A DevOps Reference Architecture for Multi-Cloud IoT Applications Deployment

Georges Bou Ghantous

Doctor of Philosophy (C02029)

Software Engineering

University of Technology Sydney

Faculty of Engineering and Information Technology

School of Computer Science

Certificate of Authorship

I, Georges Bou Ghantous, declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy in Software Engineering in the Faculty of Engineering and IT, School of Computer Science at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis. This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Signature: Production Note:
Signature removed prior to publication.

Date: 22/04/2020

Acknowledgments

I have been privileged to have Dr Asif Gill and Dr Farookh Hussain as my supervisors. I want to express my sincere gratitude to my supervisor Dr Asif Gill for providing me with an opportunity to work with him on this research project. His highly valuable support, coaching, encouragement, quick feedback, and guidance helped me to finish this research project. Dr Asif Gill was always there to provide support in meetings, phone conferences, and emails. His valuable understanding in the field of agile and software engineering as an academic, researcher, and practitioner provided a wealth of knowledge that I used in this research project.

This study would not have been possible without the support of my family. I am very grateful to my parents, Salim Bou Ghantous and Laure Mikhail, for their precious love, blessings, support, encouragement, sacrifice, and care since my childhood. I am thankful for the support of my siblings. Their genuine care gave me confidence and peace of mind during my busy life. I am blessed to be a member of such a loving, caring, and supportive family.

I want to express my sincere thanks to all of my colleagues and friends for their valuable feedback and support—especially my dearest friends, Charbel Lahoud and John Mikhael (late).

I also wish to thank the Australian Government for its support and HECS-HELP for providing the funding for my research project for the length of my PhD study period.

I am thankful to all of the reviewers for valuable feedback and comments.

Thank you all.

Research Contributions and Publications

During this PhD research project, I collaborated with my supervisor and other colleagues. I published the components of this research work (DRA) in several rigorously reviewed international conference papers and scientific journals. The papers' publications were an opportunity to present my work for review before including it in this thesis. Appendix F presents a list of the publications that have been included in this thesis.

Contents

List of Tables

List of Figures

List of Equations

List of Abbreviations

AAF	Average and Above Frequency
AAP	Average and Above Percentage
API	Application Programming Interface
App	Software Application
CD	Continuous Deployment
CI	Continuous Integration
DB	Database
DRA	DevOps Reference Architecture
DSR	Design Science Research
IaaS	Infrastructure as a Service
IoT	Internet of Things
MQTT	Message Queuing Telemetry Transport
PaaS	Platform as a Service
PC	DevOps Practice
POC	Proof of Concept
QEM	Qualitative Evaluator Matrix
QIM	Quantitative Indicator Matrix
QI	Quantitative Indicator
RDF	Resource Description Framework
Retro-QA	Retrospective Quality Assurance
RFID	Radio Frequency Identification
RPIB	Raspberry Pi Model 3 B
RQ	Research Question
SaaS	Software as a Service
SLR	Systematic Literature Review
SSH	Secure Shell
UTS	University of Technology Sydney
WAN	Wide Area Network
WSN	Wide Sensor Network
GPIO	General-purpose input/output
FEIT	Faculty of Engineering and Information Technology

Glossary

Abstraction	Abstraction refers to a logical view of entities such as objects, elements and services.
Agile	Agile is an iterative software development methodology that solves the complexity of the software project by adopting an iterative approach to 'revisit' the development process (analysis, planning, architecture, design, develop, test and deploy) and hasten the release or delivery of products to customers.
Continuous Integration (CI) Broker	A CI broker is a DevOps cloud-based tool that hosts the deployment configuration for the software applications (e.g. IoT-applications) deployment to multi-cloud in a DRA architectural model.
DevOps	DevOps is a set of software development practices that combine software development (Dev) and information technology operations (Ops) to improve agile software development.
DevOps Pipeline	A pipeline is a set of integrated DevOps tools that enable an automated software deployment process.
Framework	A framework is a set of development elements and components that are combined to produce a tailored process/method.
Internet of Things (IoT)	IoT refers to a network of physical objects that feature an IP address for internet connectivity, and the communication that occurs between these objects and other internet-enabled devices and systems.
JavaScript Object Notation (JSON)	JSON is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute-value pairs and array data types (or any other serializable value).
Multi-Cloud	Multi-cloud is the integration of several cloud computing services in a single heterogeneous architecture.
NoSQL Database (DB)	A No SQL DB stores data in a document format such as JavaScript Object Notation.
Process/Method	'Process' and 'method' are used interchangeably in this thesis (although it is acknowledged that these terms are used differently in other areas of software engineering).
Reference Architecture	Reference architecture in the field of software architecture or enterprise architecture provides a template solution for architecture for a particular domain.
Vendor Lock-In	Vendor lock-in is a situation in which a customer using a cloud product or service cannot easily transition to a competitor's cloud product or service. Vendor lock-in may occur in several cases—for instance, when a cloud in the multi-cloud system hosts the deployment configurations of the software application and when a cloud in the multi-cloud system hosts the database for the application data.
Waterfall	Waterfall methodology aligns the software development process (analysis, planning, architecture, design, develop, test, and deploy) in sequential order.

Abstract

DevOps originated in the context of agile software development, which seems an appropriate approach to enable the continuous delivery and deployment of a software application in small releases. There is growing interest among organisations in adopting the DevOps approach and a multi-cloud environment for IoT (Internet of Things) application deployment. However, the challenge is how to apply DevOps when a multi-cloud heterogeneous environment is required for IoT application deployment. To address this vital research need, this thesis applies a design science research (DSR) method. It develops the DevOps reference architecture (DRA) framework to automate IoT applications deployment to the heterogeneous multi-cloud environment. The DRA is a cloud-enabled framework that mainly focuses on the deployment part of the integrated agile–DevOps methodology. Using a DSR method, the DRA has been incrementally developed by the iterative application of build, review, and adjust research activities. The DRA is intended for use by software organisations, coaches, managers, engineers, developers, and consultants as comprehensive reference architecture for deploying IoT applications to a multi-cloud environment using the DevOps approach.

The DRA has three main components: framework characteristics, framework architecture, and framework composition. Framework characteristics incorporate nine main elements arranged into three categories: foundation (abstraction, human factor, infrastructure), core (process, tools, product), and extended (business value, rules, legal). Framework characteristics provide the building blocks necessary to create a reference architecture design using the DevOps approach and cloud infrastructure. Framework architecture is composed of five models: contextual, conceptual, logical, physical, and operational. Framework architecture is the blueprint used in the framework composition to create DevOps pipeline instances that enable IoT application deployment to the multi-cloud environment. The DRA framework composition includes three components: resources (architecture design, software, and hardware), configuration (pipeline, IoT application, IoT network), and output (DRA reference architecture, DRAv1.0 instance, DRAv2.0 instance). The framework provides implementation instructions and an evaluation template to implement and evaluate DRAv1.0 (single cloud) and DRA v2.0 (multi-cloud) instances in different organisational contexts.

The proposed DRA framework is evaluated using an empirical evaluation composed of four iterations: industry case study, research case study, teaching case study, and industry field surveys. The results of this thesis indicate that the proposed DRA framework can be considered reasonable for the successful adoption of the DevOps approach for IoT application deployment to the multi-cloud environment. The evaluation results indicate that the DRA framework is generic and can be used in different organisational contexts and technology stacks to establish a cloud-based deployment architecture that is suitable for IoT applications.

Chapter 1: Introduction

Software development is continuously evolving from documentation-driven to more agile model-driven and collaborative ways of working (Alzoubi, Gill & Al-Ani 2015; Alzoubi, Gill & Moulton 2018). Initially, software development practices mainly focused on the development (Dev) aspects of software delivery, and little attention was given to operations (Ops) (Ahmadighohandizi & Systä 2015; Ghantous & Gill 2017; Syed & Fernandez 2016). More recently, agile approaches introduced the integrated concept of DevOps (Colavita 2016; Samarawickrama & Perera 2017; Snyder & Curtis 2017; Wang & Liu 2018). Conventional development methods, including agile, are continuously challenged by the pressing need for the fast delivery of quality solutions, including both software and hardware (Qumer, Henderson-Sellers & McBride 2007; Bai et al. 2018). This has led to the emergence of an integrated DevOps automation paradigm that was born from the necessity of frequent software releases in a production environment in the context of agile development (Lwakatare, Kuvaja & Oivo 2016b; Samarawickrama & Perera 2017; Wang & Liu 2018).

Recently, there has been increasing interest among organisations to adopt DevOps for the Internet of Things (IoT) applications (Ghantous & Gill 2018; Moore et al. 2016). IoT is a new emerging paradigm that consists of connecting physical devices and virtual objects over an established network (Bradley et al. 2015; Lee & Lee 2015; Nguyen & Gendreau 2014; Ungurean, Gaitan & Gaitan 2014; Yaqoob et al. 2017). IoT presents several challenges to organisations and developers, including security, big data, heterogeneous, multiple applications, real-time deployment, logging, and integration (Ngu et al. 2016). It has been observed that '*IoT applications require the integration of development, IT operations, and quality assurance which can be achieved by practising DevOps*' (Syed & Fernandez 2016). In contrast, the cloud computing paradigm has emerged as a technology that enables broad access and resource-sharing (Jula, Sundararajan & Othman 2014).

Notably, software development effectiveness has exponentially increased as a result of cloud efficiency and services (Avram 2014; Hu et al. 2017; Moldovan et al. 2014). The cloud provides resources that are available for IoT application deployment (Botta et al. 2016; Cavalcante et al. 2016; Hughes & Lee 2010). Further, cloud computing broader access and services can be increased by using multi-cloud environments to assist IoT application deployment (Guechi & Maamri 2017; Tao et al. 2018). It is anticipated that the multi-cloud can be used to optimise IoT application deployment using automation and continuous integration (CI) (Ghantous & Gill 2018; Srirama et al. 2016).

As stated earlier, DevOps may provide integration and automation for IoT application deployment (Gendreau & Nguyen 2014). However, there is little guidance available on how to use the DevOps approach to deploy IoT applications to the multi-cloud. Hence, there is a need to construct a reference architecture using cloud technology and the DevOps approach to enable

IoT deployment to the multi-cloud. To address this critical issue, this research presents the DevOps reference architecture (DRA) framework to assist in the adoption of DevOps for IoT application deployment to the cloud (single and heterogeneous).

This chapter discusses the background of the research and related work to agile development, the DevOps approach, cloud computing, multi-cloud, and IoT. It also outlines the research ideas derived from the research background and related work analysis. Further, this chapter presents the research problem and research question, as well as the research aim, scope, and contributions. It also outlines the application and users of the DRA. Finally, it discusses the research strategy used in the thesis and outlines the organisation and structure of the rest of the chapters.

1.1. RESEARCH BACKGROUND AND RELATED WORK

The research presented in this thesis has been conducted in the area of software engineering, incremental software development, and agile software development and especially in the context of DevOps for IoT application deployment to the multi-cloud. It is necessary to understand the origin of software engineering and compare the waterfall and agile methodologies. The aim is to understand why DevOps is required for agile development and how it can help agile. This section also discusses the topics of cloud computing, multi-cloud, and IoT. This discussion conveys valuable information about the work related to the topics of this research (DevOps, multi-cloud, and IoT). The information is used in section 1.1.7 to highlight the convergence of research ideas and identify the research problem.

1.1.1. SOFTWARE ENGINEERING

Software engineering is the application of a systematic approach to developing a software artefact (Laplante 2007). The software engineering systematic approach entails the application of scientific and technical knowledge, design, implementation, testing, quality assurance, and end-user documentation of the software artefact output [ISO/IEC/IEEE std 24765:2010(E), 2010]. The emergence of early programming languages such as FORTRAN, ALGOL, and COBOL in the 1950s led to an increase in the complexity of computing (Booch 2018; Campbell-Kelly 1982). Thus, it was necessary to adopt a common engineering discipline or methodology to address the complexity of software development. The software engineering disciplines can be traced back to the 1960s. Software engineering methodology was formally presented in the NATO Science Committee conference in Garmisch, Germany, in October 1968 (Broy 2018). The conference outlined the challenges in software development: correct software requirements, adequate architecture design, practical implementation, testing, and quality of the software artefact.

Two software engineering methodologies are commonly used in software development: waterfall and agile (Bibik 2018). The waterfall methodology adopts the incremental software development approach (Basili & Larman 2003) and aligns the development process (analysis, planning,

architecture, design, develop, test, deploy) in sequential order. Thus, a project with a broad scope may take a significant amount of time to deliver the software artefact (Bibik 2018).

1.1.2. AGILE SOFTWARE DEVELOPMENT

Agile is an iterative software development methodology. It has become a popular methodology and may have surpassed the success of the waterfall model (Basili & Larman, 2003). Usage of the iterative approach to solving complex software problems dates back to the early 1970s. Agile methods adopt an iterative approach to 'revisit' the development process (analysis, planning, architecture, design, develop, test, deploy) and hasten the release or delivery of products to customers (Bibik 2018). Figure 1.1 illustrates the improvements to software development after adopting the agile methodology. The success rate increased from 11% (waterfall) to 39% (agile). The failure rate of adopting agile principles is only 9% compared with 29% for waterfall adoption.

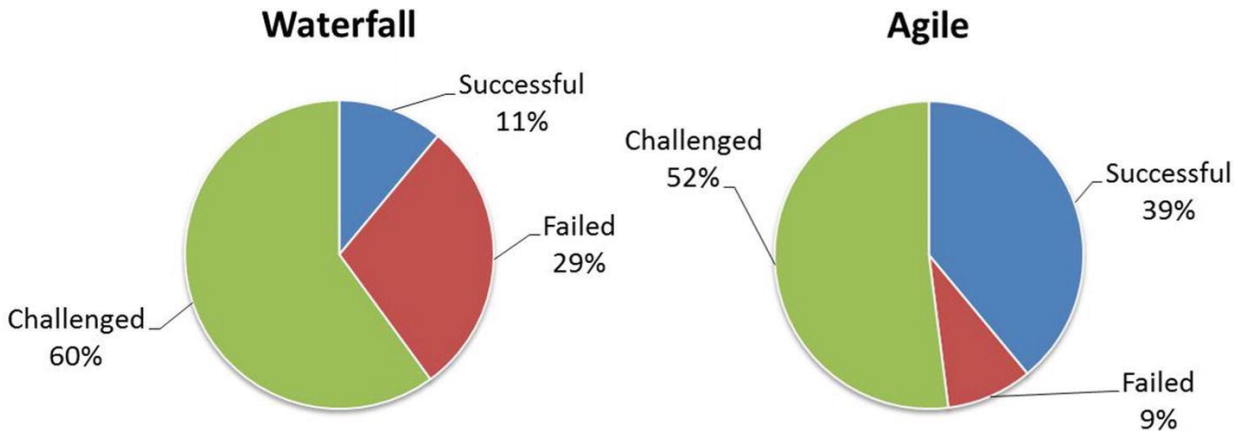


Figure 1.1: Waterfall v. Agile (Bibik 2018)

The agile manifesto for software development was published in 2001 and included 12 principles (http://agilemanifesto.org/). Agile methodologies have since become a model-driven approach that can improve the practices of software development (Brambilla et al. 2017). Model-driven engineering (MDE) has received attention from academia and industry (Favre 2005). Its success is founded on the reverse (or iterative) process model for software development. Agile methodology adopted a flexible taxonomy as MDE (Mens & Van Gorp 2006), which enabled developers to iteratively adapt to project challenges and produce improved products after repeated releases. The iterative development model required further support to increase the delivery frequency of software applications. Hence, the DevOps approach was introduced to provide agile software development with the necessary practices and tools to address the need for faster and automated software implementation (Curtis & Snyder 2017).

1.1.3. DEVOPS

The history of DevOps can be traced back to 2007–2008 and was formally acknowledged in October 2008 in Belgium (Pratibha & Khan 2018). Several definitions describe DevOps as a complex integration of development and operations capabilities (Erich, Amrit & Daneva 2014; Wettinger, Breitenbücher & Leymann 2014). For example, DevOps is a software development process that emphasises communication and collaboration between Dev and Ops (Ghantous & Gill 2017; Jabbari et al. 2016). Further, DevOps is a software development approach that enables automation for software development (Ghantous & Gill 2017; Virmani 2015). Moreover,'DevOps extends the goals of the agile movement to continuous integration and continuous delivery (Colavita 2016; Lwakatare et al. 2016b). DevOps emerged from the need for agile software development to bridge the gap between developers and information technology (IT) operations (Jha. & Khan 2018; Virmani, 2015).

Given the growing customer base and global market outreach, it became essential for software delivery to be in line with business expectations (Mohamed 2016; Perera, Silva & Perera 2017). Therefore, it was logical to introduce automation to the agile software development process. DevOps offers developers a set of well-known practices and a wide range of tools for automating the software development cycle (synchronise, build, test, deploy, release) (Ghantous & Gill 2017; Zheng et al. 2016). DevOps aims to improve the human factor experience in the context of agile software development (De Bayser, Azevedo & Cerqueira 2015; Diel, Marczak & Cruzes 2016; Sharp & Babb 2018). The automation concept of DevOps is the key to achieve CI and continuous delivery and to improve DevOps team communication and collaboration capability (Ghantous & Gill 2017).

Several frameworks claim to support DevOps in an agile environment. For instance, DICER (Artač et al. 2016) is a model-driven framework that shows how to implement DevOps practices and enable quality awareness of software application. The Continuous Scrum Framework (Samarawickrama & Perera 2017) presents a practical methodology to achieve CI and continuous delivery in an agile environment. The SQUID (Specification Quality in DevOps) (Di Nitto et al. 2016) and TOSCA (Wettinger, Breitenbücher & Leymann 2014) frameworks propose the adoption of DevOps standards based on core DevOps concepts. In summary, the DevOps approach is based on key concepts, practices, and tools (see Figure 1.2) that could be used to support a platform that enables software applications deployment and delivery in a production environment without unnecessary delays (Chen et al. 2015).

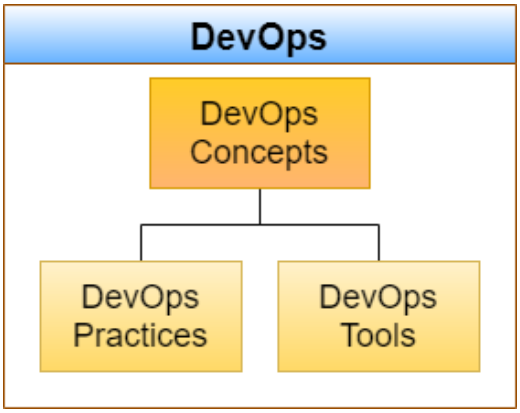


Figure 1.2: DevOps Context

1.1.4. CLOUD COMPUTING

Cloud computing is an innovative, internet-based computing model (see Figure 1.3) that provides abstract infrastructures, virtual servers, deployment platforms, and software services (Shahzad 2014). Cloud is a network of remote servers linked together to provide a single ecosystem that enables users to store data, deploy applications, and manage virtual infrastructures (Avram 2014; Jula, Sundararajan & Othman 2014). It enables ubiquitous and on-demand access to shared resources (Cloud API, software, and services). Cloud computing provides three service models (Hu et al. 2017; Moldovan et al. 2014). First, software as a service (SaaS) provides on-demand access to available cloud-based software subscriptions. Second, the platform as a service (PaaS) provides developers with a software-based environment to develop, run, manage, and test their applications without the complexity of maintaining infrastructure. Third, infrastructure as a service (IaaS) provides self-managed cloud infrastructures for developers. IaaS users are responsible for managing their applications, data, and connectivity protocols.

The cloud has five essential characteristics (Avram 2014; Hu et al. 2017): 1) resource pooling: cloud computing provides centralised access to services; 2) rapid elasticity: a fast option for users to scale their applications given the nature of the cloud as a software service (Moldovan et al. 2014); 3) measured service: resources are automatically controlled, managed and monitored by cloud providers (Lee & Hughes 2010); 4) on-demand self-service: clouds enable users to self-provision their environment and infrastructure (Hanappi, Hummer & Dustdar 2016); 5) broad network access: the cloud is globally available to users, which makes it typically suitable for agile software development (Lee & Hughes 2010). However, the adoption of cloud computing presents several challenges (Shahzad 2014): 1) security and privacy of user access, data, and applications, 2) reliability and cloud availability (no downtime), and 3) interoperability and portability standards required to increase information sharing. In essence, it may be useful to orchestrate (or group) several clouds into a multiple cloud system to increase resource-sharing (Munteanu, Şandru & Petcu 2014; Tricomi et al. 2017; Truong, Dustdar & Leymann 2016).

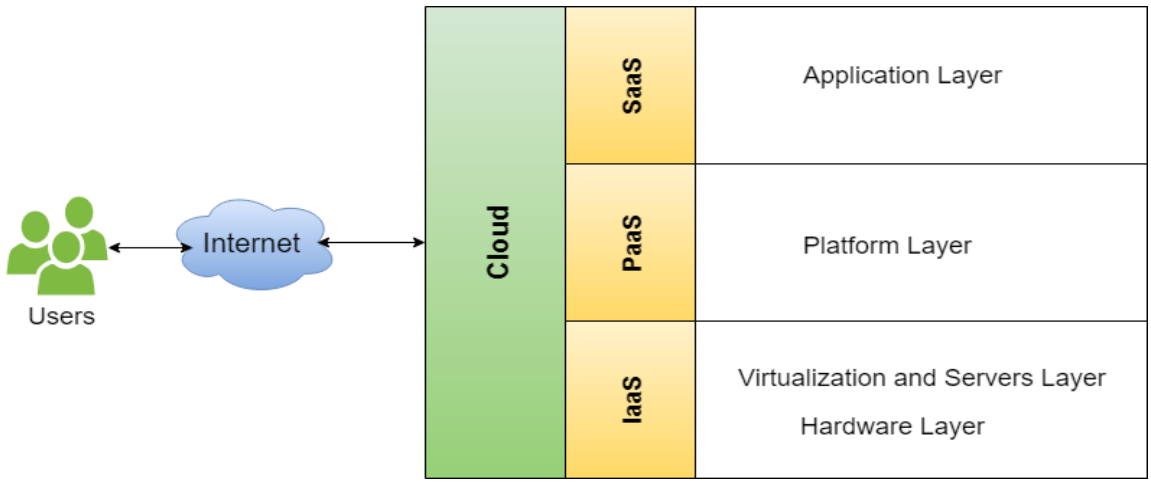


Figure 1.3: Cloud Structure

1.1.5. MULTI-CLOUD

Multiple clouds or multi-cloud (see Figure 1.4) is the integration of several cloud computing services into a single heterogeneous architecture (Hilman et al. 2020; Yang et al. 2016). Organisations, developers, and researchers can benefit from open-source cloud platforms (Munteanu, Şandru & Petcu 2014). The multi-cloud has several advantages, such as broader user access and availability of applications deployed on multiple cloud platforms (Willnecker et al. 2018; Wu & Madhyastha 2013; Yasrab & Gu 2016). Also, multi-cloud enables integration and compatibility with high-level QoS (Quality of Service) cloud services for applications (D'Agostino et al. 2013; Truong, Dustdar & Leymann 2016).

Several studies and frameworks have introduced ideas and architectures intending to achieve continuous deployment on the multi-cloud (Ferry et al. 2015; Srirama, Iurii & Viil 2016). For instance, CYCLONE (Slawik et al. 2017) is a software stack that focuses on the areas of application deployment, management, and security and user authentication on the multi-cloud. CloudMF (Ferry et al. 2018; Kumari at al. 2017) is an object-oriented domain-specific model that is tailored for applications. CloudML (Ferry et al. 2013), MCPA (Yasrab & Gu 2016), and MDE (Chondamrongkul & Temdee 2013) attempt to benefit from multi-cloud services by using automation to simplify the complexity of deployment to the multi-cloud. The deployment process of applications to the multi-cloud may follow specific migration patterns (Jamshidi et al. 2015) such as multi-cloud refactoring and multi-cloud rebinding.

In essence, the multi-cloud appears to offer many benefits to the application deployment process. However, multi-cloud adoption may present some challenges to organisations, developers, and researchers. Research shows that vendor lock-in is a central issue that may occur when adopting the multi-cloud (Ferry et al. 2013; Kritikos & Plexousakis 2015). Vendor lock-in occurs in two areas: (1) when a cloud service contains the deployment configuration of the applications (application provisioning), which prevents the other clouds in the multi-cloud model from using

the same configuration; and (2) when a cloud service hosts the database for the application, which prevents the same application deployed to other cloud vendors from storing its data in the same database. Therefore, it is essential to address both issues of vendor lock-in to benefit from multi-cloud integration for application deployment.

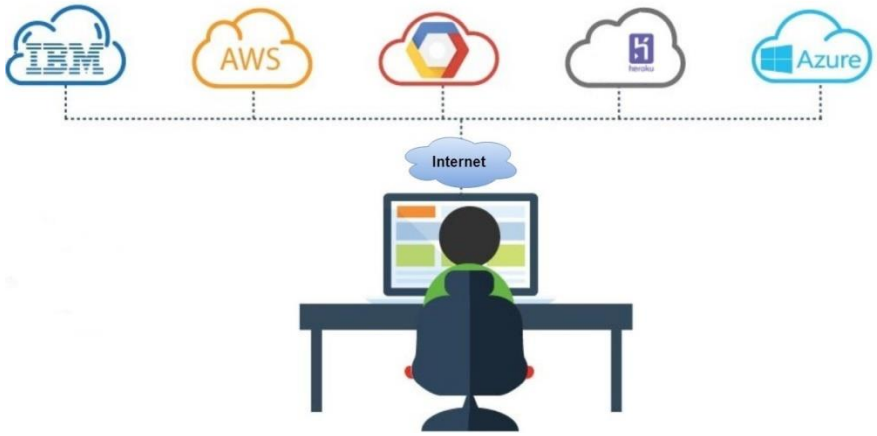


Figure 1.4: Multi-Cloud

1.1.6. INTERNET OF THINGS

IoT (see Figure 1.5) refers to a network of physical objects that feature an IP address for internet connectivity, and the communication that occurs between these objects and other internet-enabled devices (sensors and actuators) and systems (Gubbi et al. 2013; Lee & Lee 2015; Nguyen & Gendreau 2014). IoT provides human–device interactions through middleware in a secure environment (Alkhalil & Ramadan 2017; Novo and Di Francesco 2020). The middleware is the IoT applications that use existing services to provide connectivity to devices (Ngu et al. 2016). For example, Hydra provides a web service for incorporating heterogeneous physical devices into applications. In contrast, GSN (Global Sensor Network) provides a web service that delivers a stable platform for flexible integration, sharing, and deployment of heterogeneous IoT devices. Models such as SysADL (Rautmare & Bhalerao 2016) designed to preserve a system-oriented, which is an ADL (Architectural Description Language) based on SysML (Systems Modelling Language). Another model—Wi-Fi mesh infrastructure—is designed for IoT applications (Muhendra, Rinaldi & Budiman 2017). A new architecture model called 3G-PLC (Power Line Communication) (Yaqoob et al. 2017) combines two complex communication networks: 3G and PLC. These models may need a reliable gateway to control information traffic using a high-level design methodology (Chen, Lin & Guo 2017).

A study was conducted to develop an algorithm that enables an IoT gateway to automatically configure itself and perform two essential functions (Kang & Choo 2018; Rao & Shorey 2017). Ultimately, the objective is to improve human interactions with IoT devices through middleware, whether manually or automated, using an IoT sensory controller (Alowaidi et al. 2017; Russell,

Goubran & Kwamena 2015; Yonezawa et al. 2016). Connectivity of the IoT network and IoT applications requires several software and hardware technologies (RFID, WSN, Bluetooth, MQTT) (Luo & Sun 2015; Mongan et al. 2017, Muhendra, Rinaldi & Budiman 2017; Newmarch 2016; Su et al. 2014; Wu et al. 2013). Sensor nodes collect data through a WSN (Wireless Sensor Network), Bluetooth, MQTT, SSH, LPWAN (Low-Power WAN), or NFC, which enable the convergence of the IPv6 network and low-power wireless (Luo & Sun 2015; Shah & Mishra 2016). Connectivity protocols are configured on a Linux operating system running on Raspberry Pi (Newmarch 2016). A RESTful WS can be used to manage IoT devices (Sheng et al. 2015); the service can be scaled and deployed on a cloud platform.

Many practical architecture models have already been deployed on various sensor networks. First, in EasyConnect (Lin et al. 2015), devices are attached to EasyConnect using a friendly GUI (graphical user interface) that is compatible with most smart devices. Second, smart city multipurpose architecture is based on the monitoring of environmental variables in urban areas (Gómez et al. 2017). Third, SOXFire (Mongan et al. 2017) supports physical access to IoT devices that require a high-level plan for a system of systems (SoS). For this purpose, service lifecycle management (SLM) (Hefnawy, Bouras & Cherifi 2016), which is based on lifecycle modelling language (LML), was suggested to analyse, plan, design, build and maintain IoT-enabled smart city service systems. Fourth, AllJoyn (Khakimov et al. 2017) uses HTTP and MQTT protocols for communication through the gateway. Fifth, self-configuration and wisdom connection systems use WSN and actualise automated configurations based on RSSI (Hsieh et al. 2016).

The performance of IoT applications requires constant improvements to handle the increasing amount of IoT data, which are generated from interactions between IoT applications and IoT devices. These data are stored either in traditional SQL tables or using a NoSQL database (Douzis et al. 2018; Rautmare & Bhalerao 2016). Given the continued growth of IoT networks (sensors and actuators), it is necessary to enhance the performance of IoT applications using real-time monitoring and detection of critical events in the COT (Cloud of Things) (Lee & Hughes 2010). Existing models could be used to automatically monitor physical sensors such as LabVIEW (Wu et al. 2013), AnyControl (Wang et al. 2015), and TISH (Titled Time of Series Histogram) (Wu et al. 2013). Sensor events in a smart environment are monitored at run time, which enables decision-making based on data stream analysis (Agrawal et al. 2020; Kodeswaran et al. 2015; Mongan et al. 2017). However, the research shows that IoT is extensive and requires a comprehensive model to manage IoT application deployment, IoT connectivity using an adequate connectivity protocol, and IoT data storage. In essence, cloud computing (see Sections D and F) may provide IoT with the necessary services to achieve the required model.

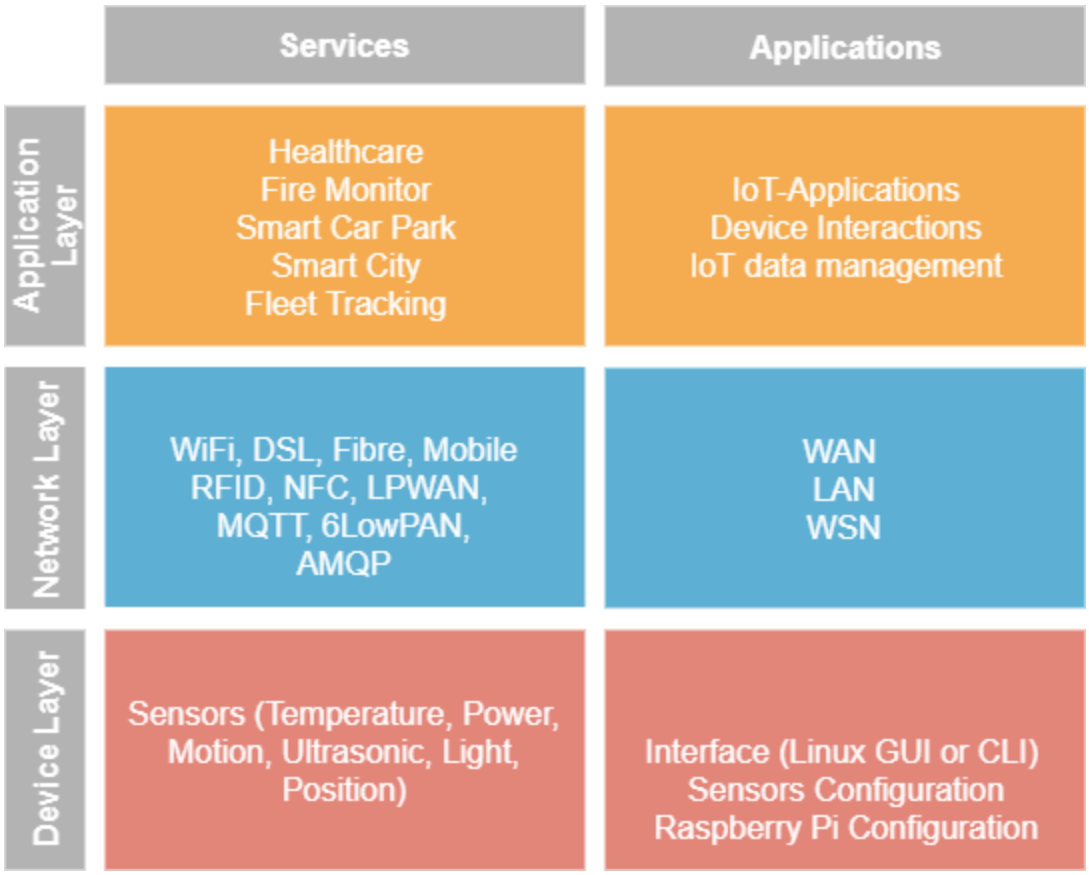


Figure 1.5: IoT Generic Structure

1.1.7. Convergence of Research Ideas

This section discusses the convergence of research ideas (see Table 1.1), which helps to identify the research aims in section 1.4. The convergence of ideas is a list of observations identified in previous sections of this chapter concerning the analysis of the background studies and related work. These ideas are selected from the analysis conducted in sections 1.1.1 to 1.1.6. Section 1.1.3 shows that DevOps concepts may vertically improve agile application deployment and delivery in a production environment. Section 1.1.4 discusses cloud computing characteristics and services. Section 1.1.5 shows that it may be useful to use the multi-cloud for application deployment; section 1.1.5 also identifies key challenges in the process. Section 1.1.6 indicates that IoT may require a model-driven architecture that benefits from cloud computing services to manage IoT data and IoT applications (Botta et al. 2016; Ray 2016). In essence, research shows that the deployment of IoT applications to the multi-cloud lacks adequate support and guidelines. Table 1.1 identifies the critical research elements that could be used to outline the research problem and highlight the research question of this thesis.

Table 1.1: Ideas Convergence

Idea	Context Relationship	Description
I_1	Multi-cloud and IoT	IoT applications can benefit from cloud services (auto-scaling, virtual servers).
I_2	Multi-cloud and IoT	IoT data can benefit from cloud data management services (cloud database, availability).
I_3	DevOps and multi-cloud	DevOps tools may support multi-cloud data management to prevent vendor lock-in.
I_4	DevOps and IoT	IoT application deployment can benefit from DevOps concepts (automation, integration).
I_5	Multi-cloud and IoT	IoT applications can benefit from multi-cloud services.
I_6	DevOps and IoT	IoT application deployment can benefit from DevOps practices that enable automation for software development chain (synchronise, build, test, deploy, deliver, and monitor).
I_7	DevOps and multi-cloud	Multi-cloud can benefit from the DevOps approach. DevOps tools may be integrated with cloud services.
I_8	DevOps and multi-cloud	DevOps tools may be used for provisioning multi-cloud application deployment to prevent vendor lock-in.
I_9	Agile and DevOps	DevOps approach may improve agile application deployment speed and frequency.

1.2. RESEARCH PROBLEM

This section discusses the main research problem. The research problem and underlying statements are identified based on the analysis conducted in the research background and related work (see Section 1.1). The adoption of the DevOps approach in agile software development improves the delivery of the software application (Lwakatare, Kuvaja & Oivo 2016b; Perera, Silva & Perera 2017). Agile software development achieves vertical improvement through the DevOps concepts of automation and CI (Schaefer, Reichenbach & Fey 2013; Snyder & Curtis 2017; Wahaballa et al. 2015; Wettinger, Breitenbücher & Leymann 2014). Hence, it is crucial to conduct an extensive investigation into the DevOps paradigm and to identify the DevOps concepts, practices, and tools.

The thesis also discusses cloud computing and the emergence of the multi-cloud phenomenon. Cloud computing provides on-demand services and abstract infrastructure for developers (Jula, Sundararajan & Othman 2014; Moldovan et al. 2014). It is essential to investigate the measures and factors required to improve application deployment to the cloud. Further, heterogeneous clouds, or the multi-cloud, have emerged as a combined abstract architecture that enables broader user access, a diverse bundle of services, and a higher level of application scalability (Papaioannou, Metallidis & Magoutis 2015; Willnecker & Krcmar 2018; Wu & Madhyastha 2013). The major obstacle for adopting multi-cloud distributed deployment is vendor lock-in, which prevents harmonious deployment and database integration for the software application (Chondamrongkul & Temdee 2013; Kritikos & Plexousakis 2015; Yasrab & Gu 2016). Hence, it is crucial to explore adequate options that support application deployment to the multi-cloud.

The third topic of discussion is IoT—in particular, the deployment of IoT applications. Section 1.1 showed that IoT applications require deployment platforms that enable auto-scaling, secure user access, and a reliable database management system (Douzis et al. 2018; Syed & Fernandez 2016). Cloud computing appears to offer the services that may be necessary to support application deployment (Atif, Ding & Jeusfeld 2016; Botta et al. 2016; Lee & Hughes 2010). Further, multi-cloud architecture appears to offer broader access, broader services, and an API (application programming interface) (Ghantous & Gill 2018; Papaioannou, Metallidis & Magoutis 2015; Slawik et al. 2017). Therefore, it is essential to understand how IoT applications can be provisioned to the multi-cloud and to discover suitable solutions to the vendor lock-in problem, and the IoT application connectivity with the physical sensors.

As stated earlier, there has been interest in using DevOps for IoT applications in the multi-cloud; however, the challenge is how to do so. Thus, the main research problem is how to create a DRA that enables IoT application deployment to the multi-cloud using the DevOps approach. This challenge leads to the definition of the research question, which is discussed in the next section.

1.3. Research Question

RQ: How can IoT applications be deployed to the multi-cloud using the DevOps approach?

The research question dimensions indicate that the adoption of DevOps may first require investigation and analysis and that a deployment method is needed to enable IoT application deployment to the multi-cloud. The research question dimensions in Figure 1.6 show how DevOps may support IoT application deployment to the multi-cloud.

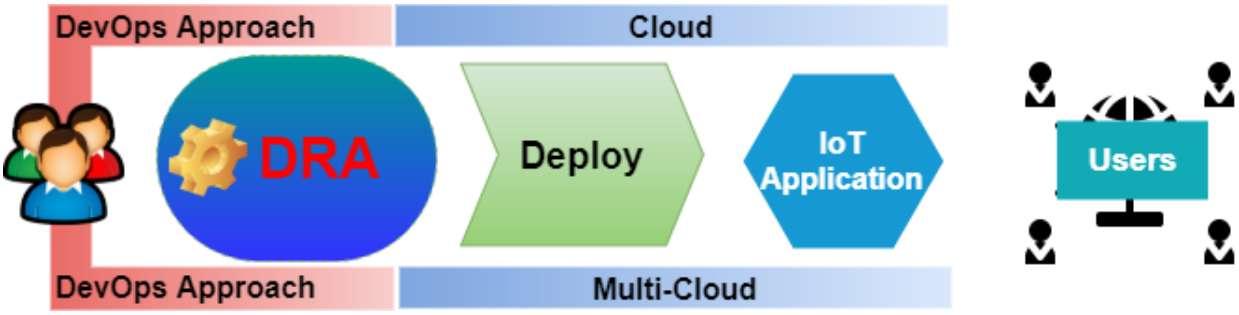


Figure 1.6: Research Question Dimensions

The research dimensions in Figure 1.6, highlights the complexity of the RQ, which requires an extensive investigation into the relationship between DevOps, the multi-cloud, and IoT. Hence, the thesis RQ is subdivided into two questions (see Table 1.2). This approach has divided the research question into an investigation about DevOps (RQ_1) and the creation of a comprehension model to enable IoT application deployment to the multi-cloud using DevOps (RQ_2). The research question underlying the objectives involves the convergence of ideas (see Table 1.1). Table 1.1 identified key research ideas that should be considered when addressing the research question. When addressing RQ_1, it is necessary to identify the benefits and challenges of adopting DevOps for IoT.

Further, the results should highlight the research gaps and identify the industry needs. Most notably, for RQ_2, it is necessary to address the issue of vendor lock-in that may occur when provisioning IoT applications to the multi-cloud, and when a cloud from the multi-cloud set hosts the IoT database. The solution for RQ_2 should contain an innovative instrument encapsulated in a comprehensive model that addresses the stated issues. The research questions RQ1 and RQ2 aims are explained in detail in section 1.4 (Research Aims).

Table 1.2: RQ Sub-Division

Label	Question	Dimension
RQ_1	What is known about DevOps?	Cloud DevOps
RQ_2	How can IoT applications be deployed to the cloud (single and heterogeneous) using the DevOps approach?	Cloud Multi-cloud IoT DevOps

1.4. RESEARCH AIM

The research aim represents the underlying objectives of the research question. With the highlighted research question in mind (see Table 1.2), the main aim of this thesis is to develop a structured framework—the DRA framework—to enable the deployment of IoT applications to the multi-cloud using the DevOps approach. The research aims to support the research question. The research aims (see Figure 1.7) are encouraged by the convergence of ideas (see Table 1.1).

The research aims (see Figure 1.7) are outlined as follows:

- Understand the DevOps approach: analyse the related studies conducted on DevOps and present the results of DevOps concepts, practices, tools, benefits, and challenges.
- Understand cloud architecture: analyse the related studies conducted on cloud computing and the multi-cloud paradigm.
- Understand IoT architecture: analyse the related studies conducted on IoT and present the findings of the IoT relationship with cloud, multi-cloud, and DevOps.

- Integrate DevOps tools with cloud services: This is the first step towards building an operational model pipeline. The DevOps tools and cloud services are identified and analysed in previous steps.
- Deploy IoT applications to a single cloud: create an operational model pipeline using the DRA design models. The DRA development pipeline is constructed using DevOps tools and cloud software and services.
- Avoid vendor lock-in issues when deploying IoT applications to the multi-cloud. In essence, the DRA framework should include an instrument that deploys IoT applications to the multi-cloud and addresses the issue of vendor lock-in of application provisioning.
- Avoid vendor lock-in issues when storing data generated by IoT applications to a database hosted by a cloud from the multi-cloud set. In essence, the DRA framework should include an instrument that addresses the issue of vendor lock-in that is caused when a cloud vendor hosts an IoT applications database.
- Deploy IoT applications to the multi-cloud using a comprehensive model based on the DevOps approach.

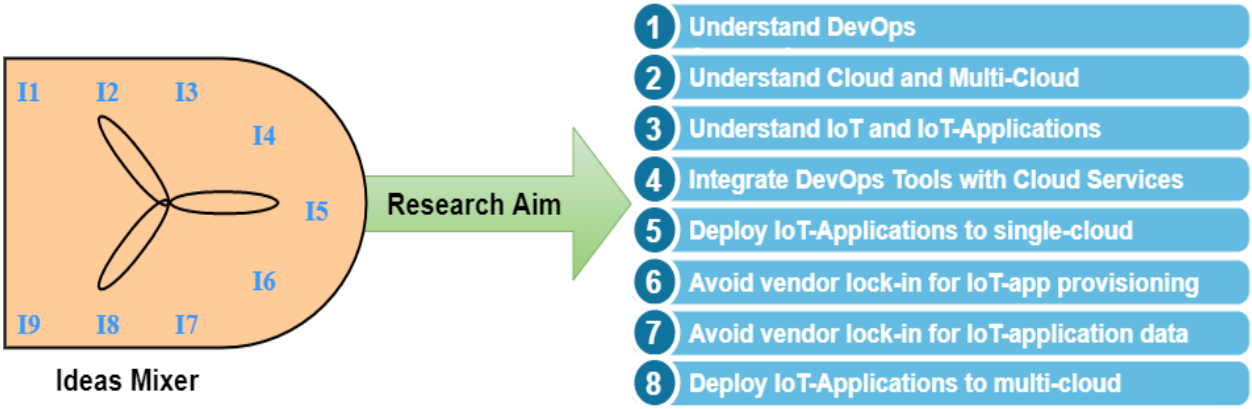


Figure 1.7: Research Aim

1.5. RESEARCH SCOPE

The research scope of this thesis is limited to DevOps, IoT applications, and their deployment to the multi-cloud (see Table 1.3). The thesis output—namely, the DRA—will cover the objectives explained in Table 1.3, which also contains the excluded research items. Table 1.3 scope includes the construction of the DRA framework as a general reference architecture that enables software application deployment to multi-cloud (including IoT-applications). A reference architecture (Chen et al. 2016) defines a cooperative relationship of technology components and methods integrated to create a comprehensive template solution for software development projects. A reference architecture is an abstract solution pattern that can be re-used by organisations to improve the interoperability, collaboration, communication, and governance of software projects. The DRA framework reference architecture aims to provide a general design

model for IoT-applications to multi-cloud and accomplish the project's aims, and scope explained in Table 1.3 and Figure 1.7.

Table 1.3: Project Scope

Inclusion	Exclusion
Investigate DevOps, multi-cloud, and IoT and their relationships. Outline DevOps concepts, practices, and tools and determine the benefits of the DevOps adoption for IoT application deployment to the multi-cloud.	The DRA framework will not provide specific and custom security measures for the DevOps pipeline, which is a separate PhD topic of research.
Use DevOps concepts and practices to define general characteristics that can be used to create a reference architecture model using the multi-cloud infrastructure.	The DRA framework will not provide specific and custom security measures for the IoT network, which is a separate PhD topic of research.
Construct a general reference architecture model that can be implemented in any context.	The DRA framework will not offer a solution for IoT data management (taking actions based on data, recycling IoT data, storage management, and data synchronisation).
Create an operational model pipeline to deploy IoT applications to the multi-cloud.	
The DRA framework should include a tool that hosts IoT application provisioning and prevents vendor lock-in. This tool should enable deployment to the multi-cloud.	The DRA will not provide a model to manage vendor lock-in caused by IoT application provisioning, which is a separate PhD topic of research.
The DRA framework should include a tool that hosts IoT data storage and prevents vendor lock-in.	The DRA will not provide a model to manage vendor lock-in caused by IoT data storage hosting, which is a separate PhD topic of research.
Provide a DRA framework composition that can be used to define architecture set output. The framework composition includes DRA design models, DRA operational model pipeline (instance), IoT application, and IoT device (IoT network). DRA composition is used to create an implementation template that can be used as a guideline to implement the DRA in the context of the current development environment.	
Analyse the applicability of the DRA based on empirical evaluation results, determine research contributions, and outline DRA limitations and possible future research ideas.	

1.6. CONTRIBUTIONS

This section outlines the main contributions of the research—the DRA framework (see Figure 1.8). The framework incorporates several ideas and methods that describe the contextual relationship of DevOps, multi-cloud, and IoT at a contextual higher level. The DRA is rich in range and includes numerous concepts, practices, and tools that have not been previously explicitly discussed or integrated into a practical or theoretical framework. The contributions of this research have been published in both industry, and academic outlets (e.g., Ghantous and Gill

2017, 2018, 2019), and they have been incorporated into the teaching of advanced software development at the University of Technology Sydney.

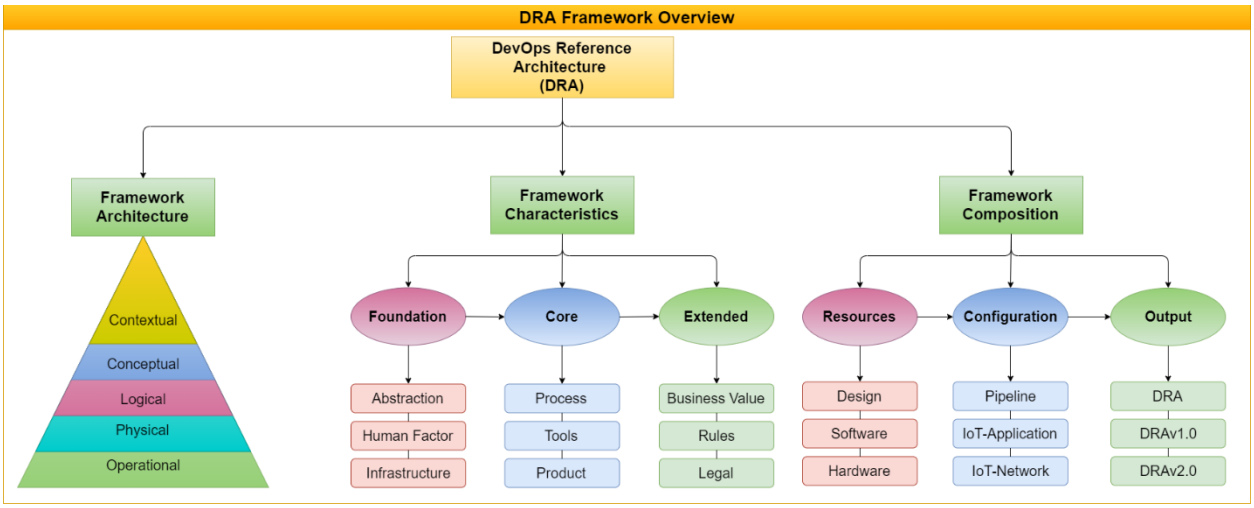


Figure 1.8: DRA Framework Overview

The DRA overview (see Figure 1.8) shows that the framework is composed of three main components: framework characteristics, framework architecture, and framework composition. Framework characteristics are general terminologies that could be used to create a reference architecture design using a DevOps approach and cloud infrastructure. Framework architecture is a general design model that can be applied in any context. DRA composition represents the resources and configurations required to create a DRA operational model instance. The DRA operational model instance enables IoT application deployment to the multi-cloud.

The framework characteristics component includes nine key elements that are organised into three categories: foundation (abstraction, human factor, infrastructure), core (process, tools, product), and extended (business value, rules, legal). Framework characteristics are defined using common terminology following the guidelines published in Berger, Häckel, and Häfner (2019) and Nickerson, Varshney, and Muntermann (2013). DevOps and cloud infrastructure support the characteristics' terminologies. The framework characteristics can be used in any context to create a reference architecture design.

The DRA architectural design is composed of five models: contextual, conceptual, logical, physical, and operational. The architecture models are published in Ghantous and Gill (2018). The DRA is created using general framework characteristics. Consequently, the DRA is a general design that represents a class of domains and is not restricted to particular instances. Instead, The DRA design model can be implemented in any development context and applied to numerous instances.

The DRA composition entails the resources (framework architecture, software, hardware) and configuration (DevOps tools integration, IoT application setup, IoT network setup) used to create a DRA operational model instance (DRAv1.0: single cloud; DRAv2.0: multi-cloud) to deploy IoT applications. The DRA composition is an application of the framework's reference architecture design model. The DRA composition outlines two outputs: DRAv1.0 instance (single-cloud deployment) and DRAv2.0 instance (multi-cloud deployment). DRAv2.0 is an updated version of DRAv1.0 that enables IoT application deployment to the multi-cloud using the DevOps approach.

The proposed framework has been evaluated using empirical evaluation (see Chapter 5), which was conducted in four iterations: industry case study, research case study, teaching case study, and industry field surveys. The encouraging results of the empirical evaluation suggest that the DRA framework may represent a high-level contextual integration of the DevOps, multi-cloud, and IoT contexts. The evaluation results are used to determine the applicability and novelty of the framework. The evaluation feedback helped to convey possible future research ideas. The results also indicate that the DRA may be considered an appropriate and practical model for IoT application deployment to the multi-cloud.

Also, serval contributions have been accomplished during this research. These contributions have been published in conference volumes and peer-reviewed journals (Ghantous & Gill 2017, 2018, 2019).

1.7. APPLICATION AND USERS

This section discusses the DRA framework application and its users. The DRA is intended for use by software organisations and experienced DevOps developers, engineers, managers, and consultants as a general reference model for IoT application deployment to the multi-cloud using the DevOps approach. The DRA should not be taken as an imposed heavy-handed step-by-step process; instead, it should be taken as a comprehensive information framework to ensure that the essential points are not missed. DevOps consultants and managers may use the proposed framework models as a general reference architecture that represents a class of domains not fixed to a particular instance. The proposed DRA instances are tailored to support numerous instances that are applicable in the contexts of the development environment.

The DRA application and its users are illustrated in Figure 1.9, which shows the interactions between DRA users or DevOps teams (consultants, developers, managers, coaches) and the framework's three mains components (characteristics, architecture, composition). Figure 1.9 outlines the user's experience with the DRA setup (using the framework characteristics), design (creating the reference architecture model), and application (configuring and implementing the DRA pipeline instance). The DevOps team sets up the DRA in the context of the organisation employing the framework's characteristics using DevOps (concepts, practices, tools) and the

cloud (infrastructure, virtual servers, services). Managers and consultants plan the DRA design based on the foundation's characteristics. At the same time, developers and coaches build the DRA pipeline and configure the IoT application deployment to the multi-cloud using a CI broker.

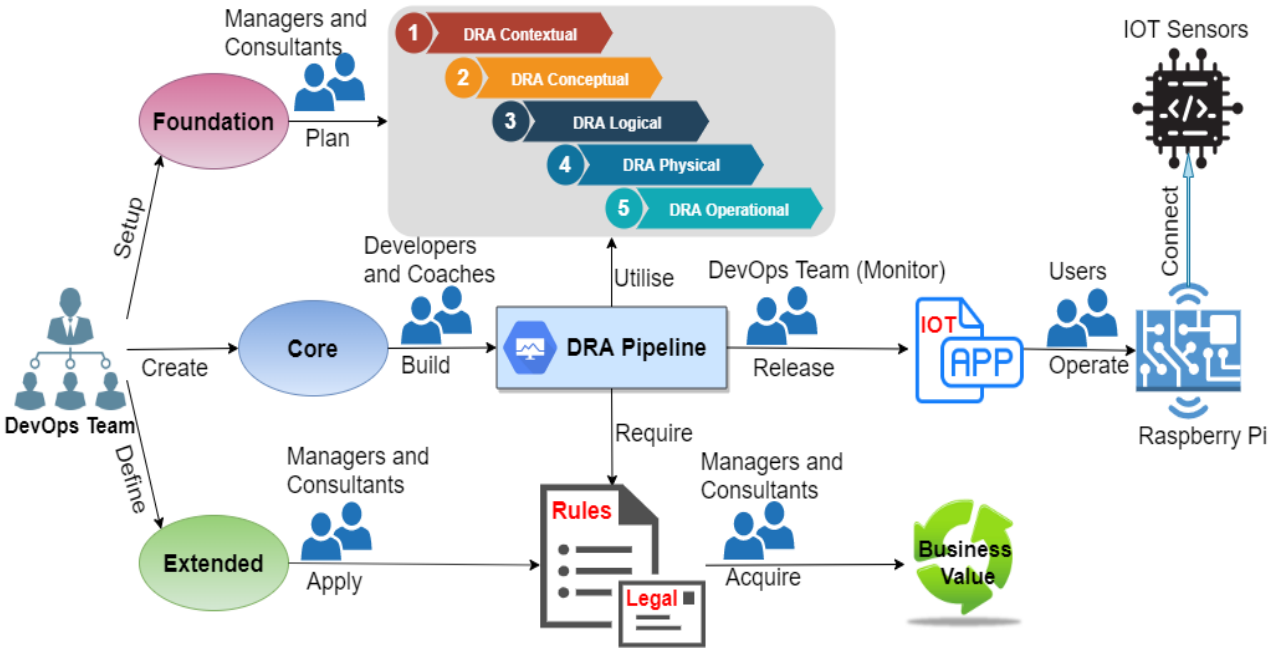


Figure 1.9: DRA Application

The CI broker hosts the deployment configurations to avoid vendor lock-in. Managers and consultants arrange the business rules and legalities applied in the context of the DRA implementation. Managers and consultants determine the business value of the DRA application in the organisation context. The DevOps team monitors the IoT application operations by end-users in real-time. The IoT application operations and interactions with the IoT network generate IoT data. The DRA pipeline also contains a tool to host IoT data storage and avoid vendor lock-in. The IoT network is composed of Raspberry Pi and IoT sensors. The Raspberry Pi computer board runs a Linux operating system and contains Python scripts that control the IoT sensors. The IoT application interacts with the Raspberry Pi board using numerous connection protocols (e.g., SSH, MQTT, AMQP, Mobile). The DevOps team has complete control over the DRA implementation using DevOps practices and tools (real-time monitoring, application logging, and collaboration tools).

1.8. RESEARCH STRATEGY

The main research question of this thesis is: 'How can IoT applications be deployed to the multi-cloud using the DevOps approach?' To answer this question, this research proposes the main research contribution—the DRA framework—which has been briefly introduced and discussed in this chapter. This section presents an overview of the research strategy that has been adopted to design, develop, and evaluate the DRA artefact output.

To address the main research question, the research question is divided into two sub-questions: RQ_1 and RQ_2 (see Table 1.2). The research question underlying the statements involves the convergence of ideas (see Table 1.1). Table 1.1 identified key research ideas that should be considered when addressing the research question. With the research question in mind, the key ideas in Table 1.1 are converted into a set of objectives (see Figure 1.7). To address these objectives, this research uses a design science research (DSR) methodology following the guidelines adopted from Gregor and Hevner (2013) and Peffers et al. (2007).

The initial step of the DSR is to conduct a systematic literature review (SLR) (see Chapter 2) adopted from guidelines by Kitchenham and Charters (2007). The SLR filters, analyses, and synthesises data from a collection of relevant studies. The SLR filtration process is composed of five stages: 1) inclusion and exclusion process of related research papers; 2) data source location and database search strategy; 3) selection criteria and inclusion decision; 4) final selection process and quality assessment based on criteria questionnaire (Dybå & Dingsøyr 2008); and 5) data extraction and synthesis from final collection of studies. The results of the SLR are used as initial data for the DSR method.

The DSR methodology is used to develop the DRA artefact. Artefact development may involve a review of existing theories and knowledge to develop a rigorous solution or artefact for the intended purpose and audience. The DSR process is composed of six steps: problem identification, analysis, design, development, evaluation and output.

The DRA framework is evaluated using an empirical evaluation composed of four iterations: industry case study, research case study, teaching case study, and industry field surveys. The DRA is then updated based on feedback received from participants. Qualitative data from the case studies are collected, analysed, and reported using the guidelines for case studies (Runeson & Höst 2009). Survey questionnaires are developed for the artefact evaluation (Prat, Comyn-Wattiau & Akoka 2014). The survey produced quantitative and qualitative data. The qualitative data are evaluated following the validation criteria for design research (Carvalho 2012), and the quantitative data are evaluated using statistical formulas for numerical data analysis (Hyndman 2008). The DRA applicability and novelty is determined using the results of the empirical evaluation.

1.9. THESIS ORGANISATION

This thesis is organised into six chapters, as illustrated in the organisation overview in Figure 1.10. Chapter 1 presents the thesis introduction and discusses the research background, research aims, research problem, research question, project scope, research contributions, and research strategy. Chapter 2 presents the SLR used in this research. Chapter 3 presents the DSR adopted in this research. Chapter 4 presents the main thesis contribution—namely, the DRA framework. Chapter 5 presents the empirical evaluation used to evaluate the DRA. Chapter 6 highlights the contributions of this thesis and outlines the framework's limitations and ideas for future research. The rest of the thesis includes a research discussion, conclusion, and appendices. The appendices contain the research data (link to CloudStor, the cloud storage recommended by the University of Technology Sydney [UTS]), publications, ethics approval and consent forms, online survey template, and case study template. The thesis also includes a brief declaration about the author.

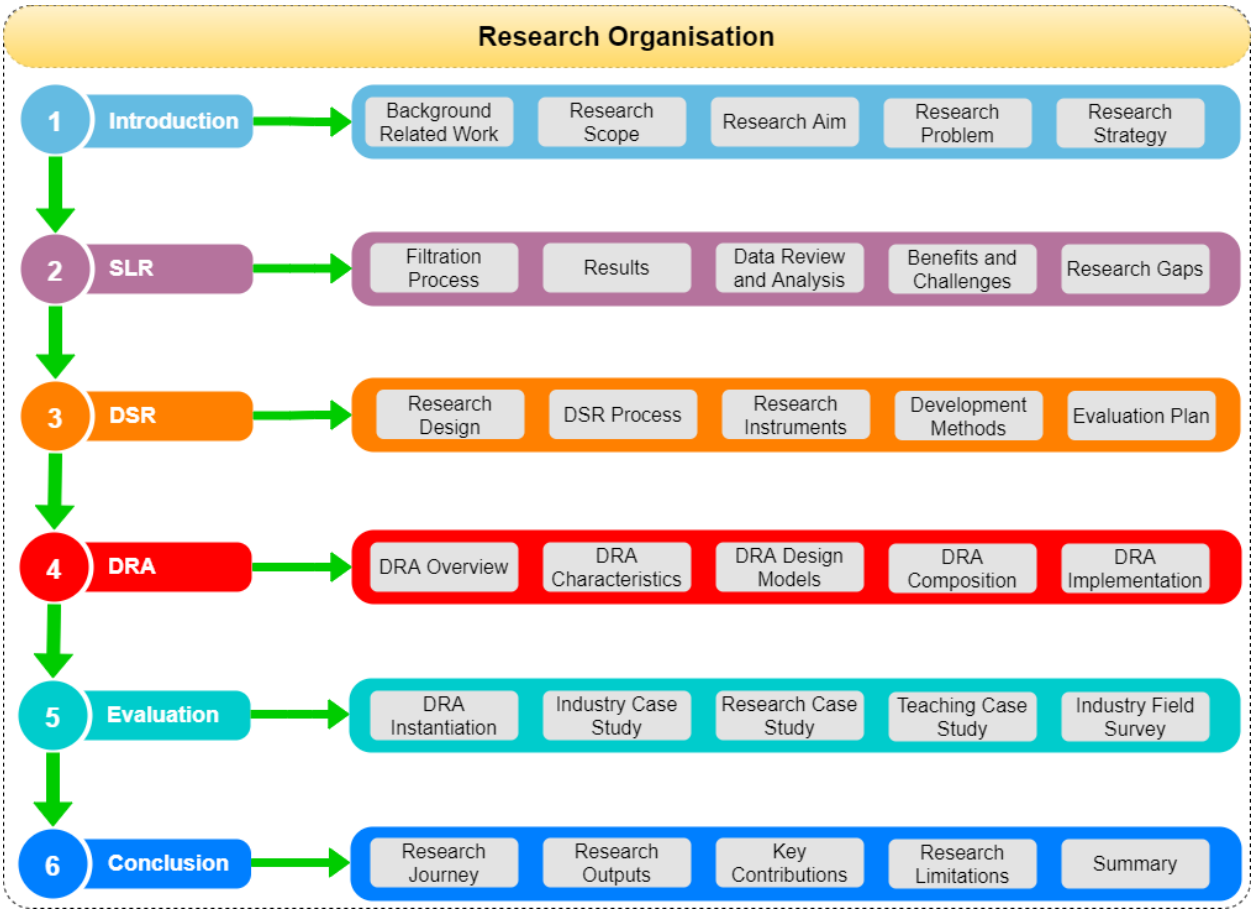


Figure 1.10: Research Organisation

1.10. SUMMARY

This chapter presented an analysis conducted in the area of software development and, in particular, the area of agile software development. It has been suggested that a DevOps approach may improve agile through automation and CI of the development process. The topics of IoT and cloud computing (single and multi-cloud) were discussed. It was suggested that DevOps might improve the deployment of IoT applications to the multi-cloud. This research has proposed the DRA as model architecture to address the main research question: How can IoT applications be deployed to the multi-cloud using the DevOps approach? The framework is composed of three main components: framework characteristics, framework architecture, and framework composition. Framework characteristics are general terminologies that can be used to create a reference architecture design using a DevOps approach and cloud infrastructure. The framework architecture is a general design model that can be applied in any context. The DRA composition represents the resources and configurations required to create DRA operational model instances. The DRA is not fixed to a particular context but can be applied to numerous instances. The DRA framework will be discussed in detail in Chapters 3–5. It is intended to be used by developers, coaches, managers, and consultants as a comprehensive template or model to deploy IoT applications to the multi-cloud in the organisation context. In Chapter 2, the adopted SLR process attempts to answer RQ_1.

Chapter 2: Literature Review

This chapter presents a systematic review of DevOps in the context of IoT application deployment to a multi-cloud environment. While DevOps practices and tools are being embraced by the industry, little research has been undertaken into how DevOps can be used for emerging IoT application deployment to a heterogeneous and complex multi-cloud environment. Here, an SLR method (Kitchenham & Charters 2007) has been used to systematically review the DevOps concepts, practices, tools, challenges, and existing solutions. This review aims to establish the current state of knowledge to locate a study area in DevOps in the context of IoT application deployment and the multi-cloud. The SLR is the first step in DSR towards the development of the initial version of the unified DRA to assist in the deployment of IoT applications to the multi-cloud (Ghantous & Gill 2017, 2018).

This research adopts an SLR to filter, analyse, and synthesise data from a collection of relevant studies. The SLR is founded on the guidelines of Kitchenham and Charters (2007). The SLR filtration process is composed of five stages: 1) inclusion and exclusion process of related research papers; 2) data source location and database search strategy; 3) selection criteria and inclusion decision; 4) final selection process and quality assessment based on criteria questionnaire (Dybå & Dingsøyr 2008); and 5) data extraction and synthesis from final collection of studies.

The SLR examines and discusses DevOps, cloud computing, multi-cloud, and IoT, as well as their relationships. The purpose of this analysis is to determine what is known about the DevOps, IoT and multi-cloud in order to the research gaps and address the research question. The SLR outlines the benefits and challenges of using the DevOps approach for IoT application deployment to the cloud/multi-cloud. The findings in the results section are used to determine the existing research gaps and address the agile software implementation (Alzoubi, Gill & Al-Ani 2015; Qumer, Henderson-Sellers & McBride 2007). The SLR augments the focus on the perspectives of DevOps, cloud/multi-cloud, and IoT and outlines the necessary steps to integrate these contexts and their relationship. The SLR results are used as initial data in the initiation step of the DSR method (Gregor & Hevner 2013; Peffers et al. 2007) adopted in this thesis to develop the proposed DRA framework.

2.1. SLR SCOPE

The scope of the SLR is based on the research aim in Section 1.4, which defines the underlying objectives of the main research question (see Section 1.3). The SLR is conducted following the guidelines published by Kitchenham and Charters (2007) to address the research questions (in particular, RQ_1) (see Table 1.2). The SLR is conducted using a list of selected studies. The selected studies are chosen based on their relevance to the topics of the research (DevOps, cloud [single and multiple], IoT), and their relationships. The selected studies in this research are

labelled $\mathbf{S_{[index]}}$, where **index = [1 ... 128]** refers to a corresponding study listed in the Bibliography of this thesis. The results of the SLR filtration process are assessed for quality using guidelines adopted from Dybå and Dingsøyr (2008). This process generates rigorous raw data, which are reviewed in the SLR results section. The results analysis intend to determine what is known about DevOps (concepts, practices, tools), identify the benefits and challenges of the adoption of the DevOps approach to deploy IoT applications to the cloud/multi-cloud, which identify and highlight the research gap that primarily describes the industry's needs.

2.2. SLR FILTRATION PROCESS

The SLR in this thesis is composed of five stages. Stage 1 discusses the type of studies included and excluded in the study. Stage 2 describes the search strategy based on a well-crafted search key. Stage 3 describes an iterative selection process conducted on a range of related studies. Stage 4 describes a further process of study inclusion and exclusion based on quality assessment criteria. Stage 5 separates the synthesised data from the selected studies into categories related to the research technologies.

2.2.1. STAGE 1: INCLUSION–EXCLUSION

The SLR only included papers that presented quality data that were deemed helpful to address the research question. The studies were selected if they satisfied the minimum quality assessment criteria, as discussed in stage 4. Papers included journal articles and conference proceedings written in English. Blogs, magazine websites, and book pages, as well as duplicates and non-English papers, were excluded from the review because of their perceived low academic study relevance and rigour. Table 2.1 shows the inclusion and exclusion benchmarks for the studies.

Table 2.1: Inclusion–Exclusion Benchmark

Inclusion Benchmark	Exclusion Benchmark
Journals	Papers from blogs
Conference proceeding	Magazine websites
Papers from IEEE, Elsevier, Springer, Google Scholar, and ACM	Book pages
Papers published between 2007 and 2009	Duplicate papers
Papers are written in English	Non-English papers

2.2.2. STAGE 2: DATA SOURCE AND RESEARCH STRATEGY

This study uses the following well-known databases as data sources:

- IEEEXplore (www.ieeexplore.ieee.org/Xplore/)
- ACM Digital Library (www.portal.acm.org/dl.cfm)
- Google Scholar (http://scholar.google.com.au/)
- Elsevier Science Direct (www.sceincedirect.com/)
- SpringerLink (www.springerlink.com/).

It was anticipated that the search results from these databases would provide sufficient data coverage for the SLR. The search key was created based on the research question (see Section 1.3) and the research aim (see Section 1.4). The search key's logical expression parameters were fit for purpose to investigate (DevOps, cloud [single and multiple], IoT). Table 2.2 shows the search categories that form the search key and its associated key terms. The initial search produced N = 950 papers.

Search Key = DevOps and IoT, DevOps concepts, DevOps tools, DevOps architecture, DevOps practices, IoT architecture, IoT sensors, IoT applications, IoT security, cloud computing, and IoT, DevOps and multi-cloud, multi-cloud and IoT.

Table 2.2: Search Categories

Search Category	Key Terms
DevOps practices	Collaboration, communication, automation, quality assurance, fast release, logging, monitoring, testing, build, deployment, database management.
DevOps architecture	Development pipeline architecture, agile architecture, DevOps framework, TOSCA, SQUID, DICER, DRA.
DevOps concepts	Automation (build, testing, deployment, monitoring, communication, collaboration, planning). Continuous integration.
DevOps tools	Travis CI, Jenkins, Codeship, Github, BitBucket, MongoDB, Docker, Azure, AWS, Heroku, Papertrail, Trello, Slack, Cucumber, Junit, TestNG, Nagios.
DevOps and IoT	PaaS, IaaS, Cloud programming, continuous updating, Cloud API, Continuous testing, continuous reporting/logging.
IoT architecture	IoT frameworks and architectures, connectivity protocols, RFID, SenML, WSN, Middleware, ADL, MQTT, LPWAN, AMQP, Mobile.
IoT devices and sensors	Proximity sensors, motion sensors, temperature sensors.
IoT applications	IoT process, IoT interactions, IoT applications connectivity, Raspberry Pi.
IoT security	P2P architecture, two-way authentication.
Cloud and IoT	MQTT on the cloud, continuous deployment, fast delivery.
DevOps and cloud	Automation, CI (cloud services, DevOps tools), real-time monitoring and logging, automated testing.
Multi-cloud	Multiple cloud services, broader access, application provisioning model.
Multi-cloud and IoT	Broader access, vendor lock-in, central database.

2.2.3. STAGE 3: STUDY SELECTION PROCESS AND INCLUSION DECISION

The selection process used in this research was based on five steps (see Table 2.3). The selection process included a system that removed duplicate papers and unrelated topic papers from the initial cohort of papers identified in stage 2.

Table 2.3: Stage 3—Filtration Process

Filtration Step	Method	Assessment Criteria (Yes/No)
Step 1	Initial search using search key: PDF or full text of relevant studies in stage 2 **(N = 950)**	Select papers: PDF or full text
Step 2	Exclude duplicate papers collected from databases	Paper is duplicate
Step 3	Include papers on a title basis	Title = key term(s)
Step 4	Include papers based on abstracts	Abstract = key term(s)
Step 5	Include papers based on the introduction	Connection to DevOps, cloud/multi-cloud and IoT

The study filtration process was applied to the N = 950 papers to identify the relevant papers for the review. The first filtration step included database results that contained full-text papers and were available for download (The result was the selection of 938 papers). The second filtration step eliminated duplicate papers through title comparisons; which lead to a selection of 868 papers. The third filtration step identified papers based on their title relevance; which lead to the selection of 391 papers. The fourth filtration step removed papers based on the abstract's relevance to the research questions; which lead to the selection of 237 papers. The fifth filtration step included papers based on the review of their introduction section and relevance to the research. This resulted in the selection of 186 papers (see Table 2.4 and Figure 2.1).

Table 2.4: Stage 3—Filtration Process Results

Database	Stage 2 Results	Step 1	Step 2	Step 3	Step 4	Step 5	Stage 3 Results	Percentage
EEEXplore	**185**	183	176	101	71	66	**66**	35.48%
ACM Digital Library	**154**	151	131	74	31	22	**22**	11.83%
Elsevier	**164**	160	153	87	74	41	**41**	22.04%
SpringerLink	**60**	59	51	32	21	19	**19**	10.22%
Google Scholar	**387**	385	357	97	40	38	**38**	20.43%
N=	**950**	938	868	391	237	186	**186**	**100%**

The filtration process steps are illustrated in Figure 2.1, which shows the iterative steps that started with N = 950 studies and finished with N = 186 selected studies.

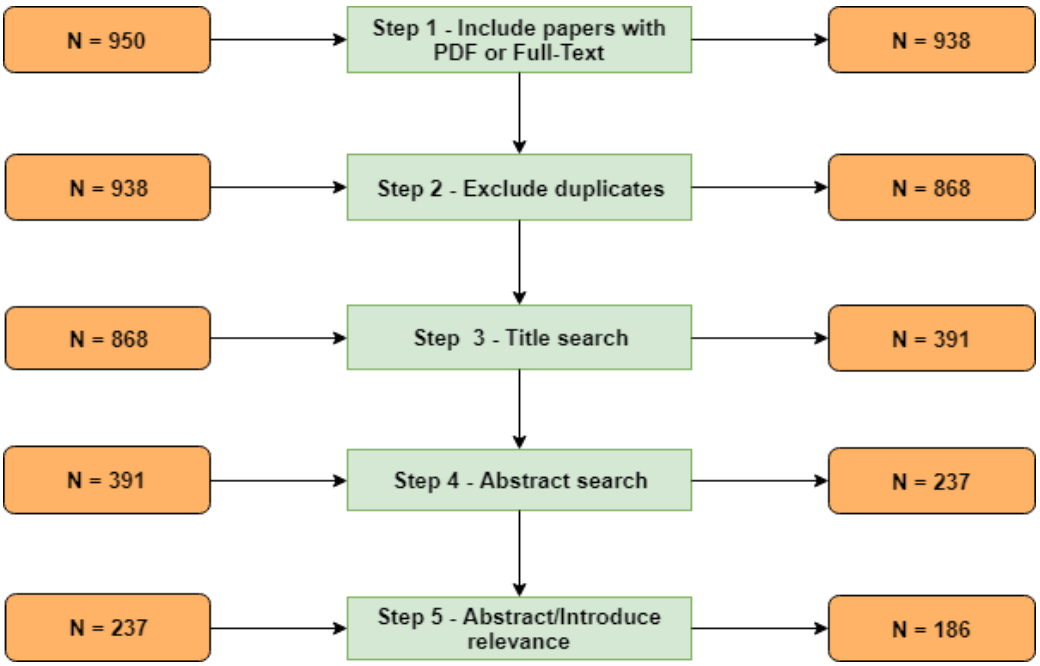


Figure 2.1: SLR Filtration Process

The filtration process results are plotted into a histogram graph relative to the percentage of papers selected from each database source (see Figure 2.2).

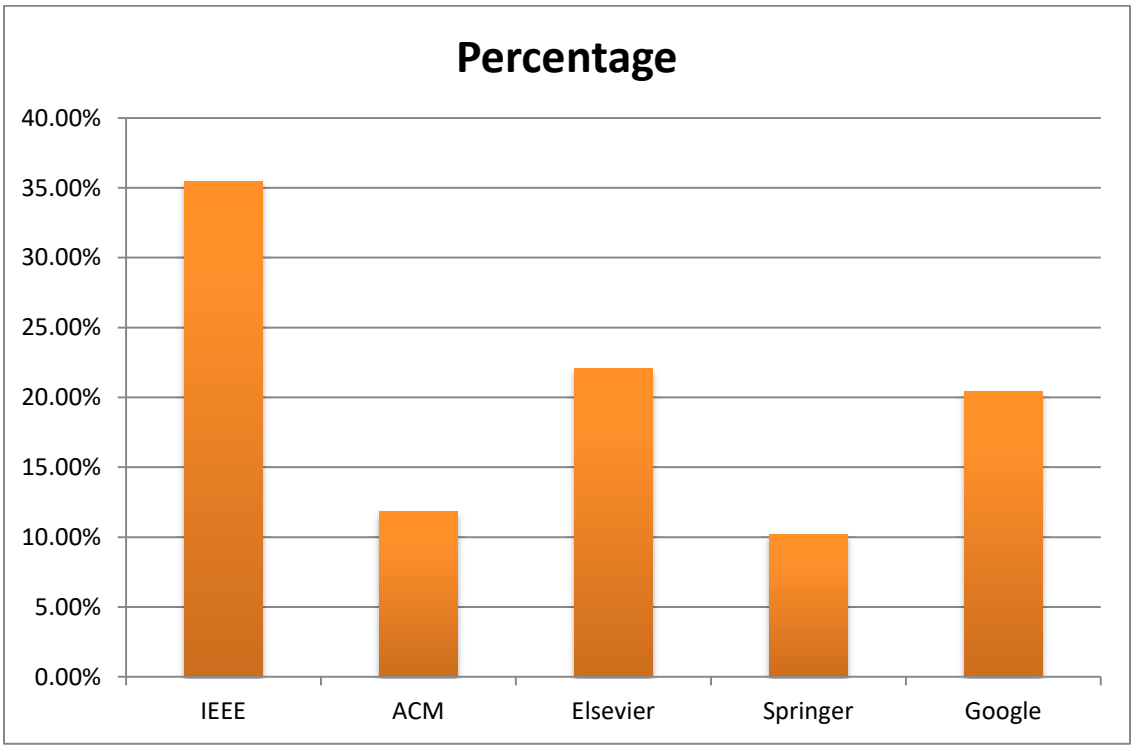


Figure 2.2: Stage 3—Filtration Results Graph

2.2.4. Stage 4: Final Selection Process and Quality Assessment

This stage uses five screening criteria (Dybå & Dingsøyr, 2008). A paper is selected if it satisfies all five quality assessment criteria (see Table 2.5). The criteria were applied to the N = 186 selected studies that passed stage 3, and 128 papers were selected for data extraction and synthesis.

Table 2.5: Stage 4—Quality Criteria

Item	Question	Answer: Yes = 1, No = 0
1	Is there a clear statement of the aims of the research?	0
2	Does the paper provide relevant data related to the research topics?	0
3	Is there a clear statement of findings?	0
4	How adequately has the research results been documented?	0
5	Is the study of value for research?	0

Finally, Table 2.6 presents the SLR selection process results at every stage. Stage 4 results are plotted in Figure 2.3 relative to its percentage.

Table 2.6: SLR Selection Process Results

Database	Stage 2	Stage 3	Stage 4	Percentage
IEEEXplore	**185**	**66**	56	43.75%
ACM Digital Library	**254**	**22**	13	10.16%
Elsevier Science Direct	**64**	**41**	35	27.34%
SpringerLink	**60**	**19**	10	7.81%
Google Scholar	**387**	**38**	14	10.94%
N=	**950**	**186**	**128**	**100%**

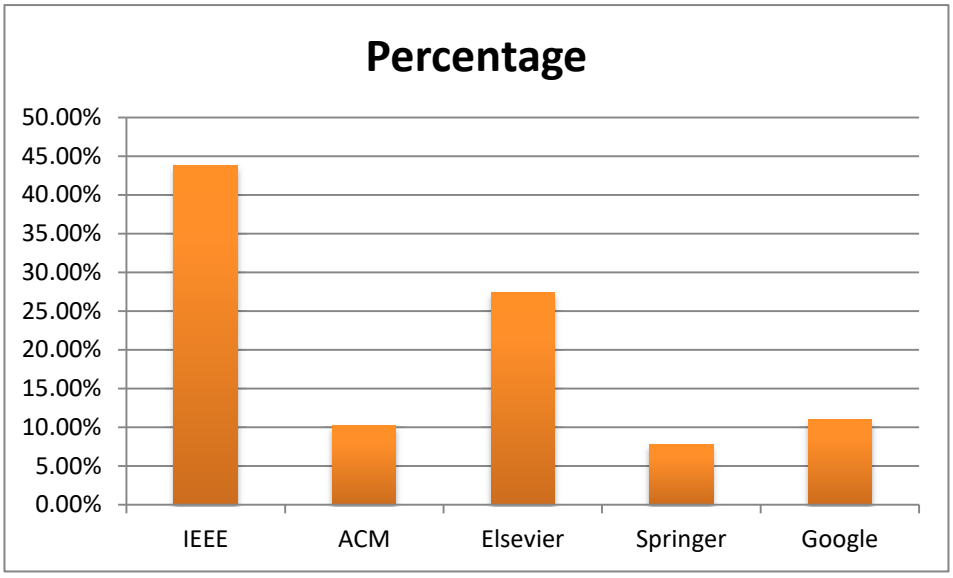


Figure 2.3: Stage 4—SLR Quality Results Graph

2.2.5. STAGE 5: DATA EXTRACTION AND DATA SYNTHESIS

During this stage, data are extracted from all 128 studies that passed stage 4. The data synthesis process summarises each paper from the selected studies. The purpose of this approach is to assemble relevant data from the selected studies to address the research problem in Section 1.2 and answer the research question (see Table 1.2). As shown in Table 2.7, the search categories of DevOps practices, continuous, automation, and tools have a high frequency or relevant sources compared with the other search categories.

The selected studies in this research are labelled $S_{[index]}$ from S_1 to $S_{128;}$ **index = [1 ... 128]** refers to a corresponding study listed in the Bibliography of this thesis. The selected studies were analysed, and the raw data were used to produce the results of this SLR (see Section 2.4).

Table 2.7: Stage 5—Selected Studies

Search Category	Frequency	Percentage	Sources
DevOps practices	20	15%	S_1, S_2, S_3, S_6, S_7, S_9, S_{10}, S_{14}, S_{19}, S_{20}, S_{22}, S_{23}, S_{24}, S_{26}, S_{27}, S_{30}, S_{43}, S_{12}, S_{107}, S_{125}
DevOps architecture	11	8%	S_1, S_2, S_3, S_7, S_9, S_{19}, S_{24}, S_{26}, S_{110}, S_{107}, S_{127}
DevOps concepts	34	26%	S_1, S_2, S_3, S_6, S_7, S_9, S_{10}, S_{12}, S_{14}, S_{15}, S_{19}, S_{20}, S_{22}, S_{23}, S_{24}, S_{26}, S_{27}, S_{30}, S_{43}, S_{107}, S_{110}, S_{116}, S_{125}, S_{126}, S_{127}, S_{130}, S_{132}, S_4, S_5 S_8, S_{13}, S_{21}, S_{25}, S_{28}
DevOps tools	17	13%	S_2, S_3, S_9, S_{14}, S_{15}, S_{19}, S_{20}, S_{22}, S_{23}, S_{24}, S_{30}, S_{43}, S_{107}, S_{110}, S_7, S_{116}, S_{127}
DevOps and IoT	3	2%	S_7, S_{11}, S_{107}
IoT architecture	14	11%	S_{32}, S_{34}, S_{37}, S_{39}, S_{40}, S_{41}, S_{47}, S_{53}, S_{56}, S_{57}, S_{60}, S_{64}, S_{69}, S_{79}
IoT devices and sensors	27	20%	S_{32}, S_{34}, S_{35}, S_{38}, S_{39}, S_{40}, S_{44}, S_{45}, S_{46}, S_{47}, S_{48}, S_{52}, S_{53}, S_{54}, S_{56}, S_{57}, S_{60}, S_{61}, S_{63}, S_{64}, S_{65}, S_{66}, S_{72}, S_{74}, S_{78}, S_7, S_{107}
IoT applications	15	11%	S_7, S_{35}, S_{36}, S_{50}, S_{60}, S_{64}, S_{71}, S_{75}, S_{77}, S_{79}, S_7, S_{93}, S_{103}, S_{107}, S_{124}
IoT security	13	10%	S_{32}, S_{36}, S_{37}, S_{39}, S_{48}, S_{50}, S_{52}, S_{69}, S_{71}, S_{77} , S_{51}, S_{76}, S_{67},
Cloud and IoT	17	13%	S_{58}, S_{62}, S_{67}, S_{68}, S_{70}, S_{83}, S_{84}, S_{85}, S_{86}, S_{90}, S_{91}, S_{92}, S_{93}, S_{102}, S_{103}, S_{11}, S_{107}
DevOps and cloud	7	5%	S_4, S_5, S_8, S_{13}, S_{21}, S_{25}, S_{28}
Multi-cloud	21	16%	S_{95}, S_{96}, S_{97}, S_{98}, S_{99}, S_{100}, S_{101}, S_{104}, S_{109}, S_{113}, S_{114}, S_{115}, S_{118}, S_{119}, S_{121}, S_{123}, S_{128}, S_{129}, S_{107}, S_{105} S_{112}
Multi-cloud and IoT	3	2%	S_{105}, S_{107}, S_{112}

The overall SLR selection progress overview from stage 1 to stage 5 is illustrated in Figure 2.4.

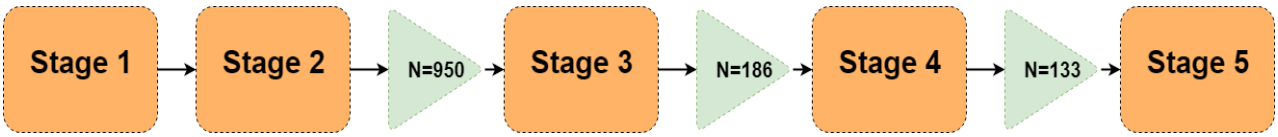


Figure 2.4: SLR Selection Progress Overview

2.3. SLR DATA REVIEW AND ANALYSIS

This section presents the analysis and review of the data collected from the selected 128 studies. The selected studies provide a rich and rigorous information platform to understand the contexts of DevOps, cloud/multi-cloud, and IoT, as well as the relationships between them. The analysis conducted in this section provides essential material to answer the research sub-question (RQ_1) (see Table 1.2). The collected data will be used as a platform to investigate the DSR method adopted by Gregor and Hevner (2013) and Peffers et al. (2007) in Chapter 4.

2.3.1. DEVOPS AND MULTI-CLOUD

DevOps has emerged as a paradigm that aims to improve collaboration and communication between Dev and Ops (Pratibha & Khan, 2018; Virmani 2015). DevOps provides a set of well-known practices (Ghantous & Gill 2017; McCarthy et al. 2015) to enable the automation of application deployment for timely release and delivery (Artač et al. 2016; Samarawickrama & Perera 2017). The DevOps approach enables automation for the application development chain (Diel, Marczak & Cruzes 2016; McCarthy et al. 2015; Nguyen & Gendreau 2014). Moreover, in a test-driven development environment, DevOps tools augment release frequency and production quality (De Bayser, Azevedo & Cerqueira 2015; Lwakatare, Kuvaja & Oivo 2016a). DevOps offers a significantly broader perspective to application deployment to the cloud (Rajkumar et al. 2016; Wettinger, Andrikopoulos & Leymann 2015) through its concepts, known practices, and a wide range of tools. Hence, the DevOps approach can be useful for the deployment of IoT applications to the multi-cloud environment. IoT application deployment to the cloud/multi-cloud requires a general DRA (Chondamrongkul & Temdee 2013; Srirama, Iurii & Viil 2016; Tao et al. 2018). The suggested reference architecture can be instantiated to fit the context of the development environment (Ghantous & Gill 2018). There are several challenges of IoT application deployment to the multi-cloud, including security, connectivity, big data exchange, automated data collection, real-time access, monitoring, scalability and heterogeneous IoT devices (Alkhalil & Ramadan 2017; Bass et al. 2015; Farahzadi et al. 2018; Ghantous & Gill 2018; Ray 2016; Stergiou et al. 2018).

DevOps appears to provide the multi-cloud environment with the methods and tools to enable automation and CI (Ghantous & Gill 2018; Singh et al. 2015; Soni 2015). For example, an open-source framework such as TOSCA (Wettinger, Andrikopoulos & Leymann 2015) enables automation across the deployment pipeline for application deployment to the cloud. In essence, the multi-cloud could benefit from DevOps concepts and practices (Alam, Sopena & El Saddik 2015; Ghantous & Gill 2018; Shekhar & Gokhale 2017). A framework model could combine DevOps and multi-cloud contexts in the general reference architecture. These contexts could serve as a general platform of characteristics defined from commonly used terminologies (Berger, Häckel & Häfner 2019). These characteristics could be used to create a general design model that could be instantiated to fit the purpose of the development context.

2.3.2. DEVOPS AND IOT

There is growing interest in the IoT community in adopting DevOps (Moore et al. 2016; Syed & Fernandez 2016). IoT, supported by cloud computing (Botta et al. 2016; Douzis et al. 2018; Ray 2016), aims to achieve interoperability and fast data exchange (Ferry et al. 2018; Yonezawa et al.). Cloud offers PaaS as a virtual development platform for IoT application developers. It also provides a back-end solution to manage the vast data streams of IoT application data through IaaS and SaaS (Alam, Sopena & El Saddik 2015; Cavalcante et al. 2016; Massonet et al. 2017). Similar to DevOps, IoT presents some challenges (Alkhalil & Ramadan 2017; Hefnawy, Bouras & Cherifi 2016; Khakimov et al. 2017) because of its large number of interconnected and heterogeneous sensors and devices (Gomes, da Rosa Righi & da Costa 2014; Su et al. 2014; Ungurean, Gaitan & Gaitan 2014).

The value of IoT for enterprises resides in the fast delivery of IoT applications. However, similar to DevOps, IoT applications are complex and involve real-time connectivity between IoT devices. IoT also requires effective seamless integration with other systems such as the cloud (Botta et al. 2016; Fan et al. 2012; Farahzadi et al. 2018). Some studies have proposed solutions to big data generated by IoT sensors such as SLM (Kang & Choo 2018) and SOXFire (Yonezawa et al. 2016), which provide a multi-community citywide sensor network for sharing big sensor data. Other studies have focused on organising and mapping future directions for better IoT infrastructure (Khakimov et al. 2017; Muhendra, Rinaldi & Budiman 2017). It has been suggested that cloud infrastructure is suitable for IoT applications (Cavalcante et al. 2016; Li et al. 2017). A reliable IoT gateway (Kang & Choo 2018) can be used to include new devices outside the network using wireless or Bluetooth connection types (Yasaki, Ito & Nimura 2015). Another study offered a management system to quickly connect IoT devices and their applications (EasyConnect) (Lin et al. 2015; Luo & Sun 2015). An end-to-end security architecture for IoT networks on the Federal Cloud (Massonet et al. 2017) is an example of how IoT transmissions can be secured with the cloud. Another way of securing sensor communication is a two-phase authentication protocol (Porambage et al. 2014). Monitoring internet traffic between the cloud and IoT applications to sensors is necessary to ensure the performance and efficiency of the IoT application (Hefnawy, Bouras & Cherifi 2016; Yonezawa et al. 2016). AnyControl (Wang et al. 2015) provides a home appliance monitoring system that observes the live data exchange between the user application and sensors.

The above information suggests that the cloud may be a practical platform for IoT application deployment. The previous section observed that DevOps and cloud/multi-cloud contexts might be combined into a general framework. Hence, IoT application deployment may benefit from the DevOps approach, which offers automation, CI, and real-time monitoring. However, IoT may pose challenges for the suggested framework, such as IoT application provisioning, IoT data storage, IoT security, and IoT connectivity.

2.3.3. IoT Sensors and IoT Applications

IoT provides human–device interactions through middleware in a secure environment (Kishore Ramakrishnan, Preuveneers & Berbers 2014; Su et al. 2014; Yaqoob et al. 2017). Middleware are IoT applications that use existing middleware service systems to provide connectivity with devices (Ngu et al. 2016). For example, Hydra provides a web service for incorporating heterogeneous physical devices into applications. GSN provides a web service that delivers a stable platform for flexible integration, sharing and deployment of heterogeneous IoT devices (Gomes, da Rosa Righi & da Costa 2014). Models such as SysADL are explicitly designed to preserve a system-oriented (Leite, Batista & Oquendo 2017), which is an ADL based on SysML architecture. Another model—Wi-Fi mesh infrastructure—is designed for IoT applications (Muhendra, Rinaldi & Budiman 2017). A new architecture model called 3G-PLC (Yaqoob et al. 2017) combines two complex communication networks: 3G and PLC. Such models may need a reliable gateway to control information traffic using a high-level design methodology (Chen, Lin & Guo 2017). A study was conducted to construct a self-configurable IoT gateway (Kang & Choo 2018). The idea was to develop an algorithm that enables an IoT gateway to automatically configure itself and perform two essential functions (Rao & Shorey 2017). Ultimately, the objective was to improve human interactions with IoT devices through middleware—whether manually or automated—using an IoT sensor controller (Alowaidi et al. 2017).

Sensor nodes collect data through WSN, Bluetooth, MQTT, etc., which enables the convergence of the IPv6 network and low-power wireless (Luo & Sun 2015). Such systems can be used on Linux running on Raspberry Pi (Newmarch 2016). To maintain IoT sensors, a RESTful WS can be used to enable the management of devices (Sheng et al. 2015); the service can be scaled and deployed on a cloud platform. The expected lifetime of low-powered IoT devices should be taken into consideration (Morin et al. 2017). Another reference architecture example is AllJoyn (Khakimov et al. 2017), which uses HTTP and MQTT protocols for communications through the gateway.

Many practical architecture models have already been deployed on various sensor networks. First, in EasyConnect (Lin et al. 2015), devices are attached to EasyConnect using a friendly GUI that is compatible with most smart devices. Second, smart city multipurpose architecture is based on the monitoring of environmental variables in urban areas (Gómez et al. 2017). Third, SOXFire (Mongan et al. 2017) supports physical access to IoT devices. For this purpose, SLM (Hefnawy, Bouras & Cherifi 2016) was suggested to analyse, plan, design, build and maintain IoT-enabled smart city service systems that could use a date sharing strategy for IoT-data (Adda, & Saad 2014).

IoT is increasingly receiving attention in the IT industry (Gutiérrez-Madroñal, Medina-Bulo & Domínguez-Jiménez 2018). It appears that a practical solution to solve the issue of substantial quantities of data is to use the concept of automation in all configuration sectors: development,

testing, data management, deployment, device integration, and connection. The configuration appears to be data-driven (Kolios et al. 2016). Actions or events are generated and analysed based on signals received from sensors through gateways (Yasaki, Ito & Nimura 2015) and are handled by IoT applications. Hence, it is essential to address the issue of connectivity between IoT applications and IoT sensors because the performance of IoT applications is dependent on the effectiveness of its connectivity with IoT devices. The performance of IoT applications is deduced by measuring their latencies using communication protocols such as MQTT, RSSI, NFC, Wi-Fi, and mobile (Babovic 2016). The performance of IoT applications requires constant improvements to handle the increasing amount of IoT data, which could reach 30 exabytes per month by 2020 (Sen 2016). IoT data are stored either in traditional SQL tables or by using a NoSQL database such as MongoDB (Alkhalil & Ramadan 2017, Rautmare & Bhalerao 2016).

2.3.4. IoT Monitoring and IoT Security

Given the continued growth of IoT networks, it is necessary to enhance the performance of IoT applications using real-time monitoring, analysis, and detection of critical events in the COT (Lee & Hughes 2010). It is challenging to keep a whole system of sensors operating effectively. Therefore, it is proposed that cloud services should be treated as sensor-generating log events during the application run time (Lee & Hughes 2010). The exchanged data with physical sensors can be maintained using a GUI such as LabVIEW (Russell, Goubran & Kwamena 2015). AnyControl (Wang et al. 2015) is used to monitor and control home-based devices by treating sensor data as event-driven objects. The action of ADLs in a smart environment is monitored at run time, which enables decision-making for generated events from sensors' RFID data streams (Kodeswaran et al. 2016; Mongan et al. 2017). Another model—TISH—has been proposed to automatically monitor and manage RFID data streams (Wu et al. 2013). This model was built on the flow of P2P objects between IoT devices.

A secure end-to-end architecture is required for all application–device connections based on the communication of various platforms (Mathur et al. 2017; Olivier et al. 2015). End-to-end security for smart IoT applications in a centralised WSN requires a useful system model that is based on a two-phase authentication solution (Porambage et al. 2014). Thus, IoT security and IoT connectivity between applications and devices are primary challenges that need to be addressed when implementing IoT applications.

2.3.5. Cloud Computing and IoT

Cloud computing is an innovative, internet-based service that provides an abstraction to infrastructures. The cloud enables ubiquitous, on-demand access to shared resources (Cloud API, configurations, and services). Cloud computing provides three service models (Jula, Sundararajan & Othman 2014): PaaS, SaaS, and IaaS.

The adoption of cloud computing for industry and research has exponentially increased due to cloud efficiency and advantages (Avram 2014), which includes: 1) immediate access to virtualised infrastructure and software; 2) enabling organisations to efficiently scale their services and applications on an available cloud platform; 3) lower costs because clouds provide virtual infrastructure and services; and 4) higher-level abstraction for existing software applications. Developers can benefit from abstract cloud computing advantages to deploy software applications to virtual servers that enable auto-scaling.

Cloud benefits have five essential characteristics (Shahzad 2014): 1) resource pooling: cloud provides centralised access to services; 2) rapid elasticity: a fast option for users to scale their applications given the nature of the cloud as a software service; 3) measured service: resources are automatically controlled, managed and monitored by cloud providers; 4) on-demand self-service: clouds enable users to provide their environment and infrastructure; 5) broad network access: the cloud is globally available to users, which makes it typically suitable for agile software development.

The adoption of cloud computing is threatened by some challenges (John et al. 2015; Shahzad 2014): 1) security and privacy: the cloud's ability to provide secure access for individual users is often questioned at an enterprise level; 2) global access: the full potential of the cloud is yet to be achieved by reaching global access; 3) reliability: on-demand 24/7 cloud availability has become critical, and cloud providers are introducing futuristic plans to counter failures, outages, and lock-in; 4) interoperability and portability: necessary and innovative standards are increasingly required to standardise information sharing.

Cloud computing is a promising solution that helps to overcome several challenges faced by the IoT paradigm, which is mainly characterised by heterogeneous physical devices with technological constraints. Cloud computing requires that IoT applications be enhanced concerning computational resources, scalability, and performance (Cavalcante et al. 2015). The interplay and cooperation between fog (edges: support IoT devices) and the core (cloud) can be characterised using three-tier COT architecture (Li et al. 2017) to establish integration between code and devices (Botta et al. 2016; Zabasta et al. 2020). Which highlights the issue of connectivity between IoT devices and IoT applications deployed to the cloud.

Several factors should be considered in the design and implementation of IoT applications. One of the most challenging problems is the heterogeneity of different objects (devices). However, this can be addressed by deploying a suitable 'middleware' that sits between devices and applications and acts as a communication platform (Farahzadi et al. 2018). Notably, middleware is designed based on different architectures, which can be classified as follows: distributed, component-based, service-based, node-based, and centralised and client-server (Farahzadi et al. 2018). There are some existing IoT middlewares, including Hydra, GSN, Google Fit, Xively, Calvin, and UPnP (Farahzadi et al. 2018; Ngu et al. 2016). The use of middleware presents a

variety of challenges that threaten the stability of IoT application interactions with IoT devices (Ray, 2016). These issues include: 1) near-real-time prioritising (resource/service allocation issues; 2) proper resource discovery implementation because there is no guarantee of the continuity of services/resources on a COT heterogeneous network; 3) user security and privacy enhancement; and 4) finding an optimal place for data analysis. There is also an issue of secure integration of applications and devices. Integration is composed of several layers (Rautmare & Bhalerao 2016): 1) sensing and smart devices; 2) gateway/connectivity nodes; and 3) remote cloud-based processing. Another connectivity protocol is mobile, which invokes the services from the multi-cloud using mobile connectivity configured on a Raspberry Pi (AlOtaibi, Lo'ai & Jararweh 2016; Farahzadi et al. 2018).

2.3.6. MULTI-CLOUD, DEVOPS AND IOT

Multiple clouds or multi-cloud is the integration of multiple cloud computing services in a single heterogeneous architecture. Organisations, developers, and researchers can benefit from open-source cloud platforms because they encourage the use of the multi-cloud through broader user access and availability to the same or different applications deployed to multi-cloud platforms (Domaschka et al. 2015; Zhang at al. 2020). Multi-cloud systems aim to enable integration and compatibility of various cloud services following high-level QoS for applications (Ferry et al. 2013; Willnecker & Krcmar 2018). However, such platforms are limited to managing single-cloud applications and often do not promote switching to other clouds because an application's code is hardwired to its cloud API (Kritikos & Plexousakis 2015).

The major obstacle for adopting multi-cloud distributed deployment is vendor lock-in, which prevents harmonious deployment and database integration for the software application (Chondamrongkul & Temdee 2013; Kritikos & Plexousakis 2015; Yasrab & Gu 2016). Vendor lock-in may occur in two cases: when a cloud from the multi-cloud cohort hosts the deployment configuration and when a cloud from the multi-cloud cohort hosts the database.

Several studies and frameworks have introduced innovative ideas and architectures to achieve heterogeneous architecture (Yang et al. 2016) for continuous deployment to the multi-cloud. For instance, CYCLONE (Slawik et al. 2017) is a software stack that focuses on the areas of application deployment, management, and security and user authentication on the multi-cloud. Another model, CloudMF (Ferry et al. 2018), is an object-oriented domain-specific model tailored for IoT applications. The deployment process to the multi-cloud can follow specific migration patterns (Jamshidi et al. 2015) such as multi-cloud refactoring and multi-cloud rebinding. The multi-cloud appears to support application deployment by offering required technology components and services (Syed & Fernandez 2016). The dynamic Data-Driven Cloud and Edge Systems (D3CES) approach enables real-time IoT data collection and provides feedback that enables effective decision-making to deploy IoT applications to the cloud (Jamshidi et al. 2015; Shekhar & Gokhale 2017). As the examples show, IoT can benefit from

cloud/multi-cloud services and techniques that enable portability and interoperability (Di Martino & Esposito 2016; Iqbal et al. 2010). Moreover, DevOps concepts (automation, CI, real-time monitoring) can support IoT application deployment (Ghantous & Gill, 2018) in a secure environment (Nguyen et al. 2020; Rahman & Williams 2016).

2.3.7. DATA ANALYSIS SUMMARY

This section summarises the SLR data analysis conducted in this section by converging the ideas and observations to tailor the set of requirements necessary for integrating DevOps, cloud/multi-cloud, and IoT. The data analysis summary is mapped in Table 2.8, which highlights the requirements and descriptions of the research contexts and their relationships.

Table 2.8: SLR Analysis Summary

Contexts	Requirement	Description
DevOps	Understand what is DevOps	Definitions of DevOps
DevOps	Outline DevOps concepts categories	DevOps concepts list
DevOps	Outline DevOps practices	Standard DevOps practices
DevOps	Catalogue and categorise DevOps tools	DevOps tools and their uses
Cloud	Understand cloud architecture	Cloud architecture standards
Cloud	Outline and discuss cloud services	Cloud services
Cloud–DevOps	Understand cloud–DevOps relationship and tools integration	Cloud–DevOps tools integration
Cloud–DevOps	DevOps adoption on the cloud	Integrating DevOps and cloud
IoT	Understand IoT connectivity	IoT network types
IoT	Understanding IoT devices and data collection	IoT-devices setup and configuration
IoT Apps	Understand IoT apps connection with IoT devices	IoT (app–device) interaction
Cloud–IoT	Understand cloud support for IoT	IoT apps deployment to the cloud
DevOps–Cloud–IoT	Enable automation and CI on the cloud for IoT	IoT apps automated deployment and automated development
Multi-cloud	Interconnect multiple clouds	Broader user access, more services
Multi-cloud	Avoid vendor lock-in	When a cloud from the multi-cloud set hosts the deployment configuration
Multi-cloud–DevOps–IoT	Enable IoT apps automated deployment and fast delivery on multiple platforms	Multi-cloud provides services and virtual infrastructure for IoT apps; DevOps provides automation, CI and real-time monitoring concepts for IoT apps
Multi-cloud–DevOps–IoT	Avoid vendor lock-in for IoT data collective	When a cloud from the multi-cloud set hosts the database
Multi-cloud–DevOps–IoT	Understand deployment parameters and IoT app connections with Raspberry Pi that control the IoT sensors	Determine the connectivity protocol suitable for IoT applications deployed to the cloud/multi-cloud.

2.4. SLR RESULTS

This section of the SLR analyses the data in Section 2.3 to further describe DevOps, cloud (single and multiple), and IoT relationships. Consequently, this section will address the research sub-question RQ_1 outlined in Table 1.2. Answering RQ_1 is the first step towards achieving the research objectives shown in Figure 1.6. The research aims are based on a set of ideas outlined in Table 1.1. This section uses the SLR analysis results and outlines what is known about DevOps. It then discusses and outlines the benefits and challenges of DevOps adoption for IoT application deployment to the cloud (single and multiple). The research gap is then outlined. The construction of the DRA framework is an attempt to produce an output that provides a solution to the research gap and answers the research question.

2.4.1. WHAT IS KNOWN ABOUT DEVOPS?

This section discusses what is known about DevOps. DevOps concepts, practices, and tools are rediscovered and refined based on the thorough analysis conducted in the previous section of this SLR (see Section 2.3).

- **DevOps Concepts**

DevOps aims to improve collaboration and communication between traditionally separated Dev and Ops (Ghantous & Gill 2017; Jabbari et al. 2016) teams through the automation of software deployment and fast product delivery (De Bayser, Azevedo & Cerqueira 2015; Schaefer, Reichenbach & Fey 2013; Wettinger, Breitenbücher & Leymann 2014). This research identified a set of 10 key concepts using the well-known generic conceptual element classes (e.g., people, event, activity, thing) that underpin the DevOps approach (see Table 2.9).

DevOps is perceived as a technology-centric venture. The results indicate that communication and collaboration (11%), along with continuous delivery (9%) and continuous deployment (10%), are essential concepts based on their frequency of appearance in the selected studies (see Table 2.9). Notably, the SLR indicates that CI (15%) and automation (15%) are vital DevOps concepts according to their frequency of appearance in the selected studies $S_{[index]}$.

DevOps teams are composed of Devs and Ops that need to effectively communicate and collaborate to improve the delivery of software and hardware infrastructure (infrastructure as a code) in production (Ghantous & Gill 2017; Rajkumar et al. 2016; Wettinger, Andrikopoulos & Leymann 2015). Thus, task synchronisation is essential to deliver software that satisfies stakeholders' requirements (Bai et al. 2018; Diel, Marczak & Cruzes 2016; Qumer, Henderson-Sellers & McBride 2007; Wahaballa et al. 2015). Therefore, continuous code management (5%) and continuous planning (7%) are essential in the adoption of DevOps. It can be concluded that all concepts require automation and CI of DevOps tools to establish a successful agile software development environment using DevOps. Hence, automation and CI are among the most cited

concepts in Table 2.9. DevOps concepts are labelled $\mathbf{C_{[index]}}$ and referenced in the first column in Table 2.9. These concepts are based on information about gathered from the selected sources (S_{index}) located in the Bibliography. DevOps concepts are the building blocks to construct a reference architecture based on the proposed DRA framework characteristics generic terminologies. The DRA general terminologies (Berger, Häckel & Häfner 2019; Nickerson, Varshney & Muntermann 2013) are not fixed to a particular instance and can be applied to any development context adopting the DevOps approach.

Table 2.9: DevOps Concepts

Element	DevOps Concepts	Frequency	Percentage	Sources
People **C_1**	Communication and collaboration	26	11%	S_{31}, S_{111}, S_{117}, S_{131}, S_1, S_2, S_3, S_6, S_9, S_{10}, S_{12}, S_{14}, S_{15}, S_{19}, S_{20}, S_{22}, S_{23}, S_{24}, S_{26}, S_{27}, S_{43}, S_{116}, S_{125}, S_{107}, S_7
Event **C_2**	Continuous deployment	23	10%	S_{31}, S_{111}, S_{117}, S_{131}, S_6, S_9, S_{10}, S_{14}, S_{15}, S_{19}, S_{20}, S_{22}, S_{23}, S_{30}, S_{43}, S_{110}, S_{125}, S_{127}, S_{130}, S_{107}, S_{132}, S_8, S_{28}
Event **C_3**	Continuous delivery	22	9%	S_{31}, S_{111}, S_{117}, S_{131}, S_6, S_{10}, S_{14}, S_{19}, S_6, S_{20}, S_{22}, S_{23}, S_{26}, S_{27}, S_{30}, S_{43}, S_{110}, S_{126}, , S_{6127}, S_{130} , S_{13}
Thing **C_4**	Continuous development	26	11%	S_{31}, S_{111}, S_{117}, S_{131}, S_6, S_{10}, S_{14}, S_{19}, S_6, S_{20}, S_{22}, S_{23}, S_{26}, S_{27}, S_{30}, S_{43}, S_{110}, S_{126}, , S_{127}, S_{130}, S_{132}, S_{13}, S_{21}
Thing **C_5**	Continuous planning	18	7%	S_{31}, S_{117}, S_{131}, S_2, S_6, S_9, S_{10}, S_{12}, S_{14}, S_{19}, S_{20}, S_{23}, S_{26}, S_{27}, S_{43}, S_{116}, S_{130}, S_{128}
Activity **C_6**	Continuous integration	35	15%	S_{31}, S_{111}, S_{117}, S_{131}, S_1, S_2, S_3, S_6, S_9, S_{10}, S_{12}, S_{14}, S_{15}, S_{19}, S_{20}, S_{22}, S_{23}, S_{24}, S_{26}, S_{27}, S_{43}, S_{116}, S_{125}, S_{107}, S_{132}, S_7, S_4, S_5, S_8, S_{13}, S_{21}, S_{25}, S_{28}, S_{11}, S_7
Thing **C_7**	Continuous code management	13	5%	S_{31}, S_{117}, S_6, S_9, S_{10}, S_{19}, S_{20}, S_{22}, S_{23}, S_{24}, S_{27}, S_{43}, S_{116}
Event **C_8**	Continuous quality assurance and review	19	8%	S_{31}, S_{117}, S_{131}, S_2, S_6, S_9, S_{14}, S_{15}, S_{19}, S_{20}, S_{22}, S_{23}, S_{26}, S_{27}, S_{43}, S_{107}, S_{125}, S_{130}, S_{128}
Activity **C_9**	Automation	35	15%	S_{31}, S_{111}, S_{117}, S_{131}, S_1, S_2, S_3, S_6, S_9, S_{10}, S_{12}, S_{14}, S_{15}, S_{19}, S_{20}, S_{22}, S_{23}, S_{24}, S_{26}, S_{27}, S_{43}, S_{116}, S_{125}, S_{107}, S_{128}, S_7, S_4, S_5, S_8, S_{13}, S_{21}, S_{25}, S_{28}, S_{11}, S_7
Event **C_{10}**	Real-time monitoring	22	9%	S_1, S_2, S_3, S_6, S_9, S_{10}, S_{12}, S_{14}, S_{15}, S_{19}, S_{20}, S_{22}, S_{23}, S_{24}, S_{26}, S_{27}, S_{43}, S_{116}, S_{125}, S_{107}, ,S_{128}, S_7
	Total Frequency Connections	**240**	**100%**	**Sources N = 128 Studies**

- **DevOps Practices**

This study identified a set of 20 DevOps practices (McCarthy et al. 2015; Soni 2015) that support the DevOps concepts (see Table 2.9). Retro-QA represents an agile retrospective improvement and quality assurance process (Rajkumar et al. 2016). Some practices focus on Dev, and others focus on Ops. DevOps practices can be considered concrete guidelines that implement DevOps concepts (Jabbari et al. 2016). The DevOps practices (**PC $_{[index]}$**) referenced in Table 2.10 are derived from the raw data synthesised from 128 selected studies for this SLR and are analysed in Section 2.3.

Table 2.10: DevOps Practices

Practices	Definitions
PC_1	Create development sandboxes for minimum code build
PC_2	Automate sandboxes deployment through the pipeline
PC_3	Provide a continuous collaboration system in real-time using DevOps tools
PC_4	Automate testing sandboxes to run in conjunction with development sandboxes
PC_5	Perform Retro-QA tests on build sandboxes
PC_6	Reduce the variance between development and production to a minimum
PC_7	Use DevOps tools to automate deployment, build, testing, synchronisation of code
PC_8	Developers must be able to access the IT operations incident reports and synchronize with operations to improve project supportability
PC_9	Testing reports (auto-generated by a DevOps tool, Sandbox test units, quality testing) must be shared between Developers and Operations
PC_{10}	Monitoring logs (generated by DevOps tools or Retro-QA monitoring logs) must be shared between Development and Operations
PC_{11}	DevOps team synchronises critical services such as transactions, performance, uptime, deployment schedule, run-time costs, version control, and project scope
PC_{12}	DevOps team uses central repository for build, deployment, testing, versioning, synchronisation, CI and continuous deployment
PC_{13}	Application release deployments must be fully automated across the pipeline
PC_{14}	DevOps team must provide overall visibility into project scope and release timing to DevOps team
PC_{15}	DevOps team must provide self-service and resources management of platform (cloud, hybrid, server) to DevOps team
PC_{16}	DevOps team must be able to increase release frequency to satisfy business demand
PC_{17}	DevOps team must have a clear insight into the SD project to ensure business reliability and application performance on a cross-platform infrastructure environment
PC_{18}	DevOps team must provide safe deployment parameters to avoid excessive workload on the infrastructure
PC_{19}	DevOps team must be able to update system iterations or sandboxes based on monitoring reports and defect logs
PC_{20}	DevOps team optimise SD project based on behaviour-driven development and Retro-QA results of a process

Table 2.11 shows the DevOps practices categories and relationships to DevOps concepts. It is essential to categorise the 20 DevOps practices and assemble them according to their relationship to DevOps concepts (see Table 2.9). Therefore, it can be concluded that DevOps practices aim to provide developers and operations with software development standards that enable the concepts shown in Table 2.8. The practices defined in Table 2.10 are a set of well-known guidelines collected from an SLR data analysis of the selected studies. It can be concluded that all practices require automation and CI to enhance the developers' experience. In the later stage of the SLR results, the review indicates that automation and CI are supported by various DevOps tools. This realisation indicates that the DevOps approach can effectively integrate with the cloud because the cloud (single and multiple) provides a wide range of services and infrastructure as code at the abstract level.

Table 2.11: DevOps Practices Categories

Practices	Category	DevOps Concepts
PC_1	Development, build	C_4; C_6; C_7; C_9
PC_2	Deployment	C_2; C_6; C_7; C_9
PC_3	Communication/collaboration	C_1; C_6; C_9
PC_4	Testing	C_4; C_6; C_9
PC_5	Retro-QA, agile planning	C_4; C_4; C_6; C_9
PC_6	Manage, synchronise	C_6; C_7; C_9
PC_7	Development, release, testing, build, synchronise, deploy, automation	C_2; C_3; C_4; C_6; C_7; C_9
PC_8	Manage, synchronise	C_6; C_7; C_9
PC_9	Testing, Retro-QA, automation	C_1; C_4; C_5; C_6; C_8; C_9
PC_{10}	Monitoring, Retro-QA, automation	C_1; C_4; C_5; C_6; C_8; C_9; C_{10}
PC_{11}	Release, manage, deploy, CI, automation, synchronise	C_1; C_2; C_3; C_4; C_5; C_6; C_7; C_9
PC_{12}	Release, manage, synchronise	C_1; C_2; C_3; C_4; C_6; C_7; C_9
PC_{13}	Release, manage, automation	C_1; C_2; C_3; C_4; C_6; C_7; C_9
PC_{14}	Manage, release, clear insight	C_1; C_2; C_3; C_4; C_6; C_7; C_9; C_{10}
PC_{15}	Manage, cloud services, resources management	C_1; C_6; C_7; C_9
PC_{16}	Release, automation	C_2; C_3; C_6; C_7
PC_{17}	Manage, clear insight, cloud platform	C_1; C_6; C_7; C_9; C_{10}
PC_{18}	Deploy, synchronise	C_2; C_3; C_6; C_7; C_9
PC_{19}	Development, testing, monitoring	C_2; C_3; C_4; C_6; C_7; C_9; C_{10}
PC_{20}	Development, Retro-QA, testing	C_2; C_3; C_4; C_6; C_7; C_9

- **DevOps Tools**

DevOps tools and practices are core to the DevOps architecture (Ghantous & Gill 2017; Domaschka et al. 2015; Wettinger et al. 2016). DevOps tools are organised according to the tool category and its connection to DevOps practices (see Table 2.10 and Table 2.11). The SLR presents 12 categories of DevOps tools (see Table 2.12). Continuous integration, deployment, IaaS/PaaS, and configuration, and provisioning are the most cited categories of tools based on

their appearance in the selected studies. These categories are vital for constructing a reference architecture based on the proposed DevOps framework generic characteristics.

DevOps tools are mapped in Tables 2.12–2.22 according to their categories and relationship to DevOps practices. As stated in the previous section, DevOps is technology-centric; in fact, it is tools-centric. A software development pipeline can be configured from a set of tools (see Tables 2.12–2.22). The pipeline's most crucial element is the automation of the development lifecycle (code synchronisation, build, test, deploy, and deliver). The automation is achieved using a robust CI of the tools involved in the pipeline. Hence, cloud and DevOps relationships are intertwined because cloud services and platforms enable end-to-end automation and provisioning of software applications (Singh et al. 2015; Soni 2015). Therefore, the effect of the DevOps culture on cloud delivery is evident (Rajkumar et al. 2016) because it enables development and deployment automation and supports the auto-scaling of applications (Cukier 2013).

Table 2.12: DevOps Tools Categories

Purpose	Tool Categories	DevOps Practices
Asset and change management	Source control management	PC_6; PC_7; PC_8; PC_{11}; PC_{12}; PC_{18}
Integration	Continuous integration	$PC_1 \rightarrow PC_{20}$
Deployment	Continuous deployment/delivery	PC_2; PC_7; PC_{11}; PC_{12}; PC_{13}; PC_{16}; PC_{18}
Platform	IaaS/PaaS	PC_{15}; PC_{17}
Control	Monitoring	PC_{10}; PC_{14}; PC_{17}; PC_{19}
Platform	Database management	PC_6; PC_8; PC_{11}; PC_{12}; PC_{13}; PC_{14}; PC_{15}; PC_{17}
Deployment	Containerisation	PC_2; PC_7; PC_{11}; PC_{12}; PC_{13}; PC_{14}; PC_{16}; PC_{18}
Audit	Logging/security	PC_{10}; PC_{14}; PC_{15}; PC_{17}; PC_{19}
Development	Build	PC_1; PC_7; PC_{19}; PC_{20}
Quality	Testing	PC_4; PC_5; PC_7; PC_9; PC_{10}; PC_{19}; PC_{20}
Team	Collaboration/communication	PC_3; PC_5; PC_6; PC_8; PC_{11}; PC_{12}; PC_{14}; PC_{17}

Table 2.13: Source Control Management

Source Control Management	
Tools	**Features**
Github https://github.com	- Github is a web-based Git (private and public accounts) repository - Github provides team collaboration - Provide logs containing (Commit history, tracking labels, pull requests, code review comments, email notifications, task lists)
Bitbucket https://bitbucket.org	- Similar features to Github - Offers both free public and private commercial accounts

Table 2.14: Continuous Integration

Continuous Integration	
Tools	**Features**
Codeship https://codeship.com	- Use Docker abilities to automate development and deployment - Enable developers to create their test units - Provide team notifications with code changes and test results - Deploy and run code in parallel simultaneously with tests - Integrate many programming languages (Java, Ruby, Python, PHP, GO) - Integrate many platforms (Heroku, AWS, GAE)
Travis CI https://travis-ci.com	- Used to build, test, deploy code hosted on Github - Notify team with test results through email, postings or any IRC channel - Support various programming languages (Java, Python, Ruby, Node.js) - Provide its command-line UI - Enable parallel deployment and testing

Table 2.15: Continuous Deployment

Continuous Deployment	
Tools	**Features**
Codeship https://codeship.com	- Enable multiple deployments sequentially or parallel - Enable developers to run deployments commands on an authenticated remote server using SSH; this feature allows developers to trigger deployment/update on external systems for stakeholders
Travis CI https://travis-ci.com	- Enable developers to setup continuous deployment schedule - Enable developments to automate deployment schedule - Integrated deployment with Github

Table 2.16: IaaS/PaaS

IaaS/PaaS	
Tools	**Features**
Heroku https://www.heroku.com	- Heroku is a PaaS that support (Ruby, Java, Node.js, Python, PHP) - Enables continuous automated deployment - Provide logs (using Logplex) and maintain version control of code
AWS https://aws.amazon.com	- AWS enables automated application deployment - AWS enables automated monitoring and reporting - AWS enables Database management - AWS enables tools integration - AWS enables auto-scaling
Google App Engine https://cloud.google.com	- Cloud database - Enable cloud and tools integration - Synchronise, Google Accounts, OpenID, and OAuth - Auto-scale Google Cloud Endpoints - Enable automated deployment - Enable real-time monitoring - GAE offers pay-per-use and a free account for researchers and students - GAE enables custom and multi-languages applications development - GAE offers services via APIs

Table 2.17: Monitoring

Monitoring	
Tools	**Features**
Nagios https://www.nagios.org	- Nagios is an open-source application that monitors systems - Nagios also provides remote monitoring through its Remote Plugin Executor, which supports SSH and SSL encrypted tunnels - Nagios enables developers to build reporting units using programming languages (Shell Scripts, C++, Perl, Ruby, Python, C#, etc.) - Provide automated logs
New Relic https://newrelic.com	- New Relic provides insight into an SD application at runtime - New Relic delivers unique monitoring log metrics of cloud application development and its deployment from UI to the backend - New Relic provides continuous automated reporting on health, status, runtime, build, deployment, and performance or a cloud application

Table 2.18: Database Management

Database Management	
Tools	**Features**
MongoDB https://www.mongodb.com	- MongoDB is a free, open-source, cross-platform, document-oriented database program - Classified as NoSQL database application, MongoDB avoids a traditional table-based relational database in favour of JSON-like documents with the dynamic schema - MongoDB provides developers with ad hoc queries, aggregation using MapReduce and Server-side JS

Table 2.19: Logging/Security

Logging/Security	
Tools	**Features**
Loggly https://www.loggly.com	- Loggly summarises the software application log and provides real-time analysis for software processes - Loggly increases delivery speed and provides a guided-data log to DevOps team based on application troubleshooting results - Loggly manages logs from any source or application test units
Papertrail https://papertrailapp.com/	- Cloud-based log monitoring system - Integrates with Heroku metrics logs - Integrates with Slack collaborative tool

Table 2.20: Build

Build	
Tools	**Features**
Codeship https://codeship.com	- Codeship provides build capability for DevOps team - Codeship provide custom build configuration for many programming languages
Travis CI https://travis-ci.com	- Travis CI provides a robust build environment that can be setup in .travis.yml

Table 2.21: Testing

Testing	
Tools	**Features**
Cucumber https://cucumber.io	- Cucumber runs automated acceptance tests written in a behaviour-driven development - Cucumber merges SD specifications and tests documentation into one cohesive log - Cucumber uses Gherkin, a language that defines Cucumber test cases, which is designed to be human-readable non-technical
Junit http://junit.org/junit4	- Junit builds test functions from normal functions by providing *@Test* annotation to the method header - Automated test units are composed of a collection of annotated Java methods that handle particular exceptions or provide run-time reports

Table 2.22: Communication and Collaboration

Communication and Collaboration	
Tools	**Features**
Trello https://trello.com/	- Trello is a web-based collaboration tool that improves DevOps team project management and project tracking by offering a shared dashboard for automated custom task cards.
Slack https://www.Slack.com	- Slack is a web-based service for internal private chat and messaging - Slack supports group chats, and support video - Slack relays messages through SMS services - Slack integrates the team progress from different repositories such as BitBucket and Github.

2.4.2. Benefits and Challenges of DevOps Adoption for Cloud IoT Apps

As discussed, DevOps is culture-driven and technology-driven. It has been suggested that the DevOps culture is a central characteristic in adopting the DevOps approach using a defined set of practices (see Table 2.10) (Ghantous & Gill 2017). The DevOps development pipeline can be constructed using a cloud-centric set of tools (see Tables 2.12–2.22). However, the adoption of DevOps for software applications (in particular, IoT) may also pose challenges for developers, especially when deploying an IoT-type application in a cloud-configured development pipeline. The SLR identified a set of 20 DevOps benefits and seven challenges (see Table 2.23 and Table 2.24). This review of selected studies indicates that the adoption of DevOps offers several benefits (72%). However, some challenges (28%) need to be considered when adopting the DevOps approach to deploy IoT applications to the cloud/multi-cloud.

Table 2.23: Benefits of DevOps Adoption

Benefits	Sources
Central code management	S_6, S_9, S_{19}, S_{20}, S_{22}, S_{23}, S_{31}, S_{43} ,S_{107}
Increase jobs processing using automated deployment	S_6, S_9, S_{19}, S_{20}, S_{22}, S_{23}, S_{31}, S_{43}, S_{12} , S_{14} , S_{30}, S_{110}, S_{107}, S_{130}
Enable automated build	S_1, S_3, S_6, S_9, S_{10}, S_{12}, S_{14}, S_{19}, S_{20}, S_{22}, S_{23}, S_{31}, S_{43} ,S_{30}, S_{110}, S_{107}, S_{130}
Enable automated unit (function) testing	S_6, S_9, S_{10}, S_{19}, S_{20}, S_{22}, S_{23}, S_{43} , S_{107}
Provide automated behaviour-driven testing	S_6, S_9, S_{10}, S_{19}, S_{20}, S_{22}, S_{23}, S_{43}, S_{107}
Provide automated Integration for pipeline tools	S_1, S_3, S_6, S_9, S_{19}, S_{20}, S_{22}, S_{23}, S_{31}, S_{43}, S_{12} , S_{14} , S_{30}, S_{110}, S_{107}, S_{130}
Provide real-time automated monitoring and log collection from application deployment	S_1, S_3, S_6, S_9, S_{19}, S_{20}, S_{22}, S_{23}, S_{31}, S_{43}, S_{12} , S_{14} , S_{30}, S_{110}, S_{107}, S_{130}
Provide real-time communication for the DevOps team	S_6, S_9, S_{10}, S_{12}, S_{19}, S_{20}, S_{22}, S_{23}, S_{27}, S_{43}, S_{107}, S_{116}, S_{132}
Provide cloud database management	S_1, S_6, S_{27}, S_{43}, S_{25}, S_{11}, S_{107}
Provide rollback of code and continuous planning	S_3, S_6, S_9, S_{12}, S_{19}, S_{20}, S_{43}, S_{116}, S_{107}
Continuous development of new ideas based on continuous planning	S_3, S_6, S_9, S_{12}, S_{19}, S_{20}, S_{43}, S_{116}, S_{107}
Rapid delivery using cycle build–test–deploy in a fully automated environment	S_6, S_9, S_{19}, S_{20}, S_{22}, S_{23}, S_{31}, S_{43}, S_{12} , S_{14} , S_{30}, S_{110}, S_{107}, S_{130}
High scalability of resources: no downtime	S_{15}, S_{43}, S_{107}, S_{25}, S_{13}, S_{21}
Provide real-time visibility of the pipeline	S_6, S_9, S_{12}, S_{19}, S_{20}, S_{22}, S_{23}, S_{27}, S_{43}, S_{107}, S_{125}, S_{25}, S_{13}, S_{21}
Provide support for IoT applications automated nature	S_7, S_{11}, S_{107}, S_{103}, S_{93}, S_{112}, S_{102}
Secure pipeline using API-key exchange, Oath, two-way authentication	S_{43}, S_{107}, S_{17}, S_{105}, S_{17}, S_{51}, S_{59}, S_{76}, S_{67}
Use cloud capabilities to provide IoT apps with necessary support using SaaS, PaaS and IaaS	S_7, S_{11}, S_{107}, S_{103}, S_{93}, S_{112}, S_{102}, S_{58}, S_{62}, S_{70}, S_{85}, S_{86}, S_{91}, S_{102}, S_{103}
Use of middleware software or cloud services to connect to IoT applications and devices	S_{91}, S_{92}, S_{93}, S_{107}, S_{11}, S_7
Use MQTT, RDF, RFID, WSN, WAN, mobile, Bluetooth and internet to connect to IoT devices and sensors	S_{32}, S_{34}, S_{38}, S_{38}, S_{53}, S_{57}, S_{66}, S_{73}, S_{78}, S_{79}, S_{72}
Exchange IoT data that can be stored and managed in the cloud using DevOps tools	S_7, S_{11}, S_{107}, S_{84}, S_{83}, S_{84}, S_{33}, S_{49}, S_{55}, S_{81}
72%	

Table 2.24: Challenges of DevOps Adoption

Challenges	Sources
Reliance on various tools: proposed automated pipeline relies mainly on the integration of many tools, which increases the risk of failure and incompatibility	S_{10}, S_{20}, S_{22}, S_{23}, S_{43}, S_{116}, S_{8}, S_{4}, S_{21}, S_{107}
DevOps team culture clashes: different languages, geographical difference, time difference	S_{10}, S_{20}, S_{22}, S_{23}, S_{43}, S_{116}, S_{107}, S_{117}, S_{131}
IoT applications: IoT-applications interactions with sensors and devices generate a large amount of data, which could prove challenging to store and process. The IoT-applications require a deployment platform that enables scalability, interoperability, and resources sharing.	S_{107}, S_{36}, S_{37}, S_{69}, S_{71}, S_{11}, S_{90}, S_{7}, S_{102}, S_{85}, S_{55}, S_{80}, S_{33}, S_{49}, S_{81}
Data mining: due to a large amount of transmitted data by IoT devices, organisations need to rely more on data mining tools for processing and discovery, which could prove challenging to integrate	S_{107}, S_{36}, S_{37}, S_{69}, S_{71}, S_{11}, S_{90}, S_{7}, S_{102}, S_{85}, S_{49}, S_{81}, S_{72}, S_{62}, S_{46}, S_{47}, S_{45}
Privacy: protecting user privacy could prove difficult due to the amount of IoT data exchanged, which may include user information	S_{36}, S_{107}, S_{37}, S_{39}, S_{7}, S_{11}, S_{69}, S_{105}, S_{17}, S_{18}, S_{51}, S_{59}
Security: security threats escalate with more IoT devices and sensors connected to the network, which exposes organisations and users to potential attacks from hackers and cybercriminals	S_{36}, S_{107}, S_{37}, S_{39}, S_{7}, S_{11}, S_{69}, S_{105}, S_{17}, S_{18}, S_{51}, S_{59}, S_{76}, S_{37}, S_{32}
Chaos: evaluation of IoT could lead to untested or incompatible devices and sensors to be added to the network without testing which could create chaos and performance inconsistency	S_{32}, S_{34}, S_{35}, S_{36}, S_{37}, S_{47}, S_{53}, S_{54}, S_{56}, S_{60}, S_{61}, S_{63}, S_{64}, S_{65}, S_{66}, S_{69}, S_{72}, S_{75}, S_{78}, S_{107}
28%	

Table 2.24 shows several challenges that could face organisations adopting DevOps for software projects. The challenges in Table 2.24 are the results of the SLR survey conducted to answer the first research question RQ1. The SLR results section contains the collective knowledge about DevOps paradigm (concepts, practices, tools). SLR results provide rigour data about DevOps that can be used to develop a reference architecture that enables the adoption of DevOps. With the research scope in mind (see Section 1.5), this thesis focuses on providing a template solution for the first three challenges listed in Table 2.24. The challenges in Table 2.24 highlight the constituents of the research gaps. The research gaps explained in section 2.4.3 indicate the need for a reference architecture to address the challenges identified in this section and to accomplish the thesis aims and scope.

2.4.3. RESEARCH GAP

The SLR analysis and review (see Section 2.3, Table 2.24) has identified gaps related to the DevOps adoption, IoT-applications deployments and interactions, and deployment platforms. An investigation into the context of DevOps, the cloud/multi-cloud and IoT indicates that DevOps adoption for IoT application deployment to the cloud/multi-cloud lacks contextual guidelines. The SLR results (see Sections 2.3.4, 2.3.5 and 2.3.6) indicate that model-driven reference architecture based on the DevOps approach may be a possible solution for IoT application deployment to the cloud/multi-cloud. The research gaps in this thesis are derived from the research problem (see Section 1.2), research aims (see Section 1.4), and research question (see Section 1.3) and SLR data analysis see Section 2.3. The proposed DRA framework aims to address the research gaps (see Table 2.25) by providing a comprehensive reference architecture that enables IoT-applications deployment to multi-cloud using DevOps approach and addresses the research challenges presented by DevOps and IoT. The research gaps are presented in Table 2.25 and linked to the corresponding DevOps and IoT challenges:

Table 2. 25: Research Gaps

DevOps Adoption Challenges (Thesis Scope)	Research Gaps	Research Question
Reliance on various tools: proposed automated pipeline relies mainly on the integration of many tools, which increases the risk of failure and incompatibility	The vast number of available DevOps tools that could be used to create a DevOps pipeline for the reference architecture could face integration and compatibility challenges.	- RQ1 must address the matter by recommending a set of DevOps and cloud tools. - RQ2 must provide a template solution for the DevOps applying using the proposed DRA reference architecture.
DevOps team culture clashes: different languages, geographical difference, time difference	There are many types of research about DevOps paradigm. Hence, it is crucial to understand the DevOps concepts and outline the DevOps practices and tools before embarking on adopting the DevOps approach.	- RQ1 must address the DevOps paradigm and provide a detailed listing of the DevOps concepts, practices and tools. - RQ1 must also highlight the importance of integrating DevOps and multi-cloud to support IoT-application deployment to multi-cloud.
IoT applications: IoT applications interactions with sensors and devices generate a large amount of data, which could prove challenging to	- Deploy IoT applications to the cloud/multi-cloud - Manage connectivity between the IoT application and IoT	- RQ2 main aim is to address the challenges of deploying IoT-applications to multi-cloud using DevOps. - RQ2 must address the

DevOps Adoption Challenges (Thesis Scope)	Research Gaps	Research Question
store and process. The IoT-applications require a deployment platform that enables scalability, interoperability, and resources sharing.	sensors configured on a Raspberry Pi; in this step, it is necessary to choose a suitable connection protocol for the organisation context. - Avoid vendor lock-in for IoT application deployment to the multi-cloud. - Avoid database vendor lock-in for IoT application deployment to the multi-cloud	connectivity challenges of between multi-could IoT-applications and devices. - RQ2 must provide an answer to the vendor lock-in challenge encountered when deploying software applications to multi-cloud. The vendor lock-in is caused when any of the clouds in the heterogeneous system hosts the deployment configurations of the application. - RQ2 must provide an answer to the vendor lock-in challenge cause when any of the clouds in the heterogeneous system hosts the IoT-database.

The research gap (see Table 2.25) can be summarised as follows:

- Deploy IoT applications to the cloud/multi-cloud
- Manage connectivity between the IoT-applications and IoT-sensors
- Avoid vendor lock-in for IoT application deployment to the multi-cloud
- Avoid database vendor lock-in for IoT application deployment to the multi-cloud
- Improve the agile adaptive implementation using DevOps (see Figure 2.5).

The proposed DRA framework may fill the research gaps. In particular, the DRA framework may provide a practical solution to the iterative approach used in the Gill Framework Adaptive Enterprise Project Management (APEM) (Alzoubi, Gill & Al-Ani 2015) for software implementation which may improve agile adoption for software development (Qumer, Henderson-Sellers & McBride 2007).

The proposed DRA framework may provide applicable solutions to the highlighted agile adaptive iteration implementation (see Figure 2.5) because it is founded on general characteristics based on DevOps concepts—most importantly, automation, CI, continuous deployment and real-time monitoring.

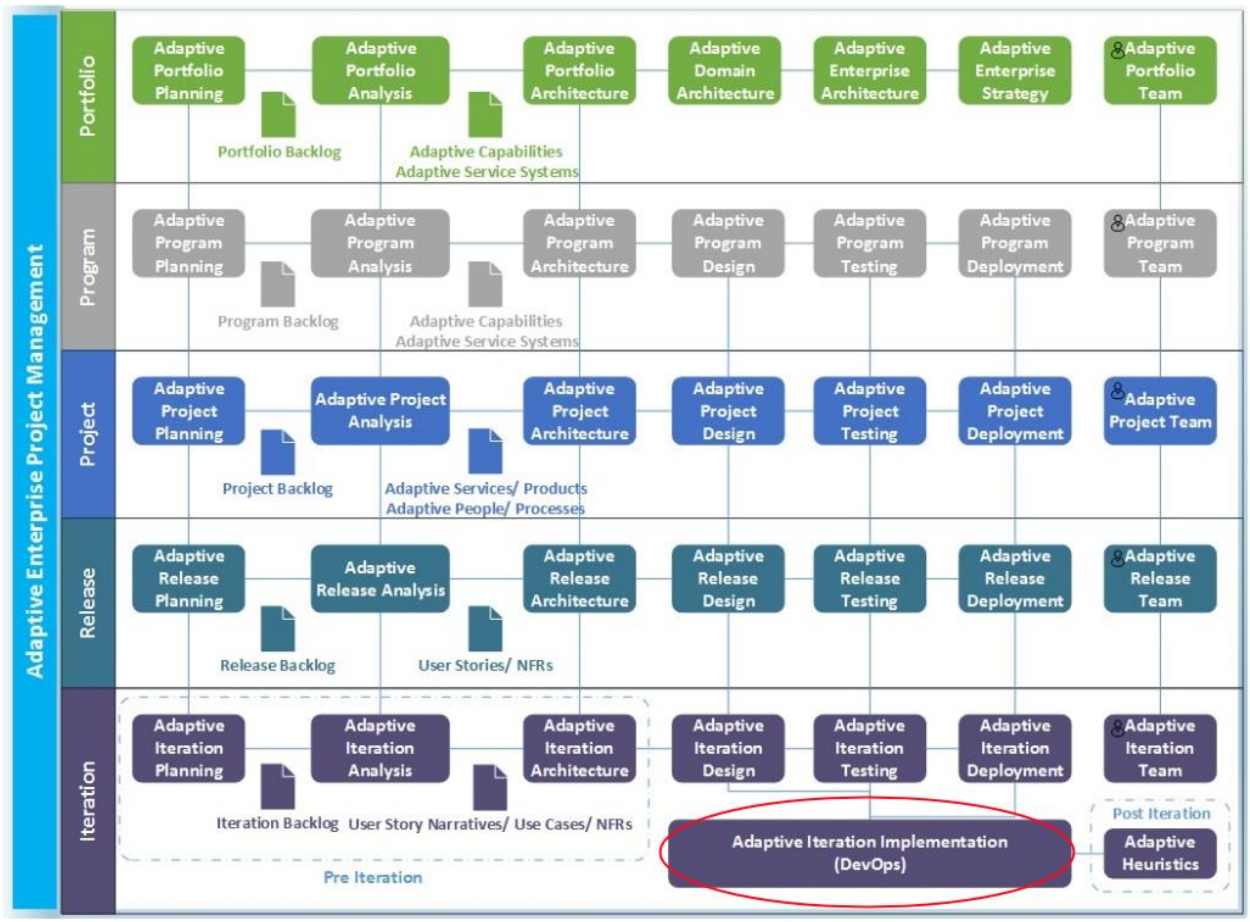


Figure 2.5: DRA for AEPM (adapted from the Gill Framework®)

2.5. SUMMARY

This chapter presented an SLR based on guidelines published by Kitchenham and Charters (2007). The SLR generated raw data from a list of selected studies (S_{index}). These data were reviewed and analysed to investigate the relationship between the contexts of DevOps, the cloud/multi-cloud and IoT. The SLR analysis produced notable results regarding DevOps concepts, practices and tools. The results also identified the benefits, challenges and research gaps based on the analysis in Section 2.3. The purpose of the SLR is to determine what is known about DevOps (see Section 2.4.1). The SLR analysis in Section 2.3 presented rigorous data that indicates that the DevOps approach might be linked to the cloud/multi-cloud and IoT. The results of the SLR addressed the first research question (RQ_1) and provided richness and rigour to the information about DevOps (concepts, practices, tools and DevOps adoption benefits and challenges to the cloud and IoT). The second research question (RQ_2) is addressed in Chapters 3–5. The answer to RQ1 and RQ_2 may provide a practical solution to the research gaps (see Table 2.25). This chapter has laid the foundation for the DSR method adopted from Gregor and Hevner (2013) and Peffers et al. (2007) in Chapter 3, which uses the DSR method to develop and evaluate the proposed DRA framework.

Chapter 3: Design Science Research Method

A research methodology explains how a researcher should proceed in the development and evaluation of an artefact. The selection of a research methodology and its research instruments depends mostly on the landscape of the research problem and its underlying objectives specified in the research aim in Chapter 1. The selection of a research methodology also depends on the availability of resources and data from the SLR conducted in Chapter 2. Many research methods could have been used to perform this research. However, the nature of the research question and its underlying objectives required a practical iterative approach to achieve the research aims specified in Chapter 1. Thus, a DSR method based on guidelines published by Gregor and Hevner (2013) and Peffers et al. (2007) was selected to develop and evaluate the proposed DRA framework. The DSR method is the most suitable approach to investigate and iteratively produce the DRA framework. This chapter discusses the potential research methodological choices for this research. It presents the iterative approach used in the DSR to identify the research problem, analyse the SLR data, design, develop and evaluate the proposed DRA and outline the thesis output.

3.1. RESEARCH DESIGN

The research objectives and motivation for the development of the proposed DRA framework were discussed in Chapters 1 and 2. This chapter outlines the methodology adopted in this thesis. The research methodology explains how a researcher should go about finding a suitable approach to develop and evaluate an artefact (Guba & Lincoln 1994). The selection of a research method depends on several factors, including the nature of the research problem and its underlying objectives, the availability of resources and data, and the research traditions that are local to that institute or organisation (Benbasat et al. 1987). The nature of agile, iterative development indicates that an iterative DSR method may be suitable for this project. An iterative DSR may be helpful to focus on the description of the study. The fundamentals of DSR include rigorous data gathering and analysis, the design and development of a general artefact and the evaluation of that artefact. The DSR method has been similarly used in previous research and studies concerning the configuration information system architecture (Gill & Chew 2019), IoT-enabled digital information systems (Dasgupta et al. 2019), the areas of cloud computing and big data (Litchfield & Althouse 2014), and the areas of adaptive enterprise management using action design research case study (Gill et al. 2016).

Consequently, it is beneficial for this project to select a research method that enables an iterative problem-solving approach. Thus, the DSR iterative method is selected in this thesis. The DSR adopted in this thesis is founded on guidelines published by Gregor and Hevner (2013) and Peffers et al. (2007). The DSR overview is composed of several steps, which are mapped in Table 3.1 and illustrated in Figure 3.1.

Table 3.1: DSR Steps

Step	Description
Purpose and scope	The purpose of the DSR is to establish a transparent systematic approach that could be used to create the DRA artefact. The DSR systematic process uses a well-known method founded on guidelines published by Gregor and Hevner (2013) and Peffers et al. (2007). The DSR approach is explained as follows: The DSR method identifies the research problem using the rigorous from the initial research (see Chapter 1, research background and related work and see Chapter 2, SLR results). The DSR scope is to achieve the research goals and create the DRA artefact that satisfies the thesis scope (see Chapter 1, Table 1.3) and research aims (see Chapter1, Figure 1.7). The DSR method includes an evaluation method to determine that DRA is fit for its purpose (address the research gaps explained in Table 2.25).
Knowledge of initial research	The DSR uses initial research from the SLR analysis and results to identify what is known about the research topics and to determine the research gaps that highlight the research problem. The initial research also includes recent publications that addressed RQ1 and RQ2 (e.g., Ghantous and Gill 2017, 2018, 2019).
Abstraction and generalisation	The DSR aims to produce or generate a comprehensive reference architecture that uses characteristics based on common terminologies. The reference architecture is explained in chapter 1, section 1.4; can be used to the DRA design models in any context to address the research gaps (see Table 2.25). The general characteristics in the reference architecture are based on DevOps concepts and cloud terminologies in the SLR Results in chapter 2.
Design and development of artefacts	The DSR aims to create a comprehensive architectural design artefact that enables DevOps adoption and address the challenges and research gaps explained in Table 2.24 and 2.25. The DSR design and development produce a new DRA framework composed of general characteristics, a general abstract design, and reusable framework composition. The DRA can be instantiated into a composition set suitable for the context of an organisation or institute. The DRA is not fixed to a particular instance. Instead, the proposed DRA can be applied to numerous instances depending on the development environment context. The DRA framework provides new knowledge about the adoption of the DevOps approach for software application (e.g. IoT-application) deployment to multi-cloud.
Evaluation	The DSR includes an evaluation method (Empirical Evaluation) to evaluate the proposed new DRA framework (see Chapter 5) and determine if the framework is fit for its purpose (e.g. answering the research question (see Section 1.3) and achieving the aims of the thesis (see Section 1.4) with the scope of the research (see Section 1.5). The evaluation of the DRA (see Chapter 5) aim to establish if the DRA framework addresses the research gaps explained in Table 2.25.
Justificatory knowledge and output	The DRA's usefulness and applicability are determined in Chapter 5 using instruments that apply indicative measurement techniques using the evaluation data. The DSR output outlines the project outputs and discusses the proposed DRA limitations and possible future research ideas.

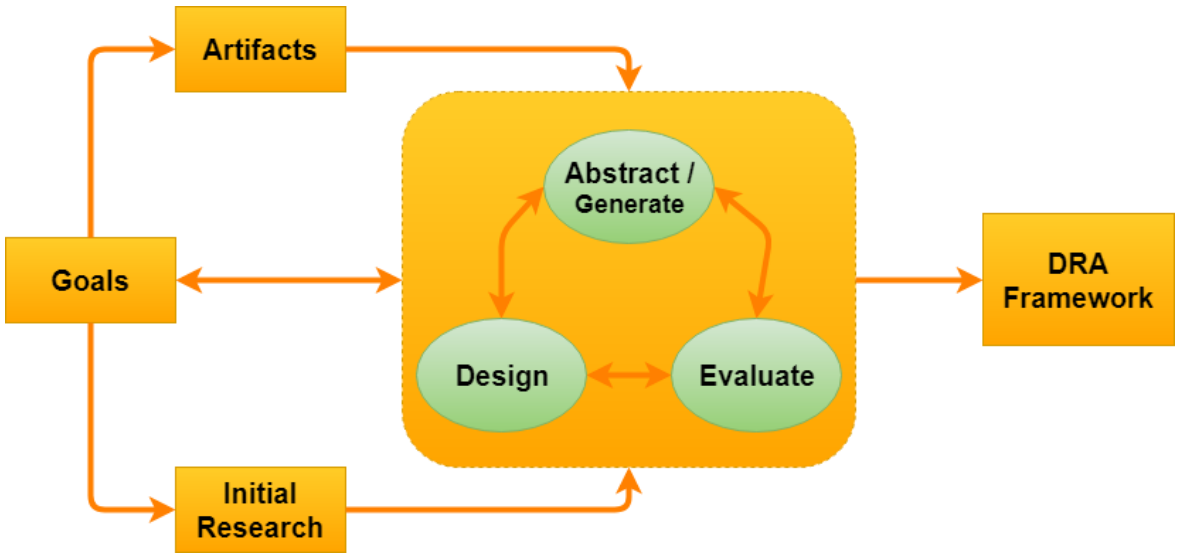


Figure 3.1: DSR Overview

3.2. DSR: METHODOLOGY

This research adopts a well-known DSR methodology (Gregor & Hevner 2013; Peffers et al. 2007), which is a system of principles, practices and procedures applied to a specific branch of knowledge to produce and present high-quality research artefacts. The DSR aims to provide verifiable contributions through the design, development and evaluation of an artefact. The artefact development may involve the review of existing theories and knowledge to develop a solution or artefact for the intended purpose and audiences. The DSR is composed of three primary process stages (see Figure 3.2):

- Stage 1—DSR main flows. There are two flows of this stage:
  - Initial research and SLR Results: Research background and related work analysis, SLR analysis and results.
  - DRA Framework: The DRA framework construction (create the DRA framework architectural model).
- Stage 2—DSR process steps. The DSR process in this thesis is composed of six steps:
  - Problem identification: In Chapter 1, the research background and related work analysis helped identify the research problem, research question and its underlying objectives. This initial research laid the foundation of the project scope and helped in recognising the research aims.
  - Analysis: The SLR conducted in Chapter 2 provided rich information about DevOps and its relationship with the cloud/multi-cloud and IoT. The SLR results identifying the challenges presented by the adoption of DevOps for software application deployment (e.g. IoT-applications) to multi-cloud. The challenges identified in chapter 2, Table 2.24 identified the problems facing the project scope and aims and consequently highlighted the research gaps (see Chapter 2, Table 2.25).

- Design: This step aims to create a comprehensive architectural design model for the proposed DRA framework. The reference architecture model is founded on general characteristics terminologies derived from the DevOps concepts, cloud infrastructure and services. The DRA design model that aims to address the research gaps (see Chapter 2, Table 2.25) within the boundaries of the research scope (see Chapter 1, Section 1.5).
- Development: Develop the DRA framework components based on the architectural design model. The new comprehensive DRA framework aims to address the research gaps (see Chapter 2, Table 2.25). The new comprehensive DRA reference architecture is not fixed to a particular situation or environment but can be applied to numerous instances in any context.
- Evaluation: Evaluate the comprehensive DRA framework and determine if the new DRA reference architecture is fit for its purpose (address the research gaps within the project scope boundaries). The evaluation step adopts an empirical evaluation method (see Chapter 3) composed of 4 case studies (Industry case study, research case study, teaching case study, and industry field survey).
- Output: The research journey and output, the project's key contributions and publications, the project's limitations, and future research.

- Stage 3—DSR outcomes. The DSR method outcomes in this thesis is composed of six outputs:
  A. Research question and research aim (see Chapter 1).
  B. Objective solutions and research gaps (see Chapter 2).
  C. Design artefact (see Chapter 4).
  D. Development artefact (see Chapter 4).
  E. Empirical evaluation (see Chapter 5).
  F. Thesis results (see Chapter 6).

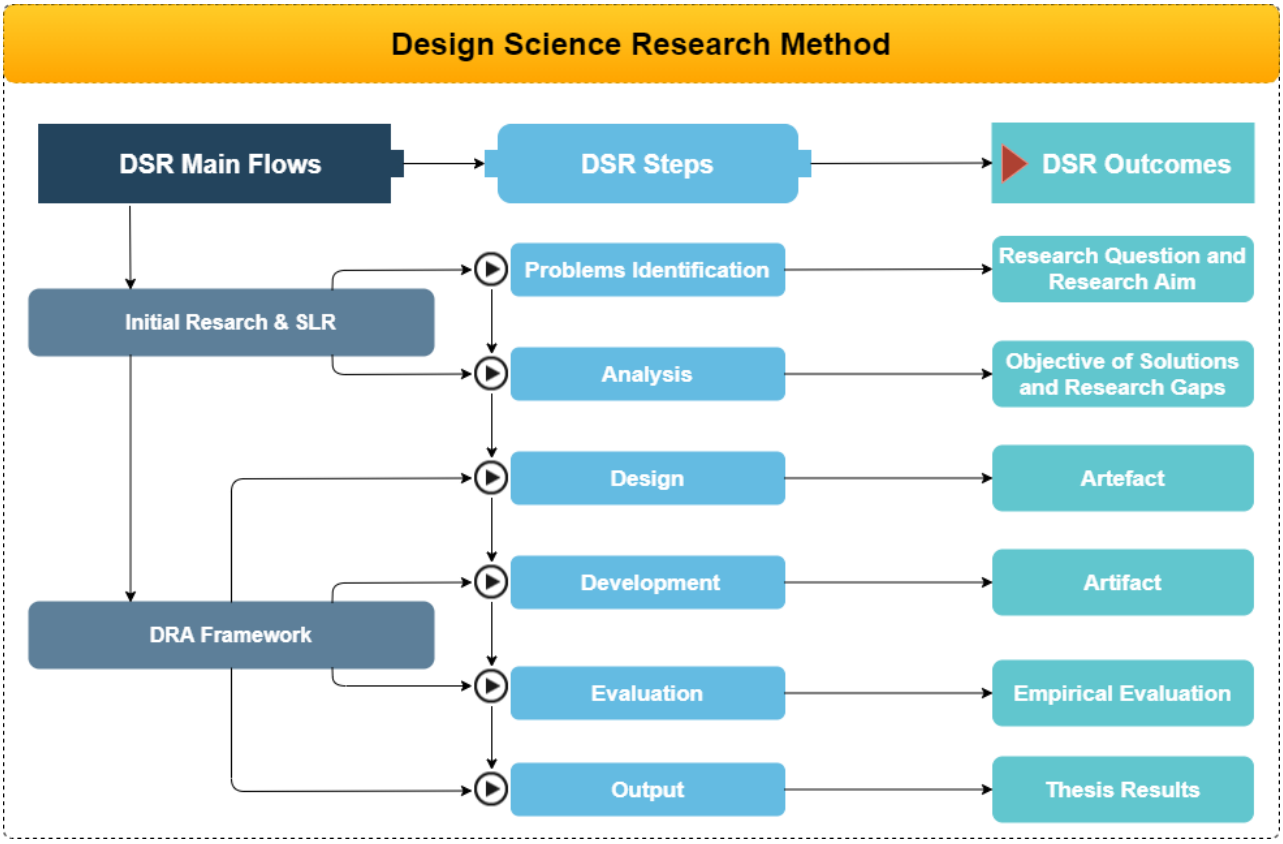


Figure 3.2: DSR Process

3.2.1. PROBLEM IDENTIFICATION

As presented in Section 1.3, the main research question is: 'How can IoT applications be deployed to the multi-cloud using the DevOps approach?'

The first stage of the DSR is about the background studies, related work analysis (see Chapter 1) and the SLR conducted in Chapter 2. The analysis conducted in Chapter 1 helped identify the research problem. The research question (and its underlying objectives) was defined based on the problems at hand explained in the research gaps (see Chapter 2, Table 2.25). The research gaps discussed in chapter 2 identify the challenges presented by the adoption of DevOps for software application (e.g. IoT-applications) deployment to multi-cloud. The research question was subdivided into RQ_1 and RQ_2:

- **RQ_1:** What is known about DevOps?
- **RQ_2:** How can IoT applications be deployed to the cloud (single and heterogeneous) using the DevOps approach?

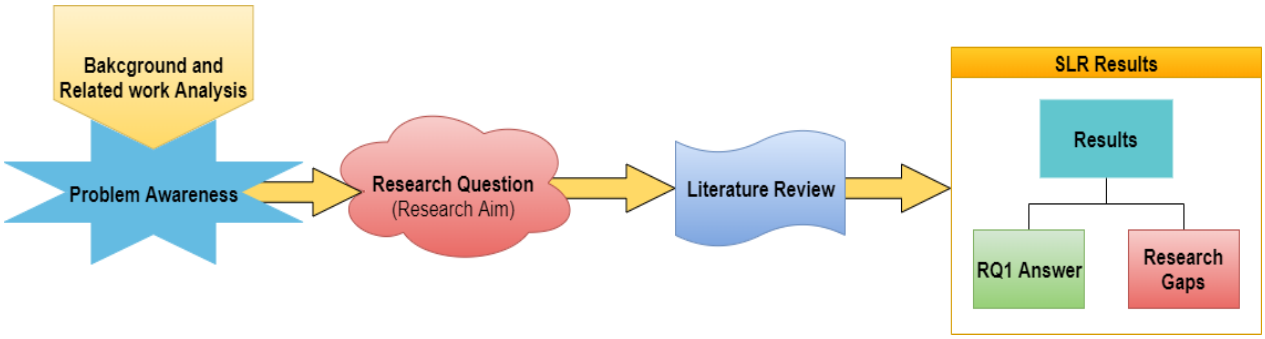


Figure 3.3: Problem Identification

3.2.2. ANALYSIS

The analysis step of the DSR method reviews and synthesises the information and resources available about cloud/multi-cloud, DevOps and IoT (see Figure 3.4). The SLR conducted in Chapter 2 provided rich information about DevOps and its relationship with the research topics (cloud/multi-cloud and IoT). The SLR results also highlighted vital research gaps (see Chapter 2, Table 2.25). The research gaps presented the challenges for research aim (see Chapter 1, Section 1.5) concerning the adoption of the DevOps approach for application deployment to multi-cloud. The suggested solutions below align with the research aims (see Chapter 1, Section 1.4) and aim to present practical solutions to the research gaps (see Chapter 2, Table 2.25):

- Define DRA set characteristics: DRA characteristics are common terminologies used to provide a foundation to create a general architectural design. The characteristics are based on DevOps concepts and use the cloud/multi-cloud infrastructure as a platform.
- Create a DRA architectural design: The DRA architectural design is a general design model that is not fixed to a particular instance but can be applied to numerous instances.
- Define DRA composition: The DRA composition is founded on DRA characteristics and development using the DRA design model as a blueprint. The DRA composition can be instantiated to fit the context of the development environment.
- Use a CI broker to host IoT applications provisioning and deployment configurations to avoid vendor lock-in.
- Use a central database to store IoT data and avoid vendor lock-in.

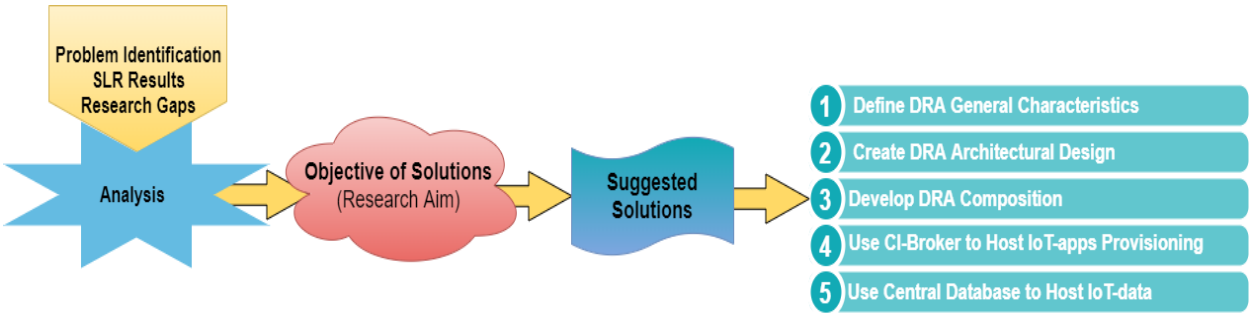


Figure 3.4: Analysis

3.2.3. DESIGN

This step focuses on developing contextual, conceptual, logical, physical and operational models of DRA for deploying IoT applications to the cloud (single and multiple). The proposed DRA architecture model is based on DRA general characteristics. The DRA architectural design is a general model that is not fixed to a particular instance but can be applied to any development context. The DRA architectural design includes a CI broker instrument that hosts application deployment configurations to avoid vendor lock-in. DRA architecture includes a central database to host the IoT application database and avoid vendor lock-in. The below DRA architectural models are illustrated in Figure 3.5:

- DRA contextual model: Describes the relationship between the research topics.
- DRA conceptual model: Represents conceptual architecture based on the DRA general characteristics, DevOps concepts and cloud/multi-cloud (infrastructure, services).
- DRA logical model: Represents the logical architecture based on DevOps practices and cloud services.
- DRA physical model: Describes the physical representation of the logical model and explains the structural deployment flow based on the logical model components.
- DRA operational model: Represents the deployment chain of the software application (IoT application). DRA pipeline instances follow the operational model flow of operations.

The DRA reference architecture design model is a template solution that can be used to address the research gaps (see Chapter 2, Table 2.25). The DRA reference architecture is not fixed to a particular situation. The objective is to design a reference architecture that could be instantiated for any development context. The DRA reference architecture model provides a practical method to how to adopt DevOps for software development and how to enable automated software applications (e.g. IoT-applications) deployment to multi-cloud. The comprehensive DRA reference architecture is an applicable answer to the research question (RQ2) of this thesis.

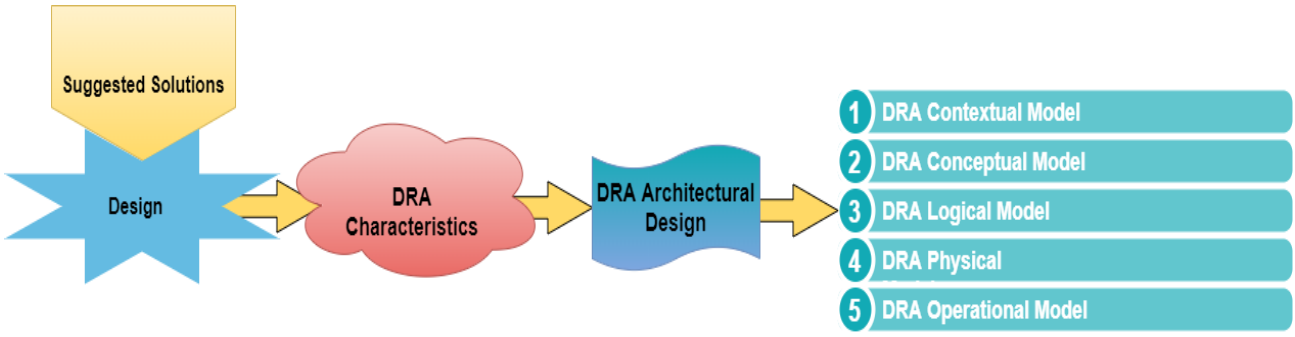


Figure 3.5: Design

3.2.4. DEVELOPMENT

The proposed DRA framework components are developed in this section. The DRA is founded on three main components:

- Resources: This component includes DRA architecture design, software and hardware.
- Configuration: This component presents the development of the IoT application, the configuration of the DRA pipeline instance and setup of the IoT network.
- Output: This step presents the DRA operational model instances DRAv1.0 and DRAv2.0.

The DRA framework is a new comprehensive artefact development in this step of the DSR method to provide practical solutions to the research gaps (see Chapter 2, Table 2.25) and address the research problem highlighted in the research questions (RQ1 and RQ2). DRAv1.0 (single-cloud) and DRAv2.0 (multi-cloud) are applicable instances of the DRA framework that explains precisely how to adopt DevOps for software applications (e.g. IoT-applications deployment to (cloud/multi-cloud).

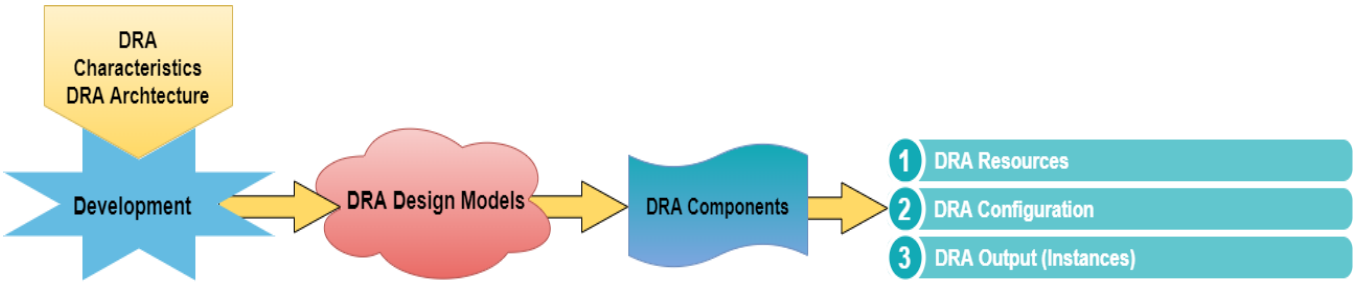


Figure 3.6: Development

3.2.5. EVALUATION

The proposed DRA framework is evaluated using an empirical evaluation (see Figure 3.7). The evaluation process is composed of four iterations:

- industry case study
- research case study
- teaching case study
- industry field survey.

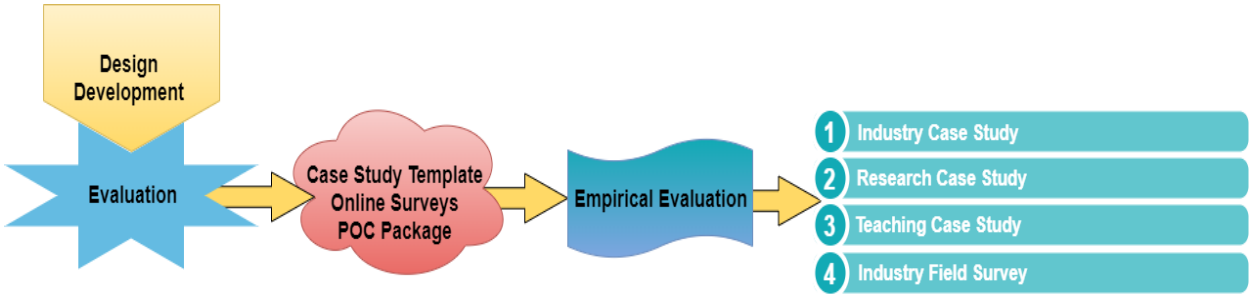


Figure 3.7: Evaluation

3.2.5.1. Case Studies Design

The case study methodology is commonly used to evaluate software engineering research phenomena (Runeson & Höst 2009). A case study is virtually any contemporary phenomenon in a real-world context (Aberdeen 2013 & Yin 2009). In software engineering, a phenomenon may involve development, operation, maintenance and related artefacts (Jedlitschka & Pfahl 2005). Case study research methodology is an empirical research method that can be used to test, generate or describe a theory or phenomenon (Runeson & Höst 2009) to determine whether the context of the phenomenon can be replicated or reused in real-world settings. The possible outcomes and contributions of case studies can be frameworks or theory (conceptual). Hence, the case study methodology is suitable for evaluating the DRA research project.

Klein and Myers (1999) defined three types of case study—positivist, critical and interpretive—depending on the research perspective:

- A positivist case study searches evidence for formal propositions, measures variables, tests hypotheses and draws inferences from a sample of a stated population. Software engineering case studies tend to lean towards a positivist perspective, especially for analytical and conceptual-type research (Runeson & Höst 2009).
- A critical case study aims at social critique by identifying different forms of social, cultural and political aspects that may hinder human ability.
- An interpretive case study attempts to understand a phenomenon from the participants' interpretation of their context.

Case studies may include qualitative and quantitative data. Conveniently, qualitative data collected from organisations are required to evaluate the DRA framework. The method used to evaluate the DRA artefact may be replicated to comply with the organisation context. The case study evaluation method is composed of five main steps (Runeson & Höst 2009):

1. **Case study plan:** Plan the case study and identify the objectives:
    - Identify the case study organisation: State the organisation and participants. Ethical considerations have been given to organisations and participants to provide anonymity. Organisations and participants have been identified with codenames. Ethical considerations must be prepared following formal ethics approval obtained from the researcher's organisation. Key ethical factors may include informed consent, confidentiality and handling of sensitive results.
    - Organisation context: Describe the organisation's environment and aims.
    - Organisation need and problem: Identify the problems and needs of the organisation.
    - Solutions: Outline the possible solutions provided by the DRA framework.
    - POC and testing: Outline the testing materials used in the organisation context to evaluate the DRA.
2. **Preparation for data collection:** Define the data collection method used in the case study. In this research, a case study template was used as a collection procedure in the case studies

(industry and research lab). The case study template is a formal template-based approach that includes structured pre-planned questions about the evaluated components of the DRA. See Appendix G for further information.

3. **Collecting data:** The case study data were collected using the case study template. The data are stored on CloudStor (UTS-recommended cloud storage). See Appendix E for further details. The participants in the case studies provided qualitative feedback about the DRA.
4. **Data analysis:** The qualitative data collected using the case study template were analysed using the hypothesis confirmation general technique of analysis (Runeson & Höst 2009). The hypotheses are the evaluation criteria (Carvalho 2012) (see Table 3.2). Participants' feedback was cross-examined against the evaluation criteria by highlighting the occurrences of the criteria in the text. The data was organised in tables, and cross-examined to determine its association with the evaluation criteria (see Table 3.2). The industry case study data and the research case study data were analysed and organised into analysis tables.
5. **Reporting:** The report communicates the findings of the case studies. The case study template used the linear-analytic approach (Runeson & Höst 2009), which is suitable for academic purposes because of its structured method of reporting experiments. The industry case study report was organised into tables, which included the testing steps of the case study and the description of each step. The report tables of the case studies followed a similar pattern to the case study design (Runeson & Höst 2009), with more emphasis on each step to highlight its interest for the audience. The report provides evidence of DRA's applicability in the context of the organisations of the case studies using the evaluation criteria (see Table 3.2). Thus, the hypotheses were correct in the setting of the DRA.

Table 3. 2: Case Study Evaluation Criteria

Criteria	Description
Generalisations	DRA is general in the sense that it is not fixed to one situation or environment.
	DRA can adapt to different situations and be used with different technology stacks.
	DRA is instantiable and applicable to a class of problem situations.
Usefulness	DRA is useful for DevOps adoption in any context
	DRA is useful for application (e.g. IoT applications) deployment to multi-cloud
	DRA reference architecture is a useful template solution for the research gaps
Novelty	DRA offers new knowledge based on DevOps practices.
	DRA offers CI broker to host multi-cloud deployment configuration
	DRA avoids vendor lock-in using the CI-Broker mechanism
Coverage	DRA provides sufficient explanation about DevOps adoption
	DRA provides setup and configuration guidelines for DRA instance pipelines
	DRA provides setup and configuration for IoT-devices (hardware)
Reusable	DRA can be replicated and reconfigured using a different combination of DevOps tools
	DRA design can be reused to create instances for a class of problem situations.
	DRA is reusable for different application-types (IoT, Web, etc.)

3.2.5.2. Survey Design

The surveys conducted in this research use the ratings described in Table 3.3. The ratings transform the participants' qualitative responses to the survey questions into numerical data (quantitative ratings). The following rating table was used in the industry field survey and the teaching case study surveys.

Table 3. 3: Survey Ratings

Qualitative Ratings	Quantitative Ratings
Strongly agree	5
Agree	4
Average	3
Disagree	2
Strongly disagree	1

The qualitative ratings were transformed into numerical data to help with the quantitative analysis of the surveys. The qualitative ratings in Table 3.3 are explained as follows:

- **Strongly agree:** The participants strongly agreed with the statement.
- **Agree:** The participants agreed with the statement.
- **Average:** The participants somewhat agreed with the statement.
- **Disagree:** The participants disagreed with the statement.
- **Strongly disagree:** The participants strongly disagreed with the statement.

The surveys followed a commonly used structure (Hyndman 2008):

- **Planning a survey:** Outline the survey objectives (purpose, need, knowledge requirements).
- **Design the sampling procedure:** Identity the target participants (ethical considerations are required).
- **Select a survey method:** Data collection plan (online method was used in this research).
- **Develop the questionnaire:**
  - SEP SFS questionnaires were predefined by UTS.
  - INP SFS questionnaires were developed by the researcher for the capstone project.
  - Industry survey questionnaires were developed by the researcher using artefact evaluation criteria (Prat, Comyn-Wattiau & Akoka 2014).
- **Conduct the survey:** Execute the survey effectively in a fixed period.
- **Collect and analyse the data:** The surveys provided quantitative and qualitative data. The surveys data analysis is composed of two main steps:
  - Survey quantitative evaluation
  - Survey qualitative evaluation.

3.2.5.2.1. Survey Quantitative Evaluation

The data generated from the surveys (industry and teaching) were categorical. The participants in the surveys contributed their responses to the survey questionnaires as qualitative ratings (see Table 3.3). This research used statistical formulas to make sense of the survey data. Statistical formulas are better suited to provide analysis of a survey's numerical data. According to Hyndman (2008), 'statistics is the study of making sense of data'. The statistical formulas used to analyse the survey data are explained in Equations 3.1–3.3.

Equation 3.1: Chi^2 Formula

Chi^2 Statistical Formula
Chi^2 or $X^2 = \sum \frac{(O-E)^2}{E}$ (O = frequency and E = expected value) (p-value < 0.01) E = ∑O/N (O = frequency and N = total number of observations) The p-value determines if the null hypothesis H_0 is accepted or rejected based on a critical value $\alpha = 0.01$ If p-value < α, then H_0 is rejected and H_1 is accepted, and there is a positive association between the test variables (DRA models) and the evaluation criteria (see Table 3.4). [If p-value < 0.000ε (ε is a small number), then p is mathematically corrected to: p < 0.001] **H_0 (null hypothesis):** there is no association between the test variables and the evaluation criteria **H_1 (alternative hypothesis):** test variables and the evaluation criteria are positively associated

Equation 3.2: Average and Above Frequency (AAF) Formula

AAF Formula
AAF = ∑Frequency (Ratings >= 3) AAF is the sum of all participants responses [Average (3) + Agree (4) + Strongly Agree (5)]

Equation 3.3: Average and Above Percentage (AAP) Formula

AAP Formula
AAP = ∑ Percentage (ratings >= 3) AAP is the sum of all percentages of responses [Average (3) + Agree (4) + Strongly Agree (5)]

3.2.5.2.2. Survey Qualitative Evaluation

The qualitative data collected in the industry survey were analysed using the hypothesis confirmation general technique of analysis (Runeson & Höst 2009). The hypotheses were the artefact evaluation criteria (Prat, Comyn-Wattiau & Akoka 2014) (see Table 3.4). Participants' feedback was cross-examined against the evaluation criteria by highlighting the occurrences of the criteria in the text. The industry feedback was organised into tables. The analysis tables include an explanation column about each item of feedback and a reference column to identify the criteria in each item of feedback. Feedback from the teaching case study surveys is quoted to reflect the participants' opinions about the subjects' (INP and SEP) contents and their overall experience. Feedback from the teaching surveys added further opinions about the usefulness of the DRA for teaching.

Table 3. 4: Survey Evaluation Criteria

Criteria	Description
Generalisations	DRA is general in the sense that it is not fixed to one situation or environment.
	DRA can adapt to different situations and be used with different technology stacks.
	DRA is instantiable and applicable to a class of problem situations.
Usefulness	DRA is useful for DevOps adoption in any context
	DRA is useful for application (e.g. IoT applications) deployment to multi-cloud
	DRA reference architecture is a useful template solution for the research gaps
Coverage	DRA provides sufficient explanation about DevOps adoption
	DRA provides setup and configuration guidelines for DRA instance pipelines
	DRA provides setup and configuration for IoT-devices (hardware)
Relevance	DRA is relevant for deployment IoT applications on the multi-cloud at the industry level.
	DRA framework new knowledge is relevant for teaching, industry and research.
	DRA is relevant for organisations seeking to improve agility using a DevOps approach.
Importance	DRA enables end-to-end automation process and allows decentralised control.
	DRA uses integrated DevOps and cloud tools to support IoT apps process.
	DRA models are a high-level design that can be replicated for any software application.

3.2.5.2.3. Survey Questionnaire Development

The industry survey evaluation criteria were used in the quantitative and qualitative evaluation. The survey evaluation criteria were designed for the DRA artefact using a well-known artefact evaluation process (Prat, Comyn-Wattiau & Akoka 2014). The survey evaluation criteria were used to develop the survey questionnaires for the industry survey. The survey questionnaires were developed to evaluate the DRA design models (see Chapter 4) both individually and overall (see Appendix D). The INP SFS (teaching capstone survey) questionnaires were developed by the researcher following similar ideas utilized by UTS SFS questionnaires. The SEP-SFS questionnaires were developed by UTS-FEIT and were used in FEIT subjects every semester (see Chapter 5, Section 5.5).

3.2.6. OUTPUT

The thesis result is a comprehensive DRA framework for IoT application deployment to the cloud/multi-cloud. The DSR outcome (see Figure 3.8) is composed of two outputs:

- DRA framework.
- Project publications.

The DSR output step presented the DRA framework. The DSR output step also discussed the project limitations and the possible future research for DRA.

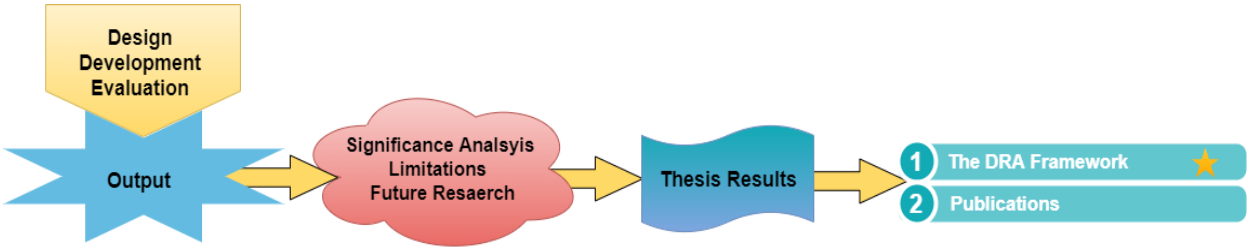


Figure 3.8: Output

3.3. RESEARCH INSTRUMENTS

This section presents the various instruments used in the construction of the proposed DRA framework. The research instruments implemented in this thesis are resources, development methods, experts and industry feedback, and research ethics.

3.3.1. RESOURCES

The essential resources used for the design, development and evaluation of the proposed DRA are mapped in Table 3.4. The resources used to develop and evaluate the DRA framework are a combination of DevOps practices and tools (see Chapter 2, SLR Results), software development and cloud tools, IoT hardware (devices and sensors), and custom survey and case study templates. The integration of the DevOps practices and tools with the cloud services, and with the IoT hardware indicates that the DRA framework development process aims to address the research question and provide a practical working solution to the research gaps determined by the challenges of the adoption of DevOps for software projects.

Table 3. 5: Resources

Resources	Description	Reference
Data	- Background and related work - SLR data review and analysis - SLR results	- Section 1.1 - Section 2.3 - Section 2.4
Method	- DSR method for DRA design, development and evaluation	- Section 3.2
DevOps concepts	- 10 concepts used to create DRA conceptual model	- Table 2.9
DevOps practices	- 20 practices used to create DRA logical model	- Table 2.10
DevOps tools	- Selection of tools used to create DevOps pipeline instances	- Tables 2.12–2.22 - Table 4.6
Software	- Development IDE (Netbeans, VS Code) - Programming languages (Java, JavaScript, Python)	- Table 4.9, Table 4.10 and 4.19
Cloud	- PaaS, IaaS, SaaS (Heroku, AWS, GAE)	- Table 4.7
Hardware	- Raspberry PI board - IoT sensors (light sensors, motion sensors)	- Table 5.4
Empirical evaluation	- Online field survey template - Case study template	- Appendix D - Appendix G

3.3.2. DEVELOPMENT PROCESS

This thesis uses an iterative development process using guidelines described in the DSR method (see Section 3.2). The DRA development process is composed of the following steps:

- Research and data collection from selected studies related to the research topics (see Background and Related in section 1.1 and SLR in chapter 2).
- Address RQ_1 and outline the research gaps using the SLR results (see Chapter 2).
- Outline and explain the DSR used in this research (see Chapter 3). The DSR uses the SLR results and the Chapter 1 analysis as initial data.
- Design and develop the DRA framework (see Chapter 4). This step includes the following items:
  - Define the general characteristic terminologies of DRA.
  - Design the architectural model of DRA.
  - Develop DRA components (DRA instances, IoT application, IoT network).
- Create an online survey (see Appendix D).
- Create an implementation process for the DRA (see Chapter 4).
- Create a case study template (see Appendix G).
- Conduct an empirical evaluation for the DRA (see Chapter 5).
- Outline the DRA limitations (see Chapter 6).
- Outline the DRA future research ideas (see Chapter 6).
- Outline the thesis contributions (see Chapter 6).

3.3.3. EXPERTS AND INDUSTRY FEEDBACK

This section describes the methods used to collect feedback from experts, researchers and industry experts. To collect feedback and opinions, this research used two types of templates:

- online field survey (see Appendix D)
- case study template (see Appendix G)

The case study template was used in the industry case study and research case study. The collected feedback from the case studies was used to determine the DRA relationship with the evaluation criteria (see Table 3.2).

The online survey contained questionnaires for experts and researchers to complete regarding the DRA architectural design. The survey also collected feedback about the framework's operational capability for IoT application deployment to the multi-cloud. The survey generated qualitative and quantitative data. Participants' feedback was analysed to determine the DRA relationship with the evaluation criteria (see Table 3.4). The quantitative data were analysed using statistical formulas (see Equation 3.1-3.3). The anonymous details of the data collection are as follows:

Industry case study participants: 1 (conducted at the organisation's premises by industry experts in the area of software engineering—in particular, in the areas of DevOps, agile, cloud and IoT).

Research case study participants: 1 (conducted at DigiSAS Lab with the help of the lab director, who was an expert in software engineering and agile).

Survey participants: 82 (offered online to industry experts in the area of software engineering—in particular, in the areas of DevOps, agile, cloud computing and IoT).

Teaching case study participants: 208 (offered online to the students of SEP48440 and INP31261 (Inter-Networking Project) at the UTS School of Software).

3.3.4. Research Ethics

Formal approval was obtained from the UTS Research Ethics Committee in compliance with the research ethics policies of the University of Technology Sydney. The approval document can be found in Appendix A. The research did not raise any ethical issues. A formal consent letter (see Appendix C) was sent to each participant. The participants were free to withdraw from the research at anytime, and could contact the supervisor or the university. Additional forms that provided information about the online survey and the DRA framework were also sent to willing participants, along with the consent form (see Appendix B). The purpose of these forms was to provide transparent details about the project, the survey questionnaires, the anonymity of the data collection, and storage.

3.4. Summary

This research was conducted to develop a new framework—the DRA framework—for IoT application deployment to the cloud/multi-cloud by using a constructive and iterative DSR method. The DSR used in this thesis was established on guidelines published by Gregor and Hevner (2013) and Peffers et al. (2007). This chapter presented the resources and development process used to construct the DRA, and it outlined the evaluation methods used in the empirical evaluation to obtain experts' feedback. The empirical evaluation was conducted by involving practitioners and experts from the software industry to acquire information regarding the applicability, novelty, relevance and usefulness of the DRA framework. The DRA is presented in detail in chapter 4 and evaluated in chapter 5.

Chapter 4: DevOps Reference Architecture Framework

This chapter presents the new DRA framework, which is the main contribution of the research. The DRA framework is a practical solution to the research question identified in chapter 1. The DRA framework was developed using the well-known DSR method discussed in chapter 3. The DRA framework aims to assist in the adoption of the DevOps to support IoT application deployment to the multi-cloud, and to address the research gaps identified in chapter 2. The framework is composed of three main components: 1) framework characteristics; 2) framework architecture; and 3) framework composition. The DRA composition incorporates the steps required to create instances of the framework. This chapter presents two instances of the DRA: DRAv1.0 (single cloud) and DRAv2.0 (multi-cloud). The DRA instances are evaluated in chapter 5, using an empirical evaluation. This chapter also includes implementation and case study templates, which can be used by experts and organisations to implement and evaluate the DRA framework in their development contexts.

4.1. DRA OVERVIEW

This thesis has observed that organisations want to determine how DevOps can be adopted to enable IoT application deployment to the cloud/multi-cloud. The analysis and review of relevant background studies and related work identified the research problem that led to the main research question (see Chapter 1). The first part of the research question was answered in the SLR in Chapter 2, which has been previously published (Ghantous & Gill, 2017). The SLR also highlighted vital research gaps (see Table 2.25) that organisations want to address. The second part of the research question aligns with the research gaps identified in the SLR in Chapter 2.

This chapter presents the new DRA framework (see Figure 4.1), which addresses the second part of the research question (see Table 1.2). The DRA has been designed and developed using a well-known DSR method explained in Chapter 3. The DRA provides comprehensive knowledge on deploying IoT applications to the cloud/multi-cloud, avoiding vendor lock-in caused by application provisioning and database hosting. The DRA implementation process needs to take into account the connectivity and interactions of IoT applications and IoT devices. The DRA also supports agile, adaptive implementation using the DevOps approach. The new DRA framework is composed of three main components:

- **Framework characteristics:** The DRA characteristics are general terminologies that can be used to create reference architecture founded on the DevOps concepts and cloud services. The DRA characteristics are not fixed to a particular context but can be used in numerous development contexts.
- **Framework architecture:** The DRA architecture is founded on the DRA characteristics. Organisations and experts can use the key elements of the DRA characteristics to design

reference architecture. The DRA architectural model is a general design that can be instantiated to fit the context of the development environment.

- **Framework composition:** The composition incorporates the essential components required to instantiate the DRA. The composition incorporates the DRA resources and configuration needed to create the framework output instances. This chapter presents two instances of the DRA model: DRAv1.0 and DRAv2.0. DRAv1.0 enables software application deployment to the single cloud, while DRAv2.0 enables IoT application deployment to the multi-cloud. DRAv1.0 and DRAv2.0 use the DevOps approach and the cloud/multi-cloud (services, infrastructure) identified in the SLR (see Chapter 2). The application of the DRAv2.0 instance, which is created using the DRA architecture model, was published in Ghantous and Gill (2018).

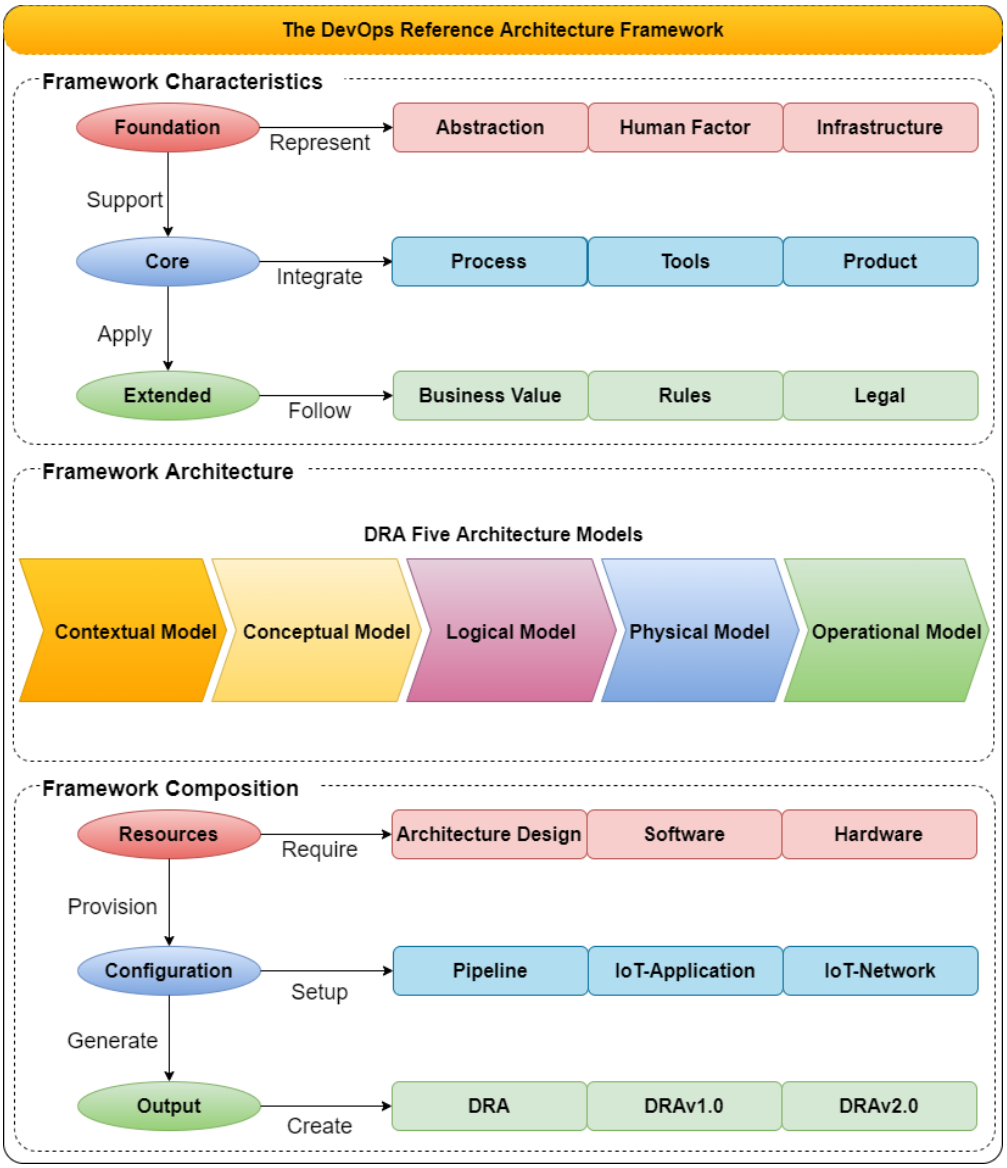


Figure 4.1: DRA Framework

4.2. DRA FRAMEWORK CHARACTERISTICS

The DRA characteristics (see Figure 4.2) represent the fundamental elements or attributes of the framework. They are defined using common terminology following the guidelines published by Berger, Häckel and Häfner (2019) and Nickerson, Varshney and Muntermann (2013). The DRA characteristics can be used to create a reference architecture design model in the organisation's context. A reference architecture model illustrates the relations of the characteristics. The reference architecture illustration of the framework characteristics uses DevOps concepts and cloud services. The framework characteristics are organised into three categories:

- The foundation is composed of the human factor, infrastructure and abstraction elements.
- The core is composed of tools, process and product elements.
- The extended is composed of business value, rules and legal elements.

The framework characteristics presented in this chapter are a result of the synthesis of the information gathered in Chapters 1 and 2. Figure 4.2 presents an example of the associations between DRA characteristics in a harmonious relationship. The DevOps concepts identified in Chapter 2 and the cloud services identified in Chapters 1 and 2 can be used in an organisation's context to create a reference architecture design founded on the characteristics illustrated in Figure 4.2. The DRA characteristics model is not fixed to a particular instance (or example view) but can be associated with numerous instances or examples to fit organisations' contexts. In Figure 4.2, the associations between DRA characteristics can be explained and connected to the relevant DevOps concepts (see Table 2.9 labelled C[index]) and cloud services as follows:

- The DevOps team represents the human factor, which is affected by geography, team communication, knowledge background, resource-sharing and team collaboration. DevOps concepts (C1, C5, C7) support this association.
- The DevOps team requires a development platform. Cloud infrastructure and services (PaaS, SaaS, IaaS) provide the necessary platform.
- The DevOps team applies the rules within the approved legal boundaries of the organisation's development context.
- The DevOps team conducts a process that is supported by the infrastructure (cloud or multi-cloud). The DevOps team uses abstract architecture design to provide a development environment (or workspace). The development environment (workspace) can support multiple processes initiated by the team. DevOps concepts (C4, C5, C6, C7, and C9) support this association.
- The DevOps team uses DevOps tools and cloud services to create a product. The selected tools are integrated into the development workspace. DevOps concepts (C1, C2, C3, C6, C7, C9, and C10) support this association.
- The DevOps team delivers a product, which is the output of the completed processes. The team uses the development workspace to design and develop a product, and uses the tools

to create a product and the infrastructure to deploy a product. DevOps concepts (C2, C3, C6, C8, C9, and C10) support this association.

- The product has a business value that is determined by the organisation. The value of a product should cover the cost of development and resources. The business value of a product is bound to the rules of the organisation context. The business value also follows the legal attributes set up by the organisation at the start of the process.

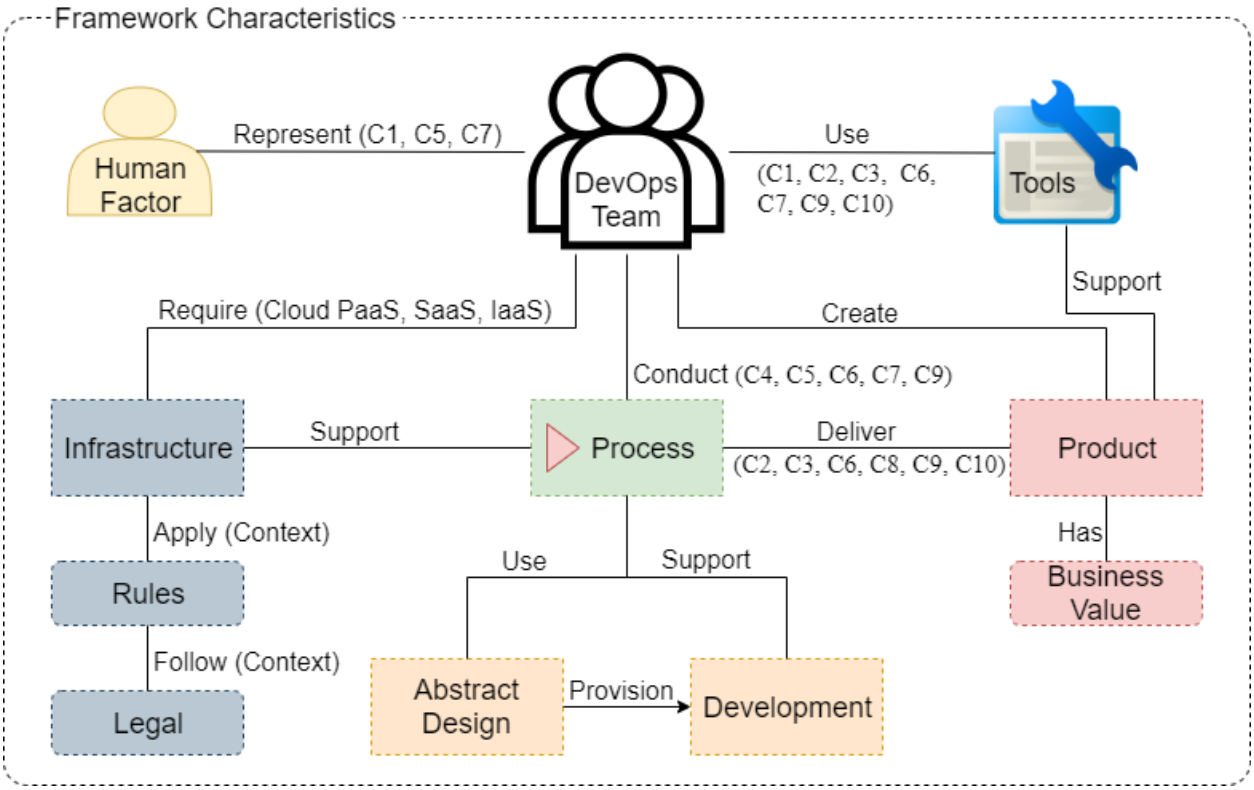


Figure 4.2: Framework Characteristics (Example View)

4.2.1. ABSTRACTION

Abstraction represents a logical view of the software development structure (Analyti et al. 2007; Theodorakis et al. 1999). The abstraction characteristic incorporates several mechanisms such as people-oriented, service-oriented, tools-oriented and process-oriented (see Figure 4.3). A framework abstract design may combine more than one abstraction mechanism. The mechanism combination is used to create the design model for the development environment (or workspace). The reason for including the abstraction in the DRA framework is based on four key factors. First, the DRA is people-oriented because it supports the human factor (people) using DevOps concepts and practices, as indicated in Chapters 1 and 2. Second, the DRA is service-oriented because it uses cloud services (PaaS, SaaS, IaaS) to set up the development workspace. Third, the DRA is tools-oriented because it uses DevOps tools to create an operational pipeline in the development workspace. Fourth, the DRA is process-oriented because it enables automated and

integrated processes to accomplish the deployment of a software application product. However, organisations are not tied to any specific abstraction mechanism mentioned in the example view (see Figure 4.3). Organisations may use, combine or integrate other abstraction mechanisms to fit the purpose of the software project. Thus, the abstraction element is linked to the human factor, process, infrastructure and product. The DRA architecture is not fixed to a particular combination of abstraction mechanisms.

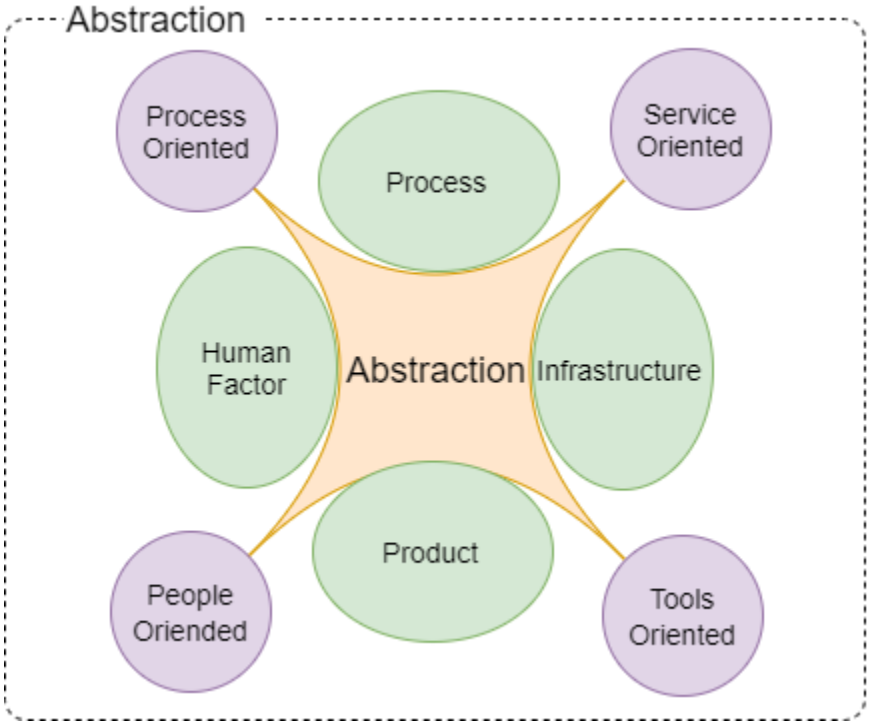


Figure 4.3: Abstraction Characteristic (Example View)

4.2.2. HUMAN FACTOR

The human factor (see Figure 4.4) is an essential element in the DevOps framework characteristics. It was observed in Chapter 1 (background and related work analysis) and Chapter 2 (SLR analysis) that the DevOps approach supports people or DevOps teams. In an organisation, a DevOps team may involve many individuals that have one or many roles. People in a DevOps team may belong to different types of communities and have different types of skills (social skills, development skills, management skills, technical skills). Regardless of the people's skills, location, languages and knowledge base, a DevOps team is expected to be involved in the entire product deployment lifecycle (or development chain), which includes code management, build, testing, deployment and monitoring. To increase the positive effect of the human factor on the project and improve people's experiences, the DRA framework uses the DevOps approach, which provides concepts, practices and tools (see Chapter 2) that enable better communication and collaboration in a team. While the DRA uses the DevOps approach tools to enable communication and collaboration, it is not tied to a fixed combination of tools.

Instead, the DRA may be implemented using numerous combinations of tools that fit the context of the organisation or project. Hence, the human factor characteristic in the DRA is affected by several entities (technology-based or non-human-based). However, the human factor in the DRA is not fixed to a particular technology. Table 4.1 incorporates the technology-related and non-technology-related entities that affect the human factor.

Table 4.1: Human Factor Entities

Entities	Description
Location and communication	- People may belong to different types of communities and speak different languages. People in a team should have the ability to work in a communications-oriented environment enabled by DevOps practices and tools.
Access and collaboration	- People may be located in different cities or countries. People in a team should have the ability to work in a collaboration-oriented environment enabled by DevOps practices and tools.
Repository	- People are required to synchronise their code in the development chain. People should have the ability to use DevOps approach practices and tools to enable code management.
Role and skills	- People may have different types of skills. A DevOps team should have the ability to use the skills of its individuals in properly assigned roles. People in a team should be encouraged to support their colleagues for the benefit of the project. - Organisations should assign roles (developer, manager, consultant, coach) according to their skills in a project.
Transparency	- People in a team should be encouraged to act with professionalism and transparency.
Sharing	People in a team should have the following options enabled in sharing: - Provide fast global resources exchange - Provide maximum possible free resources for DevOps team - Enable sharing notifications - Enable team global access to resources
Security	- People in a team should work in a secure environment. Security measures should cover resource access and role access within the organisation.

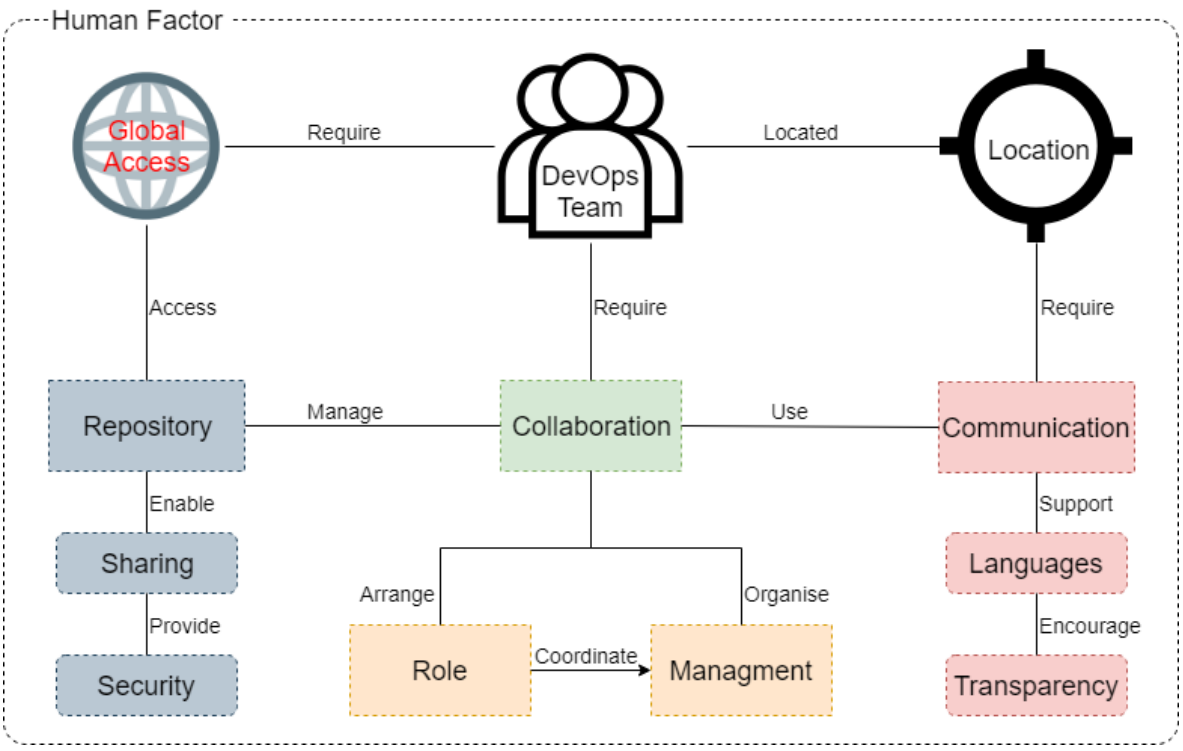


Figure 4.4: Human Factor Characteristic (Example View)

4.2.3. INFRASTRUCTURE

The infrastructure characteristic is also a foundational element in the framework's construction. As observed in Chapter 1 (background and related work analysis) and Chapter 2 (SLR Results, Section 2.4), the cloud seems to provide the required infrastructure for the DRA framework. Cloud IaaS, PaaS and SaaS offer virtual servers, resource-sharing, adaptive deployment options, software services and standard security measures. The cloud supports automation, CI, continuous deployment and monitoring. In the DRA framework, DevOps and cloud services integrate into architecture. The DRA cloud infrastructure provides the development platform for DevOps teams to use tools and create the product. The product is deployed to the virtual cloud servers. The infrastructure services are listed in Table 4.2 and shown in Figure 4.5.

Table 4.2: Infrastructure Services

Layers	Features
IaaS	- Global access, share and pay-as-you-use option - Provide virtual servers with assured availability (no downtime) - Enable scalability (scale applications up or down on demand) - Enable load balancing (distribute application traffic across used servers)
PaaS	- Provide development environment, tools and support most programming languages - Enable auto-scaling of applications - Support automated application deployment, testing, monitoring and logging - Provide database management system
SaaS	- Scalability of product and option to integrate with SaaS software services - Enable interoperability of applications - Offer configuration and setup options for various applications

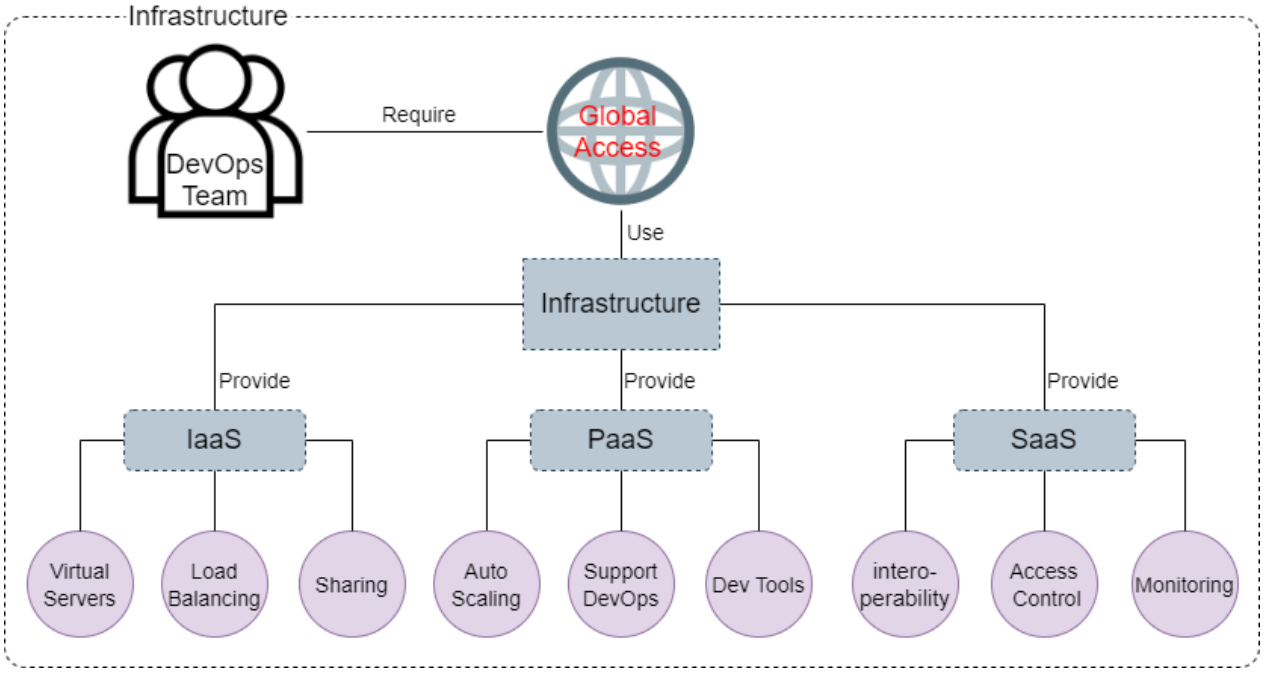


Figure 4.5: Infrastructure Characteristic (Example View)

4.2.4. PROCESS

The process characteristic is an essential element in the DRA framework. A process is composed of different sub-processes such as design, development, deployment and business. Processes are compliant to rules and legal requirements that represent the policy in an organisation context. The process characteristic can be represented by attributes explained in Table 4.3. A process has a unique process ID and a lifecycle that runs for a specific time and for a particular task (each sub-process has its lifecycle). A process lifecycle is the combination of lifecycles of all sub-processes. A process can be categorised as a software process, design process, business process or deployment process (see Figure 4.6). The software process is the development procedure used to create a software component. The design process is used to create a product architecture

model. The deployment process is a combination of the practices and functionalities required to create and deploy a product. The process is the progression of the product from development to delivery. DevOps teams perform quality assurance checks (performance, usability, sustainability, availability) as part of DevOps retrospective application review (Ghantous & Gill 2018; Perera, Silva & Perera 2017). The quality assurance can be performed for each sub-process in a particular process.

Table 4.3: Process Types

Attributes	Description
Design process	- Process ID and Name of the task [component] - The conceptual relationship between entities - Physical abstract design for entities - Logical, abstract design for entities - The operational model for the deployment process - Process lifecycle [duration] - Degree of DevOps practices adoption
Software process	- Process ID and Name of the task [component] - The development environment and tools workspace [DevOps tools] - Degree of human factor effect - Process lifecycle [duration] - Degree of DevOps practices adoption
Deployment process	- Process ID and Name of the task [component] - Deployment environment [DevOps pipeline] - Process monitoring - Software testing [Retrospective review: application health and performance] - Process lifecycle [duration] - Degree of human factor effect - Degree of DevOps practices adoption
Business process	- Process ID and Name of the task [product] - Virtual environment for the product (Cloud: IaaS, PaaS, SaaS) - Retrospective review: Product sustainability, availability, and usability - Process lifecycle [duration] - Degree of human factor effect

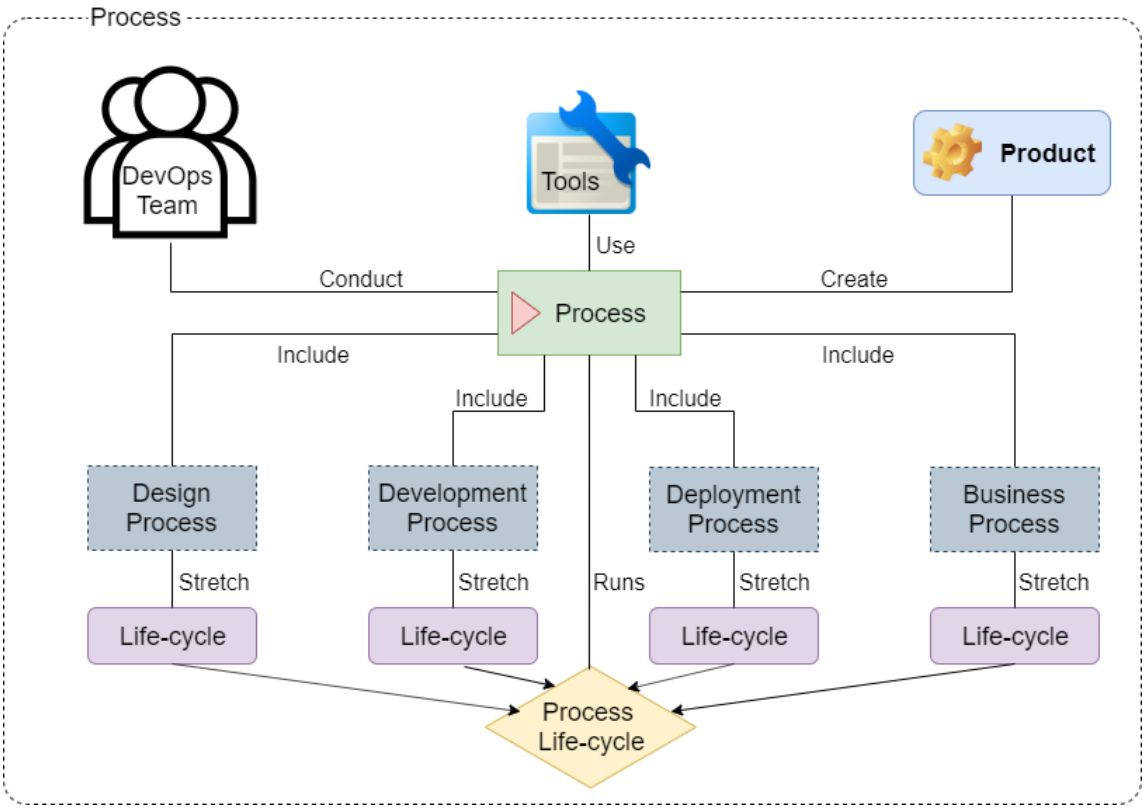


Figure 4.6: Process Characteristic (Example View)

4.2.5. TOOLS

The tools characteristic refers to DevOps and non-DevOps tools (cloud/multi-cloud services) (see Chapters 1 and 2) that are used in a process to create a product (see Figure 4.7). The tools characteristic is used in the development workspace founded on the abstract design of the framework. A configured development workspace may contain many tools, which can be used to support multiple processes. The tools should enable the DevOps concepts identified in the SLR Results in Chapter 2, Section 2.4.

In the DRA framework, tools are used to create an operational model to deploy and deliver a product (IoT application). However, the framework is not fixed to a particular set of tools. Organisations may use different tools suitable for the development context of the project. When selecting tools, organisations take into account the DevOps team's knowledge base and skills, the product type and requirements, the development environment and the workspace. Thus, the tools characteristic is linked to the human factor, process and product characteristics.

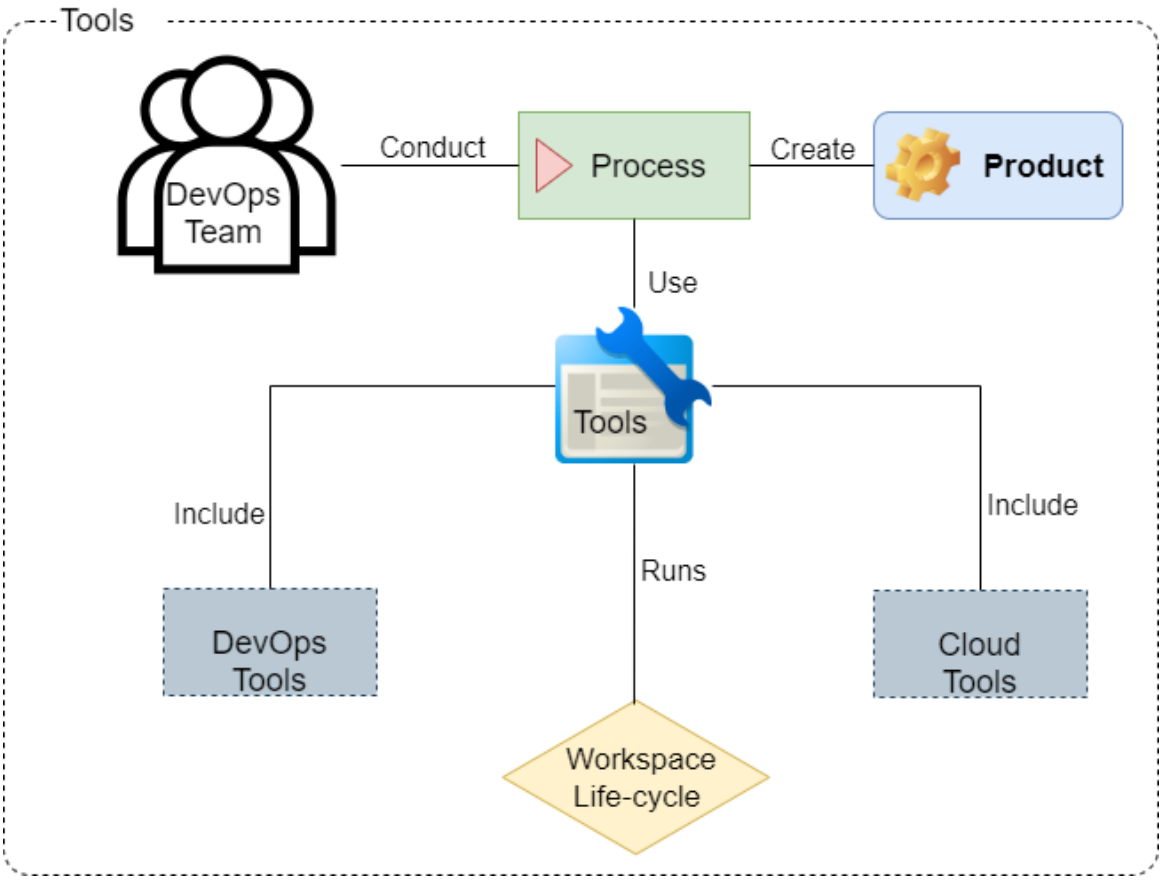


Figure 4.7: Tools Characteristic (Example View)

4.2.6. PRODUCT

The product characteristic presents the executable project output (e.g., IoT application) (see Figure 4.8). The product characteristic is the output of a process or multiple processes conducted in the development workspace. The proposed DRA framework deploys and delivers a product using an operational model composed of several tools. A product can be of different types and may contain one or many sub-products. A product is described in Table 4.4.

Table 4.4: Product Entities

Description
- Product ID and name (this rule also applies to the product's components) - Documentation (description and purpose) - Abstraction (architecture design model) - Development (technologies, tools, workspace) - Environment (deployment, synchronisation, automation) - Quality (monitoring logs, testing logs, performance logs, continuous planning) - Lifecycle (duration of development, duration of deployment, duration of delivery) - Business value (value added to a product or sub-product) - People (DevOps team creates a product, users access a product) - Risks and constraints (product disclosure for risk management, scope, limitations)

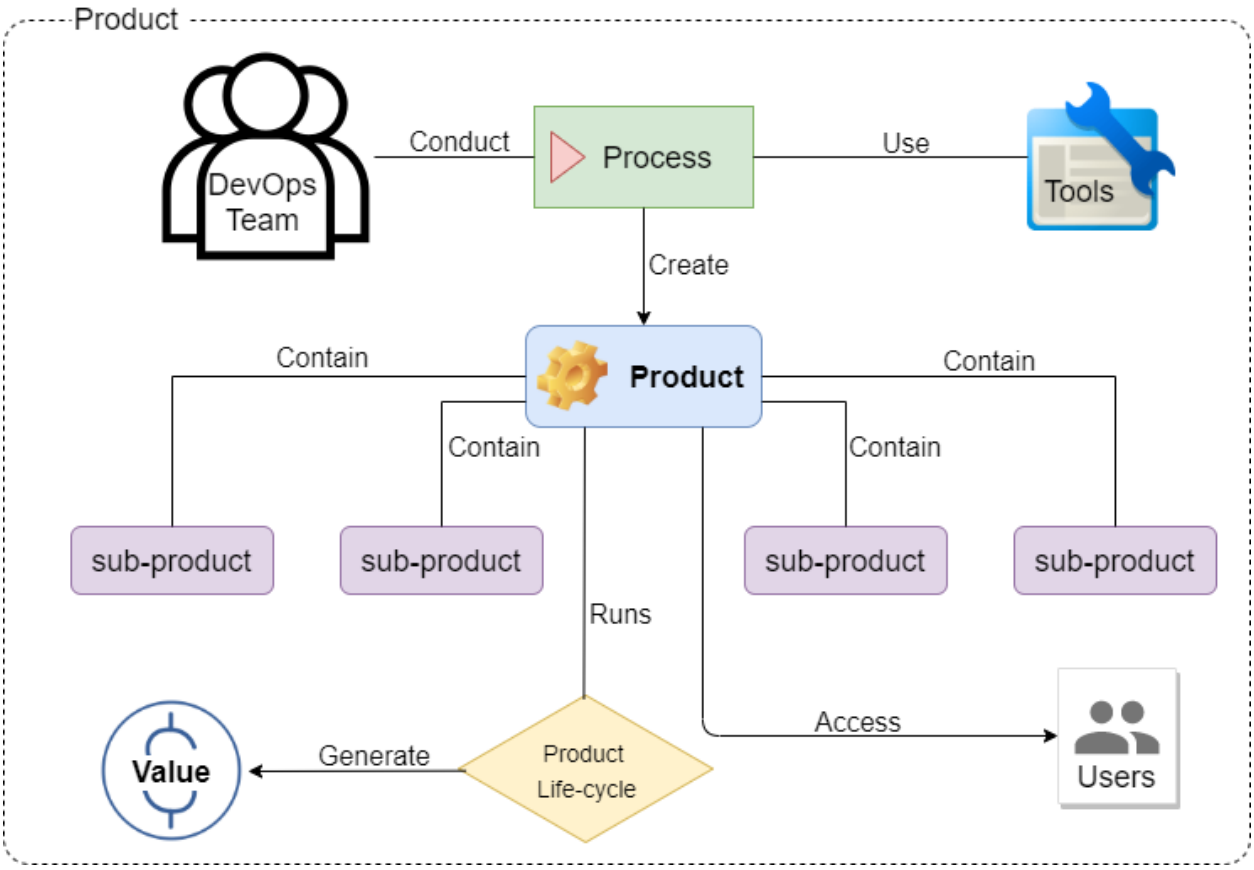


Figure 4.8: Product Characteristic (Example View)

4.2.7. BUSINESS VALUE

The business value characteristic has not been investigated to any great extent in this thesis. However, it should be included in the characteristics because it affects the deployment and delivery of the product (e.g., IoT application) to users. Business value is generated by using other characteristics such as people, tools, processes, infrastructures and products. Thus, the business value characteristic is explicitly included in the extended characteristics category of the DRA framework.

4.2.8. RULES

The rules characteristic is explicitly included in the extended characteristics category of the DRA framework. Business rules are a form of knowledge that states the guidelines being adopted in the organisation (de AR Gonçalves et al. 2011). Business rules are an essential concept and can be viewed as general declarations about the organisation's means of conducting business. Business rules affect the delivery of a product to users (deployment of IoT application) because a product is the result of a process or multiple processes, and it is associated with abstract design and tools (Cysneiros, de Macedo-Soares & do Prado Leite 1999).

4.2.9. LEGAL

The legal characteristic is explicitly included in the extended characteristics category of the DRA framework. The legal agreement and governing requirements may also affect the software development process and product (e.g., IoT application) delivery. This thesis has established that the DRA framework is not fixed to any organisation context but can be instantiated to suit the organisation's project. The DRA is implemented for organisations using the framework characteristics explained previously (abstraction, infrastructure, process, people, tools, product). Given that the DRA is not fixed to a particular context, it is the local organisation's responsibility to organise the legal regulation to manage the development process (Islam, Mouratidis & Jürjens 2011). The agile manifesto (Fowler & Highsmith 2001) provides a convenient reference for organisations to implement their rules and regulations. The following contract laws are applicable when creating a product (Hoeren & Pinelli, 2018). This list can be used as a generic set of rules to deploy and deliver a product (e.g., IoT applications) using a DRA instance.

- Specification of the accepted work: Specify the product, tools, abstract design, people and infrastructure.
- Copyright rules: Outline the copyright related to the DRA characteristics.
- Due date and payment date: Set up the product delivery date and business value.
- Risk and constraints: Identify the risks and constraints that may occur during the implementation of the DRA and the deployment of the IoT application (software build and deployment issues, internet access, IoT network configuration, connectivity protocols, vendor lock-in).
- Exclusion of known and not reserved defects: The DevOps team should identify known defects that may obstruct the product's deployment.
- Beginning of intended use of the product: Specify the deployment time and delivery time according to the development schedule of the organisation.
- Cost planning of the product and time constraint: The DevOps team should provide a detailed plan for the project schedule and cost estimation.
- Product development and implementation schedule: The DevOps team should provide a detailed development and implementation plan.
- DevOps team roles and responsibility contracts: Specify the roles of the DevOps team members in the project.
- The development environment and product delivery: Configure and set up the development workspace for the DRA implementation.
- Performance assessment and quality assurance: The DevOps team should monitor the product deployment and ensure the performance of the IoT application.

The legal characteristic may include or exclude different items depending on the current organisation context.

4.3. DRA Architecture Design

The DRA architectural design is a generic design model founded on the DRA characteristics discussed in the previous section. DRA characteristics are common terminologies used to represent the DRA elements (DevOps and cloud/multi-cloud). DevOps elements or concepts were identified in Chapter 1 (background and related work analysis) and Chapter 2 (SLR analysis and results) and were previously published by Ghantous and Gill (2017). The cloud/multi-cloud elements were identified in Chapters 1 and 2. DRA characteristics are used to create a reference architecture design to deploy a product (IoT application) to the cloud/multi-cloud (the primary objective of this research). The DRA framework design is composed of five models: contextual, conceptual, logical, physical and operational (Ghantous & Gill 2018).

4.3.1. DRA Contextual Model

The contextual model for the DRA illustrates the relationship in the context between the research topics (DevOps, cloud/multi-cloud, IoT). The contextual model is a block diagram that shows how cloud/multi-cloud and DevOps assist with the deployment of the IoT application (see Figure 4.9). The research aim (see Chapter 1) and research gap (see Chapter 2) indicated that it is vital to avoid vendor lock-in when deploying IoT applications to the multi-cloud. Thus, a CI broker mechanism is needed to perform the operations required to handle the vendor lock-in issue. The framework's characteristics represent the elements required to create a generic contextual model that is suitable for any organisation's context. For instance, the abstraction element is required to identify the combination of mechanisms used to create the contextual model (process-oriented, people-oriented, service-oriented, product-oriented). The infrastructure element represents the DRA deployment platform, the tools element identifies the DevOps and cloud tools used in the model, the process element represents the application deployment, and the product element represents the IoT application.

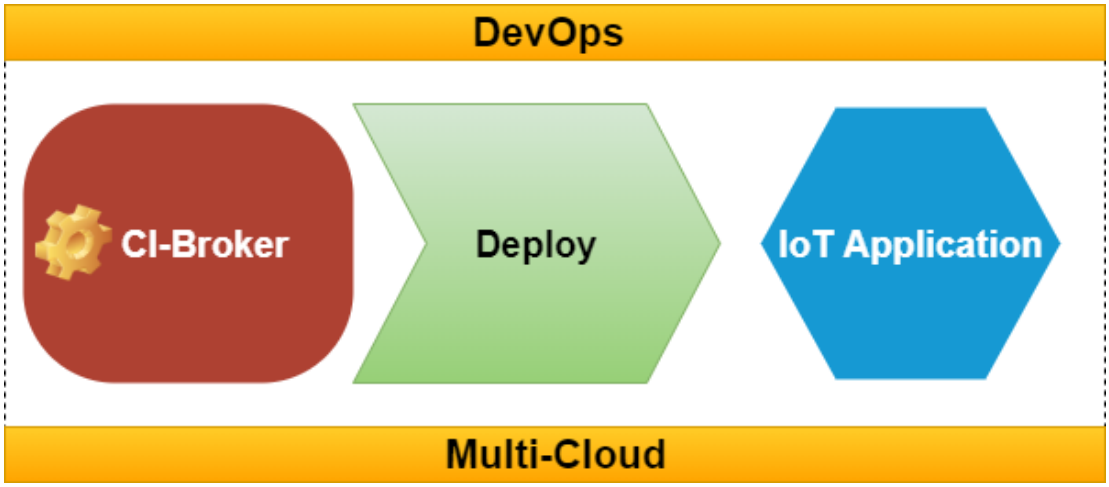


Figure 4.9: DRA Contextual Model

4.3.2. DRA Conceptual Model

The DRA conceptual model (see Figure 4.10) expands the contextual model of Figure 4.9 and incorporates the DevOps concepts and cloud services that are required to enable IoT application deployments. DevOps concepts (see Chapter 2) and cloud services (see Chapters 1 and 2) represent DRA characteristics (see Section 4.2), which are commonly used attributes that can be used to create a reference architecture model. The conceptual model is constructed by replacing the characteristics' elements with DevOps concepts and cloud services. This method indicates that the conceptual model is a generic design that can be applied to any context.

The conceptual model includes a vital mechanism—the CI broker. The CI broker idea was established in the research aim (see Chapter 1) and research gap (see Chapter 2). With the research question in mind, it was essential to devise a procedure to deploy IoT applications to the multi-cloud and avoid vendor lock-in, which occurs when a cloud vendor hosts the deployment configurations and when a cloud vendor hosts the database. The CI broker is an essential part of the DRA conceptual model because it incorporates several operations. For instance, the CI broker enables automation (build, testing, logging, deployment), CI, branching development and automated code synchronisation. Most importantly, it hosts the deployment configurations for the IoT application. The CI broker packages the IoT application in a container and deploys it to the multi-cloud platforms used in the DRA instance. This procedure prevents any of the clouds incorporated in the multi-cloud platform from hosting the IoT application deployment parameters and consequently prevents vendor lock-in.

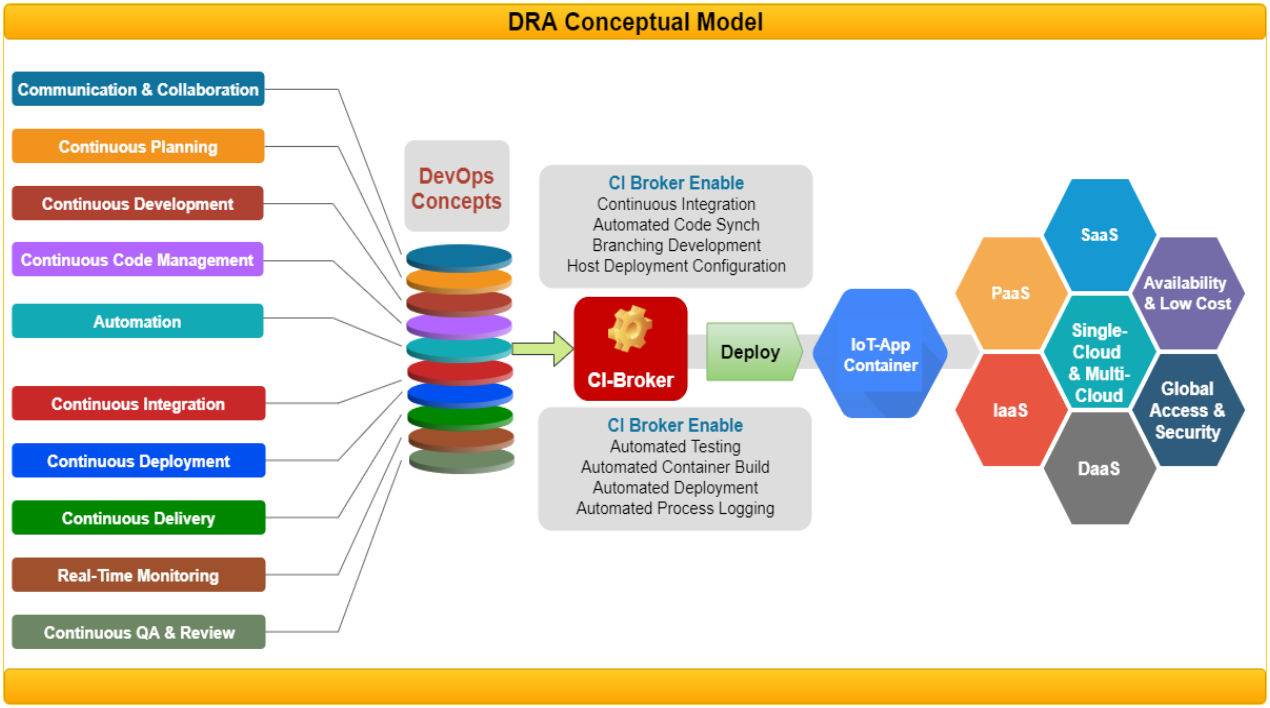


Figure 4.10: DRA Conceptual Model

4.3.3. DRA LOGICAL MODEL

The DRA logical model (see Figure 4.11) expands on the conceptual model of Figure 4.10 and incorporates DevOps practices and cloud services to represent the concepts highlighted in Figure 4.10. The DRA logical model is composed of five components (M1–M5) that represent the DRA instance tiers or phases. Each component includes functions and features related to the required DevOps practices and cloud service in that tier. The components are integrated into a logical view that represents the interactions between these components. The logical model represents the blueprint for physical model instances of the DRA. DRA instances should enable all logical model operations discussed in Table 4.5, which outline the functions and features that represent the DevOps practices and cloud services incorporated in the DRA logical model.

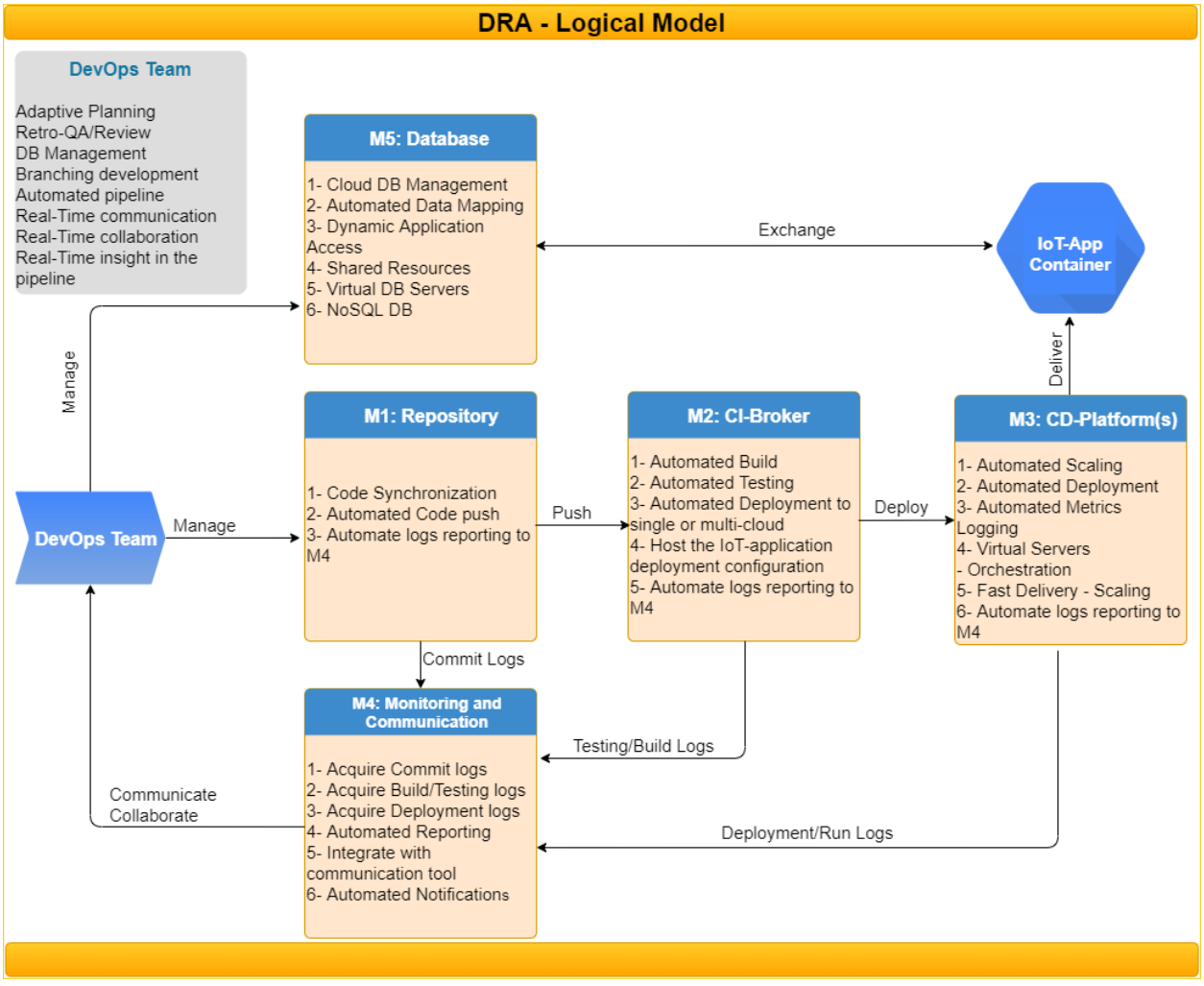


Figure 4.11: DRA Logical Model

The DRA logical model components (M1–M5) incorporate the functions and features (see Table 4.5) that are necessary to conduct the deployment process of IoT applications (product) to the multi-cloud. The logical model features (DevOps practices and cloud services) were identified in Chapters 1 and 2. The logical model features are discussed in Table 4.5, which includes a set of suggested tools that could be used in the DRA instances to enable logical model features. The suggested tools are selected from an extensive list of DevOps tools (Ghantous & Gill 2017).

The logical model components are vital for the construction of the DRA physical model and the creation of the DevOps instances. Notably, two critical components represent the key factors to enable a successful IoT application deployment to the multi-cloud. With the research gap in mind, M2 (CI broker) and M5 (database tier) help the DevOps team avoid vendor lock-in. For instance, M2 operations (features) include the hosting of IoT application deployment parameters, while M5 represents a central cloud database for IoT data. M2 and M5 give the DRA framework the necessary procedure to prevent vendor lock-in when deploying IoT applications to the multi-cloud.

Table 4.5: Logical Model Features

Components	Features	Tools
M1	- Branching development - Automated code synchronisation - Automate logs reporting to M4	- Github - BitBucket
M2	- Automated build - Automated testing - Automated deployment to multi-cloud - Host deployment configuration - Automate logs reporting to M4 - Branching development - Automated code synchronisation	- Codeship - Travis CI - Jenkins
M3	- Automated scaling - Software as services - Virtual servers—orchestration - Automated delivery - Automate logs reporting to M4	- Heroku - Google App Engine - AWS CodeDeploy
M4	- Acquire commit logs - Acquire build/testing logs - Acquire deployment logs - Automated reporting - Integrate with a communication tool - Automated notifications	- Papertrail - Nagios - New Relic - Slack - Slack
M5	- Cloud DB management - Central database - Dynamic application access - Shared resources - Virtual DB servers - NoSQL DB	- MongoDB (mLab) - DB-Maestro - Firebase

4.3.4. DRA PHYSICAL MODEL

The DRA physical model is an implementation of the DRA logical model. It creates a tangible design of the logical components (M1–M5) and their integrations. The operational workflow for the DRA physical model is illustrated in Figure 4.12. This model can be subdivided into three layers:

1. The DevOps team layer is composed of:
   - M1 enables team communication and real-time notification. M1 receives build/test logs from M2 and deployment and performance logs from M4.
2. The cloud layer is composed of:
   - M2 is the CI broker that includes the build, testing, and logging and deployment parameters of the IoT application. M2 is essential to prevent vendor lock-in.
   - M3 is the deployment cloud(s) platform for the application.
   - M4 is monitoring and tracking the progress of application deployment.
   - M5 is the database cloud for IoT data. M5 is essential to avoid vendor lock-in.
3. The user layer represents the user devices used to access and operate the IoT application.

The DRA physical model represents a harmonious integration of tangible DevOps tools and cloud services. The physical model is the template used to create the DRA operational model.

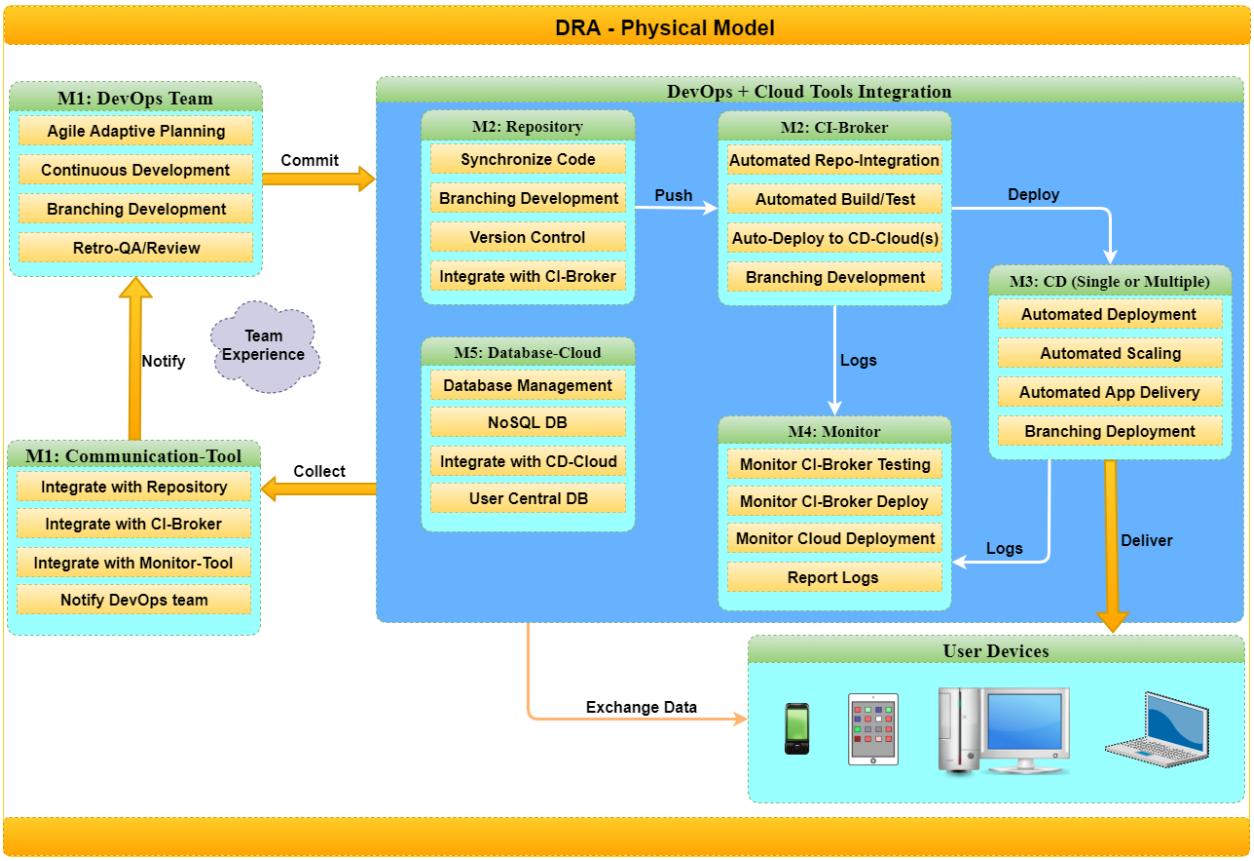


Figure 4.12: DRA Physical Model

4.3.5. DRA OPERATIONAL MODEL

The operational model of the DRA framework is based on the physical model of Figure 4.12 and enables the logical model features (DevOps practices) of Table 4.5. The DRA operational model is configured using an integrated set of DevOps tools (see Chapter 2). It provides automated IoT application deployment to the cloud/multi-cloud (Ghantous & Gill 2018). This model is the application of the logical model that represents the DRA characteristics incorporated in the DRA conceptual model. The DRA operational model is not fixed to a particular set of tools but can be configured using numerous tools to suit the context of the organisation. The DRA operational model enables a fully automated deployment process.

- **Operational model deployment process [M1 →M2→M3→M4→M5 → IoT app]**

The operational model deployment process in a pipeline begins at M1 and ends back at M1, completing a fully automated deployment process circle. The deployment process is as follows:

- The IoT app code is pushed from M1 to M2.
- The M2 model (CI broker) is a CI broker intended for automatically deploying an IoT app to M3. The CI broker includes the deployment configurations of the software applications (e.g. IoT-applications) to avoid vendor lock-in.
- M3 is the deployment platform of the DRA. DRA instances can be represented by a single cloud [DRAv1.0, see Figure 4.13] or a multi-cloud [DRAv2.0, see Figure 4.14].
- The M4 model provides the DevOps team with real-time management and clear insights across the deployment steps using DevOps monitoring and communication tools.
- Users can run or execute the IoT app from any smart device once it is deployed to the cloud/multi-cloud. The IoT app interacts with an IoT network (IoT devices). The data are collected at run-time and saved as NoSQL data entries at M5. M5 represents a centralized database to avoid vendor lock-in.
- IoT application operations logs are recorded in M4 and reported to M1. M1 also sends notifications to the DevOps team about the code synchronisation logs, build logs, testing logs and deployment logs.

- **DRA operational model instances [DRAv1.0 and DRAv2.0]**

This thesis presents two operational model instances: DRAv1.0 and DRAv2.0. The DRA instances are created using the DRA reference architecture models defined in previous sections (contextual model, conceptual model, logical model, physical model). The difference between them is that DRAv2.0 enables deployment to the multi-cloud, whereas DRAv1.0 enables deployment to a single cloud. DRAv1.0 and DRAv2.0 instances can be applied in organisations using a specific implementation, as discussed in Section 4.4. DRAv1.0 and DRAv2.0 instances were evaluated using an empirical evaluation (see Chapter 5).

4.4. DRA FRAMEWORK COMPOSITION

This section presents the DRA composition of the framework (see Figure 4.1). The DRA composition incorporates the essential elements needed to create DRA instances for deploying IoT applications to the cloud/multi-cloud. The DRA composition is founded on the DRA characteristics, which were used to create a comprehensive reference architecture for deploying IoT applications (products) to the cloud/multi-cloud (infrastructure). The DRA composition integrates the DRA architectural models to create practical implementations (instances) that are applicable in the context of the organisation. The DRA composition includes three main components:

- resources (architecture design, software, hardware)
- configuration (Pipeline, IoT application, IoT network)
- output (DRA framework architectural models, DRAv1.0 instance, DRAv2.0 instance).

4.4.1. RESOURCES

The resources element of the DRA composition incorporates the architecture design models presented in Section 4.3, as well as the software and hardware items required to implement the DRA instances. The DRA composition integrates architecture design models to create automated pipeline instances for deploying the software product to the cloud/multi-cloud. The software application (product) interacts with the hardware (infrastructure) and generates data. The data are stored in a cloud database (infrastructure) to prevent vendor lock-in issues.

4.4.1.1. Architecture Design

The architecture design represents the integration of the DRA models discussed in Section 4.3 to create a practical implementation (instance) in the context of the organisation. The architecture design is constructed to accommodate and support the framework characteristics explained in Section 4.2. The DRA architecture is not fixed to a particular instance. Instead, numerous pipeline instances (e.g., Figures 4.13 and 4.14) can be created using the DRA design models.

4.4.1.2. Software

The software element represents the product characteristic in the framework. The software element is an IoT application to be deployed to the cloud/multi-cloud in the context of the organisation.

4.4.1.3. Hardware

The hardware component is used to interact with the IoT application deployed to the cloud/multi-cloud. The hardware represents the IoT network used in this project to demonstrate the interactions of IoT applications deployed to the cloud/multi-cloud with the IoT devices.

4.4.2. CONFIGURATION

The configuration component represents the DRA composition procedures needed to create the DRA instances, configure the DRA instances' pipelines, develop the product (IoT application) and set up the hardware infrastructure (IoT network).

4.4.2.1. Pipeline

The DRA pipeline is a physical instance of the DRA operational model (see Section 4.3.E). The DRA instance pipeline enables automation and CI of the DevOps tools and cloud services (see Chapters 1 and 2). The DRA instance pipeline is created using the DRA design models defined in Section 4.3. The DRA pipeline application deployment is an end-to-end automated process that enables DevOps practices (see Table 2.10).

Two instances pipelines of the DRA are discussed in this thesis:

- DRAv1.0 instance pipeline: This instance pipeline is founded on the DRA architectural design models and is configured using set integrated tools (DevOps and cloud) to enable application deployment to the single cloud (see Figure 4.13).
- DRAv2.0 instance pipeline: This instance pipeline is founded on the DRA architectural design models and is configured using set integrated tools (DevOps and cloud) to enable application deployment to the multi-cloud (see Figure 4.14).

DRAv2.0 instance (multi-cloud) is an upgraded version of DRAv1.0 instance (single cloud). The DevOps tools used in both versions can be substituted with other tools that perform the same roles. The instances' configurations are explained as follows:

- DRAv1.0 instance pipeline (see Figure 4.13) toolsets used in the research are BitBucket, Codeship, Heroku, Papertrail, mLab MongoDB and Slack. Table 4.6 outlines the DRAv1.0 instance, toolset features and descriptions.
- DRAv2.0 instance pipeline (see Figure 4.14) toolsets used in the research are BitBucket, Codeship, Heroku, AWS CodeDeploy, Google App Engine, Papertrail, mLab MongoDB and Slack. DRAv2.0 instance pipeline is constructed from the same toolset as DRAv1.0 instance pipeline with the addition of two more tools to enable multi-cloud deployment: AWS (CodeDeploy) and GAE (Google App Engine). Table 4.6 outlines the features and descriptions of the DRAv2.0 instance toolset. The pipeline (see Figure 4.14) illustrates the deployment of the IoT app using Codeship (CI broker) to the multi-cloud (Heroku, AWS, GAE). The mLab tool (MongoDB) is used as a central database for IoT data. Hosting the IoT data in a central database prevents the vendor lock-in issue.
- For the DRA framework, the CI broker is an important mechanism. The CI broker enables automated build, testing, logging and deployment. The CI-Broker contains the deployment parameters for the software application (e.g. IoT application); which prevents the vendor lock-in issue raised in the research gaps.

Table 4.6: DRAv1.0 and DRAv2.0 Toolsets

DRAv1.0 Instance Toolsets		
Tools	**Features**	**Description**
BitBucket	- Code synchronisation - Automated code push - Automate commit logs to Slack	BitBucket is a team collaboration and code management tool. It enables code synchronisation and automatically pushes the application to Codeship. It also sends commit-logs to Slack.
Codeship	- Automated build/testing - Automated deployment - Host the deployment configuration of the IoT application - Automate build/testing logs to Slack	It is the CI broker tool and enables automated build/testing for code automatically received from BitBucket. It also enables automated deployment to clouds and sends build/testing/deployment logs to Slack.
Heroku	- Automated scaling - Virtual servers—orchestration - Fast delivery—scaling - Automate deployment logs to Papertrail	Heroku cloud enables automated scaling of the application deployed from Codeship. It enables automated scaling of the app for users. Run-time logs of the application are sent to Papertrail.
Papertrail	- Acquire deployment logs - Automated deployment logs to Slack - Automated notifications	Papertrail monitoring the deployment and execution logs of the IoT application. It sends those logs to Slack.
MLab	- Cloud DB management - Dynamic application access - Virtual DB servers - NoSQL DB - Host the database for the IoT data	MLab is a MongoDB cloud database management service. It enables automated data access and mapping. MLab collects IoT data from the IoT application and stores the data in JSON NoSQL. DevOps team can dynamically manage IoT data on MLab.
Slack	- Automated log management - Automated notifications - Real-time communication (chat, video) - Resource-sharing option - Integration with Codeship and Papertrail	Slack is a communication and collaboration tool that provides DevOps teams with real-time chat/video conference options and enables automated real-time notifications. Slack collects commit logs from BitBucket, build/test logs from Codeship and deployment/run logs from the cloud, and then notifies the team.
DRAv2.0 Instance Toolsets (Added Tools to DRAv1.0)		
AWS (CodeDeploy)	- Automated scaling - Load balancing - Virtual servers—orchestration - Fast delivery—scaling - Deployment monitoring	AWS CodeDeploy enables automated scaling of the application deployed from Codeship. It enables automated scaling of the app for users. It also provides monitoring components for the application health at run time.
GAE	- Automated scaling - User access management - Virtual servers—orchestration - Fast delivery—scaling - Deployment monitoring	GAE enables automated scaling of the application deployed from Codeship. It enables automated scaling of the app for users. It also provides monitoring components for the application health at run time.

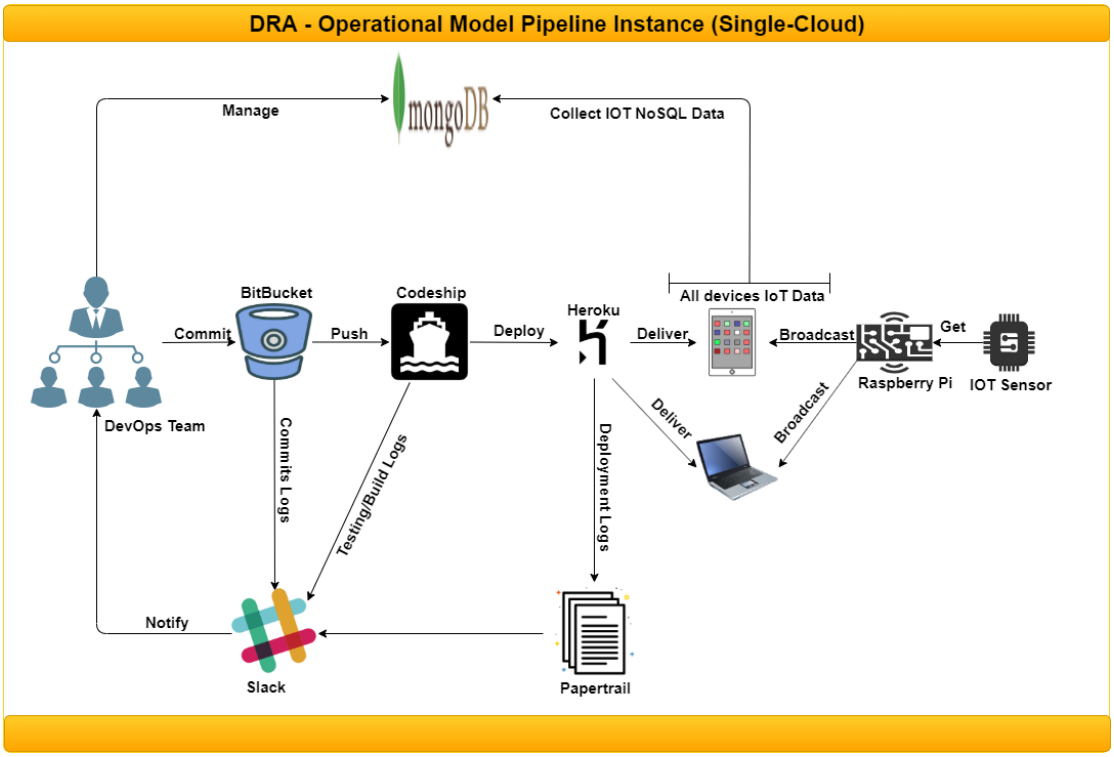


Figure 4.13: DRAv1.0 Instance Pipeline

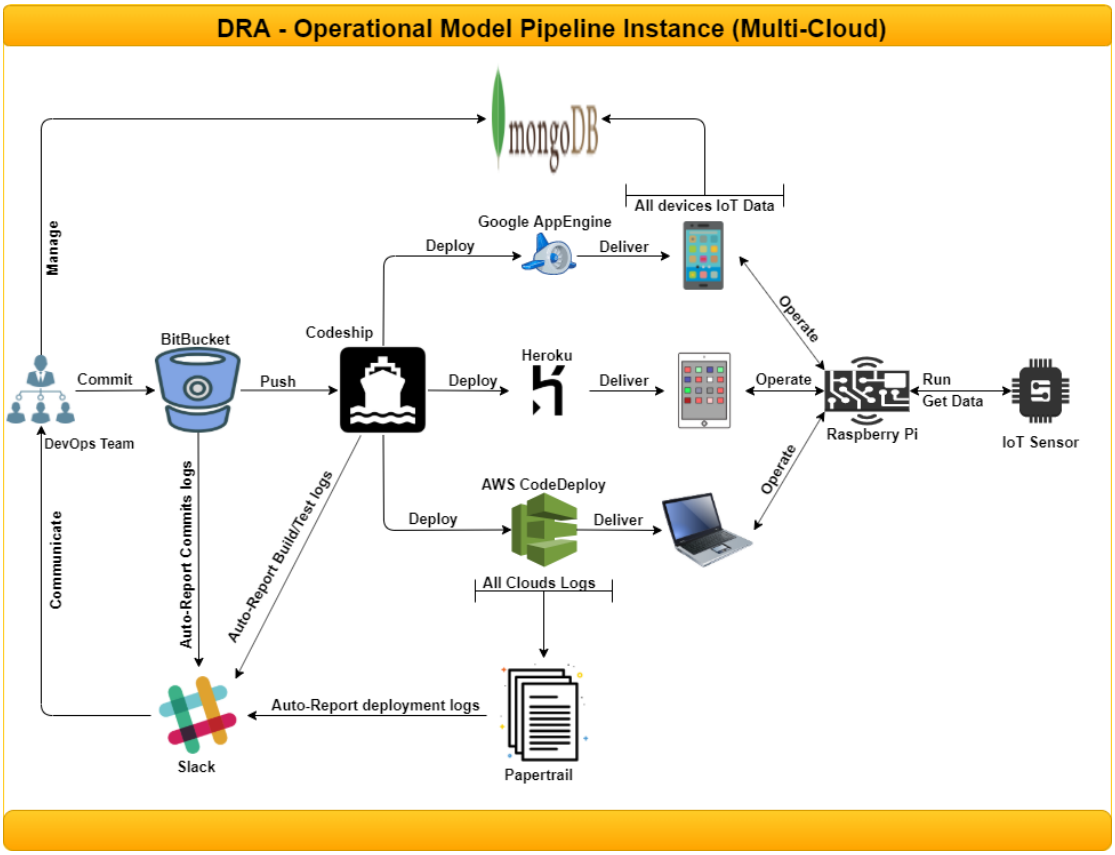


Figure 4.14: DRAv2.0 Instance Pipeline

- **DRA setup and configuration template**

The DRA framework offers a practical solution for IoT application deployment to the cloud (Ghantous & Gill 2018). This thesis provides a setup and configuration template for case studies' to prove the usability of the DRA (see Table 4.7). The DRA template will be used in case studies in Chapter 5. It provides general setup guidelines for the DRA pipeline to support the development environment for any application (supporting many programming languages). The template shows how to construct an operation pipeline that follows the conceptual model blueprint and enables the features and functions of the logical model functions. Hence, the DRA template (see Table 4.7) offers setup guidelines for the physical components of the framework: 1) DevOps team development environment, 2) repository, 3) CI broker, 4) CD platform (continuous development platform) or clouds, 5) monitoring, and 6) database. The template is an application for DRAv1.0 instance and DRAv2.0 instance (see Table 4.7).

Table 4.7: DRA Setup and Configuration Template

Step	DRAv1.0 and DRAv2.0 Instances –Setup and Configuration	
	Features	**Tools**
DevOps team	- Create Maven Project (maven-app-heroku) and configure its project descriptor **pom.xml** to include minimum required plugin plugins to enable DRA pipeline tools in the maven-app-heroku project - Create JUnit tests and Cucumber tests as part of the application - Create MongoDB (mLab) connector module **Note: Other types of applications may be programmed using various programming languages**	Maven plugin JUnit Plugin Heroku Plugin MongoDB plugin Cucumber plugin Bootstrap plugin Web jar plugin Net. ssh plugin
Repository	- Create a cloud repository for the maven-app-heroku, called 'dev-repo' on BitBucket - Integrate BitBucket with Slack and push commit logs	Bitbucket Codeship Slack
CI broker	- Setup Codeship environment script: *jdk_switcher use oraclejdk8* *mvn install* *mvn compile* *mvn test* - Setup Codeship deployment master branch to: Heroku (see documentation link) AWS (see documentation link) Google App Engine (see documentation link) - Integrate Codeship with Slack and push build/test logs	Codeship
CD platform	**Heroku setup:** - Create the maven-app-heroku project on Heroku - Add Web Dyno on Heroku for auto-scaling - Add Procfile with the same web dyno script to project root directory **AWS setup:** - Create the maven-app-heroku application on AWS - Create a user and get: the secret key and access key - Set up 2 or more EC2 instances - Set up a security group - Set up an IAM role - Set up a deployment group using EC2 instances - Set up an S3 bucket - Provide Codeship with access to S3 bucket and CodeDeploy **GAE Setup:** - Create the maven-app-heroku application on Google App Engine - Set up a bucket on Google - Provide Codeship with access to the bucket and the GAE	Heroku AWS Google App Engine
Monitoring	- Enable log capturing on Papertrail account from Heroku, AWS and Codeship - Integrate Papertrail to push deployment logs to Slack - Integrate Slack with Papertrail, Codeship and BitBucket, and collect all logs then notify users in real-time	Papertrail Slack
Database	- Create a mLab DB account through Heroku - Provide the connection link of mLab DB to the IoT application connector class	mLab (MongoDB)

In addition to the above instructions, AWS, GAE and Heroku require custom deployment setup parameters. These files are embedded within the maven-app-heroku project. Table 4.8 presents the deployment files and their descriptions for each deployment cloud.

Table 4.8: CD Platforms Parameters

Models	File Name	Code
AWS	**appsepc.yml** (Located in the root directory of IoT app)	`version: 0.0` `os: linux` `files:` `- source: /target/maven-app-heroku-1.0-SNAPSHOT` `destination: /var/www/html` `- source: /target/` `destination: /var/www/html`
GAE	**appengine-web.xml** (Located in directory WEB-INF of IoT app)	<?xml version="1.0" encoding="utf-8"?> <appengine-web-app xmlns="http://appengine.google.com/ns/1.0"> <application>propane-primacy-193602</application> <version>1</version> <runtime>java8</runtime> <threadsafe>true</threadsafe> <system-properties> <property name="java.util.logging.config.file" value="WEB-INF/logging.properties"/> </system-properties> </appengine-web-app>
Heroku	**Procfile** (Located in root directory of IoT app)	`web: java $JAVA_OPTS -jar target/dependency/webapp-runner.jar --port $PORT target/*.war`

4.4.2.2. IoT Application

The DRA software component used in this research is a Java maven web application with an integrated IoT application component (see Figure 4.15) called maven-app-heroku (https://maven-app-heroku.herokuapp.com/). The IoT app interacts with IoT sensors using Python scripts programmed on a Raspberry Pi Model 3 B (named RPIB).

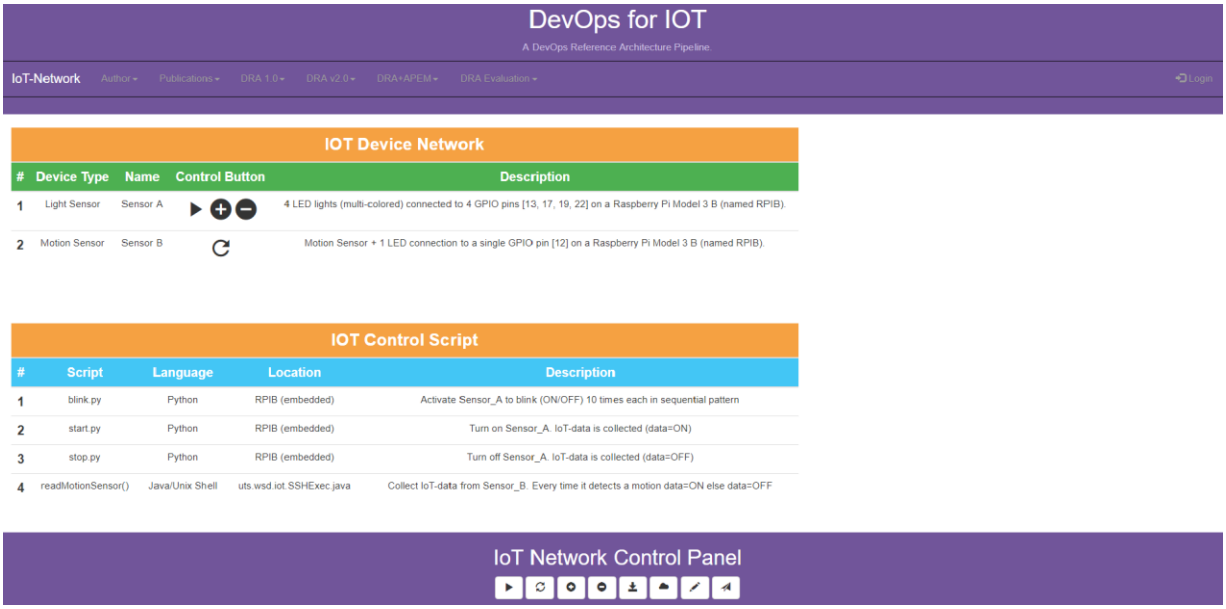


Figure 4.15: IoT App (maven-app-heroku) https://maven-app-heroku.herokuapp.com

The interaction of maven-app-heroku and the IoT devices is triggered using shell commands. The shell commands activate the Python scripts programmed on RPIB and read the RPIB GPIO pins data. Figure 4.16 shows that each pressed button on the IoT-application control panel runs an SSH command SSHC [index] from table 4.11.

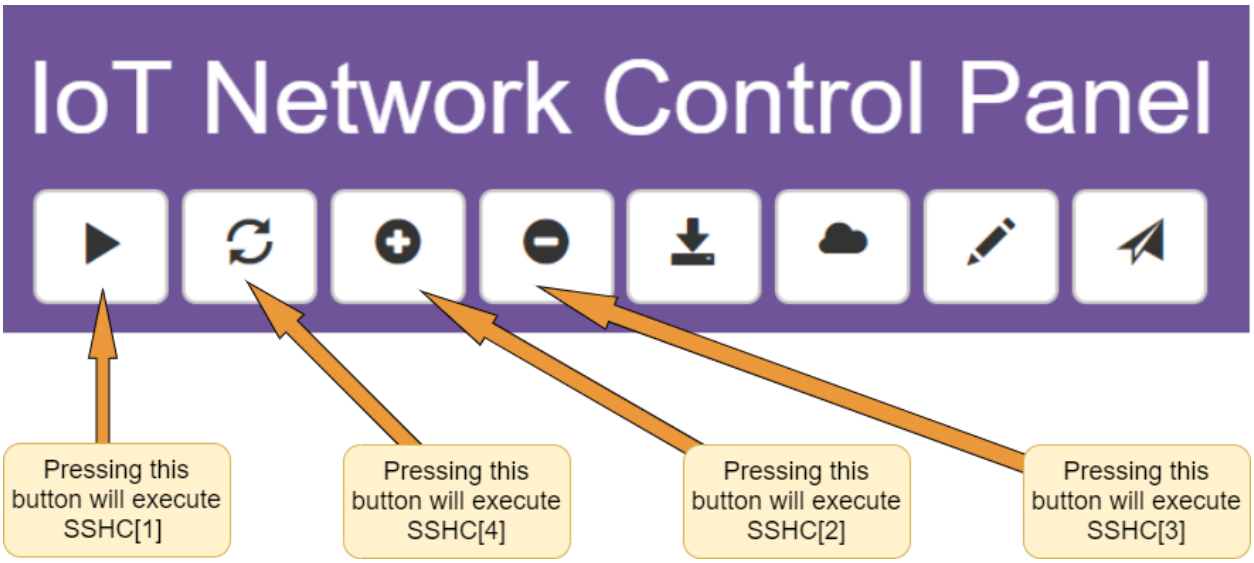


Figure 4.16: IoT App Control Panel

This section outlines the development steps and provides a development script of maven-app-heroku, the Python scripts and the shell commands used. It also provides key deployments for the IoT app setup. The complete IoT app (maven-app-heroku) code and Raspberry Pi Python scripts are stored on BitBucket.

This thesis provides 'observer' access to the project repository 'dev-repo' (see Table 4.9). It is necessary to add the correct plugins (see Table 4.7) in maven-app-heroku POM.XML. Complete details of POM.XML are located on BitBucket/dev-repo repository (see Table 4.9). The POML.XML file must be placed in the root directory of the maven-app-heroku application project.

Table 4.9: DRA Repository (dev-repo)

Software Components	IoT Application (maven-app-heroku)	
	Source Code Location	Tools
maven-app-heroku	BitBucket Link: https://bitbucket.org/product **Username**: devopsobserver@hotmail.com **Password:** observer **maven-app-heroku link:** https://maven-app-heroku.herokuapp.com/	BitBucket
Python applications	BitBucket Link: https://bitbucket.org/product **Username**: devopsobserver@hotmail.com **Password:** observer	BitBucket

- **Java components**

The main Java code components in this section are located in the following Java classes and JSP files of the maven-app-heroku project (see Table 4.9 for access to full code description):

- In package ***src.main.java.uts.wsd.iot***:
  - SSHExec.java
  - NullHostKeyVerifier.java
- In package ***src.main.java.uts.wsd.controller***:
  - IOTServlet.java
  - MongoDBConnector.java
- In package ***src.main.webapp***:
  - index.jsp
- In test Package ***uts.wsd.test***:
  - CucumberTest.Java
  - MavenJUnitTest.java
  - RunTest.java

- **Python components**

The Python scripts used to interact with the IoT sensors are programmed on RPIB (see Table 4.10). The scripts can communicate with IoT sensors set up on GPIO pins 13, 17, 19 and 22. The python scripts are labelled **PScript[index]** in table 4.10. The Python scripts are described as follows:

- blink.py: starts/stops each of the 4 LED lights with a sleep timer of 0.05 seconds. Repeat the process 10 times.
- startLED.py: starts the 4 LED lights all at once.
- stopLED.py: stops the 4 LED lights all at once.

Table 4.10: Python Scripts

PScript[1] = blink.py	PScript[2] = startLED.py	PScript[3] = stopLED.py
import RPi.GPIO as GPIO import time GPIO.setwarnings(False) GPIO.setmode(GPIO.BCM) xmas_led = (13,17,19,22) GPIO.setup(xmas_led,GPIO.OUT) j=0 while j<10: for i in range(len(xmas_led)): GPIO.output(xmas_led[i],True) time.sleep(0.05) GPIO.output(xmas_led[i],False) time.sleep(0.05) j+=1 GPIO.cleanup()	import RPi.GPIO as GPIO import time GPIO.setwarnings(False) GPIO.setmode(GPIO.BCM) xmas_led = (13,17,19,22) GPIO.setup(xmas_led,GPIO.OUT) for i in range(len(xmas_led)): GPIO.output(xmas_led[i],True)	import RPi.GPIO as GPIO import time GPIO.setwarnings(False) GPIO.setmode(GPIO.BCM) xmas_led = (13,17,19,22) GPIO.setup(xmas_led,GPIO.OUT) for i in range(len(xmas_led)): GPIO.output(xmas_led[i],False)

- **Action shell commands**

This section discusses the shell commands required to establish communication between the IoT application over the internet, and the Python scripts (see Table 4.10) programmed on an RPIB using SSH port 22. The action shell commands are executed in the Java class 'SSHExec.java' by clicking on buttons from 'index.jsp' (see Figures 4.15 and 4.16) and controlled by Java servlet 'IOTServlet.java' (see Table 4.9 for the complete code on BitBucket). The 'firecommand' method of 'SSHExec.java' class is used to execute the shell commands and establish a secure connection to RPIB and IoT sensors using port 22. The results of every shell command execution are stored as a string in a variable called 'result' in the class SSHExec.java. The IoT data stored in 'result' are automatically saved into the (MLab) DRA cloud database as JSON format. There are four SSH commands labelled **SSHC[index]** (see Table 4.11) used to communicate with IoT-sensors (Sensor_A and Sensor_B) (see Figure 4.17). The SSHC[index] commands have two roles: 1) activate the python scripts PScript[index] (see Table 4.10) and read the data from the sensors GPIO pins (see Section 4.4.B.3 IoT-Network).

Table 4.11: Action Shell Commands

SSHExec-firecommand (String Shell Command)	Shell Commands
public void fireCommand(String command) throws Exception { final SSHClient ssh = new SSHClient(); Command cmd = null; ssh.addHostKeyVerifier(new NullHostKeyVerifier()); int pn = 22; String ipaddress = "1.40.181.169"; String username = "pi"; String password = "georges034302"; try { ssh.connect(ipaddress, pn); } catch (IOException e) { } ssh.authPassword(username, password); final Session session = ssh.startSession(); cmd = session.exec(command); result = IOUtils.readFully(cmd.getInputStream()).toString(); ssh.close(); }}	*//Activate PScript[2] which runs startLED.py and read the IoT light sensor on pin 13 using the following SSH command:* **SSHC[2] =** **fireCommand("sudo python startLED.py");** **fireCommand("gpio -g read 13")** *//Activate PScript[3] which run stopLED.py and read the IoT light sensor on pin 13 using the following SSH command:* **SSHC[3] =** **fireCommand("sudo python stopLED.py");** **fireCommand("gpio -g read 13");** *//Activate PScript[1] which runs blink.py and make the IoT light sensor turn on and off for a short period using the following SSH command:* **SSHC[1] =** **fireCommand("sudo python blink.py");** *//To read data from motion sensor installed on pin 12 using the following SSH command:* **SSHC[4] =** **fireCommand("gpio -g read 12");**

4.4.2.3. IoT Network

This section discusses the IoT network (IoT devices and IoT sensors) that is used to demonstrate the IoT application interactions with IoT devices. This section outlines an IoT inventory of devices, sensors and connectivity protocols used to connect IoT applications with the IoT network. This section also shows how the devices are configured together to create a sample IoT network (IoT network). The IoT application (maven-app-heroku) interacts with the IoT network and generates IoT data, which are stored automatically in the DRA central database (MLab) in JSON format.

- **IoT network inventory**

The IoT devices and IoT sensors (IoT network) used in this research are explained as follows:

- Raspberry Pi Model 3 B (named RPIB)
- 4 LED lights connected to 4 GPIO pins (13, 17, 19 and 22) (named **Sensor_A**).
- Motion Sensor + 1 LED connection to a single GPIO pin (12) (named **Sensor_B**).

Sensor_A and Sensor_B are configured on the RPIB board (see Figure 4.17).

The IoT network is used in the empirical evaluation (see Chapter 5) to demonstrate the application of the DRA instances. The DRA instances are not fixed to a particular IoT network and can be used to deploy IoT applications that interact with numerous types of IoT devices.

- **IoT network configuration**

The IoT network is used for demonstration purposes and DRA proof of concept (POC) and evaluation (see Chapter 5). Figure 4.17 shows Sensor_A lights installed on GPIO pins (13, 17, 19 and 22) and Sensor_B installed on GPIO pin 12. These sensors are activated using Python scripts programmed on the RPIB board. The Python scripts (see Table 4.10) are triggered over the internet using shell commands executed using the Java method defined in 'SSHExec.java'.

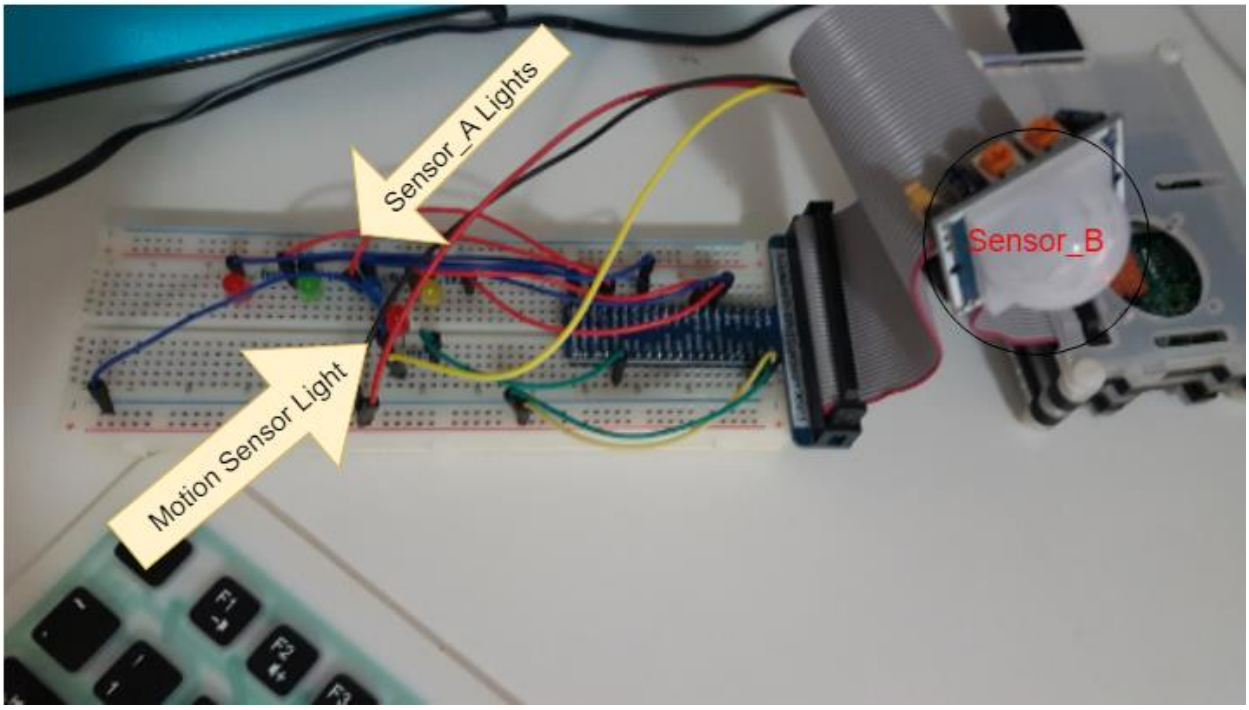


Figure 4.17: IoT Network

As noted in previous sections, the IoT network sample used for demonstration in this thesis is controlled using Python scripts. IoT network operational interaction with Python scripts (see Table 4.9) can be observed in the following YouTube video (Link).

- **Connectivity**

The IoT network is configured to connect to the internet using the organisation's local Wi-Fi network. Raspberry Pi (see Figure 4.17) can be configured on any network to obtain an IP address. The IoT application deployed to the cloud/multi-cloud interacts with the IoT network using the SSH protocol. The IoT network can be configured using different connectivity options such as mobile networks (3G, 4G, and 5G). The IoT application can interact with the IoT

network using different protocols such as MQTT or Bluetooth. Thus, IoT network connectivity options are not fixed to a particular instance and can be configured using internet connection options and connectivity protocols that are available in the organisation's context.

4.4.3. OUTPUT

This section presents the framework composition output, which are the DRA instances created using generic DRA reference architecture models (see Section 4.3). The output component is composed of three elements: DRA model, DRAv1.0 instance and DRAv2.0 instance.

4.4.3.1. DRA model

This section presents the fundamental structure for the DRA framework that can be instantiated by DevOps teams (developers, managers, coaches, and consultants) to deploy IoT applications to the cloud/multi-cloud.

A DRA model has three partitions:

a) DRA characteristics used by organisations as a foundation for DRA implementation.
b) DRA architectural design models used by organisations as a template to create instances.
c) DRA composition used by organisations to create a practical implementation of the framework.

4.4.3.2. DRAv1.0 instance (Single Cloud)

The DRAv1.0 instance (see Figure 4.18) structure is composed of four constituents or models:

- conceptual
- logical
- physical
- operational.

The operational model is an instance of the physical model that enables IoT application deployment to a single cloud (e.g., Heroku). The operational pipeline instance is built using integrated DevOps tools and cloud services. DRAv1.0 instance is an output of the DRA composition. Organisations may configure and set up the DRAv1.0 instance based on the DRA architectural models using numerous tools for different types of applications and technology stacks. Figure 4.18 shows how the conceptual model is expanded into a logical model that contains the necessary DevOps practices and cloud services to support and enable the DRA characteristics defined in Section 4.2. Figure 4.18 also shows how the physical model is created using the logical model as a template. The physical model is instantiated into an operational model pipeline that enables the product's (IoT application) deployment to a single cloud (e.g., Heroku).

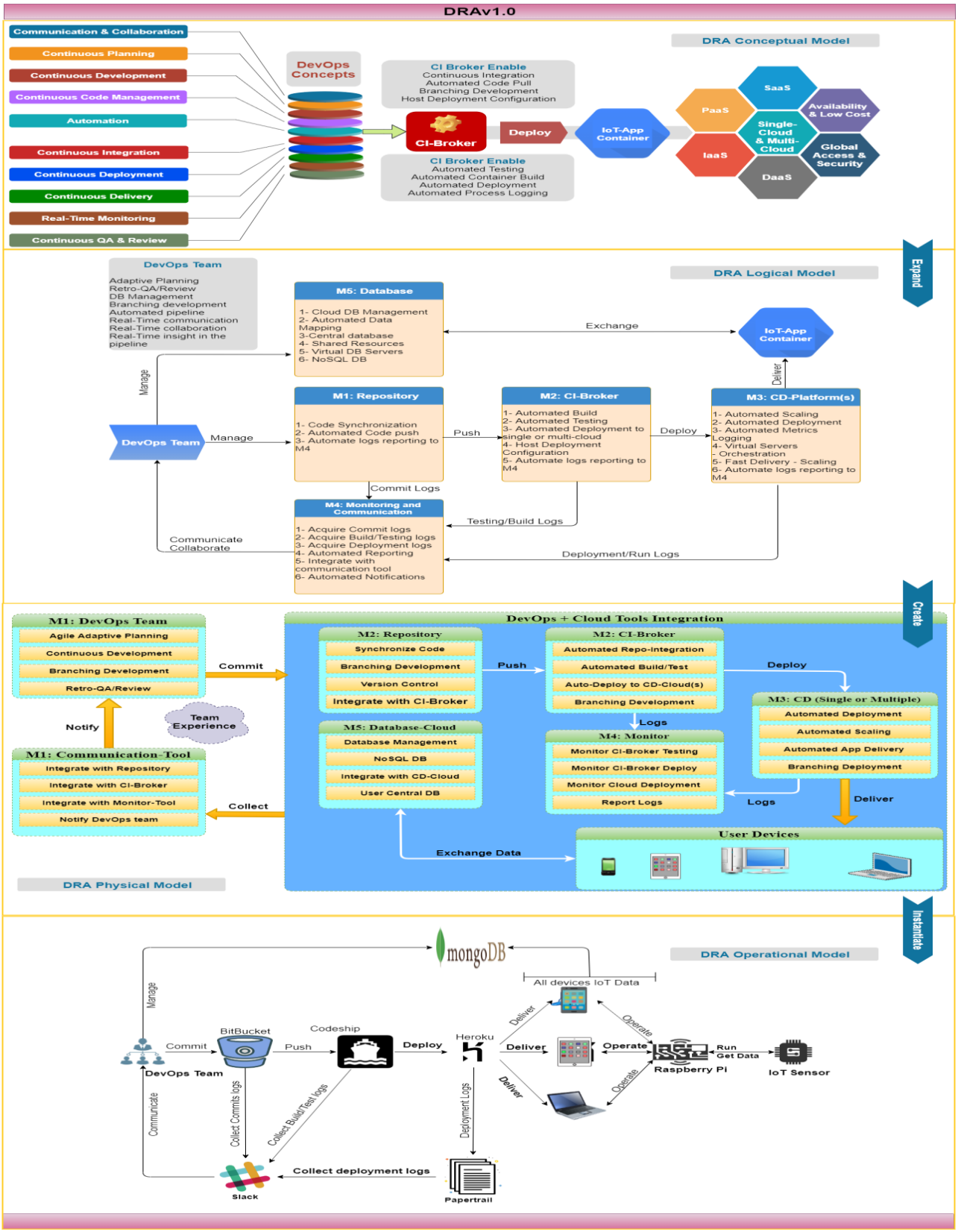


Figure 4.18: DRAv1.0 Instance

4.4.3.3. DRAv2.0 instance (Multi-Cloud)

The DRAv2.0 instance (see Figure 4.19) structure is composed of four constituents or models:

- conceptual
- logical
- physical
- operational.

The operational model is an instance of the physical model that enables IoT application deployment to the multi-cloud (e.g., Heroku, GAE). The operational pipeline instance is built using integrated DevOps tools and cloud services. The DRAv2.0 instance is an output of the DRA composition. Organisations may configure and set up the DRAv2.0 instance based on the DRA architectural models using numerous tools for different types of applications and technology stacks.

Figure 4.19 shows how the conceptual model is expanded into a logical model that contains the necessary DevOps practices and cloud services to support and enable the DRA characteristics defined in Section 4.2. Figure 4.19 also shows how the physical model is created using the logical model as a template. The physical model is instantiated into an operational model pipeline that enables the product's (IoT application) deployment to the multi-cloud (e.g., Heroku, AWS, GAE).

To prevent vendor lock-in issues, the DRAv2.0 instance contains a vital mechanism—the CI broker (e.g., Codeship)—that hosts the IoT application deployment configuration, as well as a central database (MongoDB mLab) to host the IoT data.

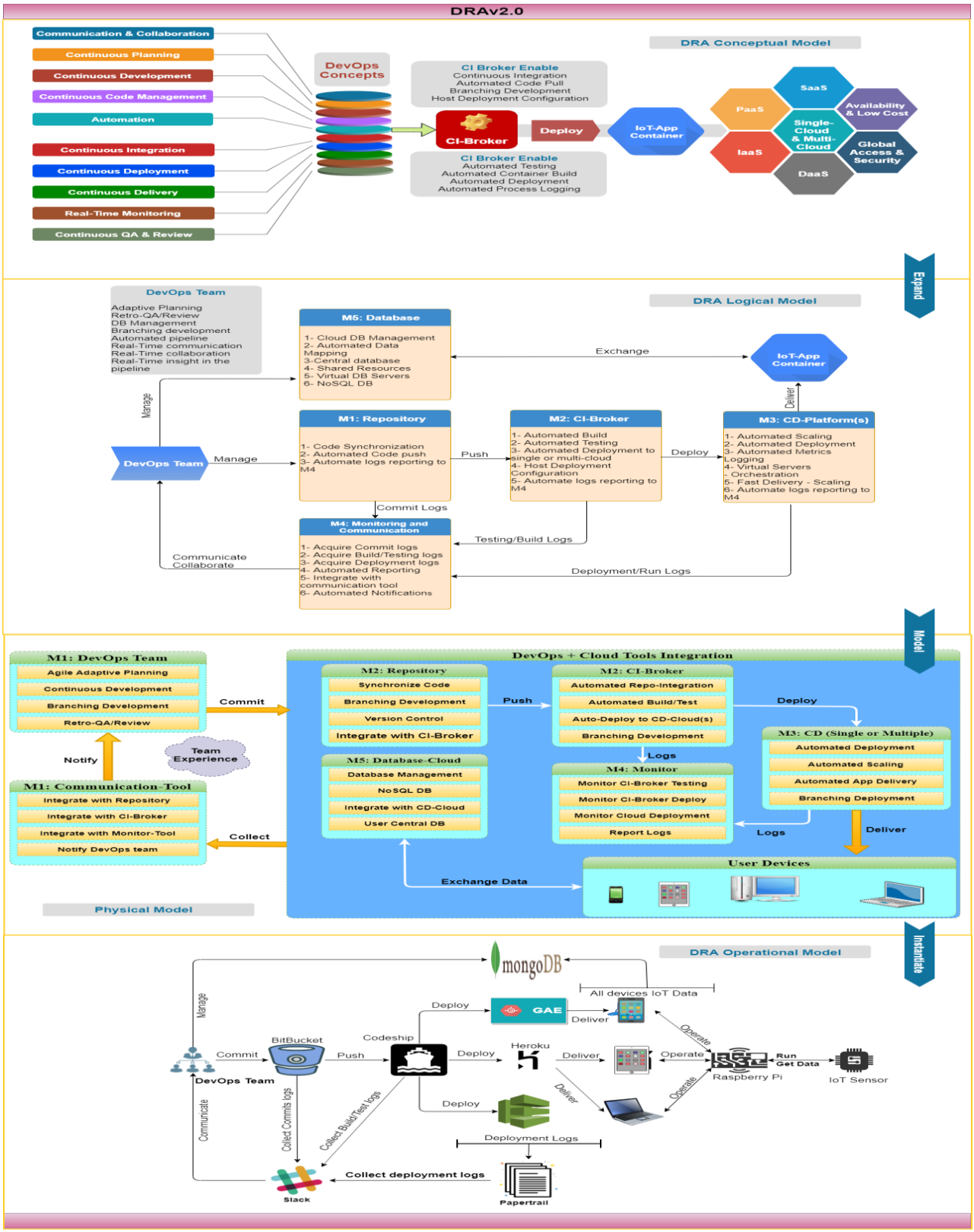


Figure 4.19: DRAv2.0 Instance

4.5. DRA FRAMEWORK IMPLEMENTATION

This section discusses the implementation steps required to create practical instances of the DRA framework in the organisation context. The DRA framework can be presented to organisations, research labs and academic courses using the implementation process and the case study template (CST) (see Appendix G). The CST is a procedure designed to guide experts, practitioners and researchers in how to apply DRA framework characteristics and use DRA architectural models to create practical instances. The implementation process is based on the DRA composition, which demonstrates how to do design, create and instantiate the DRA to enable IoT application deployment to the multi-cloud (DRAv2.0 instance). The DRA framework is composed of two steps: DRA implementation process and DRA evaluation process.

4.5.1. DRA INSTANTIATION PROCESS

The DRA instantiation process is a method aimed at helping organisations (managers, developers, engineers, coaches, consultants) create an instance of the DRA operational model for an organisation or a team. The DRA instance is tailored to adapt to the current environment and to use the DRAv2.0 instance (see Section 4.5.3.3) model to enable IoT application deployment to the multi-cloud. There are five steps to consider for the DRA instantiation process:

- initiation
- development
- pipeline configuration
- deployment
- management.

Each step has three elements:

- step goal
- step checklist
- step review.

The step goal is a unique entity that explains the purpose of a particular step. The step checklist is formed of 10 entities, which are requirements related to the particulate implementation step. Each entity description is determined by the response of the organisation team (developers, manager, engineers, consultants, coaches) to the corresponding question about that entity. The purpose of the checklist step is to provide the organisation's team with guidelines for implementing the numerous steps of the framework instance. The review step determines the level of success of the implementation of a particular step. The initiation step guidelines allow organisations and teams to decide if the implementation of DRA is achievable or feasible in the context of the organisation.

1. Initiation

Goal: To perform DevOps contextual analysis of the current environment in an organisation and determine whether a team should proceed with adopting the DRA framework.

Checklist: Table 4.12

Table 4.12: Initiation Checklist

Entity	Description
Identify the infrastructure	Which infrastructure is being used for the project (clouds)?
Identify the prospects	Who is involved in the framework adoption (consultants, managers, developers)?
Identify the hardware	What type of hardware is being used in the framework adoption (IoT devices)?
Identify DevOps concepts	Which DevOps concepts (see Table 2.9) are applicable in the adoption?
Determine business context	The business context (products, services, customers).
Determine the technology context	The technology context used in the adoption (tools, programming languages, network).
Determine prospect readiness	How ready are the prospects for the adoption (knowledge, skills, resources)?
Determine the time frame	The timeframe for the adoption (implementation, development, deployment, delivery).
Determine implementation area	The implementation area or location for the framework implementation (lab, remote).
Determine the adoption framework	Which framework will be adopted (e.g., DRA)?

Review: Organisations or teams can decide to implement the DRA framework based on the managers' and consultants' reviews of the initiation checklist success.

2. Development

Goal: To develop a product in the organisation's current environment using a DevOps approach. The prospects apply DevOps practices during the development process of the product. The DRA is founded on DevOps concepts and practices; it is relevant and practical to check the applicability of DevOps adoption during the development.

Checklist: Table 4.13

Table 4.13: Development Checklist

Entity	Description
Identify the product owner	Who owns the process (organisation, developers, managers, consultants)?
Identify the process	What is the developed product (type of application, requirements)?
Determine the environment	Where is the product being developed (cloud, lab, remote, integrated)?
Identify DevOps practices	Which DevOps practices (see Table 2.10) are applicable in the adoption?
Determine business value	What is the benefit of the product (access, data, performance)?
Determine technology	The technology used for development (tools, programming languages, libraries).
Determine prospect readiness	How ready are the prospects for the adoption (knowledge, skills, resources)?
Determine the time frame	The timeframe for development (construction, synchronisation, build, test).
Determine usability	How the product is being used and by who (purpose, users)?
Determine version control	Which version of the development product is being used (synchronisation, Retro-QA)?

Review: Organisations or teams can decide to implement the development process based on the managers' and consultants' reviews of the development checklist. The checklist success determines whether it is feasible to develop an IoT application or use a sample product that is compatible with the organisation's current environment.

3. Pipeline configuration

Goal: To develop a product in the organisation's current environment using a DevOps approach. The prospects apply DevOps practices during the development process of the product. DRA is founded on DevOps, multi-cloud and IoT integration; therefore, it is necessary to check the success of integration and application of the DRAv2.0 instance (see Table 4.10).

Checklist: Table 4.14

Table 4.14: Pipeline Configuration Checklist

Entity	Description
Identify the process owner	Who owns the process (organisation, developers, managers, consultants)?
Determine the code management tool	Which tool is being used for code synchronisation?
Determine CI broker	Which tools are being used for CI broker?
Determine CD clouds	Which clouds are being used for continuous deployment and delivery?
Determine the monitoring tool	Which tools are being used to monitor the application's deployment?
Determine the collaboration tool	Which tools are being used for communication and collaboration of prospects?
Determine DB tool	Which tools are being used for database management?
Determine IoT connection	Is the DRA pipeline connected to the IoT devices and IoT sensors?
Determine application delivery	Is the application being auto-scaled on all clouds for users?
Determine DRA instance	Which instance of DRA is being used (e.g., DRAv2.0 instance)?

Review: Organisations or teams can decide to use the operational model of the DRA based on managers' and consultants' reviews of the checklist (see Table 4.14). The checklist success determines whether it is feasible to construct a deployment pipeline for IoT application to the multi-cloud in the organisation's current environment. The deployment pipeline is a physical instance of the DRA operational model (see Chapter 4). DRAv2.0 instance pipeline configuration follows the DRA setup (see Chapter 4 for configuration guidelines).

4. Deployment

Goal: To deploy a product in the organisation's current environment using the DRA. The prospects apply DevOps practices to achieve the optimal deployment process. It is necessary to verify the effectiveness of the deployment in the pipeline based on the checklist table.

Checklist: Table 4.15

Table 4.15: Deployment Checklist

Entity	Description
Identify the process owner	Who owns the process (organisation, developers, managers, consultants)?
Verify commit logs	Can the prospects view application commit logs on the code management tool?
Verify build logs	Can the prospects view application build logs on the CI broker?
Verify testing logs	Can the prospects view application testing logs on the CI broker?
Verify deployment logs	Can the prospects view application deployment logs on the CI broker?
Verify application delivery	Can the prospects view application delivery logs on the cloud?
Verify IoT app usability	Can the prospects use the deployed application?
Verify IoT app and DB interaction	Are the data being saved to the cloud DB of the DRA?
Verify application availability	Is the application auto-scaled on the multi-cloud?
Determine deployment timeframe	How long did it take to deploy the application?

Review: Organisations or teams can identify the success of application deployment in DRAv2.0 instance pipeline is based on managers' and consultants' reviews of the checklist. The checklist determines whether the success of the deployment of the IoT application to the multi-cloud in the current organisation environment is feasible.

5. Management

Goal: To focus on facilitation, leading and supporting (management mindset) at the organisation, department or team level to deliver results and determine the business value of DRA adoption. The management scorecard is the crucial resource that can be used to determine DRA practicality and usefulness in the current environment of DRA. It is necessary to verify the level of success based on the checklist of management entities in the checklist table.

Checklist: Table 4.16

Table 4.16: Management Checklist

Entity	Description
Identify the process owner	Who owns the process (organisation, managers, and consultants)?
Manage collaboration	Does the DRA improve collaboration (developers, managers, consultants)?
Manage risk	What is the risk of DRA adoption?
Manage facilitation	Does the DRA facilitate team tasks?
Manage change	Does the DRA enable application examination and allow replanning?
Manage benchmark	How close is the DRA to a practical benchmark?
Manage business value	Does the DRA improve agile delivery?
Manage compatibility	Does the DRA have an adaptive characteristic?
Manage communication	Does the DRA enable real-time communication and notification?
Manage feasibility	Is the DRA considered a practical and useful model (DRAv2.0)?

Review: Organisations decide, or teams can measure the degree of success of DRAv2.0 instance is based on managers' and consultants' reviews of the checklist (see Table 4.16). The management review determines whether DRAv2.0 instance implementation is feasible in the organisation's current environment.

4.5.2. DRA EVALUATION PROCESS

The DRA evaluation for organisations aims to determine the applicability and usability of the DRA framework. The DRA evaluation is presented to participants (industry, research lab, and teaching) using a case study template (see Appendix G) and an industry field survey (see Appendix D). The case study template (CST) (see Appendix G) is used in chapter 5 provides industry and research experts with information and guidelines about the instantiation of the DRA architectural model. The survey questionnaires (see Appendix G) were developed to obtain the industry experts' perspectives about the DRA framework. The data collected from the case studies and surveys were analysed to determine that the DRA meets the validation criteria for the case study (see Table 3.2) and for the survey (see Table 3.4); explained in chapter 3-DSR.

4.6. SUMMARY

This chapter presented the main components of the DRA framework. The DRA framework is founded on three main components (framework characteristics, framework architecture, and framework composition). The framework characteristics are commonly used terminologies that represent the DRA entities. The DRA framework characteristics are used to create a generic architecture design model using the DevOps approach and cloud services. The DRA framework architecture model is composed of five components (contextual, conceptual, logical, physical, and operational). DRA architecture can be used as a template or blueprint to create instances in organisations' contexts. The DRA framework composition incorporates the DRA resources and configuration needed to create the framework output instances. This chapter presented two instances of the DRA model: DRAv1.0 and DRAv2.0. The chapter also presented the DRA implementation and evaluation templates. These templates are used in Chapter 5 to evaluate the DRA framework using an empirical evaluation. Information about the DRA framework development, implementation and publications are available on the development website: https://maven-app-heroku.herokuapp.com/.

Chapter 5: DRA Framework Empirical Evaluation

Chapter 4 presented the DRA framework to address the main research question (see Chapter 1). The validity of this assertion is explained in the current chapter using an empirical evaluation. This evaluation has been conducted in four iterations. First, this chapter presents an industry case study that demonstrates the applicability of the DRA to a practical context. Second, it discusses a research case study conducted in a lab environment at UTS. Third, the chapter discusses a teaching case study that demonstrates how the DRA can be used in a project-based software development environment for teaching at UTS Faculty of Engineering and Information Technology (FEIT). Fourth, this chapter presents the results of the industry survey that was conducted with software professionals from various organisations. The survey data are analysed to conclude the final evaluation of the DRA. The applicability and relevance of the DRA framework are evaluated based on the feedback collected from the case studies and field survey.

5.1. Framework Evaluation Overview

This section presents high-level testing of the DRA to provide POC of the proposed framework in the industry, research and teaching. Two types of empirical evaluations are used in this thesis: case study evaluation and survey evaluation. The results from the empirical evaluation are used in to determine that the DRA framework meets the evaluation criteria (see Chapter 3, Table 3.2, Table 3.4). The DRA empirical evaluation overview is illustrated in Figure 5.1 and represents the empirical evaluation overall results analysis to determine the applicability, reusability and novelty of the DRA. The testing was conducted in four iterations:

1. An industry case study (see Section 5.3) was conducted at the organisation code-named CPF using a case study template (see Appendix G). The participant was a DevOps engineer (DE) from the CPF organisation who evaluated the applicability of the DRA for the CPF context.
2. A research case study (see Section 5.4) was conducted at DigiSAS Lab UTS FEIT using a case study template (see Appendix G). The participant was the DigiSAS Lab director and supervisor who evaluated the applicability of the DRA for the DigiSAS Lab software development and deployment context.
3. For the teaching case study surveys (see Section 5.5) the DRA framework was integrated into two subjects at UTS FEIT (SEP 48440 School of Computer Science and INP 31261 School of Electrical and Data Engineering). Data from the SFS (student feedback survey) were collected anonymously from students in these subjects to determine the relevance and usefulness of the DRA for teaching.
4. An industry survey (see Section 5.6) was offered to a cohort of 82 professionals from local and international companies. The survey (Link) was offered to participants through LinkedIn. The data were collected anonymously to help provide POC to the DRA and to determine its relevance from the experts' point of view.

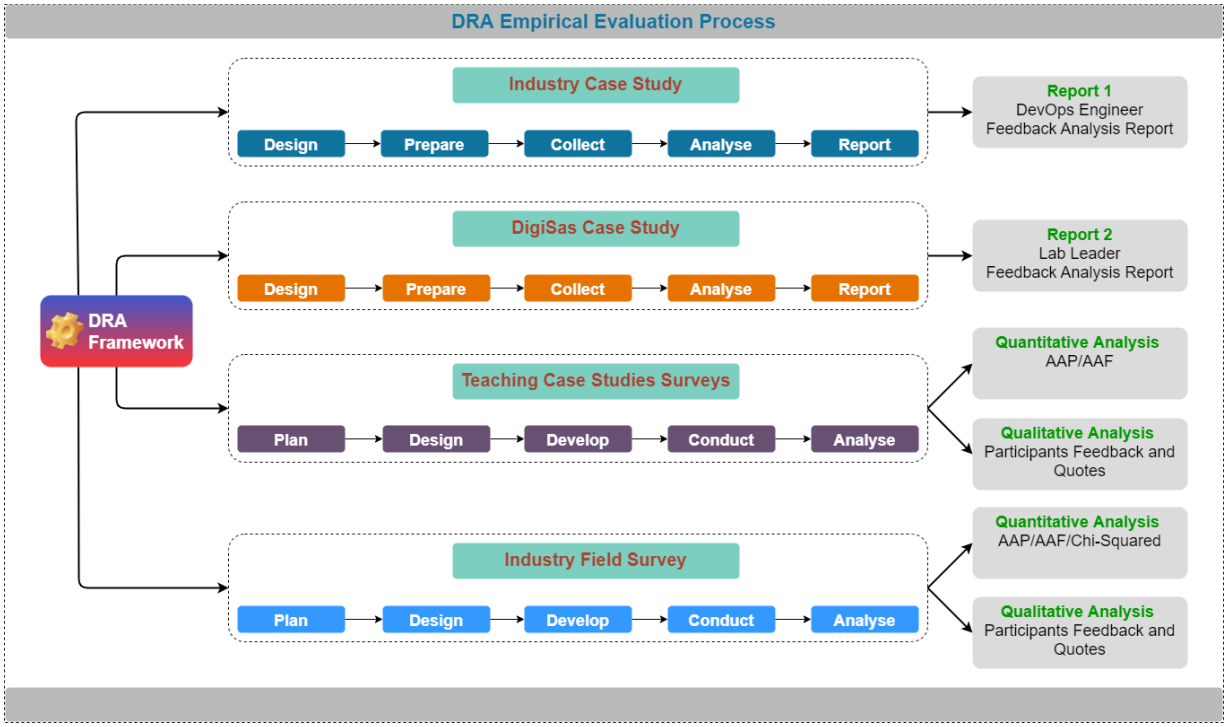


Figure 5.1: Empirical Evaluation Overview

5.2. DRA INSTANTIATION

This section presents the DRA instantiation process used to create instances for the empirical evaluation. The empirical evaluation evaluates the implementation of two DRA instances: DRAv1.0 (single-cloud) and DRAv2.0 (multi-cloud). The DRA instantiation process is explained in four tables:

- Table 5.1: Identifies and recommends commonly used DevOps and cloud tools that could be used to create an instance pipeline. Table 5.1 provides a list of features and a brief description of every tool.

- Table 5.2: Identifies the steps required to create a DRA instance and the configuration required for every recommend tool in the instance.

- Table 5.3: Describes the software application (IoT-application) used to test the DRA instances applicability. The software application is stored on BitBucket and can be accessed using and observer (guest) account setup for the evaluators.

- Table 5.4: Describes the IoT-devices (IoT-Network) used in the evaluation to test the IoT-application interactions. Table 5.4 includes two sensors (Sensor A and Sensor B) and a brief description of the sensors setup and configuration on the Raspberry Pi board **(RPIB).**

Table 5. 1: DRA Instance Toolset

Tools	Features	Description
BitBucket	1. Code synchronisation 2. Automated code push 3. Automate commit logs to Slack	BitBucket is a team collaboration and code management tool. It enables code synchronisation and automatically pushes the application to Codeship. It also sends commit- logs to Slack.
Codeship	1. Automated build/testing 2. Automated deployment 3. Hosts deployment configuration for applications 4. Enables automated testing 5. Automate build/testing logs to Slack	It is the Continuous Integration Broker (CI-Broker) tool. It enables automated build/testing for code automatically received from BitBucket. It also enables automated deployed to clouds. It also sends build/testing/deployment logs to Slack.
Heroku	1. Automated scaling 2. Virtual servers—orchestration 3. Fast delivery—staging 4. Automate deployment logs to Papertrail	Heroku cloud enables the automated staging of the application deployed from Codeship. It enables automated scaling of the app for users. Run-time logs of the application are sent to Papertrail.
Papertrail	1. Acquire deployment logs 2. Automated deployment logs to Slack 3. Automated notifications	Papertrail monitoring the deployment and execution logs of the IoT application. It sends those logs to Slack.
MLab	1. Cloud DB management 2. Dynamic application access 3. Virtual DB servers 4. NoSQL DB	MLab is a MongoDB cloud database management service. It enables automated data access and mapping. MLab collected IoT data from the IoT application and store that data in JSON NoSQL. DevOps team can dynamically manage IoT data on MLab.
Slack	1. Automated log management 2. Automated notifications 3. Real-time communication (chat, video) 4. Resource-sharing option 5. Integration with Codeship and Papertrail	Slack is a communication and collaboration tool. It provides DevOps team with real-time chat/video conference option and enables automated real-time notifications. Slack collects commit logs from BitBucket, build/test/deployment logs from Codeship, and deployment/run logs from the cloud then notifies the team.
AWS (CodeDeploy)	1. Automated scaling 2. Load balancing 3. Virtual servers—orchestration 4. Fast delivery—staging 5. Deployment monitoring	AWS CodeDeploy enables the automated staging of the application deployed from Codeship. It enables automated scaling of the app for users. It also provides monitoring components for the application health at run time.
GAE	1. Automated scaling 2. User access management 3. Virtual servers—orchestration 4. Fast delivery—staging 5. Deployment monitoring	GAE enables the automated staging of the application deployed from Codeship. It enables automated scaling of the app for users. It also provides monitoring components for the application health at run time.

Table 5. 2: DRA Instance Setup and Configuration Template

Steps	**Configuration**	**Tools** [or other tools of choice]
DevOps team	- Create an application project (IoT app) - Create testing modules - Create MongoDB (mLab) connector module - Add necessary dependencies and plugins	[Provide a list of plugins and dependencies]
Repository	- Create a cloud repository for the application - Integrate BitBucket with Slack and push commit logs	Bitbucket Slack
CI broker	- Setup Codeship environment script: [which programming language] [which compiler-compiler command] [which tester–testing command] - Setup Codeship deployment master branch to: Heroku (see documentation link) AWS (see documentation link) Google App Engine (see documentation link) - Integrate Codeship with Slack	Codeship
CD platform	**Heroku Setup:** - Create the maven-app-heroku project on Heroku - Add Web Dyno on Heroku for auto-scaling - Add Procfile web dyno script to the root directory **AWS Setup:** - Create the maven-app-heroku application on AWS - Create a user and get: the secret key and access key - Setup 2 or more EC2 instances - Setup a security group - Setup a deployment group using EC2 instances - Setup an S3 bucket - Provide Codeship with access to S3 and CodeDeploy **GAE Setup:** - Create the maven-app-heroku application on GAE - Setup a bucket on Google - Provide Codeship with access to GAE	Heroku AWS GAE
Monitoring	- Enable log capturing from Heroku, AWS and Codeship - Integrate Papertrail to push deployment logs to Slack - Integrate Slack with Papertrail, Codeship, BitBucket	Papertrail Slack
Database	- Create a mLab DB account through Heroku - Provide the connection link of mLab DB to the IoT application	mLab

Table 5. 3: Software Component

Software Component Checklist	Description
Application name *[required]*	maven-app-heroku
Application type *[IoT]*	Java Maven app with IoT module
Programming languages *[required]*	Java/Python
IDE *[required]*	Netbeans
Unit testing module *[required]*	JUnit
Acceptance testing module *[required]*	Cucumber
Resources	https://bitbucket.org/product
Access	observer
Username	devopsobserver@hotmail.com
Password	observer

Table 5. 4: Hardware Component

Step	Setup and Configuration	Tools
Sensor A	4 LED lights (multi-coloured) connected to 4 GPIO pins (13, 17, 19, 22) (named **Sensor_A**) on a **RPIB**.	LED lights Raspberry Pi
Sensor B	Motion Sensor + 1 LED connection to a single GPIO pin (12) (named **Sensor_B**) on a **RPIB**.	Motion sensor Raspberry Pi

5.3. INDUSTRY CASE STUDY

The evaluation of the DRA was conducted in real-world settings in the organisation context. The case study was conducted using a case study template tailored to the context of the industry and research (see Appendix G). The data collected from the case study were qualitative feedback. The industry case study was conducted following specific steps outlined in chapter 3 (DSR Evaluation step):

1. case study design
2. preparation for data collection
3. collecting data
4. data analysis
5. reporting.

The industry case study is illustrated in Figure 5.1 as an iteration of the empirical evaluation of the DRA framework.

5.3.1. Case Study Plan

The case study is organised to demonstrate the applicability of the DRA in a real-world context. The case study process adopted to evaluate the DRA for CPF is as follows:

- **Identify the case study organisation:**
  The case study is conducted at the organisation [CODE_NAME: **CPF**]. Date: 24/04/2019
  This evaluation involved the organisation's DE, who is involved in the company's business product models. DE's personal information was kept anonymous following the ethics approval code ETH18-2339 (see Appendix A).
- **Case study organisation context:** CPF efficiently consolidates and integrates the digital strategy, solution designs and project delivery across the portfolio. CPF aims to provide a cloud-based modelling platform to enable the business transformation from strategy to execution.
- **Need, and problem:** Need a DevOps approach to deploy their platform features to the multi-cloud environment for different customers.
- **Solutions:** The DRA seems to address the need and problems mentioned above. The DRA enables the deployment of software to the multi-cloud. It automates the process of deployment to the multi-cloud and enables faster software release. The DRA has been explained as a guideline for setting the DevOps for the multi-cloud.
- **Objective:** The objective is to evaluate the applicability of the DRA in the practical organisational context. The organisation objective is to have a working DevOps environment based on a cloud platform that enables the automation of software deployment.
- **DRA POC demo and presentation:** To evaluate the DRA framework, a presentation slide pack and demo were developed to demonstrate the deployment of a predeveloped sample IoT application to the multi-cloud environment. The POC package demonstrates the applicability and functioning of the DRA in operations:
  - Demo Video YouTube video: Link
  - Presentation Slides: Link

5.3.2. Preparation for Data Collection

The case study was conducted at organisation CPF using a case study template (see Appendix G) that was tailored for the context of the industry and research to provide a flexible application of the DRA in real-world settings. The DRA was tested using a prepared testing package at the organisation premises under the supervision of the participant (DE), who was able to assess the DRA using the POC package as a reference to understand the characteristics and concepts of the framework. The demo and presentation of the DRA ran for approximately 30 minutes. The participant also contributed to the industry field survey (duration approximately 30 minutes). The participant evaluated the following DRA components defined in Chapter 4 (duration approximately 30 minutes):

1. DRA architecture (see chapter 4)
2. DRA operational model pipeline (see Table 5.1 and Table 5.2)
3. Software components (see Table 5.3)
4. Hardware components (see Table 5.4).

5.3.3. COLLECTING DATA

After the demo and presentation, the PhD researcher organised an evaluation session with the evaluator (DE) (duration approximately 30 minutes). The total duration of data collection, including the demo, presentation, survey contribution and case study contribution, was approximately 120 minutes. The case study data were stored on CloudStor, the UTS-recommended cloud storage (see Appendix E). The participant provided qualitative feedback about the DRA components from its practical application perspectives as follows.

5.3.3.1. DRA Architecture

The DRA architecture was presented to the industry case study organisation CPF. The DRA architecture is composed of four design architecture models: conceptual, logical, physical and operational. The case study participant provided feedback on the architecture design and its applicability to their organisation. The expert from the CPF (DE) reviewed the design and provided positive feedback with further opportunities for improvements.

DE's feedback about the DRA architecture: '*It has been very well thought and process-driven. I think it would be excellent to include some controls which could be used in the case to re-deploy or even roll back to a previous version in an automated fashion*'.

5.3.3.2. DRA Operational Model Pipeline

In this step, the case study template provides a checklist for the DRAv2.0 instance pipeline implementation. The organisation's DE may reuse the recommended toolset in Table 4.6 or configure DRAv2.0 instance pipeline with other tools of choice. Table 4.7 outlines the setup process of DRAv2.0 instance pipeline. Table 4.7 is designed to facilitate the configuration of the DRA for organisation prospects.

DE's feedback about the DRA toolset: '*Tools used in operation model pipeline are industry used tools and are an excellent choice for the DRA operation mode pipeline*'.

DE's feedback about the DRA setup and configuration process: '*Configuration template is easy to use and can be replicated*'.

5.3.3.3. Software Component

DRAv2.0 instance pipeline can be configured to deploy any applications. The participant (DE) was provided with a demo application (maven-app-heroku) to test the DRA architecture (see Table 5.3). The demo application source code may be accessed on the code repository using observer public access stated in Table 5.3. However, in the case studies, prospects may use their IoT application (or non-IoT application) to test DRAv2.0 instance.

DE's feedback about the IoT app: '*Testing software component is functioning properly*'.

5.3.3.4. Hardware Component

Table 5.4 presents information about a sample IoT device network that was used to test the IoT application process. The IoT network sample (see Figure 4.17) was configured by the researcher to provide POC of the DRA operation model and its applicability for enabling IoT processes and interactions on the multi-cloud. However, in the case studies, organisations' prospects may use their IoT devices and sensors to test the IoT application deployed using DRAv2.0 instance pipeline.

DE's feedback about the IoT network: '*Testing hardware component is functioning properly and responding appropriately*'.

Overall Feedback

The participant provided feedback about the demo package and the presentation prepared for the organisation's case study and imparted overall feedback about the DRA framework. The feedback represents DE's opinion about the DRA application in the organisation context.

DE's feedback about the demo video used in the presentation: '*Demo was easy to understand and well presented*'.

DE's overall feedback about the DRA framework: '*DRA framework would help organisations understanding DevOps methodologies and agile application deployment and delivery*'.

5.3.4. DATA ANALYSIS

The case study data collected during the experiment are analysed in Table 5.5. The data were analysed using the cross-examination method between DE's feedback and the case study evaluation criteria in Table 3.2. This analysis aims to connect or relate the hypotheses (evaluation criteria) to the expert's feedback. The output of the analysis is organised into two columns: 'interpretation', which is the researchers' interpretation of the feedback, and 'DRA categories', which is the relationship between the feedback and the evaluation criteria.

Table 5.5: Industry Case study Analysis

Participant Feedback	Interpretation	DRA Categories
'It has been **very well thought and process-driven**. I think it would be excellent to include some controls which could be used in the case study to re-deploy or even roll back to a previous version in an automated fashion'.	This feedback indicates that DRA design models are well thought of and could be improved with controls to enable automated rollback. DRAv2.0 enables automated deployment and automated rollback to stable software versions.	Usefulness
'Tools used in Operation model pipeline are industry used tools and are an excellent choice for the DRA Operation Mode Pipeline'.	This feedback indicates that the DRA toolset choice is excellent for the instance pipeline.	Usefulness
'Configuration template is easy to use and can be **replicated**'.	This feedback indicates that the DRA is easy to use in the organisation. It also indicates that the DRA is applicable and instantiable.	Reusable
'Testing the software component is functioning properly'.	This comment indicates that the DRA IoT application is functioning as intended on the multi-cloud.	Usefulness
'Testing hardware component is functioning properly and responding appropriately'.	This comment indicates that the IoT devices are interacting with IoT application deployed on the multi-cloud.	Usefulness
'Demo was easy to understand and well presented'.	This feedback indicates that the organisation's DE seems to be satisfied with the DRA demo package used for presenting the framework operations.	Coverage
'**DRA framework would help organisations understanding DevOps methodologies and agile application deployment and delivery'.**	This feedback indicates that the DRA provides helpful new knowledge for organisations about the DevOps approach and agile application deployment and delivery process.	Coverage Usefulness

5.3.5. REPORTING

The case study report is an organised case study outcome presented to the audience. The report aims to draw a conclusion from the DE's point of view about the DRA framework in the context of CPF. Table 5.6 presents the systematic testing procedure of the industry case study at CPF.

Table 5.6: Industry Case Study Reporting Summary

Case Study 01	Description
Organisation	CPF [CODE-NAMED]
Test date	24/04/2019
Organisation context	CPF efficiently consolidates and integrates the digital strategy, solution designs and project delivery across the portfolio. CPF aims to provide a cloud-based modelling platform to enable business transformation from strategy to execution.
Test team (TT) (Participants)	DE at CPF who is involved in the company business product models.
Organisation need	CPF need DevOps approach to deploy their platform features to the multi-cloud environment for different customers.
Test objective	The objective is to evaluate the applicability of the DRA in the practical organisational context. The organisation objective is to have a working DevOps environment based on a cloud platform that enables the automation of software deployment.
Test case question	How can the application features be deployed to the multi-cloud using DevOps?
Test package (Pre-prepared)	To evaluate the DRA framework, a presentation slide pack and demo were developed to demonstrate the deployment of a predeveloped sample IoT application to the multi-cloud environment: - Demo YouTube video: Link - Presentation slides: Link
Main test component	The participant (DE) evaluated the following DRA components: 1. DRAv2.0 architecture (see Figure 4.19) 2. DRA operational model pipeline (see Figure 4.14) 3. Software components (see Table 5.3) 4. Hardware components (see Table 5.4)
Test method	Case study template (see Appendix G) Industry survey (see Appendix D)
Test duration	120 minutes (presentation, demo, survey, case study)
Data type	Qualitative feedback provided by the organisation's DE.
Pretesting	The researcher presented the case study project to the participant using the presentation slides. The researcher organised a question-answer session to explain necessary DRA aspects if required.
Key activities	- TT verifies that DRA supports DevOps concepts (see Table 2.9). - TT verifies that DRA enables DevOps practices (see Table 2.10). - TT verifies that DRAv2.0 toolset (see Table 4.6) are easy to configure using the guidelines provided by the case study (see Table 4.7). - TT verifies that DRAv2.0 instance pipeline enables CI using CI broker (see Table 4.7 and Figure 4.19). - TT verifies the automated and continuous deployment and delivery of the IoT app to multi-cloud (see Table 5.3 and case study demo package). - TT verifies the IoT app interaction with IoT network sample (see Tables 4.19 and 4.20 and case study demo package). - TT verifies that DRA design models can be reusable and instantiable. - TT verifies that DRA conceptual model offers new knowledge for the organisation to further understand the DevOps approach for agile application deployment and delivery.
Expected outcome	The expected outcome of the case study is to determine that the DRA provides possible solutions to CPF's needs and, by extension, verify that the framework

Case Study 01	Description
	addresses the research gaps (see Section 4.C).
Actual outcome	The actual outcome is determined in the case study analysis section (see Table 5.5). The participant (DE) from CPF imparted valuable qualitative data (as feedback) about the DRA. The cross-examination between the feedback and Table 3.2 indicates that the evaluator (DE) considers DRA design models reusable in the organisation context and easy to configure. The participant indicated that the framework offers new knowledge about DevOps methodologies for the agile application deployment and delivery process. After cross-examination with Table 3.2, this also indicates that the DRA provides a sufficient explanation about the framework elements that support the DevOps approach.
Observation and interpretation	DE contributed to the case study data collection and the industry field survey based on the information provided by the demo package. It has been observed that the framework can be easily reconfigured for an experimental situation using the DRA design models. This indicates that the framework is not fixed for a particular situation and may be generalised as a conceptual design to fit the purpose of the organisation's context. TT indicated that the DRAv2.0 instance pipeline uses excellent industry tools of choice. The pipeline demonstrated in the demo video shows that the IoT application was successfully deployed to the multi-cloud and successfully interacted with the IoT network of sample devices, which responded correctly to the IoT application actions and events. The observations are drawn from TT's feedback. They indicate that the DRA framework seems to provide an adequate solution to deploy IoT applications on the multi-cloud using the DevOps approach.

5.4. RESEARCH CASE STUDY

The evaluation of the DRA was conducted in a real-world setting of the organisation context. The case study was conducted using a case study template tailored to the context of the industry and research (see Appendix G). The data collected from the case study were qualitative. The industry case study was conducted following specific steps outlined in chapter 3 (DSR Evaluation).

1. case study design
2. preparation for data collection
3. collecting data
4. data analysis
5. reporting.

The research case study is illustrated in Figure 5.1 as an iteration of the empirical evaluation of the DRA framework.

5.4.1. CASE STUDY DESIGN

The case study at the DigiSAS Lab, UTS (15/08/2019) is organised to demonstrate the applicability of the DRA in the real-world context. This evaluation involved the lab leader who

leads and manages several software-related research and development projects at the DigiSAS Lab, including this PhD project. A summary of the process adopted for the evaluation of the DRA for the DigiSAS Lab is presented below:

- **Identify the case study organisation:** UTS SCS DigiSAS Lab. I am undertaking a PhD with this lab. Date: 15/08/2019
- **Case study organisation context:** Conduct research and development in collaboration with industry partners, who are working on several software-related projects involving web and IoT and the multi-cloud. The partners are from large to small- and medium-sized enterprises (SMEs) and start-ups.
- **Need and problem:** Need a multi-cloud deployment environment for meeting the needs of different industry partners, as they have different cloud deployment needs. The challenge or problem is how to perform multi-cloud deployments.
- **Solutions:** The DRA seems to address the abovementioned need and problem. The DRA has been explained and used as a guideline framework for setting the DevOps for the multi-cloud.
- **Objective:** The researcher's objective is to evaluate the applicability of the DRA in the research lab environment. The lab's objective is to have a working DevOps environment for multi-cloud IoT application deployments.
- **DRA POC demo and presentation:** To evaluate the DRA framework, a presentation slide pack and demo were developed to demonstrate the deployment of a predeveloped sample IoT application to the multi-cloud environment. The POC package demonstrates the application and working of the DRA in operations.
  - Demo Video YouTube video: Link
  - Presentation Slides: Link

5.4.2. Preparation for Data Collection

The case study was conducted at the UTS SCS DigiSAS Lab using a case study template (see Appendix G) that was tailored for the context of the industry and research to provide a flexible application of the DRA in a research lab setting. The DRA was tested using a prepared testing package at the lab premises under the lab leader who leads and manages several software-related research and development projects at the DigiSAS Lab, including this PhD project. The participant was able to assess the DRA using the POC demo package as a reference to understand the characteristics and concepts of the framework. The demo and presentation of the DRA ran for approximately 30 minutes. The participant (Lab Leader or LL) evaluated the following DRA components defined in Chapter 4 (duration approximately 30 minutes):

1. DRAv2.0 architecture (see Figure 4.19)
2. DRA operational model pipeline (see Table 5.1 and Table 5.2)
3. software components (see Table 5.3)
4. hardware components (see Table 5.4).

5.4.3. COLLECTING DATA

After the demo and presentation, the PhD researcher organised an evaluation session with the evaluator (LL) (duration approximately 30 minutes). The total duration of data collection, including demo, presentation and case study contribution, was approximately 60 minutes. The case study data were stored on CloudStor, the UTS-recommended cloud storage (see Appendix E). The participant (LL) provided qualitative feedback about the DRA components from its practical application perspectives as follows.

5.4.3.1. DRA Architecture

The DRAv2.0 was presented to the case study DigiSAS Lab. DRA architecture is composed of four design architecture models: conceptual, logical, physical and operational. The case study participant provided feedback on the architecture design and its applicability to their organisation. The expert (LL) reviewed the design and provided positive feedback with further opportunities for improvements.

LL's feedback about the DRA architecture:

> *The output of this research is the DRA artefacts, and the outcome of this research is new scientific or design knowledge about the DRA itself. As a research group leader, I reviewed the DRA from following four perspectives, and my comments are noted below:*
>
> ***Usefulness:*** *DRA is applicable and is fit for setting-up the DevOps multi-cloud IoT environment for lab research projects.*
>
> ***Generalisation:*** *DRA is general in the sense that it is not fixed to one situation or environment and can adapt to different situations and be used with different technology stacks as appropriate to the situation. Thus DRA is applicable to a class of problem situations and is applicable to several instantiations.*
>
> ***Novelty:*** *DRA offers new knowledge, which has not been discussed before in the form of complex DevOps for Multi-cloud and IoT. In particular, the concept of a broker DevOps Cloud in the DRA.*
>
> ***Explainability:*** *DRA models seem to provide sufficient explanation about the elements and their relationships as a 'design knowledge', which can be used or reused for a class of a problem addressed in this work.*
>
> *My overall feedback is that DRA can be successfully instantiated for the similar research lab environment needs for the deployment of IoT applications using multi-*

cloud. Overall, DRA is fit for purpose; however, the following are some opportunities for further research and development, perhaps new PhD projects.

- *Extending the DRA for handling robotics and drones, this could be a direction for further research.*
- *Extending DRA to deploy different AI/ML models and applications to heterogeneous environments.*

5.4.3.2. DRA Operational Model Pipeline

In this step, the case study template provides a checklist for the DRAv2.0 instance pipeline implementation. The DigiSAS Lab evaluator (LL) may reuse the recommended toolset in Table 4.6 or configure DRAv2.0 instance pipeline with other tools of choice. Table 4.7 contains the setup process of DRAv2.0 instance pipeline. Table 4.7 is designed to facilitate the configuration of the DRA for research prospects.

LL's feedback about the DRA toolset: '*The instance of the DRA is working fine with above technology stack*'.

LL's feedback about the DRA setup and configuration process: '*The instance of the DRA setup/ configuration is working as intended*'.

5.4.3.3. Software Component

DRAv2.0 can be configured to deploy any applications. The evaluator (LL) was provided with a demo application (maven-app-heroku) to test the DRAv2.0 architecture (see Table 5.3). The demo application source code may be accessed on the code repository using observer public access stated in Table 5.3. However, in the case studies, prospects may use their IoT application (or non-IoT applications) to test DRAv2.0 instance.

LL's feedback about the IoT app: '*The use and applicability of the DRA to deploy the sample demo application are working as intended. This seems to be used for other different types of IoT application*'.

5.4.3.4. Hardware Component

Table 5.4 presents information about a sample IoT device network that was used to test the IoT application process. The IoT network sample (see Figure 4.17) was configured by the researcher to provide POC of the DRA operation model and its applicability for enabling IoT processes and interactions on the multi-cloud. However, in the case studies, organisations' prospects may use their IoT devices to test the IoT application deployed using DRAv2.0 instance pipeline.

LL's feedback about the IoT network: '*The DRA is working as intended for the selected hardware*'.

Overall Feedback

The participant provided feedback about the demo package and the presentation prepared for the DigiSAS Lab case study and imparted overall feedback about the DRA framework. The feedback represents LL's opinion about the DRA application in the research and development context.

LL's overall feedback about the DRA framework: '*Lab is bidding for drone and robotics application development and deployment projects. This is a huge research area and has the potential to extend DRA, perhaps another PhD, for the secure deployment of drone and robotics application projects*'.

5.3.4. DATA ANALYSIS

The case study data collected during the experiment are analysed in Table 5.7. The data were analysed using the cross-examination method between LL's feedback and the case study evaluation criteria in Table 3.2. This analysis aims to connect or relate the hypotheses (evaluation criteria) to the expert's feedback. The output of the analysis is organised into two columns: 'interpretation', which is the researchers' interpretation of the feedback, and 'DRA categories', which is the relationship between the feedback and the evaluation criteria.

Table 5.7: Research Case study Analysis

Participant Feedback	Interpretation	DRA Aspects
'The output of this research is the DRA artefacts, and the outcome of this research is **new scientific or design knowledge about the DRA itself**. As a research group leader, I reviewed the DRA from following four perspectives, and my comments are noted below: **Usefulness:** DRA is **applicable and is fit for** setup the DevOps multi-cloud IoT environment for lab research projects**.** **Generalisation:** DRA is **general** in the sense that it is not fixed to one situation or environment and can adapt to different situations and **be used with different technology stacks** as appropriate to the situation. Thus DRA is applicable to a class of problem situations and is applicable to several instantiations. **Novelty:** DRA **offers new knowledge**, which has not to be discussed before in the form of complex DevOps for Multi-cloud and IoT. **In particular, the concept of a broker DevOps Cloud in the DRA**.	The detailed feedback indicates that the DRA is useful and applicable at the research level. It also shows that the DRA provides a general abstract design model that could be implemented for research labs using different technology stacks. The DRA seems to offer new knowledge about the adoption of the DevOps approach for multi-cloud IoT applications. The DRA seems to offer a sufficient explanation about the framework concepts needed for the lab. The DRA seems to be fit for purpose and seems to	Generalisations Usefulness Novelty Coverage Reusable

Participant Feedback	Interpretation	DRA Aspects
Explainability: DRA models seem to provide **sufficient explanation** about the elements and their relationships as a "design knowledge," which can be **used or reused** for a class of a problem addressed in this work**.** My overall feedback is that DRA can be **successfully instantiated for the similar research lab environment needs** for the deployment of IoT applications using multi-cloud. Overall, DRA is fit for purpose; …'	be flexible and applicable at the research level.	
'The instance of the DRA is working fine with the above technology stack'.	DRAv2.0 instance is working with the software and hardware technology stacks used in the demo.	
'The instance of the DRA setup/ configuration is working as intended'.	This feedback indicates that the DRA is configurable, and the setup is working.	
'The use and applicability of the DRA to deploy the sample demo application is working as intended. This seems to be used for other different types of IoT application'.	This comment indicates that the DRA is useful and applicable for deploying the IoT application (demo) and may be used for different types of applications.	Usefulness
'The DRA is working as intended for the selected hardware'.	This comment indicates that the IoT devices are interacting with the IoT application deployed on the multi-cloud.	
'Lab is bidding for **drone and robotics** application development and deployment projects. This is a huge research area and has the **potential to extend DRA**, perhaps another PhD (s), for the secure deployment of drone and robotics application projects'.	This feedback indicates that there is potential for improvements for the DRA. LL's suggestion is discussed as an avenue for future research in Chapter 5.	

5.3.5. REPORTING

The case study report summary is an organised case study outcome presented to the audience. The report aims to draw a conclusion from LL's point of view about the DRA framework in the context of the DigiSAS Lab. Table 5.8 presents the systematic testing procedure of the case study at the DigiSAS Lab.

Table 5.8: Research Lab Case Study Reporting Summary

Case Study 02	Description
Organisation	UTS SCS DigiSAS Lab
Test date	15/08/2019
Organisation context	DigiSAS Lab conducts research and development in collaboration with industry partners' projects. They are working on several software-related projects involving web and IoT and the multi-cloud. The partners are from large to SMEs and start-ups.
Test team (TT) (Participants)	This evaluation involved the LL who leads and manages several software-related research and development projects at the DigiSAS Lab, including this PhD project.
Organisation need	DigiSAS Lab needs a multi-cloud deployment environment to meet the needs of different industry partners, as they have different cloud deployment needs. The challenge or problem is how to perform multi-cloud deployments.
Test objective	The objective is to evaluate the applicability of the DRA in the research lab environment. The Lab's objective is to have a working DevOps environment for multi-cloud IoT application deployments.
Test case question	How can applications be deployed to the multi-cloud using DevOps?
Test package (Pre-prepared)	To evaluate the DRA framework, a presentation slide pack and demo were developed to demonstrate the deployment of a predeveloped sample IoT application to the multi-cloud environment: - Demo YouTube video: Link - Presentation slides: Link
Main test component	The participant (LL) evaluated the following DRA components: 1. DRAv2.0 architecture (see Figure 4.19) 2. DRA operational model pipeline (see Figure 4.14) 3. Software components (see Table 5.3) 4. Hardware components (see Table 5.4)
Test method	Case study template (see Appendix G) Industry survey (see Appendix D)
Test duration	60 minutes (presentation, demo, survey, case study)
Data type	Qualitative feedback provided by the evaluator (LL)
Pretesting	The researcher presented the case study project to the participant (LL) using the presentation slides. The researcher organised a question-answer session to explain necessary DRA aspects if required.
Key activities	- TT verifies that the DRA supports DevOps concepts (see Table 2.9). - TT verifies that the DRA enables DevOps practices (see Table 2.10). - TT verifies that DRAv2.0 toolset (see Table 4.6) is easy to configure using the guidelines provided by the case study (see Table 4.7). - TT verifies that DRAv2.0 instance pipeline enables CI using CI broker (see Table 4.7 and Figure 4.19).

Case Study 02	Description
	- TT verifies the automated and continuous deployment and delivery of the IoT application to the multi-cloud (see Table 5.3 and case study demo package). - TT verifies the IoT application interaction with the IoT network sample (see Tables 4.19 and 4.20 and case study demo package). - TT verifies that DRA design models can be reusable and instantiable. - TT verifies that the DRA conceptual model offers new knowledge for the organisation to further understand the DevOps approach for agile application deployment and delivery.
Expected outcome	The expected outcome of the case study is to determine that the DRA provides possible solutions to the research lab's needs and, by extension, verify that the framework address the research gaps (see Section 4.3).
Actual outcome	The actual outcome is determined in the case study analysis section (see Table 5.7). The LL of DigisSAS Lab imparted valuable qualitative data (as feedback) about the DRA. The cross-examination between the feedback and Table 3.2 indicates that the evaluator (LL) considers DRA design models reusable in the research context and provides a sufficient explanation about the concepts of the framework drawn from DevOps methodologies. TT indicated that DRA offers new knowledge base in the form of CI broker used for deploying IoT applications to the multi-cloud. TT specified that the DRA is a general concept and is not fixed to a particular situation. It can adapt to different situations and be used with different technology stacks as appropriate to the situation. Thus, the DRA applies to a class of problem situations and applies to several instantiations.
Observation and interpretation	TT contributed to the case study data collection based on the information provided by the demo package. It has been observed that the framework can be easily reconfigured and is reusable for any experimental situation. TT indicated that the DRA is a general design and is adaptable for the current situation of the experiment. TT specified that the DRA is useful and applicable for DevOps, multi-cloud and IoT projects in the research lab environment. The new knowledge in the form of CI broker offers a possible solution to the complex problem of multi-cloud deployment. TT indicated that the DRAv2.0 instance is working correctly with the technology stack of the demo. The pipeline in the demo video shows that the IoT application was successfully deployed to the multi-cloud and successfully interacted with the IoT network of sample devices, which responded correctly to the IoT application actions and events. TT outlined that the DRAv2.0 instance is working as intended with the software/hardware in the demo package. The observations are drawn from TT's feedback and indicate that the DRA framework can be successfully instantiated for similar research lab needs. Overall, the framework is fit for purpose and seems to provide an adequate solution to deploy IoT applications on the multi-cloud using the DevOps approach.

5.5. TEACHING CASE STUDY SURVEY

This section presents a teaching case study that emphasises the method of teaching the DevOps approach and agile software development. The teaching case study evaluates the integration of DRA in two subjects offered by the UTS FEIT School of Software:

- SEP 48440—Spring 2017 (http://handbook.uts.edu.au/subjects/48440.html).
- INP 31261—Autumn 2019 (http://handbook.uts.edu.au/subjects/31261.html).

The teaching case study followed a simplified case study structure based on the commonly used structure discussed in chapter 3 (DSR Evaluation step). The simplified case study for teaching is composed of two main steps:

1. Case study introduction: Introduce the subjects of the case study (SEP and INP).
2. Data collection and analysis: Present the data collection and analysis process for SFS data. The subjects' SFS data collection and analysis process are based on the commonly used survey structure discussed in chapter 3 (DSR Evaluation step):
    a. survey planning
    b. design the sampling procedure
    c. select the survey method
    d. develop the questionnaire
    e. conduct the survey
    f. collect and analyse the data.

The subjects' SFS data were transformed from categorical ratings into ordinal (numerical) ratings for the data analysis process using the survey rating table (see Table 3.3). The SFS numerical data were analysed using statistical formulas discussed in chapter 3 (see Equation 3.2 and Equation 3.3).

5.5.1. SEP CASE STUDY (DRAV1.0)

UTS offers an undergraduate subject (Software Engineering Practice) in spring sessions for approximately 200 students (SEP: http://handbook.uts.edu.au/subjects/48440.html). The DRA (see Chapter 4) was taught to students, which enabled them to apply automated deployment, CI, automated testing (acceptance and unit) and real-time monitoring practices to the delivery or release of a software product on a single cloud (Ghantous & Gill 2019).

5.5.1.1. Case Study Introduction

The SEP program organises students into groups (4–7 students per group). All group projects are independent and can be developed using a selection of software technologies (e.g., web app, IoT app, mobile app, desktop app using Java, JSP, Python, HTML, JS, Node.JS, Angular.JS, REACT, RUBY, C#, ASP.NET). The groups had to self-organise as a virtual start-up company and appoint a student project manager to coordinate the group project activities using agile

practices (e.g., Scrum). Groups were expected to deliver the software using the agile release management practice (Alzoubi, Gill & Al-Ani 2015). Students were only required to deliver release 0 (R0) and release 1 (R1). R0 deliverable is a documented software prototype based on the project proposal. R0 prototype is an excellent way to learn how to identify project-related risks and technical dependencies earlier in the project before committing too many resources upfront. R1 deliverable is a full-working software. The groups had to appoint a release manager or Scrum master to manage the delivery of releases. In R1, groups were required to implement DRAv1.0 instance and replicate the DRA operational model. Each group constructed their automated pipeline using their DevOps toolset of choice. Integrating DRAv1.0 (single cloud) in the course taught students how to apply DevOps practices (Ghantous & Gill 2017) to improve their agile development process and release management of software (Ghantous & Gill 2019).

The groups developing R0 and R1 were expected to apply:

- agile requirements analysis and planning
- agile architecture and design
- agile implementation and testing using a DevOps approach using the DRA as a template.

The DRA used for SEP (see Figure 4.18) is discussed in Chapter 4. Students were provided with a testing video that demonstrated how the operational pipeline works. They were also provided with project slides that explained how the DRA adopts DevOps concepts and practices and showed how the DRA could be configured for any software application:

- Demo YouTube video: Link
  The video demonstrates the testing of Java-web app deployment in DRAv1.0 instance. This video was provided to students for spring 2017 SEP.
- Presentation slides: Link
  The workshop slides introduce DevOps concepts and practices and show how to set up and configure the DRA for single-cloud deployment.

5.5.1.2. Data Collection and Analysis

The SFS data collection and analysis process was based on the commonly used survey structure discussed in chapter 3 (DSR Evaluation step).

a. Survey planning

The survey aimed to evaluate the subject quality and relevance from the students' perspective. UTS needs the survey data to evaluate the subjects offered every semester and determine possible improvements based on students' contributions. UTS values the quality of the student learning experience. The SFS offers students an opportunity to provide teaching staff, faculty and university management with constructive feedback to improve education at UTS.

b. Design the sampling procedure

The SFS is an anonymous survey (survey no. 200063) that allows students (total 203 enrolled in spring 2017) to input ratings based on a scale composed of five possible entries for each question (see Table 3.3). The SFS follows the UTS Vice-Chancellor's Directive: Link.

c. Select the survey method

The survey was offered online to students enrolled in SEP spring 2017 using the UTS standard SFS delivery portal: Link.

d. Develop the questionnaire

The SFS questionnaires for SEP were developed and offered by UTS FEIT. The researcher did not contribute to the development of the questionnaires. The SFS questionnaires are standard across the FEIT cohort of subjects. The SEP SFS questions are as follows:
Q1: The learning opportunities provided helped me meet the stated objectives of this subject.
Q2: I made the most of my opportunities to learn in this subject.
Q3: Overall, I am satisfied with the quality of this subject.
Q4: This subject provided practical learning activities to develop new skills and knowledge I may need in the workplace.
Q5: This subject has developed my understanding of my intended profession.
Q6: This subject's learning opportunities motivated me to conduct further self-directed learning.
Q7: This subject has developed my ability to think critically.
Q8: The layout of UTSOnline for this subject helped me navigate and locate materials with ease.
Q9: The look and feel of the UTSOnline subject site improved my user experience.
Q10: Overall, I am satisfied with how this staff member facilitated my learning.

e. Conduct the survey

The SFS for SEP was offered online to students between 9 October 2017 and 12 November 2017. The SFS is an anonymous survey (survey no. 200063) that allows students (total of 203 enrolled in spring 2017) to provide feedback and rating about the subject(s) materials.

f. Collect and analyse the data

The SFS results were published in Ghantous and Gill (2019). The students' responses (85 participants) were transformed from categorical data to ordinal (numerical) data using the survey ratings presented in Table 3.3. Table 5.9 shows the frequency and percentage value distributions for each question. The total number of answers collected from 85 students was T = 1004. The SFS generated two key values: AAF = 892 and AAP = 88.85% (see Equation 3.2 and 3.3). The AAP and AAF indicate that most of the participants in the SEP SFS were satisfied with the quality of the subject content and agreed that the subject provided practical learning that

met the stated objectives of the course. The SFS raw data are stored on CloudStor, the UTS-recommended data storage (see Appendix E).

Table 5.9: SEP SFS Spring 2017 Results Analysis

Total Answers = 1004	Strongly Disagree	Disagree	Average	Agree	Strongly Agree
Q1	5	5	21	40	29
Q2	4	5	15	41	35
Q3	7	8	24	34	27
Q4	5	2	19	44	31
Q5	5	6	20	40	29
Q6	6	5	15	45	29
Q7	6	7	19	38	31
Q8	6	6	26	38	25
Q9	6	7	26	36	25
Q10	4	7	25	27	38
Frequency	**54**	**58**	**210**	**383**	**299**
Percentage	**5.38%**	**5.78%**	**20.92%**	**38.15%**	**29.78%**

Figure 5.2 shows the distribution of the participants' answers. It illustrates the perspectives of the participants in the SFS.

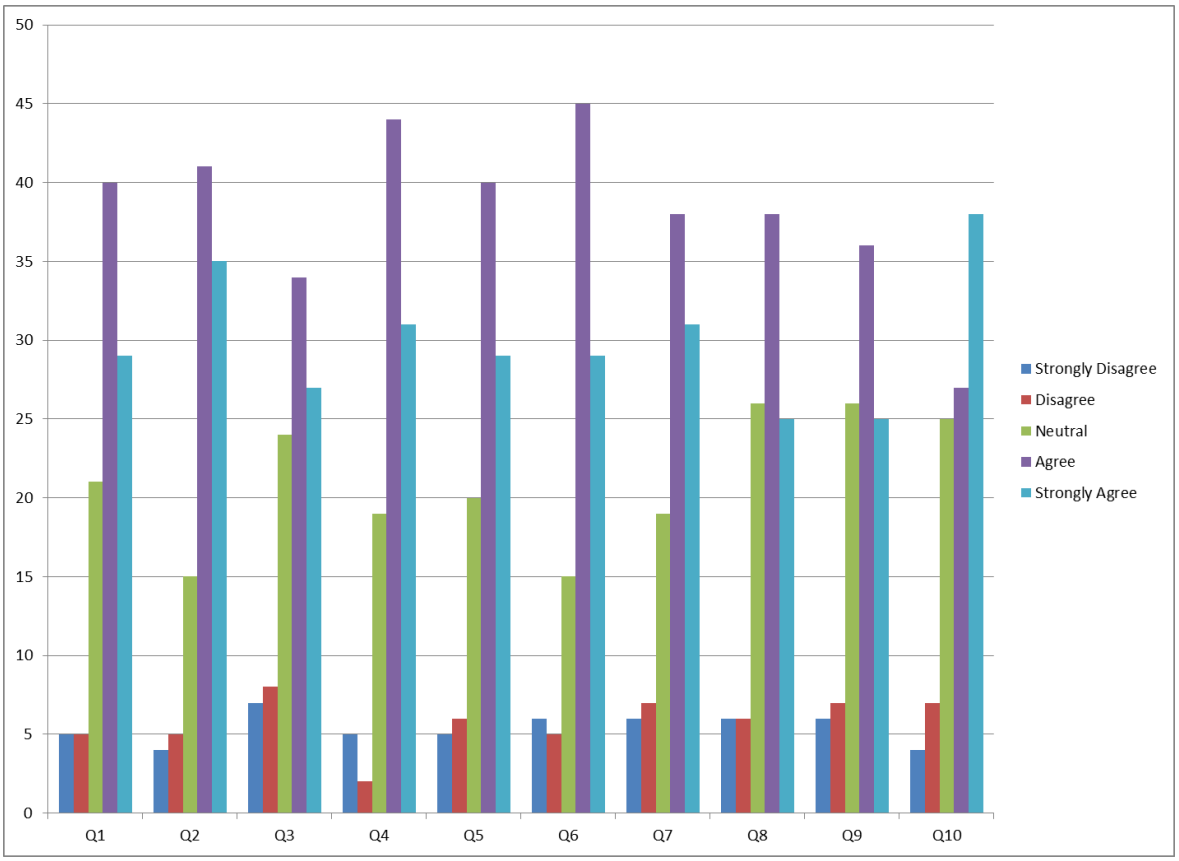


Figure 5.2: SEP Spring 2017 SFS Results Graph

Figure 5.2 shows that most of the students seem to agree or strongly agree with the survey questions about the teaching materials. Table 5.9 indicates that 88.85 % of the students scored average and above in the survey questionnaires and 67.93 % of the students scored agree/strongly agree in the survey questionnaires. The figures indicate that the students enrolled in SEP 2017 subject are satisfied with the subject materials that include teaching DevOps and DRAv1.0 instance, as explained in the case study introduction (section A.1).

A total of 1004 responses from 85 participants to the 10 questions were obtained in the SFS survey in spring 2017. The contribution to learning is reflected in the analysis of every specific questionnaire responses in the survey. The individual questionnaire analysis is explained below:

- Q1 scored 90% average and above, which means that the subject materials met the objectives stated in the subject outline of SEP.

- Q2 score 91% average and above; which means that most of the students had the opportunity to learn from the materials of SEP (including DevOps and DRAv1.0 instance).

- Q3 scored 85% average and above, which means that most of the students are satisfied with the quality of the materials taught in SEP 2017.

- Q4 scored 91% average and above, which means that the subject materials provided practical learning activities to develop new skills and knowledge needed in the workplace.

- Q5 scored 89% average and above, which means that the subject has developed the students understanding of professions related to agile and DevOps.

- Q6 scored 89% average and above, which means that learning opportunities motivated the students to conduct further self-directed learning and research in the fields related to agile and DevOps.

- Q7 scored 87% average and above, which means that the subject helped the students improve their analytical and problem-solving in the area of software development.

- Q8, Q9, Q10 are related to the subject delivery. The statistical results show that the students are satisfied with the materials delivery method (either in a class by the staff or remotely on UTSOnline blackboard).

Table 5.10 discusses and interprets the SEP surveys qualitative feedback. The students are analysed by highlighting critical phrases in the quotes related to the quality of the subject, the subject objectives, the students' experience, the flexibility of software development adopted in SEP 2017, the encouragement towards teamwork and the creativity in project design. Table 5.10 explains that the students enrolled in SEP are satisfied with the learning experience and with the teaching materials quality.

Table 5.10: SEP Students Qualitative Feedbacks

Students Quotes	Interpretation
'I liked the fact that it ran as a start-up, and there was so much **flexibility in what we were able to develop**. I liked the idea of workshops where we have time to meet our team and work on our great project. **It was a great subject for me to learn and advance in a skill which I was not familiar with before**'.	The participant seems to have learnt new skills from the course in a flexible development manner.
'How the **subject has prepared me for the real world, professional environment**. The practical experience I got from the assignments was very valuable. I've haven't had experience collaborating with others on a project before, and this was very good preparation for it'.	The participant indicated that the subject content prepared them for the real-world professional environment.
'**Industry experience being integrated into lectures**'.	The student seems satisfied with the integration of industry experience into the lecture slides.
'**The idea of creativity in project design and the use of teamwork to develop software with an agile approach**'.	The participant seems to consider the agile software approach creative.

5.5.2. INP CASE STUDY (DRAV2.0)

UTS also offers an undergraduate Inter-Networking Project (INP subject 31261), which is conducted as a capstone group project with 4–6 students (INP: http://handbook.uts.edu.au/subjects/31261.html). The INP group is expected to develop its application using agile methodology and principles and adopt DRA. The group is expected to replicate DRAv2.0 instance using the same DevOps toolset and develop an IoT application that interacts with the multi-cloud and manages (collects and saves) IoT data from sensors at run time. Data from IoT sensors are saved automatically to the NoSQL cloud database. The IoT application can be developed using any programming language (technology stack). Students in the INP group are expected to reconfigure the DRA to enable their IoT application deployment to the multi-cloud (Ghantous & Gill 2018). DRAv2.0 instance testing demonstrates how an IoT application is deployed through the DRA pipeline to the multi-cloud. The test aims to determine the usefulness of the DRA for the deployment of IoT applications on the multi-cloud. DRAv2.0 is an upgraded version of DRAv1.0 instance; the upgraded options are multi-cloud deployment options and IoT application.

5.5.2.1. Case Study Introduction

The INP capstone project aims to teach students how to develop and deploy IoT applications to the multi-cloud using the DevOps approach. The IoT application may be developed using, for example, Java, JSP, Python, HTML, JS, Node.JS, Angular.JS, REACT, RUBY, C#, ASP.NET. The IoT application was required to interact at run time with selected IoT sensors and report the

data on the application dashboard. The IoT data were also saved automatically on a NoSQL cloud database. The students learn to: 1) configure IoT devices and sensors; 2) configure their DevOps pipeline; 3) configure and develop the IoT application to interact with the IoT devices; 4) deploy the application to the multi-cloud; 5) monitor the applicaton performance in real-time; 6) apply automated testing use CI-Broker; and 6) enable automated notification. Therefore, the group had to self-organise as a virtual start-up company and appoint a student project manager to coordinate the group's project activities using agile practices (e.g., Scrum). The students were expected to deliver the software using the agile release management practice (Alzoubi, Gill & Al-Ani 2015). Students were only required to deliver release 0 (R0) and release 1 (R1). R0 deliverable is a documented software prototype based on the project proposal. R0 prototype is an excellent way to learn how to identify the project-related risks and technical dependencies earlier in the project before committing too many resources upfront. R1 deliverable is a full-working software. The groups had to appoint a release manager or Scrum master to manage the delivery of releases. In R1, students were required to implement DRAv2.0 architecture and replicate the DRA operational model. The groups were expected to construct their automated pipeline using their DevOps toolset of choice. Integrating DRAv2.0 (multi-cloud) in the course taught students how to apply DevOps practices (Ghantous & Gill 2017) to improve their agile development process and release management of software (Ghantous & Gill 2019). Groups developing R0 and R1 were expected to apply:

- agile requirements analysis and planning
- agile architecture and design
- agile implementation and testing using a DevOps approach using the DRA as a template.

The DRA used for INP (see Figure 4.19) is discussed in Chapter 4. Students were provided with a testing video that demonstrated how the operational pipeline works. They were also provided with project slides that explained how the DRA adopts DevOps concepts and practices and how the DRA can be configured for any software application:

- Demo YouTube video: Link
  The video demonstrates the testing of Java-web app deployment. This video was provided to students for autumn 2019 INP.
- Presentation slides: Link
  The workshop slides introduce DevOps concepts and practices. The slides also show how to set up and configure DRA for multi-cloud deployment.

5.5.2.2. Data Collection and Analysis

The subjects' SFS data collection and analysis process were based on the commonly used survey structure discussed in chapter 3 (DSR Evaluation step).

a. Survey planning

The survey aimed to evaluate the subject's contribution to teaching and learning and to evaluate subject content, from the students' perspective. The general need is to add further constructive feedback and data to support the relevance and usefulness of the DRA for teaching.

b. Design the sampling procedure

The SFS is an anonymous survey that allows a group of five students (enrolled in INP autumn 2019) to input ratings based on a scale composed of five possible entries for each question (see Table 3.3). The SFS follows the guidelines of the ethics approval ETH18-2339 (see Appendix A and Appendix B).

c. Select the survey method

The survey was offered online to students enrolled in INP autumn 2019. The SFS can be viewed on the following web page: Link.

d. Develop the questionnaire

The SFS questionnaire was developed by the researcher using artefact evaluation criteria for software engineering (133). There were two types of questionnaires in the survey:

- Quantitative data questionnaires (ratings): level of effort, contribution to learning, skill and responsiveness of the instructor, course content and overall ratings
- Qualitative data questionnaires (feedbacks): course useful aspects, suggested improvements.

Note: Data related to the SFS instructor and students (i.e., level of effort, skill and responsiveness of instructor) were omitted from the analysis because they were not related to the DRA evaluation. Data relating to contribution to learning, course content data, overall ratings, course useful aspects and suggested improvements were considered in this evaluation.

- **Contribution to learning questionnaire set**

The contribution to learning is a set of seven questionnaires listed as follows:
Q1: Contribution to learning [The learning opportunities provided helped me meet the stated objectives of this subject].
Q2: I made the most of my opportunities to learn in this subject.
Q3: This subject provided practical learning activities to develop new skills and knowledge I may need in the workplace.
Q4: This subject has developed my understanding of my intended profession.
Q5: This subject's learning opportunities motivate me to conduct further self-directed learning.
Q6: This subject has developed my ability to think critically.
Q7: Overall, I am satisfied with the quality of this subject.

The course content is a set of ten questionnaires listed as follows:
Q1: The project learning objectives were clear.
Q2: R0 and R1 report required materials helped me to understand agile development process.
Q3: DRA helped with agile development, testing, deployment, delivery.
Q4: DRA architecture was easy to configure for my IoT application.
Q5: DRA CI broker feature helps with multi-cloud deployment.
Q6: Learning how to setup and configure IoT devices was interesting.
Q7: DRA is adaptive to any application type.
Q8: DRA is adaptive to any DevOps toolset.
Q9: I learnt a lot about DevOps, multi-cloud and IoT by using DRA.
Q10: Overall, I am satisfied with DRA as INP subject architecture.

The course useful aspects question is: What aspects of this course were most useful or valuable?

The course suggested improvements question is: How would you improve this course?

e. Collect and analyse the data

The students' responses (five participants) were transformed from categorical data to ordinal (numerical) data using the survey ratings in Table 3.3.

The contribution to learning data was analysed in Table 5.11 using the statistical equations discussed in chapter 3 (see Equation 3.2 and 3.3). Table 5.11 results are computed to calculate AAF = 20 and AAP = 100%. AAF and AAP indicate that the students are satisfied with the course's contribution to their learning. The students responded with 98% score of agree/strongly agree that the subject has motivated them to think critically and improved their understanding of software development using the agile-DevOps approach. The students strongly agreed (75%) that the subject has provided practical learning modules that helped to improve their skills and knowledge in software development.

Table 5.11: Contribution to Learning Results Analysis

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Frequency	Percentage
Strongly Disagree	0	0	0	0	0	0	0	0	0.00%
Disagree	0	0	0	0	0	0	0	0	0.00%
Neutral	0	0	0	2	0	0	0	2	10.00%
Agree	2	2	1	0	1	0	1	3	15.00%
Strongly Agree	2	2	3	2	3	4	3	15	75.00%

The course content value to students' learning was represented in Table 5.12 using the statistical equations discussed in chapter 3 (see Equation 3.2 and 3.3). The results in Table 5.12 are computed to calculate AAF = 26 and AAP = 100%. AAF and AAP indicate that the students are satisfied with the course contribution to their learning.

Table 5.12: Course Content Results Analysis

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Frequency	Percentage
Strongly Disagree	0	0	0	0	0	0	0	0	0	0	0	0.00%
Disagree	0	0	0	0	0	0	0	0	0	0	0	0.00%
Neutral	0	0	0	1	0	0	0	0	0	0	1	3.85%
Agree	0	2	1	3	2	2	2	1	1	1	7	26.92%
Strongly Agree	5	3	4	1	3	3	3	4	4	4	18	69.23%

The contribution to learning data is plotted in Figure 5.3. The graph shows that the group either agrees or strongly agrees that the DRA has contributed to their learning. Figure 5.3 clearly shows that the students agree or strongly agree with the level of course contribution to learning.

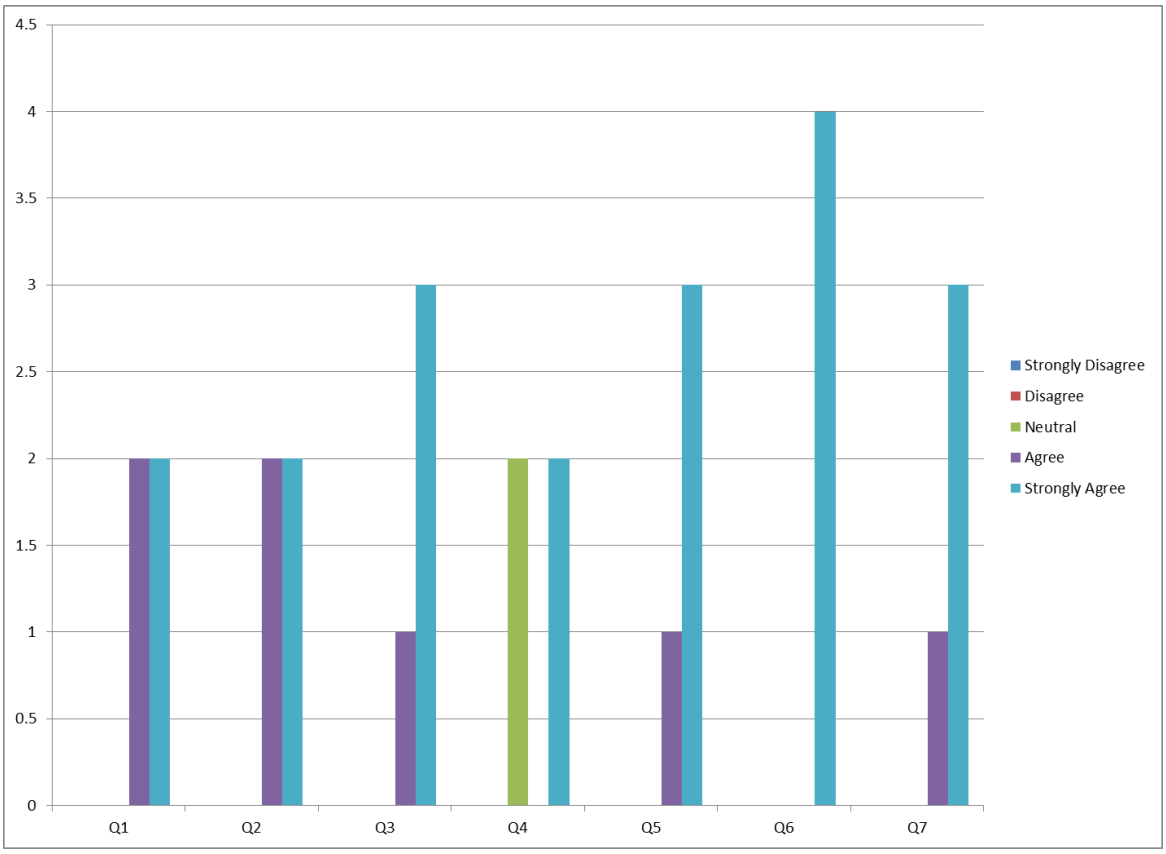


Figure 5.3: Contribution to Learning Graph

The course content data are plotted in Figure 5.4. The graph shows that the group either agrees or strongly agrees that the DRA framework contents (architecture, pipeline, and demo) helped improve their skills in agile software development using the DevOps approach. Figure 5.4 illustrates that 96.15% of students agree/strongly agree that the course content of INP provides a comprehensive new mechanism (CI-Broker) that improved software application deployment to multi-cloud. The students agree/strongly agree (96.15%) that DRA is not fixed to a particular situation. It can be concluded from the survey results that students consider DRA instantiable and applicable in the context of a software development project.

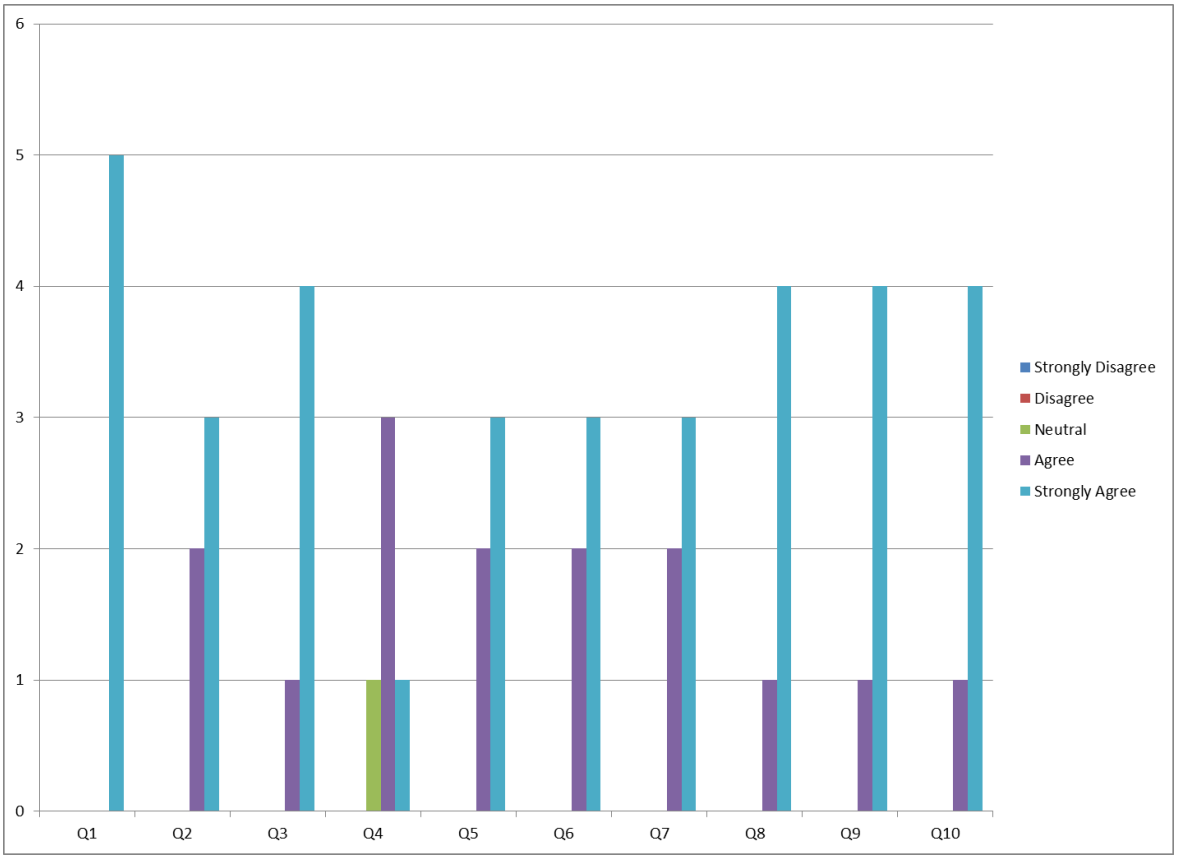


Figure 5.4: INP Survey Results Charts

The INP case study qualitative evaluation is based on the students' overall feedback and overall ratings to the DRA used in their capstone project. Feedback is quoted from the students who participated in the INP SFS survey for autumn 2019. The overall rating data are shown in Table 5.13, which shows that, overall, the group is 100% strongly satisfied with the course. The course content included instantiating DRA architecture to create DRAv2.0 development pipeline to deploy the project's IoT-application to multi-cloud.

Table 5.13: Overall Rating Results

	Overall how satisfied are you with the course
Strongly Disagree	0
Disagree	0
Neutral	0
Agree	0
Strongly Agree	5

The participants contributed to the qualitative questionnaire in the SFS by providing feedback about the DRA's useful aspects and suggesting possible improvements to the DRA from their perspective. The students' feedback is quoted and mapped in Table 5.14.

Table 5.14: INP Students Qualitative Feedbacks

Questions	Students Quotes	Interpretation
What aspects of this course were most useful or valuable?	- The testing and the knowledge of **how the DRA V2.0 connects/works together**. - I learnt how to understand code much better than I ever have before as I was thrown in the deep end and had to use various types of technology in order to develop my understanding of learning to write code - The most valuable aspect of this course was able to work as a group on a real project. It helps me build my communication and time management skills. - Regular contact with supervisor - Weekly meetings with the supervisor **helped my understanding of IoT devices and how multi-cloud works**.	The participants considered that knowledge of the DRA operational model is useful. The course also helped them further understand the concepts of IoT and multi-cloud.
How would you improve this course?	- Clearer understanding and communication from the subject coordinator to our group and supervisor - The subject coordinator can be more responsive. - More IOT resources available through the university - More communication from the subject coordinator	The students seem to require IoT resources at the faculty level.

5.5.3. DRAv1.0 v. DRAv2.0

This section analyses the results of the teaching case study (see Sections 5.1.2 and 5.1.3). The data extracted from the student survey in subject SEP47440 (see Figure 5.2) and INP31261 (see Figures 5.3 and 5.4) indicated that the integration of the DRA in agile software engineering at the academic level has been successful. The quantitative results from both surveys indicated that the DRA helps agile application delivery by automating the development process. The qualitative survey feedback showed that students were satisfied with the subject materials.

This section also presents a comparison between the DRA architecture instances: DRAv1.0 and DRAv2.0. The cross-comparison in Tables 5.15 and Table 5.16 shows that DRAv2.0 instance includes all DRAv1.0 instance features. DRAv2.0 pipeline is an extended version of DRAv1.0 and enables multi-cloud deployment features using a CI broker for CI. Further, DRA2.0 uses IoT application and IoT devices and sensors for testing and demonstration, which makes it more tailored to address the research question. Hence, it can be concluded that DRAv2.0 instance is more suitable to use in the industry case study for evaluation and review. Section 5.5 uses DRAv2.0 instance as a case study in the industry.

Table 5.15: DRAv1.0 vs DRAv2.0 (Design)

DRAv1.0 VS DRAv2.0						
DRA Characteristics	**Instance of DRAv1.0**	**Instance of DRAv2.0**		**DRA Architecture**	**Instance of DRAv1.0**	**Instance of DRAv2.0**
Human factor	☑	☑		Contextual model	☑	☑
Infrastructure	☑	☑		Conceptual model	☑	☑
Tools	☑	☑		Logical model	☑	☑
Process	☑	☑		Physical model	☑	☑
Product	☑	☑		Operational model	☑	☑
Abstraction	☑	☑				
Business value	☑	☑				
Rules	☑	☑				
Legal	☑	☑				

Table 5.16: DRAv1.0 v. DRAv2.0 (Instance)

DRA Pipeline	**Instance of DRAv1.0**	**Instance of DRAv2.0**	**DRA Demo Application Software/Hardware**	**DRAv1.0**	**DRAv2.0**
Repository	☑	☑	Java-web app	☑	☒
Collaboration	☑	☑	IoT app	☒	☑
CI broker	☑	☑	IoT network sample	☒	☑
NoSQL DB	☑	☑			
Monitoring	☑	☑			
CD-cloud1	☑	☑			
CD-cloud2	☒	☑			
CD-cloud3	☒	☑			

5.6. INDUSTRY FIELD SURVEY

The field industry survey is the fourth iteration in the empirical evaluation conducted in this thesis. The survey is a collection of specified information offered to specialised and specific populations (Runeson & Höst 2009; Sjøberg et al. 2005). The survey was offered online to a cohort of domestic and international industry experts. It was constructed using a commonly survey design as follows (Hyndman 2008).

a. Survey planning

The aim was to obtain experts' feedback and opinions about the DRA phenomenon. The survey plan was to attain qualitative and quantitative data from participants. The survey data analysis aims to support that the DRA meets the evaluation criteria (see Chapter 3, Table 3.4).

b. Design the sampling procedure

The survey (Link) was offered to participants and experts in the IT industry who specialise in the areas of software engineering, DevOps, cloud computing and architecture, and IoT. The participants came from a cohort of companies located in Australia, the US, UK, Russia, Japan, Spain, Switzerland, India, Portugal, Sweden, Brazil, Costa Rica, Italy, South Korea, Canada, Germany and the Netherlands. The participants were initially contacted via LinkedIn using the formal invitation letter approved by the UTS ethics approval **ETH18-2339** (see Appendix A, Appendix B and Appendix C). In line with the ethics approval outlined in Appendix A, no personal information was collected about the participants. The survey was offered to participants after they replied to the survey invitation letter (see Appendix C) and agreed to participate in the survey and receive the survey form (see Appendix B). The original survey data were stored on CloudStor (see Appendix E). The demographic representation of the participants is mapped in Table 5.17, including information about their professional experience, organisation location and years of experience in their IT field (see Table 5.17). The minimum number of years of experience in the industry was three years. The participants' years of experience, as shown in Table 5.17 ranged from 3 to 16 years. It indicates that the participants may provide abundant feedback and comments based on their years of experience and their expertise in the IT industry.

Table 5.17: Participants Demographic Distribution

Participant	Organisation Location	Area of Expertise	Experience
1	Denmark	DevOps Engineer, Cloud Engineer, Developer	3 years+
2	India	DevOps Engineer, Cloud Engineer	5 years +
3	US	Sr. DevOps Engineer, Software Engineer	9 years +
4	India	Thought Leadership, Agile DevOps Coach, AWS Engineer	5 years +
5	US	Senior DevOps Engineer	4 years+
6	Sweden	DevOps Engineer (Cloud Lead, Security Lead)	3 years +

Participant	Organisation Location	Area of Expertise	Experience
7	India	Director of Development Operations, Head of DevOps	5 years +
8	India	Senior DevOps Engineer, AWS Certified Solutions Architect	4 years+
9	India	DevOps Consultant, Consulting & Services Integration (C&SI)	7 years +
10	Australia	Senior Developer, Senior DevOps Engineer	7 years +
11	Australia	Senior DevOps Engineer	5 years +
12	Australia	Software Architect, Senior Cloud Engineer	7 years +
13	Australia	Software Engineer—Mobile, Web, IoT	3 years +
14	Netherlands	Software Developer, Software Engineer	3 years +
15	Costa Rica	Software Engineer	7 years +
16	UK	Reliability Engineer, System Engineer, Continuous Integration	7 years +
17	Italy	Agile DevOps Coach	5 years +
18	Australia	DevOps Engineer, Developer, Robotic Process Automation	4 years +
19	Australia	Agile Coach, Senior Developer	7 years +
20	Australia	Senior Software Engineer	4 years +
21	US	Software/Build & Infrastructure Engineer	7 years +
22	Costa Rica	Cloud Architecture, DevOps Engineer, Professional Scrum Master	4 years +
23	Costa Rica	DevOps Engineer	7 years +
24	India	DevOps Engineer, Cloud Engineer	5 years +
25	South Korea	Software Engineer, Developer	5 years +
26	Australia	DevOps Engineer	4 years +
27	US	DevOps Engineer, Product Transformation, Consultant	6 years+
28	India	DevOps Consultant, Engineer	4 years +
29	Netherlands	DevOps, Team Lead Big Data Engineer	12 years +
30	Japan	Head of DevOps	3 years +
31	India	Developer (Python) & DevOps Engineer	3 years +
32	Australia	Cloud Architect, Senior DevOps Consultant	4 years +
33	Australia	BizDevOps, Enterprise Technology	11 years +
34	US	Cloud Consultant, DevOps Engineer	3 years +
35	Portugal	Senior DevOps Engineer, Site Reliability Engineer	3 years +
36	US	DevOps Engineer, Automation Developer	3 years +
37	US	Software Engineer	4 years +
38	Australia	Lead Tech. Consultant, Senior Software Engineer	4 years +
39	Australia	Principal Software Engineer, Software Architect, Director	11 years +
40	Australia	CEO, IoT	14 years +
41	Australia	Senior Test Consultant, Technical Manager, Author & Blockchain, Cryptocurrency Advisor, Director	16 years +
42	Costa Rica	Site Reliability & DevOps Engineer, Developer	8 years +
43	Australia	Lead Technical Architect—Azure Apps & Infrastructure	5 years +
44	Australia	DevOps and Automation Specialist, Senior Software	7 years +

Participant	Organisation Location	Area of Expertise	Experience
		Engineer	
45	Australia	Software Delivery Manager, Senior Consultant	6 years +
46	Australia	Senior DevOps Engineer	4 years +
47	Australia	Senior Architect—AWS (CSA-A), TOGAF9	6 years +
48	Australia	Cloud Consultant—DevOps Engineer	5 years +
49	Germany	DevOps Engineer, Cloud Engineer	4 years +
50	Netherlands	Head of DevOps	6 years +
51	Australia	DevOps Engineer	6 years +
52	Australia	Senior DevSecOps Engineer	3 years +
53	Australia	Software Engineering Leader, Digital Transformation, Lean Product Development	11 years +
54	Australia	DevOps Engineer	6 years +
55	Australia	Senior Program Architect, DevOps, Cloud Solution Architect	4 years +
56	Australia	Senior Cloud Architect (AWS, Azure, GCP), Technical Architecture	6 years +
57	Brazil	Principal Engineer, Consultant	7 years +
58	Australia	DevOps, Data Engineer	5 years +
59	Russia	Senior Application Support Analyst	7 years +
60	Australia	DevOps Consulting Partner, Cloud Migration/Native	8 years +
61	Switzerland	Product Owner, Agile Coach, DevOps Engineer	13 years +
62	US	Senior Software Engineer	11 years +
63	Canada	Senior DevOps, Cloud Architect	7 years +
64	Switzerland	Head of Development Platform Services, Senior Engineer, QA Test Automation Manager	13 years +
65	Australia	Senior Software Engineer, DevOps Engineer, Senior Software Developer	15 years +
66	Australia	Cloud AppOps & DevOps Lead	10 years +
67	Australia	Development Lead (DevOps)	6 years +
68	Brazil	CTO, IT Executive, Agile and Digital Transformation	6 years +
69	Australia	Cloud Solutions, DevOps Engineer	8 years +
70	Australia	DevOps Tech Lead	8 years +
71	Australia	Cloud Solution Architect—Application Development	4 years +
72	Australia	Lean-Agile Coach, Service Transformation, Project Manager	6 years +
73	Australia	Head of Cloud Transformation, Enterprise DevOps Architect	4 years +
74	Australia	Senior Technical Engineer (Cloud), DevOps Practice Lead	6 years +
75	Australia	Enterprise Agile Coach	8 years +
76	Australia	Software Engineer, Senior Tech. Specialist	10 years +
77	Australia	Head of Engineering	10 years +
78	Germany	DevOps Engineer, Consulting Manager	10 years +
79	Australia	Senior Software Engineer	7 years +
80	Australia	Senior Developer, DevOps Architect, Cloud Specialist	5 years +
81	Spain	DevOps Engineer, Solutions Architect	4 years +
82	Brazil	Senior Consultant, Software Engineer, Developer	11 years +

c. Select the survey method

The DRA framework was evaluated using a field survey (Link) (see Appendix D) and was offered online to industry experts who were contacted via LinkedIn (see Table 5.17).

The survey was opened in January 2019 and closed in June 2019. A total of 82 participants completed the survey online.

d. Develop the questionnaire

The survey was composed of nine questionnaire sets (see Appendix D):

- Q1 set: DRA contextual model questionnaire set (5 questions)
- Q2 set: DRA conceptual model questionnaire set (6 questions)
- Q3 set: DRA logical model questionnaire set (5 questions)
- Q4 set: DRA logical model features (9 questions)
- Q5 set: DRA physical model (5 questions)
- Q6 set: DRA operational model (8 questions)
- Q7 set: DRA usefulness feedbacks and ratings (2 questions)
- Q8 set: DRA-suggested improvement (1 question)
- Q9 set: DRA overall feedbacks and ratings (2 questions).

e. Collect and analyse the data

The survey questionnaire sets (see Appendix D) generated two types of data:

- Quantitative data: rating data or categorical data transformed into ordinal data (participants' ratings in sets Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q9)
- Qualitative data (participants' feedback in sets Q7, Q8, Q9)

The survey evaluation is composed of two main steps:

- survey data collection
- survey data analysis.

5.6.1. Survey Data Collection

The survey collection process presents the procedure used in the data collection. The collected data from the survey can be categorised into two types: quantitative and qualitative.

The quantitative data sources are the ratings collected from the survey questionnaire sets Q1, Q2, Q3, Q4, Q5, Q6, Q7 and Q9 (see Appendix D).

The quantitative data sources are the feedback collected from the survey questionnaire sets Q7, Q8 and Q9 (see Appendix D).

The collected data were organised into groups according to the questionnaire related to the survey evaluation criteria (see Table 3.4). The questionnaire was organised as follows.

- **DRA Models Questionnaire Groups (Sets Q1–Q6)**

The DRA survey questionnaires (sets Q1–Q6) are organised into Tables 5.18 to 5.23. The questions in these sets offer the survey participants the option to evaluate DRA models against the criteria in Table 3.4 (coverage, relevance and importance). The questionnaires are grouped as follows:

Table 5.18: DRA Contextual Model Questions Group

Question	Description	Category
Q1	Does the contextual model provide the overall scope and purpose of using a DevOps approach for IoT app and multi-cloud at a high level?	Coverage
Q2	Do you think DevOps is appropriate for deploying IoT apps to the multi-cloud environment?	Coverage
Q3	Are the model elements (technologies) sufficient for the context?	Coverage
Q4	Is the contextual model relevant to the DRA framework?	Relevance
Q5	Are the contextual model elements important to the DRA framework?	Importance

Table 5.19: DRA Conceptual Model Questions Group

Question	Description	Category
Q1	Does the conceptual model provide enough components for DevOps?	Coverage
Q2	Does the conceptual model provide enough components for the cloud?	Coverage
Q3	Does the conceptual model provide enough components for the multi-cloud deployment platform of IoT apps?	Coverage
Q4	Is the conceptual model relevant for DRA framework?	Relevance
Q5	Is the conceptual model important for DRA framework?	Importance
Q6	Is CI-Broker an important component for deploying IoT apps on multi-cloud?	Importance

Table 5.20: DRA Logical Model Design Questions Group

Question	Description	Category
Q1	Does the logical model provide enough components for DevOps?	Coverage
Q2	Does the logical model provide enough components for IoT apps deployment?	Coverage
Q3	Does the logical model provide enough components for the cloud platform?	Coverage
Q4	Is the logical model relevant to the DRA framework?	Relevance
Q5	Is the logical model important to the DRA framework?	Importance

Table 5.21: DRA Logical Model Functions Questions Group

Question	Description	Category
Q1	DRA M1 automate code synchronisation for DevOps team.	Coverage
Q2	DRA M2 enable automation for: repository update, build, testing.	Coverage
Q3	DRA M2 enables deployment to M3 (using CI broker).	Coverage

Question	Description	Category
Q4	DRA M3 automate scaling and application staging for users.	Coverage
Q5	DRA M4 enable automated log capture from build, testing and deployment of IoT app.	Coverage
Q6	DRA M5 provides cloud database management for DevOps team.	Coverage
Q7	Do you think that the M1–M5 sub-models provide enough functions for the DRA framework?	Coverage
Q8	Do you think that the M1–M5 sub-models are relevant for the DRA framework?	Relevance
Q9	Do you think that the M1–M5 sub-models are important for the DRA framework?	Importance

Table 5.22: DRA Physical Model Questions Group

Question	Description	Category
Q1	Does the physical model provide enough features for DevOps?	Coverage
Q2	Does the physical model provide enough features for the cloud?	Coverage
Q3	Does the physical model provide enough features for IoT apps deployment?	Coverage
Q4	Is the physical model relevant to the DRA framework?	Relevance
Q5	Is the physical model important for the DRA framework?	Importance

Table 5.23: DRA Operational Model Questions Group

Question	Description	Category
Q1	Does the pipeline provide enough components to support DevOps?	Coverage
Q2	Does the pipeline provide enough components to support multi-cloud deployment?	Coverage
Q3	Does the pipeline provide enough components to enable IoT app deployment on multi-cloud?	Coverage
Q4	Does DRA pipeline enable automated IoT app deployment on multi-cloud using Codeship as CI broker?	Importance
Q5	Is DRA pipeline tools integration relevant for the framework?	Relevance
Q6	Are the DevOps tools in the pipeline sufficient for the framework?	Importance
Q7	Does the DRA pipeline reflect the conceptual design model?	Relevance
Q8	Does the DRA pipeline provide all the functions and features defined in the Logical model?	Importance

- **DRA Overall Questionnaire Groups (Sets Q7–Q9)**

The DRA survey questionnaires (sets Q7–Q9) are organised into one group. The questions in these sets offer the survey participants the option to evaluate DRA models against the criteria in Table 3.4. The qualitative data collected from these questionnaire sets were analysed to determine the relationship between the participants' quotes and comments and DRA categories represented by the evaluation criteria in Table 3.4. This process aimed to make sense of the qualitative data and correlate the participants' feedback with the framework. The quantitative data collected from sets Q7–Q9 provide overall ratings that aim to determine the usefulness and applicability of the DRA based on the participants' ratings.

5.6.2. SURVEY DATA ANALYSIS

The survey evaluation process is composed of two main phases:

- Survey quantitative evaluation: Participants' ratings were transformed from categorical data to ordinal data (numerical) using the survey ratings in Table 3.3. The ordinal data were used in statistical formulas to evaluate the survey results (see Equation 3.1–3.3).
- Survey qualitative evaluation: Participants' feedback was analysed using the hypothesis confirmation general technique of analysis (Runeson & Höst 2009). The hypotheses are the artefact evaluation criteria (Prat, Comyn-Wattiau & Akoka 2014) (see Table 3.4). Participants' feedback was cross-examined against the evaluation criteria by highlighting the occurrences of these criteria in the text. The industry feedback is organised in tables.

5.6.2.1. Survey Quantitative Evaluation

The quantitative evaluation process is composed of two sections:

1. Individual DRA design model evaluation based on the data collected from sets Q1–Q6.
2. Combined DRA design model evaluation based on the total responses from participants collected in sets Q1–Q6.

5.6.2.1.1. Individual DRA Models Evaluation

The individual DRA model evaluation has six steps (based on sets Q1–Q6). The survey data are located on CloudStor (Link). The individual evaluation process is as follows:

- Collect and map the survey rating into tables labelled **RT[Index]**.
- Group the ordinal data from the rating tables into category rating tables labelled **CT[Index]** based on the questionnaire sets (see Tables 5.18–5.23).
- Plot the RT [index] tables into a bar graph representation of the data labelled **RF[Index]**.
- Calculate the AAP and AAF statistical values for all RT[Index] tables and calculate goodness-of-fit Chi^2 for all CT[Index] (see Equation 3.1-3.3). The aim is to determine whether the DRA models meets the evaluation criteria positively (see Table 3.4):
  - AAF determines the frequency of participants agreeing or strongly agreeing that the DRA models meets the evaluation criteria positively.
  - AAP determines the percentage of participants agreeing or strongly agreeing that the DRA models meets the evaluation criteria positively.
  - Goodness-of-fit Chi^2, and p-value for each of the CT[Index] tables.

  H_0 (null hypothesis)*: There is no association between the DRA models and the evaluation criteria.*

  H_1 (alternative hypothesis)*: The DRA models meets the evaluation criteria positively.*

  If p-value < α, then H_0 is rejected and H_1 is accepted, and the DRA models meets the evaluation criteria positively (Coverage, Importance, Relevance).

  [If p-value < 0.000ε (ε is a small number), then p is corrected to: p < 0.001].

- **Contextual Model**

Table 5.24: Contextual Questionnaire Data (RT1)

	Q1	Q2	Q3	Q4	Q5	Row Total	Percentage
Strongly Disagree	3	1	5	3	3	15	3.66%
Disagree	9	1	9	3	3	25	6.10%
Average	14	3	17	15	15	64	15.61%
Agree	40	23	32	44	42	181	44.15%
Strongly Agree	16	54	19	17	19	125	30.49%
Column Total	82	82	82	82	82	410	100.00%

AAF = 370

AAP = 90.25%

Table 5.25: Contextual Group Data (CT1)

Category →	**Coverage (Q1, Q2, Q3)**		**Relevance (Q4)**		**Importance (Q5)**	
N = 5; E=∑O/N	**O**	**E**	**O**	**E**	**O**	**E**
Strongly Disagree	9	49.2	3	16.4	3	16.4
Disagree	19	49.2	3	16.4	3	16.4
Average	34	49.2	15	16.4	15	16.4
Agree	95	49.2	44	16.4	42	16.4
Strongly Agree	89	49.2	17	16.4	19	16.4
H_0 is rejected for p < 0.01	**Chi^2 = 130.911**	**p < 0.00001**	**Chi^2 = 68.488**	**p < 0.00001**	**Chi^2 = 62.39**	**p < 0.00001**

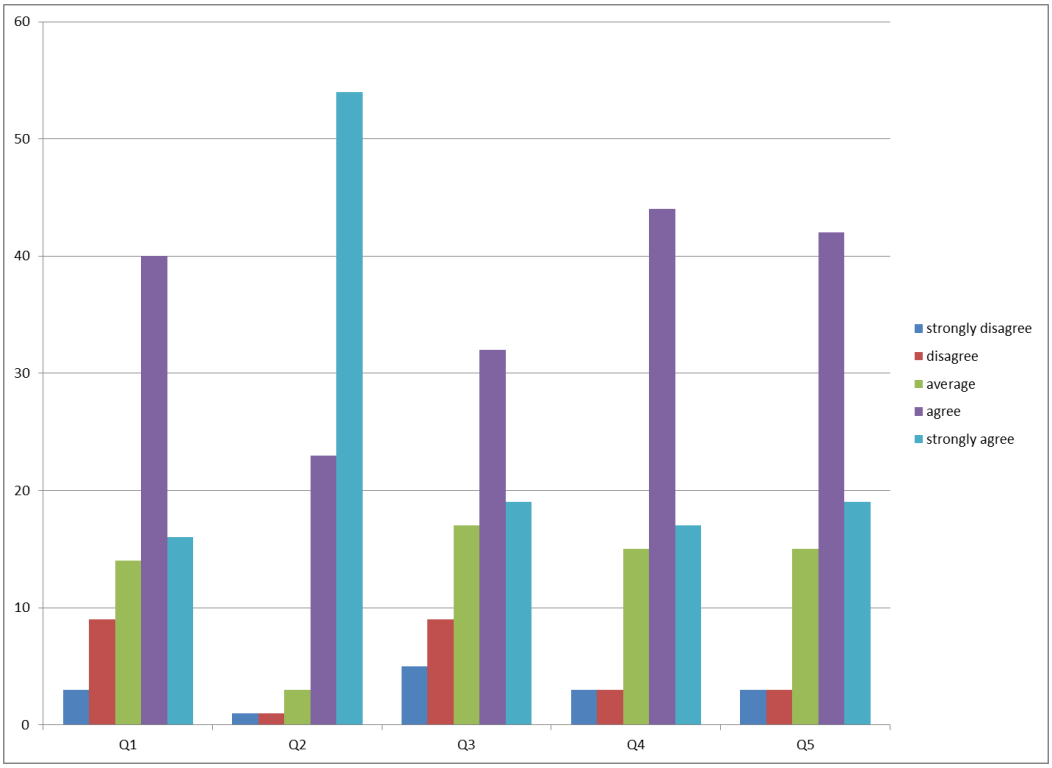


Figure 5.5: Contextual Data Graph (RF1)

REVIEW

The ordinal data in Table 5.24 (RT1) and Table 5.25 (CT1) produced fundamental statistical values based on participants' responses. The contextual model evaluation results can be interpreted as follows:

- AAF = 370 out of 410 response indicates that most of the participants agree the DRA contextual model meets the coverage evaluation criteria positively.
- AAP = 90.25% indicates that a high percentage of participants agree that the DRA contextual model meets the coverage evaluation criteria positively.
- The p-value for the test variables:
  - Coverage p-value is set at $0.001 < \alpha=0.01$. It means that H_0 is rejected, H_1 is accepted, and the DRA contextual model meets the coverage evaluation criteria positively.
  - Relevance p-value is set at $0.001 < \alpha=0.01$. It means that H_0 is rejected, H_1 is accepted, and the DRA contextual model meets the relevance evaluation criteria positively.
  - Importance p-value is set at $0.001 < \alpha=0.01$. It means that H_0 is rejected, H_1 is accepted, and the DRA contextual model meets the importance evaluation criteria positively.

The statistical values indicate that the participants consider the DRA contextual model relevant and an important design and that it covers the industry needs. Figure 5.5 illustrates the frequency of participants' responses to add further visual insight to the results.

- **Conceptual Model**

Table 5.26: Conceptual Questionnaire Data (RT2)

	Q1	Q2	Q3	Q4	Q5	Q6	Row Total	Percentage
Strongly Disagree	1	1	2	1	1	2	7	1.71%
Disagree	4	5	9	2	2	3	21	5.12%
Average	15	14	17	13	12	11	67	16.34%
Agree	39	44	40	46	42	31	203	49.51%
Strongly Agree	23	18	14	20	25	35	112	27.32%
Column Total	82	82	82	82	82	82	410	100.00%

AAF = 382

AAP = 93.17%

Table 5.27: Conceptual Group Data (CT2)

Category →	**Coverage (Q1, Q2, Q3)**		**Relevance (Q4)**		**Importance (Q5, Q6)**	
N = 5; E=∑O/N	**O**	**E**	**O**	**E**	**O**	**E**
Strongly Disagree	4	49.2	1	16.4	3	16.4
Disagree	18	49.2	2	16.4	5	16.4
Average	46	49.2	13	16.4	23	16.4
Agree	123	49.2	46	16.4	73	16.4
Strongly Agree	55	49.2	20	16.4	60	16.4
H_0 is rejected for $p < 0.01$	**Chi^2 = 172.902**	**p < 0.00001**	**Chi^2 = 82.024**	**p < 0.00001**	**Chi^2 = 125.39**	**p < 0.00001**

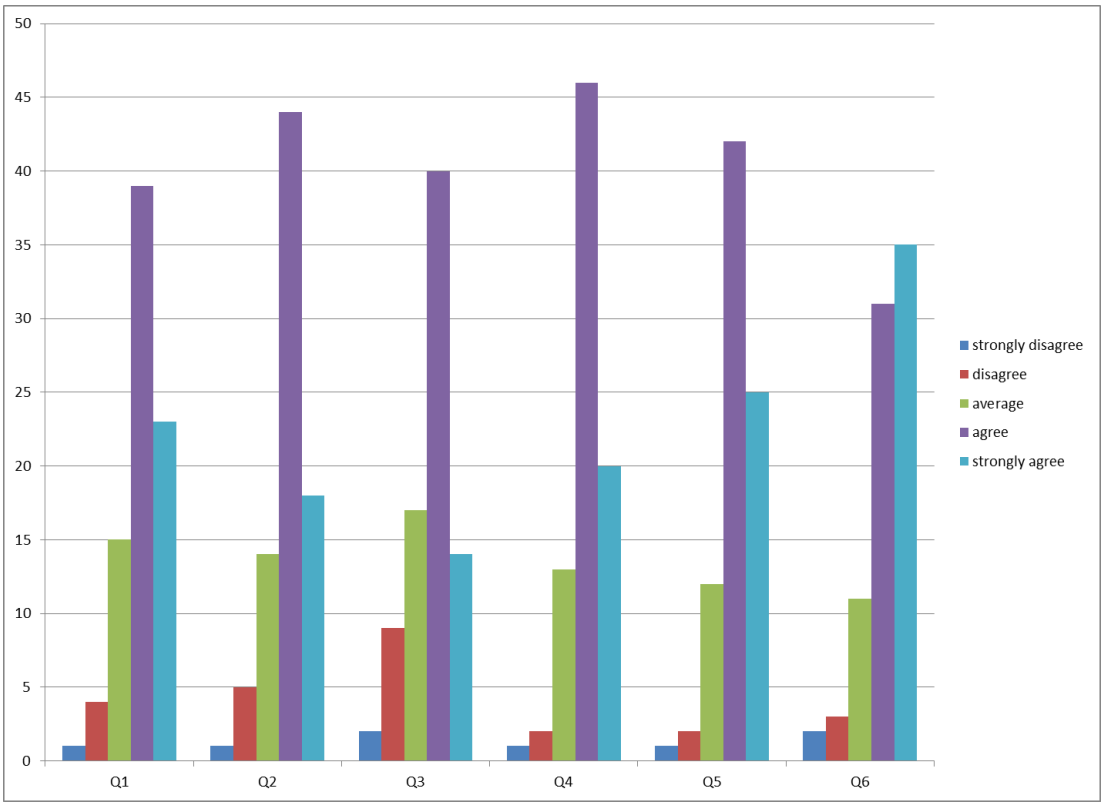


Figure 5.6: Conceptual Data Graph (RF2)

REVIEW:

The ordinal data in Table 5.26 (RT2) and Table 5.27 (CT2) produced key statistical values based on participants' responses. The conceptual model evaluation results can be interpreted as follows:

- AAF = 382 out of 410 response indicates that most of the participants agree the DRA conceptual model meets the coverage evaluation criteria positively.
- AAP = 93.17% indicates that a high percentage of participants agree that the DRA conceptual model meets the coverage evaluation criteria positively.
- The p-value for the test variables:
  - Coverage p-value is set at $0.001 < \alpha=0.01$. It means that H_0 is rejected, H_1 is accepted, and the DRA conceptual model meets the coverage evaluation criteria positively.
  - Relevance p-value is set at $0.001 < \alpha=0.01$. It means that H_0 is rejected, H_1 is accepted, and the DRA conceptual model meets the relevance criteria positively.
  - Importance p-value is set at $0.001 < \alpha=0.01$. It means that H_0 is rejected, H_1 is accepted, and the DRA conceptual model meets the importance criteria positively.

The statistical values indicate that the participants consider the DRA conceptual model relevant and an important design and that it covers the industry needs. Figure 5.6 illustrates the frequency of participants' responses to add further visual insight to the results.

- **Logical Model Design**

Table 5.28: Logical Design Questionnaire Data (RT3)

	Q1	Q2	Q3	Q4	Q5	Row Total	Percentage
Strongly Disagree	2	2	3	1	1	9	2.20%
Disagree	3	4	5	1	2	15	3.66%
Average	8	14	17	5	7	51	12.44%
Agree	48	44	40	52	45	229	55.85%
Strongly Agree	21	18	17	23	27	106	25.85%
Column Total	82	82	82	82	82	410	100.00%

AAF = 386

AAP = 94.14%

Table 5.29: Logical Design Group Data (CT3)

Category →	**Coverage (Q1, Q2, Q3)**		**Relevance (Q4)**		**Importance (Q5)**	
N = 5; E=∑O/N	**O**	**E**	**O**	**E**	**O**	**E**
Strongly Disagree	7	49.2	1	16.4	1	16.4
Disagree	12	49.2	1	16.4	2	16.4
Average	39	49.2	5	16.4	7	16.4
Agree	132	49.2	52	16.4	45	16.4
Strongly Agree	56	49.2	23	16.4	27	16.4
H_0 is rejected for p < 0.01	**Chi^2 = 206.724**	**p < 0.00001**	**Chi^2 = 116.78**	**p < 0.00001**	**Chi^2 = 89.22**	**p < 0.00001**

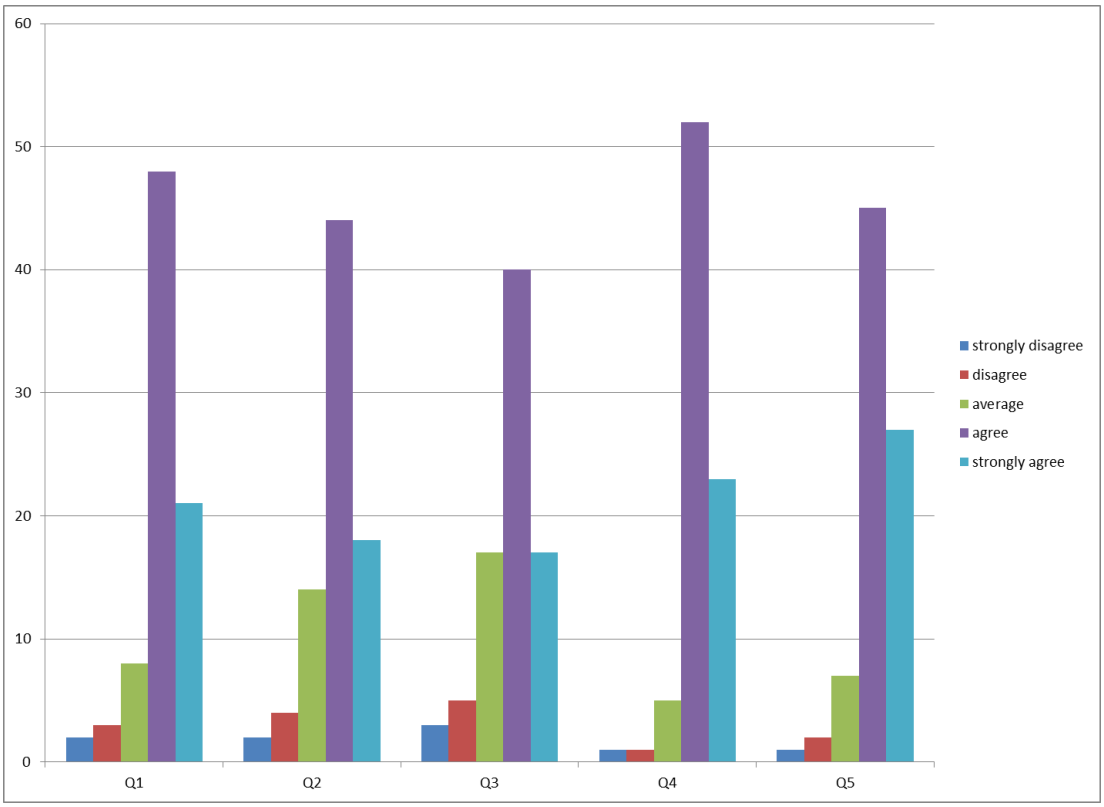


Figure 5.7: Logical Design Data Graph (RF3)

REVIEW

The ordinal data in Table 5.28 (RT3) and Table 5.29 (CT3) produced key statistical values based on participants' responses. The logical model evaluation results can be interpreted as follows:

- AAF = 386 out of 410 response indicates that most of the participants agree the DRA logical design model meets the coverage evaluation criteria positively.
- AAP = 94.14% indicates that a high percentage of participants agree that the DRA logical design model meets the coverage evaluation criteria positively.
- The p-value for the test variables:
  - Coverage p-value is set at $0.001 < \alpha=0.01$. It means that H_0 is rejected, H_1 is accepted, and the DRA logical design model meets the coverage evaluation criteria positively.
  - Relevance p-value is set at $0.001 < \alpha=0.01$. It means that H_0 is rejected, H_1 is accepted, and the DRA logical design model meets the relevance criteria positively.
  - Importance p-value is set at $0.001 < \alpha=0.01$. It means that H_0 is rejected, H_1 is accepted, and the DRA logical design model meets the importance criteria positively.

The statistical values indicate that the participants consider the DRA logical design model relevant and an important design and that it covers the industry needs. Figure 5.7 illustrates the frequency of participants' responses to add further visual insight to the results.

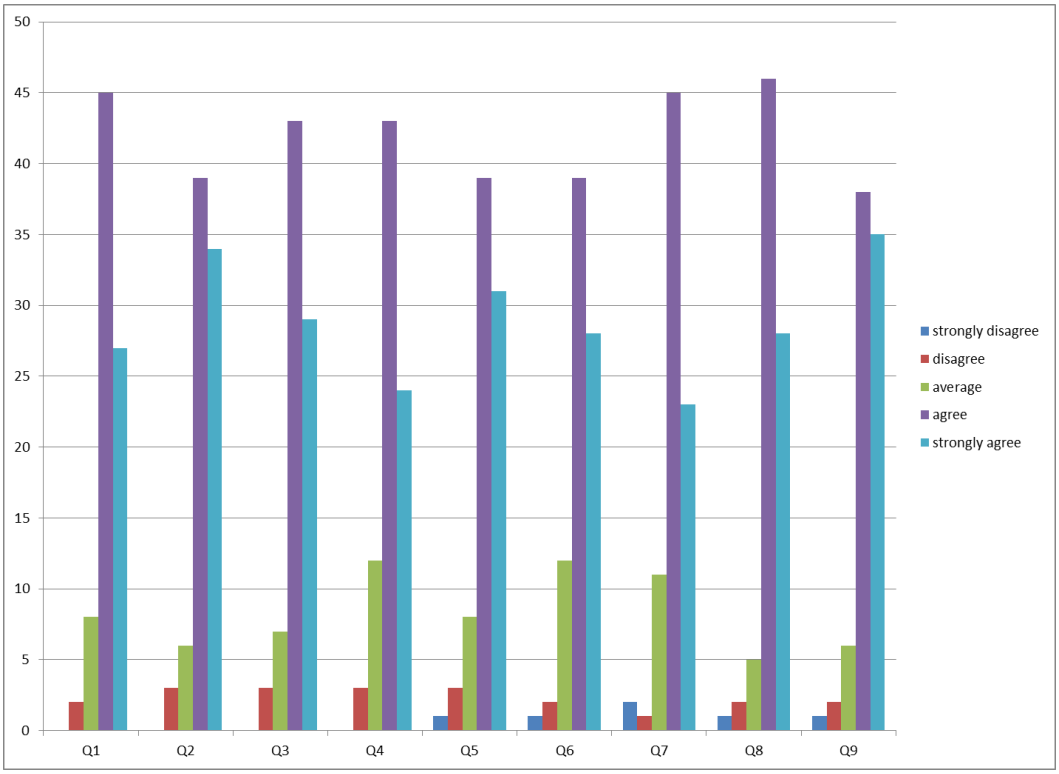
- **Logical Model Features**

Table 5.30: Logical Features Questionnaires Data (RT4)

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Row Total	Percentage
Strongly Disagree	0	0	0	0	1	1	2	1	1	6	0.81%
Disagree	2	3	3	3	3	2	1	2	2	21	2.85%
Average	8	6	7	12	8	12	11	5	6	75	10.16%
Agree	45	39	43	43	39	39	45	46	38	377	51.08%
Strongly Agree	27	34	29	24	31	28	23	28	35	259	35.09%
Column Total	82	82	82	82	82	82	82	82	82	738	100.00%

Table 5.31: Logical Features Group Data (CT4)

Category →	**Coverage (Q6, Q7)**		**Relevance (Q8)**		**Importance (Q1-Q5, Q9)**	
N = 5; E=∑O/N	**O**	**E**	**O**	**E**	**O**	**E**
Strongly Disagree	3	32.8	1	16.4	3	114.8
Disagree	3	32.8	2	16.4	18	114.8
Average	23	32.8	5	16.4	59	114.8
Agree	84	32.8	46	16.4	286	114.8
Strongly Agree	51	32.8	28	16.4	208	114.8
H_0 is rejected for p < 0.01	**Chi^2 = 147.098**	**p < 0.00001**	**Chi^2 = 96.659**	**p < 0.00001**	**Chi^2 = 547.596**	**p < 0.00001**



AAP = 711

AAF = 96.33 %

Figure 5.8: Logical Features Data Graph (RF4)

REVIEW

The ordinal data in Table 5.30 (RT4) and Table 5.31 (CT4) produced key statistical values based on participants' responses. The logical features evaluation results can be interpreted as follows:

- AAF = 711 out of 738 response indicates that most of the participants agree the DRA logical model features meet the coverage evaluation criteria positively.
- AAP = 96.33% indicates that a high percentage of participants agree that the DRA logical model features meet the coverage evaluation criteria positively.
- The p-value for the test variables:
  - Coverage p-value is set at $0.001 < \alpha=0.01$. It means that H_0 is rejected, H_1 is accepted, and the DRA logical model features meet the coverage evaluation criteria positively.
  - Relevance p-value is set at $0.001 < \alpha=0.01$. It means that H_0 is rejected, H_1 is accepted, and the DRA logical model features meet the relevance criteria positively.
  - Importance p-value is set at $0.001 < \alpha=0.01$. It means that H_0 is rejected, H_1 is accepted, and the DRA logical model features meet the importance criteria positively.

The statistical values indicate that the participants consider the DRA logical model feautures are relevant and an important design and that it covers the industry needs. Figure 5.8 illustrates the frequency of participants' responses to add further visual insight to the results.

- **Physical Model**

Table 5.32: Physical Model Questionnaires Data (RT5)

	Q1	Q2	Q3	Q4	Q5	Row Total	Percentage
Strongly Disagree	3	2	3	1	3	12	2.93%
Disagree	2	4	7	4	1	18	4.39%
Average	9	9	8	10	11	47	11.46%
Agree	42	41	39	42	38	202	49.27%
Strongly Agree	26	26	25	25	29	131	31.95%
Column Total	82	82	82	82	82	410	100.00%

AAF = 380

AAP = 92.68%

Table 5.33: Physical Model Group Data (CT5)

Category →	**Coverage (Q1, Q2, Q3)**		**Relevance (Q4)**		**Importance (Q5)**	
N = 5; E=∑O/N	**O**	**E**	**O**	**E**	**O**	**E**
Strongly Disagree	8	49.2	1	16.4	3	16.4
Disagree	13	49.2	4	16.4	1	16.4
Average	26	49.2	10	16.4	11	16.4
Agree	122	49.2	42	16.4	38	16.4
Strongly Agree	77	49.2	25	16.4	29	16.4
H_0 is rejected for $p < 0.01$	**Chi^2 = 195.504**	**p < 0.00001**	**Chi^2 = 70.805**	**p < 0.00001**	**Chi^2 = 65.317**	**p < 0.00001**

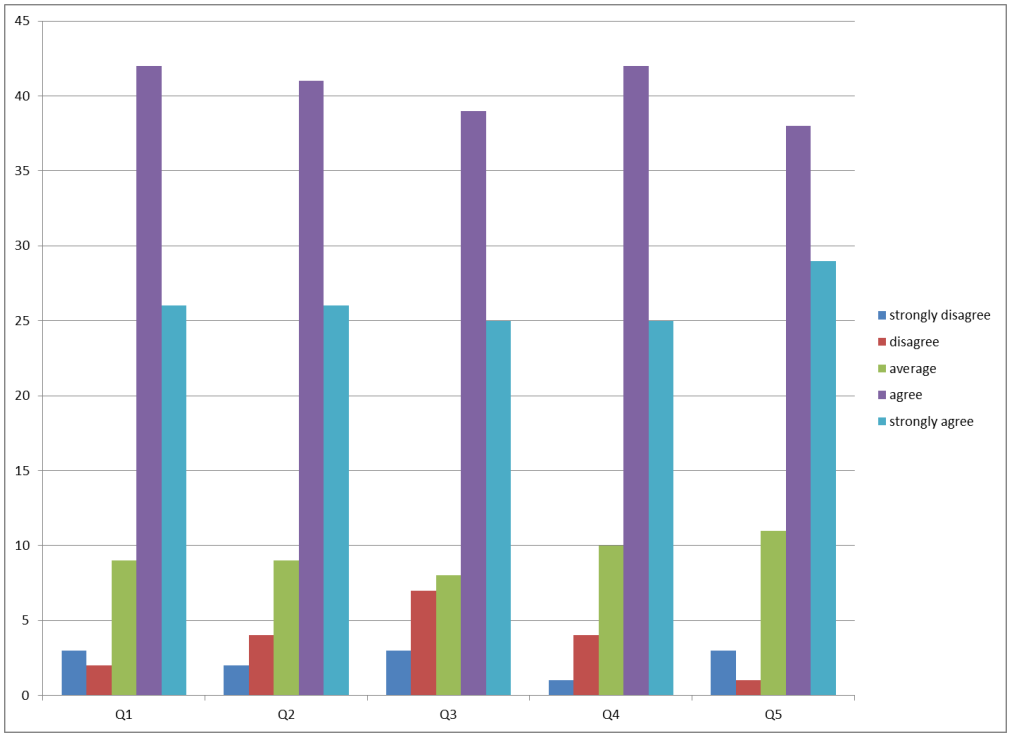


Figure 5.9: Physical Model Data Graph (RF5)

REVIEW

The ordinal data in Table 5.32 (RT5) and Table 5.33 (CT5) produced key statistical values based on participants' responses. The logical features evaluation results can be interpreted as follows:

- AAF = 380 out of 410 response indicates that most of the participants agree the DRA physical model meets the coverage evaluation criteria positively.
- AAP = 92.68% indicates that a high percentage of participants agree that the DRA physical model meets the coverage evaluation criteria positively.
- The p-value for the test variables:
  - Coverage p-value is set at $0.001 < \alpha=0.01$. It means that H_0 is rejected, H_1 is accepted, and the DRA physical model meets the coverage evaluation criteria positively.
  - Relevance p-value is set at $0.001 < \alpha=0.01$. It means that H_0 is rejected, H_1 is accepted, and the DRA physical model meets the relevance evaluation criteria positively.
  - Importance p-value is set at $0.001 < \alpha=0.01$. It means that H_0 is rejected, H_1 is accepted, and the DRA physical model meets the importance evaluation criteria positively.

The statistical values indicate that the participants consider the DRA physical model relevant and an important design and that it covers the industry needs. Figure 5.9 illustrates the frequency of participants' responses to add further visual insight to the results.

- **Operational Model**

Table 5.34: Operational Model Questionnaires Data (RT6)

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Row Total	Percentage
Strongly Disagree	2	3	3	2	2	1	2	2	17	2.59%
Disagree	0	1	2	1	2	3	2	3	14	2.13%
Average	11	11	12	9	10	13	10	10	86	13.11%
Agree	37	38	39	42	38	41	34	38	307	46.80%
Strongly Agree	32	29	26	28	30	24	34	29	232	35.37%
Column Total	82	82	82	82	82	82	82	82	656	100.00%

AAF= 625

AAP= 95.28%

Table 5.35: Operational Model Group Data (CT6)

Category →	Coverage (Q1, Q2, Q3, Q6)		Relevance (Q5, Q7)		Importance (Q4, Q8)	
N = 5; E=∑O/N	O	E	O	E	O	E
Strongly Disagree	9	65.6	4	32.8	4	32.8
Disagree	6	65.6	4	32.8	4	32.8
Average	47	65.6	20	32.8	19	32.8
Agree	155	65.6	72	32.8	80	32.8
Strongly Agree	111	65.6	64	32.8	57	32.8
H_0 is rejected for $p < 0.01$	**Chi^2 = 261.512**	**p < 0.00001**	**Chi^2 = 132.098**	**p < 0.00001**	**Chi^2 = 142.159**	**p < 0.00001**

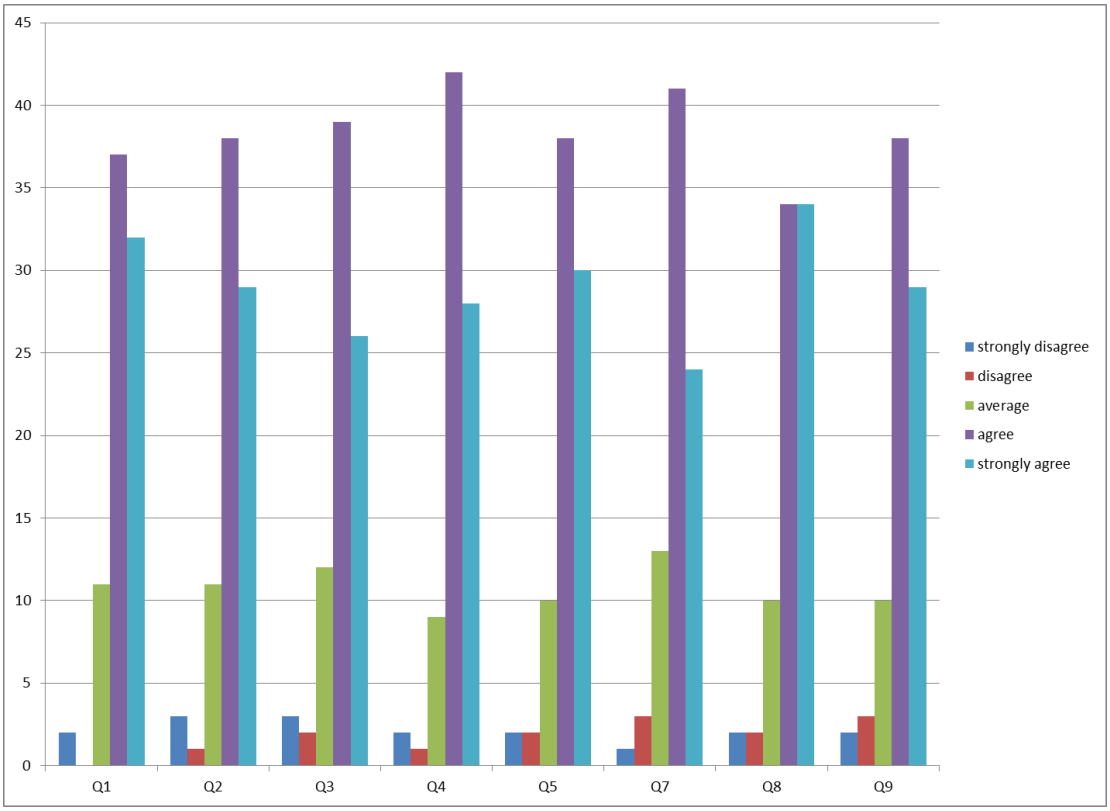


Figure 5.10: Operational Model Data Graph (RF6)

REVIEW

The ordinal data in Table 5.34 (RT6) and Table 5.35 (CT6) produced key statistical values based on participants' responses. The operational model evaluation results can be interpreted as follows:

- AAF = 625 out of 656 response indicates that most of the participants agree the DRA operational model meets the coverage evaluation criteria positively.
- AAP = 92.28% indicates that a high percentage of participants agree that the DRA operational model meets the coverage evaluation criteria positively.
- The p-value for the test variables:
  - Coverage p-value is set at $0.001 < \alpha=0.01$. It means that H_0 is rejected, H_1 is accepted, and the DRA operational model meets the coverage evaluation criteria positively.
  - Relevance p-value is set at $0.001 < \alpha=0.01$. It means that H_0 is rejected, H_1 is accepted, and the DRA operational model meets the relevance criteria positively.
  - Importance p-value is set at $0.001 < \alpha=0.01$. It means that H_0 is rejected, H_1 is accepted, and the DRA operational model meets the importance criteria positively.

The statistical values indicate that the participants consider the DRA operational model relevant and an important design and that it covers the industry needs. Figure 5.10 illustrates the frequency of participants' responses to add further visual insight to the results.

5.6.2.1.2 Combined DRA Models Evaluation

The combined DRA models evaluation uses the total categorical data from the six individual DRA models questionnaires (sets Q1–Q6). The combined data collected from these questionnaires are evaluated as follows:

- Add all survey rating from every category questionnaire into Table 5.37, labelled **RT [7]**.
- Plot RT [7] data into a bar graph Figure 5.1,1 labelled RF [7].
- Calculate the statistical values AAP and AAF for each RT [7] table:
  - AAF determines the frequency of participants that may agree or strongly agree that the DRA models meet the evaluation criteria positively. (Table 3.4).
  - AAP determines the percentage of participants that are likely to agree or strongly agree that the DRA model meets the evaluation criteria positively (see Table 3.4).
- Calculate the 5x5 Chi^2 (see Equation 3.1) for RT [7] data and map the results into Table 5.37 labelled CT [7].

The Chi^2 test is a 5x5 table test that aims to evaluate the DRA design models against the evaluation criteria (see Table 3.4) using the total responses of participants in the survey for the six category questionnaires.

- **H_0 (null hypothesis)**: There is no association between the DRA models and the evaluation criteria in Table 3.4.
- **H_1 (alternative hypothesis)**: The DRA models meet the evaluation criteria positively:
  - DRA contextual model
  - DRA conceptual model
  - DRA logical model + logical features
  - DRA physical model
  - DRA operational model

The overall p-value is calculated to determine whether to accept or reject the null hypothesis at a critical value $\alpha = 0.01$. The test calculates the Chi^2 and the p-value for each cell in Table 5.37.
If p-value < α, then H_0 is rejected, H_1 is accepted, hence then participants agree or strongly agree that the DRA models meet the evaluation criteria positively. (**coverage, relevance, importance**).
[If p-value < 0.000ε (ε is a small number), then p is mathematically corrected to p: < 0.001].

Table 5.36: DRA Total Data Results (RT7)

	Contextual	Conceptual	Logical	Physical	Operational	Rows	Percentage
Strongly Disagree	15	7	18	12	17	69	2.27%
Disagree	25	21	39	18	14	117	3.86%
Average	64	67	122	47	86	386	12.72%
Agree	181	203	579	202	307	1472	48.52%
Strongly Agree	125	112	390	131	232	990	32.63%
Column Total	**410**	**410**	**1148**	**410**	**656**	**3034**	**100.00%**

Table 5.37: DRA Total Data Chi^2-Test (CT7)

The chi^2 test is 41.4224. The p-value is 0.000481. p < 0.01 and H_0 is rejected						
	Contextual	**Conceptual**	**Logical**	**Physical**	**Operational**	**Row Total**
Strongly Disagree	15 (9.32) [3.45]	7 (9.32) [0.58]	18 (26.11) [2.52]	12 (9.32) [0.77]	17 (14.92) [0.29]	69
Disagree	25 (15.81) [5.34]	21 (15.81) [1.70]	39 (44.27) [0.63]	18 (15.81) [0.30]	14 (25.30) [5.05]	117
Average	64 (52.16) [2.69]	67 (52.16) [4.22]	122 (146.05) [3.96]	47 (52.16) [0.51]	86 (83.46) [0.08]	386
Agree	181 (198.92) [1.61]	203 (198.92) [0.08]	579 (556.97) [0.87]	202 (198.92) [0.05]	307 (318.27) [0.40]	1472
Strongly Agree	125 (133.78) [0.58]	112 (133.78) [3.55]	390 (374.59) [0.63]	131 (133.78) [0.06]	232 (214.05) [1.50]	990
Column Total	410	410	1148	410	656	**3034**

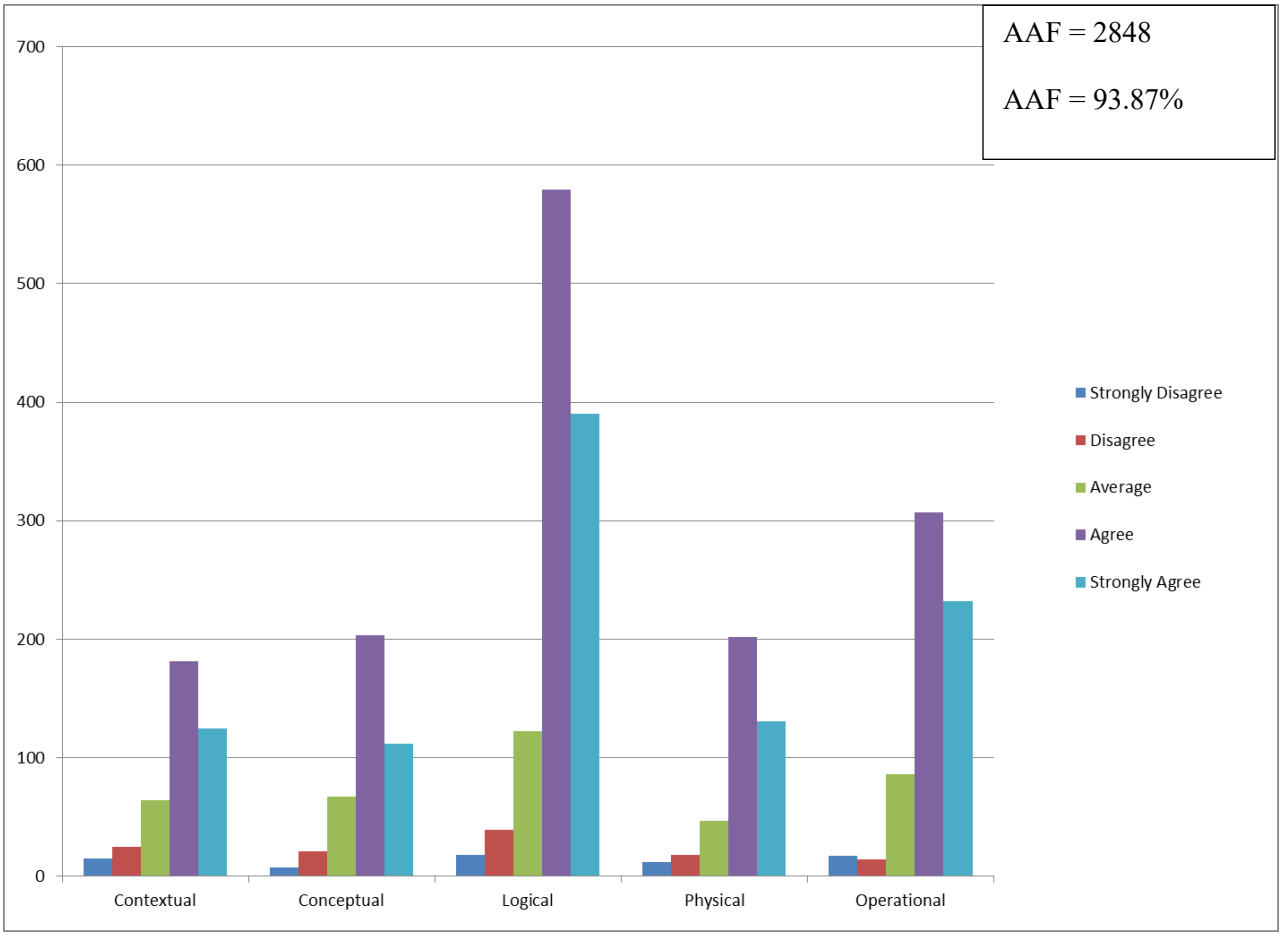


Figure 5.11: DRA Models Combined Data (RF7)

REVIEW

The ordinal data in tables Table 5.36 (RT7) and Table 5.37 (CT7) produced key statistical values based on participants' responses. The DRA design models evaluation results can be interpreted as follows:

- AAF = 2848 out of 3034 responses indicates that most of the participants agree that that the DRA models meet the evaluation criteria positively.
- AAP = 93.87% indicates that a high percentage of participants agree that the DRA models meet the evaluation criteria positively.
- The Chi^2 test is 41.4224. The p-value is .000481. The result is significant at $p < 0.01$. H_0 is rejected, and H_1 is accepted.

The statistical values indicate that the participants consider the DRA design models a relevant and important design and may cover industry needs. Figure 5.11 illustrates provide further visual insights that the DRA models positively meet the evaluation criteria (see Table 3.4).

5.6.2.2. Survey Qualitative Evaluation

This section presents the qualitative evaluation of the participants' feedback and comments about the DRA. The survey qualitative evaluation is based on the participants' feedback provided about DRA design models in the questionnaire sets (Q7–Q9). The survey feedback is located on CloudStor (Link).

The qualitative evaluation process is composed of two sections:

- DRA usefulness (for teaching, research and industry) evaluation [Q7 set]
- DRA overall feedback and ratings [Q9 set].

Note: DRA-suggested improvements [Q8 set] are used in Chapter 6 to determine future research based on participants' suggestions in this question.

5.6.2.2.1 DRA Usefulness Feedback and Rating

This section evaluates the participants' responses about DRA usefulness in industry, teaching and research. The evaluation process is as follows:

- Collect and map the feedback about DRA usefulness into Table 5.38, labelled FT1.
- Analyse FT1 feedback based on the occurrences of the criteria (see Table 3.4) in the text using the cross-examination method (see in chapter 3, DSR Evaluation step).
- Collect and map the usefulness rating as numerical data in Table 5.39, labelled RT8.
- Plot Table 5.39 (RT8) data into a bar graph in Figure 5.12, labelled RF8.
- Calculate the statistical values AAF and AAP from Table 5.39 (RT8) data (see Equation 3.1-3.2):

- AAP determines the frequency of participants that consider the DRA useful for industry, teaching and research (see Equation 3.2).
- AAF determines the percentage of participants that consider the DRA useful for industry, teaching and research (see Equation 3.3).
- Calculate the goodness-of-fit Chi^2 and p-value for each test variable (teaching, industry, research) at a critical value $\alpha = 0.01$ (see Equation 3.1).

If $p < \alpha$, then the null hypothesis H_0 is rejected and H_1 is accepted.

H_0: There is no association between the test variables.

H_1: The DRA models meet the evaluation criteria positively (usefulness for teaching, industry, research).

Table 5.38: DRA Usefulness (FT1)

Category	Participants' Feedbacks	Interpretation
Importance Generalisations	One particular aspect that I think is very **important in the DRA** framework is the **flexibility** to choose the **instantiations of each component** (or indeed have several instantiations). Given the **heterogeneous** nature of the cloud and IoT environments, I think this is a critical feature.	Flexibility Instantiations of each component Heterogeneous nature Cloud and IoT relationship important in the DRA
Usefulness Relevance Coverage	The **models provided are all very useful** as they **deconstruct and disambiguate what is required** in order to **deploy any code as a single or multi-cloud application**. In essence, this is achieved through presenting both abstract and concrete examples and clearly defining steps involved at each stage of the **process.**	Useful models Disambiguate Deploy any code Adaptive to technology Process What is required
Usefulness Importance	The **continuous integration** and **automated deployment to multi-cloud are** very **useful.**	Continuous integration Useful Automated deployment
Coverage Generalisations	All of it, it provides **a comprehensive overview** of **what's needed for** an **IoT application deployment and management**.	What is needed for IoT apps Comprehensive overview
Relevance Importance	**Decentralized** logging, **cloud-hosted CI**, **deployment to multiple hosting vendors**, etc. are all **acceptable modern solutions for the demonstrated problem**.	Decentralised logging Cloud CI Deployment to multiple vendors Acceptable modern solutions
Usefulness	DRA logical model specifications and DRA **pipeline instance** are **very useful** in the enterprise world, as they focus on the **low-level implementation scheme.**	DRA logical, pipeline instance is very useful Low-level implementation
Usefulness	**The video was good with the Raspberry Pi implementation**, although implementing a 'real' code change, e.g., changing the frequency of flickering LEDs and showing the before and after results of the change would have been more concrete rather than just changing a text file, which we cannot	Acceptable demo

Category	Participants' Feedbacks	Interpretation
	verify has been deployed to the target destination in the video.	
Relevance	The fact that the entire **deployment process** is **automated** and seamless is really nice.	Automated deployment process Seamless deployment process
Relevance	An interesting approach to **combine DevOps and IoT**.	DevOps and IoT combination IoT interaction
Coverage Usefulness Generalisations	I do like this model for the **deployment of applications**. I think it is quite **extensive** and **applicable** to developers (ops) and **IT management,** including **project management**.	Extensive model Applicable IT and project management
Usefulness	It provides a **framework** that I'm pretty sure would be **invaluable** for people that don't know all the components and would like to implement it.	Invaluable framework Easy to implement
Usefulness Relevance	An IDE integrated plugin which can **deploy code readily to an IoT device**, without being able to manage the entire infrastructure pipeline in between is a **great value addition for any start-up working on Io**T.	IoT code deployment A valuable addition to IoT
Coverage	Correctly identifies **the benefits of DevOps**.	DevOps benefit
Importance	Very **appropriate tools** and real **use case implementation.**	Appropriate tools Use case implementation
Relevance	It gives a clear idea of a **path to production** that could be used in **IoT development process**.	Path to production IoT development process
Usefulness	**Useful for IoT cloud solutions**.	Useful for cloud–IoT
Importance Usefulness	The **DevOps** section is very good as all the **tools are perfectly used**. As it will make good **automation in architecture.**	Correct use of tools Automated architecture Useful
Generalisations Relevance Usefulness	Having a **high-level view of specifications**, **logic, modelling**, and behaviour of **the whole system is paramount to a solid implementation of the pipeline between IoT and multi-cloud.**	High-level view, logic, models IoT and multi-cloud IoT interaction
Relevance Importance	It's good that you're creating a **platform on multi-cloud** level. The idea of **Automating end to end pipeline** is a good thing.	Multi-cloud platform End-to-end automation
Usefulness	I am really **impressed with DRA** I think the IoT environment are missing some DevOps structure like that provided by DRA, so it is very **useful** to guide teams to have a simple workflow to **implement DevOps in IoT projects.**	Significance of DRA Useful DevOps implementation IoT projects
Relevance Usefulness	The value proposition that is to enable **IoT architecture by the DevOps efforts** in order to start using this amazing technology for an application like smart cities or even the set a **cloud architecture** for a smart house system.	DevOps for IoT Cloud architecture IoT interaction May be used for the smart house system
Importance Usefulness	Definitely **automation**. When IoT makes to the top, it will be so hard to test things manually, and those things are going to **be interacting directly and**	Automation Interaction It's going to be a very good

Category	Participants' Feedbacks	Interpretation
	physically with users... So the **automation process** it's **going to be a very good friend for the UX of anything related IoT.**	friend for the UX of anything related IoT.
Coverage	My feedback would be the same for DRA as well as for the **framework**, **this info and the project itself is excellent**, but has a lot of info, a lot of information very wide. Would also be good to watch more real-life scenarios.	Framework Information is excellent
Usefulness Relevance Importance	This can be a **useful reference** document for **IoT based Apps deployment** with **DevOps culture in the team process and toolset**.	Useful IoT apps deployment DevOps culture
Generalisations	I thought it was a good **overview** of how **DevOps can be applied with IoT.**	DevOps applied with IoT Overview
Importance	It is a good **high-level design** for the **IoT cloud app deployment** workflow.	High-level design IoT cloud app deployment IoT interaction
Relevance Importance	**Agility, CI/CD, automation, Speed.**	Agility, speed CI/CD
Importance Relevance	Allowing users to **interact with IoT devices** remotely using **cloud services**. The ability to **track usage** through the use of **Paper-trail and automation of IoT device by using the cloud**.	IoT interaction Cloud services Tracking Cloud–IoT monitoring Useful
Relevance Importance Coverage	Perfect **pick up as industry is facing these issues** as to how to **integrate or implement DevOps transformation** when it comes to **Cloud and IoT related Apps**. In a nutshell its part of **Digital Transformation** which is one of future in Industry.	Industry needs Implement DevOps transformation Cloud and IoT relationship IoT interaction Digital transformation

Table 5.39: DRA Usefulness Ratings (RT8)

DRA Usefulness	Research		Teaching		Industry			
N = 5; E=∑O/N	**O**	**E**	**O**	**E**	**O**	**E**	**Rows Total**	**Percentage**
Strongly Disagree	1	16.4	2	16.4	2	16.4	5	2.03%
Disagree	1	16.4	4	16.4	7	16.4	12	4.88%
Average	12	16.4	10	16.4	21	16.4	43	17.48%
Agree	32	16.4	37	16.4	27	16.4	96	39.02%
Strongly Agree	36	16.4	29	16.4	25	16.4	90	36.59%
H_0 is rejected for p < 0.01	**Chi^2 = 68.366**	**p < 0.001**	**Chi^2 = 60.073**	**p < 0.001**	**Chi^2 = 30.683**	**p < 0.001**	**246**	**100.00%**

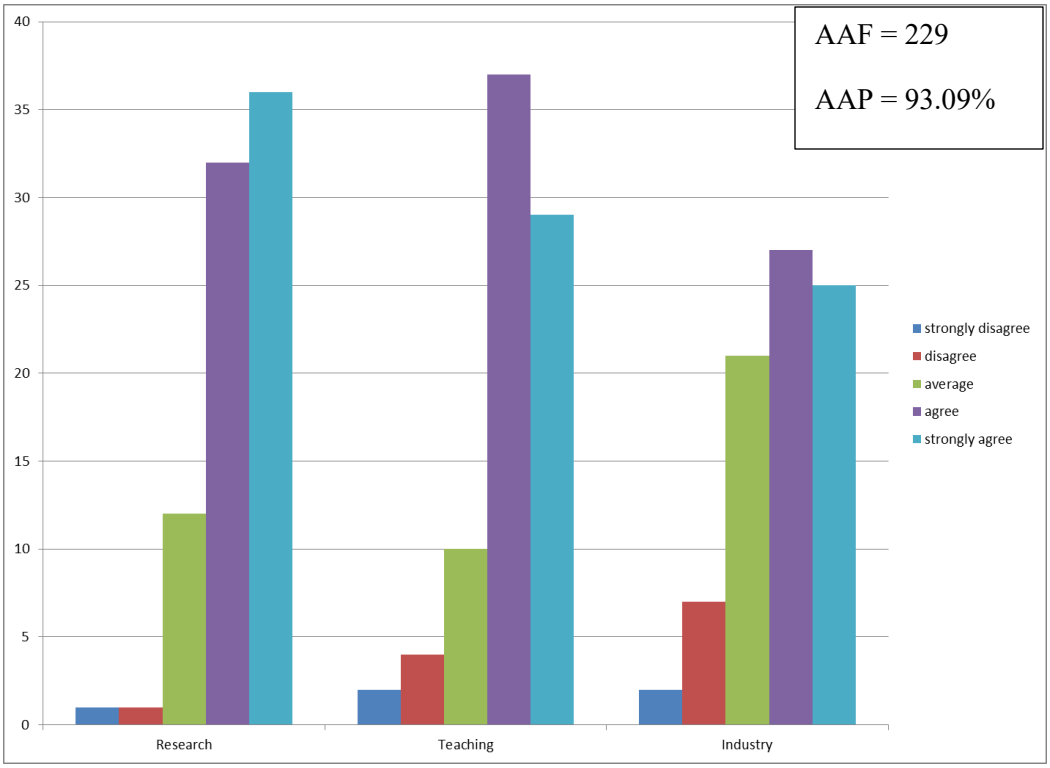


Figure 5.12: DRA Usefulness Ratings Graph (RF8)

REVIEW

The ordinal data in Table 5.38 (RT8) key statistical values based on participants' responses ordinal data. The DRA usefulness questionnaire results can be interpreted as follows:

- AAF = 229 out of 246 responses indicates that most of the participants agree or strongly agree that the DRA is useful for teaching, research and industry.
- AAP = 93.09% indicates that a high percentage of participants agree or strongly agree that the DRA usefulness for teaching, research and industry.
- The p-value for the test variables:
  - Research p-value is set at $0.001 < \alpha=0.01$. H_0 is rejected, H_1 is accepted, and the DRA models meet the evaluation criteria positively (usefulness for research).
  - Teaching p-value is set at $0.001 < \alpha=0.01$. H_0 is rejected, H_1 is accepted, and the DRA models meet the evaluation criteria positively (usefulness for teaching).
  - Industry p-value is set at $0.001 < \alpha=0.01$. H_0 is rejected, H_1 is accepted, and the DRA models meet the evaluation criteria positively (usefulness for the industry).
- In Table 5.38 (FT1), there are T = 50 referenccing of the evaluation criteria (see Table 3.4). The frequency distribution of FT1 categories is mapped in Table 5.40. The results in Table 5.40 can be interpreted as follows: The participants consider DRA models useful (28.00%), relevant (26.00%) and important (24.00%). The DRA seems to be a general design (10.00%) that provides what is needed (12.00%) for the industry.

Table 5.40: DRA Usefulness Categories Frequencies

T = 50	Generalisation	Usefulness	Coverage	Relevance	Importance
Frequency	5	14	6	13	12
Percentage	10.00%	28.00%	12.00%	26.00%	24.00%

5.6.2.2.2 DRA Overall Feedback and Rating

This section evaluates the participants' overall feedback and rating about the DRA models. The evaluation process is as follows:

- Collect and map the feedback provided about the DRA into Table 5.41, labelled FT2.
- Analyse FT2 feedback based on the frequency criteria occurences in the text (see Table 3.4) using the cross-examination method (see in chapter 3, DSR Evaluation step).
- Collect DRA overall rating and map them as numerical data in Table 5.42, labelled RT9.
- Plot Table 5.42 (RT9) data into a bar graph representation in Figure 5.13, labelled RF9.
- Calculate the statistical value AAP from Table 5.42 (RT9) data (see Equation 3.2):
  - AAP determines the frequency of participants satisfied with the DRA overall.

Table 5.41: Survey Overall Feedback (FT2)

Category	Overall Participants Feedbacks	Interpretation
Usefulness Coverage Relevance	Overall it **seems very well thought-out** and could be extremely **useful** in practice. However, it is fairly complex and might not be ideal for a newcomer. Perhaps some of the graphics in this form could be further simplified to present DRA's ideas in a cleaner way. I strongly agreed to the '**provide enough components**' statements; are there too many components? Could it be made simpler? Aside from this, the **DRA framework** certainly **makes sense**.	DRA is very well thought is Extremely useful in practice I strongly agreed to the '**provide enough components**' **DRA framework** certainly **makes sense**
Relevance Usefulness	This work illustrates how the DevOps methodology can be applied to IoT development in a **multi-cloud environment**. It will be **useful** for students. It can evolve into a **practical solution**.	DevOps applied to IoT in multi-cloud. A useful, practical solution
Usefulness Relevance	I'm afraid it's just not the kind of thing relevant to my work, but I can see how it might be **useful** to someone starting up a **DevOps, multi-cloud, IoT project.**	Useful for DevOps, multi-cloud IoT
Relevance	I am not technical, so a lot of what is being proposed seemed **practical** from my view. How would this work with mainframes? I deal with the cultural aside of organisations so while this looks pretty good on paper, I wonder how practical it would be to implement. We get involved a lot in implementing transformations. The people side of things is the hardest. Before we even attempt to introduce technical changed, e.g., dev-ops plus CI/CD, we need to set the organisation up to be able to adapt to the new changes that are about to come. Culture is sometimes the biggest impediment	DRA seems Practical

Category	Overall Participants Feedbacks	Interpretation
	to a successful transformation as you cannot change the culture without first changing the organisation, then this becomes a question of 'what is your appetite to change'. My point is that this needs to happen prior to introducing any change to an organisations technical practices and approach if you want it to be successful. Perhaps this is a given as a prerequisite, if not then it should be unless of course, it is a start-up.	
Coverage Usefulness	I really appreciated the breakdown of the theory behind the **DevOps culture**; **it's very good to explain for teaching reasons**. In the enterprise world, most of the companies already take it as a **practice,** and it is consolidated and evolved further with a 'You Build It You Run It' paradigm where operations are part of development, including maintenance and incident management. This is somewhat new to developers to respond to incidents on shifts, without delegating to an ops team designated for maintenance only. This includes the adoption of **paging tools** (such as OpsGenie or PagerDuty) to be able to manage an on-call roster for 24/7 response. In the enterprise world, this **framework** would be separate into two parts, one being technical **implementation**, and the other one being organisation and practices. Architects define the **practices** and the required organisation, while engineers look deeper into the design and implementation of the technical part of the software.	DevOps Culture Very good for teaching reasons In the enterprise world, this framework would be separate in two parts: implementation and practice.
Usefulness Coverage	Great idea about **organizing the structure for an IoT project**, the main reason for projects failures of IoT is the lack of structure.	IT projects Organising the DRA project for IoT
Generalisations	I feel this is a more **general architectural reference** than specific to IoT devices. I feel there are further unsolved challenges there.	General architecture
	Self-remediation, infrastructure provisioning should be a part of the framework.	DRA allows retrospective approach for DevOps team
Relevance	Looks **effective** and definitely have invested a lot of efforts and work but utmost wonderfully presented **Many of Industries are already trying to get on these paths**. Keep up the good work.	Effective Industries are already trying to get on these paths
Importance	It will be great **automated IoT deploy tool** if you keep work on it. It would be good if add a case study of build failure. Sometimes build fails it without reasons.	Automation
Usefulness	Happy to see more researches like this. The reason I put average for the **usefulness** of this framework in the industry is that technological innovation happens every day and new and better **tools** are constantly introduced into the **DevOps space**; therefore some of the tools mentioned in this framework could be replaced and become irrelevant at some point, maybe soon. With that said, I agree the **concept remains valid and could**	Useful DevOps Practices Appropriate Significant concepts

Category	Overall Participants Feedbacks	Interpretation
	be very helpful.	
Relevance	This is an **excellent project**. Brilliant.	Excellent project
Relevance	Thanks for sharing your research with me. Unfortunately, I don't have the technical experience to provide more thorough and valuable feedback. But at a **conceptual level, it all makes sense**. Thanks again, and best of luck!	conceptual level all makes sense
Relevance	Nice **project**, I never imagined using **DevOps for IoT**.	DevOps for IoT
Relevance	**Very interesting**, but I think that one should consider **GitLab CI/CD** as well.	Interesting
Relevance Importance	I have learnt a lot within using the **DRA** approach, and it has enabled me to better understand the way code works and also how these **clouds are integrated together.**	Framework Heterogeneous

Table 5.42: Survey Overall Feedbacks Ratings (RT9)

On a scale of 1 to 5. Please provide an overall rating for the DRA framework	
Strongly Disagree	1.00%
Disagree	2.00%
Average	11.00%
Agree	46.00%
Strongly Agree	22.00%

Overall AAP =

77.00%

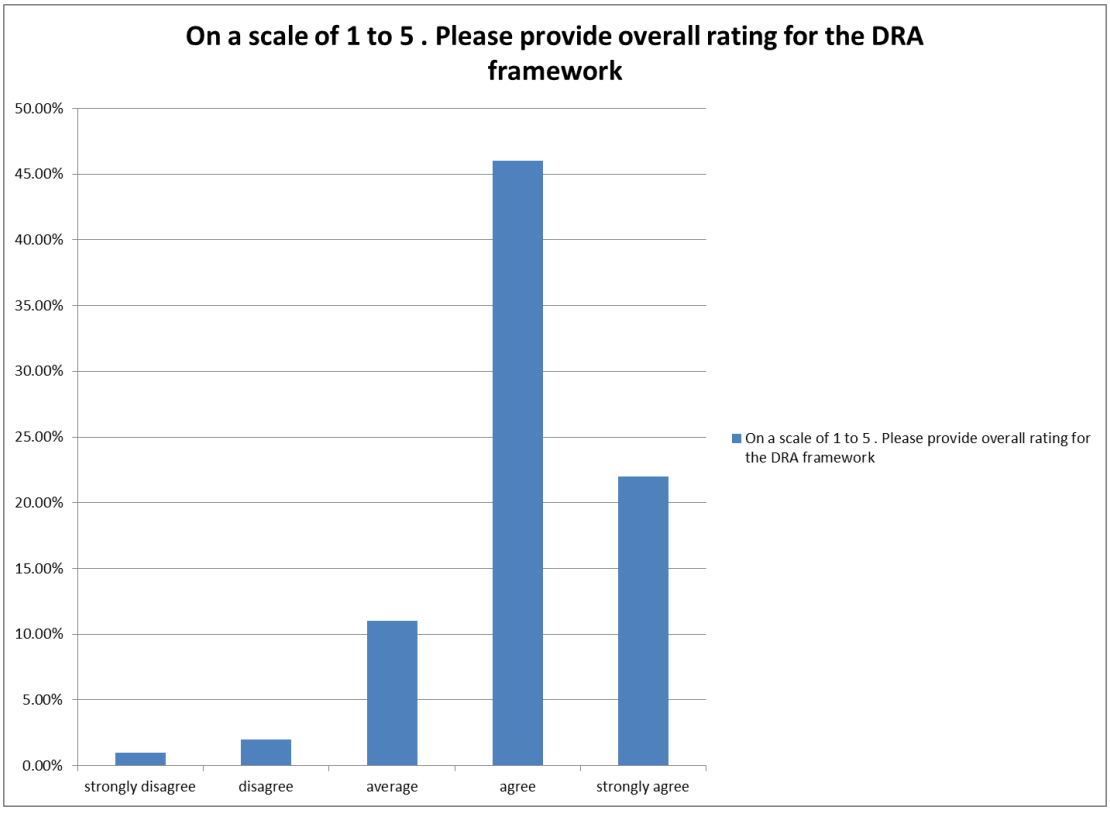


Figure 5.13: Survey Overall Feedbacks Graph (RF9)

Table 5.43: DRA Overall Criteria Occurrences

T = 22	Generalisation	Usefulness	Coverage	Relevance	Importance
Frequency	1	6	3	10	2
Percentage	4.55%	27.27%	13.64%	45.45%	9.09%

REVIEW

The ordinal data in Table 5.42 (RT9) produced a statistical value based on participants' responses. Overall, the DRA Q9 set showed that:

- AAP = 77.00%, indicating that a percentage (77%) of participants agree that the DRA architectural models meet the evaluation criteria (see Chapter 3, Table 3.4).
- In Table 5.41 (FT2), there are T = 22 related references to the evaluation criteria (see Table 3.4). The frequency distribution of the FT2 categories is mapped in Table 5.43. Table 5.43 shows that participants seem to consider the DRA useful (27.27%) and relevant (45.45%) to the industry, and it may be useful for their organisation's context.

5.7. EMPIRICAL EVALUATION OVERALL ANALYSIS

This section presents the overall analysis of the empirical evaluation results. The purpose of this investigation is to provide more insight and evidence of the DRA applicability and novelty based on the evaluation data collected in this chapter. The DRA empirical evaluation was organised into four iterations: industry case study, research case study, teaching case study and industry field survey. The data from the evaluation section are combined into an indicative matrix to provide overall evidence that the DRA meets the overall evaluation criteria (see Table 3.2 and Table 3.4). There are two types of indicative matrices:

1. Quantitative indicator matrix (QIM) (see Table 5.44)
2. Qualitative evaluator matrix (QEM) (see Table 5.45).

5.7.1. QUANTITATIVE INDICATOR MATRIX

The quantitative indicator matrix (QIM) contains the combined numerical data results reported in the empirical evaluation. Three data streams are feeding into the QIM. The data streams are the results of the analysis conducted on numerical data in the teaching case study surveys (SEP and INP) and the industry field survey. The collected data are the:

- AAF from all survey result tables
- AAP from all survey result tables
- Chi^2 test results from all field survey result tables.

The QIM (see Table 5.44) aims to determine that there is a probability of 75% or above that the participants may agree with that the DRA architectural models meet the survey criteria (see Chapter 3, Table 3.4) The QIM is organised as follows:

- reference column: indicates the tables from which the data originated
- AAP column: indicates the AAP from each data source table
- AAF column: indicates the AAF from each data source table
- coverage p-value column: indicates the coverage criteria p-value from each source table
- relevance p-value column: indicates the relevance criteria p-value from each source table
- importance p-value column: indicates the criteria p-value from each source table
- combined DRA models p-value column: indicates the overall two-way p-value
- industry usefulness p-value column: indicates the industry usefulness p-value
- research usefulness p-value column: indicates the research usefulness p-value
- teaching usefulness p-value column: indicates the teaching usefulness p-value
- indicator column P(X): indicates the probability of participants agreeing with AAP.

The Quantitative Indicator (QI) formula is an average value that indicates that the DRA models meet the evaluation criteria (see Chapter 3, Table 3.4) at a probability above 75%. The QI formula is described in Equation 5.1:

Equation 5.1: QI Formula

QI Formula
QI definition: The Quantitative Indicator (QI) is the average probability (AAP) for every table source in chapter 5, the QIM table (see Table 5.44). The probability of a source table is: [Probability (X) or P(X)] = AAP[Table $_{(index)}$] QI = $\sum [P(X)]_{row[i]}$ / Count(row[i]); where (i = the row index in QIM and X = participants score ≥ 3) **QI objective:** Determine the average probability of participants' likely to agree with the data results from source tables (Table ($_{index}$)). It indicates that the DRA models meet the evaluation criteria positively (see Chapter 3, Table 3.4). **Specific QI condition:** Participants agree with a table results: if [AAF > 75%, p-value < 0.01] **QI Result:** QI = $\sum [P(X) > 75\%]_{row[i]}$ / Count(row[i]) [Indicates the average probability of participants scoring ≥ 3 in a particular survey table] **Conclusion:** QI > 75 % then DRA is significant. **Data Source:** Table 5.9, Table 5.11, Table 5.12, Table 5.24, Table 5.25, Table 5.26, Table 5.27, Table 5.28, Table 5.29, Table 5.30, Table 5.31, Table 5.32, Table 5.33, Table 5.34, Table 5.35, Table 5.36, Table 5.37, Table 5.39

Table 5. 44: Quantitative Indicator Matrix (QIM)

Table$_{(index)}$	AAP	AAF	Coverage p-value	Relevance p-value	Importance p-value	DRA Models	Useful for Research	Useful for Teaching	Useful for Industry	P(X)
Table 5.9	892	88.85%								88.85%
Table 5.11	20	100%								100%
Table 5.12	26	100%								100%
Table 5.24 Table 5.25	370	90.25%	< 0.001	< 0.001	< 0.001					90.25%
Table 5.26 Table 5.27	382	93.17%	< 0.001	< 0.001	< 0.001					93.17%
Table 5.28 Table 5.29	386	94.14%	< 0.001	< 0.001	< 0.001					94.14%
Table 5.30 Table 5.31	711	96.33%	< 0.001	< 0.001	< 0.001					96.33%
Table 5.32 Table 5.33	380	92.68%	< 0.001	< 0.001	< 0.001					92.68%
Table 5.34 Table 5.35	625	95.28%	< 0.001	< 0.001	< 0.001					95.28%
Table 5.36 Table 5.37	2848	93.87%				<0.001				93.87%
Table 5.39	229	93.09%					< 0.001	< 0.001	< 0.001	93.09%
QI Result	**QI= $\sum$ [P(X) > 75%]$_{row[i]}$ / Count(row[i]) = 86.47 %**									

REVIEW

The QI = 86.47%, indicates that the cohort of participants in the surveys seem to agree that the DRA models meet the evaluation criteria at a probability [Probability > 75% and/or p-value < 0.001]. Overall it can be concluded that participants seem to consider the DRA framework applicable and reusable in a class of situations and provide a novelty knowledge base about the DevOps adoption in any context.

5.7.2. QUALITATIVE EVALUATOR MATRIX

The qualitative data evaluator is a combination of feedback gathered in Chapter 5 and the aim of providing a further indication that the DRA architectural models meet the evaluation criteria (see Chapter 3, Table 3.2). Three data streams are feeding into the qualitative evaluator matrix (QEM). The data streams are acquired from Tables 5.6, 5.7, 5.38 and 5.41, and they are a combination of feedback and comments provided by the industry case study, research case study and the survey participants. QEM (see Table 4.45) feedback was analysed using the cross-examination method between the participants' feedback and the combined evaluation criteria in Table 3.2. This analysis aims to connect or relate the hypotheses (evaluation criteria) to the experts' feedback. The QEM (see Table 5.45) aims to provide overall insights into the DRA applicability and novelty from the participants' perspectives.

Table 5. 45: Qualitative Evaluator Matrix

Criteria	Feedback—Participants' Quotes
Importance Reusable Generalisations	'One particular aspect that I think is very **important in the DRA framework is the flexibility** to choose the **instantiations of each component** (**or indeed have several instantiations**). Given the **heterogeneous** nature of the **cloud and IoT environments**, I think this is a critical feature'.
Usefulness Coverage Relevance	'The **models provided are all very useful** as they **deconstruct and disambiguate what is required** in order to **deploy any code as a single or multi-cloud application** - this is achieved through presenting both abstract and concrete examples and clearly defining steps involved at each stage of the **process**'.
Usefulness Importance	'The **continuous integration** and automated deployment to multi-cloud are very **useful**'.
Generalisations Coverage	'All of it, it provides a **comprehensive overview** of **what's needed for** an IoT application deployment and management'.
Novelty Importance Relevance	'**Decentralized** logging, **cloud-hosted CI, deployment to multiple hosting vendors, etc. are all acceptable modern solutions for the demonstrated problem**'.
Usefulness	'**DRA logical model specifications and DRA pipeline instance** are **very useful in the enterprise world**, as they focus on the low-level **implementation scheme**'.
Relevance	'The fact that the entire **deployment process** is **automated** and seamless is really nice'.
Relevance	An interesting approach to **combine DevOps and IoT**.
Coverage Usefulness Generalisations	'I do like this **model for the deployment of applications**. I think it is quite **extensive** and **applicable** to developers, (ops) and **IT management,** including **project management**'.
Usefulness	'It provides a **framework** that I'm pretty sure would be **invaluable** for people that don't know all the components and would like to **implement** it'.
Coverage	'Correctly identifies **the benefits of DevOps**'.
Importance Reusable	'Very **appropriate tools** and real **use case implementation**'.
Relevance	'It gives a clear idea of a **path to production** that could be used in **IoT development process**'.
Usefulness	'**Useful for IoT cloud solutions**'.
Importance Usefulness	'The **DevOps** section is very good as all the **tools are perfectly used**. As it will make good **automation in architecture**'.
Generalisations Relevance Usefulness	'Having a **high-level view of specifications**, **logic, modelling**, and behaviour of the whole system is paramount to a **solid implementation** of the pipeline between **IoT and multi-cloud**'.
Relevance Importance	'It's good that you're creating a **platform on multi- cloud** level. The idea of **Automating end to end pipeline** is a good thing'.
Usefulness	'I am really **impressed with DRA** I think the IoT environment are missing some DevOps structure like that provided by DRA so it is very **useful** to guide teams to have a simple workflow to **implement DevOps in IoT projects**'.
Relevance Usefulness Reusable	'The value proposition that is to enable **IoT architecture by the DevOps efforts** in order to start using this amazing technology for application like smart cities or even the set an **cloud architecture** for a smart house system'.
Importance Usefulness	'Definitely **automation**. When IoT makes to the top, it will be so hard to test things manually, and those things are going to be interacting directly and physically with

Criteria	Feedback—Participants' Quotes
	users... So the **automation process** it's going to be a very good friend for the UX of anything related IoT'.
Usefulness Relevance Importance	'This can be **useful referenc**e document for **IoT based Apps deployment** with **DevOps culture in team process and tool set**'.
Generalisations	'I thought it was a good **overview** of how **DevOps can be applied with IoT**'.
Importance	'It is a good **high level design** for **IoT cloud app deployment** workflow'.
Relevance Importance	'**Agility, CI/CD, automation, Speed**'.
Importance Relevance	'Allowing users to **interact with IoT devices** remotely using **cloud services**. The ability to **track usage** through the use of **Paper-trail and automation of IoT device by using the cloud**'.
Relevance Importance Coverage Novelty	'Perfect **pick up as industry is facing this issues** as to how to **integrate or implement DevOps transformation** when it comes to **Cloud and IOT related Apps**. In a nutshell its part of **Digital Transformation** which is one of future in Industry'.
Usefulness Coverage Relevance	'**Overall it seems very well thought-out** and could be **extremely useful** in practice. However, it is fairly complex and might not be ideal for a newcomer. Perhaps some of the graphics in this form could be further simplified to present DRA's ideas in a cleaner way. I strongly agreed to the "**provide enough components**" statements; are there too many components? Could it be made simpler? Aside from this, the **DRA framework** certainly **makes sense**'.
Coverage Usefulness	'I really appreciated the breakdown of the theory behind the **DevOps culture**; it's very good to explain for teaching reasons. In the enterprise world most of the companies already take it as a **practice** and its consolidated and evolved further with a "You Build It You Run It" paradigm where operations are part of development, including maintenance and incident management …etc'.
Relevance	'Looks **effective** and definitely have invested lot of efforts and work but utmost wonderfully presented Many of Industries are already trying to get on these path. Keep up the good work'.
Usefulness	'Happy to see more researches like this. The reason I put average for the **usefulness** of this framework in the industry is because technological innovation happens every day and new and better **tools** are constantly introduced into the **DevOps space**, therefore some of the tools mentioned in this framework could be replaced and become irrelevant at some point, maybe soon. With that said I agree the **concept remains valid** and could be very helpful'.
Usefulness	'**Tools** used in Operation **model pipeline** are **industry used** tools and are **excellent choice** for the **DRA Operation Mode Pipeline**'.
Reusable	'**Configuration template** is easy to use and can be **replicated**'.
Coverage Usefulness	'**DRA framework** would help organisations understanding **DevOps methodologies** and **agile application deployment and delivery**'.
Generalisations Usefulness Novelty Coverage Reusable	'**DRA is applicable and is fit for purpose to setup the DevOps multi-cloud**'. 'DRA is general in the sense that it is not fixed to one situation or environment and **can adapt to different situations** and be used with different technology stacks as appropriate to the situation. Thus **DRA is applicable** to a class of problem situations and is applicable to several instantiations'. '**DRA offers new knowledge**, which has not been discussed before in the form of complex **DevOps for Multi-cloud and IoT**'.

Criteria	Feedback—Participants' Quotes
	'DRA models seem **to provide sufficient explanation** about the elements and their relationships as a "design knowledge", which can be **used or reused for a class of a problem addressed in this work**'. '**My overall feedback is that DRA can be successfully instantiated for the similar research lab environment needs for the deployment of IoT applications using multi-cloud. Overall DRA is fit for purpose**'.

Table 5. 46: QEM Criteria Occurrences

T = 67	Generalisations	Usefulness	Coverage	Relevance	Reusable	Novelty	Importance
Frequency	6	18	9	14	5	3	12
Percentage	8.96%	26.87%	13.43%	20.90%	7.46%	4.48%	17.91%

REVIEW

Table 5.45 shows T = 67 related references to the evaluation criteria (see Chapter 3, Table 3.2 and Table 3.4). The occurrences of the evaluation criteria in the QEM is mapped in Table 5.46, which shows that participants seem to consider the DRA useful (26.87%), relevant (20.90%) and important (17.91%) to research and industry, and they believe it may be useful for their organisation's context.

QEM (Table 5) quotes seem to acknowledge the following elements:

- DRA is instantiable and easy to implement, using various tools and technologies.
- DRA is flexible and may be reconfigured with another technology stack.
- DRA includes integrated tools that enable automated deployment to the multi-cloud.
- DRA is based on high-level modelling that supports DevOps concepts and practices.
- DRA enables IoT processes and IoT interactions in IoT projects.
- DRA enables DevOps, cloud and IoT relationships.
- DRA enables decentralised logging and notifications.
- DRA supports DevOps culture and the human factor.
- DRA is an extensive comprehensive architecture that supports digital transformation.
- DRA provides a new knowledge base regarding adopting the DevOps approach.
- DRA provides a fast and clear path to software production in agile.

5.8. FUTURE RESEARCH

The second version of the DRA framework (DRAv2.0) was incrementally developed, refined and presented at the international conference CBI2018, the 20th IEEE International Conference on Business Informatics 11–13 July 2018, Vienna, Austria; (Ghantous & Gill 2018). DRAv2.0 is an upgrade from single-cloud deployment (DRAv1.0) to multi-cloud deployment (see Chapter 4). The evaluation of the DRA in Chapter 5 determined its relevance, usefulness and applicability for the industry, research and teaching contexts.

The DRA framework answered the main research question and achieved the thesis objectives (see Chapter 1). However, as noted in the previous section and the research scope (see Chapter 1), there are limitations to the project, which indicate that there is room for improvement in the framework. Several future research options can be suggested as improvements to the DRA. The suggested future research topics are outside the scope of this thesis but are recommended by industry survey participants and industry experts. The implementation of the suggested topics could be considered an important upgrade to the current version of the DRA.

This section evaluates the participants' responses regarding suggested improvements to the DRA at the industry level. The feedback was in response to the Q8 set (see Appendix D). Feedback to the Q8 set provided key suggestions to further improve the DRA, as well as valuable ideas that may be considered as future research projects or DRA upgrades. The evaluation data results for the Q8 set are presented in Table 5.47, which is organised as follows: the participants' suggestions column, which contains the participants' feedback and comments; and the interpretations column, which includes the researcher's answers to suggestions. In this column, the researcher identifies the key ideas considered possible future research topics.

Table 5. 47: Suggested Improvements

Suggested Improvements	Interpretations
The only issue I've seen is the **scope**; this **framework** is useful **beyond the scope of just IoT.**	DRA output is the architecture design; it can deploy other application types, not just IoT.
Need to look more at the **IoT device** part of the lifecycle to elaborate how that is managed.	DRA output is the architecture design; IoT devices management and software patching may be considered a future research.
Including more **open-source tools** and including the **private/on premise cloud** may be a nice touch.	DRA can be configured with any tools set. The tools used in the DRA pipeline are for demo and testing.
The **technology stack** is acceptable however I would be using AWS ECS for long running services and Lambda/API Gateway where applicable, Heroku's performance in Australia is subpar and cost model is poor for large scale projects, most companies I've worked for don't tend to use Cloud hosted CI for security reasons and the most fit for purpose solution seems to be Buildkite. Papertrail/Loggly/etc. aren't generally used for cost reasons at scale however SumoLogic is priced more effectively. HipChat is an abomination and end-of-life, use Slack.	DRA can be configured with any tools set. The tools used in the DRA pipeline are for demo and testing. CI broker security is a two-way authentication system enabled by exchanging API tokens. Although security is not in this project scope.
Multi stage deployment would need more attention in terms of testing the app before releasing.	DRA BitBucket and CI broker both enable in-house branching. It is a common practice to test features on development branches before committing to the production master. The video shows ready IoT app features deployed. The video is used for pipeline testing only. The project aim is not to address the best practices of software application development.

Suggested Improvements	Interpretations
Think about the activities of the developers up front of the **DRA such as using TDD or BDD** activities and how they will influence how the DRA is used e.g., writing tests first, writing code to satisfy the tests, checking in -> **deployment, then refactoring the code, running the tests, checking in -> new deployment** etc. etc.	DRA output is the architecture design; how DRA may be adopted in a current organisation environment. Testing scripts and code are used within the DRA design.
This is more of a question than an improvement. I saw that there are multiple steps in the **deployment** process. What if it fails at one of the last steps? When you re-run the **deployment process** does it start from the first step or continue from the last successful step? **Is it configurable**? Maybe something you want to look into?	DRA enables dev/prod (branching). DevOps team may work on the production branch and test their features. Application new features are released to production after passing the tests. Git technology enables the team to revert to previous stable software versions.
What strategies are there for **controlling and monitoring** the **multi-cloud environments** in the context of **DRA Framework**?	DRA enables real-time application monitoring. Logs of incidents are collected and sent to Slack. DevOps team can act in real-time to fix application issues based on incidents reports.
The only thing missing in the **framework** is the addition of **high availability**. We are using **multi-cloud** but what is not covered is failure. This is extremely important if you want to promote this **framework** from research to industry.	Multi-cloud ensures multiple service availability for the application. IT means that the application can be accessed from the different vendor if one or more has failed.
The **tools** change all the time; I feel that is quite a challenge to keep up to date. I think there are some unsolved challenges handling **multi-cloud deployments**. I would suggest taking a look at Artifactory. A lot of steps in **the pipeline** may generate artefacts which are good to keep somewhere. I feel that while the **technology side has been covered**, in **practice getting a DevOps team** to work with a few other teams can be challenging. I would separate practices into a different set of models and contrast what different practices say. For example, I'm about to take some SAFE training in **DevOps! (agile**), but there are books in other materials and other ways to work under different methodologies.	DRA output is the architecture design; tools and services can be added or replaced within the design.
Self-remediation, infrastructure provisioning should be a part of the **framework**	DRA CI broker enables clouds provisioning. DRA requires CD clouds to be a deployment platform only. CD clouds do not host database or provision **deployments in DRA**.
Address **deployment to multiple environments (e.g., test, staging, prod)**; **configuration management**; access to shared infrastructure - brokers, MQs, etc.	DRAv2.0 enables multi-cloud provisioning and deployment. The deployment infrastructure is intentionally kept to the CD cloud.
An important aspect that should be considered is **governance**. Although it does not play a role in **automation**, it is a very important component in real businesses. Could be an improvement for future	<Possible Future Research Idea>

Suggested Improvements	Interpretations
versions.	
Firstly I would ask you to concentrate on the **IoT application** as the **deployment strategies**, build tools differ for android, IOS and windows. Try using some kind of Artifactory or the roll back purpose if the release doesn't go well. Research more about the tool set and will that particular tool set can be used with your **application pipeline**.	DRA output is the architecture design; tools and services can be added or replaced within the design. Organisations adopting DRA may use their in-house applications and tools into DRA framework configuration. Roll-back option is enabled on Git if DevOps team decided to revert to stable application version based on current testing logs.
Consider have a step to create and **monitoring error logs** from production devices.	<Possible Future Research Idea>
Missing from the **framework** and the **pipeline** it the focus on the telemetry and analytics that would be **retrieved from IoT devices**. This usually involves large analytics platforms like Big Query, Apache SOLR or Hadoop. CDNs are usually employed to provide low-latency updates to IoT devices as well.	DRA output is the architecture design; tools and services can be added or replaced within the design. Organisations adopting DRA may use their in-house applications and tools into DRA framework configuration.
Once **deployed**, how do you **address hot fix**?	DRA enables dev/prod (branching). DevOps team may work on the production branch and test their features. Application new features are released to production after passing the tests. Git technology enables the team to revert to previous stable software versions.
DevOps is all about improvement and there can much addition to this **framework** in coming future. Using feedback loops can be helpful in improving processes and further learning which specifically found for an **IoT app infrastructure**. This is a real good start indeed and helps many people.	DRA framework is based on DevOps practices and enables collaboration and communication. This help with feedback and logging.
Security and Metrics are an important piece of **DevOps**.	<Possible Future Research Idea>
Security for infrastructure as well as application and **deployment speed improvements**	<Possible Future Research Idea>
Scope for improvement. **DevSecOps or DevQAOps** can be included as part of detailing. E.g., **TDD / BDD Framework or Security** related **tools** can be added.	<Possible Future Research Idea>
Like I said before, I did not see any **IoT specific design** in **DRA framework**.	IoT specific design is outside the project purpose. The aim is to deploy IoT apps to multi-cloud using DevOps.

Future research topics identified in this research are shown in Table 5.48.

Table 5. 48: Future Research Ideas

Future Research	Source
To provide IoT devices management system as an upgrade to DRA IoT application that aims to automate IoT sensors discovery and connectivity to a network	Table 5.47
An important aspect that should be considered is **governance**. Although it does not play a role in **automation**, it is a very important component in real businesses. Could be an improvement for future versions.	Table 5.47
DRA security improvements. DevOps security	Table 5.47
DRA design to enable parallel application deployment to multi-cloud which improves speed of deployment and enable parallel branching	Table 5.47
DRA security improvements. IoT security	Table 5.47
Extend DRA to support **drone and robotics** application development and deployment projects. It is a huge research area and perhaps another PhD (s) idea, for the secure deployment of drone and robotics application projects.	Table 5.7

5.9. SUMMARY

This chapter evaluated the DRA framework using an empirical evaluation, which was composed of four iterations: industry case study, research case study, teaching case study and industry field survey. The collected data from the evaluation iterations were reviewed to determine the relevance and importance of the DRA and whether it covered the institution's (industry, research, teaching) needs. The industry case study results appeared to indicate that the DRA is applicable and flexible at the industry level. The research case study results indicated that the DRA is fit for purpose and useful. The results from the research case study also showed that the DRA could be part of future innovative projects. The teaching case study (for SEP and INP) indicated that students were satisfied with their course content and have learnt a practical DevOps approach. Finally, the industry field survey data were organised into quantitative and qualitative data. The evaluation of the quantitative data indicated that most participants appeared to agree that the DRA was applicable and fit for purpose at the industry level. The evaluation of the survey qualitative data showed that the participants appeared to consider the DRA useful and believed that it might cover industry needs. The empirical evaluation combined data was analysed to determine that the DRA architectural models meet positively the evaluation criteria discussed in chapter 3 (DSR). The survey participants imparted valuable comments and suggested further improvements for the DRA that could be future research projects. The thesis output, limitations and key contributions are discussed in Chapter 6.

Chapter 6: Discussion and Summary

This chapter outlines the research journey that started in August 2016. This chapter also presents the main output of the research—the DRA. The contributions and publications are also listed in this chapter. The DRA limitations are discussed based on the feedback from the empirical evaluation. Finally, this section includes an overall summary of the research project.

6.1. Research Journey and Output

This section discusses the research journey and its outputs. It is organised as follows:

- The research journey, which started in August 2016
- The main output of this research—the DRA framework.

6.1.1. Research Journey

The research journey is illustrated in Figure 6.1. It began in August 2016 at the higher degree research level and was upgraded to PhD in autumn 2017. The research thesis is expected to be submitted for review by the end of spring 2019.

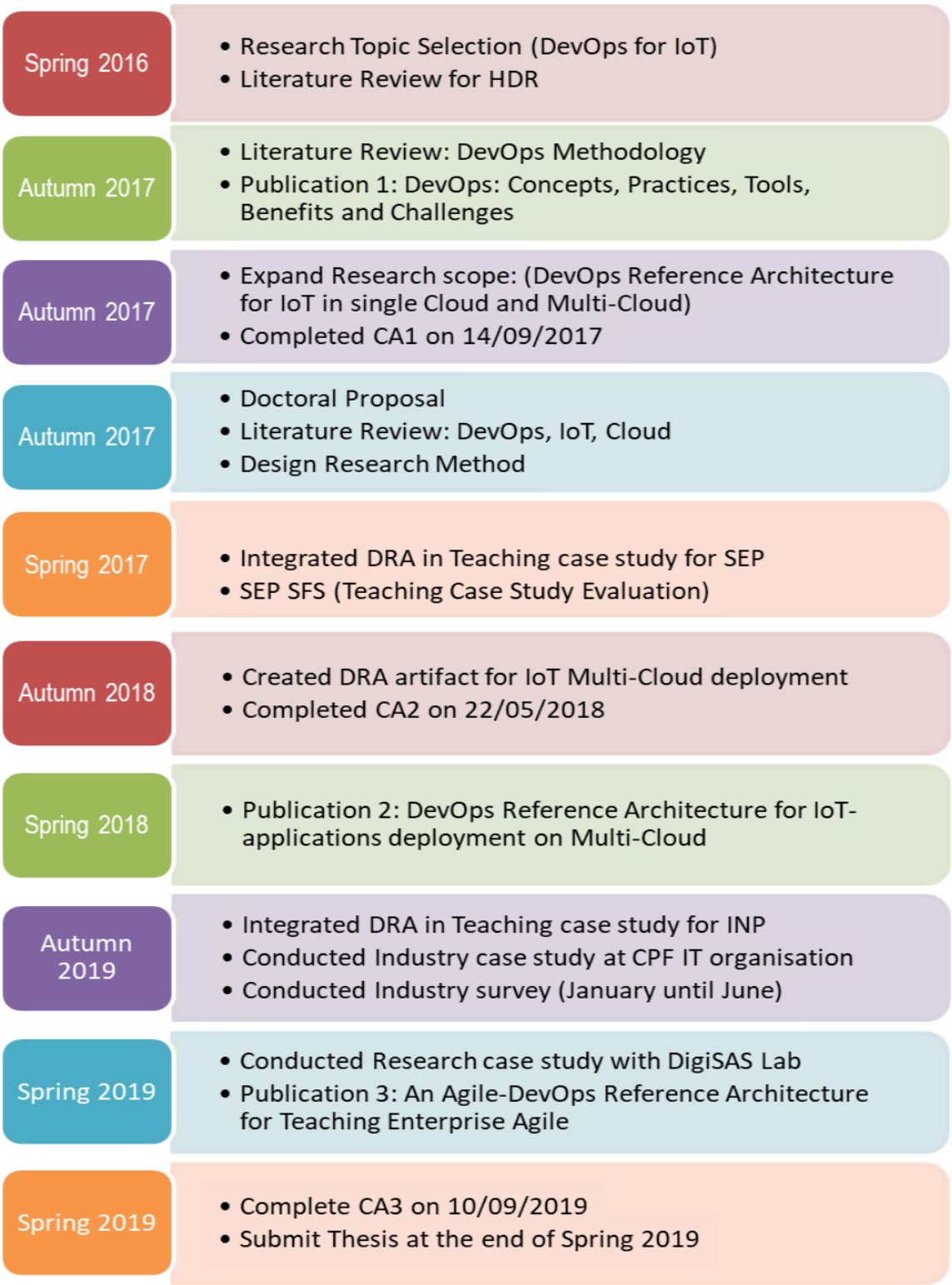


Figure 6.1: Research Journey

6.1.2. RESEARCH OUTPUT

The main research output of this thesis is the DRA framework for IoT application deployment on the multi-cloud. The framework was discussed in Chapter 4 and evaluated in Chapter 5 using an empirical evaluation. The DRA framework was constructed using the DSR method outlined in Chapter 3. The following sections outline the output of this research. The DRA framework has three main components (see Figure 4.1): framework characteristics, framework design and framework composition. The DRA framework logo is illustrated in Figure 6.2.

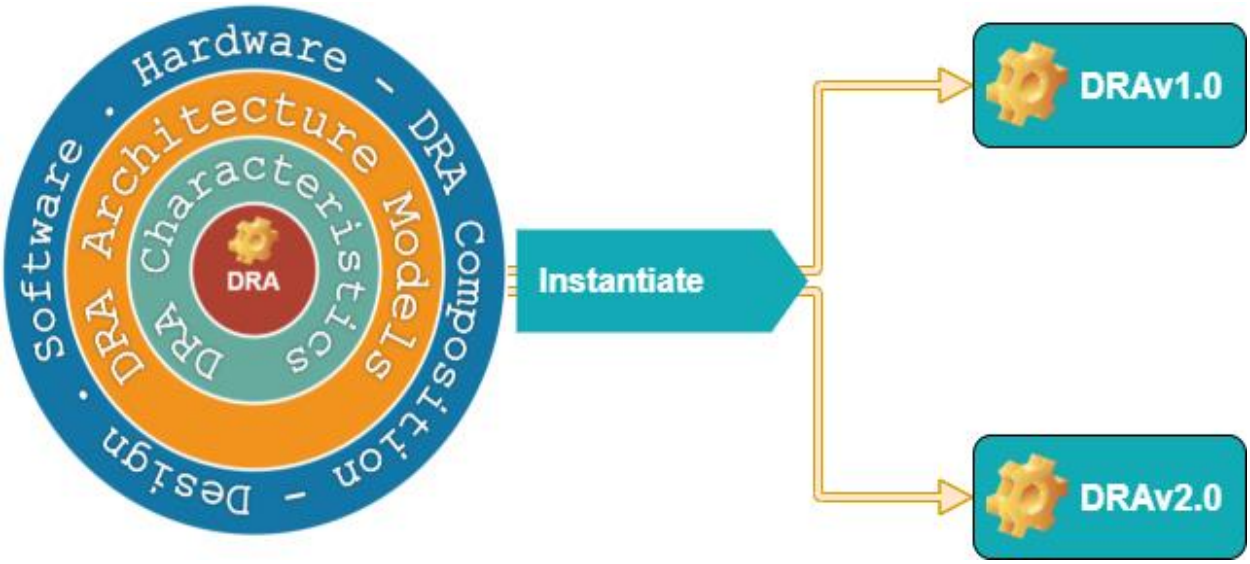


Figure 6.2: DRA Framework Overview

- **Framework Characteristics**

The framework characteristics were discussed in Chapter 4. They contain a set of nine elements: human factor, infrastructure, abstraction, process, tools, product, business value, rules and legal. The DRA can be tailored using its characteristics as guidelines that support the relationship between DevOps, IoT and the multi-cloud. DRA characteristics can be used in the implementation process of the DRA for a situation in the organisation context. The framework characteristics' output is mapped and explained in Table 6.1, which outlines the characteristics, a brief description, and where the characteristics are located in this thesis.

Table 6. 1: DRA Characteristics Output

Characteristic	Description	Source
Abstraction	Abstraction represents a logical view of the software development structure. The abstraction characteristic incorporates several mechanisms such as people-oriented, service-oriented, tools-oriented and process-oriented. A framework abstract design may combine than one abstraction mechanism. The mechanism combination is used to create the design model for the development environment (or workspace).	Chapter 4 Section 4.2.1 Figure 4.3
Human factor	The human factor is an essential element in the DevOps framework characteristics. DevOps approach supports people or DevOps team. In an organisation, a DevOps team may involve many individuals that have one or many roles. People in a DevOps team may belong to different types of communities and have different types of skills (social skills, development skills, management skills, technical skills). Regardless of the people skills, location, languages or knowledge base, a DevOps team is expected to be involved in the entire product deployment lifecycle (or development chain), which includes code management, build, testing, deployment and monitoring.	Chapter 4 Section 4.2.2 Table 4.1 Figure 4.4
Infrastructure	The infrastructure characteristic is also a foundational element in the framework construction. Cloud seems to provide the required infrastructure for the DRA framework. Cloud IaaS, PaaS and SaaS offer virtual servers, resource-sharing, adaptive deployment options, software services, and standard security measures. Cloud supports automation, CI, continuous deployment, monitoring. In the DRA framework, DevOps and cloud services integrate into architecture. The DRA cloud infrastructure provides the development platform for the DevOps team to use the tools and create the product.	Chapter 4 Section 4.2.3 Figure 4.5
Process	The process characteristic is an important element in the DRA framework. A process is composed of different sub-processes such as design, development, deployment, business. Processes are compliant to rules, legal requirements that represent the policy in an organisation context. A process can be categorised as a software process, design process, business process, and deployment process. The software process is the development procedure used to create a software component. The deployment process is combination practices and functionalities required to create and deploy a product. The process is the progression of a product from development to delivery.	Chapter 4 Section 4.2.4 Table 4.3 Figure 4.6
Tools	Tools characteristic refers to DevOps and non-DevOps tools (cloud/multi-cloud services) that are used in a process to create a product. The tools characteristic is used in the development workspace founded on the abstract design of the framework. In the DRA framework, tools are used to create an operational model to deploy and deliver a product (IoT application). However, the framework is not fixed to a particular set of tools. Organisations may use different tools suitable for the development context of the project.	Chapter 4 Section 4.2.5 Figure 4.7

Characteristic	Description	Source
Product	The product characteristic presents the executable project output (e.g., IoT application). It is the output of a process or multiple processes conducted in the development workspace. The proposed DRA framework deploys and delivers a product using an operational model composed of several tools.	Chapter 4 Section 4.2.6 Table 4.4 Figure 4.8
Business Value	The business value characteristic has not been investigated to any great extent in this thesis. However, the business should be included in the characteristics because it affects the deployment and delivery of the product (e.g., IoT application) to users. The business value is generated by using other characteristics such as people, tools, process, infrastructure and product.	Chapter 4 Section 4.2.7
Rules	The rules characteristic is explicitly included in the extended characteristics category of the DRA framework. Business rules are a form of knowledge that states the guidelines being adopted in the organisation. Business rules are an essential concept and can be seen as general declarations about the organisations' means of conducting business.	Chapter 4 Section 4.2.8
Legal	The legal characteristic is explicitly included in the extended characteristics category of the DRA framework. The legal agreement and governing requirements may also affect the software development process and product (IoT application) delivery.	Chapter 4 Section 4.2.9

- **Framework Design**

The DRA design is a generic design model founded on the DRA characteristics discussed in Section 4.2. DRA characteristics are common terminologies used to represent DRA elements (DevOps and cloud/multi-cloud). DRA characteristics are used to create a reference architecture design to deploy a product (IoT application) to the cloud/multi-cloud (the primary objective of this research). Table 6.2 also includes the two versions of DRA architecture sets: DRAv1.0 and DRAv2.0.

Table 6. 2: DRA Design Models

Model	Description	Source
Contextual model	The DRA contextual model describes the relationship between DevOps, multi-cloud and IoT at a higher level. This model shows that DevOps can be used for deploying IoT applications to the multi-cloud environment.	Chapter 4 Section 4.3.1 Figure 4.9
Conceptual model	The DRA conceptual model describes the integration of elements of the three technologies (DevOps, multi-cloud, IoT) into one blueprint tailored to support the DRA characteristics. The conceptual model includes a vital mechanism—the CI broker. The CI broker is an essential part of the DRA conceptual model because it incorporates several operations. For instance, it enables automation (build, testing, logging, deployment). The CI broker enables CI, branching development and automated code synchronisation.	Chapter 4 Section 4.3.2 Figure 4.10

Model	Description	Source
	Most importantly, it hosts the deployment configurations for the IoT application. The CI broker packages the IoT app in a container and deploys it to the multi-cloud platforms used in the DRA instance. This procedure prevents any of the clouds incorporated in the multi-cloud platform from hosting the IoT application deployment parameters and consequently prevents vendor lock-in.	
Logical model	The DRA logical model is founded on the conceptual model. It expands the conceptual model and incorporates DevOps practices. The DRA logical model is composed of five components (M_1–M_5). The components represent the DRA instance tiers or phases. Each logical model component includes functions and features related to the required DevOps practices and cloud service in that tier. The logical model components are integrated into a logical view that represents the interactions between these components. The logical model represents the blueprint for physical model instances of the DRA.	Chapter 4 Section 4.3.3 Figure 4.11 Table 4.5
Physical model	The DRA physical model layer is an implementation of the DRA logical model. It represents a harmonious integration of tangible DevOps tools and cloud services. The physical model is the template used to create the DRA operational model.	Chapter 4 Section 4.3.4 Figure 4.12
Operational model	The operational model of the DRA framework is based on the physical model that enables the logical model features (DevOps practices). The DRA operational model is configured using an integrated set of DevOps tools (see Chapter 2). The DRA operational model provides automated IoT application deployment to the cloud/multi-cloud. It is the application of the logical model that represents the DRA characteristics incorporated in the DRA conceptual model. The DRA operational model is not fixed to a particular set of tools but can be configured using numerous tools to suit the context of the organisation.	Chapter 4 Section 4.3.E
DRAv1.0	The DRAv1.0 structure is composed of four constituents or models: 1) conceptual, 2) logical, 3) physical, 4) operational. The operational model is an instance of a physical model that supports IoT application deployment on the single cloud (e.g., Heroku). The operational pipeline construct is built using DevOps tools and cloud services.	Chapter 4 Section 4.5.1 Figure 4.18
DRAv2.0	The DRAv2.0 (see Figure 4.19) structure is composed of four constituents or models: 1) conceptual, 2) logical, 3) physical, 4) operational. The operational model is an instance of a physical model that supports IoT application deployment on the multi-cloud (e.g., Heroku, AWS, GAE). The operational pipeline construct is built using DevOps tools to provide end-to-end automation of IoT app deployment on the multi-cloud (Heroku, AWS, and GAE). DRAv2.0 is an upgraded version of DRAv1.0.	Chapter 4 Section 4.5.2 Figure 4.19

- **Framework Composition**

This section presents the DRA composition of the framework (see Figure 4.1). The DRA composition incorporates the essential elements needed to create DRA instances for deploying IoT applications to the cloud/multi-cloud. The DRA composition is founded on the DRA characteristics that were used to create a generic reference architecture for deploying IoT applications (products) to the cloud/multi-cloud (infrastructure). The DRA composition integrates the DRA models to create practical implementation (instances) applicable in the context of the organisations. The DRA composition includes three main components:

- resources (architecture design, software, hardware)
- configuration (Pipeline, IoT application, IoT network)
- output (DRA models, DRAv1.0 instance, DRAv2.0 instance).

The DRA composition used for creating an instance of the framework is mapped in Table 6.3.

Table 6. 3: DRA Instance Components

Component	Description	Source
DRA abstract architecture	DRA abstract architecture is composed of the five main design models: contextual, conceptual, logical, physical and operational.	Chapter 4 Section 4.4.1
DRA pipeline structure	The DRA pipeline is a physical instance of the DRA operational model. The DRA pipeline enables automation and CI for IoT application deployment. The DRA pipeline lifecycle supports DevOps concepts. There are two versions of the DRA pipeline: DRAv1.0 and DRAv2.0. Both versions are founded on the DRA architecture. DRAv2.0 (multi-cloud) is an upgraded instance of DRAv1.0 (single cloud). The DevOps tools set used in both versions can be substituted with other tools that perform the same roles.	Chapter 4 Section 4.4.2 Figure 4.13 Figure 4.14 Table 4.7 Table 4.8
DRA software component	The DRA software component used in this research is a Java maven web application called maven-app-heroku https://maven-app-heroku.herokuapp.com/. IoT app interacts with IoT sensors using Python scripts. Note: The DRA applies to many software situations and can adapt to various programming languages. The application used in this research is designed for evaluation and POC purposes.	Chapter 4 Section 4.4.3 Table 4.9 Table 4.10 Table 4.11
IoT devices and sensors	The IoT devices and IoT sensors (IoT network) used in this research are the following: - RPIB - 4 LED lights connected to 4 GPIO pins (13, 17, 19, 22) (named Sensor_A). - Motion Sensor + 1 LED connection to a single GPIO pin [12] (named Sensor_B).	Chapter 4 Section 4.4.4 Figure 4.17

6.2. KEY CONTRIBUTIONS AND PUBLICATIONS

The DRA framework outlined in the previous section was evaluated and tested in Chapter 5. Three types of testing and evaluation were used to determine the validity and relevance of the framework in the industry. Key publications contributed to the construction of the DRA framework. The conference publications were peer-reviewed by key international researchers and experts. This section presents the key contributions of the research (see Table 6.4):

Table 6.4: Thesis Key Contributions

Contribution	Reference	Source
DRA framework	The main contribution of this thesis is the construction and development of the DRA framework for IoT application deployment on the multi-cloud. It is also important to add to this key contribution to the empirical evaluation results conducted in Chapter 5 (industry case study, teaching case study, research case study, industry field survey).	Thesis output Chapter 4 Chapter 5 Chapter 6.1.2
Conference	Ghantous, G. & Gill, A. 2017, 'DevOps: concepts, practices, tools, benefits and challenges', *PACIS2017*. http://aisel.aisnet.org/pacis2017/96	PACIS(2017)
Conference	Ghantous, G.B. & Gill, A.Q. 2018, 'DevOps reference architecture for multi-cloud IOT Applications', *IEEE 20th Conference on Business Informatics 2018*, vol. 1, pp. 158–167. https://ieeexplore.ieee.org/abstract/document/8452669	CBI2018 IEEE
Journal	Ghantous, G.B. and Gill, A.Q. 2019, 'An agile-DevOps reference architecture for teaching enterprise agile', International Journal of Learning, Teaching and Educational Research, vol. 18, no. 7. https://opus.lib.uts.edu.au/handle/10453/135151	IJLTE
Journal (in progress)	Title: 'An Empirical Evaluation of the DevOps Reference Architecture for Multi-Cloud IoT applications' Authors: Ghantous, G.B. and Gill, A.Q. Journal home-page: https://www.computer.org/csdl/magazine/so	IEEE Transactions
Journal (in progress)	Title: 'The DevOps Reference Architecture Framework' Authors: Ghantous, G.B. and Gill, A.Q. Journal home-page: https://www.computer.org/csdl/magazine/so	IEEE Software
Conference (in progress)	Title: The DevOps Reference Architecture Evaluation – A Design Science Research Case Study Authors: Ghantous, G.B. and Gill, A.Q. Conference home-page: https://www.uantwerpen.be/en/conferences/business-informatics/	IEEE CBI 2020

6.3. RESEARCH LIMITATIONS

The DRA framework in this thesis has been tested in the case studies (see Chapter 5) and assessed by experts using survey questionnaires. The framework has also been peer-reviewed at renowned conferences (Ghantous & Gill 2018). The DRA has been evaluated using an empirical evaluation conducted in Chapter 5. It was noted in Chapter 1 that the current version of the framework (DRAv2.0) has a few limitations. Table 1.3 indicated that the DRA has four current limitations that have been excluded from the research objectives. This section discusses the project's exclusions (DRA limitations) as outlined in Table 1.3.

- **Custom Pipeline Security**

The DRA operational model (see Chapter 4) provides an end-to-end automated pipeline instance created by integrating DevOps tools (see Tables 2.12–2.22). A DRA pipeline toolset is integrated by exchanging the API key (access key). The integration of tools provides standard security to the automated deployment process chain consisting of **Code → Build → Provision → Test → Defect Fix → Deploy → Release.** The DRA framework relies on the standard security measures provided by the integrated tools and does not provide custom or additional security measures to the deployment process chain.

- **Custom IoT Security**

This thesis uses a sample IoT network (see Chapter 4) to test the interaction of IoT app (maven-app-heroku) deployed on the multi-cloud with IoT devices and sensors. The IoT sample network was used to demonstrate and test the DRA (see Chapter 5). The IoT app (maven-app-heroku) used for demonstration in this project interacts with the IoT network over the internet using SSH commands (see Table 4.10) through port 22. The SSH commands activate Python scripts that are programmed to activate and engage with IoT sensors installed on an RPIB. The IoT sample network used in this research is built for DRA testing and does not provide custom or additional security measures for IoT sensors. IoT security is a research topic of its own. The flow of the interaction of **Multi-cloud ↔ IoT app ↔ IoT network** is secured by the current network on which the DRA is tested. The SSH commands enable a standard secure interaction between the IoT app and IoT sensors. The data collected from the IoT sensors has been stored on the MongoDB cloud (mLab), which provides users with secure data storage.

- **IoT Data Management**

The complexity of IoT introduces several challenges to software systems. IoT data management is one of the main concerns for organisations and practitioners (see Chapter 2). In this research, the DRA architecture does not provide a solution for IoT data management.

- **DevOps–Agile Integration**

The integration of DevOps–agile methods into a combined architecture or framework may prove to be effective may vertically augment software engineering. The transformation to DevOps–agile, integrated architecture is not handled in this thesis. However, it can be considered the next step towards future research and further improvement of the DRA.

- **IoT application Scalability**

The complexity of IoT increases the scalability challenge to the software system deployed to multi-cloud and the cloud database (Mongo DB). In this research, the DRA architecture does not provide a solution to the growing IoT data and a large amount of IoT interactions.

6.4. SUMMARY

This research was conducted between 2016 and 2019. Firstly, the background study and related work analysis conducted in Chapter 1 revealed that DevOps, cloud/multi-cloud and IoT appear to be connected. Further investigation into the related work suggested that the multi-cloud warranted further research to determine its relationship with DevOps and IoT. The analysis led to the convergence of ideas (see Chapter 1).

Further analysis of these ideas (see Table 1.1) revealed the research problem of the thesis. This helped in determining the research question (RQ): How can IoT applications be deployed to the multi-cloud using the DevOps approach? The study dimensions explained in Figure 1.5 showed the complexity of the research question. Thus, the research question was further subdivided into two questions (see Table 1.2):

1. RQ_1: What is known about DevOps?
2. RQ_2: How can IoT applications be deployed to the cloud (single and heterogeneous) using the DevOps approach?

Second, the research aimed to understand DevOps and its effect on software development. For this reason, an SLR was conducted in Chapter 2. The SLR was founded on the guidelines adopted by Kitchenham and Charters (2007). It was used to systematically review the DevOps concepts, practices, tools, challenges and existing solutions. The SLR examined and discussed the topics of DevOps, cloud computing, multi-cloud and IoT. The results of the adopted SLR helped to answer the first sub-question (RQ_1). They provided rich information about DevOps (concepts, practices, tools and adoption benefits and challenges to cloud and IoT). Part of the SLR results that provided rich information about the DevOps approach was refined and published in the international PACIS2017 (proceedings 96) conference in Malaysia. The SLR data analysis also helped to reveal the research gaps of the thesis, which were addressed in Chapters 3–5, consequently answering the second sub-question (RQ_2).

Third, this research adopted a DSR process based on the guidelines derived from Gregor and Hevner (2013) and Peffers et al. (2007). The DSR process steps are mapped in Table 3.1 and Figure 3.1. The thesis DSR uses the background study and related work analysis (see Chapter 1) and SLR data analysis and results (see Chapter 2) as initial data. The DSR aims to provide a verifiable contribution through the development and evaluation of an artefact. The DSR follows six steps (see Figure 3.2): problem identification, analysis, design, development, evaluation and outcome. The DSR process produced several artefacts, most notably the new DRA framework, which is the main contribution of the thesis.

Fourth, this research was carried out to develop a new framework called the DRA framework for IoT by using a constructive and iterative DSR (see Chapter 3). The framework aims to assist in the adoption of the DevOps approach for software development. The DRA is tailored to support IoT application deployment to the cloud. The framework has three main components: framework characteristics, framework architecture and framework composition. The DRA framework has two versions (instances discussed in this thesis): DRAv1.0 (deployment to a single cloud) and DRAv2.0 (deployment to the multi-cloud). Chapter 4 also included an implementation set that could be used to help with the application and evaluation of the framework in organisational contexts. The implementation set (see Chapter 4) was used to create a case study template, which was used in the industry case study and the research lab case study (see Chapter 5) to evaluate the DRA.

Fifth, an empirical evaluation was conducted to determine the relationship between the DRA elements and the evaluation criteria (see Tables 5.1 and 5.4). The empirical evaluation of the DRA has four iterations: 1) industry case study, 2) research case study, 3) teaching case study, and 4) industry field survey. The case studies used a case study template (see Appendix G) to provide instructive guidelines for DRA implementation and evaluation in organisations' contexts. The industry case study obtained feedback from a DE expert at CPF. The feedback was organised in Table 5.5 and reported in Table 5.6 and acknowledged that the DRA is useful, reusable and covers the organisation's needs. The feedback provided by the lab leader at the research lab (DigiSAS Lab) was organised in Table 5.7 and reported in Table 5.8. The analysis report conceded that the DRA offers new and useful knowledge that may cover what is needed to consider the DRA a general template that could be instantiated for IT project situations with various technology stacks. The teaching case study was conducted in two iterations: SEP case study and INP case study. SEP and INP are subjects offered by UTS FEIT. The SFS results for SEP produced an AAF = 88.85% (see Table 5.9) and the SFS results for INP produced an AAF = 100% (see Tables 5.11 and 5.12). These results indicated that the DRA is useful for teaching. The qualitative feedback in Tables 5.10 and 5.14 indicated that the students were satisfied with the subject content quality.

The industry field survey used in this research (see Appendix D) was anonymous. The survey was offered online via LinkedIn to industry experts from organisations located locally and

internationally (see Table 5.17). The survey evaluation process had two phases: quantitative evaluation process and qualitative evaluation process. The quantitative evaluation process collected rating data from the individual DRA models questionnaires (see Appendix D). The collected ratings were mapped as numerical data into results (see Tables 5.24–5.37. The RT and CT tables produced key statistical values (AAP, AAF and Chi^2 p-value). The AAP values in the results tables were above 90% overall, and the p-value for each test variable was less than the critical value $\alpha = 0.01$. The qualitative evaluation process collected feedback and ratings from the overall survey questionnaires and mapped the data in Tables 5.38 and 5.40.

The data in these tables were analysed to determine the relationship between the feedback and the evaluation criteria (see Table 3.2). Table 5.39 calculated an AAP = 93.09% and p-value < 0.01. This result indicated that the participants considered the DRA useful for the industry, teaching and research contexts. The analysis of the empirical evaluation combined data using QIM (see Table 5.44) and QEM (5.46) showed that the participants agree that the DRA meets the evaluation criteria positively (see Table 3.2 and Table 3.4) at a probability of QI = 86.47%. The QEM results (see Table 5.45) provided further insights from the participants' perspectives about the DRA architectural models relationships with the evaluation criteria. Chapter 5 also discussed future research based on suggested improvements to the DRA framework based on the feedback collected in the empirical evaluation. Chapter 6 presented the research journey, research output and key contributions. Chapter 6 also discussed the framework's limitations and research contributions.

Finally, and based on the empirical evaluation results, it appears that the proposed DRA framework is a practical, applicable comprehensive solution for IoT application deployment to the multi-cloud. Practitioners and researchers may benefit from the framework and the empirical evaluation results to understand the usefulness and applicability of the DevOps approach on the multi-cloud platform for the automated deployment of IoT applications.

Conclusion

This thesis presented a DRA framework that appears to provide a practical solution for deploying IoT applications to the cloud/multi-cloud. The DRA was developed iteratively using a well-known DSR. The DRA framework is intended for use by DevOps teams (managers, consultants, engineers, developers) as a practical guide for using DevOps, cloud/multi-cloud and IoT contexts in software development environments. The DRA offers a comprehensive architectural design model that could be implemented and applied in any organisational context using numerous technology stacks. The DRA operational model instance may vary from each implementation; however, the architecture is generic and fixed. The architecture models of the DRA are effectively used as a blueprint. This framework should not be taken as an imposed heavy-handed step-by-step process; rather, it should be understood as a comprehensive guide to ensure that the important points are not ignored. DevOps teams may use some or all of the framework's characteristics and components to generate their operational models that are suitable for the development environment of their organisation. This framework will be further extended based on future learning, research and experience.

Declarations

This section contains information about the thesis author. It includes a brief bio about the thesis author.

A. Author Information

Author: Georges Bou Ghantous

Thesis: Doctor of Philosophy (C02029)

Thesis Title: A DevOps Reference Architecture for Multi-Cloud IoT-applications Deployments.

Email: Georges.BouGhantous-1@uts.edu.au and Georges.BouGhantous@student.uts.edu.au
UTS Profile: https://www.uts.edu.au/staff/georges.boughantous-1
Linked In Profile: https://www.linkedin.com/in/georges-bou-ghantous
Development Website: https://maven-app-heroku.herokuapp.com

B. Availability of Data and Materials

The data collected during the evaluation process of the DRA framework have been securely stored on 'CloudStor' by AARNet (Link) the UTS-recommended cloud storage to researchers and academics. The collected data consists of:

- Industry survey data is stored on CloudStor. The collected data from the survey is completely anonymous. No information about participants was collected.
- Industry case study data and feedback. No information about organisations or participants was recorded.
- Research case study data and feedback. No information about organisations or participants was recorded.
- Teaching case study data is stored on CloudStor. No information about organisations or participants was recorded.

Please Note: Only the author (Mr Georges Bou Ghantous) and the principal supervisor (Dr Asif Gill) have access to the original thesis data is stored on CloudStor. The anonymous results data are available for public access (see Appendix E).

Bibliography

Aberdeen, T. 2013, 'Yin, R. K. 2009. Case study research: design and methods (4th ed.). Thousand Oaks, CA: Sage', *Canadian Journal of Action Research*, vol. 14, no. 1, pp. 69–71.

Adda, M. & Saad, R. 2014, 'A data sharing strategy and a DSL for service discovery, selection and consumption for the IoT', *Procedia Computer Science*, vol. 37, pp. 92–100. (S33).

Ahmadighohandizi, F. & Systä, K. 2015, 'ICDO: integrated cloud-based development tool for DevOps', *Published in SPLST 2015*, pp. 76–90. (S21).

Alam, K.M., Sopena, A. & El Saddik, A. 2015, 'Design and development of a cloud based cyber-physical architecture for the Internet-of-Things', *IEEE International Symposium on Multimedia 2015*, IEEE, Piscataway, NJ, pp. 459–464. (S60).

Alkhalil, A. & Ramadan, R.A. 2017, 'IoT data provenance implementation challenges', *Procedia Computer Science*, vol. 109, pp. 1134–1139. (S53).

AlOtaibi, M., Lo'ai, A.T. & Jararweh, Y. 2016, 'Integrated sensors system based on IoT and mobile cloud computing', *IEEE/ACS 13th International Conference of Computer Systems and Applications 2016*, IEEE, Piscataway, NJ, pp. 1–5. (S66).

Alowaidi, M., Rahman, M.A., Hassanain, E. & El Saddik, A. 2017, 'Demo abstract: a semantic notification approach for IoT-based sensory data', *IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation 2017*, IEEE, Piscataway, NJ, pp. 289–290. (S42).

Alzoubi, Y.I., Gill, A.Q. & Al-Ani, A. 2015, 'Distributed agile development communication: an agile architecture driven framework', *Journal of Software*, vol. 10, no. 6, pp. 681–694. (S113).

Alzoubi, Y.I., Gill, A.Q. & Moulton, B. 2018, 'A measurement model to analyze the effect of agile enterprise architecture on geographically distributed agile development', *Journal of Software Engineering Research and Development*, vol. 6, no. 1, p. 4. (S127).

Analyti, A., Theodorakis, M., Spyratos, N. & Constantopoulos, P. 2007, 'Contextualization as an independent abstraction mechanism for conceptual modeling', *Information Systems*, vol. 32, no. 1, pp. 24–60.

Anatolijs Zabasta, Nadezda Kunicina, Kaspars Kondratjevs, and Leonids Ribickis. 2020. 'Adaptive Workflow of Service Oriented IoT Architectures for Small and Distributed Automation Systems'. *In Proceedings of the 3rd International Conference on Applications of*

Intelligent Systems (APPIS 2020). Association for Computing Machinery, New York, NY, USA, Article 5, 1–6. DOI:https://doi-org.ezproxy.lib.uts.edu.au/10.1145/3378184.3378189

Artač, M., Borovšak, T., Di Nitto, E., Guerriero, M. & Tamburri, D.A. 2016, 'Model-driven continuous deployment for quality DevOps', *Proceedings of the 2nd International Workshop on Quality-Aware DevOps*, ACM, New York, NY, pp. 40–41. (S106).

Atif, Y., Ding, J. & Jeusfeld, M.A. 2016, 'Internet of Things approach to cloud-based smart car parking', *Procedia Computer Science*, vol. 98, pp. 193–198. (S82).

Avram, M.G. 2014, 'Advantages and challenges of adopting cloud computing from an enterprise perspective', *Procedia Technology*, vol. 12, pp. 529–534. (S86).

Babovic, Z.B., Protic, J. & Milutinovic, V. 2016, 'Web performance evaluation for Internet of Things applications', *IEEE Access*, vol. 4, pp. 6974–6992. (S75).

Bai, X., Li, M., Pei, D., Li, S. & Ye, D. 2018, 'Continuous delivery of personalized assessment and feedback in agile software engineering projects', *IEEE/ACM 40th International Conference on Software Engineering: Software Engineering Education and Training 2018*, IEEE, Piscataway, NJ, pp. 58–67. (S107).

Bass, L., Holz, R., Rimba, P., Tran, A.B. & Zhu, L. 2015, 'Securing a deployment pipeline', *Proceedings of the Third International Workshop on Release Engineering*, IEEE, Piscataway, NJ, pp. 4–7. (S17).

Berger, S., Häckel, B. & Häfner, L. 2019, 'Organizing self-organizing systems: a terminology, taxonomy, and reference model for entities in cyber-physical production systems', *Information Systems Frontiers*, pp. 1–24.

Botta, A., De Donato, W., Persico, V. & Pescapé, A. 2016, 'Integration of cloud computing and Internet of Things: a survey', *Future Generation Computer Systems*, vol. 56, pp. 684–700. (S89).

Bradley, D., Russell, D., Ferguson, I., Isaacs, J., MacLeod, A. & White, R. 2015, 'The Internet of Things—the future or the end of mechatronics', *Mechatronics*, vol. 27, pp. 57–74. (S38).

Carvalho, J.Á. 2012, 'Validation criteria for the outcomes of design research', a Pre-ECIS and AIS SIG Prag Workshop on IT Artefact Design & Workpractice Intervention, Barcelona, 10 June.

Cavalcante, E., Pereira, J., Alves, M.P., Maia, P., Moura, R., Batista, T., Delicato, F.C. & Pires, P.F. 2016, 'On the interplay of Internet of Things and cloud computing: a systematic mapping study', *Computer Communications*, vol. 89, pp. 17–33. (S83).

Chen, C.H., Lin, M.Y. & Guo, X.C. 2017, 'High-level modeling and synthesis of smart sensor networks for Industrial Internet of Things', *Computers & Electrical Engineering*, vol. 61, pp. 48–66. (S52).

Chen, H.M., Kazman, R. and Haziyev, S., 2016. 'Agile big data analytics for web-based systems: An architecture-centric approach'. *IEEE Transactions on Big Data*, 2(3), pp.234-248.

Chen, H.M., Kazman, R., Haziyev, S., Kropov, V. & Chtchourov, D. 2015, 'Architectural support for DevOps in a neo-metropolis BDaaS platform', *IEEE 34th Symposium on Reliable Distributed Systems Workshop 2015*, IEEE, Piscataway, NJ, pp. 25–30. (S3).

Chondamrongkul, N. & Temdee, P. 2013, 'Multi-cloud computing platform support with model-driven application runtime framework', *13th International Symposium on Communications and Information Technologies 2013*, IEEE, Piscataway, NJ, pp. 715–719. (S114).

Colavita, F. 2016, 'DevOps movement of enterprise agile breakdown silos creates collaboration, increase quality, and application speed', *Proceedings of 4th International Conference in Software Engineering for Defence Applications*, Springer, Cham, pp. 203–213. (S128).

Cukier, D. 2013, 'DevOps patterns to scale web applications using cloud services', *Proceedings of the 2013 Companion Publication for Conference on Systems, Programming, & Applications: Software for Humanity*, ACM, New York, NY, pp. 143–152. (S25).

Cysneiros, L.M., de Macedo-Soares, T. & do Prado Leite, J.C.S. 1999, 'Using ISO 9000 to elicit business rules', *Proceedings 4th IEEE International Software Engineering Standards Symposium and Forum*, IEEE, Piscataway, NJ, pp. 88–98.

D'Agostino, D., Galizia, A., Clematis, A., Mangini, M., Porro, I. & Quarati, A. 2013, 'A QoS-aware broker for hybrid clouds', *Computing*, vol. 95, no. 1, pp. 89–109. (S124).

Dasgupta, A., Gill, A. and Hussain, F., 2019. A Conceptual Framework for Data Governance in IoT-enabled Digital IS Ecosystems.

de AR Gonçalves, J.C., Santoro, F.M. & Baião, F.A. 2011, 'Collaborative narratives for business rule elicitation', *IEEE International Conference on Systems, Man, and Cybernetics 2011*, IEEE, Piscataway, NJ, pp. 1926–1931.

De Bayser, M., Azevedo, L.G. & Cerqueira, R. 2015, 'ResearchOps: the case for DevOps in scientific applications', *IFIP/IEEE International Symposium on Integrated Network Management 2015*, IEEE, Piscataway, NJ, pp. 1398–1404. (S14).

Dev Agrawal, Rahul Bhagwat, Rajdeep Bandopadhyay, Vineela Kunapareddi, Eric Burden, Shane Halse, Pamela Wisniewski, and Jess Kropczynski. 2020. 'Enhancing Smart Home Security using Co-Monitoring of IoT Devices'. *In Companion of the 2020 ACM International Conference on Supporting Group Work (GROUP '20)*. Association for Computing Machinery, New York, NY, USA, 99–102. DOI:https://doi-org.ezproxy.lib.uts.edu.au/10.1145/3323994.3369883

Di Martino, B. & Esposito, A. 2016, 'Semantic techniques for multi-cloud applications portability and interoperability', *Procedia Computer Science*, vol. 97, pp. 104–113. (S109).

Di Nitto, E., Jamshidi, P., Guerriero, M., Spais, I. & Tamburri, D.A. 2016, 'A software architecture framework for quality-aware DevOps', *Proceedings of the 2nd International Workshop on Quality-Aware DevOps*, ACM, New York, NY, pp. 12–17. (S2).

Diel, E., Marczak, S. & Cruzes, D.S. 2016, 'Communication challenges and strategies in distributed DevOps', *IEEE 11th International Conference on Global Software Engineering 2016*, IEEE, Piscataway, NJ, pp. 24–28. (S12).

Domaschka, J., Griesinger, F., Baur, D. & Rossini, A. 2015, 'Beyond mere application structure thoughts on the future of cloud orchestration tools', *Procedia Computer Science*, vol. 68, pp. 151–162. (S29).

Douzis, K., Sotiriadis, S., Petrakis, E.G. & Amza, C. 2018, 'Modular and generic IoT management on the cloud', *Future Generation Computer Systems*, vol. 78, pp. 369–378. (S56).

Dybå, T. & Dingsøyr, T. 2008, 'Empirical studies of agile software development: a systematic review', *Information and Software Technology*, vol. 50, no. 9–10, pp. 833–859.

Erich, F., Amrit, C. & Daneva, M. 2014, *Report: DevOps literature review*, Technical Report, University of Twente, The Netherlands (S23).

Fan, T.R., Feng, G.A.O., Zhang, X. & Xu, W.A.N.G. 2012, 'Integration of IoT and DRAGON-lab in cloud environment', *Journal of China Universities of Posts and Telecommunications*, vol. 19, no. 2, pp. 87–91. (S90).

Farahzadi, A., Shams, P., Rezazadeh, J. & Farahbakhsh, R. 2018, 'Middleware technologies for Cloud of Things: a survey', *Digital Communications and Networks*, vol. 4, no. 3, pp. 176–188. (S91).

Ferry, N., Chauvel, F., Song, H., Rossini, A., Lushpenko, M. & Solberg, A. 2018, 'CloudMF: model-driven management of multi-cloud applications', *ACM Transactions on Internet Technology*, vol. 18, no. 2, p. 16. (S97).

Ferry, N., Chauvel, F., Song, H. & Solberg, A. 2015, 'Continuous deployment of multi-cloud systems', *Proceedings of the 1st International Workshop on Quality-Aware DevOps*, ACM, New York, NY, pp. 27–28. (S101).

Ferry, N., Rossini, A., Chauvel, F., Morin, B. & Solberg, A. 2013, 'Towards model-driven provisioning, deployment, monitoring, and adaptation of multi-cloud systems', *IEEE Sixth International Conference on Cloud Computing 2013*, IEEE, Piscataway, NJ, pp. 887–894. (S94).

Fowler, M. and Highsmith, J., 2001. The agile manifesto. *Software Development*, *9*(8), pp.28-35.

Ghantous, G.B. & Gill, A. 2017, 'DevOps: concepts, practices, tools, benefits and challenges', *Pacific Asia Conference on Information Systems 2017*, AISeL https://aisel.aisnet.org/pacis2017/. (S41).

Ghantous, G.B. & Gill, A.Q. 2018, 'DevOps reference architecture for multi-cloud IOT applications', *IEEE 20th Conference on Business Informatics 2018*, IEEE, Piscataway, NJ, pp. 158–167. (S104).

Ghantous, G.B. & Gill, A.Q. 2019, 'An agile-DevOps reference architecture for teaching enterprise agile', *International Journal of Learning, Teaching and Educational Research*, vol. 18, no. 7, pp. 128–144.

Gill, A.Q., Chew, E.K., Kricker, D. and Bird, G., 2016, August. Adaptive enterprise resilience management: Adaptive action design research in financial services case study. In 2016 IEEE 18th Conference on Business Informatics (CBI) (Vol. 1, pp. 113-122). IEEE.

Gill, A.Q. and Chew, E., 2019. Configuration information system architecture: Insights from applied action design research. Information & Management, 56(4), pp.507-525.

Gomes, M., da Rosa Righi, R. & da Costa, C.A. 2014, 'Internet of Things scalability: analyzing the bottlenecks and proposing alternatives', *6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops 2014*, IEEE, Piscataway, NJ, pp. 269–276. (S120).

Gómez, J.E., Marcillo, F.R., Triana, F.L., Gallo, V.T., Oviedo, B.W. & Hernández, V.L. 2017, 'IoT for environmental variables in urban areas', *Procedia Computer Science*, vol. 109, pp. 67–74. (S54)

Gregor, S. & Hevner, A.R. 2013, 'Positioning and presenting design science research for maximum impact', *MIS Quarterly*, vol. 37, no. 2, pp. 337–355.

Gubbi, J., Buyya, R., Marusic, S. & Palaniswami, M. 2013, 'Internet of Things (IoT): a vision, architectural elements, and future directions', *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660. (S37).

Guechi, F.A. & Maamri, R. 2017, 'Secure self-destruction of shared data in multi-cloud IoT', *IEEE 5th International Conference on Future Internet of Things and Cloud 2017*, IEEE, Piscataway, NJ, pp. 161–168. (S102).

Gutiérrez-Madroñal, L., Medina-Bulo, I. & Domínguez-Jiménez, J.J. 2018, 'IoT–TEG: test event generator system', *Journal of Systems and Software*, vol. 137, pp. 784–803. (S55).

Hanappi, O., Hummer, W. & Dustdar, S. 2016, 'Asserting reliable convergence for configuration management scripts', *ACM SIGPLAN Notices*, vol. 51, no. 10, pp. 328–343. (S16).

Hefnawy, A., Bouras, A. & Cherifi, C. 2016, 'IoT for smart city services: lifecycle approach', *Proceedings of the International Conference on Internet of Things and Cloud Computing*, ACM, New York, NY, p. 55. (S45).

Hoeren, T. & Pinelli, S. 2018, 'Agile programming—introduction and current legal challenges', *Computer Law & Security Review*, vol. 34, no. 5, pp. 1131–1138.

Hsieh, W.K., Hsieh, W.H., Chen, J.L. & Lin, C.Y. 2016, 'Self-configuration and smart binding control on IOT applications', *18th International Conference on Advanced Communication Technology 2016*, IEEE, Piscataway, NJ, pp. 80–85. (S73).

Hu, P., Dhelim, S., Ning, H. & Qiu, T. 2017, 'Survey on fog computing: architecture, key technologies, applications and open issues', *Journal of Network and Computer Applications*, vol. 98, pp. 27–42. (S103).

Hyndman, R.J. 2008, *Quantitative business research methods*, Department of Econometrics and Business Statistics, Monash University, Melbourne, Vic.

Iqbal, A., Ullah, F., Anwar, K.H. & Sup, K. 2010, 'Interoperable Internet-of-Things platform for Smart', *Networks*, vol. 54, no. 15, pp. 2787–2805. (S99).

Islam, S., Mouratidis, H. & Jürjens, J. 2011, 'A framework to support alignment of secure software engineering with legal regulations', *Software & Systems Modeling*, vol. 10, no. 3, pp. 369–394.

Jabbari, R., bin Ali, N., Petersen, K. & Tanveer, B. 2016, 'What is DevOps? A systematic mapping study on definitions and practices', *Proceedings of the Scientific Workshop Proceedings of XP2016*, ACM, New York, NY, p. 12. (S20).

Jamshidi, P., Pahl, C., Chinenyeze, S. & Liu, X. 2015, 'Cloud migration patterns: a multi-cloud service architecture perspective', *Service-Oriented Computing-ICSOC 2014 Workshops*, Springer, Cham, pp. 6–19. (S98).

Jedlitschka, A. & Pfahl, D. 2005, 'Reporting guidelines for controlled experiments in software engineering', *International Symposium on Empirical Software Engineering 2005*, IEEE, Piscataway, NJ, pp. 95–104.

Jha, P. & Khan, R. 2018, 'A review paper on DevOps: beginning and more to know. *International Journal of Computer Applications*, vol. 180, no. 48, pp. 16–20. (S1).

John, W., Meirosu, C., Pechenot, B., Sköldström, P., Kreuger, P. & Steinert, R. 2015, 'Scalable software defined monitoring for service provider DevOps', *Fourth European Workshop on Software Defined Networks 2015*, IEEE, Piscataway, NJ, pp. 61–66. (S15).

Jula, A., Sundararajan, E. & Othman, Z. 2014, 'Cloud computing service composition: a systematic literature review', *Expert Systems with Applications*, vol. 41, no. 8, pp. 3809–3824. (S85).

Kang, B. & Choo, H. 2018, 'An experimental study of a reliable IoT gateway', *ICT Express*, vol. 4, no. 3, pp. 130–133. (S48).

Khakimov, A., Muthanna, A., Kirichek, R., Koucheryavy, A. & Muthanna, M.S.A. 2017, 'Investigation of methods for remote control IoT-devices based on cloud platforms and different interaction protocols', *IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering 2017*, IEEE, Piscataway, NJ, pp. 160–163. (S68).

Kishore Ramakrishnan, A., Preuveneers, D. & Berbers, Y. 2014, 'Enabling self-learning in dynamic and open IoT environments', *Procedia Computer Science*, vol. 32, pp. 207–214. (S35).

Kitchenham, B. & Charters, S. 2007, *Guidelines for performing systematic literature reviews in software engineering*, EBSE Technical Report, Keele University and University of Durham.

Klein, H.K. & Myers, M.D. 1999, 'A set of principles for conducting and evaluating interpretive field studies in information systems', *MIS Quarterly*, vol. 23, no. 1, pp. 67–94.

Kodeswaran, P.A., Kokku, R., Sen, S. & Srivatsa, M. 2016, 'Idea: a system for efficient failure management in smart IoT environments', *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, ACM, New York, NY, pp. 43–56. (S43).

Kolios, P., Panayiotou, C., Ellinas, G. & Polycarpou, M. 2016, 'Data-driven event triggering for IoT applications', *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1146–1158. (S78).

Kritikos, K. & Plexousakis, D. 2015, 'Multi-cloud application design through cloud service composition', *IEEE 8th International Conference on Cloud Computing 2015*, IEEE, Piscataway, NJ, pp. 686–693. (S92).

Kumari, S., Li, X., Wu, F., Das, A.K., Choo, K.K.R. & Shen, J. 2017, 'Design of a provably secure biometrics-based multi-cloud-server authentication scheme', *Future Generation Computer Systems*, vol. 68, pp. 320–330. (S111).

Lee, I. & Lee, K. 2015, 'The Internet of Things (IoT): applications, investments, and challenges for enterprises', *Business Horizons*, vol. 58, no. 4, pp. 431–440. (S36).

Lee, K. & Hughes, D. 2010, 'System architecture directions for tangible cloud computing', *First ACIS International Symposium on Cryptography, and Network Security, Data Mining and Knowledge Discovery, E-Commerce and Its Applications, and Embedded Systems 2010*, IEEE, Piscataway, NJ, pp. 258–262. (S80).

Leite, J., Batista, T. & Oquendo, F. 2017, 'Architecting IoT applications with SysADL', *IEEE International Conference on Software Architecture Workshops 2017*, IEEE, Piscataway, NJ, pp. 92–99. (S77).

Li, W., Santos, I., Delicato, F.C., Pires, P.F., Pirmez, L., Wei, W., Song, H., Zomaya, A. & Khan, S. 2017, 'System modelling and performance evaluation of a three-tier Cloud of Things', *Future Generation Computer Systems*, vol. 70, pp. 104–125. (S84).

Litchfield, A. and Althouse, J., 2014. A systematic review of cloud computing, big data and databases on the cloud. *AMCIS (2014)*

Lin, Y.B., Lin, Y.W., Chih, C.Y., Li, T.Y., Tai, C.C., Wang, Y.C., Lin, F.J., Kuo, H.C., Huang, C.C. & Hsu, S.C. 2015, 'EasyConnect: a management system for IoT devices and its applications for interactive design and art', *IEEE Internet of Things Journal*, vol. 2, no. 6, pp. 551–561. (S62).

Luo, B. & Sun, Z. 2015, 'Enabling end-to-end communication between wireless sensor networks and the internet based on 6LoWPAN', *Chinese Journal of Electronics*, vol. 24, no. 3, pp. 633–638. (S64).

Lwakatare, L.E., Kuvaja, P. & Oivo, M. 2016a, 'An exploratory study of DevOps extending the dimensions of DevOps with practices', *ICSEA 2016: The Eleventh International Conference on Software Engineering Advances*, IARIA, 2016, pp. 91–99. (S27).

Lwakatare, L.E., Kuvaja, P. & Oivo, M. 2016b, 'Relationship of DevOps to agile, lean and continuous deployment', *International Conference on Product-Focused Software Process Improvement*, Springer, Cham, pp. 399–415. (S126).

Massonet, P., Deru, L., Achour, A., Dupont, S., Levin, A. & Villari, M. 2017, 'End-to-end security architecture for federated cloud and IoT networks', *IEEE International Conference on Smart Computing 2017*, IEEE, Piscataway, NJ, pp. 1–6. (S65).

Mathur, A., Newe, T., Elgenaidi, W., Rao, M., Dooly, G. & Toal, D. 2017, 'A secure end-to-end IoT solution', *Sensors and Actuators A: Physical*, vol. 263, pp. 291–299. (S49).

McCarthy, M.A., Herger, L.M., Khan, S.M. & Belgodere, B.M. 2015, 'Composable DevOps: automated ontology based DevOps maturity analysis', *IEEE International Conference on Services Computing 2015*, IEEE, Piscataway, NJ, pp. 600–607. (S6).

Minh Nguyen, Saptarshi Debroy, Prasad Calyam, Zhen Lyu, and Trupti Joshi. 2020. 'Security-aware Resource Brokering for Bioinformatics Workflows across Federated Multi-cloud Infrastructures'. *In Proceedings of the 21st International Conference on Distributed Computing and Networking (ICDCN 2020)*. Association for Computing Machinery, New York, NY, USA, Article 26, 1–10. DOI:https://doi-org.ezproxy.lib.uts.edu.au/10.1145/3369740.3369791

Mohamed, S. 2016, 'DevOps maturity calculator DOMC-value oriented approach', *International Journal of Engineering Science and Research*, vol. 2, no. 2, pp. 25–35. (S26).

Moldovan, D., Copil, G., Truong, H.L. & Dustdar, S. 2014, 'On analyzing elasticity relationships of cloud services', *IEEE 6th International Conference on Cloud Computing Technology and Science 2014*, IEEE, Piscataway, NJ, pp. 447–454. (S116).

Mongan, W.M., Rasheed, I., Ved, K., Vora, S., Dandekar, K., Dion, G., Kurzweg, T. & Fontecchio, A. 2017, 'On the use of radio frequency identification for continuous biomedical monitoring', *IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation 2017*, IEEE, Piscataway, NJ, pp. 197–202. (S46).

Moore, J., Kortuem, G., Smith, A., Chowdhury, N., Cavero, J. & Gooch, D. 2016, 'DevOps for the urban IoT', *Proceedings of the Second International Conference on IoT in Urban Space*, ACM, New York, NY, pp. 78–81. (S7).

Morin, E., Maman, M., Guizzetti, R. & Duda, A. 2017, 'Comparison of the device lifetime in wireless networks for the Internet of Things', *IEEE Access*, vol. 5, pp. 7097–7114. (S59).

Muhendra, R., Rinaldi, A. & Budiman, M. 2017, 'Development of WiFi mesh infrastructure for Internet of Things applications', *Procedia Engineering*, vol. 170, pp. 332–337. (S51).

Muhammad H. Hilman, Maria A. Rodriguez, and Rajkumar Buyya. 2020. 'Multiple Workflows Scheduling in Multi-tenant Distributed Systems: A Taxonomy and Future Directions.' *ACM Comput. Surv. 53, 1, Article 10 (February 2020), 39 pages*. DOI:https://doi-org.ezproxy.lib.uts.edu.au/10.1145/3368036

Munteanu, V.I., Şandru, C. & Petcu, D. 2014, 'Multi-cloud resource management: cloud service interfacing', *Journal of Cloud Computing*, vol. 3, no. 1, p. 3. (S125).

Newmarch, J. 2016, 'Low power wireless: 6LoWPAN, IEEE802. 15.4 and the Raspberry Pi', *Linux Journal*, vol. 2016, no. 271, p. 1. (S76).

Ngu, A.H., Gutierrez, M., Metsis, V., Nepal, S. & Sheng, Q.Z. 2016, 'IoT middleware: a survey on issues and enabling technologies', *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 1–20. (S69).

Nguyen, V. & Gendreau, A. 2014, 'A vision of a future IoT architecture supporting messaging, storage, and computation', *International Journal of Future Computer and Communication*, vol. 3, no. 6, p. 405. (S32).

Nickerson, R.C., Varshney, U. & Muntermann, J. 2013, 'A method for taxonomy development and its application in information systems', *European Journal of Information Systems*, vol. 22, no. 3, pp. 336–359.

Olivier, F., Carlos, G. & Florent, N. 2015, 'New security architecture for IoT network', *Procedia Computer Science*, vol. 52, pp. 1028–1033. (S39).

Oscar Novo and Mario Di Francesco. 2020. 'Semantic Interoperability in the IoT: Extending the Web of Things Architecture'. *ACM Trans. Internet Things 1, 1, Article 6 (March 2020), 25 pages*. DOI:https://doi-org.ezproxy.lib.uts.edu.au/10.1145/3375838

Papaioannou, A., Metallidis, D. & Magoutis, K. 2015, 'Cross-layer management of distributed applications on multi-clouds', *IFIP/IEEE International Symposium on Integrated Network Management 2015*, IEEE, Piscataway, NJ, pp. 552–558. (S118).

Peffers, K., Tuunanen, T., Rothenberger, M.A. & Chatterjee, S. 2007, 'A design science research methodology for information systems research', *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77.

Perera, P., Silva, R. & Perera, I. 2017, 'Improve software quality through practicing DevOps', *Seventeenth International Conference on Advances in ICT for Emerging Regions 2017*, IEEE, Piscataway, NJ, pp. 1–6. (S121).

Porambage, P., Schmitt, C., Kumar, P., Gurtov, A. & Ylianttila, M. 2014, 'Two-phase authentication protocol for wireless sensor networks in distributed IoT applications',

IEEE Wireless Communications and Networking Conference 2014, IEEE, Piscataway, NJ, pp. 2728–2733. (S74).

Prat, N., Comyn-Wattiau, I. & Akoka, J. 2014, 'Artifact evaluation in information systems design-science research—a holistic view', *Pacific Asia Conference on Information Systems 2017*, AIS eLibrary https://aisel.aisnet.org/pacis2014/23/, p. 23.

Qumer, A., Henderson-Sellers, B. & McBride, T. 2007, 'Agile adoption and improvement model', *Proceedings of European and Mediterranean Conference on Information Systems.* (S31).

Rahman, A.A.U. & Williams, L. 2016, 'Software security in DevOps: synthesizing practitioners' perceptions and practices', *IEEE/ACM International Workshop on Continuous Software Evolution and Delivery 2016*, IEEE, Piscataway, NJ, pp. 70–76. (S18).

Rajkumar, M., Pole, A.K., Adige, V.S. & Mahanta, P. 2016, 'DevOps culture and its impact on cloud delivery and software development', *International Conference on Advances in Computing, Communication, & Automation 2016*, IEEE, Piscataway, NJ, pp. 1–6. (S13).

Rao, S. & Shorey, R. 2017, 'Efficient device-to-device association and data aggregation in industrial IoT systems', *9th International Conference on Communication Systems and Networks 2017*, IEEE, Piscataway, NJ, pp. 314–321. (S63).

Rautmare, S. & Bhalerao, D.M. 2016, 'MySQL and NoSQL database comparison for IoT application', *IEEE International Conference on Advances in Computer Applications 2016*, IEEE, Piscataway, NJ, pp. 235–238. (S79).

Ray, P.P. 2016, 'A survey of IoT cloud platforms', *Future Computing and Informatics Journal*, vol. 1, no. 1–2, pp. 35–46. (S88).

Runeson, P. & Höst, M. 2009, 'Guidelines for conducting and reporting case study research in software engineering', *Empirical Software Engineering*, vol. 14, no. 2, p. 131.

Russell, L., Goubran, R. & Kwamena, F. 2015, 'Personalization using sensors for preliminary human detection in an IoT environment', *International Conference on Distributed Computing in Sensor Systems 2015*, IEEE, Piscataway, NJ, pp. 236–241. (S72).

Samarawickrama, S.S. & Perera, I. 2017, 'Continuous scrum: a framework to enhance scrum with DevOps', *Seventeenth International Conference on Advances in ICT for Emerging Regions 2017*, IEEE, Piscataway, NJ, pp. 1–7. (S123).

Schaefer, A., Reichenbach, M. & Fey, D. 2013, 'Continuous integration and automation for DevOps', *IAENG Transactions on Engineering Technologies*, Springer, Dordrecht, pp. 345–358. (S30).

Sen, S. 2016, 'Context-aware energy-efficient communication for IoT sensor nodes', *53nd ACM/EDAC/IEEE Design Automation Conference 2016*, IEEE, Piscataway, NJ, pp. 1–6. (S44)

Shah, J. & Mishra, B. 2016, 'Customized IoT enabled wireless sensing and monitoring platform for smart buildings', *Procedia Technology*, vol. 23, pp. 256–263. (S50).

Shahzad, F. 2014, 'State-of-the-art survey on cloud computing security challenges, approaches and solutions', *Procedia Computer Science*, vol. 37, pp. 357–362. (S87).

Sharp, J. & Babb, J. 2018, 'Is information systems late to the party? The current state of DevOps research in the Association for Information Systems eLibrary', Twenty-fourth Americas Conference on Information Systems, New Orleans, 2018, DevOps Research in the AISeL. (S22).

Shekhar, S. & Gokhale, A. 2017, 'Enabling IoT applications via dynamic cloud-edge resource management', *IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation 2017*, IEEE, Piscataway, NJ, pp. 331–332. (S100).

Sheng, Z., Wang, H., Yin, C., Hu, X., Yang, S. & Leung, V.C. 2015, 'Lightweight management of resource-constrained sensor devices in Internet of Things', *IEEE Internet of Things Journal*, vol. 2, no. 5, pp. 402–411. (S70).

Singh, N.K., Thakur, S., Chaurasiya, H. & Nagdev, H. 2015, 'Automated provisioning of application in IAAS cloud using Ansible configuration management', *1st International Conference on Next Generation Computing Technologies 2015*, IEEE, Piscataway, NJ, pp. 81–85. (S5).

Sjøberg, D.I., Hannay, J.E., Hansen, O., Kampenes, V.B., Karahasanovic, A., Liborg, N.K. & Rekdal, A.C. 2005, 'A survey of controlled experiments in software engineering', *IEEE Transactions on Software Engineering*, vol. 31, no. 9, pp. 733–753.

Slawik, M., Blanchet, C., Demchenko, Y., Turkmen, F., Ilyushkin, A., de Laat, C. & Loomis, C. 2017, 'CYCLONE: the multi-cloud middleware stack for application deployment and management', *IEEE International Conference on Cloud Computing Technology and Science 2017*, IEEE, Piscataway, NJ, pp. 347–352. (S96).

Snyder, B. & Curtis, B. 2017, 'Using analytics to guide improvement during an agile–DevOps transformation', *IEEE Software*, vol. 35, no. 1, pp. 78–83. (S122).

Soni, M. 2015, 'End to end automation on cloud with build pipeline: the case for DevOps in insurance industry, continuous integration, continuous testing, and continuous delivery', *IEEE International Conference on Cloud Computing in Emerging Markets 2015*, IEEE, Piscataway, NJ, pp. 85–89. (S8).

Srirama, S.N. 2017, 'Mobile web and cloud services enabling Internet of Things', *CSI Transactions on ICT*, vol. 5, no. 1, pp. 109–117. (S81).

Srirama, S.N., Iurii, T. & Viil, J. 2016, 'Dynamic deployment and auto-scaling enterprise applications on the heterogeneous cloud', *IEEE 9th International Conference on Cloud Computing 2016*, IEEE, Piscataway, NJ, pp. 927–932. (S119).

Stergiou, C., Psannis, K.E., Kim, B.G. & Gupta, B. 2018, 'Secure integration of IoT and cloud computing', *Future Generation Computer Systems*, vol. 78, pp. 964–975. (S57).

Su, X., Zhang, H., Riekki, J., Keränen, A., Nurminen, J.K. & Du, L. 2014, 'Connecting IoT sensors to knowledge-based systems by transforming SenML to RDF', *Procedia Computer Science*, vol. 32, pp. 215–222. (S34).

Syed, M.H. & Fernandez, E.B. 2016, 'Cloud ecosystems support for Internet of Things and DevOps using patterns', *IEEE First International Conference on Internet-of-Things Design and Implementation 2016*, IEEE, Piscataway, NJ, pp. 301–304. (S11).

Tao, M., Zuo, J., Liu, Z., Castiglione, A. & Palmieri, F. 2018, 'Multi-layer cloud architectural model and ontology-based security service framework for IoT-based smart homes', *Future Generation Computer Systems*, vol. 78, pp. 1040–1051. (S108).

Theodorakis, M., Analyti, A., Constantopoulos, P. & Spyratos, N. 1999, 'Contextualization as an abstraction mechanism for conceptual modelling', *International Conference on Conceptual Modeling*, Springer, Berlin, pp. 475–490.

Tricomi, G., Panarello, A., Merlino, G., Longo, F., Bruneo, D. & Puliafito, A. 2017, 'Orchestrated multi-cloud application deployment in OpenStack with TOSCA', *IEEE International Conference on Smart Computing 2017*, IEEE, Piscataway, NJ, pp. 1–6. (S117).

Truong, H.L., Dustdar, S. & Leymann, F. 2016, 'Towards the realization of multi-dimensional elasticity for distributed cloud systems', *Procedia Computer Science*, vol. 97, pp. 14–23. (S110).

Ungurean, I., Gaitan, N.C. & Gaitan, V.G. 2014, 'An IoT architecture for things from industrial environment', *10th International Conference on Communications 2014*, IEEE, Piscataway, NJ, pp. 1–4. (S40).

Virmani, M. 2015, 'Understanding DevOps & bridging the gap from continuous integration to continuous delivery', *Fifth International Conference on the Innovative Computing Technology*, IEEE, Piscataway, NJ, pp. 78–82. (S10).

Wahaballa, A., Wahballa, O., Abdellatief, M., Xiong, H. & Qin, Z. 2015, 'Toward unified DevOps model', *6th IEEE International Conference on Software Engineering and Service Science 2015*, IEEE, Piscataway, NJ, pp. 211–214. (S9).

Wang, C. & Liu, C. 2018, 'Adopting DevOps in agile: challenges and solutions', Master thesis, Blekinge Institute of Technology, Sweden. (S112).

Wang, D., Lo, D., Bhimani, J. & Sugiura, K. 2015, 'Anycontrol—IoT based home appliances monitoring and controlling', *IEEE 39th Annual Computer Software and Applications Conference 2015*, IEEE, Piscataway, NJ, vol. 3, pp. 487–492. (S58).

Wettinger, J., Andrikopoulos, V. & Leymann, F. 2015, 'Automated capturing and systematic usage of DevOps knowledge for cloud applications', *IEEE International Conference on Cloud Engineering 2015*, IEEE, Piscataway, NJ, pp. 60–65. (S4).

Wettinger, J., Breitenbücher, U., Kopp, O. & Leymann, F. 2016, 'Streamlining DevOps automation for cloud applications using TOSCA as standardized metamodel', *Future Generation Computer Systems*, vol. 56, pp. 317–332. (S28).

Wettinger, J., Breitenbücher, U. & Leymann, F. 2014, 'Standards-based DevOps automation and integration using TOSCA', *IEEE/ACM 7th International Conference on Utility and Cloud Computing 2014*, IEEE, Piscataway, NJ, pp. 59–68. (S19).

Willnecker, F. & Krcmar, H. 2018, 'Multi-objective optimization of deployment topologies for distributed applications', *ACM Transactions on Internet Technology*, vol. 18, no. 2, p. 21. (S93).

Wu, Y., Sheng, Q.Z., Shen, H. & Zeadally, S. 2013, 'Modeling object flows from distributed and federated RFID data streams for efficient tracking and tracing', *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 10, pp. 2036–2045. (S71).

Wu, Z. & Madhyastha, H.V. 2013, 'Understanding the latency benefits of multi-cloud webservice deployments', *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 2, pp. 13–20. (S105).

Xiantao Zhang, Xiao Zheng, Zhi Wang, Hang Yang, Yibin Shen, and Xin Long. 2020. 'High-density Multi-tenant Bare-metal Cloud'. *In Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*

(ASPLOS '20). Association for Computing Machinery, New York, NY, USA, 483–495. DOI:https://doi-org.ezproxy.lib.uts.edu.au/10.1145/3373376.3378507

Yang, C., Shen, W., Lin, T. & Wang, X. 2016, 'A hybrid framework for integrating multiple manufacturing clouds', *International Journal of Advanced Manufacturing Technology*, vol. 86, no. (1–4), pp. 895–911. (S95).

Yaqoob, I., Ahmed, E., Hashem, I.A.T., Ahmed, A.I.A., Gani, A., Imran, M. & Guizani, M. 2017, 'Internet of Things architecture: recent advances, taxonomy, requirements, and open challenges', *IEEE Wireless Communications*, vol. 24, no. 3, pp. 10–16. (S67).

Yasaki, K., Ito, H. & Nimura, K. 2015, 'Dynamic reconfigurable wireless connection between smartphone and gateway', *IEEE 39th Annual Computer Software and Applications Conference 2015*, IEEE, Piscataway, NJ, vol. 3, pp. 228–233. (S61).

Yasrab, R. & Gu, N. 2016, 'Multi-cloud PaaS Architecture (MCPA): a solution to cloud lock-in', *3rd International Conference on Information Science and Control Engineering 2016*, IEEE, Piscataway, NJ, pp. 473–477. (S115).

Yonezawa, T., Ito, T., Nakazawa, J. & Tokuda, H. 2016, 'Soxfire: a universal sensor network system for sharing social big sensor data in smart cities', *Proceedings of the 2nd International Workshop on Smart*, ACM, New York, NY, p. 2. (S47).

Zheng, J., Liu, Y. & Lin, J. 2016, 'Exploring DevOps for data analytical system with essential demands elicitation', International Journal of Software Engineering and Knowledge Engineering (*SEKE)*, Vol. 26, No. 09n10, pp. 1453-1472. (S24).

Appendices

The appendices contain the information required to evaluate the framework. They also contain the contribution research papers.

Appendix A: Ethical Approval

Your ethics application has been approved as low risk—ETH18-2339

From: research.ethics@uts.edu.au

Mon 11/06/2018

To: Georges Bou Ghantous; Asif Gill

Dear Applicant

Your local research office has reviewed your application titled, 'DevOps Reference Architecture for IoT (single and multi-cloud)', and agreed that the application meets the requirements of the National Statement on Ethical Conduct in Human Research (2007). I am pleased to inform you that ethics approval has now been granted.

Your approval number is UTS HREC REF NO. ETH18-2339

You should consider this your official letter of approval. If you require a hardcopy please contact your local research office.

Approval will be for a period of five (5) years from the date of this correspondence subject to the provision of annual ethics reports to your local research office.

Your approval number must be included in all participant material and advertisements. Any advertisements on the UTS Staff Connect without an approval number will be removed.

Please note that the ethical conduct of research is an on-going process. The National Statement on Ethical Conduct in Human Research (2007) requires us to obtain reports about the progress of the research, and in particular about any changes to the research which may have ethical implications. You will be contacted when it is time to complete your first report.

Please refer to the AVCC guidelines relating to the storage of data, which require that data be kept for a minimum of 5 years after publication of research. However, in NSW, longer retention requirements are required for research on human subjects with potential long-term effects, research with long-term environmental effects, or research considered of national or international

significance, importance, or controversy. If the data from this research project falls into one of these categories, contact University Records for advice on long-term retention.

To access this application, please follow the URLs below:

* If accessing within the UTS network: https://rm.uts.edu.au

* If accessing outside of UTS network: https://vpn.uts.edu.au, and click on 'RM6 – Production 'after logging in.

If you have any queries about this approval, or require any amendments to your approval in future, please do not hesitate to contact your local research office or Research.Ethics@uts.edu.au.

REF: 12a

Appendix B: Consent Form

Participant Information Sheet

DevOps Reference Architecture for Multi-Cloud IoT Applications
[UTS HREC NO. ETH18-2339]

WHO IS DOING THE RESEARCH?

My name is *Georges Bou Ghantous* and I am an academic/student at UTS. My supervisor is Dr. Asif Q Gill.

WHAT IS THIS RESEARCH ABOUT?

The aim of the research is to create a DevOps Reference Architecture (DRA) that provides automation for application deployment using DevOps tools, and practices. The research utilizes cloud platform (single and multi-cloud) for deployment.

FUNDING

Funding for this project has been received from Commonwealth Government as student funding help for higher education and research students.

WHY HAVE I BEEN ASKED?

You have been invited to participate in this study because of your distinguished experience in IT/ Software Development field and understanding of DevOps, Cloud based software development and IoT applications. Your contact details were obtained from LinkedIn and/or by Supervisors' Industry Contacts as he is actively engaged with the industry on similar projects.

IF I SAY YES, WHAT WILL IT INVOLVE?

If you decide to participate, I will invite you to kindly participate in the evaluation of my research outcome using the method of online survey google form.

The google online form includes:

- A detailed description of the research project.
- A You Tube video that demonstrate the deployment of IoT application through DRA pipeline implementation.
- A list of publications about the research project.
- A set of questionnaires designed to evaluate and rate the research outcome artefact (DRA)

I will ask you to:

- Read the project description PDF and refer to the publications' list for further information.
- Watch the demo/testing YouTube video.
- Answer online questionnaire and rating survey questions in the online form.

Further information:

- The completion of the google online survey form may require 60 to 90 mins.
- No travelling or payments are required.
- The form is sent to you by email. The surveys will be conducted online. Upon completion the data will be sent back to me.
- The data will not include any information that may identify you in any way. No personal data will be collected; the survey collected data is technical and completely anonymous.
- The data will be stored in UTS systems as per UTS research data management policy on the UTS-recommended cloud storage CloudStor https://cloudstor.aarnet.edu.au/. Only my supervisor and I have access to data via UTS secure login to CloudStor.
- The collected technical/anonymous data will be used for publications of conference papers, journal papers and the research thesis.

ARE THERE ANY RISKS/INCONVENIENCE?

There is no risk, (low category) because it only involves online survey and video viewing of the DRA for IoT. It is only a technical and software content. Therefore, it is highly unlikely for any risk to occur.

DO I HAVE TO SAY YES?

Participation in this study is voluntary. It is completely up to you whether or not you decide to take part.

WHAT WILL HAPPEN IF I SAY NO?

If you decide not to participate, it will not affect your relationship with the researchers or the University of Technology Sydney. If you wish to withdraw from the study once it has started, you can do so at any time without having to give a reason, by contacting the researcher (Georges Bou Ghantous, email: Georges.BouGhantous-1@gmail.com).

If you withdraw from the study, you can do so at any time, the participation in this study is voluntary. However, it may not be possible to withdraw your response-data from the study results. Your response-data collected from the online survey will not contain any personal information about you. The collected data is technical and anonymous.

CONFIDENTIALITY

By signing the consent form you consent to the research team collecting and using online survey anonymous response-data for the research project. All this information will be treated confidentially. The data will be stored in UTS systems as per UTS research data management policy. Only my supervisor and I have access to data via UTS secure login. The collected anonymous data from your response to the online survey form will not identify you in any way and will only be used for the purpose of this research project (thesis) and papers publications (conference and journal).

WHAT IF I HAVE CONCERNS OR A COMPLAINT?

If you have concerns about the research that you think I or my supervisor can help you with, please feel free to contact us on [Mr. Georges Bou Ghantous (researcher): Georges.BouGhantous-1@uts.edu.au

Dr. Asif Q. Gill (supervisor): Asif.Gill@uts.edu.au]

You will be given a copy of this form to keep.

NOTE:

This study has been approved by the University of Technology Sydney Human Research Ethics Committee [UTS HREC]. If you have any concerns or complaints about any aspect of the conduct of this research, please contact the Ethics Secretariat on ph.: +61 2 9514 2478 or email: Research.Ethics@uts.edu.au], and quote the UTS HREC reference number. Any matter raised will be treated confidentially, investigated and you will be informed of the outcome.

Appendix C: Survey Invitation Letter

INFORMATION SHEET AND CONSENT FORM FOR ONLINE SURVEYS

DevOps Reference Architecture for Multi-Cloud IoT Applications

(UTS HREC REF NO. ETH18-2339)

My name is Georges Bou Ghantous and I am an academic/student at UTS. (My supervisor is Dr. Asif Q. Gill at UTS.)

I am conducting research in the area of software engineering and developed a DevOps reference architecture for Multi-Cloud IoT Applications deployment and would welcome your assistance. In order to evaluate the design and applicability of reference architecture, I would like to request you to participate in my research, review the reference architecture and provide your feedback via online google survey form. The research review and evaluation will involve an online google survey form will not take more than 60-90 minutes of your time. I kindly request you to participate in this research because of your expertise in the field of software engineering. No travels or payment will be required by participants.

Only technical data will be collected about the evaluation of the research artefact. No personal information will be collected. The data will be stored in UTS systems as per UTS storage and archiving policy. Only my supervisor and I have access to data via UTS secure login.

You can change your mind at any time and stop completing the survey without consequences.

If you agree to be part of the research and to research data gathered from this survey to be published in a form that does not identify you, please continue with answering the survey questions. The survey form will be sent to you by email.

If you have concerns about the research that you think I or my supervisor can help you with, please feel free to contact us.

Mr. Georges Bou Ghantous (researcher): Georges.BouGhantous-1@uts.edu.au

Dr Asif Q. Gill (supervisor): Asif.Gill@uts.edu.au

If you would like to talk to someone who is not connected with the research, you may contact the Research Ethics Officer on 02 9514 9772 or Research.ethics@uts.edu.au and quote this number: (UTS HREC REF NO. ETH18-2339)

Appendix D: Online Industry Survey Questionnaire

The following is a sample of the online Google survey that has been used to record the participants' responses. The survey participants are industry experts in the fields of software engineering, software architecture, DevOps, cloud computing and IoT. The participants are from Australia, the US, UK, Russia, Japan, Spain, Switzerland, India, Portugal, Sweden, Brazil, Costa Rica, Italy, South Korea, Canada, Germany and the Netherlands.

To access the online survey, please follow: https://goo.gl/JgQaUa

----------Start Survey----------

DRAv2.0 Framework Evaluation

Please submit feedback regarding DevOps Reference Architecture Framework for IOT-app deployment to Multi-Cloud (DRAv2.0)

DRA Framework Design Models

1. DRA Contextual Model
2. DRA Conceptual Model
3. DRA Logical Model
4. DRA Physical Model
5. DRA Pipeline Instance (Operational Model)

Survey Rating Factors

Qualitative Rating	Quantitative Rating
strongly disagree	1
disagree	2
average	3
agree	4
strongly agree	5

DRA Framework Description

1. Research Project Outline: https://goo.gl/WWiwNA
2. IoT app deployment in DRA on Multi-Cloud demo video: https://youtu.be/DmrIAciPKAU
3. DRA Framework development website: https://maven-app-heroku.herokuapp.com/

Publications

Publication 1:
Ghantous, Georges Bou and Gill, Asif, 'DevOps: Concepts, Practices, Tools, Benefits and Challenges' (2017). PACIS 2017 Proceedings. 96.
http://aisel.aisnet.org/pacis2017/96 OR: https://goo.gl/R9edjS

Publication 2:
G. Bou Ghantous, A. Gill, 'DevOps Reference Architecture for Multi-Cloud IOT Applications', 20th IEEE International Conference on Business Informatics CBI2018 Vienna Austria, 2018.
https://ieeexplore.ieee.org/abstract/document/8452669 OR: https://goo.gl/ya8J61

Q1 set: DRA Contextual Model Questionnaires

DRA Contextual Model (Describes in context the relationship between the tri-topology DevOps, Multi-Cloud, IoT at higher level). This model shows that DevOps can be used for deploying IoT apps to Multi-Cloud environment.

	Strongly disagree	Disagree	Average	Agree	Strongly agree
Does the contextual model provide the overall scope and purpose of using DevOps approach for IoT app and Multi-Cloud at the high level?	○	○	○	○	○
Do you think DevOps is appropriate for deploying IoT apps to multi-cloud environment?	○	○	○	○	○
Are the model elements (technologies) sufficient for the context?	○	○	○	○	○
Is the contextual model relevant to the DRA framework?	○	○	○	○	○
Are the contextual model elements important to the DRA framework?	○	○	○	○	○

Please provide comments or suggested improvements to DRA Contextual Model
Your Answer: ……..

Q2 set: DRA Conceptual Model Questionnaires

DRA Conceptual Model (Describes the Tri-Topology concept relationship and environment). DRA Conceptual Model is composed of two main tiers: 1) DevOps: enable concepts that evolve around team support and management. Communication and Collaboration: enable Dev and Ops team work Manage: Project management in real-time. Monitoring: monitor project development and deployment process in real-time. Sharing: enable global access for the team Human factor and culture: support human diverse cultures and geographical differences. Automation: enable automate project development and deployment throughout the life-cycle. Integration: integrate various aspect of project development to form one entity using automation. QA/Testing: provide logs and reporting for project deployment and application behaviour and health. Planning: enable retrospective agile planning based on QA/Testing reports. 2) Cloud: Abstract platform that

provides services and virtual servers for DevOps team. Cloud offer services such as PaaS, SaaS, DaaS, IaaS which provide DevOps team with features, practices, and functions that help enable DevOps approach such as (Monitoring, Security, Low Cost, Auto-Scaling, Automation and Integration) IoT app deployment on Multi-Cloud is achieved using CI-Broker (a DevOps tool) that enables the deployment functionality for the development life-cycle.

	Strongly disagree	Disagree	Average	Agree	Strongly agree
Does the conceptual model provide enough components for DevOps?	○	○	○	○	○
Does the conceptual model provide enough components for Cloud?	○	○	○	○	○
Does the conceptual model provide enough components for the Multi-Cloud deployment platform of IoT apps?	○	○	○	○	○
Is the conceptual model relevant for DRA framework?	○	○	○	○	○
Is the conceptual model important for DRA framework?	○	○	○	○	○
Is CI-Broker an important component for deploying IoT apps on Multi-Cloud?	○	○	○	○	○

Please provide comments or suggested improvements to DRA Conceptual Model
Your Answer: ……..

Q3 set: DRA Logical Model Questionnaires

DRA Logical Model (DRA Logical Model is based on the interactions of 5 sub-models M1-M5. The sub-models apply DevOps practices for software development). M1 enables code synchronization, automated code push to CI-Broker, automated commit logging. M4 enable communication of DevOps team and provide real-time logging and reports. M2 enable automation of build/test of IoT app and deploy to multi-cloud. M3 provide virtual servers of various cloud for IoT app deployment and auto-scaling. M5 provide cloud NoSQL DB and management of IoT data for DevOps team.

	Strongly disagree	Disagree	Average	Agree	Strongly agree
Does the logical model provide enough components for DevOps?	○	○	○	○	○
Does the logical model provide enough components for IoT apps deployment?	○	○	○	○	○
Does the logical model provide enough components for cloud platform?	○	○	○	○	○
Is the logical model relevant to the DRA framework?	○	○	○	○	○
Is the logical model important to the DRA framework?	○	○	○	○	○

Please provide comments or suggested improvements to DRA Logical Model
Your Answer: ……..

Q4 set: DRA Logical Model Specifications Questionnaires

DRA Logical Model Specifications (Organized into 5 sub-models M1-M5 which provide development features using suggested DevOps tools and practices). M1-M5 sub-models are based on cloud tools integration. The M1-M5 integrations enable DevOps practices.

	Strongly disagree	Disagree	Average	Agree	Strongly agree
DRA M1 automate code synchronization for DevOps team	○	○	○	○	○
DRA M2 enable automation for: repository update, build, testing	○	○	○	○	○
DRA M2 enable deployment to M3 (using CI-Broker)	○	○	○	○	○
DRA M3 automate scaling, and application scaling for users	○	○	○	○	○
DRA M4 enable automated log capture from: build, testing and deployment of IoT app	○	○	○	○	○
DRA M5 provide cloud Database management for DevOps team	○	○	○	○	○
Do you think that the M1-M5 sub-models provide enough functions for the DRA framework?	○	○	○	○	○
Do you think that the M1-M5 sub-models are relevant for the DRA framework?	○	○	○	○	○
Do you think that the M1-M5 sub-models are important for the DRA framework?	○	○	○	○	○

Please provide comments or suggested improvements to DRA Logical Sub-Models
Your Answer: ……..

Q5 set: DRA Physical Model Questionnaires

DRA Physical Model (This is a physical illustration for applying DevOps logical model to create a cloud-centric pipeline). DRA Physical Model is a tangible cloud structure based on multiple tools integration that provide DevOps teams with automated development practices and real-time management features. This model is based on the DRA Logical 5 sub-models and illustrates the development life-cycle of an application using M1-M5.

	Strongly disagree	Disagree	Average	Agree	Strongly agree
Does the physical model provide enough features for DevOps?	○	○	○	○	○
Does the physical model provide enough features for cloud?	○	○	○	○	○
Does the physical model provide enough features for IoT apps deployment?	○	○	○	○	○
Is the physical model relevant for the DRA framework?	○	○	○	○	○
Is the physical model important for the DRA framework?	○	○	○	○	○

Please provide comments or suggested improvements to DRA Physical Model
Your Answer: ……..

Q6 set: DRA Operational Model Questionnaires

DRA Pipeline Instance (Instance of DRA physical model which enables automation, continuous integration, real-time monitoring and communication for IoT app deployment on Multi-Cloud). DevOps team can manage IoT app automated build, testing and deployment on three clouds (AWS, GAE, Heroku). DevOps team can monitor application build, testing and heath using Slack (collects build, deployment logs). The IoT app interacts with IoT sensors through the Raspberry Pi. The central component in the pipeline is CI-Broker (i.e., Codeship) which enables automation of test/build and deploy to Multi-Cloud.

	Strongly disagree	Disagree	Average	Agree	Strongly agree
Does the pipeline provide enough components to support DevOps?	○	○	○	○	○
Does the pipeline provide enough components to support Multi-Cloud deployment?	○	○	○	○	○
Does the pipeline provide enough components to enable IoT app deployment on Multi-Cloud?	○	○	○	○	○
Does DRA pipeline enable automated IoT app deployment on Multi-Cloud using Codeship as CI-Broker	○	○	○	○	○
Is DRA pipeline tools integration relevant for the framework?	○	○	○	○	○
Are the DevOps tools integrated in the pipeline sufficient for the framework?	○	○	○	○	○
Does the DRA pipeline reflect the conceptual design model?	○	○	○	○	○
Does the DRA pipeline provide all the functions and features defined in the Logical model?	○	○	○	○	○

Please provide comments or suggested improvements to DRA Operational Model
Your Answer: ……..

Q7 set: DRA Usefulness Feedback and Ratings

Q7. What aspects are useful or valuable about DRA?
Your Answer: ……..

Do you consider DRA framework useful?	Strongly disagree	Disagree	Average	Agree	Strongly agree
For research proposes?	○	○	○	○	○
For teaching purposes?	○	○	○	○	○
In the industry?	○	○	○	○	○

Q8 set: DRA Suggested Improvements Feedback

Q8. What improvements would you suggest to DRA Framework?
Your Answer: ……..

Q9 set: Overall Feedback and Ratings

DRA Framework Overall Rating	1	2	3	4	5
On a scale of 1 to 5 please provide overall rating for the DRA framework	○	○	○	○	○

Q9. Comments?

……..……..

……..……..

……..……..

……..……..

……..……..

----------End Survey----------

Appendix E: Empirical Study Data

This section contains the source of the data used in the empirical evaluation in this thesis. The data have been stored on CloudStor, the UTS-recommended cloud storage service. Only the thesis author (Georges Bou Ghantous) and the principle supervisor (Dr Asif Q. Gill) have access to the data files on CloudStor. The empirical original data files are organised as follows:

- Case Study Template (CST):
  http://tiny.cc/nyu9iz
- Industry case study data files:
  https://cloudstor.aarnet.edu.au/plus/s/qSTTY96UgjBNDaC
- Research Lab case study files:
  https://cloudstor.aarnet.edu.au/plus/s/BdkVGQaSzqGLEnm
- Teaching survey files:
  - SEP files: https://cloudstor.aarnet.edu.au/plus/s/XPe8sd3tUGMSNyt
  - INP files: https://cloudstor.aarnet.edu.au/plus/s/tn9MYctbTNBwydZ
- Industry Survey:
  http://tiny.cc/99dldz
- Industry survey files:
  https://cloudstor.aarnet.edu.au/plus/s/plwJtmRbUM8oaJ0
- DRA Application Demo Video (Used in CST and Survey):
  https://youtu.be/JN38xS27ek0
- DRA Presentation Slides (Used in CST and Survey):
  http://tiny.cc/pcv9iz
- Ethical approval letter, invitation letters, PIS form:
  https://cloudstor.aarnet.edu.au/plus/s/KMSxYyO8HWKfXe5

Note: The author's LinkedIn page has been used to communicate with professionals from the IT industry and offer the industry field survey to the participants using the survey invitation letter (see Appendix C).

Note: All links in Appendix E were active at the time of thesis publications.

Appendix F: Research Papers

This appendix lists the conference papers that have been published as part of the ongoing research process during the development of this thesis (2016–2019):

Publication 1: Ghantous, G.B. & Gill, A. 2017, 'DevOps: concepts, practices, tools, benefits and challenges', Pacific Asia Conference on Information Systems 2017, AISeL. https://aisel.aisnet.org/pacis2017/96/

Publication 2: Ghantous, G.B. & Gill, A.Q. 2018, 'DevOps reference architecture for multi-cloud IOT applications', *IEEE 20th Conference on Business Informatics 2018*, IEEE, Piscataway, NJ, vol. 1, pp. 158–167. https://ieeexplore.ieee.org/abstract/document/8452669

Publication 3: Ghantous, G.B. & Gill, A.Q. 2019, 'An agile-DevOps reference architecture for teaching enterprise agile', *International Journal of Learning, Teaching and Educational Research*, vol. 18, no. 7. https://www.ijlter.org/index.php/ijlter/article/view/1499

Appendix G: Case Study Template

This appendix contains the case study template that will be used by organisations to determine the applicability of DRA implementation in the organisation's current environment. As explained in Chapter 5, the case study template is composed of two main steps: the DRA implementation process and the DRA evaluation process. Participants in case studies may use the DRA implementation process checklist (see Chapter 4) to manage the process of the DRA operational model implementation for the case study host (IT organisation, research lab, teaching lab). The DRA evaluation process is used by the case study participants to provide a qualitative review of the applicability of the DRA framework and test the DRAv2.0 instance.

DRAv2.0 Case Study Template: **Date: --/--/----**

The case study at the [organisation] is organised to demonstrate the applicability of the DRA in a real-world context. This evaluation involved the [participants] [identify the role of the participants]. A summary of the process adopted for the evaluation of DRA for [organisation] is presented below.

A. Case Study Introduction

- **Identify the case study organisation:** [organisation brief information]
- **Case study organisation context:** Conduct research and development [purpose and with whom]. [Identify the case study context in the organisation and identify]
- **Need, and problem:** [Identify the problem or the need of the hosting organisation. Identify how the problem or the organisation need is related to DRA]
- **Solutions:** [Briefly highlight how DRA can help the organisation and how DRA seem to answer the organisation needs].
- **Objective:** [Identify the DRA evaluation object and the organisation goal and gain from helping with the evaluation].
- **DRA Proof of Concept (POC) demo and presentation:** To evaluate the DRA framework, a presentation slide pack and demo was developed demonstrating the deployment of a predeveloped sample IoT application to a multi-cloud environment. This demonstrated the application and working of the DRA in operations.
    - Demo Video YouTube video: Link
    - Presentation Slides: Link

B. Evaluation Feedback

After the demo and presentation, the PhD researcher organised an evaluation session with the evaluator [participants] [identify the role of the participants]. The participants from the case study organisation [identify the role of the participants] provided qualitative feedback on the

DRA from a practical application perspective. The [organisation] [participants] evaluated the following DRA components:

- DRA architecture
- DRA operational model pipeline
- Software components
- Hardware components

1. DRA Architecture:

The DRA architecture was presented to the case study [organisation]. DRA architecture is composed of four design architecture models: Conceptual, Logical, Physical and Operational. The case study participants provided feedback on the architecture design and its applicability to their organisation. The expert from the [organisation] [participants] reviewed the design and provided feedback and comments.

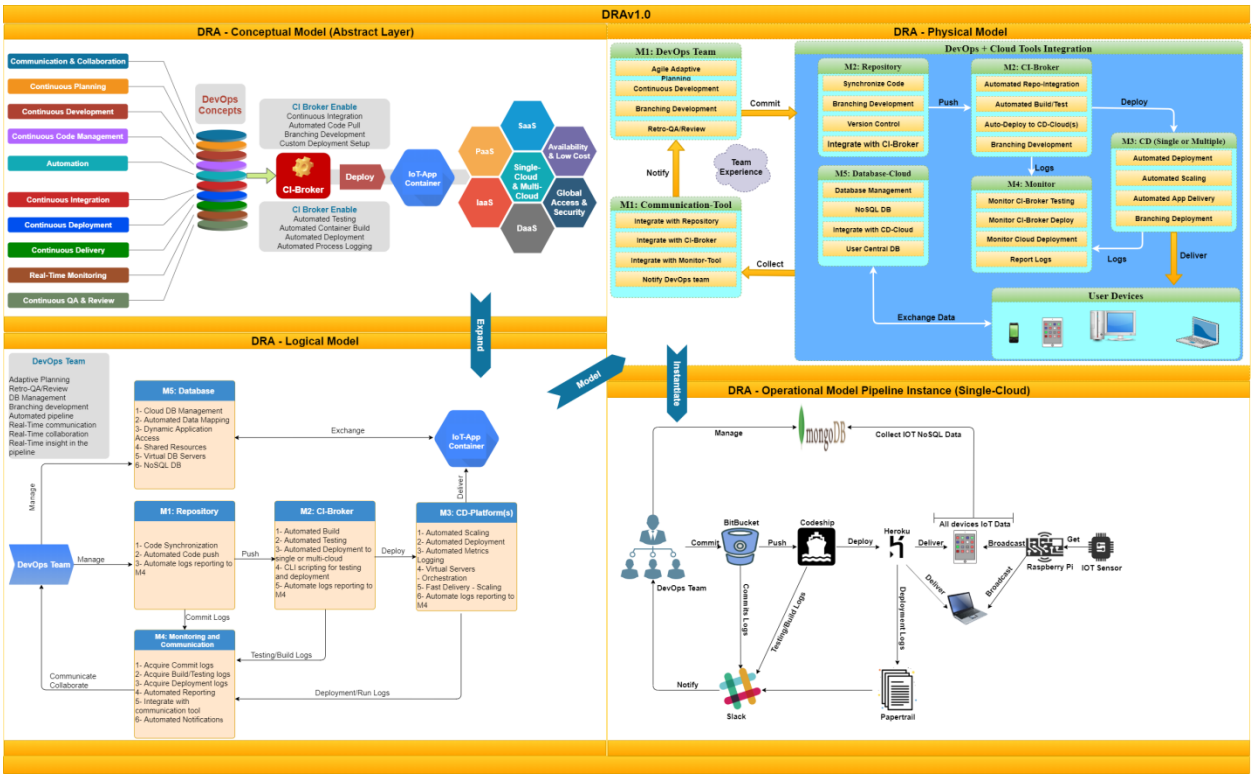


Figure G.1: DRAv2.0 Instance

Comments: Please provide comments about the DRA architecture

2. DRA Operational Model Pipeline:

In this step, the case study template provides a checklist for the DRAv2.0 instance pipeline implementation. The [organisation] [participants] may reuse the recommended toolset in Table 4.6 or configure DRAv2.0 instance with other tools of choice. Table G.1 shows the setup process of DRAv2.0 instance. Table G.2 is designed to facilitate the configuration of DRA for [organisation] [participants].

Table G.1: DRAv2.0 Instance Toolset

Tools	Features	Description
BitBucket	1. Code synchronisation 2. Automated code push 3. Automate commit logs to Slack	*BitBucket is a team collaboration and code management tool. It enables code synchronisation and automatically pushes the application to Codeship. It also sends commits logs to Slack.*
Codeship	1. Automated build/testing 2. Automated deployment 3. CLI scripting for testing/ deployment 4. Automate build/testing logs to Slack	*It is the Continuous Integration Broker (CI-Broker) tool. It enables automated build/testing for code automatically received from BitBucket. It also enables automated deployed to clouds. It also sends build/testing/deployment logs to Slack.*
Heroku	1. Automated scaling 2. Virtual servers—orchestration 3. Fast delivery—scaling 4. Automate deployment logs to Papertrail	*Heroku cloud enables automated scaling of the application deployed from Codeship. It enables automated scaling of the app for users. Run-time logs of the application are sent to Papertrail.*
Papertrail	1. Acquire deployment logs 2. Automated deployment logs to Slack 3. Automated notifications	*Papertrail monitoring the deployment and execution logs of the IoT application. It sends those logs to Slack.*
MLab	1. Cloud DB management 2. Dynamic application access 3. Virtual DB servers 4. NoSQL DB	*MLab is a MongoDB cloud database management service. It enables dynamic data access and mapping. MLab collected IoT data from the IoT application and store that data in JSON NoSQL. DevOps team can dynamically manage IoT data on MLab.*
Slack	1. Automated log management 2. Automated notifications 3. Real-time communication (chat, video) 4. Resources sharing option 5. Integration with Codeship and Papertrail	*Slack is a communication and collaboration tool. It provides DevOps team with real-time chat/video conference option and enables automated real-time notifications. Slack collects commit logs from BitBucket, build/test/deployment logs from Codeship and deployment/run logs from the cloud then notifies the team.*
AWS (CodeDeploy)	1. Automated scaling 2. Load balancing 3. Virtual servers—Orchestration 4. Fast delivery 5. Deployment monitoring	*AWS CodeDeploy enables automated scaling of the application deployed from Codeship. It enables automated scaling of the app for users. It also provides monitoring components for the application health at run-time.*
GAE	1. Automated scaling 2. User access management 3. Virtual servers—orchestration 4. Fast delivery 5. Deployment monitoring	*GAE enables automated scaling of the application deployed from Codeship. It enables automated scaling of the app for users. It also provides monitoring components for the application health at run time.*

Comments: Please provide comments about the DRA tools set

Table G.2: DRAv2.0 Instance Setup and Configuration Template

Step	**Features**	**Tools** [or other tools of choice]
DevOps Team	• Create an application project (IoT app) • Create testing modules • Create MongoDB (mLab) connector module • Add necessary dependencies and plugins	[Provide a list of plugins and dependencies]
Repository	• Create a cloud repository for the application • Integrate BitBucket with Slack and push commit logs	Bitbucket Slack
CI-Broker	• Setup Codeship environment script: *[which programming language]* *[which compiler – compiler command]* *[which tester – testing command]* • Setup Codeship deployment master branch to: Heroku (see documentation link) AWS (see documentation link) Google App Engine (see documentation link) • Integrate Codeship with Slack and push build/test logs	Codeship
CD Platform	**Heroku Setup:** • Create the maven-app-heroku project on Heroku • Add Web Dyno on Heroku for auto-scaling • Add Procfile web dyno script to the root directory **AWS Setup:** • Create the maven-app-heroku application on AWS • Create a user and get: the secret key and access key • Setup 2 or more EC2 instances • Setup a security group • Setup a deployment group using EC2 instances • Setup an S3 bucket • Provide Codeship with access to S3 and CodeDeploy **GAE Setup:** • Create the maven-app-heroku application on GAE • Setup a bucket on Google • Provide Codeship with access to the bucket and the GAE	Heroku AWS GAE
Monitoring	• Enable log capturing from Heroku, AWS, and Codeship • Integrate Papertrail to push deployment logs to Slack • Integrate Slack with Papertrail, Codeship, BitBucket	Papertrail Slack
Database	• Create a mLab DB account through Heroku • Provide the connection link of mLab DB (MongoDB) to the IoT application	mLab

Comments: Please provide comments about the DRA setup and configuration process

3. Software Component:

DRAv2.0 instance pipeline can be configured to deploy any type of applications. Table G.3 provides the [organisation] [participants] with a demo application (maven-app-heroku) to test the DRA. However, in case studies, [organisation] [participants] may use their own IoT application (or no IoT applications) to test DRAv2.0 instance pipeline.

Table G.3: Software Component

Software Component Checklist	Description
Application Name *[required]*	maven-app-heroku
Application Type *[IoT]*	Java Maven app with IoT module
Programming Languages *[required]*	Java/Python
Unit Testing Module *[required]*	JUnit
Acceptance Testing Module *[required]*	Cucumber
Resources	https://bitbucket.org/product
Access	observer
Username	devopsobserver@hotmail.com
Password	observer

Comments: Please provide comments about the IoT app

4. Hardware Component:

Table G.4 provides a sample IoT device network (see Copy of Figure 4.17) to test the IoT application process. However, in case studies, [organisation] [participants] may use their own IoT devices and sensors to test the IoT application deployed using DRAv2.0 instance pipeline.

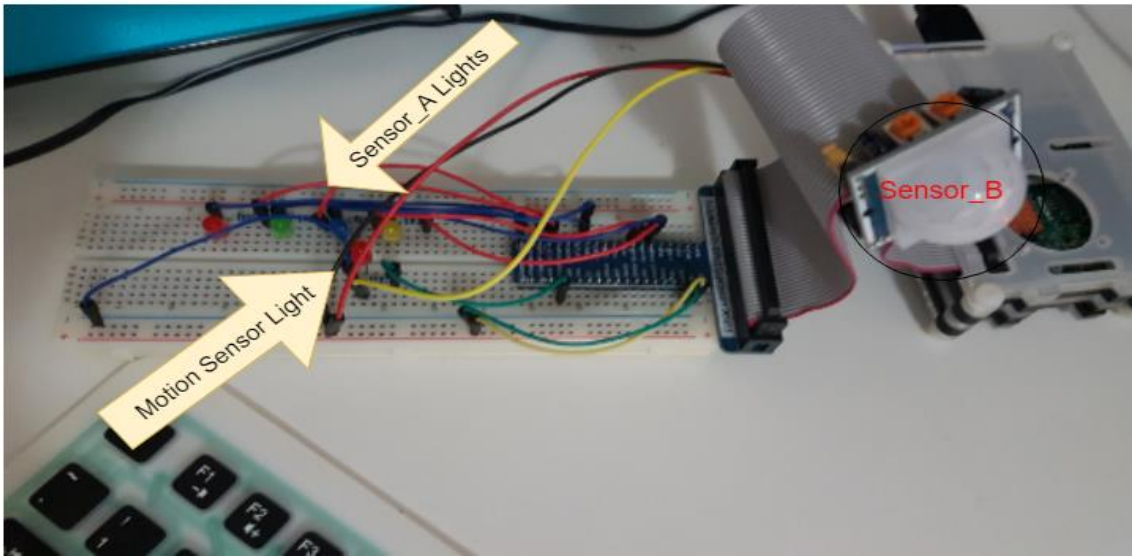


Figure G.2: IoT Network

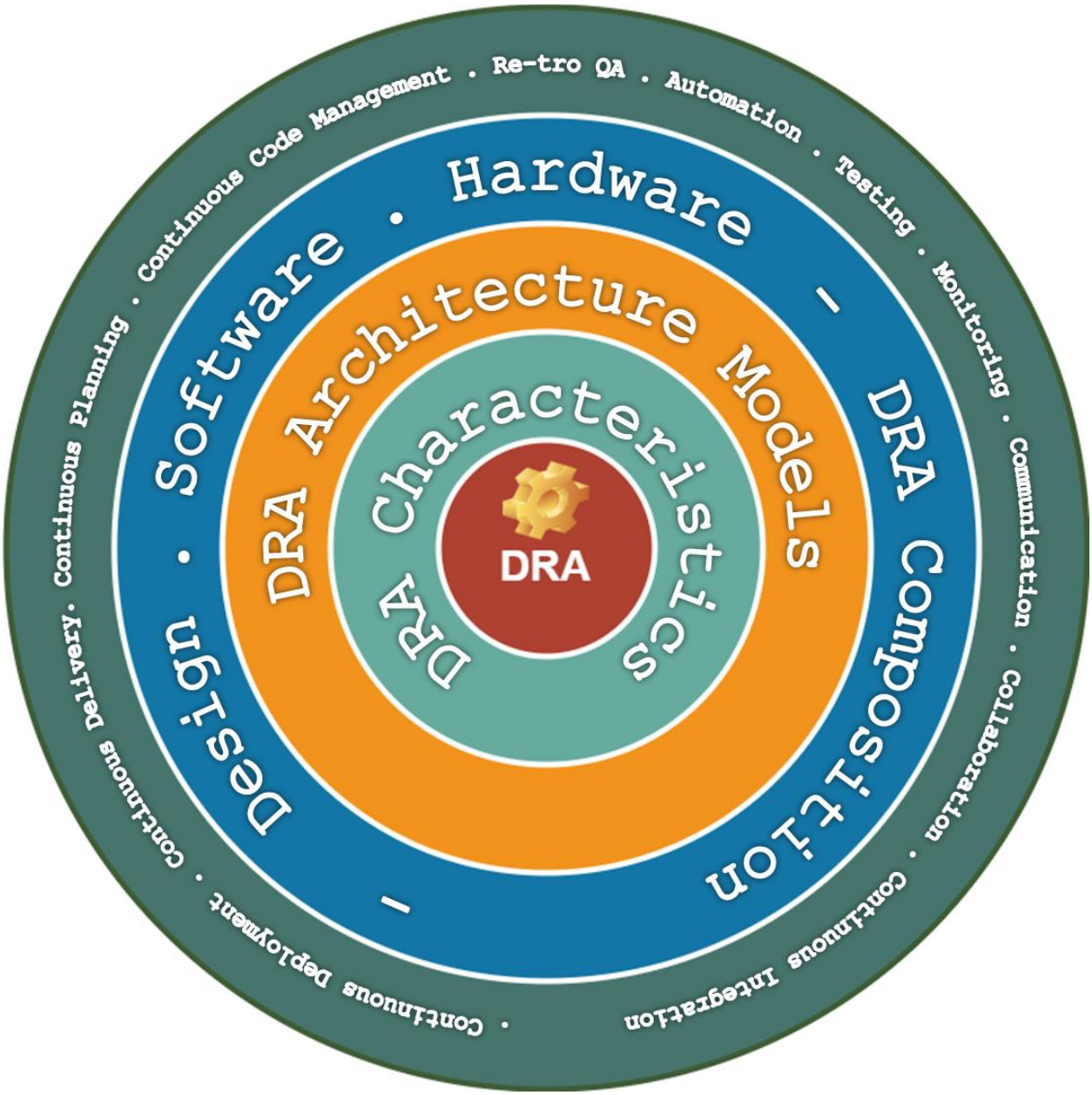
Table G.4: Hardware Component

Sensor	Setup and Configuration	Type
Sensor A	4 LED lights (multi-coloured) connected to 4 GPIO pins [13, 17, 19, 22] (named **Sensor_A**) on a Raspberry Pi Model 3 B (named **RPIB**).	LED lights Raspberry Pi
Sensor B	Motion Sensor + 1 LED connection to a single GPIO pin [12] (named **Sensor_B**) on a Raspberry Pi Model 3 (named **RPIB**).	Motion Sensor Raspberry Pi

Comments: Please provide comments about the IoT network

Comments: Please provide comments about the demo video

Comments: Please provide overall comments about DRA


DRA
DRA Characteristics
DRA Architecture Models
DRA Composition
Design - Software • Hardware -
Continuous Code Management . Re-tro QA . Automation . Testing . Monitoring . Communication . Collaboration . Continuous Integration . Continuous Deployment . Continuous Delivery . Continuous Planning