, this layer can become a bottleneck in the enterprise message flow due to the complex and intensive processing logic of converting messages from and to many formats [18]. This challenge can be addressed if the centralised IT systems and devices can send the messages in a canonical form (or two formats, if IoT devices are involved) to significantly reduce the message formatting logic.

4.2 Foundational Principles for DIF

The DIF is designed according to the interoperability principles replicated and extended from available DLT solutions and the interoperability requirements of modern enterprises. Enterprises use various architectural styles, design and their combinations, including layered architecture, Microservice architecture, event-driven architecture, and distributed architecture to fulfil their business requirements [70]. Furthermore, the enterprise architecture approach, and the solution design and implementation can differ based on each business's unique needs and requirements, even within the same business domain. The foundational principles of the interoperability framework are flexible and accommodate these diverse and sometimes contradicting expectations by maintaining its focus on interoperability.

The DIF is based on interoperability principles replicated and extended (see Section 2.8) from available interoperability solutions (see Section 2.5), and the interoperability requirements of contemporary organisations (see Section 2.2) as explained next.

1. **Enterprise Centric Framework**: Enterprises or organisations are the centre of the DLT interoperability framework (DIF). Enterprises undertake prototyping of DLT-based solutions, invest money, time, and their workforce, to implement and

operationalise DLT-based solutions. Enterprises can be part of multiple DLT networks, which can operate independently or communicate among themselves (inter-BC interoperability). DLT network solution must synergize with centralised IT capability to benefit an organisation and business unit with better and more efficient business processes and capabilities, and swift decision-making. The DLT interoperability framework (DIF) visualises DLT solution capability as any other technical capability, enabling its integration with non-DLT systems.

2. **Core Business Functionality in DLT**: DLT network solution is challenging to implement and operationalise compared to centralised IT systems in terms of organisation collaboration, development and implementation cost, human resource skill set, and the need to maintain the desired level of IT operations [4]. Hence, a DLT solution have to be lean, lightweight, and cost-effective, including limited and carefully chosen functionality in DLT network. This is beyond the control of a single organisation in a multi-organisation DLT network. However, enterprises is expected to keep only the business process and capability that directly enhances the trust, transparency, and distributed processing amongst the network participants. The rest of the functionality and business capability is expected to continue running on centralised IT systems. For example, in the HDSP explained in the next chapter, the customer (a honey consumer) requirement is managed outside DLT network because it is not the core functionality of the supply chain process.
3. **API-driven Design**: Contemporary enterprise IT systems are based on distributed architecture using SOA patterns [99]. The services and business capabilities are distributed and can be developed on various technology platforms, multiple programming languages, tools, and development frameworks, starting with the monolithic Mainframe systems in the early 1970s, Microservices more recently, and everything in between. These capabilities are exposed as an API request and response messaging format to integrate the core business capabilities to create more complex and sophisticated business processes. The DIF uses the same API-driven design to integrate DLT solutions with the implemented IT capability.
4. **Selective Business Data in DLT Ledgers**: The business data and information that directly contributes to improving the transparency and trust between the network participants have to be included in DLT data structure. The centralised IT systems can continue to manage the rest of the data.

5. **Store Hashing or Signature in BC**: Many business domains, e.g., the supply chain business process, involve extensive data management and significant document handling. These documents and large data are vulnerable to forgery and fraud; to avoid this, a mutually agreed algorithm signature or hash value have to be created for each document and stored in DLT Ledger. Before processing or relying on the document, the system must verify the ledger's signature to establish document authenticity. To store documents or massive amounts of data, organisations are expected to continue using specialised centralised tools such as ECM (Enterprise Content Management), document management systems and similar.
6. **Cloud-Native Framework**: Almost all organisations have either adopted or are planning to adopt Cloud technology [100]. The interoperability framework is Cloud-native, can be deployed on any Cloud environment, and can integrate among Cloud networks using the API-driven design.

4.3 Integration Architectures based on DIF

This section applies the DLT interoperability framework (DIF) to design integration architectures. The DIF is applied to popular interoperability architecture and solution patterns widely deployed by modern enterprises. The DIF is elaborated in Section 4.1, and the framework's foundational principles are explained in Section 4.2. These principles and framework formulate the appropriate architecture model and design pattern to design and develop interoperability architecture. This DIF based integration design enables the adaptability of the framework among broader business domains, utility across the various architecture patterns, and applicability to the diverse IT system landscape.

4.3.1 Interoperability Architecture using API Gateways, ESB, and Service Mesh

In the last decade, most IT systems evolved as applications with a web-based front-end layer and business services and processes supported by distributed back-end systems and applications. As the IT systems became distributed, and organisations adopted SOA, the services and business capabilities spread across the globe [30]. These capabilities were based on various architecture patterns, developed in many programming languages on diverse operating systems and platforms [85]. These services, functionalities, programs, and system components, were integrated to exchange the data, reuse the business capabilities, and create new and improved business processes. Enterprises achieve such a complex level of interoperability by using (Section 2.2):

- application programming interface (API) Gateways [35] (e.g., Apigee, Kong, Del Boomi, Akana, Cloud APIs like Azure, AWS, Oracle)
- enterprise service bus [24] (Mule ESB, IBM Websphere ESB, Microsoft Biz Talk, Oracle ESB)
- Service Mesh tools for integrating Microservices [36] (e.g., Istio, Conduit, Envoy, AWS App Mesh).

These capabilities can be used standalone or in combination, if the service and integration landscape of the enterprise is complex, global, distributed, and diverse.

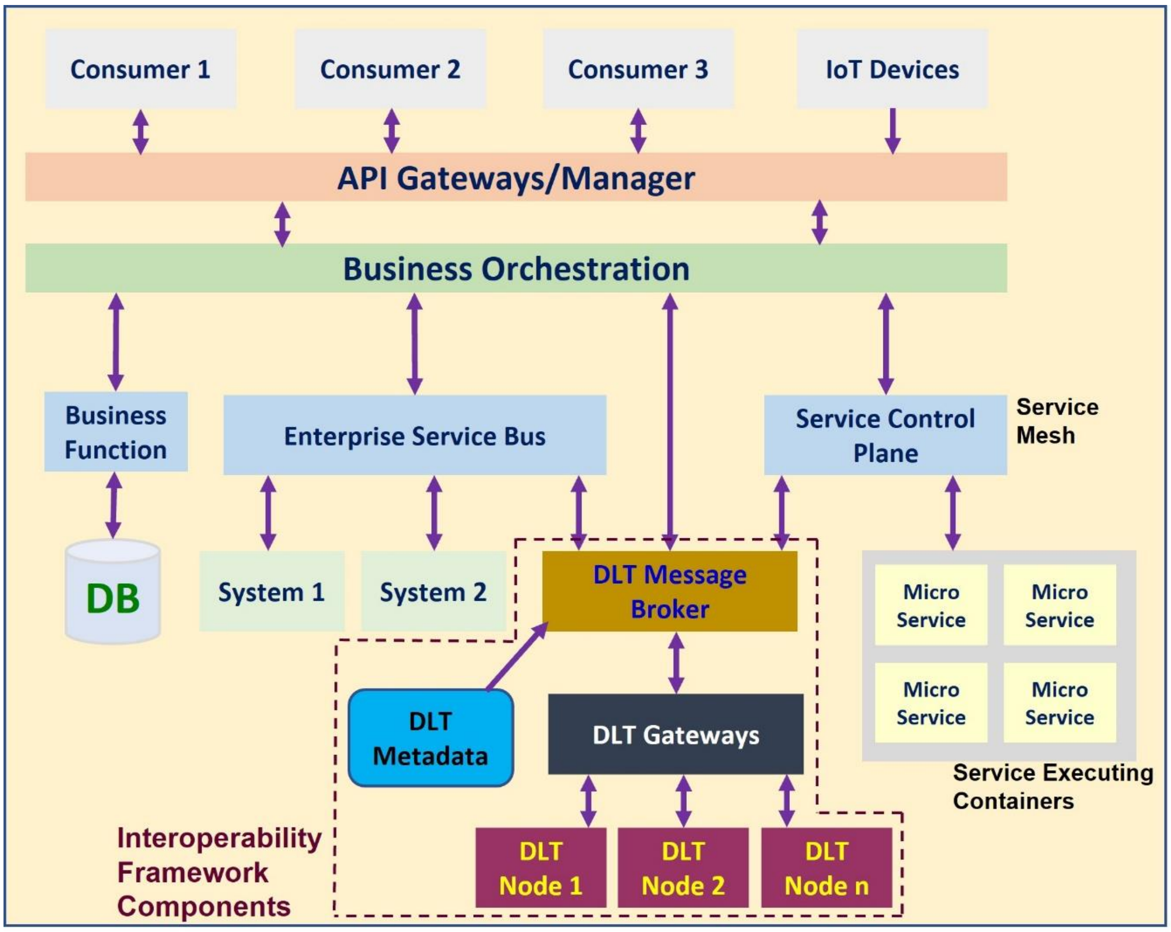


Figure 4.2. Integration architecture, based on DIF, enabling interoperability among DLT and non-DLT systems using API gateways, ESB, and Service Mesh (Service Control Plane).

The design presented in Figure 4.2 integrates the DIF components into the IT ecosystem to integrate DLT solutions with the centralised IT systems. The DIF design is API-driven (based on the 3rd principle in Section 4.2), matching the API gateways' principles to interact within and outside the organisation. The architecture also provides the capability of designing the IT ecosystem and customer-centric digital products to encourage the integration of current and new APIs (REST, SOAP). This API'sation supports both approaches: 1) a bottom-up or asset utilisation by making the published APIs available to API Gateways, and 2) a top-down or usage first approach in which the APIs are defined to meet needs and docked to an existent back-end business system. In each approach, the message exchanged with Business Orchestration and API Gateways will interact with the Message Broker to transform into a consumable format. The DLT Metadata feed can parameterise to automate the message handling functionality in both directions.

For practical business purposes, an enterprise will participate in the many DLT networks. These different ledgers have multiple data formats, and their semantics and syntax are different. The DLT Message Broker component addresses this problem or gap, by transforming the messages into the desired or standard format for interacting with the ESB. The messages received from DLT message broker are treated like any other message in the system by the ESB.

A Microservice or Service Mesh approach restructures the monolithic systems into many fine or coarse-grained, autonomous, scalable, minimal, resilient, and integrable services [36]. This microservice approach helps improve the modularity and agility of applications and systems [17]. Also, Microservice technology uses a framework (e.g., Spring Boot) and containers (e.g., Docker and Kubernetes) to build the business interface, which can be part of the Business Orchestration module. Thus, the service mesh is focused on service-to-service interaction that, along with DLT Message Broker component, enables the API data to push into DLT for transactional purposes, or consume a DLT transaction using a Microservice via a Service Control Plane. Service Mesh can facilitate and control the sharing of the application data (including DLT ledger data). Service Mesh does not implement integration logic; each Microservice must implement the integration logic by becoming an intelligent endpoint, and similarly for DLTs, this logic can be taken care of by DLT gateways and DLT message broker component.

Figure 4.2 also illustrates a flexible architecture showing the capability of the DIF to work in synergy with current enterprise interoperability practices, tools, and designs. Based on the component design of the systems, one DLT Message Broker component can integrate with the APIs or messages coming straight from API Gateways or the Business orchestration layer.

Another module or sub-component in the message broker component can seamlessly interact with the ESB. The design and implementation of this sub-component can depend on the enterprise ESB ecosystem, the number of systems it interacts with, the scope of functionality included in the ESB (e.g., routing, messaging transformation, messaging brokering parameters), and its technology footprint. For the service mesh configuration, the service control plane can have a dedicated component(s) to deal with DLT related messages, a component for incoming and outgoing DLT API messages, or two separate components. If DLT Message Broker components are too complex and heavy, some non-core functionality can be moved to DLT gateways to meet business, architecture, and performance expectations.

4.3.2 Event-driven Distributed Architecture

Event-driven Distributed Architecture (EDA) is a software architecture paradigm and system design pattern to develop the IT systems to detect, react, and consume business events [38]. Event-driven distributed architecture enables horizontal scaling of the systems in a distributed architecture, making them more failure resistant by making multiple copies of the application available among parallel systems. Event-driven distributed architecture promotes loose-coupling architecture because events are not aware of the consequence of their trigger and destination processing system. Many techniques and tools are available to implement EDA in enterprises, e.g., Apache Flink, Apache Kafka, AWS Kinesis, Rabbit MQ. Apache Kafka (AK) is one of the most popular industry tools used to implement EDA [39].

The grey portion of Figure 4.3 represents the pub/sub (public and subscribe) integration pattern for event-driven architecture for non-DLT based systems. This part of the figure consists of REST (Representational State Transfer), MQTT (Message Queue Telemetry Transport) broker, Kafka Clients and Connector as incoming gates through which inflowing messages flow to the Kafka cluster. These messages are routed to specific AK topics, subscribed to and accessed by Micro Services, Systems and Applications, data consumers, and big data or Cloud storage input. Zookeeper makes AK's configuration simple and efficient, and the schema registry provides the meta-data for the incoming messages in the Kafka cluster. Furthermore, the message sequence in AK is fixed once a message is received in a partitioned cluster, which maintains a permanent sequence of messages, similar to a sequence of transactions or messages in a DLT ledger.

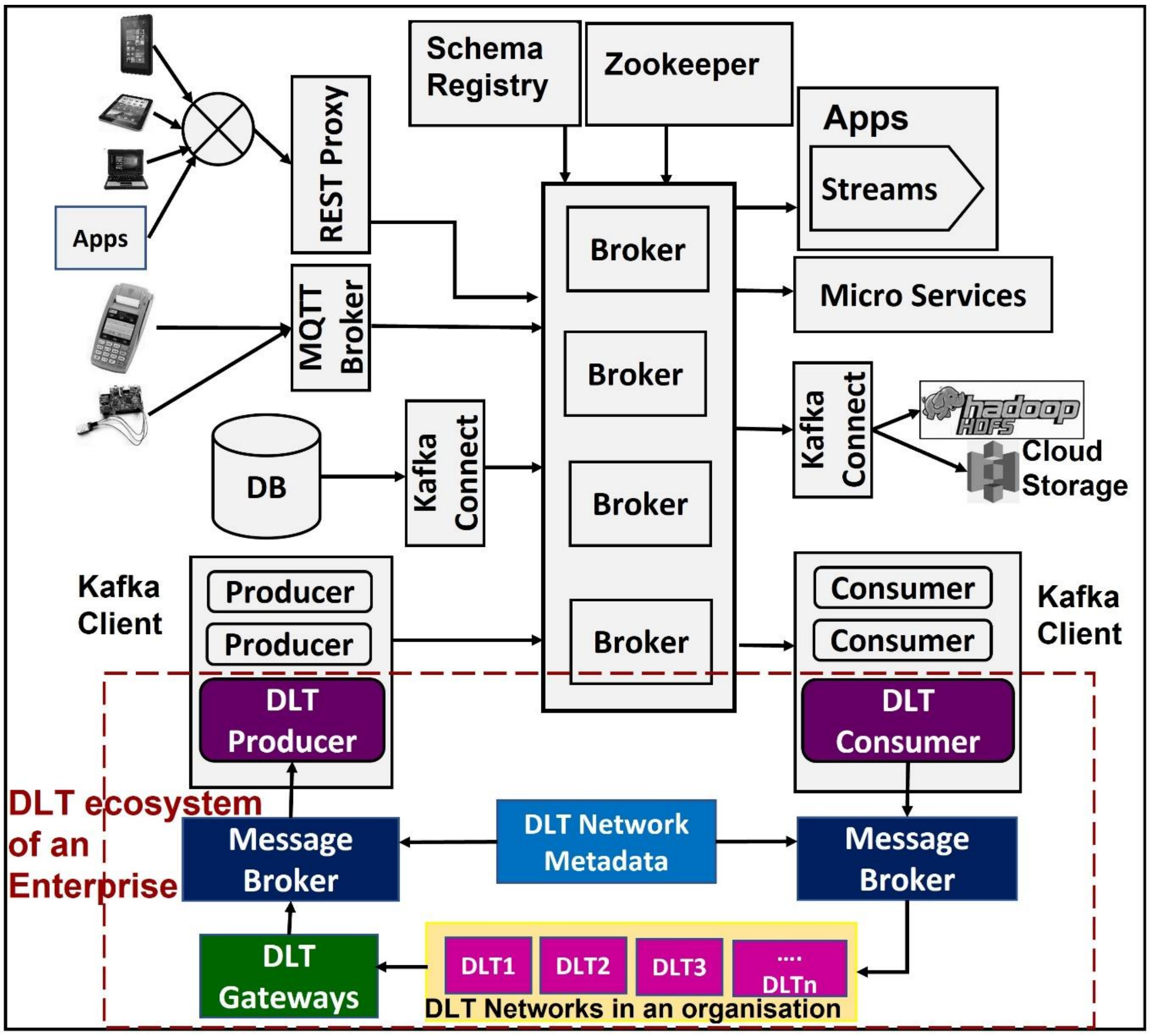


Figure 4.3. Apache Kafka DLT interoperability Architecture displays the integration of an organisation's multi-DLT network with its centralised IT systems. Apache Kafka is a broker receiving incoming messages from diverse protocols and their associated devices, including DLT networks. These incoming messages are routed to specific AK topics as output, and subscribed to by both DLT and non-DLT systems for further processing.

The AK DLT interoperability architecture in Figure 4.3 displays the integration of DLT solutions in a typical distributed event-driven architecture. This diagram includes the coloured DLT components and the centralised non-DLT grey components; the integration pattern is consistent for DLT and non-DLT systems. The DLT Gateways component extracts the events or transactions from DLT networks. Message Broker act as an interface, bridging the message formatting, and the syntactic and semantic gap between DLT and centralised systems. The Message Broker can also convert the message protocol, and grant enterprise users access for DLT system. The DLT Producers receive DLT transaction data from the Message Broker in a standard format to send the records (i.e., event messages) to an AK topic based on the record key. These messages can be consumed by enterprise non-DLT systems such as ERP, CRM, and the DLT Consumer. The DLT Consumer is responsible for delivering an event record (e.g., a sell order generated by the enterprise ERP system, payment fulfilled by a payment system, or an event from a DLT network) in a standard messaging format to the Message Broker. The Message Broker re-formatted this message based on the destination DLT network, using the DLT Network Metadata. The integration architecture is flexible enough to design one or multiple DLT Producers and DLT Consumers, based on DLT type, technology, and business functionality of the network, batch processing or high-speed online transaction processing, and similar considerations. The component design is flexible enough to offload or exchange the capability among DLT Gateways, Message broker, and DLT Producers and Consumers to format and transmit data between the systems, determine the access points for DLT and non-DLT systems, and capture event metrics.

Distributed ledger technology and AK share affinity characteristics; for example, both share the concept of an immutable append-only log. Apache Kafka is an immutable, ordered sequence of records continually added to a structured commit log, and DLT is a continuously growing list of sequentially linked and secured blocks of records. Apache Kafka provides high throughput and horizontal scalability, and DLT excels in the secured order and structure of transactions and events blocks. By integrating these complementing capabilities, synergy can be created from combining these technology solutions. Many organisations use AK in their event-driven distributed enterprise architecture to integrate diverse systems [39]. These same blue-chip organisations are investing significantly in DLT solutions, so continuing to use AK as an integration pattern seems to be a logical choice [6].

Apache Kafka enables interoperability by supporting the system transaction processing capability, which DLT integration components illustrated in Figure 4.3 may extend to process and integrate DLT transactions along with non-DLT transactions. Apache Kafka provides in-stream non-DLT data processing and can produce integrated data processing by merging the incoming DLT transactions on the distributed ledger with implemented systems. Apache Kafka has a flexible and scalable capability for capturing and ingesting the high volume stream of non-DLT transactions. This capability can extend to integrating a large volume of DLT ledger transactions into big data and data lakes such as Hadoop and Spark, stored in RDBMS, Cassandra, or AWS S3 Cloud storage [101]. The log aggregation capability of AK enables the cumulative and integrated logging of all DLT and non-DLT transactions. Both AK and DLT have a unique ability of transaction order preservation, which allows for consistent processing of high volume fixed sequence transactions. Furthermore, AK provides the capability to transform, process, and load the data, which is well suited to the requirements of DLT integration, where the vast number of transactions from different types of ledgers need consistent transformation, processing, and loading into current systems and storage.

Under a heavy load, AK provides outstanding performance; its flexible architecture provides scalability, which can be helpful when there is a sudden spike in transaction flow from DLT nodes or significant data flows from IoT devices. Apache Kafka provides sequential access and immutable commit log, ensuring that immutable transactions in DLT are processed, and their logs are maintained in AK. The capability of AK to process the transaction in sequence, which is one of the primary reasons for its immense processing performance, is of enormous benefit for processing the sequential transactions in DLT ledger. The DLT Smart Contracts can produce business events (pre-defined or custom events) listened to by DLT or non-DLT applications to take action. The consumer uses these events for business processing, which can be further delivered as events using AK; Hyperledger Fabric provided AK-based orderer implementations for a fault-tolerant production environment, which can be immensely helpful in processing such events.

4.3.3 Generic Interoperability Architecture

This section discusses a generic and flexible interoperability architecture an enterprise can consider implementing if it does not want to implement the prevailing integration design patterns. Using technology, tools, the IT landscape, and systems expertise, an organisation can design and execute its customised interoperability architecture. Many technical components exist in today's business world, such as payment engines, ERP (Enterprise Resource Planning) systems, transaction processors, decision-making applications, and intelligent autonomous systems. These are expected to expose services and consume external services using industry-established protocols (e.g., REST) and APIs. The generic architecture have to integrate the current IT capabilities and DLT network nodes; enterprises are supposed to carefully consider the network participating nodes and the current system landscape requiring interoperability.

The system design of the HDSP is discussed in Section 5.1 in the next chapter and illustrated in the left section of Figure 4.4. It features a Node.js based API layer to connect RESTful API services using an HF SDK (Software Development Kit) library, which provides a powerful, service-oriented, and easy-to-use capability for injecting the transactions/events into the HF network [102]. Supply chain network participants have the flexibility to use a variety of devices and messaging protocols. For example, a manuka honey distributor can use IoT devices and related protocols to push jar information into the network when the warehouse receives the manuka honey jar boxes. The honey producer can then bulk transfer the jar information from its database (DB) (e.g., Access DB or MS SQL) into the HF network. However, a honey retailer might also decide to make the honey jar entries into its ERP/inventory systems first and then provide the API feed to the HF DLT network.

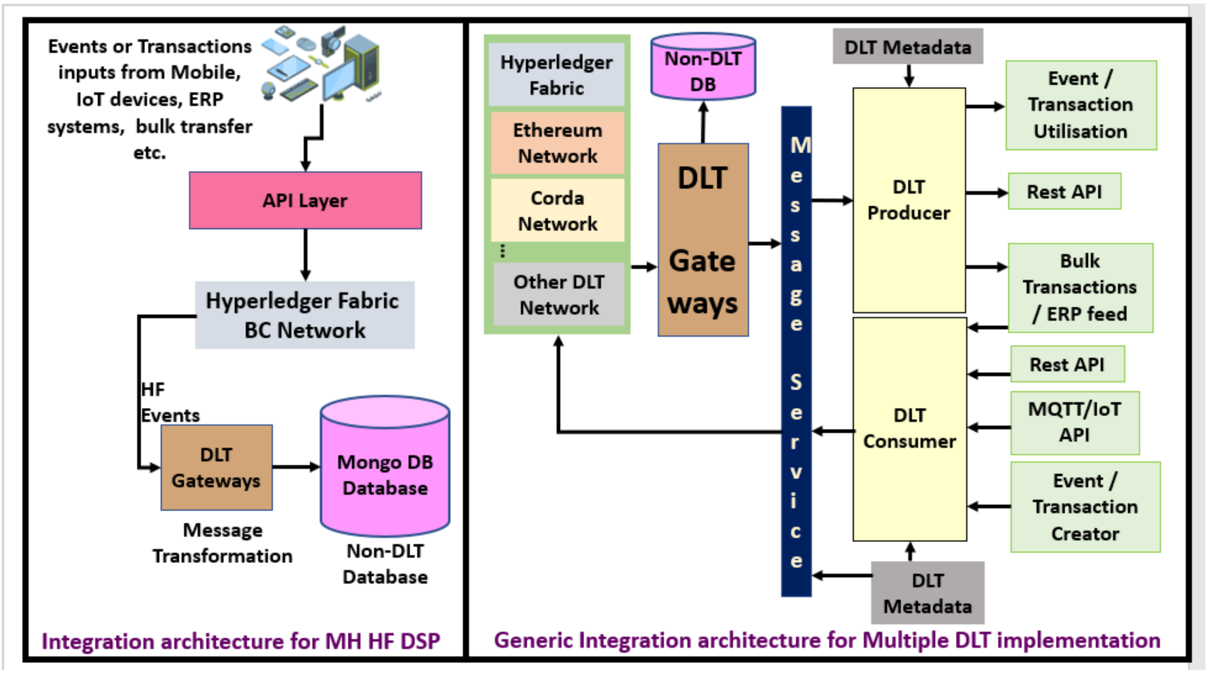

Events or Transactions inputs from Mobile, IoT devices, ERP systems, bulk transfer etc.
API Layer
Hyperledger Fabric BC Network
HF Events
DLT Gateways
Mongo DB Database
Message Transformation
Non-DLT Database
Integration architecture for MH HF DSP
Hyperledger Fabric
Ethereum Network
Corda Network
Other DLT Network
Non-DLT DB
DLT Gate ways
Message Service
DLT Metadata
DLT Producer
DLT Consumer
Event / Transaction Utilisation
Rest API
Bulk Transactions / ERP feed
Rest API
MQTT/IoT API
Event / Transaction Creator
DLT Metadata
Generic Integration architecture for Multiple DLT implementation

Figure 4.4. Generic interoperability architecture. The left section demonstrates the design of HDSP, which is enhanced and extrapolated in the right section of the diagram for a business organisation participating in a multiple DLT network.

The API layer in the left section of Figure 4.4 is flexible to push the transactions/events from various devices such as mobile telephones, IoT, IT systems by network peers. These non-DLT data are processed by CC (Chain Code, also called “Smart Contract SC”) to store in the HF ledger and Level-DB. For every transaction within DLT, an HF event will be triggered by SubEvent() function to receive two parameters: the event's name, and the payload in bytes to input in JSON format. When the transaction is completed successfully by CC, the event is sent to the SDK to unmarshal the payload and complete the required business processing. This event may create various transactions and associated payloads to be sent to the SDK after the block is committed. The SDK listens to the event, accepts it, unpacks the payload, and decides what to do, such as sending an email, writing to an external system, initiating another process, or sending a notification. A simple one line of code can efficiently integrate the HF with a range of systems, and the DLT gateways further process events, converting DLT data to non-DLT data for storage in the MongoDB database.

For businesses participating in multiple DLT networks, the right section of Figure 4.4 suggests the integration design based on the design of the HDSP implementation shown in the left section of the figure. The essential components of the integration design are DLT gateways for extracting DLT data entities from the multiple DLT networks. Another component, Message Service, bridges the differences in technology protocol between DLT and non-DLT systems. It can also grant enterprise users access to a DLT system, transmit external data (e.g., transmit a near field communication (NFC) tag detection event to DLT network), transmit DLT data to an external system (e.g., approved purchase order placed by an external organisation on DLT network) and enable the data auditing transferred to and from DLT networks. Irrespective of the originating network, DLT Producers receive DLT transaction data from DLT Gateway via Message Service. The DLT producer functionality is similar to the In-Flow Broker in DIF in that it transforms the message based on the originating DLT network and the message destination system. The DLT Consumer functionality is similar to that of the Out-Flow Broker in that it delivers the message to the Message Service after appropriate transformation. In addition to its other critical responsibility, the Message Service layer can also undertake message transformation, depending on architecture requirements. The feed of DLT Metadata to the Message Service layer can parameterise this responsibility for incoming and outgoing messages. The DLT gateway can also push the ledger data into single or multiple non-DLT databases (e.g., MongoDB in HDSP) for further data consumption by systems like Big Data.

Chapter 5: Hyperledger Fabric DLT Solution Prototype (HDSP)

This chapter explains the Hyperledger Fabric 1.4 (HF) DLT solution prototype (HDSP) developed using the DLT interoperability framework (Section 4.1). The HDSP utilises the DIF interoperability principles (Section 4.2) to design the MHSC solution. The solution contributes to addressing the research gap by building a transparent and trusted supply chain network, integrating DLT solution with non-DLT centralised IT systems. The solution also validates that the DIF and its interoperability principles are implementable for creating an integrated solution.

An extensive survey of DLT interoperability by Belchior et al. [7] suggested that organisations apply DLT to use cases with only one DLT platform. There is considerable interest in deploying multiple DLTs in a relevant business context [63]; however, examples of multiple DLT platform implementation in an organisation are rare. It is reasonable to assume that enterprises will implement a DLT solution and then add more DLT networks as their experience and capability in implementing DLT solutions grow. The HDSP explained in this chapter can be considered the first implementation; hence only one DLT network is considered in DLT solution prototype instantiation. Furthermore, implementing a multi-DLT network solution needs expertise in multiple DLT platforms; the IT resource requirements are significantly higher (e.g., Cloud servers, network, and peers), and the timelines to implement the solution are substantially longer. This research has taken a balanced approach by considering the resources available, such as technical expertise, finance, timelines, and previous research feedback, and decided to implement a DLT solution prototype with a single DLT platform. The platform evaluation in Section 5.2.2 and Section 5.2.1 guided the decision to choose Hyperledger Fabric 1.4 as DLT platform and the MHSC use case, respectively.

This chapter presents the HDSP and validates the DIF and its principles by demonstrating the integrated DLT solution with non-DLT technologies and tools. Section 5.1 describes the solution architecture and design of the Hyperledger Fabric (HF) based HDSP for the MHSC use case. It explains the message and data flow among DLT and non-DLT systems and presents the HDSP solution characteristics. Section 5.2 describes the rationale for choosing the MHSC use case and selecting a DLT platform to implement the solution prototype. Section 5.3 delves into the HF network's prototype development, HF components, node interactions, and event handling. Finally, the knowledge gathered during the prototype's design, development, and implementation is documented in Section 5.4. As expected in the DSR methodology, this research knowledge contribution (Section 5.4) can be used by practitioners to build future interoperable DLT solutions.

5.1 HDSP Architecture and Solution Design

This research uses an MHSC use case to build an interoperable HDSP based on the DIF. Although the solution establishes trust and transparency amongst the network participants/nodes, the study focuses on demonstrating the interoperability of DLT-based solutions with centralised IT systems. The solution architecture, end-to-end transaction and data flow, and the solution characteristics of the HDSP are explained in this section.

5.1.1 Solution Architecture

This sub-section elaborates on the solution architecture and component design of the HF-based solution for an MHSC business process. The HDSP involves the interaction between the supply chain network's three stakeholders (honey producer, distributor, and retailer are the most important stakeholders in the honey supply chain). The customer user interface is provided to display the honey product quality and provenance information. The solution can augment the supply chain business capability by using DLT-based solutions to create or improve the business process by integrating the HDSP with centralised systems.

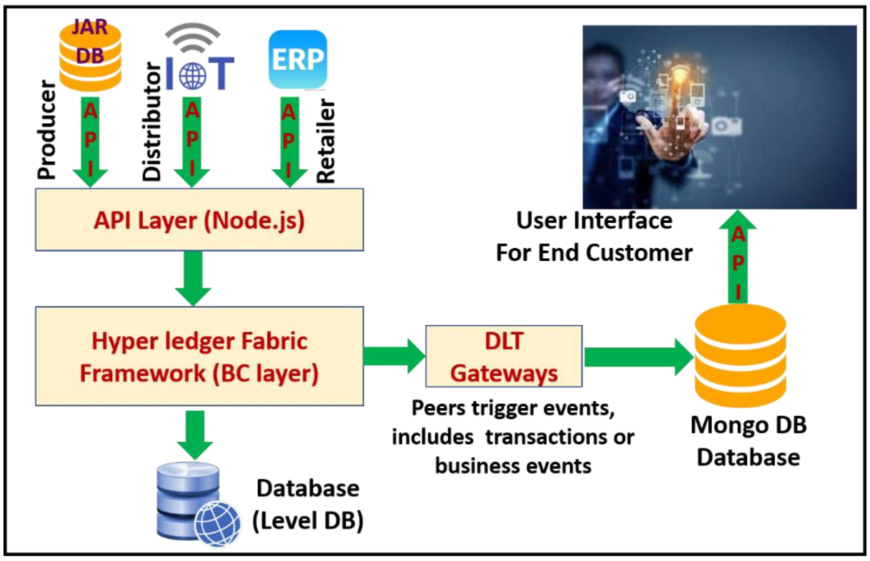

JAR DB
IoT
ERP
Producer
Distributor
Retailer
API
API
API
API Layer (Node.js)
Hyper ledger Fabric Framework (BC layer)
DLT Gateways
Peers trigger events, includes transactions or business events
Database (Level DB)
User Interface For End Customer
API
Mongo DB Database

Figure 5.1. Solution Architecture displaying the end-to-end message flow of transactions. The figure also shows the interaction points of system (DLT and non-DLT) components and the message flow across them.

The end-to-end flow of transactions across DLT and centralised system components in the HDSP are illustrated in Figure 5.1. The transaction can trigger from any system or device capable of generating messages. The message is saved in the ledger and LevelDB of the HF network via the API Layer, further getting pushed into the MongoDB database by the DLT Gateways layer. The data in MongoDB are accessible to any user interface.

The honey producer can send a pre-formatted message containing jar information from the database on its local system. The transaction message sends the transaction to the HF DLT layer via the Node.js API layer (*app.js* component). The transaction is approved, endorsed, and committed into the HDSP ledger copy for all three HF nodes/peers. The data from the HF network are stored in the Level-DB database, offering the data query functionality to query DLT data. Every committed transaction is passed to DLT gateways (component invoice-transaction.js), from where the data are stored in the MongoDB database. All the transactions committed into the HF network flow to the MongoDB via DLT gateways will be inserted or updated. Data are inserted if the producer adds new jar information; data from other network participants are updated in the MongoDB based on the unique jar identity (ID) or digital jar identity.

A similar process is followed when the honey distributor or retailer inserts the HF network data, which flows from the API layer to the HDSP network and is committed to the ledger of all the network peers. The HF transactions get pushed to the DLT gateways to update the jar record (the honey producer has already inserted the jar information) into the MongoDB database. The customer user interface (UI) is provided for viewing the jar details for the end-user or honey customer. The AngularJS based UI accesses the MongoDB database via the API to fetch the information after the jar ID is entered or scanned by the customer.

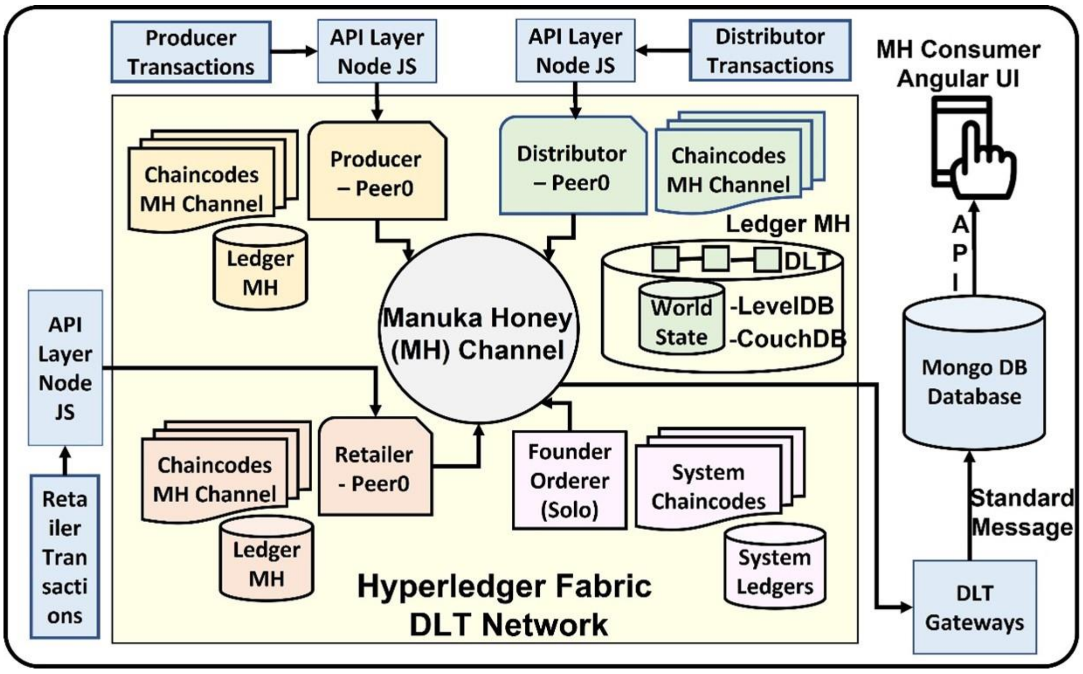


Figure 5.2. Solution Architecture of the HDSP technical prototype instantiation. The figure emphasises the internal components of HF, their interaction with each other and non-DLT components of the system.

The components of the HDSP are presented in Figure 5.2. The component in the centre is the manuka honey channel (MH channel), three peer nodes (honey producer, distributor, and retailer), and a transaction orderer is connected to this MH channel. Multiple channels can be defined in HF depending on business requirements (e.g., another channel can be added to share sensitive data such as honey's cost); the HDSP technical prototype uses a channel to share the data. Every peer node deploys, instantiates, and executes the Chaincode (Smart Contract SC); the functionality of the SC in the HDSP is kept simple to enable the exchange of transactions across the peer nodes and non-DLT systems. All three HF network nodes store an identical local copy of the ledger. As represented by the distributor peer in Figure 5.2, internally, the HF ledger (on each node) consists of two related parts: DLT ledger and the world state, as follows:

1. **DLT ledger** logs all transaction records executing in the HDSP. Transactions are stored inside blocks and appended in a fixed sequence forming DLT ledger, and it cannot modify its data structure; it is immutable.

2. **The World State** is a database holding the current values of a set of ledger states. It makes it easy for an SC or other program to directly access the current state value instead of calculating it by traversing the complete transaction log from a DLT ledger.

The world state can be stored in either the LevelDB or CouchDB; the HDSP uses the LevelDB.

Another vital component in the HF DLT Network in Figure 5.2 is Founder Orderer, which forms an ordering service to undertake the transaction ordering for the transactions or events to be included in the blocks. It also enforces access control for channels, managing who can read, write, and configure the data. It consists of a system Chaincode (CC) that defines a low-level program code corresponding to domain-independent system interactions. The solution prototype uses the solo orderer to order the transactions into blocks; the production implementation is expected to use fault-tolerant Raft or Apache Kafka (AK) to order the transactions.

The components outside the HF DLT Network section in Figure 5.1 are centralised (non-DLT) components. The API Layer Node JS enables the seamless message flow from the messaging enabled device to the HF network. For simplicity, three API Layer Node JS components (equivalent to Message Service Layer in DIF, explained in Section 4.1.5) are shown for every node; in implementation, it is a single component with which the three nodes interact. The DLT Gateways (equivalent to the DLT Gateways Layer in the DIF, explained in Section 4.1.2) component uses the event function to push the transactions from the MH channel into the MongoDB database in a standard (canonical) message format. The honey data from the MongoDB database are accessed by API messages to display on the customer web user interface (manuka honey consumer AngularJS UI component). The HDSP uses a single DLT platform; for such solutions, the In-Flow Broker or Out-Flow Broker components of the DIF (see Sub-sections 4.1.4 and 4.1.6) can be considered dummy components that need to be developed only for multi-DLT platform integration implementation.

5.1.2 Transaction and Data Flow

The HDSP end-to-end transaction flow from DLT network nodes to the HF Ledger, MongoDB database, and customer UI is represented in Figure 5.3. The transaction messages flow into the HF network in a standard format (due to which the In-Flow and Out-Flow Broker components from the DIF are not required in this implementation) and include transactions executed by the network peers. For example, for a particular jar, the producer injects the API message, Transaction 1, into the HF network. This transaction and other transactions are executed simultaneously to form an HF block in the network and get added to HF Ledger. For the same jar, the distributor injects Transaction 2 into the HF network once the distributor has received the jar; the retailer pushes Transaction 3 after it receives the jar. These three transactions can be part of the same or other blocks, most likely in different blocks, because transactions from three nodes for the same jar will trigger at different times.

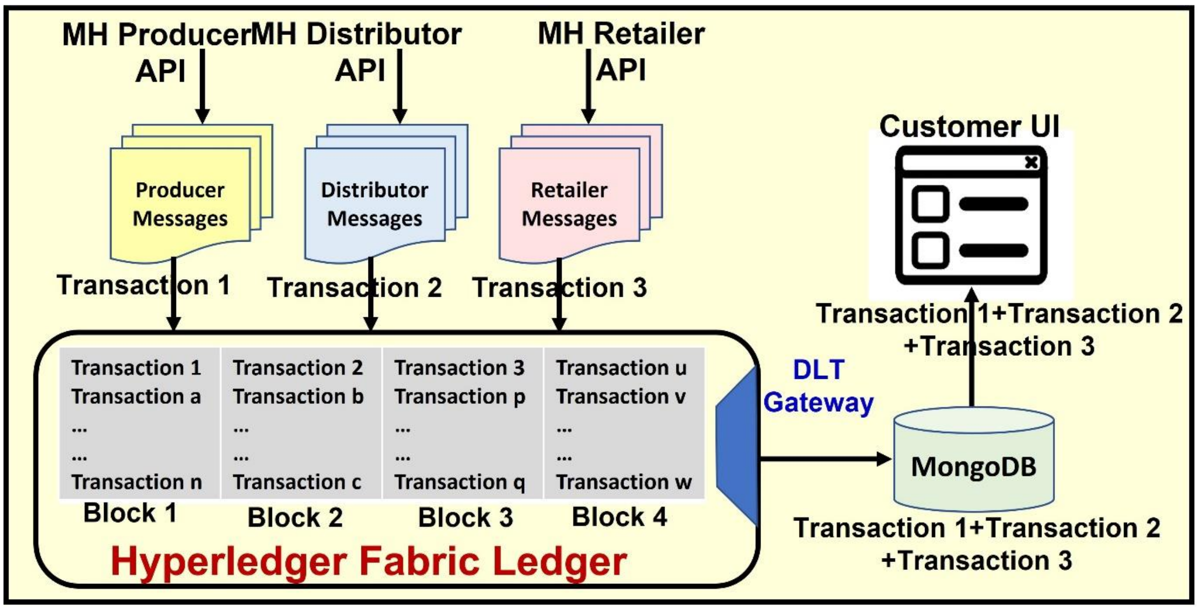

MH Producer API
MH Distributor API
MH Retailer API
Producer Messages
Distributor Messages
Retailer Messages
Transaction 1
Transaction 2
Transaction 3
Customer UI
Transaction 1+Transaction 2 +Transaction 3
Transaction 1 Transaction a … … Transaction n
Transaction 2 Transaction b … … Transaction c
Transaction 3 Transaction p … … Transaction q
Transaction u Transaction v … … Transaction w
Block 1
Block 2
Block 3
Block 4
Hyperledger Fabric Ledger
DLT Gateway
MongoDB
Transaction 1+Transaction 2 +Transaction 3

Figure 5.3. Transactions flow from the external devices or systems to HF Ledger, MongoDB database and customer user interface.

After storing the transaction in the HF ledger, each transaction in the HF network is pushed into the MongoDB database via the DLT Gateway. In MongoDB, the record is stored at the jar level, combining the jar's information from the honey producer, distributor, and retailer. In this example illustrated in Figure 5.3, a row is stored combining Transactions 1, 2 and 3 in the MongoDB for a particular jar. The jar information on the customer UI is displayed at the row or record level from the MongoDB (centralised IT systems component from DIF), presenting all the information for a jar coming from all the nodes of the HF network in the JSON message format.

The major components of the HDSP for processing the messages from the HF network nodes are represented in Figure 5.4. First, the invoking messages trigger the *app.js* (Message Service Layer in the DIF) to perform the initial set-up and configuration before sending the message to the *invoke-transaction.js* program. Because the Broker component from the DIF is not required for a single DLT network, the *invoke-transaction.js* program overlaps the DIF’s Message Service Layer and DLT Gateways Layer component. Next, the Chaincode or Smart Contract is triggered to endorse and validate the transaction and store it in the HF Ledger (Step 1).

Finally, after successful execution, the CC triggers the DLT Gateway function, *invoke-transaction.js*, to process and store the message in the MongoDB database (Step 2).

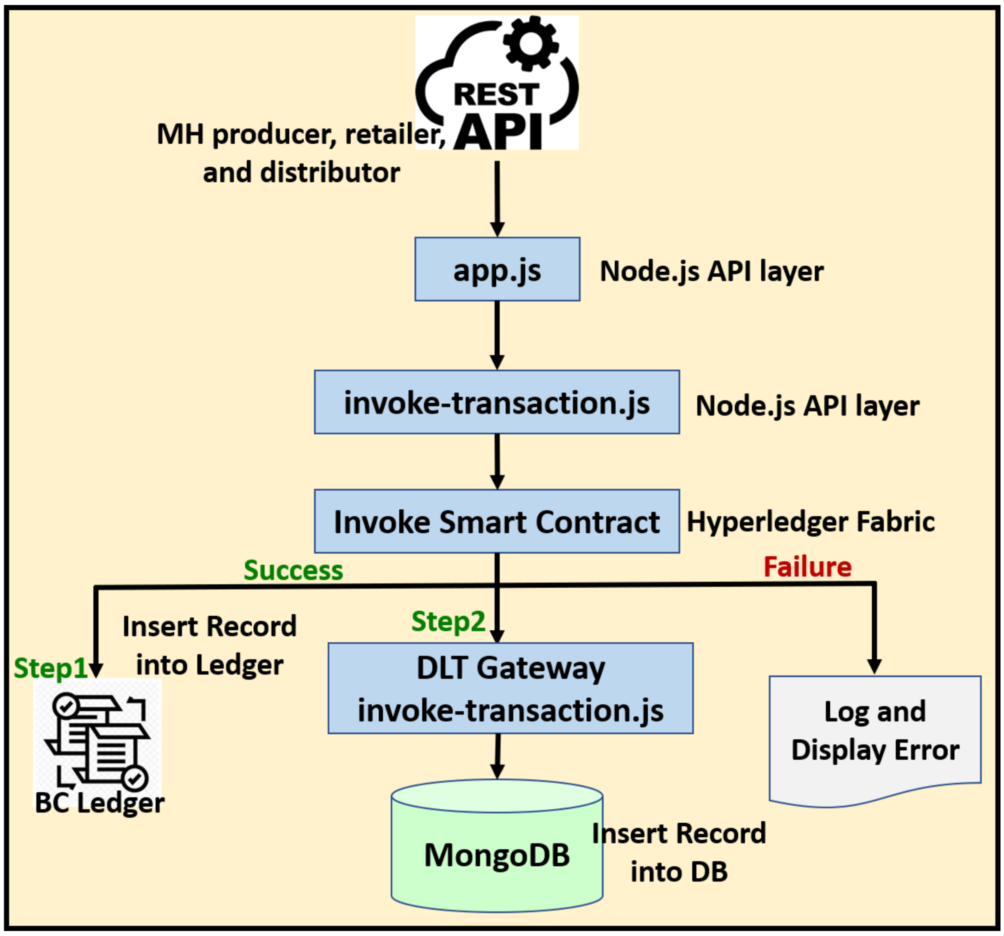


Figure 5.4. The major software components implemented in the HDSP. This presents the key components of processing the transaction and data flowing from external devices or systems and saved in the HF Ledger and MongoDB database.

5.1.3 HDSP Solution Characteristics

The distributed ledger technology interoperability framework (DIF) components and their interoperability principles are the foundational characteristics for establishing the interoperability of DLT solutions with centralised systems. The HDSP, its solution architecture in Figures 5.1 and 5.2 and the end-to-end transaction flow in Figure 5.3 are based on these critical aspects of the DIF. The HDSP design makes HF data and DLT solution an extension to non-DLT data and centralised solutions and establishes the pathway for DIF to enable the integrated DLT-based business solutions with centralised IT systems. Furthermore, the HDSP imbibes and extends the interoperability principles of the DIF and exhibits the following characteristics:

1. **API Centric Design**: The architecture and solution design are developed using Service Oriented Architecture (SOA) principles. Every service capability of the solution is wrapped around the service wrapper and exposed as an API request and response message. Any device, service, component, database, which interacts synchronously or asynchronously in a request and response message format, can interact with the system. This capability enables the services or business capabilities developed on any technology, platform or programming language and can be exposed as APIs to interact with DLT-based solutions.
2. **Internet of Things (IoT) enabled solution:** The architecture can integrate with IoT devices even though the IoT solution is not instantiated in the HF DLT solution prototype (HDSP). Radio frequency identification (RFID) or near field communication (NFC) can be implemented to assign a unique digital identity to the jar in the business process flow. The design to enable scanning of the honey jar by the distributor to feed the data into the HF DLT network demonstrates the IoT enablement of the solution. Most IoT devices can exchange data as API messages, making them interoperable with the solution design of the HDSP.
3. **JSON data format**: This makes the solution interoperable at the data level. The JSON format is the most widely used, accepted, and user-friendly data format for exchanging and transforming data. It is the format adopted by or compatible with most IT systems, applications and IoT devices [73].
4. **Business-critical network participants**: DLT network nodes and participants are limited to those who directly contribute to increasing trust and transparency in the business context (supply chain network). For example, the end customer (honey consumer) is kept out of DLT network because its sole interest is the quality and authenticity of the product, and this information is made visible by data integration and a web user interface (customer UI).
5. **Regulated DLT Data**: Processing, transporting, and storing the data in DLT network is expensive and not convenient to consume (compared to centralised application data). Therefore, only data contributing directly to improve the business process using DLT solution (e.g., enhancing trust and transparency in the supply chain) is part of DLT data structure (ledger). Hence, business-critical and minimal data are stored on DLT ledger

storage. Other essential and business-relevant data to the producer, retailer, or distributor, is expected to be stored on their own IT systems. This segregation enables the management of DLT ledger's size and the network's data bandwidth, enabling the solution's scalability.

6. **Cloud-Enabled Solution**: The end-to-end implementation of the HDSP is developed and implemented on AWS (Amazon Web Services) Cloud infrastructure. The API-centric design enables the solution to be workable on any Cloud, hybrid Clouds, or on-premises infrastructure. In addition, instead of installing the HF on a Linux medium size AWS server, which was done for the HDSP, any Cloud provider's Blockchain as a Service (BaaS) offering can be used to implement the solution.
7. **Contemporary Technology Stack**: Contemporary enterprise technology stack is used to implement the solution. This includes AngularJS designing the lean web interface for customers, Node.js to implement the API layer, JSON data format, MongoDB database, Hyperledger Fabric 1.4, Docker containers, Linux operating system, AWS Cloud, and enabling IoT technology (NFC or RFID). This solution demonstrates that DLT solutions can be integrated with enterprises' modern and popular technology stacks.
8. **Contemporary Enterprise Architecture**: Modern enterprise architecture concepts are implemented in the solution and include the solution based on SOA, API driven architecture, Cloud implementation, distributed systems, Messaging layer to enable interoperability, IoT enabled design, and NoSQL MongoDB database (even though the solution is processing structured data).

5.2 Business Use Case and DLT Platform Selection

An integrated DLT solution is built to validate the DIF and its interoperability principles. A relevant business use case is required to demonstrate the integration of DLT solution with centralised systems. The technical stack used to develop the IT solution for the selected business use case have to be relevant to establishing and proving the interoperability of DLT solutions with popular technologies used in modern enterprises. The business process or part that can be improved and made more effective using the distributed technology must be developed on DLT; the rest is expected to be implemented on a traditional technology stack. Therefore, selecting a business use case, a DLT platform and traditional technologies to implement the business use case, based on the DIF and its principles, becomes a crucial decision factor in demonstrating the interoperable solution. This section details the rigorous process of choosing an appropriate business use case and assessing various DLT platforms to select one suited for building the HDSP.

5.2.1 Supply Chain Use Case Selection

A suitable use case needs to be considered to design, develop, and instantiate the solution prototype for a specific business context. Without such a use case, the solution will remain at the architecture, framework, or similar abstract level, without providing the technical feasibility, technical design, and technology implementation insights needed to prove the utility of the DIF in enabling the design and implementation of interoperable solutions. Detailed technical implementation for a use case provides better insight into the interoperability potential at the solution's data, semantic, syntactic, application, tools, technology, interface, and infrastructure levels. Successful instantiation of a technical solution for a use case is expected to provide the knowledge and clarity for enterprises to consider the DIF in designing DLT solutions for their requirements.

A supply chain is one of the most complex business processes and is currently managed by centralised IT systems [103]. A supply chain's business processes and transactions span countries and continents, involve many non-trusted, governmental, intermediary organisations, and invisible stakeholders, and involve a significant amount of paperwork; a business transaction can last weeks or months. Such a complex process comes with inefficiencies, fraud, pilferage, significant trust deficits, ever-increasing compliance requirements, regulations, and stringent monitoring, all of which add costs to a business [104]. Current technology capability does not effectively address the severe challenges associated with a supply chain [55]. Therefore, it is timely that the supply chain process is made simple and transparent to establish trust amongst non-trusted and remotely located participants and stakeholders. This will enable monitoring and tracking of the provenance of the supply chain products, and how the products were produced or manufactured to help identify any slavery, child labour, environmental impacts, fertilizers or chemicals, component authenticity, or animal exploitation involved in production [2].

Distributed technology can simplify the supply chain processes and answer many of the industry's teething problems [105]. Distributed ledger technology based solutions can improve transparency and data sharing, establish trust in non-trusted participants, enhance security, and enable product visibility from start to finish [106]. A DLT transaction can record goods transferred among multiple parties, each identified by a unique address in DLT network. The relevant information of the supply chain, such as quantity, date, location, price and similar, can be lodged into the distributed ledger as a transaction [107]. The information in the ledger will then be transparent and visible to all participants in DLT network, making it feasible to establish the traceability of every transaction to the grassroots of raw material.

Many solutions are suggested and implemented in the supply chain domain based on DLT. Chen [108] designed autonomous, IoT-based fuzzy cognitive maps and an agent-based tracing system for a product usage life cycle. Chen discussed agricultural-based food products to simulate a complex food tracing system. The RFID and DLT technology-based food traceability systems were discussed by Tian [109] in a system that covered information management and data gathering processes for every link in the agriculture food supply chain. This achieved monitoring, tracking management, and traceability for the safety and quality of food from farm to table. A TraceFood system was proposed by Storoy et al. [110] to ease the automated electronic data interchange for supply chain products, and was based on non-proprietary international standards. Their system included the principles for uniquely identifying food items, traceability information exchange based on generic standards, and the relationship between the data elements was defined for sector-specific ontology. A BC-based traceability system for wine supply was proposed by Biswas et al. [111]. The immutable block of information was recorded in a DLT system as a transaction. This traceability system enabled safety and transparency in the overall process of wine preparation from the grape to the bottle. An AgriBlockIoT traceability solution based on a decentralised DLT was presented by Caro et al. [62] for Agri-food supply chains. Other researchers (e.g. Bahga et al. [112], Olsen et al. [113], Lin et al. [114], Kamoun et al. [115]) provided traceability solutions to improve trust and transparency in a supply chain. The solutions' focus is to establish and convince of the usefulness of the BC/DLT based solution in the supply chain domain, with little or no discussion on systems or solutions interoperability. However, the design of the HDSP focuses on establishing the interoperability of DLT-based solutions with traditional systems by using the DIF and its interoperability principles.

5.2.2 Distributed Ledger Technology Platform Selection

Many DLT platforms are available to implement solutions at the technical level. For supply chain use cases and enterprise-scale implementation, the following five prominent platforms were evaluated, and summarised in Table 5.1:

- **Ethereum**: This is one of the most popular BC platforms [114], widely used in a permissionless network; however, recently, its usage has also increased in permissioned enterprise network setups [116]. Ethereum can create almost any application; the software runs on its own Ethereum Virtual Machine, using various programming languages. Ethereum introduced the concept of a Smart Contract, now widely used in almost all DLT platforms, and can automate many business processes. The platform is used in supply chain solution implementation like Treiblmaier et al. [117], Ge et al. [118], Mao et al. [119], Kshetri [120], Galvez et al. [105], Casey et al. [121], Want et al. [122], Hewett et al. [22], Lin et al. [60], Biswas et al. [111].
- **Hyperledger**: This is an enterprise grade and one of the most renowned DLT platforms offering modular system design and architecture with greater resiliency, flexibility, confidentiality, security, and scalability [123]. It is designed to support pluggable applications with various components and accommodates enterprise systems' significant complexities and intricacies. In addition, Hyperledger supports data storage in multiple formats [102]. It can create different ledgers for personal channels to manage sensitive data, is strongly supported by IBM and Linux, and is used in many research projects like Lin et al. [114], Ge et al. [118], Kshetri [120], Wang et al. [122], and Petersen et al. [124].
- **R3 Corda**: R3 Corda is one of the leading DLT consortia [63], and was formed by a financial organisation's collaboration to design and develop the Corda network [125]v. The platform was designed for financial purposes; however, its usage in other domains, including that of the supply chain, is increasing [63]. Corda uses known identities to establish trust in the network, and the transactions are kept private between network participants; only parties part of a transaction have access to that transaction. This helps maintain data consistency, a higher volume of transactions, and better scalability.
- **IBM Blockchain**: This platform offers robust utility in many domains such as supply chains, trade finance, governance, oil, and gas [6]. It enables higher value for business models by creating synergies with traditional technologies and other permissioned DLT implemented in organisations. It offers a powerful SDK and interface capability

offering the agility and flexibility to deliver integrated solutions. It also allows either joining an available DLT network or creating a new channel inviting interested organisations to join, supported by on-Cloud or on-premises infrastructure and easy to instantiate networks.

- **Multichain**: This is an open-source DLT platform for building and operationalising permissioned DLT applications to function between or within organisations [1]. High-end features such as data streams, native assets, simple configuration per-chain and permissions management, enable an enterprise to build scalable, integrated, and function-rich applications. It is considered one of the most effective permissioned enterprise DLT networks and is used by around a hundred organisations for financial transactions [125]. However, its usage in business domains beyond the financial industry has not reached a critical stage in selecting this DLT platform for supply chain use case.

DLT Platform	**Public / Permissioned**	**Language Supported**	**Consensus**	**Support for Smart Contract?**
Ethereum	Public and Permissioned	Flint, Solidity, Scilla	PoS, PoW	Yes
Hyperledger Fabric	Permissioned	Go, Java, Node.js	PBFT	Yes
R3 Corda	Permissioned	Java, Kotlin	PoS, PoW	Yes
IBM Blockchain	Permissioned	Go, Java	PoS, PoW	Yes
Multichain	Permissioned	C#, Go, Java, Python, PHP	PBFT	No

Table 5.1. The evaluation of five DLT platforms used to select a suitable one, Hyperledger Fabric, for the MHSC use case. Note: PoS – Proof of Stake, PoW – Proof of Work, PBFT – Practical Byzantine Fault Tolerance.

Hyperledger Fabric (HF) is an enterprise-grade DLT platform with advantages over other similar platforms. Hyperledger Fabric allows modular architecture, which enables developers to create plug-in components. Companies want to reuse available capabilities (e.g., Identity Management), and the modular architecture of HF enables those capabilities to integrate well. While building permissioned DLT, HF offers an advantage by assigning different roles to the nodes, as a Client to invoke transactions, Orderer to update transaction data, Peer to receive the update and commit transactions, and an Endorser to validate the transaction authenticity. HF does not indulge in PoW (Proof of Work), so it can attain high throughput and scalability. Sensitive data (e.g., commodity price) can be kept private with the involved nodes in the transaction by establishing separate Channels. Hyperledger Fabric allows rich querying capability; its LevelDB enables keyed queries via its key-value DB, can process critical range and composite key queries, and can be used with JSON. The Hardware Security Model feature helps manage and safeguard the digital keys for authentication and, if used along with Identity Management, can increase the security of sensitive data and keys [126]. Backed by IBM and Linux, HF offers rich community support, and its development community is vibrant and experienced. This research used HF as a DLT platform to establish the interoperability amongst DLT solution and traditional technologies in the MHSC use case.

5.3 Hyperledger Fabric for HDSP Development

Hyperledger is an umbrella project started by Linux Foundation and consists of open-source DLTs and related tools. Among the many frameworks developed and supported by Hyperledger, the Hyperledger Fabric (HF) is prominent, popular, and widely used in enterprises [63]. Hyperledger Fabric is based on a modular architecture, provides the capability to execute Smart Contracts (called Chaincode CC in HF), provides membership services, configurable consensus, and supports open and flexible development capability supporting Java, JavaScript and Go [123] [50].

5.3.1 Hyperledger Fabric Components

The main components of HF developed in the HDSP are presented in Figure 5.5 and explained next.

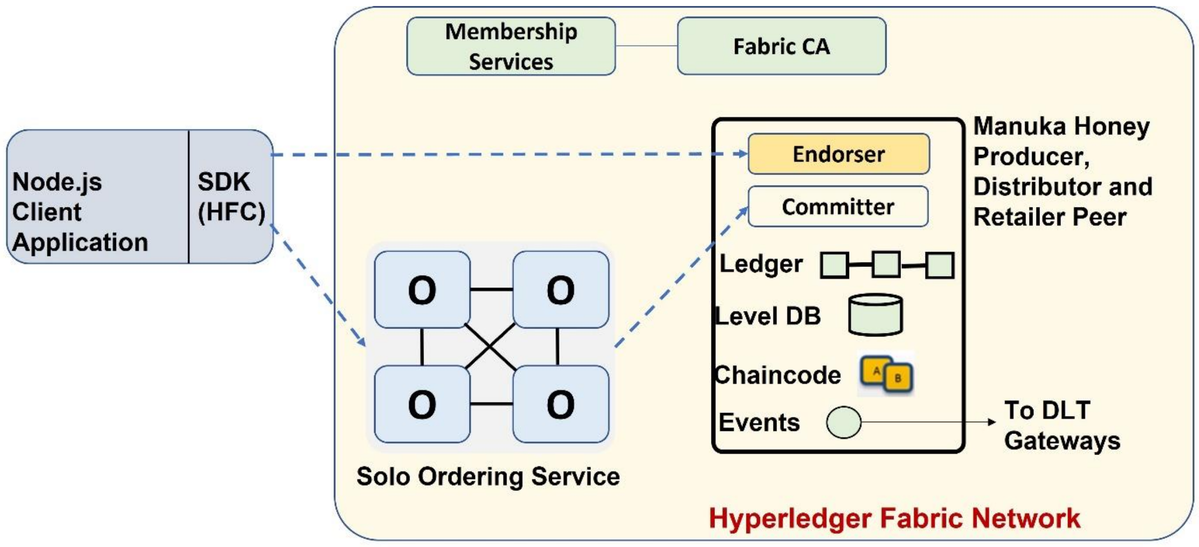


Figure 5.5. The components of the Hyperledger Fabric 1.4 developed for the HDSP. It illustrates the interaction between the external client application and components of the HDSP. Adapted from [123].

Membership Services: These provide an identity for the honey producer, distributor, and retailers transacting on the HF network. This identity is a digital certificate that network peers use to sign transactions and submit to the HF. One MSP (Membership Service Provider) is defined for every honey producer, distributor and retailer organisation, with one or multiple users or participants (this research defined one participant or node for each MSP). The MSP authenticates the legitimate peer and provides appropriate access to participants. In the HDSP, each participant is an anchor peer, endorser, and committer that can authorise and commit transactions. The Membership Services can either interact with internal Fabric-CA (certification authority) or plug into an External CA to generate membership certificates. A client application can be on any system; HF client is an SDK provided to interact with the HF peer or ordering service.

This research used the Fabric provided CA to generate the MSP certificates. This capability is comparable to enterprises' security certificates for HTTPS or TLS (Transport Layer Security) and relies on public and private key associations.

Client Application: The client application can be written in any language; the HDSP utilised the popular enterprise language Node.js to develop the application. It provides SDKs to interact with DLT, known as “Hyperledger Client” (HC) and can be developed using Java and Python.

Peer: Multiple organisations will participate in the HF network, and each organisation can have multiple users, peers, or network participants. Separate functionality or roles can be assigned to users within an organisation with appropriate authorisation. There are three organisations in the HDSP network; each has one peer as a honey producer, distributor, and retailer. The peer node manages the HF ledger storing all the transactions and state information, and executes Chaincode on each node of the HF network. Once the transactions in the HDSP are committed to the HDSP ledger, they can be emitted as events to integrate with external, DLT, or non-DLT applications (e.g., MongoDB in HDSP).

A peer can have one or more functions; one function is an endorser. An endorser executes the CC on a node, signs the output of that CC execution using the certificate, and endorses the transaction output. Every signed transaction from all the endorsers goes to the ordering service to commit the transaction on the network. The ordering service transactions are committed to the HF ledger by the Committer, as shown in Figure 5.5. Peers can distribute the responsibility as endorsers or committers; however, for simplicity, each peer is defined as an endorser and committer in the HDSP.

Ordering Service: This component aims to provide the ordered set of transactions. The HDSP peers can submit the transactions in random orders, and the ordering service puts the transactions in the correct order. It approves the transaction block inclusion into the HF ledger and communicates with endorsing and committing peer nodes. This ensures that all the nodes on the HDSP network receive the transactions in the same and correct order to commit on the HF ledger, guaranteeing the HF ledger’s consistency among all the nodes. The dedicated ordering node need not hold a ledger or smart contract.

Transaction Consensus Sequence: The sequence in which the transaction consensus is achieved is shown in Figure 5.6. For example, the honey producer API submits the transaction to all the endorser nodes in the client application. They execute the transactions, agree on the identical output produced by those nodes, and add the signature to the output produced. Next, the honey producer application/API collects the endorsement from the other two peers for the transaction submitted. Once sufficient signatures are collected, the transactions are presented to the ordering service (order) to ensure the correct order of the transactions. Then the transactions are validated, for example, to ensure no duplicated spending on the same account in the transaction block.

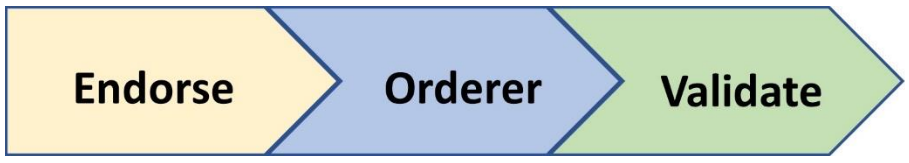


Figure 5.6. Order of transactions to achieve the consensus to commit the transaction to the ledger.

5.3.2 HDSP Nodes Interactions

This section explains the flow of the honey jar into DLT network to track the jar. The solution assumes that the manuka honey is handled at jar level instead of batch or container level. At every stage of the flow of jars in the HF network, it explains the involvement and interaction of honey producers, distributors, and retailers. The relevance and interaction point of the customer UI interface demonstrates the efficient design of the business solution.

Manuka honey producer: The producer produces the honey from its own, rented, or contracted bee hive. The honey is sent to a designated government laboratory for testing to authenticate its quality. The honey is then packaged in a jar based on its quality and specially packaged with a unique ID assigned to the jar. The industry-wide adopted technology used to assign a unique ID is RFID (radio frequency identification) or NFC (near field communication); in many situations, NFC is preferred because of its low cost and ease of use. Near field communication can be as thin as paper and designed to disable once the customer opens the honey jar lid. This feature makes the NFC non-usable after the jar lid is opened, avoiding duplication or reusing the NFC identification or honey jar.

The detail of the jar is pushed into DLT network by the producer. Honey jar details need not be pushed one by one; for example, a producer can store the NFC or RFID details in the database. Additional information (e.g., honey production date, batch number, producer information) can be automatically suffixed to unique jar IDs to push the data into DLT network in a batch process. Every network participant is also an endorser that can approve the transactions to be inserted into DLT to keep the design of the HDSP simple. This inserted information by the producer is committed to the ledger of all three network peers.

Manuka honey distributor: The distributor receives the honey jar from a producer(s) to distribute to the retailer(s). Distributors can capture the jar information using an IoT scanner/device and append the jar information with distributor information (e.g., distributor name, date/time, transport quality) to push the data into DLT network in a batch process. The distributor is also an endorser and can approve and inject the transactions into the network. The ledger updates the insertion of jar information fed by the distributor for all three network participants. If the jar is damaged, the seal is broken, or the lid is open, the distributor can decide not to inject that jar information into the network but return the same to the producer.

Manuka honey retailer: The retailer receives the jar batch from the distributor and can decide to update its retailer ERP system first. The retailer can build an automated process of injecting the appropriate ERP records into DLT network. Because the retailer is also a network endorser, the HF network will commit the transactions, and the ledgers of all three network participants are updated with the new transactions.

- For the customer UI, the honey jar data are accessed from MongoDB. The Events() function pushes the HF transaction data into the MongoDB database via DLT Gateways. The AngularJS web UI extracts the data from the MongoDB database each time the customer scans a purchased honey jar. Thus, the information displayed for a customer is the consolidated information from the honey producer, distributor, and retailer, for the specific jar.

5.4 HDSP Based Knowledge Contribution

The development of the HDSP utilised system design knowledge, code development skills, testing abilities and tacit skills. The development was a technical activity undertaken in collaboration with industry experience people. The HDSP build process stimulated many discussions, alternative designs, operational models, Smart Contract designs, generating helpful knowledge that can be useful for the research community. From a DSR perspective, this section elaborates on the prescriptive and descriptive knowledge acquired from the research contribution of building the HDSP. Prescriptive knowledge is interested in achieving the specified goals efficiently and effectively and explains the ideas, structure, concepts, and practices to be used in future research for possible instantiations. Descriptive knowledge or truth value illustrates the observational facts, empirical observations, and causal laws. The knowledge explained in the remaining section can be used by practitioners to instantiate solutions or by researchers to build further advanced solutions based on the knowledge gained.

5.4.1 API Driven Architecture

The architecture of the HF DLT solution prototype is presented in Figures 5.1 and 5.2. To simplify the prototype, only one peer (Peer0) in an organisation is allowed to be associated with the channel (however, there will be multiple peers on the channel in a production implementation). The components outside the yellow box in Figure 5.2, Hyperledger Fabric DLT network, are off-chain components, and the remainder is on-chain components. Every peer ledger contains two major interior components, although Figure 5.2 represents the internal components of only the distributed peer ledger, and the MHSC channel holds the history of all transactions.

The Linux Foundation (LF) designed the HF as a modular system to plug different API methods. Using the same principles, the HF DLT is designed as a modular system using the APIs to promote the components of HF: Chaincode (CC), HDSP ledger, communication layer, data store, pluggable cryptography, and consensus layer. This design enables not only backward interoperability but also integration with different flavours of DLT implementation. This design abstracts the complexity of the HF business network and enables solution implementation. This simplicity and solution feasibility is possible due to easy-to-use APIs by Linux Foundation to access new systems, network participants, middleware, and business networks. The research recommends either a token coin or crypto-asset-agnostic approach to set the asset tokenization notion to represent virtual and physical assets.

While instantiating, the network design is expected to be centred around the participating organisations instead of peers, as they are the delivery unit with a secured domain and credentials. This will facilitate governing one or more peers depending on the MSP (Membership Service Provider) to issue certificates and identities for the network peers and clients for CC access privilege. The HF orderer is a separate organisation (including the MSP) and is one of the critical components of the HF for ordering the transactions in sequence. The HDSP implemented the orderer using Solo. However, Apache Kafka on Zookeeper or Raft protocol must be used for a production-grade orderer system to provide a Crash Fault Tolerance (CFT) system and robust and stable implementation. From the CAP (consistency, availability, partition tolerance) principal perspective, HF is designed as the AP system, like many other DLT technologies. The “A” in AP is for availability where the ledger copy is available on all the peers, and “P” denotes partition tolerance to maintain network operation despite failed nodes. In addition, HF uses version control and transaction order to manage consistency.

Executing an HF peer is a costly and heavy business, which can be reduced by grouping the entities with trusted parties into an organization, which decreases the number of network peers. For example, a financial institution and its customer may be combined as one organisation, as a financial institution will probably have resources to run the peer for itself and its client. Two possible implementation designs are proposed:

1. Middleware and application layers can embed the access control logic. For example, the login credentials or IDs can distinguish users and mapped IDs or control the access of permitted CC functions.
2. Distinguishing attributes embed within the certificates issued to organisations' members by an organisation MSP (acting as a CA server). CC or middleware can implement the access control functionality to disallow or permit the operation by parsing the attributes according to the application policy.

The HDSP was designed using distributed system architecture principles and developed on an AWS Cloud medium Linux server. AWS is a leading cloud service provider offering IaaS, SaaS, PaaS and function execution platform. The HDSP supports using IoT devices to assign a digital identity to a honey jar to monitor and track the movement of the honey jar. Various programming languages (e.g., Golang, JavaScript, Node.js) are supported for the organisation to use their choice of programming language based on their technical stack and programming skills. The customer UI was developed using the AngularJS framework. Docker containers were utilised to execute the HF CC with Mongo DB. The APIs provided loosely coupled architecture to enable the servers/solutions to be geographically distributed as per organisation infrastructure requirements, service exposures, technical skillsets, and data bandwidth constraints.

5.4.2 Data Design

The data design is expected to inject data into the HF network to contribute to trust improvement, transparency, and collaboration amongst peers. An off-chain/non-DLT database have to store the rest of the data and can use the HF data partitioning capability to control access to sensitive and confidential data with selected peers. This feature also contributes to managing the data bandwidth for geographically distributed servers, maintaining the latency and increased transaction volume.

The HF DLT solution prototype (HDSP) uses an HF event to copy the data into the MongoDB database via DLT Gateways for further processing by centralised systems. DLT world state maintains the current state, using Level DB or Couch DB to store DLT data. Couch DB can support JSON-based rich query operations. However, the HDSP uses Level DB for design simplicity and because MongoDB was used to store transactions and events to support complex relational database queries. A business have to consider using CouchDB if not planning to use a query-friendly database or if it needs a reasonable data query capability.

Data must not be stored on DLT for big data or file operations. However, the file's fingerprint or data may be stored on a DLT network and the signature verified before processing to assure data integrity. Flexible implementation parameters may manage the performance and scalability of the system (e.g., BatchTimeout 1s, maximum message count 50) as blocks are committed to the ledger. These parameters can significantly influence the system's performance, throughput, and scalability.

5.4.3 Smart Contract (Chaincode) Design

After an organisation deploys DLT solutions, it will require ongoing maintenance and enhancements to Chaincode (CC) due to regulation changes, new peers in DLT network, and bugs. This requirement can make development and operational practices challenging, especially when organisations with varying velocities and cultures collaborate to develop and operationalize the solution within time, cost, and quality constraints. This research endorses the following recommendations to mitigate these challenges.

While designing DLT data, ensure that CC owns the data and the underlying ledger accessible directly only by APIs to track, manage and measure the data access from the ledger using API Gateway. When the CC is enhanced for the new version, it migrates the previous CC data to the new one. To remedy this situation, consider splitting the CC in the business contract (to address the business logic) and data contract (to record the data into the ledger). Because the data contract does not change as often as a business contract, it addresses the need to access the most-used operations and data attributes. However, if the CC needs to change frequently, a CC Registry (a CC ID can be assigned to each CC for better management) may be created to register all the deployed CC. Each time a new CC version is deployed, a transaction can be invoked to register the newly deployed CC and update the previous versions.

The fundamental CC development principle is to have a deterministic operation to produce an identical output of code execution on different peers/machines. Everything required for transactions/operations must depend on the current state of the ledger and the parameters provided by the invocation. Non-deterministic operations such as random number generation, network calls, and file operations in CC must be moved to DLT network or provided as parameters to the executing code. Also, validating or sanitising the parameters is essential before executing the business logic based on the current state of DLT network. Avoid long-running tasks like network operations or complex cyclic logic in CC. Keep the process simple to receive the data, validate it, do the appropriate conversion, extract DLT state data, merge the parameters according to the business logic, and serialise the data before storing it into DLT and processing the next operation. DLT network peers may exchange business information running on CC in completely isolated and independent containers (not running inside the channel).

The HDSP in Figure 5.1 incorporates the business logic into CC, and most DLT solutions use a Smart Contract (SC) to implement business processing. These are executed in an independent Docker container, managed as an isolated entity, and loosely connected with other system components to exchange information. CC may be written in various languages (e.g., Golang, JavaScript, Node.js) to support development in the technology of choice. Service-Oriented Architecture (SOA) practice will help develop the services at the right granularity, enabling the efficient usage of distributed employees' skill sets and modular, service-oriented, and distributed architecture in the organisation. CC in the HDSP is developed in Golang and uses the HF libraries and toolkit for application development. Golang is recommended for CC because Node.js uses the npm (node package manager) package when container building, impacting network response for multiple peers on the network. Java is a new entrant, and its technology robustness and community support for HF have not yet matured.

Chaincode is a vital component for business and network peers, defining the conditions and rules for transaction validity, which all network participants must agree upon in advance to build trust. To support this, the following suggestions may be helpful.

1. Record the approvals from the relevant parties on CC and encourage code review to avoid misinterpretation.
2. If possible, publish the test cases, automate them, and make the test result available to network participants for their approval/review.

3. Use test case results to assess the CC change impact and any backward compatibility required.
4. Carefully review the scope of CC for the entire network or a set of participants.
5. Establish the traceability of the CC change to the underlying new requirement or bug-fixing.

Reach consensus on the frequency of CC code promotion in the production and change process, and release checks and conditions to avoid ambiguity for participating organisations.

5.4.4 Deployment and Operational Aspects

Instead of Docker containers, from HF 2.0 onwards, organisations may also use other external non-Docker containers. The HDSP executes the CC, peers, orderer, and fabric certification authority in independent Docker containers managed as isolated entities. The HF community publishes stable and tested commonly used Docker images and Docker Compose definitions to build the binaries, which along with the capability of spinning operating system instances and scripting, may enable easy deployments and operations. Kubernetes or a similar container orchestration tool have to be used for production-grade deployment to efficiently manage many containers, based on the organisation's scalability requirement. Furthermore, the DevOps tools, such as Jenkins, Ansible/Chef/Puppet. may enable Infrastructure as a Code (IaaC) practice. Continuous Integration (CI) tools such as Jenkins may automatically deploy Docker images and CC executables to different application environments. Additionally, the recent versions of HF, 1.4 and 2.0, focus on production operations and stability, which is expected to increase practitioners' trust in implementing and operationalising the HF-based solutions [123].

Hyperledger Fabric uses a multi-version concurrency control (MVCC) mechanism to prevent double-spending and enforce ledger consistency. A network may face critical collisions (modifying the key-value pair simultaneously), unintentional or intentional modification of transaction sequences or delays in transaction calculation and commitment. These issues can be addressed by versioning the stored keys on the ledger, algorithm checks to match the result of executing range query, projecting the WriteSet onto the current Worldstate, deriving the key from the transaction, multiple keys usage, transaction queuing, and splitting assets. It can be relatively easy to exploit user profiles created by an organisation to create hostnames and specifications, to organise the complex network and its structures for the CI, automation, and testing. This design may use the definition in the profile to generate the genesis block and channel.

During prototype development, logging was a vital function that assisted with analysing and detecting runtime problems. Writing the log messages (collected from all components) in a standard error file (stderr) controlled by the peers and modules configuration is recommended. Provide a unique name to each object logged that is used as the prefix for each record. Do not send all the log records to the output, but control by setting a logging severity for each object. CC may control the severity of logging by setting the SetLoggingLevel API function. Ensure that the standard output function is disabled in the production environment for security reasons when the peer manages the CC process. The outputs of CC and peers have to be disabled and only used for debugging.

Certificate generation defines the domain tied to YAML (yet another markup language) configuration files to offer flexibility and tie to the specific domain for the orderer organisation. The certificates can streamline many securities and related checks by attaching them to the organisation, preventing data leakage. For example, if a hacker steals a peer’s certificate for use on a completely different network, perhaps to spoof the issuing organization, it exposes the certificate’s content to unauthorized access. Because certifications are tied to a domain, if somebody tries to use this certificate outside the organisation’s domain, the connection will be refused, and data leakage will be prevented, providing robust security.

For security purposes, a good HF design must not enable peers from one organisation to communicate with a peer of another organisation, except via anchor peers, which can be defined during the MSP setup. An anchor peer acts as a fulcrum in an organisation by using a gossip protocol to synchronise the cross-organisation ledger. Hyperledger Fabric also provides the hardware-based protection of a digital key, resulting in better security of assets. Also, practitioners must use traditional security practices like SFTP, database encryption, HTTPS, two-factor authentications, SAML, to synergise with the security features of DLT network.

Chapter 6: Framework and Solution Prototype Evaluation

This chapter explains the methods and techniques used to evaluate the two artefacts developed in this research, the DLT interoperability framework (DIF) and the Hyperledger DLT solution prototype (HDSP). The evaluation validates that the artefacts produced are practical and relevant to the research problem and were not created by random steps, but by executing established research steps following a trusted methodology. Design science has always emphasised the need to assess the utility and efficacy of artifacts and knowledge produced, as suggested in Larsen et al.'s research [127]. Evaluation of research artefacts is an integral part of research activity to assure research rigour and provide feedback for future development [128]. Following the advice of Cleven et al. [88], this research did not undertake evaluation as an isolated activity but was designed to be conducted iteratively from the beginning of the artefact design, development, and instantiation process.

Section 6.1 of this chapter explains the evaluation methods and techniques followed in this research. The subsequent section elaborates on the formative and summative evaluations undertaken to validate the DIF. Sections 6.3 and 6.4 overview the assessment of the DIF by IT professionals and instantiate the solution prototype designed based on the DIF. The following section explains the evaluation undertaken to choose the business use case, technology, architecture, and data format to create the HDSP. Section 6.6 explains the formative and summative assessment undertaken for the HDSP; Section 6.7 covers the working demonstration of the HDSP and its test cases execution. Section 6.8 validates that the artefact evaluation conducted in this research is adequate, and section 6.9 summarises the overall evaluation undertaken for both artefacts.

6.1 Artefact Evaluation Methods

The evaluation of artefacts is crucial to establish the relevance and practical utility of the artefacts to address the targeted problem. Evaluation involves the design research outputs, including artefacts and the research process. Research by Alturki et al. [129] published an overall DSR roadmap after analysing and reviewing sixty articles to document the key activities, tasks, and steps involved in DSR. Alturki et al.'s work was based on March et al.'s study [104], suggesting that acceptable research need not extend to evaluation if the design solution is novel. It is argued that both aspects of DSR, construction (i.e., design and development) research, and evaluation research, have to be encouraged. However, the separation between them is an important decision, as evaluation requires different expertise and entail substantial resources [129]. This research balanced the approach to evaluating the artefacts (the DIF and HDSP) against available resources, such as evaluation and testing knowledge, project time, and cost, and utilised selective and relevant evaluation methods to assess the artefacts produced. Early and iterative assessments reduced time and expense by evaluating the decisions and choices (e.g., Business use case, DLT platform) before the design and development of the artefacts.

Research by Pries-Heje et al. [130] integrated a significant proportion of the extant research work on artefact evaluations. Their study explained two evaluation dimensions:

- the method used (naturalistic or artificial), and
- at what time to conduct the evaluation. When to conduct evaluations can be categorised as follows:
    - ex-ante – before the design or development of artefacts and a synonym for formative evaluation.
    - ex-post – when the artefacts are developed or constructed or reached a particular milestone/build stage. It is synonymous with summative evaluation.

Both natural and artificial evaluation methods have their advantages. An artificial (technical) evaluation has a lower cost and is quick to evaluate, but a naturalistic evaluation is more practical and relevant. However, the nature of the artefacts also determines the appropriate evaluation method; the technical nature of the artefact produced in this research demanded technical (artificial) evaluations. The selection of modern architecture, design, integration patterns, technology platforms, databases, and tools popular in modern enterprises covered the natural aspect of both artefacts' evaluations.

The research also utilised the maturity model for enterprise interoperability (MMEI), artefact assessment by IT professionals, validation by practical instantiation of the solution, demonstration of the solution prototype, and MHSC test cases execution. These are the widely discussed and adopted evaluation methods and techniques popular in Design Science literature. [92] [93][131]

6.2 DIF Formative and Summative Evaluation

The formative and summative evaluation of the DIF was designed and undertaken from the design stage of the DIF. The formative evaluation was undertaken to design and enhance the components of the DIF to meet the interoperability principles criteria (see Section 4.2.2) and to meet the integration needs of modern organisations (see Section 2.2). When the DIF reached a particular development milestone, a summative assessment was conducted. The DIF is implemented to design interoperability architecture for popular enterprise architecture and interoperability design patterns and tools (see Section 4.3). For each design implementation of the DIF, a summative assessment was triggered. As a part of the iterative evaluation process, as shown in Figure 6.1, the outcome of the summative assessment was fed into the formative evaluation of the DIF. This iterative evaluation process enabled the refinement of the DIF components and the final interoperability framework.

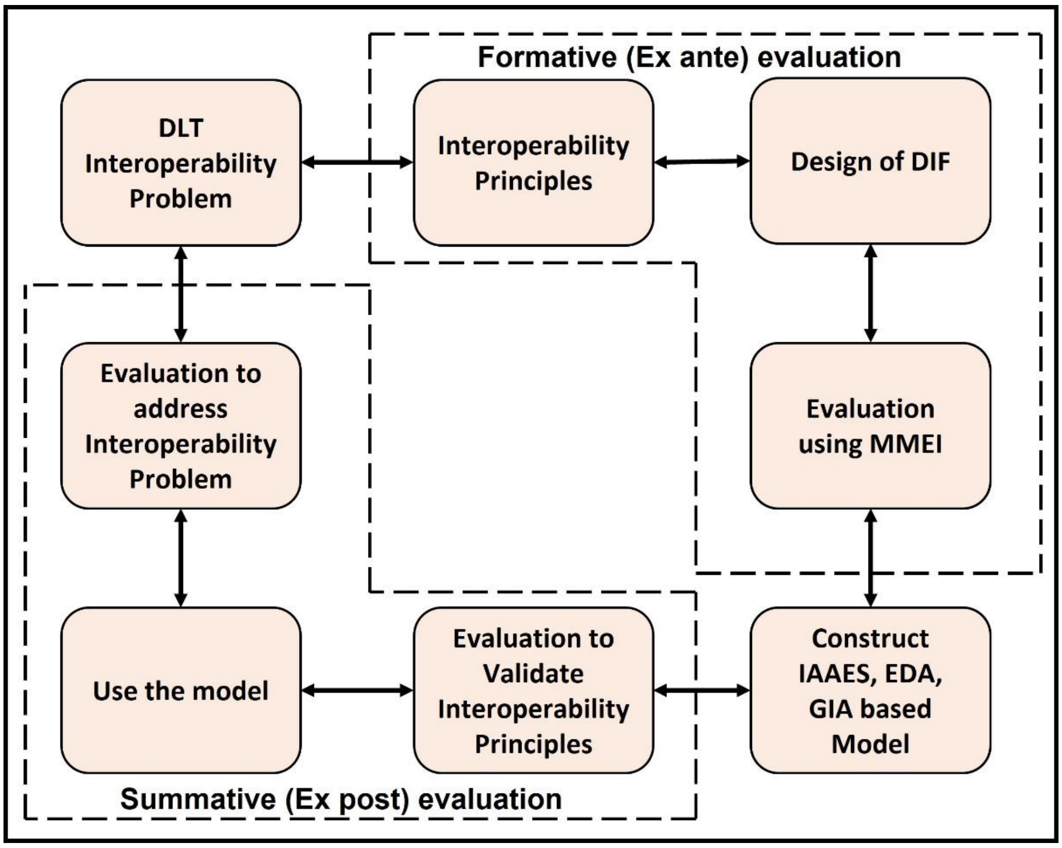

Formative (Ex ante) evaluation
DLT Interoperability Problem
Interoperability Principles
Design of DIF
Evaluation to address Interoperability Problem
Evaluation using MMEI
Use the model
Evaluation to Validate Interoperability Principles
Construct IAAES, EDA, GIA based Model
Summative (Ex post) evaluation

Figure 6.1. The cyclic formative and summative evaluation process for the DIF. The figure shows that each significant activity follows the DIF evaluation exercise to undertake formative (ex-ante) and summative (ex-post) evaluations. Adapted from [86].

As a part of the formative assessment, the maturity model for enterprise interoperability (MMEI) framework was used to evaluate the interoperability capability of the DIF. An iterative formative evaluation of the DIF was undertaken to validate that:

- the design of the DIF and its components met the interoperability principles criteria, as explained in Section 4.2.2
- the DIF and its components aligned with the modern interoperability approach and met the integration requirements of modern enterprises, as described in Section 2.2

6.2.1 Summative Evaluation

Three interoperability architectures, as explained in Section 4.3, were created by using the DIF. After every architecture design, a summative evaluation was undertaken to validate the architecture against the interoperability principles and interoperability requirements of enterprises. After each summative assessment, the DIF was enhanced, and the formative assessment was repeated. This iterative assessment enabled significant improvements in the DIF design, supporting the interoperability for widely used architecture practices, design patterns, tools, and IT systems used in contemporary organisations.

a. **Enhancements due to the First Iteration**: During the conceptualisation of the GIA (Generic Interoperability Architecture), a layer of organisational DLT network nodes (similar to the Hyperledger component in the HDSP) was formed. This layer participates in many DLTs that interact with DLT Gateways layers (similar to DLT Gateways in the HDSP). The interaction of the DLT Gateways layer with the Message service layer (similar to the API layer, *app.js* component in the HDSP) was established and interacted with centralised IT systems/applications or devices.
b. **Enhancements due to the Second Iteration**: After designing the Interoperability Architecture using API Gateways, ESB and Service Mesh (IAAES) architecture, some potential shortcomings and implementation challenges were identified. The output of the DLT Gateways layer is supposed to go to an additional layer to handle the message in a format suitable for the destination system. The earlier Messaging Service Layer (MSL) was designed to handle the message conversion; however, this can make the MSL layer CPU intensive and complex, causing end-to-end message flow bottlenecks. Also, many-to-many message conversions would add overheads in the MSL and risk distributing the functionality into various

components or layers. This additional layer of the Message Broker clearly defined the separation of concern for the components in the DIF.

An interface for External Oracles was added for the dynamic data that need to be fed into DLT, such as currency rates, stock prices, improving the practicality and utility of the framework.

c. **Enhancements due to the Third Iteration**: A single layer of Message Broker became heavy during the EDA design, and implementation was not straightforward. Also, there was no need to bring the DLT Gateways layer between DLT and the outgoing Message Broker. Therefore, to ease these concerns, the message broker component was broken into two: In-Flow and Out-Flow Brokers. Only the incoming messages (towards the centralised IT systems) would get a feed from the DLT Gateways layer to the In-Flow Broker; the outgoing message (from centralised IT systems) directly feeds the API/messages to DLT network layer via the Out-Flow Broker. This segregation provided significant flexibility during solution implementation, and the layer dependencies were reduced. An additional input block, DLT Metadata, was added to feed the input to the outgoing canonical messaging layer. These metadata parameterise the message conversion into the destination DLT message format definition based on DLT network and the message's route. This layer can also add additional parameters to the outgoing messages for DLT to enable the enterprises to tag the messages to automate further business processing. The message reformatting is parameterised when the message flows from the DLT Metadata to the In-Flow Broker. The parameters and associated reformatting are based on the source DLT network from where the message originates.

6.2.2 Maturity Model for Enterprise Interoperability (MMEI)

Research by Leal et al. [132], Guedria et al. [28], Rezaei et al. [98] and Leal et al. [95] discussed the maturity model for enterprise interoperability (MMEI) framework to measure the interoperability degree of enterprise systems. There are five levels of interoperability maturity of any solution [28]. Level 0 is the unprepared and non-relevant solution from an interoperability perspective. Level 1 is the defined capability of correctly modelling and describing systems to prepare interoperability. Level 2 is an aligned solution capable of making necessary changes to align to common formats or standards. Level 3 is an organized solution capable of metamodeling for necessary mapping to integrate with multiple heterogeneous systems. Level 4 is an adapted solution capable of negotiating and dynamically accommodating heterogeneous systems.

Table 6.1 presents the three critical barriers for enterprise systems interoperability: conceptual, technological, and system components. Each barrier has four significant concerns: business, process, service, and data integration. The maturity model for enterprise interoperability framework measures the interoperability degree of the enterprise systems in all the four major concern areas for each barrier.

The maturity model for enterprise interoperability framework is utilised, as a part of formative assessment, to measure the interoperability capability of the DIF. Section 4.3 demonstrated that the DIF is relevant and feasible to implement contemporary and modern architectural styles and integration patterns utilised by organisations. The interoperability maturity mapping table, Table 6.1, measures the interoperability of these DIF based architecture and system design implementations. The table illustrates the interoperability maturity mapping of these three integration architectures, based on:

- comparative logical reasoning of these architecture and design implementations as explained in Section 4.3
- practical experience of the author in developing the Hyperledger DLT solution prototype (HDSP)
- the industry experience of the author in integrating the IT systems

Type ---->	Conceptual	Technological	System Components
Concern			
Business	4/3/4	3/4/3	3/3.5/4
Processes	2.5/4/3.5	3/4/3	3/4/4
Service	4/3/4	3.5/4/3.5	3/3.5/4
Data	3/3.5/3.5	4/4/3	3.5/4/3.5

Table 6.1. Mapping the maturity levels for the **EDA**, **IAAES**, and **GIA** interoperability solutions, respectively. This table covers three major interoperability problem areas in the columns and four major concerns in the rows.

The interoperability maturity of EDA (see Section 4.3.2) is high on the conceptual interoperability of businesses because it provides the flexibility to connect among diverse business systems. The services can integrate seamlessly with the flow of events, and data can be exchanged across the systems by Kafka producers and consumers via a Kafka Broker. However, the interoperability maturity of EDA is relatively low in terms of integrating the processes because the events can be too granular, and stringing them together into business processes is comparatively more complex (this is an inherent drawback of event-driven architecture). A similar result is observed for system components interoperability, as the architecture expects interoperability to occur via a Kafka broker with an additional intermediate layer. The technological interoperability barrier is addressed well because of the seamless event flow among the systems using pub/sub integration patterns.

The IAAES's (see Section 4.3.1) interoperability maturity model score is high on technological and system components interoperability because it uses a combination of API Gateway, ESB, and Service Mesh to integrate systems and technologies, providing more capabilities and tools to integrate. The score is comparatively low for conceptual interoperability because the various layers (ESB, Service Control Plane, Orchestration, API Gateways) can become complex with time. Moreover, the functionality of these layers can become heavy and overlapping. Enterprises need to put more effort into designing the interoperability of business, service, and data, as these layers can obscure underlying system components' scope and capability. However, this problem is not due to DLT integration, and it is expected that enterprises will have sufficient clarity and governance on the functionality to be included in their DLT because other organisations are part of the distributed network. The IAAES involves two-way communication for all the components, so its interoperability capability is high; however, it also has the risk of complexity. Enterprises can integrate DLT system via ESB, Service Mesh, or API Gateways. However, a choice must be made regarding solution effectiveness, complexity, and operational manageability from a long-term perspective.

Generic Interoperability Architecture (see Section 4.3.3) is a flexible architecture design an enterprise can consider for its interoperability solution. It offers a simplistic conceptual and system component view, so the interoperability score is high. However, this flexibility comes with the technology choice and its design implementation cost, due to which the score is low on technology parameters. Generic Interoperability Architecture can adopt API Gateways, ESB, Service Mesh or Apache Kafka to implement their interoperability requirements. However, judiciously choosing an approach and consistently applying for the majority of the enterprise interoperability requirement will help to reduce the technology implementation risk. The DLT producer and consumer can become a heavy layer as enterprises add interoperability solutions and interacting systems or devices. Enterprise can consider breaking it into multiple layers based on the technology approach for implementation and architecture requirements.

6.3 DIF Evaluation by IT Professionals

Three IT professionals (Participants 1 to 3) were selected to assess the integration capability of the DIF. This selection is based on the guidelines suggested by Prat et al. [89] to execute the generic evaluation method by practitioners. The research adopted the approach to undertake evaluation by a small set of practitioners, those who are very experienced and expert in their domain areas, instead of evaluating from a larger set of relatively less experienced practitioners. The brief experience of the three participants is summarised next.

Participant 1 had 25 years of professional experience in the IT industry as an integration specialist, data migration expert, and ERP implementation and software development manager.

Participant 2 had 20 years of professional experience as a data expert, data integration specialist, data warehousing consultant, and IT manager.

Participant 3 had 28 years of experience in the IT industry in broad and diverse roles such as executive manager, solution architect, business analyst, and development manager.

A one-hour meeting was scheduled with the three participants to:

- explain the aims and objectives of the research
- explain the rationale of the DIF principles and the interoperability framework
- demonstrate the working of the HDSP
- explain the maturity model for enterprise interoperability (MMEI), as summarised in Table 6.1

In the last 20 minutes of the meeting, each participant was requested to independently assess the interoperability capability of the DIF using the MMEI. The interoperability score on MMEI framework will indicate the integration effectiveness of DLT solutions with non-DLT technologies in HDSP. They were requested to rate the DIF in increments of 0.5 to address the four interoperability concerns (i.e., data, service, process, and business) among the three integration barriers (i.e., conceptual, technological, and system components). The results are presented in Table 6.2.

		Participant 1	Participant 2	Participant 3	Average
Conceptual	Business	3	3.5	4	3.5
	Processes	3	3	3.5	3.2
	Service	3.5	3	3	3.2
	Data	4	3.5	3.5	3.7
Technological	Business	2.5	3	2	2.5
	Processes	2.5	2.5	2.5	2.5
	Service	3	3.5	3	3.2
	Data	3.5	4	3.5	3.7
System Components	Business	3	3	3.5	3.2
	Processes	3.5	3	3	3.2
	Service	2.5	3.5	3	3.0
	Data	3	3.5	3.5	3.3

Table 6.2. Mapping of the maturity levels of the Interoperability Framework by three professional participants, covering three significant interoperability barriers and four major concerns for each barrier.

The last column in Table 6.2 shows the average score of the participants for a particular concern within an integration barrier. The average interoperability score of the DIF is "good" (3.0 and above, based on MMEI model explained in [28]) in all areas except business and processes concerns for the technological barrier. Before concluding the meeting, a brief discussion of the possible reason for the low score for business and processes concerns happened. Conceptually and at the components level, its comparatively easy to integrate the solutions; however, the integration landscape becomes very broad and diverse from a technology perspective. To integrate businesses and processes (from a technology perspective), many technology platforms, operating systems, protocols, a lack of standards, implementation challenges, and operational complexities must be overcome. Because DLT is a new technology, there are few accepted standards and the experience in integrating DLT services and solutions is limited, resulting in low confidence and a low score on these two concerns.

6.4 DIF Validation by Instantiating Solution Prototype

The HDSP is an interoperable DLT solution prototype for the MHSC use case. The interoperability of the HDSP is due to the architecture, solution design, and technical instantiation derived from the DIF design, components, and its interoperability principles, and not from some circumstances or independent confounding variables. These factors established the interoperability of the DLT-based solution, the HDSP, with centralised technologies. The design and integration pattern of the HDSP is in line with the SOA and modern distributed architecture, establishing its relevance to contemporary organisations.

The Hypderledger Fabric DLT solution prototype (HDSP) is an object of the technical instantiation of the DIF and its interoperability principles. Design science research researchers such as Herselman et al. [133], Cleven et al. [88], and Venable et al. [128] imparted significant importance to instantiation validity if an artefact (the HDSP in this research) instantiates design theory, model (the DIF in this research) or design principles (interoperability principles in this research). As per Venable et al.'s research [128], technological artefacts such as the HDSP must undertake artificial or technical evaluation. The HDSP represents a physical realisation of the DIF (design representation of the solution) and its principles functioning in the natural world and has been widely accepted as a core type of DSR contributions [127]. Modern and contemporary technology stack was used to implement the HDSP using a naturalistic evaluation, in addition to a technical evaluation of the HDSP.

6.5 Evaluation to Assess HDSP Design Options

The assessment to decide the appropriate business use case, DLT platform, non-DLT technology stack, architecture, and data format choice, was undertaken before initiating the design of the HDSP. The correct decisions during this assessment enabled the research to execute the research process in a cost-effective and timely manner, enhancing the quality of design artefacts and design process and reducing the calendar timelines of the project. This initial assessment assisted in improving the outcome of the design research process and provided a foundation for the other assessment techniques, triggering continuous improvement of the HDSP. The following evaluation was undertaken:

1. **Evaluation to Determine Instantiation Use Case**

   A business use case was required to design DLT-based solution and integrate it with the centralised IT system. As explained next, the literature guided the choice of an appropriate business context in which to implement the HDSP based on the DIF.

   a. Casino et al. [134] conducted a systematic literature review of DLT-based applications among multiple domains. Their study presented an inclusive classification of DLT-enabled systems across various sectors, such as supply chains, healthcare, governance, humanitarian aid supply chains, and finance. Casino et al. [134] claimed that DLT was expected to improve transparency, accountability, and trust amongst the global and unknown supply chain stakeholders, enabling more trusted, efficient, and transparent value chains. The supply chain domain is one of the few business domains on which DLT is expected to provide the most positive impact.

b. Foreseeing the possible positive impact of DLT on food supply and safety, IBM and Walmart established a collaboration to implement BC technology to develop food provenance, traceability, and transparency solutions in the food supply chain. In addition, they aimed to digitise the food supply, safety, product information, and food processing using a DLT-based solution [6].
c. Larsen et al. [105] provided recommendations, suggestions, and guidelines for the design and structure of DLT case studies in a supply chain business context as compared to other business domains. Furthermore, Peterson et al.'s [103] 's research emphasised that implementing DLT to supply chain and logistics provides a wealth of opportunities for enterprises in this business domain.

This evaluation and Section 5.2.1 (Supply Chain Use Case Selection) helped decide to choose the manuka honey supply chain (MHSC) domain to instantiate a DLT-based technical prototype.

2. **Evaluation to Determine DLT Platform**

   The research evaluated prominent DLT platforms to implement and instantiate the technical prototype of the MHSC use case. Section 5.2.2 (Distributed Ledger Technology Platform Selection) details the evaluation undertaken for different DLT platforms before choosing a suitable one (Hyperledger Fabric 1.4) for the HDSP instantiation.

3. **Evaluation To Choose Technology Stack**

   The modern popular technology stack used in contemporary enterprises was chosen to develop the HDSP [135]. It enabled the establishment of the interoperability capability of DLT platform with cutting-edge and futuristic IT systems and technology stacks. The website WebFX [136] and the review of internet sources, such as Basias et al. [30] and Biel et al. [27], enabled the choice of the right technology stack. By adopting the modern web-based, API-centred, and Cloud-enabled technology stack used by enterprises, the natural evaluation factor was included while undertaking the artificial (technical) evaluation of the HDSP.

4. **Choosing Architecture for the Design of the HDSP**

   Service Oriented Architecture (SOA) is widely used for designing modern distributed systems, Basias et al. [30], Leotta et al. [16], and Yarberry et al. [99] emphasise the success and wide adoption of SOA principles and practices in enterprises. The HDSP followed the SOA architecture style for designing the solution.

5. **Choosing a Data Format:**

   This research evaluated the XML and JSON data formats and followed Svetashova et al.'s [73] suggestion to use the JSON data format. This format enabled a well-defined data structure, making data interoperable and easy to understand, and helping to streamline the data processing. The JSON data format and API-driven design of the system enabled data interoperability among the system components. The sample RESTful input message from the producer, in JSON format, is presented in Table 6.3.

```
{    “jarId”: “50”,
     “umf”: “150”,
     “jarWt”: “1000”,
     “batch”: “D21062020”,
     “lab”: “Analytica”,
     “floral”: “Mono”,
     “tutin”: “1”,
     “pname”: “Comvita”,
     “paddress”: “Hamilton”,
     “pcontact”: “0295412547”,
     “plicence”: “NZ0123470235”,
     “cost”: “250”    }
```

Table 6.3. Sample RESTful input message from the producer, in JSON format.

6.6 HDSP Formative and Summative Evaluation

Formative evaluation of the HDSP validated the decisions taken before and during the design phase, evaluating the design research process and ensuring the desired level of rigour. The summative assessment of the HDSP was executed when the first iteration of product development was completed, instantiated, and ready for testing. As shown in Figure 6.2, the insights from the summative evaluation were fed as input to the formative assessment to improve the artefact design and development for enhanced research rigour and final product. The HDSP was developed using the modern and popular contemporary technology stack employed by current organisations, covering the naturalistic evaluation aspect of the HDSP [135].

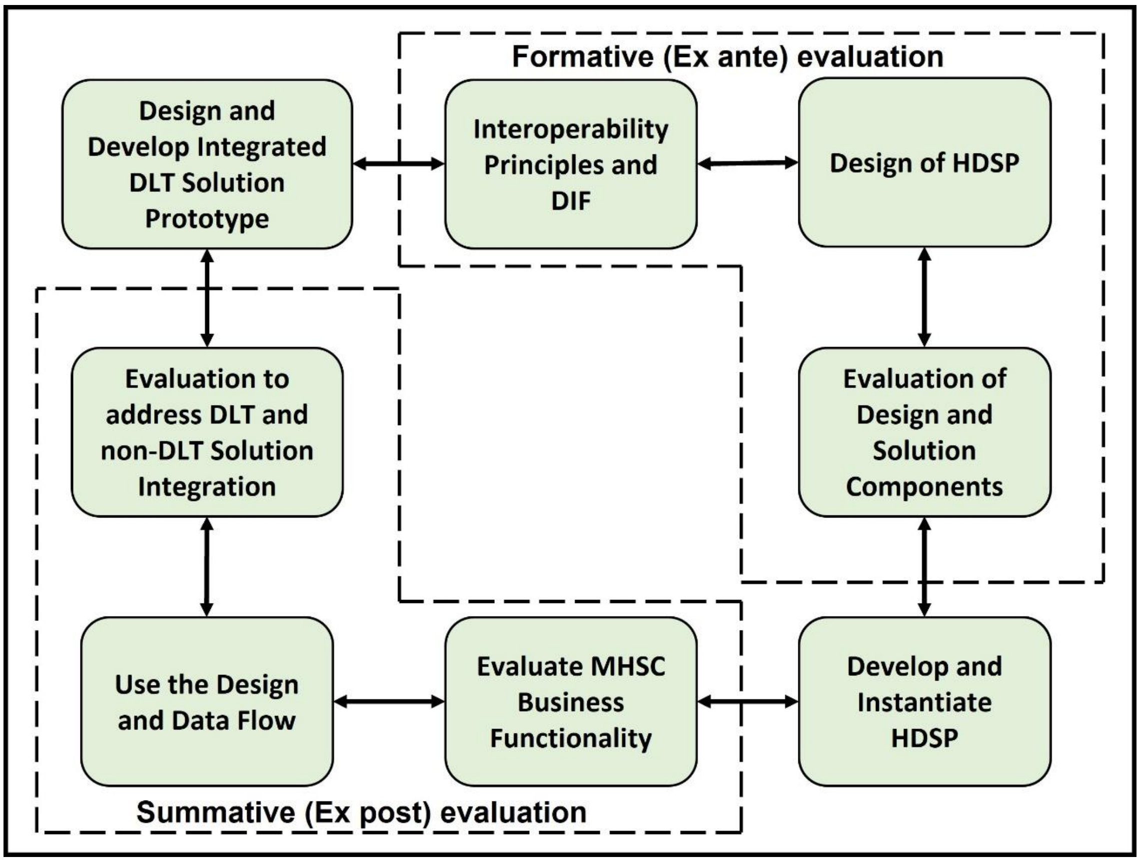

Formative (Ex ante) evaluation
Design and Develop Integrated DLT Solution Prototype
Interoperability Principles and DIF
Design of HDSP
Evaluation to address DLT and non-DLT Solution Integration
Evaluation of Design and Solution Components
Use the Design and Data Flow
Evaluate MHSC Business Functionality
Develop and Instantiate HDSP
Summative (Ex post) evaluation

Figure 6.2. Iterative formative and summative assessment undertaken for the HDSP. Adapted from [86].

6.6.1 HDSP Formative Evaluation

The architecture design and solution components of the HDSP were evaluated in the formative evaluation of the artefact. The authors comparative logical reasoning and practical experience were utilised to validate that the HDSP solution design was in line with the interoperability principles of the DIF. The solution components and their interactions were evaluated to verify that the HDSP followed the DIF components, purpose, and message interactions. Two cycles of summative assessment were undertaken; after each cycle, the formative assessment was repeated to enhance the HDSP.

6.6.2 HDSP Summative Evaluation

Two iterations of summative evaluation of the HDSP were conducted. The focus of each iteration stage was to evaluate the interoperability of DLT solution with the centralised IT systems. The HDSP demonstration (Section 5.1.2) validated that the solution fulfilled the essential MHSC business criteria (criteria validation). The solution met the semantic and data interoperability expectations for the business use case and validated that the DIF based DLT solution integrated with the current IT system, and therefore the overall solution augmented the current business's capability. Koussouris et al. [149] discussed other types of interoperabilities, like rules, object and software interoperabilities. Many of these interoperabilities are covered implicitly by the artefacts, e.g.:

- Rules Interoperability: When enterprises join DLT network, they agree on specific fundamental rules based on which the network functions and Smart Contract executes.
- Objects Interoperability: Every object (IoT, systems, services, NFC etc.) is interconnected via APIs. Earlier the APIs were not standardised, the newly established RESTful standard simplified the object interoperability.
- Software System Interoperability: Modern software systems are based on SOA, so integration is enabled by APIs, ESB or microservices design patterns.

The evaluation methods executed and their outcomes are discussed further. The first summative evaluation was initiated after the initial design, development, and instantiation of the HDSP. The outcome of the first summative evaluation triggered changes in the artefacts design. The first summative evaluation changes were included in the HDSP before initiating the second summative evaluation. The process was repeated to incorporate the enhancements from the second summative evaluation of the HDSP.

a. **Enhancements due to the First Iteration**: This evaluation was triggered when the HDSP was instantiated the first time to undertake the testing. The program *invoke-transaction.js* was activated from the HF node with the virtual layer of *app.js*. The application was functioning; however, this enabled the tight coupling of the HF node interface with the HF core component. The first summative assessment identified the need to provide a flexible interface from the devices and DLT nodes (user) to the HF network. This requirement was fulfilled by adding API Layers (Node.js) to integrate the enhanced system from various devices and systems to the HF network. The addition of the *app.js* component provided the flexibility to choose the user interface the peer node wished to use to inject the transaction.
b. **Enhancements due to the Second Iteration**: The *app.js* layer was made functional. This API layer provides the flexibility to interact with any message or

API-driven system, protocol or application from the HF node to interact with fabric core components and smart contract (chain code). The customer (MH end-user) node was part of the HF network.

Based on the test cases execution and literature review, the customer node was removed from the HF network. This node did not improve the trust and transparency provided by DLT solution to the MHSC. Instead, a user interface (UI) was provided to meet customers' MH jar information requirements.

6.7 HDSP Demonstration and Test Cases Execution

The HDSP instantiation demonstrated that the DIF promoted the architecture and solution design to integrate DLT solution with centralised IT systems. The working functionality of the instantiation of the HDSP was presented to seven department academic staff, four fellow researchers, and five industry peers. The honey producer, distributor, and retailer could push the transaction into the HF network, and transactions were committed to the ledger on all the nodes, improving the trust and transparency amongst the HDSP network participants. Transactions were reliably pushed into the network, committed on the HF ledger, and saved on the MongoDB database to be accessed by centralised systems (customer UI). The MHSC testes cases executed during the HDSP demonstration were captured, confirming the solution met the business criteria. The transaction execution time of these test cases validated the solution's practicality endorsing the working of a DLT interoperable solution. The transaction time between 2 to 3.5 seconds is pragmatic considering that the HDSP and its three nodes are executed on a medium-sized Linux server on an EC2 instance. It would indicate the scalability potential of the solution if more IT infrastructure resources were deployed for DLT solution instantiation. The detailed analysis of DLT application performance, throughput and scalability is considered out of the scope of this research, which is discussed in the studies conducted by Kuzlu et al. [139] and Kuhi et al. [140].

The test cases executed are documented in the Appendix, and the transaction execution time is shown in Table 6.4. This demonstration meets the evaluation objective of DIF by solution instantiation.

Transaction	Event	MH Producer Transaction	Exececution Time(sec)	MH Distributor Transaction	Exececution Time(sec)	MH Retailer Transaction	Exececution Time(sec)
Trans 1	Start	"2020-09-23 22:42:41.579"	2.184	"2020-09-23 22:46:40.147"	2.170	"2020-09-23 22:48:25.599"	2.172
	End	"2020-09-23 22:42:43.763"		"2020-09-23 22:46:42.317"		"2020-09-23 22:48:27.771"	
Trans 2	Start	"2020-09-23 22:15:34.785"	2.230	"2020-09-23 22:21:57.843"	2.812	"2020-09-23 22:23:45.563"	2.200
	End	"2020-09-23 22:15:37.015"		"2020-09-23 22:22:00.031"		"2020-09-23 22:23:47.763"	
Trans 3	Start	"2020-10-09 14:10:56.876"	2.355	"2020-10-09 14:12:43.334"	2.228	"2020-10-09 14:15:01.745"	3.060
	End	"2020-10-09 14:10:59.231"		"2020-10-09 14:12:45.562"		"2020-10-09 14:15:04.805"	
Tran 4	Start	"2020-10-09 16:01:12.985"	2.122	"2020-10-09 16:04:36.678"	2.406	"2020-10-09 16:05:55.412"	2.467
	End	"2020-10-09 16:01:15.107"		"2020-10-09 16:04:39.084"		"2020-10-09 16:05:57.879"	
Tran 5	Start	"2020-10-15 11:32:16.541"	2.917	"2020-10-15 11:36:41.746"	2.428	"2020-10-15 11:41:51.214"	2.244
	End	"2020-10-15 11:32:19.458"		"2020-10-15 11:36:44.174"		"2020-10-15 11:41:53.458"	
Tran 6	Start	"2021-04-24 21:05:54.984"	3.118	"2021-04-24 21:10:25.478"	3.280	"2021-04-24 21:14:47.057"	2.901
	End	"2021-04-24 21:05:58.102"		"2021-04-24 21:10:28.758"		"2021-04-24 21:05:49.958"	
Tran 7	Start	"2021-04-24 19:07:34.214"	2.91	"2021-04-24 19:11:41.415"	2.694	"2021-04-24 19:15:21.745"	3.333
	End	"2021-04-24 19:07:37.124"		"2021-04-24 19:11:44.109"		"2021-04-24 19:15:25.078"	
Tran 8	Start	"2021-04-24 18:41:14.478"	2.77	"2021-04-24 18:47:25.749"	3.434	"2021-04-24 18:53:55.358"	2.989
	End	"2021-04-24 18:41:17.248"		"2021-04-24 18:47:29.183"		"2021-04-24 18:53:58.347"	
Tran 9	Start	"2021-04-24 18:20:52.364"	2.325	"2021-04-24 18:26:14.873"	2.478	"2021-04-24 18:35:23.785"	2.471
	End	"2021-04-24 18:20:54.689"		"2021-04-24 18:26:17.351"		"2021-04-24 18:35:26.256"	
Tran 10	Start	"2021-04-25 13:52:45.781"	2.943	"2021-04-25 13:58:62.278"	2.403	"2021-04-25 14:02:39.459"	2.798
	End	"2021-04-25 13:52:48.724"		"2021-04-25 13:58:64.681"		"2021-04-25 14:02:42.257"	

Table 6.4. Transaction execution time for the HF networks three peers, honey producer, distributor, and retailer nodes.

6.8 How much Evaluation is Enough?

Design science research puts significant emphasis on evaluating and validating the artefacts produced in research. However, the question, “how much evaluation is enough?” for artefact evaluation activity, needs to be addressed for any study limited in financial, time, and other resources. The five Design Evaluation Methods (DEM) suggested by Hevner et al. [80] were followed to validate that the research followed a sufficient evaluation process. The evaluation strategy adopted and the evaluation activities conducted in this research met the evaluation expectation set by Hevner et al., as explained next.

1. **Observational DEM**: In the observational method, the case study approach is suggested for evaluating the artefact in a business environment. The usefulness of the HDSP in the supply chain business context (use case of MHSC) validated the utility of the DIF in developing interoperable solutions.
2. **Analytical DEM**: Analytical analysis is suggested to evaluate artifacts' fitment into the information system architecture. The DIF was implemented to design the enterprise interoperability architecture, and HDSP architecture was utilised to enhance the Generic Interoperability Architecture, described in Section 4.3.3.
3. **Experimental DEM**: This method recommends artefact simulation and experimentation in a controlled business environment. The HDSP artefact was simulated for the MHSC use case by using artificial data. It was executed in a controlled environment to test the interoperability of the HDSP without paying attention to other less relevant aspects of this study, such as trust and transparency.
4. **Testing DEM**: This method suggests black box (functional) and white box (structural) testing. The HDSP executed functional testing to verify that both DLT solutions and centralised systems executed the transactions performed by the three nodes on the HF network. The HDSP structural testing was undertaken to trace the transactions' execution path and analysed to optimise the HDSP (e.g., designing the components of HDSP and moving the customer node out of the HF network).
5. **Descriptive DEM**: Informed argument was utilised to use the available knowledge base of DLT integration solutions to define the interoperability principles of the DIF. The informed argument used the current interoperability practices (architecture, design, and tools) to design the interoperability architecture in Section 4.3 (based on the DIF) to establish a convincing argument for the DIF utility. The MHSC business scenario was created to implement the DIF by instantiating the HDSP to demonstrate the utility and usefulness of the DIF.

6.9 Summary of Artefact Evaluation

The artefacts produced by this study, the DIF and HDSP, utilised relevant evaluation methods and techniques to validate the research process and usefulness of the artefacts. The summary of these evaluation techniques is plotted in Figure 6.4.

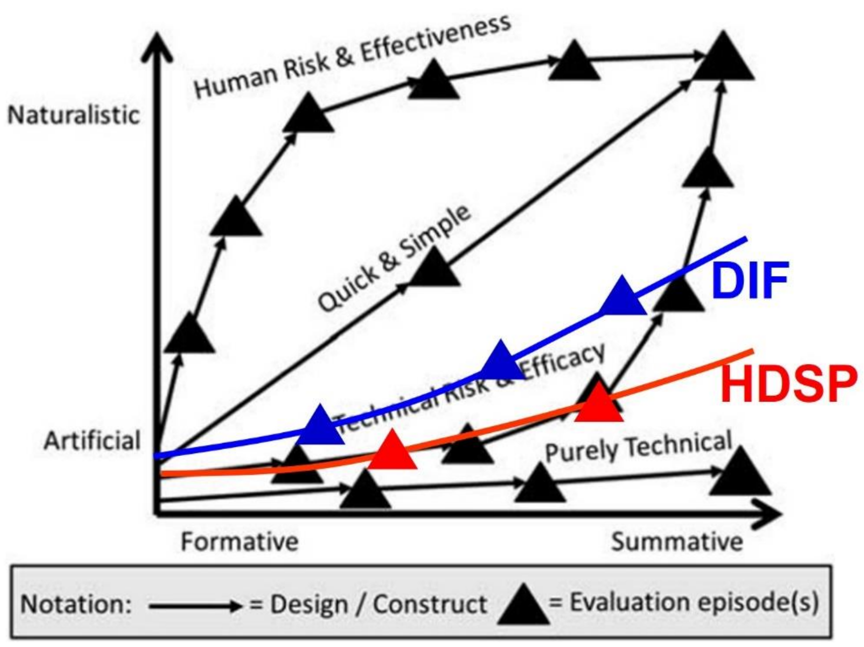

Human Risk & Effectiveness
Naturalistic
Quick & Simple
DIF
Technical Risk & Efficacy
HDSP
Artificial
Purely Technical
Formative
Summative
Notation: = Design / Construct = Evaluation episode(s)

Figure 6.3. Evaluation Framework with suggested evaluation summary. The curves for the DIF and the HDSP represent the evaluation summary for the two artefacts; the triangles on these curves are the summative evaluation of the respective artefacts. Adapted from [128].

The formative evaluation of the DIF enabled the design and enhancement of its components. When the DIF reached a particular development milestone, a summative assessment was conducted. The DIF was implemented to design interoperability architecture for popular enterprise architecture, interoperability design patterns and tools (see Section 4.3). For each design implementation of the DIF, a summative assessment was triggered. As part of this iterative evaluation process, the outcome of the summative assessment was fed into the formative evaluation of the DIF. This iterative evaluation process enabled the refinement of the DIF components to produce a robust and practical integration framework.

The application of the maturity model for enterprise interoperability (MMEI) framework to measure the degree of interoperability specifies the integration strength of the DIF. This is expected to provide the motivation to utilise the framework to design integrated solutions. The evaluation undertaken by three IT professionals established the relevance and strength of the suggested framework from an industry perspective. The implementation of the HDSP, based on the DIF, assured a practical and implementable solution when using the framework.

Formative evaluation of the HDSP validated the decisions taken before and during its design and development, evaluating the design research process and ensuring the desired level of rigour. The summative assessment of the HDSP was executed when the product development had been completed, instantiated, and was ready for testing. The insights from the summative evaluation were fed as input to the formative assessment, improving the design and instantiation of the final product (the HDSP). In addition, the MHSC testes cases executed during the HDSP demonstration were captured, confirming the solution met the business criteria and endorsing the working of a DLT interoperable solution. The transaction execution time of these test cases validated the solutions practicality and indicated the scalability potential if more IT resources were deployed for DLT solution instantiation.

Chapter 7: Research Summary

During the last decade, DLT became popular by establishing trust amongst non-trusted or unknown peers through real-time, distributed, digital, immutable and secure transactions and the technology capability has been maturing. Many technology characteristics have emerged distinguishing DLTs based on their usefulness for the specific business domain. The major category is public and private DLT network technology: public DLT is useful for a business where untrusted or unknown persons or entities can join DLT network. A private DLT network is applicable where the participants are trusted or partially trusted to execute the transactions. Hyperledger Fabric, Ethereum, Corda R3, are suitable technology platforms for private DLT networks. Integrating DLT-based solutions with the traditional IT system is crucial for enterprises looking to implement DLT solutions in their organisations. The analysed DLT integrated solutions are not comprehensive and complete, and to address this gap, this research developed an interoperability framework (the DIF) based on interoperability principles. This framework enables enterprises (participating in a private DLT network) to design an integrated solution, establishing a synergy with the current non-DLT based IT capability with emerging DLT solutions. To provide the evidence of DIF usefulness, the HDSP solution was built, demonstrating the successful integration of the Hyperledger Fabric 1.4 solution prototype with the traditional IT systems.

This chapter summarises the research undertaken, its artefact outcome, the contribution made in the DSR domain, the research limitations, and suggestions for future research. The following section recaps the identified research problem and its significance. Section 7.2 describes the research contribution of developing the DIF and interoperability principles and building the HDSP artefact. The evaluation of both artefacts to justify their utility is revisited in Section 7.3. The next section summarises the DSR knowledge contribution made by this research, and Section 7.5 explains the research limitations. The final section suggests the potential for future research to extend the knowledge developed in this thesis.

7.1 Research Problem and its Significance

DLT offers significant benefits to organisations wishing to improve their current business processes or establish new ones to enhance and expand their business capabilities [63]. Almost all universities, research entities, and organisations invest in or closely monitor technologys progress and breakthroughs [141]. These entities experiment with the utility and usefulness of DLT in every possible domain to realise its claimed benefits. High-profile collaborations such as IBM and Walmart's partnership to improve the supply chain process are taking place to leverage the strength of each other's capability so they can realise the significant business benefits of DLT implementation [6].

However, it is unlikely that DLT and its solutions will replace the current functioning solutions based on centralised IT systems. Instead, DLT-based solutions will augment business capability by improving the current processes or defining new ones. Organisations will not expect DLT-based solutions to work in isolation or silos, but extend the capability of implemented IT solutions. For this, DLT-based solutions must collaborate and integrate with centralised IT solutions. Without the seamless integration between DLT-based and centralised IT systems, enterprises will be less interested in investing and implementing DLT-based solutions. This research was focused on this critical challenge of establishing interoperability between these DLT and non-DLT based solutions. Based on this, the research developed a DLT interoperability framework (DIF). It proposed the integration architecture designs and integrated DLT-based solutions that enterprises can consider implementing into their current integration architecture and IT solution stacks.

To realise the benefits of DLT, the research community is recommended to address the interoperability challenge of DLT solutions with the current IT ecosystem. To design the solutions and artefacts to meet these challenges is a complex process and it expects a creative advance in the solution and artefacts domain. To address this problem, DSRM is utilised, which is a problem-solving paradigm rooted in the science of artificial and engineering [91]. The design science research methodology (DSRM) addresses wicked and real-world problems by designing and developing innovative artifacts and system knowledge, following the design science research process, methodology, and activities. The DSRM is used to define the problem and its importance, establish solution objectives, design artefacts to meet the objectives, evaluate the artefacts, demonstrate the solution and its benefits, and communicate the research outcomes to the research community. The research also followed the seven guidelines and fundamental principles of DSR to establish the artefacts to meet the research objectives. In designing the two artefacts, the generated knowledge was able to contribute to DSR project design knowledge.

7.2 Research Contribution

The literature review of academic and professional journals conducted in this research identified that:

- the lack of integration of DLT-based solutions with traditional IT systems is one of the primary obstacles to the broader adoption and implementation of DLT
- analysed DLT integrated solutions from the literature review are inadequate for addressing the interoperability challenges faced by organisations

This research examined the available DLT solutions integrated with centralised IT systems. Based on these solutions and considering the integration requirements and expectations of modern enterprises, interoperability principles were formed and explained. Based on these principles and integration criteria, an integration framework, the DIF, was conceptualised and modelled. The framework has significant implications for practitioners and researchers to:

- enable enterprises to design an integrated solution integrating a DLT-based solution with traditional IT systems;
- provide pathway in synergising the new DLT-based capability with current non-DLT based capabilities in an organisation;
- demonstrate that the contemporary interoperability architecture, integration patterns and tools popular in modern enterprises are relevant and fit for purpose with the suggested integration framework;
- instantiate a practical, interoperable technical solution integrating DLT system with centralised IT systems to provide implementation example of DIF-based solutions; and
- future research and a DIF-based implementation are expected to further refine the components of the DIF. This might establish what might be termed a "golden interoperability framework" that professionals and enterprises can reference to enhance trust in an integration solution.

As these points suggest, this research designed, developed, and instantiated the HF DLT solution prototype (HDSP) for the MHSC business case to:

- validate that the integrated DIF-based designed solutions are practical and implementable; and
- ensure that DIF-based solutions establish seamless interoperability between DLT and non-DLT technologies at the technical, semantic, syntactic, data, and message levels.

7.3 DIF and HDSP Evaluation

In design science research, the evaluation of artefacts includes evaluating the design research process, artefacts produced, and knowledge contribution. Artefact evaluations require different expertise and entail substantial resources; research need not extend to evaluation if the design solution is novel or offers a significant improvement (as explained in Section 6.1). However, a reasonable level of evaluation provides trust to the artefact users in deploying the artefacts and utilising the knowledge supporting the artefacts. This research utilised formative (ex-ante) and summative (ex-post) evaluation, artefact instantiation and maturity model for enterprise interoperability (MMEI) evaluation techniques to evaluate the artefacts. The established DSR process was followed, as explained in Chapter 3, and the research knowledge contribution was documented.

The DLT interoperability framework (DIF) was evaluated rigorously and iteratively:

1. The DIF components were validated against the interoperability principles to ensure the DIF met the required expectations.
2. The DIF was evaluated based on the interoperability maturity model, the maturity model for enterprise interoperability (MMEI). This method was applied for event-driven Distributed Architecture (EDA), Interoperability Architecture using API Gateways, ESB and Service Mesh (IAAES), and Generic Interoperability Architecture (GIA). This provided the evidence that the DIF was relevant to contemporary interoperability architecture, integration patterns, and tools used in modern enterprises.
3. Information technology professionals utilised the MMEI framework to assess the interoperability capability of the DIF. This established the utility of the framework in industry and professional domains.
4. An interoperable DLT solution (the HDSP) was instantiated based on the DIF. This demonstrated the implementability and practicality of the solutions designed based on the DIF.

The HDSP's formative assessment was undertaken to:

- identify the relevant use case for the solution instantiation in the domain where DLT-based solution could produce the most favourable business impact

- determine DLT platform most appropriate and relevant to modern enterprises for the selected MHSC use case
- choose the centralised technology stack to design the solution prototype that modern organisations commonly adopt
- determine the HF network peers or nodes to enable the accurate selection of network participants to optimise the solution's benefits by managing the technical complexity

Two iterations of the summative evaluation of the HDSP were undertaken, and the outcomes were incorporated to iteratively enhance the HDSP solution. These also evaluated the extent to which the interoperability principles (e.g., API driven design) and the integration requirements of modern enterprises were incorporated into the design of the HDSP. A criterion validation was undertaken to ensure that the HDSP instantiation met the supply chain functionality criteria and integrate DLT and non-DLT solutions. Semantic and data interoperability was undertaken, and the technical interoperability was validated by the fact that the prototype demonstration worked as expected. The working demonstration of the HDSP was recorded, and test cases execution data and transaction execution times were reported.

7.4 Knowledge Contribution

This research has made a significant contribution to knowledge in the DSR project knowledge base. The DLT interoperability framework and its interoperability principles have been fully explained. Based on the solution context and prerequisites, these principles and rules can be modified, dropped, or enhanced to meet the specific requirements of the enterprise solution. These principles also enable the DIF components and their interactions to be sufficiently flexible to meet the expectations of a particular enterprise's IT architecture or ecosystem. The interoperability architecture was developed and explained for commonly used integration patterns and tools popular in modern enterprise IT ecosystems. This will help enterprises modify and enhance the integration architecture and its component, to build an appropriate architecture for their specific needs.

The solution architecture and the end-to-end message flow of the HDSP can be replicated from the MHSC solution to other similar use cases. The prototype's messages and data flow pattern are centred around the API-driven design of the solution, a popular system design pattern in the industry. The characteristics of the solution explain the foundation principles based on which the solution prototype was built, and are significantly relevant for DLT-based interoperable solutions. The event handling process has strong utility for many business situations among the different DLT platforms available to manage business events originating from IoT devices, centralised systems, or DLT-based solutions. The recommended software practices to build DLT-based solutions can benefit practitioners wishing to build similar solutions. It details the best practices and explains the learning outcome of this research to develop API-driven architecture, data design, building a smart contract, and deploying and operationalising DLT-based solutions.

7.5 Research Limitations

This research has made a significant contribution by developing the interoperability principles, the DIF based on these principles, and the HDSP solution design and implementation based on the DIF. However, the research has three limitations due to the constraints on the project's timeframe, cost, and related resources. After reflecting on the research process and outcomes, these limitations were identified as follows:

1. The artificial or technical evaluation processes and techniques used in the research have limitations. Natural evaluation is supposed to have been undertaken by implementing the DIF framework and its interoperability principles for an enterprise IT ecosystem. Ideally, the research is supposed to have chosen an organisation interested in DLT solution implementation, studied its IT eco-system and integration architecture, and implemented a multi-DLT network solution to improve its business process and integrate this DLT-based system with the current IT system in the selected organisation. However, an organisation needs a business commitment and significant investment to implement such a broader impacting solution.
2. Instead of a single DLT network, the HDSP solution have been implemented for a multi-DLT network. This could have enabled the design, development, and instantiation of all the components of the DIF. However, there are currently few multi-DLT network implementations; hence, this must not discourage organisations from implementing the interoperable solution provided in this research. However, this can be considered as a future research opportunity to advance multi-DLT interoperability.
3. This research has not delved into the performance analysis of HDSP's transaction throughput, latency and scalability. The study of improvement in the number of transactions per second, transaction response time per second and number of

participants a platform can serve is beyond the scope of this research. It can be referred to in the research conducted by Kuzlu et al. [139] and Kuhi et al. [140].

7.6 Suggestions for Future Research

Interoperability is a major concern and focus area of many enterprises, especially with emerging technologies such as the IoT and DLT. For DLT to be a mainstream and successful technology, it must co-exist and communicate with current technologies and IT ecosystems. This research has focussed on this aspect and designed the DLT interoperability framework (DIF) and built the HDSP, establishing an integrated DLT solution with centralised IT technologies. However, more research is required to provide trust and conviction to enterprise stakeholders and decision-makers to invest in and implement this technology. The following sections explain the possible research areas researchers can focus on to improve the adoption and implementation of this promising DLT.

7.6.1 Interoperability Framework Instantiation

This research developed a DLT interoperability framework that can be instantiated in an enterprise environment. Future research can consider an enterprise use case to design DLT architecture and instantiate its components to design and develop the integration architecture for their IT ecosystem. Such a study would demonstrate the working instantiation of the interoperability architecture based on the DIF. It could also conduct structured or semi-structured interviews and data collection involving the organisation’s IT staff, business analysts, and business stakeholders to measure the utility and effectiveness of the DIF. Based on the study, the DIF and its components can be further refined and improved.

This research suggested software engineering practices for DLT solution design, development, and instantiation, which can be experimented with and validated in the enterprise environment. This knowledge can further feed enterprises' learning outcomes and DLT solution development experience into the DSR project knowledge base. This research has also suggested the DIF-based interoperability architecture for prevalent integration patterns and tools in modern enterprises. The learning outcomes from these designs, enhancement in the architecture, and the insights gained into the components and layers of the DIF can further feed the expertise into the knowledge base to improve the DIF. This can help establish a golden interoperability framework that can be referred to by broader organisations and practitioners, boosting trust in the interoperability of DLT-based solutions and systems.

7.6.2 Cross-DLT Interoperability

Organisations are endeavouring to leverage the technological capabilities of DLT and attempting to build distributed solutions. However, it is also acknowledged that no single solution will be perfect for addressing all or most issues. For example, the IOTA DLT uses direct acyclic graph (DAG) technology to improve the micro-payment capabilities among IoT devices and networks. Stellar DLT has been built to smoothen the global payment network functioning and use VeChain DLT system to strengthen supply chain business management. With the proliferation of such DLT projects, various technology flavours are used to handle specific business needs, data quantity, network expectations, and the specific requirements of governments, businesses, and social organisations.

In the near future, there will be an expectation of inter-DLT interoperability among these specialised DLT platforms. This integration will enable the transfer and exchange of messages and data among different DLT networks. Even though some DLT networks might work in isolation to meet specialised purposes, the data and processing capability will be helpful for other DLT networks and centralised IT systems, which must be transferred and shared among the networks. Without developing such an ability, the full benefits of DLT will not be realised. Therefore, future research can focus on cross-DLT interoperability so that these diverse DLT platforms can communicate with one another with simple standards, building a truly decentralised and integrated network.

7.6.3 Decentralised Applications Interoperability

DApps (decentralised applications) run on decentralised or distributed technology. The next stage of technological innovation is where DLT forms the infrastructure, and the Smart Contracts bind the application users and network participants. Soon, DApps may mimic or replace applications like Facebook, Uber, Airbnb, because they are cost-effective, trustworthy, and running on decentralised infrastructure. There are many popular DApps based on specialised DLT, meeting the specific demands of businesses and users. However, few DApps use the limited integration capability across the applications; for example, HyperDragon can use kittens or tokens collected from CryptoKitties to progress functionality. However, this area has a significant research gap for building the integrated DApps integrating among DLT platforms and centralised IT systems.

Until recently, interoperability within and among DLT and centralised IT systems was considered a significant challenge. However, recent advancements in integrating these diverse technologies have provided conviction to researchers and practitioners in using the cross-technological capabilities. In DApps, integration offers significant benefits such as:

a) smooth information sharing among the systems and applications
b) execution of Smart Contracts across DLTs
c) sharing DLT-based solutions and collaborating on enterprise development; and
d) gaining deep knowledge in a few DLT platforms and reusing the capabilities of other DLT platforms.

The need for the DApps interoperability framework will grow as more platforms, applications, distributed networks, and systems are developed. Platforms like Polkadot, Ark, Cosmos, have realised this importance; however, more researchers and institutes need to collaborate to enable practitioners to build interconnected applications to tap the capability of DLT and non-DLT based applications.

References

[1] C. Cachin and M. Vukolić, “Blockchain Consensus Protocols in the Wild.,” *IBM Research*, p. 24, 2017.

[2] A. Kumar, R. Liu, and Z. Shan, “Is Blockchain a Silver Bullet for Supply Chain Management? Technical Challenges and Research Opportunities,” *Decision Sciences*, vol. 51, no. 1, pp. 8–37, 2020.

[3] W. D. Du, S. L. Pan, D. E. Leidner, and W. Ying, “Affordances, Experimentation and Actualization of FinTech: A Blockchain Implementation Study,” *Journal of Strategic Information Systems*, vol. 28, no. 1, pp. 50–65, 2019.

[4] A. Tezel, P. Febrero, E. Papadonikolaki, and I. Yitmen, “Insights into Blockchain Implementation in Construction: Models for Supply Chain Management,” *Journal of Management in Engineering*, vol. 37, no. 4, pp. 1–19, 2021.

[5] S. Seebacher, R. Schüritz, and G. Satzger, “Towards an Understanding of Technology Fit and Appropriation in Business Networks: Evidence from Blockchain Implementations,” *Information Systems and e-Business Management*, vol. 19, no. 1, pp. 183–204, 2021.

[6] D. Galvin, “IBM and Walmart: Blockchain for Food Safety,” *IBMBlockchain*, p. 15, 2017.

[7] R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia, “A Survey on Blockchain Interoperability: Past, Present, and Future Trends,” *Association for Computing Machinery*, no. May, 2020.

[8] Gartner, “Gartner Hype Cycle 2019,” *Gartner.com*, 2019. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2019-10-08-gartner-2019-hype-cycle-shows-most-blockchain-technologies-are-still-five-to-10-years-away-from-transformational-impact. [Accessed: 29-Oct-2019].

[9] E. Abebe *et al.*, “Enabling Enterprise Blockchain Interoperability with Trusted Data Transfer (industry track),” *Middleware Industry 2019 - Proceedings of the 2019 20th International Middleware Conference Industrial Track, Part of Middleware 2019*, pp. 29–35, 2019.

[10] P. Zhang, J. White, D. C. Schmidt, and G. Lenz, “Applying Software Patterns to Address Interoperability in Blockchain-based Healthcare Apps,” no. 5 Jun, 2017.

[11] T. Hardjono, A. Lipton, and A. Pentland, “Toward an Interoperability Architecture for

Blockchain Autonomous Systems," *IEEE Transactions on Engineering Management*, pp. 1–12, 2019.

[12] T. Macheel, "Will Open Source Drive Blockchain Interoperability ?," *American Banker*, vol. 181, no. 227, pp. 1–3, 2016.

[13] P. Lafourcade and M. Lombard-Platet, "About Blockchain Interoperability," *Information Processing Letters*, vol. 161, p. 105976, 2020.

[14] E. J. Scheid, T. Hegnauer, B. Rodrigues, and B. Stiller, "Bifröst: a Modular Blockchain Interoperability API," *Proceedings - Conference on Local Computer Networks, LCN*, vol. 2019-Octob, pp. 332–339, 2019.

[15] W. Yang, E. Aghasian, S. Garg, D. Herbert, L. Disiuta, and B. Kang, "A Survey on Blockchain-Based Internet Service Architecture: Requirements, Challenges, Trends, and Future," *IEEE Access*, vol. 7, pp. 75845–75872, 2019.

[16] M. Leotta, F. Ricca, M. Ribaudo, G. Reggio, E. Astesiano, and T. Vernazza, "An Exploratory Survey on SOA Knowledge, Adoption and Trend in the Italian Industry," *Proceedings of IEEE International Symposium on Web Systems Evolution, WSE*, pp. 21–30, 2012.

[17] S. Gadge and V. Kotwani, "Microservice Architecture : API Gateway Considerations," *Globallogic.Com*, pp. 1–13, 2017.

[18] E. Djogic, S. Ribic, and D. Donko, "Monolithic to Microservices Redesign of Event Driven Integration Platform," *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2018 - Proceedings*, pp. 1411–1414, 2018.

[19] E. Morris, L. Levine, C. Meyers, P. Place, and D. Plakosh, "System of Systems Interoperability (SOSI): final report (No. CMU/SEI-2004-TR-004)," *Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.*, no. April, 2004.

[20] B. Pillai and B. Kamanashis, "Blockchain Interoperable Digital Objects," in *International Conference on Blockchain*.

[21] V. J. Morkunas, J. Paschen, and E. Boon, "How Blockchain Technologies Impact Your Business Model," *Business Horizons*, vol. 62, no. 3, pp. 295–306, 2019.

[22] N. Hewett, W. Lehmacher, and Y. Wang, "Inclusive Deployment of Blockchain for Supply Chains," *World Economic Forum*, no. March, p. 26, 2019.

[23] P. Wegner, "Interoperability," *ACM Computing Surveys*, vol. 28, no. 1, pp. 285–287, 1996.

[24] W. He and Xu, "Integration of Distributed Enterprise Applications: A Survey," *IEEE*

Transactions on Industrial Informatics, vol. 10, no. 1, pp. 35–42, 2012.

[25] F. B. Vernadat, "Interoperable Enterprise Systems: Architectures and Methods," *IFAC Proceedings Volumes (IFAC-PapersOnline),* vol. 12, no. Part 1, pp. 13–20, 2006.

[26] M. Health, "Interoperability roadmap:Accelerating the Shift to a Fully Interoperable Digital Health Ecosystem," 2020.

[27] D. Biel, M. Delone, W. Gerhardt, and C. Chang, "Radical interoperability Picking up Speed to Become a Reality for the Future of Health," no. Feature, pp. 1–20, 2019.

[28] W. Guédria, Y. Naudet, and D. Chen, "Maturity Model for Enterprise Interoperability," *Enterprise Information Systems*, vol. 9, no. 1, pp. 1–28, 2015.

[29] F. Lampathaki, S. Koussouris, C. Agostinho, R. Jardim-Goncalves, Y. Charalabidis, and J. Psarras, "Infusing scientific foundations into Enterprise Interoperability," *Computers in Industry,* vol. 63, no. 8, pp. 858–866, 2012.

[30] N. Basias, M. Themistocleous, and V. Morabito, "SOA Adoption in E-banking," *Journal of Enterprise Information Management*, vol. 26, no. 6, pp. 719–739, 2013.

[31] Q. Wu, S. Ying, Y. C. Ni, and H. Cui, "The architecture framework with exception handing in SOA," *Applied Mechanics and Materials*, vol. 20–23, pp. 992–997, 2010.

[32] I. Redbooks, *Patterns : Implementing an SOA using an Enterprise Service Bus*. IBM Redbooks, 2004.

[33] K. Thramboulidis, D. C. Vachtsevanou, and A. Solanos, "Cyber-Physical Microservices," *IEEE*, vol. 18, pp. 232–239, 2018.

[34] R. Chandramouli, Z. Butcher, and U. NIST, "Building Secure Microservices-based Applications Using Service-Mesh Architecture," *Nist*, no. NIST Special Publication 800-204A Building, pp. 1–19, 2020.

[35] X. Zuo, Y. Su, Q. Wang, and Y. Xie, "An API Gateway Design Strategy Optimized for Persistence and Coupling," *Advances in Engineering Software,* vol. 148, no. April, pp. 1–11, 2020.

[36] R. Mara Jösch, "Managing Microservices with a Service Mesh An Implementation of a Service Mesh with Kubernetes and Istio," *Degree Project Computer Science and Engineering*, 2020.

[37] W. Li, Y. Lemieux, J. Gao, Z. Zhao, and Y. Han, "Service Mesh: Challenges, State of the Art, and Future Research Opportunities," *Proceedings - 13th IEEE International Conference on Service-Oriented System Engineering, SOSE 2019,* pp. 122–127, 2019.

[38] B. Stopford, *Designing Event-Driven Systems.*, First. O'Reilly, 2018.

[39] K. Foundation, "Apache Kafka a Distributed Streaming Platform," *AK Platform,* 2021.

[Online]. Available: https://kafka.apache.org. [Accessed: 01-Feb-2021].

[40] Cloudurable, “Cloudurable Introduction to Kafka,” 2018.

[41] M. Matłoka, “Who and why uses Apache Kafka?,” *SoftwareMill Tech Blog*, 2020. [Online]. Available: https://blog.softwaremill.com/who-and-why-uses-apache-kafka-10fd8c781f4d. [Accessed: 21-Jul-2020].

[42] A. Reyna, C. Martín, J. Chen, E. Soler, and M. Díaz, “On Blockchain and its Integration with IoT. Challenges and Opportunities,” *Future Generation Computer Systems*, vol. 88, pp. 173–190, 2018.

[43] D. Yaga, P. Mell, N. Roby, and K. Scarfone, “Blockchain Technology Overview,” *National Institute of Standards and Technology US Department of Commerce*, 2019.

[44] J. Yli-huumo, D. Ko, S. Choi, S. Park, and K. Smolander, “Where Is Current Research on Blockchain Technology ?— A Systematic Review,” *PLOS One*, pp. 1–27, 2016.

[45] F. Tian, “An Information System for Food Safety Monitoring in Supply Chains based on HACCP, Blockchain and Internet of Things,” *Doctoral Dissertation ePubWU Institutional Repository*, vol. 1, no. March, pp. 1–164, 2018.

[46] L. Lamport, R. Shostak, and M. Pease, “The Byzantine Generals Problem,” *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382–401, 1982.

[47] E. Ak and B. Canberk, “BCDN: A Proof of Concept Model for Blockchain-aided CDN Orchestration and Routing,” *Computer Networks*, vol. 161, no. June 2018, pp. 162–171, 2019.

[48] H. Jin, X. Dai, and J. Xiao, “Towards a Novel Architecture for Enabling Interoperability Amongst Multiple Blockchains,” *Proceedings - International Conference on Distributed Computing Systems*, vol. 2018-July, pp. 1203–1211, 2018.

[49] J. Kwon and E. Buchman, “A Network of Distributed Ledgers.”

[50] L. Foundation, “Hyperledger Architecture, Volume 1,” 2017.

[51] L. Foundation, “Hyperledger Architecture, Volume II,” 2018.

[52] J. Brogan, I. Baskaran, and N. Ramachandran, “Authenticating Health Activity Data Using Distributed Ledger Technologies,” *Computational and Structural Biotechnology Journal*, vol. 16, pp. 257–266, 2018.

[53] C. Gorenflo, S. Lee, L. Golab, and S. Keshav, “FastFabric: Scaling Hyperledger Fabric to 20,000 Transactions per Second,” *ICBC 2019 - IEEE International Conference on Blockchain and Cryptocurrency*, pp. 455–463, 2019.

[54] S. Pearson *et al.*, “Are Distributed Ledger Technologies the panacea for food

traceability?," *Global Food Security*, vol. 20, no. November 2018, pp. 145–149, 2019.

[55] A. Banerjee, "Integrating Blockchain with ERP for a Transparent Supply Chain," *Infosys*, no. Perspective, pp. 1–8, 2017.

[56] P. Schneider, *Managerial challenges of Industry 4.0: an empirically backed research agenda for a nascent field*, vol. 12, no. 3. Springer Berlin Heidelberg, 2018.

[57] M. Andoni *et al.*, "Blockchain Technology in the Energy Sector: A Systematic Review of Challenges and Opportunities," *Renewable and Sustainable Energy Reviews*, vol. 100, no. October 2018, pp. 143–174, 2019.

[58] A. Reyna, C. Martín, J. Chen, E. Soler, and M. Díaz, "On Blockchain and its Integration with IoT. Challenges and Opportunities," *Future Generation Computer Systems*, vol. 88, pp. 173–190, 2018.

[59] A. Patki and V. Sople, "Indian Banking Sector: Blockchain Implementation, Challenges and Way Forward," *Journal of Banking and Financial Technology*, vol. 4, no. 1, pp. 65–73, 2020.

[60] J. Lin, Z. Shen, A. Zhang, and Y. Chai, "Blockchain and IoT based Food Traceability for Smart Agriculture," *International Conference on Crowd Science and Engineering, Singapore*, p. 6, 2018.

[61] Á. Rocha, A. M. Correia, H. Adeli, L. P. Reis, and M. M. Teixeira, "New Advances in Information Systems and Technologies," *Advances in Intelligent Systems and Computing*, vol. 445, pp. 275–286, 2016.

[62] M. P. Caro, M. S. Ali, M. Vecchio, and R. Giaffreda, "Blockchain-Based Traceability in Agri-Food Supply Chain Management: A Practical Implementation," *2018 IoT Vertical and Topical Summit on Agriculture - Tuscany, IOT Tuscany 2018*, pp. 1–4, 2018.

[63] H. Anwar, "Distributed Ledger Technology: Where Technological Revolution Starts," *101Blockchains.Com,* 2019. [Online]. Available: https://101blockchains.com/distributed-ledger-technology-dlt/#3. [Accessed: 20-Apr-2020].

[64] E. Chapkanovska, "Blockchain Adoption [The LATEST Statistics 2021]," *https://spendmenot.com/,* 2021. [Online]. Available: https://spendmenot.com/blog/blockchain-adoption/. [Accessed: 25-May-2021].

[65] C. Lima, "Developing Open and Interoperable DLT/Blockchain Standards [Standards]," *Computer*, vol. 51, no. 11, pp. 106–111, 2018.

[66] O. Rodríguez-Espíndola, S. Chowdhury, A. Beltagui, and P. Albores, "The Potential of

Emergent Disruptive Technologies for Humanitarian Supply Chains: the Integration of Blockchain, Artificial Intelligence and 3D Printing," *International Journal of Production Research*, vol. 58, no. 15, pp. 4610–4630, 2020.

[67] V. Lopes and L. A. Alexandre, "An Overview of Blockchain Integration with Robotics and Artificial Intelligence," *Ledger Journal*, vol. 5980, 2018.

[68] B. Chavali, S. K. Khatri, and S. A. Hossain, "AI and Blockchain Integration," *ICRITO 2020 - IEEE 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)*, pp. 548–552, 2020.

[69] A. Panarello, N. Tapas, G. Merlino, F. Longo, and A. Puliafito, *Blockchain and Iot Integration: A systematic Survey*, vol. 18, no. 8. 2018.

[70] K. C. Chan, X. Zhou, R. Gururajan, X. Zhou, M. Ally, and M. Gardiner, "Integration of Blockchains with Management Information Systems," *Proceedings of the 2019 International Conference on Mechatronics, Robotics and Systems Engineering, MoRSE 2019*, no. December, pp. 157–162, 2019.

[71] A. J. Yewale, "Study of Blockchain-as-a-Service Systems with a Case Study of Hyperledger Fabric Implementation on Kubernetes," 2018.

[72] E. W. K. Tsang, "Realist Perspective," *Academy of Management Review*, vol. 24, no. 4, pp. 759–781, 1999.

[73] Y. Svetashova, S. Schmid, and Y. Sure-Vetter, "New Facets of Semantic Interoperability: Adding JSON - JSON-LD Transformation Functionality to the BIG IoT API," *CEUR Workshop Proceedings*, vol. 1963, pp. 1–4, 2017.

[74] Cheryl Tatano Beck, "Replication Strategies for Nursing Research," *Journal of Nursing Scholarship*, vol. 26, no. 3, pp. 191–194, 1994.

[75] S. W. Brown and K. A. Coney, "Building a Replication Tradition in Marketing.," *American Marketing Association*, pp. 1776–1976, 1976.

[76] Scott M. Fuess, "On Replication in Business and Economics Research: The QJBE Case," *The Marketing Concept in Perspective*, vol. 13, no. 22. pp. 5416–5421, 1988.

[77] H. H. Hyman and C. R. Wright, "Evaluating Social Action Programs. The Uses of Sociology," *NY Basic*, pp. 769–777, 1967.

[78] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Integration of Blockchain and Cloud of Things: Architecture, Applications and Challenges," *IEEE Communications Surveys and Tutorials*, pp. 1–36, 2019.

[79] L. Besancon, C. F. Da Silva, and P. Ghodous, "Towards Blockchain Interoperability: Improving Video Games Data Exchange," *ICBC 2019 - IEEE International Conference*

on Blockchain and Cryptocurrency, pp. 81–85, 2019.

[80] A. R. Hevner, S. T. March, J. Park, and S. Ram, “Design Science in Information Systems Research,” *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004.

[81] R. Winter, “Design Science Research in Europe,” *European Journal of Information Systems*, vol. 17, no. 5. Palgrave Macmillan Ltd., pp. 470–475, 2008.

[82] H. Simon, *The Sciences of the Artificial*, Third. Cambridge, MA, USA, 1996.

[83] S. T. March and G. F. Smith, “Design and Natural Science Research on Information Technology,” *Decision Support Systems*, vol. 15, no. 4, pp. 251–266, 1995.

[84] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, “A Design Science Research Methodology for Information Systems Research,” *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007.

[85] G. Muller, “Accelerating a Connected New Zealand,” p. 96, 2018.

[86] R. Vidgen, B. Donnellan, S. Matook, and K. Conboy, “Practical Aspects of Design Science,” *Communications in Computer and Information Science*, vol. 286, no. March, pp. 171–177, 2012.

[87] S. Gregor and D. Jones, “The Anatomy of a Design Theory,” *Journal of the Association for Information Systems*, vol. 8, no. 5, pp. 312–335, 2007.

[88] A. Cleven, P. Gubler, and K. M. Hüner, “Design Alternatives for the Evaluation of Design Science Research Artifacts,” *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology, DESRIST ’09*, 2009.

[89] N. Prat, I. Comyn-Wattiau, and J. Akoka, “Artifact Evaluation in Information Systems Design-Science Research - A Holistic View,” *Association for Information Systems AIS Electronic Library (AISeL)*, p. 2014, 2014.

[90] G. F. S. Salvatore T. March, “Design and Natural Science Research on Information Technology,” *Decision Support Systems*, vol. 10, pp. 501–509, 1964.

[91] S. Gregor and A. R. Hevner, “Positioning and Presenting Design Science Research for Maximum Impact,” *MIS Quarterly*, vol. 37, no. 2, pp. 337–355, 2013.

[92] Hevner, March, Park, and Ram, “Design Science in Information Systems Research,” *MIS Quarterly*, vol. 28, no. 1, p. 75, 2004.

[93] R. Baskerville, A. Baiyere, S. Gergor, A. Hevner, and M. Rossi, “Design Science Research Contributions: Finding a Balance between Artifact and Theory,” *Journal of the Association for Information Systems*, vol. 19, no. 5, pp. 358–376, 2018.

[94] V. K. Vaishnavi, V. K. Vaishnavi, and W. Kuechler, *Design Science Research Methods*

and Patterns. 2015.

[95] G. S. S. Leal, W. Guédria, and H. Panetto, “Enterprise Interoperability Assessment: A Requirements Engineering Approach,” *International Journal of Computer Integrated Manufacturing,* vol. 33, no. 3, pp. 265–286, 2020.

[96] A. Drechsler and A. R. Hevner, “Utilizing, Producing, and Contributing Design Knowledge in DSR Projects,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics),* vol. 10844 LNCS, pp. 82–97, 2018.

[97] H. Avdiji and R. Winter, “Knowledge gaps in design science research,” *40th International Conference on Information Systems, ICIS 2019,* pp. 1–17, 1984.

[98] R. Rezaei, T. K. Chiew, and S. P. Lee, “An Interoperability Model for Ultra Large Scale Systems,” *Advances in Engineering Software,* vol. 67, pp. 22–46, 2014.

[99] G. Feuerlicht and S. Govardhan, “SOA: Trends and Directions,” *System Integration,* pp. 149–154, 2009.

[100] Z. Li, A. V. Barenji, and G. Q. Huang, “Toward a blockchain cloud manufacturing system as a peer to peer distributed network platform,” *Robotics and Computer-Integrated Manufacturing,* vol. 54, no. July, pp. 133–144, 2018.

[101] G. Büyüközkan and F. Göçer, “Digital Supply Chain: Literature review and a proposed framework for future research,” *Computers in Industry,* vol. 97, pp. 157–177, 2018.

[102] JS Doc, “Hyperledger Fabric SDK for node.js,” *Hyperledger Github,* 2021. [Online]. Available: https://hyperledger.github.io/fabric-sdk-node/release-1.4/index.html. [Accessed: 05-Dec-2021].

[103] S. Chen, H. Wang, and L.-J. Zhang, “Blockchain – ICBC 2018,” *Proceedings Blockchain-ICBC,* vol. 10974, no. January 2019, pp. 3–17, 2018.

[104] G. Gebresenbet and T. Boso, “Logistics and Supply Chains in Agriculture and Food,” *Pathways to Supply Chain Excellence,* pp. 125–148, 2012.

[105] J. F. Galvez, J. C. Mejuto, and J. Simal-Gandara, “Future Challenges on the Use of Blockchain for Food Traceability Analysis,” *TrAC - Trends in Analytical Chemistry,* vol. 107, pp. 222–232, 2018.

[106] F. Holotiuk, F. Pisani, and J. Moormann, “Unveiling the Key Challenges to Achieve the Breakthrough of Blockchain: Insights from the Payments Industry,” *Proceedings of the 51st Hawaii International Conference on System Sciences,* pp. 3537–3546, 2018.

[107] D. Pigini and M. Conti, “NFC-based Traceability in the Food Chain,” *Sustainability (United States),* vol. 9, no. 6, pp. 1–20, 2017.

[108] R. Y. Chen, “Autonomous Tracing System for Backward Design in Food Supply Chain,” *Food Control*, vol. 51, pp. 70–84, 2015.

[109] F. Tian, “An Agri-Food Supply Chain Traceability System for China Based on RFID & Blockchain Technology,” *2016 13th International Conference on Service Systems and Service Management, ICSSSM 2016*, pp. 1–6, 2016.

[110] J. Storoy, M. Thakur, and P. Olsen, “The TraceFood Framework - Principles and Guidelines for Implementing Traceability in Food Value Chains,” *Journal of Food Engineering*, vol. 115, no. 1, pp. 41–48, 2013.

[111] K. Biswas, V. Muthukkumarasamy, and W. L. Tan, “Blockchain based Wine Supply Chain Traceability System,” *Future Technologies Conference*, no. November, pp. 56–62, 2017.

[112] A. Bahga and V. K. Madisetti, “Blockchain Platform for Industrial Internet of Things,” *Journal of Software Engg and Applications*, pp. 533–546, 2016.

[113] P. Olsen and M. Borit, “The Components of a Food Traceability System,” *Trends in Food Science and Technology*, vol. 77, no. June 2017, pp. 143–149, 2018.

[114] Q. Lin, H. Wang, X. Pei, and J. Wang, “Food Safety Traceability System Based on Blockchain and EPCIS,” *IEEE Access*, vol. 7, pp. 20698–20707, 2019.

[115] F. Kamoun, O. Alfandi, and S. Miniaoui, “An RFID Solution for the Monitoring of Storage Time and Localization of Perishable Food in a Distribution Center,” *GSCIT 2015 - Global Summit on Computer and Information Technology - Proceedings*, pp. 1–6, 2015.

[116] M. Kim, B. Hilton, Z. Burks, and J. Reyes, “IoT to Design a Food Traceability Solution,” *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, no. Figure 1, pp. 335–340, 2018.

[117] H. Treiblmaier, “The Impact of the Blockchain on the Supply Chain: A Theory-Based Research Framework and a Call for Action,” *Supply Chain Management*, vol. 23, no. 6, pp. 545–559, 2018.

[118] L. Ge, C. Brewster, J. Spek, A. Smeenk, and J. Top, *Blockchain for Agriculture and Food*. Wageningen University & Research, 2017.

[119] D. Mao, Z. Hao, F. Wang, and H. Li, “Innovative Blockchain-based Approach for Sustainable and Credible Environment in Food Trade: A case Study in Shandong Province, China,” *Sustainability (Switzerland)*, vol. 10, no. 9, 2018.

[120] N. Kshetri, “Blockchain’s Roles in Meeting key Supply Chain Management Objectives,” *International Journal of Information Management*, vol. 39, no. June 2017,

pp. 80–89, 2018.

[121] M. J. Casey and P. Wong, "Global Supply Chains are about to get Better, Thanks to Blockchain," *Harward Business Review Press,* pp. 85–92, 2019.

[122] Y. Wang, M. Singgih, J. Wang, and M. Rit, "Making Sense of Blockchain Technology: How will it Transform Supply Chains?," *International Journal of Production Economics,* vol. 211, no. February, pp. 221–236, 2019.

[123] H. Fabric, "Hyperledger Fabric," *Hyperledger*. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-1.4/whatis.html. [Accessed: 19-Apr-2020].

[124] M. Petersen, N. Hackius, and B. Von See, "Mapping the Sea of Opportunities : Blockchain in Supply Chain and Logistics," *Research Gate - Information Technology,* p. 9, 2018.

[125] X. Xu, I. Weber, M. Staples, X. Xu, I. Weber, and M. Staples, *Existing Blockchain Platforms*. 2019.

[126] H. Fabric, "Hardware Security Module (HSM)," *Hyperledger Foundation*, 2020. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2/hsm.html.

[127] K. R. Larsen, R. Lukyanenko, R. M. Mueller, and V. C. Storey, "Validity in Design Science Research," *International Conference on Design Science Research in Information Systems and Technology*, no. June, 2020.

[128] J. Venable, J. Pries-Heje, and R. Baskerville, "FEDS: A Framework for Evaluation in Design Science Research," *European Journal of Information Systems*, vol. 25, no. 1, pp. 77–89, 2016.

[129] A. Alturki, G. G. Gable, and W. Bandara, "The Design Science Research Roadmap: In Progress Evaluation," *Proceedings - Pacific Asia Conference on Information Systems, PACIS 2013*, 2013.

[130] J. Pries-Heje, R. Baskerville, and J. Venable, "Strategies for Design Science Research Evaluation," *16th European Conference on Information Systems, ECIS 2008*, 2008.

[131] R. Baskerville, A. Baiyere, S. Gregor, A. Hevner, and M. Rossi, "Design science research contributions: Finding a balance between artifact and theory," *Journal of the Association of Information Systems*, vol. 19, no. 5, pp. 358–376, 2018.

[132] G. da Silva Serapião Leal, W. Guédria, and H. Panetto, "Interoperability Assessment: A Systematic Literature Review," *Computers in Industry*, vol. 106, pp. 111–132, 2019.

[133] M. Herselman and A. Botha, "Evaluating an Artifact in Design Science Research," *ACM International Conference Proceeding Series*, p. 10, 2015.

[134] F. Casino, T. K. Dasaklis, and C. Patsakis, "A Systematic Literature Review of

Blockchain-based Applications: Current Status, Classification and Open Issues," *Telematics and Informatics*, vol. 36, no. November 2018, pp. 55–81, 2019.

[135] O. Source, "Most popular software tech stack in 2020-2021.," *Quora*, 2020. [Online]. Available: https://www.quora.com/What-is-the-most-popular-software-tech-stack-in-2020-2021.

[136] WebFX, "A Guide to Technology Stacks (Infographic)," 2020. [Online]. Available: https://www.webfx.com/blog/web-design/technology-stacks-infographic/. [Accessed: 19-Apr-2021].

[137] Stackshare, "Track and Collaborate on Tech Stack Decisions," *Stackshare*, 2020. [Online]. Available: https://stackshare.io/. [Accessed: 07-Jan-2021].

[138] H. Shah, "Tech Stack of 5 Popular Billion Dollar Companies," *SIMFORM*, 2020. [Online]. Available: https://www.simform.com/tech-stack-billion-dollar-companies/. [Accessed: 07-Jan-2021].

[139] M. Kuzlu, M. Pipattanasomporn, L. Gurses, and S. Rahman, "Performance Analysis of a Hyperledger Fabric Blockchain Framework: Throughput, Latency and Scalability," *Proceedings - 2019 2nd IEEE International Conference on Blockchain, Blockchain 2019*. pp. 536–540, 2019.

[140] K. Kuhi, K. Kaare, and O. Koppel, "Ensuring Performance Measurement Integrity in Logistics using Blockchain," *Proceedings of the 2018 IEEE International Conference on Service Operations and Logistics, and Informatics, SOLI 2018*, pp. 256–261, 2018.

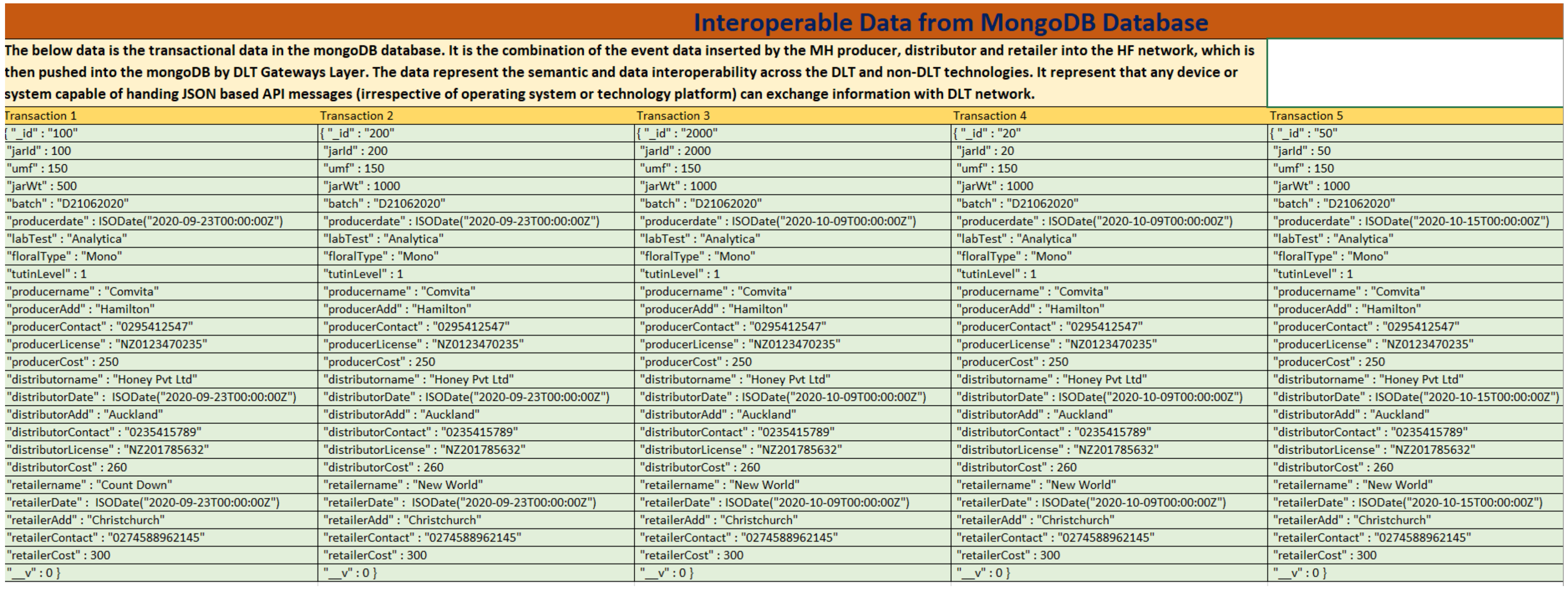
[141] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends," *Proceedings - 2017 IEEE 6th International Congress on Big Data, BigData Congress 2017*, pp. 557–564, 2017.

Appendix A

The research has developed a Hyperledger Fabric DLT Solution Prototype (HDSP), explained in detail in Chapter 5. The source code of the solution, both front-end (HDSP_UI) and back-end (HDSP_Hyperledger_Code), is available on GitHub at the below mentioned link:

- HDSP_Prototype https://github.com/hushare1/PhD-Prototype-HDSP

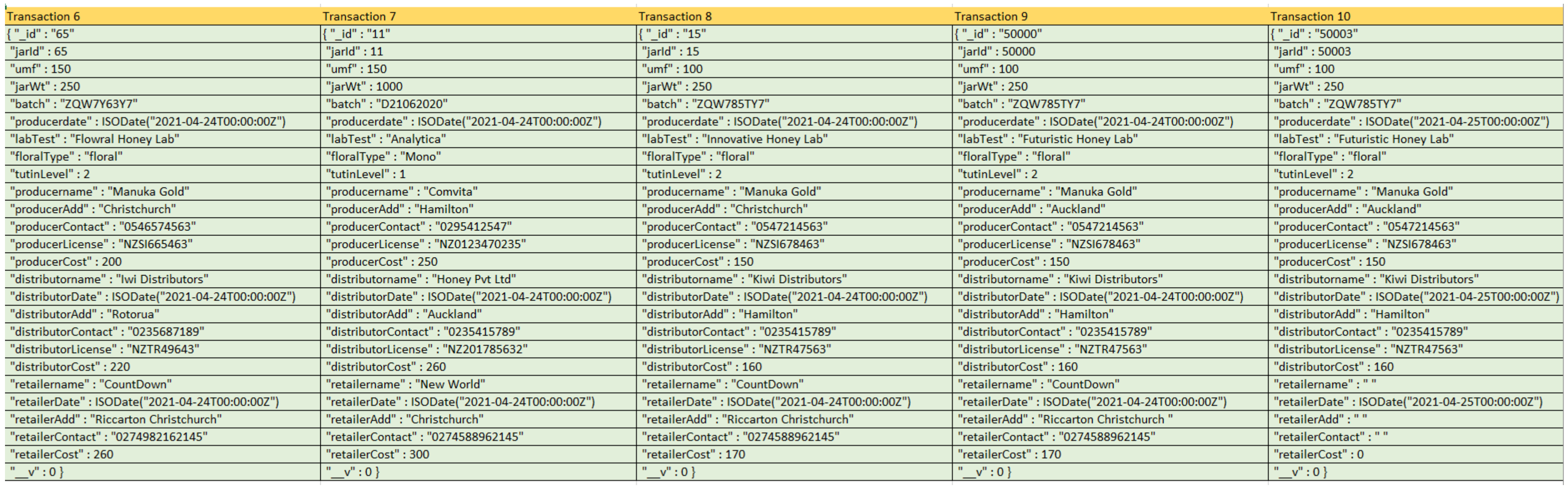
Appendix B



Interoperable Data from MongoDB Database

The below data is the transactional data in the mongoDB database. It is the combination of the event data inserted by the MH producer, distributor and retailer into the HF network, which is then pushed into the mongoDB by DLT Gateways Layer. The data represent the semantic and data interoperability across the DLT and non-DLT technologies. It represent that any device or system capable of handing JSON based API messages (irrespective of operating system or technology platform) can exchange information with DLT network.

Transaction 1	Transaction 2	Transaction 3	Transaction 4	Transaction 5
{ "_id" : "100"	{ "_id" : "200"	{ "_id" : "2000"	{ "_id" : "20"	{ "_id" : "50"
"jarId" : 100	"jarId" : 200	"jarId" : 2000	"jarId" : 20	"jarId" : 50
"umf" : 150	"umf" : 150	"umf" : 150	"umf" : 150	"umf" : 150
"jarWt" : 500	"jarWt" : 1000	"jarWt" : 1000	"jarWt" : 1000	"jarWt" : 1000
"batch" : "D21062020"	"batch" : "D21062020"	"batch" : "D21062020"	"batch" : "D21062020"	"batch" : "D21062020"
"producerdate" : ISODate("2020-09-23T00:00:00Z")	"producerdate" : ISODate("2020-09-23T00:00:00Z")	"producerdate" : ISODate("2020-10-09T00:00:00Z")	"producerdate" : ISODate("2020-10-09T00:00:00Z")	"producerdate" : ISODate("2020-10-15T00:00:00Z")
"labTest" : "Analytica"	"labTest" : "Analytica"	"labTest" : "Analytica"	"labTest" : "Analytica"	"labTest" : "Analytica"
"floralType" : "Mono"	"floralType" : "Mono"	"floralType" : "Mono"	"floralType" : "Mono"	"floralType" : "Mono"
"tutinLevel" : 1	"tutinLevel" : 1	"tutinLevel" : 1	"tutinLevel" : 1	"tutinLevel" : 1
"producername" : "Comvita"	"producername" : "Comvita"	"producername" : "Comvita"	"producername" : "Comvita"	"producername" : "Comvita"
"producerAdd" : "Hamilton"	"producerAdd" : "Hamilton"	"producerAdd" : "Hamilton"	"producerAdd" : "Hamilton"	"producerAdd" : "Hamilton"
"producerContact" : "0295412547"	"producerContact" : "0295412547"	"producerContact" : "0295412547"	"producerContact" : "0295412547"	"producerContact" : "0295412547"
"producerLicense" : "NZ0123470235"	"producerLicense" : "NZ0123470235"	"producerLicense" : "NZ0123470235"	"producerLicense" : "NZ0123470235"	"producerLicense" : "NZ0123470235"
"producerCost" : 250	"producerCost" : 250	"producerCost" : 250	"producerCost" : 250	"producerCost" : 250
"distributorname" : "Honey Pvt Ltd"	"distributorname" : "Honey Pvt Ltd"	"distributorname" : "Honey Pvt Ltd"	"distributorname" : "Honey Pvt Ltd"	"distributorname" : "Honey Pvt Ltd"
"distributorDate" : ISODate("2020-09-23T00:00:00Z")	"distributorDate" : ISODate("2020-09-23T00:00:00Z")	"distributorDate" : ISODate("2020-10-09T00:00:00Z")	"distributorDate" : ISODate("2020-10-09T00:00:00Z")	"distributorDate" : ISODate("2020-10-15T00:00:00Z")
"distributorAdd" : "Auckland"	"distributorAdd" : "Auckland"	"distributorAdd" : "Auckland"	"distributorAdd" : "Auckland"	"distributorAdd" : "Auckland"
"distributorContact" : "0235415789"	"distributorContact" : "0235415789"	"distributorContact" : "0235415789"	"distributorContact" : "0235415789"	"distributorContact" : "0235415789"
"distributorLicense" : "NZ201785632"	"distributorLicense" : "NZ201785632"	"distributorLicense" : "NZ201785632"	"distributorLicense" : "NZ201785632"	"distributorLicense" : "NZ201785632"
"distributorCost" : 260	"distributorCost" : 260	"distributorCost" : 260	"distributorCost" : 260	"distributorCost" : 260
"retailername" : "Count Down"	"retailername" : "New World"	"retailername" : "New World"	"retailername" : "New World"	"retailername" : "New World"
"retailerDate" : ISODate("2020-09-23T00:00:00Z")	"retailerDate" : ISODate("2020-09-23T00:00:00Z")	"retailerDate" : ISODate("2020-10-09T00:00:00Z")	"retailerDate" : ISODate("2020-10-09T00:00:00Z")	"retailerDate" : ISODate("2020-10-15T00:00:00Z")
"retailerAdd" : "Christchurch"	"retailerAdd" : "Christchurch"	"retailerAdd" : "Christchurch"	"retailerAdd" : "Christchurch"	"retailerAdd" : "Christchurch"
"retailerContact" : "0274588962145"	"retailerContact" : "0274588962145"	"retailerContact" : "0274588962145"	"retailerContact" : "0274588962145"	"retailerContact" : "0274588962145"
"retailerCost" : 300	"retailerCost" : 300	"retailerCost" : 300	"retailerCost" : 300	"retailerCost" : 300
"__v" : 0 }	"__v" : 0 }	"__v" : 0 }	"__v" : 0 }	"__v" : 0 }



Transaction 6	Transaction 7	Transaction 8	Transaction 9	Transaction 10
{ "_id" : "65"	{ "_id" : "11"	{ "_id" : "15"	{ "_id" : "50000"	{ "_id" : "50003"
"jarId" : 65	"jarId" : 11	"jarId" : 15	"jarId" : 50000	"jarId" : 50003
"umf" : 150	"umf" : 150	"umf" : 100	"umf" : 100	"umf" : 100
"jarWt" : 250	"jarWt" : 1000	"jarWt" : 250	"jarWt" : 250	"jarWt" : 250
"batch" : "ZQW7Y63Y7"	"batch" : "D21062020"	"batch" : "ZQW785TY7"	"batch" : "ZQW785TY7"	"batch" : "ZQW785TY7"
"producerdate" : ISODate("2021-04-24T00:00:00Z")	"producerdate" : ISODate("2021-04-24T00:00:00Z")	"producerdate" : ISODate("2021-04-24T00:00:00Z")	"producerdate" : ISODate("2021-04-24T00:00:00Z")	"producerdate" : ISODate("2021-04-25T00:00:00Z")
"labTest" : "Flowral Honey Lab"	"labTest" : "Analytica"	"labTest" : "Innovative Honey Lab"	"labTest" : "Futuristic Honey Lab"	"labTest" : "Futuristic Honey Lab"
"floralType" : "floral"	"floralType" : "Mono"	"floralType" : "floral"	"floralType" : "floral"	"floralType" : "floral"
"tutinLevel" : 2	"tutinLevel" : 1	"tutinLevel" : 2	"tutinLevel" : 2	"tutinLevel" : 2
"producername" : "Manuka Gold"	"producername" : "Comvita"	"producername" : "Manuka Gold"	"producername" : "Manuka Gold"	"producername" : "Manuka Gold"
"producerAdd" : "Christchurch"	"producerAdd" : "Hamilton"	"producerAdd" : "Christchurch"	"producerAdd" : "Auckland"	"producerAdd" : "Auckland"
"producerContact" : "0546574563"	"producerContact" : "0295412547"	"producerContact" : "0547214563"	"producerContact" : "0547214563"	"producerContact" : "0547214563"
"producerLicense" : "NZSI665463"	"producerLicense" : "NZ0123470235"	"producerLicense" : "NZSI678463"	"producerLicense" : "NZSI678463"	"producerLicense" : "NZSI678463"
"producerCost" : 200	"producerCost" : 250	"producerCost" : 150	"producerCost" : 150	"producerCost" : 150
"distributorname" : "Iwi Distributors"	"distributorname" : "Honey Pvt Ltd"	"distributorname" : "Kiwi Distributors"	"distributorname" : "Kiwi Distributors"	"distributorname" : "Kiwi Distributors"
"distributorDate" : ISODate("2021-04-24T00:00:00Z")	"distributorDate" : ISODate("2021-04-24T00:00:00Z")	"distributorDate" : ISODate("2021-04-24T00:00:00Z")	"distributorDate" : ISODate("2021-04-24T00:00:00Z")	"distributorDate" : ISODate("2021-04-25T00:00:00Z")
"distributorAdd" : "Rotorua"	"distributorAdd" : "Auckland"	"distributorAdd" : "Hamilton"	"distributorAdd" : "Hamilton"	"distributorAdd" : "Hamilton"
"distributorContact" : "0235687189"	"distributorContact" : "0235415789"	"distributorContact" : "0235415789"	"distributorContact" : "0235415789"	"distributorContact" : "0235415789"
"distributorLicense" : "NZTR49643"	"distributorLicense" : "NZ201785632"	"distributorLicense" : "NZTR47563"	"distributorLicense" : "NZTR47563"	"distributorLicense" : "NZTR47563"
"distributorCost" : 220	"distributorCost" : 260	"distributorCost" : 160	"distributorCost" : 160	"distributorCost" : 160
"retailername" : "CountDown"	"retailername" : "New World"	"retailername" : "CountDown"	"retailername" : "CountDown"	"retailername" : " "
"retailerDate" : ISODate("2021-04-24T00:00:00Z")	"retailerDate" : ISODate("2021-04-24T00:00:00Z")	"retailerDate" : ISODate("2021-04-24T00:00:00Z")	"retailerDate" : ISODate("2021-04-24T00:00:00Z")	"retailerDate" : ISODate("2021-04-25T00:00:00Z")
"retailerAdd" : "Riccarton Christchurch"	"retailerAdd" : "Christchurch"	"retailerAdd" : "Riccarton Christchurch"	"retailerAdd" : "Riccarton Christchurch "	"retailerAdd" : " "
"retailerContact" : "0274982162145"	"retailerContact" : "0274588962145"	"retailerContact" : "0274588962145"	"retailerContact" : "0274588962145"	"retailerContact" : " "
"retailerCost" : 260	"retailerCost" : 300	"retailerCost" : 170	"retailerCost" : 170	"retailerCost" : 0
"__v" : 0 }	"__v" : 0 }	"__v" : 0 }	"__v" : 0 }	"__v" : 0 }