Toward a Framework for the Composition and Assessment of IoT and CPS Capabilities : Smart Cities Applications

Khalid Halba

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

Spécialité : Informatique
École doctorale : MSTII - Mathématiques, Sciences et technologies de l'information, Informatique
Unité de recherche : Laboratoire d'Informatique de Grenoble

Présentée par :

Khalid HALBA

Toward a Framework for the Composition and Assessment of IoT and CPS Capabilities: Smart Cities Applications

Vers un Framework pour la composition et l'évaluation des fonctionnalités des IoT et CPS : Applications dans le contexte des villes intelligentes

Direction de thèse :

Ahmed LBATH Université Grenoble Alpes	Directeur de thèse
Edward GRIFFOR National Institute of Standards and Technology (NIST)	Codirecteur de thèse
Anton DAHBURA Johns Hopkins University	Co-encadrant de thèse

Rapporteurs :

Abdelaziz BOURAS Professeur des Universités, Qatar University	Rapporteur
Samia BOUZEFRANE Professeur des Universités, Conservatoire National des Arts et Métiers (CNAM Paris)	Rapporteure

Thèse soutenue publiquement le **1 juin 2023**, devant le jury composé de :

Saddek BENSALEM Professeur des Universités, Université Grenoble Alpes	**Président**
Ahmed LBATH Professeur des Universités, Université Grenoble Alpes	Directeur de Thèse
Abdelaziz BOURAS Professeur des Universités, Qatar University	Rapporteur
Samia BOUZEFRANE Professeur des Universités, Conservatoire National des Arts et Métiers (CNAM Paris)	Rapporteure

Invités :

Edward GRIFFOR
Associate Director for CPS & IoT, SCS division, CTL, National Institute of Standards and Technology.

Abdella BATTOU
Division Chief, SCS division, CTL, National Institute of Standards and Technology.

Anton DAHBURA
Co-Director of the Institute for Assured Autonomy, Johns Hopkins University.

"Information is the resolution of uncertainty."

Claude Elwood Shannon(1916-2001) - The father of Information Theory.

Acknowledgements

First, I would like to express my gratitude and appreciation to my advisor, Professor Ahmed Lbath, for the opportunity to pursue my Ph.D. under his supervision. Over the last few years, he gave me invaluable research advice and continuous support during my Ph.D. study.

I am also very grateful to my supervisors:

Dr. Edward Griffor at the National Institute of Standards and Technology (NIST) and Dr. Anton Dahbura at The Johns Hopkins University for the excellent research environment and for ensuring I have the resources necessary to accomplish my research activities.

Additionally, I would like to thank all the administrative staff from the University of Grenoble, particularly Mrs. Zilora Zouaoui and Mrs. Maud Chaurier, for their help and assistance.

Next, I would like to thank Dr. Abdella Battou and Dr. Lotfi Benmohamed for their technical support and helpful advice during the initial phase of my Ph.D.

Moreover, I am grateful to my colleagues and supervisors at NIST, including Dr. Thomas Roth, Dr. Dave Wollman, Dr. Chris Greer, Dr. James Filliben, Dr. Charif Mahmoudi, Dr. Siham Khoussi, Dr. Hasnae Bilil, Omar Ilias El Mimouni, Dr. Lilia Hannachi, Mheni Merzouki, and Asmaa Hailane. I have learned many things from them that helped me improve and implement my work.

I am very grateful to all the reviewers and jury members for agreeing to review this work and for their valuable feedback on the manuscript.

Finally, I would like to thank my wife, Hadhoum, and my family, including my father, Mohamed, my mother, Aicha, my sister, Khadija, and my brother Hassan, for their love and support throughout the years. None of my accomplishments would have been possible without them.

UNIVERSITÉ GRENOBLE ALPES

Abstract

l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique

Doctor of Philosophy

Toward a Framework for the Composition and Assessment of IoT and CPS Capabilities: Smart Cities Applications

by Khalid HALBA

By 2030, over half a trillion devices will be connected to the internet. With so many devices providing a wide range of services, a framework for prototyping, verifying, and assessing Internet of Things (IoT) and Cyber-Physical Systems (CPS) capabilities is needed. Based on our research, existing IoT and CPS service composition frameworks either lack solid composition foundations that consider the various stakeholders' concerns, don't propose modeling semantics, or don't provide the means to formally verify the different states a composite system might yield. A comprehensive composition framework should facilitate the composition of capabilities and give the stakeholders means to model and verify compositions reliably. An IoT and CPS Composition Framework (ICCF) is proposed to achieve this goal. ICCF is based on the NIST CPS framework composition guidelines, mPlane-based composition semantics, and TLA-based (Temporal Logic of Actions) formal verification descriptors and tools. For experimental validation, we propose an implementation of ICCF, named IoTCaP (IoT Capabilities Platform), which takes into consideration ICCF foundations and enables the composition of novel capabilities in various domains. We have implemented ICCF in three domains of interest: Well-being in smart buildings, safety in Autonomous Driving Systems, and health improvement in Intensive Care Units.

Contents

Abstract **vii**

1 Context, Identified Problems, and Work Plan **1**
1.1 Context: Smart Cities: a source of diverse IoT and CPS capabilities. . . 1
1.1.1 Smart Buildings . . . 2
1.1.2 Smart Transportation Systems . . . 3
1.1.3 Smart Health Applications . . . 3
1.2 Identified Problems . . . 3
1.3 Work Plan . . . 4

2 Introduction **7**
2.1 What's IoT or CPS ? . . . 7
2.2 What's an IoT capability ? . . . 7
2.3 Capabilities Classification: . . . 8
2.4 Composition and Decomposition Functions: . . . 8
2.5 A Layered model for IoT Capabilities Composition and Decomposition 10

3 State of the art and gaps in IoT service composition SLRs, Surveys, Frameworks, and Platforms **13**
3.1 Systematic Literature Reviews (SLRs) and Surveys: State of the art and identified gaps . . . 13
3.2 Service Composition Frameworks and Platforms: State of the art and identified gaps . . . 16
3.3 Summary and takeaways from related work . . . 21

4 Answering scientific questions related to Formal, Technical, and QoS aspects in service composition. **23**
4.1 SLR : Research Methodology . . . 24
4.1.1 Formulating the research questions . . . 24
4.1.2 Explaining the Search process and the Inclusion/Exclusion criteria . . . 24
4.1.3 Quality assessment . . . 30
4.1.4 Limitations . . . 31
4.1.5 Data Extraction . . . 31
4.1.6 Analyzing Data . . . 31
4.1.7 Execution . . . 31
4.2 SLR Results: Taxonomy and Surveyed Aspects Data . . . 32
4.2.1 A Taxonomy of the SLR research questions aspects and sub-aspects and extracted data distribution. . . . 32
4.2.2 Formal Aspects . . . 34
4.2.3 Technical Aspect . . . 39

4.2.4 QoS Aspect . . . 46
4.3 SLR : Discussion . . . 50
4.3.1 Answering SLR Questions . . . 50
4.3.2 SLR Trends, Gaps, and Threats to Validity of the study . . . 72
4.4 Conclusion . . . 74
4.4.1 Summary . . . 75
4.4.2 Benefits of the study for different stakeholders . . . 75

5 Toward an IoT and CPS Capabilities Composition Framework (ICCF). 77
5.1 Introduction: Pillars for a comprehensive and trustworthy service composition framework for IoT and CPS capabilities. . . . 78
5.1.1 Composition Guidelines . . . 80
5.1.2 Modeling Semantics . . . 82
5.1.3 Formal specification languages and formal verification tools . . 83
5.1.4 The NIST CPS Framework guidelines for building IoT and CPS Platforms that meet stakeholders' concerns. . . . 86
5.1.5 mPlane semantics for describing capabilities algebra, objects, components, and operations . . . 88
5.1.6 Formal Specification and Verification through PLUSCAL and TLA+ . . . 93
5.1.7 Implementation efforts . . . 98
5.1.8 IoTCaP: An ICCF-based Platform For Implementing The Composite Capability Entities And Interactions using mPlane Semantics. . . . 98

6 ICCF Applications in smart cities 103
6.1 Well-being as a Composite Capability in the Smart Building Domain: A Formal and Technical Study. . . . 103
6.1.1 Introduction . . . 104
6.1.2 Related Work . . . 105
6.1.3 An ICCF definition of Well-Being as a composite capability . . 108
6.1.4 Formal Specification And Verification . . . 114
6.1.5 IoTCaP Implementation, Results, and Analysis . . . 119
6.1.6 Conclusion, Challenges, And Future Work . . . 124
6.2 Safety as a Composite Capability in the Smart Transportation Domain. 125
6.2.1 Introduction . . . 125
6.2.2 Related Work . . . 125
6.2.3 An ICCF definition of Safety as a Composite Capability: . . . 126
6.2.4 Formal Specification And Verification . . . 128
6.2.5 IoTCaP Implementation, Results, and Analysis . . . 128
6.2.6 Conclusion, Challenges, And Future Work . . . 130
6.3 Ongoing work: Health improvement as a Composite Capability in the Smart Health Domain. . . . 134

7 Conclusion 137
7.1 Summary of the work . . . 137
7.2 Ongoing and future work . . . 139

List of Figures

1.1 Smart cities leverage many systems and sensors (data sources) to implement applications of interest in multiple domains, including residential buildings, healthcare facilities, and transportation infrastructure. 2

2.1 An illustration of IoT capabilities modeling, composition, and decomposition functions 9
2.2 Modeling, composition, and decomposition layers and functions . . . 11
2.3 Composition of atomic services into composite services. 11
2.4 decomposition of composite services into atomic capabilities. 11

3.1 Distribution of previous SLRs questions per the proposed taxonomy aspects and the SLR methodology. 16
3.2 Requirements for building a trustworthy composition framework and platform. 20

4.1 Search Strings and filtering method. 27
4.2 Count of primary studies per year and per publisher. 28
4.3 Distribution of primary studies per search method: Main SCOPUS (MS), Snowballing SCOPUS (SS), Manual Google Scholar (MGS). . . . 29
4.4 Multi-stage selection process with the inclusion/exclusion criteria . . . 29
4.5 Proposed Taxonomy for the IoT/CPS capabilities composition and decomposition topic (root): Aspects (leaves), Sub-aspects (sub-leaves). . . 33
4.6 Formal verification of IoT/CPS capabilities workflow. 36
4.7 AQ1: motivations for the native support of service composition/decomposition mechanisms in standards, frameworks, and reference architectures. 51
4.8 AQ2: Formal representations modeling and verification properties and techniques and corresponding modeled/verified properties in service composition. 52
4.9 AQ3: formal representations: modeling and formal verification trends. 54
4.10 AQ4: trends related to the state space explosion problem approaches and outcomes. 55
4.11 AQ5: primary studies trends related to stakeholders' concerns in different domains. 57
4.12 AQ6: implementation trends in service composition. 59
4.13 AQ7: trends in service composition automation level. 60
4.14 AQ11: benefits of building platforms with decomposition in mind. . . 65
4.15 AQ12: identified roles of AI/ML in service composition. 66
4.16 AQ14: interoperability challenges and solutions in service composition. 69
4.17 AQ15: privacy challenges and solutions in service composition. 72

4.18 Primary studies trends in IoT/CPS capabilities composition and decomposition based on the number of primary studies that partially or thoroughly addressed a particular formal, technical, or QoS sub-aspect, including non-addressed sub-aspects in this study (Other Formal, Other Technical, Other QoS. 73

5.1 ICCF foundations, semantics, and formal verification pillars (foundations, semantics, formal verification) and parameters to be respected by these pillars. 80
5.2 ICCF's selected pillars based on the comparative analysis: foundations (NIST CPS Framework), semantics (mPlane), and formal verification (PLUSCAL/ TLA / TLA+) 86
5.3 NIST CPS Framework Composition-specific aspects and Concerns to consider 88
5.4 Space of Capabilities and Entities 89
5.5 Composition Hierarchy 91
5.6 Specification and Result model. 91
5.7 Virtual Composite Service model in PlusCal and its translation to TLA 95
5.8 Atomic and Composite Capabilities and their range of possible and trustworthy values. 96
5.9 Symbolic Execution Results: Error status (1), number of states (2), user output (3) 96
5.10 Deadlock and Trustworthiness Verification 97
5.11 Sequence diagram for IoTCaP interactions and entities: crafting a composite capability from atomic functions is highlighted using mPlane composition interactions. 100
5.12 Filled Requirements for building a trustworthy composition framework and platform. 101

6.1 Composition of the well-being composite service using atomic capabilities that are weighted based on user/stakeholder preferences. . . . 113
6.2 Sequence diagram for the well-being capability expressed using mPlane interactions. 114
6.3 Modeling the state space for the well-being composition values based on corresponding atomic capabilities values, ranges, projections, and scores. 116
6.4 Screenshot from TLA+ indicating the functions to verify described using the PLUSCAL language. 118
6.5 Screenshot from TLA+ indicating the composition functions to verify translated to TLA specification. 119
6.6 Experiment devices, boards, and atomic sensors: 2 X ESP8266, 1 X DHT22, 1 X SDS011, 1 X GLiNET WiFi Access, 1 X KY038, 3 X 5V DC outlets. 120
6.7 TLC model checking results show historical data of the well-being model's total number of generated states, distinct states, and the state's queue handling. 121

6.8 Screenshot of the Vue.JS IoTCaP front-end, an actor can fill in the weights for the capabilities based on his/her preferences and submit a specification to the well-being back-end, a result is returned that carries both atomic capabilities as well as the composite capability of well-being based on inserted weights. 123
6.9 An illustration of the IoTCaP implementation layers for collecting and composing smart building data into a composite capability. 124
6.10 Sequence diagram for the safety assessment capability expressed using mPlane composition interactions. 129
6.11 PLUSCAL description translated to a TLA formal specification for the Safety use-case. Safety is a binary composite capability of two discreet atomic capabilities, speed, and distance. 130
6.12 Symbolic execution of the safety composite capability specification with 20000: the number of possible unique outcomes, the log file shows the safety assessment/value for each pair of speed and distance values. 131
6.13 Emergency braking experiment simulation environment. 132
6.14 At T=99, the vehicle detected the obstacle, we vary vehicle speed at the moment of detection and distance to the obstacle to assessing safety 133
6.15 Color Coded scheme for assessing the safety of a maneuver in the smart transportation domain. 133
6.16 A smart ICU testbed, with multiple data sources or sensors 135
6.17 Sequence diagram for the Health improvement capability expressed using mPlane interactions. 136

List of Tables

3.1 Related work: Literature and SLR studies that tackled the IoT/CPS capabilities composition topic. . . . 17
3.2 Comparing existing IoT/CPS composition environments, platforms, and frameworks . . . 19

4.1 Systematic literature review (SLR) questions (RQs) and corresponding motivations. . . . 25
4.2 Criteria, Primary Studies, and scope of the SLR: **P**opulation, **I**ntervention, **C**omparison, **O**utcome, **C**ontext (**PICOC**). . . . 26
4.3 Composition and decomposition aware standards, frameworks, and reference architectures . . . 35
4.4 Formal representations and composition algebras. . . . 36
4.5 Formal verification techniques (Model Checking: MC | Equivalence Checking: EC | Theorem Proving: TP), verified properties, applications, and tools. . . . 37
4.6 The state space explosion problem in service composition: description, techniques for resolution, and outcomes. . . . 38
4.7 Capabilities composition domains and corresponding stakeholders concerns. . . . 40
4.8 Composition platforms, engines, and implementations. . . . 41
4.9 Service composition process types and automation levels. . . . 42
4.10 Communication protocols roles in IoT/CPS service composition/decomposition. . . . 43
4.11 Capabilities data models/schemas properties, roles, and examples of attributes. . . . 44
4.12 Service decomposition efforts description and benefits . . . 45
4.13 AI/ML use in service composition/decomposition: description of efforts, orientation, applications, and mechanisms. . . . 46
4.14 Scalability challenges and solutions in IoT/CPS service composition. . . 47
4.15 Interoperability challenges and solutions in IoT/CPS service composition. . . . 48
4.16 Privacy challenges and solutions in IoT/CPS service composition. . . . 49

5.1 Comparing frameworks composition foundations in IoT and CPS . . . 82
5.2 Comparing IoT service composition semantics. . . . 83
5.3 Comparing IoT service composition formal specifications languages. . 85
5.4 Aspects (including composition) and concerns related to the NIST CPS framework. . . . 87

6.1 State space of values, ranges, and scores related to well-being atomic capabilities in a residential building. . . . 117

6.2 Comparing implementation technologies. 122

List of Abbreviations

IoT	**I**nternet **o**f **Th**ings
CPS	**C**yber **P**hysical **S**ystems
SLR	**S**ystematic **L**iterature **R**eview
ICCF	**I**oT **C**PS **C**omposition **F**ramework or **I**oT **C**apabilities **C**omposition **F**ramework
IoTCaP	**IoT Ca**pabilities **P**latform

Chapter 1

Context, Identified Problems, and Work Plan

In this chapter, we introduce the context of our study: smart cities generate diverse and large amounts of data through various data sources in the form of IoT devices, cyber-physical systems, or full-fledged connected services. Capabilities extracted from these data sources can be composed to build value-added services in multiple domains.

In this chapter, we a) explain the context of this work through 3 examples of smart city environments, b) briefly explain gaps in existing research efforts from both: i) a knowledge perspective (SLRs and literature reviews) and an implementation perspective (weaknesses and gaps in existing capabilities composition frameworks and platforms) the problem with existing composition platforms (a detailed discussion is provided in the following chapters), and finally, we describe this work's plan to tackle gaps in the knowledge and implementation sides as well as proposing a novel composition framework with implementations in various smart city domains.

1.1 Context: Smart Cities: a source of diverse IoT and CPS capabilities.

Smart cities in the current era provide a broad set of services targeting multiple domains, including smart living, intelligent transportation, and intelligent health (Check Figure 1.1). These intelligent services rely on connected IoT or CPS systems which generate data that, when aggregated or composed, yields the possibility for innovating services with added value to end users [240]. Furthermore, complex IoT and CPS systems can also be decomposed -from a capability or computation perspective- into atomic capabilities that can be reused or recycled in other systems for cost-saving and computation efficiency purposes [14] [15].

The composition of IoT or CPS capabilities can either occur with one domain (intra-domain) or between multiple domains (inter-domain). An example of domains that can benefit from an inter-domain composition is smart buildings with high air quality standards and extremely low noise levels. This specific composition would require input from the transportation infrastructures nearby to assess whether pollution or noise levels -caused by a specific density of vehicles on nearby roads or highways- are below the stakeholders' defined minimum

thresholds [**umair2021impact**] [265].

In this work, we focus on compositions within a single, smart city domain and leave inter-domain compositions for future work.

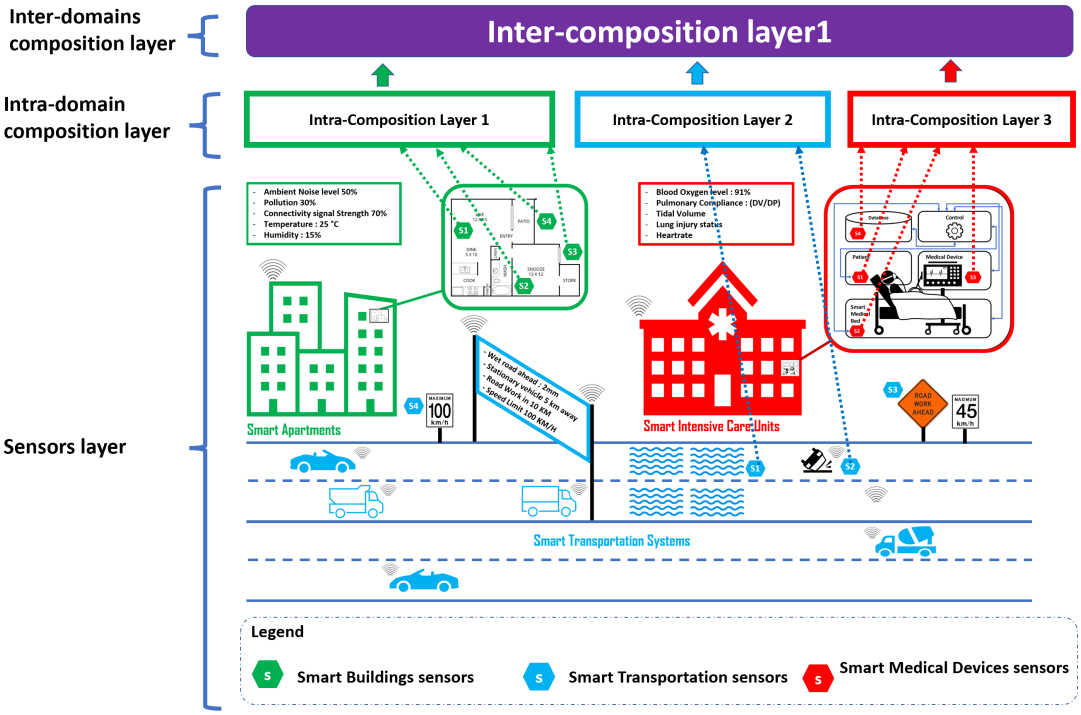

FIGURE 1.1: Smart cities leverage many systems and sensors (data sources) to implement applications of interest in multiple domains, including residential buildings, healthcare facilities, and transportation infrastructure.

We discuss efforts in three smart-city domains where the composition of IoT and CPS capabilities was leveraged to innovate new services and applications based on composable atomic features.

1.1.1 Smart Buildings

For smart living, multiple connected appliances and sensors provide a wide range of services that ensure comfort or well-being. Examples of devices or sensors include temperature/humidity probes, noise and WiFi Signal Strength sensors, and fire alarms, to mention a few. Examples of efforts that leveraged service composition in the smart building domain include [29], where the researchers focused on automating service composition processes (discovery, selection, aggregation). Another service composition effort that focuses on smart building applications is highlighted in [169][171], where the authors proposed a composition platform -named IoT Composer-which focuses on creating smart building services in a way that ensures ease of use for end users. Similarly, authors in [84] focused on automating temperature and light management in smart buildings using ranking and filtering blueprints or abstractions that are intelligible to end users. Finally, in [9], researchers

addressed person detection as a smart composite feature in smart buildings while enhancing the user-friendliness of the platform's graphical user interface.

1.1.2 Smart Transportation Systems

Smart transportation services and sensors leverage IoT sensors and devices to inform drivers or autonomous vehicles about potential road hazards or weather/road conditions which can impact people and property safety. When combined/composed, data generated by all the IoT sensors can help provide a framework for safety in transportation infrastructure for smart cities. Many research efforts discussed service composition in the smart transportation domain, including [121], where authors leveraged the UCEF environment to compose an autonomous driving experiment to assess safety in an emergency braking scenario. Another example is found in [196], where a platooning feature that enables cooperation during adaptive cruise control for a series of vehicles is composed. Insurance companies leverage service composition to predict insurance quotes based on sensor data and predefined driving behaviors and thresholds; this was implemented in the DRACENA composition platform [310]. Service composition in the smart transportation domain is illustrated in other research efforts such as [264], where traffic congestion at the city level leverages Node-RED's distributed API to estimate traffic congestion using data collected from traffic cameras. Similarly, researchers in [68] proposed an estimator for parking, traffic, and noise using FIWARE's composition capabilities; these estimators are considered atomic capabilities that, when composed, provide a modular open trip planner with reusable components. Finally, in [182] and [118], researchers proposed a travel reservation composite service based on service-oriented computing architectures or web-based architectures where users benefit from the ease-of-use and plug-and-play automated features of the composition platform.

1.1.3 Smart Health Applications

Hospitals and Intensive Care Units (ICUs) can leverage data provided by sensors that are either connected to patients or to smart medical devices to help improve patients' health or help predict hospital occupancy based on patients' health. Data aggregated and composed from different sensors can be used to that end in a way that renders care facilities more efficient and also helps track and improve the patients' health. Many research efforts tackled the service composition role in smart health applications. An example of such efforts includes [196], where a personal health management service was conceived based on a distributed microservices platform, with a focus on the privacy of patients' data required for smart health applications. Another example that addressed service composition in the smart health applications domain was identified in [26], where a cloud-based health monitoring system was introduced which composes health/medical prescription based on the patient data and the medical team input.

1.2 Identified Problems

Based on the research we have done, we identified two problems that this work strives to add value and contribute to:

I) The topic of IoT service composition still presents many scientific and technical gaps to be filled out. Based on a comprehensive study of existing research, we classified these gaps under formal (e.g., how state space explosion in service composition is addressed), technical (service decomposition categories and roles), or QoS aspects (privacy, interoperability, and scalability challenges in service composition).

After discussing gaps in SLRs and literature reviews in Chapter 3 Section (3.1), 15 formal, technical, and QoS questions were analyzed and addressed using the systematic literature review approach in Chapter 4.

II) From an implementation and prototyping of composite capabilities perspective, the existing IoT or CPS composition frameworks and platforms either lack composition foundations and guidelines, use complicated and non-human-readable or non-composition-friendly semantics that isn't accessible, or isn't easily convertible to specifications that can be automatically verified, which takes us to the 3rd point which is formal verification, where we noticed in some frameworks the lack of accessible and rapid formal verification mechanisms of the proposed composition models, especially from a measurability perspective: making sure the composite capability doesn't yield unexpected results and respects user or stakeholder's expectations is important when crafting and implementing novel services. We explain these problems in detail in Chapter 3 Section (3.2), and based on a comparative study in Chapter 5, a framework for composing novel smart city capabilities (ICCF) is proposed to address all these concerns and gaps in existing frameworks, and a platform (IoTCaP) for implementing the newly proposed framework foundations is also presented.

1.3 Work Plan

To achieve the objectives of this work, we explained in this chapter (**Chapter 1**) the context of the work and identified problems to be addressed.

In **Chapter 2**, we provide definitions -used throughout this work- and explanations for the different components and operations related to IoT or CPS capabilities composition and decomposition.

In **Chapter 3**, we present related work which identifies gaps in two research areas:

a) SLRs and literature reviews, with the main objective of identifying formal, technical, and QoS questions that, when answered, will strengthen the body of knowledge related to service composition and decomposition in IoT and CPS. Identified gaps will be addressed through the SLR methodology in **Chapter 4**.

b) Service composition frameworks and platforms, with the main objective of identifying weaknesses in service composition foundations, semantics, and

formal verification components. Identified gaps will be addressed by proposing a new IoT and CPS Capabilities framework in **Chapter 5**.

In **Chapter 4**, we express formal, technical, and QoS questions through the gaps identified in Chapter 3 Section (3.1); we answer those questions based on the SLR methodology and primary studies obtained from trusted research databases.

In **Chapter 5**, we leverage gaps identified in Chapter (3.2) to propose a novel IoT and CPS capabilities composition framework (ICCF). A comparative study is conducted to determine the main pillars of ICCF, including foundations, semantics, and formal verification techniques. An implementation and assessment platform that we called IoTCaP is proposed to implement the ICCF pillars and help researchers and engineers craft IoT and CPS capabilities in different domains of interest, including smart buildings, smart transportation, and smart health environment domains such as Intensive Care Units (ICU).

Chapter 6 leverages the ICCF foundations and the proposed IoTCaP service composition platform -which implements the ICCF foundations- to propose, describe, model, verify, implement, and assess smart cities composite capabilities in three domains:

a) Well-being as a composite capability in the smart building domain.

b) Manoeuver Safety as a composite capability in the smart transportation domain.

c) Health Improvement as a composite capability in the smart health domain.

In **Chapter 7**, we summarize our work and highlight ongoing and future work.

Chapter 2

Introduction

In the previous chapter, we defined the context of this work: smart city capabilities in different domains can be composed -within the same domain or across multiple domains- to create novel services or be decomposed for reuse purposes.

We explained the nature of the gaps we address in this work (knowledge gaps and implementation/platforms gaps), and we summarized the work plan we adopted to tackle those gaps.

In this chapter, we define jargon, concepts, and elements that will be instrumental in understanding the topic of IoT Capabilities Composition and Decomposition. These concepts include capabilities in the context of IoT or CPS, classes of capabilities from a measurability perspective, and composition and decomposition functions. We leverage a layered architecture to better represent and understand the definitions and concepts we introduced.

2.1 What's IoT or CPS ?

According to [239], the Internet of things (IoT) refers to a network that connects various types of devices and services with the Internet via communication protocols with the goal of conducting information exchange and communications to achieve smart functions such as recognition, positioning, tracing, monitoring, and administration. An IoT device is typically a device that's connected to the IoT network or the Internet in a broad sense. On the other hand, Cyber-Physical Systems (CPS) are -according to [50]- the next step into globally integrated software systems, as they exhibit the result of combining embedded systems with cyberspace or the Internet. Characteristics of Cyber-Physical Systems include support of real-world awareness and access to global data and services by embedded systems. IoT and CPS can be used interchangeably when discussing connected devices and services, and they share a lot of characteristics and challenges, such as real-time constraints, functional safety, security, dependability, and quality of service.

2.2 What's an IoT capability ?

A capability of an Internet of Things (IoT) device [30], or a Cyber-Physical System (CPS) -terms used interchangeably [110][53][124][241][143]- can represent a simple feature, service, or measurement (temperature, humidity, pressure, etc.). IoT

capabilities can also refer to full-fledged complex applications provided by one or multiple devices interconnected at the network layer and orchestrated at the application layer to generate a value-added feature, such as thermal comfort in a smart building [267].

We define in the following sections capabilities categories and explain -using a layered architecture- important concepts related to capabilities composition and decomposition operations.

2.3 Capabilities Classification:

Capabilities can be classified into two categories based on how their features can be measured.

i) Tangible capabilities: provide a quantitative metric, such as temperature or round-trip time [287]. Tangible capabilities are usually a characteristic of basic/atomic services such as sensors or network probes. Keeping an open mind when defining what an atomic or a composite service is, is necessary because some full-fledged services and applications can be defined as atomic in the context of an even more complex system.

ii) Abstract capabilities: are typically complex services not measured using traditional units. These abstract capabilities are recognized across multiple domains, including smart cities (quality of life [45]), IoT infrastructures (scalability [5]), and smart transportation systems (traffic jam prediction accuracy and driver risk level [310]). Abstract capabilities measurements are often a user or programmer-defined in a way that simplifies and makes their assessment intelligible.

2.4 Composition and Decomposition Functions:

IoT capabilities composition is the art of aggregating existing IoT capabilities in order to come up with a novel service with added value [87], whereas decomposition aims to reuse a subset of the capabilities of a complex service or distribute its computation [267]. A general diagram is provided in Figure 2.1, which shows how IoT capabilities are modeled, composed, or decomposed.

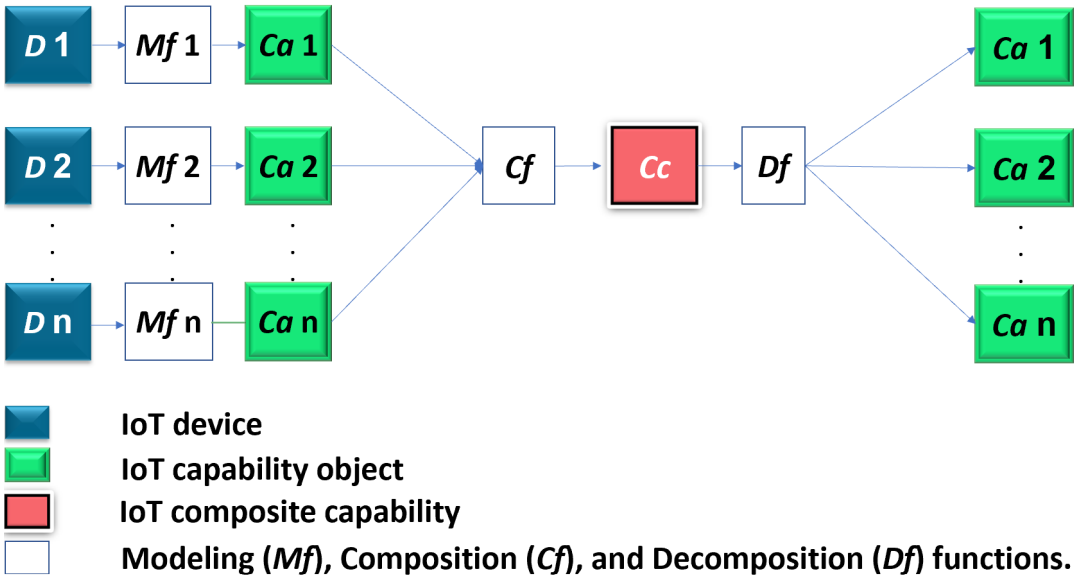

FIGURE 2.1: An illustration of IoT capabilities modeling, composition, and decomposition functions

To formally explain the different concepts of the topic, let us consider the example of IoT capabilities $Ca1$, $Ca2$, and $Ca3$ and two functions, Cf, and Df, representing IoT capabilities composition and decomposition functions, respectively. $Ca1$, $Ca2$, ..., Can the atomic capabilities that generate the composite capability Cc. Cc can be considered atomic in the context of a service or an application with a higher level of complexity. In [67], single-function components that are reusable by other city services are packaged and published as standalone components or atomic services, considered as single functional blocks that consume data and implement a feature, such as managing, enriching, or filtering input, and are similar to the concept of a microservice in terms of being a reusable, self-contained piece of software targeting a specific task. In the same effort [67], eight atomic services addressing smart city challenges in data analytics, evaluation, integration, validation, and visualization were pointed out (parking data prediction atomic service, traffic flow predictor; 2 atomic visualization services; 1 data elaboration atomic service; and 3 data transformation atomic services). Functions performed on atomic capabilities include the composition and decomposition functions Cf and Df, which can be synchronous [84], asynchronous [315], serial or sequential, probabilistic or alternative, parallel, circular, or cyclic [29] [28]. The success of Cf and Df requires multiple processes, including computations, filtering, ranking, composing, and verifying atomic and composite features [84].

Let us consider Figure 2.2: $a1$ and $a2$, two applications (e.g., hiking recommendation application and traffic condition application) that rely on a composite capability $Cc2$ (e.g., $Cc2$ provides weather information necessary to decide whether or not to travel/hike). $Cc2$ is a composition of multiple atomic capabilities(e.g., $Ca3$ provides temperature information, $Ca4$ predicts precipitation, and Cak predicts the fog level). Composition is a bottom-up process that leverages lower-level atomic capabilities to build value-added capabilities that can be composed to create even more complex capabilities or applications. Decomposition is a top-bottom process that analyzes

existing composite capabilities components to determine reusable functions or distribute computation. Dotted lines in Figure 2.2 trace composition and decomposition paths. Figure 2.3 and Figure 2.4 highlight a high-level, step-by-step approach to the composition or decomposition of IoT/CPS capabilities.

2.5 A Layered model for IoT Capabilities Composition and Decomposition

Researchers in [289] organized service composition into three layers: physical, virtual object, and service composition. This model is representative of the composition/decomposition problem because it does not include networking complexities. In [28], the authors based their composition survey on a layered architecture that also considers network and application aspects. In [286], the proposed layered approach, which represents the journey of a capability message, has three levels: i) an information level where the message parameters and temporal scope are defined; ii) a representation protocol level where the message is serialized as a JSON object and made ready for composition by a composition engine; and iii) a session layer where the composite capability is securely delivered to a client or service. A smart home composition framework was proposed in [171] [169], which uses the Majord'home platform as SDN middleware between the data plane layer, where IoT sensors objects reside, and the service composition layer. Other layered models for the composition and decomposition of IoT capabilities were discussed in [310] [5][226].

Based on the aforementioned efforts, we proposed a layered architecture that focuses on composition and decomposition operations (Figure 2.2), which illustrates a layered, high-level, and hierarchical architecture within which devices, capabilities, applications, and required functions to transition from one layer to another -Modeling Mf (turning devices capabilities into composition-ready objects or data models), Composition Cf, and Decomposition Df- are highlighted.

The bottom layer is the **Devices**, which provides raw and non-composition-ready capabilities. The next layer is the **Capability** model or object layer, where the atomic capabilities are composition-ready using different data models. The third layer is the **Composite Capabilities** layer that incorporates value-added services composed of aggregated capabilities. These composite capabilities are generated using processes or engines that leverage composition-ready capabilities models to aggregate atomic capabilities and provide novel and composite features which can be further composed into more complex entities or **Applications** (e.g., $a1$, $a2$, ..., an in Figure 2.1), which represent the fourth or upper layer. These applications can be decomposed into less complex or atomic capabilities, or their computation can be distributed across multiple nodes [14]. **Composite capabilities** can also be decomposed into atomic capabilities via decomposition processes.

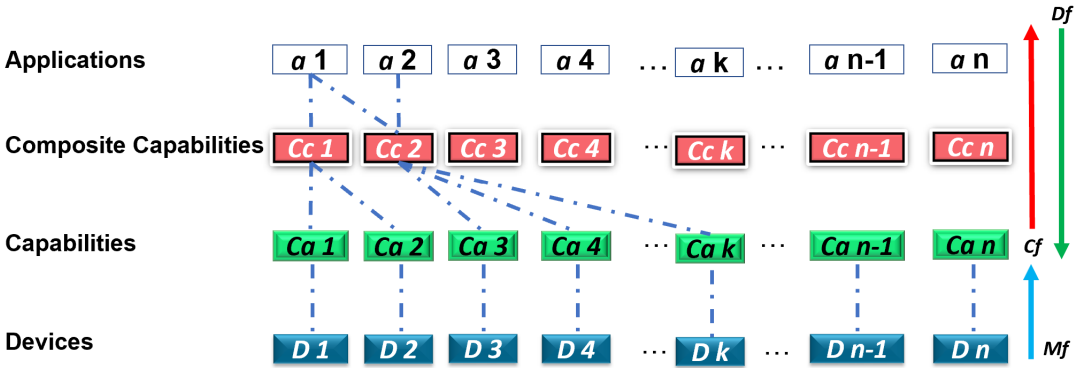

FIGURE 2.2: Modeling, composition, and decomposition layers and functions

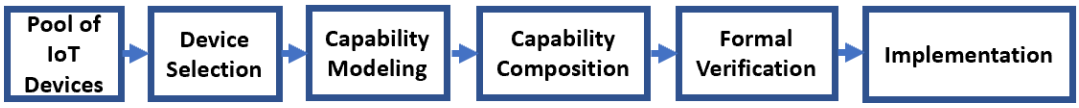

FIGURE 2.3: Composition of atomic services into composite services.

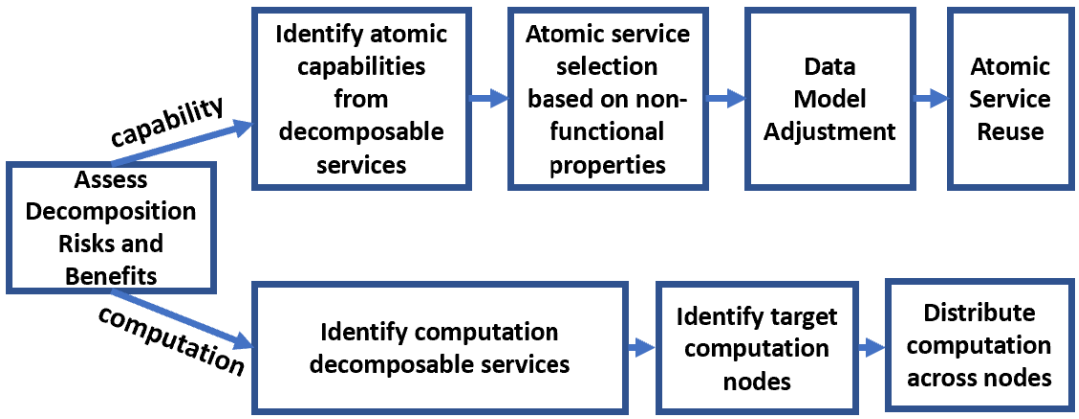

FIGURE 2.4: decomposition of composite services into atomic capabilities.

In this chapter, we have introduced concepts related to the topic of IoT capabilities composition and decomposition. In the next chapter, we provide related work to identify:

a) Formal, Technical, and QoS Gaps in scientific questions within SLRs and literature reviews. The goal is to discuss these gaps through the SLR methodology in a way that adds to the body of knowledge and helps future readers, and researchers better understand the topic of IoT and CPS service composition and decomposition

b) Gaps in foundations, semantics, and formal verification techniques in service composition frameworks and platforms. The goal is to propose a novel service composition framework that fills out those gaps and enables the composition of novel capabilities in different smart city domains.

Chapter 3

State of the art and gaps in IoT service composition SLRs, Surveys, Frameworks, and Platforms

After introducing the definitions and concepts related to IoT capabilities service composition and decomposition, we provide in this chapter the state-of-the-art related to:

1) Systematic literature reviews and surveys: The goal is to find gaps in SLRs and literature surveys and craft questions based on these gaps to be discussed and answered through the SLR methodology in Chapter 4.

2) IoT and CPS capabilities composition frameworks and platforms: The goal is to identify gaps in modeling foundations, including reference architectures, formal and knowledge aspects, and propose a service composition framework that will fill those gaps in Chapter 5.

3.1 Systematic Literature Reviews (SLRs) and Surveys: State of the art and identified gaps

In this section, we discuss previous SLR studies and Literature reviews that addressed an aspect or more of the topic of IoT capabilities composition and decomposition. Based on the discussion of each SLR and literature review, we revealed components for research questions (RQ) that received little to no attention in previous efforts (see Table 3.1).

One major aspect that we identified as a lacking component in existing SLRs and literature reviews is a general and comprehensive taxonomy that organizes the different aspects related to IoT and CPS capabilities composition and decomposition. We consider this as one of the items we contributed to in this work.

We organized the different aspects related to studying service composition into three aspects: **Formal**, **Technical**, and **QoS**. In Chapter 4, we explain in detail the reasons behind adopting such taxonomy by comparing it against existing taxonomies.
.

The topics discussed in previous SLRs and literature reviews will be discussed according to the taxonomy proposed in Chapter 4, i.e., topics will be placed under three buckets: **Formal**, **Technical**, and **QoS**.

Regarding the **Formal** aspect, many previous works tackled corresponding sub-aspects, including IoT composition standards and frameworks [21][262][227], composition algorithms (heuristic, meta-heuristic, exact, hybrid) [135], formal verification tools, techniques, and properties verification such as correctness [271][270] or security [129]. However, none of the SLR surveys addressed the motivation behind native support for composition or decomposition guidelines and mechanisms by standards, frameworks, and architectures (RQ1). In addition, previous SLR questions did not comprehensively explain the main properties of formal representations in service composition from a modeling and formal verification perspective (RQ2). Discussing recent formal trends in service composition, including the properties type, formal modeling approaches, formal verification tools, and implementations, lacks a comprehensive outlook (RQ3), and the question of state space explosion and how it was tackled and to what extent it was solved was not discussed in previous efforts (RQ4). We put service composition sub-aspects that relate to standards, frameworks, architectures, and formal verification techniques and challenges under the Formal aspect as they relate to the formalization, knowledge, background, and foundations basics of the topic.

For the **Technical** aspect, the sub-aspects previously discussed in other SLRs and literature reviews include service composition in the cloud [295], and industrial environments [126], composition service types, attributes, domains of application [132], composition planning and strategies [144] [162] [225], composition platforms [262], and composition models, techniques, and tools [252][184]. However, some Technical sub-aspects weren't addressed, including stakeholders' concerns regarding service composition (RQ5). Similarly, no SLR question comprehensively discussed the nature of composition platforms and how composition implementations differ within these different categories of platforms (RQ6). In addition, the relationship between composition automation levels and composition process types was not highlighted in previous SLRs (RQ7). Similarly, the role of communication protocols in composing services at different layers of IoT environments needs a discussion (RQ8). A comparative study of the different roles a data model performs in service composition is also lacking (RQ9). From a measurability perspective, composite services typically reflect a metric that is difficult to assess using conventional metrics, and this aspect also needs a discussion (RQ10). Similarly, the decomposition of services for reuse or resource optimization has never been surveyed (RQ11). Finally, the role of artificial intelligence and machine learning (AI/ML) in capabilities composition, either in the composition process or the nature of capabilities themselves, was not surveyed (RQ12).

Regarding the **QoS** aspect, different SLRs addressed key QoS questions under different themes, including functional and non-functional properties. For example, the availability of composite services was studied in surveys [271] [144] [126]. The cost was studied in [271] [144] [126] [295], time-related QoS questions, including execution time, response time, and latency, were addressed in [271] [295] [144]

[126], reliability and reputation were both discussed in [144] [126], and scalability was addressed in [295],[162],[126]. Unique QoS properties that were addressed in SLR questions in previous surveys include performance parameters such as completeness, distribution, dynamicity, level of automation, maturity, QoS awareness [162], efficiency, and optimization [295], and security and throughput [144]. By looking at what was covered in previous SLR efforts as well as the existing literature, we identified QoS questions that were not addressed previously; this includes scalability challenges and solutions when composing capabilities in the IoT or CPS space (RQ13). Although the authors in [28][162] discussed which composition efforts provided high, low, or no scalability, they did not address the challenges that face composition approaches to satisfy increased levels of scalability (in terms of latency, computation, etc.). Similarly, in [271] and [126], ensuring a high level of scalability while maintaining a low response time and low verification cost are examples of scalability challenges for service composition that need to be addressed. Another significant aspect that was not given a comprehensive assessment was the interoperability challenges and solutions (RQ14). Different efforts addressed specific areas of the interoperability question, which is the case with SLR [132], which mentioned some interoperability challenges, including differences in network protocols, data models, and service types. SLR [132] also defines a fully interoperable composition as "service type heterogeneity." Similarly, SLR [28] suggested open-source frameworks and a dynamic service composition ensuring interoperability. SLR [271] addressed formal verification challenges related to the interoperability of to-be-composed IoT capabilities. SLR [225] mentioned the integration, selection, and discovery of services as challenges to interoperability. By answering (RQ14), we provide a consolidated response to interoperability challenges and solutions in one discussion. Finally, an aspect of crucial importance in the age of data sharing is the privacy challenges and solutions when composing services. Although many studies have addressed this concern from a composition perspective [82] [192], none of the SLR surveys addressed privacy-related service composition questions. We address this aspect in RQ15, try to understand how different research efforts improve privacy in terms of service composition and explain how new technologies, such as blockchain, can be leveraged to improve privacy.

Table 3.1 aggregates SLR and literature reviews that tackled IoT and CPS capabilities composition. We classified these efforts based on the survey type (SLR or literature review), the year when the research was conducted, the topics covered in the survey, the covered period of the study, and strengths as well as gaps in each study that inspired the SLR questions in this work that we will discuss and answer in Chapter 4.

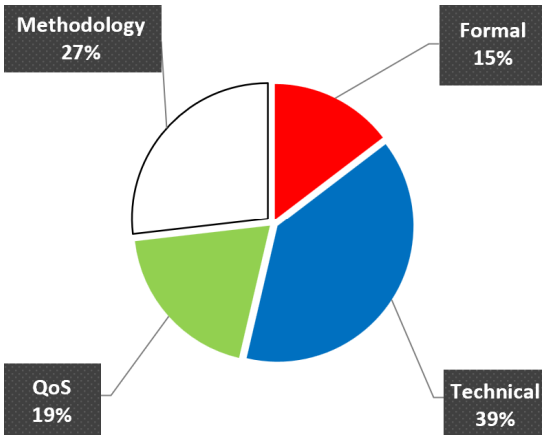

FIGURE 3.1: Distribution of previous SLRs questions per the proposed taxonomy aspects and the SLR methodology.

3.2 Service Composition Frameworks and Platforms: State of the art and identified gaps

After discussing surveys and systematic literature reviews in service composition, as well as the gaps left by these studies, we address in this section the gaps in service composition platforms and frameworks. We focus on gaps related to composition foundations, modeling semantics, and formal verification aspects as these components represent the initial phases of IoT or CPS capabilities service composition, as they precede the deployment and implementation of composite capabilities. Identifying gaps and weaknesses of existing IoT and CPS capabilities composition frameworks and platforms would help in understanding the requirements for future composition frameworks and filling related constraints in terms of accessibility to end users, developers, and city planners, as well as meeting comprehensiveness and trustworthiness requirements.

Composing IoT or CPS capabilities in the smart city domain requires following guidelines and best practices that satisfy different stakeholders' concerns and criteria. These concerns and criteria include making sure users, developers, or city planners can easily craft composition models and translate their composition ideas into a semantic model, which can later be semantically specified and verified for potential implementation.

Examples of IoT and CPS frameworks or platforms for capabilities composition include OneM2M environment [313]: it leverages the NIST CPS framework [112] [113] [307] to define composition guidelines -including those relative to time synchronization between atomic capabilities-.

The OneM2M environment can use the M2M semantics provided by the industry segment that uses corresponding data. This makes its semantics domain-specific [238]; a higher abstraction layer might be needed to simplify the rapid prototyping

TABLE 3.1: Related work: Literature and SLR studies that tackled the IoT/CPS capabilities composition topic.

Ref	Type	Year	Topic	Period	Strengths (✓) and Gaps (x)
Our SLR Study	SLR	2022	Formal, Technical, and QoS aspects of IoT Capabilities Composition and Decomposition	2006-2022	(✓) A new reference taxonomy based on Formal, Technical, and QoS aspects. (✓) Based on gaps in existing research, 15 formal, technical, and QoS questions were identified and answered. (Check Table 4.1 for motivations behind each SLR question). **(x)** Check gaps in Chapter 4).
[28]	SLR	2018	Functional and non-functional service composition properties	1993-2018	(✓) Multiple formal, technical, and QoS sub-aspects addressed. **(x)** QoS: Scalability challenges related to increased latency and computation burden not addressed. **(x)** QoS: privacy was considered a challenge and was left for future work.
[132]	SLR	2019	Service composition in interoperable and heterogeneous environments.	1992-2019	(✓) Strong focus on service composition interoperability challenges in specific service types (REST, SOAP, EOS). **(x)** The SLR question does not answer how data and protocol interoperability challenges in service composition have been addressed or resolved in previous efforts.
[21]	Literature Review	2019	Composition types, models, standards, and QoS sub-aspects.	1996-2018	(✓) A comparative study of service composition approaches. **(x)** Doesn't explain why composition foundations are fundamental when proposing an IoT platform or composite service.
[271]	SLR	2018	Formal verification role in assessing service composition correctness.	1999-2018	(✓) Formal sub-aspects in service composition addressed. **(x)** An SLR question to address the state space explosion problem in service composition is missing.
[295]	SLR	2017	Cloud services composition	2003-2017	(✓) Focus on cloud composition technologies. **(x)** Compositions at the Edge, Fog, SDN and simulation platforms were not addressed.
[129]	Literature Review	2020	Formal verification of IoT protocols	1976-2020	(✓) Focus on verifying the security of compositions and tools leveraged for this end. **(x)** Missing challenges and solutions to the state space explosion problem when formally verifying complex systems with a large state space.
[284]	Literature Review	2017	Interoperability approaches in the IoT application layer	2009-2016	(✓) Application layer composition standards and frameworks compared **(x)** Focus on interoperability aspects (properties, behavior, semantics, message, protocol.), but a discussion around solutions to interoperability challenges is missing.
[227]	Literature Review	2020	SOA Capabilities composition and formal specifications.	2003-2020	(✓) Explaining the differences between SOA service composition languages. **(x)** An architecture agnostic comparison cloud has been more comprehensive (SOA, REST, ..).
[270]	Literature Review	2019	Formal verification approaches for composed IoT services.	2015-2019	(✓) IoT composite services correctness verification was the focus of the study. **(x)** state space explosion issue concerns and solutions not addressed.
[252]	Literature Review	2021	Comparison of enterprise service composition models in IoT	1992-2020	(✓) Composition techniques, models, and tools were highlighted and compared. **(x)** Formal sub-aspects and measurability/assessment of QoS metrics are not addressed.
[144]	SLR	2015	QoS-aware web service composition	2005-2015	(✓) A comprehensive study of QoS-Aware service composition algorithms (heuristic, meta-heuristic, etc.)
[162]	SLR	2022	Web service composition	1994-2021	(✓) Focus on Technical sub-aspects of web-service composition. **(x)** Formal sub-aspects are missing.
[126]	SLR	2022	Service composition methods in cloud manufacturing systems	2008-2021	(✓) Focus on Technical sub-aspects in composing cloud manufacturing capabilities. **(x)** Formal and measurability aspects are missing.
[262]	Literature Review	2014	Web services composition	1997-2014	(✓) A summary of standards, prototypes, and web-service composition platforms. **(x)** Motivations for native support of composition by standards is missing
[184]	Literature Review	2015	Web services composition tools and techniques	1974-2015	(✓) Overview of service composition techniques, technologies, and tools. **(x)** Technical aspects, including the service decomposition role of AI/ML, are not discussed.
[225]	SLR	2021	QoS-Aware Service Composition	2008-2020	(✓) Focus on the technical and QoS aspects related to Hybrid meta-heuristic composition algorithms in SOA. **(x)** Formal sub-aspects not covered.

of composition for different IoT and CPS domains.

Fiware [244] was coupled with the IoT-A framework, which supports IoT capabilities composition and semantic specification using Business Process Model and Notation (BPMN) 2.0; it also supports powerful features such as synchronous and asynchronous compositions [40]. The BMPN semantics, however, make it challenging to formally verify compositions as that involves converting the BPMN notation to the Generic Property Specification Language (GPSL formal) specification language, which can improve expressiveness but might add complexity, impact performance or limit expressiveness when converting BMPN to a graphically verifiable model such as Property Sequence Chart (PSC) [51].

For the CIM (Context Information Management) environment [190], the foundations for composition are provided by the CIM NGSI evolution framework; it uses RDF (Resource Description Framework) to semantically describe the capabilities of a system. RDF is a graph-based descriptive language; it can be converted to a formally verifiable specification such as ShEx 2.0 (Shape expression schemas v2.0) [48]. ShEx expressions can be used both to describe RDF and to automatically check the conformance of RDF data; however, ShEx checks whether RDF data respects the schema requirements as it is data-oriented, not composition function-oriented, which makes it challenging to model checking the system features.

VITAL is another project that supports the IoT-A framework, W3C SSN semantics, but recommendation on formal specification and verification languages and tools to use are not the focus of the framework [157]. The same case for FogFlow [63], an environment that leverages the NGSI framework for IoT capabilities composition foundations and YAML as a capability descriptor.

AWS [226] is a commercial environment for cloud services; it leverages PlusCal semantics [176] and TLA [175] formal specification techniques to verify the correctness of properties such as fault tolerance in their storage services. Still, this benefit didn't cover functional capabilities of microservices in IoT composition solutions such as GreenGrass [173].

mPlane [286][287] is a network measurement platform that comes baked with service composition capabilities, objects, semantics, components, and operations. The semantics of mPlane are particularly interesting from a service composition standpoint as they are very natural to operations needed for discovering, requesting, composing, and consuming composite capabilities. However, the mPlane platform only mentions formal verification in the context of ensuring the safety of access control (non-functional property) and doesn't recommend formal verification tools or techniques to make sure compositions are correct from a functional standpoint.

The NIST CPS Framework [112][113] does a great job of highlighting the different requirements to think about when it comes to building new IoT or CPS capabilities, including detailed concerns related to composition aspects and concerns (discoverability, reachability, complexity, and constructivity). However, The NIST CPS framework doesn't specify which semantics to use or which formal verification techniques to adopt for building services as it stands as a reference for researchers and engineers to keep in mind key aspects and concerns when building novel IoT or CPS services or systems.

Table 3.2 summarizes the foundations, semantics, and formal verification properties of the discussed composition frameworks and environments.

Based on studying the aforementioned frameworks and platforms, there's a need for an IoT and CPS capabilities platform that meets all the identified constraints and fills all the identified gaps including:

TABLE 3.2: Comparing existing IoT/CPS composition environments, platforms, and frameworks

IoT service composition frameworks and environments	Ref	Composition Foundations and Guidelines	Modeling and Semantics	Formal Verification Languages and tools
OneM2M environment	[313][238]	• Leverages the NIST CPS framework to define the composition guidelines.	• M2M Domain-specific semantics provided by the industry segment.	• Verification and validation through implementation (Mobius) (not formally verified)
Fiware	[244]	• Coupled with the IoT-A framework	• Business Process Model and Notation (BPMN) 2.0.	• BPMN converted to PSC for formal verification.
IoT Architectural Reference Model (IoT ARM)	[40]	• IoT-A: The Service Organization Functional Group (FG) implements composition functions and mechanisms.	• BPMN 2.0	• N/A.
Context Information Management (CIM)	[190]	• Provided by the CIM NGSI evolution framework	• RDF (Resource Description Framework) is used to semantically describe the capabilities of a system.	• No formal verification. Data validation using ShEx 2.0.
VITAL	[157]	• leverages IoT-A framework for composition foundations.	• W3C SSN	• N/A.
FOGFLOW	[63]	• NGSI framework for IoT capabilities composition foundations	• YAML	• N/A.
AWS Microservices and Storage (S3) load balancing	[226] [173]	• Follows Amazon Web Services design recommendations.	• PLUSCAL is used to bridge the gap between code semantics and formal specifications to be verified	• TLA/TLA+ (from a functional perspective, formal verification wasn't used in IoT microservices functionality verification or in IoT GreenGrass applications)
mPlane Measurement Platform	[79] [286] [287]	• Describes components for building a composition platform for the measurement of network performance (components include clients, probes, supervisors and reasoners, and repositories.).	• mPlane semantics are composition friendly, with operations geared towards requesting capabilities and returning results.	• highlights the importance of leveraging formal specifications to model and verify the safety of access control mechanisms but doesn't provide examples or recommendations for formal specifications or tools.
NIST CPS Framework	[112][113]	• Provides complexity, constructivity, reachability, and discoverability guidelines for IoT and CPS compositions.	• Doesn't recommend specific descriptors or semantics for service description and modeling.	• Doesn't recommend a specific formal verification language or tool but highlights the importance of building verified and trustworthy systems.

a) Comprehensive composition foundations and guidelines.

b) Expressive and composition-friendly semantics for modeling capabilities, operations, and components involved when discovering, registering, aggregating, and serving services to end users.

c) Support for a seamless transition from modeling semantics to formal specifications and verification of composition operations before deployment.

d) An implementation platform that takes into consideration the identified service composition guidelines (a), leverages the expressive semantics when crafting code for novel capabilities composition (b), and allows the testing and assessment of formally verified capabilities(c).

Figure 3.2 summarizes elements to consider when proposing a novel service composition framework that should facilitate prototyping and verification of the novel capabilities while respecting different stakeholders' concerns and requirements from the conception phase to the deployment phase.

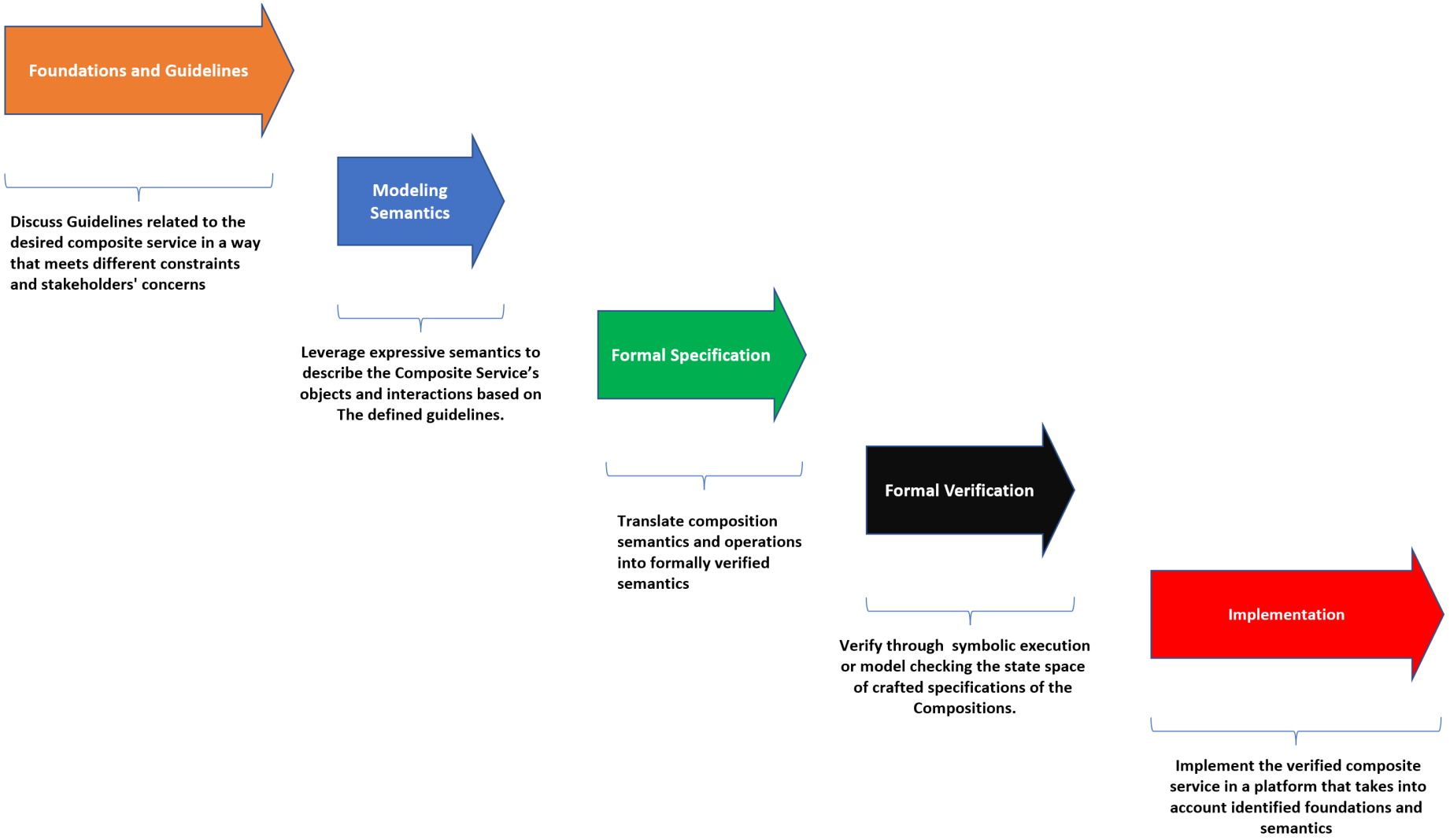

FIGURE 3.2: Requirements for building a trustworthy composition framework and platform.

3.3 Summary and takeaways from related work

In this chapter, we discussed related work and identified gaps in both:

i) Systematic literature reviews and surveys:

The takeaways concluded from Table 3.1 include i) a strong interest in answering specific questions related to the topic (SLR surveys) compared to simply summarizing aspects related to it (Literature reviews), ii) there is a strong interest in the Technical aspect of the topic compared to Formal and QoS aspects and iii) the topic of IoT capabilities composition and decomposition is still trending as it was continuously discussed as early as 1992 until this year (2022) and still attracts the interest and curiosity of researchers. To complete the related work analysis in this section, SLR efforts investigated in Table 3.1 addressed a total of 41 SLR questions. None of the research questions identified (RQs) have been addressed previously. Figure 3.1 shows the distribution of the previous SLR questions based on the aspect under which they fall. The questions related to the methodology were geared toward the SLR methodology itself or were too general to organize under a specific aspect.

ii) IoT or CPS capabilities composition frameworks and platforms.

To address the gaps in these frameworks and provide a robust framework for the composition of IoT and CPS capabilities, an IoT and CPS Composition Framework (ICCF) is proposed in Chapter 5. This framework leverages the NIST CPS framework composition and trustworthiness recommendations, uses strong semantics inspired by the mPlane protocol [286], and relies on the intuitive PlusCAL/TLA/TLA+ package to prototype, formally specify, and model check capabilities and assess their trustworthiness. None of the frameworks mentioned above provides a complete set of powerful foundations, modeling semantics, and formal verification techniques, which should facilitate prototyping, composing, assessing, and verifying novel capabilities in the IoT or CPS space.

To address gaps in i) and ii):

In Chapter 4, we address the identified formal, technical, and QoS gaps in surveys and systematic literature reviews (RQs) by discussing these gaps in-depth and providing answers and discussions based on the systematic literature review methodology.

In Chapter 5, we address identified gaps in service composition platforms and frameworks by proposing a new IoT and CPS capabilities composition framework that meets the foundations, semantics, and formal verification needs -required for prototyping and implementing novel capabilities in the IoT and CPS space- while respecting different stakeholders concerns.

Chapter 4

Answering scientific questions related to Formal, Technical, and QoS aspects in service composition.

In Chapter 3 Section (3.1), we identified fifteen formal, technical, and QoS gap in SLRs and surveys (RQ1 to RQ15). These gaps represent important questions that when answered will enrich the body of knowledge related to the topic of IoT and CPS capabilities composition and will help future researchers and engineers craft and design comprehensive, trustworthy, and reliable services and composition platforms.

In this chapter, we rely on the Systematic Literature Review (SLR) methodology to address the identified gaps in Chapter 3 Section (3.1), to achieve this goal, we follow the organization below :

Section (4.1) discusses the SLR research methodology based on which we provide data and answers to the research questions. This discussion includes the formulation of research questions, explaining the search process and the inclusion/exclusion criteria, assessing the quality of data sources, highlighting the limitations, data extraction and analysis approach, and the execution timeline of the systematic literature review.

Section (4.2) includes two major components of the SLR conduced in this chapter:

i) Proposes a new taxonomy for the different aspects and sub-aspects of the IoT and CPS capabilities composition and decomposition.

ii) Provides through tabulated data information from primary studies that will be leveraged in discussing and answering the formal, technical, and QoS questions. Each table is preceded by a discussion explaining the main elements of the question and the components required for discussing it.

Section (4.3) Answers the identified SLR questions based on the aforementioned primary studies, and highlights trends and gaps in primary studies as well as threats to the validity of this SLR effort.

Section (4.4) Summarizes the SLR study and showcases its benefit to different stakeholders.

Information, discussions , and answers presented in this chapter benefit curious end-users of IoT systems as it explains the role of capabilities composition and decomposition in building value-added services or reusing existing ones for resource optimization or cost reduction. For researchers, it answers critical questions related to the topic's formal, technical, and QoS sub-aspects and indicates the corresponding trends and gaps.

Another benefit for answering SLR questions in this chapter is enriching the discussion around formal, technical, and QpS aspects for the proposed ICCF framework discussed in Chapter 5 and improving its aspects and applications through the discussions and answers provided.

4.1 SLR : Research Methodology

The SLR approach uses an objective research methodology to answer specific research questions based on relevant papers on that topic. SLR reviews require expertise in the domain of study, search in different databases, and require years to produce. However, literature reviews can use subjective research methods to summarize topics using informal approaches. The SLR approach was deemed the most suitable for answering the identified gaps for its comprehensiveness and rigorousness. The guidelines proposed by Kitchenham [165][166] were used, as well as guidelines from the SLR studies in the related work for respecting the SLR methodology: i) formulating the research questions based on the PICOC approach [103][49][277], and ii) explaining the search process while highlighting inclusion and exclusion criteria, iii) performing Quality assessment, iv) discussing the effort limitations, v) explaining the data collection process, vi) explaining data analysis process, and vii) explaining the execution timeline of the SLR.

4.1.1 Formulating the research questions

Based on the gaps and weaknesses of related work, formal (RQ1-RQ4), technical(RQ5-RQ12), and QoS(RQ13-R15) questions -that were not addressed in previous SLR efforts- were pointed out, and the list of these questions was elaborated in Table 4.1 along with corresponding motivations.

4.1.2 Explaining the Search process and the Inclusion/Exclusion criteria

4.1.2.1 Explaining the search process

The research questions (RQs) in Table 4.1 and the corresponding taxonomy aspects and sub-aspects presented in Figure 4.5 are the foundations of the SLR review because they guide the search process by guaranteeing that the selection of primary studies is directly related to the SLR research questions. The search process was performed in 6 stages:

- In Stage 1, the SLR questions and the corresponding taxonomy aspects and sub-aspects are identified.
- In Stage 2, the search databases, corresponding search string, and filtering formula, are selected, as illustrated in Figure 4.1. The search string incorporated

TABLE 4.1: Systematic literature review (SLR) questions (RQs) and corresponding motivations.

Research Questions	Motivations
RQ1: What is the motivation for native support for IoT capabilities composition/decomposition mechanisms by standards, reference models/architectures (RMAs), and frameworks?	• Service composition/decomposition platforms rely on different frameworks, standards, or reference architectures that may -or not- support composition mechanisms and guidelines. • Answering this question would encourage future standard groups to consider composition/decomposition benefits during the research and writing phase and platform builders to adopt standards or frameworks that keep composition and decomposition guidelines in mind during the implementation.
RQ2: What are the main properties of formal representations leveraged in service composition?	• Explaining the role formal representations play in modeling and explaining composite capabilities from functional and non-functional perspectives, and highlighting how formal representations can be leveraged in verifying composite capabilities' formal properties such as correctness.
RQ3: What are the current trends in formal representations modeling and formal verification?	• As there is a wide range of techniques and tools adopted for enabling formal specification and studying different properties, the goal is to make researchers and engineers recognize these elements for potential use in their research efforts or industrial applications.
RQ4: What are the different ways and how effective formal verification techniques and tools tackled the state-space explosion problem in service composition?	• What motivates this question is informing researchers of the different solutions used to solve or minimize the impact of the state space explosion on service composition verification, as the number of states explodes with the complexity that comes with composing different atomic capabilities with a wide range of values and states.
RQ5: What are the different stakeholders' categories and concerns when it comes to composing or consuming capabilities in different domains?	• Different stakeholders deal with service composition challenges from their own perspectives. Developers, Users, City Planners, and Researchers each have their own concerns and expectations. Understanding these concerns from the get-go would help in addressing them and taking them into consideration either while developing new platforms for composite services or when using these solutions by end users.
RQ6: What are the technical differences in capabilities composition implementation in different platforms?	• Composing capabilities in the cloud differs from composing capabilities in the edge or the fog. Responding to this question would enlighten researchers and engineers on which processes or services should be implemented in which composition layer.
RQ7: What are the different composition process types, and how do they differ in terms of automation level?	• Service composition can be synchronous or asynchronous, rule-based, or programming-based, among other process types. These process types are discussed in light of the automation level. Based on existing literature, we check whether automating composition can be better performed under a particular composition process type.
RQ8: What roles do communication protocols play in composing or decomposing IoT capabilities?	• Besides ensuring communication between different components involved in service composition, the other roles communication protocols play -either in improving certain QoS properties or enabling some other capabilities- are discussed.
RQ9: What are the roles of data models leveraged in service composition and decomposition?	• IoT data models differ in terms of expressiveness and complexity, among other properties. Leveraged data models in service composition are highlighted as well as their roles in the context in which they were leveraged. This will help developers make informed choices relative to capabilities data models when building new capabilities in the IoT space.
RQ10: How are atomic or composite capabilities quantified or measured?	• Composite capabilities typically lack conventional methods of measurement or assessment compared with atomic capabilities. Answering this question would inspire and inform researchers about different ways of assessing and measuring the performance or levels of composite capabilities.
RQ11: What are the benefits of building IoT platforms and complex services with decomposition in mind?	• Service composition has been extensively discussed in previous surveys and literature. However, to the best of our knowledge, this is the only study that extensively addresses service decomposition. We explain decomposition flavors and the benefits it brings (including reuse) when services are built with it in mind.
RQ12: What role can AI/ML techniques play in shaping or improving service composition?	• To the best of our knowledge, this is the only survey to address the role of AI/ML in service composition. Two ways in which AI/ML plays a role were identified: improving service composition workflow components (e.g., service selection) or building services with AI/ML capabilities.
RQ13: What are the main scalability challenges and solutions adopted when composing IoT and CPS capabilities?	• Different Technical sub-aspects can either improve or hinder scalability when composing services. Based on existing efforts, these challenges and solutions are revealed to help composite capabilities stakeholders build and use features that scale.
RQ14: What are interoperability challenges and solutions when composing capabilities from heterogeneous environments?	• Composing capabilities requires interoperable data models, network APIs, and synchronized data, among other requirements. Those requirements are exposed to inform IoT platforms builders of interoperability considerations when it comes to composing novel IoT capabilities.
RQ15: What are the main privacy challenges and solutions in service composition?	• Privacy is an end-user concern that is receiving increasing attention, especially with the advent of new standards such as GDPR that impact how IoT systems should be built to address privacy [31]. We identify privacy concerns in service composition as well as the role of new technologies or best practices, such as the blockchain or regulations, in addressing these concerns.

initial inclusion and exclusion criteria.

• In Stage 3, we ran the SCOPUS search script, which focused on the title, abstract, and manuscript keywords, which yielded 2805 manuscripts.

• In Stage 4, the search results from Stage 3 are narrowed by applying different filtering keywords in column **Population** from Table 4.2. For example, QoS/privacy-related primary studies were extracted by adding different related keywords (privacy, private, etc.) to the search string. Performing filtering on the different aspects and sub-aspects resulted in 553 manuscripts. Although some papers contained keywords related to the research questions, 503 manuscripts were excluded as they did not sufficiently answer the SLR RQs, leaving only 50 primary studies that substantially addressed one or more RQ in a specific paragraph or as the main topic of the manuscript.

TABLE 4.2: Criteria, Primary Studies, and scope of the SLR: **P**opulation, **I**ntervention, **C**omparison, **O**utcome, **C**ontext (**PICOC**).

Criteria	Primary Studies	Population	Intervention	Comparison	Outcome	Context
RQ1	Check Table 4.3 Column **Ref**	Formal: Frameworks, Architectures, Standards	Classifying and characterizing IoT service composition and decomposition efforts based on a taxonomy, and answering key questions related to the taxonomy aspects based on relevant literature by extracting and discussing data.	A comparison study connecting primary studies in the IoT/CPS service composition and decomposition topic to come up with answers to important formal, technical, and QoS questions	Classifying, comparing, and analyzing data from different research efforts to answer the key questions identified. Trends, gaps, and Future work will also be discussed.	The systematic literature review approach is leveraged to investigate and consolidate resulting primary studies in IoT or CPS service composition within the context of 3 aspects : Formal, Technical, and QoS.
RQ2	Check Table 4.4 and Table 4.5 Column **Ref**	Formal : Formal Representations				
RQ3	Check Table 4.4 and Table 4.5 Column **Ref**	Formal : Formal Verification Techniques, aspects, applications, tools				
RQ4	Check Table 4.6 Column **Ref**	Formal : State Space Explosion				
RQ5	Check Table 4.7 Column **Ref**	Technical : Stakeholders				
RQ6	Check Table 4.8 Column **Ref**	Technical :Composition Platforms				
RQ7	Check Table 4.9 Column **Ref**	Technical : Composition Processes and Automation				
RQ8	Check Table 4.10 Column **Ref**	Technical : Communication Protocols				
RQ9	Check Table 4.11 Column **Ref**	Technical : Data Models				
RQ10	Check Table **??** Column **Ref**	Technical : Measurability				
RQ11	Check Table 4.12 Column **Ref**	Technical : Decomposition				
RQ12	Check Table 4.13 Column **Ref**	Technical : AI/ML				
RQ13	Check Table 4.14 Column **Ref**	QoS : Scalability				
RQ14	Check Table 4.15 Column **Ref**	QoS : Interoperability				
RQ15	Check Table 4.16 Column **Ref**	QoS : Privacy				

• In Stage 5, more relevant manuscripts were included using the forward and backward snowballing techniques based on the 50 primary studies in Stage 4 to find more answers to the research questions, which required reading the full text of these publications instead of focusing on the abstract, title, or keywords, which added 103 more manuscripts.
• In Stage 6, 29 additional manuscripts were included using a manual search in the Google Scholar database for completion.

Figure 4.4 highlights the search stages: the 182 primary studies are highlighted in the **Ref** column from Table 4.3 to 4.16, with some primary studies providing answers to more than one RQ. The primary studies for each RQ are highlighted in Table 4.2 column **Primary Studies**. For more details about the selected primary studies, readers can refer to [160], where we put together the list of primary studies, as well as related information (year of publication, the source database, how each study was extracted (Main Search, Snowballing, Manual), publisher, as well as its role in answering a Research Question (RQ)). For the other referenced material in this chapter, publications and links mentioned when introducing certain concepts, web sources, and Git repositories are not counted as primary studies, but they are components for explanation and completion.

4.1.2.2 Inclusion Criteria

From a content perspective, the inclusion criteria require:
• Relevance to the 15 formal, technical, and QoS research questions or the taxonomy sub-aspects.
• The manuscript addresses an RQ or taxonomy sub-aspect as the main component or at least in a specific section/paragraph.
• The manuscript exists in the SCOPUS or Google Scholar Database.

4.1.2.3 Exclusion Criteria

• SCOPUS main search: non-peer-reviewed publications and document type limitations: sources that are not Conference Papers (cp), Articles (ar), Book Chapters (ch), or Books (bk).
• Google Scholar Manual Search: respected the same criteria as in the SCOPUS main search while tolerating a few important technical reports.
• All databases: document Type limitations: MS or PhD dissertations, white papers, SLR and Literature reviews, documents that are not in the field of Computer Science, Engineering, or Mathematics, and manuscripts written in languages other than English.
• All databases: the full text of the candidate primary study does not provide sufficient information to allow classification of the studied sub-aspect properties.
• All databases: for manuscripts selected manually or using snowballing, the full text of the candidate primary study could not be obtained by contacting the authors or other means.

(A) : SCOPUS Search String

```
TITLE-ABS-KEY ( ( ( service* ) OR ( capabilit* ) OR (
feature* ) OR ( function* ) ) AND ( ( composition* ) OR (
aggregation* ) OR ( decomposition* ) ) AND ( ( iot ) OR (
internet AND of AND things ) OR ( cps ) OR ( cyber AND
physical AND system* ) ) ) AND ( LIMIT-TO ( DOCTYPE , "ar" )
OR LIMIT-TO ( DOCTYPE , "cp" ) OR LIMIT-TO ( DOCTYPE , "ch" )
OR LIMIT-TO ( DOCTYPE , "bk" ) ) AND ( LIMIT-TO ( SUBJAREA ,
"COMP" ) OR LIMIT-TO ( SUBJAREA , "ENGI" ) ) AND ( LIMIT-TO (
LANGUAGE , "English" ) )
```

(B) : Google Scholar Search String

```
( ( ( service* ) OR ( capabilit* ) OR ( feature* ) OR (
function* ) ) AND ( ( composition* ) OR ( aggregation* ) OR (
decomposition* ) ) AND ( ( iot ) OR ( internet AND of AND
things ) OR ( cps ) OR ( cyber AND physical AND system* ) ) )
```

(C) : Filtering sub-aspects or RQs

```
(Search String ((A) OR (B)) (AND) (RQ Sub aspect Keywords)
```

FIGURE 4.1: Search Strings and filtering method.

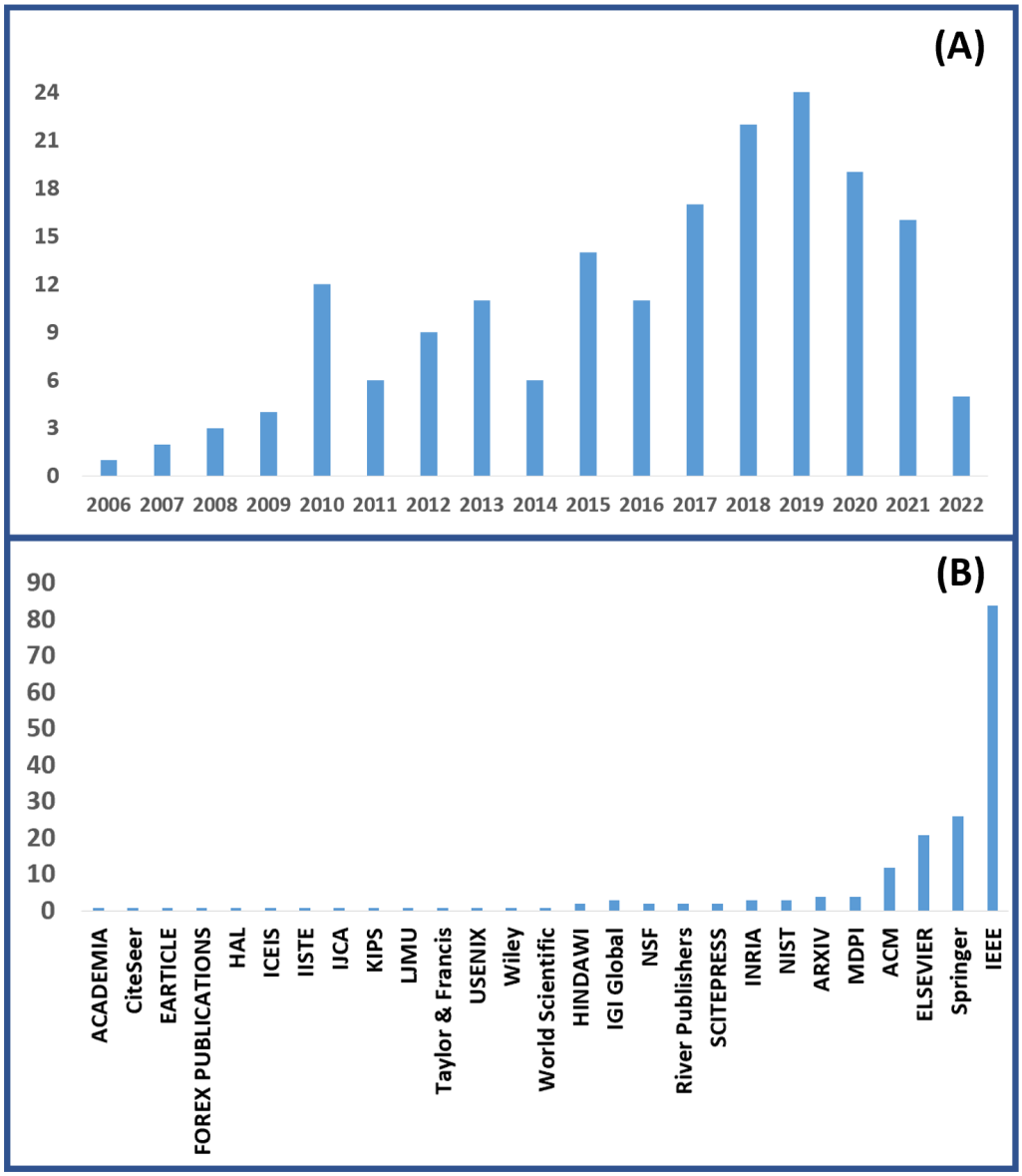

FIGURE 4.2: Count of primary studies per year and per publisher.

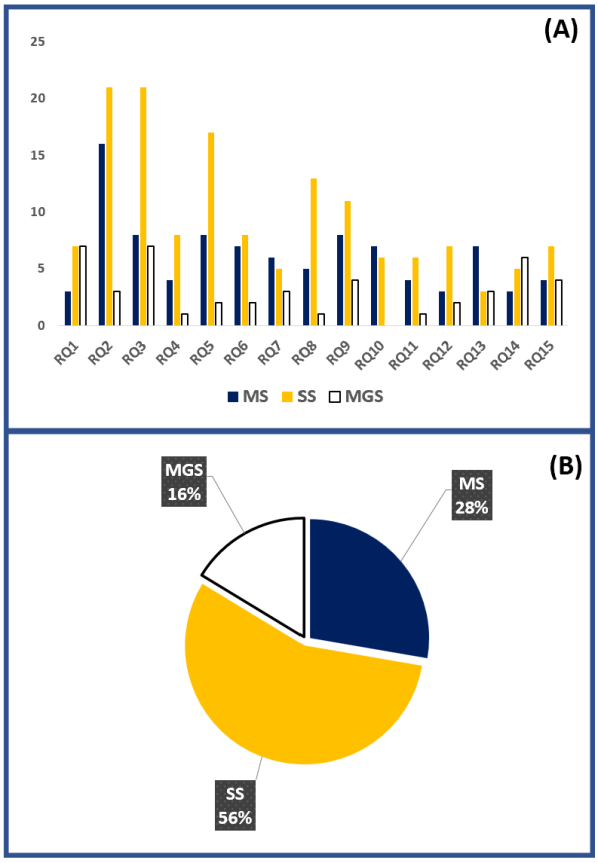

FIGURE 4.3: Distribution of primary studies per search method: Main SCOPUS (MS), Snowballing SCOPUS (SS), Manual Google Scholar (MGS).

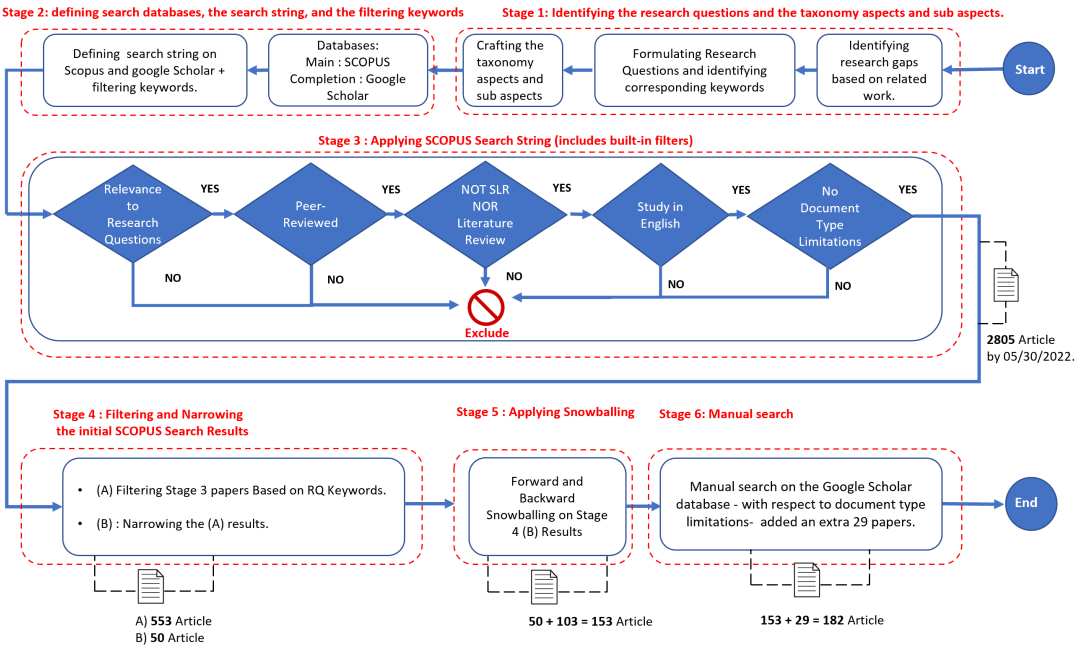

FIGURE 4.4: Multi-stage selection process with the inclusion/exclusion criteria

4.1.3 Quality assessment

The six steps followed in the research process were intended to ensure that only relevant and high-quality manuscripts were selected as primary studies. Techniques to ensure quality include full-text reading, forward and backward snowballing, and manual selection of relevant papers that add value to the RQs answers. We were inspired by the quality assessment elements proposed in IEEE Access SLR [225] to ensure that the selected articles respect a minimum quality threshold of 75% of the criteria below :

i) Validating the data source: Queried databases and journals are well-known and trusted by the research community through indicators such as the impact factor.

ii) Relevance to the research domain (IoT and Cyber-physical Systems).

iii) Presence of substantial information: The sole presence of RQ keywords doesn't imply inclusion.

iv) Primary studies selected provide solid contributions that address the SLR objectives.

To ensure that the SLR is inclusive, efforts that do not originate from well-known -but genuine and trusted- publishers were included; only 16% of the primary studies were obtained using a manual search on the Google Scholar database to ensure that the majority of primary studies were systematically selected while guaranteeing a level of quality and completeness by including manually selected -and RQ relevant- manuscripts. Although important, the number of citations was not taken into consideration as an exclusion criterion as that would discriminate against high quality or newer publications that might have important data; the same reasoning applies to the year of publication of the manuscripts, which could limit the scope of the study with no concrete benefit. The resulting primary studies were published between 2006 and 2022, with more than 70% being published between 2015 and 2022, and for each SLR question, the majority of primary studies that address them span this period which would reveal the latest advances in the topic and provide up-to-date answers to the identified SLR questions.

Figure 4.2 (A) shows the number of publications per year; this distribution shows that more than 70% of scientific publications on this topic were published after 2015; hence, the continued relevance of the topic in recent years.

Figure 4.2 (B) shows the count of primary studies publications per publisher, primary studies from less-known publishers -representing 21% of primary studies- to achieve a higher level of completeness and to account for the importance of these studies, while 79% of the manuscripts were extracted from well-known publishers (ACM, ELSEVIER, IEEE, and SPRINGER) to guarantee a high level of quality.

Figure 4.3 (A) accounts for primary studies cited in more than one RQ and shows the distribution of primary studies search methodology per RQ: Main (MS), Snowballing (SS), Manual (MGS).

Figure 4.3 (B) highlights the percentage of the primary studies search methodology: 84% of the primary studies were either obtained using the direct search string

and filtering keywords or the snowballing technique, and all the results of the main search or the snowballing search are indexed in the SCOPUS database.

4.1.4 Limitations

The limitations of this SLR are as follows:
• Not including manuscripts ruled out by exclusion criteria.
• Some formal, technical, and QoS sub-aspects (referred to in the taxonomy as Other Formal, Other Technical, and Other QoS) are not discussed (out of scope)
• Databases queried (Only SCOPUS and Google Scholar).
• Some papers were not probably considered due to human error while generating results using the search strings or while selecting papers.
• Although we strongly believe that we extensively covered the studied sub-aspects, relevant papers after May 2022 might have been missed due to the consolidation phase.

4.1.5 Data Extraction

Data in the Results section were extracted from 182 primary studies; the methods used included filtering on the SCOPUS and Google Scholar databases. Keywords leveraged for extraction include those related to research questions but also : (i) title, (ii) names of authors, (iii) year of publication, (iv) Publication venue and related quality index (v), and approaches, criteria, and parameters for each sub-aspect or research question.
Extracted data were placed in tables by referencing the primary studies, as well as other columns, to classify and compare the different studies based on each RQ requirement. For accuracy purposes, the extracted data were reviewed by the main author and agreed upon by the co-authors.

4.1.6 Analyzing Data

Answering the research questions of this SLR required analyzing tabulated data resulting from data extraction and synthesizing their content based on the RQ requirements and response elements. The main output of the analysis is exposing techniques, constraints, solutions, and other aspects and properties that provide elements for answering each RQ. All documents were subject to classifications in the tabulated data for each RQ, and this classification was re-evaluated by all authors for refinement.

4.1.7 Execution

This SLR was conducted in five incremental updates; the first execution was done in April 2018 (yielded 61% of the 182 primary studies), the second in February 2020, the third in June 2021, and the 4th in December 2021 after the reviewers' feedback, and the last update was performed on May 30th, 2022, with a full reevaluation of the abstracts. After the last update, a re-evaluation of the primary studies identified a total of 11 false exclusions, which were later included in the final result of 182 selected documents.

4.2 SLR Results: Taxonomy and Surveyed Aspects Data

In this section, we **1)** propose a taxonomy for the IoT capabilities composition and decomposition topic based on the studied RQs, and we highlight the references cited in this chapter, including primary studies. In subsection **2**, we provide extracted data for the Formal sub-aspects, which answers formal questions RQ1-RQ4. In subsection **3**, the extracted data that would answer Technical questions RQ5-RQ12 is provided, and finally, in subsection **4**, we provide tabulated data for the QoS sub-aspects that would help answer RQ questions RQ13-RQ15.

4.2.1 A Taxonomy of the SLR research questions aspects and sub-aspects and extracted data distribution.

4.2.1.1 Taxonomy

Based on related work and experts opinion, the issues and research questions addressed in this chapter/SLR study are organized based on three aspects: Formal (RQ1-RQ4), Technical (RQ5-RQ12), and QoS (RQ13-RQ15). We were inspired by previous SLRs on how they organized topics into taxonomies [132] [21] [295] [129] [284] [144] [162] [126] [262], and we proposed in Figure 4.5 a taxonomy for the IoT capabilities composition and decomposition topic (root), with an indication of which sub-aspect relates to which research question. The taxonomy would help in the search/filtering steps and guide the discussion and trend analysis. We believe that the proposed taxonomy can be extended by researchers to become comprehensive (with the inclusion of the other formal, technical, and QoS sub-aspects not discussed in this SLR) and can be leveraged by researchers to build a full picture of service composition formal, technical, and QoS sub-aspects. The taxonomy's three aspects (leaves) and corresponding sub-aspects (sub-leaves) discussed in this SLR are as follows:

•**Formal Aspect**: We include service composition-related standards, frameworks, and reference architectures as well as formal verification -which includes formal specification languages, formal verification techniques, and challenges-, under one aspect as they all aim to provide knowledge and common ground for representing or building a certain concept (composition algorithms fall under this aspect but are not addressed in the SLR study). The Formal aspect of the taxonomy contains the following RQs-related / sub-aspects: building composite capabilities based on a certain standard, framework, or reference architecture (RQ1); highlighting formal representations properties (RQ2); identifying trends related to formal verification of capabilities composition or decomposition (RQ3); and studying formal verification constraints (the state space explosion problem) (RQ4).
•**Technical Aspect**: represents sub-aspects including service composition domains and stakeholders (RQ5), service composition platform nature (RQ6), service composition automation and process type (RQ7), communication protocols leveraged in service composition (RQ8), capabilities data models (RQ9) and measurability aspects (RQ10), the role service decomposition plays in distributing capabilities or computation (RQ11), and the role of AI/ML in crafting novel services or improving the service composition/decomposition process (RQ12). Other Technical sub-aspects not discussed in this SLR include service discovery and selection.

•**QoS Aspect**: The taxonomy adopted in this survey considers QoS as a full-fledged aspect of the IoT capabilities composition and decomposition topic. One reason is that, for example, as more capabilities are composed into value-added services and applications, concerns such as privacy (RQ15) are of great concern because composition formulas and preferences might give away stakeholders' personal preferences [99][285]. In addition to privacy, scalability (RQ13) and interoperability (RQ14) are two QoS sub-aspects that we will address, as they relate to the identified research questions.

As indicated above, some Formal (composition algorithms, etc.), Technical (service selection, service discovery, etc.), and QoS (security, cost, energy efficiency, fault tolerance, response time, etc.) sub-aspects are not discussed in this SLR as the main topic or research question; they are represented in the taxonomy as "Other (Formal, Technical, QoS) Aspects." These other sub-aspects were either sufficiently discussed in the previous SLRs or were outside of the scope of this SLR. One goal of this SLR study is to encourage researchers to use and be inspired by the proposed taxonomy to build a comprehensive picture of IoT or CPS service composition and decomposition.

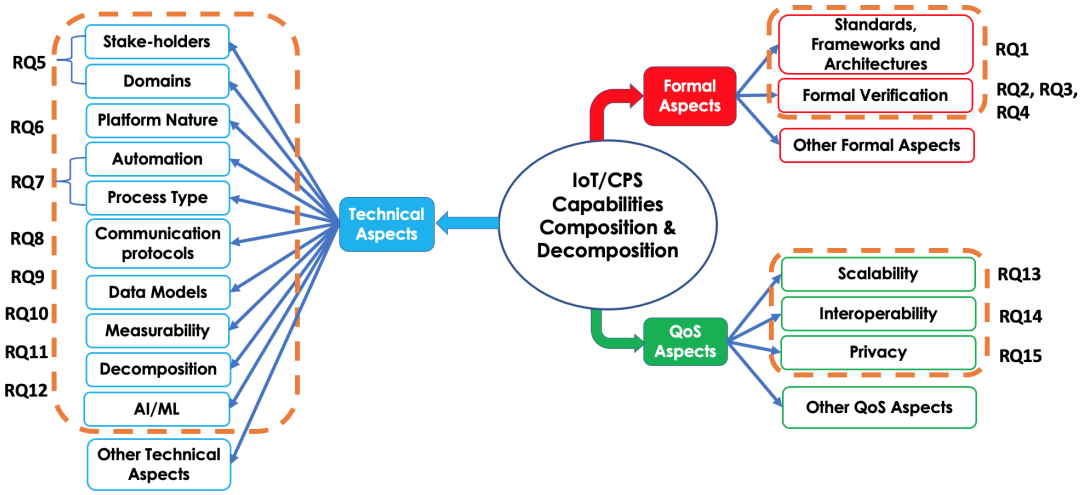

FIGURE 4.5: Proposed Taxonomy for the IoT/CPS capabilities composition and decomposition topic (root): Aspects (leaves), Sub-aspects (sub-leaves).

4.2.1.2 Manuscripts role and distribution

In the following subsections, tabulated results extracted from the primary studies -related to the research questions presented earlier- are exposed including :

• 182 primary studies that met the inclusion/exclusion/filtering criteria as highlighted in Figure 4.4. These primary studies are referenced in column **Ref** from Table 4.3 to Table 4.16.
• 15 references were leveraged to explain certain aspects of the SLR methodology or to introduce or explain certain topics.
• 32 references to GitHub repositories were cited to enrich the discussion around certain composition platforms implementations and projects, formal verification

tools, and source code. GitHub repositories also include [160], which contains extra details about the primary studies leveraged in this SLR.
• 12 online references that point to certain IoT composition tools or formal verification software.

The results of this SLR search are presented in the form of tables, which would be instrumental in answering the SLR questions in the discussion section. The next three subsections (**2**, **3**, **5**) contain data extracted for each aspect and sub-aspect mentioned in the taxonomy and relate to the SLR questions we seek to answer. Each sub-aspect of the taxonomy is introduced, and we identify which data -or Table- answers which RQ, while explaining the columns in the tabulated data and specifying whether these tables provide answers to more than one RQ.

4.2.2 Formal Aspects

In this subsection, data related to the Formal sub-aspects of IoT/CPS capabilities composition and decomposition are extracted to answer RQ1, RQ2, RQ3, and RQ4.

First, the standards, frameworks, and reference architectures supporting the composition and decomposition concepts of IoT and CPS capabilities are addressed. Next, the key characteristics and implementations of the algebraic and graphical formal representations were analyzed. Formal verification techniques used to verify composite services as well as tools and technologies that support such operations, are presented. Finally, the state-space explosion was given special consideration, with efforts and methods for solving this issue being discussed.

4.2.2.1 Standards, reference architectures, and frameworks

Standards, reference architectures, and frameworks provide foundations and guidance for building IoT or CPS platforms while respecting and guaranteeing certain aspects and constraints of interest to stakeholders. We list the different IoT and CPS capabilities frameworks that provide guidance on how to build composition environments with some criteria in mind [244]. Frameworks, standards and reference architectures that propose composition/decomposition guidelines ensure that platforms built have certain beneficial properties (listed in column **Composition/Decomposition Enabled Properties** in Table 4.3), which would serve as a reason and motivation for the native support of composition/decomposition guidelines.

Table 4.3 shows primary studies that addressed this sub-aspect, and its data will be used to answer RQ1.

4.2.2.2 Composition algebras and formal representations

Algebra or formal representations can be used to shape algorithms for composition and can be leveraged as formal specifications in formal verification tools for assessing different properties of interest [135]. Formal descriptions of objects and their interactions, leading to composing and decomposing IoT capabilities, are performed

TABLE 4.3: Composition and decomposition aware standards, frameworks, and reference architectures

Standards, frameworks, architectures	Ref	Composition/Decomposition Enabled Properties
IoT Architecture (IoT-A)	[244]	• Native support for composition engines: FIWARE Functionality Groups (FG) & The Management FG.
Architecture Reference Model (ARM) and Web Ontology Language (OWL) based Frameworks	[231] [40] [105]	• Compatible with hierarchical and distributed systems. • Supports the conversion of OWL descriptors to Domain Specific Languages via different tools. • Assimilate composition and decomposition mechanisms to programming paradigms: classes and subclasses.
On-The-Fly (OTF) Computing Reference Architecture	[146] [145] [155] [133] [123]	• Brings on-the-fly automatic service composition to IoT platforms. • Simplifies composition rules regardless of platforms complexity.
NIST CPS Framework	[112] [113] [307]	• Defines service composition requirements in IoT and CPS environments. • Supported composition requirements: adaptability, complexity, constructivity, service discoverability, and selection.
Service-oriented service of cloud manufacturing (CMfg) CPS	[313]	• Inherits the NIST CPS Framework properties and integrates them into the OneM2M platform. • CMfg enables the composition of trustworthy and large-scale industrial cloud applications.
Internet of Smart City Objects (ISCO)	[266]	• Performs service composition with SCOs that satisfy functional and qualitative requirements at runtime. • Adaptable, flexible, and suits different composition contexts in a wide range of applications.
IoT and CPS Composition Framework (ICCF)	[120]	• Leverages the NIST CPS Framework composition guidelines. • Exploits the mPlane semantics to describe composition operations. • Relies on formal verification tools such as TLA+ to verify the composition models.
Web Service Decomposition Architecture	[283]	• Enables computation decomposition of complex and computation-intensive services from the cloud to edge nodes.

using algorithms that rely on the algebraic representations of objects and services. Algebraic representations can also be derived from graphical representations using conversion [115].

In this paragraph, data that would help partially answer RQ2 is extracted, that is, recognizing the main properties of formal representations leveraged in service composition. Table 4.4 presents the efforts that have discussed formal representations and composition algebra. The data extracted from Table 4.4 provides an idea of how these formal representations are leveraged, whether graphical or algebraic in nature and their important characteristics. For completion, a column for the source code related to composition algebra was provided to help the researcher use and explore implementations and use cases of these formal representations.

4.2.2.3 Service composition formal verification aspects

Formally verifying composed IoT services properties is a mechanism that aims to verify the properties of atomic or composed IoT capabilities using models in the format of formal specifications, and running these models in tools to verify certain properties (deadlock freeness, correctness, fairness, etc.) [171]. The formal verification process is performed after the capabilities specifications are sketched to describe composed services using compatible composition algebra. The capabilities of IoT objects are typically described using a data model. This model is then converted into an algebraic language, which is translated into a formal specification language supported by a formal verification tool. The formal specification is later subject to a formal verification technique (model checking, equivalence checking, theorem proving) to verify -using a formal verification technique - that a certain property is met. The formal verification workflow for service composition is illustrated in Figure 4.6.

TABLE 4.4: Formal representations and composition algebras.

Algebraic or Graphical Representation	Ref	Nature	Description and relevant aspects	Src Code
Iterative Weighted Relaxation Service Composition (IWRSC)	[14]	Hybrid	• IWRSC algebra uses a directed graph for modeling services and operations • IWRSC is used to model power consumption inefficiency in large-scale IoT environments.	N/A
Real-Maude	[84][177][32]	Algebraic	• Real-Maude is used to reason about real-time systems and interactions in terms of time. • Real-Time Maude tool supports LTL model checking commands.	[20] [137]
LOTOS/ LOTOS New Technology (LNT)	[171][169][316] [219][59][269]	Algebraic	• Algebraic modeling style of IoT objects, interactions, states, and actions. • This simple modeling enables the verification of nondeterministic and concurrent systems.	[7][6] [207]
Temporal Logic of Actions (TLA) / PLUSCAL	[226][120]	Algebraic	• Model checking of composite services using TLA after PLUSCAL conversion. • The TLA+ tool converts the PLUSCAL model to a TLA specification.	[185] [279]
HTN-MLS (Hierarchical Task Planning for Machine Learning Services)	[221][220]	Algebraic	• HTN-MLS is an algorithm for automated service composition applied to the area of ML. • HTN-MLS recursively decomposes complex tasks into subtasks until only atomic tasks remain.	[92]
Recursive Composition Algebra (RCA) and interaction graph (RCIG)	[245] [246]	Hybrid	• RCA yields a directed tree with a root service representation which allows traceability of services in distributed systems. • RCA traces recursively services that compose a complex application in distributed systems built as a tree-leaf-root model.	N/A
DX-MAN (Distributed X-MAN)	[24][25] [22][23]	Hybrid	• DX-MAN is based on X-MAN, a component-based system modeling tool. • DX-MAN is suited for specifying multi-workflow services during runtime.	[75]
Markov Decision Process(MDP)	[189][258]	Algebraic	• MDP is used on top of FSM and Probabilistic Computation Tree Logic(PCTL) to model formal properties such as reliability and cost. • MDP uses states, actions, and rewards concepts to model discrete-time stochastic processes.	[78]
Finite State Machine (FSM)	[308][215]	Algebraic	• FSM enables the modeling of composite system states and the transitions between these states. • FSM is especially suited for deterministic, interoperable, and complete systems.	[259]
Pi-Calculus	[191][62]	Algebraic	• Pi-Calculus is a refined classical logic that provides a method for tracking resources. • Linear logic is leveraged to represent non-functional attributes, including cost and price.	N/A
Communicating Sequential Processes (CSP)	[312][319] [181]	Algebraic	• CSP is a mathematical theory for specifying complex patterns during concurrent interactions.	[150]
Extended Control Flow Graph (XCFG)	[187]	Graphical	• XCFG is an extension of CFG, which adds concurrency and synchronization dependency to model workflows of composite web services. • XCFG models BPEL workflows and ensures synchronization among concurrent activities.	N/A
Business Process Model and Notation (BPMN)	[35][296]	Graphical	• An Energy Efficiency Algorithm (E2C2) based on the BPMN graphic formalism was used to model an event-based choreography of decoupled microservices compositions.	[233]
Directed Acyclic Graph (DAG)	[179][314]	Graphical	• Compositions are modeled using the DAG as a chain of services invoked successively. • The result of the execution of one service invokes the next one.	[76]
Petri Nets/Colored Petri Nets (CPN) with a Kripke specification.	[115]	Hybrid	• CPN is a concurrent (as opposed to single-threaded FSMs) model. • CPN is converted into an algebraic specification (Kripke) to describe and model-check a customer service system model. • The Kripke specification is used to verify the reachability from and to other system states.	[151]
Vector Symbolic Architecture (VSA)	[264]	Algebraic	• VSA enables the compression of large volumes of data into a fixed-size vector. • VSA hierarchically models composite features which can be decomposed into atomic vectors.	N/A
Calculus of Communicating Systems (CCS)	[89]	Algebraic	• CCS includes primitives for describing parallel compositions. • CCS preserves synchronization and parallelism properties when converted to an LTS. • CCS evaluates the qualitative correctness of properties such as a deadlock or livelock.	[17]
Process Meta Language (PROMELA)	[65][282]	Algebraic	• PROMELA was used in this example to model and check the properties of Advanced Electric Power Grids such as noninterference. • RT-SPIN tool checks the correctness of the PROMELA model and determines whether it encounters the state space explosion problem.	[71] [250]
GALLINA	[237]	Algebraic	• Gallina, the specification language of Coq, was used to specify and prove distributed services mathematical theories based on building blocks including axioms, functions, etc.	[140][139] [320]
Intelligible semi-automated reasoning(Isar)	[275]	Algebraic	• Theorem proving of axioms using the proof language Isar within the Isabelle tool. • Isar is known for its easy readability by humans and machines.	[141][54] [305]

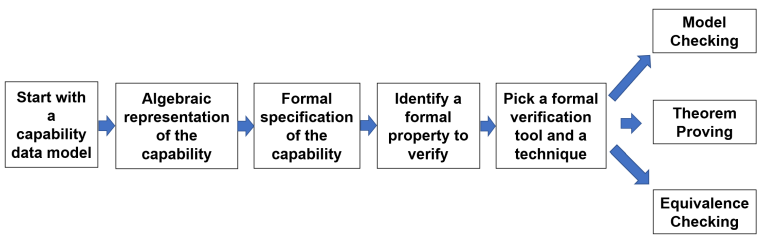

FIGURE 4.6: Formal verification of IoT/CPS capabilities workflow.

In this paragraph, data that would help partially answer RQ3 is presented, that is, recognizing formal verification techniques, applications, and tools leveraged in service composition. This would also help in understanding the extent of use of these techniques in research, service prototyping, or industrial applications. Table 4.5 aggregates primary studies that would contribute to answering this question; it provides data related to formal verification techniques, the properties they address, their domains of application, and the tool used to perform formal verification.

TABLE 4.5: Formal verification techniques (Model Checking: MC | Equivalence Checking: EC | Theorem Proving: TP), verified properties, applications, and tools.

Ref	Techniques	Verified formal properties	Applications	Tool
[84]	MC	Deadlock Freeness + Unmatched Sent Messages	Mozilla Project Things Smart Home application	Maude
[171]	MC	Deadlocks+Livelocks+Safety+Liveness+Fairness	Majord'Home: SDN-based Smart Home platform	CADP
[169]	MC	Compatibility + Deadlock Freeness + Correctness	Majord'Home: SDN-based Smart Home platform	CADP
[226]	MC	Correctness	Fault-tolerance checking and bug detection in AWS systems	TLA+
[313]	TP	Trustworthiness	OneM2M NIST-CPS Framework requirements verification.	Mobius
[189][174]	MC	Reliability	Probabilistic model checking formal properties of IoT services	PRISM
[170]	MC	Deadlock Freeness + Correctness	Light control in Smart Homes	CADP
[272]	MC	Correctness	Service Composition in Multi-Cloud Environments	NuSMV
[256]	MC	Safety	Model checking safety and states of an air conditioner system.	CLEM
[104]	TP	Correctness + Reliability	Multi-stage composition formulas in smart health systems	Coq
[154][275]	TP	Correctness + Security	Privacy-Oriented Smart Health Application	Isabelle
[242]	MC + EC	Realizability	BPMN 2.0 Choreographies	CADP
[254]	EC	Bisimulation	An Intrusion detection system BPMN model specified via LTS	CADP
[42]	MC + EC	Deadlock Freeness + Liveness	Sequence Diagrams and Pi-Calculus Comparison	MWB
[232][237]	TP	Correctness	CPS Designs Formal Verification	Coq
[214]	MC	Compliance	NetBill Communication Protocol	CWB-NC
[65][282]	MC	Correctness+Deadlock Freeness+Liveness+Safety	Formal verification applied to MQTT-CV/CPS Applications	SPIN
[216][163]	MC	Correctness+Reliability+Consistency+Completeness	BPEL Processes and events in Web Services	Maude
[197]	MC	Correctness + Security	Verifying IEEE 802.11i correctness and security mechanisms	UPPAAL
[4]	MC	Correctness + Reliability	Online Ticket System represented as a Process Algebra	MWB
[80]	MC	Correctness	Online Book Purchase System	CWB-NC
[2]	MC	Correctness	E-Health	CADP
[64]	MC	Compatibility	Distributed Systems	TLA+
[172]	MC	Liveness + Safety	TLA+ Specification and Applications	TLA+
[300]	MC	Correctness	Hotel Room Reservation System	TLA+
[101]	MC + TP	Correctness	CADP Specification and Applications	CADP
[222]	MC	Safety	Cross Road Smart Transportation System	UPPAAL
[3]	MC	Correctness	Online Book Purchase System	MWB
[83]	MC + EC	Reliability + Compatibility	Addressing State Space Explosion in Petri-Nets services	Multiple
[125]	MC	Security + Privacy + Reliability	Formal verification of composed IoT Services properties	CompoSec
[205]	MC	Correctness + Reliability	Verifying a SysML Model after ACME/ARMANI conversion	AcmeStudio
[321]	MC + TP	Safety	Model checking non-functional properties of software systems.	TLA+
[57]	MC	Correctness + Safety + Security	Model checking functional/non-functional properties of IoT.	NuSMV
[74]	MC	Dependability	Model Checking industrial CPS properties	nuXmv

4.2.2.4 State Space Explosion

As the number of state variables in the composite system increases, the size of the system's state space increases exponentially, which makes it challenging to formally verify composed systems' properties. This is called the "state explosion problem." Much of the research on model checking over the past 30 years has involved developing techniques to address this problem [70].

Any composite system can have a large number of states. The size of the state-space of a composed IoT system tends to grow exponentially as a function of the number of its capabilities, processes, and variables. The base of exponentiation depends on the number of local states of a capability or a variable, and the number of values a capability or variable may store [226]. State-space methods have motivated researchers to efficiently reduce the number of states while remaining faithful to system design.

Previous surveys did not give this topic the attention it deserves and classified it as an open research problem [271].

Table 4.6 aggregates the efforts that tackled the problem of state space explosion in service composition and based on which an answer for RQ4 will be provided, i.e., the different methods for resolution and the extent of success of such methods in solving the state-space explosion problem in service composition.

TABLE 4.6: The state space explosion problem in service composition: description, techniques for resolution, and outcomes.

Ref	Description	Resolution Techniques and remarks	Outcome
[246]	• Modeling and verification of composed IoT services	• Trace Merging in Recursive composition algebra (RCIG) partially solved the state space explosion problem by reducing the order of the (RCIG)	Reduced
[316]	• Composite systems parallel model-checking and property verification.	• CADP Evaluator: proven to prevent a state-space explosion by enabling the detection of errors in systems with a large state space	Eliminated
[219]	• State-space explosion in IoT capabilities composition	• CADP toolbox: breaks the verification process into simpler verification problems	Eliminated
[312]	• FDR Models specification scalability	• FDR2 specification proposed: yielded fewer states, thus contributing to better scalability of the model	Reduced
[187]	• Shortcomings study (including state space explosion) of BPEL and Petri Nets specifications of concurrent systems.	• Limiting the size of the specification is a proposed solution to the state space explosion problem	Reduced
[214]	• GCTL introduced an improvement to the Computation Tree Logic (CTL) specification language.	• CWB-NC model checker: alleviates the state explosion problem of automata-based techniques. • The space requirements for Boolean functions used in the symbolic technique are exponentially smaller than those that use explicit representation. • The proposed technique cannot eliminate the state explosion problem because the state space still increases when the model becomes larger	Reduced
[300]	• Composing and verifying TLA specified composite services	• TLA's model checker TLC: equipped with a multithreaded concurrent verification mechanism that alleviates the state-space explosion problem • TLA+ tool allows offloading computation to AWS EC2 instances, which provides more resources to alleviate the state space explosion problem.	Reduced
[130] [301]	• State-space explosion in model checking for service composition models	• ML algorithms: applied to find the optimal service composition. • For future work, prediction methods such as deep learning will be leveraged to avoid the state-space explosion in model checking for service composition models.	Reduced
[73]	• Errors during microservice choreography composition in Cyber-Physical Social Systems (CPSS)	• The asynchronous compositions increase exponentially with the size of the simulator buffers. • PAT Simulator: stops when the number of states exceeds the simulator buffer.	Managed
[47]	• State-space explosion in cryptographic IoT protocols • Specifications of these protocols are not described in simple rules	• Proverif tool: its pi-calculus algorithms efficiently simplify the protocol's specifications (unification)	Eliminated
[282]	• The state space of the model of a CPS (Smart Grid) was large and could not be verified using the available computation resources.	• The model was decomposed into multiple sub-models, each with a smaller state space that can be checked individually.	Reduced

4.2.3 Technical Aspect

In this subsection, the Technical sub-aspects of the composition and decomposition of IoT capabilities are discussed. Domains of application, stakeholders' concerns, and real-world implementations (e.g., AWS GreenGrass + Lambda [276][300][226]) or efforts that provide substantial code-base or interesting prototypes (e.g., MCC Cloudlets [202]) were explored. IoT platforms for service composition and related communication protocols, data models, schemas, and engines will be discussed. The composition process types and automation, as well as the measurability of the novel capabilities, are investigated. Novel Technical sub-aspects, including the decomposition of capabilities and the use of AI/ML in composing smart services or improving the composition process, are key contributions of this subsection. This subsection provides data that answers the technical questions RQ5 to RQ12.

4.2.3.1 Domains and Stakeholders

The applications of IoT composition cover multiple domains, including cities, buildings, transportation, health, farming, and manufacturing [152][29][313]. Different stakeholders have diverse expectations and requirements regarding building or leveraging composite capabilities. These stakeholders include end-users[9], developers[310], and city managers[68], to mention a few. Table 4.7 aggregates the efforts that addressed domains of applications of the capabilities composition or decomposition and highlights the stakeholders' interests in each domain for each use case. Data in Table 4.7 are instrumental in answering RQ5, i.e., Understanding the major stakeholders' concerns regarding composing capabilities in different domains.

4.2.3.2 Composition platforms, engines, and implementations

Composition platforms addressed the composition and decomposition of IoT capabilities at different complexity layers, including edge [202], fog[63], and cloud[272] layers. The takeaway that can be concluded by studying composition and decomposition in these layers is the fact that the complexity of a service increases when its atomic capabilities are composed of edge devices, creating a fog service, or composed of fog services to create cloud services. However, this same complexity can overwhelm the upper layers, particularly the cloud layer [196]. Offloading computations through decomposition from the cloud nodes to the fog nodes or even to edge devices can prove necessary to perform capabilities in the most computationally-efficient manner. Table 4.8 classifies platforms based on the composition engine, the nature of the platform (simulation, centralized, decentralized), as well as the composition layers targeted (edge, fog, cloud), and provides the reader with implementation details and source code references. The data in Table 4.8 were used to answer RQ6, that is, understanding the technical differences in capabilities composition implementation in different platforms.

4.2.3.3 Composition Process Type and automation level

The process type refers to how the composition is triggered or processed and what its workflow looks like. Services can be composed in a serial [171], parallel [310], rule-based [128], or folow-based fashion [221], among other process types. Composition is a complex process that involves several steps, including discovery, selection,

TABLE 4.7: Capabilities composition domains and corresponding stakeholders concerns.

Domain	Ref	Description/Application	Stakeholders' Concerns/Interests
Smart Buildings	[29] [169][171] [84] [9]	• Composed smart building services and scenarios. • IoT Composer: smart building services composition. • Automatic temperature/light management in a smart building. • Person detection in a smart room	• Users expect a highly automated composition process. • Users require ease of use of the platform. • Ranking and filtering facilitate the selection of composition abstractions that are intelligible to users. • User-friendliness of the platform's GUI.
Environment Monitoring	[308]	• FSM model-driven service composition architecture for the rapid prototyping of IoT services. Environment monitoring stations were implemented using three types of wireless sensor networks and deployed on a university campus.	• Developers and researchers can customize and quickly prototype composite services. These stakeholders also showed interest in cost reduction, reusability, and cross-domain interoperability.
Smart Transportation	[121] [196] [310] [264] [68] [182][118]	• UCEF, a co-simulation environment, allows the composition of features of complex systems such as autonomous vehicles. • A platooning feature is composed: a technique that enables cooperation during adaptive cruise control for a series of vehicles. • Dracena was used to compose vehicle sensor data for decision support and to predict insurance quotes based on thresholds. • Traffic congestion at the city level uses Node-RED for a distributed implementation, with input from a traffic camera API. • An estimator for parking, traffic, and noise was composed via FIWARE to provide an open trip planner. • A travel booking/reservation composite service based on service-oriented computing architectures or web-based architectures.	• Assessing the trustworthiness of the functions of a CPS • Road users expect improved safety and mobility. Developers benefit from the time efficiency gained by shifting from hardware and communication composition complexity to the ease of microservices composition. • Insurance companies use Dracena to generate custom, fine-grained recommendations on insurance fees. • Developers benefit from improved collaboration when Migrating the Node-RED implementation from a centralized to a decentralized paradigm. • City planners seek cost optimization, thereby encouraging developers to build modular designs for their applications and identify reusable IoT services. • Users benefit from the ease-of-use and plug-and-play automated features of the composition platform.
Smart Farming	[152]	• Gaiasense offers novel, inexpensive composite smart farming services by facilitating data interoperability for smart-farming systems using techniques such as "Data Interoperability Zone" and "Information Management Adapter."	• Farmers' interest lies in the zero cost of implementing Gaiasense. Producers of farming systems benefit from the reduced manufacturing cost. The environment also benefits from systems cooperation as it can reduce fossil fuel emissions.
Smart Cities	[63] [294] [67] [281][135] [58][16] [240]	• Application: FogFlow-based anomaly detection of energy consumption in a smart city. • Adaptive service composition framework that supports dynamic reasoning. This allows mobile users to perform their daily tasks dynamically by integrating the services available in their vicinity. • Smart city services include human-centric mobility, multimodal transport (parking, disabled people's navigation), and community policy applications (noise monitoring, agile governance). • Energy-efficient IoT service composition algorithms are discussed with a focus on data sharing, fog-enabled, and mobile-based IoT applications. • Smart-city composition platforms are discussed along with different impacted domains and design challenges.	• FogFlow's model allows IoT service developers to program elastic IoT services easily over the cloud or the edge. • End-users discover and investigate more composition opportunities owing to the dynamic reasoning support of wEASEL-based composite heterogeneous systems. • Developers are interested in publishing atomic services -leveraged by small companies to speed up composition- in a one-stop-shop repository. • Energy-optimized composite services -achieved by adopting resource-efficient platforms (FSCA-EQ,..) and efficient composition algorithms (CRIO,...)- are the main concern for different stakeholders. • Users are concerned about the composition platform performance, environment friendliness, security, low cost, and reliability.
Smart Health	[196]	• IoT capabilities were re-conceived as a "microservice" in contrast to the "thing" concept. This allows IoT to benefit from features such as the distribution of services and service discovery. The approach is illustrated in a personal health management service.	• Developers access ready-to-compose distributed, and secure microservices features using API Gateways. Users benefit from the enforcement of access control to composite microservices, which improves privacy protection for the data owner required for smart health applications.
Smart/Cloud Manufacturing	[313] [309] [195][13] [273]	• A large-scale composition platform for cloud manufacturing is introduced by connecting multiple remote factories and establishing a collaborative connection through the cloud. • QoS stability algorithms were proposed to minimize supply chain manufacturing errors and, as a result, improve the competitiveness of the manufacturing facility. • The difference between traditional manufacturing and service-oriented manufacturing was explained, and the benefits of cloud services in enabling collaboration and fault-tolerant composition of services between different providers were highlighted. • SOCRADES, a smart manufacturing architecture, is presented. It relies on smart connected objects and tagged raw materials to build services that enable agile manufacturing of goods that accommodate customers' needs.	• This composition paradigm uses the NIST CPS framework to address the concerns of trustworthiness/stakeholders' (i.e., functional, human, timing, interoperability, and intelligence). • Users and developers requirements: QoS stability, service collaboration, and service composition failures are discussed within a cloud manufacturing application. • Recommendation algorithms are proposed to allow customers to compose products online based on their concerns (time, cost, reliability, availability, and throughput) in a fault-tolerant fashion. • User-friendliness, ease of use, and customization are the major user concerns related to the proposed smart manufacturing composition platform.

filtering, composing, verifying, and delivering. Some of these steps can be automated or semi-automated (requiring the input of a user or another entity). Similarly, each composition process has a set of tasks, each with a certain level of automation. Different efforts have tackled the composition process type and automation level independently: we try to find synergies and relationships between these two Technical sub-aspects. Table 4.9 highlights efforts that addressed composition process

TABLE 4.8: Composition platforms, engines, and implementations.

Platform	Ref	Platform nature	Composition Engine	Implementation Details	Src Code
mPlane	[287]	Microservices distributed as nodes (Consumer, Producer, Registry, Supervisor, Reasoner)	The **Supervisor** aggregates the lower-level capabilities into higher-level capabilities.	Implementation is available in Python3 [95], or NodeJS [86]. Requires a supervisor for composing capabilities, a capability source, and a requesting client. These client-server implementations can be run through an IP architecture or over a CDN.	[86] [95]
Fast IoT Composition	[14]	Distributed IoT services composition platform based on user-defined QoS requirements	The **Service Composition Process** composes atomic services based on various QoS needs.	Implemented using Java under a Windows 7 OS, and machine specifications were provided. No Git repository or reference to code is provided.	N/A
IoT Composer	[169]	Web application with a GUI for composing capabilities and verifying their correctness.	The **Majord'Home** platform and the **CADP** correctness verifier.	Web application hosted on Apache Tomcat. API uses jQuery and Semantic UI; calls are done using the REST API, serializing JSON objects	[8] [138]
FIWARE	[244]	Open-source decentralized/scalable IoT platform for data management and composition of smart applications.	The IoT Broker hosts the **Entity Composer Plugin** that composes services by aggregating attribute values.	Using the FIWARE composition plugin (iotbroker, entitycomposer) requires installing the IoT agent and enabling the entity to be composed.	[94] [93]
MCC Cloudlets	[202]	The platform provides composition at the edge via mobile devices, fog (virtual representations), and the cloud.	The **Central Cloud** composes Virtual Device Representations at the fog into applications	Networking between edge, fog, and cloud is ensured by OpenStack. Linux and android containers host virtual device representations residing in the fog. The Src Code column points to the platform's UPPAAL model and to the UPPAAL tool used for modeling.	[61] [292]
IFTTT	[146]	IFTTT is a cloud-based app available on iOS and Android. It provides access to public and private APIs.	**IFTTT** sets responses for events, connects to service providers, and executes commands.	Install the application on a supporting device and connect to APIs of interest. IFTTT applications include remotely controlling devices when a condition is met.	[136]
Home Assistant	[308]	Web-based home automation software for central control of smart home devices.	Home Assistant **Scripts**, **Scenes**, and **Automations** compose services similar to IFTTT's.	The home assistant software comes in different flavors. The most stable one uses an always-on VM that comes pre-installed and requires network configuration only.	[97]
Node-RED	[264]	Flow-based Web tool for composing services and connecting hardware and software services.	**JavaScript Functions** are inserted from a web interface to compose capabilities.	Install Node-RED using npm. On the web interface, drag and drop features from different APIs. Integration with IBM Watson's data analytics tool is available.	[278]
UCEF	[121] [251]	Simulation framework, which provides tools for creating simulations for CPS	Simulates a CPS via a federation: a composition of federates that interact via the **HLA Protocol**.	A template federation for a CPS is created using WebGME. The capabilities of each federate are populated using an IDE. ADS example in [121].	[161]
AWS Greengrass	[276]	A commercial cloud solution that provides various services, including storage and computing.	**Greengrass** is a solution for connecting and composing IoT services using **Lambda** scripts.	Install AWS Greengrass dependencies, certificates, and software for target devices. Leverage Lambda scripts to automatically compose services.	[260]
Microsoft Azure IoT	[318] [183]	Microsoft cloud service for building, testing, deploying, and managing services.	**Azure IoT Hub** enables bidirectional communications and composition between IoT devices.	Setting up an Azure IoT hub requires an Azure account, Azure portal, Azure IoT Tools, Azure PowerShell, Azure CLI, Azure REST API, and .NET templates.	[218] [217]
Cloud CAMP	[44]	An open-source cloud platform that automatically delivers composite applications in the cloud	**Abstraction of Business Model**, **Configurator**, **Enactor**, and the **Knowledge Base**	WebGME MDE is used to define the metaModel, MongoDB is used to store the model, and MySQL is used to store the knowledge base.	[18]
Vert.X	[304]	Message-driven toolkit for creating reactive, elastic, resilient, and responsive microservices	**Composition functions** implemented within Vert. X microservices.	Install Vert.X, and use its REST libraries (e.g., Axios) to connect to sources of data. The code references a well-being application that uses the Vert.X toolkit.	[159]
Thing Orchestration	[201]	A composition platform geared toward centralized computing architectures such as the cloud.	The **Orchestrator** is a central component coordinating a concurrent sequence of things to call during composition at runtime.	Python scripts were used to model a centralized architecture with a pool of resources with services to be composed. No Git repository or a reference to code is provided	N/A
MMESCN	[229]	A service composition **M**echanism based on collaborative **M**obile **E**dge **S**ervers and **C**ache **N**odes (MMESCN).	**Mobile Edge Servers (MES)** compose capabilities provided by IoT devices. **Cloud Center Nodes** take over when MES are not capable enough.	Network Simulator 3 (NS3) tool is used to simulate networks and edge or cloud composition nodes. One cloud center node is connected to 8 mobile edge servers. Each mobile edge server is connected to 20 IoT devices.	N/A

types and the automation level as well as manifestations for each topic. Data in Table 4.9 will be instrumental in answering RQ7, i.e., understanding how composition processes differ in terms of the automation level.

4.2.3.4 Communication protocols in service composition

From a communication perspective, capabilities composition and decomposition platforms adopt multiple communication paradigms and play different roles that

TABLE 4.9: Service composition process types and automation levels.

Composition Process Type	Ref	Process Manifestation	Automation Level	Automation Manifestation
Sync/Async Support	[287]	• The data and control flow within mPlane supports both cyclic workflows in the "foreground" and continuous/periodic measurements in the "background". • mPlane interfaces facilitate the flow of control messages that trigger new measurements and data reception, supporting both synchronous and asynchronous modes.	semi-automatic	• mPlane is highly programmable and supports component-initiated workflows (no client interference required) and client-initiated workflows.
Flow-based and Programming-based	[14]	• The composition flow requires rendering constrained devices' capabilities into virtual objects before the composition and defining their QoS in a way that contributes to either their selection or exclusion. • An Iterative Weighted Relaxation Service Composition (IWRSC) algorithm is proposed that allows users to customize QoS parameters that contribute to the selection of atomic capabilities participating in a composition.	semi-automatic	• Composite services are realized as a succession of tasks. • Tasks represent abstract services that can be realized by the best-fitting concrete service. • The user decides on the fitness of a service based on its QoS properties. • QoS properties are either device-specific or user-defined; hence, they are semi-automatic.
Asynchronous or Parallel	[310]	• Dracena compositions enabled the asynchronous and cooperative use of large amounts of IoT data among disparate services in real-time.	semi-automatic	• The composition development environment provides Kafka plugins that perform compositions on basic and complex capabilities. • It also provides service designers APIs to inject hypotheses and verify output correctness.
Rule-based	[84]	• Automated techniques for building compositions of devices represented as abstract objects are proposed. • Filtering and Ranking rule out non-desirable compositions as the number of possible compositions is high.	automatic	• For the Node-RED deployment, human intervention is only required to define the goal of the composition. The other composition steps were then automated.
Flow-Based	[46] [264] [157]	• Node-RED, WoTKit, and VITAL-OS are flow-based IoT programming tools. In particular, Node-RED models composition task logic as a directed flow chart that consists of sequences of interconnected service nodes. • Flow-based composition can be easily represented in GUIs, which increases usability and provides more intuitive user interaction. • When it comes to complicated composition task logic, flow-based composition tools cannot avoid introducing additional programming.	automatic	• The WotKit Processing Engine automates data collection and processing using scripts. • The VSA approach applied in Node-RED enables automated service composition by combining and self-describing services. • VITAL-OS, an open-source operating system for smart cities, aims to automate smart city services in a three-phase process: infrastructure deployment, deployment of vertical applications, and deployment of city-wide integrated applications.
Synchronous or Serial or Sequential	[171] [169]	• Composition aims to connect various objects through their APIs. • The objects involved in a composition interact through bindings in a synchronous manner.	automatic	• IoT Composer graphically exposes the interfaces of the selected objects and allows the user to bind interfaces concerning the IoT service expected from the composition. • It is worth noting that only this first step of the approach requires human intervention; all the subsequent steps are fully automated.
Flow-based	[221] [220]	• MLS-PLAN: a Machine learning services planner that leverages an algorithm for automating machine learning data classification.	automatic	• The automated task composes an ML pipeline (consisting of ML services) that maximizes classification accuracy over new data from a data source
Process or Programming Based	[308]	• Programming or process-based composition is the most widely used method in which the business logic of the service is composed mainly by manual programming or using scripts. • This method adopts either traditional programming languages or domain-specific languages.	semi-automatic	• This method allows the encapsulation of heterogeneous sensors, actuators, and IoT devices into composable Web services with uniform model-driven,user-customizable, semi-automatic compositions
Rule-Based	[128] [293]	• In openHAB 2 and IFTTT, rules are usually predefined and represented as events, conditions, formulas, or symbolic logic. • Whenever a rule is met during runtime, the corresponding operation is automatically triggered.	automatic	• IFTTT and openHAB only require setting up simple rules via a GUI that trigger events in the context of home automation. This approach is deemed automatic owing to its GUI accessibility and low complexity level.

vary depending on the context and other factors, including power consumption [289] or the environment type (production[169], simulation[251], ..), among other factors. Table 4.10 presents primary studies that tackled the communication protocols in service composition, their use cases, and their workflows. Table 4.10 data will be leveraged to answer RQ8, i.e., the role communication protocols play in the composition or decomposition of IoT/CPS capabilities.

4.2.3.5 Data Models or Schemas

The capability extracted from a device must be properly represented using a data model or schema to facilitate composition or decomposition. Efforts that tackle IoT capabilities composition typically tend to use data models based on known formats such as JSON, XML flavors, WSDL, HTML, and other options to represent IoT capabilities [248]. Device characteristics are described through ontologies that provide

TABLE 4.10: Communication protocols roles in IoT/CPS service composition/decomposition.

Communication Protocol	Protocol description	Ref	Protocol Use case / Workflow
mPlane Protocol	• mPlane refers to a protocol in which entities request, receive, compose, and store data by leveraging multiple components.	[287]	• This example leverages mPlane as a measurement platform. Specifications are sent by a consumer to request measurements from probes capabilities.
Representational state transfer (REST/HTTP)	• A software architectural style that guides the design and development of web services	[45] [315] [149]	• RESTful communication technologies are used to connect the IoT server proxy to both the smart city controller (using HTTP) and the Smart Service Provider (using CoAP). • A solution for composing asynchronous RESTful web services, making use of various IoT services, invoking the RESTful web services from the IoT, and publishing a BPEL process as a RESTful web service by extending BPEL. • The proposed composition platform enables transparent access to heterogeneous IoT networks -including CoAP- using "interworking" proxies to the HTTP protocol.
multicast DNS protocol (mDNS)	• mDNS resolves hostnames to IP addresses within small networks that do not include a local name server.	[84]	• mDNS is used to enable the discoverability of IoT devices.
Constrained Application Protocol (CoAP)	• An internet application protocol (RFC 7252) that enables communication between constrained devices.	[289]	• SVOM (Service and Virtual Objects Management system) is introduced with a CoAP-based fault management component that captures the status of the registered devices and services.
Software-Defined Networks (SDN)	• A network management technology that enables dynamic, programmable, and efficient network configuration.	[169] [313]	• Majord'Home leverages a software-defined LAN (SD-LAN) for its simplified interfaces required for network configuration of IoT services involving multiple smart environments. • A testbed based on SDN is constructed for experimental verification in a flexible network configuration with Mininet as an emulator for configuring and deploying virtual devices
Simple Object Access Protocol (SOAP)	• A messaging protocol specification that enables the exchange of structured data in web services	[312] [88]	• The service requester and service provider exchange data using SOAP service request and response messages. • Genetic algorithms and case-based reasoning are leveraged to define automatic composition workflows in SOA and select suitable atomic services based on select QoS requirements.
Message Queuing Telemetry Transport (MQTT)	• A lightweight, publish-subscribe network protocol that transports messages between devices and usually runs over TCP/IP.	[63] [128]	• Fogflow's NGSI-based context management system provides a global view for all system components, enables querying, subscribing, and updating context entities, and supports MQTT pub-sub message brokers such as Mosquitto and Apache Kafka. • MQTT is used as middleware for its pub/sub-broker, which runs on top of a TCP/IP network for the openHAB automated home system.
High-Level Architecture (HLA)	• A standard for the distributed simulation used when building a simulation for a larger purpose by combining several simulations	[251] [121]	• HLA is used as a medium for exchanging data between federates composing CPS in UCEF simulations.
WebSocket	• A communications protocol capable of bidirectional communications over a single TCP connection (RFC 6455).	[304]	• Netty is compared with other WebSocket frameworks, including Undertow, Vert.X, Grizzly, and Jetty, to decide which situations each framework ought to be used for.
Named Data Networks (NDN)	• Named Data Networks (NDN) is an Information-Centric Networking (ICN) protocol that is seen as an alternative to TCP/IP.	[302] [164]	• NDN provides the benefit of node caching and service decomposability by tracing back its interest requests. • Improving NDN's high-speed forwarding capabilities using NDN-DPDK techniques offers faster forwarding speeds than known IP protocols. These benefits can be extended to other applications, such as IoT communications capabilities.
Wireless Sensor Networks (WSNs)	• Wireless sensor networks are dispersed sensors that track physical environment data and forward it to a central location.	[224] [117]	• Principles of the WSNs and SOA are exploited to enable energy-efficient composition features in the Networks On a Chip (NOC) paradigm. • WSNs and smart connected devices (IoT devices) are leveraged to develop new applications in a specific domain using the concept of "mashup" architectures or user-generated composite applications.
GS1 EPCglobal	• A set of protocols (RFID, GDSN) and standards (EPCglobal) that facilitate data exchange between different stakeholders in the supply chain domain.	[107]	• An EPCglobal-compliant IoT Middleware (Fosstrac) was used to validate a proposed IoT Infrastructure model which uses the Application Level Events (ALE) and the Electronic Product Code Information Services (EPCIS) components of the EPCglobal standard to support service decomposition, multi-threading, elasticity, and cloud virtualization capabilities.

a shared vocabulary to model different objects and concepts and their relationships [10]. Table 4.11 presents these efforts, and based on the results obtained from it RQ9 is answered, i.e., understanding the roles of the different IoT data models and schemas leveraged in capabilities composition or decomposition.

';'

4.2.3.6 Decomposition

The decomposition of IoT services and capabilities is the process of deconstructing a complex service into small services or atomic capabilities [15][283]. This process

TABLE 4.11: Capabilities data models/schemas properties, roles, and examples of attributes.

Data model/schema	Ref	Data model/schema role	Examples of data models attributes
XML	[286] [190] [317]	• XML was used in mPlane for integration with XML-based systems. • RDF/XML provides APIs for backward compatibility with other platforms. • Describing and discovering services from different Wireless Sensor Networks (WSN) nodes and networks using XML and SOAP standards.	RDF/XML schema attributes: rdf:statement; rdf:subject; rdf:predicate; rdf:object
WSDL	[246][234][156]	• WSDL is used with SOAP and XML schemas to describe web services through a document that lacks implementation details and logic, which is why WSDL is limited to modeling and not formally verifying operations.	WSDL schema parameters: Port Type; Operation; input; output
JSON-based schemas: JSON JSON-WoT mPlane-JSON JSON for linking data (JSON-LD)	[171] [169] [100] [84] [286] [190] [264]	• The JSON schema represents the objects, the bindings, steps, and strength of the bindings and the composition plan. • The JSON-Web of Things schema is used to describe IoT objects' capabilities and also the IFTTT rules that describe the synchronous compositions. • The mPlane reference architecture leveraged JSON as the default schema for its simplicity, parseability, and efficiency. • For the evolution of context information representation, JSON-LD provides more powerful options and is designed to be easily integrated into the existing Context Information Management (CIM) platforms. • JSON is used for a service description of the Node-RED object detectors. JSON service descriptions were later converted into semantically comparable service vector descriptions.	mPlane capability Attributes in the JSON schema: parameters start: now...+inf end: now...+inf source.ip4: 192.0.2.3 destination.ip4: octets.count: 28...65535 period.s:
YAML	[286]	• YAML's main use cases for the mPlane platform include documenting and debugging for its improved human readability and writability.	YAML's schema attributes : template; interfaceMap; activeTimeout; maxFlows; silkCompatible; ...
HSML/HTML	[308]	• HSML is a domain-specific language with HTML-like syntax for describing the state of an IoT service and the state transfer chain between services.	HSML Schema Attributes : loc (x,y), id, src (ip,protocol) type , filter
OWL-based schemas: OWL-S OWLS-TC4	[300] [85] [303] [294]	• OWL-S documents describe services in terms of predefined upper ontology, such as service profiles, grounding, and processes. A formal TLA specification is defined based on the OWL-S description. • IoT services are defined as a subclass of the service class defined in OWL-S. An IoT Service has a Service Profile and a Process that describes its functional and nonfunctional properties (inherited from the OWL-S Service class). • OWLS-TC4 was used as a data model to create composite services in the context of smart cities. Authors argued that data models such as OWL-S provide simple capability aggregation mechanisms for non-complex compositions.	OWL-S Service schema attributes: service name, port types, service operations, input message, output message, implementation descriptions
WoTDL	[230]	• The Web of Things Description Language (WoTDL) ontology is an alternative to existing WoT models such as OWL-S and WSDL, which are not suitable for describing AI planning concepts for automatic WoT composition.	WoTDL schema attributes: hasActuator; hasSensor; name; hasMeasurement; hasPreCondition; hasActuation; hasEffect
BPEL-based schemas: BPEL WS-BPEL BPEL-TC	[280] [199][200] [208] [72]	• WS-BPEL is assimilated to UML as it allows the description of services and enables the description and execution of composite services. • Composite services based on WS-BPEL were tested under load in a travel-agency composition prototype. • BPEL-TC is an adaptation of WS-BPEL that accommodates composition and decomposition requirements for temporally customized web services. • BPEL process models are transformed into composite executable service templates after three steps (template creation, composition, and installation.)	BPEL schema attributes: Travel Request; Invoke FS; resp FS; type input; Travel Response; Type output
Semantic Sensor Network (SSN)	[38]	• The SSN ontology (part of the W3C Semantic Sensor Networks Incubator Group) represents context information for sensor devices, including deployment attributes.	SSN schema attributes : available battery, deployment attributes, location, time

contributes to cost-efficiency, scalability, and interoperability between IoT systems when complex services are built with decomposability in mind. Based on the research we performed, decomposition efforts are either i) capability oriented: which means a complex feature is decomposed into sub-features or atomic capabilities for reuse by other systems, or ii) computation oriented: a complex service might run in the cloud and consume more resources than allowed by a single service, decomposing computation allows its distribution on fog or edge devices in a way that renders resource consumption more efficient.

Decomposition is challenging, because the building blocks of complex services might not be easily decomposable, especially when their APIs lack loose coupling

support [196]. Decomposition also comes with nonfunctional challenges such as scalability, frequency of data generation, and security requirements. Table 4.12 aggregates service decomposition primary studies, and the data it contains will be leveraged to answer RQ11. i.e., exposing the benefits of building IoT platforms and complex services with decomposition in mind.

TABLE 4.12: Service decomposition efforts description and benefits

Ref	Platform	Decomposition Description	Decomposition Topic	Decomposition Benefit
[14]	Energy-efficient and fast IoT service composition	• Relaxation is a technique used to improve the flexibility and scalability of service composition. • Relaxation decomposes users' requirements and QoS constraints into local constraints of services.	Capability	• Decomposition points to energy-efficient atomic capabilities that compose energy-efficient services.
[15]	FOG-IoT and Linked-Microservices	• Linked microservices are built with decomposition in mind, contributing to the computation across different computing nodes in the IoT architecture and benefiting from fog and SOA strengths.	Computation	• Congestion reduced by (70%) at the cloud.
[308]	FSM model-driven service composition and decomposition architecture	• From a machine state perspective, composing or decomposing IoT devices can be described as building or breaking the linkage of states between FSMs.	Capability	• Reusability of FSM model-driven services.
[264]	Node-RED/VSA	• The proposed scheme converts existing Node-RED microservices into a cooperating set of decomposed and decentralized proxies. • These proxies are instantiated into the CORE environment by adding a cognitively aware wrapper around each service to facilitate decentralized discovery and execution.	Capability	• Decomposition decentralizes services and improves collaboration between capabilities.
[196]	Microservices-based IoT platform	• Containers leveraged in microservices to enhance safe service decomposition and enhance the security of IoT infrastructure. • Desirable characteristics of microservice-based systems include the decomposition of larger services into small, focused, self-contained services with loose coupling.	Capability	• The loose coupling allows fast decomposition and, as a result, faster reuse of existing capabilities for further compositions
[302]	Named Functions Networks Computation Service Management (NFN CS-Man)	• Decomposition is built into the information request of the NDN IoT network model. • Sending a request (interest) is later subject to decomposition to sub-interests, each requesting an aspect of the composite service.	Capability	• Interests retrieve cached capabilities from a nearby cache. • Traceability of atomic capabilities by checking the interests in the Pending Interest Table (PIT).
[267]	iKaaS functional decomposition	• Each simple service runs in its process and communicates using lightweight mechanisms. • The overall high-level service logic is decomposed into multiple software modules, delivered as independent runtime services.	Capability	• Service functional decomposition ensures agile, autonomous, flexible, and scalable services.
[81]	Fog-IoT ORchestrator (FITOR)	• Decomposition of application computation (CPU, memory y) on different nodes is performed e using optimized fog service provisioning g (O-FSP)	Computation	• Resources/cost optimization. • Stakeholder acceptance
[107]	EPCES-ALE	• Service decomposition is considered one of the pillars for a "Future IoT Infrastructure" along with virtualization and multi-threading in order to distribute computation optimally.	Computation	• Computation optimization
[198]	Hierarchical IoT	• A middle-layer service can be decomposed into sub-services which can be integrated to complete the pieces of another service with higher complexity.	Capability	• Reusing atomic capabilities from a complex service into another service
[37]	Decomposing monoliths into smaller microservices using domain-driven design (DDD)	• For monolithic software systems with a complex domain, decomposition faces many challenges, including the low comprehensibility of its source code. • Add to these challenges that every change in the system needs a whole redeployment, and all parts of the system are not equal in terms of change frequency. • DDD is a good option for the initial decomposition of a system as it can be applied to a subdomain to decompose it into smaller chunks.	Capability	• DDD ensures reusability through the decomposition of small chunks of a system.

4.2.3.7 AI/ML

Researchers have addressed AI and ML use in service composition from two perspectives. The first perspective is capability-oriented [67], where the AI/ML capabilities are aggregated into value-added features with AI/ML capabilities. The second perspective is process-oriented [221]: AI and ML improve the composition process, especially service selection, where filtering criteria are considered for picking a particular capability over another. Table 4.13 illustrates the primary studies related to

IoT capabilities composition that tapped into the AI/ML potential, and based on the data it presents RQ12 is answered, i.e., understanding the role AI/ML techniques play in shaping novel capabilities or improving the composition process.

TABLE 4.13: AI/ML use in service composition/decomposition: description of efforts, orientation, applications, and mechanisms.

Ref	Description	Orientation	AI/ML applications and mechanisms
[67]	Using AI/ML atomic service to compose intelligent and complex capabilities in the smart-city domain (EU's SynchronyCity project).	Capability	• Composing ML complex capabilities, including parking area availability, traffic flow, and noise level predictions based on time-series data. • These complex capabilities will be used as atomic capabilities to build smarter and cross-city applications
[15]	Facial recognition service that is composed of ML tasks, including facial feature extraction, data fusion, data filtering, and face detection algorithms	Capability	• Services decomposition via fog computing was introduced to shift computation latency and burden from the centralized cloud to decentralized fog nodes regardless of data types (numerical, text, image) and the different ML algorithms. • Decomposition of facial recognition capabilities works with different ML algorithms and data types.
[221] [220]	Composing ML pipelines to improve classification outcomes	Process	• Given sample data, the task is to compose an ML pipeline (consisting of ML services) that maximizes classification accuracy over new data from the same source. • AI-based Classification can be leveraged in service selection, thereby improving the composition process
[125]	AI is used to verify nonfunctional properties of a composed system, including security, privacy, and dependability, and track its changes over time	Process	• CompoSecReasoner framework uses AI algorithms to monitor the composite system and its changes and evaluates its security, privacy, and dependability status. • CompoSecReasoner utilizes an event and model-based approach and implements dynamic system composition verification, properties validation, and automated administration based on metrics.
[68]	Reusable AI atomic smart-city services.	Capability	• AI-enabled smart city services were analyzed and compared collaboratively. This resulted in atomic services such as parking and traffic estimators that use AI. • These atomic AI capabilities can be used to enable more complex capabilities in a collaborative platform.
[88]	Genetic algorithms are used for service selection based on QoS defined by the user in SOA-based environments.	Process	• An approach to deal with the dynamic service composition problem is presented based on a genetic algorithm and case-based reasoning, which supports the flexible service workflow according to the user's requirements.
[90]	AI algorithms are leveraged for geospatial web service selection and automatic composition	Process	• Integration of well-described web services into feasible workflows can be achieved using AI planning, a computational deliberation process that chooses and organizes actions to achieve predefined objectives.
[33]	Service selection using an agent that uses reinforcement learning to select devices for composition based on specific user-defined criteria	Process	• A reinforcement learning technique was adopted to effectively deal with highly dynamic situations in mobile IoT environments. • A reinforcement learning agent was proposed that selects services in terms of spatial cohesiveness and number of handovers while providing the services.
[253]	Predictive maintenance in Industrial IoT platforms enabled by A Federated Learning Framework.	Process	• IoT platforms maintenance impacts not only nonfunctional properties such as the QoS but also process-related operations such as service discovery/availability. •Maintenance and fault data are collected over time and based on which a prediction is made to infer when the next fault will occur
[247]	Integrating and adapting Machine Learning for security monitoring in Big Data IoT data aggregation platforms.	Process	• Ensuring the security of IoT platforms through AI can be instrumental in guaranteeing certain composition processes are not compromised (service selection: a compromised atomic service might lead to compromised composite service.). • Phases for implementing security-aware machine learning processes within IoT Data aggregation platforms are explained.
[228]	Composing real-time AI capabilities within the edge layer microservices.	Capability	• The platform implements AI capabilities at the edge layer (called ROOF) and only calls upper layers (Fog, Cloud) for composition or computation-intensive tasks. • This platform was used to analyze incoming AC, Speed, and Fuel Consumption data from simulated vehicles with the goal of leveraging these time-critical atomic capabilities to build composite features in an automated driving scenario.

4.2.4 QoS Aspect

In this subsection, the important QoS sub-aspects of scalability, interoperability, and privacy in service composition or decomposition are investigated. A unified approach is followed for studying these QoS properties via the SLR questions RQ13, RQ14, and RQ15, respectively.

For each concern, the context of the study is presented, challenges encountered against its realization, and how researchers tackled this concern in their service composition efforts.

This subsection will provide input for answering QoS questions RQ13, RQ14, RQ15 based on the tabulated results in Tables 4.14, 4.15, and 4.16.

4.2.4.1 Scalability

Scalability in the context of capabilities composition is the ability of a particular IoT composition platform, composition algebra, or a composition technical implementation, to function properly regardless of the number of composite capabilities [24]. Table 4.14 shows the capabilities compositions scalability challenges and the different ways researchers addressed those challenges. Data in Table 4.14 is extracted from primary studies that not only discussed scalability challenges in the context of service composition but also provided solutions to address those challenges. Table 4.14 will also help address RQ13, i.e., recognizing the main scalability challenges and adopting solutions to improve scalability in composition platforms.

TABLE 4.14: Scalability challenges and solutions in IoT/CPS service composition.

Ref	Context	Scalability Challenges	How Addressed
[171] [169]	IoT Composer scalability in smart homes	Formal verification scalability challenge: number of states.	• Formal verification: CADP scales well as it can handle hundreds of parallel compositions.
[112] [113] [307]	NIST CPS Framework Scalability Constraints	Billions of connected IoT devices present scalability challenges: (i) Network performance (ii) Authenticating large numbers of users	• Building IoT/CPS capabilities composition platforms with the NIST CPS Framework guidelines in mind, especially those related to trustworthiness, would make these platforms more scalable.
[24] [25] [22]	DX-MAN scalability compared to existing paradigms	Composing IoT systems from an ultra-large number of services	• Scalability requirements: (i) explicit control flow (ii) distributed workflows (iii) location transparency (iv) decentralized data flows (v) separation of control, data, and computation; (vi) workflow variability
[199]	Testing WS-BPEL compositions under load (Number of Requests).	An online travel recommender tool was used to test the WS-BPEL compositions triggered by the user requests; each represents a composition of user travel criteria.	• The system scalability and performance were satisfactory for small-scale compositions • For higher load compositions, load balancing was proposed as a solution.
[267]	IoT Big Data Analytics Platform	Variable number of devices, services, and users	• Data management, storage, and processing services need to be dimensioned dynamically
[261]	secureSVM: a privacy preserving Data Distribution Platform for ML Data Training and Composition	Training data originates from multiple providers. This can overwhelm training/composition algorithms.	• A Gradient Descent Optimization algorithm was adopted for training and composing data, making the platform scale with many data providers.
[255]	Composition algorithms that scale for services that capture qualitative user preferences.	Proving that using qualitative preferences can help improve the quality of the generated composition algorithms, including their ability to scale.	• Integrating CP-net-based reasoning with traditional approaches to composition would improve the scalability of the composition processes.
[213]	Wearable health devices' scalability constraints	(i) High cost of wearable devices (ii) When widely adopted, computing environments might not handle the large volume of generated data, especially when aggregation is required.	• Picking cost-effective devices. • The computing architecture consists of a web service that leverages Edge, Fog, and Cloud layers to achieve scalability with real-time and computation-intensive constraints.

4.2.4.2 Interoperability

Enabling the full compositionality potential of the future CPS and IoT platforms requires enhanced interoperability between software and hardware elements, supported by new reference architectures and common definitions and lexicons [109][152]. Addressing this challenge requires broad collaboration to develop a consensus around key concepts and build a common understanding of the underlying composition technologies[19]. Interoperability in the context of IoT composition refers to the ability of a certain platform to compose capabilities from devices with different data models or that, in general, are not compatible or not built for straightforward composition.

Table 4.15 aggregates efforts that addressed interoperability aspects, challenges, and potential solutions; as a result, it will help answer RQ14, i.e., understanding interoperability challenges and adopting solutions to improve interoperability aspects in composition platforms.

TABLE 4.15: Interoperability challenges and solutions in IoT/CPS service composition.

Ref	Context	Interoperability Challenges	How Addressed
[235]	Smart End-to-end Massive IoT Interoperability, Connectivity and Security(SEMIoTICS)	- Technical: various device interfaces - Syntactic: various data formats or encoding - Semantic: various data models/ontologies - Organizational: heterogeneous APIs prevent communication between organizations.	• Standard device interfaces would address technical challenges. • Unified data formats/encoding would address syntactic challenges. • Unified data models/ontologies address the semantic concerns. • Standard APIs would solve organizational concerns.
[116]	DIY IoT Networks and Architectures	- Heterogeneous IoT systems and Platforms APIs.	• Interoperability in IoT across stakeholders and producers/consumers will only be achieved if standardized interfaces are provided.
[40]	Fair VS Full interoperability in ARM-enabled IoT Architectures	-IoT Systems with different Architectures.	• IoT ARM is a tool that allows fair interoperability by enabling bridges between systems that don't share the same architecture.
[112] [113] [307]	NIST CPS Framework enabled Platforms	- Heterogeneous components and systems. - Data Interoperability issues.	• External interoperability achieved by standardized APIs. • Providing a common language for describing composite CPS.
[152]	Gaiasense: Smart Farming (SF) IoT Platform.	- Lack of Interoperability APIs due to the cost of networks and sensors needed to allow communication between SF systems	• The "Data Interoperability Zone" and the "Information Management Adapter" are introduced to facilitate data interoperability between SF systems within an NGSIv2-FIWARE implementation.
[157]	VITAL-OS: Operating System for Smart Cities.	- Semantic (Data-Models) and organizational (Cross-domain) interoperability challenges	• VITAL leverages W3C SSN-compliant semantics (including JSON-LD, known for its flexibility and simplicity) to ensure interoperability across diverse IoT streams and domains.
[267]	IoT platform for composing analytics data.	-Unstructured data -Heterogeneous Semantics	• Semantic Interoperability and Data Homogenization techniques exploit both structured and unstructured data.
[109]	Fiware Composite IoT Applications	- Heterogeneity of communication protocols and data formats.	• Lightweight and reusable interoperability models that support a broad range of applications.
[268]	Interoperability and composability of IoT Web Services	-Traditional way of ensuring network interoperability (using standards) has shortcomings, including a lack of compatibility between some standards and specifications.	• A Technique that wraps service semantics into middleware at the application layer, which automatically builds APIs allowing interoperability without modifications to existing standards, devices, or technologies.
[11]	SenaaS: Event-driven IoT Sensor Virtualization/Cloud Approach	- Technical interoperability challenges related to the heterogeneity of IoT devices.	• A hard-coded adapter is used to mitigate the diversity issues related to sensor platforms (Technical) by selecting only compatible devices. Automated selection is planned for future work.
[153]	Composition of heterogeneous IoT services in smart homes.	- Heterogeneous devices come with different communication APIs and supported protocols.	• Using a unified communication protocol (UPnP) at the pervasive device layer to achieve communication between the heterogeneous smart home devices. • Once communication complexity is handled at the device layer, an AI planning technique is used to compose services during runtime.
[19]	Interoperability challenges when composing smart-city services in a multi-platform environment	- Lack of interoperability among IoT smart city platforms and services prevents IoT from reaching its full potential as computation load distribution is not managed efficiently.	• Analyzing the individual performance of a single IoT platform (normal workload, parallel services workload) and the aggregation of requirements when binding all platforms services within a smart city. • A Poisson Distribution function was provided to estimate load and ensure that one platform interface doesn't carry most of the load.

4.2.4.3 Privacy

For IoT/CPS to be trusted by different stakeholders, privacy must be preserved when sensitive and Personally Identifiable Information (PII) is exchanged [154]. IoT and CPS capabilities are typically associated with privacy requirements, especially in the case of applications that involve the collection of personal data, hence, there is a need for anonymization techniques as well as techniques for encryption and secure data storage. In [60], privacy is also considered the key to maintaining the success of capabilities composition in the cloud and its impact on sharing information for social networking and teamwork on a specific project. One way to maintain this

success is to allow users to choose when and what they wish to share, in addition to allowing encryption and decryption facilities to protect specific data.

Table 4.16 is an aggregation of primary studies that discussed the capabilities composition privacy aspects, challenges, and solutions adopted to address privacy. Data in Table 4.16 will constitute a basis for addressing RQ15, i.e., recognizing privacy challenges that arise while composing capabilities in IoT platforms and understanding the different solutions the researchers adopted to improve privacy and tackle its challenges.

TABLE 4.16: Privacy challenges and solutions in IoT/CPS service composition.

Ref	Context	Privacy Challenges	How Addressed
[235]	Privacy aspects and dependencies in IoT and Industrial IoT Platforms	- Composing and facilitating the design of IoT platforms that are secure and privacy-aware.	• Privacy in IoT composition platforms is defined as a higher property and is decomposed into specific and low-level aspects each studied thoroughly, including authentication, authorization, data protection, unobservability, anonymity, unlinkability, undetectability, pseudonymity.
[244]	Privacy-aware IoT-A based FIWARE cloud IoT platform.	- Allowing trusted devices and objects to anonymously join and be composed within IoT-A compliant platforms (FIWARE).	• The Security Functional Group (FG) handles security and privacy issues in IoT-A-compliant IoT systems. • The Identity Management Functional Component (FC) within the Security FG issues/manages pseudonyms/accessory information to trusted subjects, thereby ensuring anonymous operations and privacy.
[116]	Privacy best practices during the development of IoT Platforms.	- Developers focus more on functionality and less on ethical values related to the use of communication technologies. - Privacy policies may be in place, but their technical implementation is neither supervised nor adhered to.	• Human-Centric Computing is proposed as a solution for developing privacy-aware platforms from the architecture/development phase. • Examples of Human-Centric Computing include the “Human Centric Systems Development Life Cycle”.
[40]	Privacy in ARM-based IoT Platforms	- Subject’s privacy is an "Element to Protect": data that a user or a device does not explicitly agree to make publicly available. It is specifically impacted when the composite system’s requirement includes non-repudiation.	• Privacy is addressed by leveraging an Identity Management component run by a trusted third party for user privacy protection and tracking malicious actions.
[112] [113]	Addressing Privacy in Cyber-Physical Systems using the NIST CPS Framework	- Certain types of CPS data may present little or no privacy concerns in isolation, but when aggregated with other data, they might become privacy intrusive.	• Privacy risk management guidelines include analyzing privacy risks throughout the entire data lifecycle: creation, collection, composition, exploitation, and disposal.
[125]	AI-driven composition and privacy validation in IoT	- Tangibly assessing Privacy is a challenge in IoT Platforms. - There is a need for IoT design and management frameworks that implement mechanisms to assess privacy.	• CompoSecReasoner addresses privacy concerns by deriving and validating privacy computation/estimation post-composition. • Privacy was computed and estimated based on tangible vulnerability, exposure, and disclosure metrics. • "Attack Surface" and "Medieval Castle" are security approaches used to derive measurements for privacy aspects based on "Attackable Points" and "Protection Levels."
[267]	Building IoT Analytics Platforms with privacy in mind	- Users need privacy policies that are permissive enough for services to work while also restrictive enough that their privacy is not compromised.	IoT Analytics platform builders must keep privacy in mind by: • Anonymization of personal data. • Encrypting and securing data storage • Implementing user-customizable data sharing mechanisms.
[261]	Privacy-Preserving Smart-City IoT Platform based on ML and Blockchain	- SVM ML classifiers require labeled IoT data that may contain privacy-sensitive elements.	• secureSVM: a privacy-preserving SVM training scheme over blockchain-based encrypted IoT data. • IoT data are encrypted and then recorded on a distributed ledger.
[180] [108]	Privacy-preserving in distributed and cloud IoT platforms using AI and Blockchain	- Data containing sensitive personal information can reveal the identity of IoT stakeholders, including consumers and producers.	• Human-centric approaches: user consent. • Technological approaches: cryptography, blockchain networks. • Regulatory approaches: GDPR. • Blockchain provides a distributed way to protect privacy, as there is no centralized entity to manage the credentials of devices and stakeholders.
[285]	Consent and Trust related to Personal Data Sharing in IoT	- The IoT data volume doubles every two years, with more than 60% generated by individuals, with wearable medical devices data, particularly raising privacy concerns.	• The EU’s GPDR is proposed as a framework for addressing privacy by enforcing mechanisms of Transparency(how data are processed), consent(user’s ability to opt-in or opt-out), and erasure(right of the users to delete data).
[82] [192] [27]	Privacy-Aware Cloud or Cross-Cloud Service Composition.	- Private clouds don’t disclose QoS details of their service owing to business privacy concerns in cross-cloud scenarios. - Cross-clouds credibility might be doubted if the advertised service composition SLR doesn’t meet real specifications.	• The HIstory REcord-based Service Optimization MEthod (HireSomeII) protects cross-clouds privacy by evaluating their QoS history records rather than their advertised QoS values, which enhances the credibility of the composition plans they provide.

4.3 SLR : Discussion

Section (4.1) describes the adopted SLR methodology, and Section (4.2) presents the results with tabulated data that would help provide answers (AQs) to the pointed out SLR questions (RQs). In this section, **1)** the data from Section (4.2) is analyzed and consolidated to provide answers to SLR questions RQ1-R15, and **B)** trends, gaps, and threats to the validity of this study are discussed.

4.3.1 Answering SLR Questions

In this subsection, we provide answers to the SLR questions by referencing relevant primary studies while highlighting in figures trends for topics of interest.

4.3.1.1 AQ1: What is the motivation for native support for IoT capabilities composition/decomposition mechanisms by standards, reference models/architectures (RMAs), and frameworks?

IoT/CPS standards, frameworks, and reference architectures not only define data sharing, interoperability, and security specifications of exchanged messages over a composition platform but also incorporate mechanisms for composing or decomposing novel capabilities in different platforms.

The different standards, frameworks, and reference architectures mentioned in Table 4.3 provide motivations for native support for IoT/CPS capabilities composition/decomposition through their enabled properties. The motivations include: a) the ability to address the composition of functional, business, human, trustworthiness, timing, data, boundaries, composition, and life-cycle concerns of an IoT or a CPS, and b) taking into consideration the complexity, discoverability, adaptability, and constructivity of the composite capabilities [112][113][307]; c) enabling computation distribution of computation-intensive services running in cloud nodes to low computation nodes at the Fog/Edge level [283]; d) addressing composite services functional and qualitative properties during runtime to enable flexible and adaptable compositions [266] and e) incorporating composition-friendly ontologies and composition mechanisms in distributed environments through hierarchical structures such as classes and subclasses [231][40][105]; f) providing automatic composition mechanisms to build modular software capabilities and from heterogeneous service marketplaces and locations [146] [145] [155] [133][123]; g) enabling reusability of small, atomic, reusable components through decomposition[283] [283][244]; and h) guiding atomic service discovery, selection, and complex services prototyping and composition in the cloud environments [283] [313] [307]. Figure 4.7 summarizes the motivations above.

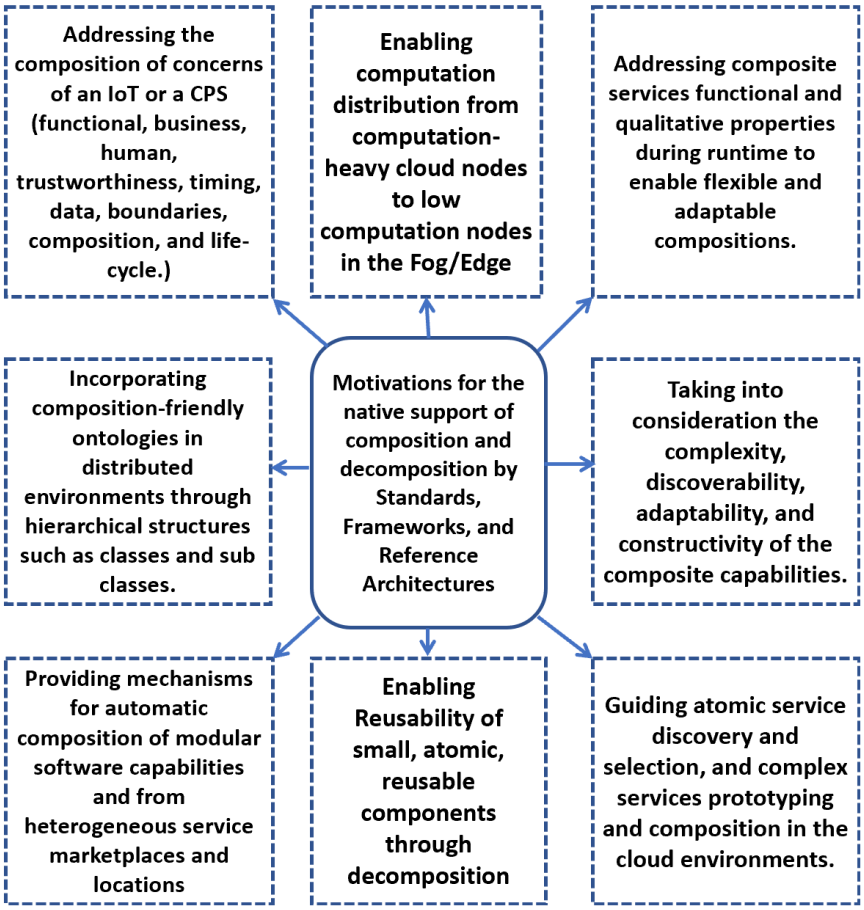

FIGURE 4.7: AQ1: motivations for the native support of service composition/decomposition mechanisms in standards, frameworks, and reference architectures.

4.3.1.2 AQ2: What are the main properties of formal representations leveraged in service composition ?

Based on data from Tables 4.4 and 4.5, formal representations have two main properties : the **a) Modeling** property, and the **b) Formal verification** property, given that the formal model can be used in formal verification tools, and as a result, the properties of interest can be formally assessed.

a) Modeling : can be performed in three ways: algebraic [171] [319] [177] [269] [32] [258] [62] [181], graphical [35][179][314], or hybrid [115][22][24][25][23] . Modeling targets two elements:
→ **Functional components** : include modeling system's components (e.g.: modeling system buffers [242]), service types (e.g.: describing and tracing atomic or composite services[301][14][245]), data format [312], composition interactions (e.g., querying and registering [308][73] [246]), composition operations (e.g., discovery [35]) , behavior (e.g., traffic congestion was described using VSA [264]), functionality (e.g., actuation, communication, controllability, manageability, measurability, monitorability, physical context, sensing, states and uncertainty [313]) or all what preceded (Using LNT to model a composite system's objects, interactions, states,

and actions [59]). It's worth mentioning that formal modeling typically preserves compositions process type: for example, in [171], the concurrent/parallel nature of the composition is preserved by the model.
→ **Non-Functional Properties** : this includes QoS properties of a composite system, including cost and price of a composition [191], energy efficiency [296] or power inefficiency [14], safety [80] (analyzed using the CWB-NC tool [291]) , privacy [154], security [232], scalability [25], interoperability [104], and system performance indicators such as response time [301].

b) Formal Verification: to perform model checking -for verifying formal properties or assessing the model against state space explosion-, the formal model needs to be loaded into a formal verification tool after translation to a formal specification. This is the case, for example, for PLUSCAL -that is converted to the TLA specification in the TLA+ tool- or LNT -a description language, which is later translated to LTS, a formal specification language- to formally verifiy the different properties of interest [215]. Formally verifiable properties mentioned in the primary studies discussed in Table 4.5 include correctness [272](verified in cloud environments using tools such as NuSMV[158]), compliance [115] , liveness (the overall study of formal verification problems such as starvation, deadlocks, and livelocks) [172], compatibility [64], non-link redundancy[187], privacy [125], security [154], safety [222], reachability [115] , livelocks [89], dependability [74], reliability (verified in PRISM [174]) [189][104], realizability [242] , deadlock freeness [42], consistency [216] , non-conflict [187], conformance [312] , completeness [216], fairness [171], trustworthiness [313] , and communication properties such as unmatched send messages [84]. It is worth mentioning that verification of some these properties can be automatically achieved when other properties are verified. This is the case in [65] where safety was verified through the verification of the non-reachability of deadlock states by leveraging the PROMELA formal specification.

Figure 4.8 presents the main properties of formal representations as learned from the primary studies results in Tables 4.4 and 4.5.

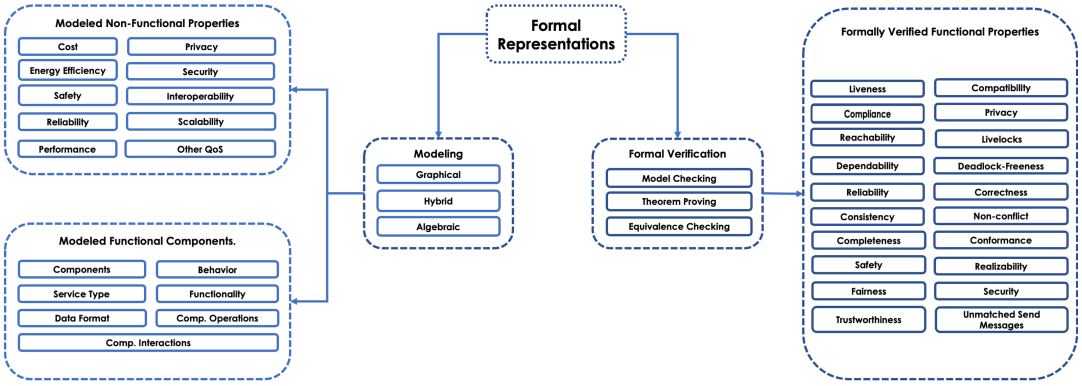

FIGURE 4.8: AQ2: Formal representations modeling and verification properties and techniques and corresponding modeled/verified properties in service composition.

4.3.1.3 AQ3: What are the current trends in formal representations modeling and formal verification?

Five trends were pointed out from the data extracted in Tables 4.4 and 4.5. These trends are represented in Figure 4.9 as follows: formally verified functional properties (A), formal verification tools (B), formal verification techniques (C), formal representations modeling approaches (D), and formal representations - used in service composition- source code availability (E).

For formal verification techniques, model checking [170][256][197][4][163][2][3] [205][57] represents the major technique for assessing properties of interest (74% of primary the studies in Table 4.5), while equivalence checking [254][83] and theorem proving [101][321] represent the remaining 26%.
70% of formal representations are of algebraic nature, while graphical or hybrid representations representing are used in 30% of the identified primary studies: the use of algebraic solutions in most primary studies can be explained by the fact that they can be easily translated to formal specifications compared to their graphic counterparts, which require additional interpretations and transformations (e.g., in [242], BPMN 2.0 is converted to LOTOS NT, and the latter is transformed to the formal specification LTS using the CADP tool).
Verifying certain properties of the composed systems is the main goal of formal verification. From this perspective, correctness remains the most verified property (%31 of primary studies), while safety, security, deadlock freeness, and reliability -combined- are addressed in 38% of the primary studies that addressed formal properties assessment. Properties such as privacy can be considered emerging properties in which researchers need to invest more effort (formally verified in one manuscript [125]), which also explains why it was studied in a specific RQ (RQ15).
In terms of formal verification tools, CADP, TLA+, Maude, CoQ, and MWB [290] combined represented 55% of the tools leveraged in the primary studies, followed by Isabelle, UPAAL, CWB, NUSMV, and PRISM (5% each). These tools are used by both the research community and industry stakeholders.

From an implementation perspective, the data in Table 4.4 shows that 75% of the formal representations can be traced to a Git repository containing substantial examples of formally specified models for different composite systems. This provides resources to inspire researchers to model and verify the properties of novel services. Figure 4.9 highlights all these trends.

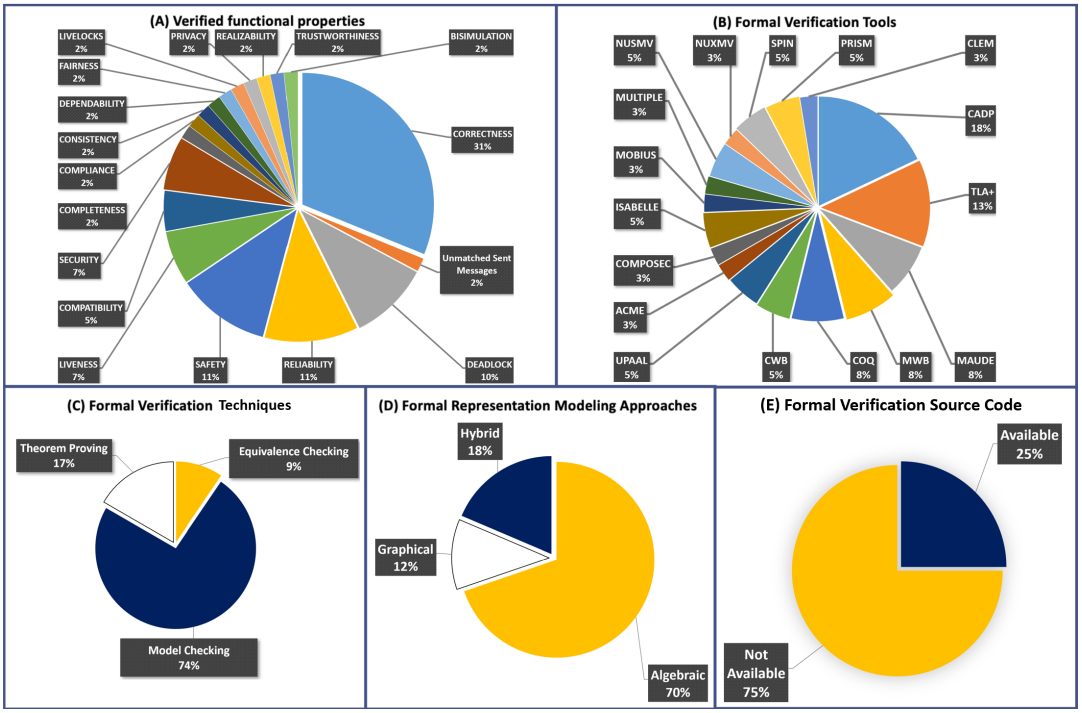

FIGURE 4.9: AQ3: formal representations: modeling and formal verification trends.

4.3.1.4 AQ4: What are the different ways formal verification techniques and tools tackled the state-space explosion problem in service composition?

Data extracted from primary studies in Table 4.6 show that the state space explosion problem in service composition can be tackled using three different approaches:

a) Model simplification:

This approach works better for models that cannot be decomposed into simpler problems. Examples of this approach include [47] (where a pi-calculus composition algorithm was simplified using an approach called unification that eliminated the state space explosion problem), [214] (where the state space was reduced by using Boolean functions instead of explicit representations with more states), [312] (where the FDR2 algorithm replaced the old FDR, yielding simpler models and as a result fewer states), [246] (where the algebra's order was reduced causing fewer states), [130] and [301] (where machine learning is used to determine the optimal service composition, which in turn simplifies the model), and [316] (the model was simplified by anticipating and eliminating errors in systems with a large state space).

b) Model decomposition:

This approach works better with models with a large state space that can be decomposed into simpler problems, making model checking much simpler and preventing state space explosion. Examples of this approach include [219], where the CADP toolbox uses software mechanisms to decompose the model to be

verified, also [187], where the size of the specification was reduced as a result of decomposing the model, and [282], where the model was decomposed into multiple sub-models, each with a smaller state space that can be checked individually, thereby reducing the state space and as a result preventing the state space explosion problem.

c) Resource management:

This approach leverages available hardware or cloud resources to alleviate the computation of formally specified models or to put up measures against eventual crashes when the state space computation exceeds the available resource capacity. Examples of this approach include [216], where the TLA specification makes use of multi-threading and allows access to multi-core processing, or by offloading and distributing computation among multiple AWS EC2 cloud instances, or [73], where the simulator manages the state space explosion problem by stopping the simulated model to prevent crashes when the space is too large, which doesn't reduce or eliminate the state space explosion problem.

To conclude this analysis, the outcomes of these three approaches vary from reducing the state space explosion problem by making the model work for low-scale models, completely eliminating it if the model's state space is small enough to not cause the state space explosion, or managing it by preventing crashes in formal verification tools. The trends in Figure 4.10 (A) show that model simplification remains a widely adopted approach to tackle the state space explosion problem, and in 65% of the cases, the problem is reduced, as Figure 4.10 (B) shows.

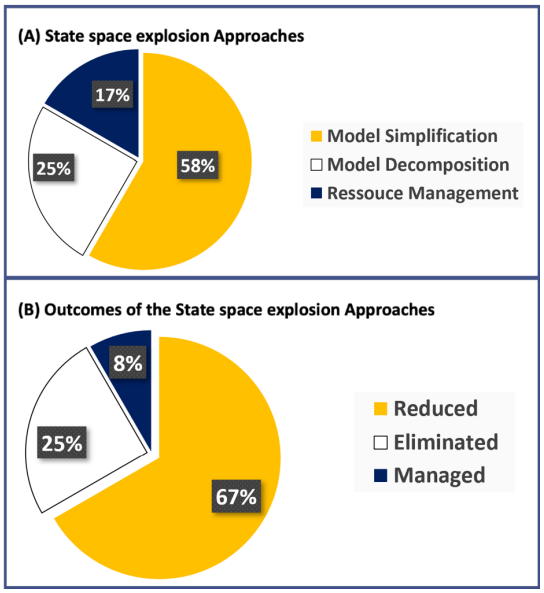

FIGURE 4.10: AQ4: trends related to the state space explosion problem approaches and outcomes.

4.3.1.5 AQ5: What are the different stakeholders' categories and concerns when it comes to composing or consuming composite capabilities in different domains?

Table 4.7 presents data from a set of primary studies that focused on different stakeholders' concerns when it comes to composing or consuming composite capabilities in different domains. To better understand these trends, we put up Figure 4.11, which matches pointed out stakeholders and domains with corresponding concerns. Although Figure 4.11 doesn't represent a full picture of the pointed out domains, stakeholders, and concerns, it highlights key relationships and synergies between these sub-aspects. Three main stakeholders were recognized: users, developers, and city planners (or platform managers), with the majority of efforts focusing on users (47%) and developers (42%) concerns.

For **Users**, the friendliness of composition platforms from a GUI perspective represents an important concern, this is the case for smart buildings applications such as [9], and this has been mentioned in other domains including smart transportation [182][118]. Customization and ease of use are also user concerns in the smart manufacturing domain [273]: customers use recommendation algorithms to customize products during pre-production in terms of cost, reliability, and delivery time, among other criteria [13][195]. Users also expect energy efficient composite systems in smart cities [281][135][58][16], less human interaction and more automation in composition platforms processes [29], ease of use of composition platforms[171][169], cost reduction or zero-cost implementation [152], trustworthiness concerns that include the human/user factor in smart manufacturing [313], accessible safety assessment especially for critical metrics such as security in smart cities [240], safety in smart transportation applications, data privacy in smart health applications [196], environment friendliness and security [240] for smart buildings and cities.

For **Developers**, most concerns relate to the ease and time to develop, customize, prototype and publish composite services. This typically requires ready-to-use APIs or at least available and less complex software objects for hardware and communication components. Ease of development, customization, prototyping, and publishing of services for developers were pointed out as concerns in applications related to the domains of smart health [196], environment monitoring [308], and smart cities [67][68]. Other developers' concerns that are worth mentioning are collaboration and interoperability in smart manufacturing [309], smart transportation [264], and environment monitoring domains [308]. These concerns contribute to a faster and easier composite service development process.

For **City Planners and Platforms Managers**, four concerns were identified: customization of services to clients, which is the case for insurance companies that can use composition platforms such as Dracena to generate costumer-specific fine-grained recommendations on insurance fees [310]. Another concern is cost reduction. An example of such concern is addressed in [68], where city planners encouraged developers to adopt modular and reusable designs for composite services for cost optimization purposes. An additional identified concern is ensuring trustworthiness in smart manufacturing. An example of this concern was mentioned in [313],

where researchers proposed a cloud manufacturing platform that takes into consideration trustworthiness pillars as indicated by standards such as the NIST CPS Framework. Finally, energy efficiency in smart cities was recognized as a major concern by both users and platform managers in [281] [135][58][16].

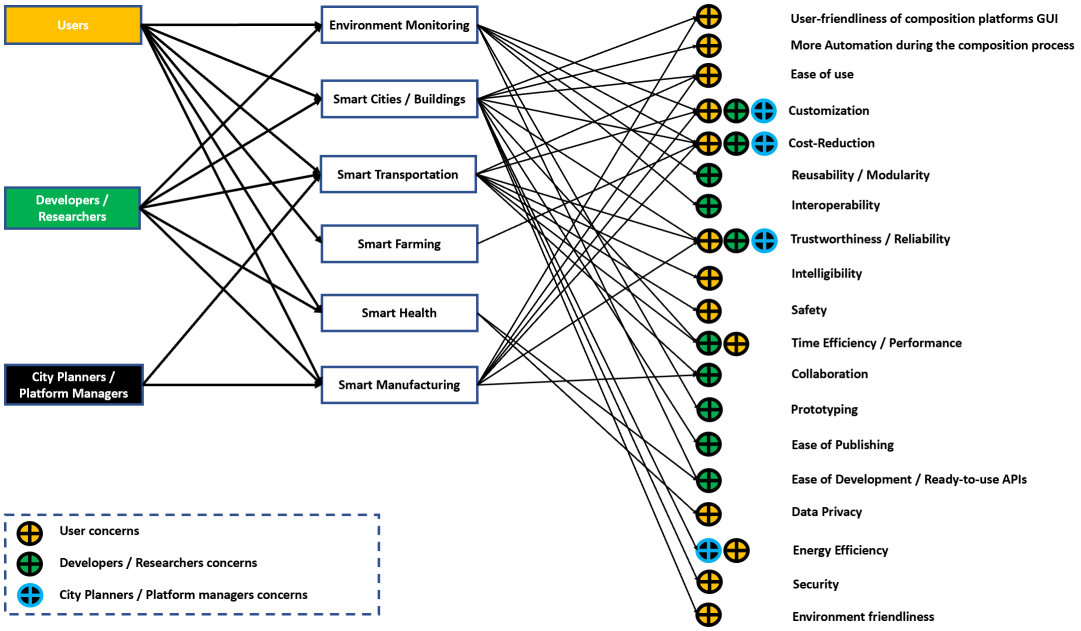

FIGURE 4.11: AQ5: primary studies trends related to stakeholders' concerns in different domains.

4.3.1.6 AQ6: What are the technical differences in capabilities composition implementation in different platforms ?

Primary studies pointed out in Table 4.8 were selected to answer this question as they provide sufficient implementation details, including where the services are implemented from an IoT layer perspective, which composition engine is used, and implementation instructions that would highlight more details about the technical differences in implementation.

From a platform nature perspective, four implementation trends were recognized:

a) Cloud/fog implementations:

The implementation of these platforms leverages cloud services, communications, and capabilities to provide value-added services. Examples of such implementations include MCC Cloudlets [202], where researchers have created virtual objects representing edge devices at the fog level. These virtual objects were later composed into applications in the Central Cloud. Networking between the edge, fog, and cloud components is ensured via OpenStack, and the repository for virtual objects is maintained at the fog level. Thing Orchestration [201], along with Cloud CAMP [44] are two other examples of composition performed at the cloud level. Another example of the use of cloud computing capabilities to compose

novel services or applications can be found in both commercial solutions: AWS GreenGrass [276], and Microsoft Azure IoT [183] [318]. AWS GreenGrass leverages Lambda scripts to compose value-added services and applications by connecting different AWS services; this requires the installation of the different GreenGrass dependencies and certificates related to the target devices and services. Similarly, Microsoft Azure IoT relies on the Azure IoT Hub cloud component to connect and compose capabilities and services at the cloud level. This also requires the installation of different Microsoft Azure components and command line capabilities. A rule-based cloud solution, IFTTT [146], has also been presented as a cloud composition service. Users can set sensing or actuation rules anywhere on their apps, and the IFTTT service composes the desired rules or outcomes. Finally, Fiware [244] is also a cloud composition platform that leverages multiple plugins, including the Entity Composer Plugin, to compose capabilities provided by devices or services at the edge or fog level, which requires special agents to be installed on edge devices or services to allow the composition of capabilities at the cloud level.

b) Edge/Local containers implementations:

Edge implementations rely on edge devices to perform a subset of compositions and computations when these edge devices satisfy minimum computation requirements. Edge computing or composition can also be leveraged to minimize delays in safety-critical applications. This is the case for [229], where mobile edge servers are proposed as a solution for composing capabilities at the edge when the edge servers have sufficient computational power. When edge servers are overwhelmed, only then do cloud center nodes take over the composition computation. This type of composition is typically suited for delay-sensitive applications such as smart transportation applications. Another example of the use of edge computing to perform service composition is highlighted in [308], where authors discussed Home Assistant: a free and open-source software for home automation designed for central control in smart homes. Home Assistant is hosted on a local machine and doesn't require internet connectivity. Edge devices and the Home Assistant node are located in the home LAN and are managed in a way that ensures privacy and security. mPlane is another measurement platform that can be used for network measurements and compositions at the edge [287] and requires a local node running the composition engine (Supervisor) on a competent machine connected via different types of APIs (TCP/IP, REST) to network probes, services, and clients. A similar concept is highlighted in [304], where the Vert.X toolkit is used to create and compose local microservices, with the ability to call remote atomic capabilities via different APIs such as Axios. Finally, the publication [14] highlights an example of edge/local composition using a composite engine (Fast IoT Composer) running on a local machine to compose smart home services.

It can be concluded that edge/container composition only serves as a substitute for cloud composition when the resources associated with edge/container nodes are sufficient to perform the composition computation. From a performance perspective, edge composition is faster and provides faster results and services than its cloud counterparts. Finally, it is also safe to assume that edge/local containers computing provides more privacy when the data doesn't leave the user's vicinity.

c) Simulation/prototyping implementations:

These platforms are used to compose simulations or prototypes for composite services and systems. Examples of these platforms include [251] [121], where a CPS representing an autonomous vehicle for safety assessment was simulated in UCEF using atomic components called federates that communicate using the pub-sub HLA protocol. Another example can be found in [264], where services and sensors can be fetched using REST APIs from Node-RED's web interface, which enables the simulation and prototyping of composite services. It is worth mentioning that Node-RED can be used for production as well for less-complex home applications, but there are other mature solutions that are well-optimized for such purposes such as Home Assistant.
Trends in Figure 4.12 show that from a platform nature perspective, cloud composition platforms represent (43%) of the primary studies identified: this can be explained by the fact that most studied compositions are computation-intensive and need resources that typically reside in the cloud. In addition, cloud platforms are mostly commercial with high maturity and reach as opposed to edge/container solutions which are mostly academic efforts. Edge or local containers composition solutions represent (36%) of primary studies, whereas service composition simulation platforms represent only (21%) of primary studies.

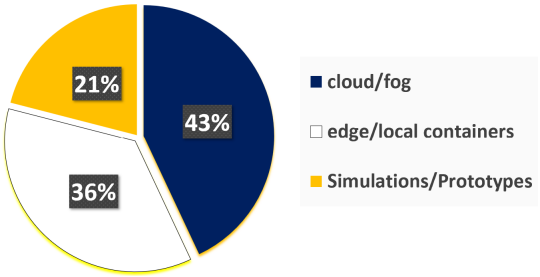

FIGURE 4.12: AQ6: implementation trends in service composition.

4.3.1.7 AQ7: What are the most common composition processes, and how do they differ in terms of automation level?

The primary studies in Table 4.9 were selected as they provided input on both the composition level and the composition process type. By analyzing Table 4.9 columns, especially the automation and process type manifestation columns, the depth of complexity and automation in each process type can be learned. Two trends were pointed out:
a) Low complexity processes => Require no/basic user input: These processes facilitate the automation of composition owing to their low complexity, and they are ready to exploit APIs and interfaces to atomic services [171][169], or by automating complex composition steps, including data collection and composition [46][264][25][221][220]. Rule-based processes can be considered the least complex of all composition process types because they only require setting up simple rules or composition goals by the user from a GUI to trigger desirable events [128][293][84].

b) High complexity processes => Require user or developer input and programming These processes have a higher level of complexity due to the fact that

they require extra programming to achieve the composition goals. Examples of such composition process types include process or programming-based composition processes, which -as the name suggests- require manual programming or crafting composition scripts to achieve the compositions goal [308]. Asynchronous or parallel compositions also show a higher level of complexity compared to their synchronous counterparts, which was indicated in [310] and [287] where additional development on specific highly programmable platforms is required. Finally, some flow-based composition processes can have a high automation level but still require user input on specific QoS parameters to successfully achieve the goal of the composition, as in [14], where user input is required during the service selection step to recognize specific atomic services that meet certain QoS properties of interest.

Figure 4.13 aggregates these trends and shows that among the 14 primary studies addressed in Table 4.9, 28% are semi-automatic for their higher complexity and their need for user or developer input, while 71% of the primary studies were automatic for their lower complexity and their need for basic or no user input to achieve the composition goals.

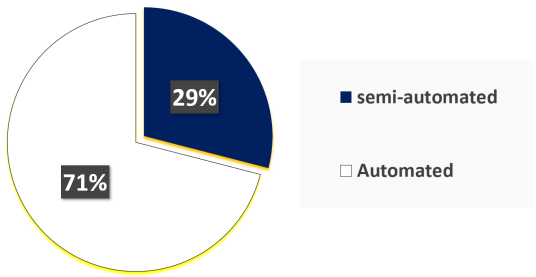

FIGURE 4.13: AQ7: trends in service composition automation level.

4.3.1.8 AQ8: What roles communication protocols play in composing or decomposing IoT capabilities ?

Table 4.10 contains the primary studies that provide insights into the role of communication networks in service composition or decomposition. Three categories of roles were pointed out.

a) Communication roles: This role is manifested in i) enabling a wide variety of services such as microservices (Vert.X's Axios, [304]) and pub/sub Java federates (HLA [251],[121]) to communicate and exchange data with composition engines in order to build composite services, or in [117], where smart connected IoT devices leverage the mashup paradigm to communicate and exchange data, leading to value-added services through compositions in WSNs; ii) facilitating the communication of heterogeneous IoT networks through transparent access (CoAP/HTTP, [149]); iii) enabling the discoverability of IoT devices capabilities (mDNS, [84]); iv) simplifying interfaces and enabling flexible network management (SDN, [169][313]); v) acting as middleware between publish-subscribe brokers and services (MQTT,[63][128]); and vi) offering commands that enable requesting, receiving, composing, and storing data by leveraging multiple network components (mPlane protocol, [287]).

b) QoS-related roles: This role is observed in i) (CoAP, [289]) which allows communication between energy and computation constrained devices, or in ii) (SVOM, a CoAP-based protocol, [289]) which captures the status of registered devices to achieve fault management. Similarly, in [224], energy efficiency was enabled in SOA composite applications in WSNs by leveraging the NOC paradigm.

c) Process roles: In addition to communication and QoS roles, two process roles were identified in primary studies that communication protocols facilitate in service composition: i) enabling asynchronous communications (REST, [315]), and ii) enabling service decomposability by tracing back interests requests to reveal which capabilities (atomic or composite) exchange data with the composition engine or clients (NDN [302][164]).

4.3.1.9 AQ9: What are the main roles of data models leveraged in service composition and decomposition ?

Primary studies used in the discussion below were selected because they provide the reasons behind using a given data model for composing capabilities. These primary studies also provided concrete examples of such data models, as shown in the last column of Table 4.11, where a subset of the data models used and their attributes are represented. Some attributes are shared by most data models, including capabilities ID, location, and timestamp. Based on the data provided in Table 4.11, the following data models and schemas roles were pointed out:

a) Modeling services, composition, decomposition, processes, and operations: The ease of composing atomic capabilities is one of the key enablers for creating value-added applications, which can only be done through the use of data models that facilitate this task by representing capabilities data and operations. This is highlighted in [286] where JSON is used to model mPlane capabilities thanks to its simplicity, parseability, and efficiency. JSON is also used to represent services, bindings between services, as well as the strength of these bindings [171][169][100]. In IFTTT [84], JSON Web of Things (JWT) is used to model objects and composition rules. The same applies to WSDL [246][156], which is typically used with SOAP and XML [234][317] schemas to describe and enable the discovery of web-services. BPEL data models describe composite services processes and enable their deployment in three steps: process template creation, process composition, and process installation. Another flavor of BPEL, WS-BPEL [200][72], describes services and their execution and compositions [280], with BPEL-TC upgrading the features of WS-BPEL to include composition and decomposition requirements for temporally customized web services [208]. OWL-S enables nested classes description to model functional and non-functional properties of IoT devices [85][303], and OWLS-TC4 provides simple capability aggregation mechanisms for non-complex compositions [294]. Finally, HSML/HTML [308] describes the state of IoT services whereas SSN [38] represents context information for capabilities including deployment attributes. The attributes of data models illustrated in Table 4.11 show that there are synergies and common attributes (id or name, timestamp or start time, location or ip, etc.) between most of the different data models when it comes to describing services.

b) Enabling formal specification: After adequate translation, some data models can enable formal specification. The translation process complexity varies in terms of difficulty from one data model to another. Examples of data models that were converted into a formal specification include JSON [264] which was used for a service description of the Node-RED object detector; which were later converted into semantically comparable service vector descriptions (VSA: Vector Symbolic Architecture). Another example of this role can is found in OWL-S [300], which enables services description (service profile, service processes) based on which a TLA formal specification is defined. Some efforts mentioned challenges that complicate the conversion of data models to formal specification: an example of such complications can be found in WSDL [246], which cannot be converted into formal specification because of its lack of implementation details and logic.

c) Facilitating interoperability between different platforms through integration APIs: To allow different platforms to compose their capabilities when their data models differ, an interoperability bridge between these platforms must be established. Data model integration APIs are one of the bridge techniques used in service composition to accommodate this requirement. Examples of such techniques can be found in [190], where JSON-LD was easily integrated with other Context Information Management (CIM) platforms, and RDF/XML (Resource Description Framework) was used to integrate APIs with external XML platforms to ensure backward compatibility. XML has also been used in mPlane [286] to allow integration with external platforms using XML data models.

d) Documenting services and debugging processes and compositions: YAML's improved human readability and writability, making it suitable for documenting and debugging [286].

e) Special roles: WoTDL [230] is an alternative to existing WoT models such as OWL-S and WSDL, which are unsuitable for describing AI planning concepts for automatic WoT compositions.

4.3.1.10 AQ10: How are atomic or composite capabilities quantified or measured?

Data from the primary studies identified in Table **??** is used to shed light on the measurability aspect of service composition. Regarding atomic or composite capabilities, the following measurability trends were recognized:

a) Atomic capabilities:
↪ The are typically concrete and tangible capabilities with standardized metrics and units.
↪ They are typically generated by devices by converting physical environment information to a digital form and associating it with a standardized unit.
↪ Cannot be further decomposed into other atomic capabilities.

The best example of atomic capabilities measurability approaches that satisfy the trends above is illustrated in [5], where functional atomic capabilities (temperature, humidity, carbon dioxide, wind speed) and non-functional atomic capabilities

(RTT, latency, and drop rate) of a thermal comfort system were measured using different weather and home sensors as well as system performance probes.

b) Composite capabilities:
↪ Are typically abstract/non-tangible capabilities; the metrics and units used are user-defined and non-standardized.
↪ Are generated by combining and aggregating atomic capabilities in a way that can be assessed by end users.
↪ Can be decomposed into atomic capabilities.

Examples of composite capabilities measurability approaches that satisfy the elements above include the two non-functional capabilities mentioned in [171][169]: i) "Tool Usability": was assessed using many atomic factors derived from the user's experience including the number of clicks to achieve a composition task and the easiness of interacting with the UI. ii) "Platform Performance": was assessed by assessing and aggregating multiple factors, including the time for the deployment and the time to generate a specification, as well as the number of states and bindings processed by the platform. Examples of functional composite capabilities include Traffic Jam Trend prediction [310] (composed based on speed and location atomic capabilities provided by the SUMO simulator probes) and Thermal Comfort [5] (composed based on atomic capabilities provided by weather sensors: temperature, humidity, carbon dioxide, and wind speed). The functional and non functional composite capabilities identified in Table **??** follow the aforementioned trends.

It can be concluded that measurability trends apply regardless of whether atomic or composite capabilities are **i) functional** (inherent features of IoT devices and represent their main output. For example, a temperature sensor provides a functional capability: measuring temperature, which is a measurement that can be expressed with a quantitative value and a unit (Celsius or Fahrenheit)) or **ii) non-functional** (QoS and performance metrics associated with the IoT devices -or the infrastructure they run on top of- rather than their main feature itself, these non-functional properties include the SLA level, latency, redundancy, scalability, interoperability, security, and privacy, among other non-functional features [299]). As for the composite IoT capabilities network infrastructure, measured aspects include service quality and network capacity [193][194] by tracking resource utilization. Other non-functional measurements in web-services platforms include performance, resource utilization, dependability, fidelity, response time, availability, and cost [134][106]. Finally, an interesting example of measuring latency in mPlane is worth mentioning as latency measurement in this example is a functional property as it represents the main feature of mPlane as a network performance measurement platform [287]. This same metric (latency) is considered a non-functional property in other platforms, such as the thermal comfort measurement platform [5] as it doesn't contribute to the calculation of comfort but rather provides an idea of the platform's performance.

4.3.1.11 AQ11: What are the benefits of building IoT platforms and complex services with decomposition in mind ?

The primary studies in Table 4.12 showcase publications focused on service decomposition. This is a particularly interesting topic that has not been investigated in

previous SLRs. Based on collected data, the following main benefits were pointed out:

a) Capabilities reusability: Platforms with decomposition support benefit from this property as a decomposed complex service into atomic capabilities can see its atomic services reused in other compositions, omitting the need for implementing redundant services and saving relative costs. This is the case in a hierarchical IoT platform [198] where complex capabilities are decomposed and reused to complete the pieces of another service with a higher complexity. Another example of reusing capabilities owing to built-in decomposition mechanisms can be found in [196]: a microservices-based platform allows larger services to be decomposed into small, focused, self-contained services with loose coupling, which facilitate their reuse. The same case applies to the use of Domain-Driven-Designs (DDD) to decompose monolithic software [37], which makes it possible to use pieces of monolithic software in other compositions. Finally, in [308], the FSM model-driven services decomposition broke their linkage, which enabled their reusability.

b) Resource optimization: The decomposition of complex capabilities leads to resources optimization not only in terms of costs associated with deploying novel services [81] but also in terms of reducing network congestion as monolithic services consume more bandwidth compared to atomic capabilities[15]. The same concept applies to [107] where computation-intensive virtualized services leverage decomposition to optimize computation. Another case of resource optimization was encountered in [14] where decomposition leads to the identifying energy-efficient atomic capabilities that can be later used to compose efficient complex services.

In addition to capabilities reusability and resource optimization (main benefits), other benefits of building composition platforms with decomposition in mind were mentioned, including **stakeholders acceptance** as described in [81] where platform users can benefit from the reduced cost of exploitation due to reusability and resource optimization. Other benefits include **improved QoS parameters** such as platforms agility, flexibility, and scalability [267]. **Improving collaboration** between capabilities has also been pointed out as a benefit of decomposition in [264]. Finally, the **traceability** of the atomic capabilities that contribute to a certain composite service was mentioned as a benefit in Information-Centric Networks (ICN) such as Named Data Networks (NDN) [302] [164].

Figure 4.14 shows the distribution of the main benefits of building composition platforms with decomposition in mind as extracted from Table 4.12. Efforts that leveraged decomposition to break complex services into atomic capabilities for reusability purposes represent 72% of primary studies, while efforts that focused on computation decomposition for resource optimization purposes represent the remaining 28%.

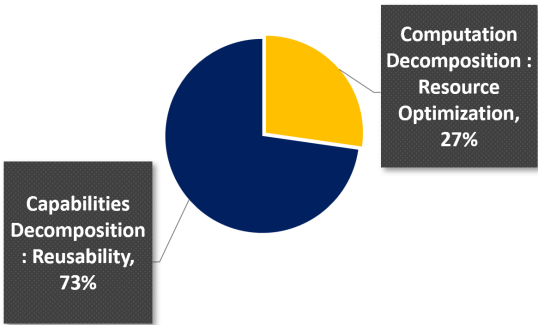

FIGURE 4.14: AQ11: benefits of building platforms with decomposition in mind.

4.3.1.12 AQ12: What role can AI/ML techniques play in shaping or improving service composition?.

The role of Artificial Intelligence (AI) and Machine Learning (ML) in service composition was not surveyed in previous service composition SLRs. We presented primary studies that tackled this particular aspect in Table 4.13; these primary studies show the roles AI/ML techniques play in service composition, including:

a) Composing capabilities with AI/ML features.

Primary studies in Table 4.13 highlight this role. In [228], AI/ML capabilities were implemented in a smart transportation platform that analyzed incoming vehicle data to build real-time automated driving features. In [67], complex atomic capabilities with AI/ML features were built and made available to developers to compose smarter systems and platforms in smart city or smart transportation domains. Examples of these atomic capabilities with AI/ML features include multiple predictors such as noise level and free parking area predictors. Another example of the role of AI/ML in building complex capabilities was referred to in [15], where a cloud facial recognition composite service leveraged AI/ML atomic capabilities at the fog level, including facial features extraction, data fusion, data filtering, and face detection algorithms. Finally, in [68], an example of reusable AI/ML smart city services was studied in the context of a collaborative platform. This platform enables the composition of smart city services based on atomic AI capabilities such as traffic and parking estimators.

b) Improving composition platforms and processes.

This is the most common role among the pointed out AI/ML primary studies aggregated in Table 4.13, which shows how AI/ML can play a role in improving composition platforms as in [253], where AI/ML is leveraged to improve maintenance of IoT composition platforms and in [247] where AI/ML techniques were used to improve security. To improve the composition processes, in [221] and [220], an AI-based classification method applied to a pipeline consisting of ML services was used to maximize the classification accuracy, leading to an improved service selection process. Similarly, a service composition technique called AI Planning was used in [90] for automatically composing well-described web services into feasible workflows and for selecting and organizing web services based on location. [88] is another primary study where genetic algorithms were used in SOA-based

environments to perform service selection based on QoS properties defined by the user. Finally, another example of AI/ML-based service selection was presented in [33], where a reinforcement learning agent was used in the dynamic of Mobile IoT to perform service selection based on user-defined criteria (including spacial cohesiveness, number of handovers).

A secondary role of AI techniques in service composition was discussed in [125], the CompoSecReasoner framework leveraged AI algorithms to assess, monitor, and verify properties such as security, privacy, and dependability. The components of the CompoSecReasoner framework also ensure dynamic system composition verification, property validation, and metrics-based automated administration.

Based on the primary studies in Table 4.13, Figure 4.15 shows the distribution of the main roles of AI/ML techniques in service composition: i) improving composition processes (67%) and ii) composing AI/ML enriched or improved capabilities (33%).

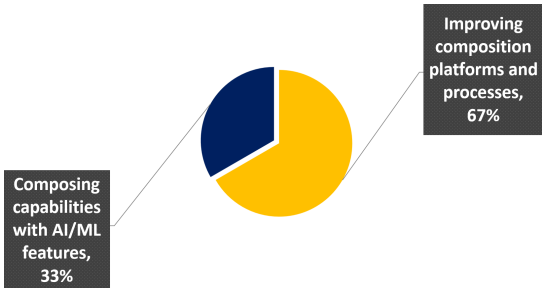

FIGURE 4.15: AQ12: identified roles of AI/ML in service composition.

4.3.1.13 AQ13: What are the main scalability challenges and solutions adopted when composing IoT and CPS capabilities?.

As novel services with high value require the composition of multiple atomic capabilities, scalability can arise as a challenge if the platform fails to scale for performance, QoS, network, or other constraints. Based on data from Table 4.14, the **growing number of services, users, and providers, generating or requesting composed and value-added services** represent the main causes for scalability issues. These challenges and potential solutions are discussed to provide answers to RQ14.

- **a) Scalability Challenges in service composition**

In [267], IoT BigData analytics platform scalability was impacted by a large number of devices, services, and users. Similarly, a large number of wearable smart health devices cause scalability issues not only from a computation perspective, but also from a budget perspective [213]. For [261], the training data originating from multiple providers overwhelmed the training and composition algorithms in the secureSVM platform. Similarly, large user requests are considered the root for overwhelming a WS-BPEL-based online travel recommender [199]. Another example of the scalability issues caused by high numbers of IoT objects was mentioned

in [171][169], which not only impacted IoT smart home applications deployments but also generated a large state space that caused the state space explosion problem when formally verifying composite capabilities properties. In [255], composition algorithms that did not scale for reasons associated with large user quantitative constraints were the subject of analysis. Finally, ultra-large numbers of services were also identified as the root for scalability issues in [24][25][22].

• **b) Scalability Solutions in service composition**

To address scalability challenges in the above primary studies, researchers have adopted the following solutions:

i) Properly adjusting and dimensioning resources: An example of such a solution was adopted in [267], where researchers proposed a dynamic solution for dimensioning resources, including storage and processing, to anticipate scalability issues. Another example of such an approach was pointed out in [199], where researchers proposed redundancy and load balancing at the level of computation and network resources to tackle higher loads and user requests for composite services.
ii) Adopting optimized and enhanced composition algorithms to overcome scalability issues: This is the case in [255], where CP-net algorithms were proven to have a better capacity to handle compositions that require a large number of user constraints. Similarly, an optimized Gradient Descent Optimization algorithm was used in [261] for training and composing data, creating a platform that scales with a large number of data providers. Another example was identified in [171][169], where CADP's parallel algorithms and composition capabilities were leveraged to overcome scalability issues when formally verifying properties of interest.

iii) IoT capabilities layers, data, computation, and workflows management: Adopting a layered architecture that distributes computation across different layers (cloud, fog, edge) was adopted as a solution in [213] to handle the large amount of data generated by smart health wearable devices. The research around the DX-MAN composition platform also provides an example of this solution [24][25][22] where researchers highlighted scalability requirements that, when applied to data and workflows associated with IoT devices, contribute to better scalability, these requirements are: explicit control flow, distributing workflows among multiple compute nodes, localization transparency, decentralized data flows, separation of control, data, and computation, and workflows variability.

iv) Building IoT composition platforms based on scalability-aware frameworks: frameworks with scalability recommendations and guidelines can help researchers, developers, and composition platforms providers build platforms that scale from the get-go. An example of this approach is illustrated in the NIST CPS Framework [112][113][307], where building IoT/CPS capabilities composition platforms with the NIST CPS Framework guidelines in mind -especially those related to trustworthiness- would provide elements to enhance scalability in terms of network constraints and in the case of large numbers of users.

4.3.1.14 AQ14: What are interoperability challenges and solutions when composing capabilities from heterogeneous environments?

Capabilities in IoT or CPS may originate from heterogeneous devices and services. This can create a challenge when attempting to compose these capabilities to

innovate value-added capabilities. Primary studies in Table 4.15 discussed both challenges and solutions to interoperability. These aspects are presented below to provide answers to RQ14.

- **a) Interoperability challenges in service composition**

Two main challenges to service composition interoperability were identified:

i) Heterogeneous services, systems, networks, and components: lead to a lack of communication/network interoperability. This was recognized as a challenge in multi-platform environments discussed in [19], where the lack of interoperability impacts load distribution. The same issue was noticed in the NIST CPS Framework enabled platforms when it comes to heterogeneous components and systems [112][113][307] or IoT Systems with different architectures [40]. Similarly, researchers in [235] and [152] mentioned two challenges, one technical, related to the various device interfaces in heterogeneous IoT systems and platforms APIs, and the second organizational, illustrated by the heterogeneous service APIs that prevent communication between organizations. Another case of this challenge was identified in [109][153] related to the heterogeneity of communication protocols and the challenges it represents to interoperability, and in [116] where heterogeneous APIs of certain IoT systems and platforms contribute to the lack of interoperability and as a result constitutes challenges to service composition.

ii) Heterogeneous and unstructured semantics and data: Unstructured and heterogeneous data and data models were exposed as a challenge in the NIST CPS Framework [112][113][307], and in [267] as a challenge to easily compose useful analytics in IoT platforms. Similarly, the researchers in [235] and [157] discussed syntactic and semantic challenges, including various data formats, data encoding, data models, and ontologies as a hindrance to composing cross-domain IoT services. The lack of compatibility between data formats and data models standards and specifications also falls under the same umbrella as discussed in [268] and [109].

- **b) Interoperability solutions in service composition**

Two main solutions were identified to address the aforementioned interoperability challenges.

i) Standardized or custom communication APIs and suitable interfaces were identified as a solution for bridging interoperability gaps and challenges in [109] [235] and would solve related technical and organizational concerns. In [19], understanding the load each platform interface handles in a multi-platform environment and providing suitable interfaces that fairly distribute this load is key to improving interoperability between the different platforms. In [116], providing standardized interfaces was pointed out as a measure to achieve interoperability in IoT across stakeholders and producers/consumers. Another similar solution was identity in ARM-based IoT platforms [40], by using the IoT ARM tool to allow fair interoperability by enabling bridges between systems that don't share the same architecture. [152] is another effort that adopted the same strategy; the "Information Management Adapter" was introduced to facilitate interoperability

between smart farming systems within an NGSIv2-FIWARE implementation, and in [11] a hard-coded adapter was used to mitigate the diversity issues related to sensor platforms by picking compatible devices. Finally, standards such as the NIST CPS Framework [112][113][307] proposed standardized APIs to achieve external interoperability and, as a result, facilitate service composition in cyber-physical systems.

ii) Standardized or common description languages and data semantics: is also a technique adopted in multiple efforts such as [90] to ensure capabilities are composable by leveraging data homogenization techniques to exploit both structured and unstructured IoT devices data in analytics. In [109], lightweight and reusable interoperability models were used to support the composition of a broad range of applications, and in [235], unified data formats/encoding were leveraged to address syntactic and semantic challenges to facilitate service composition. Similarly, researchers in [157] leveraged W3C SSN-compliant semantics (JSON-LD) in the VITAL platform in order to ensure interoperability across diverse IoT streams and domains, and in [152] a "Data Interoperability Zone" was introduced to ensure data interoperability between smart farming systems within an NGSIv2-FIWARE implementation. In [268], a technique that wraps service semantics into middleware at the application layer automatically generates APIs allowing interoperability without modifications to existing standards, devices, or technologies. Finally, The NIST CPS framework [112][113][307] proposed providing common languages for describing services to ensure an easy composition of CPS capabilities. Figure 4.16 shows the above trends related to interoperability challenges and solutions in service composition.

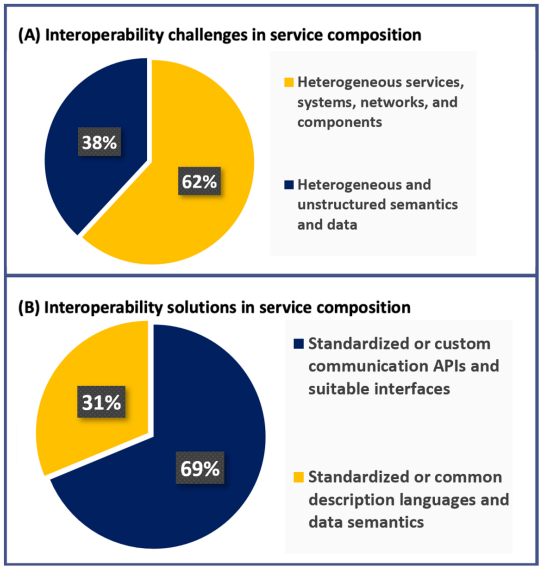

FIGURE 4.16: AQ14: interoperability challenges and solutions in service composition.

4.3.1.15 AQ15: What are the main privacy challenges and solutions in service composition?

Primary studies in Table 4.16 focused on privacy challenges and solutions in service composition.

- **a) Privacy challenges in service composition**

Four categories of privacy challenges were identified:

i) Non-trusted objects/Devices Challenges: non-trusted objects anonymously joining composition platforms can lead to privacy issues when collecting or processing IoT Data. An example of this challenge was highlighted in IoT-A-based composition platforms such as Fiware [244].

ii) Data Challenges: data in IoT or CPS composition platforms can reveal private information when labeled, exchanged, or composed. In [112][113], the NIST CPS framework highlighted the privacy threat that originates from composing or aggregating certain types of CPS data, which may present little or no privacy concerns in isolation. Another example of data challenges leading to privacy concerns is found in [261], where the use of labels on data to allow Machine learning classification might compromise privacy if those labels contain privacy-sensitive elements. In [180], the ever-growing number of IoT devices generating privacy-sensitive information is considered a privacy concern if the data are not properly processed throughout its life cycle: a Special case of this issue is discussed in [285] where the ever-growing number of medical IoT devices constitutes a privacy sensitive challenge if the data are not properly handled.

iii) Platforms Design challenges: platforms that lack privacy components by design are the most vulnerable to different privacy challenges as discussed in [235]. In other instances, Privacy policies might be in place, but their technical implementation is neither supervised nor adhered to by developers, who typically focus more on functionality and less on ethical values related to the use of communication technologies [116]. Cross-domain IoT platforms can also advertise good QoS metrics, including privacy, but these metrics might not be credible, especially when profitability is threatened [82], and In [125], researchers stressed the need for IoT design and management frameworks that implement mechanisms for assessing privacy in a tangible fashion.

iv) Legal challenges: This case is particularly crucial when the composite system's requirements include non-repudiation [40]. This goes against the users' or devices' privacy which is an "Element to Protect".

- **b) Privacy solutions in service composition**

Three solutions were pointed out to address privacy challenges in service composition:

i) Service and component-based solutions : By implementing components and services that enable, manage, and assess privacy within composition platforms. In [244], an IoT-A-based IoT composition platform (Fiware) leveraged the "Identity Management Functional Component (FC)" within the "Security Functional Group" to issue pseudonyms and manage accessory information to trusted subjects to ensure anonymous operations and as a result protect privacy. In [40], an Identity Management component, ran by a trusted third party, was leveraged for user privacy protection and for tracking malicious actions. Similarly, the CompoSecReasoner [125] component addressed privacy through the computation and validation of privacy metrics post-composition. Privacy was computed/estimated based on tangible metrics associated with vulnerability, exposure, and disclosure.

ii) Best practices, Standards, and Regulatory solutions: this solution is illustrated in [112][113], where the NIST CPS Framework provides privacy risk management guidelines, including the analysis of privacy risks throughout the entire data life-cycle: creation, collection, composition, exploitation, and disposal. In [180] and [285], The EU's GPDR is proposed as a framework for addressing privacy by enforcing mechanisms of **Transparency** (how data are processed), **Consent** (user's ability to opt-in or opt-out), and **Erasure** (the right of the users to delete data). Human-Centric Computing, proposed in [116], also proposed architectural best practices for developing privacy-aware composition platforms from the development phase, and in [180][116], user consent was highlighted as an important component to protect users' privacy when accessing IoT cloud platforms. Similarly, researchers in [82] proposed a history record-based service optimization method (HireSomeII) that protects cross-clouds privacy by evaluating their QoS history records instead of relying on their advertised QoS values, which would enhance the credibility of the composition plans they provide. Another effort that tackled the problem of protecting user privacy in cloud platforms while enhancing other QoS parameters was mentioned in [27], where researchers used privacy preserving mechanisms to rank compositions based on there privacy preservation level. Other best practices that enhance privacy in IoT composition platforms were discussed in [267], where researchers highlighted recommendations including the anonymization of personal data, encrypting and securing data storage components, and implementing user-customizable data sharing mechanisms

iii) Technology-based solutions: Including **blockchain, encryption, and cryptography** as discussed in [261] and [180], where IoT data was encrypted to preserve privacy in an ML/BLockchain based smart-city composition platform, and in a cloud IoT platform. Blockchain have also been used in [261] and [180] to protect privacy by either storing sensitive data on a distributed ledger, making it difficult to trace; or by leveraging the decentralized nature of the blockchain to avoid the case of a single entity managing devices and stakeholders credentials. **Decomposition** of privacy into atomic properties is another technical solution leveraged in [235] to address privacy concerns in IoT composition platforms. Researchers have decomposed privacy into atomic problems or properties, including authentication, authorization, data protection, unobservability, anonymity, unlinkability, undetectability, and pseudonymity. Each of these low-level atomic properties was studied individually and thoroughly to improve privacy. Figure 4.17 shows trends related to IoT/CPS service composition privacy challenges (A) and solutions (B).

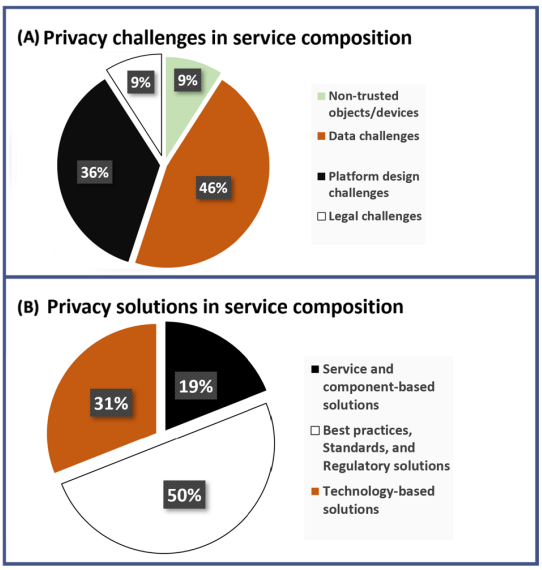

FIGURE 4.17: AQ15: privacy challenges and solutions in service composition.

4.3.2 SLR Trends, Gaps, and Threats to Validity of the study

4.3.2.1 SLR Trends

We can recognize what's trending in a certain topic and how important it is by running an advanced search -using different flavors of each sub-aspect's vocabulary- on the pool of primary studies (182 publications) using the Adobe Advanced Search tool.
The advanced search parses primary studies and searches for keywords related to the different taxonomy sub-aspects -mentions in the bibliography not considered- including some that were not addressed in this SLR. As opposed to the discussion section, we include in this analysis primary studies that not only discussed an aspect thoroughly but also papers that partially addressed it while discussing other sub-aspects. These trends are illustrated in Figure 4.18.
For the Formal aspect, almost 99% of primary studies mentioned a framework, a standard, or an architecture, which shows the importance of these components in guiding service composition. Composition algorithms -although not addressed in this SLR- are also a major aspect of the discussed primary studies, with more than 70% of primary studies mentioning at least one composition algorithm.
For Technical sub-aspects, composition process type and data models are the most discussed aspects (more than 95% of primary studies), whereas service discovery and service selection were discussed in only 26% of primary studies or less. A reason for this is that we didn't pick primary studies based on these sub-aspects' keywords.
Finally, for the QoS aspect, cost, security, and reliability were mentioned in 56% of the primary studies; although they weren't the main subject of this study, this shows their importance and involvement when discussing other service composition sub-aspects.

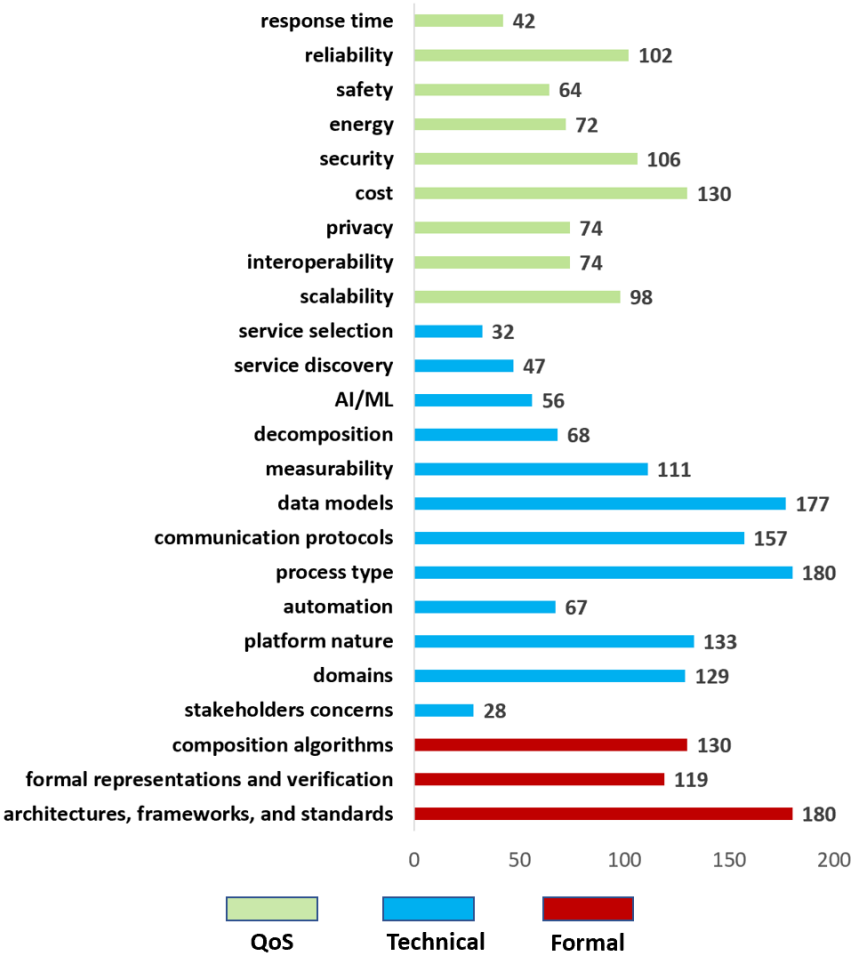

FIGURE 4.18: Primary studies trends in IoT/CPS capabilities composition and decomposition based on the number of primary studies that partially or thoroughly addressed a particular formal, technical, or QoS sub-aspect, including non-addressed sub-aspects in this study (Other Formal, Other Technical, Other QoS.

4.3.2.2 SLR Gaps

As mentioned in the trends section and based on the taxonomy proposed in this SLR study, many formal, technical, and QoS sub-aspects were not addressed through SLR questions but rather discussed partially within other sub-aspects.

For the Formal aspect, although composition algorithms were extensively discussed in [225] -specifically on how to leverage meta-heuristics algorithms to efficiently select capabilities based on user-defined QoS parameters-, service selection remains only one of many service composition steps. An SLR question that addresses how different service composition algorithms efficiently intervene during the other composition steps is a topic researchers need to invest effort in.

Regarding the Technical sub-aspects of service composition, automation and process types in service decomposition, and the level of automation of each step of the composition process are worth investigating by researchers, and we consider

these topics as open issues.

For the QoS aspect, the security of composed systems was addressed in many non-SLR/Literature publications, including [125], where authors discussed the security of IoT systems of systems (SoS) and highlighted the importance of calculating the security level of the final/composed system, taking into consideration the security of its subsystems. However, security-specific SLR questions, such as the security mechanisms required during each composition or decomposition step, were not addressed in this SLR study or in other SLR studies, which makes it a gap worth filling by the research community.

4.3.2.3 SLR Threats to Validity

In this paragraph, possible threats to the validity of this SLR are presented while mentioning some corrective strategies.

For the document sources, only SCOPUS and Google Scholar databases were queried; however, SCOPUS alone generates results from more than 5000 publishers (including the main publishers), which should provide substantial results along with the search performed -for completion- in Google Scholar.

For the SCOPUS and Google Scholar search strings: they were designed in a way that produces as many results as possible, with extra keywords added to filter specific sub-aspects questions. Not using all synonyms for a certain sub-aspect might result in missing a certain study. The search string automates the selection process as much as possible, but given the large number of papers and the different addressed questions, human error/bias during the non-automatic phases of the search process (manual selection, forward and backward snowballing) cannot be completely ruled out.

For the selection procedure, it was partially automated as some stages require researchers' involvement and refinement (snowballing forward and backward and manual evaluation of the large number of initial results from both databases). The manual stages were repeated to minimize error and bias.

As for the possibility of false negatives, which could cause the exclusion of important studies, we ran the search strings multiple times and during multiple periods while conducting this study, which would reduce the chances of excluding important manuscripts.

As for the focus of the study, this SLR did not exclude non-academia efforts and cited not only scientific and academic publications but also industry solutions (especially in the platform type sub-aspect) to provide comprehensive results.

4.4 Conclusion

In this section, we provide a summary of the contributions presented in this chapter, explain their benefit to different stakeholders, and highlight our ongoing and future work.

4.4.1 Summary

This chapter provides an SLR study that addresses formal, technical, and QoS sub-aspects in the topic of IoT and CPS capabilities composition and decomposition using the rigorous SLR methodology and based on gaps in previous SLR efforts. From hundreds of related publications, 182 primary studies were carefully selected to provide a high-quality study that answers novel and important questions. Fifteen questions related to various aspects illustrated in a potentially comprehensive taxonomy were identified and answered. The taxonomy classifies the different aspects of the topic in three main aspects: i) a Formal aspect where questions related to the role of frameworks, standards, and references architectures were addressed, as well as questions regarding formal specification and formal verification techniques, including issues such as the state space explosion. ii) A rich technical study addressed 10 Technical sub-aspects to answer 8 SLR questions related to stakeholders' concerns in service composition domains, the different service composition platforms, the relationship between service composition process types and the composition automation level, the role of communication protocols, data models, AI/ML, and decomposition in creating novel applications or reusing existing ones, in addition to investigating measurability aspects in service composition. iii) For QoS sub-aspects, three major components were addressed in SLR questions, mainly exposing scalability and interoperability challenges in service composition and how researchers addressed privacy while composing IoT or CPS capabilities. After answering the 15 SLR questions, trends and gaps in the topic were pointed out based on the pool of primary studies, and suggested a few research topics worth investigating by the research community.

4.4.2 Benefits of the study for different stakeholders

This study will benefit different stakeholders. For example, engineers, developers, and city planners will learn about the various aspects, challenges, and solutions related to the innovation of composite services or the reuse of atomic ones. This study also represents a valuable contribution -through the proposition of a potentially comprehensive taxonomy to guide research efforts- for researchers interested in this topic's formal, technical, and QoS sub-aspects. Finally, this survey educates end-users on how the composition of services drives innovation by generating value-added services and making them accessible to the general public, as is the case with well-known platforms, such as IFTTT and Home Assistant. In addition, it highlights general-public concerns that stem from exposing one's capabilities preferences, which could expose end-users to privacy and security issues. In addition, discussions in this SLR will shed more light on the ICCF framework and its IoTCaP implementations in different smart city domains.

Chapter 5

Toward an IoT and CPS Capabilities Composition Framework (ICCF).

In Chapter 1, we explained the smart-city context of the topic of IoT and CPS capabilities composition and decomposition; in Chapter 2, we introduced the main concepts around it by leveraging a layered architecture. In Chapter 3, we presented related work in systematic literature reviews and surveys -to identify the topic's gaps and non-addressed scientific questions- and in frameworks and platforms for service composition -to identify gaps in foundations, semantics, and formal verification stages while composing new services-. In Chapter 4, we performed a systematic literature review based on gaps in SLRs, and we addressed unanswered formal, technical, and QoS questions related to service composition and decomposition.

In this chapter, we introduce the IoT and CPS Composition Framework (ICCF), a framework that aims to facilitate the innovation and reuse of Internet of Things (IoT) and Cyber-Physical Systems (CPS) capabilities in various smart city domains.

ICCF is an answer to gaps identified in Chapter 3 Section (3.2) and is based on the NIST CPS Framework's recommendations and composition guidelines, intuitive composition semantics inspired by the mPlane protocol, and strong formal verification capabilities of the Temporal Logic of Actions (TLA) formal descriptors and tools. We demonstrate why such guidelines, semantics, and formal specification and verification components form a powerful and intuitive composition framework that enables trustworthy modeling and composition of capabilities in the smart-city domain and satisfies different stakeholders' concerns.

To achieve the goals of this chapter, we adopt the following organization:

section (5.1) explains the ICCF framework's required components and pillars based on gaps identified in Chapter 3 Section (3.2).

section (5.2) presents a comparative study of ICCF's pillars: guidelines and foundations frameworks, modeling and description semantics, and formal specification and verification languages and tools.

section (5.3) introduces ICCF based on the comparative study outcomes.

section (5.4) highlights a prototype of a virtual composite capability within a smart-city domain based on the ICCF pillars.

section (5.5) showcases a proposed implementation platform, IoTCaP, which implements the ICCF framework pillars and enables the composition of composite capabilities in different smart-city domains.

5.1 Introduction: Pillars for a comprehensive and trustworthy service composition framework for IoT and CPS capabilities.

IoT or CPS capabilities composition is the process of generating a value-added capability based on atomic measurements or services. Throughout this chapter, IoT and CPS can be used interchangeably [110]. A framework for capabilities composition that addresses different stakeholders' concerns would serve as a foundation for open innovation and re-purposing of IoT and CPS capabilities of an expected half-trillion IoT and CPS devices by 2030 [69]. Examples of target domains include smart buildings (well-being), transportation (safety), and healthcare (bed occupancy prediction). Verifying such novel compositions and making sure their deployment won't cause errors is crucial for a trustworthy implementation. For this reason, there is a need for a framework for composing IoT or CPS systems capabilities regardless of their complexity or domain. This work proposes an **I**oT and **C**PS **C**omposition **F**ramework (**ICCF**) that addresses these goals. Such a framework should lay the groundwork for composition and trustworthiness assessment, use straightforward semantics to help developers prototype novel capabilities, and describe tools for the formal verification of such novel composite capabilities.

We prove in this section why equipping a framework with composition foundations and adopting expressive semantics that can be easily translated to a formal specification for verification purposes constitute strong foundations for a robust composition framework that allows reliable prototyping of IoT and CPS capabilities.

Building novel services in smart cities requires awareness of different stakeholders' concerns and systems criteria. Being aware of such concerns and criteria requires deep expertise with IoT and CPS frameworks and systems while making sure a comprehensive outlook of the stakeholders' needs is adopted.

Satisfying different stakeholders' concerns when proposing a novel composition framework means relying on a knowledge database that provides capabilities composition foundations, which enables the satisfaction of all stakeholders' concerns. The expression of these concerns while modeling or implementing the novel service is the reason why it's important to adopt a comprehensive knowledge database or foundation source. For example, users of a novel capability should be able to tune it in a way that satisfies their needs, a well-being capability in a smart building can be a subjective concept (a temperature value might not be comfortable

across different users), and being able to satisfy different users outlook on the same concept is a crucial element when attempting to build a comprehensive framework that addresses thermal comfort.

Once the right foundation source or knowledge database is identified, choosing the right semantics to express and model the objects, mechanisms, and operations of the composite capability is the next crucial step. The model, at this point, should be expressive and user-readable. The expression of the stakeholders' concerns needs to be enabled by the chosen semantics: the expression of the importance of a certain component of a composite capability can be achieved through weighting mechanisms, and the semantics must enable algebraic expressions that allow that weighting mechanism to take place.

Semantics needs to be composition and algebra friendly, lightweight, and expressive. The algebraic interpretation of the semantics is important in enabling measurability of the composite capability down the road and transitioning from a model to a verifiable specification.

Semantics should describe and model atomic components of the composite service, operations on services, including service selection and discovery, composition and decomposition operations and operators, as well as interactions between users and services.

After performing semantic modeling and description of the composite service, translating the semantic model into a formally verifiable specification is important when it comes to services for which the range of values must be contained within user or system acceptable envelops or thresholds.

Based on the discussion above and the gaps identified in existing service composition platforms and frameworks (check Chapter 3 Section (3.2)), the design and prototyping of novel IoT and CPS capabilities requires foundations, modeling, specification, verification, and implementation components:

a) Foundations and Guidelines: Discuss Guidelines related to the desired composite service in a way that meets different constraints and stakeholders' concerns.

b) Modeling Semantics: Leverage expressive semantics to describe the Composite Service's objects and interactions based on The defined guidelines.

c) Formal Specification: Translate composition semantics and operations into formally verified semantics.

d) Formal Verification: Verify through symbolic execution or model checking the state space of crafted specifications of the Compositions.

e) Implementation: Implement the verified composite service in a platform that takes into account identified foundations and semantics.

Figure 5.1 highlights the main pillars to be considered when constructing a trustworthy framework for IoT and CPS capabilities composition and decomposition: i) composition foundations and guidelines, ii) composition modeling semantics, and iii) composition formal verification languages and tools.

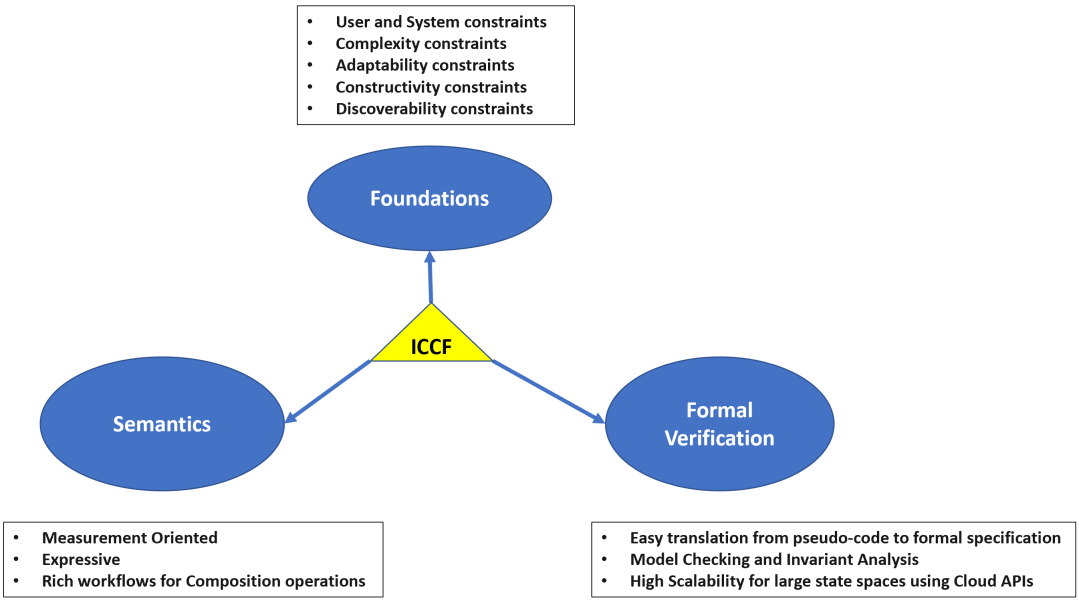

FIGURE 5.1: ICCF foundations, semantics, and formal verification pillars (foundations, semantics, formal verification) and parameters to be respected by these pillars.

Figure 3.2 summarizes the thinking process and steps to follow from the initial thinking about a novel capability's foundations to its implementation phase.

section A comparative analysis of existing service composition guidelines, description semantics, and formal specification and verification languages and tools.

After listing gaps in existing service frameworks and platforms, we perform a comparative analysis in this section to pick foundations, semantics, and formal specification and verification languages and tools for a service composition framework that fills the identified gaps and facilitates the composition of novel capabilities in the IoT and CPS space. Below is a comparison of foundations frameworks, modeling semantics, and formal verification languages and techniques: the goal is to select suitable composition components for the ICCF framework we propose.

5.1.1 Composition Guidelines

A framework provides foundations for service composition when it takes into consideration i) concerns related to the ability of the IoT/CPS to achieve an intended purpose in the face of changing external conditions such as the need to upgrade or otherwise reconfigure an IoT/CPS to meet new conditions, needs, or objectives (adaptability), ii) concerns related to our understanding of the behavior of IoT/CPS

due to the richness and heterogeneity of interactions among its components, such as the existence of legacy components and the variety of interfaces (complexity), iii) concerns related to the ability to combine IoT/CPS modular components (hardware, software, and data) to satisfy user requirements (constructivity), vi) concerns related to the ease and reliability with which an IoT/CPS component can be observed and understood (for purposes of leveraging the component's functionality) by an entity (human, machines), and v) concerns related to the ease and reliability with which an IoT/CPS component's functions can be ascertained (for purposes of leveraging that functionality) by an entity (discoverability). We can add to that the ability of the framework to address concerns related to users as composite capabilities measurability tends to be subjective (well-being in a smart building, safety in smart transportation, health improvement in smart medical facilities).

Examples of frameworks with composition foundations include IoT-A, a reference model and architecture (RMA) designed to allow the generation of different IoT architectures tailored to specific scenarios. Using IoT-A with Fiware in [244] enabled the creation of architectures with different functional groups, each serving a specific purpose and enabling interoperability; the composition of functions is intrinsic to Fiware using the service organization FG (functionality group), but stakeholders are concerns when composing IoT capabilities aren't explicitly addressed.

In [231], an OWL-based ontological framework for the opportunistic composition of IoT systems was introduced. The framework leverages holons, which are programming entities used to model distributed systems. Designing holons uses CoAMOS and A3ME ontologies. The resulting ontologies can then be converted to UML or domain-specific languages for further exploitation or composition in the IoT domain. While the discoverability of capabilities or composition complexity is addressed in this framework, the adaptability of the composition isn't addressed.

In [266], ISCO, (Internet of Smart City Objects), a distributed framework for service discovery and composition was introduced with three major enablers: a functional semantic description of city objects, representing physical devices or abstract services, a distributed service directory that embodies available city services for service lookup and discovery, and planning tools for selecting and chaining basic services to compose new complex services, this effort provides rich implementation aspects in the smart city context but the trustworthiness of composed capabilities isn't discussed.

The NIST CPS framework [112] defines criteria that contribute to CPS composition trustworthiness, taking into consideration functional, human, trustworthiness, timing, data, and composition concerns. The composition concern addresses the adaptability, complexity, constructivity, and discoverability of CPS capabilities; hence, the NIST CPS framework composition foundations are leveraged in ICCF to guide stakeholders' concerns for composing capabilities in the IoT or CPS Space.

Based on the discussion above and what we learned about IoT standards, frameworks, and references architectures in Chapter 4, Table 5.1 compares frameworks that provide foundations for IoT and CPS service composition in order to select a

foundations source that meets service composition needs in IoT and CPS.

TABLE 5.1: Comparing frameworks composition foundations in IoT and CPS

IoT/CPS service composition foundations	**Ref**	**Discussed strengths (✓) and Gaps (x)**
IoT-A	[244]	(✓) Supports interoperability mechanisms between heterogeneous IoT devices. (✓) Adopted by popular platforms such as Fiware. (x) Adaptability to user concerns and preferences not discussed, especially fine-tuning atomic capabilities weights.
OWL-based Ontological framework	[231]	(✓) Tailored for opportunistic, complex, and distributed IoT systems composition, with built-in discoverability mechanisms. (x) OWL-specific with UML interpretations, which might prevent the framework from being adapted to other modeling languages, reducing its interoperability.
ISCO	[266]	(✓) Provides enablers for semantic description, pools of capabilities for service discovery, and mechanisms for chaining and composing services. (x) Trustworthiness of composed capabilities isn't discussed.
NIST CPS Framework	[112]	(✓) Supports user, system, complexity, adaptability, constructivity, and discoverability constraints (✓) Defines different aspects related to an IoT or a CPS including functional, business, human, trustworthiness, timing, data, boundaries, composition, and lifecycle aspects. (✓) Supported by popular platforms such as OneM2M for smart manufacturing (CMfg) [313]. (x) Doesn't tell which semantics or formal verification tools to use, which can be beneficial in preventing scope reduction.

5.1.2 Modeling Semantics

Semantics for comprehensive, rapid, measurement-oriented, and accessible prototyping of capabilities composition suggest that the descriptions are lightweight and expressive enough to represent capabilities, interactions, compositions, and workflows necessary to compose value-added features in IoT and CPS.

The W3C Web Ontology Language (OWL) [212] is a semantic web language designed to represent complex and rich IoT capabilities, groups of things, and relations between things. However, it is more geared toward web services, and it is not a lightweight approach to composing services.

In [263], CyPhyML, a CPS capability description language, was discussed with formal specification capabilities supported. However, the language was more geared towards a formal description of CPS systems for model checking purposes

and not for IoT services description.

mPlane semantics [286] allow the representation of value-added capabilities using a set of operations designed to facilitate service composition. These compositions can be applied to measurement environments, IoT services, and CPS. mPlane semantics are simple, expressive, and lightweight compared to other description languages investigated; as a result, it satisfies the human aspect of the NIST CPS framework.

Based on the discussion above, Table 5.2 compares service composition semantics leveraged in describing IoT or CPS capabilities, their compositions, and interactions.

TABLE 5.2: Comparing IoT service composition semantics.

IoT/CPS service composition semantics	**Ref**	**strengths (✓) and weaknesses (x)**
W3C Web Ontology Language (OWL)	[212]	(✓) semantic web language designed to represent complex and rich IoT capabilities, groups of things, and relations between things. (x) geared toward web services, and it is not a lightweight approach to composing services.
CyPhyML	[263]	(✓) Formal specification capabilities supported. (x) As it's geared toward formal specification and verification of a CPS, it's not as accessible as other high-level IoT service descriptors.
mPlane	[286]	(✓) Allows the representation of value-added capabilities using a set of operations designed to facilitate service composition. (✓) Semantics facilitate measurability. (✓) Simple, expressive, and lightweight semantics compared to other description languages. (x) Doesn't recommend a particular formal specification language to transition from semantics to specification.

5.1.3 Formal specification languages and formal verification tools

Building correct compositions and verifying their properties should not be a daunting experience for engineers and developers. These stakeholders should be able to easily use semantics and service descriptors to formally specify and prototype composite services.

In [191], authors introduced linear logic LL based on pi-Calculus to describe and formally verify non-functional attributes such as the credibility/trustworthiness of the service composition.

Linear temporal logic (LTL) was introduced in [177]: Real-Time Maude formal verification tool that is based on LTL was used to formally verify properties of interest.

In [179], Directed acyclic graphs were used to formally model dynamic service discovery, invocation, and composition in opportunistic networks.

Petri Nets [115] were used as an algebra to formally model services and processes where the main goal was to check the compliance of compositions with the ever-changing regulations on IoT.

Temporal Logic of Actions (TLA) formal specification was used to formally specify and verify critical properties on services within the AWS echo-system [226].

The common aspect between these formal specification languages is how difficult it is to transition from description semantics to formal specification of composite services. TLA is an exception to this remark, as it has strong software support using PlusCAL: a high-level language that is comparable to pseudocode and which enables the fast translation of mPlane composition semantics to TLA formal specification.

TLA+ is the formal verification tool that uses notations similar to natural mathematic operations. CoQ [274], or Isabelle [12] use relatively daunting notations that are challenging for stakeholders, which might impact the developer's ability to write verifiable compositions and, as a result, might limit innovation.

Table 5.3 compares service composition semantics leveraged in describing IoT or CPS capabilities, their compositions, and interactions.

section ICCF: An IoT and CPS Capabilities Composition Framework

In this section, we select the ICCF pillars based on what was discussed in the previous section.

Pillars of ICCF include:

i) foundations for composition: including parameters to build composition platforms and composite services in a way that meets different stakeholders' concerns, including users, developers, and city planners.

ii) semantics for capabilities, interactions, and compositions, based on which composition algebra is crafterd.

iii) formal specification languages and formal verification tools used to translate the semantics of composition into specifications for performing model checking and trustworthiness assessment via assertions or deadlock analysis.

Based on the comparative study of IoT and CPS foundations and guidelines frameworks, service description semantics and modeling languages, and formal specification and verification techniques, we proposed a novel IoT and CPS Composition Framework that we call ICCF, a framework for the composition of IoT and CPS capabilities that is based on:

TABLE 5.3: Comparing IoT service composition formal specifications languages.

Formal specification and verification languages/tools	**Ref**	**strengths (✓) and weaknesses (x)**
Linear Logic and Pi-Calculus / No tool was used	[191]	(✓) based on pi-Calculus to describe and formally verify non-functional attributes such as the credibility/trustworthiness of the service composition. (x) Functional attributes verification not addressed.
Linear Temporal Logic / Real-Time-Maude	[177]	(✓) Used to reason about real-time systems and interactions in terms of time. (✓) Real-Time Maude tool supports LTL model checking commands. (x) Semantics might not be easily intelligible for new developers/researchers.
Directed acyclic graphs / VeriDAG	[179][1]	(✓) Compositions are specified using the DAG as a chain of services invoked successively. (x) DAG, a graphical representation, might need further algebraic interpretation to be formally verified.
Petri Nets / No tool was used	[115]	(✓) Specifies and verifies services and processes comply with the ever-changing regulations on IoT. (x) Doesn't provide a mechanism to translate semantics to formal specification.
Temporal Logic of Actions (TLA) / TLA+	[226]	(✓) formally specifies and verifies critical properties (load balancing) on services within the AWS echo-system. (x) Verification wasn't applied to functional properties in AWS IoT platforms such as GreenGrass.
Gallina/CoQ	[274][237]	(✓) Used to specify and prove distributed services mathematical theories based on building blocks including axioms and functions. (x) uses relatively daunting notations that are challenging for stakeholders, which might impact the developer's ability to write verifiable compositions and, as a result, might limit innovation.
Isar/Isabelle	[12][275]	(✓) Isar is known for its easy readability by humans and machines. (x) Doesn't provide mechanisms to deal with high complexity specifications that require cloud EC2 instances to run.

• Strong composition foundations provided by the NIST CPS framework.

• Simple, expressive, composition-friendly, and lightweight semantics for objects and operations inspired by the mPlane protocol.

• Easily interpretable modeling semantics (mPlane) to a pre-specification language (PLUSCAL).

• User-readable pre-specification language (PLUSCAL) which automatically translates to a Temporal Logic of Actions (TLA) formal specification thanks to

PlusCAL's translation capabilities of composition semantics to TLA specification.

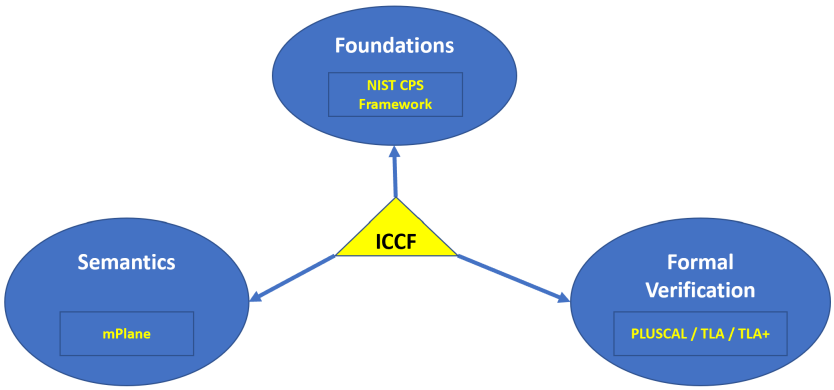

FIGURE 5.2: ICCF's selected pillars based on the comparative analysis: foundations (NIST CPS Framework), semantics (mPlane), and formal verification (PLUSCAL/ TLA / TLA+)

ICCF components are complementary:

• The NIST CPS Framework only provides comprehensive guidelines for building trustworthy CPS and IoT platforms but doesn't specify which description semantics to use for describing a composite service.

• mPlane semantics are very suitable for describing composite services objects, components, and interactions, but a language for interpreting these semantics into formal specifications is lacking in the mPlane platform.

• PLUSCAL, a language supported by the TLA and TLA+ suite for formal specification and verification, bridges the gap between mPlane semantics and formal specifications of algebraic composition operations.

section An ICCF description of a virtual composite service in a smart-city domain

Based on what we learned so far and from the SLR study's response to RQ1, IoT/CPS standards, frameworks, and reference architectures not only define data sharing, interoperability, and security specifications of exchanged messages over a composition platform but also incorporate mechanisms for composing or decomposing novel capabilities in different platforms.

5.1.4 The NIST CPS Framework guidelines for building IoT and CPS Platforms that meet stakeholders' concerns.

In this subsection, we highlight the NIST's CPS Framework general guidelines for building IoT or CPS composition frameworks, as well as specific guidelines for ensuring that compositions are built with many solutions to specific composition concerns in mind.

5.1.4.1 General guidelines for building IoT or CPS composition platforms.

Table 5.4 summarizes aspects and concerns to be aware of when composing capabilities based on the NIST CPS Framework guidelines. For more details about concerns related to each aspect, you can refer to [112][307][113].

TABLE 5.4: Aspects (including composition) and concerns related to the NIST CPS framework.

Functional	**Concerns about function performed by the IoT or CPS, including sensing, actuation, control, communications, etc.**
Business	Concerns about cost, enterprise, time to market, environment, regulation, etc.
Human	Concerns about human interaction with and as part of an IoT or CPS
Trustworthiness	Concerns about the trustworthiness of an IoT or a CPS Composite service, including safety, privacy, security, reliability, and resilience
Timing	Concerns about time and frequency in CPS and composite IoT services, including the generation and transport of time and frequency signals, timestamping, managing latency, timing composability, etc.
Data	Concerns about data interoperability, including fusion, metadata, type, identity, etc.
Boundaries	Concerns related to demarcations of topological, functional, organizational, or other forms of interactions
Composition	Concerns related to the ability to compute selected properties of a component assembly from the properties of its components. Compositionality requires components that are composable: they do not change their properties in an assembly. Timing composability is particularly difficult, hence the need for synchronization when it comes to timing constraints.
Lifestyle	Concerns about the lifecycle of CPS, including its components.

5.1.4.2 Composition-specific guidelines.

The NIST CPS Framework provides four composition-specific guidelines:

- Composition adaptability:

Addresses concerns related to the ability of the CPS to achieve an intended purpose in the face of changing external conditions, such as the need to upgrade or otherwise reconfigure a CPS to meet new conditions, needs, or objectives.

- Composition complexity:

Highlights concerns related to our understanding of the behavior of the CPS or IoT composite service due to the richness and heterogeneity of interactions among its components, such as the existence of old/legacy components and the variety of communication interfaces.

• Composition discoverability:

Identifies concerns related to the ease and reliability with which an IoT or a CPS component can be discovered and exploited (for purposes of leveraging the component's capability) by a composite service or by an agent (human, machine).

• Composition constructivity:

Showcases concerns related to the ability to compose IoT or CPS modular components including hardware, software, and data, to meet user requirements.

These guidelines, aspects, and concerns can be interpreted -In the context of the ICCF proposal- as seen in Figure 5.3 below:

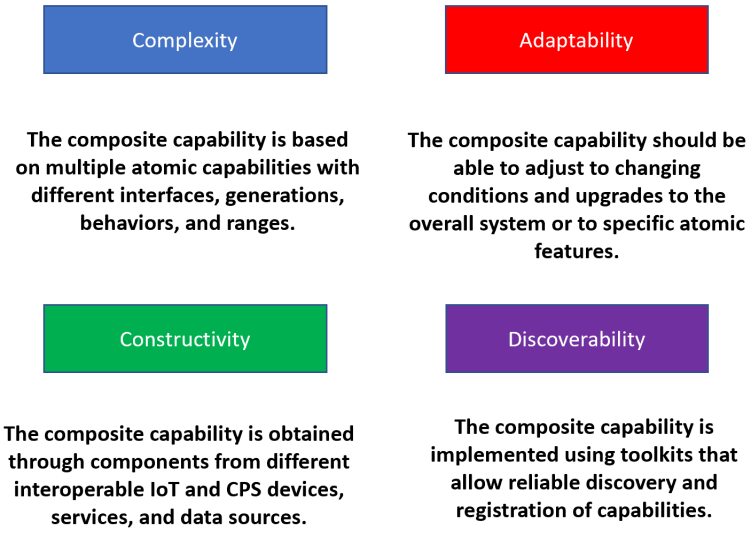

FIGURE 5.3: NIST CPS Framework Composition-specific aspects and Concerns to consider

5.1.5 mPlane semantics for describing capabilities algebra, objects, components, and operations

This subsection defines the mPlane semantics for describing capabilities algebra, interactions, and compositions.

The mPlane semantics choice is motivated by what we learned from the comparative study in this chapter as well as the knowledge we learned from answering SLR questions:

• RQ1/AQ1 : mPlane provides native support for composition functions.

• RQ6/AQ6: mPlane provides a composition engine (reasoner/supervisor).

• RQ7/AQ7: mPlane provides mechanisms for semi-automated compositions as the user initiates the composition process.

• RQ8/AQ8: mPlane offers commands that enable requesting, receiving, composing, and storing data by leveraging multiple network components (mPlane protocol, [287]).

• RQ9/AQ9: as a data model, mPlane semantics leverages JSON/XML schemas to describe capabilities.

• RQ10/AQ10: as a measurement platform, mPlane measures Traffic volume by composing bandwidth and latency atomic capabilities.

5.1.5.1 The space of capabilities and descriptors

Figure 5.4 shows the space of entities and capabilities. $\mathscr{E}$ represents the space of IoT entities, while $\mathscr{D}$ represents the space of capability descriptors. There is a space $\mathscr{R}$ in $\mathscr{D}$ that meets *ICCF* requirements. $\mathscr{R}$ elements can be composed and decomposed using the ICCF framework specification algebra, inspired by the mPlane semantics.

There is a *surjective* relationship between $\mathscr{D}$ and $\mathscr{E}$: one or more *CapabilityDescriptors* are provided by a single entity (*Ca*1 and *Ca*2 provided by *Ea*). In the implementation, using microservices, an exception to this rule are those microservices that provide a single and unique *CapabilityDescriptor* (*Cb*1 $\rightarrow$ *Eb* represents this case).

If $\mathscr{E}$ is composed of such microservices, then the relation between $\mathscr{R}$ and $\mathscr{E}$ is *injective*. Capabilities in $\mathscr{R}$ are either atomic (or non-decomposable) or composite.

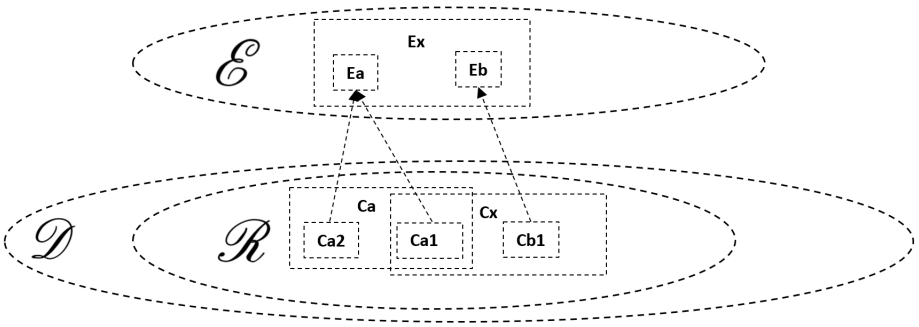

FIGURE 5.4: Space of Capabilities and Entities

5.1.5.2 Composition operators and descriptors

Let's define an operator ψ, which represents a k-ary composition operator. To illustrate composition, an assumption of k=2 is considered (which renders ψ a binary composition), Ca1, Ca2, and Ca are represented as JSON objects (with simple key-value pairs representing the *CapabilityDescriptor* parameters), the composition is an operator on values obtained after sending a specification to all atomic capabilities and receiving results.

Let's consider *Ca*1 and *Ca*2 from Figure 5.4 two atomic capabilities and *Ca* a composite capability obtained as follows:

```
The composition operator ψ has outcomes in ℛ:
ψ:                ℛ² → ℛ.
          (Ca1,Ca2) → Ca
                    → ψ (Ca1,Ca2)
```

This composition generates the *CapabilityDescriptor* of the composite capability *Ca* described below:

```
{"ID":  "Ca_ID",
  "Organization":  "Ca_O",
  "NAME":  "Ca_N",
  "TIMESTAMP":  "Ca_TS",
  "LOCATION":  "Ca_L",
  "REFRESH_RATE":  "Ca_RR",
  "UNIT":  "Ca_U",
  "VALUE":  "Ca_V",
  "SIGNATURE":  "Ca_S"}
```

Ca_ID: represents the ID of the composite capability. It is an increment of the last ID registered in the *CMr* registry. *Ca_N* and *Ca_O* are the Name and the Organization of the new composite capability, respectively; a new name and organization are attributed to the composite parameters when the atomic capabilities have different ones. *Ca_TS*: time of arrival of the composite capability. *Ca_L* represents the physical location (geographical in terms of latitude and longitude) or logical location (IP address).

In the case of a geographical address, the composite location is the location that comprises atomic capabilities' locations. For logical locations, If the sensors reside in the same IP Subnet, then the subnet that comprises their IP address becomes their composite location. *Ca_RR* represents the frequency at which a measurement is received. A composite value for this parameter should be the longest refresh rate:

$Ca\_RR \longleftarrow$ MAX{$Ca1\_RR$,$Ca2\_RR$}. *Ca_U* reflects the nature and unit of the composite capability. The simple example of power consumption as a composite capability of both current and voltage takes the "Watt" as the composite Unit of "Amperes" and "Volts". In other cases such as well-being in a smart building, atomic capabilities such as temperature, humidity, and air quality have different units, and the composition algorithm depicts the composite unit. *Ca_V*: represents the value of the composite capability.

IoT providers have the flexibility to define and introduce parameters customized to their composition needs. One such customization is the introduction of weights and multipliers. For example, $Ca\_V \leftarrow \alpha Ca1\_V + \beta Ca2\_V$ where α and β are two doubles that represent the weight of C_a1V and C_a2V, respectively, and (+) a composition operator. Composition rules and parameters are nested in the programmable extension of the atomic capabilities descriptors. This addresses the constructivity concern of the NIST CPS framework, as the ability to compose capabilities of different units and sources in a modular way would allow more innovation.

5.1.5.3 Capability Hierarchy and Level

Composite capabilities can be further composed into more complex capabilities. $C_{1,1}$ and $C_{1,2}$ in Figure 5.5 are an example of this case. The hierarchy level σ ranks the capabilities' complexity. Every capability can be expressed as follows: $C\sigma,y$, where σ is the hierarchy level and y is the id of the capability at that level. If $\sigma(C)$ =0, the capability is atomic. The composite capability descriptor enables tracking the ancestry of the capabilities and verification of their source without directly sending a request to the producing entities. This paradigm addresses the composition complexity concern of the NIST CPS framework.

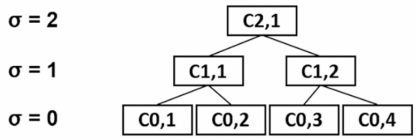

FIGURE 5.5: Composition Hierarchy

5.1.5.4 Specifications, results, and interactions

5.1.5.4.1 Specifications and results A *Specification* (*sC*) for a capability with a descriptor (*C*) is a request sent from an entity to a resource to get information. The *Specification* contains information about the capability that helps intermediate entities (including proxies and capability managers) to locate the requested capability. A *Result* is a *Specification* for which all the parameters are known. The *Result* (*rC*) can be represented as the solution for a system of equations with all the parameters resolved. The space of solutions can contain a unique element, multiple elements, or no element. Below is an example of a *Specification* represented as a system of equations where the only unknown parameter is C_V: the value of the capability. The other parameters are known as depicted in equation 1. The *Result* is a unique solution to the *Specification* as depicted in equation 2.

5.1.5.4.2 Discovery interaction The capability manager discovers entities that verify the following rule: $CapabilityDescriptor \in \mathscr{R}$. The discoverability function is defined as Disc(*CM*, E); it takes CM, a capability manager, and E, an entity, as input and returns a binary based on whether or not a capability is discovered. This addresses discoverability, one of the NIST CPS framework composition concerns.

$$sC \leftrightarrow \begin{cases} C_{ID} = ID_{value} \\ C_O = O_{value} \\ C_N = N_{value} \\ C_{TS} = TS_{value} \\ C_L = L_{value} \\ C_{RR} = RR_{value} \\ C_U = U_{value} \\ C_V = ? \\ C_S = S_{value} \end{cases} \quad (1) \qquad rC \leftrightarrow \begin{cases} C_{ID} = ID_{value} \\ C_O = O_{value} \\ C_N = N_{value} \\ C_{TS} = TS_{value} \\ C_L = L_{value} \\ C_{RR} = RR_{value} \\ C_U = U_{value} \\ C_V = V_{value} \\ C_S = S_{value} \end{cases} \quad (2)$$

FIGURE 5.6: Specification and Result model.

5.1.5.4.3 Registration interaction $C \in \mathscr{R} \implies C$ can be registered in CMr. The above implies that all k-ary compositions ψ can be applied to C. The Reg(CM,C) function is a binary function that takes CM (a capability manager) and C (the entity's $CapabilityDescriptor$) as inputs and returns True or false depending on whether or not the descriptor is stored in the CMr and the composition algorithms nested in its programmable extension are stored in the CMt.

5.1.5.4.4 sendSpec(Src,Dst,Specification) interaction It is a request sC sent to a resource, a proxy, or a capability manager. If C represents a composite capability $CapabilityDescriptor$, the $Specification$ sC will be decomposed to its atomic $Specifications$ $(sC1,sC2,...,sCk)$ first by applying the decomposition operator ψ^{-1} as follows:

$$\begin{aligned} \psi^{-1}\textbf{:} \quad & \mathscr{R} \rightarrow \mathscr{R}^k. \\ & (sC) \rightarrow (sC1, sC2, \ldots, sCk) \\ & \rightarrow \psi^{-1}(sC) \end{aligned}$$

5.1.5.4.5 sendResult(Src,Dst,Result) interaction Entities directly provide a $Result$ for the $Specification$ (check 5.6) if it doesn't require a further composition or if it is available in the cache CMc. This explains how the adaptability concern of the CPS framework is addressed. Composite $Results$ require composition.

$$\begin{aligned} \psi\textbf{:} \quad & \mathscr{R}^k \rightarrow \mathscr{R}. \\ (rC1, rC2, \ldots, rCk) & \rightarrow rC \\ & \rightarrow \psi(rC1, rC2, \ldots, rCk) \end{aligned}$$

5.1.5.5 ICCF semantics or algebra example

$ICCF$ algebra helps in expressing abstract interactions (discovery, registration, composition, decomposition, specification, results) and enables formal verification, symbolic execution, and making sure the outputs of a system fall within trustworthy values. The pseudocode in Algorithm 1 summarizes all these operations in a use case explained in the section. The space of capabilities is described above via an example depicted in Figure 5.4.

Let's consider CM, a capability manager, Ex an entity that provides a composite $CapabilityDescriptor$ Cx from atomic capabilities $Ca1$ and $Cb1$. These atomic capabilities are provided by entities Ea and Eb. Ex requests a composite capability Cx from the nearest CM. CM checks its CMc as to whether a copy of the $Result$ rCx is available. If this is the case, CM returns the data to Ex. Otherwise, CM sends requests to entities Ea and Eb based on information in the CMr. These latter respond by sending their $Results$ back to CM. The capability manager performs the composition of the $Result$ rCx based on algorithms in the CMt and sends it back to Ex.

So far, ICCF composition interactions and operations have been described using semantics inspired by the intuitive mPlane platform and following the capabilities composition guidelines of the NIST CPS framework.

Algorithm 1 ICCF Protocol

if $Ca1 \in \mathscr{R}$ and $Cb1 \in \mathscr{R}$ **then**
  Disc(CM, (Ea,Eb))$\leftarrow$ true and
  Reg(CM,($Ca1$,$Cb1$))$\leftarrow$ true
  sendSpec(Ex, CM, sCx)
if $rCx \in CMc$ **then**
  sendResult(CM, Ex, rCx)
else
  ψ^{-1}(sCx)$\rightarrow$($sCa1$, $sCb1$)
  sendSpec(CM, Ea, sCa1) and
  sendSpec(CM, Eb, $sCb1$)
  sendResult(Ea, CM, $rCa1$) and
  sendResult(Eb, CM, $rCb1$)
  ψ($rCa1$,$rCb1$)$\rightarrow$(rCx)
  sendResult(CM, Ex, rCx)

5.1.6 Formal Specification and Verification through PLUSCAL and TLA+

5.1.6.1 Usecase description

In this paragraph, a virtual composite service is formally studied. This virtual service involves composing multiple atomic capabilities to get a value-added feature.

Potential real-world use cases include composite well-being as a composite capability in a smart building, safety as a composite capability in the smart transportation domain, and health improvement as a composite capability in the smart health domain.

In this example, we leverage an abstract composite capability that depends on multiple sensor inputs from multiple entities.

These entities can be temperature, humidity, and pollution sensors in the case of smart buildings.

For smart transportation, atomic capabilities can originate from vehicles, road users, or transportation infrastructure.

For the medical domain, atomic capabilities can originate from the patient or the ICU's different smart devices.

The value rCn of a virtual composite capability Cn can be represented as the composition ψ of n atomic capabilities as follows:

$$\psi: \quad (rC0\,, rC1\,, \ldots\,, rCn-2\,, rCn-1\,, rCn) \;\rightarrow\; rCn$$

To simplify and illustrate the formal specification of such capability, we suppose n=3:

ψ:

$$(rC0, rC1, rC2) \rightarrow rC3$$

ψ is the composition operator which represents, in this case, arithmetic and logical operations on the results of atomic capabilities $C0$, $C1$, $C2$ that generate the result $rC3$ of the composite capability $C3$.

The goal is to prototype a composition with an assured outcome, which means the composite feature's values must fall in a trustworthy range.

Assuming discovery, registration, and other mPlane protocol interactions are already performed, we focus on the composition operations on the capabilities results.

The composite feature's value (or state) is comprised between a minimum value and a maximum value, with desired and trustworthy values that meet stakeholders' concerns falling between these two extremes.

The impact of each atomic capability on the state of the composite capability can be represented using multiple techniques, including the weighting mechanism as follows:

ψ:

$$(rC0, rC1, rC2) \rightarrow (W0 * rC0) + (W1 * rC1) + (W2 * rC2) \rightarrow rC3$$

As an example, let's consider the set of values and ranges below for the atomic capabilities $C0$, $C1$, and $C2$:

$$1 \leq rC0 \leq 5$$

$$2 \leq rC1 \leq 4$$

$$1 \leq rC2 \leq 3$$

And let's consider that according to the engineer's tuning preferences, the weight of each capability is distributed as follows:

$$WC0 : 2\ \%$$

$$WC1 : 3\ \%$$

$$WC2 : 1\ \%$$

In this case, the range of $C3$ values is comprised between [min((W0*$rC0$)+(W1*$rC1$)+(W2*$rC2$)) and max((W0*$rC0$)+(W1*$rC1$)+(W2*$rC2$))], ie [min=2+6+1,max=10+12+3]=[9,25].

$$9 \leq rC3 \leq 25$$

For any composite capability, there's a threshold that defines acceptable values for a certain stakeholder; if values of $rC3$ that are < 15 are the desired (or trustworthy) states for the composite capability, any other outcome should raise flags when formally verifying the composite capability. This is achieved through invariants analysis.

This threshold analysis is part of the human concern analysis that we take into consideration as we respect the NIST CPS framework implementation guidelines. The timestamp is synced across all capabilities values, which are discrete.

Figure 5.7 shows the composite capability model in PlusCal and its translation to TLA+.

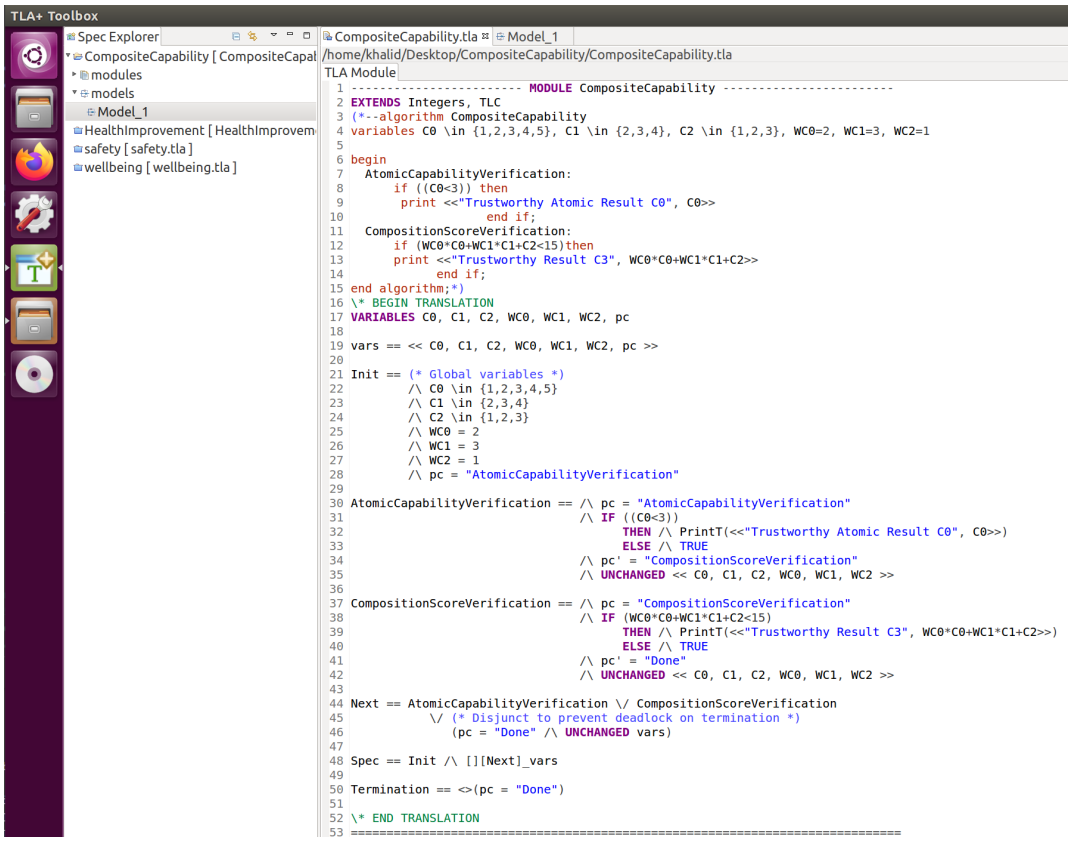

FIGURE 5.7: Virtual Composite Service model in PlusCal and its translation to TLA

Figure 5.8 shows the range of values generated by each atomic capability and the trustworthy boundaries for the composite capability. In some applications, trustworthy boundaries can be set for atomic capabilities as well.

mPlane Semantics	PLUSCAL/TLA representation	Description	Unit	Complexity	Possible Values/States	Trustworthy boundaries
rC0	C0	Atomic capability 1	Unit 1	Atomic	[1,5]	[1,2]
rC1	C1	Atomic capability 2	Unit 2	Atomic	[2,4]	-
rC2	C2	Atomic capability 3	Unit 3	Atomic	[1,3]	-
rC3	C3	Composite capability 1	Unit 4	composite	[9,25]	[9,15]

FIGURE 5.8: Atomic and Composite Capabilities and their range of possible and trustworthy values.

Figure 5.9 shows results after running symbolic execution. The model was run on an Ubuntu 16.04 with eight i7 CPU cores and 10 GB of RAM. TLA+ allows connection to remote AWS EC2 instances to analyze super complex and highly demanding specifications.

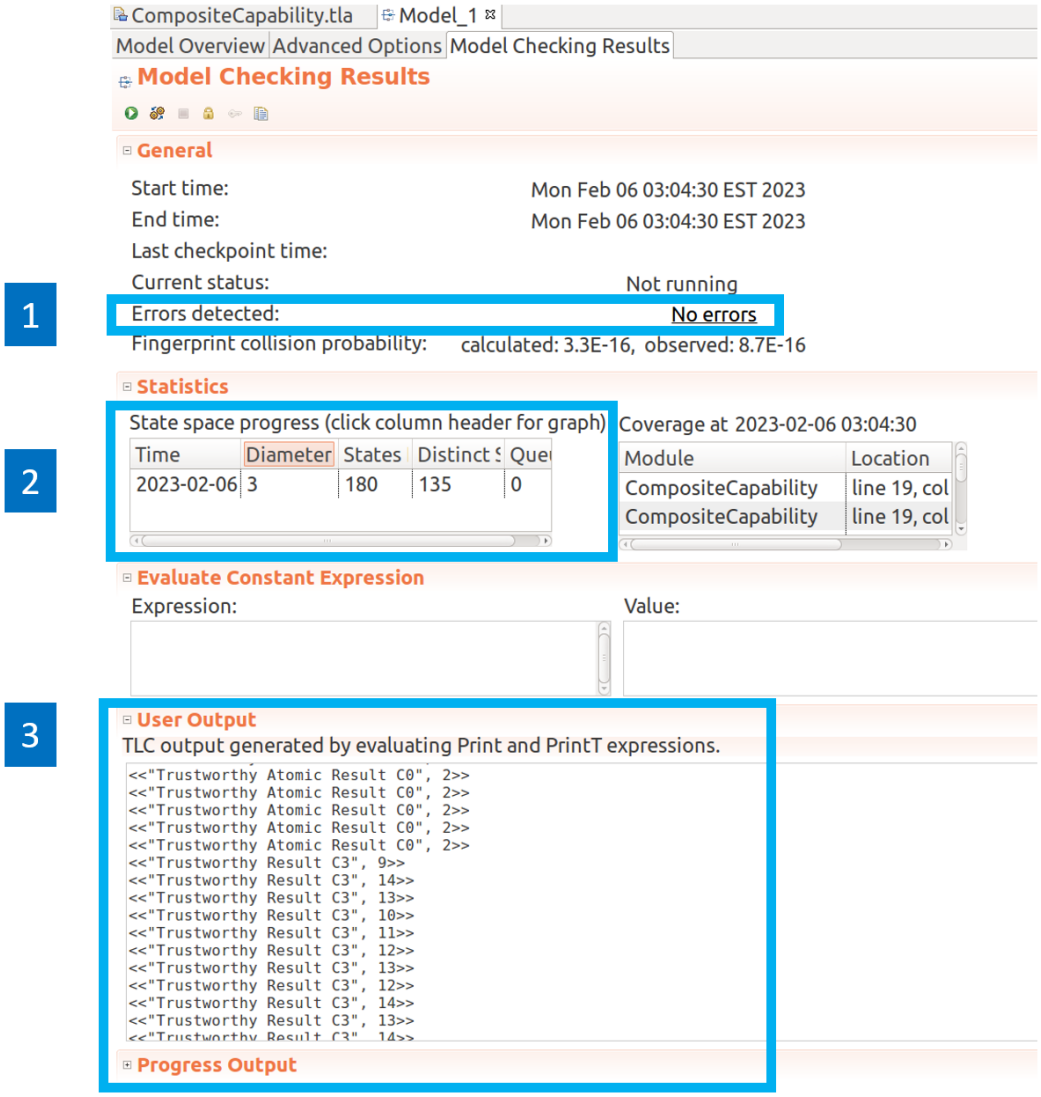

FIGURE 5.9: Symbolic Execution Results: Error status (1), number of states (2), user output (3)

To test the virtual composite capability against errors or unwanted states, we use

deadlock invariants. Figure 5.10 shows an instance of checking a deadlock state that yields a non-trustworthy / desired outcome.

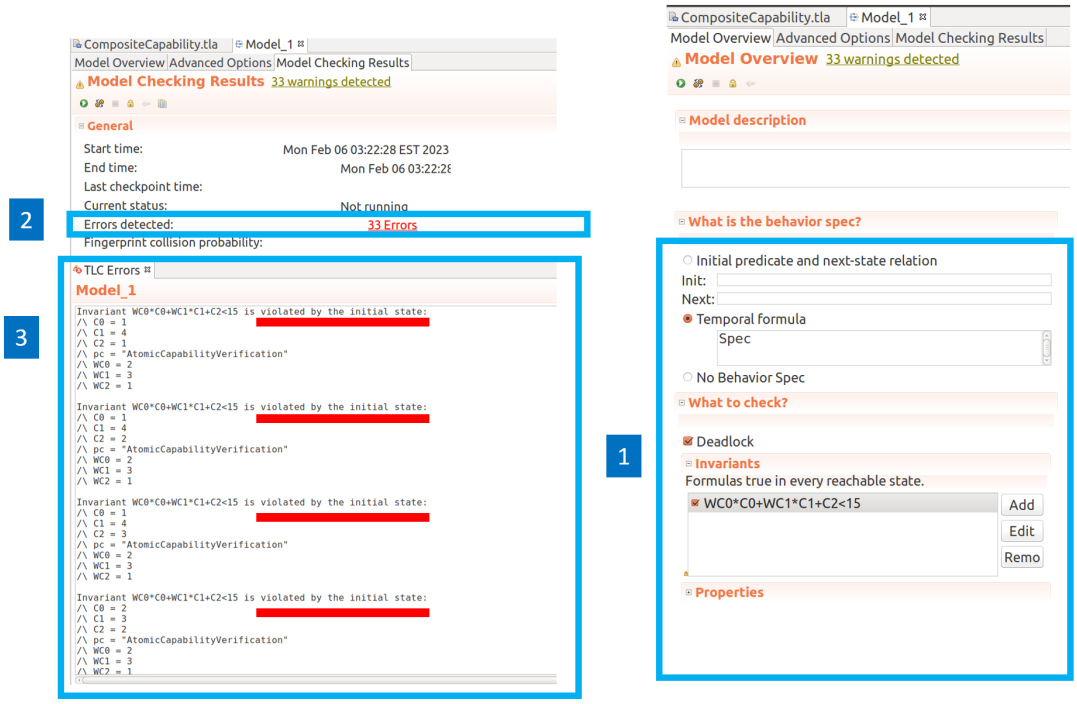

FIGURE 5.10: Deadlock and Trustworthiness Verification

5.1.6.2 Qualitative and Quantitative Analysis

The symbolic execution of the model results in running combinations of all atomic capabilities values to determine the composite capability state space. In Figure 5.8, it took 1 second to perform symbolic execution as the model is simple and was provided for demonstrating the usefulness of the TLA+ ecosystem. APALACHE Model Checker can replace TLC (default TLA+ model checker) to improve execution time [168] when the system's model is too complex. From Figure 5.9, the number of states generated across all combinations is 180, with 135 distinct states. The queue didn't suffer from congestion as the system presented few variables with fewer states.

5.1.6.3 Takeaway from the use case

Through this use case, we demonstrated how to prototype a composition based on the framework and semantics proposed, run symbolic execution, analyze trustworthy results, and reveal errors using a deadlock invariant. Examples that could benefit from this framework include composing a well-being capability in the smart building domain, analyzing oxygen toxicity in autonomous ventilators (smart health applications), or studying braking time as a composite feature in an autonomous vehicle to evaluate the braking amount required to prevent collisions (smart transportation applications). Minimizing execution time, queue congestion, and the effect of state-space explosion can be done by optimizing the model, space reduction, or leverage of better hardware (local/cloud) which TLA+ allows

by calling EC2 instances that can run TLC as we learned by answering research questions RQ4/AQ4 in Chapter 4.

section IoTCaP: an implementation platform for ICCF-described composite services.

5.1.7 Implementation efforts

The ICCF composition framework is agnostic from the implementation perspective, and its principles can be implemented in multiple environments. Vert.X, a reactive and event-driven programming toolkit, is used to implement the well-being composite features in a smart building application based on the ICCF foundations. The well-being verticle receives data from temperature and humidity sensors (provided by sensor DHT22 AM2302) and air-quality sensors (provided by sensor SDS011 PM2.5). Code for the project is available in the GitHub repository [159]. An Automated Driving System testbed [121] on the NIST's UCEF co-simulation environment is also being built [52] to provide atomic capabilities for a safety assessment verticle: The goal is to simulate autonomy functions as a composite feature which will enable trustworthiness assessment of safety-critical maneuvers such as emergency braking.

5.1.8 IoTCaP: An ICCF-based Platform For Implementing The Composite Capability Entities And Interactions using mPlane Semantics.

We worked on an implementation for the ICCF framework that we called IoTCaP, which stands for the IoT Capabilities Platform. IoTCaP leverages a front-end interface built using the VUE.JS toolkit and a backend that leverages Vert.X verticles to connect to the front-end interface via the Axios Library and to the devices layer through endpoints compatible sensors.

The sequence diagram in Fig. 5.11 provides a high-level outlook on IoTCaP and summarizes its composition-related interactions between its entities, it is inspired by the mPlane protocol reference architecture that:

• Defines the role of the client in requesting capabilities (replaced in the IoTCaP by a stakeholder).

• Defines the role of the probes and repositories in making capabilities results available for requesting clients and services.

• Defines the role of the supervisor and reasonser in composing atomic capabilities into value-added services.

In IoTCaP, the "Stakeholder" can be any actor with interest in the composition platform; for example, it can be a user requesting a capability or an engineer tuning its parameters through weights. Typically a stakeholder authenticates securely to his/her profile in IoTCaP and then inputs parameters that describe the desired capability. Once these parameters are submitted to the front-end interface and communicated to the backend composite service "via specifications", the IoTCaP composite capability backend computes one or multiple composite metrics and

returns the "result" to the stakeholder, with the possibility to display atomic capabilities contributing to the composite service. Figure 5.11 doesn't show some mPlane operations such as capabilities registration or discovery, and it also omits user authentication as the main goal here is to show composition-specific operations. These operations will be added to the real-world examples that we will discuss in Chapter 6.

section*Summary of the ICCF framework proposal
In this chapter, we introduced the ICCF framework as well as its formal and knowledge criteria derived from a composition-enabling framework (NIST CPS Framework), straightforward and expressive semantics (mPlane), and strong formal verification language and techniques (TLA+, TLA, TLC, PLUSCAL). A comparison of existing environments, frameworks, semantics, and formal specification and verification techniques enabled the selection of formal components of the ICCF composition framework that enables specification, prototyping, and assessment of IoT and CPS capabilities. The goal is to provide stakeholders with the tools to innovate in the IoT and CPS space while addressing their corresponding concerns. NIST CPS framework composition guidelines and powerful semantics inspired by the mPlane protocol, as well as formal specification and verification techniques provided by the TLA/PlusCal package, enable such a framework. Composition requirements, services, and interactions were described, and based on that, a composite capability example was studied. Results of model checking were generated, and an analysis of the state space was performed to understand non-trustworthy results through a deadlock invariant.

Figure 5.12 fills up the components chosen for the ICCF framework and summarizes the steps to follow to prototype and implement a novel composite service.

In Chapter 6, we discuss in depth a real-world composite service in the smart city domain:

- Assessing well-being as a composite capability in the smart building domain.

And we discuss components for assessing two more smart-city applications within the ICCF framework:

- Assessing Autonomous Vehicle Manoeuvers Safety as a composite metric in the smart transportation domain.

- Assessing Health improvement as a composite capability in the smart health domain.

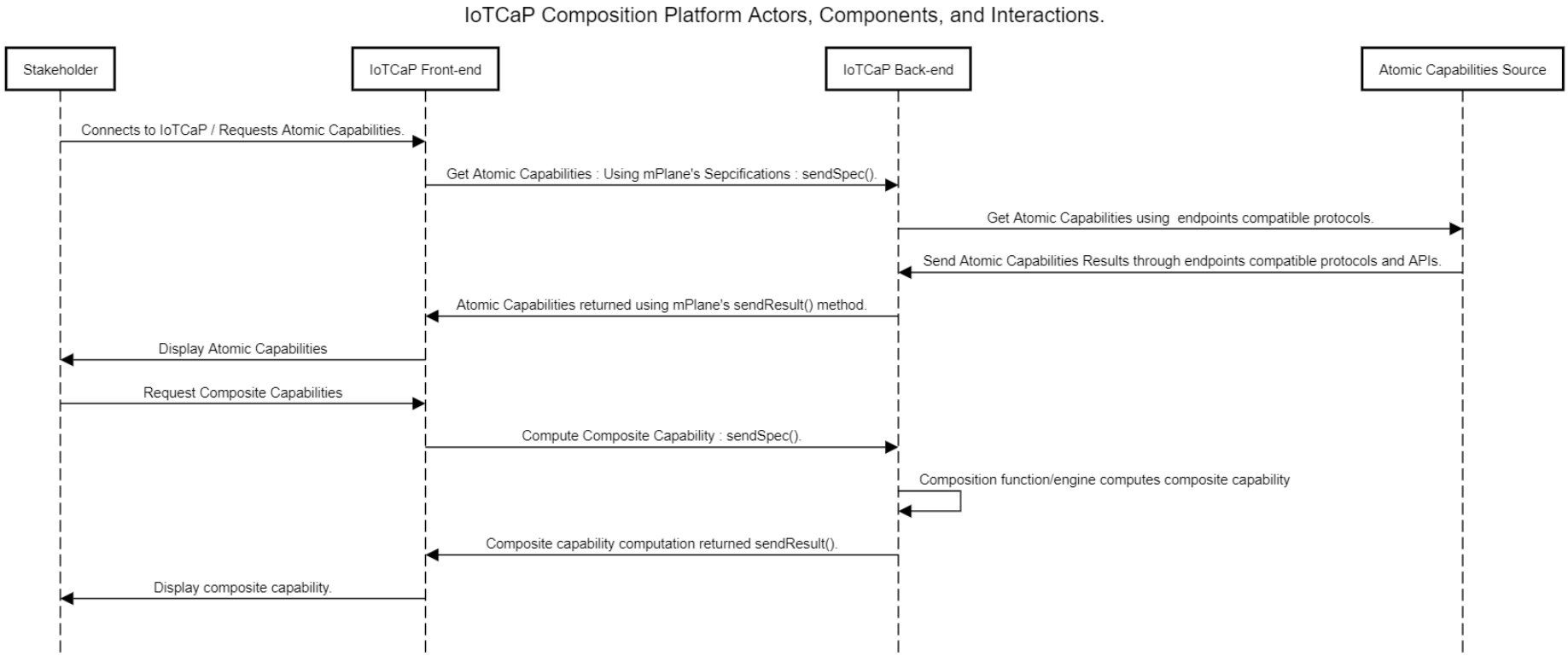

FIGURE 5.11: Sequence diagram for IoTCaP interactions and entities: crafting a composite capability from atomic functions is highlighted using mPlane composition interactions.

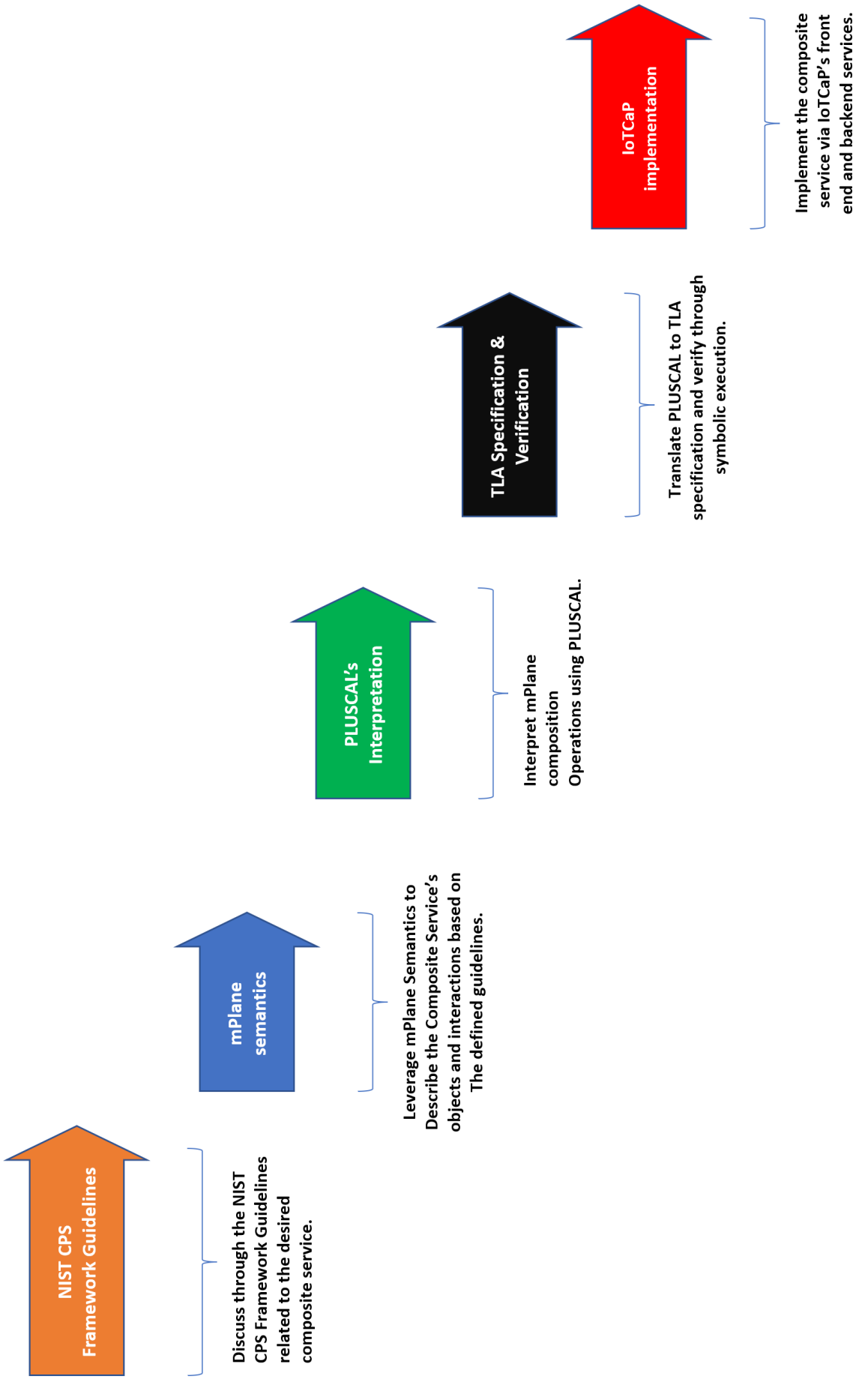

FIGURE 5.12: Filled Requirements for building a trustworthy composition framework and platform.

Chapter 6

ICCF Applications in smart cities

After introducing and explaining the ICCF framework in Chapter 5, we devote this chapter to discussing applications of the proposed framework in different smart city domains. In section 6.1, we start by discussing well-being as a composite capability in smart buildings, section 6.2 discusses safety as a composite capability in the smart transportation domain, and section 6.3 discusses ongoing work on a health improvement composite capability in the smart health domain.

6.1 Well-being as a Composite Capability in the Smart Building Domain: A Formal and Technical Study.

Smart building value-added capabilities are gaining significant attention from various stakeholders, including the general public, researchers, and the industry. One such capability is well-being, a composition of multiple atomic capabilities that characterize a smart building. Atomic functions that compose a well-being capability include temperature, noise level, pollution level, and humidity, to name a few. Multiple efforts have addressed this specific capability and its composition requirements and techniques from standardization, technical, and quality of service aspects. One such effort is the IoT and CPS Composition Framework (ICCF), a novel framework for rapid modeling, specifying, verifying, and prototyping IoT and CPS capabilities, which we discussed in Chapter 5. ICCF relies on the NIST CPS Framework guidelines to address different stakeholders' concerns; it also leverages composition semantics inspired by the mPlane platform to describe entities and interactions intuitively. In addition, it uses the Temporal Logic of Actions + (TLA+) formal verification techniques to verify the correctness of core functions.

In this chapter, we leverage the ICCF framework to provide the following contributions: i) a description of a stakeholder-defined well-being composition capability based on the ICCF framework foundations, ii) an in-depth characterization of the well-being capability, iii) considerations regarding the formal aspects of the well-being capability, including verifying its correctness, deadlock, and state-space, iv) implementation of the composite capability using a lightweight microservices environment and real sensors, v) discussion of results based on the different domains of interest including residential buildings and factories. Finally, a summary of this chapter is provided, and challenges to capabilities composition, as well as future plans for improvement and expansion, are highlighted.

6.1.1 Introduction

Innovating value-added capabilities in Internet of Things (IoT) or Cyber-Physical Systems (CPS) - concepts that are converging over time [110] - through composition is gaining more and more interest as a wide range of sensors and appliances are generating data that can be exploited to improve various features within the different types of environments including smart buildings. Smart buildings are emerging as complex cyber-physical systems with humans in the loop [148], that should provide a safe and comfortable space for inhabitants. Well-being is a function that represents the quality of living perceived by different entities (residents, maintenance, owners, etc.) in a particular building with specific properties; this means that well-being requirements in a residential building can differ from those of a factory or a hospital. In other words, as buildings vary in terms of shape, characteristics, and purposes, well-being definition varies accordingly to accommodate those particularities. For example, well-being in a residential building has the primary purpose of addressing the occupants' comfort. Consequently, it requires reasonable levels of temperature, humidity, air quality, noise level [257, 306, 131], and WiFi signal strength as wireless internet access in smart buildings is common practice as opposed to wired communications and represents a crucial feature that appliances for well-being or entertainment rely on [77].

On the other hand, for hospitals, well-being is based on ensuring sanitary conditions, which may require more adjustable levels of temperature, humidity, and air quality compared to those of residential buildings [186]. The same can be said about clean rooms in which semiconductors are built where certain products require temperature, humidity, and air quality (dust, particulate matter) levels that differ from those of residential buildings or hospitals [188].

Factors that influence well-being can be classified into two categories: i) static factors that a stakeholder has little to no ability to change or improve; these include the shape of the building, and its location, to mention a few, and ii) dynamic factors, which the stakeholder can alter and improve using multiple techniques or systems and those factors include temperature, noise, humidity, and air quality. This work focuses on dynamic factors that impact well-being in a given type of building.

Modeling, prototyping, and rapidly implementing composite capabilities in IoT and CPS are essential for innovation in smart buildings. Therefore, following streamlined and easy-to-follow guidelines is crucial to ensure adherence to standards [211] and best practices and achieve an acceptable level of correctness and reliability.

In this chapter, the IoT and CPS Composition Framework (ICCF) [120] -a framework for composing novel IoT and CPS capabilities- is leveraged for guiding, modeling, prototyping, composing, and verifying an IoT composite capability called well-being. This capability addresses concerns in different types of buildings. We illustrate an end-to-end composition effort; to the best of our knowledge, the well-being capability has never been formally specified as a full-fledged capability; this effort provides a detailed prototype for this novel feature. The ICCF framework guidelines followed in this chapter derive from the NIST CPS

Framework -a framework that provides a comprehensive analysis of a CPS and captures its generic functionalities, activities, and artifacts needed to support its conceptualization, realization, and assurance-[114, 112, 307], make use of intuitive and comprehensive mPlane semantics [286] to describe entities and interactions and leverages robust, straightforward, and easy to use Temporal Logical of Actions (TLA) formal verification techniques and tools [175, 168]. It was demonstrated in [120] that these foundations, guidelines, and tools form a powerful framework that would help researchers, developers, and engineers better frame and organize their composition and innovation efforts and adhere to procedures and semantics to improve the capabilities quality, correctness, and reliability.

The contributions presented in this section are organized as follows: Subsection 6.1.2 provides a comprehensive related work about well-being in IoT. Subsection 6.1.3 explains the introduced definition of well-being and its relation to different stakeholders' concerns and describes entities and interactions contributing to its composition and computation in an ICCF implementation called IoTCaP. In Subsection 6.1.4, mPlane semantics for the well-being function are formally specified using the PlusCal language, and model checking and deadlock analysis of the formal specification are done using TLA+. In Subsection 6.1.5, an implementation of IoTCaP is provided, as well as a discussion of the results obtained. Finally, a summary of this chapter discussing encountered challenges and future work is provided in Subsection 6.1.6.

6.1.2 Related Work

This subsection provides a comprehensive review of capabilities composition in the smart building domain. In addition, it highlights efforts addressing well-being or comfort as a capability in the IoT or CPS space.

6.1.2.1 Smart Building Applications And Composition

Different research efforts addressed smart building applications from a service composition perspective. In [36], *Brick*, a uniform metadata schema for representing smart buildings components -including sensors and subsystems, and describing the different interactions that occur between these components- is proposed. The goal of such schema is to provide APIs that enable the creation of portable energy efficiency applications.

In [148], a platform-based methodology for smart building design (PBD) was proposed, which promotes the reuse of hardware and software on shared infrastructures, enables rapid prototyping of applications, and involves extensive exploration of the design space to optimize design performance.

In [43], IoT was leveraged to build an Energy Management System (EMS) that takes into account the behavior of individual customers who occupy smart buildings. This idea is extended in this work to allow different stakeholders to customize well-being to address their needs.

In [122], full-IP IoT with real-time Web protocols is discussed as a major enabler for efficient and meaningful aggregation of services known as composition and how that would impact domains such as smart buildings.

In [203], energy cost, thermal comfort, and social IoT (SIoT) concepts are composed to provide a Smart Heating, Ventilation, and Air Conditioning (HVAC) system for smart buildings.

In [56], Fault Maintenance Trees (FMTs) and probabilistic model checking are used to evaluate various dependability metrics and maintenance strategies of Heating, Ventilation, and Air-Conditioning of a smart building.

In [294], an adaptive service composition framework that supports dynamic reasoning on numerous smart city IoT services was proposed, and a contExt Aware Web Service Description Language (wEASEL) abstraction was leveraged to represent services, compositions, and interactions. For evaluating the composition framework, an OWLS-TC4 testbed was proposed to evaluate the accuracy and novelty of simple and composite services.

In [209], a model was proposed for simulating the core engineering subsystems of a smart building. The model is based on Matlab Simulink, the Simscape physical modeling library, and the Stateflow library. The goal of the model is to simulate coordination between subsystems and assess power consumption for optimization purposes.

6.1.2.2 Previous Efforts On assessing Well-Being

In this paragraph, previous efforts that tackled comfort or well-being as an IoT capability are discussed, in particular, the IoT and CPS Composition Framework (ICCF) [120].

Research on well-being usually addresses a single atomic capability that contributes to well-being instead of studying well-being as a composite function which this work addresses.

In [223], the role of smart urban technologies was demonstrated, especially in ensuring sustainable cities and well-being for the citizens; however, the study focused on sustainability and energy efficiency aspects and didn't address well-being concerns.

In [147], a summary of IoT technologies enabling smart buildings was provided, and a referenced study [311] found that people spend 80 percent of their lifetime inside buildings; the study showed the extent to which comfort is a key component that IoT researchers and engineers must address to improve the well-being of smart building residents.

In [41], highlighting the differences between smart buildings and autonomous buildings was performed to clear the confusion associated with these two concepts; they also introduced a measurement to quantitatively assess the building's

"intelligence". One key criterion contributing to the building intelligence metric is "inhabitants' comfort." This component is defined as the ability of "Smart" homes to learn from inhabitants' behavior and maximize their comfort.

In [142], the book on Green and Smart buildings discusses the shift buildings are witnessing as they become more and more people-centric rather than engineering, construction, or technology-centric. This is exemplified in how building owners, developers, and facility managers focus on increasing occupant well-being and comfort by building smarter spaces that engage occupants, understand how they use buildings in new ways, and get them involved in implementing sustainable practices.

In [178], a focused study on addressing thermal comfort was done within an occupied building and how that requires energy and, thus, an optimized solution balancing energy use with indoor environmental quality (adequate thermal comfort, lighting, etc.). In [119], a quantitative composite air quality metric was introduced as a contributor to well-being.

In [102], balancing comfort requirements with environmental and energy constraints was discussed by leveraging a Context-Aware Framework for Collaborative Learning Applications (CAFCLA) to combine various technologies that simplify the creation of context-awareness and social computing systems that influence user behavior to favor efficient energy resources without compromising comfort in the workplace. In [236], comfort and well-being were addressed from a privacy perspective. By setting and interacting with smart devices, inhabitants risk giving away details about their preferences that compromise elements of their privacy. A framework that forces IoT Assistants -when capturing and managing the privacy preferences of their users- was proposed to communicate privacy-sensitive information to privacy-aware systems.

In [39], a discussion around extracting information that enhances inhabitants' comfort from smart buildings was done. Smart buildings generate huge amounts of data, and the integration of Big Data Analytics (IBDA) and IoT to address the large volume and velocity of real-time data in the smart building domain is proposed to enhance well-being.

In [288], the health and well-being of smart city residents are discussed as a social aim of IoT rather than the usual sustainability and ecology aim. A case study of smart health and well-being in Kashiwanoha Smart City in Japan was discussed, and the impact on resident lifestyles was assessed. Findings suggest that smart cities have great potential to be designed and executed to tackle social problems and realize more sustainable, equitable, and livable cities. In [206], comfort, among other qualities, is discussed as a crucial property in educational buildings; comfort contains thermal, acoustic, visual, and air quality components; the paper presents a case study in Nuevo León, Mexico, where a comparative study was conducted to assess the teaching-learning process in different environments with different health, safety, and comfort criteria.

In [204], a new metric was proposed: the Smart Readiness Indicator (SRI). SRI shows whether or not a building respects different criteria defined by the European Union Standards; the criteria contributing to the SRI include energy, flexibility for the grid, self-generation, comfort, convenience, well-being and health, maintenance, and fault prediction, information to occupants. The SRI methodology differentiates between three weights for all functional levels and impact criteria: equal weights, residential, and non-residential buildings.

In [91], smart buildings are discussed as well as several technologies that support their functionalities, including the automated building management system (BMS), which aims to achieve the well-being of occupants, promoting a comfortable environment while ensuring efficient use of building resources. Context-based reasoning as a modeling paradigm for smart buildings is proposed as a supportive mechanism to realize smart building applications.

6.1.2.3 Main takeaway

As a summary of the related work above, well-being or comfort was addressed as a qualitative feature, with some efforts addressing single aspects of well-being such as air quality. However, and to the best of our knowledge, none of the research efforts investigated tackled well-being as a dynamic quantitative composite measurement that can be formally specified and verified, which this effort aims to achieve.

6.1.3 An ICCF definition of Well-Being as a composite capability

This subsection proposes an ICCF-based definition of well-being as a quantitative, dynamic, and composite capability. Next, a discussion sheds light on this capability from different stakeholders' perspectives and how these perspectives can be quantified and incorporated into the well-being metric computation. After that, semantics representing the requirements mentioned above using mPlane notations and algebra are provided. Finally, an illustration is done for the required operations using a sequence diagram and pseudo-code in the context of a platform called the IoT Capabilities Platform (IoTCaP).

6.1.3.1 A Proposed Definition For Well-Being Based On Stakeholders Concerns

A set of metrics that contribute to well-being in smart buildings were identified; some of these metrics are found in the WELL Building Standard version 2™ [297]. These metrics include temperature, humidity, air quality, noise level, and WiFi signal strength. These capabilities change over time, and sometimes they fall under comfort levels that compromise well-being as a whole. In this work, *weights* and *scores* characterize, quantify, and contribute to the computation of the $well - being$ metric as seen in equations (6.1) below:

$$
\begin{aligned}
&1 \leq st,sh,sa,sn,ss \leq 5 \\
&Wt{+}Wh{+}Wa{+}Wn{+}Ws{=}20 \\
&0 \leq Wt,Wh,Wa,Wn,Ws \leq 20 \\
&(st,sh,sa,sn,ss,Wt,Wh,Wa,Wn,Ws) \in \mathbb{N} \\
&WB{=}Wt * st{+}Wh * sh{+}Wa * sa{+}Wn * sn{+}Ws * ss
\end{aligned}
\tag{6.1}
$$

Where WB, a value between 0 and 100, represents well-being (higher is better). st, sh, sa, sn, and ss represent scores assigned to the values of temperature, humidity, air quality, noise, and WiFi signal strength, respectively. These scores must be between 1 and 5, with a higher score representing a range of more comfortable values for the stakeholder in a given domain.

Finally, Wt, Wh, Wa, Wn, and Ws represent individual weights for temperature, humidity, air quality, noise, and WiFi signal strength, respectively. Individual weights represent the importance a stakeholder assigns to a given metric and can be between 0 and 20, but their sum is always equal to 20.

For example, in a residential building, the weights for the different metrics would all have the same importance, which means the weight is equal to 4 for each metric.

If a single metric is crucial and the others are negligible, the value 20 is assigned as the weight for that critical metric, while the other metrics will have a weight equal to zero.

6.1.3.2 Respecting the NIST CPS Framework guidelines when modeling the well-being capability in the smart building domain.

The NIST CPS framework provides best practices to compose IoT and CPS capabilities while making sure important aspects are taken into consideration. We use the nine aspects of the NIST CPS Framework to discuss the well-being capability, and we explain how each aspect is addressed during the modeling phase. We will select a few concerns for each aspect to avoid exhaustiveness, and we will focus on the composition aspect as that's the most important one.

6.1.3.2.1 Functional

We leverage a set of sensors, including temperature, humidity, air quality, noise level, and WiFi signal strength, to compose a well-being capability in the smart building domain.

6.1.3.2.2 Business

Cost is addressed in two ways: the ICCF framework encourages the decomposition of complex services into reusable atomic capabilities, and a complex that contains temperature for example can provide APIs to other services to use its temperature feed, saving costs for other stakeholders related to implementing a brand new temperature service. The other way cost is saved through the use of energy-constrained sensors and low-computation devices (raspberry pie) to achieve the goals of this composition.

6.1.3.2.3 Human

Humans interact with the well-being capability in different ways depending on the environment. Hence, the need for weights related to each atomic capability to reflect the importance of a certain metric in a specific area of operation. Dust, for example, can be tolerated in a residential area but not in a chip factory. Similarly, certain levels of temperature can be accepted in a hospital, but they might feel uncomfortable in a residential building. Building composition with the human aspect in mind contributes to a successful and useful deployment.

6.1.3.2.4 Trustworthiness

The reason we run formal verification and algebraic specifications when describing the well-being capability is to make sure our assessment can be trusted as it runs through formal specification and verification pipelines. A well-being service should always yield results that are deemed comfortable by end users, hence the importance of implementing trustworthiness mechanisms (including formal verification and software validation through testing) during the modeling, prototyping, and testing phase.

6.1.3.2.5 Timing

The composite capability of well-being requires feeds of data from multiple sources, which might have different time parameters and refresh rates. Adapting the timing parameters in a way that's compatible and intelligible by the composition engine is crucial for a successful and meaningful composition.

6.1.3.2.6 Data

Data associated with the composite well-being capability must arrive at the composition engine in a timely and correct fashion for the system to correctly compose data feeds into a meaningful composite feature.

6.1.3.2.7 Boundaries

Data related to well-being should respect certain boundaries in terms of values envelop, units, and refresh rate. Measures should be in place to make sure composed data is in the right range and respects the anticipated properties.

6.1.3.2.8 Composition

This is the main element in the NIST CPS Framework which supports the composition foundations of the ICCF framework. The NIST CPS framework provides four

"4" guidelines for service composition in ICCF: i) **adaptability**: well-being information can come from different sensor technologies and different communication protocols, and the composite capability (well-being) must work seamlessly and adapt to these differences. ii) **complexity**: well-being information can come from complex or legacy systems and sensors and might require layers of simplification to be useable. iii) **constructivity**: the well-being composite metric must be able to leverage and combine modular data and device components with satisfying stakeholders' requirements. iv) **discoverability**: the ability of the composite well-being capability to connect to the underlying atomic services and sensors contributing to its calculation.

6.1.3.2.9 Lifecycle

This aspect ensures that the composite well-being service is reliable when deployed and safe to disconnect from the building when there's no need for it.

6.1.3.3 IoTCaP: An ICCF-based Platform For Implementing The Composite Capability Entities And Interactions using mPlane Semantics.

This paragraph describes entities and interactions within IoTCaP using the mPlane semantics and algebra, describes those interactions using a sequence diagram, and highlights the overall behavior using pseudo code.

6.1.3.3.1 Algebraic Description Of IoTCaP Entities and interactions

$IoTCaP$ is the platform built to implement the ICCF-based composite capability. $IoTCaP$ implements the following entities: $Actor$: logs into the IoTCaP front-end interface and selects domains or custom weights, $Auth$: authentication mechanism, $IoTCaP$: dashboard for visualizing the composite capability as well as inserting weights either manually or by selecting a specific domain, $well-being$: back-end process that implements the mPlane protocol by posting, $specification$ to sensors. The $well-being$ back-end also computes the composition based on the received weights and the composition formula, $Sensors$: receive specifications from the $well-being$ back-end and return $Results$ based on the $Capability$ schema. Consider $\mathscr{R}$, the space of capabilities that can be composed and decomposed using the ICCF framework.

Consider Ct, Ch, Ca,Cn, Cs, the Capabilities descriptors generated by the temperature, humidity, air quality, noise, and signal strength sensors, respectively.
Consider sCt, sCh, sCa,sCn, sCs, the specification posted to the sensors of temperature, humidity, air quality, noise, and signal strength respectively.
Consider rCt, rCh, rCa,rCn, rCs, the result obtained from the sensors of temperature, humidity, air quality, noise, and signal strength, respectively.

6.1.3.3.2 Algebraic Description Of IoTCaP Interactions Using mPlane Semantics

The different interactions that occur within IoTCaP can be described as follows:

• Authentication interaction: The $Auth$ entity authenticates an $Actor$ to the IoTCaP front-end; the $Actor$ can be a user or a process. The Authentication function is defined as Authenticate($Actor$, $Auth$); it compares $Actor$ credentials against $Auth$ database and allows access to the $IoTCaP$ front-end if there is a match.

• Discovery interaction: The $well-being$ back-end discovers entities that contribute to computing $well-being$. The discoverability function is defined as Disc($well-being$, C); it takes $well-being$, a composition manager, which also represents the $well-being$ back-end and C, a capability as input and returns a binary that shows whether or not that capability is discovered.

• SendSpec(Src,Dst,Specification) interaction: It is a request sC used in two instances: i) $sCwb$: a well-being specification (in the mPlane semantics, a specification is a request sent from a particular entity or a service to get data values or results from a sensor or another service) sent from the $IoTCaP$ front-end to the $Well-being$ back-end, or ii) sCt, sCh, sCa,sCn, sCs, sent from the $Well-being$ backed to the different $Sensors$ providing atomic capabilities.

• sendResult(Src,Dst,Result) interaction: According to the mPlane semantics, processes or sensors provide a $Result$ for the $Specification$ they receive based on their $Capability$. Two instances of this operation are witnessed in our implementation: i) results of sensor values returned upon processing specifications sent by the $well-being$ back-end. ii) results of the computed well-being value returned to the IoTCaP front-end.

• Composition function: Consider an operator ψ, which represents a k-ary composition operator. To illustrate composition, we assume that k=5, representing the five capabilities contributing to the composite capability of well-being, is made. The composition is an operator on values obtained after sending a specification to all atomic capabilities and receiving results.

• Capability weight and composition computation: Consider Wt , Wh, Wa, Wn, Ws, the weights of rCt, rCh, rCa,rCn, rCs respectively, the well-being composite result $rCwb$ can be expressed as seen in equation (6.2):

$$\begin{aligned} &\psi : \mathbb{N}^5 \rightarrow \mathbb{N} \\ &(rCt,rCh,rCa,rCn,rCs) \rightarrow rCwb \\ &\rightarrow Wt * rCt+Wh * rCh+Wa * rCa+Wn * rCn+Ws * rCs \end{aligned} \tag{6.2}$$

• Specification decomposition function: since the well-being Cwb represents a composite capability $CapabilityDescriptor$, the $Specification$ $sCwb$ will be decomposed to its atomic $Specifications$ (sCt, sCh, sCa,sCn, sCs) by applying the decomposition operator ψ^{-1} as seen in expression (6.3):

$$\begin{aligned} &\psi^{-1} : \mathscr{R} \rightarrow \mathscr{R}^5 \\ &(sCwb) \rightarrow (sCt, sCh, sCa, sCn, sCs) \\ &\qquad \rightarrow \psi^{-1}(sCwb) \end{aligned} \tag{6.3}$$

Figure 6.1 illustrates graphically the composition of atomic capabilities contributing to the calculation of well-being in a smart building after being weighted by the user/stakeholder.

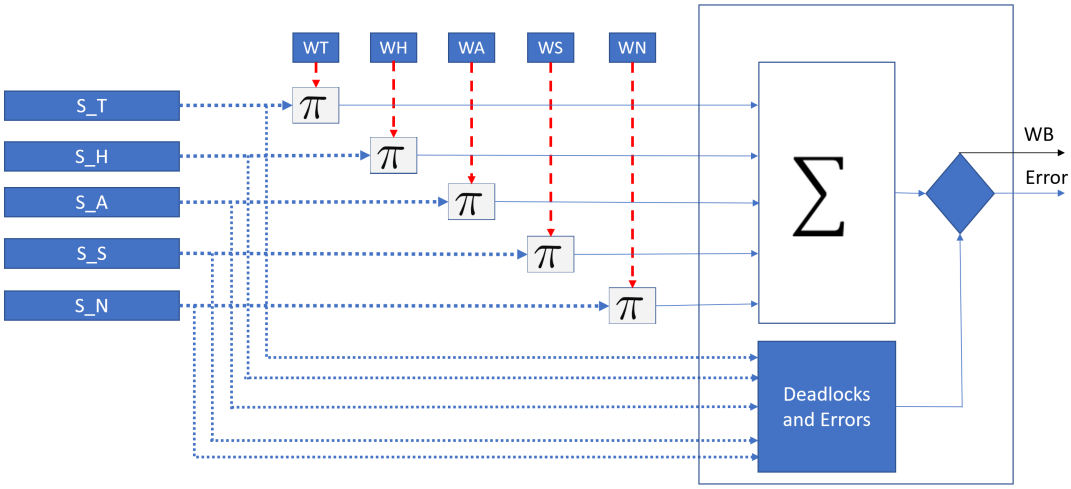

FIGURE 6.1: Composition of the well-being composite service using atomic capabilities that are weighted based on user/stakeholder preferences.

6.1.3.3.3 Illustrating Entities And Interactions Using A Sequence Diagram

The sequence diagram in Fig. 6.2 summarizes the interactions between IoTCaP entities. The "Actor" authenticates to its profile in IoTCaP and then selects weights for different metrics based on their importance. Once weights are submitted, the "Well being" composition engine computes a well-being value and returns it to the Actor as well as values for atomic metrics. In the case where the sum of weights exceeds 20, an error is displayed to the Actor.

6.1.3.3.4 Illustrating mPlane-described IoTCaP Entities And Interactions Using Pseudo-code

The pseudo-code in Algorithm 2 summarizes the well-being composition operations. In particular, line 15 refers to the computation of the composite capability of well-being.

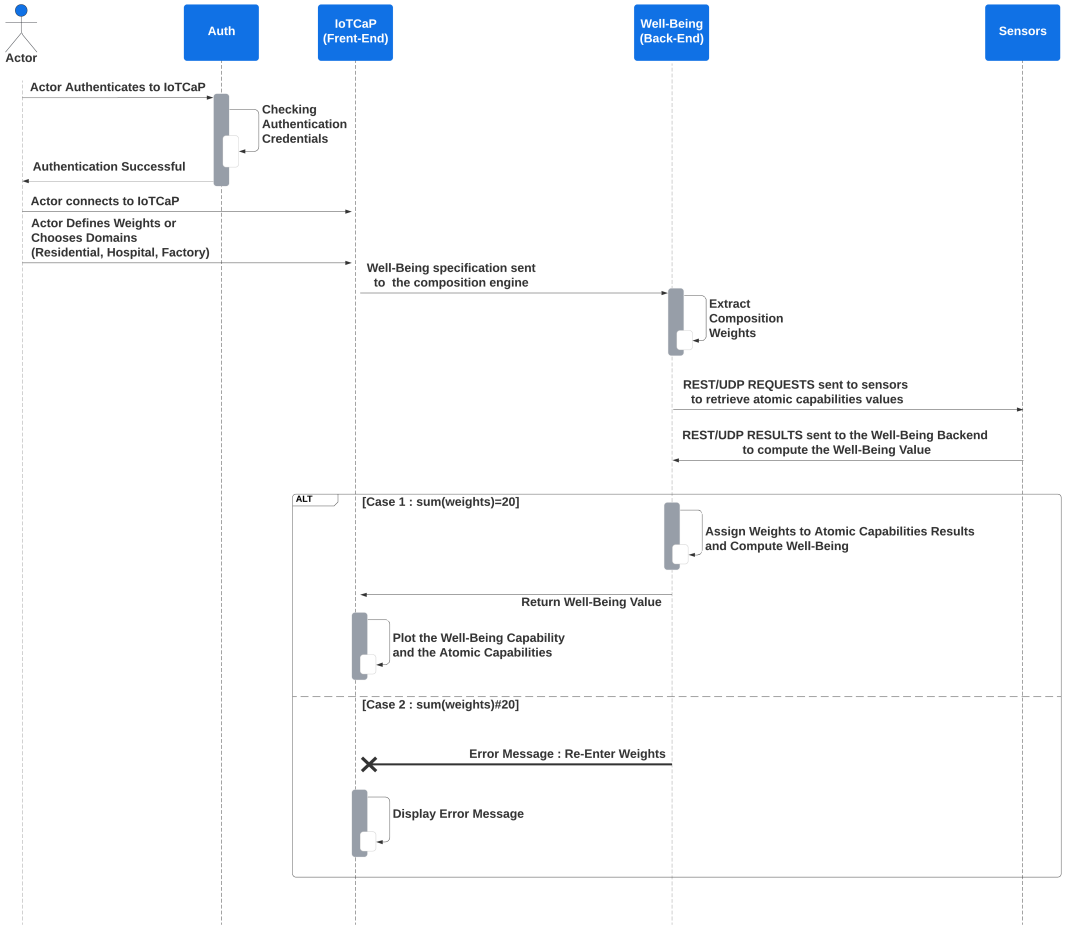

FIGURE 6.2: Sequence diagram for the well-being capability expressed using mPlane interactions.

6.1.4 Formal Specification And Verification

Formal verification is necessary because it is a reliable way to verify the mPlane-modeled functions. Furthermore, fixing and optimizing the well-being model becomes possible by studying deadlocks and in-variants. This subsection converts mPlane semantics defined previously to PLUSCAL language, which is translated to TLA specification. Then the model checker yields the state space, and by studying invariants, assessing the correctness of the model, or proposing corrections in the case of errors or deadlocks is possible.

The choice of TLA+ was based on the fact that it is a trusted tool for verifying microservices, including in commercial solutions such as AWS [226], where it was leveraged to verify the correctness of properties such as fault tolerance in storage services.

6.1.4.1 Formal Verification Environment

TCL version 1.5.7, TLA+'s model checker, is executed in a four-core CPU-equipped Linux Ubuntu 14.04 VM with 8GB RAM.

Algorithm 2 IoTCaP platform mPlane interactions.

if $Ct, Ch, Cs, Ca, Cn \in \mathscr{R}$ and
Disc($well-being$,(Ct, Ch, Cs, Ca, Cn))$\leftarrow$ true and Authenticate($Actor$, $Auth$)$\leftarrow$ true **then**
    sendSpec($IoTCaP$, $well-being$, $sCwb$)
ψ^{-1}(sCwb)$\rightarrow$(sCt, sCh, sCa,sCn, sCs)
sendSpec($well-being$, Ct, sCt) and
    sendSpec($well-being$, Ch, sCh) and
    sendSpec($well-being$, Ca, sCa) and
    sendSpec($well-being$, Cn, sCn) and
    sendSpec($well-being$, Cs, sCs)
sendResult(Ct, $well-being$, rCt) and
    sendResult(Ch, $well-being$, rCh) and
    sendResult(Ca, $well-being$, rCa) and
    sendResult(Cn, $well-being$, rCn) and
    sendResult(Cs, $well-being$, rCs)
ψ(rCt,rCh,rCa,rCn,rCs)$\rightarrow$($rCwb$)
sendResult($well-being$, $IoTCaP$, $rCwb$)

6.1.4.2 Adapting The State Space For PLUSCAL's Requirements

Converting mPlane algebra to PLUSCAL language requires some modifications, including adapting the state space of sensor values to include positive values only. Fig. 6.3 describes a pipeline of operations that yields values compatible with PLUSCAL. The outcome is shown in Table 6.1 where sensor values -provided by the temperature and humidity sensor (DHT22), noise sensor (KY-038), air quality sensor (SDS011), and WiFi signal strength sensor (ESP8266 SOC)- were translated into positive values and distributed among ranges representing different well-being areas. These areas are assigned scores that reflect how they contribute to the well-being metric.

For example, the temperature sensor provides temperature values tv between -40 ºC and 125 ºC. To make these values positive, the value 40 ºC is added to both ends of this range to make all values positive (because the minimum temperature value is -40 ºC). The best values of temperature that contribute to well-being are between 20 ºC and 22 ºC. This range RT of values is assigned a score RTs of 5; the equivalent range in PLUSCAL is the positive range RpT with values between 60 ºC and 62 ºC. A score $RpTs$ of 5 is assigned to this positive range as well, and this value is used to compute the overall well-being after subjecting all the sensor values to the same pipeline processes described in Fig. 6.3. In the case of humidity, sensor values hv won't require adaptation as the values are already positive, as it can be seen in Table 6.1.

6.1.4.3 Interpreting Core Composition Functions Into PLUSCAL And Translation Into TLA Specification

Two core functions for composing the well-being capability are interesting from a formal verification perspective:

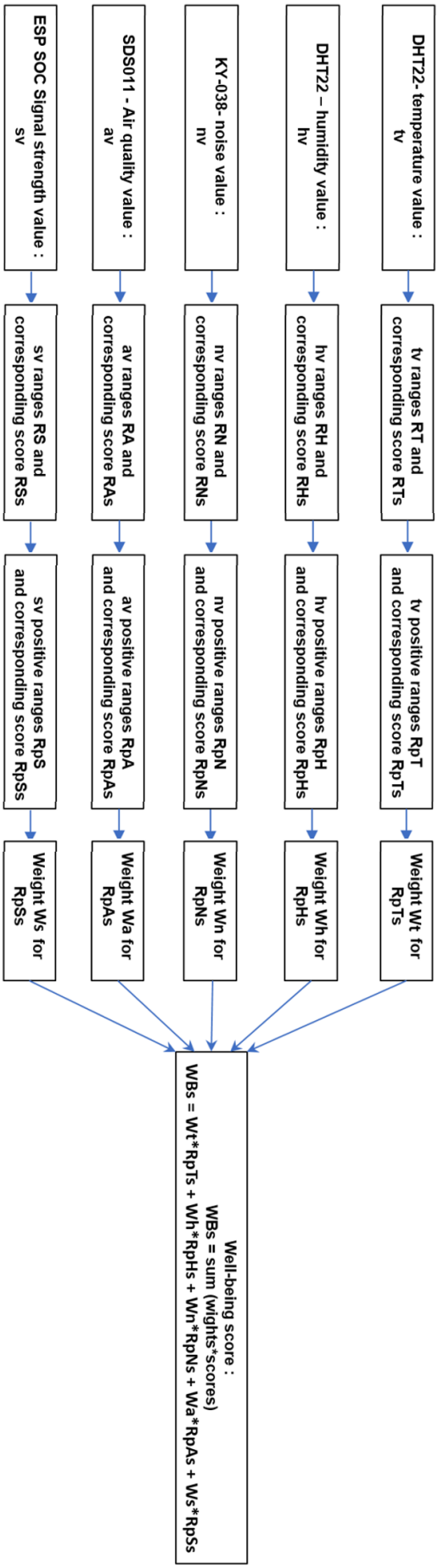

FIGURE 6.3: Modeling the state space for the well-being composition values based on corresponding atomic capabilities values, ranges, projections, and scores.

TABLE 6.1: State space of values, ranges, and scores related to well-being atomic capabilities in a residential building.

Atomic Capability	Sensors	Units	Low value	High Value	Real sensor values ranges (RT, RH, RN, RA, RS) and score assignment per range (RTs, RHs, RNs, RAs, RSs)					TLA+ compatible sensor values ranges (RpT, RpH, RpN, RpA, RpS) and score assignment per range (RpTs, RpHs, RpNs, RpAs, RpSs				
					1	2	3	4	5	1	2	3	4	5
Temperature	DHT22	$^{\circ}C$	tv		RT					RpT				
			-40	125	[-40,-1] or [41,125]	[0,9] or [31,40]	[10,16] or [26,30]	[17,19] or [23,25]	[20,22]	[0,41] or [81,165]	[40,49] or [71,80]	[50,56] or [66,70]	[57,59] or [63,65]	[60,62]
Humidity	DHT22	%	tv		RH					RpH				
			0	100	[0,10] or [80,100]	[11,20] or [61,79]	[21,25] or [55, 60]	[26,29] or [51,54]	[30,50]	[0,10] or [80,100]	[11,20] or [61,79]	[21,25] or [55, 60]	[26,29] or [51,54]	[30,50]
Noise	KY-038	dB	tv		RN					RpN				
			23	129	[81,129]	[51,80]	[41,50]	[31,40]	[25,30]	[81,129]	[51,80]	[41,50]	[31,40]	[25,30]
Air Quality	SDS011 2.5UG	$\mu g/\mathrm{m}^3$	tv		RA					RpA				
			0	999	[101,999]	[36,100]	[21,35]	[13,20]	[0,12]	[101,999]	[36,100]	[21,35]	[13,20]	[0,12]
WiFi Signal Strength	ESP8266 WiFi SOC	dBm	tv		RS					RpS				
			-80	-5	[-80,-61]	[-60,-51]	[-50,-41]	[-40,-31]	[-30,-5]	[0,19]	[20,29]	[30,39]	[40,49]	[50,75]

6.1.4.3.1 The score assignment function

The role of this function is to assign scores from 1 to 5 for sensor values based on how much they contribute to well-being.

6.1.4.3.2 The Composition computation function

The role of this function is to compute well-being based on both the assigned scores of sensor ranges of values and metric weights.

Fig. 6.4 shows both of these functions as described in the PLUSCAL language.

After translating the PLUSCAL description, the TLA specification is generated, as seen in Fig. 6.5.

6.1.4.4 Model Checking And State-Space Analysis

6.1.4.4.1 Running Symbolic Execution And Discussing Results Of The State Space

The symbolic execution of the model results runs combinations of all atomic capabilities scores and weights to assign score values to humidity ranges and determine the well-being state space. In Fig. 6.7, it took 4 minutes and 31 seconds to perform symbolic execution; TLC can improve its execution time when it leverages multiple EC2 or Azure cloud instances. The number of states generated across all combinations is 60750000, with 35437500 distinct states, which means more optimization can be done on the model. The queue experienced congestion 31 seconds after execution, but it was emptied over. This simple but efficient method of calculating well-being using TLC's model checker was verified, and the results show the correctness of the core functions of the composite capability.

```
1  ---------------------------- MODULE wellbeing ---------------------
2  EXTENDS Integers, TLC
3  (*--algorithm wellbeing
4  variables hv \in 1..100, RpHs \in {1,2,3,4,5},
5            RpTs \in {1,2,3,4,5}, RpNs \in {1,2,3,4,5},
6            RpAs \in {1,2,3,4,5}, RpSs \in {1,2,3,4,5},
7            WT \in 0..8, WH=4, WS \in 0..8, WA=4, WN=4
8  begin
9    HumidtyRangeVerification:
10     if ((hv<51)/\(hv>29)) then
11         RpHs := 5;
12         else if (((hv<30)/\(hv>25))\/((hv<55)/\(hv>50))) then
13         RpHs := 4;
14         else if (((hv<26)/\(hv>20))\/((hv<61)/\(hv>54))) then
15         RpHs := 3;
16         else if (((hv<21)/\(hv>10))\/((hv<80)/\(hv>60))) then
17         RpHs := 2;
18         else if (((hv<11)/\(hv>0))\/((hv<101)/\(hv>79))) then
19         RpHs := 1;
20         end if;
21         end if;
22         end if;
23         end if;
24         end if;
25   CompositionScoreVerification:
26       if (WH+WT+WS+WA+WN<21)then
27           print ((WH*RpHs)+(WT*RpTs)+(WA*RpAs)+(WN*RpNs)+(WS*RpSs));
28             end if;
29 end algorithm;*)
```

FIGURE 6.4: Screenshot from TLA+ indicating the functions to verify described using the PLUSCAL language.

6.1.4.4.2 Preventing State Space Explosion

Score assignment was executed for humidity values alone to reduce the state space and prevent state space explosion. Once humidity is verified, swapping ranges to compute score assignment for the other sensor values is straightforward. This technique saves hours of symbolic execution run-time. TLA+ can also save time by running the symbolic execution on cloud instances such as EC2 or Azure instances.

6.1.4.5 Deadlock Case And Corrective Measures

As software developers, applying formal methods and model checking enables thinking above the code level, validating the understanding of the composite capabilities, and finding critical bugs that are difficult to spot. For example, the invariant study aims to test the capability in case well-being weights aren't equal to 4. i.e., *Wt* and *Ws* randomly vary between 0 and 8.

```
\* BEGIN TRANSLATION (chksum(pcal) = "b4c94829" /\ chksum(tla) = "f7f24b1")
VARIABLES hv, RpHs, RpTs, RpNs, RpAs, RpSs, WT, WH, WS, WA, WN, pc

vars == << hv, RpHs, RpTs, RpNs, RpAs, RpSs, WT, WH, WS, WA, WN, pc >>

Init == (* Global variables *)
        /\ hv \in 1..100
        /\ RpHs \in {1,2,3,4,5}
        /\ RpTs \in {1,2,3,4,5}
        /\ RpNs \in {1,2,3,4,5}
        /\ RpAs \in {1,2,3,4,5}
        /\ RpSs \in {1,2,3,4,5}
        /\ WT \in 0..8
        /\ WH = 4
        /\ WS \in 0..8
        /\ WA = 4
        /\ WN = 4
        /\ pc = "HumidtyRangeVerification"

HumidtyRangeVerification == /\ pc = "HumidtyRangeVerification"
                            /\ IF ((hv<51)/\(hv>29))
                            THEN /\ RpHs' = 5
                            ELSE /\ IF (((hv<30)/\(hv>25))\/((hv<55)/\(hv>50)))
                            THEN /\ RpHs' = 4
                            ELSE /\ IF (((hv<26)/\(hv>20))\/((hv<61)/\(hv>54)))
                            THEN /\ RpHs' = 3
                            ELSE /\ IF (((hv<21)/\(hv>10))\/((hv<80)/\(hv>60)))
                            THEN /\ RpHs' = 2
                            ELSE /\ IF (((hv<11)/\(hv>0))\/((hv<101)/\(hv>79)))
                            THEN /\ RpHs' = 1
                            ELSE /\ TRUE
                            /\ RpHs' = RpHs
                            /\ pc' = "CompositionScoreVerification"
                            /\ UNCHANGED << hv, RpTs, RpNs, RpAs, RpSs, WT, WH,
                                            WS, WA, WN >>

CompositionScoreVerification == /\ pc = "CompositionScoreVerification"
        /\ IF (WH+WT+WS+WA+WN<21)
        THEN /\ PrintT(((WH*RpHs)+(WT*RpTs)+(WA*RpAs)+(WN*RpNs)+(WS*RpSs)))
        ELSE /\ TRUE
        /\ pc' = "Done"
        /\ UNCHANGED << hv, RpHs, RpTs, RpNs, RpAs,
                        RpSs, WT, WH, WS, WA, WN >>

(* Allow infinite stuttering to prevent deadlock on termination. *)
Terminating == pc = "Done" /\ UNCHANGED vars

Next == HumidtyRangeVerification \/ CompositionScoreVerification
        \/ Terminating

Spec == Init /\ [][Next]_vars

Termination == <>(pc = "Done")

\* END TRANSLATION
=============================================================================
```

FIGURE 6.5: Screenshot from TLA+ indicating the composition functions to verify translated to TLA specification.

However, if well-being is set as an invariant where its value is strictly inferior to 100, and with random weights or without proper controls, the symbolic execution of the well-being model might yield values superior to 100, which is an incorrect outcome that causes TLC to throw deadlock errors. Adding a control instruction that forces TLC to check whether the sum of weights equals 20 would prevent this deadlock case.

6.1.5 IoTCaP Implementation, Results, and Analysis

This subsection describes the experiment environment, hardware, and toolkit used to implement the IoTCaP platform. The results obtained are also discussed in light of other efforts and based on different stakeholders' perspectives.

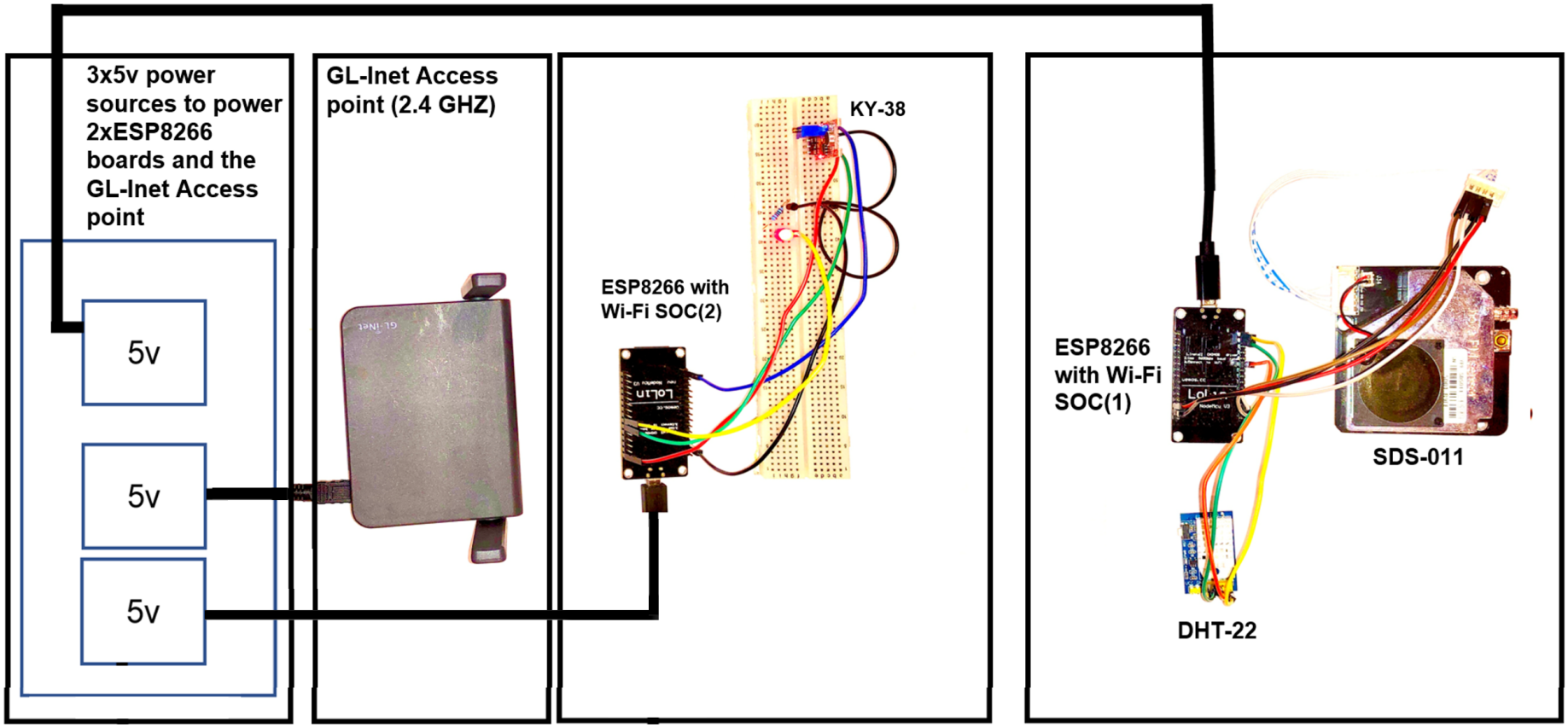

FIGURE 6.6: Experiment devices, boards, and atomic sensors: 2 X ESP8266, 1 X DHT22, 1 X SDS011, 1 X GLiNET WiFi Access, 1 X KY038, 3 X 5V DC outlets.

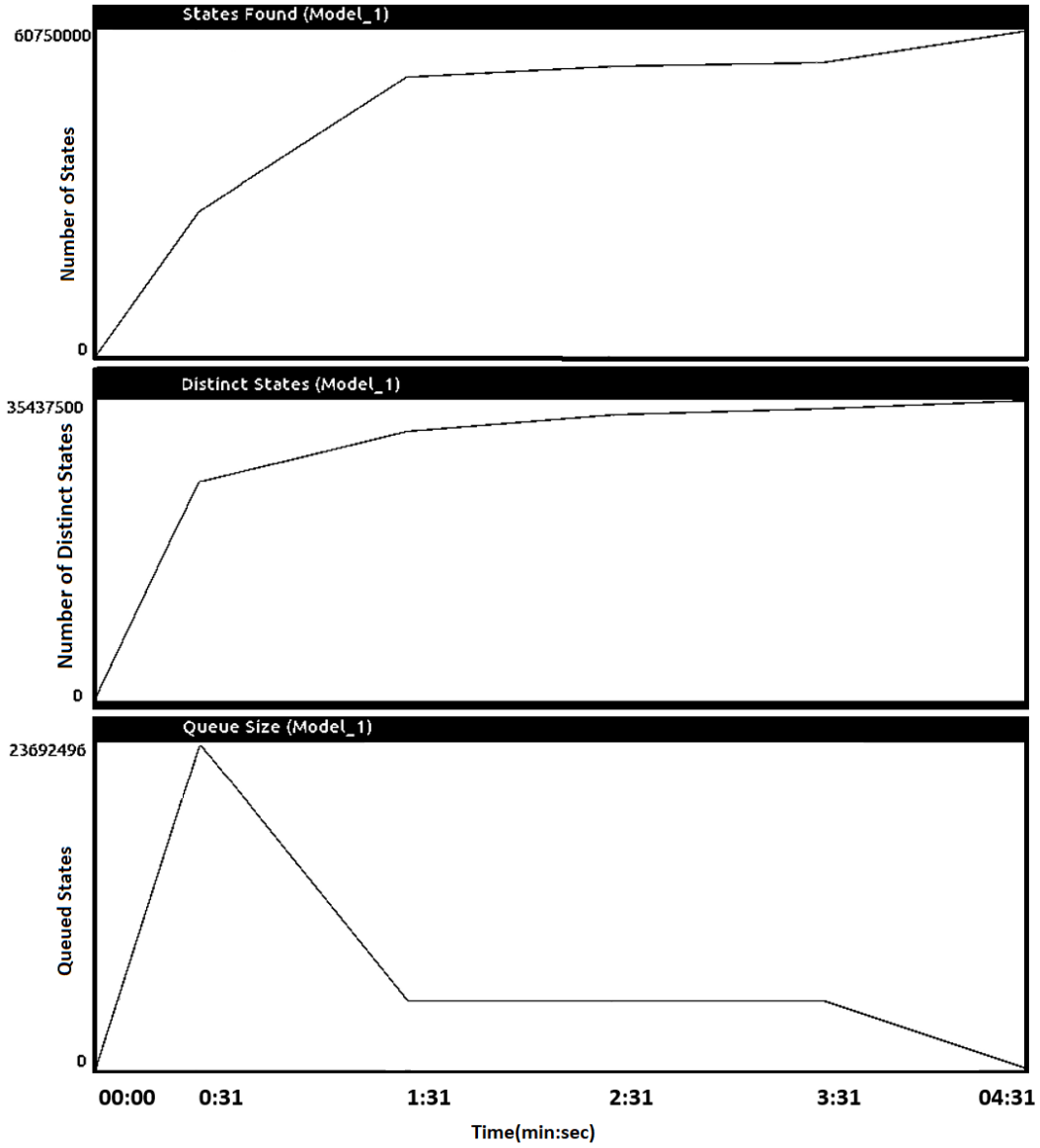

FIGURE 6.7: TLC model checking results show historical data of the well-being model's total number of generated states, distinct states, and the state's queue handling.

6.1.5.1 Experiment Environment, Hardware, and Toolkits

An Ubuntu 18.04 desktop machine with an i711700 eight-core processor and 32 GB of ram is running eclipse, where Vert.X verticles are collecting data from hardware described in Fig. 6.6.

The sensors used include a DHT-22 temperature and humidity sensor, a KY-38 noise sensor, an SDS-011 particulate matter sensor for measuring air quality, a GL-Inet Acces point that provides network access to two ESP8266 boards that play two roles: provide signal strength values to the IoTCaP platform and also running a web-server/client that collect all sensor data and sends it to a custom API in the Linux-Machine where vert.X verticles are running.

TABLE 6.2: Comparing implementation technologies.

Technology	Ref	Performance	Long-running Processes	Logging	Latency [ms]	Application Related Pros(+) and Cons(-)
Vert.X	[243]	EventLoop model (less overhead, less CPU usage)	Supported (JVM Support)	Vert.X verticles offer significantly faster messages for error recovery compared to Akka	1.8	(+)More flexibility with callbacks, futures, Java Completion Stage, Kotlin coroutines, RxJava, and fibers support. (-) Many conflicting versions.
Node.js	[55]	reactor pattern (High CPU usage, high memory usage)	Supported	Errors in Node.js are handled through exceptions.	1.2	* Both Node.js and Vert.X give scalable server-side applications, but Vert.X scales better than Node.js.
Akka	[249]	Actor model (less overhead, moderate CPU usage)	Supported (JVM Support)	Akka actors offer messages for error recovery	2.8	* Vert.X is more suited for a wider variety of tasks, while Akka fits more for designing large systems with several subsystems that handle humongous concurrency.
Spring Framework	[298]	(Slow startup time, high load, High Heap memory usage)	Supported (JVM Support)	Default Logback Logging	4.2	(-)Spring is less flexible and slower than vert.X and uses blocking APIs
Quarkus	[167]	Vertx EventLoop (less overhead, high CPU usage)	Supported (JVM Support)	JBoss Log Manager	4.7	(+)Runs exceptionally well in container environments like Kubernetes. Vert.X is used to powering the Quarkus networking stack.
Netty	[210]	memory and CPU overhead	Supported (JVM Support)	JdkLoggerFactory	1.3	(+)Provides non-blocking I/O APIs for the JVM. Slightly faster than Vert.X (-)APIs low-level compared to Vert.X

6.1.5.2 Micro-services Toolkits

ICCF is platform agnostic; platforms are swapped when better ones are available. For this project's microservices toolkit needs, Vert.X is picked as it satisfies latency, performance, and logging requirements needed for IoTCaP as concluded from Table 6.2. In addition, Vert.X is an approachable and efficient toolkit for writing asynchronous and reactive applications on the JVM [243]. Vue.JS, a lightweight front-end platform, is used for its compatibility with the libraries needed to transmit data between Vert.X verticles and the front-end interface.

Code for both IoTCaP back-end (Vert.X) and IoTCaP front-end (Vue.JS) is available in Github [159].

6.1.5.3 Running IoTCaP and discussing stakeholders' concerns and requirements

Figure 6.9 illustrates IoTCaP components upon implementation:

• Different users connect to IoTCaP's Vue.JS front end via the Keycloak [66] authentication API; this ensures that privacy concerns that we discussed in RQ15/AQ15 in Chapter 4 are addressed. as different user profiles store different preferences in terms of the well-being atomic capabilities weights.

• Meanwhile, the HazelCast layer ensures that registered services are secure and trustworthy. This is guaranteed by registering trusted objects using JSON Web Tokens (JWTs) [127].

• The devices layer represents sensors and data sources, each are using custom/-factory protocols to connect to their corresponding Vert.X verticle, which makes the capabilities of the sensors composition-ready.

• The Vert.X back-end connects to the sensors and devices layer to retrieve capabilities data; it also implements composition functions to aggregate sensor data

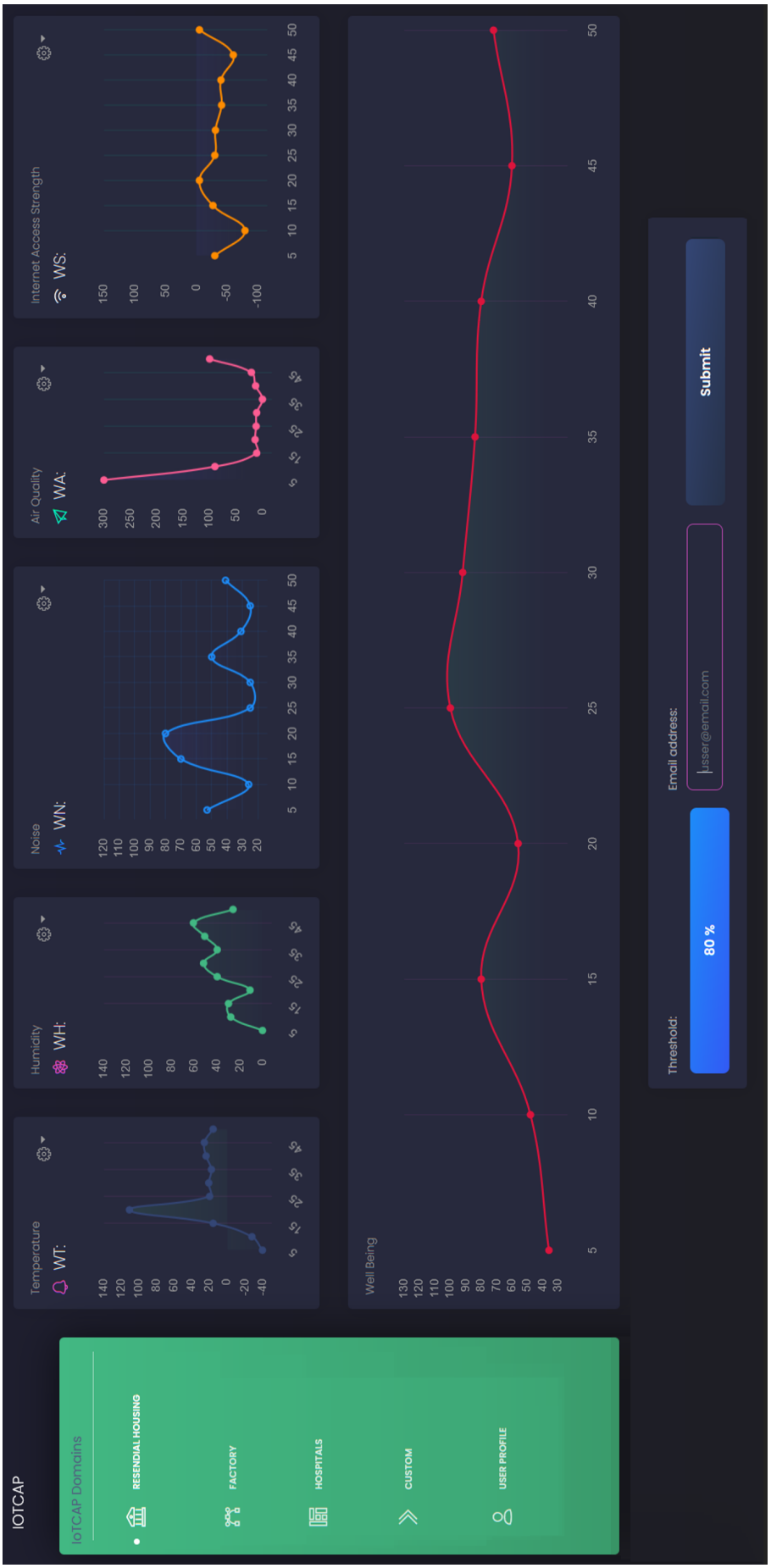

FIGURE 6.8: Screenshot of the Vue.JS IoTCaP front-end, an actor can fill in the weights for the capabilities based on his/her preferences and submit a specification to the well-being back-end, a result is returned that carries both atomic capabilities as well as the composite capability of well-being based on inserted weights.

into value-added capabilities such as well-being.

• The event-bus or the Axios library enables the Vue.JS front-end and the Vert.X back-end to communicate and exchange data.

• The Vue.JS front end allows users to select different views: atomic capabilities views or composite capabilities views, or both.

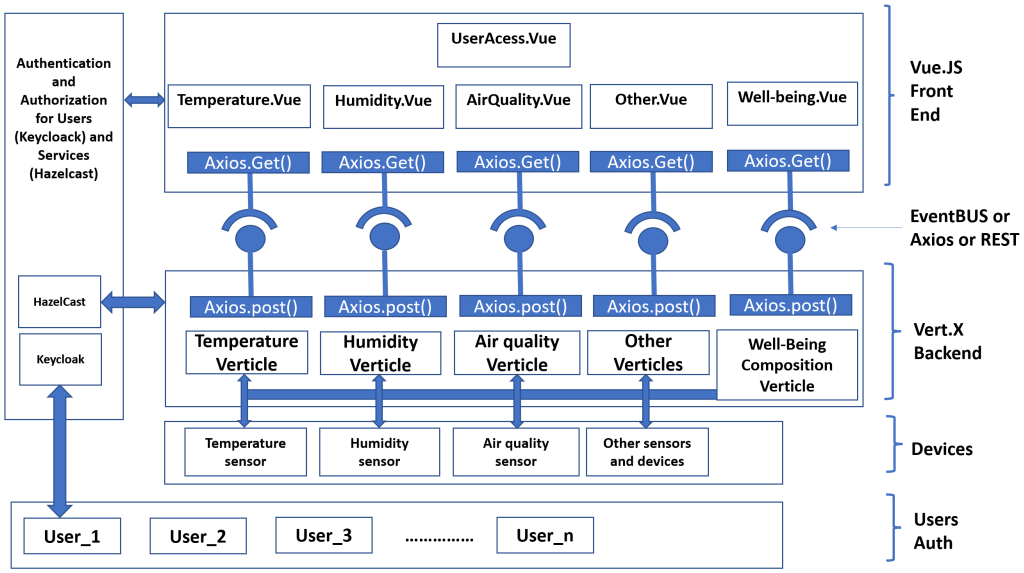

FIGURE 6.9: An illustration of the IoTCaP implementation layers for collecting and composing smart building data into a composite capability.

After powering the sensors, running Vert.X verticles, and launching the GUI, the IoTCaP platform is up, as seen in Fig. 6.8. An *Agent* picks domains of preference based on the smart building requirements. For example, choosing a smart building type of factory for semiconductors manufacturing would suggest a higher weight for air quality as dust is intolerable in clean rooms. On the other hand, picking a residential building as a domain suggests equal weight for all metrics as they are equally important.

The front-end interface also allows users to set thresholds for well-being and get e-mail warnings when the value sinks below the desired values.

6.1.6 Conclusion, Challenges, And Future Work

This subsection introduced a stakeholder-defined well-being composition capability based on the ICCF framework foundations. First, scores and weights to quantitatively compute the metric of the composite well-being capability are introduced. Second, an algebraic specification is developed using the ICCF framework semantics to describe entities and interactions. Third, formal verification is executed on the well-being model. Fourth, the core composition operations and the state-space dimensions are analyzed. Fifth, A Vert.X/Vue.JS implementation for the well-being composite capability is built and presented, and run-time behavior is discussed

based on two stakeholders' perspectives: residential buildings and factories.

Two challenges related to this work were identified: i) when composing atomic capabilities, gaining insight into the composite capability is straightforward, but keeping sight of what caused a particular state of the composite capability isn't always an easy task. A suggested solution is to leverage Artificial Intelligence by assigning profiles to atomic capabilities, which would lead to a better understanding of what caused a particular state of a composite capability. An identical issue was addressed in [98], where researchers recognized appliances that consume the most energy based on their energy profile. ii) The second challenge is when running a model checking for many variables and values; it might take a long time to execute the symbolic execution. A solution to this challenge would involve leveraging cloud instances to run TLC on multiple AWS or Azure instances and assessing the benefits of this approach.

For future work, the QoS aspect of the studied composition will be assessed, especially from a privacy and scalability perspective.

In the next sections, composite capabilities in other domains of interest will be explored: safety assessment in smart transportation systems (Section 6.2) and health improvement in the smart health domain (Section 6.3).

6.2 Safety as a Composite Capability in the Smart Transportation Domain.

6.2.1 Introduction

After we discussed well-being as a composite capability in the smart building domain, we discuss in this section another application of interest that we are going to describe according to ICCF's foundations, semantics, and formal specifications and verification. The application we will discuss in this section is assessing safety during an autonomous vehicle emergency braking experiment. The manoeuver of emergency braking can have two outcomes based on different environment parameters: Safe or NOT Safe.

6.2.2 Related Work

6.2.2.1 Exisiting efforts on assessing safety in smart transportation applications

Multiple efforts have attempted to assess safety as a composite metric in the smart transportation domain; the NIST OES Framework [111] is a recent approach to describe the Operational Design Domain (ODD) components in a tangible and measurable fashion, and in a way that facilitates the description of a safe manoeuver, by identifying the envelope -or range- of acceptable values related to the different components of an autonomous driving scenario during a safety-critical manoeuver. Similarly, researchers in [310] leveraged atomic capabilities such as speed and position of vehicles to interpret traffic jams on a certain road, but a formally specified safety function wasn't provided. In [96], RAND corporation proposed a framework

for measuring safety in autonomous vehicles, but no tangible composite capability was provided to address this concern and formal specification and verification of such a capability weren't proposed.

6.2.3 An ICCF definition of Safety as a Composite Capability:

6.2.3.1 A proposed definition of safety based on stakeholders concerns:

In our work, we leverage specific components in the related work to define a safety metric for a given autonomous driving manoeuver.

One way to simplify safety assessment during an autonomous driving manoeuver is to treat it as either a binary property:

In this case, a manoeuver can be either safe or not safe. We express this outcome in a tangible fashion based on the ICCF description of a composite capability aggregated through atomic features of the studied domain.

6.2.3.2 Respecting the NIST CPS Framework guidelines when modeling the safety composite capability:

The NIST CPS framework provides best practices to compose IoT and CPS capabilities while making sure key aspects are respected. We go through the nine elements of the NIST CPS Framework and we explain how each aspect is going to be addressed during the modeling phase. We will select a few concerns for each aspect to avoid exhaustion.

6.2.3.2.1 Functional

For the functional aspect, the safety composite capability will leverage speed and distance sensors to retrieve information related to these atomic capabilities necessary to assess the safety of a vehicle or its occupants during an emergency braking maneuver.

6.2.3.2.2 Business

Cost is one concern among multiple business concerns; as we are measuring the safety of autonomous vehicles in a simulation setting, the cost is reduced compared to real-world testing.

6.2.3.2.3 Human

Humans interact with the safety metric in two ways:
i) As readers of the safety metric, the assessed results need to be accessible and easily readable for end users. Binary outcomes (Safe/Not safe) or color schemes that represent how safe the ADS is are examples of accessible ways to inform humans about the safety metric. i) As subjects of an assessment: when the human driver or passenger's safety is impacted (or improved) during an autonomous driving experiment (regardless of the integrity of the car).

6.2.3.2.4 Trustworthiness

The reason we include formal verification and algebraic specifications when describing the safety capability is to make sure our assessment can be trusted as it runs through formal specification and verification pipelines.

6.2.3.2.5 Timing

Timing for the Safety Composite Capability during an emergency braking experiment is of crucial importance. A distracted driver may not detect an obstacle earlier, and as a result, a few seconds can make the difference between survival and non-survival. This can also occur for an ADS if the system responsible for detecting obstacles fails.

6.2.3.2.6 Data

Data associated with the safety composite metric must arrive at the composition engine in a timely and correct fashion for the system to correctly compose data feeds into a meaningful composite feature.

6.2.3.2.7 Boundaries

Data related to safety should respect certain boundaries in terms of values envelop, units, and refresh rate. Measures should be in place to make sure composed data is in the right range and respects the anticipated properties.

6.2.3.2.8 Composition

This is the most important element in the NIST CPS Framework with regard to the theme of the topic we are addressing in this effort. The NIST CPS framework provides four "4" guidelines for service composition: i) **adaptability**: safety information can come from different sensor technologies and different communication protocols, and the safety composite capability must work well and adapt to these differences. ii) **complexity**: safety information can come from complex or legacy systems and sensors and might require layers of simplification to be useable. iii) **constructivity**: the safety composite metric must be able to leverage and combine modular data and device components with satisfying stakeholders' requirements. iv) **discoverability**: the ability of the safety composite capability to connect to the underlying atomic services and sensors contributing to its calculation.

6.2.3.2.9 Lifecycle

This aspect ensures that the composite safety service is reliable when deployed and safe to disconnect from the ADS when there's no need for it.

6.2.3.3 IoTCaP: An ICCF-based Platform For Implementing The Composite Capability Entities And Interactions using mPlane Semantics.

6.2.3.3.1 Illustrating Entities And Interactions Using A Sequence Diagram

The sequence diagram in Fig. 6.10 summarizes the interactions between IoTCaP entities. The "Actor", typically a simulation or test engineer, authenticates to its profile in IoTCaP and then inputs parameters that describe the vehicle, the manoeuver, the environment, and the obstacles, if any. Once these parameters are

submitted to the front-end interface and communicated to the back-end composite service, the safety assessment composition function computes one or multiple safety factors and returns the result to the Actor, with the possibility to display atomic capabilities values related to the maneuver.

6.2.4 Formal Specification And Verification

If, for a given environment, we elect to study safety based on two parameters: how far the vehicle is from an obstacle (distance, with discreet values ranging from 1 to 100 meters) and how fast the vehicle is at the moment of detecting the obstacle (speed, with discreet values ranging from 1..100 mph), we can express the Safety as a composite capability of two atomic features (speed, distance), with the safety capability having a binary outcome (as opposed to well being which has a range of 0%-100%)).

6.2.4.1 Interpreting Core Composition Functions Into PLUSCAL And Translation Into TLA Specification

Figure 6.11 shows the translation of the experiment semantics from natural language to PLUSCAL and from PLUSCAL to TLA within the TLA+ toolkit.

6.2.4.2 Model Checking And State-Space Analysis

Figure 6.12 shows the outcome of the symbolic execution for the given state space: the state of values yields 20000 unique cases, with the safety taking two outcomes as seen in the log output ("SAFE", "NOT Safe").

6.2.5 IoTCaP Implementation, Results, and Analysis

In [121][161], we built an experiment that simulates the emergency braking of an autonomous vehicle when an obstacle is detected. The simulation environment leveraged IGNITE, a proprietary vehicle dynamics physics engine, to simulate acceleration, braking, and deceleration of the vehicle, while the control signals, including the speed and braking requests, were originating from the UCEF [251] environment thanks to co-simulation capabilities.

Parameters that influence the "time to stop" include vehicle mass, distance to the obstacle, vehicle aerodynamics, vehicle speed at the moment of detection of the obstacle, tire rolling resistance, and wind speed.

To take into consideration all the components of the simulation when calculating the stopping distance -and, as a result, assess safety- the function for assessing safety was simplified to take into consideration two variables instead of the full state space that impacts an emergency braking experiment.

As explained earlier, to simplify the specification of the safety composition function, we only consider two discrete atomic capabilities, i) distance to the

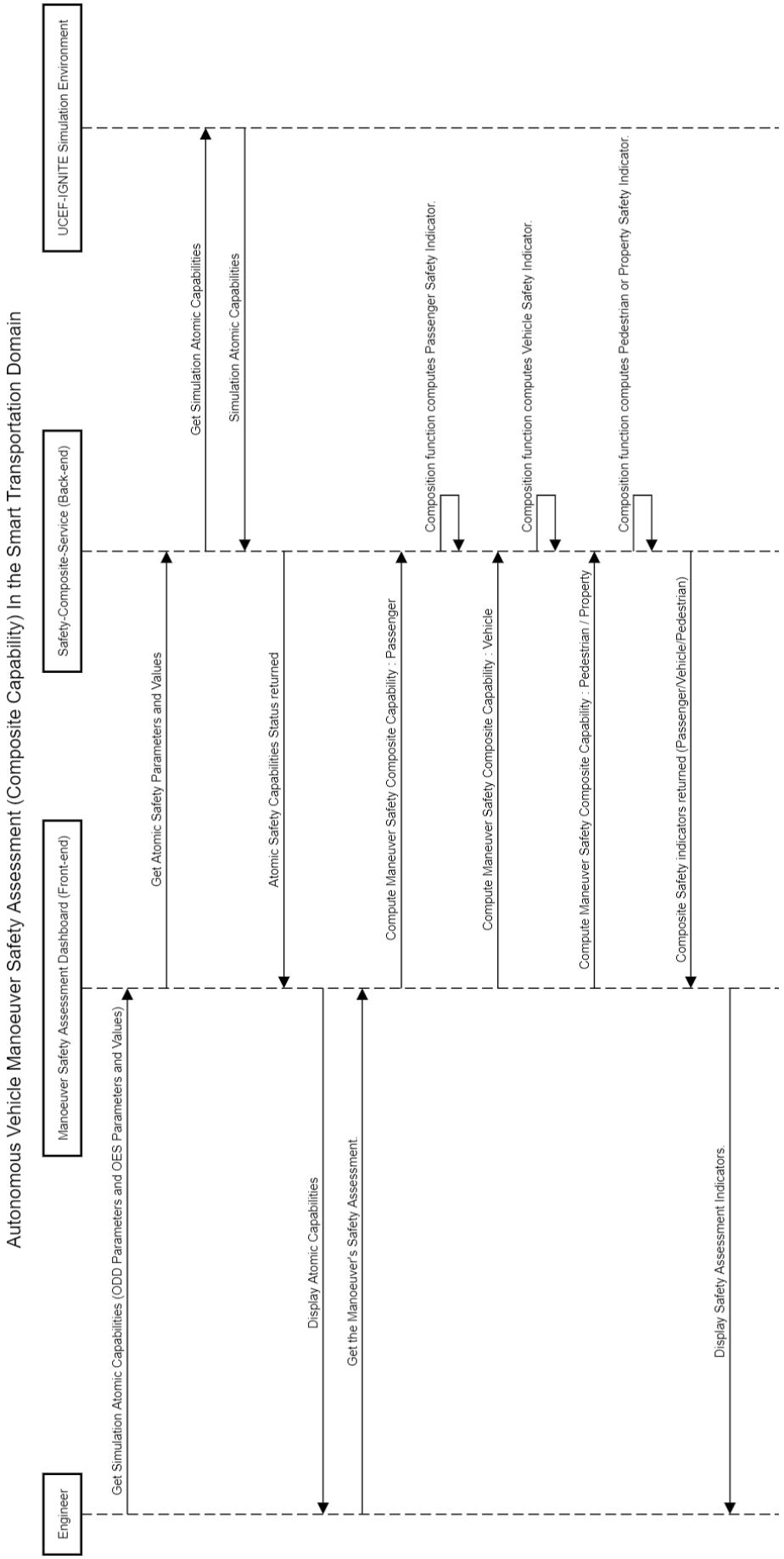

FIGURE 6.10: Sequence diagram for the safety assessment capability expressed using mPlane composition interactions.

safety.tla | Model_1

/home/khalid/Safety/safety.tla

TLA Module

```
1  ---------------------------- MODULE safety ----------------------------
2  EXTENDS Integers, TLC
3  (*--algorithm safety
4  variables speed \in (1..100), distance \in (1..100)
5
6  begin
7    SafetyRangeVerification:
8        if ((speed<60)/\(distance>50)) then
9            print <<"SAFE", "speed", speed, "distance", distance>>;
10           else
11                    print <<" NOT Safe", "speed", speed, "distance", distance>>;
12
13               end if;
14
15 end algorithm;*)
16 \* BEGIN TRANSLATION
17 VARIABLES speed, distance, pc
18
19 vars == << speed, distance, pc >>
20
21 Init == (* Global variables *)
22         /\ speed \in (1..100)
23         /\ distance \in (1..100)
24         /\ pc = "SafetyRangeVerification"
25
26 SafetyRangeVerification == /\ pc = "SafetyRangeVerification"
27                            /\ IF ((speed<60)/\(distance>50))
28                                  THEN /\ PrintT(<<"SAFE", "speed", speed, "distance", distance>>)
29                                  ELSE /\ PrintT(<<" NOT Safe", "speed", speed, "distance", distance>>)
30                            /\ pc' = "Done"
31                            /\ UNCHANGED << speed, distance >>
32
33 Next == SafetyRangeVerification
34           \/ (* Disjunct to prevent deadlock on termination *)
35              (pc = "Done" /\ UNCHANGED vars)
36
37 Spec == Init /\ [][Next]_vars
38
39 Termination == <>(pc = "Done")
40
41 \* END TRANSLATION
42 ===============================================================================
43 \* Modification History
44 \* Last modified Fri Jan 27 03:48:16 EST 2023 by khalid
45 \* Created Fri Jan 27 03:14:59 EST 2023 by khalid
```

FIGURE 6.11: PLUSCAL description translated to a TLA formal specification for the Safety use-case. Safety is a binary composite capability of two discreet atomic capabilities, speed, and distance.

obstacle, and ii) the speed at the moment of detecting the obstacle. We consider the other parameters as constant and not contributing to the output of the experiment.

Figure 6.13 shows the experimentation setting we adopted.

Figure 6.14 describes the moment T=97 when the vehicle has detected the obstacle. The time ΔT

6.2.6 Conclusion, Challenges, And Future Work

In this section, we proposed a formal specification of the safety metric during an autonomous driving maneuver.

As the composition function of atomic capabilities during an emergency braking experiment is still a work in progress, we highlighted a simple example that only considers two parameters to define whether or not the manoeuver was safe.

The model we adopted reduces the scope of parameters that influence emergency braking, but it provides a glimpse into what formal specification and verification of composite capabilities would add to this field of study. We consider our effort among the very few attempts to study safety as a composite and tangible

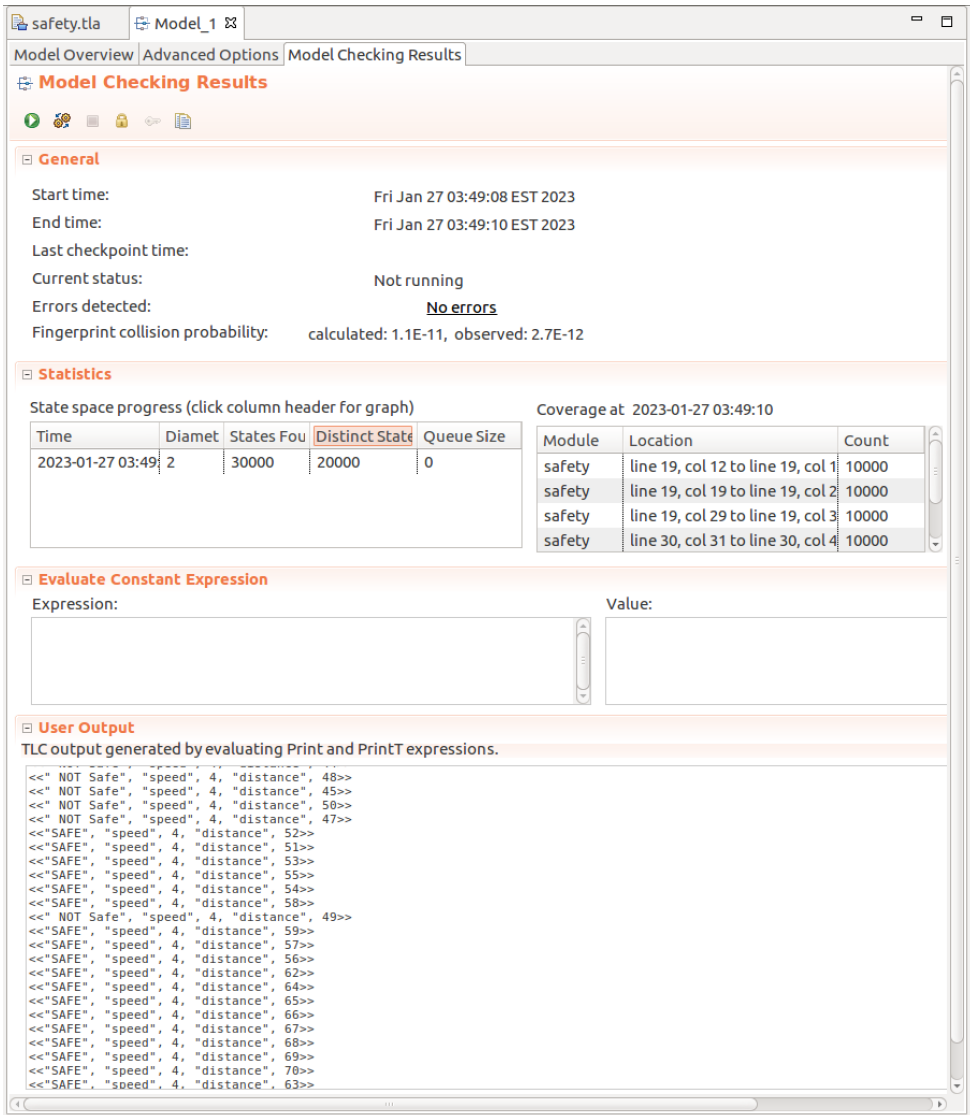

FIGURE 6.12: Symbolic execution of the safety composite capability specification with 20000: the number of possible unique outcomes, the log file shows the safety assessment/value for each pair of speed and distance values.

capability that can be measured in a specific scenario and under specific constraints.

For future work, we will continue improving our model to take into consideration all the possible properties impacting the emergency braking experiment while making sure we test a large variety of values for each parameter and trying to overcome the state space explosion problem that can occur as a byproduct of extending the state space of parameters and values, and which can be mitigated or reduced by connecting our formal verification machines to extra EC2 instances or by optimizing further our composite capability function.

Another domain of contribution we are trying to contribute to is the adoption of a color-coded scheme for representing safety as a composite capability in the smart transportation domain. The choice for color-coded schemes subscribes to the NIST CPS Framework guidelines that encourage building IoT Composition Platforms

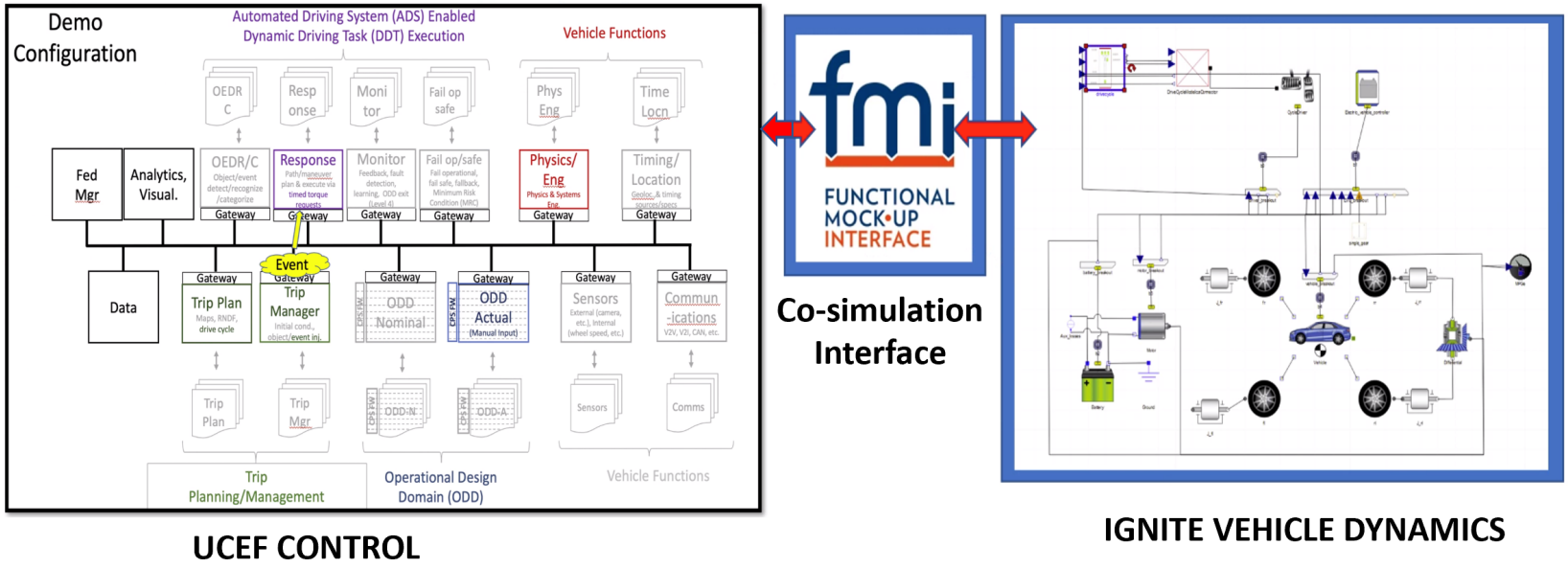

FIGURE 6.13: Emergency braking experiment simulation environment.

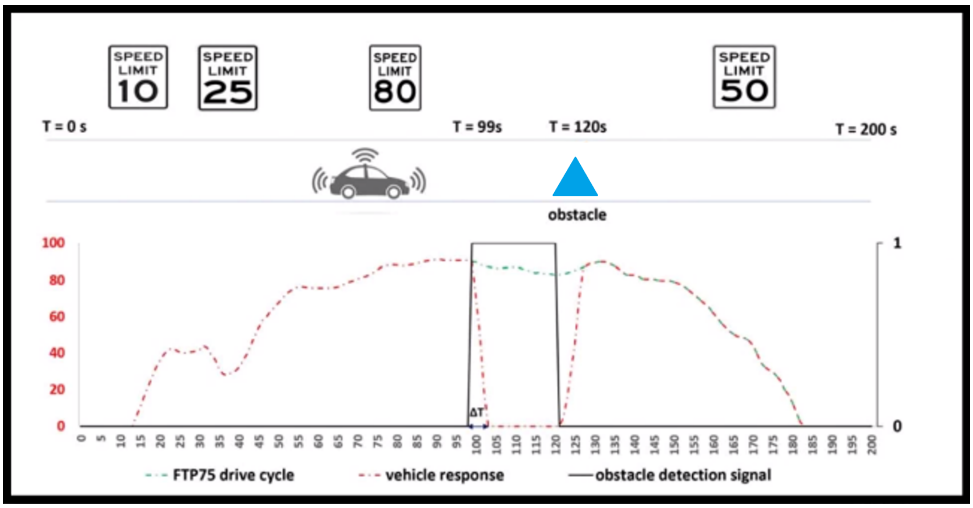

FIGURE 6.14: At T=99, the vehicle detected the obstacle, we vary vehicle speed at the moment of detection and distance to the obstacle to assessing safety

that provide value-added services with assessments that are user-friendly, and color-coded schemes for assessing non-tangible capabilities are one way of ensuring this goal is met.

Once the full state space of parameters that influence safety in a smart transportation experiment is taken into consideration, and composition functions that assess safety based on this state space are identified and verified, Figure 6.15 provides a glimpse for a user-friendly color-coded scheme that can be leveraged to assess safety in the smart transportation domain.

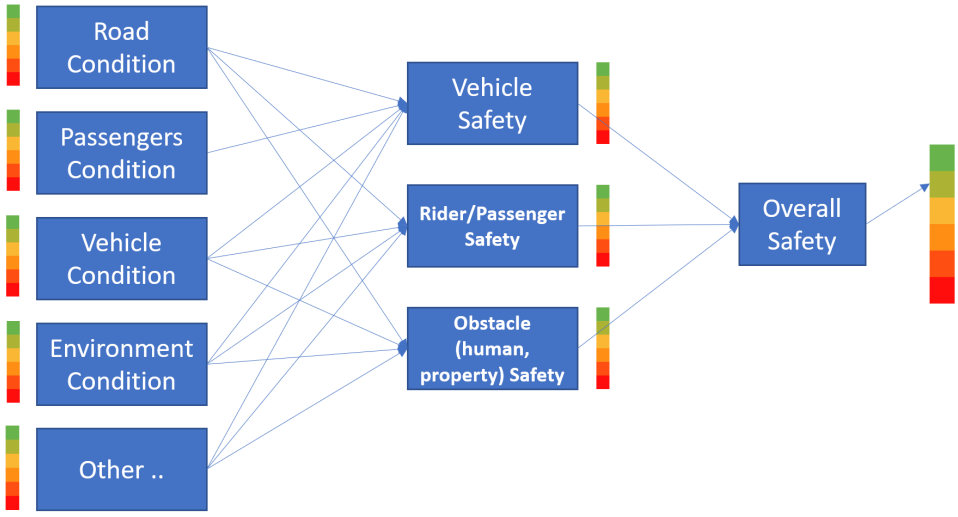

FIGURE 6.15: Color Coded scheme for assessing the safety of a maneuver in the smart transportation domain.

6.3 Ongoing work: Health improvement as a Composite Capability in the Smart Health Domain.

After we discussed well-being as a composite capability in the smart building domain and safety as a composite capability in the smart transportation domain, we discuss in this section an ongoing work relative to service composition in the smart health domain: the assessment of health improvement for a patient in a smart ICU, we examine the example of a patient suffering from respiratory disease (e.g., asthma, chronic obstructive pulmonary disease (COPD), pulmonary fibrosis, pneumonia, and lung cancer, etc.) and connected to a ventilator with autonomy functions, as part of a smart Intensive Care Unit (ICU) setting. Figure 6.16 explains the context of the study:

The interactions between entities contributing to composing the health indicator are showcased in the sequence diagram in Fig. 6.17, which highlights a potential implementation using the ICCF-based IoTCaP platform:

The "Actor", which is, in this case, a Medical Practitioner, authenticates to its profile in IoTCaP and then selects a patient using a patient ID. The back end of IoTCaP connects to individual atomic capabilities and implements a composition function that can also return a health indicator.

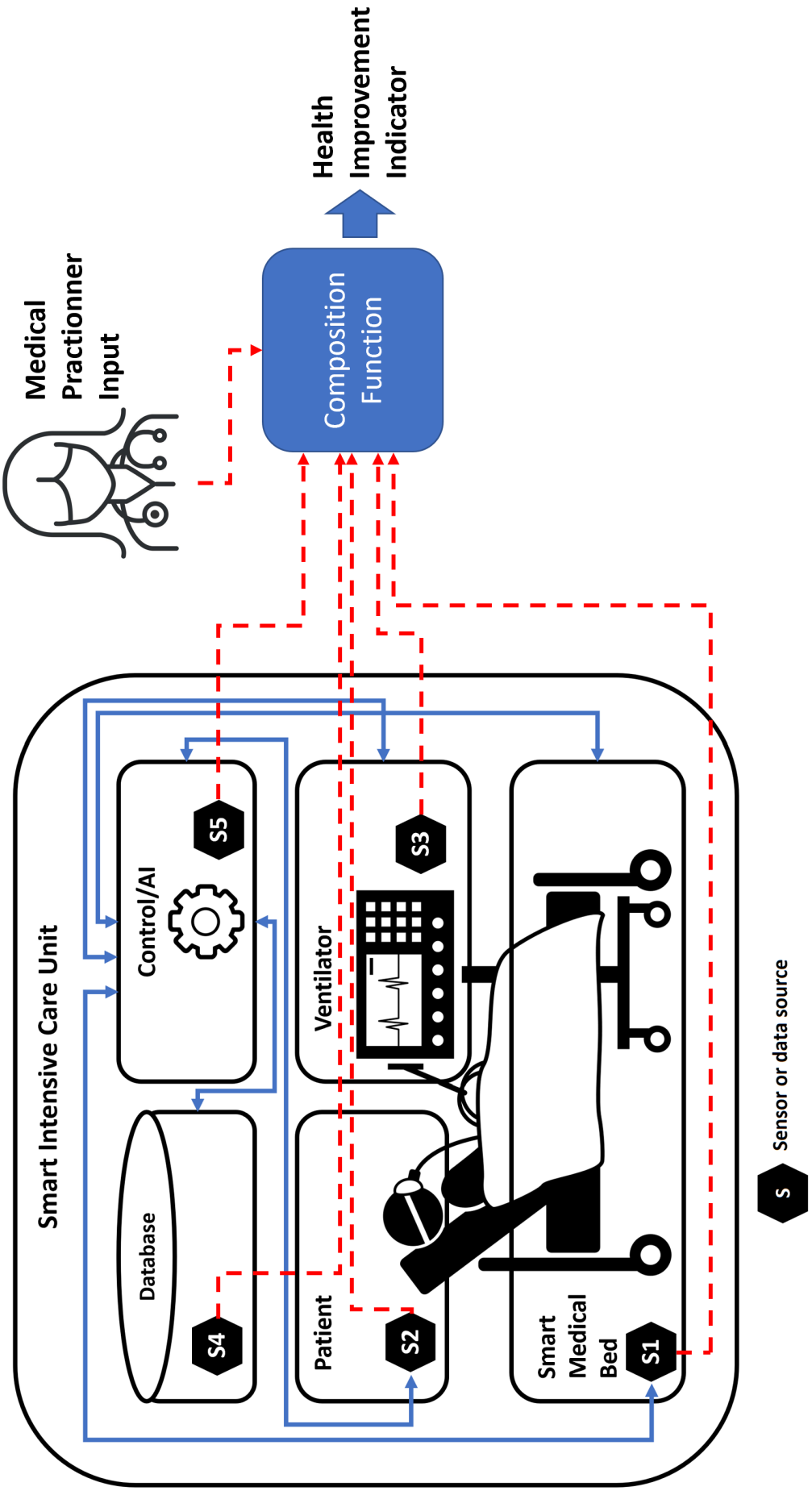

FIGURE 6.16: A smart ICU testbed, with multiple data sources or sensors

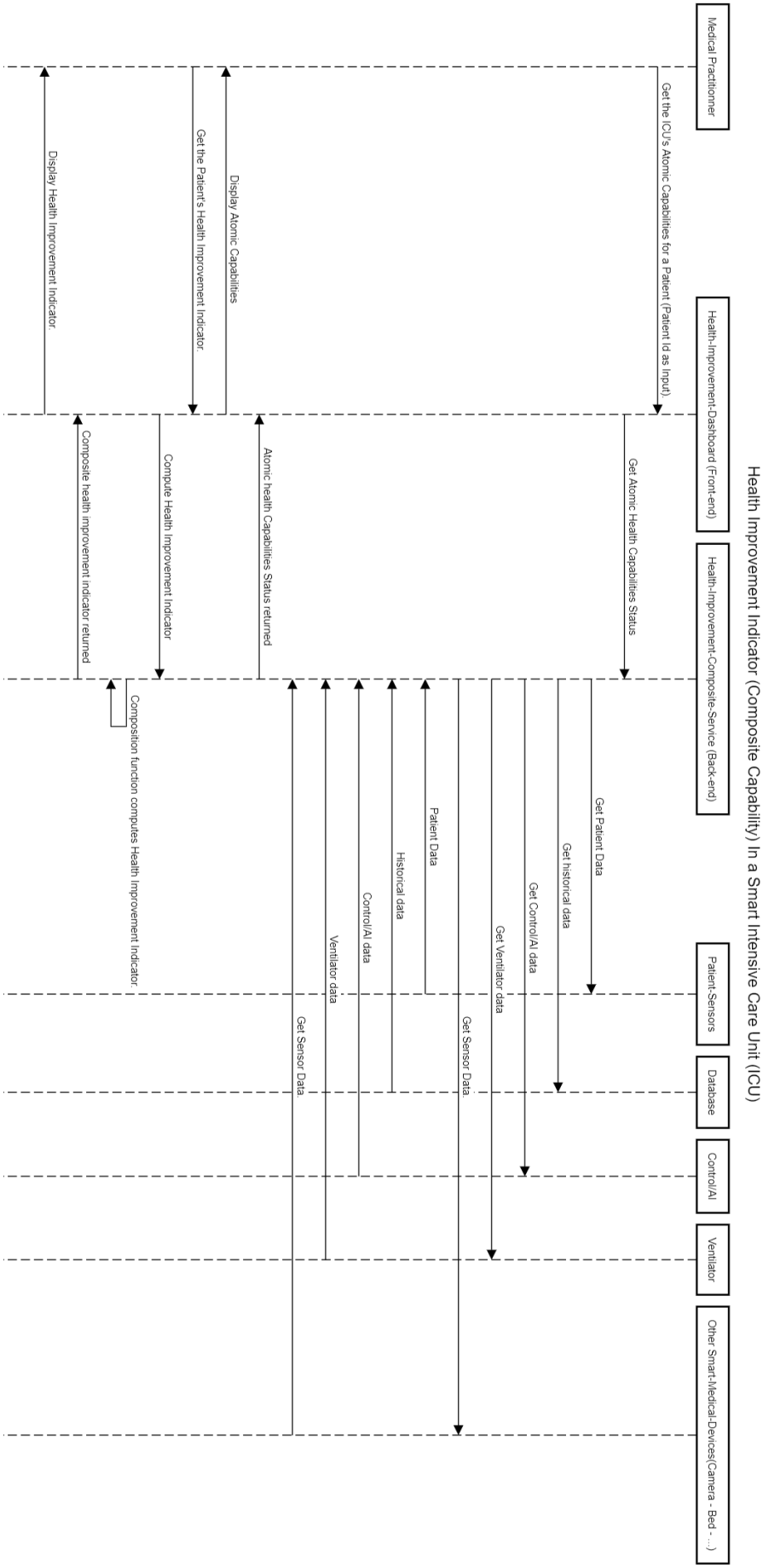

FIGURE 6.17: Sequence diagram for the Health improvement capability expressed using mPlane interactions.

Chapter 7

Conclusion

7.1 Summary of the work

In this work, we started by explaining in chapter 1 the context that motivated this study: smart cities offering a tremendous number of data sources via IoT sensors and devices and the potential this creates in innovating new services that can satisfy multiple stakeholders' needs.

In chapter 2, we explained the concepts related to service composition in IoT and CPS through a layered model, and we explained service composition actors and operations.

In chapter 3, we discussed related work relative to:

i) Systematic Literature Reviews:

The goal is to identify gaps in the literature to be filled through a systematic literature review that we performed in chapter 4, section 1.

ii) Service Composition Platforms and Frameworks:

The goal is to identify gaps and weaknesses in existing service composition frameworks and platforms with the objective of defining a new service composition framework that we propose and discuss in chapter 5.

In chapter 4, we performed a systematic literature review (SLR) to provide answers to fifteen formal, technical, and QoS questions. The identified questions represent gaps in previous systematic literature reviews that weren't addressed, as we concluded from chapter 4, section 1. The discussions and answers provided for the different questions would benefit different stakeholders, including researchers willing to propose new frameworks or end users that need to learn about IoT service composition frameworks. The knowledge gathered from the SLR would also help in implementing best practices when it comes to proposing or improving future iterations of composition frameworks such as the ICCF composition framework we propose.

In chapter 5, we used knowledge gathered from chapters 3 and 4, and we compared existing composition environments and frameworks: corresponding gaps in formal and knowledge foundations were identified, and a new IoT and

CPS composition framework were proposed to fill those gaps. We call the newly proposed framework ICCF. Its foundations are based on the NIST CPS Framework, providing guidelines for respecting different stakeholders' concerns and pointing out characteristics to be addressed relative to service composition. ICCF uses mPlane for semantics as mPlane provides composition-friendly jargon that covers composed capabilities, composition operations, and composition engines. mPlane is also measurability friendly as it was originally conceived to assess network performance KPIs such as the RTT. Finally, TLA+, a formal verification toolkit, was proposed as it offers a bridge (PLUSCAL) between human-readable composition semantics described in the mPlane semantics and the TLA specification, which is used to formally verify the composition model and test its invariants. A simple well-being example was provided to give a glimpse of what the semantics of a composite capability would look like under ICCF, and a formal specification model was executed to model check the proposed composite metric.

In chapter 6, three examples of the ICCF framework were highlighted:

→ Well-being as a composite capability in a smart building: which leverages five atomic capabilities (humidity, temperature, noise level, wifi signal strength, and air quality). An implementation in Vert.X using real sensors was also provided, which shows how the composite metric can satisfy multiple stakeholders in different categories of smart buildings (residential, factory, hospital, ...).

→ Safety as a composite capability in the smart transportation domain: Based on research we have done associated with the study of autonomous vehicles functions such as emergency braking [121][161], we are working on a composition formula for safety that takes as an input component of the simulation environment and runs an experiment that yields results reflecting whether or not the manoeuver was safely executed. Levels of safety are defined as either a spectrum or as a binary function. The measurability aspect of this effort will use elements of the CVSS vulnerability measurement framework.

→ Health improvement as a composite capability in the smart medical domain: this is an ongoing collaborative research effort between researchers and scientists at the Johns Hopkins Computer Science School and the Johns Hopkins Medical school. The effort aims to introduce an autonomous ventilator with trustworthiness and assuredness capabilities. The simulation environment is based on Biogears [34], with plans to co-simulate it with MATLAB models and hardware in the loop. The simulation environment allows the simulation of multiple actors and situations, including patients, medical devices, health preconditions, and real-time insults. The focus of the simulation is on respiratory systems, and data generated by the simulator is compared with real-world data -from patients suffering from respiratory health conditions- for data validation purposes. A health improvement composite capability based on the ICCF Framework is under development which will take into consideration data input from sensors attached to patients, data collected from medical devices, and also input from experienced medical staff and practitioners to determine whether a patient's health has improved. The health improvement capability would also serve as an indicator or predictor for the hospital's bed

occupancy.

Chapter 7 concludes this effort and provides highlights of ongoing and future work.

7.2 Ongoing and future work

In this work, we proposed a novel IoT and CPS composition framework called ICCF. We focused on the knowledge and formal aspects of the framework as the main aim is to provide researchers and interested parties with formal and knowledge tools to prototype novel capabilities while respecting certain foundations for a trustworthy and robust implementation. In the future iterations of ICCF, we define Technical and QoS components, and we discuss those components under the same applications we proposed in chapter 6.

We will also continue to develop and improve the composition formulas for the three domains of interest in a way that best reflects their meaning to a large extent and equip the composition formulas with means that enable customization to address different user views as composite capabilities such as well-being tend to mean different things for different stakeholders; ensuring certain flexibility in the composition formula is an important aspect that we will take into account.

Formally verifying compositions is another item of interest for future work as we intend to include other properties to verify -in addition to invariants checking- including correctness, safety, and reliability.

Bibliography

[1] Tatsuya Abe. "A verifier of directed acyclic graphs for model checking with memory consistency models". In: *Haifa Verification Conference*. Springer. 2017, pp. 51–66.

[2] N Adadi et al. "Proposition of web services composition approach basing of model-driven approach and multi-agent systems". In: *International Journal of Computer Modelling & New Technologies* (2017).

[3] Saurabh Agarwal and Koshel Agarwal. "Pi-calculus based formal verification of web services composition". In: *International Journal of Grid and Distributed Computing* 8.5 (2015), pp. 137–140.

[4] Saurabh Agarwal, N Tiwari, and M Dubey. "Checking of Web Services Composition through Online Movie Ticket Booking Example". In: *International Journal of Computer Science Research* (2017), pp. 1–5.

[5] Shabir Ahmad and DoHyeun Kim. "Design and implementation of thermal comfort system based on tasks allocation mechanism in smart homes". In: *Sustainability* 11.20 (2019), p. 5849.

[6] Krishna Ajay. *Advanced iot compositions in webthings gateway*. 2020. URL: https://github.com/ajaykrishna/mozart.

[7] Krishna Ajay. *Compose and build iot applications using lnt models*. 2019. URL: https://github.com/ajaykrishna/iotcomposer.

[8] Krishna Ajay. *Iot composer*. 2019. URL: https://www.github.com/ajaykrishna/iotcomposer.git.

[9] Charilaos Akasiadis et al. "Developing complex services in an IoT ecosystem". In: *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. IEEE. 2015, pp. 52–56.

[10] Ismaeel Al Ridhawi et al. "A blockchain-based decentralized composition solution for IoT services". In: *Icc 2020-2020 ieee international conference on communications (icc)*. IEEE. 2020, pp. 1–6.

[11] Sarfraz Alam, Mohammad MR Chowdhury, and Josef Noll. "Senaas: An event-driven sensor virtualization approach for internet of things cloud". In: *2010 IEEE International Conference on Networked Embedded Systems for Enterprise Applications*. IEEE. 2010, pp. 1–6.

[12] Tamleek Ali, Mohammad Nauman, and Masoom Alam. "An accessible formal specification of the UML and OCL meta-model in Isabelle/HOL". In: *2007 IEEE International Multitopic Conference*. IEEE. 2007, pp. 1–6.

[13] Osama Almurshed et al. "A fault tolerant workflow composition and deployment automation IoT framework in a multi cloud edge environment". In: *IEEE Internet Computing* (2021).

[14] Osama Alsaryrah, Ibrahim Mashal, and Tein-Yaw Chung. "A fast iot service composition scheme for energy efficient qos services". In: *Proceedings of the 2019 7th International Conference on Computer and Communications Management*. 2019, pp. 231–237.

[15] Badraddin Alturki et al. "Exploring the effectiveness of service decomposition in fog computing architecture for the Internet of Things". In: *IEEE Transactions on Sustainable Computing* (2019).

[16] Khaled Alwasel et al. "IoTSim-Osmosis: A framework for modeling and simulating IoT applications over an edge-cloud continuum". In: *Journal of Systems Architecture* 116 (2021), p. 101956.

[17] Bacciu Andrea. *An sml library for calculus of communicating systems*. 2019. URL: https://github.com/andreabac3/Calculus-of-Communicating-Systems.

[18] Bhattacharjee Anirban. *Automated designing and provisioning platform for cloud applications*. 2022. URL: https://github.com/doc-vu/DeploymentAutomation.

[19] Aleksandar Antonić, Martina Marjanović, and Ivana Podnar Žarko. "Modeling aggregate input load of interoperable smart city services". In: *Proceedings of the 11th ACM International Conference on Distributed and Event-Based Systems*. 2017, pp. 34–43.

[20] Iannopollo Antonio. *Pyco ltl composition*. 2017. URL: https://github.com/fmohr/ML-Plan.

[21] Idir Aoudia et al. "Service composition approaches for internet of things: a review". In: *International Journal of Communication Networks and Distributed Systems* 23.2 (2019), pp. 194–230.

[22] Damian Arellanes and Kung-Kiu Lau. "Algebraic service composition for user-centric IoT applications". In: *International Conference on Internet of Things*. Springer. 2018, pp. 56–69.

[23] Damian Arellanes and Kung-Kiu Lau. "D-XMAN: a platform for total compositionality in service-oriented architectures". In: *2017 IEEE 7th International Symposium on Cloud and Service Computing (SC2)*. IEEE. 2017, pp. 283–286.

[24] Damian Arellanes and Kung-Kiu Lau. "Decentralized data flows in algebraic service compositions for the scalability of IoT systems". In: *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*. IEEE. 2019, pp. 668–673.

[25] Damian Arellanes and Kung-Kiu Lau. "Evaluating IoT service composition mechanisms for the scalability of IoT systems". In: *Future Generation Computer Systems* 108 (2020), pp. 827–848.

[26] Parvaneh Asghari, Amir Masoud Rahmani, and Hamid Haj Seyyed Javadi. "A medical monitoring scheme and health-medical service composition model in cloud-based IoT platform". In: *Transactions on Emerging Telecommunications Technologies* 30.6 (2019), e3637.

[27] Parvaneh Asghari, Amir Masoud Rahmani, and Hamid Haj Seyyed Javadi. "Privacy-aware cloud service composition based on QoS optimization in Internet of Things". In: *Journal of Ambient Intelligence and Humanized Computing* (2020), pp. 1–26.

[28] Parvaneh Asghari, Amir Masoud Rahmani, and Hamid Haj Seyyed Javadi. "Service composition approaches in IoT: A systematic review". In: *Journal of Network and Computer Applications* 120 (2018), pp. 61–77.

[29] Tatiana V Atanasova, Stoyan A Poryazov, and Emiliya T Saranova. "Problems with quality enabling of information functions composition in smart buildings". In: *2016 24th Telecommunications Forum (TELFOR)*. IEEE. 2016, pp. 1–4.

[30] Luigi Atzori, Antonio Iera, and Giacomo Morabito. "The internet of things: A survey". In: *Computer networks* 54.15 (2010), pp. 2787–2805.

[31] Claudio Badii et al. "Smart city IoT platform respecting GDPR privacy and security aspects". In: *IEEE Access* 8 (2020), pp. 23601–23623.

[32] Kyungmin Bae et al. "Verifying hierarchical Ptolemy II discrete-event models using Real-Time Maude". In: *Science of Computer Programming* 77.12 (2012), pp. 1235–1271.

[33] KyeongDeok Baek and In-Young Ko. "Spatio-cohesive service selection using machine learning in dynamic IoT environments". In: *International Conference on Web Engineering*. Springer. 2018, pp. 366–374.

[34] Austin Baird et al. "BioGears: A C++ library for whole body physiology simulations". In: *Journal of Open Source Software* 5.56 (2020), p. 2645.

[35] Thar Baker et al. "An energy-aware service composition algorithm for multiple cloud-based IoT applications". In: *Journal of Network and Computer Applications* 89 (2017), pp. 96–108.

[36] Bharathan Balaji et al. "Brick: Metadata schema for portable smart building applications". In: *Applied energy* 226 (2018), pp. 1273–1292.

[37] Armin Balalaie et al. "Microservices migration patterns". In: *Software: Practice and Experience* 48.11 (2018), pp. 2019–2042.

[38] Payam Barnaghia et al. "Sense2web: A linked data platform for semantic sensor networks". In: *Semantic Web-Interoperability, Usability, Applicability an IOS Press Journal* 2.1 (2011), pp. 1–11.

[39] Muhammad Rizwan Bashir and Asif Qumer Gill. "Towards an IoT big data analytics framework: smart buildings systems". In: *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE. 2016, pp. 1325–1332.

[40] Alessandro Bassi et al. *Enabling things to talk*. Springer Nature, 2013.

[41] Eugeny I Batov. "The distinctive features of "smart" buildings". In: *Procedia Engineering* 111 (2015), pp. 103–107.

[42] Aissam Belghiat and Allaoua Chaoui. "A Pi-calculus-based approach for the verification of UML2 sequence diagrams". In: *2015 10th International Joint Conference on Software Technologies (ICSOFT)*. Vol. 2. IEEE. 2015, pp. 1–8.

[43] Paolo Bellagente et al. "Adopting IoT framework for Energy Management of Smart Building: A real test-case". In: *2015 IEEE 1st International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*. IEEE. 2015, pp. 138–143.

[44] Anirban Bhattacharjee et al. "(wip) cloudcamp: Automating the deployment and management of cloud services". In: *2018 IEEE International Conference on Services Computing (SCC)*. IEEE. 2018, pp. 237–240.

[45] Seyed Arshia Hosseini Bidi, Zeinab Movahedi, and Zahra Movahedi. "Qoe-aware service composition in smart cities using restful iot". In: *Electrical Engineering (ICEE), Iranian Conference on*. IEEE. 2018, pp. 1559–1564.

[46] Michael Blackstock and Rodger Lea. "IoT mashups with the WoTKit". In: *2012 3rd IEEE International Conference on the Internet of Things*. IEEE. 2012, pp. 159–166.

[47] Bruno Blanchet et al. "ProVerif 2.00: automatic cryptographic protocol verifier, user manual and tutorial". In: *Version from* (2018), pp. 05–16.

[48] Iovka Boneva, Jose E Labra Gayo, and Eric G Prud'hommeaux. "Semantics and validation of shapes schemas for RDF". In: *International Semantic Web Conference*. Springer. 2017, pp. 104–120.

[49] Pearl Brereton et al. "Lessons from applying the systematic literature review process within the software engineering domain". In: *Journal of systems and software* 80.4 (2007), pp. 571–583.

[50] Manfred Broy. "Engineering cyber-physical systems: Challenges and foundations". In: *Complex Systems Design & Management: Proceedings of the Third International Conference on Complex Systems Design & Management CSD&M 2012*. Springer. 2013, pp. 1–13.

[51] Mihal Brumbulli, Emmanuel Gaudin, and Ciprian Teodorov. "Automatic Verification of BPMN Models". In: *10th European Congress on Embedded Real Time Software and Systems (ERTS 2020)*. 2020.

[52] Martin Burns et al. "Universal CPS Environment for Federation (UCEF)". In: *2018 Winter Simulation Innovation Workshop*. 2018.

[53] Davide Calvaresi et al. "The challenge of real-time multi-agent systems for enabling IoT and CPS". In: *Proceedings of the international conference on web intelligence*. 2017, pp. 356–364.

[54] University of Cambridge. *isabelle installation guide*. 2022. URL: https://isabelle.in.tum.de/installation.html.

[55] Mike Cantelon et al. *Node. js in Action*. Manning Greenwich, 2014.

[56] Nathalie Cauchi et al. "Efficient probabilistic model checking of smart building maintenance using fault maintenance trees". In: *Proceedings of the 4th ACM International Conference on Systems for Energy-Efficient Built Environments*. 2017, pp. 1–10.

[57] Z Berkay Celik, Patrick McDaniel, and Gang Tan. "Soteria: Automated {IoT} Safety and Security Analysis". In: *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. 2018, pp. 147–158.

[58] Zheng-yi Chai, Meng-meng Du, and Guo-zhi Song. "A fast energy-centered and QoS-aware service composition approach for Internet of Things". In: *Applied Soft Computing* 100 (2021), p. 106914.

[59] David Champelovier et al. *Reference Manual of the LNT to LOTOS Translator*. 2018.

[60] Victor Chang and Muthu Ramachandran. "Towards achieving data security with the cloud computing adoption framework". In: *IEEE Transactions on services computing* 9.1 (2015), pp. 138–151.

[61] Mahmoudi Charif. *UPPAAL model for Mobile Cloud (Cloudlets) verification*. 2017. URL: https://github.com/charifmahmoudi/CloudletModel.

[62] Feng Chen et al. "Modeling cross-organizational services composition with pi-calculus". In: *Proceedings of 2011 IEEE International Conference on Service Operations, Logistics and Informatics*. IEEE. 2011, pp. 51–56.

[63] Bin Cheng et al. "FogFlow: Easy programming of IoT services over cloud and edges for smart cities". In: *IEEE Internet of Things journal* 5.2 (2017), pp. 696–707.

[64] Florent Chevrou, Aurélie Hurault, and Philippe Quéinnec. "Automated verification of asynchronous communicating systems with TLA+". In: *Electronic Communications of the EASST* 72 (2015), pp–1.

[65] Samir Chouali et al. "Formal Verification and Performance Analysis of a New Data Exchange Protocol for Connected Vehicles". In: *IEEE Transactions on Vehicular Technology* 69.12 (2020), pp. 15385–15397.

[66] Marcus A Christie et al. "Using keycloak for gateway authentication and authorization". In: (2017).

[67] Flavio Cirillo et al. "Atomic Services: sustainable ecosystem of smart city services through pan-European collaboration". In: *2019 Global IoT Summit (GIoTS)*. IEEE. 2019, pp. 1–7.

[68] Flavio Cirillo et al. "Smart city IoT services creation through large-scale collaboration". In: *IEEE Internet of Things Journal* 7.6 (2020), pp. 5267–5275.

[69] Cisco. *Cisco IoT Prediction for 2030. A technical Report*. https://www.cisco.com/c/en/us/products/collateral/se/internet-of-things/at-a-glance-c45-731471.pdf.

[70] Edmund M Clarke et al. "Model checking and the state explosion problem". In: *LASER Summer School on Software Engineering*. Springer. 2011, pp. 1–30.

[71] Nimble Code. *An Efficient Logic Model Checker for the Verification of Multithreaded Code*. 2022. URL: https://github.com/nimble-code/Spin.

[72] Zhu Da et al. "Future service provision: Towards a flexible hybrid service supporting platform". In: *2010 IEEE Asia-Pacific Services Computing Conference*. IEEE. 2010, pp. 226–233.

[73] Fei Dai et al. "A choreography analysis approach for microservice composition in cyber-physical-social systems". In: *IEEE Access* 8 (2020), pp. 53215–53222.

[74] Loris Dal Lago et al. "Dependability assessment of SOA-based CPS with contracts and model-based fault injection". In: *IEEE Transactions on Industrial Informatics* 14.1 (2017), pp. 360–369.

[75] Arellanes Damian. *Dx-man platform for building iot systems using algebraic service composition*. 2019. URL: https://github.com/damianarellanes/dxman.

[76] Hershcovich Daniel. *modelling composition for directed acyclic graphs in python*. 2015. URL: https://github.com/ninjin/nerv.

[77] Santanu Das. "Technology for Smart Home". In: *Proceedings of International Conference on VLSI, Communication, Advanced Devices, Signals & Systems and Networking (VCASAN-2013)*. Springer. 2013, pp. 7–12.

[78] Kerkkamp David. *Learning State Representations for Formal Verification*. 2020. URL: https://github.com/davidkerkkamp/representation-learning.

[79] SSBG De Rosa and SSBS Pentassuglia. "Specification of mPlane Access Control and Data Protection Mechanisms". In: ().

[80] Mofleh Al-Diabat et al. "Automatic Verification of Communicative Commitments using Reduction". In: ().

[81] Bruno Donassolo et al. "Fog based framework for IoT service provisioning". In: *2019 16th IEEE annual consumer communications & networking conference (CCNC)*. IEEE. 2019, pp. 1–6.

[82] Wanchun Dou et al. "HireSome-II: Towards privacy-aware cross-cloud service composition for big data applications". In: *IEEE Transactions on Parallel and Distributed Systems* 26.2 (2013), pp. 455–466.

[83] Yanhua Du, Wei Tan, and MengChu Zhou. "Timed compatibility analysis of web service composition: A modular approach based on Petri nets". In: *IEEE Transactions on Automation Science and Engineering* 11.2 (2013), pp. 594–606.

[84] Francisco Durán, Gwen Salaün, and Ajay Krishna. "Automated composition, analysis and deployment of IoT applications". In: *International Conference on Objects, Components, Models and Patterns*. Springer. 2019, pp. 252–268.

[85] Rasool Esmaeilyfard and Mahshid Naderi. "Distributed composition of complex event services in IoT network". In: *The Journal of Supercomputing* 77.6 (2021), pp. 6123–6144.

[86] Invernizzi Fabrizio. *mplane node js implementation*. 2015. URL: https://github.com/finvernizzi/mplane.

[87] Bob Familiar. *Microservices, IoT, and Azure*. Springer, 2015.

[88] Yong-Yi Fanjiang et al. "Genetic algorithm for QoS-aware dynamic web services composition". In: *2010 International Conference on Machine Learning and Cybernetics*. Vol. 6. IEEE. 2010, pp. 3246–3251.

[89] Elie Fares, Jean-Paul Bodeveix, and Mamoun Filali. "Event algebra for transition systems composition-application to timed automata". In: *2013 20th International Symposium on Temporal Representation and Reasoning*. IEEE. 2013, pp. 125–132.

[90] Mahdi Farnaghi and Ali Mansourian. "Automatic composition of WSMO based geospatial semantic web services using artificial intelligence planning". In: *Journal of Spatial Science* 58.2 (2013), pp. 235–250.

[91] Pedro Fazenda, Paulo Carreira, and Pedro Lima. "Context-based reasoning in smart buildings". In: *Proceedings of the first international workshop on information technology for energy applications*. Vol. 923. 2012, pp. 131–142.

[92] Mohr Felix. *ML plan*. 2020. URL: https://github.com/fmohr/ML-Plan.

[93] FIWARE. *A PLUGIN FOR COMPOSING ENTITIES FROM OTHER ENTITIES*. 2022. URL: https://fiware-iot-broker.readthedocs.io/en/latest/entitycomposition/.

[94] FIWARE. *Installing Fiware IoT Broker*. 2022. URL: https://fiware-iotagent-ul.readthedocs.io/en/1.10.0/installationguide/.

[95] fp7mplane. *mplane python3 sdk*. 2015. URL: https://github.com/fp7mplane/protocol-ri.

[96] Laura Fraade-Blanar et al. *Measuring automated vehicle safety: Forging a framework*. 2018.

[97] Nijhof Franck et al. *Home assistant add-ons*. 2022. URL: https://github.com/hassio-addons/hassio-vagrant.

[98] Patricia Franco et al. "A Framework for IoT Based Appliance Recognition in Smart Homes". In: *IEEE Access* 9 (2021), pp. 133940–133960. DOI: 10.1109/ACCESS.2021.3116148.

[99] Michael Friedewald et al. "The brave new world of ambient intelligence: An analysis of scenarios regarding privacy, identity and security issues". In: *International Conference on Security in Pervasive Computing*. Springer. 2006, pp. 119–133.

[100] Jeff Friesen. *Java XML and JSON*. Springer, 2016.

[101] Hubert Garavel et al. "CADP 2011: a toolbox for the construction and analysis of distributed processes". In: *International Journal on Software Tools for Technology Transfer* 15.2 (2013), pp. 89–107.

[102] Oscar Garcıéa et al. "A serious game to reduce consumption in smart buildings". In: *International Conference on Practical Applications of Agents and Multi-Agent Systems*. Springer. 2017, pp. 481–493.

[103] Alicia Garcıéa-Holgado, Samuel Marcos-Pablos, and Francisco Garcıéa-Peñalvo. "Guidelines for performing systematic research projects reviews". In: (2020).

[104] Arthur Gatouillat and Youakim Badr. "Verifiable and resource-aware component model for IoT devices". In: *Proceedings of the 9th International Conference on Management of Digital EcoSystems*. 2017, pp. 235–242.

[105] Noreddine Gherabi and Mohamed Bahaj. "A new method for mapping UML class into OWL ontology". In: *Spec. Issue Int. J. Comput. Appl.(0975 8887) Soft. Eng. Databases Expert Syst. SEDEXS* (2012), pp. 5–9.

[106] Mostafa Ghobaei-Arani and Alireza Souri. "LP-WSC: a linear programming approach for web service composition in geographically distributed cloud environments". In: *The Journal of Supercomputing* 75.5 (2019), pp. 2603–2628.

[107] Márcio Miguel Gomes, Rodrigo da Rosa Righi, and Cristiano André da Costa. "Future directions for providing better IoT infrastructure". In: *Proceedings of the 2014 ACM international joint conference on pervasive and ubiquitous computing: Adjunct Publication*. 2014, pp. 51–54.

[108] Deeksha Gorige et al. "Privacy-risk detection in microservices composition using distributed tracing". In: *2020 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE)*. IEEE. 2020, pp. 250–253.

[109] Paul Grace et al. "Taming the interoperability challenges of complex iot systems". In: *Proceedings of the 1st ACM Workshop on Middleware for Context-Aware Applications in the IoT*. 2014, pp. 1–6.

[110] Christopher Greer et al. *Cyber-physical systems and internet of things*. 2019.

[111] Edward Griffor, David Wollman, and Christopher Greer. "Automated Driving System Safety Measurement Part I: Operating Envelope Specification". In: *NIST Special Publication* 1900 (2021), p. 301.

[112] Edward Griffor et al. *Framework for Cyber-Physical Systems: Volume 1, Overview*. en. June 2017. DOI: https://doi.org/10.6028/NIST.SP.1500-201.

[113] Edward Griffor et al. *Framework for Cyber-Physical Systems: Volume 2, Working Group Reports*. en. June 2017. DOI: https://doi.org/10.6028/NIST.SP.1500-202.

[114] Edward R Griffor et al. "Framework for Cyber-Physical Systems: Volume 1, Overview, Version 1.0". In: *NIST Special Publication* (2017), pp. 1500–201.

[115] Heerko Groefsema, Nick RTP van Beest, and Marco Aiello. "A formal model for compliance verification of service compositions". In: *IEEE Transactions on Services Computing* 11.3 (2016), pp. 466–479.

[116] Dominique Guinard et al. "From the internet of things to the web of things: Resource-oriented architecture and best practices". In: *Architecting the Internet of things*. Springer, 2011, pp. 97–129.

[117] Dominique Guinard et al. "Towards physical mashups in the web of things". In: *2009 Sixth International Conference on Networked Sensing Systems (INSS)*. IEEE. 2009, pp. 1–4.

[118] J Octavio Gutiérrez-Garcıéa, Félix F Ramos-Corchado, and Jean-Luc Koning. "Obligation-based agent conversations for semantic web service composition". In: *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*. Vol. 1. IEEE. 2009, pp. 411–417.

[119] Quang Phuc Ha, Santanu Metia, and Manh Duong Phung. "Sensing data fusion for enhanced indoor air quality monitoring". In: *IEEE Sensors Journal* 20.8 (2020), pp. 4430–4441.

[120] Khalid Halba et al. "A Framework for the Composition of IoT and CPS Capabilities". In: *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE. 2021, pp. 1265–1272.

[121] Khalid Halba et al. "Using statistical methods and co-simulation to evaluate ads-equipped vehicle trustworthiness". In: *2019 Electric Vehicles International Conference (EV)*. IEEE. 2019, pp. 1–5.

[122] Son N Han et al. "Service composition for IP smart object using realtime Web protocols: Concept and research challenges". In: *Computer Standards & Interfaces* 43 (2016), pp. 79–90.

[123] Markus Happe et al. "On-the-fly computing: A novel paradigm for individualized it services". In: *16th IEEE International Symposium on Object/component/service-oriented Real-time distributed Computing (ISORC 2013)*. IEEE. 2013, pp. 1–10.

[124] George Hatzivasilis, Ioannis Papaefstathiou, and Charalampos Manifavas. "SCOTRES: secure routing for IoT and CPS". In: *IEEE Internet of Things Journal* 4.6 (2017), pp. 2129–2141.

[125] George Hatzivasilis et al. "Artificial intelligence-driven composition and security validation of an internet of things ecosystem". In: *Applied Sciences* 10.14 (2020), p. 4862.

[126] Vahideh Hayyolalam et al. "Single-objective service composition methods in cloud manufacturing systems: Recent techniques, classification, and future trends". In: *Concurrency and Computation: Practice and Experience* 34.5 (2022), e6698.

[127] David R Heffelfinger. "Security with JSON Web Tokens". In: *Payara Micro Revealed: Cloud-Native Application Development with Java*. Springer, 2022, pp. 191–203.

[128] Florian Heimgaertner et al. "Scaling home automation to public buildings: A distributed multiuser setup for OpenHAB 2". In: *2017 Global Internet of Things Summit (GIoTS)*. IEEE. 2017, pp. 1–6.

[129] Katharina Hofer-Schmitz and Branka Stojanović. "Towards formal verification of IoT protocols: A Review". In: *Computer Networks* 174 (2020), p. 107233.

[130] Mehdi Hosseinzadeh et al. "A hybrid service selection and composition model for cloud-edge computing in the internet of things". In: *IEEE Access* 8 (2020), pp. 85939–85949.

[131] Kailiang Huang et al. "Indoor air quality analysis of residential buildings in northeast China based on field measurements and longtime monitoring". In: *Building and Environment* 144 (2018), pp. 171–183.

[132] Alexis Huf and Frank Siqueira. "Composition of heterogeneous web services: A systematic review". In: *Journal of Network and Computer Applications* 143 (2019), pp. 89–110.

[133] Zille Huma, Christian Gerth, and Gregor Engels. "On-The-Fly computing: automatic service discovery and composition in heterogeneous domains". In: *Computer Science-Research and Development* 30.3 (2015), pp. 333–361.

[134] San-Yih Hwang et al. "A probabilistic approach to modeling and estimating the QoS of web-services-based workflows". In: *Information Sciences* 177.23 (2007), pp. 5484–5503.

[135] Godar J Ibrahim, Tarik A Rashid, and Mobayode O Akinsolu. "An energy efficient service composition mechanism using a hybrid meta-heuristic algorithm in a mobile cloud environment". In: *Journal of parallel and distributed computing* 143 (2020), pp. 77–87.

[136] IFTTT. *Ifttt github repository*. 2021. URL: https://github.com/IFTTT.

[137] University of Illinois. *Maude download and installation*. 2022. URL: http://maude.cs.illinois.edu/w/index.php.

[138] INRIA. *Cadp online request form*. 2022. URL: https://cadp.inria.fr/registration.

[139] INRIA. *The coq assistant installation guide*. 2022. URL: https://coq.inria.fr/download.

[140] Inria. *Coq*. 2021. URL: https://github.com/coq/coq.

[141] Isabelle. *Isabelle*. 2022. URL: https://github.com/isabelle-prover.

[142] Nilesh Y Jadhav. *Green and smart buildings: advanced technology options*. Springer, 2016.

[143] Parita Jain et al. "Convergence of IoT and CPS in Robotics". In: *Emergence of Cyber Physical System and IoT in Smart Automation and Robotics: Computer Engineering in Automation*. Springer, 2021, pp. 15–30.

[144] Chandrashekar Jatoth, GR Gangadharan, and Rajkumar Buyya. "Computational intelligence based QoS-aware web service composition: a systematic literature review". In: *IEEE Transactions on Services Computing* 10.3 (2015), pp. 475–492.

[145] Bahar Jazayeri. "Architectural management of on-the-fly computing markets". In: *Proccedings of the 10th European Conference on Software Architecture Workshops*. 2016, pp. 1–2.

[146] Bahar Jazayeri and Simon Schwichtenberg. "On-the-fly computing meets IoT markets—towards a reference architecture". In: *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*. IEEE. 2017, pp. 120–127.

[147] Mengda Jia et al. "Adopting Internet of Things for the development of smart buildings: A review of enabling technologies and applications". In: *Automation in Construction* 101 (2019), pp. 111–126.

[148] Ruoxi Jia et al. "Design automation for smart building systems". In: *Proceedings of the IEEE* 106.9 (2018), pp. 1680–1699.

[149] Wenquan Jin et al. "Integrated Service Composition Approach Based on Transparent Access to Heterogeneous IoT Networks Using Multiple Service Providers". In: *Mobile Information Systems* 2021 (2021).

[150] Harlow Joe. *A library for communicating sequential processes in node.js, built on top of async/await*. 2020. URL: https://github.com/f5io/csp.

[151] Harlow Joe. *Build petri net models compositionally*. 2020. URL: https://github.com/AlgebraicJulia/AlgebraicPetri.jl.

[152] Nikos Kalatzis, Nikolaos Marianos, and Fotis Chatzipapadopoulos. "IoT and data interoperability in agriculture: A case study on the gaiasense TM smart farming solution". In: *2019 Global IoT Summit (GIoTS)*. IEEE. 2019, pp. 1–6.

[153] Eirini Kaldeli et al. "Interoperation, composition and simulation of services at home". In: *International Conference on Service-Oriented Computing*. Springer. 2010, pp. 167–181.

[154] Florian Kammüller. "Human centric security and privacy for the iot using formal techniques". In: *International Conference on Applied Human Factors and Ergonomics*. Springer. 2017, pp. 106–116.

[155] Holger Karl et al. "A case for a new IT ecosystem: On-The-Fly computing". In: *Business & Information Systems Engineering* 62.6 (2020), pp. 467–481.

[156] Stamatis Karnouskos et al. "Towards the real-time enterprise: service-based integration of heterogeneous SOA-ready industrial devices with enterprise applications". In: *IFAC Proceedings Volumes* 42.4 (2009), pp. 2131–2136.

[157] Aqeel Kazmi, Martin Serrano, and John Soldatos. "Vital-os: An open source iot operating system for smart cities". In: *IEEE Communications Standards Magazine* 2.2 (2018), pp. 71–77.

[158] Fondazione Bruno Kessler. *Nusmv download and installation guide*. 2022. URL: https://nusmv.fbk.eu/NuSMV/download/getting-v2.html.

[159] HALBA Khalid. *IoT Capabilities Composition Framework*. 2021. URL: https://github.com/usnistgov/ICCF.

[160] HALBA Khalid. *SLR Primary Studies Data*. 2022. URL: https://github.com/KhalidHALBA/SLR.

[161] Halba Khalid. *Ucef ADS Testbed*. 2022. URL: https://github.com/usnistgov/avchallenge.

[162] Morteza Khani Dehnoi and Saeed Araban. "A systematic literature review on web services composition". In: *International Journal of Nonlinear Analysis and Applications* 13.1 (2022), pp. 2821–2855.

[163] Ahmed Kheldoun and Malika Ioualalen. "Transformation BPEL processes to recatnet for analysing web services compositions". In: *2014 2nd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*. IEEE. 2014, pp. 425–430.

[164] Siham Khoussi et al. "Performance evaluation of the ndn data plane using statistical model checking". In: *International symposium on automated technology for verification and analysis*. Springer. 2019, pp. 534–550.

[165] Barbara Kitchenham et al. "Systematic literature reviews in software engineering–a systematic literature review". In: *Information and software technology* 51.1 (2009), pp. 7–15.

[166] Barbara Kitchenham et al. "Systematic literature reviews in software engineering–a tertiary study". In: *Information and software technology* 52.8 (2010), pp. 792–805.

[167] Tayo Koleoso. "Microservices with Quarkus". In: *Beginning Quarkus Framework*. Springer, 2020, pp. 51–132.

[168] Igor Konnov, Jure Kukovec, and Thanh-Hai Tran. "TLA+ model checking made symbolic". In: *Proceedings of the ACM on Programming Languages* 3.OOPSLA (2019), pp. 1–30.

[169] Ajay Krishna et al. "IoT Composer: Composition and deployment of IoT applications". In: *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE. 2019, pp. 19–22.

[170] Ajay Krishna et al. "MOZART: design and deployment of advanced IoT applications". In: *Companion proceedings of the web conference 2020*. 2020, pp. 163–166.

[171] Ajay Krishna et al. "Rigorous design and deployment of IoT applications". In: *2019 IEEE/ACM 7th International Conference on Formal Methods in Software Engineering (FormaliSE)*. IEEE. 2019, pp. 11–20.

[172] Markus Alexander Kuppe, Leslie Lamport, and Daniel Ricketts. "The TLA+ toolbox". In: *arXiv preprint arXiv:1912.10633* (2019).

[173] Agus Kurniawan. *Learning AWS IoT: Effectively manage connected devices on the AWS cloud using services such as AWS Greengrass, AWS button, predictive analytics and machine learning*. Packt Publishing Ltd, 2018.

[174] Marta Kwiatkowska, Gethin Norman, and David Parker. "PRISM 4.0: Verification of probabilistic real-time systems". In: *International conference on computer aided verification*. Springer. 2011, pp. 585–591.

[175] Leslie Lamport. *Introduction to TLA*. Digital Equipment Corporation Systems Research Center [SRC], 1994.

[176] Leslie Lamport. "The PlusCal algorithm language". In: *International Colloquium on Theoretical Aspects of Computing*. Springer. 2009, pp. 36–60.

[177] Cosimo Laneve and Luca Padovani. "An algebraic theory for web service contracts". In: *Formal Aspects of Computing* 27.4 (2015), pp. 613–640.

[178] Thomas M Lawrence et al. "Ten questions concerning integrating smart buildings into the smart grid". In: *Building and Environment* 108 (2016), pp. 273–283.

[179] Nicolas Le Sommer, Yves Mahéo, and Fadhlallah Baklouti. "Multi-strategy dynamic service composition in opportunistic networks". In: *Information* 11.4 (2020), p. 180.

[180] GM Lee and U Jayasinghe. "AI and blockchain enabled edge of things with privacy preserving computation". In: *Conference Proceedings of The XIII International scientific conference Perspective technologies in the information transfer means*. 2019, pp. 39–43.

[181] Kuan-Rong Lee, Meng-Hsuan Fu, and Yau-Hwang Kuo. "A hierarchical scheduling strategy for the composition services architecture based on cloud computing". In: *The 2nd International Conference on Next Generation Information Technology*. IEEE. 2011, pp. 163–169.

[182] Ye Lei, Yu MingYuan, and Huang Qing Zhang. "A Domain-oriented Business-Standardization Service Organization Model". In: *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*. Vol. 1. IEEE. 2010, pp. V1–666.

[183] Frédéric Lemoine, Tatiana Aubonnet, and Noëmie Simoni. "IoT composition based on self-controlled services". In: *Journal of Ambient Intelligence and Humanized Computing* 11.11 (2020), pp. 5167–5186.

[184] Angel Lagares Lemos, Florian Daniel, and Boualem Benatallah. "Web service composition: a survey of techniques and tools". In: *ACM Computing Surveys (CSUR)* 48.3 (2015), pp. 1–41.

[185] Lamport Leslie. *TLA+*. 2022. URL: https://github.com/tlaplus/tlaplus.

[186] Michael Leung, Alan Chan, and Paul Lam. "Control and management of hospital indoor air quality". In: *Med Sci Monit* 12.3 (2006), p. 23.

[187] Bixin Li et al. "Verifying the concurrent properties in BPEL based web service composition process". In: *IEEE Transactions on Network and Service Management* 10.4 (2013), pp. 410–424.

[188] Jianfeng Li and Ya-Fei Zhou. "Occupational hazards control of hazardous substances in clean room of semiconductor manufacturing plant using CFD analysis". In: *Toxicology and industrial health* 31.2 (2015), pp. 123–139.

[189] Lixing Li et al. "Modeling and analyzing the reliability and cost of service composition in the IoT: A probabilistic approach". In: *2012 IEEE 19th International Conference on Web Services*. IEEE. 2012, pp. 584–591.

[190] Wenbin Li et al. "Graph-based semantic evolution for context information management platforms". In: *2018 Global Internet of Things Summit (GIoTS)*. IEEE. 2018, pp. 1–6.

[191] Yuemin Li et al. "A formal validation method for trustworthy services composition". In: *2016 International Conference on Networking and Network Applications (NaNA)*. IEEE. 2016, pp. 433–437.

[192] Li Lin, Jian Hu, and Jianbiao Zhang. "Packet: a privacy-aware access control policy composition method for services composition in cloud environments". In: *Frontiers of Computer Science* 10.6 (2016), pp. 1142–1157.

[193] Jianhua Liu and Weiqin Tong. "Adaptive service framework based on grey decision-making in the internet of things". In: *2010 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM)*. IEEE. 2010, pp. 1–4.

[194] Jianhua Liu and Weiqin Tong. "Dynamic service model based on context resources in the internet of things". In: *2010 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM)*. IEEE. 2010, pp. 1–4.

[195] Zhengchao Liu et al. "A multi-attribute personalized recommendation method for manufacturing service composition with combining collaborative filtering and genetic algorithm". In: *Journal of Manufacturing Systems* 58 (2021), pp. 348–364.

[196] Duo Lu et al. "A secure microservice framework for iot". In: *2017 IEEE Symposium on Service-Oriented System Engineering (SOSE)*. IEEE. 2017, pp. 9–18.

[197] Yuteng Lu and Meng Sun. "Modeling and verification of IEEE 802.11 i security protocol in UPPAAL for Internet of Things". In: *International Journal of Software Engineering and Knowledge Engineering* 28.11n12 (2018), pp. 1619–1636.

[198] Weigong Lv et al. "A general architecture of IoT system". In: *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*. Vol. 1. IEEE. 2017, pp. 659–664.

[199] Afef Jmal Maâlej and Moez Krichen. "Study on the limitations of WS-BPEL compositions under load conditions". In: *The Computer Journal* 58.3 (2015), pp. 385–402.

[200] Afef Jmal Maâlej et al. "Distributed and Resource-Aware Load Testing of WS-BPEL Compositions." In: *ICEIS (2)*. 2018, pp. 29–38.

[201] Zakaria Maamar et al. "Orchestration-and choreography-based composition of Internet of Transactional Things". In: *Service Oriented Computing and Applications* 15.2 (2021), pp. 157–170.

[202] Charif Mahmoudi, Fabrice Mourlin, and Abdella Battou. "Formal definition of edge computing: An emphasis on mobile cloud and IoT composition". In: *2018 Third international conference on fog and mobile edge computing (FMEC)*. IEEE. 2018, pp. 34–42.

[203] Claudio Marche, Michele Nitti, and Virginia Pilloni. "Energy efficiency in smart building: a comfort aware approach based on Social Internet of Things". In: *2017 Global Internet of Things Summit (GIoTS)*. IEEE. 2017, pp. 1–6.

[204] Elena Markoska et al. "Assessment of building intelligence requirements for real time performance testing in smart buildings". In: *2019 4th International Conference on Smart and Sustainable Technologies (SpliTech)*. IEEE. 2019, pp. 1–6.

[205] Raoudha Maroui and Bechir Ayeb. "Integrating the sysml and acme in a model driven engineering approach to verify the web service composition". In: *2015 IEEE 24th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises*. IEEE. 2015, pp. 183–189.

[206] Teresa Martinez, Miriam Duarte, and Ana Cristina Garcia-Luna. "How using smart buildings technology can improve indoor environmental quality in educational buildings". In: *SHS Web of Conferences*. Vol. 102. EDP Sciences. 2021, p. 03003.

[207] Marvin and LeMaria. *Parse a simple custom description language for labeled transition systems, create composites (synchronize on identical transition symbols), and graph them.* 2019. URL: https://github.com/marvk/Labelled-Transition-System.

[208] Preeti Marwaha, Hema Banati, and Punam Bedi. "FORMALIZING BPEL-TC THROUGH [PI]-CALCULUS". In: *International Journal of Web & Semantic Technology* 4.3 (2013), p. 11.

[209] O Yu Maryasin, AS Kolodkina, and AA Ogarkov. "Computer Simulation of a Smart Building". In: *Automatic Control and Computer Sciences* 53.7 (2019), pp. 787–793.

[210] Norman Maurer and Marvin Wolfthal. *Netty in action*. Simon and Schuster, 2015.

[211] Sue LT McGregor and Elizabeth B Goldsmith. "Expanding our understanding of quality of life, standard of living, and well-being". In: *Journal of Family and Consumer Sciences* 90.2 (1998), p. 2.

[212] Deborah L McGuinness, Frank Van Harmelen, et al. "OWL web ontology language overview". In: *W3C recommendation* 10.10 (2004), p. 2004.

[213] Fabrıécio Martins Mendonça et al. "Wearable Devices in Healthcare: Challenges, Current Trends and a Proposition of Affordable Low Cost and Scalable Computational Environment of Internet of Things". In: *Brazilian Congress on Biomedical Engineering*. Springer. 2022, pp. 1301–1308.

[214] Mohamed El-Menshawy et al. "Verifying conformance of multi-agent commitment-based protocols". In: *Expert Systems with Applications* 40.1 (2013), pp. 122–138.

[215] Marjan Mernik. "Formal and Practical Aspects of Domain-Specific Languages: Recent Developments: Recent Developments". In: (2012).

[216] Hamza Merouani, Farid Mokhati, and Hassina Seridi-Bouchelaghem. "Towards formalizing web service composition in Maude's strategy language". In: *Proceedings of the 1st International Conference on Intelligent Semantic Web-Services and Applications*. 2010, pp. 1–6.

[217] Microsoft. *Azure IoT*. 2022. URL: https://github.com/Azure/iot.

[218] Microsoft. *Create an IoT hub using the Azure portal*. 2022. URL: https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-create-through-portal.

[219] Hana Mkaouar et al. "From AADL model to LNT specification". In: *Ada-Europe International Conference on Reliable Software Technologies*. Springer. 2015, pp. 146–161.

[220] Felix Mohr, Marcel Wever, and Eyke Hüllermeier. "Automated machine learning service composition". In: *arXiv preprint arXiv:1809.00486* (2018).

[221] Felix Mohr et al. "(WIP) towards the automated composition of machine learning services". In: *2018 IEEE International Conference on Services Computing (SCC)*. IEEE. 2018, pp. 241–244.

[222] Vince Molnár et al. "The Gamma statechart composition framework: design, verification and code generation for component-based reactive systems". In: *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)*. IEEE. 2018, pp. 113–116.

[223] Maria V Moreno, Miguel A Zamora, and Antonio F Skarmeta. "User-centric smart buildings for energy sustainable smart cities". In: *Transactions on emerging telecommunications technologies* 25.1 (2014), pp. 41–55.

[224] Guido Moritz et al. "Differences and commonalities of service-oriented device architectures, wireless sensor networks and networks-on-chip". In: *2009 International Conference on Advanced Information Networking and Applications Workshops*. IEEE. 2009, pp. 482–487.

[225] Hadi Naghavipour et al. "Hybrid Metaheuristics for QoS-Aware Service Composition: A Systematic Mapping Study". In: *IEEE Access* (2021).

[226] Chris Newcombe et al. "How Amazon web services uses formal methods". In: *Communications of the ACM* 58.4 (2015), pp. 66–73.

[227] Mahdi Saeedi Nikoo, Önder Babur, and Mark Van Den Brand. "A survey on service composition languages". In: *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*. 2020, pp. 1–5.

[228] Sewwandi Nisansala et al. "Microservice Based Edge Computing Architecture for Internet of Things". In: *2022 2nd International Conference on Advanced Research in Computing (ICARC)*. IEEE. 2022, pp. 332–337.

[229] Danmei Niu et al. "A Service Composition Mechanism Based on Mobile Edge Computing for IoT". In: *2019 6th International Conference on Information Science and Control Engineering (ICISCE)*. IEEE. 2019, pp. 982–985.

[230] Mahda Noura and Martin Gaedke. "WoTDL: web of things description language for automatic composition". In: *2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. IEEE. 2019, pp. 413–417.

[231] Vatsala Nundloll et al. "An ontological framework for opportunistic composition of IoT systems". In: *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*. IEEE. 2020, pp. 614–621.

[232] Samir Ouchani, Khaled Khebbeb, and Meriem Hafsi. "Towards Enhancing Security and Resilience in CPS: A Coq-Maude based Approach". In: *2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA)*. IEEE. 2020, pp. 1–6.

[233] Valderas P. *Microservice composition based on the choreography of bpmn fragments*. 2020. URL: https://github.com/pvalderas/microservices-composition-example.

[234] Michael P Papazoglou et al. "Service-oriented computing: a research roadmap". In: *International Journal of Cooperative Information Systems* 17.02 (2008), pp. 223–255.

[235] Manos Papoutsakis et al. "Towards a collection of security and privacy patterns". In: *Applied Sciences* 11.4 (2021), p. 1396.

[236] Primal Pappachan et al. "Towards privacy-aware smart buildings: Capturing, communicating, and enforcing privacy policies and preferences". In: *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*. IEEE. 2017, pp. 193–198.

[237] Zoe Paraskevopoulou, John M Li, and Andrew W Appel. "Compositional optimizations for certicoq". In: *Proceedings of the ACM on Programming Languages* 5.ICFP (2021), pp. 1–30.

[238] oneM2M Partners. *OneM2M Technical Report : Study of Abstraction and Semantics Enablements*. https://onem2m.org/images/files/deliverables/Release2/TR-0018-Industrial_Domain_Enablement-V2_0_0.pdf. 2016.

[239] Keyur K Patel, Sunil M Patel, and P Scholar. "Internet of things-IOT: definition, characteristics, architecture, enabling technologies, application & future challenges". In: *International journal of engineering science and computing* 6.5 (2016).

[240] Sonam Pathak and Manish Pandey. "Smart cities: Review of characteristics, composition, challenges and technologies". In: *2021 6th International Conference on Inventive Computation Technologies (ICICT)*. IEEE. 2021, pp. 871–876.

[241] Marcos A Pisching et al. "An architecture based on IoT and CPS to organize and locate services". In: *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE. 2016, pp. 1–4.

[242] Pascal Poizat and Gwen Salaün. "Checking the realizability of BPMN 2.0 choreographies". In: *Proceedings of the 27th Annual ACM Symposium on Applied Computing*. 2012, pp. 1927–1934.

[243] Julien Ponge. *Vert. x in Action*. Manning Publications, 2020.

[244] Alexandros Preventis et al. "IoT-A and FIWARE: Bridging the Barriers between the Cloud and IoT Systems Design and Implementation." In: *CLOSER (2)*. 2016, pp. 146–153.

[245] Gopal N Rai, GR Gangadharan, and Vineet Padmanabhan. "Algebraic modeling and verification of web service composition". In: *Procedia computer science* 52 (2015), pp. 675–679.

[246] Gopal N Rai et al. "Web service interaction modeling and verification using recursive composition algebra". In: *IEEE Transactions on Services Computing* 14.1 (2018), pp. 300–314.

[247] Ratchana Rajendran et al. "An Exploratory analysis of Machine Learning adaptability in Big Data Analytics Environments: A Data Aggregation in the age of Big Data and the Internet of Things". In: *2022 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM)*. Vol. 2. IEEE. 2022, pp. 32–36.

[248] Carlos Rodrigues, José Afonso, and Paulo Tomé. "Mobile application web-service performance analysis: Restful services with json and xml". In: *International Conference on ENTERprise Information Systems*. Springer. 2011, pp. 162–169.

[249] Raymond Roestenburg, Rob Williams, and Robertus Bakker. *Akka in action*. Simon and Schuster, 2016.

[250] Spin Root. *Spin download page*. 2022. URL: http://spinroot.com/spin/Src/index.html.

[251] Thomas Roth, Christopher Lemieux, and Martin Burns. "The UCEF Approach to Tool Integration for HLA Co-Simulations". In: *2020 IEEE Workshop on Design Automation for CPS and IoT (DESTION)*. IEEE. 2020, pp. 10–18.

[252] Alireza Safaei, Ramin Nassiri, and Amir Masoud Rahmani. "Enterprise service composition models in IoT context: solutions comparison". In: *The Journal of Supercomputing* 78.2 (2022), pp. 2015–2042.

[253] Hamza Safri et al. "A Federated Learning Framework for IoT: Application to Industry 4.0". In: *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE. 2022, pp. 565–574.

[254] Gwen Salaün. "Quantifying the similarity of non-bisimilar labelled transition systems". In: *Science of Computer Programming* 202 (2021), p. 102580.

[255] Ganesh Ram Santhanam, Samik Basu, and Vasant Honavar. "On utilizing qualitative preferences in web service composition: a cp-net based approach". In: *2008 IEEE Congress on Services-Part I*. IEEE. 2008, pp. 538–544.

[256] Ines Sarray et al. "Safe composition in middleware for the internet of things". In: *Proceedings of the 2nd Workshop on Middleware for Context-Aware Applications in the IoT*. 2015, pp. 7–12.

[257] Alexandra Schieweck et al. "Smart homes and the control of indoor air quality". In: *Renewable and Sustainable Energy Reviews* 94 (2018), pp. 705–718.

[258] Seyedsalar Sefati and Nima Jafari Navimipour. "A qos-aware service composition mechanism in the internet of things using a hidden-markov-model-based optimization algorithm". In: *IEEE Internet of Things Journal* 8.20 (2021), pp. 15620–15627.

[259] Sedelnikov Serge. *.net implementation of the finite state machine framework*. 2020. URL: https://github.com/serge-sedelnikov/xstate.net.

[260] Amazon Web Services. *Green grass IoT*. 2022. URL: https://github.com/aws-greengrass.

[261] Meng Shen et al. "Privacy-preserving support vector machine training over blockchain-based encrypted IoT data in smart cities". In: *IEEE Internet of Things Journal* 6.5 (2019), pp. 7702–7712.

[262] Quan Z Sheng et al. "Web services composition: A decade's overview". In: *Information Sciences* 280 (2014), pp. 218–238.

[263] Gabor Simko et al. "Specification of cyber-physical components with formal semantics–integration and composition". In: *International Conference on Model Driven Engineering Languages and Systems*. Springer. 2013, pp. 471–487.

[264] Christopher Simpkin et al. "Dynamic distributed orchestration of node-red iot workflows using a vector symbolic architecture". In: *2018 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS)*. IEEE. 2018, pp. 52–63.

[265] Tarana Singh, Arun Solanki, and Sanjay Kumar Sharma. "Role of smart buildings in smart city—components, technology, indicators, challenges, future research opportunities". In: *Digital Cities Roadmap: IoT-Based Architecture and Sustainable Buildings* (2021), pp. 449–476.

[266] Fikret Sivrikaya et al. "Internet of smart city objects: A distributed framework for service discovery and composition". In: *IEEE Access* 7 (2019), pp. 14434–14454.

[267] John Soldatos. *Building Blocks for IoT Analytics*. River Publishers, 2016.

[268] Zhexuan Song, Alvaro A Cárdenas, and Ryusuke Masuoka. "Semantic middleware for the internet of things". In: *2010 Internet of Things (IOT)*. IEEE. 2010, pp. 1–8.

[269] Alireza Souri and Mostafa Ghobaei-Arani. "Cloud manufacturing service composition in IoT applications: a formal verification-based approach". In: *Multimedia Tools and Applications* (2021), pp. 1–20.

[270] Alireza Souri and Monire Norouzi. "A state-of-the-art survey on formal verification of the internet of things applications". In: *Journal of Service Science Research* 11.1 (2019), pp. 47–67.

[271] Alireza Souri, Amir Masoud Rahmani, and Nima Jafari Navimipour. "Formal verification approaches in the web service composition: a comprehensive analysis of the current challenges for future research". In: *International journal of communication systems* 31.17 (2018), e3808.

[272] Alireza Souri et al. "A hybrid formal verification approach for QoS-aware multi-cloud service composition". In: *Cluster Computing* 23.4 (2020), pp. 2453–2470.

[273] Luciana Moreira Sá de Souza et al. "Socrades: A web service based shop floor integration infrastructure". In: *The internet of things*. Springer, 2008, pp. 50–67.

[274] Matthieu Sozeau et al. "Coq coq correct! verification of type checking and erasure for coq, in coq". In: *Proceedings of the ACM on Programming Languages* 4.POPL (2019), pp. 1–28.

[275] Maria Spichkova and Heinrich Schmidt. "Towards logical architecture and formal analysis of dependencies between services". In: *2014 Asia-Pacific Services Computing Conference*. IEEE. 2014, pp. 121–128.

[276] Josef Spillner. "Quantitative analysis of cloud function evolution in the AWS serverless application repository". In: *arXiv preprint arXiv:1905.04800* (2019).

[277] Zlatko Stapic et al. "Performing systematic literature review in software engineering". In: *Central European Conference on Information and Intelligent Systems*. Faculty of Organization and Informatics Varazdin. 2012, p. 441.

[278] Koch Stefan et al. *Node-RED and ibm watson integration*. 2020. URL: https://github.com/acoustic-content-samples/sample-node-red-contrib-wch.

[279] Merz Stephan and Alexander Kuppe Markus. *PlusCal Cheat Sheet by Stephan Merz*. 2021. URL: https://github.com/tlaplus/PlusCalCheatSheet.

[280] Chang-ai Sun et al. "Automated testing of WS-BPEL service compositions: a scenario-oriented approach". In: *IEEE Transactions on Services Computing* 11.4 (2015), pp. 616–629.

[281] Mengyu Sun et al. "Energy-efficient IoT service composition for concurrent timed applications". In: *Future Generation Computer Systems* 100 (2019), pp. 1017–1030.

[282] Yan Sun et al. "Verifying noninterference in a cyber-physical system the advanced electric power grid". In: *Seventh International Conference on Quality Software (QSIC 2007)*. IEEE. 2007, pp. 363–369.

[283] Junichi Tatemura and Wang-Pin Hsiung. "Web service decomposition: Edge computing architecture for cache-friendly e-commerce applications". In: *Electronic Commerce Research and Applications* 5.1 (2006), pp. 57–65.

[284] Varun M Tayur and R Suchithra. "Review of interoperability approaches in application layer of Internet of Things". In: *2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*. IEEE. 2017, pp. 322–326.

[285] Elias Z Tragos et al. "Trusted IoT in the complex landscape of governance, security, privacy, availability and safety". In: *Digitising the Industry-Internet of Things Connecting the Physical, Digital and Virtual Worlds. River Publishers Series in Communications* (2016), pp. 210–239.

[286] B Trammell et al. *mPlane Architecture Specification*.

[287] Brian Trammell et al. "mPlane: an intelligent measurement plane for the internet". In: *IEEE Communications Magazine* 52.5 (2014), pp. 148–156.

[288] Gregory Trencher and Andrew Karvonen. "Stretching "smart": Advancing health and well-being through the smart city agenda". In: *Local Environment* 24.7 (2019), pp. 610–627.

[289] Israr Ullah, Muhammad Sohail Khan, and DoHyeun Kim. "IoT services and virtual objects management in hyperconnected things network". In: *Mobile Information Systems* 2018 (2018).

[290] UPPSALA UNIVERSITAT. *Mwb'99 tool installation*. 2022. URL: https://www.it.uu.se/research/group/concurrency/mwb/installation.

[291] Michigan State University. *Cwb-nc download folder*. 2022. URL: http://sens.cse.msu.edu/Software/CWB-NC/.

[292] Uppsala University and Aalborg University. *Uppaal download and installation*. 2022. URL: https://uppaal.org/downloads/.

[293] Blase Ur et al. "Trigger-action programming in the wild: An analysis of 200,000 ifttt recipes". In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 2016, pp. 3227–3231.

[294] Aitor Urbieta et al. "Adaptive and context-aware service composition for IoT-based smart cities". In: *Future Generation Computer Systems* 76 (2017), pp. 262–274.

[295] Asrin Vakili and Nima Jafari Navimipour. "Comprehensive and systematic review of the service composition mechanisms in the cloud environments". In: *Journal of Network and Computer Applications* 81 (2017), pp. 24–36.

[296] Pedro Valderas, Victoria Torres, and Vicente Pelechano. "A microservice composition approach based on the choreography of BPMN fragments". In: *Information and Software Technology* 127 (2020), p. 106370.

[297] Richard Vedvik. "Understanding WELL v2 certification: As WELL v2 certification becomes more prevalent, project managers, engineers and designers need to understand which preconditions and optimization features are impacted by the systems they design." In: *Consulting Specifying Engineer* 58.6 (2021), pp. 24–31.

[298] Craig Walls. *Spring Boot in action*. Simon and Schuster, 2015.

[299] Hongbing Wang, Peisheng Ma, and Xuan Zhou. "A Quantitative and Qualitative Approach for NFP-Aware Web Service Composition". In: *2012 IEEE Ninth International Conference on Services Computing*. 2012, pp. 202–209. DOI: 10.1109/SCC.2012.16.

[300] Hongbing Wang, Qianzhao Zhou, and Yanqi Shi. "Describing and verifying web service composition using TLA reasoning". In: *2010 IEEE International Conference on Services Computing*. IEEE. 2010, pp. 234–241.

[301] Hongbing Wang et al. "Integrating recurrent neural networks and reinforcement learning for dynamic service composition". In: *Future Generation Computer Systems* 107 (2020), pp. 551–563.

[302] Qian Wang et al. "CS-Man: Computation service management for IoT in-network processing". In: *2016 27th Irish Signals and Systems Conference (ISSC)*. IEEE. 2016, pp. 1–6.

[303] Wei Wang et al. "A comprehensive ontology for knowledge representation in the internet of things". In: *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE. 2012, pp. 1793–1798.

[304] Yukun Wang et al. "Performance comparison and evaluation of websocket frameworks: Netty, undertow, vert. x, grizzly and jetty". In: *2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN)*. IEEE. 2018, pp. 13–17.

[305] Makarius Wenzel. "Miscellaneous Isabelle/Isar examples". In: (2021).

[306] Peder Wolkoff. "Indoor air humidity, air quality, and health–An overview". In: *International journal of hygiene and environmental health* 221.3 (2018), pp. 376–390.

[307] David Wollman et al. *Framework for Cyber-Physical Systems: Volume 3, Timing Annex*. en. Sept. 2017. DOI: https://doi.org/10.6028/NIST.SP.1500-203.

[308] Ruowei Xiao, Zhanwei Wu, and Dongyu Wang. "A Finite-State-Machine model driven service composition architecture for internet of things rapid prototyping". In: *Future Generation Computer Systems* 99 (2019), pp. 473–488.

[309] Na Xie et al. "An efficient two-phase approach for reliable collaboration-aware service composition in cloud manufacturing". In: *Journal of Industrial Information Integration* 23 (2021), p. 100211.

[310] Hisatoshi Yamaoka et al. "Dracena: A real-time iot service platform based on flexible composition of data streams". In: *2019 IEEE/SICE International Symposium on System Integration (SII)*. IEEE. 2019, pp. 596–601.

[311] Rui Yang and Lingfeng Wang. "Multi-agent based energy and comfort management in a building environment considering behaviors of occupants". In: *2012 IEEE Power and Energy Society General Meeting*. IEEE. 2012, pp. 1–7.

[312] Wing Lok Yeung. "A formal and visual modeling approach to choreography based web services composition and conformance verification". In: *Expert Systems with Applications* 38.10 (2011), pp. 12772–12785.

[313] Seongjin Yun et al. "A novel reference model for cloud manufacturing cps platform based on onem2m standard". In: *KIPS Transactions on Computer and Communication Systems* 8.2 (2019), pp. 41–56.

[314] Baili Zhang et al. "A top-K QoS-optimal service composition approach based on service dependency graph". In: *Journal of Organizational and End User Computing (JOEUC)* 33.3 (2021), pp. 50–68.

[315] Lili Zhang et al. "Research on IOT RESTful web service asynchronous composition based on BPEL". In: *2014 Sixth International Conference on Intelligent Human-Machine Systems and Cybernetics*. Vol. 1. IEEE. 2014, pp. 62–65.

[316] Huiqun Zhao et al. "Research on formal modeling and verification of BPEL-based web service composition". In: *2012 IEEE/ACIS 11th International Conference on Computer and Information Science*. IEEE. 2012, pp. 631–636.

[317] Hua Zhou, Zhiqiu Huang, and Guoan Zhao. "A service-centric solution for wireless sensor networks". In: *2010 5th International ICST Conference on Communications and Networking in China*. IEEE. 2010, pp. 1–5.

[318] Jiehan Zhou et al. "Cloud architecture for dynamic service composition". In: *International Journal of Grid and High Performance Computing (IJGHPC)* 4.2 (2012), pp. 17–31.

[319] Ming Zhu et al. "Modeling and verification of response time of QoS-aware web service composition by timed CSP". In: *Procedia Computer Science* 141 (2018), pp. 48–55.

[320] Paraskevopoulou Zoe. *CertiCoq*. 2022. URL: https://github.com/CertiCoq/certicoq.

[321] Steffen Zschaler. "Formal specification of non-functional properties of component-based software systems". In: *Software & Systems Modeling* 9.2 (2010), pp. 161–201.