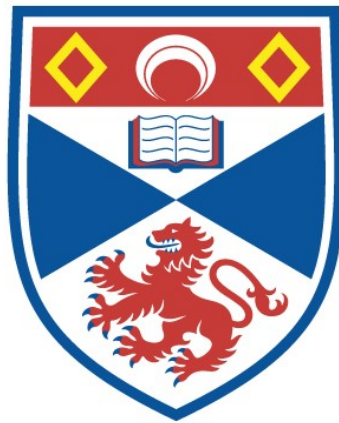


A CONCEPTUAL FRAMEWORK FOR UNCERTAINTY IN SOFTWARE SYSTEMS AND ITS APPLICATION TO SOFTWARE ARCHITECTURES

Chawanangwa Lupafya

A Thesis Submitted for the Degree of PhD
at the
University of St Andrews



2023

Full metadata for this item is available in
St Andrews Research Repository
at:

<http://research-repository.st-andrews.ac.uk/>

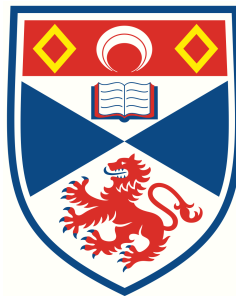
Identifiers to use to cite or link to this thesis:

DOI: <https://doi.org/10.17630/sta/264>
<http://hdl.handle.net/10023/26909>

This item is protected by original copyright

A Conceptual Framework for Uncertainty in Software Systems and its Application to Software Architectures

Chawanangwa Lupafya



University of
St Andrews

This thesis is submitted in partial fulfilment for the degree of
Doctor of Philosophy
at the University of St Andrews

July 2022

Abstract

The development and operation of a software system involve many aspects including processes, artefacts, infrastructure and environments. Most of these aspects are vulnerable to uncertainty. Thus, the identification, representation and management of uncertainty in software systems is important and will be of interest to many stakeholders in software systems. The hypothesis of this work is that such consideration would benefit from an underlying conceptual framework that allows stakeholders to characterise, analyse and mitigate uncertainties. This PhD proposes a framework to provide a generic foundation for the systematic and explicit consideration of uncertainty in software systems by consolidating and extending existing approaches to dealing with uncertainty, which are typically tailored to specific domains or artefacts. The thesis applies the framework to software architectures, which are fundamental in determining the structure, behaviour and qualities of software systems and are thus suited to serve as an exemplar artefact. The framework is evaluated using the software architectures of case studies from 3 different domains. The contributions of the research to the study of uncertainty in software systems include a literature review of approaches to managing uncertainty in software architecture, a review of existing work on uncertainty frameworks related to software systems, a conceptual framework for uncertainty in software systems, a conceptualisation of the workbench infrastructure as a basis for building an uncertainty consideration workbench of tools for representing uncertainty as part of software architecture descriptions, and an evaluation of the uncertainty framework using three software architecture case studies.

Acknowledgements

My gratitude goes to my supervisor, Dharini, for the guidance and support throughout the PhD process. To you, I dedicate the following quote: *at times our own light goes out and is rekindled by a spark from another person. Each of us has cause to think with deep gratitude of those who have lighted the flame within us.* - Albert Schweitzer. Thank you!

To the University of St Andrews and to the School of Computer Science, thank you for the Scholarship which made my studies possible and the various funding opportunities which have supported me through my studies.

To my family, Mother, Brother and Sisters, thank you for being there for me throughout!

To my Auntie(Mum - Mercy Msiska, Mrs Mwandira), Oh, how much I was looking forward to telling you that I completed my studies, but life had a different plan: *I still miss those I loved who are no longer with me but I find I am grateful for having loved them. The gratitude has finally conquered the loss.* - Rita Mae Brown

To all those I have met during the Ph.D. period, both virtual and physically, too many to mention, I dedicate the following quote because you have been good to me and I am grateful: *cultivate the habit of being grateful for every good thing that comes to you, and to give thanks continuously. And because all things have contributed to your advancement, you should include all things in your gratitude.* - Ralph Waldo Emerson

To my friends who we met in St Andrews, Cupar and in Scotland in general, and back home in Malawi, thank you for being wonderful, you have contributed in so many ways to making my PhD experience great! In particular, I would like to mention the only friend from Malawi who we met here in St Andrews, Kondwani Katundu, I couldn't have asked for a better friend!

Candidate's declaration

I, Chawanangwa Lupafya, do hereby certify that this thesis, submitted for the degree of PhD, which is approximately 45,000 words in length, has been written by me, and that it is the record of work carried out by me, or principally by myself in collaboration with others as acknowledged, and that it has not been submitted in any previous application for any degree. I confirm that any appendices included in my thesis contain only material permitted by the 'Assessment of Postgraduate Research Students' policy.

I was admitted as a research student at the University of St Andrews in September 2017.

I received funding from an organisation or institution and have acknowledged the funder(s) in the full text of my thesis.

Date 24 July 2022

Signature of candidate

Supervisor's declaration

I hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of PhD in the University of St Andrews and that the candidate is qualified to submit this thesis in application for that degree. I confirm that any appendices included in the thesis contain only material permitted by the 'Assessment of Postgraduate Research Students' policy.

Date 6 January 2023

Signature of supervisor

Permission for publication

In submitting this thesis to the University of St Andrews we understand that we are giving permission for it to be made available for use in accordance with the regulations of the University Library for the time being in force, subject to any copyright vested in the work not being affected thereby. We also understand, unless exempt by an award of an embargo as requested below, that the title and the abstract will be published, and that a copy of the work may be made and supplied to any bona fide library or research worker, that this thesis will be electronically accessible for personal or research use and that the library has the right to migrate this thesis into new electronic forms as required to ensure continued access to the thesis.

I, Chawanangwa Lupafya, confirm that my thesis does not contain any third-party material that requires copyright clearance.

The following is an agreed request by candidate and supervisor regarding the publication of this thesis:

Printed copy

No embargo on print copy.

Electronic copy

No embargo on electronic copy.

Date 24 July 2022

Signature of candidate

Date 6 January 2023

Signature of supervisor

Underpinning Research Data or Digital Outputs

Candidate's declaration

I, Chawanangwa Lupafya, understand that by declaring that I have original research data or digital outputs, I should make every effort in meeting the University's and research funders' requirements on the deposit and sharing of research data or research digital outputs.

Date 24 July 2022

Signature of candidate

Permission for publication of underpinning research data or digital outputs

We understand that for any original research data or digital outputs which are deposited, we are giving permission for them to be made available for use in accordance with the requirements of the University and research funders, for the time being in force.

We also understand that the title and the description will be published, and that the underpinning research data or digital outputs will be electronically accessible for use in accordance with the license specified at the point of deposit, unless exempt by award of an embargo as requested below.

The following is an agreed request by candidate and supervisor regarding the publication of underpinning research data or digital outputs:

No embargo on underpinning research data or digital outputs.

Date 24 July 2022

Signature of candidate

Date 6 January 2023

Signature of supervisor

*I dedicate this work to my Mother (Esther), Chimwemwe, Brenda,
Gomezgani and Mercy!*

Publications

Publication 1 Lupafya C. A framework for managing uncertainty in software architecture. In Proceedings of the 13th European Conference on Software Architecture-Volume 2 2019 Sep 9 (pp. 71-74).

Publication 2 Lupafya C, Balasubramaniam D. A Framework for Considering Uncertainty in Software Systems In2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC) 2022 Jul. IEEE.

Submitted to Journal 1 Lupafya C, Balasubramaniam D. Towards an Uncertainty Framework for Software Systems, Journal - Information and Software Technology, Submitted in June, 2022

Submitted to Journal 2 Lupafya C, Balasubramaniam D. Uncertainty in Software Architecture: A Survey, Journal - ACM Computing Surveys, Submitted in July 2022

COVID STATEMENT OF IMPACT

I received twelve months of COVID Discount Time due to both personal health and research-related issues during the COVID period. The main impact on my research was in terms of project timelines and the evaluation approach.

CONTENTS

Contents	i
List of Figures	vi
List of Tables	vii
Acronyms	ix
1 Introduction	1
1.1 Research context and motivation	1
1.2 Research Questions	3
1.2.1 Research Question 1	3
1.2.2 Research Question 2	3
1.3 Contributions	3
1.4 Organisation of Dissertation	4
2 Methodology	7
2.1 Scope and approach	7
2.2 Phases and steps	8
2.2.1 Phase one	8
2.2.2 Phase two	8
2.3 Discussion	10
2.4 Conclusions	10
3 Uncertainty and software systems Background - illustrative example	11
3.1 Overview	11
3.2 Uncertainty and software systems	11
3.3 Uncertainty and its expression - illustrative example	12
3.4 Managing uncertainty in software systems	14
3.5 Conclusion	15
4 Uncertainty in Software Architecture: A Survey	17
4.1 Overview	17
4.2 Related work	19
4.3 Survey design	19
4.3.1 Identifying aspects of software architecture	20

4.3.2	Survey questions	21
4.3.3	Inclusion and exclusion criteria	21
4.4	Survey process	23
4.4.1	Data collection and trends	23
4.4.2	Limitations and threats to validity	25
4.5	Survey Data	28
4.6	Overview of software architecture aspects	28
4.7	Categorisation of software architecture aspects	29
4.8	Uncertainty in software architecture aspects	32
4.8.1	Architecture Definitions and Concepts	32
4.8.2	Architecture Activities	36
4.8.3	Architecture Artefacts	39
4.8.4	Architecture Tools and Notations: Uncertainty capabilities	40
4.8.5	Summary	42
4.9	Risks and opportunities of uncertainty in software architecture aspects	42
4.10	Mitigation and exploitation of uncertainty in software architecture aspects	43
4.11	Discussion: Gaps and directions for Future Work	44
4.12	Conclusions	47
5	Uncertainty characterisation concepts: state of the art	49
5.1	Overview	49
5.2	Existing approaches for considering uncertainty in software systems	50
5.2.1	Uncertainty in cognitive science	51
5.2.2	Uncertainty in complex systems	52
5.2.3	Uncertainty in cyber-physical systems	53
5.2.4	Uncertainty in self-adaptive systems	54
5.3	Summary of existing frameworks	56
5.4	Conclusion	57
6	Framework	59
6.1	Overview	59
6.2	A framework for considering uncertainty in software systems	60
6.2.1	Framework definition	60
6.2.2	Applying the framework	64
6.2.3	Uncertainty representation and analysis	66
6.3	Evaluation and discussion	67
6.3.1	Case studies	67
6.3.2	Illustration of uncertainties in case studies	69
6.3.3	Demonstration of extensibility: Future studies concept	72
6.3.4	Discussion	73
6.4	Conclusion	74
7	The Workbench Infrastructure	75
7.1	Overview	75
7.2	Concepts of the workbench infrastructure	76

7.3	Design of the workbench infrastructure	76
7.4	Realisation of the workbench infrastructure	78
7.5	Implementation of the uncertainty capture tool	79
7.6	Critical discussion of the workbench infrastructure	82
7.6.1	Validity of the workbench infrastructure concept	82
7.6.2	Workbench infrastructure tools and notations support	82
7.6.3	Application of workbench infrastructure	82
7.7	Conclusion	83
8	Evaluation strategy	85
8.1	Introduction	85
8.2	Evaluation strategy	85
8.3	Architecture case study selection criteria	86
8.4	The case studies and evaluation	87
8.5	The case studies evaluation approaches	90
8.5.1	Case study 1: Knowledge generation: uncertainty capturing and documentation	90
8.5.2	Case study 2: Uncertainty analysis of software architecture	90
8.5.3	Case study 3: Alternative candidate architecture uncertainty ranking and selection	92
8.6	The ATAM overview and customisation	93
8.6.1	ATAM Customisation	93
8.6.2	The uncertainty framework and the ATAM	94
8.6.3	Independent application of the uncertainty framework - MSc project vs An Architecture and Open-Source Software Framework for Aerial Robotics (AEROSTACK)	96
8.7	Conclusion	97
9	Case Study 1: AEROSTACK - Uncertainty Knowledge Generation and Documentation	99
9.1	Overview	99
9.2	AEROSTACK Architecture Description	100
9.3	Logical view	102
9.4	Development view	105
9.5	Discussion	106
9.5.1	Encoding and interpreting uncertainty knowledge and documentation	106
9.5.2	Uncertainty documentation and knowledge	110
9.6	Conclusions	111
10	Case Study 2: IMPALA - uncertainty architecture analysis	113
10.1	Overview	113
10.2	IMPALA platform architecture description	113
10.3	Logical view	116
10.3.1	Uncertainty of the <i>Capture</i> component	117
10.3.2	Uncertainty of the <i>Transport</i> component	117

10.3.3	Uncertainty of the <i>Refine</i> component	117
10.3.4	Uncertainty of the <i>Store</i> component	118
10.3.5	Uncertainty of the <i>Analyze</i> component	118
10.3.6	Uncertainty of the <i>Distribute</i> component	118
10.3.7	Uncertainty of the <i>Manage</i> component	119
10.4	Functional view	122
10.5	Infrastructure view	126
10.6	Network view	129
10.7	Security view	130
10.8	Discussion	130
10.8.1	Uncertainty data overview	132
10.8.2	Uncertainty knowledge and Documentation	133
10.8.3	Uncertainty data quality	133
10.8.4	IMPALA platform uncertainty consideration architecture analysis - Report	133
10.8.5	IMPALA platform ATAM architecture analysis - Report	137
10.8.6	Comparison of analysis reports insights: uncertainty consideration framework vs the customised ATAM	142
10.9	Conclusions	143
11	Case study 3: IoV - comparison of candidate architectures on uncertainty	145
11.1	Overview	145
11.2	IoV Architectures	147
11.2.1	Candidate Architecture One	147
11.2.2	Candidate Architecture Two	150
11.3	Candidate Architectures Comparison approach	153
11.3.1	Uncertainty influence measure attribute	153
11.3.2	Framework extensibility - <i>Influence measure</i> attributes	153
11.3.3	The common uncertainties	154
11.3.4	Uncertainties analysis - ordinal data generation	155
11.3.5	Ranking candidate architecture: comparison	163
11.4	Discussion	167
11.5	Conclusions	168
12	Conclusions	171
12.1	Overview	171
12.2	Summary of main contributions	171
12.2.1	A literature survey of uncertainty in software architecture from 1991 to 2021	171
12.2.2	A literature survey of uncertainty frameworks for software system . . .	172
12.2.3	A definition of a conceptual framework for considering uncertainty in software systems	172
12.2.4	The specification and development of the workbench infrastructure concept	172
12.2.5	Evaluation of the uncertainty consideration framework on software architecture case studies	173
12.2.6	Critical reflection on the uncertainty consideration framework	173

12.2.7	Final thoughts on the main contributions	174
12.3	Future work	174
12.3.1	Continuous uncertainty management	175
12.3.2	Uncertainty and Agile software development	175
12.3.3	Uncertainty framework automation	175
12.3.4	Uncertainty framework customisation	175
12.3.5	Final thought on future work	176
12.4	Conclusions	176
References		177
Appendix A	Appendix-A - Architecture Survey Literature Sources	189
Appendix B	Appendix-B Uncertainty framework data and meta-data	195
B.1	Uncertainty Attributes - JSON OBJECT template	195
B.2	XML data for screenshot	198
Appendix C	Appendix-C: Case Study 3 Uncertainty framework data	227

LIST OF FIGURES

4.1	Distribution of literature sources per year from 1991 - 2021	25
5.1	Taxonomy of Ignorance (Reproduced from [1])	51
5.2	Uncertainty in complex systems (Based on [2]	52
5.3	Conceptual model (Reproduced from [3])	53
5.4	MAPE-K Loop (Image from [4])	54
6.1	Conceptual model: characterisation of uncertainty in systems	62
7.1	Logical view of the technology stack of the workbench infrastructure	77
7.2	Technology stack of the workbench infrastructure	77
7.3	Screenshot of the workbench (XML source code for the screenshot is in Appendix B Listing B.2)	80
9.1	Logical view of AEROSTACK from the original image in [5]	101
9.2	Development view of AEROSTACK Architecture from the original image in [5] . .	105
10.1	Logical view of IMPALA platform from the original in SDD ¹	115
10.2	Functional view of IMPALA platform from the original image in SDD ¹	122
10.3	Infrastructure view of IMPALA platform from the original image in SDD ¹	126
10.4	Network view of IMPALA platform from the original image in SDD ¹	129
10.5	Security view of IMPALA platform from the original image in SDD ¹	131
11.1	Universal IoV for smart cities overview (Image from [6])	148
11.2	Universal IoV Layers (Image from [6]))	149
11.3	Candidate architecture two - IoV Layers (Image from [7])	151

LIST OF TABLES

4.1	Databases search results from 1991-2021	22
4.2	Coverage of aspects in literature sources	26
4.3	Languages, notations, and tools designed for <i>explicitly</i> working with uncertainty in software architecture	41
5.1	Literature sources on uncertainty frameworks	50
6.1	Consolidated uncertainty characterisation attributes. CgS - cognitive science, CxS - complex systems, CPS - cyber-physical systems, SAS - self-adaptive systems, REL - the RELAX language	65
6.2	Examples of uncertainty framework values	70
6.3	Extending mapping - Mitigation attribute	72
8.1	Selection criteria for the three case studies summary	89
8.2	Comparison of the completed uncertainty attributes	96
9.1	Uncertainties of the Logical view	103
9.2	Uncertainties of the Logical view	104
9.3	Uncertainties of the development view	107
10.1	Uncertainties of the Logical view	120
10.2	Uncertainties of the Logical view	121
10.3	Uncertainties of the Functional view	125
10.4	Uncertainties of the Infrastructure view	128
10.5	Uncertainties of the Network view	130
10.6	Uncertainties per architecture component	132
10.7	IMPALA customised Architecture Tradeoff Analysis Method (ATAM) analysis results	139
10.8	IMPALA customised ATAM analysis Results	140
11.1	Uncertainties and their influence measures	155
11.2	Uncertainties, Influence, and Candidate Architectures	156
11.3	Uncertainties, influence, and candidate architectures with encoded values	164
11.4	Uncertainty weighing digits	164
11.5	Candidate architecture and uncertainty weighing digits	165
11.6	Candidate architecture and uncertainty weighing digits	165
11.7	Uncertainties with similar influence scores eliminated	167
11.8	Candidate architecture and uncertainty weighing digits with similarities eliminated	167

A.1	<i>Part 1 of 5: Research focus of literature source: aspects</i>	190
C.1	Internet of Vehicles common uncertainties data	228
C.2	Internet of Vehicles common uncertainties	229

ACRONYMS

ADL Description Language

ADLs Description Languages

AEROSTACK An Architecture and Open-Source Software Framework for Aerial Robotics

AHP Analytic Hierarchy Process

API Application Programming Interface

ARA Aerial Robotics Architecture

ATAM Architecture Tradeoff Analysis Method

BI Business Intelligence

BSC Battlefield Control System

Cd-Sys Connected systems

CHAM Chemical Abstract Model

COTS Commercially available Off-The-Shelf

CPS Cyber-Physical Systems

CRUD create, read, update, and delete

D2D Device-to-Device

ERD Entity Relationship Diagram

ERP Enterprise Resource Planning

ETL Extract, Transform, Load

FAME Fuzzy self-Adaptation ModEling

GPS Global Positioning System

IDE Integrated Development Environment

IMPALA The Information Management Platform for Data Analytics and Aggregation

IoT Internet of Things

IoV Internet of Vehicles

ITS Intelligent Transport Systems

JSC Johnson Space Centre

JSON JavaScript Object Notation

MASH Mission Associated Summary of Health

MEDB Medical Evaluation Document, Part B

MEME Mission Extended Medical Enterprise

NAMS NASA Access Management System

NASA National Aeronautics and Space Administration

NIC Network Interface Cards

OED The Oxford English Dictionary

OLAP Online analytical processing

RADAR Requirements and Architecture Decision AnalyseRl

RAID Redundant Array of Independent Disks

SA-LC Stages Uncertainty in software architecture lifecycle stages

SDD System Design Document

SoADL Systems-of-Systems Architecture Description Language

SoS Systems-of-Systems

SysML Systems Modeling Language

Sys-Q System qualities

UAS Unmanned Aerial Systems

UAV Unmanned Aerial Vehicle

UIoV Universal Internet of Vehicles

UML Unified Modeling Language

V2B Vehicle-to-Building

V2D Vehicle-to-Device

V2G Vehicle-to-Grid

V2H Vehicle-to-Home

V2I Vehicle-to-Infrastructure

V2R Vehicle-to-Road

V2S Vehicle-to-Sensor

V2V Vehicle-to-Vehicle

V2X Vehicle-to-Everything

VPN Virtual Private Network

XML Extensible Markup Language

INTRODUCTION

This chapter introduces the work conducted to define a conceptual framework for uncertainty in software systems and its application to software architectures. Software architecture is one of the many facets of a software system. Thus, this chapter introduces software architecture in the context of a software system, and then outlines the research context and motivation, the research questions, main contributions of the work and organisation of the rest of the thesis.

1.1 Research context and motivation

The development and operation of software systems involve many facets, such as processes, requirements, software architecture, infrastructure and environments [8]. All such facets are subject to uncertainty which can potentially cause positive or negative consequences [2, 9].

On the one hand, it may be possible to exploit the positive consequences of uncertainty as opportunities [2]. For example, excess resources from over-estimation during project planning or system design may be re-invested to increase productivity or improve system quality, instead of being treated as waste or losses.

On the other hand, it may be necessary to mitigate the adverse effects of uncertainty, such as network disruptions, data loss or security breaches because these can lead to risks to system functionality and performance, among other consequences [2, 10]. Thus, it is important to understand and manage uncertainty in all facets of software systems.

One of the key facet of any software systems is its architecture, since it represents the foundation of the software system [11]. Any risks of uncertainty in the software architecture can have significant consequences for the overall system. Therefore, it is important to attempt to anticipate

and mitigate the impact of uncertainty risk on the software architecture and, consequently, the system [12].

This work presents our efforts to develop a generic conceptual framework for considering uncertainty in software architecture, which we have developed in the context of software systems. To the best of our knowledge, currently, no generic uncertainty characterisation framework exists for characterising uncertainty in various facets of software systems such as software architecture.

Uncertainty is a broad concept with multiple dimensions [1, 2, 13–15]. It has been studied from different perspectives in different contexts of software systems, including its architecture, resulting in different approaches to characterising and managing it [1, 2, 12–15]. However, such characterisation approaches are not generic, but target specific domains [12]. Chapter 3, discusses the challenge of expressing uncertainty in software systems.

In our search for the definition of the framework, our first step was to explore uncertainty in software architecture to understand its specific consideration within it. To achieve this, we conduct a survey which provides insight into uncertainty in software architecture. Chapter 4 presents the Uncertainty in Software Architecture Survey.

As previously stated, considering that software architecture is the foundation of any software systems, and that uncertainty is also studied in software systems in general, we further explored uncertainty consideration frameworks in software systems in general, in our effort to develop the generic uncertainty framework. In this regard, we have developed a generic uncertainty framework for software systems, which we evaluate in the context of software architecture.

Thus, this thesis presents an extensible conceptual framework for uncertainty in software systems and its application to software architectures. The framework defines a foundation for systematic and explicit consideration of uncertainty in facets of software system such as software architecture. In this work, the framework is specifically discussed and evaluated in the context of software architecture.

The proposed framework was built by consolidating and extending existing uncertainty concepts and attributes from various domains related to software systems.

In general, this work aims to answer the two research questions presented in Section 1.2. The contributions of the thesis are presented in Section 1.3. The audience of the thesis includes stakeholders in the fields of software systems architecture, such as - analysts, architects, evaluators, etc. The organisation of the rest of the chapters is presented in Section 1.4.

1.2 Research Questions

Research question one guides the definition of the uncertainty framework. Then research question two guides the evaluation of the uncertainty consideration framework through software architecture case studies.

1.2.1 Research Question 1

How can definitions, conceptualisations and characterisations of uncertainty from specific contexts of software systems be consolidated to create a viable generic uncertainty framework which can be used to identify, represent and characterise uncertainty in software architecture?

1.2.2 Research Question 2

How can an uncertainty framework that defines, conceptualises, and characterises uncertainty in software systems be used to identify and represent uncertainty as part of software architecture documentation so that it can contribute to the analysis of known uncertainties in the software architecture?

1.3 Contributions

The following 5 elements are the key contributions of the work:

Contribution 1 A literature survey of uncertainty in software architecture from 1991 to 2021.

This period spans from the inception of software architecture as a formal research discipline in software engineering to the presents (in the contexts of this thesis when we conduct the literature search). The survey provides insight into uncertainty in aspects of software architecture.

Contribution 2 A literature survey of uncertainty frameworks for software systems. We conducted an open literature search in established computer science literature sources databases. The survey explores the existence of frameworks for explicitly considering uncertainty in software systems.

Contribution 3 A definition of a conceptual framework for considering uncertainty in software systems. This is an extensible framework which we apply and evaluate in the contexts of software architecture. However, the framework can be used in other aspects of software development, such as requirements, user stories, and design.

Contribution 4 The specification and development of the infrastructure for building a work-bench of tools for considering uncertainty in software systems, including a prototype tool to capture uncertainty in software architecture design based on the conceptual framework and generate uncertainty data for the system.

Contribution 5 Evaluation of the uncertainty consideration framework on software architecture case studies. We conduct architecture analysis to illustrate the application of the uncertainty framework to uncertainty knowledge generation and documentation, uncertainty architecture analysis, and uncertainty candidate architecture selection.

1.4 Organisation of Dissertation

A short description of the rest of the chapters of the thesis is given below.

Chapter 2 discusses the research methodology, which is split into two phases. First, the research focused on the literature reviews. The results of the literature reviews were then used as a basis to define the uncertainty framework. The second phase of the research focused on the application and evaluation of the uncertainty framework.

Chapter 3 presents, using an illustrative example, the challenge of expressing uncertainty in software system. This provides a background for the definition of the uncertainty framework. The papers *Publications 1 and 2*, and *Submitted to Journal 1 and 2*, cover this chapter.

Chapter 4 presents a survey which explores uncertainty in software architecture. It highlights the aspects where uncertainty is considered in software architecture. The survey identifies and presents a comprehensive list of aspects of software architecture where uncertainty is considered. The systematic literature review explores uncertainty in software architecture in literature sources from 1991 to 2021. The contents of this chapter are cover in the survey paper *Submitted to Journal 2*.

Chapter 5 presents existing literature about uncertainty characterisations in software systems. The chapter reviews uncertainty characterisation approaches to identify relevant concepts and notions for characterising uncertainty. These concepts are then used as a basis for proposing the uncertainty framework in Chapter 6. The contents of this chapter are covered in *Publication 2* and the paper under *Submitted to Journal 1*.

Chapter 6 presents the uncertainty consideration framework. It discusses the rationale for defining the framework, relates the various concepts and attributes of the framework,

and discusses their literature sources. Further, using illustrative examples, this chapter conceptually evaluates the framework through demonstrating how the individual attributes can be used to capture uncertainty in software architecture. The contents of this chapter are covered in *Publication 2* and the paper under *Submitted to Journal 1*.

Chapter 7 discusses the infrastructure for building a workbench for considering uncertainty in software architecture. It includes a prototype tool for capturing uncertainty in software architecture description using the attributes of the framework. The concepts of the workbench introduced in *Publication 1 & 2* and the paper under *Submitted to Journal 1*.

Chapter 8 presents the evaluation strategies of the uncertainty framework, including the use of ATAM. The evaluation is based on three case studies. The evaluation strategy and objectives are discussed for each of the three case studies. The case studies are from AEROSTACK [5], The Information Management Platform for Data Analytics and Aggregation (IMPALA)¹, and Internet of Vehicles (IoV) [6, 7, 16]. An overview of the evaluation approach is discussed in the *Publication 2* and the paper under *Submitted to Journal 1*. Further, as part of the evaluation strategy, we highlight that the framework defined in this research has been applied else where [17], in an independent MSc research project - *A Visual Representation of Uncertainty in Software Systems* [17].

Chapter 9 presents the first case study. This exemplary evaluation uses the uncertainty framework to generate architecture knowledge and document the identified uncertainties of the AEROSTACK architecture. The aim of the exemplary evaluation is to illustrate a fundamental application of the framework, which is to generate uncertainty data.

Chapter 10 presents the second case study, the evaluation of IMPALA. This evaluation aims to illustrate the application of the uncertainty framework data to conduct architecture analysis to assess an architecture design. As a baseline for the validity of the IMPALA analysis results, we conducted a parallel analysis of the IMPALA platform using the established ATAM approach. The results of the two approaches are discussed to assess the insight of the analyses and the feasibility of using the uncertainty framework in conducting architecture evaluation. An overview of the case study is discussed in the the paper under *Submitted to Journal 1*.

Chapter 11 presents the case study of IoV. The aim of this evaluation is to use the uncertainty framework to assist with selecting a candidate architecture from alternative designs, based on its potential capabilities for risk mitigation. Thus, the framework is used to contribute towards addressing one of the familiar challenges in software architecture: *How to select*

¹<https://ntrs.nasa.gov/citations/20160011412>

an implementation architecture from a set of alternative candidate architectures? In the process, we discuss three algorithms for conducting the evaluation basing on the framework data.

Chapter 12 presents the conclusion with a discussion of main contributions, critical reflections and future work, and the conclusion of the thesis.



CHAPTER TWO

METHODOLOGY

The research was organised in two phases. Phase one addressed the first research question. Phase two addressed the second research question. In addition, we proposed an uncertainty consideration workbench infrastructure.

2.1 Scope and approach

Phase one of the methodology involved reviewing the literature on uncertainty in software architecture and software systems. Realising this gap, which we express in research question one, we worked systematically to propose a foundational conceptual framework for considering uncertainty in software systems, which we specifically apply in the context of software architecture.

Although defining the framework was the main goal of phase one, we also conducted systematic literature surveys that provide 1) an overview of uncertainty in aspects of software architecture and 2) reviews uncertainty consideration frameworks in software systems. The results of the two surveys contribute to Chapter 4 and Chapter 5, respectively.

Phase two of the methodology addressed research question two. Application and evaluation of the uncertainty framework in the context of three software architecture case studies. In addition, it discussed the development of tools for incorporating uncertainty in software architecture.

The case studies explored three scenarios: 1) application of the uncertainty framework to generate architecture uncertainty knowledge and documentation. 2) application of the uncertainty framework to conduct an architecture uncertainty analysis, and 3) application of the uncertainty data to select a candidate architecture which is less likely to be vulnerable to the known uncertainties.

In addition, as part of the framework application, we proposed the infrastructure for building a workbench of tools for incorporating uncertainty in software architecture descriptions. A prototype tool is defined, based on the framework, to support the capturing and representation of uncertainty data in the architecture description of the software system.

2.2 Phases and steps

The following are the two phases of the research methodology that address research questions one and two, respectively.

2.2.1 Phase one

Phase one of the methodology addressed research question one, the definition of the framework. The following outline presents the steps we took to define the uncertainty framework.

1. Survey uncertainty in software architecture through a systematic literature survey to understand existing work on uncertainty in software architecture and research gaps.
2. Conduct a literature review on uncertainty frameworks in software systems through a systematic literature survey to understand the definitions, conceptualisations, and characterisations of uncertainty in software systems.
3. Consolidate the results of the literature search to define an uncertainty framework for considering uncertainty in aspects of software systems, such as software architecture.
4. Conceptually evaluate the uncertainty framework through illustrative examples to assess the feasibility of capturing uncertainty details in software architecture.

2.2.2 Phase two

Phase two addressed the second research question - applying and evaluating the uncertainty framework in the context of software architecture on three software architecture case studies. Research work of phase two also involved specifying the infrastructure for developing a workbench of tools for incorporating uncertainty in software architecture descriptions.

Phase two involved selecting the systems to evaluate, identifying uncertainties about the architecture and analysing the results. Below is an overview of the steps of phase two that are discussed in detail in the Evaluation chapter, Chapter 8.

1. Select systems to use as case studies on the basis of availability of the architecture description details, among other factors.
2. Identify, capture and represent the software architecture related uncertainties from the system information sources like documentation and architecture designs details.
3. Analyse the uncertainty data to generate insight about uncertainty in software architecture. The analysis involves exploring risks, mitigations, uncertainty metrics, and other relevant analysis dimensions.
 - Also, checking whether the uncertainty framework allows the capturing and representation of all known uncertainties?
4. In addition to the evaluation of the case studies, we proposed the infrastructure for building a workbench of tools for incorporating uncertainty in software architecture. We present this in Chapter 6.
5. Then we conduct a critical qualitative assessment of the general strengths and weaknesses of the framework.
 - a) What is the significance of the framework? How effective was the application of the framework in achieving the following:
 - i. Capturing, representing, and documenting uncertainty
 - ii. Generating uncertainty data
 - iii. Analysing architecture uncertainty
 - iv. Enabling or facilitating software architecture improvements such as uncertainty mitigations and exploitations
 - We use the term enabling, so we do not suggest that the framework supports automatic recommendation.
 - b) Finally we assess the results and draw some conclusions about the application of the uncertainty framework. Specifically, we discuss the following points:
 - i. Is the framework a feasible approach to the identification and representation of uncertainty in software architecture to positively contribute to uncertainty management?
 - What do we conclude after applying the approach to the case studies?

- ii. What are the aspects of the uncertainty framework that are not possible to implement?
- iii. In general, where did the uncertainty framework succeed and fail?

2.3 Discussion

The first research question focuses on understanding existing work and the challenges of considering uncertainty in software systems and software architecture. The research methodology addresses the first research question in phase one.

Phase one, Section 2.2.1, of the methodology covers the literature search that is the basis for the definition of the uncertainty framework. The key anticipated results for phase one are comprehensive reviews of the literature on uncertainty in software architecture and software systems and the definition of the uncertainty framework. Furthermore, in phase one, we use a conceptual approach to evaluate the feasibility of the framework by demonstrating that the framework can be used to identify and represent uncertainties in software architecture.

Phase two, Section 2.2.2, of the methodology explores the application of the framework and its evaluation in the software architecture. This includes evaluating the usefulness of the framework for managing uncertainty in software architecture. This is achieved by applying the framework to three case studies.

Thus, together, phase one and two in Sections 2.2.1 and 2.2.2 of the methodology address the definition of the uncertainty framework and its evaluation. The detailed evaluation strategy is discussed in Chapter 8.

2.4 Conclusions

The methodology presents the approach to addressing the two research questions. Each research question is addressed separately in one of the two main phases of the methodology. The first phase of the methodology focuses on the literature surveys and the definition of the framework. The second phase then focuses on the application of the framework in the context of software architecture and its evaluation. Finally, conclusions are drawn from the application of the framework.

UNCERTAINTY AND SOFTWARE SYSTEMS BACKGROUND - ILLUSTRATIVE EXAMPLE

3.1 Overview

This chapter presents a background of uncertainty in software systems and the challenge of expressing uncertainty using an illustrative example. Thus, presenting the foundation for defining the uncertainty framework with multiple uncertainty attributes and dimensions.

3.2 Uncertainty and software systems

While formulating his Laws of Software Evolution, Lehman divided software systems into three types: S, P and E [8]. S-type systems can be defined and derived from a specification. P-type systems deal with problems which can be precisely stated, but whose solution depends on an approximation of the real-world. In contrast, E-type systems have neither a precise definition nor a formal specification. Instead, they are real-world systems, inherently prone to changes, and designed to target complex problems, such as automating human or societal activities [8].

Lehman later refined his classification to reflect the vulnerability of the different types of systems to uncertainty [18]. S-type systems are ideal; they operate in a closed context and are thus

effectively isolated from uncertainty. E-type systems are real-world systems and are the most exposed to uncertainty, which can emerge from diverse sources including project management, political environments, operating conditions, supply chains and geographical factors. In this context, P-type systems are subsumed by E-type systems. Indeed, uncertainty is an inherent feature of most real-world software systems.

The development and operation of software systems involve many facets, which can be considered in categories such as processes, artefacts, infrastructure and environments. All such facets are subject to uncertainty, potentially with positive or negative consequences [2, 12, 19, 20]. On the one hand, it may be possible to exploit the positive consequences of uncertainty as opportunities. For example, excess resources from over-estimation during project planning or system design may be re-invested to increase productivity or improve system quality, instead of being treated as waste or losses. On the other hand, it may be necessary to mitigate the adverse effects of uncertainty, such as network disruptions, data loss or security breaches because these can lead to risks to system functionality and performance, among other consequences [2, 10]. Thus, it is important to understand and manage the uncertainty in all facets of software systems.

The scope of the framework we define is uncertainty in general, thus in the context of E-type systems and their facets, such as software architecture. The rest of the chapter is organised as follows. Section 3.3 describes the challenges of expressing uncertainty. Section 3.4 discusses uncertainty in software systems. Section 3.5 is the Conclusion.

3.3 Uncertainty and its expression - illustrative example

The Oxford English Dictionary (OED)¹ defines uncertainty as “the quality of being uncertain in respect of duration, continuance, occurrence, etc.; liability to chance or accident. Also, the quality of being indeterminate as to magnitude or value; the amount of variation in a numerical result that is consistent with observation”.

The multiple dimensions of uncertainty in this definition demonstrate its wide scope and highlight the need for precision and rigour in its treatment. For instance, this definition can apply in the following ways in relation to software systems:

- the uncertainty about duration might relate to the length of time for which a system operates under stress conditions before failing;
- uncertainty about continuance might relate to whether a system continues to operate until or

¹<https://www.oed.com/>

despite a specified event;

- uncertainty about occurrence might relate to an event such as a possible but not guaranteed increase in demand for a service during a particular period;
- uncertainty in terms of chance and accident often relates to the probability or possibility of an event occurring during system operation; and
- uncertainty about numerical variations might be concerned with observed quantities or values in the system such as network bandwidth.

As illustrated through the definition and illustrative examples, uncertainty is a common and familiar concept in both casual and formal settings. Its familiarity makes it prone to ambiguity when used without precision [1].

In casual everyday conversations, people instinctively grasp the essence of the uncertainty when situations or events are referred to as being uncertain. For example, let us assume that Person A and Person B previously agreed to meet at a specific time. Sometime later, Person A is concerned that they may not be able to make the meeting on time. When Person A tells Person B that they may not arrive on time, Person B understands that the essence of the uncertainty is temporal and responds appropriately.

Person B's understanding of Person A's situation may involve considering a number of factors, including the reasons for Person A's uncertainty, possible ways of mitigating the uncertainty, and the consequences of Person A not arriving on time. They can even use Person A's uncertainty regarding arrival time as an opportunity to complete other tasks, since they no longer have to commit to the meeting time agreed initially.

The nature of an uncertainty is critical to understanding it. Person A's uncertainty is due to a lack of knowledge about the exact time they will arrive. As a consequence of this lack of information, Person B is also uncertain from their viewpoint of Person A's arrival time. Given more information, Person A can reduce the uncertainty and then tell Person B the precise, possibly different, time of their arrival, or they can try to find a mitigation against the risk of being late so that they arrives at the originally planned time. This kind of uncertainty depends on available knowledge, and is called *epistemic* uncertainty [21, 22].

In contrast, there is another kind of uncertainty, which is due to inherent variability [21, 22]. Such uncertainty is a feature of the item, event, or thing, instead of being due to the state of knowledge. For instance, when tossing a fair coin, the uncertainty of getting either a head or a tail, in the long run, can be measured using probability as 50% [23]. Statistically, this is simply

the reality of tossing a fair coin; any discrepancy of the odds would mean bias. This kind of uncertainty, which is due to inherent variability or phenomenon, is called *aleatory* uncertainty.

In casual contexts, such as the situation with Person A and Person B, uncertainties can be handled in an ad-hoc manner, without rigorous consideration of their nature and other characteristics. However, in software systems, particularly those used for critical purposes, systematic and rigorous approaches for the consideration of uncertainty are necessary [2, 15].

3.4 Managing uncertainty in software systems

As previously described, uncertainty is an inherent feature of E-type software systems. It can be handled in an ad-hoc manner during the life of a system through the vigilance of stakeholders such as architects, designers, developers and testers [19]. However, as software systems become more ubiquitous and decoupled, and are deployed in a variety of fast-changing environments, more systematic approaches to handling uncertainty are needed [24].

Existing approaches typically address uncertainty in specific facets of software systems or target particular challenges. Examples include addressing uncertainty through requirements management in self-adaptive systems [25], uncertainty management in early architecture decisions [26], evaluation of software architecture under uncertainty [27], classification of uncertainty in adaptive systems with multiple quality requirements [21] and understanding uncertainty in cyber-physical systems through a conceptual model [3]. Mathematical approaches, such as probability, can be used to measure specific uncertainties in software systems.


One consequence of this specialisation is that there is no foundational approach to the explicit and systematic consideration and analysis of uncertainty for software systems in general. As decoupled E-type systems become more common, it is likely that domain-specific frameworks are not sufficient to address the risks of uncertainties that arise at different stages of the system lifecycle. The focus of this work is a more comprehensive treatment of uncertainty that subsumes mathematical approaches. The inclusion of additional attributes such as the nature and location of an uncertainty will allow users to not only identify and measure it but also devise mitigations against its risks. Our approach aims to help software engineers *identify, represent and manage* uncertainty.

- Identification refers to how uncertainty is recognised in the different facets or aspects of a system;
- Representation refers to how details of uncertainties are captured and modelled in system artefacts; and

- Management refers to the process of analysing uncertainty and exploiting or mitigating its risks in a system.

3.5 Conclusion

This work contributes towards addressing the lack of a generic foundational and customisable framework for the consideration of uncertainty in software systems. We hypothesise that such a framework will help software engineers determine the consequent risks and possible mitigations more accurately.



CHAPTER FOUR

UNCERTAINTY IN SOFTWARE ARCHITECTURE: A SURVEY

Researchers have long recognised the influence of uncertainty on different aspects of software architecture. Uncertainty in some aspects of software architecture, such as architecture design and evaluation, has typically generated more research interest and results than in others. This chapter presents a systematic survey of uncertainty in software architecture aspects, focusing on the period between 1991 to 2021 inclusive. It identifies twenty five (25) aspects of software architecture with existing work on uncertainty. The survey discusses existing work on uncertainty in these software architecture aspects, provides insights into the state of the art, and suggests potential directions for future work in addressing uncertainty in software architecture.

4.1 Overview

Since its formalisation as a research area in the 1990s, the software architecture of a system has been defined in various ways [28, 29]. Examples include definitions that treat software architecture as the high-level structure and behaviour of the system, the key design decisions about the system, and the configuration of the system in terms of components linked by connectors based on rationale [29–31]. In essence, all such definitions agree that software architecture represents the foundation of a software system [28].

Consequently, risks from uncertainty in software architecture, such as those from unanticipated changes, incomplete documentation, or variability in the operating environment that affect architectural assumptions, can pose a threat to the quality of the system and lead to high costs [2]. These costs can be in terms of finance [32], performance [33], reliability [34], legal issues [35] and business disruptions [9]. Therefore, uncertainty in software architecture should be identified and avoided, or at least its risks mitigated, as much as possible [2].

The aim of this survey is to review existing work on uncertainty in software architecture. The survey is novel in recognising that since software architecture is multifaceted, a comprehensive review of uncertainty in software architecture will also be multifaceted. Thus, the survey analyses uncertainty in individual aspects of software architecture to build up a comprehensive overview of uncertainty in software architecture [36, 37].

The process of defining, using and maintaining a software architecture includes various activities such as design and evaluation. These activities generate artefacts, such as documentation, and require tools to support the activities and notations to represent the outcomes of the activities. All these are examples of aspects of software architecture.

Understanding uncertainty in software architecture aspects can be crucial to minimising its negative consequences [26, 38]. For instance, knowledge of uncertainties during software architecture design can help architects invest in appropriate mitigations, learning cycles and feedback loops in agile development to manage the uncertainty [39]. This survey chapter explores existing work on uncertainty in all aspects of software architecture found in literature.

The remainder of this chapter is structured as follows. Section 4.2 outlines related work in the form of other surveys with similar motivation. Section 4.3 describes the design of the survey, while Section 4.4 presents the survey process. Section 4.5 shares the survey data. Section 4.6 provides an overview of the survey results. Section 4.7 categorises the aspects of software architecture based on the survey results. Section 4.8 discusses the uncertainty in the identified aspects, according to their respective categories. Section 4.9 discusses uncertainty risks and opportunities in the context of software architecture aspects. Section 4.10 presents an overview of the mitigation and exploitation approaches of uncertainty in software architecture aspects. Section 4.11 provides a discussion of the general findings, including the strengths and limitations of existing approaches and future work. Finally, some conclusions are given in Section 4.12.

4.2 Related work

We are not aware of any existing work which reviews uncertainty in diverse software architecture aspects. The two closest outputs we have identified are: 1) a literature review by Mahdavi-Hezaveh et al. [21], in which the authors explore uncertainty in architecture-based self-adaptive systems with multiple requirements and define a classification, and 2) a systematic literature review by Sobhy et al. [40] in which the authors explore uncertainty in the context of software architecture evaluation.

In contrast to these works, this chapter reviews uncertainty in diverse aspects of the software architecture of software systems in general and attempts to build a holistic picture of how uncertainty is currently addressed in software architecture. We empirically identify the aspects of software architecture through reviewing literature sources. Aspects thus identified include *Activities, Artefacts, Tools and Notations*.

Other researchers have pointed out the need for software engineering to consider uncertainty [19] and explored the understanding of uncertainty from a self-adaptive perspective [41]. Our work contributes towards such objectives with a focus on exploring uncertainty in aspects of software architectures.

4.3 Survey design

The objectives of this survey were to identify literature sources related to uncertainty in software architecture and to analyse the data from these sources to identify aspects of software architecture that have been considered, their relationship to uncertainty and the approaches that have been proposed to manage uncertainty in these aspects. Literature sources were identified through the following two main steps.

The first step was an automated search of six databases (IET Digital Library¹, Web of Science², DBLP³, Scopus⁴, ACM Library⁵, and IEEE Xplore⁶). These databases are among those regarded as reliable literature data sources in Computer Science [42, 43]. This range of databases increase the thoroughness and likelihood of finding relevant literature sources.

The search was made through two queries, one with filters and the other with wildcards. We

¹<https://digital-library.theiet.org/>, last accessed on 08 April, 2022

²<https://apps.webofknowledge.com>, last accessed on 08 April, 2022

³<https://dblp.org/>, last accessed on 08 April, 2022

⁴<https://www.scopus.com>, last accessed on 08 April, 2022

⁵<https://dl.acm.org/>, last accessed on 08 April, 2022

⁶<https://ieeexplore.ieee.org>, last accessed on 08 April, 2022

compared the results of these two queries as one way of demonstrating the comprehensiveness and validity of the systematic search. The key words in the queries are based on the survey objectives and questions (discussed in Section 4.3.2).

1. `uncertainty AND "software architecture" AND published between 1991 and 2021`
2. `uncertainty AND "software architecture" AND (language OR capture OR document OR model OR description OR stakeholder OR representation) AND published between 1991 and 2021`

The second step was a manual filtering process. This was conducted through reading the literature sources identified in the first step. During this process, we identified other relevant literature sources from references in the original set. Section 4.3.2 presents the survey questions and Section 4.3.3 covers inclusion and exclusion criteria.

4.3.1 Identifying aspects of software architecture

In the context of the survey chapter, we define architectural aspects as specific areas in which uncertainty is considered or discussed in the identified literature sources related to software architecture. For example, uncertainty might be discussed in relation to architecture requirements, documentation or evaluation. All these are specific aspects.

However, since there is no established list of aspects where uncertainty is studied in software architecture, the survey uses a manual process to identify such aspects. The author read and analysed the literature sources to identify the relevant aspects.

Query 1 uses a wildcard in its search, and thus potentially identifies most of the literature sources which discuss uncertainty and software architecture. Query 2 includes a subset of some familiar terms or words in software architecture, as potential aspects, and thus is likely to generate a subset of the results of Query 1.

We can make the list of terms in Query 2 as long as possible to refine the search and minimise the effort for manual search. However, this will search for only known aspects. Query 1 identifies most results because it includes both the aspects that are known to us and those unknown to us, by definition of using the wildcard. Therefore, as part of validation test of the search results, we expected that all the literature sources identified by Query 2, should be in Query 1 results. And this was generally the case. Section 4.3.3 further elaborates on the inclusion and exclusion criteria of the literature sources and the aspects

4.3.2 Survey questions

To guide the review, we defined a set of questions to achieve our objectives, focusing on uncertainty, sources of uncertainty, aspects of software architecture, the impact of uncertainty on software architecture and the representation and mitigation of uncertainty risks in software architecture, thus exploring uncertainty in software architecture aspects [2, 12].

The questions are in two categories – specific and general. Specific questions address individual concerns relating to uncertainty in software architecture. The general question aims to provide broad insights into uncertainty in software architecture, and thus is addressed throughout the chapter and particularly in Section 4.11.

These questions are partly based on the framework by McManus and Hastings [2] for understanding uncertainty in complex systems, which states that uncertainty causes risks or opportunities which are handled by mitigations or exploitations to, ideally, result in the desired outcomes.

Specific questions:

1. What are the sources of uncertainty in software architecture?
2. How is uncertainty represented in software architecture?
3. What are the specific aspects of software architecture where uncertainty research is focused?
4. To what extent does the uncertainty research in software architecture cover these aspects?
5. What are the risks and opportunities of uncertainty at the software architecture level?
6. How are the risks and opportunities of uncertainty mitigated and exploited in software architecture?

General question:

7. How has the study of uncertainty evolved in the last 30 years and to what extent is uncertainty addressed in software architecture and its aspects?

4.3.3 Inclusion and exclusion criteria

Software architecture emerged as a research discipline in the 1990s [29]. To cover the earliest work in the area, the survey starts the literature search from 1991 and continues up to and including 2021, as the first step of data collection. Table 4.1 shows a summary of the search

Table 4.1: Databases search results from 1991-2021

	Database Name	Results (Query 1)	Results (Query 2)	Relevant (Query 1)	Relevant (Query 2)
1	IET Digital Library	87	86	4	6
2	Web of Science	170	120	22	15
3	DBLP	12	0	10	0
4	Scopus	322	194	36	20
5	ACM Library	904	902	16	14
6	IEEE Xplore	225	155	19	13

results. We identify pertinent literature sources by manually reviewing the relevant database search results, and following up their relevant references to ensure that we identify other literature sources which might be linked to uncertainty and software architecture.

Query 1 identifies literature sources with terms "software architecture" and "uncertainty" between 1991 and 2021. Query 2 adds context to the search string through the inclusion of key words about familiar terms in software architecture: language, capture, document, model, description, stakeholder and representation. *software architecture* and *uncertainty* are the minimum terms to identify relevant literature sources. For the additional key words, *language* comes from software architecture description language, *model* from architecture models, *description* from architecture description, *stakeholders* are participants in the software architecting process, and so on.

On the one hand, Query 1 uses the key search terms without further constraints, thus implicitly including all potential aspects of software architecture related to architecture requirements, activities, artefacts, tools, notations and others. In Section 4.7 of the results, we provide an analysis of the categories of software architecture aspects.

On the other hand, Query 2 explicitly includes familiar specific terms of software architecture and filters the results. In addition, we used the following variants of the word uncertainty together with software architecture: *unpredictability*, *indeterminate*, *variability* and *unreliability*. However, using these synonyms did not find additional relevant literature sources. Most literature sources discussing uncertainty use the term *uncertainty* and not its variants.

In fact, we were more likely to find literature sources which include the terms *uncertainty* and *architecture* but are not necessarily within the scope of this survey as shown in Table 4.1. Therefore it is reasonable to only consider *uncertainty* and exclude its synonyms.

4.4 Survey process

4.4.1 Data collection and trends

As previously stated, we used data from six academic databases, executing Queries 1 and 2 on them with last access on 08 April, 2022. For each database, Table 4.1 shows the results of each query and the total relevant literature sources found through manual review. Executing these queries through the search feature of each database should generate similar results to the survey results column in Table 4.1. Below are the exact queries applied to each of the databases:

IET Digital library Queries:

1. from all fields including fulltext for 'uncertainty AND "software architecture"' published between 1991 and 2021
2. from all fields including fulltext for 'uncertainty AND "software architecture" AND (language OR capture OR document OR model OR description OR stakeholder OR representation)' published between 1991 and 2021

Web of Science Queries:

1. ALL=(uncertainty AND "software architecture")
2. ALL=(uncertainty AND "software architecture" AND (language OR capture OR document OR model OR description OR stakeholder OR representation)) Timespan: 1991-01-01 to 2021-12-31 (Publication Date)

DBLP Queries:

1. uncertainty AND "software architecture"
2. uncertainty AND "software architecture" AND (language OR capture OR document OR model OR description OR stakeholder OR representation)

Scopus Queries:

1. TITLE-ABS-KEY (uncertainty AND "software architecture") AND PUBYEAR > 1990 AND PUBYEAR < 2022

2. (TITLE-ABS-KEY (uncertainty AND "software architecture") AND TITLE-ABS-KEY ((language OR capture OR document OR model OR description OR stakeholder OR representation))) AND PUBYEAR > 1990 AND PUBYEAR < 2022

ACM Library Queries:

1. [All: uncertainty] AND [All: "software architecture"] AND [Publication Date: (01/01/1991 TO 31/12/2021)]
2. [All: uncertainty], AND [[All: "software architecture"] OR [All: uncertainty]] AND [All: "software architecture"] AND [[All: language] OR [All: capture] OR [All: document] OR [All: model] OR [All: description] OR [All: stakeholder] OR [All: representation]] AND [Publication Date: (01/01/1991 TO 31/12/2021)]

IEEE Xplore Queries:

1. ("All Metadata":uncertainty AND "All Metadata":"software architecture") Filters Applied: 1991 - 2021
2. ("All Metadata":uncertainty AND "All Metadata":"software architecture") AND ("All Metadata":language OR "All Metadata":capture OR "All Metadata":document OR "All Metadata":model OR "All Metadata":description OR "All Metadata":stakeholder OR "All Metadata":representation) Filters Applied: 1991 - 2021

Figure 4.1 shows an overview of the final data in terms of literature sources per year. Overall, the trend in Figure 4.1 shows a positive growth in the distribution of references per year. This trend suggests a rising interest in research or discussion of uncertainty in software architecture, particularly from 2011.

After the manual review, we identified eighty two (82) relevant literature sources in total. Table A.1 provides an overview of literature sources for each aspect we identified.

There are relatively few sources between 1991 to 1997, averaging around one per year with exceptions as shown in the chart. The low number of literature sources during this period suggests

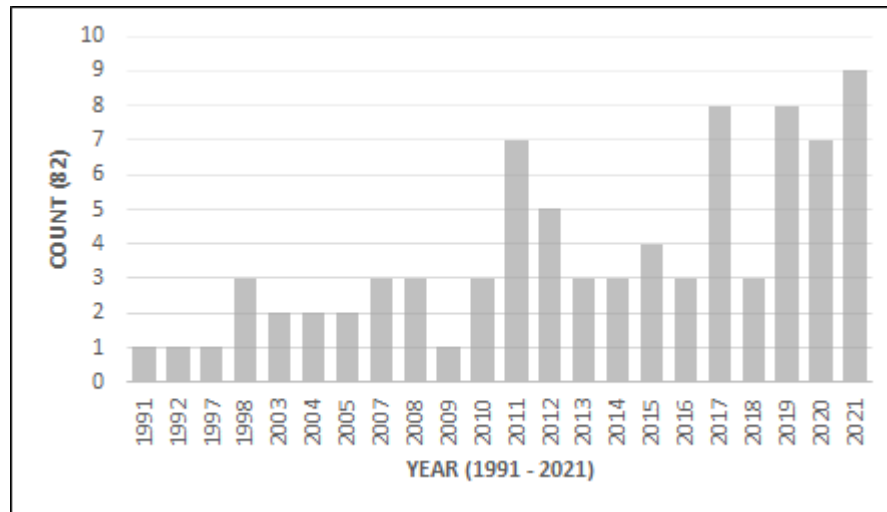


Figure 4.1: Distribution of literature sources per year from 1991 - 2021

that it was uncommon for uncertainty and software architecture to be explicitly associated at the time.

Indeed, we note that most of the search results during this period were related to the establishment of the study of software architecture. Therefore, they focus on the concept of software architecture and stages of architecture process such as design, analysis, evaluation and modelling using architecture description languages [28, 29].

Many of the sources from 1998 to 2004 discuss uncertainty in architecture analysis and evaluation approaches and architecture requirements [44–48]. From 2004 to 2021, we see discussions about uncertainty in diverse software architecture aspects including architecture design methodologies, run-time system architecture adaptation, analysis of uncertainty in architecture artefacts, communication of architectural uncertainty among stakeholders and conceptualisation of new approaches to designing software architecture of specific applications or system domains such as Systems-of-Systems (SoS).

4.4.2 Limitations and threats to validity

While our objective was to be as thorough as possible in identifying and analysing relevant literature sources with the available resources, we are aware of limitations and threats that can impact the validity of the survey. They relate to bias and domain of study. To enable replication of the results of the analysis, Section 4.5 includes a description of the repository of the data of references that we analysed to produce the insights presented in this survey.

Table 4.2: Coverage of aspects in literature sources

Aspects	Count	Coverage
Definitions and Concepts		
1 Classification	9	11%
2 General	13	16%
3 Systems of systems	6	7%
4 Internet of Things	3	4%
5 Knowledge	3	4%
6 SE Methodology	3	4%
7 Performance	4	5%
8 Reliability	5	6%
9 Stability	3	4%
10 Risk	3	4%
Activities		
11 Communication	1	1%
12 Decisions & Tradeoffs	13	16%
13 Analysis	17	21%
14 Deployment	2	2%
15 Design	26	32%
16 Evaluation	18	22%
17 Evolution	2	2%
18 Run-time	21	26%
19 Self-Adaptation	21	26%
Artefacts		
20 Description	11	13%
21 Documentation	6	7%
22 Requirements	8	10%
23 Traceability	2	2%
Tools and Notations		
24 Tools	21	26%
25 Notation	21	26%
Number of literature sources from 1991 - 2021	82	

4.4.2.1 Bias

The literature search, identification of relevant literature sources and interpretation of the results were carried out by the author. Thus, the survey may be biased towards their perceived notions of uncertainty and software architecture. Ideally, at least a second person would have also done an independent literature selection to compare with the current results to minimise the potential effect of bias.

The use of a systematic process is likely to have mitigated the potential personal biases. In addition, the literature sources were identified from six reputable Computer Science literature databases, using a systematic approach to increase the likelihood that search results and selection are independent and comprehensive. The details of these queries and selection are provided in the Data Collection Section 4.4.1.

4.4.2.2 Domain of study

Even though uncertainty and software architecture are associated with each other in literature sources, the exact aspect of software architecture is sometimes not explicitly stated or there are multiple aspects affected by the uncertainty.

This is a challenge for the survey since we analyse and collate literature sources which might not be explicitly related. Therefore, some of the literature sources we have reviewed may be classified in ways that are broader or more specific than intended by the original authors. Nevertheless, we have focused on discussing software architecture aspects which are acknowledged or can be justified by the literature sources and we provide references to sources for further information, in addition to our interpretation. This should minimise the threat of misinterpretation.

Further, the terminology used in the survey, such as uncertainty, software architecture and various aspects of software architecture, are well-established but they lack standardised and precise definitions.

Uncertainty is defined in a number of ways. Similarly, software architecture and its constituents do not have universal definitions [49]. We have cited a number of sources to highlight the diversity of definitions and meanings, and specified our interpretation of concepts in the context of this survey.

4.5 Survey Data

To promote the validity and reproducibility of the analysis and insights presented in this chapter, we include a repository⁷ of the raw data which we analysed to gain insight into the uncertainty aspects. Each row of the data represents an individual literature source. There are a total of eighty two (82) rows and one (1) header row.

Each row includes the following details: the year of publication of the literature source (year), the reference ID, the architecture aspect(s) in which uncertainty is considered (Aspects), name of the tool or notation if the literature source contains one, the uncertainty context, the mitigation strategy, the desired outcome, the nature of the uncertainty considered, and the full reference details.

We use these details to analyse and identify the uncertainty aspects, show the trend of uncertainty consideration in software architecture and generate the graphs and tables included in this survey chapter. As the data is empirically generated, it is not perfect. It contains some null values, some of the data is free text notes without format, and the identified architecture aspects might not be exhaustive.

4.6 Overview of software architecture aspects

As stated in Section 4.3.1, we manually analysed each literature source to identify the specific architecture aspect(s) in which uncertainty is considered. Table A.1 shows the individual literature sources and their aspects where uncertainty is studied in software architecture.

In total, we identified twenty five (25) unique aspects through this analysis. Table 4.2 presents a summary of the aspects and their associated references.

Since multiple aspects can appear within a single literature source, each literature source is mapped to its aspects, as shown in Table A.1. This mapping was a manual process, which involved reading a literature source and tagging it with relevant architecture aspects.

Many of the software architecture aspects identified in Tables 4.2 and A.1, such as architecture requirements, self-adaptation and deployment, are reasonably well-known terms in software architecture. In addition to these, we identify a few less established aspects such as communication, classification, general and Systems of Systems, which require clarification. Therefore, Section 4.7 provides a description of the individual aspects and their categorisation. After that, Sections 4.8 to 4.10 present answers to the survey questions introduced in Section 4.3.2.

⁷<https://bit.ly/3z4WNWv>, last accessed on 25 July, 2022

4.7 Categorisation of software architecture aspects

Table 4.2 shows the extent of uncertainty consideration in software architecture aspects through their coverage in the literature sources. The coverage is the percentage of literature sources focusing on an aspect of software architecture out of the total 82 literature sources.

Table A.1 shows a more detailed analysis of each literature source. The initial set of the 25 architecture aspects were distinct items, without an explicit relationship with each other. Thus, they were open to various categorisations, depending on the basis of the categories.

Defining a classification or categories of items, things or concepts in information systems is a challenging problem due to, among other reasons, the process being subjective and empirical [50]. Our approach to defining the categories of aspects is based on identifying similarities or common elements of the architecture aspects, which are used as the basis for the categories [50]. As such, the categorisation is from empirical to conceptual [50].

Ideally, when conducting a classification, besides identifying common elements as the basis for classification, its important to get external input from others, conduct discussions, and go through a number of interactions to improve the classification and minimise the subjective influence [50].

In this case, the categories were defined among the two authors. However, we base the classification on existing software architecture lifecycle terminology, with the exception of *Concepts and Definition* category which is generic and miscellaneous [49].

In this survey, we propose the following five categories of the architecture aspects: architectural *Definitions and Concepts*; architectural *Activities*; architectural *Artefacts*; and architectural *Tools and notations*. Below is a description of each of these categories of architectural aspects, the aspects contained in the category, and the justification of the category.

The aspects in the *Definitions and Concepts* category aim to present different perspectives or approaches to managing uncertainty in software architecture. This category is miscellaneous as it includes various aspects which do not fit into the other specific categories. However, within this category, we organise the individual aspects into related sub categories as discussed below. These include:

- Classification of uncertainty in software architecture (Classification)
- General discussion about uncertainty and software architecture (General)
- Connected systems (Cd-Sys) such as Internet of Things (IoT) and SoS, in which architecting in the presence of uncertainty may require novel approaches

- The conceptualisation of architecture knowledge under uncertainty (Knowledge)
- The conceptualisation of uncertainty and software engineering methodologies such as Agile and their resulting impact on software architecture (Methodology)
- System qualities (Sys-Q) of software architecture such as performance, reliability and stability, which may be affected by uncertainty
- Risks arising from uncertainty in software architecture thus threatening the foundation of a system (Risk).

The category of *Architecture Activities* includes the following:

- Uncertainty arising in architecture communications such as discussions about architecture design decisions among stakeholders (Communication)
- Decision-making and trade-offs in the presence of uncertainty (Decisions)
- Uncertainty in software architecture lifecycle stages (SA-LC Stages) such as design, evaluation, deployment and evolution. This category includes activities relating to managing architecture at run-time and in self-adaptive systems, such as the evaluation of self-adaptive policies at run-time. The list is as follows:
 - Trade-offs
 - Analysis
 - Deployment
 - Design
 - Evaluation
 - Evolution
 - Run-time
 - Self-adaptation

The activities identified above can be conducted by architecture stakeholders, the system or both. For instance, activities related to self-adaptation at run-time will be carried out by the system while design decisions during development are made by stakeholders.

The category of *Architecture Artefacts* includes the set of tangible products or by-products of the architecting process. These include:

- Uncertainty in architecture description (Architecture description)
- Uncertainty in architecture documentation (Documentation)
- Uncertainty related to architecturally significant requirements (Requirements)
- Uncertainty about architecture traceability such as among architecture models and other artefacts (Traceability)

Although the Architecture description and Documentation aspects might appear similar, we recognise them as distinct aspects since architecture description approaches such as Description Languages (ADLs) focus on the representation of the architecture models, while documentation is more general with items such as business documents which include contracts or stakeholder communications, among other relevant architecture related documents.

The *Tools and Notations* category includes aspects which facilitate the architecting process. Architecture processes require tools to support functionality such as diagramming, traceability and visualisation. Similarly, software architecture may be represented using various notations. Some of these, such as ADLs with precisely-defined syntax and semantics, are formal, some, such as Unified Modeling Language (UML) diagrams, are semi-formal, and others, such as boxes-and-lines diagrams, are casual but often widely adopted in practice.

The aspects in this category focus on the capabilities of architecting tools and notations to represent uncertainty in software architecture. This category is somewhat different from the others in the sense that uncertainty may not directly affect them and we are concerned with their capabilities to model and analyse uncertainty.

In Section 4.8.4, the survey presents and discusses the tools and notations identified for working with uncertainty in software architecture. It also specifies the specific aspects in which the tools and notations are applied to support the management of uncertainty.

A key challenge in categorising architecture aspects relates to the *Definitions and Concepts* category, in which there is a collection of architecture quality attributes such as reliability and stability, architecture application domains such as IoT or SoS, architecture development methodology and architecture risk among others, as shown in Table 4.2. We divide them into specific subcategories for ease of understanding. This was deemed a suitable compromise in presenting an overview of the topic.

From Table 4.2, we note that most of the discussion about uncertainty in software architecture relates to Design, with a coverage of 32%. The least covered aspects are *Activities - Deployment and Evolution* and *Artefacts - traceability*, both at 2% coverage.

4.8 Uncertainty in software architecture aspects

Uncertainty in software architecture emerges from various sources and influences diverse architecture aspects. In this section we consider uncertainty in the four categories of aspects introduced in Section 4.7, based on the results of our survey. For each aspect, an overview of the relevant literature sources and specific uncertainty details are provided. For each category, the discussion is through presenting an overview of literature sources, thus presenting a justification of our identification of the aspect.

4.8.1 Architecture Definitions and Concepts

4.8.1.1 General

The general aspect highlights literature sources which are concerned with discussions of uncertainty in software architecture and systems, in the general sense. Uncertainty in software architecture and software engineering is often discussed in a general context to advocate and highlight that software systems should consider uncertainty as a first-class concern [19]. Various research results have advocated for a paradigm shift in software engineering, systems and architecture to incorporate uncertainty [19, 20, 40, 51]. Software practitioners and researchers have long recognised the threats of uncertainty in the software development process, which can result in bugs, delays in projects, high costs and other negative implications to the delivery of quality software [52].

4.8.1.2 Classification

The classification aspect relates to understanding, considering and interpreting uncertainty in various contexts of software architecture [12]. Uncertainty classifications can be identified within literature sources discussing uncertainty in specific domains or in relation to software architecture [9, 12, 15, 21]. For instance, uncertainty classification is discussed in self-adaptive systems, including in relation to their architecture [14, 21, 22]. Classification of uncertainty is also discussed in specific system domains such as complex systems [9] and then adopted for application in software architecture [53].

Uncertainty can be classified as aleatory or epistemic [2, 53]. However, software architecture requires a more specific and specialised classification [21].

Uncertainty has also been classified according to its risk mitigations: *whether the uncertainty can be reduced or is irreducible [2]. In self-adaptive system requirements gathering, uncertainty can be classified as being based on ambiguity or false assumption; under design, uncertainty can be about unexplored alternatives or untraceable design; and under run-time, uncertainty can be about the operating environments, or relate to various system components such as sensor failure and sensors noise [14, 22]. In a general model, uncertainty can be classified in terms of dimensions such as location, level and nature [15].* The classification framework of uncertainty in architecture-based self-adaptive systems with multiple quality requirements considers similar dimensions of uncertainty but with specific options for self-adaptive systems [21].

There is currently no uniform approach to classifying uncertainty in software architecture, although there are efforts to explore this topic [12]. Uncertainty classification in software architecture is an ongoing research area with open discussions on the definitions, conceptualisations and classification of uncertainty [12, 21].

4.8.1.3 Connected systems - SoS

The SoS aspect considers uncertainty in the software architecture of systems of systems. Such systems are common in the domains of IoT and Cyber-Physical Systems (CPS) [54–56].

The SoS software architecture composition is prone to epistemic uncertainty because some of the constituent elements of their architecture cannot be determined at design time, but can only be realised at run-time [57]. Since software architecture may be used as the foundation of the composition of a system, epistemic uncertainty during design time raises the challenge of how to design the architecture of a system whose actual architecture elements and configuration are dynamic and can only be adequately known in the future, for instance, during runtime [54, 57, 58]. Therefore, instead of designing a traditional architecture, the SoS design approach uses an abstract architecture, which serves as the template for the real architecture that can only be fully realised at runtime [58].

Research on handling uncertainty in SoS architecture includes the definition of SoS architecture description languages such as Systems-of-Systems Architecture Description Language (SoADL) to formally describe the SoS software architecture [54]. In addition, other approaches address uncertainty in SoS through tools to support architecture decision making during its design under uncertainty. Examples of these tools include a fuzzy logic expert system to support architecture design in terms of documentation and decision making through reuse of knowledge

of recurring decisions [59] and a tool called Ark which supports a constraint-based method for architecture synthesis of a smart SoS [58].

4.8.1.4 Connected systems - IoT

In terms of the literature sources identified during the survey, the IoT aspect is closely related to SoS. There are some literature sources with exclusive focus on each while others are shared [54, 55]. An example of a shared research area is a fuzzy architecture description for handling uncertainty in IoT Systems-of-Systems [55]. This extends SoADL with uncertainty capability to support IoT systems [55]. Uncertainties in IoT can emerge from the operating environment, loose coupling of system components, deployment, security threats and user interactions [24, 55, 60]. Thus, software engineering for IoT requires a paradigm-shift from traditional approaches to handle the inherent uncertainty [24]. Research questions in IoT include how to define software architectures in the context of ubiquitous uncertainties in IoT and how to control or handle the uncertainty in the software architecture of IoT [24]. Further, IoT introduces the challenge of software deployment under uncertainty [60]. One proposed solution for deployment is the goal-driven approach for deploying self-adaptive IoT systems [60].

4.8.1.5 Knowledge

Software architecture knowledge is generated and documented throughout the lifetime of a system and its architecture [11, 49]. Sources of uncertainty in architecture knowledge include the state of the knowledge itself, such as knowledge availability, validity, completeness, quality and accuracy [35, 39]. Uncertainty in knowledge impacts architecture design as well as other architecture aspects [26, 35, 39, 61]. For example, design decisions such as deciding on elements of a system to out-source [35] can be affected by uncertainty in knowledge. In addition, the requirements may be unknown at the beginning of a project, yet legal agreements must be made and costs estimated [35].

4.8.1.6 Methodology

Software systems are often developed in an environment where there is uncertainty due to lack of knowledge on the software's structure, behaviour and execution context, which need to be represented as part of the software architecture [24]. This presents a challenge to software development methodologies relating on how to strategically handle uncertainty.

A specific challenge, in terms of methodologies, is the relationship between software architecture and the Agile software development methodology [62]. By definition, software architecture is the foundation of the software system, and thus, it may be hard to change later. However

the Agile methodology encourages continuous iterative development of software and embraces change. This contradiction needs to be handled [11, 28, 62].

Addressing the relationship between the Agile methodology and software architecture is one of the research topics relevant to handling uncertainty in software architecture. One practical approach to handling this is to design an agile architecture so that it is modifiable and change-tolerant [62]. This approach is based on the following proposed five guiding tactics to building an agile architecture: keep designs simple, improve the architecture code iteratively, use good design practices, delay decision making and plan for options [62].

4.8.1.7 System quality - Performance

Performance is a software architecture quality attribute. Uncertainty about performance in software architecture arises in relation to architecture analysis and evaluation activities since performance is one of the key quality attribute for assessing the suitability of a candidate architecture [30, 46]. A specific example of research on uncertainty and software architecture performance is in the performance analysis of architecture models under uncertainty [33, 63]. Performance results of a system can be analysed and traced to specific architecture components of a system to understand the relationship between the performance of a system and its architecture model [64]. However, there is often uncertainty in precisely identifying such a relationship, and, as such, fuzzy approaches are required to implement and handle the mapping [64].

4.8.1.8 System quality - Reliability

Similar to performance, reliability is a quality attribute which is significant to architecture analysis and evaluation [30]. Uncertainties relating to the reliability of architecture components can result in reliability issues for the overall system [65, 66]. Evaluating architectures through early predictions of software component reliability is vulnerable to uncertainty [66]. The lack of information about architecture components during the early phases of software architecture development makes it hard to assess the reliability of an architecture [65, 66]. The lack of data on intended behaviour of an architectural component, its operational profile, implementation details and failure behaviour can make the architecture reliability assessment process unreliable [65, 66]. The accuracy of architecture-based reliability evaluations depends on a number of parameters that need to be estimated, such as environmental factors or system usage [34, 67]. Techniques used to mitigate the risks from lack of reliability data include estimating components parameters [68] and modeling and simulation to handle uncertainty in architectural parameters [67].

4.8.1.9 System quality - Stability

Stability is another quality attribute of software architecture. Uncertainty in the context of stability arises from techniques for predicting the stability of an architecture. Modelling techniques such as the use of a model based on the theory of real options to predict the stability of software architectures have been explored [47, 69]. The uncertainty in predicting architecture stability is related both to the accuracy of the prediction functions and to the input into the prediction or analysis function [47].

4.8.1.10 Risks

This aspect relates to risk in architectures due to uncertainty. Risk is a direct consequences of uncertainty [2]. "Architectural risk, intuitively, is the degree to which the performance of a design is fragile in the face of unknowns" [70]. Risk analysis aims to estimate and understand architectural risk from uncertainty [70]. There are a range of sources of risks in software including project uncertainty, process uncertainty and design uncertainty [70]. Software architecture design in the context of rapidly evolving technologies involves dealing with the risk that a design will fail to meet its performance objectives [65, 66, 70].

4.8.2 Architecture Activities

4.8.2.1 Communication

Uncertainty in communication arises during the exchanging of architecture information through speaking, writing or using some other medium. Sources of communication uncertainty include language use and interpretation [61, 71]. In terms of verbal or written language, uncertainty may arise from the meaning of uncertain language key words such as *may*, *might* and *maybe* in sentences which can signal uncertainty [71]. Besides, the presence of such uncertainty key words in architecture documentation and presentations can signal feedback, preference, opinion, reassurance or figure of speech [71]. For instance, during an architecture presentation to stakeholders, the presenter might use uncertain language key words to solicit feedback or opinion [71]. In addition to language use, uncertainties in communication arise within architecture documents due to missing information, assumptions or ambiguities [61].

4.8.2.2 Decisions

There are many areas where decisions are vulnerable to uncertainty in software architecture [72]. These include uncertainty in documenting architecture decisions, uncertainty in domain specific architectural decisions (such as mobile systems), uncertainty in decisions that aim to

achieve specific quality attributes (reliability or scalability), and uncertainty in making decisions and grouping architecture decisions [42, 72, 73]. During architecture design and evaluation, uncertainty may emerge while selecting implementation architectures among alternatives due to lack of information; for example, there may be uncertainty about the implications of a decision [39, 72–74]. Uncertainty in decision making can be handled through its reduction by gathering knowledge through modelling, simulation and testing [74]. Software development methodologies can support decision making under uncertainty through feedback loops on architecture decisions, thus enabling the progressive maturity of the architectural knowledge base [39, 75].

4.8.2.3 Trade-offs

Trade-offs are a part of the essence of architecture design and are closely related to architecture decisions and analysis [76]. Trade-offs are prone to uncertainty from lack of knowledge during design, evaluation and architecture execution in the context of self-adaptive systems [48, 77, 78]. There is need for tools and techniques to facilitate the management of trade-offs in software architecture [78]. Recent work on a related topic considers the feasibility of the use of a machine learning approach to explain the architecture trade-off design space in the context of uncertainty [76].

4.8.2.4 Analysis

Analysis is one of the core activities in software architecture. Uncertainty in analysis may emerge from its techniques and methods, such as quantification of values or approximation, as well as stakeholder input into the analysis process among other aspects [26, 30, 44, 48].

We have previously discussed uncertainty in analysis through other aspects such as reliability [65, 68], trade-off [48, 76–78], stability [79] and performance [33]. Besides, analysis is discussed in literature in the context of software architecture and security in terms of confidentiality [80], and uncertainty in analysis of architecture during run-time [44, 45].

4.8.2.5 Deployment

Deployment uncertainty in architecture is discussed in the contexts of self-adaptive systems, SoS and IoT [60, 81]. Deployment uncertainty relates to available resources, components and quality attributes of the system [81] and both structure and behaviour of software architecture [81]. Uncertainty can arise due to the topology of deployed components being unpredictable in dynamic environments, particularly in the context of self-adaptive systems [60] and the reconfiguration and dynamic synthesis of architecture in operational systems [81].

4.8.2.6 Design

In the design aspect, discussion about uncertainty often concerns other architectural aspects. These include in SoS [54, 57–59], IoT [55], tools [58, 82, 83], analysis [76, 78], evaluation [84], decisions [75, 83] and knowledge [35]. An additional area of uncertainty in architecture design is in relation to machine learning architectural components which might be affected by uncertain behaviour from learning [84]. An architecture with machine learning components has an inherent uncertainty regarding functional stability which makes architecture evaluation and trade-off analysis challenging [84, 84].

4.8.2.7 Evaluation

Uncertainty in evaluation techniques emerge from the challenge of forecasting the impact of the architecture decisions on the software architecture quality attributes and the assessment of alternative candidate architectures to select a suitable architecture [32, 85, 86]. Evaluation relates to other architecture aspects such as performance [46], stability [69, 69], reliability [34, 66–68], and analysis [68]. In a recent systematic review, the concept of continuous evaluation of software architecture from design time to runtime is proposed to handle uncertainty in software architecture evaluation [40]. Continuous evaluation of software architecture is necessary given the dynamic and uncertain nature of complex systems such as IoT, SoS and self-adaptive systems, which influences their architecture evaluation results [40].

4.8.2.8 Evolution

Software changes with time as new requirements are introduced, features are modified and errors corrected. Some such changes arise due to uncertainty and may themselves be subject to uncertainty. Thus, predicting the stability of a software architecture is a major source of uncertainty in evolution [47, 87, 88]. Uncertainties in software architecture evolution arise from the fact that the understanding of how architectural decisions map to quality attribute responses, in terms of costs and benefits, depends on intuition and expert opinion [87]. Thus, the subjective nature of the process leads to uncertainty [87]. Evolution can cope with uncertainty through, for instance, determining the appropriate degree of architectural flexibility and balance with economic consideration of the mitigations [47, 87].

4.8.2.9 Run-time

This aspect relates to uncertainty in run-time analysis [44, 45], deployment [81] and evaluation [40] of the architecture of the system. Discussions about uncertainty at run-time in software architecture often relate to self-adaptation, which we discuss in the next sub-section [89].

4.8.2.10 Self-adaptation

Self-adaptation is one of the architecture aspects in which uncertainty in general and uncertainty in software architecture are widely considered [41, 90–93]. Often, uncertainty is considered in the context of the operational environment at run-time, the adaptation-function execution, and variable user needs [41, 94].

In self-adaptive systems, uncertainty can emerge from system parameters, uncertainty in analytical models and uncertainty in user preferences [95]. Uncertainty in such systems can be internal (such as predicting the impact of a system configuration change on quality attributes) or external (such as weather conditions requiring adaptation) [95].

Literature sources on self-adaptation contain a range of discussions of uncertainty in relation to other aspects: design [96], cyber-physical systems [97, 98], requirements and runtime [99] and machine-learning techniques for proactive architectural adaptation [100] and deployment [60]. However, despite the range of such discussions, there is still a need to clarify the concept of uncertainty in self-adaptive systems [41].

4.8.3 Architecture Artefacts

4.8.3.1 Architecture description

Architecture description is the main artefact of software architecture [57, 92]. An area of research interest about uncertainty in architecture description is the ability of the description to include and represent uncertainty details [57, 92]. There are a few existing approaches that can explicitly describe and capture uncertainty in architecture description. These include using ADLs and enhanced UML notation, such as SoADL[57] and Fuzzy self-Adaptation ModELing (FAME) [96]. Besides, there are analytical architecture description approaches, which include mathematical characterisation of uncertainties and fuzzy representation of uncertainty [26, 27].

There are two kinds of approaches to representing uncertainty in architecture description: encoding the uncertainty as part of the architecture description artefact or representing the uncertainty using a mathematical or analytical approach [92]. In Section 4.8.4, where we consider tools and notations, we will provide a description of some of these tools and notations, which are used to generate description artefacts.

4.8.3.2 Documentation

We consider documentation in the context of the following research areas: automatic uncertainty detection in software architecture documentation [61] and uncertainty expressions in software

architecture group decision making [71]. In these approaches, uncertainty in architecture documentation relates to architecture communication artefacts such as stakeholder discussions and business contracts relating to architecture. This is in contrast to architecture descriptions, which are about architecture models such as architecture views and ADLs [96, 101]. Sources of uncertainty in documentation often relate to lack of information resulting in communication uncertainties such as ambiguity and confusion [61].

4.8.3.3 Requirements

Uncertainty may influence architecturally significant requirements [38, 102–104]. Uncertainty in requirements impacts architecture design since requirements are the primary input into the design process [25]. Therefore, even though uncertainty representation in requirements might not always be specific to software architecture, it still is significant in informing the architecture process [25]. Software architecture should be designed for change since requirements are often volatile [38]. Designing for change requires that architectures are flexible, modifiable and scalable, to support the incorporation of uncertainty [38, 104].

4.8.3.4 Traceability

Finally, uncertainty in traceability applies when mapping between architecture description elements and other systems aspects such as performance and code [64, 105]. For instance, following literature sources highlight traceability between architecture models and code [105] and between performance results and architecture components [64].

4.8.4 Architecture Tools and Notations: Uncertainty capabilities

Given the range of architecture aspects, there are different approaches to handling uncertainties, such as analytical approaches or explicit recording of uncertainty details [9]. Mathematical or analytical techniques can be used to represent uncertainty in software architecture. These include probability theory [95], Bayesian theoretical approaches such as Bayesian networks [73, 79, 84, 99], variations of Markov chains [99], Evidence theory [34, 106], Possibility theory [9, 54, 106] and fuzzy approaches such as fuzzy logic and numbers [26, 32, 55, 59, 77, 86, 96].

Explicit uncertainty handling approaches encode uncertainty in artefacts through notations and architecture languages with capabilities to capture or include uncertainty details. Examples of these include architecture description languages based on fuzzy concepts [55], fuzzy UML notation [96] and fuzzy traceability notation to capture uncertainty between architecture components and performance or sources code [64, 105].

Table 4.3: Languages, notations, and tools designed for *explicitly* working with uncertainty in software architecture

	Target Aspects	Description - central concept	Ref
Languages			
1. SoADL	Systems of System Design	SoADL can represent epistemic uncertainty in the composition of SoS architectures	[54, 57]
2. Fuzzy Sos Architecture Description Language	SoS & IoT Design	Fuzzy SoADLis used to model architectures of System of systems of IoT incorporating uncertainty	[55]
3. RELAX	Self-adaptation requirements	RELAX specifics variant and invariant requirements and captures uncertainty in the operating environment	[25]
4. Stitch language	Self-Adaptation	Stitch, which has uncertainty capabilities, represents architecture-based self-adaptation to automate system administration	[91]
5. Architecture and code traceability language	Traceability and Code	Traceability language for mapping between source code and architecture model components. It handles uncertainty in the mapping i.e, ambiguity, incompleteness and assumptions.	[105]
6. Architecture and performance traceability language	Traceability and Performance	Traceability language for mapping between performance results and architecture model components. It handles uncertainty in the mapping i.e, ambiguity, incompleteness.	[64]
Notations			
7. FAME	Self-Adaptation	FAME is a UML like enhanced modelling notation which supports diagrams such as Fuzzy Case Diagram, Fuzzy Class Diagram and Fuzzy Sequence Diagram for self-adaptive systems	[96]
8. Chemical Abstract Model (CHAM)	Self-Adaptation	CHAM is a computational model notation which uses a chemical reaction analogy to model possible configurations of architectural self-adaptation, thus incorporating uncertainty	[92]
Tools			
9. GuideArch	Evaluation	GuideArch is an architecture analysis and evaluation tool which ranks candidate architecture alternatives using fuzzy notation to select a suitable implementation architecture	[27]
10. iArch-U Integrated Development Environment (IDE)	Design	iArch-U is an IDE for managing uncertainty in a module fashion in all phases in software development. It supports addition or deletion of uncertain concerns to/from models, code and tests whenever these concerns arise or are fixed to concerns.	[82]
11. MU-MMINT: An IDE for Model Uncertainty	Design	MU-MMINT IDE allows developers to express their design time uncertainty within software artifacts and perform a variety of model management tasks such as reasoning, transformation and refinement in an interactive environment	[83]
12. Requirements and Architecture Decision AnalyseRl (RADAR)	Requirement and decision Analysis	The RADAR tool includes the RADAR modelling language to consider impact of uncertainty in requirements and architecture decisions.	[73]

Notations that can represent uncertainty can be incorporated in architecture development tools to enhance them with capabilities to handle uncertainty. Tools like GuideArch explore the architecture solution space under uncertainty using a fuzzy notation which represents uncertainty about architecture candidate alternatives [27]. Stitch is a language for self-adaptive system administration in the context of uncertainty [91].

Besides, there are IDEs with capabilities to model uncertainty when developing architectures [82, 83]. However, such tools are not widely adopted in practice [107], an example of an uncertainty consideration IDE is the Interface-Centric Integrated Uncertainty Aware Development Environment (iArch-U) [82].

RELAX is a language to model the requirements of self-adaptive systems [25] which incorporates

uncertainty. While not all RELAX-ed requirements are architectural, some of them are likely to be architecturally significant in the context of self-adaptive systems.

Table 4.3 is a list of languages, notations and tools which are designed explicitly for working with uncertainty in various aspects of software architecture under the categories identified in Section 4.7 - indicated in the Target Aspects column. We present these tools, notations, and languages to illustrate the research on the consideration of uncertainty in software architecture. Data on their adoption or use in practice is not available.

4.8.5 Summary

In this section, we have discussed works on uncertainty related to software architecture aspects in the categories of *Definition and Concepts*, *Activities*, *Artefacts*, and *Tools and Notations*. For each category, we have summarised the research relevant to identifying, representing, analysing and managing uncertainty where relevant.

A number of diverse approaches have been described in literature with these aims. One implication of such diversity is that, when the term *uncertainty* is used, it is not always clear what it exactly refers to, unless specific details are provided from the specific context. This ambiguity is one reason why it is challenging to categorise works discussing uncertainty in software architecture, software systems and software engineering [12, 19]. Indeed, *Ambiguity* too is a form of uncertainty [1].

4.9 Risks and opportunities of uncertainty in software architecture aspects

Uncertainty can cause risks or can create opportunities [2, 9]. In this context, a risk is a negative influence of uncertainty and opportunities are its positive consequences [2, 9]. Discussions of uncertainty in the literature generally deal with its risks. All the literature sources we found in this survey discuss uncertainty only with respect to its negative consequences. Opportunities of uncertainties are not considered. Consequently, the exploitation of uncertainty is not discussed in any of the 25 aspects.

Risk mitigation is an integral part of managing uncertainty [2]. In fact, there is a software architecture development methodology driven by risk called risk-driven architecture [108]. There are many possible risks that can arise from uncertainty in software architecture with the potential to result in significant negative consequences. Discussions of such risks in literature tend to

be specific to an aspect, such as during software architecture analysis [30, 48] and evaluation [26, 47].

For example, during architecture analysis using techniques such as ATAM [30], one potential risk from uncertainty might be identifying incorrect architecturally significant requirements or priorities, and this might have serious consequences. In contrast, during evaluation [27], the risk might be selecting the wrong candidate architecture for deployment, and this too, might have serious consequences.

Thus, uncertainties in software architecture such as those arising from unanticipated changes, incomplete documentation or variability in the operating environment, create risks to the quality of the system with consequences of potentially high costs [2]. These costs can be in terms of finance [32], performance [33], reliability [34], legal issues [35] and business disruptions [9]. Therefore, uncertainty in software architecture should be identified and avoided, or at least mitigated, as much as possible [2].

4.10 Mitigation and exploitation of uncertainty in software architecture aspects

Since opportunities are rarely discussed in literature with respect to uncertainties, their exploitation is also not discussed. The main focus is on risks and their mitigations, as we have demonstrated in this chapter.

Mitigation strategies manage risks from uncertainty in software architecture aspects such as requirements, analysis, design, evaluation, run-time, documentation, traceability and classification. For instance, risk from uncertainty to interpretation and understanding can be mitigated through classifications or taxonomies [15, 21]. Often, such classification approaches target specific areas, such as uncertainty in self-adaptive systems or complex systems [2, 14].

Classifications can provide a basis for defining the scope of uncertainty, with some mitigation strategies targeting epistemic uncertainties while others target aleatory uncertainties or both [54]. Likewise, the lack of knowledge and the use of imprecise techniques can cause epistemic uncertainty in architectural decision making. Therefore, approaches for dealing with architecture knowledge can be used to mitigate such uncertainties during decision making or trade-offs [35].

Similarly, methodological approaches such as agile development can mitigate epistemic uncertainty through feedback loops on architecture decisions and alternatives [39, 62]. Moreover, modelling, simulation and testing strategies can be used to fill the knowledge gap as mitigations

in decision making [33, 67].

Other uncertainty mitigation approaches include design tools and techniques which help architects work with uncertainty. These include tools such as IDEs, techniques for reasoning about uncertainty in design and frameworks for synthesising architectures in uncertain conditions to achieve the desired balance of quality attributes. This survey identifies some tools, notations and languages, for explicitly working with uncertainty in Table 4.3.

Multiple mitigation approaches exist for uncertainty in architecture evaluation. These include the use of fuzzy representation to aid in selecting candidate architecture [27], Analytic Hierarchy Process (AHP) [86], possibility theory (an alternative to probability theory for handling uncertainty) [109] or simulations such as the Monte-Carlo method [106], continuous software architecture evaluation [40], among others.

Mitigations in architecture design include conceptual approaches. For instance, designing for stability is important for the durability of software architecture as it evolves. Similarly, mitigation approaches for architecture evolution focus on architecture flexibility in the context of stability so that the architecture can handle unanticipated requirements [47]. Architecture flexibility in the context of evolution is considered as an investment in the system, returning benefits from future adaptations and reducing maintenance costs [47].

Mitigations in methodology focus on recommendations and good practices such as keeping architecture designs simple, delaying decision making and planning multiple options [62]. Testing, prototyping and designing for flexibility mitigate uncertainty risks from architecturally significant requirements [38]. Prototyping and incremental development help with identifying uncertainty and coping with them, respectively [102].

Overall, some mitigations may target specific aspects of software architecture, and others may target a range of aspects such as run-time and self-adaptation. Mitigations can also be vulnerable to uncertainties. For example, within the self-adaption mechanism, there may be uncertainties about the adaptation-function itself as it executes, and thus it may require further mitigations to guarantee satisfactory execution [94].

4.11 Discussion: Gaps and directions for Future Work

This survey identifies aspects of software architecture from the results of a literature survey, categorises these aspects and discusses current work on uncertainty in their respective contexts.

The architecture aspects within the identified categories of *Definition and Concepts*, *Activities*,

Artefacts, and *Tools and Notations* are inter-related. They share data, techniques and feedback mechanisms. Thus, uncertainty identified in one aspect is potentially uncertainty in multiple aspects of the software architecture.

Recognising the influence of uncertainty within each category and within each aspect is a positive step in considering uncertainty in software architecture. Other significant steps are its systematic identification, minimisation and resolution. The last two involve mitigations of uncertainty risks [83]. Potential future work for considering uncertainty as a first class concern in software architecture therefore involves uncertainty management in all categories and individual aspects of software architecture.

This chapter demonstrates that uncertainty in software architecture is an open research area with increasing interest and effort, but with many outstanding issues to address, including approaches for identifying, representing, analysing and managing uncertainty in various aspects of software architecture. Many of the available approaches target uncertainty in specific aspects and are limited in scope because they do not share their results with other architecture aspects, which may also be impacted by the same or similar uncertainties.

In terms of *Definitions and Concepts*, there are many novel and innovative approaches being proposed to address uncertainty in software architecture. However, there is a lack of clarity on how to organise, select and apply such approaches. As discussed in Section 4.8.1, there are General, Classification, Cd-Sys such as SoS or IoT, Knowledge, Methodology, System qualities and Risks aspects, each with distinct approaches to addressing uncertainty in software architecture. Thus, there is a need for a systematic approach to organising and integrating this work instead of treating them in miscellaneous categories.

In defining uncertainties, it is important to understand their sources. Some sources of uncertainty in software systems have received particular attention, such as the operating environment in self-adaptive systems architecture [94]. Progress has been made in designing architectures of adaptive systems.

In architecture *Activities*, which we discussed in Section 4.8.3, specific approaches exist to address uncertainty in aspects such as Communication, Decision making, Trade-offs, Analysis, and others in their respective context. However, most of these works are done in isolation and will require coordination approaches to be integrated and support each other to manage uncertainty in a systematic manner.

In architecture *Artefacts*, such as architecturally significant requirements and architecture descriptions, uncertainties, their mitigation and desired outcomes should be represented at

the relevant level of detail. For example, uncertainties and outcomes may be part of requirements while architecture descriptions and resulting detailed designs may include details of mitigation. Existing approaches support this for specific domains or artefacts and thus there is a lack of support for uncertainty representation in software architecture artefacts in general. Artefacts which include uncertainty information are likely to help minimise future risks through making the uncertainty knowledge explicitly available to stakeholders throughout the software lifecycle. The incorporation of uncertainty information in artefacts requires tools to support all the activities of this process.

Further, there is a general lack of traceability of uncertainty information among the aspects. However, there are some attempts towards addressing this, such as tracing between architecture documentation and code or performance under uncertainty [64, 105].

In most of the aspects considered in the chapter, support from *tools and notations* for considering and managing uncertainty in software architecture is lacking. Some work on tools and notations is discussed in Section 4.8.4; however, we did not find evidence of these approaches being widely adopted by practitioners.

Based on these observations, we suggest the following research directions for the future:

- *Definitions and Concepts*: More work is required on the interpretation and understanding of uncertainty as an architecture concern with precise definitions and characterisations. A generic conceptual model for uncertainty that considers its different dimensions and attributes would provide a robust basis for considering uncertainty in architecture artefacts and processes. We suggest more research on explicitly identifying and exploiting opportunities afforded by uncertainty since there appears to be little work published on this topic.
- *Activities*: More research is needed to investigate how the explicit consideration of uncertainty can be embedded in all architecture activities and broader software engineering processes to produce outcomes that are robust in the face uncertainty.
- *Artefacts*: Information about uncertainties need to be represented in a usable and explicit manner in architecture artefacts to support their use in architecture activities. In addition, more work on traceability among software artefacts is needed in general, with a focus in this particular context of traceability of uncertainty information within architecture artefacts and with other artefacts. This information must be accessible to all relevant stakeholders and be human or machine readable depending on their proposed use in the software lifecycle.
- *Tools and Notations*: All of the above require support from the notations and tools used in

software architecture and broader development activities. A particular concern here is that such notations and tools must be integrated into existing development infrastructure since the need to learn and use additional notations and tools is likely to hinder their adoption.

We note that the elements of the suggested potential future work are interconnected. In a broader sense, we would thus propose the development of a generic but customisable approach to uncertainty management in software systems. Such an approach would include addressing uncertainty in software architecture as a vital and foundational artefact in the software lifecycle.

The consideration of uncertainty as a first-class concern in software architecture requires understanding of its nature, sources, risks, and mitigation in all aspects [2, 21]. In addition, its management to achieve the desired outcomes, such as system qualities, requires the definition and development of notations, mitigation tactics, activities and tools with capabilities of representing, analysing and mitigating uncertainty in software architecture. This PhD thesis contributes towards this vision.

4.12 Conclusions

This chapter reviewed literature sources from 1991 to 2021 to explore uncertainty in software architecture aspects, including architecture *Definitions, Concepts, Activities, Artefacts, Tools and Notations*. The survey resulted in 82 literature sources, which highlighted 25 specific software architecture aspects where uncertainty is explicitly considered and provided a potential categorisation of these aspects. Each of these aspects has received varying degrees of focus in terms of research interest and results. The survey identified existing approaches for addressing uncertainty in these aspects of software architecture as well as gaps in the state of the art.

There is a need for further research to enhance uncertainty consideration, representation, and management in all the aspects of software architecture. In addition to the gaps highlighted in the survey, a systematic approach to the management of uncertainty in software architecture also needs to be devised. Future research directions include addressing uncertainty within each of the four categories of aspects and within each individual aspect. In the Discussion Section, we highlighted potential future research directions for each category.

This PhD thesis presents our current work on developing a generic framework for considering uncertainty in software systems, with a particular focus on software architecture. This framework enables the generation of uncertainty data for software architecture to support software architecture uncertainty documentation, representation, analysis and management.

UNCERTAINTY CHARACTERISATION CONCEPTS: STATE OF THE ART

Few approaches, which explicitly represent the various dimensions of uncertainty in software architecture or software systems, exist. This chapter presents specific approaches that consider uncertainty in its multiple dimensions in software systems. The chapter builds on the background from Chapter 3.

5.1 Overview

Section 5.2 presents an overview of the existing uncertainty consideration approaches from the background discussed in Chapter 3. Section 5.3 presents an overall summary of the existing approaches. Section 5.4 is the conclusion.

Table 5.1: Literature sources on uncertainty frameworks

ID	Focus area	Title	Year	Reference
1	Cognitive Science	Ignorance and uncertainty: Emerging paradigms	1989	[1]
3	Complex Systems	A framework for understanding uncertainty and its mitigation and exploitation in complex systems	2004	[2]
4	Complex Systems	A classification of uncertainty for early product and system design	2007	[9]
10	Cyber-Physical Systems	Understanding uncertainty in cyber-physical systems: a conceptual model	2016	[3]
11	Self-Adaptive Systems	A classification framework of uncertainty in architecture-based self-adaptive systems with multiple quality requirements	2017	[21]
5	Self-Adaptive Systems	Relax: Incorporating uncertainty into the specification of self-adaptive systems	2009	[25]

5.2 Existing approaches for considering uncertainty in software systems

To identify these approaches, we searched the following academic databases: IET Digital Library¹, Web of Science², DBLP³, Scopus⁴, ACM Library⁵, and IEEE Xplore⁶. The search terms we used to find literature sources in the databases were: ‘*uncertainty AND (software OR system) AND framework*’. In addition, we conducted a manual search to obtain more comprehensive results by following up references from results returned by the search. We then analysed the search results manually to identify primary references, which either propose or consolidate concepts for characterising uncertainty, as shown in Table 5.1. Our approach to determining primary sources included 1) checking for citations among results to check dependencies among literature sources, and 2) checking whether further useful information would be gained by retaining a source.

The results of our search show that there are not many existing approaches that consider uncertainty in software systems as a first class concern. We identified one source from cognitive science which has been cited in literature related to software systems, two generic frameworks

¹<https://digital-library.theiet.org/>

²<https://apps.webofknowledge.com>

³<https://dblp.org/>

⁴<https://www.scopus.com>

⁵<https://dl.acm.org/>

⁶<https://ieeexplore.ieee.org>

from complex systems, a conceptual model from cyber-physical systems, a literature review of uncertainty in self-adaptive systems and an approach for characterising uncertainty in self-adaptive systems. Thus, four contexts contribute to our study of existing approaches.

The identified approaches consider uncertainty in the following contexts. Cognitive science is related to human thought and behaviour [1]. There is a strong intertwined relationship between physical and software components in cyber-physical systems [3]. Complex systems comprise numerous complex components, interconnections and dynamism [2]. Self-adaptive systems automatically adapt to counter the influence of uncertainties against performance goals [94, 95].

Our work consolidates and extends these individual frameworks into a generic customisable framework for considering uncertainty. In Sections 5.2.1 to 5.2.4, we examine and compare the approaches from these contexts to extract concepts and attributes of uncertainty for consolidation.

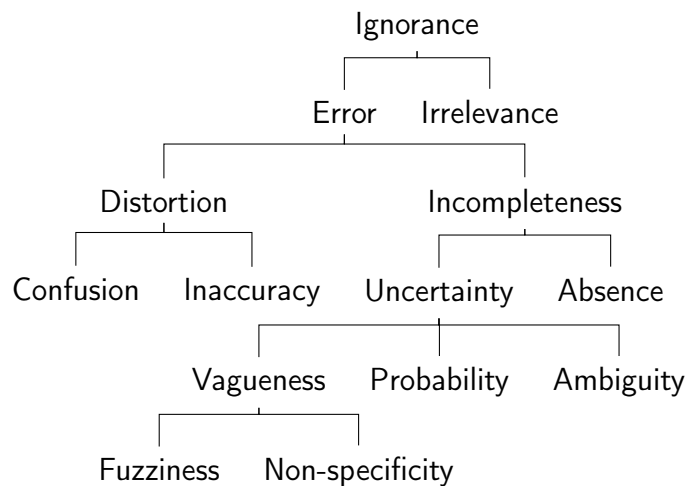


Figure 5.1: Taxonomy of Ignorance (Reproduced from [1])

5.2.1 Uncertainty in cognitive science

In "Ignorance and uncertainty: emerging paradigms", Smithson defines uncertainty in the context of ignorance [1]. Their working definition of ignorance states that “A is ignorant from B’s point of view if A fails to agree with or show awareness of ideas which B defines as actually or potentially valid” [1].

One stated benefit of this definition is that it avoids an absolutist problem by placing the onus on the agent to define what they mean by ignorance [1]. This definition implies a subjective perspective to uncertainty, such that different agents may not identify or perceive an uncertainty in the same way.

This definition focuses on knowledge; that is, ignorance can be due to “ignoring” or “being ignorant” [1]. “Ignoring” implies an active conscious ignorance or Irrelevance. On the other hand, “being ignorant” means a passive ignorance resulting in a flawed state of knowledge or Error.

Figure 5.1 shows the taxonomy of uncertainty from ignorance. It shows that uncertainty is a form of incompleteness of knowledge, where incompleteness itself is classified as an error that causes ignorance. Besides, uncertainty has three measurement subcategories: vagueness, probability and ambiguity. Vagueness has two categories: fuzziness and nonspecificity (generality) [1]. Since this classification is centred on knowledge, it deals with epistemic uncertainty.

5.2.2 Uncertainty in complex systems

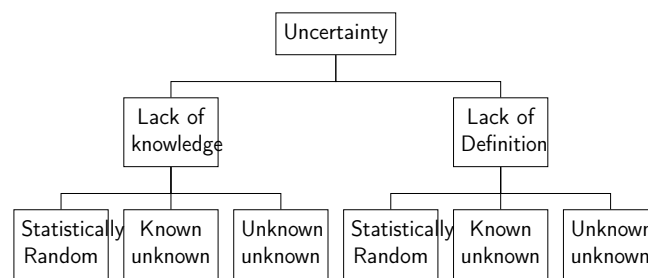


Figure 5.2: Uncertainty in complex systems (Based on [2])

Hastings and McManus define a framework for managing uncertainty, and techniques for mitigating and even taking advantage of it, in complex systems [2]. In this framework, uncertainty is defined as both the lack of knowledge and the lack of definition. Each of these in turn has three subcategories: statistically characterised (random) variability or phenomena, known unknowns and unknown unknowns, as illustrated in Figure 5.2. There is an emphasis on knowledge-based or epistemic uncertainty, though the subcategory of statistical variability or phenomena can characterise aleatory uncertainty.

Furthermore, Hastings and McManus state that *uncertainty* leads to *risks* or *opportunities*, which are handled by *mitigation* or *exploitations*, ideally leading to desired *outcomes* [2]. Risks are the potential negative consequences of uncertainty while opportunities are potential positive consequences of uncertainty. Mitigations are the approaches to risk minimisation while exploitations refer to approaches to value or opportunity enhancement. The outcomes are the desired results of mitigating risks or exploiting opportunities. Thus, the work by Hastings and McManus suggests a broad and systematic approach to considering uncertainty in systems.

In addition, there are numerous individual sources of uncertainty in complex systems. de Weck

et al. [9] identify and classify sources of uncertainty into two broad categories: exogenous and endogenous. Endogenous uncertainties arise within the system while exogenous uncertainties are external to the system. Typically, the degree of influence in mitigating risks or exploiting opportunities arising from uncertainties decreases from endogenous to exogenous uncertainties. For example, an organisation can better control uncertainties within the development environment but might have limited influence on external uncertainties from natural disasters [9].

5.2.3 Uncertainty in cyber-physical systems

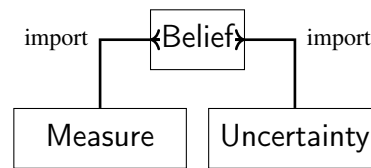


Figure 5.3: Conceptual model (Reproduced from [3])

Zhang et al. [3] describe a conceptual model for uncertainty in cyber-physical systems. They consider uncertainty to be subjective, thus dependent on the beliefs of an agent, such as a human, a system, or an object, whose state of knowledge represents a particular world view [3]. Uncertainty is considered and represented at three logical levels of a cyber-physical system: application, infrastructure and integration [3, 110, 111]. Their model has three main components as shown in Figure 5.3: belief, uncertainty and measure.

Belief represents an agent's subjective view, with its beliefs being valid or invalid. Beliefs are represented using belief statements. Beliefs can change with time as an agent's state of knowledge changes [3]. Since belief statements can be either valid or invalid, they have to be validated. For this purpose, the model acknowledges an objective world with facts independent of the beliefs of the agent.

The known objective facts constitute the evidence, which is used to assess the validity of belief statements. The objective relationship between a belief statement and its evidence is called evidence knowledge. The existence of evidence can be known or unknown, thus resulting in uncertainty [3].

Uncertainty is based on an agent's confidence over its belief depending on the available evidence [3]. Specifically, uncertainty occurs when an agent lacks confidence in its belief statement due to lack of evidence to validate its beliefs [3]. Besides, uncertainty arises through indeterminacy: a state where full knowledge does not exist to determine the validity of a belief statement. Indeterminacy manifests through an indeterminacy source, which results in lack of confidence in the belief statement and thus uncertainty [3].

Measure defines a non-exhaustive list of scales that can be used for quantifying uncertainty [3]. These include probability, ambiguity and vagueness, with vagueness categorised into fuzziness and non-specificity [1].

The conceptual model also identifies the following attributes of uncertainty:

- *Lifetime* - the duration of the existence of uncertainty. For instance, it may exist for a while and then disappear, or it may live in perpetuity until resolved.
- *Pattern* - how the uncertainty might change with time, such as in a systematic or aperiodic manner. A systematic pattern can be periodic or persistent, while aperiodic patterns can be transient or sporadic. Additional patterns may also exist.
- *Locality* - where an uncertainty occurs in the belief statement.
- *Risk* - the level of risk from low, medium, high to extreme, with the extreme case requiring particular attention.

5.2.4 Uncertainty in self-adaptive systems

In this section, we present the following two approaches from Table 5.1: uncertainty in self-adaptive systems with multiple quality requirements in section 5.2.4.1 and uncertainty in self-adaptive system requirements in section 5.2.4.2.

5.2.4.1 Self-adaptive systems with multiple quality requirements

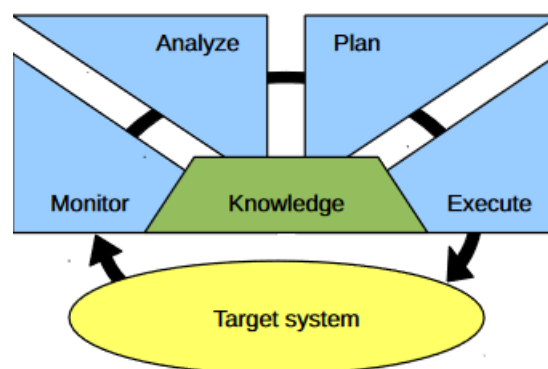


Figure 5.4: MAPE-K Loop (Image from [4])

Mahdavi-Hezavehi et al. [21] consolidate existing works in the domain to define a framework for characterising uncertainty in self-adaptive systems with multiple quality requirements [21]. Here, uncertainty is discussed within the context of the adaptation model such as the one depicted by

Figure 5.4. The dimensions of uncertainty include location, nature, level/spectrum, emerging time and sources. Each of the dimensions has specific options:

- *Location is where uncertainty manifests itself within the overall adaptation process. It has six options: model, goal, function, environment, resource and managed system locations.*
- *Nature specifies whether the uncertainty is due to the imperfection of available knowledge i.e. epistemic, or the inherent variability of the phenomena described, i.e. aleatory.*
- *Level or spectrum indicates the position of uncertainty along the spectrum between deterministic knowledge and total ignorance.*
- *Emerging time is when the existence of uncertainty is acknowledged or appears during the lifecycle of the system, such as design or run-time.*
- *Sources are the circumstances affecting the adaptation decision. These are organised into classes based on the origin of uncertainty including model, adaptation function, goal and environment. Each of these classes can have further sub-options [21].*

5.2.4.2 Self-adaptive system requirements: RELAX

RELAX is a requirements language with the capacity to express uncertainty in self-adaptive systems. It provides explicit constructs for uncertainty: operators to encode uncertainty in requirements statements and environmental factors to capture the uncertain environment.

RELAX transforms the traditional SHALL requirements statements into RELAX-ed requirements. These are requirements that can be temporally relaxed when operating conditions of the system change so that the critical non-relaxable requirements can continue to be strictly adhered to [25]. The relaxable requirements are called variant, while the non-relaxable requirements are invariant.

Relaxable requirements are specified by applying uncertainty operators to the SHALL statements in compliance with the RELAX grammar [25]. RELAX defines three types of operators: modal, temporal and ordinal.

- The modal operator expresses the possibility of functionality and actions through modal verbs such as MAY.
- The temporal operator expresses uncertainty related to time through operators from fuzzy temporal logic such as EVENTUALLY, UNTIL, BEFORE, AFTER, IN, AS EARLY AS

POSSIBLE, AS LATE AS POSSIBLE and AS CLOSE AS POSSIBLE TO some frequency value.

- The ordinal operator expresses quantitative uncertainty through fuzzy terms such as AS CLOSE AS POSSIBLE TO some quantity and AS MANY or AS FEW AS POSSIBLE.

Factors of the uncertain operating environment are identified by the following attributes: environment (ENV), environment monitors (MON), relationship (REL) and dependencies (DEP).

The environment is the operating environment, which is a source of uncertainty. Environment monitors are used to monitor for uncertain conditions in the environment. Since it may not always be possible to identify uncertainty directly from the environment, the relationship defines the connection between the environment and monitors for detecting uncertainty. Finally, dependencies state which related requirements are affected by the uncertainty in the current requirement statement [25].

5.3 Summary of existing frameworks

The work described in the preceding sections influenced the framework proposed in this thesis. Hastings and McManus' theory that *uncertainty leads to risks or opportunities, which are handled by mitigation or exploitation, leading to desired outcomes* underpins our framework. While the frameworks discussed in this section are aimed at specific contexts, their characterisation of uncertainty contributed to the set of uncertainty attributes in the proposed framework which we present in Chapter 6.

There are various approaches that conceptualise uncertainty in specific domains or contexts, such as policy modelling [15], self-adaptive systems [14, 22, 112], complex systems [2, 9], cyber-physical systems [3, 110] and cognitive science [1]. The major difference between these solutions and our approach is that our framework consolidates, generalises and extends existing concepts so that they can be used regardless of context. Our conceptual model can also be adapted as necessary.

Walker et al. [15] define a conceptual basis for uncertainty management in model-based decision support systems. Mahdavi-Hezaveh et al. [21] define uncertainty and consolidate dimensions of uncertainty with a focus on self-adaptive systems with multiple quality requirements. Their review is used in this work and we discuss their results in section 5.2.4.1. The framework from complex systems by Hastings and McManus [2], discussed in section 5.2.2, specifies the relationship between uncertainty, risks or opportunities, mitigation or exploitation and outcomes. Zhang et al. [3] define a conceptual approach for managing uncertainty in cyber-physical

systems, with a focus on epistemic uncertainty, as discussed in section 5.2.3. Smithson [1] defines uncertainty as a subcategory of ignorance. We discuss this work in section 5.2.1. Whittle et al. [25] define a language, RELAX, for explicitly representing uncertainty in self-adaptive system requirements, as discussed in section 5.2.4.2. Finally, we previously discussed the need for clarity in characterising uncertainty in software architecture in [12].

5.4 Conclusion

Often, uncertainty is treated using mathematical or analytical approaches. However, such approaches do not capture or highlight other details about the uncertainty, such as its location, nature and more. The details about the uncertainty can be captured or characterised using the various attributes of uncertainty. In this chapter, we explored existing works which explicitly state attributes and characteristics of uncertainty. These include sources from complex systems, self-adaptive system, cognitive science and cyber-physical systems. We use the concepts and notions of uncertainty identified in this chapter as a basis for defining an uncertainty framework for software systems.



CHAPTER SIX

FRAMEWORK

This chapter presents the uncertainty consideration framework and conceptually evaluates its application using software architecture illustrative examples. The framework is built from the concepts and observations discussed in Chapter 5.

6.1 Overview

This chapter presents an extensible conceptual framework, which defines a foundation for the systematic and explicit consideration of uncertainty in software systems. The proposed framework consolidates and extends uncertainty concepts and attributes from the contexts of cognitive science, complex systems, self-adaptive systems and cyber-physical systems, which have been identified following a review of existing work in the area in Chapter 5.

The identified attributes are organised into broad categories, to which additional uncertainty attributes can be added. The extensibility property addresses the need for customising the framework, since the pre-defined list of attributes in the context of open-ended E-type systems might not be exhaustive. Three case studies from different application contexts of software architecture are used to conceptually evaluate the framework.

The novel contribution of the work is a generic but customisable framework for considering uncertainty in software systems. While the characteristics of uncertainties present in a specific system may be context-dependent, we hypothesise that the use of a generic framework will help the systematic consideration of uncertainties and mitigation of their risks. With software systems becoming more ubiquitous and decoupled [24], domain- or context-specific uncertainty frameworks may be too narrow in scope to identify, analyse and mitigate associated risks.

The rest of the chapter is as follows: Section 6.2 proposes the new framework of consolidated uncertainty attributes, and a suggested workflow of applying it. Section 6.3 provides illustrative examples and a demonstration of the extensibility of the framework. Finally, Sections 6.3.4 and 6.4, are discussion and conclusions, respectively.

6.2 A framework for considering uncertainty in software systems

In this section, we consolidate and extend the concepts discussed in Section 5.2 with the aim of using them to systematically consider uncertainty in software systems in general.

6.2.1 Framework definition

We define uncertainty as the lack of certainty arising from both lack of knowledge, which subsumes lack of definition as well as active and passive ignorance, and inherent variability. Each uncertainty has the potential to cause negative or positive consequences depending on the specific system and its facets. The consequences can be risks or opportunities, which require mitigation or exploitation, targeting one or more specific outcomes. In this framework, uncertainty is a concept characterised by multiple attributes.

The resulting conceptual model is shown as an Entity Relationship Diagram (ERD) in Figure 6.1, with attributes of uncertainty detailed in Table 6.1. The framework includes details of the software system, its facets, which are aspects of development and operation where uncertainty might be present, and viewpoints, which are the perspectives from which uncertainty might be considered. A system has at least a facet where uncertainty is identified from at least a viewpoint. Each system may also have many uncertainties.

Each uncertainty is characterised by a set of attributes. These attributes were identified in the following two ways.

First, as discussed under Chapter 5, from the contexts of cognitive science, complex systems, self-adaptive systems and cyber-physical systems. The basis for the identification of these attributes was one of the hypothesis of this thesis, captured in Question 1, that if an attribute can be used to characterise uncertainty in a specific contexts of software systems, it can also be used to characterise uncertainty in other contexts of software systems. Later, we evaluate the application of the attributes to assess their suitability for characterise uncertainty in the context of software architecture.

Secondly, some of these attributes were extended or customised depending on identified additional information from various literature sources in the course of this research. Of course, the attributes are empirical thus not exhaustive. Table 6.1 presents the list of attributes and their sources, together with their extension status.

Figure 6.1 shows the mapping among the various uncertainty characterisations concepts. The rest of the entities of the AEROSTACK in Figure 6.1, from the *causes* to *Outcome*, define the consequences of the uncertainty.

The values of attributes defined in Table 6.1 can provide useful information when decisions need to be made in the presence of uncertainty. Some details of an uncertainty can be identified by asking questions such as *what* for its description, *when* for its emerging time, *where* for its location, and *why* for its cause.

Each attribute is annotated with a superscript denoting its cardinality constraint: $+$, $*$, n , $n+$, such that $+$ means at least one, $*$ means zero or more, n is an exact positive integer and $n+$ means at least n .

Out of the twenty seven (27) proposed attributes in Table 6.1, all but one can apply to both (B) epistemic (E) and aleatory (A) uncertainties. The attribute Evidence is specific to epistemic uncertainty since it depends on the availability of specific knowledge.

As indicated in Section 5.3, this model uses the work of Hastings and McManus [2] as the basis of the framework. Each uncertainty can result in multiple risks or opportunities. These in turn require one or more mitigations or exploitations with the goal of achieving specific outcomes. The main entities of the framework are therefore, the system, its facets, viewpoints, risks, opportunities, mitigations, exploitations and outcomes.

A comprehensive consideration of uncertainty requires details of these concepts as well as their relationships to be defined. Since the proposed framework is generic, we do not prescribe specific processes for applying it. However, there is existing work relating to these concepts for some artefacts. For instance, viewpoints are often analysed and characterised in the context of software architectures [37]. The software architecture of a system can be represented in terms of a viewpoint set, such as the 4+1 model, which prescribes the physical, logical, process and development views, in addition to scenarios [37]. The framework can be used to explicitly consider and represent uncertainties in each of these views using the attributes of Table 6.1 and relationships in Figure 6.1. Where necessary, additional attributes maybe added or removed to suit the context.

The rest of the section describes the uncertainty attributes in the framework. We have

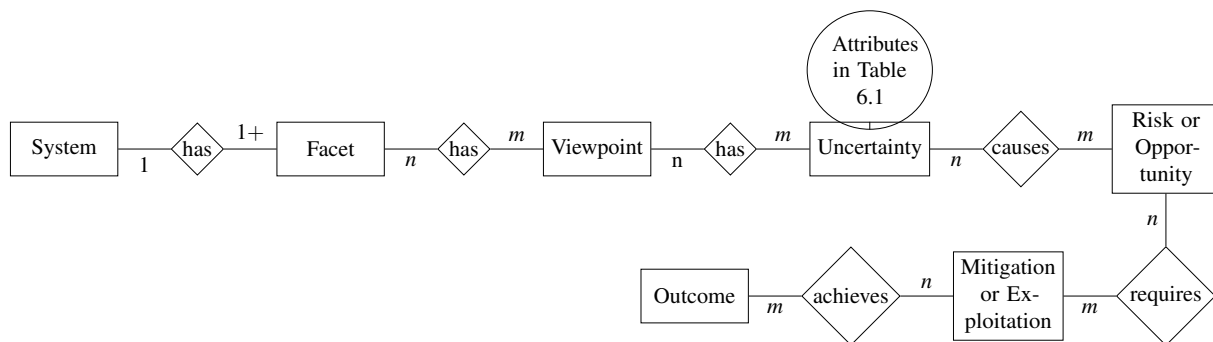


Figure 6.1: Conceptual model: characterisation of uncertainty in systems

organised these attributes into six categories, as shown in Table 6.1: description, source, system, manifestation, time and mapping.

- Uncertainty description attributes:
 - *Description* captures the specifics of the uncertainty in natural or structured language.
 - *Nature* of the uncertainty is either epistemic or aleatory.
 - *Bounds* are the known limits or scope of uncertainty as it occurs.
 - *Perspective* determines the interpretation of uncertainty. A subjective uncertainty depends on the agent defining the uncertainty and is therefore relative, while objective uncertainty is absolute.
 - *Awareness* is an agent's consciousness of uncertainty, which can be a known unknown or an unknown unknown.
 - *Level* of uncertainty is the position of uncertainty in a spectrum from certainty to total uncertainty. This can be expressed in various ways, for instance, using fuzzy values such as High, Medium and Low.
- Uncertainty source attributes:
 - *Source* type of uncertainty can be internal (endogenous) or external (exogenous).
 - *Cause* is the trigger of the uncertainty, such as humans in the loop, operational interference and business changes.
- Uncertainty system attributes:

- Uncertainty influences *systems*. A software system might be composed of sub-systems and their components. In addition, the system can be characterised through its name, purpose, facets and viewpoints. Uncertainty can influence both the structure and the behaviour of a system.
- A *viewpoint* is a standpoint from which we consider and analyse uncertainty. Interests of stakeholders typically determine system viewpoints. Each viewpoint in a system may have multiple uncertainties and an uncertainty may be relevant to multiple viewpoints.
- We use the term *facet* to refer to a specific aspect of a software system. For instance, facets of a system may include activities related to development and operation, artefacts, infrastructure and environments. All such facets can be influenced by uncertainty.
- *Location* is where the uncertainty appears in the system or its facets.
- Uncertainty manifestation attributes:
 - *Manifestation* represents how the uncertainty appears or emerges. Possible options include system states, nature of input, availability of resources or future outcomes.
 - *Measure* is the expression of the degree of uncertainty. Depending on the uncertainty, measures might include probabilistic, ambiguous, non-specific and fuzzy values.
 - *Monitors* detect uncertainty manifestation in the sources of uncertainty such as the operational environment, business context and processes. The output of monitors contributes to evidence.
 - *Evidence* demonstrates the existence of uncertainty. Additional evidence, as objective facts, can reduce epistemic uncertainty.
 - *Relationship* is an intermediary function which can be used to confirm uncertainty manifestation in the absence of direct feedback of detecting uncertainty from monitors. Thus, a relationship can be used for inferring uncertainty from its manifestation location.
- Uncertainty time attributes:
 - *Emerging time* is when the uncertainty manifests during the system lifecycle, such as development time or run-time.
 - *Lifetime* indicates whether the uncertainty has a limited lifetime or exists in perpetuity.
 - *Change* indicates how uncertainty is influenced by time, determining whether it is dynamic

or static. Some uncertainties may change with time. For instance, epistemic uncertainty can disappear with more evidence as knowledge improves.

- *Pattern* represents the behaviour of uncertainty with time, such as systematic or aperiodic or other custom trends.
- Uncertainty mapping attributes:
 - *Dependency* identifies the uncertainties which are related to the current uncertainty. This information can facilitate tasks such as impact analysis using an uncertainty dependency graph and allow users to derive additional information regarding the system as a whole.
 - Uncertainty causes *Risks* or *Opportunities* which require *Mitigation* or *Exploitation* to achieve one or more specific *Outcomes*. Each of these can be individually characterised with further attributes in the context of specific systems.
 - *Operators* are key words, phrases or terms that can express or signal uncertainty or its constraints. The specific syntax and semantics of operators may require a grammar for their application and interpretation. Operators may also be used in ordinary or structured language to signal uncertainty [71]. We have discussed modal, temporal and ordinal categories of operators, as defined in RELAX [25]. In addition to these operators, ordinal operators AT LEAST and AT MOST, as used in software architecture facets for traceability [64, 105], and temporal operator AROUND <time> are included in the framework. Others may be added by users as required. We can also use operators to signal the scope or limits of uncertainty in desired uncertainty management outcomes. Table 6.2 of section 6.3.2 illustrates the use of operators in the description of outcomes.

Table 6.1 shows the consolidated attributes and their origin. Some attributes are used exactly as they are defined in the frameworks described in section 5.2. Others have been extended or introduced. The last column in table 6.1 distinguishes between the *Extended* or introduced attributes and those from existing work.

6.2.2 Applying the framework

This section outlines a possible process which can be used to apply the framework for the consideration, representation and management of uncertainty. This is not prescriptive and other workflows may be used depending on the specific context and need. This process illustrates the approach we used to identify and represent the uncertainty values of the case studies from Section 6.3.1.

Table 6.1: Consolidated uncertainty characterisation attributes.

CgS - cognitive science, CxS - complex systems, CPS - cyber-physical systems, SAS - self-adaptive systems, REL - the RELAX language

Attributes	Options	CgS	CxS	CPS	SAS	REL	Extended
Uncertainty description attributes							
Description ⁺	[What]	✓	✓	✓	✓	✓	No
Nature ¹	Aleatory or Epistemic	✓	✓	✓	✓		No
Bound [*]	Known limits around the uncertainty					✓	Yes
Perspective ¹	Objective or Subjective	✓		✓			No
Awareness ¹	Known unknown or Unknown unknown	✓	✓	✓			No
Level ¹	Certainty to uncertainty		✓	✓	✓		No
Uncertainty source attributes							
Source type ⁺¹	Endogenous or Exogenous		✓				No
Cause ⁺¹	[Why]	✓	✓	✓	✓	✓	No
Uncertainty system attributes							
Viewpoint ⁺¹			✓	✓	✓		Yes
Facets ⁺¹		✓	✓	✓	✓	✓	Yes
Location ⁺¹	[Where]			✓	✓		No
Uncertainty manifestation attributes							
Manifestation ⁺¹	Observable impact	✓	✓	✓	✓	✓	No
Measure ⁺¹		✓		✓	✓		No
Monitor [*]						✓	Yes
Evidence [*]				✓			No
Relationship ⁺¹	Between evidence and monitor			✓			Yes
Uncertainty time attributes							
Emerging time ⁺¹	[When]		✓	✓	✓		Yes
Lifetime ⁺¹	Limited or Perpetuity			✓	✓		No
Change ⁺¹	Dynamic or Static		✓	✓			No
Pattern [*]	Systematic - periodic, Systematic - persistence, Aperiodic - transient, or Aperiodic - Sporadic			✓			No
Uncertainty mapping attributes							
Dependencies [*]	with other uncertainties of the system					✓	No
Risk ⁺¹ or Opportunity ⁺¹	Negative or positive consequences of uncertainty		✓				No
Mitigation ⁺¹ or Exploitation ⁺¹			✓				No
Operator ⁺¹	Modal, ordinal and temporal operators to express managed uncertainty outcomes					✓	Yes
Outcome ⁺¹	Uncertainty management objectives		✓				No

Firstly, the target system and its boundaries are identified. By definition, we can only consider known uncertainties. However, stakeholders must be aware that currently unknown uncertainties are likely to emerge in future. The complete set of uncertainties comprises known and unknown uncertainties.

Secondly, the facets, in which uncertainties are recorded and analysed, are identified. For instance, facets of interest may include system requirements, architecture, design, implementation, hardware infrastructure and operating environment. All such aspects of the systems are vulnerable to uncertainty, and the framework can be used with them.

While facets are identified, it might be useful to consider the perspective or viewpoint from which uncertainties in each facet will be identified and captured. In the case of complex facets, uncertainties can be identified in individual viewpoints and later aggregated to understand or analyse the whole facet.

Thirdly, we identify the uncertainties in the system. Software engineers can use different approaches to identify these uncertainties, such as scenarios, brainstorming, experience reports, user feedback, technical specifications, budget details and risk assessment reports. Different sources of uncertainty information may need to be considered for different facets.

Fourthly, for each uncertainty identified, its specific details, as listed in Table 6.1 and Figure 6.1, are recorded. Uncertainties may be associated through the dependency attribute if there is a relationship amongst them.

Finally, depending on the context of the system and its environments, some of the attributes may be irrelevant and additional attributes may need to be introduced. The framework allows customisation in such cases.

The application of the framework can be an iterative process. It is possible and even likely that information about uncertainties is gained throughout the lifecycle of a system and it can be added at any stage. Uncertainty information about an existing or old system can be useful in anticipating uncertainties in new development scenarios.

6.2.3 Uncertainty representation and analysis

While details of representations and analyses of uncertainty data are outwith the scope of this chapter, this section briefly discusses possible options in relation to the proposed framework. The detailed illustrative evaluation is in Chapters 9, 10, and 11.

The uncertainty details identified using the framework can be represented in different ways to

suit the system context and anticipated uses including analysis. Our approach does not stipulate any specific notations. Possible forms of representations include tables, property graphs, data interchange formats such as JSON, and visualisations. The rest of the thesis uses tables to present the uncertainty data.

Different kinds of analyses can be carried out on the uncertainty data to support these tasks. Depending on stakeholder needs, these can include determining the most vulnerable parts of a system and calculating overall uncertainty measures for a system including the probability of specific outcomes. Techniques from simple counts to Bayesian inference and logics for knowledge and belief can be used for such analyses.

Mitigations for uncertainties that lead to unacceptable risks can be determined based the analyses carried out. Again, the framework does not stipulate specific techniques or tactics for mitigating risks since these will be context and system dependent.

6.3 Evaluation and discussion

In this section, we illustrate the application of the framework to case studies (sections 6.3.1 and 6.3.2), demonstrate the extensibility of the framework (section 6.3.3), and discuss the efficacy of the overall framework (section 6.3.4).

6.3.1 Case studies

This section introduces case studies to illustrate and evaluate the use of the framework to consider and characterise uncertainty in different contexts. Specifically, we use a Big Data platform by the National Aeronautics and Space Administration (NASA), an Aerial Robotics Architecture (ARA) and IoV. In these contexts, the framework is used as the basis for generating data on uncertainties and their characteristics in the system architecture design facet. This data can be used to facilitate processes such as uncertainty analysis in the architecture, risk assessment and evaluation of system qualities. Such explicit and systematic characterisation of uncertainty has potential for other uses, some of which are presented in the Discussion and Future Work sections. Although the examples we identify in these case studies are from the system architecture facet, other facets, such as design, requirements and hardware, can be similarly explored.

6.3.1.1 The Information Management Platform for Data Analytics and Aggregation

IMPALA¹ is a Big Data reservoir for medical and health data of astronauts used by NASA. We identified some of the uncertainties applicable to the architecture of IMPALA by reviewing its System Design Document (SDD), which describes its requirements, system design and data design.

The SDD is a live document that provides the latest information about IMPALA to stakeholders, including the IMPALA infrastructure team, data architecture team, system integration team, security management team, project manager, NASA data scientists and users. Therefore, the uncertainties identified from it are likely to be relevant to a range of stakeholders.

The IMPALA architecture facet has the following views: Logical, Functional, Infrastructure, Network and Security. Each viewpoint is described in the SDD using a diagram showing the relevant architecture elements from the perspective of the viewpoint. We identify uncertainties from each of these viewpoints, and list two uncertainties in the logical view in Table 6.2. While uncertainties have been identified from the description in the SDD, we infer the values of some of the attributes of the uncertainty, such as the the nature of the uncertainty, from the context when they are not explicitly expressed.

6.3.1.2 An Architecture and Open-Source Software Framework for Aerial Robotics [5]

Unmanned Aerial Systems (UAS), more popularly known as Unmanned Aerial Vehicle (UAV), are now used as part of critical systems in various domains from business activities and health to military. However, their use needs to be simplified through fully autonomous operation to facilitate their adoption in different domains. AEROSTACK is one of the proposed autonomous architectures for UAS and aims to provide potential benefits such as reducing limitations to its customisation and supporting versatility since it is not designed for a specific application or aerial platform. Since AEROSTACK operates in dynamic environments, has a complex system configuration, and is an open-source multi-purpose software framework for autonomous multi-UAS operation, it can be vulnerable to a range of uncertainties.

The AEROSTACK architecture includes self-adaptation aspects and is organised in five layers: Reactive, Executive, Deliberative, Reflective and Social [5]. In its physical infrastructure, sensors and actuators are some of the components vulnerable to uncertainty. The AEROSTACK architecture has a range of communication channels through a wireless network, including human-robot (human sending commands to UAS) and robot-robot communication.

¹<https://ntrs.nasa.gov/citations/20160011412>

We have analysed the description of AEROSTACK in literature [5] and related GitHub documentation² to identify uncertainties and express them using the uncertainty framework. Such knowledge can be used for a range of purposes, such as tracking the status of uncertainty risk mitigations, and marking handled uncertainties as closed.

6.3.1.3 Internet of Vehicles

This case study is from Intelligent Transport Systems (ITS) [6, 7, 16]. IoV is a specific category of ITS and an application of the IoT. In the context of IoV, each vehicle is considered a node in the network of the IoT with vehicles connected to the Internet. IoV makes use of concepts and technologies such as intelligent network systems and applications and software defined networks, and they operate in complex environments. IoV have various potential communication channels such as between vehicles and other vehicles, among devices within a specific vehicle, between vehicles and road infrastructure and pedestrians. In fact, the potential communication in IoV is defined as between vehicle and everything.

IoV have a range of potential benefits including the sharing of information and communication among vehicles, and between vehicles and transport infrastructure or pedestrians or other relevant entities. However, potential uncertainties relating to aspects such as coordination between vehicles, insufficient information and scalability, also arise from these multiple interactions. Nonetheless, advances in communication and computing technologies such as cloud and/or fog computing, Big Data processing and analytics, machine learning, and artificial intelligence present opportunities to mitigate such challenges. Here, we identify uncertainties which relate to IoV in literature [6, 7, 16]. A list of common uncertainties can be used to compare the designs of IoVs, based on their ability to handle and control such common uncertainties, as an example of using the uncertainty data for system design analysis.

6.3.2 Illustration of uncertainties in case studies

Table 6.2 contains two uncertainties from each of the case studies. It illustrates how uncertainties may be characterised using the attributes of the framework. The first column lists the uncertainty attributes and the rest of the columns contain data from the case studies corresponding to these attributes.

Uncertainties in each system were identified though analysing available documentations for the system. As indicated in Table 6.2, values of some of the uncertainty attributes were explicitly identified in the documentation, while others were inferred from the context. While the

²<https://bit.ly/3MGgJV7>

Table 6.2: Examples of uncertainty framework values

Attribute	Uncertainties						
System	IMPALA platform architecture		Aerial Robotics architecture		Internet of Vehicles (IoV) architecture		
Example #	1	2	1	2	1	2	Status
Description	Future sources of data	Data importation capacity	timing and sequence of UAV swarm interactions	Network availability	Data volume growth	IoV computation capability	Explicit
Nature	Epistemic	Epistemic	Epistemic	Aleatory	Epistemic	Epistemic	Inferred
Bound		up to 100% data				up to 100% capacity	Inferred
Perspective	Objective	Objective	Objective	Subjective	Objective	Objective	Inferred
Awareness	Known unknown	Known unknown	Known unknown	Known unknown	Known unknown	Known unknown	Inferred
Level	Medium	Medium	Medium	Medium	High	Medium	Inferred
Source	Exogenous	Exogenous	Exogenous	Endogenous	Exogenous	Endogenous	Explicit
Cause	Emergence of new data source(s)	Data volume growth from sources	UAV joining or disconnecting from the swarm	Network hardware issue	Increase in data collection	Limited computing process power	Inferred
Viewpoints	Logical, Deployment, Physical	Logical, Deployment, Physical, Process	Logical, Physical, Process	Logical, Physical, Network, Security	Logical, Deployment, Physical, Process	Logical, Physical, Process	Explicit
Facets	Architecture	Architecture	Architecture	Architecture	Architecture	Architecture	Explicit
Location	Capture component	Transport component	Social layer	Reflective layer	Data acquisition layer	Processing layer	Explicit
Manifestation	New data sources	Data import delays	UAV coordination and interaction failure	Failure to receive commands	Data volume	Processing speed	Inferred
Measure	Probability	Importation fluctuation	UAV swarm timing misalignment	Timeout period	Data growth rate	Processing fluctuations	Inferred
Monitor	Governance - data integration requests	Data importation rate	Coordination failure report	Connectivity monitor	Data volume against storage and processing	Processor monitor	Inferred
Evidence	New data source	Data importation history/trend	UAV coordination	Connectivity	Data volume	Processing trends	Inferred
Relationship		Data importation congestion					Inferred
Emerging time	Run-time	Run-time	Development, & Run-time	Run-time	Run-time	Run-time	Inferred
Lifetime	Perpetuity	Perpetuity	Perpetuity	Perpetuity	Perpetuity	Perpetuity	Inferred
Change	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic	Inferred
Pattern	Aperiodic-sporadic	Aperiodic-sporadic	Aperiodic-sporadic	Aperiodic-sporadic	Aperiodic-sporadic	Aperiodic-sporadic	Inferred
Dependencies	Transport & Refine components uncertainties	Refine component uncertainties	Interactions with other UAV uncertainties	Communication uncertainties	Data use applications uncertainties	Security, control and management uncertainties	Explicit
Risks or Opportunities	Risk: Data Integration failure	Risk: Data loss & import delays	Risk: Data import delay and failure	Risk: Accidents among UAV and coordination failure	Risk: Loss control of UAVs	Risk: IoV slow, real-time processing failure, potential accidents etc.	Inferred
Mitigation or Exploitation	Mitigation: Extensible design to integrate with new data sources	Mitigation: Transport layer extensible design to increase capacity through custom scripts	Mitigation: UAV autonomy to independently enable and manage coordination	Mitigation: UAV autonomy to complete mission without human intervention	Mitigation: data processing and filtering layer for removing irrelevant data	Mitigation: Cloud computing - both private and public	Explicit
Outcome (with operators)	Integration with AS MANY future data sources AS POSSIBLE	AS MUCH data AS available from current sources	Enable UAS coordination AS EARLY AS POSSIBLE	MAXIMISE UAV independence to accomplish missions	MINIMISE irrelevant data capturing AS MUCH AS POSSIBLE	MAXIMISE IoV components' and application's computation power and speed	Inferred

framework proposes a core set of attributes for characterising uncertainty, it does not prescribe a data format in which attribute values should be stored. For readability, a tabular format is used here.

We will now go through the values assigned to uncertainty 'Example #1' of the IMPALA case study to illustrate how framework attribute values may be determined.

This uncertainty relates to *Future sources of data*, as represented by the *Description* attribute. Since we cannot fully know which data sources might have to be integrated within the IMPALA platform in future, the *Nature* of this uncertainty is *Epistemic* due to the lack of knowledge.

The *Bound* attribute is blank because there are no specific boundaries to the uncertainty.

Current and new data sources can be specifically and explicitly identified, and therefore, the *Perspective* of this uncertainty is *Objective* as its identification does not depend on knowledge of an individual stakeholder or agent.

In terms of *Awareness*, this is a *Known unknown* uncertainty as it is anticipated. The *Level* of uncertainty is estimated to be *Medium* and its *Source* is external or *Endogenous* to the IMPALA platform. The *Cause* of the uncertainty is the *Emergence of new data source(s)*.

The logical Viewpoint is where we explicitly identify the uncertainty. However, this uncertainty may also impact *Deployment and Physical* views. All these views are related to the *Architecture Facet* in which we consider this uncertainty.

Specifically, the *Location* of the uncertainty is the *Capture component* where various data sources are integrated with the IMPALA platform, and its *Manifestation* is through the appearance of *New data sources*.

The *Monitor* of the uncertainty is through *Governance in the form of data integration requests* since new additions of data sources to the IMPALA platform will have to be approved by the Governance team for security purposes before they are added to the system.

Similarly, the evidence which we can use to reduce the uncertainty is the knowledge of *New data sources*. The *Measure* of this uncertainty is *Probability* and it does not have any specific *Relationships*.

The *Emerging time* of the uncertainty is during system *Run-time* and the uncertainty will exist (its *Lifetime*) as long as the system is in operation, thus it is in *Perpetuity*.

The uncertainty might *Change* with time, therefore, it is *Dynamic*. The *Pattern* of change is irregular since new data sources might be available on demand but not on particular intervals,

Table 6.3: Extending mapping - Mitigation attribute

Attribute	Options
Mitigation	Preferred
	Possible
	Plausible
	Probable

thus it is *Aperiodic - sporadic*.

Dependencies include *uncertainties in Transport & Refine components*. The uncertainty causes the *Risk of Data Integration failure*. This requires *Mitigation* in the form of *Extensible design to integrate new data sources* with the goal of achieving the *Outcome of Integration with AS MANY future data sources AS POSSIBLE*.

Thus, uncertainty relating to *Future data sources* of the IMPALA platform is captured with the framework. The other uncertainties can be interpreted in a similar way from Table 6.2.

Among the benefits of this encoding of these uncertainties is to support uniformity of expressing the uncertainties. The data thus generated can then be used to analyse and manage the uncertainty, as we discuss under the *Discussion*, Section 6.3.4.

6.3.3 Demonstration of extensibility: Future studies concept

To demonstrate extensibility of the framework, we use the concept of futures. Futures Studies (FS) recognise the existence of infinitely many potential futures, rather than a single future [113]. The futures concept has four categories: *possible*, *plausible*, *probable* and *preferable*.

Possible futures are the set of all futures imaginable, including those we cannot currently realise because they depend on future knowledge. *Plausible* futures are the set of futures that we can achieve, i.e. those that are feasible. *Probable* futures represent likely futures which we can realistically achieve, for example, based on current trends. Finally, *Preferred* futures represent the futures we desire.

The concept of futures can be used to extend the mitigation attribute of the framework such that we can represent *Preferred*, *Probable*, *Possible* and *Plausible* mitigations, as shown in Table 6.3. This extension allows system architects and designers to capture not only the mitigation that is finally chosen but the other options considered and their viability.

6.3.4 Discussion

The central hypothesis of this work is that a generic and useful conceptual framework for considering uncertainty in software systems can be defined. We have observed that uncertainty is characterised with distinct attributes in different contexts, potentially omitting useful details. We have consolidated and extended attributes from four relevant contexts found in literature to propose a generic, foundational and customisable framework of uncertainty for the broader context of software systems. The use of the framework is demonstrated using case studies from three different application domains.

The process of providing values for the proposed attributes contributes to the identification, representation and management of known and anticipated uncertainties. Management includes the analysis of the data to control or mitigate uncertainties and make more informed decisions throughout the software lifecycle. For example, a record of aleatory and epistemic uncertainties as part of the knowledge database of a system would facilitate the assessment of risks and consideration of mitigations, and help track the status of known uncertainties.

The framework is not limited to specific development methodologies, system artefacts, notations or levels of abstraction. It is intended to be applied to different artefacts and stages of the software lifecycle.

Examples from case studies demonstrate that descriptions of uncertainty, its sources, manifestation, timings and dependencies together with risks, opportunities, mitigation, exploitation and expected outcomes can be gathered using the framework, thus providing data for further analysis. The framework provides a basis for extensibility since attributes can be included or removed based on emerging information.

However, manually compiling all uncertainties associated with any non-trivial system would be a time-consuming activity, in addition to being error-prone. Since the framework provides a structured and systematic basis for capturing uncertainty data, automation may be a feasible alternative for applying the framework on a larger scale. With automation, rules for associating uncertainties with specific patterns of structure and interactions in the system can be specified and used to generate an initial set of uncertainty data that can be manually augmented by stakeholders.

While we have provided a definition of each attribute in the framework, we have deliberately avoided stipulating any processes or guidelines for populating these attributes. Therefore, the granularity of the data might depend on the context of the specific system, the stage of the lifecycle at which uncertainty is considered, and the aims of the users of the framework. It is likely that values of attributes with well-defined options such nature, which can be either aleatory

or epistemic, will be uniform across systems. However, details such as description, dependencies and outcomes might vary from the examples provided and across different systems.

For example, if the framework is used to capture uncertainties in the software architecture of a system, attribute values are likely to depend on the specific architecture viewpoint and the architecture model chosen. The logical viewpoint might have uncertainties relating to the system topology or configuration during runtime, while the process viewpoint might focus on uncertainties relating to system processes and interactions at runtime.

This chapter illustratively demonstrates the feasibility of defining a generic and extensible framework from individual frameworks in existing literature sources. The framework is not designed to be exhaustive or mandated on specific methodologies, but can be customised to specific use cases.

The data generated can be used to assess the uncertainty in different system facets. For example, in the case of software architectures, uncertainty information can be used in the evaluation of candidate architectures and the selection of a suitable architecture design solution. Such evaluation can be stand-alone focusing on uncertainty, or carried out as part of a broader evaluation process such as ATAM [32, 114].

Besides, we anticipate that a conceptual framework such as the one proposed can also aid in solving problems such as planning under epistemic uncertainty [115].

As the framework is applied to different facets of software systems, its effectiveness and viability will be continuously evaluated and refined. As far as we are aware, no work currently exist that expresses uncertainty attributes in a generalised format for software systems.

6.4 Conclusion

This chapter has presented a generic foundational conceptual framework for the systematic consideration of uncertainty in software systems. This is the beginning of what we believe to be an exciting and important avenue of future research. The rest of the thesis will demonstrate the feasibility of applying the framework in software architecture on three case studies. As the uncertainty framework is applied to different facets of software systems, its effectiveness and viability will be continuously evaluated and refined. As far as we are aware, based on our comprehensive literature reviews, no work currently exist that expresses uncertainty and its attributes in a generalised format for uncertainty consideration in facets of software systems such as software architecture.

THE WORKBENCH INFRASTRUCTURE

This chapter presents the workbench infrastructure we propose as a foundation for the identification, enhancement, and realisation of software architecture tools and notations that can support the representation of uncertainty in software architecture descriptions based on the proposed framework.

7.1 Overview

Our vision is to develop a workbench of tools that can support the consideration of uncertainty in software architecture. To achieve this, we first have to define a foundation on which the workbench of tools will be developed. We refer to such a foundation as the workbench infrastructure.

This chapter presents the workbench infrastructure and an illustrative example tool. The workbench infrastructure is designed as a proof of concept to demonstrate one of the potential applications of the uncertainty framework. So far, we have implemented a tool for capturing the uncertainty framework attributes in software architecture description on the workbench infrastructure. The infrastructure itself is realised using a existing open-source software diagramming tool.

The rest of the chapter is presented as follows: Section 7.2 presents the concepts of the workbench infrastructure, Section 7.3 presents the design of the workbench infrastructure, Section 7.4 discusses our proposed solution to the realisation of the workbench infrastructure. Section 7.5

discusses the implementation of the workbench. Then Section 7.6 presents a critical discussion of the workbench infrastructure. Finally, Section 7.7 is the conclusion.

7.2 Concepts of the workbench infrastructure

There are various tools and notations which are used to capture and document software architecture. Broadly, these are categorised into formal, semi-formal and casual, and these, in industry, have been adopted with various degrees of success [107].

In practice, semi-formal approaches, such as diagramming tools, which include UML notation, or other similar notations, are often used. In addition, casual notations, such as boxes-and-lines, are common approaches to diagramming software architecture, as they are not prescriptive and thus convenient for quick sketches in architectural design discussions or documentation [107].

Formal notations such as Description Language (ADL) are often research-focused and are rarely adopted in practice [107]. Various formal notations have been proposed over the years, but these have not been widely adopted [74, 107], as we previously discussed in Chapter 4.

In general, there is a challenge with the adoption of various software architecture tools and notations by practitioners [107]. Thus, our approach to considering uncertainty in software architecture does not introduce new specialised tools or notations. Instead, we propose that the framework should be incorporated in existing tools and notations which software architects use to capture software architecture.

However, if existing approaches are to achieve such enhancements, they have to meet some specifications for data collection and analysis. In this regard, we define the workbench infrastructure, which is a framework that specifies the criteria that architecture tools have to meet so that they can be enhanced with uncertainty data capture and analysis capabilities.

Thus, we propose the workbench infrastructure. In Section 7.3, we present a layered workbench infrastructure architecture and discuss the rationale for its design. Then in Section 7.4, we present a possible approach to its realisation.

7.3 Design of the workbench infrastructure

The workbench infrastructure is based on a web-based model, considering that software architecture design is a collaborative process, therefore, an online solution can conveniently support such collaborative features. Figure 7.1 shows the logical structure of the workbench

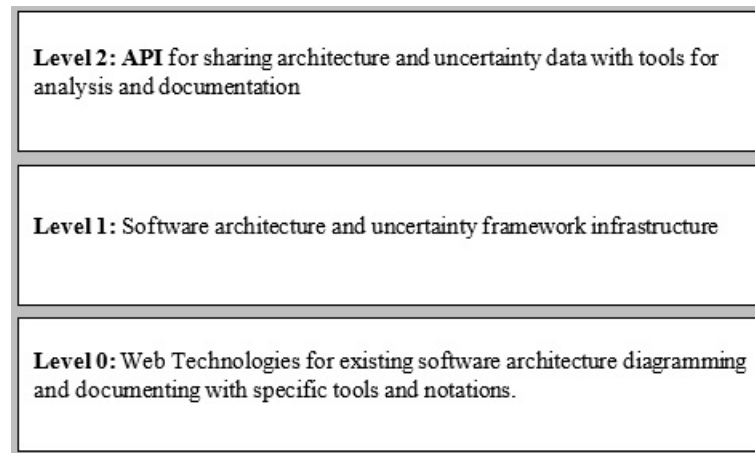


Figure 7.1: Logical view of the technology stack of the workbench infrastructure

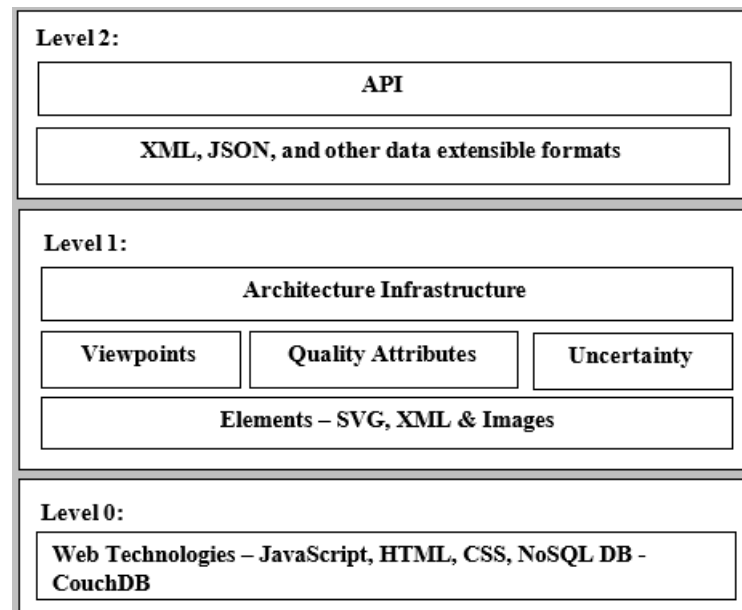


Figure 7.2: Technology stack of the workbench infrastructure

infrastructure. In Figure 7.2 we present the architecture of the workbench based on the stack of the proposed implementation technologies.

The workbench infrastructure has three layers, as logically presented in Figure 7.1. These layers focus on web technologies, the uncertainty framework, and sharing or interoperability of the software architecture uncertainty data for analysis or documentation.

Level 0 - proposes that the workbench should be designed using web technologies such as JavaScript, HTML, etc. so that the workbench is web-based. As stated earlier, a web-based approach allows the workbench to include the benefits of the Web, including accessibility,

modifiability, collaboration, and interoperability, among other benefits.

Level 1 - is the software architecture and uncertainty framework layer. This layer includes software architecture features such as diagramming elements, architecture representation elements such as viewpoints, and architecture quality attributes. Furthermore, the layer includes the uncertainty framework attributes to capture details about uncertainties of specific architecture elements or views. A key feature in this layer is that the elements for diagramming architecture should be in an extensible data format so that the data can be available for analysis.

Level 2 - focuses on enabling the sharing and interoperability of the uncertainty framework data with other relevant applications for analysis and documentation. We envision that the uncertainty data will provide a platform for various software architecture uncertainty analysis. Therefore, the API should be able to share uncertainty data in extensible formats such as JavaScript Object Notation (JSON) or Extensible Markup Language (XML) to support integration with external analysis applications.

As part of the thesis evaluation, we demonstrate some of the possible analyses that can be conducted on the basis of the uncertainty data. However, the evaluations are performed using manual processes, with the uncertainty framework data captured in tabular format.

7.4 Realisation of the workbench infrastructure

An existing open-source diagramming software, *diagrams.net* (previously known as *draw.io*)¹, has been cloned and extended to implement the workbench infrastructure. Tools can be implemented in this implementation of the workbench infrastructure. Currently, we have implemented a single tool on it as an exemplar. The tool can be used to capture the details of uncertainty in software architecture diagrams.

diagrams.net uses an extensible representation format in which each element of a diagram is represented as a vertex of a graph, and connections are represented as edges of the graph, based on the JavaScript mxGraph Library². The range of diagramming notations within *diagrams.net* is not prescriptive, since it supports many of the major diagramming notations, including boxes-and-lines, Systems Modeling Language (SysML) and UML.

Due to its design, *diagrams.net* meets the specifications of each of the levels of the technology stack defined in the workbench infrastructure of Figures 7.1 and 7.2.

¹<https://github.com/jgraph/drawio>

²<https://jgraph.github.io/mxgraph/docs/manual.html>

First, for *Level 0*, *diagrams.net* is a web-based diagramming software with various diagramming tools and notations built using open-standard web technologies such as JavaScript, HTML, and CSS. Of course, it has an equivalent Java implementation; however, our work focuses on its JavaScript implementation.

In terms of *Level 1*, *diagrams.net* stores its various images and data in extensible formats such as XML. Besides, this data can be seamlessly converted or integrated with JSON representation, as we will demonstrate with the implementation of the uncertainty capture tool in Section 7.5.

Finally, due to the ability to capture details in an extensible format in *diagrams.net* and since each element or object within the workbench infrastructure can be uniquely identified with an identifier, *diagrams.net* enables the support of an Application Programming Interface (API) which can be used for create, read, update, and delete (CRUD) functionalities to export, document and analyse the uncertainty data. This meets the requirements of *Level 2*.

7.5 Implementation of the uncertainty capture tool

Based on the workbench infrastructure, we implemented a prototype tool to capture uncertainty within an architecture description through *diagrams.net*. Each uncertainty is associated with the relevant architecture element, and its attributes are recorded. Uncertainties can be created, read, updated, and deleted from an architecture element. These actions are achieved through the manipulation of the XML and JSON data using JavaScript methods and the web user interface to interact with the data and the graphics.

In addition to recording uncertainty data as part of the architecture model, users can calculate metrics such as the number of uncertainties per architecture component.

Since *diagrams.net* supports the encoding of architecture diagrams in data formats such as XML and JSON, the architecture models and the associated uncertainty data can be exported for further processing in other tools or be recorded as part of the architecture knowledge for uncertainty management activities.

Figure 7.3 shows a screenshot of the logical view of an architecture with the uncertainty framework attributes on the right panel. The XML source code of this screenshot which includes the uncertainty details is in Appendix B, under Listing B.2. As can be seen from the encoding in the Listing B.2, the data and meta-data of a view is a large file and presents an opportunity for automated processing.

Listing 7.1 shows the partial JSON template of the uncertainty data storage format. The complete

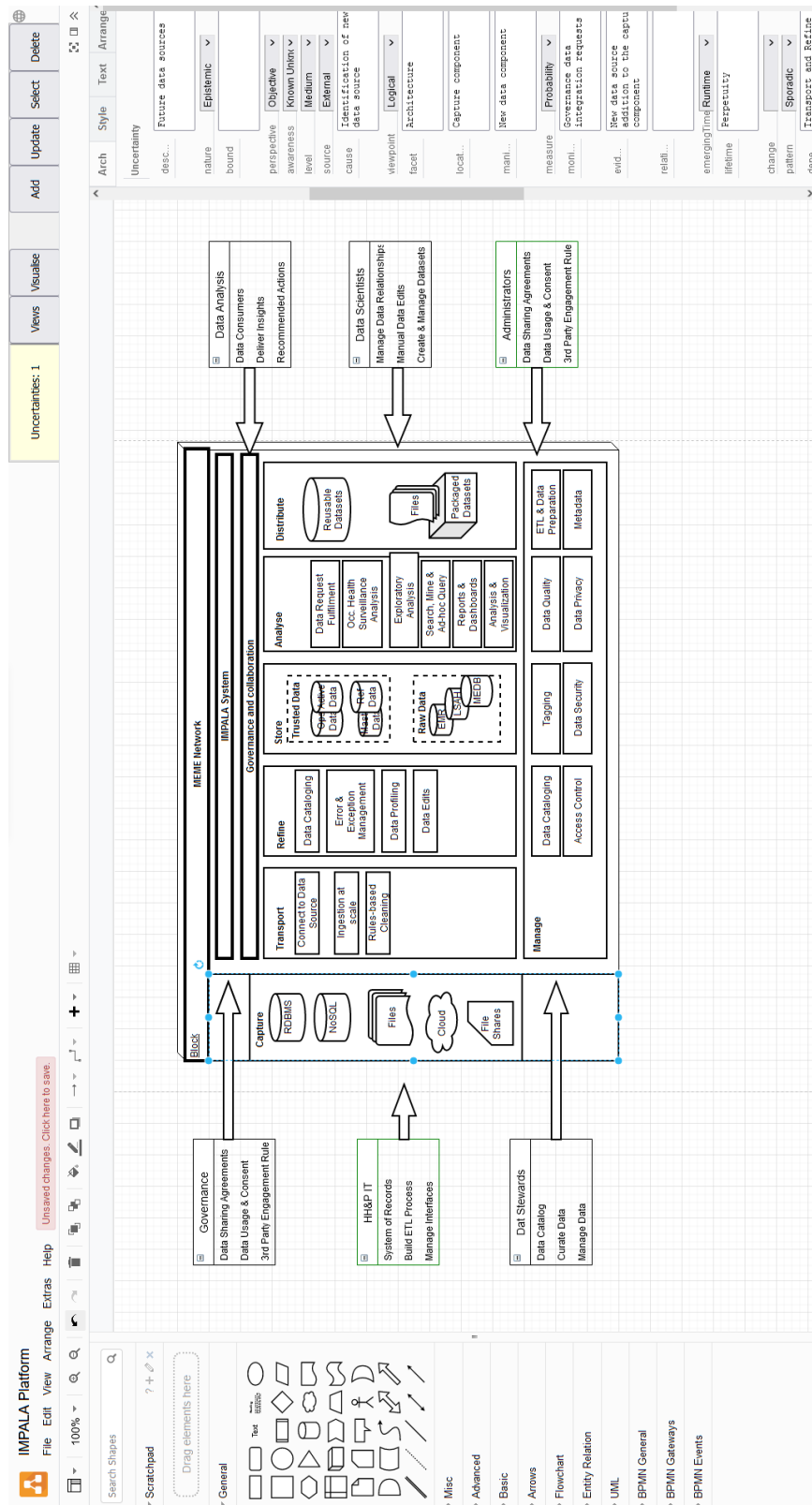


Figure 7.3: Screenshot of the workbench (XML source code for the screenshot is in Appendix B Listing B.2)

JSON object of an individual uncertainty is in Appendix B - Listing B.1. The workbench relates uncertainties, architecture views, and components. Each uncertainty has a unique ID that can be used, for example, to map the relationships between the uncertainties.

diagrams.net handles each element of an architecture as a vertex of a graph, and the connections are represented as edges of a graph. Therefore, our approach to incorporating uncertainty involves encoding the uncertainty data of an architecture component into a JSON format and associating it with its relevant component.

```
{
  "id":1642559068879,
  "facet":"",
  "manifestation":"",
  "monitor":"",
  "bound":"",
  "nature":{
    "options":[
      "Aleatory",
      "Epistemic"
    ],
    "value":""
  },
  "perspective":{
    "options":[
      "Subjective",
      "Objective"
    ],
    "value":""
  },
  "source":{
    "options":[
      "External",
      "Internal"
    ],
    "value":""
  },
  "viewpoint":{
    "options":[
      "Logical",
      "Physical",
      "Process"
    ],
    "value":""
  },
  .
  .
  .
}
```

Listing 7.1: Partial JSON data

7.6 Critical discussion of the workbench infrastructure

In this chapter, we have introduced the concept of a workbench infrastructure. In this section, we present its critical discussion in terms of the following: First, the validity of the concepts of the workbench infrastructure. Second, its implementation: What are the suitable tools and notations to implement the workbench infrastructure? Third, the application of the workbench infrastructure.

7.6.1 Validity of the workbench infrastructure concept

Chapter 4, which presents a survey of uncertainty in software architecture, presented among other aspects of software architecture which deal with uncertainty, the tools and notations aspects. There, we highlighted 12 tools which incorporate various capabilities of considering uncertainty in software architecture. However, none of these has the capabilities to capture uncertainty data to generate architecture uncertainty knowledge documentation. As such, the workbench infrastructure addresses a significant gap in terms of tools and notations support for uncertainty consideration in software architecture. Of course, the idea of an architecture workbench is well established [116, 117]. However, our specific approach is in the context of software architecture and architecture uncertainty data generation and analysis.

7.6.2 Workbench infrastructure tools and notations support

The three levels of the workbench infrastructure provide a criterion for identifying suitable software architecture tools and notations to support the incorporation of uncertainty in software architecture. *Level 0*, *Level 1* and *Level 2* define the specification of existing architecture tools and notations that can support documentation and analysis of uncertainty in architecture uncertainty. We illustrate the application of this criterion to identify *diagrams.net* as a proof of concept.

7.6.3 Application of workbench infrastructure

The workbench infrastructure forms a base for building tools to support the incorporation and analysis of uncertainty in software architecture. So far, we have built a tool to capture the uncertainty details of the software architecture based on the workbench infrastructure.

As a proof of concept, the implementation of the workbench infrastructure has been demonstrated as feasible, based on its realisation through the *diagrams.net* platform. As discussed in Section 7.5, *diagrams.net* meets the requirements of the implementation stack. Uncertainty data are captured in interoperable and extensible data formats such as XML and JSON so that it can be exported to external tools for documentation or analysis. The Listings in Appendix B presents an overview of the data and meta-data of the realisation of uncertainty framework.

7.7 Conclusion

In this chapter, we have demonstrated the concept of the workbench infrastructure, which can be a basis for building tools with capabilities to incorporate uncertainty into software architecture. Representing the uncertainty data into extensible formats such as XML and JSON, while associating it with architecture elements, can facilitate architecture uncertainty data documentation and analysis automation. The automated data can be exported to tools for further processing. The application of the framework does not depend on it being automated or incorporated into a workbench. But for effective and efficient application of the uncertainty framework, future work will require automation, considering the size of the data.

EVALUATION STRATEGY

We qualitatively evaluate the uncertainty framework by applying it in the context of software architecture in three case studies. These case studies cover the following scenarios: 1) Architecture uncertainty knowledge generation and documentation. 2) Architecture uncertainty analysis, and 3) a comparison of candidate architectures to select an architecture likely to be less vulnerable to the known uncertainties. The framework has been applied in an independent project elsewhere, in this Chapter, we also highlight these results [17].

8.1 Introduction

The evaluation aims to illustrate the application of the framework as an approach to uncertainty analysis of software architecture. The evaluation is based on systems with existing architecture artefacts, such as views and requirements. The rest of the chapter presents the following: Section 8.2 discusses the evaluation strategies. Section 8.3 presents the selection criteria for the case studies for the evaluation. Then 8.4 presents the three case studies. Section 8.5 is the evaluation process. Section 8.6 discusses the ATAM and its customisations. Section 8.6.3 highlight independent research results which used the framework from published work [17, 118]. Finally, Section 8.7 is the conclusion.

8.2 Evaluation strategy

In conducting the architecture uncertainty documentation and analysis, our goal is to generate insight into the software architecture case studies with respect to uncertainty. Stakeholders can consider the analysis results to make decisions to improve the architecture or simply as architecture knowledge and documentation for various uses.

The first scenario of the evaluation aims to illustrate the use of the framework to capture and present uncertainty data. The results are presented in a table format and we discuss these.

The second evaluation scenario aims to conduct a software architecture uncertainty analysis. The analysis aims to illustrate that the uncertainty data can be used to identify potential areas of focus in a software architecture which might for instance require improvement or special attention.

To critically review the validity of the results of the uncertainty framework analysis, we compared the framework results with the results of applying a customised version of ATAM on the same architecture case study. ATAM is a generic qualitative analysis approach for software architecture [119]. We used the ATAM as a baseline for assessing the uncertainty framework analysis results.

ATAM is a comprehensive architecture analysis method with specific steps, but these steps can be customised before being applied to specific projects and context [119]. In our evaluation, we customised the ATAM to meet one of the key constraint of our evaluation, that the evaluation is being conducted by an individual person, the author. In Section 8.6 we present the customised version of the ATAM with justifications of the customisation decisions.

The third evaluation case study assesses the use of the framework to compare candidate architectures. In this regard, we identify known uncertainties, and assess which architecture best addresses the identified uncertainty.

It is important to note that architecture analysis in general and, with respect to uncertainty in particular, is about providing insights into the architecture to stakeholders [119]. The follow up actions after the analysis report depend on the priorities of stakeholders [119]. The analysis only identifies potential issues or areas of focus in the software architecture [119].

8.3 Architecture case study selection criteria

The evaluation used systems whose software architecture might be vulnerable to significant uncertainties. The following are the key points in the selection criteria of the case studies:

1. Architectures of open source systems with characteristics such as being distributed across multiple machines or operating on a network with multiple components or interacting with multiple activities from system agents, users and data sources.
2. The availability of the software architecture information: one challenge with evaluating the architecture of software systems is the lack of comprehensive documentation about the system design. Often, even if the source code of a system is openly available on public repositories

such as GitHub, there is generally a lack of architecture and design documentation for the software system.

3. The architecture needs to have at least one architecture viewpoint documented: software architectures are presented from multiple viewpoints to capture interests of stakeholders. These include logical, physical, process and development viewpoints. Each of such viewpoints might help with considering specific uncertainties.
4. The architecture documentation should include details of architecture styles, quality attributes and use case scenarios so that the architecture can be analysed with the customised version of the ATAM.
5. The architecture uncertainty context should be explicit. We need to specify the uncertainty environment or sources where we are exploring the uncertainty. Uncertainty can manifest in various areas such as security, data, hardware, development and others. The case studies should present a definite context where we explore the uncertainty.

8.4 The case studies and evaluation

The following is an overview of each of the three case studies for evaluating the uncertainty consideration framework. Table 8.1 shows the selection criteria for these three case studies.

1. *An Architecture and Open-Source Software Framework for Aerial Robotics* [5].

In this evaluation, we analyse the documentation of AEROSTACK to extract uncertainties and encode them using the uncertainty framework. Thus aiding with uncertainty knowledge generation and documentation about the AEROSTACK architecture. Such knowledge can be used for a range of purposes, for instances tracking the status of implementing the uncertainty risk mitigations, and the closing the handled uncertainties.

2. *The Information Management Platform for Data Analytics and Aggregation (IMPALA)*¹

In this evaluation, we identify the uncertainties applicable to the architecture of IMPALA by reviewing its SDD, which describes its requirements, system design and data design.

The SDD is a living document that captures capability enhancements and system improvements to provide accurate details about IMPALA to stakeholders, including the IMPALA Platform infrastructure team, data architecture team, system integration team, security

¹<https://ntrs.nasa.gov/citations/20160011412>

management team, project manager, NASA data scientists and users. Therefore, the uncertainties identified from it are relevant to a range of stakeholders.

The IMPALA platform architecture description has the following views - Logical, Functional, Infrastructure, Network and Security. Each viewpoint is described in the SDD using a diagram showing the relevant architecture elements from the perspective of the viewpoint.

3. *Comparison of candidate architectures for emerging Internet of Vehicles (IoV)* [16]

Various architectures of IoV have been proposed with different strengths. Under this evaluation, we compare such candidate architectures of the emerging IoV basing on whether they handle the common identified IoV uncertainties. Our approach is to identify common uncertainties about IoV from the documentations then compare and evaluate the IoV candidate architectures based on their handling of such common uncertainties. Thus, we illustrate the potential application of the uncertainty framework data to compare and rank candidate system architectures.

Table 8.1: Selection criteria for the three case studies summary

	Criteria	IMPALA	AEROSTACK	IoV
1	System architecture overview and complexity	Big Data analytics system built from assembly of Commercially available Off-The-Shelf (COTS) components, multiple data sources including future unknown data sources, infrastructure includes networking, distributed storage and computing, and security consideration	Architecture for UAS is design to be hardware independent thus flexible to operate on a range of UAS hardware and sensors. Its self-adaptation and configured software components	IoV system architecture
2	System Documentation	Architecture design documentation within the SDD including architecture views, architecture styles, quality attributes, and scenarios, suitable to support the ATAM	Existing documentation include published AEROSTACK architecture design and evaluation research papers and a GitHub repository which includes, code, user manual and other documentation	Existing literature survey and research papers
3	Architecture Viewpoint	Logical, Functional, Infrastructure, Network & Security views	Logical (with Reactive, Executive, deliberative, reflective and social layers) & development views	Logical & network views
4	Uncertainty context	Uncertainties including those related to data and data sources, networking, hardware, security, system processing and performance	UAS uncertainty include vulnerable from environment, sensors, intelligence, cognition, compatibility with other system, autonomy etc.	Deployment environment & architecture components

8.5 The case studies evaluation approaches

8.5.1 Case study 1: Knowledge generation: uncertainty capturing and documentation

The evaluation for the first case study focuses on demonstrating the application of the uncertainty framework to capturing and representing uncertainties. Such architecture knowledge and documentation can be useful to stakeholders that use the architecture documentation. Steps for uncertainty identification and capturing are as follows:

1. First identify the system whose uncertainty is to be documented
2. Then decide the viewpoints from which the uncertainty is to be identified and presented
 - The decision of the viewpoints is based available architecture description artefacts - for example, we may consider the logical, physical, or data views.
3. Decide the sources of uncertainty information about the system. This depends on the available resources, such as documentation and other artefacts of the systems.
4. After the uncertainties are captured and represented, we need to check the following about the uncertainty data.
 - a) On the uncertainty data quality: assess the completeness of the attributes for each unit uncertainty. If an attribute is blank, what information can we infer from this? For instance if the migration is blank then probably the uncertainty is a high threat to the system?
 - b) While capturing uncertainties, it should be acknowledged if the uncertainty and its attributes are explicitly identified in the system or inferred.
5. Finally, assess whether the evaluation demonstrates that the uncertainty framework feasibly captures and represents uncertainty to generate and document architecture knowledge.

8.5.2 Case study 2: Uncertainty analysis of software architecture

Uncertainty analysis utilises the software architecture uncertainty data to calculate sensible software architecture uncertainty metrics. Uncertainty metrics are quantitative or qualitative indicators which can provide insight about the uncertainty in the software architecture description. They are based on the unit uncertainties and the various uncertainty attributes.

Below are the steps of the analysis which include pre-define metrics. In practice, other sensible metrics can be defined depending on the context:

1. First identify the system whose uncertainty is to be documented and document the uncertainties as discussed in Case study 1 - section 8.5.1.
2. After the uncertainties are captured and represented, we can define the metrics.
 - a) Calculate the uncertainty metrics using the uncertainty data. The metrics are defined such that they convey information about the uncertainty of the software architecture. For example, the number of uncertainties per architecture component of a particular view informs us about the numbers of known uncertainties of that architecture component depending on the particular viewpoint.

Additional metrics can be derived from existing metrics through for instance aggregation. For example, the total count of unit uncertainties of an architecture component from different views. Using such metrics, we can deduce information such as which component has the highest number of known uncertainties.

- b) Some software architecture uncertainty metrics include the following:
 - i. Count of uncertainties per component
 - ii. Count of types of uncertainties per component
 - iii. Among other possible metrics and aggregation of uncertainty metric calculations.
 - c) Analysis report - stakeholders can use the uncertainty information for various purposes. For instance, developers may use the uncertainty data as input to their risk consideration during development, operations team may consider operational risks and opportunities from the uncertainty. Similarly, other stakeholders can use the uncertainty data to explore opportunities, risks, mitigation and outcomes in addition to other sensible information.
 3. Finally, assess whether the evaluation demonstrates that the uncertainty framework feasibly captures and represents uncertainty to document architecture knowledge, and that the analysis generates useful insight. In this regards, we use an customised version of the ATAM to assess the validity, soundness and sanity of our approach.

8.5.3 Case study 3: Alternative candidate architecture uncertainty ranking and selection

One of the main challenges in architecture design is *how to compare and select a candidate architecture from comparable alternative architectures?* In the context of uncertainty in software architecture, we consider the use of the uncertainty details to address this challenge so that we select a candidate architecture that is likely to be less vulnerable to uncertainty risks.

Among the attributes of an uncertainty, we introduce an influence measure attribute that captures the potential influence of uncertainty. We can reason about uncertainty by aggregating uncertainties to compare or rank candidate architectures based on the mitigated or unmitigated uncertainties and their potential influence of the system. The architecture designer is meant to assign the value of the influence measure attribute.

The following steps outline the process of ranking and selecting a candidate architecture from comparable architectures:

1. Select a system whose candidate architecture are to be compared basing on uncertainty.
2. Define or identify, if they already exist, alternative candidate architectures to compare and rank.
3. Identify common uncertainties of the domain from sources such as system documentation and others. This will define a common basis for comparing the candidate architectures.
 - For each uncertainty, include an uncertainty influence attribute. This attribute captures the potential influence an uncertainty is expected to have on the system if its risk is realised.
 - The influence attribute is defined by the architecture designer and assigned values basing on a rationale.
4. For each candidate architecture assess how it handles the known uncertainties.
5. The criteria for ranking candidate architectures is based on their mitigations or handling of the common uncertainties. A candidate architecture which is less vulnerable to the uncertainties ranks top.
 - For each uncertainty conduct an architecture analysis
 - Rank the candidate architectures based on the uncertainty data and select the most suitable architecture

- The most suitable architecture is the candidate architecture most likely to be uncertainty robust depending on the uncertainty data.
6. To control and assess the validity of ranking approach, we propose three variations of the ranking algorithm, which we apply separately, and compare their results.
 7. This ranking only provides an uncertainty dimension to the candidate architecture selection problem. Other approaches can contribute information to making the overall final decision on the suitability of an implementation candidate architecture.

8.6 The ATAM overview and customisation

This section provides an overview of the ATAM and how it was customised for the purpose of this thesis. We change the ATAM to adapt it so that it is suitable for application by an individual, instead of a team of stakeholders.

8.6.1 ATAM Customisation

In this research we use a customisation of the ATAM omitting scenario generation activities, quality attributes identification and architecture definition. The case studies we analyse already have these details defined. The existing documentation of each case study provides such details. Indeed the ATAM authors made the following observation about its application in practice and customisation, while applying the ATAM to analyse the Battlefield Control System (BSC) [119]:

In describing the BSC ATAM, we will not describe every step of the method [the ATAM]. This is for two reasons. The first reason is that the steps involve a reasonable amount of repetition (of activities such as scenario elicitation); in describing the method we simply describe a single instance of each activity. The second reason is that no particular instance of the ATAM slavishly follows the steps as stated. Systems to be evaluated are in different stages of development, customers have different needs, and there will be different levels of architectural documentation in different development organisations.

Thus, our approach to applying the ATAM involves applying existing scenarios to the existing quality attribute and architecture styles to identify ATAM key results. As always, the final outcome or results of the ATAM are the architecture risks, sensitivity points and trade-off points with respect to quality attributes and architecture styles.

We use these ATAM analysis results to compare with the results from the uncertainty data from the framework to identify insight about the uncertainty status of the case studies. The key for such an evaluation is to qualitatively assess the usefulness of the uncertainty framework data and its potential application to uncertainty consideration in software architecture.

Again, the point of considering uncertainty in software architecture is not to identify all uncertainties and eliminate or mitigate them. Simply, architecture analysis cannot achieve such a goal. The point is to generate insight so that the architecture stakeholders are aware of the uncertainty within the system and can take other proactive approaches to manage it. Similarly, the point of the ATAM is not perfect architecture analysis, as discussed by the following quote from the authors [119]:

The point of this example [application of the ATAM on the BSC example] is not to show which alternative the contractor chose, for that is relatively unimportant and relied on their organizational and mission-specific constraints. The point here is to show that the process of performing an ATAM on the BSC raised the stakeholder's awareness of critical risk, sensitivity, and trade-off issues. This, in turn, focused design activity in the areas of highest risk and caused a major iteration within the spiral process of design and analysis.

Finally, it has to be realised that architecture analysis depends on the inputs of the process. The quality of the results and usefulness of the analysis is a direct output of the analysis function and its input- as noted by the following quote from the ATAM authors:

An architecture analysis method, any architecture analysis method, is a garbage-in-garbage-out process. The ATAM is no different.

Therefore, the uncertainty consideration framework is no different, too. Its results depend on the input. Of course, it might be difficult to generate all uncertainties during the initial architecture review. As such, continuous uncertainty identification and analysis might be an appropriate strategy in practice - for instance, by following the agile development methodology and update the uncertainties through the multiple iterations.

8.6.2 The uncertainty framework and the ATAM

In the ATAM, its inputs include the architecture description, quality attributes, architecture styles and scenarios to generate architecture risks, sensitivity points and trade-off points. In the

uncertainty consideration framework evaluation, the inputs are the architecture description and the output is the uncertainty data which can be analysed to generate the uncertainty insight about the software system using various metrics.

As previously stated, our interest in applying the framework is to characterise the uncertainty in the system so that stakeholders are aware of it for their uncertainty consideration. This is similar to the ATAM interest expressed in the following quote:

What we are interested in doing - in the spirit of a risk identification activity - is learning where an attribute of interest is affected by architectural design decisions, so that we can reason carefully about those decisions, model them more completely in subsequent analyses, and devote more of our design, analysis, and prototyping energies on such decisions.

Thus, what we aim to do in the ATAM, in addition to raising architectural awareness and improving the level of architectural documentation, is to record any risks, sensitivity points, and trade-off points that we find when analyzing the architecture.

The uncertainty framework characterises the uncertainty of a system using various attributes: uncertainty descriptive attributes, uncertainty source attributes, uncertainty time attributes, uncertainty mapping attributes. Characterising the uncertainty of an architecture and using sensible metrics can highlight significant uncertainty risk areas of the software architecture for further consideration.

The key outputs of the ATAM are its identification of the architecture risks, sensitivity points and trade-off points. In addition the ATAM assists with the architecture documentation. Similarly, the purpose of the uncertainty consideration framework is to characterise the known uncertainties of the software architecture with the goal of focusing attention on specific areas of an architecture based on the awareness of the architecture uncertainties. Besides, such characterisation assist with improving the documentation of the architecture uncertainties.

8.6.2.1 Summary of ATAM and the uncertainty framework

This section has discussed the customisation we made to the ATAM for its application in the evaluation. Our adaptation of the ATAM takes advantage of the available existing results. We use existing architecture descriptions, quality attributes, architecture styles and scenarios to apply the ATAM. The objective of applying both the ATAM and the uncertainty consideration framework is to compare the results of the two approaches.

8.6.3 Independent application of the uncertainty framework - MSc project vs AEROSTACK

Table 8.2: Comparison of the completed uncertainty attributes

	Attribute	MSc - ERP [17]	Coverage (%)	MSc -IoT [17]	Coverage (%)
	Count of uncertainties	33	100%	31	100%
1	Description	33	100%	31	100%
2	Nature	33	100%	31	100%
3	Bound	-		-	
4	Perspective	33	100%	31	100%
5	Awareness	33	100%	31	100%
6	Level	33	100%	31	100%
7	Source type	33	100%	31	100%
8	Cause	33	100%	31	100%
9	Viewpoints	33	100%	31	100%
10	Facets	33	100%	31	100%
11	Location	33	100%	31	100%
12	Manifestation	33	100%	31	100%
13	Measure	33	100%	31	100%
14	Monitor	33	100%	31	100%
15	Evidence	31	94%	31	100%
16	Relationship	-		-	
17	Emerging time	33	100%	31	100%
18	Lifetime	33	100%	31	100%
19	Change	33	100%	31	100%
20	Pattern	33	100%	31	100%
21	Dependencies	23	70%	22	70%
22	Risk or Opportunities	33	100%	31	100%
23	Mitigation or Exploitation	33	100%	31	100%
24	Outcome (with operators)	33	100%	31	100%

The MSc project titled *A Visual Representation of Uncertainty in Software Systems* used the framework as a foundation to generate and document uncertainty knowledge [17]. The uncertainty information was then used to develop a website which offers automated models and graphics for visualising the uncertainty data.

In the MSc project, [17], uncertainty data was generate for two systems: a cloud-based Enterprise Resource Planning (ERP) system and an IoT consisting of sensors, devices, connectivity, data processing and various interfaces. Thirty three (33) uncertainties were identified for the ERP system, while thirty one (31) uncertainties were generated for the IoT system.

As shown in Table 8.2, in terms of these two projects: 1) The MSc - ERP system has all attribute completed at 100% except the *Bound*, *Relationship* which are empty, and *Evidence* at 94% and *Dependencies* at 70% [17]; 2) The MSc - IoT equally, has all attributes covered at 100% except *Bound*, *Relationship* which are empty, and *Dependencies* at 70% [17].

As an empirical process, these incompleteness are expected, and the framework allows for customisation. The MSc examples demonstrate that the framework has been used independently by other to generate results and uncertainty knowledge. Table 8.2 shows an aggregate analysis of the values identified under each of the examples for each of the framework's attributes.

Together, the insight from both the case studies and the MSc projects [17] is the feasible application of the framework to generate uncertainty knowledge for further analysis.

8.7 Conclusion

The evaluation focuses on applying the uncertainty framework in software architecture analysis on three case studies: the AEROSTACK architecture, the IMPALA platform, and the IoV architecture. The AEROSTACK evaluation focuses on generating knowledge and documentation. The case study on the IMPALA platform focuses on uncertainty data analysis to gain insight into the software architecture. Finally, the IoV case study conducts an analysis on candidate architectures and uses the results to select a candidate architecture which is likely to be less vulnerable to uncertainty. Besides, in this chapter, we illustrated the feasibility of the framework by highlighting independent results of an MSc dissertation which is based on the application of the framework.

CASE STUDY 1: AEROSTACK - UNCERTAINTY KNOWLEDGE GENERATION AND DOCUMENTATION

In this evaluation, we evaluate the use of the uncertainty framework to explore uncertainty of AEROSTACK, which is documented in an open-source repository on GitHub¹ and published work[5]. This case study demonstrates a foundational use of the framework to generate uncertainty knowledge and documentation.

9.1 Overview

The AEROSTACK case study focuses on identifying uncertainties and capturing them using the uncertainty framework to generate uncertainty knowledge and documentation. The case study evaluation follows the strategy described in Section 8.5.1. The objective of the case study is to assess the feasibility of using the uncertainty framework to generate architecture uncertainty knowledge and documentation. The data are captured and presented in a table.

¹<https://bit.ly/3MGgJV7>

AEROSTACK is designed to enhance the independent operation of UAS. Fully autonomous operation is needed to simplify the usage of UAS and to flexibly extend UAS adoption to a range of applications. Thus, both autonomy and versatility are among the main quality attributes of the AEROSTACK architecture. In this evaluation, we identify and characterise the uncertainties of AEROSTACK in the context of it being a multi-purpose software framework for autonomous multi-UAS operation.

9.2 AEROSTACK Architecture Description

We identify the uncertainties in the documentation of AEROSTACK in its logical and development view of the architecture. The logical view presents a layered architecture of the main components of the AEROSTACK. The development view presents the modules, libraries and application components of the AEROSTACK.

The AEROSTACK architecture considers uncertainty in its design as part of the various architecture design decisions. One of the key non-functional requirements of the AEROSTACK architecture is that it should be hardware independent. This ensures that the architecture can operate with UAS from various manufacturers and with a range of sensor hardware technology, thus handling uncertainty from possible UAS model variations and supporting versatility.

The AEROSTACK architecture is designed to run on the operative level of the UAS software, instead of the firmware and middle-ware which are hardware dependent. A UAS has three software levels - firmware, middle-ware and operative system.

The firmware runs on the UAS onboard computer, and it directly depends on the hardware and its time-critical. The middle-ware is also time-critical to the control of the system, but it depends on the firmware. In contrast, the operative system is computationally intensive but non-time-critical and is used for UAS system management. AEROSTACK architecture runs at the operative level and is hardware independent to cope best with UAS hardware variations.

Such AEROSTACK architecture design decisions as these may include both uncertainty consideration and its mitigation. In the just-presented architecture decisions, the potential uncertainty is about the UAS hardware variation, and its mitigation includes hardware independent design, with the goal or outcome of supporting versatility or flexibility.

The case study identifies such uncertainties and characterises them using the uncertainty framework. In Section 9.3 we identify uncertainties in the logical view of AEROSTACK. Similarly, Section 9.4 identifies and documents uncertainties in the development view of the AEROSTACK.

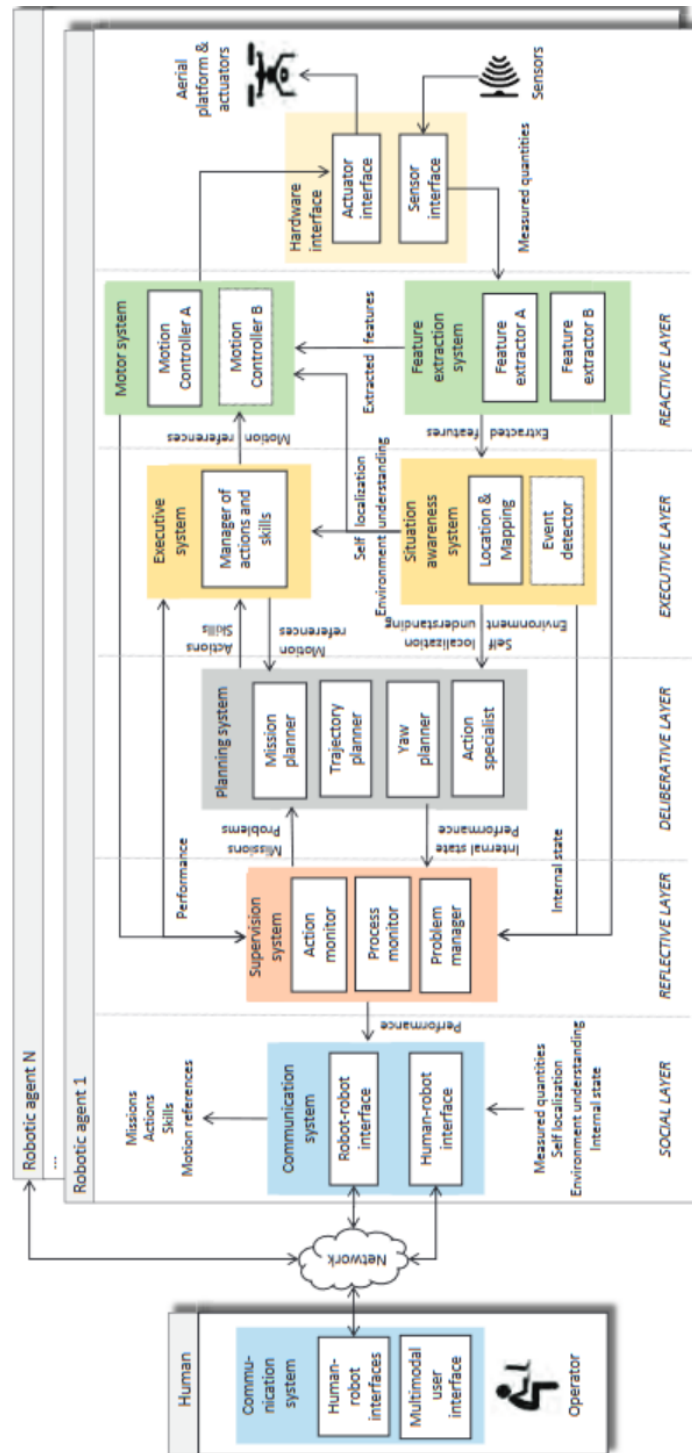


Figure 9.1: Logical view of AEROSTACK from the original image in [5]

9.3 Logical view

The logical view of AEROSTACK is organised into layers as shown in Figure 9.1. The main layers of the logical view of the AEROSTACK architecture are *Reactive*, *Executive*, *Deliberative*, *Reflective* and *Social* layers.

The AEROSTACK architecture is designed to be ready for use or adoption without the need for specific customisations. It includes the main components within each layer to execute a fully autonomous mission of a UAS or swarms of UAS . Furthermore, it has a collection of ready-to-use and flight-tested modular components that can be reused by users and developers. AEROSTACK supports compatibility with other aerial platforms and a large number of sensors.

In terms of layers, *Reactive layer* functions in the present - reacting to sensor input in a sensor-action loop, while *Deliberative layer* uses information from the past and projection to the future to generate complex solutions such as trajectory planning and other planning tasks for UAS .

To increase the degree of autonomy of UAS , AEROSTACK includes a *Reflective layer* based on cognitive architectures to simulate self-awareness to supervise the other layers: check progress and react to problems - unexpected obstacles, faults, etc. - with recovery actions. The *Social layer* is to support multi-agent systems communication with human operators and other UAS . *Executive layer* is responsible for navigation systems, including situation awareness, location and mapping, and event detection.

The logical view of the AEROSTACK architecture shows a collection of highly interrelated and specialised components organised to achieve a full autonomous UAS architecture. Such an organisation of a solution creates challenges such as adaptability to different problems, performance of the various components, efficiency, scalability, and others. The AEROSTACK architecture addresses these challenges.

We identify uncertainties in the context of such challenges. Tables 9.1 and 9.2 show uncertainties that we have identified related to the logical view of AEROSTACK , that is, known uncertainties.

Table 9.1: Uncertainties of the Logical view

Attribute	Uncertainties					
Uncertainty ID	1	2	3	4	5	6
Description	Network connection loss	New UAS joins swarm	Obstruction appearance	Localisation precision	Mapping environments	Trajectory and mission planning
Nature	Aleatory	Epistemic	Aleatory	Aleatory	Epistemic	Aleatory or epistemic
Bound						
Perspective	Objective	Objective	Object	Subjective	Objective	Objective
Awareness	Known unknown	Known unknown	Known unknown	Known unknown	Known unknown	Known unknown
Level	Low	Medium	Medium	Medium	High	low
Source type	Exogenous or Endogenous	Exogenous	Exogenous	Endogenous	Exogenous	Endogenous
Cause	Hardware or software failure & UAS out of range	Data growth from new and existing sources	Unexpected object on UAS path	Human error or mistake	an unstructured and changing environments	unanticipated changes and poor data
Viewpoints	Logical	Logical	Logical	Logical	Logical	Logical
Facets	Architecture	Architecture	Architecture	Architecture	Architecture	Architecture
Location	Reactive and Social layers	UAV route	UAV route	Executive layer	Executive and Reactive layer	Deliberative and Reflective layer
Manifestation	No network connection	UAS in or out of swarm	Collusion	Inaccuracy	Lack of awareness	Wrong trajectory and failures
Measure	Probability	Probability	Probability	Error margin	Mapping quality	Errors rates
Monitor	Network check - ping command		Collision rates			Trajectory and planning outcomes
Evidence		UAS Swarm numbers		Localisation data	Mapping data	trajectory and planning data
Relationship			Object detection sensor feedback			
Emerging time	Run-time	Run-time	Run-time	Run-time	Run-time	Run-time
Lifetime	Perpetuity	Perpetuity	Perpetuity	Limited	Perpetuity	Perpetuity
Change	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic
Pattern	Aperiodic-sporadic	Aperiodic-sporadic	Systematic	Aperiodic- Sporadic	Aperiodic- Sporadic or systematic	Aperiodic- Sporadic or systematic
Dependencies	Social, Executive, and other UAS uncertainties	Navigation and social uncertainties	Vision system and radar system uncertainties	Localisation services uncertainties	Environmental awareness services uncertainties	Navigation awareness
Risk or Opportunities	Risk: Communication loss, mission failure, and accident	Risk: UAS collisions & mission failure	Risk: Crash accidents	Risk: Errors in trajectory	Risk: misleading mapping, lack of awareness, data loss	Risk: Errors in trajectory mission failure
Mitigation or Exploitation	Mitigation: Abnormal situations problem manager	Mitigation: Fully autonomous operation in UAS	Mitigation: Precise control of the aircraft	Mitigation: High maneuverability capabilities	Mitigation: High definition and accuracy sensors	Mitigation: Situation awareness system
Outcome (with operators)	MINIMISE mission dependency on external communication	Support MULTI-robot swarming POSSIBILITIES	MINIMISE obstruction collision chances	MAXIMISE localisation precision and accuracy	MAXIMISE and enhance environmental mapping	MINIMISE trajectory errors and MAXIMISE accuracy

Table 9.2: Uncertainties of the Logical view

Attribute	Uncertainties					
Uncertainty ID	7	8	9	10	11	12
Description	Multi-tasking executions	New functionality to UAS	Hardware and sensor compatibility	Firmware change	Sensor failure	Environmental instability
Nature	Epistemic	Epistemic	Epistemic	Epistemic	Aleatory	Aleatory
Bound						
Perspective	Objective	Objective	Objective	Objective	Objective	Subjective
Awareness	Known unknown	Known unknown	Known unknown	Known unknown	Known unknown	Known unknown
Level	High	Medium	Medium	Medium	High	low
Source type	Exogenous	Exogenous	Exogenous	Endogenous	Exogenous	Endogenous
Cause	Parallel execution requirements	UAS updates or enhancements	Hardware and sensors variety or diversity	Updates or hardware changes	Hardware or software issues	For example, weather
Viewpoints	Logical	Logical	Logical	Logical	Logical	Logical
Facets	Architecture	Architecture	Architecture	Architecture	Architecture	Architecture
Location	Robotic agent	AEROSTACK layers	Social and Reactive layers - interfaces	Reactive layer	Reactive layer	Executive layer
Manifestation	Execution conflicts	Incompatibility	Incompatibility	Firmware changes causing issues	Robotic agent issues	Hazardous environmental conditions
Measure	Probability	Updates rate			Failure rate	Weather conditions
Monitor	Multi-tasking demands	Updates requirements	Compatibility issues		Sensor data input issues	Hazardous environmental warnings
Evidence	Multi-tasking data	New functionality requests data	Compatibility and incompatibility issues	Firmware updates data	Sensor data	Environmental data
Relationship						
Emerging time	Run-time	Run-time	Run-time	Data usage	Run-time	Run-time
Lifetime	Perpetuity	Limited	Perpetuity	Limited	Perpetuity	Perpetuity
Change	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic
Pattern	Aperiodic-sporadic	Systematic	Aperiodic- sporadic	Systematic	Aperiodic- Sporadic	Aperiodic- Sporadic
Dependencies	All layers uncertainties	All layers uncertainties	Reactive and Social layers uncertainties	Hardware and software interfaces uncertainties	Reactive layer uncertainties	Executive, Deliberative and Reactive layers uncertainties
Risk or Opportunities	Risk: System low efficiency	Risk: New functionality failure	Risk: Incompatibility with associated results	Risk: Firmware change failure	Risk: data and signal corruption or loss	Risk: mapping and localisation failure and accidents
Mitigation or Exploitation	Mitigation: Distributed processing, separate common and optional processes	Mitigation: Components arranged by functionality and dependency level	Mitigation: Standardisation & extensible design	Mitigation: Independence to hardware and firmware	Mitigation: Reflective layer - failure detection, notification, and recovery	Mitigation: Fully autonomous operation of UAS
Outcome (with operators)	Scalability: UAS needs to be AS efficient and effective AS POSSIBLE in multi-tasking	Versatility: introduce AS MANY new functionalities AS POSSIBLE	AEROSTACK needs to be compatible to AS MANY hardware and sensor variety AS POSSIBLE	AEROSTACK needs to be AS independent from hardware AS POSSIBLE	MINIMISE risks of UAS from sensor failure	MINIMISE risks of UAS failure from instability

9.4 Development view

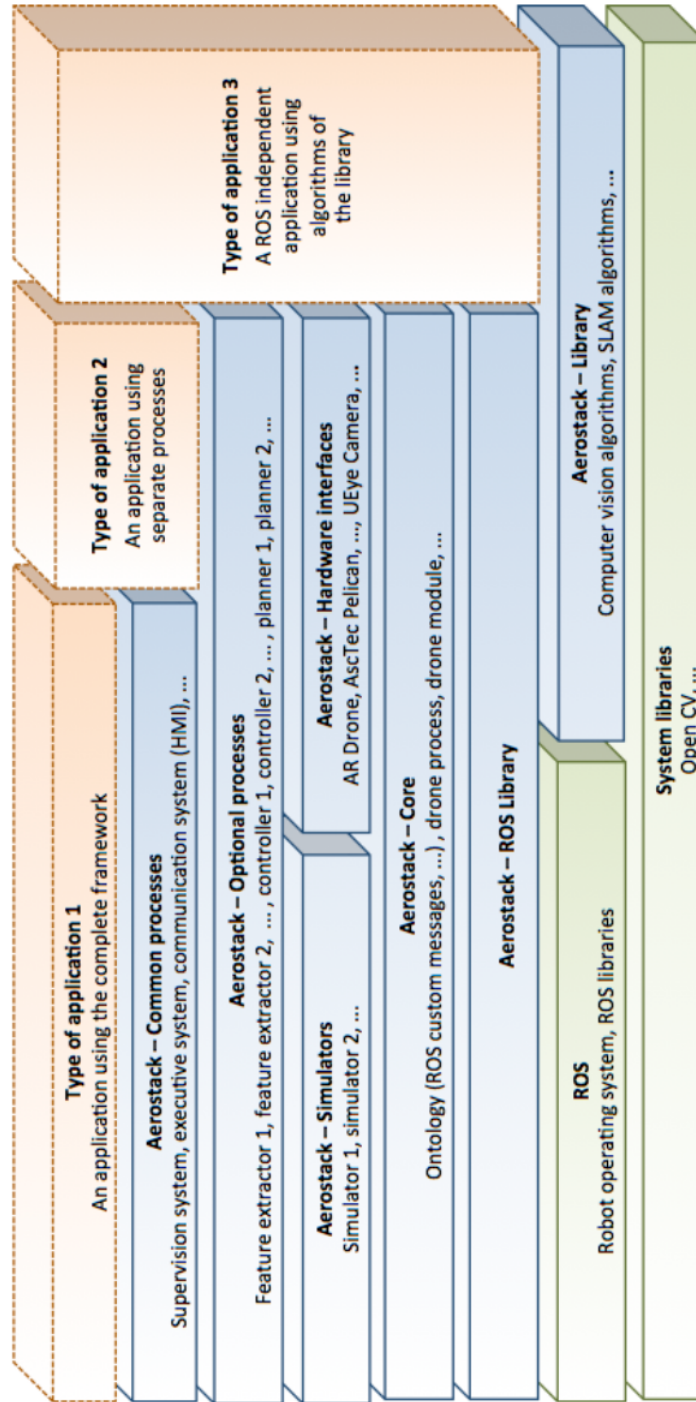


Figure 9.2: Development view of AEROSTACK Architecture from the original image in [5]

Figure 9.2 shows the development view of the AEROSTACK architecture. All AEROSTACK components run above the system libraries, thus achieving hardware independence. The development view includes three categories of components.

First are hardware dependent and operating system components, the System libraries, and the Robot operating system (ROS). Next, we have the AEROSTACK architecture components. There are seven components: from the AEROSTACK library to the AEROSTACK common processes, as shown in Figure 9.2. Finally, there are the application components where software developers can develop applications of UAS using the AEROSTACK architecture.

The main non-functional requirements of the architecture is to ensure that AEROSTACK is hardware independent and that the components in the architecture can be easily modified. Table 9.3 present uncertainties in the context of the development view.

9.5 Discussion

In the previous two sections, Logical view Section 9.3 and Development 9.4, we identified and encoded uncertainty data using the uncertainty framework as uncertainty knowledge and documentation. In this section, we illustrate the interpretation of uncertainty in Section 9.5.1 and then discuss the uncertainty documentation in 9.5.2. The other two case studies, in Chapters 10 and 11, build on the foundation in this case study to facilitate architecture analysis and evaluation.

9.5.1 Encoding and interpreting uncertainty knowledge and documentation

Sections 9.2 and 9.3, together, they include sixteen (16) uncertainties, with twelve (12) identified in the logical view and the rest identified in the development view.

Individually, each recorded uncertainties captures and documents specific uncertainty knowledge or data. Collectively, the uncertainties provide comprehensive uncertainty documentation of the AEROSTACK systems from the two perspectives - logical and development views.

Let us walk through the following uncertainties from each of the two views and analyse their knowledge and data captured for illustration proposes:

First, we begin with the *Uncertainty ID 1* from the Logical view, Table 9.1. The overall uncertainty is about *Network connection loss* which is captured by the *Description* attribute. Since network failure may occur from a variety of issues, such as various hardware failures, we assign its *Nature* as *Aleatory*. On the contrary, *Epistemic* uncertainty is due to lack of knowledge, where the uncertainty can be reduced by additional knowledge.

Proceeding with the uncertainty example, for this uncertainty, the *Bound* attribute is blank because there are no specific boundaries to the uncertainty. When a network connection is lost,

Table 9.3: Uncertainties of the development view

Attribute	Uncertainties			
Uncertainty ID	13	14	15	16
Description	Update ROS	System libraries change	Flight security attacks	AEROSTACK library changes
Nature	Epistemic	Epistemic	Epistemic	Epistemic
Bound				
Perspective	Subjective	Subjective	Objective	Subjective
Awareness	Known unknown	Known unknown	Known unknown	Known unknown
Level	Medium	Medium	High	Medium
Source type	Endogenous or Exogenous	Endogenous or Exogenous	Endogenous or Exogenous	Endogenous or Exogenous
Cause	Modifiability issues and human error	Need for modifications or new change requests	Code vulnerabilities which can be exploited	Need for modifications or new change requests
Viewpoints	Development	Development	Development	Development
Facets	Architecture	Architecture	Architecture	Architecture
Location	ROS	System libraries	Flight control and applications	AEROSTACK library
Manifestation	Update failure	Change causing bugs	Detected exploitations	Modification failure
Measure	Updates failure rates			Modification failure rate
Monitor	Modification or update issues	Modification or update issues	Count of detected exploitations	Modification or update issues
Evidence	Update data	Change data	Threats data	Modification or change requests data and plans
Relationship				
Emerging time	Development	Development	Development & Run-time	Run & development times
Lifetime	Perpetuity	Perpetuity	Perpetuity	Perpetuity
Change	Dynamic	Dynamic	Dynamic	Dynamic
Pattern	Aperiodic- sporadic	Aperiodic- sporadic	Aperiodic- sporadic	Aperiodic- sporadic
Dependencies	Uncertainties above ROS layer	Uncertainties above system libraries	Applications and ROS uncertainties etc.	UAS application uncertainties
Risk or Opportunities	Risk: ROS update failure	Risk: Update costs and failure	Risk: Successful software security attacks	Risk: incompatibility with various UAS platforms
Mitigation or Exploitation	Mitigation: layered, modular, extensible design	Mitigation: layered, modular, extensible design	Mitigation: Fully autonomous flights to avoid attacks	Mitigation: modular, extensible design & hardware independence
Outcome (with operators)	Update ROS AS seamlessly AS POSSIBLE	MINIMISE effort to change library source code	MAXIMISE AEROSTACK flight security from attacks	MAXIMISE AEROSTACK modifiability

it can be specifically identified; therefore, *Perspective* of this uncertainty is *Objective* as its identification does not depend on the knowledge of an individual stakeholder or agent.

In terms of *Awareness*, this is a *Known unknown* uncertainty. The *Level* of the uncertainty is *low* and its *Source* can be internal or external or *Endogenous* or *Exogenous*, respectively, to AEROSTACK. The *Causes* of the uncertainty includes *Hardware or software failure* and *UAS being out of range*.

The *Viewpoint* where we explicitly identified the uncertainty is in the *Logical* view; however, this uncertainty can also impact views such as the *Deployment*, *Network* and *Physical* views. All these views are of the *Architecture Facet* where we consider this uncertainty. Specifically, *Location* of the uncertainty is the *Reactive and social* layers through which the network interfaces with the UAS, and its *Manifestation* is through *No network connection*.

The *Monitor* of the uncertainty can be done through tools such as the *Network check - ping command*. The evidence we can use to reduce the uncertainty is *blank*. The *Measure* of this uncertainty is *Probability* and does not have any specific *Relationship* attribute value.

The uncertainty *Emerging time* is during the UAS *Run-time* and the uncertainty will exist or its *Lifetime* is as long as the system is in operation, so it is in *Perpetuity*.

The uncertainty might *Change* over time; therefore, it is *Dynamic*. Its *Pattern* is irregular since network connections can be lost abruptly and without particular regular intervals, so its *Aperiodic-sporadic*.

In terms of influence, the *Network connection loss* uncertainty *Dependencies* includes *Social*, *Executive*, and other UAS uncertainties of the architecture.

For *Risks or Opportunities*, the uncertainty causes *Risk: Communication loss, mission failure, and accident*. The AEROSTACK architecture includes a specific component to handle risks: *Mitigation: Abnormal situations problem manager* with the goal to *MINIMISE mission dependency on external communication* as the *Outcome (with operators)*.

Thus, the *Network connection loss* uncertainty is captured with the framework in this way. Similarly, the other uncertainties can be interpreted in the same way from the encoding of the uncertainty framework in Table 9.1.

Next, we consider one of the uncertainties from the development view - the last one. Specifically, we look at *Uncertainty ID 16* in Table 9.3.

The overall uncertainty is about *AEROSTACK library changes* which is captured by the *Description* attribute. Since some changes can be planned, while others can be abrupt, therefore,

we assign its *Nature* as *Epistemic* - the more changes are planned and schedule, the less the uncertainty. The *Bound* attribute is blank because there are no specific boundaries to the uncertainty.

Different stakeholders might arrange or plan for different changes, depending on the priority; therefore, *Perspective* of this uncertainty is *Subjective* as its identification depends on the knowledge of individual stakeholders.

In terms of *Awareness*, this is a *Known unknown* uncertainty. The *Level* of the uncertainty is *Medium* and its *Source* can be internal or external or *Endogenous* or *Exogenous*, respectively, to AEROSTACK . The *Causes* of the uncertainty includes *Need for modifications or new change requests*.

The *Viewpoint* where we explicitly identified the uncertainty is in the *Development* view, however, this uncertainty can also impact such views as *Deployment*, and *process* views. All these views are of the *Architecture facet* where we consider this uncertainty. Specifically, *Location* of the uncertainty is the *AEROSTACK library* layer, and its *Manifestation* is through *Modification failure or change issues*.

The *Monitor* of the uncertainty can be expressed by *modification issues*. The evidence we can use to reduce the uncertainty is *modifications or change requests data and plans*. The *Measure* of this uncertainty is *Modification failure rates* and does not have any specific *Relationship* attribute value.

The uncertainty *Emerging time* is during the *UAS Run and development times* and the uncertainty will exist or its *Lifetime* is as long as the system is in operation; thus, it is in *Perpetuity*.

The uncertainty might *Change* over time; therefore, it is *Dynamic*. Its *Pattern* is irregular since some change will occur abruptly, on demand, and without particular regular intervals; therefore, its *Aperiodic- sporadic*.

In terms of influence, the *AEROSTACK library changes* uncertainty *Dependencies* includes *UAS applications uncertainties* which are built on top of the libraries.

For *Risks or Opportunities*, the uncertainty causes *Risk: incompatibility issues with various UAS platforms*. The AEROSTACK architecture includes the following approaches to handle the risk: *Mitigation: modular, extensible design & hardware independence* with the goal to *MAXIMIZE AEROSTACK modifiability* as the *Outcome (with operators)*.

Thus, the *AEROSTACK library changes* uncertainty is captured with the framework. In this way, as illustrated through these uncertainties: *Uncertainty ID 1* from the Logical view, Table 9.1 and

Uncertainty ID 16 in Table 9.3 of the Development view, uncertainties can be captured.

9.5.2 Uncertainty documentation and knowledge

The uncertainty framework provides a basis for a range of fundamental uncertainty data that can be used for various analyses to gain insight into the uncertainty of a software system. The opportunity to use the framework includes both qualitative and quantitative analyses. Additionally, the process of capturing and completing the uncertainty attributes provides an opportunity to think about or consider uncertainty in the context of the system.

The framework data defines the ground-data for further analysis. For example, considering the current attributes of the framework, each of these contains information about the individual uncertainty. Collectively, the group of attributes contains adequate information to characterise an individual uncertainty, from its description, to risks or opportunities, mitigation or exploitation, and outcomes.

As an aggregate, the uncertainty data can provide a basis which can be used to characterise the uncertainty of the the architecture. For example, currently, for the AEROSTACK architecture, we identified the sixteen uncertainties (16), all with different characteristics. This provides an overview of the uncertainty with respect to the individual view points. We can analyse the data on individual attributes to assess which location, for instance, in the architecture has the most uncertainties, thus might likely be vulnerable to uncertainty risk.

More qualitative and quantitative analyses could be conducted to explore the uncertainty in the architecture. The remaining two case studies conduct such detailed analyses. While applying the uncertainty framework, we also observed some of its challenges and limitations in applying it. For instance, the framework might require the use of structured texts for easier automated analysis.

However, currently, this is not a major limitation as the framework has been designed to be generic without specific restrictions on its application. But individual users are free to introduce the necessary customisation for better usability in specific contexts, as illustrated by the framework's application in the MSc dissertation [17]. The framework does not prescribe specific steps to follow when applying it. However, in Chapter 6, we specify a suggested approach to applying the it.

9.6 Conclusions

The aim of this case study was to demonstrate the use of the framework to document and generate uncertainty knowledge in software architecture. In the context of the case study, we captured uncertainties of the AEROSTACK architecture from its logical and development views. Additionally, we highlight some of the uncertainties to demonstrate the encoding and interpretation of the uncertainty data. In addition, we discussed the potential uses of the data that are developed in the other two case studies.

CASE STUDY 2: IMPALA - UNCERTAINTY ARCHITECTURE ANALYSIS

The IMPALA case study focuses on analysis. The evaluation follows the strategy described in Section 8.5.2. The objective of the case study is to evaluate the feasibility of using the uncertainty framework to generate architectural uncertainty knowledge and documentation, and then to apply it for architecture analysis.

10.1 Overview

In this evaluation, we use the uncertainty framework to explore the uncertainty of the architecture of the IMPALA platform, which is documented in the SDD¹. The SDD provides information on the IMPALA architecture design from a number of architectural viewpoints. The IMPALA platform is developed from a consolidation of COTS components. Its architecture is built from the composition and integration of various COTS components.

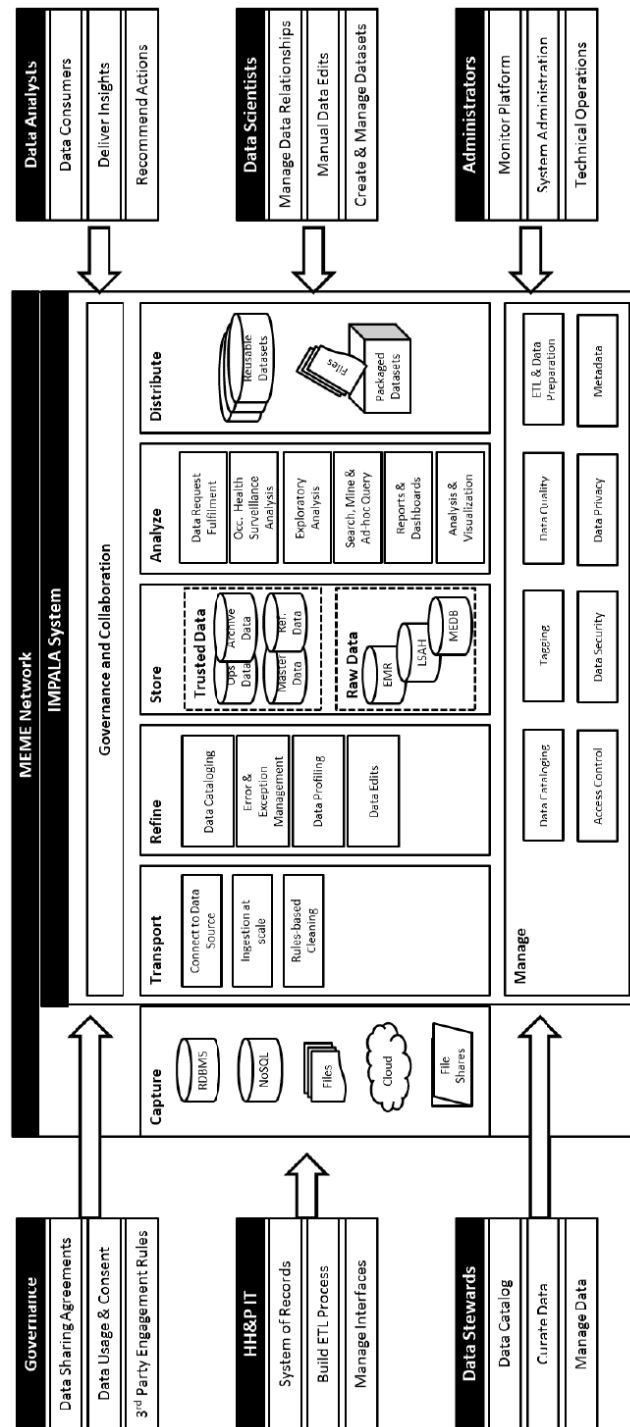
10.2 IMPALA platform architecture description

The IMPALA platform architecture description has five views - Logical, Functional, Infrastructure, Network and Security. Each view is described with a diagram showing the components and connectors that capture the architecture from the perspective of a viewpoint.

¹<https://ntrs.nasa.gov/citations/20160011412>

In the following sections, we review each architecture view and identify uncertainties within it. We capture and represent uncertainties using the uncertainty framework.

Each view is presented from its original diagram in the SDD. Its uncertainty data are captured in the corresponding table. The components in the architecture description are associated with the uncertainties according to the uncertainty framework attributes, as discussed in the uncertainty framework design Chapter 6.

Figure 10.1: Logical view of IMPALA platform from the original in SDD¹

10.3 Logical view

Figure 10.1 shows the logical architecture view of the IMPALA platform. It has seven main components: *Capture*, *Transport*, *Refine*, *Store*, *Analyze*, *Distribute* and *Manage*. The IMPALA platform operates within the Mission Extended Medical Enterprise (MEME) network, that is why all the components are enclosed within its block. The six smaller boxes in Figure 10.1 highlight the stakeholders; each of these has a pointer to a logical level at which they operate.

Each of the seven components have specific purposes. All components are internal to the IMPALA platform except the *Capture* component, which is partially external to the IMPALA platform because it includes external data sources for the IMPALA environment.

- *Capture* component interfaces with the current and future data sources, whose data is imported into the IMPALA platform. Examples include RDBMS, NoSQL, files, cloud sources, and others.
- The *Transport* component is responsible for the transmission (importing) of the captured data through the *Refine* component to the *Store* component.
- The *Refine* component is responsible for various data cleaning tasks before the data is sent to the *Store* component.
- The *Store* component is responsible for storage and includes data backup, redundancy, and other data storage functionalities.
- The *Analyze* component is responsible for data use activities, including creating datasets and their management, performing statistical and machine learning analyses, etc.
- The *Distribute* component is for data sharing to interested stakeholders, which may include through other systems and user agents.
- And finally, the *Manage* component supports the internal components of the IMPALA platform with various management tasks, such as monitoring the system and data health.

Each of these components can have associated uncertainties. In the following sections, we identify uncertainties under each of the main components of the logical view. The uncertainties are in the context of the description of the components highlighted in the SDD.

The identified uncertainties are not exhaustive by definition because there may be unknown unknown uncertainties. In addition, some uncertainties might not be identified since this is an empirical process. The uncertainties are characterised using the uncertainty framework attributes.

Tables 10.1 and 10.2 show uncertainties we have identified for the components in the logical view and express them using the framework attributes. The name of the relevant component(s) is in the *Location* attribute of each uncertainty - a unit uncertainty is within a column.

10.3.1 Uncertainty of the *Capture* component

The *Capture* component relates to data sources; therefore, some of its uncertainties relate to future data sources that must be integrated into the IMPALA platform. The SDD states that data will initially be captured from existing data sources within the NASA environment, including the EMR (Electronic Medical Record), the LSAH (Medical Evaluation Document, Part B (MEDB) Lifetime Surveillance of Astronaut Health) and MEDB Sharepoint.

In addition to these sources, the IMPALA platform is extensible and capable of capturing data from other types of data sources. The platform supports relational data sources such as Microsoft SQL Server, and Microsoft Access, as well as data located in file shares, cloud-based data stores, and local files of varying formats (XML, CSV, JSON, PDF, and more).

10.3.2 Uncertainty of the *Transport* component

Since uncertainty about future data sources in the *Capture* component could influence the *Transport* component, therefore, future data sources uncertainty also belongs to the *Transport* component. According to Table 10.1, future data sources uncertainty is linked to both the *Transport* and *Refine* components because its dependency attribute includes both of these components.

Uncertainties about the *Transport* component also relate to its capacity to handle traffic and the range of data source connections. Capacity is related to the current maximum throughput of the *Transport* component and the possibility that data input increases to the extent that the *Transport* component becomes a bottleneck of the IMPALA platform.

10.3.3 Uncertainty of the *Refine* component

The *Refine* component is a transient phase in which the data is processed to be used within the IMPALA platform for analysis. After refining, the data are sent to storage, the *Store* component. Refinement can be manual or automated with activities such as data review, cleaning, transformation and labelling (for use in machine learning).

Data processing in the *Refine* component is performed on a copy of the original data from *Transport* to mitigate against uncertainties from refinement errors. Users such as Developers,

Data Owners, Data Stewards, Data Scientists and Data Analysts collaborate to refine and catalogue the data.

10.3.4 Uncertainty of the *Store* component

The *Store* component is for persistent data storage after the data move from the *Capture*, *Transport* and *Refine* components. The key principles guiding this component are scalability, redundancy and performance of the data and storage infrastructure.

Scalability is handled through Apache Hadoop². In addition, redundancy against data loss is achieved through Apache Hadoop replication, across clusters of servers.

Uncertainty in performance is handled with the distributed processing capability of Apache Hadoop. The IMPALA platform leverages the parallel processing feature of Apache Hadoop, allowing the search, enrichment, cataloguing, and transformation of data by nodes in parallel.

10.3.5 Uncertainty of the *Analyze* component

The *Analyze* component is about the usage of the data from the *Store* component. Data usage includes various tasks such as ad hoc queries, data mining, machine learning, searching, and other data functions.

Taking into account the interactive nature of the analysis, a key aspect of the *Analyze* component is the user interface. At the architecture level, architectural performance is critical to the effectiveness of system interactions and feedback since these depend on the processing speed of handling analysis operations such as queries.

Furthermore, considering that security is a concern at the user interface level, the architecture needs to be secure against exploitation. Thus, uncertainty risks within the *Analyze* component relate to the performance and security of the IMPALA platform, among other concerns.

10.3.6 Uncertainty of the *Distribute* component

The *Distribute* component is responsible for sharing or publishing datasets. Users can work with the data in the *Distribute* component to gain insight, for example, into functionalities such as visualisation. The uncertainty in the *Distribute* component relates to the capability of sharing data and interoperability with other external systems.

²https://en.wikipedia.org/wiki/Apache_Hadoop

10.3.7 Uncertainty of the *Manage* component

Manage components serve the *Capture*, *Transport*, *Refine*, *Store*, *Analyze* and *Distribute* components, as shown in Figure 10.1. IMPALA platform system administrations, Data Stewards and Developers use scripts to execute functionalities of the management component.

Such functionalities include data loading and transformation, data cataloguing, and domain-knowledge-based tagging. The scripts are tested and deployed in their respective relevant components for execution, either through a web interface or command line interfaces. Therefore, uncertainty in management is related to the successful execution of management scripts.

In addition, system administrators use the management components for security in terms of access and privacy control. Therefore, it is necessary to ensure the security of the data. As such, data privacy and security are another source of uncertainty in the *Manage* component.

Table 10.1: Uncertainties of the Logical view

Attribute	Uncertainties					
Uncertainty ID	1	2	3	4	5	6
Description	Future data sources	Data import jobs capacity to handle data growth	Connections to future data sources	Refinement mistakes & errors	Data rapid growth	Data storage failure
Nature	Epistemic	Epistemic	Epistemic	Aleatory	Epistemic	Aleatory
Bound		Up to 99% of capacity			98% above capacity	66% or 2 to 3 storage failure
Perspective	Objective	Objective	Subjective	Subjective	Objective	Objective
Awareness	Known unknown	Known unknown	Known unknown	Known unknown	Known unknown	Known unknown
Level	High	Medium	Medium	Medium	High	low
Source type	Exogenous	Exogenous	Endogenous	Endogenous	Exogenous	Endogenous
Cause	Identification of new data source	Data growth from new and existing sources	Non-standard data sources	Human error or mistake	New data sources or increase in data collection	Store hardware or software failure
Viewpoints	Logical	Logical	Logical	Logical	Logical	Logical
Facets	Architecture	Architecture	Architecture	Architecture	Architecture	Architecture
Location	Capture component	Transport component	Transport component	Refine component	Store component	Store component
Manifestation	New data component	Data import delays	IMPALA failure to connect with new data sources	Demand supply	Storage capacity full	Data availability
Measure	Probability	Possibility		Refinement correctness	Data growth rate	Storage failure rate
Monitor	Governance - data integration requests	Demand for data import	Integration failures reports	Request for correction	Data growth & against storage	Redundant clusters failure rate
Evidence	New data source addition to Capture component	Data import job history/trend	Pending data sources due to connection failure		Route preview	Price difference
Relationship		Import congestion	Backlog of out-standing connections issues			
Emerging time	Run-time	Run-time	Development, Deployment & Run-time	Data usage	Run-time	Run-time
Lifetime	Perpetuity	Perpetuity	Perpetuity	Limited	Perpetuity	Perpetuity
Change	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic
Pattern	Aperiodic-sporadic	Aperiodic-sporadic	Systematic	Aperiodic- Sporadic	Aperiodic- Sporadic	Aperiodic- Sporadic
Dependencies	Transport & Refine components	Refine components	Refine components	Store & Analysis components	Distribute, & Analyse	Distribute, & Analyses
Risk or Opportunities	Risk: Data Integration failure or unknown data formats	Risk: Data loss & delay import	Risk: Data import delay and failure	Risk: data corruption	Risk: data loss due to lack of storage	Risk: Data loss due to storage failure
Mitigation or Exploitation	Mitigation: Extensible and able to capture data from other types of sources	Mitigation: Extensible design	Mitigation: Standardisation & extensible design	Mitigation: Use a copy of original data from transport component	Mitigation: Apache Hadoop cluster to support scaling	Mitigation: Apache Hadoop replication
Outcome (with operators)	Integration with AS MANY future data sources AS POSSIBLE	Import AS MUCH data as POSSIBLE from AS MANY sources AS AVAILABLE	Connect with AS MANY new data sources AS POSSIBLE or Require MINIMISE work to connect to new data sources	MINIMISE risk of loosing data from corruption due to refinement errors & mistakes	Scalability: Handle ALL future data storage demands	MINIMISE risks of data loss through redundancy

Table 10.2: Uncertainties of the Logical view

Attribute	Uncertainties				
Uncertainty ID	7	8	9	10	11
Description	System performance may degenerate	IMPALA's capacity to handle analysis demands and security	Data distribution interoperability	Management scripts execution success	Data access and privacy assurance
Nature	Aleatory	Epistemic	Epistemic	Aleatory	Epistemic
Bound		Up to 99% of capacity			
Perspective	Objective	Objective	Subjective	Objective	Subjective
Awareness	Known unknown	Known unknown	Known unknown	Known unknown	Known unknown
Level	Medium	Medium	Medium	Medium	High
Source type	Exogenous (New data source & usage demand)	Exogenous (Analysis demand growth)	Exogenous (external systems)	Endogenous (execution environment)	Endogenous (Humans in the loop)
Cause	System computation demands	Analysis operations on data	Non-standard data sharing request	Management script execution failure	Data privacy and security violations
Viewpoints	Logical	Logical	Logical	Logical	Logical
Facets	Architecture	Architecture	Architecture	Architecture	Architecture
Location	Store component	Analysis component	Distribute component	Manage component	Manage component
Manifestation	System slowing-down	Analysis performance and security	IMPALA failure to share data with external systems	Management functionality failure	Data security breach
Measure	Performance fluctuations	Performance fluctuations		Execution statistics	Data breach detection rate
Monitor	Performance data graph		Integration failures reports	Management functionality results	User access logs
Evidence	Performance trends	Performance trends	Pending data sharing due to interoperability issues	Management results	
Relationship			Backlog of out-standing interoperability issues		
Emerging time	Run-time	Run-time	Deployment & Run-time	Run-time	Run-time
Lifetime	Perpetuity	Perpetuity	Perpetuity	Perpetuity	Perpetuity
Change	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic
Pattern	Aperiodic-sporadic	Aperiodic-sporadic	Systematic	Aperiodic-Sporadic	Aperiodic-Sporadic
Dependencies	Analyse, Manage & Distribute components	Distribute components	External systems	All components	All components
Risk or Opportunities	Risk: IMPALA performance degeneration	Risk: IMPALA performance degeneration and security	Risk: Interoperability failure	Opportunity: earlier delivery	Risk: accident
Mitigation or Exploitation	Mitigation: Apache Hadoop distributed processing for performance	Mitigation: Apache Hadoop distributed processing for performance	Mitigation: Standardisation & extensible design	Exploitation: testing	Mitigation: Administration through access control
Outcome (with operators)	Handle MAX POSSIBLE system demand	Handle MAX POSSIBLE analysis operations securely	Require MINIMUM customisations to share or publish data	MAXIMISE successful execution of scripts	MINIMISE data access and privacy violations

10.4 Functional view

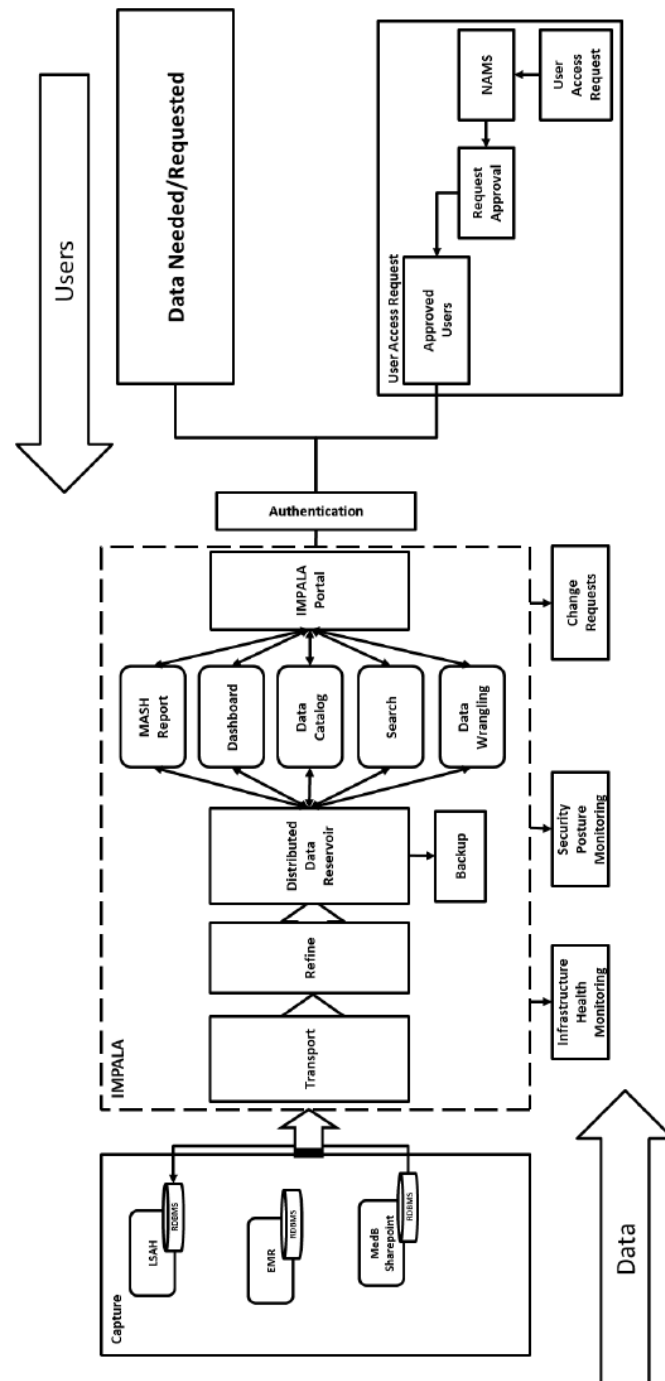


Figure 10.2: Functional view of IMPALA platform from the original image in SDD¹

The functional view in Figure 10.2 presents the components of the IMPALA platform from the perspective of data importation and user interactions. It describes the functionality of each of the architecture components and the user access of the IMPALA platform.

The main components of the functional view are *Authentication*, *IMPALA Portal*, *Analyze*, *Distributed Data Reservoir*, *Backup*, *Refine*, *Transport*, *Infrastructure Health Monitoring*, *Security Posture Monitoring and Change Requests*, *Data Needed/Requested* and *User Access Request*. Some of these components have sub-components which we discuss in the specific context. We consider uncertainty about each of these components and their associated components.

As previously stated, the IMPALA platform is designed to use COTS and other already existing components. For user management in the *Authentication* component, the IMPALA platform will use the existing Johnson Space Centre (JSC) NASA Access Management System (NAMS) as part of *User Access Requests*. Users with credentials interact with the IMPALA platform through the JSC network from onsite workstations or through the JSC Virtual Private Network (VPN) by accessing the IMPALA platform web-portal - the *IMPALA portal* component.

The *Analyze* component consists of the following functional components of the architecture: *Mission Associated Summary of Health (MASH) Report*, *Dashboard*, *Data Catalogue*, *Search and Data Wrangling*. All these components are designed to be customisable and flexible, thus allowing users to explore data even with unanticipated user needs or requests. Users can create information dashboards and visualisations, connect to multiple data sources, define and generate a data catalogue, search for data, and wrangle(or clean) data before or after search.

The *Transport* component is responsible for transmitting the data from the *Capture* component. It is implemented using Pentaho³ Data Integration tools to move / copy data, which is a *Business Intelligence (BI) software that provides data integration, Online analytical processing (OLAP) services, reporting, information dashboards, data mining and Extract, Transform, Load (ETL) capabilities*³.

Furthermore, the *Transport* component helps with scheduling external transformation scripts from the *Refine* component to be called from outside the standard copy of the data. This feature is implemented using Trifacta⁴ which is designed for analysts to explore, transform and enrich raw data in clean and structured formats.

The *Refine* component is implemented based on Trifacta and Pentaho to create and apply rule-based data transformation before loading it into storage. Once created, some transformation can be repeated for the cleaning of the data. Transformations that are saved for reuse are called static transformations.

The *Distributed Data Reservoir* component and its associated *Backup* component are related

³<http://www.pentaho.com/>

⁴ <https://www.trifacta.com/>

to the *Store* component. In addition to storing the data, these storage components support authentication and authorisation. The authentication is controlled using Kerberos⁵, which is a computer network security protocol that authenticates service requests between two or more trusted hosts in an untrusted network.

The authorisation uses role-based access control using Sentry⁶ which is a granular role-based authorisation module for Hadoop. It provides the ability to control and enforce precise levels of privileges on data for authenticated users and applications on a Hadoop cluster. The Backup is implemented through the IMPALA Network Attached Storage.

IMPALA management consists of the *Infrastructure Health Monitoring*, *Security Posture Monitoring*, and *Change Requests* components. These are implemented through existing infrastructure management functionality within the MEME environment.

The *Infrastructure Health Monitoring* uses SolarWinds⁷ which is monitoring software to detect, diagnose, and resolve system issues such as network performance problems and outages. *Security Posture monitoring* uses an agent-based log aggregation tool for security auditing and uses DELL KACE for inventory and system management. DELL KACE⁸ is a system management and deployment product that provides inventory and asset management, software distribution and patch management. The tracking of issues is provided by Jira⁹, which is a software application for issue tracking and project management.

These are the components of the functional view shown in Figure 10.2 and whose known uncertainties we have identified in Table 10.3. Thus, together, the functional view shows the functional interaction and capability of the IMPALA platform. The uncertainties identified and captured in this view have the potential to directly impact the functionality of the system.

⁵<https://www.simplilearn.com/what-is-kerberos-article>

⁶<https://cwiki.apache.org/confluence/display/sentry/sentry+tutorial>

⁷<https://www.solarwinds.com>

⁸ <https://www.quest.com/kace/>

⁹<https://www.atlassian.com/software/jira>

Table 10.3: Uncertainties of the Functional view

Attribute	Uncertainties					
Uncertainty ID	12	13	14	15	16	17
Description	NAMS disruption	User access request approval time	Access connection to the IMPALA platform	Unanticipated data requests	Untrusted access connections	Introduce components to the IMPALA platform
Nature	Epistemic or Aleatory	Aleatory	Epistemic	Epistemic	Epistemic	Epistemic
Bound						
Perspective	Objective	Objective	Objective	Subjective	Objective	Objective
Awareness	Known unknown	Known unknown	Known unknown	Known unknown	Known unknown	Known unknown
Level	High	Medium	Medium	Medium	High	low
Source type	Exogenous or Endogenous	Endogenous	Endogenous	Exogenous	Exogenous	Exogenous
Cause	NAMS access failure	Approval governance committee not available	VPN network done	New data use cases and datasets requests	Internet access connections	Store hardware or software failure
Viewpoints	Functional	Functional	Functional	Functional	Functional	Functional
Facets	Architecture	Architecture	Architecture	Architecture	Architecture	Architecture
Location	User Access Request & NAMS components	User Access Request component	IMPALA portal	Analyse component	Distributed Data Reservoir	IMPALA Platform & MEME network
Manifestation	User access failure	Delays in granting user access	Not connection	Existing features not supporting new request	Connections	Data availability
Measure	Probability	Possibility			Probability	
Monitor	Authentication failure rates	Pending user assess requests	Network status	Frequency of data analysis component use	Data growth & External connections	IMPALA platform changes
Evidence	System access failure report data	Approvals pending data	No connection		Count of non-standard requests	Additional components
Relationship			out-standing connection issues			
Emerging time	Run-time	Run-time	Run-time	Data usage	Run-time	Deployment & Run-time
Lifetime	Perpetuity	Perpetuity	Perpetuity	Perpetuity	Perpetuity	Perpetuity
Change	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic
Pattern	Aperiodic-sporadic	Aperiodic-sporadic	Systematic	Aperiodic- Sporadic	Aperiodic- Sporadic	Aperiodic- Sporadic
Dependencies	IMPALA Portal	IMPALA Portal	IMPALA platform	Distributed Data Reservoir	IMPALA portal, Backup & Analyse	Distribute, & Analyses
Risk or Opportunities	Risk: Users cannot access and use IMPALA	Risk: Delays negatively affecting system use	Risk: IMPALA inaccessible	Risk: Data and system exploitation	Risk: Data security breach	Risk: High cost changes
Mitigation or Exploitation	Mitigation:	Mitigation:	Mitigation: Alternative access through JSC network	Analyse components are customisable and adaptable	Mitigation: Kerberos Authentication & Sentry authorisation	Mitigation: Configurable platform
Outcome (with operators)	IMPALA should be able available AS CLOSE to 100% AS POSSIBLE	Access should be grant WITHIN approved TIMELINES	IMPALA should be able available AS CLOSE to 100% AS POSSIBLE	MAXIMISE data use and exploitation	MINIMISE risk of external data security breaches	MAXIMISE the use of off-the-self-components and existing infrastructure

10.5 Infrastructure view

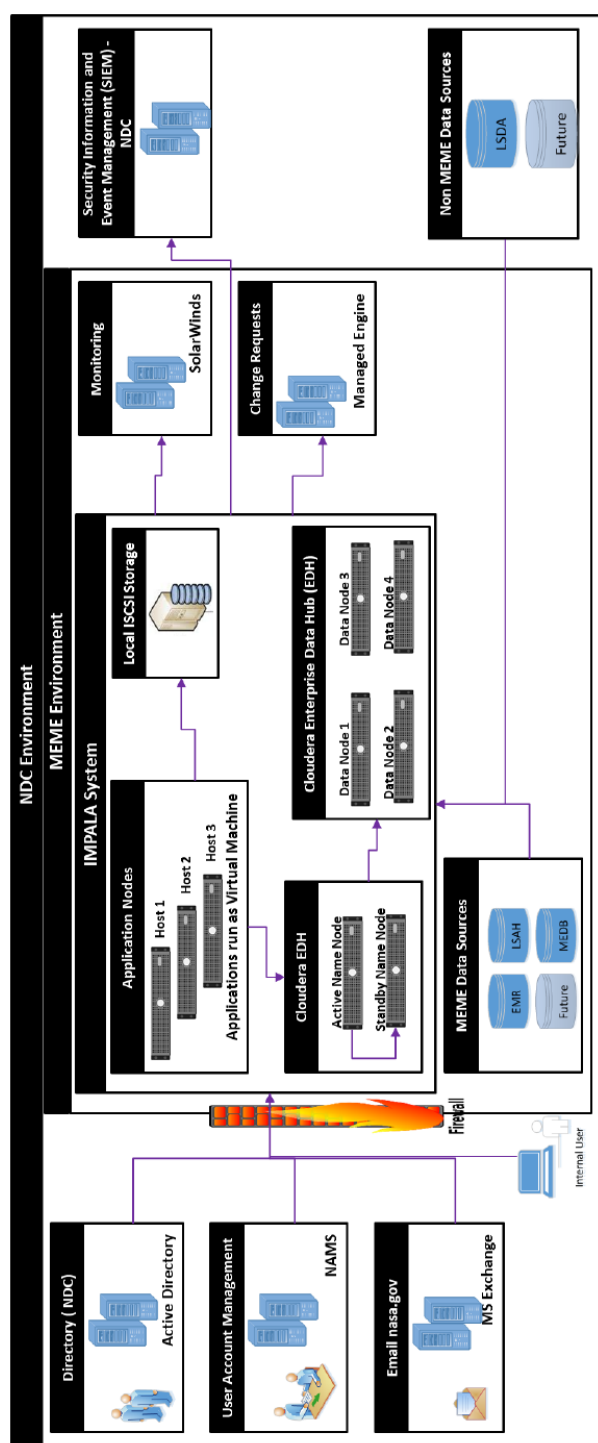


Figure 10.3: Infrastructure view of IMPALA platform from the original image in SDD¹

The infrastructure of the IMPALA platform is self-contained and is housed on a rack within a room. It reside within the MEME network environment behind a firewall. The infrastructure

sub-components are deployed as both virtual machines and physical servers.

The application nodes, which are user-facing, are deployed on virtual machines across three host machines. IMPALA *Distributed Data Reservoir* is deployed on two Master Nodes and four Data Nodes. The network link between the application nodes and the data nodes within the rack is through services on the Master nodes over a 10GB network.

In terms of connectivity infrastructure, the IMPALA rack has four switches, two 10GB switches for intra-rack communication, and two 1GB switches for user and management access, both in a High Availability (HA) configuration. The connection between the MEME firewall is through the user/management switch.

Overall, the IMPALA platform infrastructure supports high performance through the following attributes: clustering, Redundant Array of Independent Disks (RAID) configuration, Distributed storage, Distributed processors, and Network throughput. These attributes contribute to the handling of uncertainties according to the uncertainties of the infrastructure identified in Table 10.4.

Table 10.4: Uncertainties of the Infrastructure view

Attribute	Uncertainties					
Uncertainty ID	18	19	20	21	22	23
Description	Application node access	Link between application nodes and data nodes	Infrastructure performance - Data Nodes	Infrastructure performance - OS hard disks	Infrastructure performance - Processors	Infrastructure performance - Network Throughput
Nature	Aleatory	Aleatory	Aleatory	Aleatory	Aleatory	Epistemic
Bound						
Perspective	Objective	Objective	Objective	Objective	Objective	Subjective
Awareness	Known unknown	Known unknown	Known unknown	Known unknown	Known unknown	Known unknown
Level	Low	Low	Medium	Medium	High	Medium
Source type	Exogenous or Endogenous	Endogenous or Exogenous	Endogenous	Endogenous	Endogenous or Exogenous	Endogenous or Exogenous
Cause	Hardware or software failure	Network link failure or disconnection	Data node failure	Hard disk failure	Increase in processing demand	Data transmission growth
Viewpoints	Infrastructure	Infrastructure	Infrastructure	Infrastructure	Infrastructure	Infrastructure
Facets	Architecture	Architecture	Architecture	Architecture	Architecture	Architecture
Location	Application nodes	10GB network	Nodes	Nodes	Nodes	10GB Network
Manifestation	System down	Loss of connectivity	IMPALA down	IMPALA down	Slow processing	Slow network
Measure	Probability	Possibility	Possibility	Possibility	Processor speed	Bandwidths fluctuation
Monitor	Down-time	Down-time	Down-time	Down-time	Processing speed	Network speed
Evidence	Down-time data	Connection time-out data	No connection		Count of non-standard requests	Additional components
Relationship						
Emerging time	Run-time	Run-time	Run-time	Data usage	Run-time	Deployment & Run-time
Lifetime	Perpetuity	Perpetuity	Perpetuity	Perpetuity	Perpetuity	Perpetuity
Change	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic
Pattern	Aperiodic-sporadic	Aperiodic-sporadic	Aperiodic-sporadic	Aperiodic-Sporadic	Aperiodic-Sporadic	Aperiodic-Sporadic
Dependencies	User terminals	Application & Data Reservoir nodes	IMPALA system	Nodes	Nodes	Nodes
Risk or Opportunities	Risk: Users cannot access and use IMPALA applications	Risk: IMPALA application inaccessible	Risk: Data loss	Risk: Data loss	Risk: System failure	Risk: Network failure
Mitigation or Exploitation	Mitigation: Redundancy - two virtual machines & three host machines	Mitigation: Redundancy - two connections and switches 10GB	Mitigation: nodes are clustered	Mitigation: RAID Configuration ensures continued operations	Mitigation: Distributed Processors	Mitigation: 10GB intra-rack backbone
Outcome (with operators)	MAXIMISE IMPALA application infrastructure reliability	Connection between Application and Data nodes MUST ALWAYS BE AVAILABLE	Avoid drop in unavailability of ANY of the nodes services	ENSURE continued operations in event of failure	Leverage processor across ALL servers to MAXIMISE performance	ENSURE high speed data transfer BETWEEN nodes, data nodes and application nodes

10.6 Network view

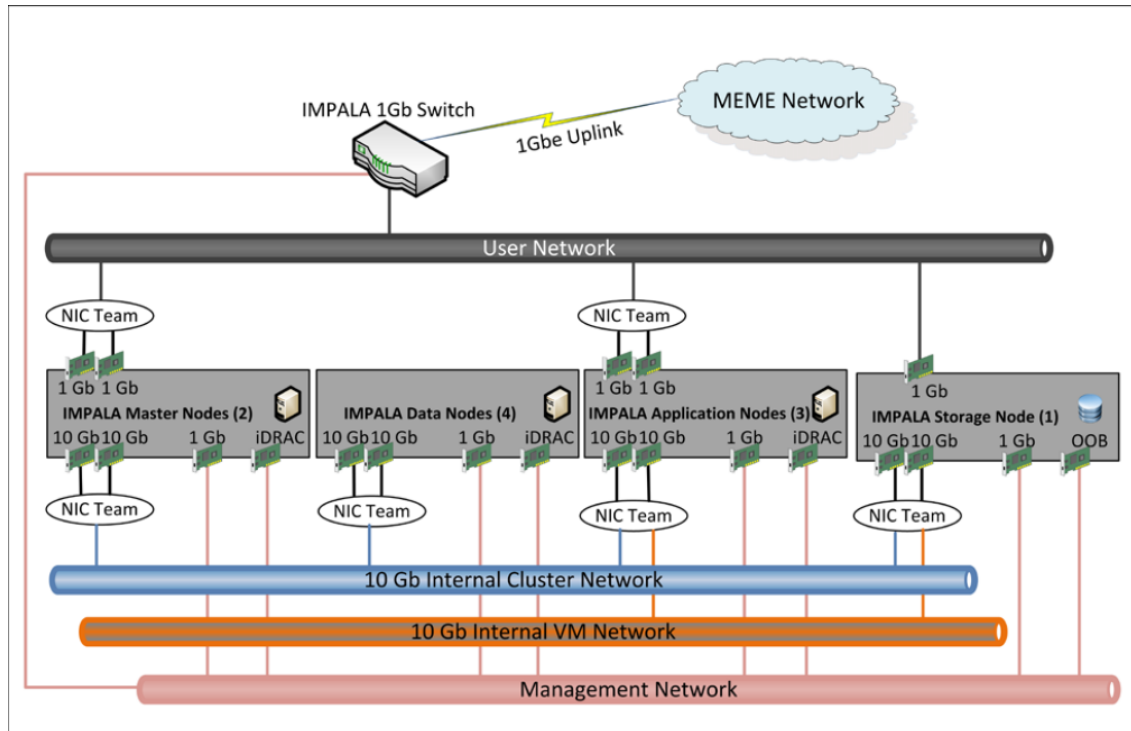


Figure 10.4: Network view of IMPALA platform from the original image in SDD¹

The network architecture view shows the network components of the IMPALA platform. Section 10.5 of the infrastructure view implicitly refers to the components of the network architecture and even identified some related uncertainties in Table 10.4 related to connectivity and throughput. The network view explicitly shows the consideration of multiple Network Interface Cards (NIC) teaming. The uncertainty related to the teaming of network cards to provide redundancy is identified in Table 10.5.

Table 10.5: Uncertainties of the Network view

Attribute	Uncertainties
Uncertainty ID	24
Description	Network Card Status
Nature	Aleatory
Bound	
Perspective	Objective
Awareness	Known unknown
Level	Low
Source type	Exogenous or Endogenous
Cause	Hardware or software failure
Viewpoints	Network
Facets	Architecture
Location	Network Interface Cards (NIC)
Manifestation	Connection down
Measure	Probability
Monitor	Down-time
Evidence	Down-time data
Relationship	
Emerging time	Run-time
Lifetime	Perpetuity
Change	Dynamic
Pattern	Aperiodic- sporadic
Dependencies	Network connections & Nodes
Risk or Opportunities	Risk: Users cannot access and use IMPALA applications
Mitigation or Exploitation	Mitigation: Redundancy: Network Interface Cards Teaming
Outcome (with operators)	MAXIMISE Network availability

10.7 Security view

The security architecture view shows five components of security: *Perimeter Security, Authentication, Authorisation, Encryption and Policy*. Figure 10.5 shows each of these components and their associated implementation technologies/components.

The IMPALA platform is protected from network access vulnerabilities through *Perimeter Security*: Within the MEME environment, IMPALA is protected through the firewall and external user access through the VPN connection.

Section 10.4 previously discussed the implementation of authentication and authorisation security and identified associated uncertainties. The security view clarifies the presentation of these components by explicitly identifying their constituent components. Data in *Data Reservoir* are encrypted, and data transmission is also in an encrypted. The security policy includes the use of antivirus, logarithmic analysis, auditing, and Dell KACE, as shown in Figure 10.5.

10.8 Discussion

The preceding sections generated the uncertainty data based on specific architecture viewpoints. This section assesses the data to evaluate their usefulness in considering uncertainty in software

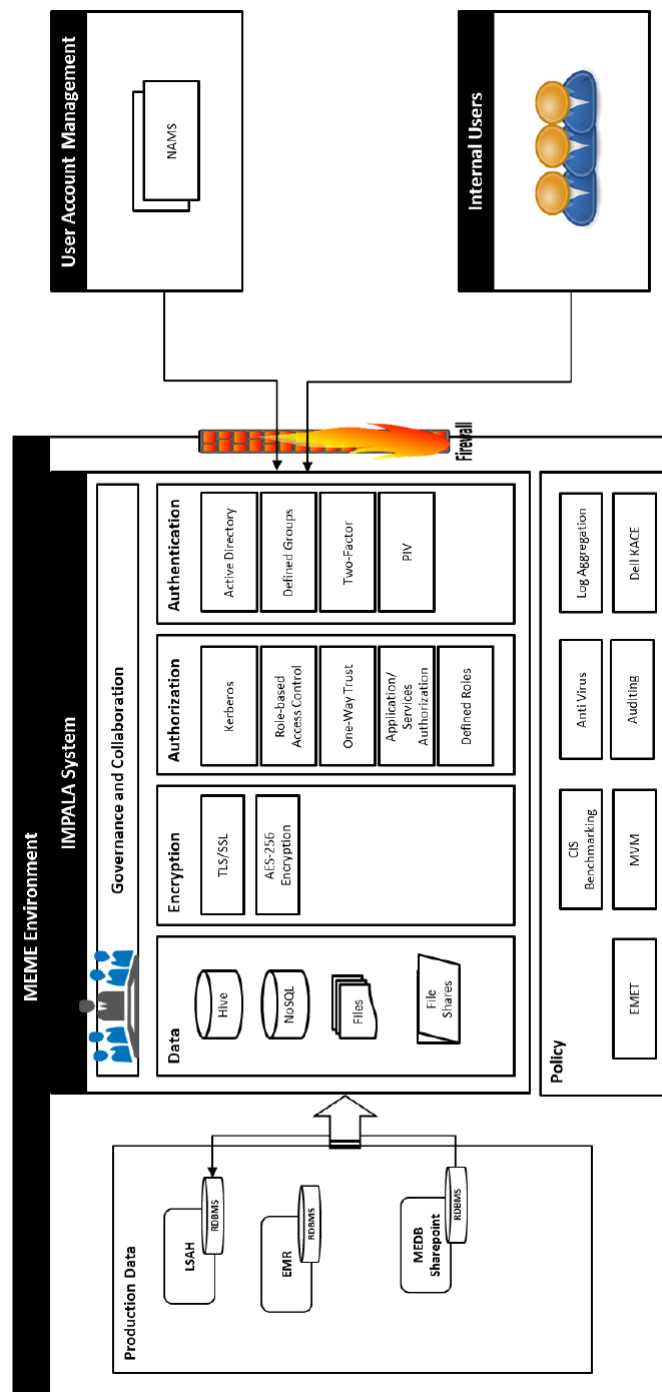


Figure 10.5: Security view of IMPALA platform from the original image in SDD¹

architecture. The analysis aims to raise awareness of the uncertainty within the architecture for its consideration.

The discussion focuses on the following aspects: it presents an overview of the uncertainty data in 10.8.1. Then it presents the uncertainty data as architecture knowledge and documentation in

10.8.2. Section 10.8.3 reviews the quality of the uncertainty data. Section 10.8.5 presents the customised ATAM analysis of the IMPALA platform with quality attributes, architecture styles or decisions, risks, sensitivity points, trade-off points, and rationale. Section 10.7, compares the results of the uncertainty consideration framework with the results of the customised ATAM . Finally, Section 10.9 is the conclusion.

10.8.1 Uncertainty data overview

In total, there are twenty four (24) uncertainties together from Table 10.1, 10.2, 10.3, 10.4, and Table 10.5, identified in the Logical, Functional, Infrastructure and Network architecture views, respectively. These are the known uncertainties. Table 10.6 lists the number of uncertainties per architecture component.

The logical view has eleven (11) identified uncertainties, the functional view has six (6) uncertainties, the infrastructure view has six (6) uncertainties and the network view has one (1) uncertainty.

There are sixteen (16) unique architecture components where we identified the uncertainties. The *Nodes* components found in the Infrastructure view and the *Store* component of the Logical view have the relative highest number of uncertainties of three (3). The architecture design includes mitigations for most of the uncertainties, except for two that are identified in the functional view Table 10.3 with IDs 12 and 13, both of which are related to the *User Access Request* component.

Table 10.6: Uncertainties per architecture component

Architecture Component	Uncertainties
Nodes	3
Store component	3
10GB Network	2
Analyze component	2
Manage component	2
Transport component	2
Application nodes	1
Capture component	1
Distribute component	1
Distributed Data Reservoir	1
IMPALA Platform and MEME network	1
IMPALA portal	1
NIC component	1
Refine component	1
User Access Request and NAMS components	1
User Access Request component	1
Total	24

10.8.2 Uncertainty knowledge and Documentation

As we illustrated in the overview of the uncertainty consideration data in Section 10.8.1, by simply capturing the uncertainties, we create architecture uncertainty knowledge and documentation. Analysing these data can provide potential information, which can be used to help focus the efforts of the architecture designers and stakeholders to identify high-uncertainty areas or components in the software architecture.

10.8.3 Uncertainty data quality

By definition, the set of the identified uncertainties is less than or equal to all uncertainties (the set of all uncertainties is equal to known uncertainties and unknown uncertainties). Considering that we cannot guarantee that the identified uncertainties are exhaustive because at least there is possibility of the existence of the unknown unknown uncertainty; therefore, the current list of uncertainties gives us a partial view of the system's uncertainty status.

Furthermore, like most architecture analysis and evaluation approaches, the process depends on the skills of stakeholders, the available resources to conduct the process, and other subjective factors; thus, the quality of the data could be influenced by these factors. However, various mitigations can be employed, such as an input from a second opinion.

In terms of the individual framework attributes, as has been illustrated through the uncertainties we identified in the views, the attributes are applicable in various contexts, as they are generic to uncertainties. However, customisations can be added to the framework depending on the need, as we discuss later in this section.

10.8.4 IMPALA platform uncertainty consideration architecture analysis - Report

In terms of the uncertainty framework, each of the attributes is important as they provide information about uncertainty or its risks and mitigation as discussed in Chapter 6, the definition of the uncertainty framework. Therefore, in this uncertainty analysis, we review the data on some of the attributes.

The *description* attribute tells us about the individual uncertainties; therefore, these details are specific. However, in the case of a repository of uncertainties, it might be important to perform text processing with the description attribute to identify similarities among the uncertainties or other relationships.

For most of the uncertainties, the *bound* attribute is blank. Two (2) uncertainties with ID 2 and 8 of the logical view, have their bounds limit *up to 99%*. Uncertainty ID 2 is about *Data import jobs capacity to handle data growth* and uncertainty ID 8 is about *IMPALA's capacity to handle analysis demands and security*. The two uncertainties with bounds are uncertainty ID 5 and 6 which are about *Data rapid growth* and *Data storage failure*, respectively.

In terms of the *Nature* attribute, there are twelve (12) aleatory uncertainties and twelve (12) epistemic uncertainties. This is a coincidence that the types are evenly distributed. However, knowing either nature of the uncertainties helps us to think about possible mitigations if required. Aleatory uncertainties arise from variability, thus might require different mitigation approaches to epistemic, which arise from lack of knowledge, therefore can be reduced by managing the knowledge gap. Each individual uncertainty requires a specific approach, as captured by the framework.

In contrast to the *nature* attribute, for the *perspective* attributes, there are eighteen (18) objective uncertainties and six (6) subjective uncertainties. Ideally, there should be consensus among stakeholders about objective uncertainties, since these do not depend on the knowledge or opinion of stakeholders. Subjective uncertainties are often relative and thus might not always generate consensus among stakeholders.

All uncertainties are *known unknowns*, so, for the attribute *Awareness*, the identified uncertainties are known unknowns. However, for the consideration of uncertainty, it is important to know that there are unknown unknowns, and these can equally have a huge risk to the system. Examples of these include natural disasters. In physical engineering construction or complex systems, often structures are built with significant margins to mitigate against unknown uncertainties [2, 9].

An attribute that captures the expert opinion of stakeholders on uncertainty is *Level* of the uncertainty attribute, which is a degree of uncertainty. The uncertainty can be high, low, or medium. These are relative values and should not be considered absolute. Currently, the IMPALA platform has six(6) high uncertainties, thirteen (13) medium uncertainties, and five (5) low uncertainties. When considering uncertainties in the architecture, depending on the interest of stakeholders, uncertainties may be prioritised based on level, in addition to other factors.

Out of the twenty four (24) uncertainties, nine (9) uncertainties are endogenous, and equally, nine are (9) exogenous and six (6) are both exogenous or endogenous relative to the architecture component. The uncertainty *Source type* provides useful information as it can tell stakeholders where the vulnerability of the architecture uncertainties emerges the most. If most of the uncertainties are from external sources, and that is a concern, then an alternative candidate architecture can be proposed which reduces these. For the IMPALA architecture, from the

current data, it seems that the influence of both endogenous and exogenous uncertainties is even. Therefore, the specific details of the uncertainties may need further analysis from the other attributes to inform uncertainty management decisions.

Such specific details about the uncertainty could include understanding the underlying issue, which is captured in the *Cause* attribute. Each one of the twenty four (24) IMPALA platform uncertainty has a cause captured. It may be possible that, at times, the cause of an uncertainty might not be obvious and will require investigation. However, knowing the cause of the uncertainty is important, as this will inform the mitigations. Furthermore, in a larger dataset, data about causes can be analysed to highlight key or frequent causes of uncertainties. Such an analysis might require accumulation of the uncertainty data over a relatively longer period to have sufficient data for significant statistical analysis.

In the Data Overview Section 10.8.1, we highlight the number of uncertainties identified from each architecture *viewpoint* attribute. The *Logical* view has eleven (11) identified uncertainties, the *Functional* view has six (6) uncertainties, the *Infrastructure* view has six (6) uncertainties, and the *Network* view has a single uncertainty. It is important to note that each architecture view presents architecture decisions or styles captured in terms of architecture components and connectors. Therefore, the number of uncertainties identified in a view can be used as a good heuristic to alert stakeholders to review the decisions made within an architecture view. We use the term *heuristic* to emphasise that the uncertainty count only suggests a potential possible area of concern.

In the context of the case study, the attribute *facet* is constant, as the uncertainty characterisation we have carried out is only related to the software architecture.

Again, in the Data Overview Section 10.8.1 we highlighted the individual *locations* or architecture components where the uncertainty manifests itself. Some components of the architecture have more uncertainties than others, as highlighted in Table 10.6. In total, there are sixteen (16) unique architecture components(*locations*) where we identified uncertainties. The *Nodes* components found in the *infrastructure* view and the *Store* component of the *Logical* view have the relative highest number of uncertainties of three (3). The architecture design includes mitigations for most of the uncertainties, except for two that are identified in the functional view Table 10.3 with IDs 12 and 13, both of which are related to the *User Access Request* component.

The *Manifestation* attribute indicates the appearance of the uncertainty or how the uncertainty can be observed after its risk is achieved. Some uncertainties might have a similar manifestation, for example, different uncertainties that cause the IMPALA platform to go down. Uncertainties ID 20 and 21, for instance *Infrastructure performance - OS hard disks* and *Infrastructure*

performance - Data Nodes, respectively, might result or cause the *IMPALA platform to go down*.

Ten (10) of the uncertainties identified have the *Measure* attribute as Probability. The other measure values we have identified are bandwidth fluctuation (1), data breach detection rate (2), data growth rate (1), execution statistics (1), performance fluctuations (2), processor speed (1), refinement correctness (1) and storage failure rate (1). Some of the uncertainties do not have any specific measure.

The *Measure* attribute defines a scale which we can use to assess the uncertainty. In contrast, the *Monitor* attribute is an indicator that we can use to detect uncertainty. Out of the twenty four (24) uncertainties we have identified, most of them have their specific monitors, however, some monitors are similar. For example, *Down-time* of the IMPALA platform is a monitor to five (5) uncertainties, *Integration failure reports* to two (2) uncertainties, and the rest are mapped one-to-one between the monitor and the respective uncertainty.

Both the *Evidence* and *Relationship* attributes can be used to manage a specific uncertainty and therefore are tailored to the uncertainty. Similarly, *Emerging time* shows when uncertainty is expected to manifest itself. For IMPALA, seventeen (17) uncertainties have the emerging time during run-time, while three (3) are during data usage, and another three (3) during deployment and run-time, and finally one (1) is during development, deployment, or run-time. Therefore, it seems likely that the IMPALA platform might be more vulnerable to uncertainties during runtime.

Closely related to *Emerging time*, are the *life-time*, *Change*, and *Pattern* attributes of uncertainty. Out of the twenty four (24) uncertainties, all of them have a life-time of Perpetuity, but one (1) which is limited. All uncertainties are expected to *Change* dynamically, with twenty one (21) with a pattern of Aperiodic-sporadic and three (3) with a systematic pattern.

All twenty four (24) uncertainties have at least a value in the *Dependencies* attribute. This highlights that the uncertainties in the IMPALA platform from one location are likely to impact another uncertainty. The architecture components with the relatively high number of dependent uncertainties are the *Nodes*, the *Distribute* and the *Analyse* components.

Uncertainties can have a positive or negative consequence on the system. Currently, twenty three (23) uncertainties in the IMPALA are associated with risks. Only one uncertainty, ID 10, is associated with a potential opportunity of earlier delivery of results if the management script execution is successful within the *Manage* component of the Logical view.

One of the strengths of the IMPALA platform architecture is that twenty one (21) uncertainties have their mitigations considered within the architecture design. One (1) uncertainty is related to

an opportunity, and thus it has an exploitation considered. Two (2) uncertainties, ID 12 and 13 do not have mitigations defined. These uncertainties relate to *NAMS disruption* and *User access request approval time*, respectively.

Finally, for each uncertainty, we have specific its *Outcome* attribute. This is specific to the uncertainty and related to the quality attributes of the architecture. Most of the outcomes are related to availability, performance, security, extensibility, and scalability.

Therefore, by reviewing the uncertainty data on the IMPALA platform, we can generate uncertainty information about the architecture, which will help stakeholders become more aware of the uncertainty status of the architecture.

In the following Section 10.8.5, we re-analyse the IMPALA platform with the customised ATAM approach to generate a report on the architecture. Then finally, in Section 10.8.6 we compare the insight from the customised ATAM report with the uncertainty data report in terms of the insight into the architecture.

10.8.5 IMPALA platform ATAM architecture analysis - Report

In terms of the ATAM, we highlight the IMPALA platform architecture risks, sensitivity points, and trade-off points with respect to specific architecture styles and quality attributes.

In the ATAM, architecture styles are defined or identified with respect to how they aim to achieve quality attributes. For example, one of the main quality attributes of the IMPALA platform is performance. To achieve this, the general architecture approach of the IMPALA platform is to use COTS architecture components with the Hadoop framework as a key component in the architecture. This approach to achieving high performance is an architecture style. ATAM identifies specific architecture styles in an architecture with respect to quality attributes for analysis.

Thus, in this example, the ATAM identifies the quality attribute as performance and the architecture style as the COTS components built on Hadoop framework. The ATAM analysis of the performance example then proceeds to identify risks, sensitivity points, and trade-off points if present so that stakeholders are aware of them when implementing the architecture. The ATAM only presents information and insight in a report to stakeholders. The next action will depend on what the stakeholders consider appropriate.

Architecture styles represent architecture decisions with respect to quality attributes. The analysis of an architecture decision with respect to a quality attribute results in the identification of its risks or non-risks, sensitivity points or trade-off points.

In terms of the performance quality attribute, using Hadoop might be a good decision without obvious risks, as Hadoop is an established Big Data high performance framework. However, if we analyse the same architecture style (decisions of COTS Hadoop), in terms of the modifiability attribute, we might realise that modifying the architecture by removing the Hadoop framework might not be easy to achieve.

Thus, the same architecture style which works best for the performance quality attribute introduces a risk for modifiability and a sensitivity point - since in terms of modifiability, there is a limit to the customisations which can be applied on COTS components, depending factors such licences and skills. Therefore, in implementing the COTS style, we have to deal with a trade-off between the modifiability and performance quality attributes.

We can conduct a similar analysis of the IMPALA security quality attribute with respect to the IMPALA security architecture style. In ATAM, the overall architecture of a software system is considered an aggregation of architecture styles that focus on one or more quality attributes [119].

In the following analysis of the IMPALA platform using the customised ATAM we introduced in Section 8.6, we will present the architecture styles (which are decisions of the architecture) with respect to one or more quality attributes. For each combination of style and quality attributes, we will present some identified sensitivity points, trade-off points, and risks, if they are available. Each of these will be justified with a rationale, as illustrated in the justification of the previous performance quality attribute example.

Finally, we will compare the results of the ATAM with the uncertainty consideration framework data to check if both approaches can be used to raise awareness to stakeholders to improve the architecture by highlighting key areas of concern. As has already been highlighted through the ATAM discussion in Section 8.6, the purpose of *analysis is to help identify potential areas where special attention might be required within the architecture as the system architecture is implemented or evolves* [119].

Table 10.7: IMPALA customised ATAM analysis results

#	Quality attribute	Architecture style (Decision)	Risk	Sensitivity point	Trade-off point	Rationale
1	Scalability, performance, and extensibility	Assemble and install the IMPALA platform COTS component on the Hadoop framework	Hadoop is a non-risk since its among the top Big data storage and analytics framework	Hadoop is a sensitivity point as the scalability, performance and extensibility of the IMPALA platform depend on it	There is a trade-off in terms of performance and scalability vs extensibility. To take full advantage of the latest release of Hadoop you need to limit the local customisations to compatible extensions with future updates, patches and releases	Hadoop is a high performance, scalable and available framework to store, process and analyze data
2	Availability, performance and scalability	The IMPALA platform server racks will connect through the existing MEME network	The MEME network capability might not cope with future growth and demand on IMPALA	The MEME is a sensitivity point due all the three quality attribute depending on it	The MEME is not a trade-off point with respect to these quality attributes as they are all negatively affected when MEME network is under performing or vice versa	This decision takes advantage of the existing NASA infrastructure
3	Scalability, Availability, Security	Infrastructure services such as DNS, Active Directory are not part of the IMPALA platform.	If the infrastructure services were out sourced, they would have been a risk of control, however, the current services are within NASA, thus this is a non-risk	Infrastructure services are sensitivity point with respect to scalability, availability and security	Security might be enhance with having infrastructure services as part of IMPALA. But excluding the improves scalability and availability of the services	Instead, they are made available from the Resource Directorate through the Network Access Control Board
4	Security	The IMPALA platform will not automatically patch or update application or operating systems	Might miss important update			For security and governance consideration, changes need to be approved before being effected
5	Performance and availability	Power consumption for each IMPALA rack is not to exceed 5000 watts	Risk due to future growth	Power limit	High performance might require more power but if power demands exceed limit, this might negatively affect availability	Limited due to design consideration
6	Accessibility and security	IMPALA access only through NASA network or VPN	Non-risk as since this optimises Security	IMPALA access:network control and restriction	Increase accessibility through widening access can risk security	Access control to improve availability and security

Table 10.8: IMPALA customised ATAM analysis Results

#	Quality attribute	Architecture style (Decision)	Risk	Sensitivity point	Trade-off point	Rationale
7	Extensibility, scalability and performance	Capture component to handle a range of present and future data sources	Unsupported future data sources formats	The capture component range of possible data sources is sensitive to extensibility, scalability and performance	The more the diverse the type of data sources supported, the better the extensibility and scalability. However, this can impact the performance due to the requirement to manage a variety data formats	Big data system have to be flexible to incorporating a range of data sources in addition to relational databases
8	Performance and Security	Transport component to handle importation of data sources	Non-risk since the nature of importation is well understood	The Transport component is a sensitivity point to performance since it can be a bottleneck and a single point of failure	Transport performance may need to compromise with security since transmission security checks and validation can impact performance	Mechanism of running importation scripts to transmit data to destination
9	Availability and scalability	The Store component is built with redundancy and backup on the Hadoop framework	Non-risk as storage is robust in the Hadoop distributed file system	Failure and performance of the storage will impact availability and scalability	No trade-off in terms of availability and scalability	Storage has to be robust and recoverable
10	Performance, scalability, security, availability	Analyze component made up of modular components of MASH Report, Dashboard, Data Catalog, Search and Data Wrangling	Analysis user interface might be vulnerable to user exploitation which can threaten system quality attributes	The Analyze component is a sensitivity point to the IMPALA use interaction which influence all the quality attributes	The more security measure are introduced for users to use the Analysis functionality the less the system is availability and also might affect performance and scalability	The Analyze component architecture design is to be flexibility and customisation to support data exploration and unanticipated use cases or requests
11	Performance, security, and availability	Distribute component is deployed on a master slave style with two master and four slave nodes. Further data is encrypted	Non-risk	The distribute component is a sensitivity areas with respect to performance due to data growth, security due to exploitation and availability due to failure	Encrypting Big Data can affect performance thus are is a trade-off between security and performance	The architecture design ensures robust and high performance through redundancy and parallel process
12	Scalability and availability	Manage component is a composition of various components: security & infrastructure monitor, and issue tracking	Backup or alternative services of management is not discussed	Each module is a sensitivity point with regards to the availability of the management services it provides	No trade-offs	Management functionality are deployed as independent or modules services thus support scalability and availability

Table 10.7 shows the ATAM results applied to the IMPALA platform with respect to the following quality attributes: *Scalability, Performance, extensibility, Availability, Modifiable and Security*. For each of these, we specify the architecture style used to achieve the quality attribute, any potential risks, sensitivity points, and trade-offs, together with a *rationale*.

In total, we include twelve (12) analyses of the various combinations of quality attributes and architecture style. Table 10.7 presents each of these. Overall, there are seven (7) architecture styles that have risks; the rest are non-risk decisions. This means that about 58% of the identified architecture decisions could be risky and therefore could require a further review of these. Of course, this does not automatically mean that the architecture is flawed. The analysis is only to highlight areas of the architecture which require attention. In ATAM, risks are architecture decisions that might have negative consequences and those decisions that have not yet been made [119]. In the case of the IMPALA platform, since the architecture is complete, all the risks are based on decisions made, and thus their consequences have the potential to impact the system while in operation.

In terms of sensitivity points, these are architecture points that have an impact on achieving a quality attribute. The correlation might be positive or negative. The important point about a sensitive point is to know that the desired behaviour of a quality attributes depends on them. For example, if we look at the ATAM analysis ID 1 in Table 10.7: Hadoop is a sensitivity point to achieving scalability, performance, and extensibility of the IMPALA platform. Changing the Hadoop platform impacts these quality attributes. Table 10.7 presents the rest of the sensitivity points.

Sensitivity points are directly related to trade-off points. When there are at least two (2) quality attributes associated with a sensitivity point but in a contradicting manner, it raises a trade-off point. For example, consider the analysis of ID 1 again. In terms of performance and scalability, the use of Hadoop positively facilitates these. However, extensibility of the Hadoop framework might raise a constraint such that the extensions that can be made on the IMPALA platform should be compatible with the Hadoop updates. Otherwise, if the local customisations are incompatible with Hadoop, future updates of Hadoop related to performance or scalability might fail, thus negatively impacting these attributes. Therefore, a trade-off exists in this context. Similar trade-offs on the rest of the architecture have been discussed in Table 10.7.

As we have seen, conducting architecture analysis provides extra insight into the architecture, which, if acted on, can improve the existing architecture. Not all results of the analysis will require immediate action. However, since the architecture of a system is an investment, it makes sense to have this documentation and insight into the system. In the next Section, we compare

the insight we have identified from the ATAM and the uncertainty data to assess if the results of the uncertainty data provide any useful or additional analysis insights.

10.8.6 Comparison of analysis reports insights: uncertainty consideration framework vs the customised ATAM

Software architecture analysis is a multi-stage incremental process that can go over iterations. It requires inputs, processing, and outputs. The first aspect is the input, then the analysis itself, with the final output as a report which shares the results of the analysis.

The uncertainty consideration framework and the customised ATAM analysis have different inputs and analysis processes, so the two aspects of the analysis functions cannot be easily compared. Therefore, we remain with the output of the two analyses to compare the two approaches. However, again, the specific details about the output of the two analysis approaches are different, thus, the individual specific details of the output might not be a suitable basis for comparison.

But a solid basis for comparing the two analysis approaches is whether they contribute to the goal or objective of architecture analysis. As we discussed in the evaluation strategy chapter, Chapter 8, the goal of architecture analysis is to identify insights into the software architecture and to make stakeholders aware of the point of interest in the architecture for its potential improvement. Given this information, a stakeholder may decide to take action, ignore the information, or reserve any action until specific conditions are met.

Therefore, being aware of this rationale for conducting software architecture analysis, our approach to compare the two approaches is to assess whether either approach generates insights that could inform stakeholders about the architecture.

In terms of the ATAM process, in the customised version that we applied it, the input included the software architecture, quality attributes, scenarios, and the analysis results are the risks, sensitivity points, and trade-off points. Taking into account Table 10.7, there are some clear insights from the analysis output that stakeholders should take into account when implementing the IMPALA architecture.

For instance, consider the ATAM analysis ID 5, the architecture constraint on power that it should not exceed 5000 watts per rack is a significant limitation, which requires careful consideration of its consequences. Similarly, the ATAM result ID 6, with respect to accessibility and security of the IMPALA platform, is a significant sensitivity point and raises a useful trade-off between these attributes for consideration.

The results of Table 10.7 provide insight into the IMPALA architecture which stakeholders could find useful and relevant. ATAM is an established architecture analysis methodology; thus, in practice, it is highly likely that such output insights can influence the architecture of a system.

Similarly, if we consider the results of the uncertainty data analysis, the data provide a unique perspective to the analysis of the software architecture. The uncertainty analysis approach is unique, as no such an approach exists that characterises the uncertainty of a software architecture into specific attributes and uses them for architecture analysis. The individual attributes of the uncertainty framework, as we discuss in Section 10.8.4, provide information about the architecture that can be considered to process and manage the uncertainty in the software architecture.

From the uncertainty data analysis report, which we presented in Section 10.8.4, a number of insights have been generated about the IMPALA architecture. Not only in terms of the characterisation of the uncertainty, but also in terms of identifying uncertainties that lack mitigation or might require exploitation. Of course, taking advantage of the analysis result depends on other factors and priorities such as budget, time-lines, and implementation constraints. However, having the uncertainty information in a concise and precise manner ensures that the architecture details are documented and preserved as architecture knowledge for future use.

Thus, comparing the uncertainty consideration framework analysis from Section 10.8.4 and the customised ATAM results from Section 10.8.6, both approaches provide documentation, generate architecture knowledge, and can be exploited for insight to improve or understand the software architecture of a system.

10.9 Conclusions

The IMPALA case study aimed to evaluate the uncertainty consideration framework as an approach to considering uncertainty in software architecture and to assess the feasibility of the approach. We identified uncertainties of the IMPALA platform architecture with respect of various viewpoints. Then we aggregated the uncertainty data for analysis to generate architecture knowledge, documentation, and analysis. To assess the soundness of our approach, we conducted a separate analysis of the IMPALA architecture using a customised version of the ATAM to generate architecture, knowledge, documentation, and analysis. Comparing the results from the two approaches, we note that each provides a unique perspective to the architecture, but both provide insightful information which can inform the architecture stakeholder about its state.

CASE STUDY 3: IoV - COMPARISON OF CANDIDATE ARCHITECTURES ON UNCERTAINTY

This case study is from ITS. A specific category of ITS is the concept of IoV, which is an application of IoT. Various layered candidate architectures for IoV have been proposed with different capabilities. These range from three-layer architectures to seven-layer architectures. In this evaluation, we compare two state-of-the-art 7-layered candidate architectures of IoV based on their potential to handle or mitigate specific common uncertainties of IoV. We identify common uncertainties in the literature on the IoV architecture. The object of the case study is to demonstrate that, using the uncertainty framework data, we can compare candidate architectures and gain insight which can contribute to selecting a particular candidate architecture among alternatives.

11.1 Overview

IoV are intelligent network systems with software-defined networks, software applications, and operate in complex environments [6]. IoV have a variety of potential benefits, including the

sharing of information and communication between vehicles and between vehicles and the transport infrastructure, pedestrians or other relevant entities [6, 7, 16].

However, from these multiple interactions also arise potential uncertainties, such as coordination between vehicles, insufficient information, scalability, etc. However, advances in communication and computing technologies such as cloud and/or fog computing, Big Data processing and analytics, machine learning, and artificial intelligence present opportunities to mitigate such challenges [16].

Various architectures of IoV have been proposed with different strengths [7, 16]. In this evaluation, we compare such candidate architectures of the emerging IoV based on whether they handle the common identified IoV uncertainties. Our approach is to identify common uncertainties about IoV from the documentation and then compare and evaluate the IoV candidate architectures based on their handling of such common uncertainties. Thus, we illustrate a potential application of the uncertainty framework data to compare and rank candidate system architectures.

The whole process has steps from step 1 to step 8 where we present the comparison or ranking result:

1. Identify uncertainties and assign each of them an *Influence measure value*, as illustrated in Table 11.1.
 - a) Decide the ordinal measure to use for *Influence measure value*
2. Identify the candidate architecture to compare or rank
3. Analyse the candidate architecture to assess its mitigations to handling the uncertainty risks.
 - a) Decide the ordinal measure to use for *candidate architecture mitigation degrees*
4. Assign ordinal scores to the candidate architecture to indicate the degree to which they can potentially mitigate against uncertainty risks.
5. Generate the ground data of the architecture as illustrated in Table 11.2
6. Encode the ordinal data of Table 11.2 into numerical values, as illustrated in Table 11.3.
7. Apply specific algorithms details to determine the ranking of the candidate architecture. In this case, we have three algorithms.
 - a) Algorithm one - weighting algorithm, Section 11.3.5.1

- b) Algorithm two - enhanced weighing algorithm with mean, Section 11.3.5.2
 - c) Algorithm three - eliminate similar uncertainties and then weigh, Section 11.3.5.3
8. Present the results which might be valid, invalid, or inconclusive.

In the following section, we introduce each of the architectures and the associated uncertainties. Then we define the basis for comparing the uncertainties. After that, we compare the architecture and present the results of the analysis.

11.2 IoV Architectures

IoVs are complex cyber-physical systems with numerous components that interact and a huge amount of data generated to facilitate vehicle operation activities such as driver safety, traffic efficiency and infotainment [6, 7, 16]. IoV possible range of communications include inter-communications between vehicles and other objects using Vehicle-to-Vehicle (V2V), Vehicle-to-Road (V2R), Vehicle-to-Infrastructure (V2I), Vehicle-to-Building (V2B), Vehicle-to-Home (V2H), Vehicle-to-Everything (V2X) and Vehicle-to-Grid (V2G). Also within the vehicle are intra-communications such as information exchanges between Vehicle-to-Device (V2D), Vehicle-to-Sensor (V2S) and Device-to-Device (D2D) [6, 7, 16].

In smart cities, inter-communication could also include other environmental elements, such as monitoring pollution and using vehicles for environmental awareness with the range of sensors, including visual monitoring [6]. Figure 11.1 shows an overview of the IoV environment. In this section, we introduce the two IoV architectures which we evaluate in this case study.

11.2.1 Candidate Architecture One

Figure 11.2 shows the components of one of the proposed Universal Internet of Vehicles (UIoV) architecture, which considers the operation of IoV in the context of smart cities as illustrated in Figure 11.1.

One of the challenges with architecture representation in practice is that architecture is rarely represented with a formal consistent notation. Often, architecture is represented using informal notation with a focus on presenting the idea while generally ignoring formal notations. In this case study, we use the original image of the UIoV architecture as shown in Figure 11.3.

The issue of consistency among architecture representation approaches is one reason why we designed the uncertainty framework to be generic without prescription for specific guidelines for

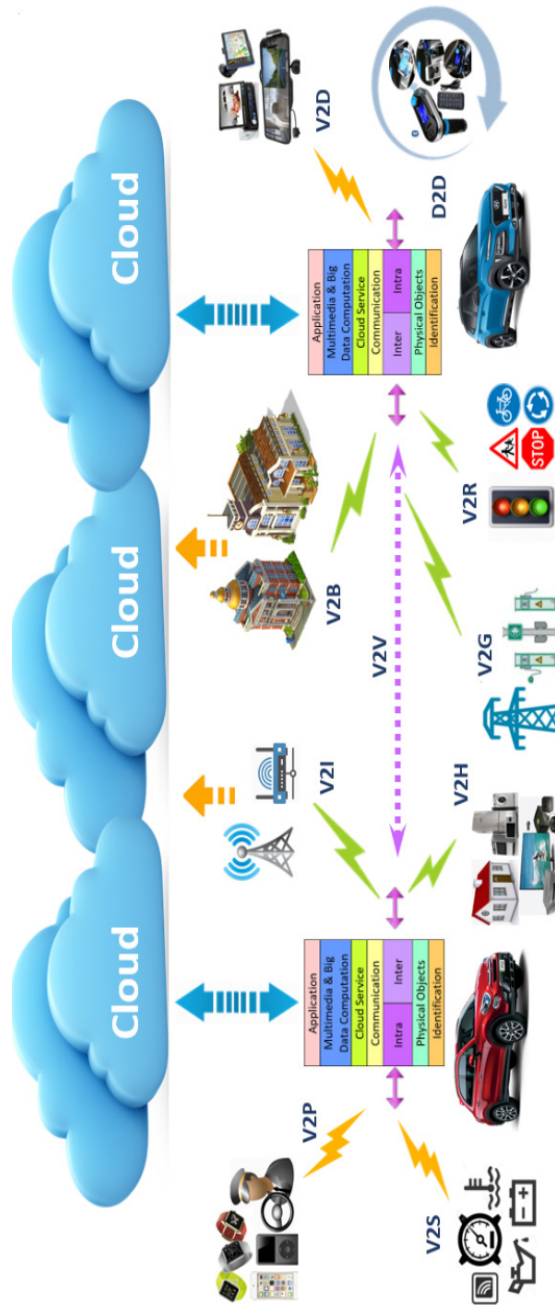


Figure 11.1: Universal IoV for smart cities overview (Image from [6])

its application. Thus, the framework can be applied in a range of different contexts with minimal customisations.

The seven(7) layers that make up the UIoV architecture include the *Identification*, *Physical Object*, *Inter-intra Devices*, *Communication*, *Cloud Services*, *Multimedia & Big Data Computation* and *Applications* layers. In the following listing, we present the functionality and responsibility of

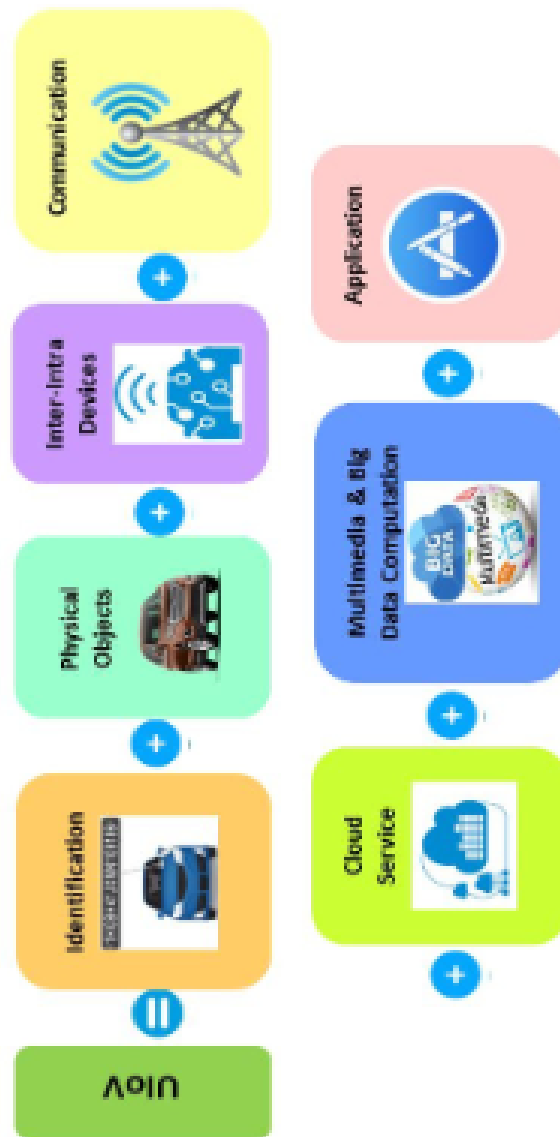


Figure 11.2: Universal IoV Layers (Image from [6])

each of the layers.

1. *Identification layers* assigns identifiers to IoV components. Within the IoV, there are two categories of components, vehicles and non-vehicles. The Identification layer is responsible for assigning such objects unique IDs for global identification.
2. *Physical Objects layer* is responsible for collecting data from all components, vehicles and non-vehicles, within the UIoV environment. The data are sent to the intra- or inter-device layer for processing. Thus, the Physical Objects layer is responsible for collecting data from all UIoV components for the intra-inter layer.

3. *Inter-intra devices layer* is unique to UIoV. It works in coordination with *Communication layer* to support UIoV communications. Intra components refer to sub-components of an UIoV component. For example, a vehicle can have intra-components such as a traffic warning system or a direction system like Global Positioning System (GPS) . The inter-components refer to external components such as vehicles, pedestrians, road infrastructure, etc.
4. *Communication layer* is responsible for communicating among the IoV components, including handling communication between heterogeneous components. Therefore, this layer includes adapters to manage translation between heterogeneous components and accommodate diversity and variety of components. In addition, this layer needs to handle low-power communication in a noisy environment.
5. *Cloud services layer* support scalability and performance of UIoV by providing both computing power, storage, and infrastructure. Cloud services can be implemented through both public clouds and private clouds.
6. *Multimedia and Big Data Computation layer* is composed of three sub-layers which highlight its functionality and performance: data pre-processing sub-layer, big data computation sub-layer and intelligent transportation sub-layers. Together with the appropriate hardware, this layer is designed to be high-performance for Big data processing, analytics and intelligence libraries.
7. *Applications layer* is user orientated where smart application operate to provide services ranging from traffic safety and efficiency to multimedia base infotainment.

Thus, these seven layers present the architecture of the UIoV. In the next section, we present the architecture of the second candidate of the IoVs, which also has seven layers.

11.2.2 Candidate Architecture Two

Figure 11.3 shows the seven (7) layers of the second candidate architecture. The architecture is designed with the seven layer to support the transparent interconnection of all components of the IoV network and the distribution of data within the IoV environment [7]. Of particular interest in this candidate architecture is the security layer, which is responsible for the following: managing authentication, authorisation, and monitoring and auditing all transactions between all entities in the IoV environment [7].

The other architecture layers are: *User interface, Data Acquisition, Filtering & Pre-processing, Communication, Control & Management, and Processing* layers and a cross-cutting *Security*

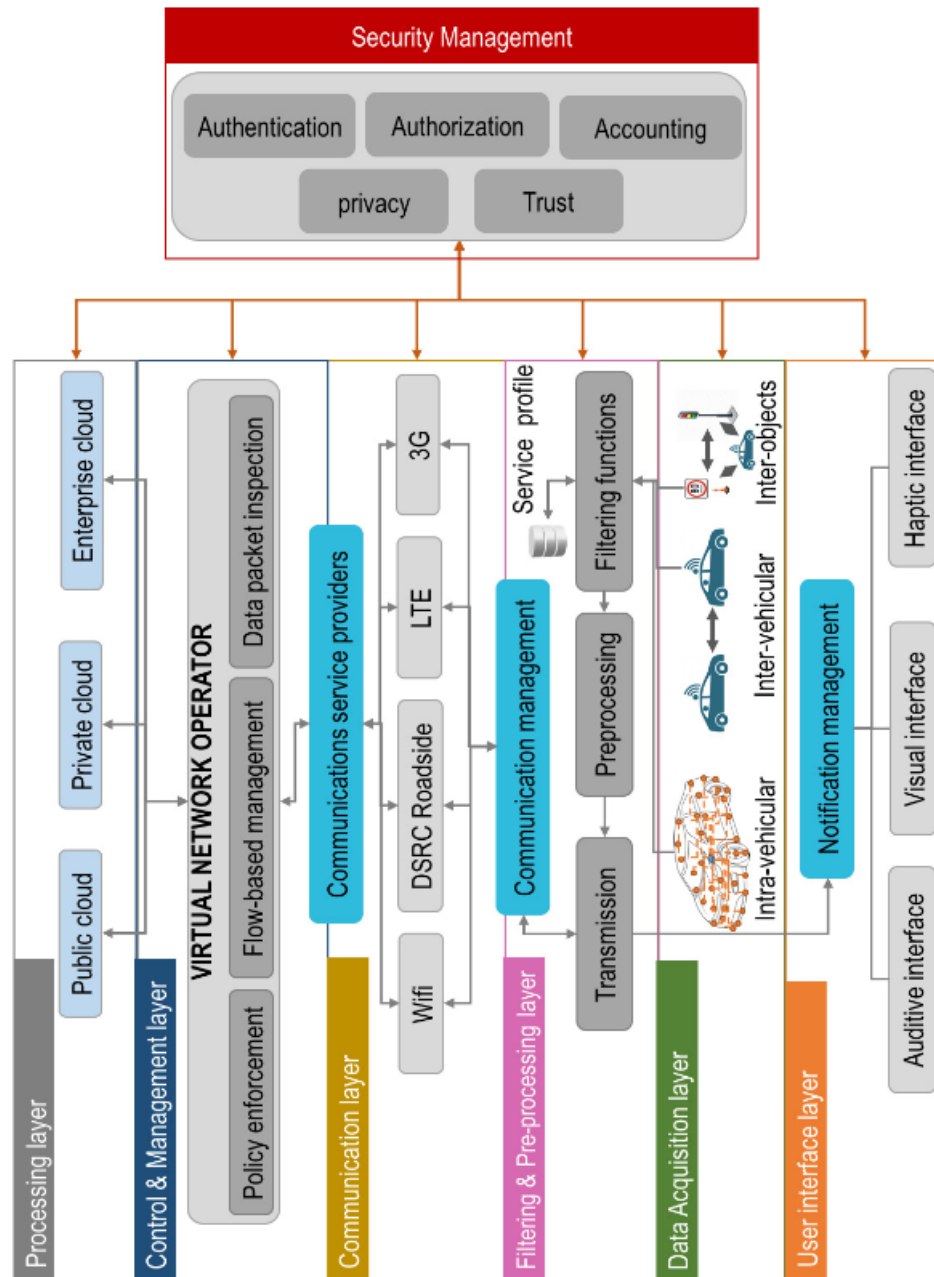


Figure 11.3: Candidate architecture two - IoV Layers (Image from [7])

Management layer. All of these layers work together to support the seamless and explicit interconnection of all IoV elements in all environments. Each of these has the following responsibilities:

1. *User interface* or vehicle interface which handles communication between driver and vehicle. It includes a management unit to coordinate notifications to the driver and a user interface to minimise distractions.

2. Next, we have the *Data Acquisition* layer, which aims to collect data from a variety of sources. Data sources include internal vehicle sensors, the GPS, traffic lights, road signals, etc.
3. Taking into account the data collected in the *Data Acquisition* layer, the next layer is the *Data Pre-processing* layer, which filters out irrelevant data and thus reduces data growth. Data filtering is configured through services created for the vehicle that has subscribed or activated services.
4. After the data have been filtered, the next step is to transmit them to their destination. The *Communication* layer is responsible for this role. One of its key functions is to select an optimal network to send the data. Several factors are considered in making the decision for the optimal path: These include congestion and quality of service levels over network options, information relevance, privacy, and security, among other considerations.
5. Once the information is sent, the next activities relate to controlling and managing the transmission networks. This is achieved with the *Control and Management* layer. At this layer, the various network service providers within the IoV are controlled and managed. Examples of such functionality include traffic management and traffic engineering.
6. The *Processing* layer follows the *Control and Management* layer. Its purpose is to process large amounts of data and information. This is where the cloud computing infrastructure is based. The key features of this layer include the storage, processing, and analysis of the information received from the other layers.
7. Finally, the *Security Management* layer extends to the other layers since security is a consideration in each aspect. It provides features such as data authentication, integrity, non-repudiation and confidentiality, access control, availability, etc. In general, the security layer aims to provide mitigation solutions to address security threats from various possible attacks.

These seven layers present the second candidate architecture. Similar to the first architecture, this architecture is presented with semi-formal notation with the architecture providing a logical overview of the IoV system.

In the next section, we describe the approach we used to compare the two candidate architectures based on the common uncertainties that we identified. Additionally, we introduce specific attributes in addition to the existing attributes of the uncertainty consideration framework to

capture the potential influence of an uncertainty on the architecture. This illustrates the use of the extensibility feature of the framework.

11.3 Candidate Architectures Comparison approach

Uncertainty can have various degrees of influence on a system. Among the attributes of a unit uncertainty, for this case study, we have introduced an influence attribute. This attribute is used to capture the potential influence of individual uncertainty, as we discuss in the next section.

11.3.1 Uncertainty influence measure attribute

The case study uses the uncertainty influence attribute to reason about the potential vulnerability of a system's architecture to uncertainty. The architecture designer is meant to assign the value of the influence measure attribute. To do this, they should assess the unit uncertainty information to come up with a rationale to justify their influence value choice. Thus, their choice depends on both their expertise and the details of the unit uncertainty.

The use of user input is familiar in software architecture analysis and evaluation activities. Various software architecture evaluation and analysis methods use the input or opinion of stakeholders to reason about architecture analysis and evaluation [40, 119]. However, one limitation of such approaches is that they depend on the knowledge of stakeholders, making them prone to subjectivism and bias. Some mitigations to such threats include techniques such as voting, averaging, and aggregation of input to achieve consensus [27, 86, 120].

In the context of this case study, the framework uses rationale and uncertainty details to justify a user assignment of an uncertainty influence measure value. This does not guarantee accuracy and objectivity, but encourages the architect to consider the available information on the uncertainty while assigning the *Influence measure*.

11.3.2 Framework extensibility - *Influence measure* attributes

As stated above, the uncertainty *influence measure* captures the potential impact of an uncertainty on the architecture. The architecture designer assesses the attributes of the uncertainty to arrive at a judgement of the influence values of the unit uncertainty. Each category of uncertainty attributes can provide details that can help the expert make a judgement about the uncertainty.

The uncertainty attributes, as defined in the framework, can contribute the following details: the descriptive attributes can provide factual details of the uncertainty. The source attribute can provide details on the cause of the uncertainty. The manifestation attribute can deal with details

about the materialisation of the uncertainty. Mapping attributes can contribute to the influence assessment by presenting information about the relationship and impact of the unit uncertainty with other uncertainties and architectural elements.

11.3.3 The common uncertainties

So far, we have identified twelve (12) common uncertainties for IoV, which we use to assess candidate architectures. These uncertainties were identified in a similar approach we followed to identify uncertainties in the previous two case studies. The raw uncertainty data are available in the Appendix C in Tables C.1 and C.2.

The description attributes of each of the twelve (12) uncertainties have the following values: *IoV network bandwidth*, *IoV computation capability*, *IoV storage capacity*, *IoV network failure*, *IoV coordination failure*, *IoV vehicle data and other components growth*, *IoV future services*, *IoV protocols variations and heterogeneous*, *IoV range of communication or interconnection channels*, *IoV real-time operation*, *IoV security* and *IoV mapping localisation*.

For each of these uncertainties, we assigned an ordinal¹ value to its *Influence measure* attribute as shown in Table 11.1. Each *influence measure value* is based on the expert opinion given the uncertainty details to justify its assessment. We use the following ordinal¹ values: *Very High*, *High*, *Medium* and *Low* to indicate the potential impact that uncertainty could have on a system if its not mitigated and its risks manifest.

Since these are ordinal values, by definition, the differences between the values, for example, between *Very high* and *high* do not have a significant meaning. The purpose of the individual ordinal values is to signal that a degree of difference exists between values with one more extreme than the other, therefore, supporting ordering or ranking. Thus, in terms of uncertainties, having, for example, a *Very high influence measure* simply means that the particular uncertainty is more significant to the architecture than the others with lower values, such as *High or Low*. The use of an ordinal measure ensures that individual uncertainties can at least be ordered or ranked¹.

Table 11.1 shows each uncertainty description and its associated potential *influence measure* in the system. In the next section, we explore each of these uncertainties with respect to the two candidate architectures, and then compare and rank the architectures depending on their potential vulnerability to these uncertainties.

As indicated previously, an existing challenge in architecture evaluation and analysis is that, most of the time, the source of ground truth data depends on expert opinion or stakeholder interest,

¹https://en.wikipedia.org/wiki/Ordinal_data, last accessed on 05 June, 2022

Table 11.1: Uncertainties and their influence measures

	Uncertainty description	Influence measure
1	IoV network bandwidth	High
2	IoV computation capability	Medium
3	IoV storage capacity	High
4	IoV network failure	Medium
5	IoV coordination failure	Medium
6	IoV vehicle, data, and other components growth	High
7	IoV future services	Medium
8	IoV protocols variations and heterogeneous	Medium
9	IoV range of communication or interconnection channels	Medium
10	IoV real-time operation	Very high
11	IoV security	High
12	IoV mapping localisation	Medium

thus, the validity of most analyses or evaluation results are vulnerable to this threat [27, 120]. Our assignment of the *Influence measure* values to the twelve common uncertainties vulnerable to this threat.

In the next step of the evaluation, which we present in Section 11.3.4, for each uncertainty in Table 11.1, we assess the degree to which a candidate architecture mitigates the risks from each of these twelve (12) uncertainties. For each candidate, we assign a corresponding score to indicate the degree to which its mitigates against the risks from the uncertainty. Section 11.3.4 presents the justification or rationale for assigning a particular ordinal mitigation value to each candidate. The resulting data are captured in Table 11.2 which is used as the basis for the final analysis.

11.3.4 Uncertainties analysis - ordinal data generation

For each uncertainty handled, we assign it an ordinal measure, which signifies to what extent the uncertainty is handled or controlled; collectively, we use the term mitigated. The ordinal measures that we use to express this are the following: *Strongly Mitigated*, *Mitigated*, *Partially Mitigated* and *Unmitigated*.

In the context of this case study, *Strongly mitigated* refers to a solution that explicitly considers uncertainty and has been verified as a suitable mitigation. For example, consider an architecture decision on a vehicle computation or processing requirements. If an architecture proposes a solution that requires a minimum computational capacity for the vehicles to join the IoV, this could *Strongly mitigate* against the risks of not meeting this computational requirement standard.

Table 11.2: Uncertainties, Influence, and Candidate Architectures

	Uncertainty description	Influence measure	Candidate One	Candidate Two
1	IoV network bandwidth	High	Partially Miti- gated	Mitigated
2	IoV computation capability	Medium	Mitigated	Mitigated
3	IoV storage capacity	High	Mitigated	Mitigated
4	IoV network failure	Medium	Mitigated	Strongly miti- gated
5	IoV coordination failure	Medium	Mitigated	Partially miti- gated
6	IoV vehicle, data, and other components growth	High	Strongly miti- gated	Strongly miti- gated
7	IoV future services	Medium	Strongly miti- gated	Mitigated
8	IoV protocols variations and heterogeneous	Medium	Mitigated	Mitigated
9	IoV range of communi- cation or interconnection channels	Medium	Strongly miti- gated	Mitigated
10	IoV real-time operation	Very high	Mitigated	Mitigated
11	IoV security	High	Partially miti- gated	Mitigated
12	IoV mapping localisation	Medium	Mitigated	Mitigated

Thus, this *Strongly mitigates* against the computational uncertainty risks.

The *Mitigated* ordinal measure signals a specific mitigation against an uncertainty that has been explicitly considered to handle or control the uncertainty. It does not *Strongly mitigate* the uncertainty, but it does mitigate it. For example, when there is uncertainty about *Quality of services regarding communication*, the architect may consider a decision about a layer or component that aims to select the most optimal communication option available. This might be a good solution to *Mitigate* against uncertainty of the quality of services. But this does not guarantee *Strong mitigation*, as a suitable option may not always be available to select from among the set of available options.

The *Partially mitigated* refers to mitigations that can potentially be assigned or claimed to have the capacity to handle or control some known uncertainty. Such mitigations are often not explicitly defined for the purpose of mitigating the uncertainty but can be inferred or demonstrated from the context to have the appropriate mitigation effect. For example, given an uncertainty related to *Network bandwidth* fluctuations, an architecture with a design to handle or control communication within a noisy environment might also handle or control communication

disruptions arising from *Network bandwidth* issues. Although the mitigation is not specifically designed for this purpose, we consider this *Partially mitigated*.

Finally, the *Unmitigated* ordinal measure is an extreme where an uncertainty exists without an explicit or implicit and obvious mitigation. This uncertainty exposes the system to the consequences of unmitigated risks. Such unmitigated uncertainties could also arise from known unknowns.

In the following list, we consider each of the two candidate architectures and assess how they mitigate the risks of the uncertainties in Table 11.1. The twelve (12) uncertainties in Table 11.1 are assigned a corresponding mitigation score for each candidate architecture: *Candidate one* and *Candidate two*. We use this analysis as a basis to compare and rank the two candidate architectures with respect to the degree with which they are likely to be vulnerable to these twelve (12) known uncertainties. The results of the analysis are captured in Table 11.2.

1. The *IoV network bandwidth* uncertainty is aleatory, with the network bandwidth variation between the maximum available bandwidth and zero. The fluctuation of the bandwidth risks the quality of services of IoV degeneration. Ideally, the IoV should aim to maximise the use of the bandwidth to ensure network reliability and quality.

Candidate One architecture manages this uncertainty in its *Communication layer* where issues such as noisy communication in a low-power environment are considered. However, this mitigation is not explicitly for *IoV network bandwidth* uncertainty, thus we assign it a score of *Partially mitigated*.

Candidate two similarly has a *Communication layer* where network related uncertainties are mitigated. Specifically, with regard to bandwidth, communication layers include explicit consideration of quality of service where data transmission is optimised by sending it through the most effective option, depending on factors such as congestion and quality of service levels over the network options, information relevance, privacy, and security, among other considerations. Thus, we assign it a score of *Mitigated*.

2. The *IoV computation capability* is related to the computational power that could be required to process data and run IoV applications, in addition to the real-time operational requirements. Considering that vehicles may not have high computational power at their disposal, solutions within the architecture are required that mitigate such an uncertainty.

Candidate One has a specific layer dedicated to cloud computing which handles scalability or IoV computational capability. This layer considers various cloud computing technologies including the public and private cloud. Vehicles and other elements can take advantage

of cloud computing infrastructure to enhance their computational capabilities. Thus, in case of manifestation of risks from *IoV computation capability* uncertainty, the additional computation capacity from cloud resources can help mitigate, if the connection is available. Thus, this uncertainty is *Mitigated*.

Candidate two architecture has *Processing layer* where computational uncertainty can be handled by the IoV using cloud computing infrastructure (private, public and enterprise). Cloud computing is one of the key enablers of IoV; however, when the network is not reliable, it can pose challenges in terms of real-time processing. Another enabling technology is fog computing, which prioritises infrastructure near the IoV nodes, thus mitigating issues such as latency. However, mitigation in the context of *Candidate two* only focuses on cloud computing, which still handles this uncertainty, thus it is *Mitigated*.

3. *IoV storage capacity* is an area of uncertainty in IoV, considering the amounts of data growth, individual IoV elements or components are likely to generate or require more data than they can efficiently store or process, thus there is a need for Big Data solutions within IoV. Among the possible solutions to Big Data challenges are the cloud solutions which both of the two candidate architectures incorporate in their design; thus, they both mitigate this uncertainty.

Candidate One architecture, as discussed previously, incorporates the cloud architecture style within its architecture, thus *mitigates* this uncertainty.

Candidate two architecture, equally, as previously discussed, incorporates the cloud architecture style within its architecture, thus *Mitigates* this uncertainty. We consider it not *Strongly mitigated* as the cloud infrastructure could be unavailable due to uncertainties such as *IoV Network failure*.

4. The *IoV network failure* uncertainties are related to the range of potential sudden connectivity issues that might arise within the IoV. The cause of these could include hardware failures of IoV component network receivers, transmitters, and relays, among other possibilities. The risks of *IoV network failure* include data loss, related disruptions, and accidents.

Candidate One architecture handles uncertainty in network issues through abstracting the communication within the *Communication layer* and the *Inter-intra devices layer* such that both individual IoV elements and sub-elements communicate at an abstract level. Of course, in the case of actual physical damage, in the absence of redundancy, the failure of the network can be permanent; thus, we consider this uncertainty *Mitigated* instead of *Strongly mitigated*.

Candidate two architecture includes a more explicit consideration of the robustness of the

network within its *Communication layers* which has a *Communications service provider sub-layer* that has communication options such as Wi-Fi, DSRC Roadside, LTE, and 3G. This provides explicit alternatives in the case of a specific network failure due to the hardware of some of these alternatives. Thus, we assign this architecture a score of *Strongly mitigated*.

5. The *IoV coordination failure* uncertainty might arise when interactions among IoV components fail. These include inter-component coordination, such as between vehicles, and intra-component such as among devices within a vehicle and inter-object, for example, between a vehicle and road-traffic-light infrastructure or cameras. Coordination is a key part of IoV applications, which facilitate driving safety and congestion management, among other applications. For coordination to be operational, there must be both network and storage capabilities to support coordination, in addition to the implementation of specific coordination software applications.

Candidate One architecture supports coordination through unique identifiers of IoV components, both vehicles and non-vehicles, specified within the *Identification layers*. In particular, coordination is supported through the robustness of *Communication layer* and *Intra-inter devices layers* where coordination challenges *Mitigated*. Additional coordination challenges might arise for applications. This architecture considers an *Applications layer* but does not discuss specific details that we can relate to the *IoV coordination failure* uncertainty. Thus, we consider this uncertainty *Mitigated*.

Candidate two architecture considers coordination at the user level with *User interface layer* which handles coordination of notifications between the driver of the vehicle and the various interfaces within the vehicle to minimise distractions. The *Communication layers* and *Control and Management layers* facilitate coordination between components through their effective communication processing. However, this observation is not explicitly stated within the design; therefore, we assign the mitigation measure of *Partially mitigated*.

6. The *IoV vehicles and other components growth* uncertainty is related to the potential uncertain growth pattern that can emerge within the IoV due to demand and other factors. The number of vehicles might grow; similarly, the number of components - whether within a vehicle or the supporting infrastructure such as traffic lights, cameras, smart-road technology - might as well grow in uncertain or unforeseen patterns causing pressure of the IoV, such as negatively affecting the quality of services and management of scalability.

Candidate One architecture has been designed with scalability and Big Data management as key drivers. Individual layers support specific scalability and growth characteristics. Specifically, the *Cloud services layers* and the *Multimedia and Big Data Computation layer*

define specific capability to handle uncertainties of potential growth requirements. The architecture is designed with specific capabilities to handle growth uncertainty; therefore, we consider this *Strongly mitigated*.

Candidate two architecture is designed similarly with growth as one of its key quality attributes. Each of its layers specifies a specific capability to support scalability and handle growth in terms of Big Data. Some specific layers include specific capabilities, such as within *Data Acquisition layer*, which collects data from a range of sources. Then *Filtering & Pre-processing layer* removes irrelevant data to manage data growth. The *communication layer* is designed to handle a range of possible communications and therefore supports scalability. Each layer is designed with such specific capabilities to handle growth uncertainty; therefore, we consider this *Strongly mitigated*.

7. The *IoV future services* uncertainty related to potential new services that could emerge over time as the IoV transform. Initially, the IoV could focus on essential driving services, such as road safety and traffic management, among other road services. However, as IoV develops and evolves, other services may need to be introduced, such as providing a comfortable driving experience, including infotainment and efficient driving for environmental protection.

Candidate One architecture has been explicitly designed with future services in mind considering the evolution of IoV with smart cities. In this context, in addition to essential driving services, the architecture considers infotainment services and using IoV in smart cities to manage services such as environmental pollution monitoring. Thus, this uncertainty is *Strongly mitigated*.

Candidate two architecture has been designed with explicit consideration of essential services and infotainment future services. However, it does not consider future services such as using the IoV to manage the environment within a smart-city. Therefore, this architecture handles essential and information services but could be vulnerable to the requirements of new future services, so we consider the uncertainty *Mitigated*.

8. The *IoV protocols variations and heterogeneous* relates to the number of different vehicles or components that operate or can connect to the IoV and associated services. Such differences might introduce uncertainty in terms of variations and heterogeneity. Thus, an IoV architecture should be designed to support variations and heterogeneity uncertainty.

Candidate One architecture has *Identification layer* which allows the abstraction of IoV components such that they can be uniquely but uniformly identified, independent of their heterogeneity and specific protocol variations. Furthermore, the other layers support variation

and heterogeneity by providing generic approaches of interactions among objects, such as *Physical Object layer* and *Communication layer*. However, considering that heterogeneity and protocol variation can manifest in a range of unknown ways, we consider it *Mitigated*, as it would be hard to justify *Strong mitigation* considering the range of possibilities.

Candidate two architecture *Mitigates* the uncertainty from heterogeneity and variations in its various layers through generic interfaces and standardisation. It does not have a specific layer which abstracts IoV components and objects. Communication depends on the standard network infrastructure from *Communication layer* and also the generic process from *Data acquisition layer*, which can collect data from a range of heterogeneous and various protocols, including vehicle internal sensors, GPS, traffic lights, road signals, etc.

9. The *IoV range of communication or interconnection channels* uncertainty relates to the possible communication channels with which IoV can be requested to support. Ideally, the universal connection should be between *Vehicle and everything* [16]. However, specific architectures are designed to handle a sub-set of the possible connections [16]. With the *Vehicle and everything* model, new connections might emerge from unexpected sources or objects, such as environmental monitors to monitor air pollution using vehicles or various motion sensors in addition to driving-related activities [6].

Candidate One architecture is design as a universal architecture and, in the context of smart cities, therefore, considers the possibility of communication with a range of channels. Its communications depend on *Identification layer*, in addition to *Physical objects*, *Inter-intra device* and *Communication layers*. Thus, this uncertainty is *Strongly mitigated*.

Candidate two architecture is designed for IoV with a focus on communication between vehicles, within a vehicle, and the communication between a vehicle and inter-connected infrastructure. It does not explicitly consider connections with everything in a smart city, but its standard network technologies might still support most of the demands from such an unexpected connection request, if required. Thus, we consider the uncertainty *Mitigated*.

10. The *IoV real-time operation* uncertainty relates to the requirement that IoV must operate in real time for some applications, such as traffic management or vehicle warning systems, to avoid or mitigate accidents. The real-time operation is related to both network connectivity and the available capacity to process data or information in real-time, in a timely way.

Candidate One architecture includes various architecture styles and elements, which enhance the robustness of the real-time operation of the IoV. These include the use of *Cloud services layer* to enhance processing and storage, *Communication layer* to facilitate data transmission,

and *Multimedia and Big Data Computation layer* for high performance processing. With real-time uncertainty, there might be many unknowns that might cause the failure; thus we assign it *Mitigated*.

Candidate two architecture similarly, has various mitigations that can support the operation of the IoV in the context of real-time processing uncertainty. These include *Processing layer* which has various cloud computing services and *Communication layer* which supports various robust data transmission technologies. Thus, *mitigating* against real-time operational uncertainties.

11. The *IoV security* uncertainties, relate to the various security threats to which the IoV and its individual elements might be exposed or threatened. Realistically, it is almost impossible to *Strongly mitigate* against general security, as threats arise from many sources; however, there are always specific measures that a system can take to mitigate the risks of security uncertainty.

Candidate One architecture, does not, specify a specific layer to handle security, however, the design implies security consideration within most of its layers from their design and discussion thus its *Partially mitigated*.

Candidate two architecture does explicitly includes a *security layer* that is cross-cutting among the other layers. The *Security layer* spans the rest of the layers and includes features dealing with data authentication, integrity, non-repudiation and confidentiality, access control, availability, etc. Thus, we consider the risks of security *Mitigated*.

12. The *IoV mapping localisation* uncertainty is related to a possible application of IoV to map and localise vehicles for various conveniences, such as finding the nearest services or managing traffic congestion. As mapping and localisation are key services to the successful operation of an IoV, architectures must be designed to support such a service and mitigate its uncertainty in terms of accuracy and availability, among other factors.

Candidate One architecture uses existing services such as GPS to manage mapping and localisation. Furthermore, it suggests that local sensors might be used to provide information on mapping and localisation within a smart city environment. The uncertainty of mapping localisation is *Mitigated* with such entities.

Candidate two architecture uses GPS services with *Data acquisition layer* to handle localisation and mapping, thus relying on the robustness of GPS to *Mitigate* risks of *IoV mapping localisation* uncertainty.

Thus, so far, we have individually assessed each candidate architecture and assigned it a degree

to which it mitigates against the individual risks of the twelve uncertainties. The next step is to analyse these data so that we can compare the two candidates. We conduct this analysis in Section 11.3.5. To achieve this, we define algorithms to guide us in the analysis.

11.3.5 Ranking candidate architecture: comparison

Table 11.2 shows the ordinal data we have generated from the uncertainty details of the two candidate architectures. In this section, we use the data as ground data to perform an uncertainty comparison and ranking of the two candidates. The results of the analysis aim to determine which, if any, of the two candidate architectures is likely to be less vulnerable to uncertainty. To conduct these comparisons and rankings, we propose three algorithms to produce the ranking or comparison of the candidate architectures.

Before we proceed, let us first look at an overview of the process from the start to the end so that we get a clear picture of the current stage, which is Step 7 in the procedure introduced in Section 11.1.

11.3.5.1 Ranking and comparison: algorithm one - weighing algorithm

The weighing algorithm works by ranking the candidate architecture based on their weights *candidate score* as captured in Table 11.3. The higher *candidate score*, the better the candidate architecture is ranked. Candidate architectures are equivalent if they mitigate similar uncertainties with equal scores as indicated by the ordinal score of the mitigation. To perform this evaluation, the following process is described:

Initially, the data are in the format indicated in Table 11.2. To transform the data for analysis, we encode them using numerical values. For example, *Influence measures (Very High, High, Medium, Low)* is encoded in the numerical *Influence values (4, 3, 2, 1)*, in descending order, respectively. Equally, *Candidate measures (Strongly mitigated, Mitigated, Partially mitigated, Unmitigated)* are encoded in *Candidate values (4, 3, 2, 1)*, in descending order, respectively. Table 11.3 shows the encoded data. Having the encoding is relevant for the algorithm, as will be illustrated in the following paragraphs.

After encoding, the next step is to generate what we call *Uncertainty weighing digits*. This is a placeholder for an architecture comparison or ranking number. We generate *Uncertainty weighing digits* by sorting the uncertainty according to *Influence score*.

If we consider the data in Table 11.3 and sort the uncertainties according to column number 3, (*Influence score*), the uncertainty IDs will be ordered according to Table 11.4. This sorting is in ascending order as illustrated in Table 11.4, below.

Table 11.3: Uncertainties, influence, and candidate architectures with encoded values

ID	Uncertainty description	Influence score	Candidate One score	Candidate Two score
1	IoV network bandwidth	3	2	3
2	IoV computation capability	2	3	3
3	IoV storage capacity	3	3	3
4	IoV network failure	2	3	4
5	IoV coordination failure	2	3	2
6	IoV vehicle, data, and other components growth	3	4	4
7	IoV future services	2	4	3
8	IoV protocols variations and heterogeneous	2	3	3
9	IoV range of communi- cation or interconnection channels	2	4	3
10	IoV real-time operation	4	3	3
11	IoV security	3	2	3
12	IoV mapping localisation	2	3	3

Table 11.4: Uncertainty weighing digits

Uncertainty ID	10	1	3	6	11	2	4	5	7	8	9	12
Influence score	4	3	3	3	3	2	2	2	2	2	2	2
Uncertainty weighing digits	—	—	—	—	—	—	—	—	—	—	—	—

As stated above, the uncertainties are sorted by *Influence score* in ascending order, as shown in Table 11.4. The effect of sorting is to group the uncertainties based on *Influence score*, from highest *Influence score* to lowest *Influence score*. As indicated in Table 11.4, the uncertainty with ID 10 has an *Influence score* of 4 while the uncertainty with ID 12 has an *Influence score* of 2. The rest of the uncertainties are as mapped in Table 11.3.

Important note: Through sorting the individual uncertainties basing on the *Influence score*, in an ascending order, implicitly, the *uncertainty weight digits* generated implies that the higher its value, the more significant the uncertainties. This is the motivation to sort and define *uncertainty weight digits*.

We then use the pattern of *uncertainty weight digits* to generate the weighting score for the individual candidate architecture.

To generate a candidate *uncertainty weight digits*, we simply map *Candidate score* for each *Candidate architecture* to the respective *uncertainty weight digits* placeholder for the relevant

Table 11.5: Candidate architecture and uncertainty weighing digits

Uncertainty ID	10	1	3	6	11	2	4	5	7	8	9	12
Influence score	4	3	3	3	3	2	2	2	2	2	2	2
Candidate one weighing digits	3	2	3	4	2	3	3	3	4	3	4	3
Candidate two weighing digits	3	3	3	4	3	3	4	2	3	3	3	3

uncertainty ID. Table 11.5 shows the two candidate architectures and their associated *uncertainty weighing digits*. We then compare these two digits to find which candidate has a better score.

Candidate one weighing digits of (323423334343) is less than *Candidate two weighing digits* of (333433423333), which suggests that the candidate two architecture is likely to be less vulnerable to known uncertainties than candidate one architecture.

However, this value is vulnerable, since, in essence, it is only considering the comparison between the highest influence uncertainties digits. The moment there is a difference, the rest of *Candidate weighing digits* is ignored after that. This means that we lose all the information without further insight.

Therefore, the analysis algorithm for *uncertainty weight digits* requires improvement on this weakness. For example, instead of using individual uncertainty digits, we can consider calculating the mean of the uncertainties with similar *Influence score*.

In fact, in the next section, we adapt and improve this algorithm by calculating the mean of the values within *uncertainty weight digits* with a similar *Influence score*. In this way, we can compare categories and determine in which aspects a candidate architecture better mitigates uncertainties than the other. Thus, we proceed to algorithm two.

11.3.5.2 Ranking and comparison: algorithm two - enhanced weighing with mean/average

Table 11.6: Candidate architecture and uncertainty weighing digits

Uncertainty ID	10	1	3	6	11	2	4	5	7	8	9	12
Influence score	4	3	3	3	3	2	2	2	2	2	2	2
Candidate one weighing digits	3	2	3	4	2	3	3	3	4	3	4	3
Candidate two weighing digits	3	3	3	4	3	3	4	2	3	3	3	3
Candidate one weighing mean	3	2.75				3.29						
Candidate two weighing mean	3	3.25				3.00						

Continuing with the analysis to improve Algorithm one, we proceed as follows.

We group uncertainties similar *Influence scores* into a single category. This is rational since *Influence score* indicates that such uncertainties have a potential similar impact on the system. In Table 11.5 we have three distinct influence scores among which the twelve uncertainties belong: 4, 3, 2. In Table 11.6 we have introduced columns to indicate these groups or categories.

The next step is to calculate the average / mean values for each of these uncertainties groups to generate a *weighted mean score*.

After calculating the *Candidate weighing mean* values for each of the two candidate architectures as shown in Table 11.6, the result now changes to candidate one having the following mean values (3, 2.75, 3.29) and candidate two having mean values (3, 3.25, 3.00).

Indeed, this calculation is more robust, as it shows that, in terms of mitigating the highest influence uncertainties, the two candidate architectures are similar, both with a score of 3.

However, when it comes to mitigating the *High* uncertainties and the *Medium* uncertainties, *Candidate architecture two*(3.25) is better than *Candidate architecture one*(2.75) at the *High*. Although *Candidate architecture one*(3.29) is better at handling *Medium* than *Candidate architecture two*(3.00).

Currently, we do not consider the significance of the number or differences. In addition, of course, these results depend on several factors, including the information available during the analysis, the subjective technical expertise of the analysis, and the validity of the uncertainties.

Next, we finally look at the third and last algorithms that optimise the analysis.

11.3.5.3 Ranking and comparison: algorithm three - elimination then weigh algorithm

In this algorithm, we propose an approach that improves on previous algorithms by first eliminating or removing uncertainties with same *Candidate score* and then assessing the architectures based on uncertainties with different *Candidate scores*.

For example, if we consider in Table 11.3, the uncertainty with ID 3, since this uncertainty has similar scores for both candidate architectures, we eliminate or remove this uncertainty from consideration. The results after removing same uncertainties are shown in Table 11.7.

In addition to improving efficiency in the calculation of the weight score, by removing uncertainties with similar scores, we highlight the uncertainties that can make a difference for further analysis. These uncertainties may be explored further to understand the different approaches to their mitigations in the different candidate architecture.

For example, with the help of Table 11.7, the architect could try to explore the underlying

Table 11.7: Uncertainties with similar influence scores eliminated

ID	Uncertainty description	Influence score	Candidate One score	Candidate Two score
1	IoV network bandwidth	3	2	3
4	IoV network failure	2	3	4
5	IoV coordination failure	2	3	2
7	IoV future services	2	4	3
9	IoV range of communication or interconnection channels	2	4	3
11	IoV security	3	2	3

differences between the candidate architectures to understand why uncertainties are handled differently. This information could be used to improve either candidate architecture, if possible, from the observation of the other candidate architecture.

Table 11.8: Candidate architecture and uncertainty weighing digits with similarities eliminated

Uncertainty ID	1	11	4	5	7	9
Influence score	3	3	2	2	2	2
Candidate one weighing digits	2	2	3	3	4	4
Candidate two weighing digits	3	3	4	2	3	3
Candidate one weighing mean	2		3.5			
Candidate two weighing mean	3		3.00			

Next, we analyse the uncertainty data in Table 11.7, using the weighted mean approach as presented in the previous algorithm. Table 11.8, shows the results of the analysis from which we can make the following ranking observations:

1. The candidate two architectures are better at handling uncertainty with the *High influence score* (3) compared to the candidate one architecture.
2. Candidate one architecture is better at handling the uncertainty with *Medium influence score* (2) compared to candidate two architecture.
3. Both candidate architectures have a similar capacity to handle *Very high* uncertainties.

11.4 Discussion

This evaluation has been an interesting exercise to illustrate another potential application of the uncertainty framework. We aimed to compare two candidate architectures from the IoV domain.

In terms of the two candidate architectures, in general, we can evaluate that *Candidate architecture two* is likely to be better at handling uncertainties than *Candidate architecture one*. *Algorithm one* indicated that *Candidate architecture two* was the preferable candidate based on the *Candidate weighting digits* score. Also, *Algorithm two* indicated that the two candidates had similar abilities to handle *Very high influence* but *Candidate architecture two* was better with *High influence score* uncertainties. And finally, *Candidate architecture two* was better with *High influence score* uncertainties under *Algorithm three*. Thus, heuristically, our analysis suggests that *Candidate architecture two* is preferable to manage uncertainty than *Candidate architecture one*.

Of course, this is not the final decision because, as we have seen in Table 11.7, there may be other factors to consider, such as the number of uncertainties available in each category that can change and impact the results. Ideally, considering that uncertainties are dynamic, they change with time and new uncertainties might be discovered, existing uncertainties might become certainties, and it might make sense to consider additional sources of information before making the final selection.

As such, even though the uncertainty data provide enough information to provide uncertainty analysis for the candidate architectures, it might be best to use it in conjunction with other analysis approaches to make the final decisions. Besides, similar to most architecture analysis results, this is a heuristic decision which might not always be correct, depending on the input of the process. As always, it is important to keep in mind, during any architecture analysis process, that the analysis is *a garbage in garbage out process*. Otherwise, this case study has been successful from the perspective of illustrating a potential use case for the framework data.

11.5 Conclusions

In this case study, we illustrate the use of the uncertainty framework to compare and rank candidate architectures from the IoV domain. Our approach involved first identifying common uncertainties of the IoVs, which we used as a basis to compare the candidate architectures. Based on the common uncertainties, we analysed the individual candidate architectures to assess how well they mitigate the common uncertainty, which we recorded using ordinal values. By encoding both the common uncertainties and their mitigation results for each candidate architecture, we generated ground data, which we analysed to rank the candidate architectures. The ranking was achieved through three related but improved versions of the weight algorithms. Overall, we found that it was possible to analyse candidate architectures based on uncertainty data; however, further details from another analysis approach might be required to provide the final results.

However, still, the uncertainty analysis provides useful information which can inform the overall candidate architecture ranking and comparisons process.

12

CHAPTER TWELVE

CONCLUSIONS

In this chapter we present the conclusions of the work, including an overview of the insights, a summary of the main contributions, critical reflections and future work.

12.1 Overview

Overall, this research has generated interesting insights and observations on uncertainty in software systems in general and in software architecture in particular. As part of this research, the following work was undertaken: literature surveys of uncertainty in software systems and in software architecture, the definition of a framework for considering uncertainty in software systems and its illustration in the context of software architecture, the design and development of a workbench infrastructure to facilitate future addition of tools for incorporating uncertainty in software architecture, the implementation of a proof-of-concept of tool based on the framework to represent uncertainty in software architecture specifications, the evaluation of the uncertainty framework through applying it to three architecture case studies and a consideration of potential future work.

12.2 Summary of main contributions

The following five elements are the key contributions of the work:

12.2.1 A literature survey of uncertainty in software architecture from 1991 to 2021

This chapter reviewed literature sources from 1991 to 2021 to explore uncertainty in software architecture aspects, including architecture *Definitions, Concepts, Activities, Artefacts, Tools*

and Notations. The survey resulted in 82 literature sources, which highlighted 25 specific software architecture aspects where uncertainty is explicitly considered and provided a potential categorisation of these aspects. Each of these aspects has received varying degrees of focus in terms of research interest and results. The survey identified existing approaches for addressing uncertainty in these aspects of software architecture as well as gaps in the state-of-the-art. There is a need for further research to improve the consideration, representation and management of uncertainty in all aspects of software architecture.

12.2.2 A literature survey of uncertainty frameworks for software system

Often, uncertainty is treated using mathematical approaches. However, such approaches do not capture or highlight other details about the uncertainty, such as its location, nature and more. The details about the uncertainty can be captured or characterised using the various attributes of uncertainty. In this contribution, we explored existing work which explicitly defines uncertainty characterisation attributes in software systems. These include sources from complex systems, self-adaptive system, cognitive science and cyber-physical systems. We used these concepts and notions of uncertainty to define the uncertainty consideration framework.

12.2.3 A definition of a conceptual framework for considering uncertainty in software systems

The definition of a framework for considering uncertainty in software architecture was the inspiration for this research. In this regard, we defined a generic framework for considering uncertainty in software systems and applied it to software architecture for evaluation. The uncertainty consideration framework is defined through consolidating attributes from existing works, including self-adaptive systems, complex systems, cognitive systems and cyber-physical systems. The uncertainty framework can be used in various contexts, including architecture analysis or to enhance tools and notations with the capabilities to incorporate uncertainty.

12.2.4 The specification and development of the workbench infrastructure concept

Based on the uncertainty framework, we explored the concept of the workbench infrastructure: specifically, we explored the realisation of the uncertainty framework through the workbench infrastructure, which defines the foundation for building an architecture workbench of tools incorporating uncertainty in software architecture. The implementation of tools based on the workbench infrastructure contributes to the proposed future work. The realisation of tools and

notations that support uncertainty in software architecture and software systems in general is one of the key points which this work advocates.

12.2.5 Evaluation of the uncertainty consideration framework on software architecture case studies

Finally, we evaluated the uncertainty consideration framework on three case studies. The first case study illustrates a potential application of the uncertainty framework to generate architecture uncertainty knowledge and documentation. The second case study uses the uncertainty data for architecture analysis. To achieve this, the case study first generates uncertainty data about the NASA IMPALA platform architecture. Then the evaluation focuses on analysing the architecture design based on the uncertainty details. The validity of the analysis is compared with a similar analysis using a customised version of the ATAM. Finally, the third case study applies the uncertainty framework to the problem of comparing and ranking candidate architectures of a system. This case study compares two candidate architectures from the domain of IoV using three algorithms. The results of the evaluations suggest that the framework can provide useful information to support evaluation of candidate architectures.

12.2.6 Critical reflection on the uncertainty consideration framework

In this section, we critically review the uncertainty consideration framework. First, we consider the significance of our work.

This work has develop a framework for considering uncertainty in software architecture. This is significant work since we introduce a novel approach for considering uncertainty in software systems and software architecture. The motivation of the research was to develop such a framework; therefore, we consider that we have achieved the objective of our work.

The evaluation of the framework was performed on three case studies, and the results demonstrated that the framework is a feasible approach to considering uncertainty in software architecture. In addition, considering the variation in the case studies, the evaluation has demonstrated that the framework has potential for use in a variety of use cases.

However, there is still a lot of work to be done. Currently, the framework is generic, with little constraint on its application. But in some contexts, the framework might require to be customised. For instance, to manage data quality, in an automated context, the fields of the attributes of the framework can be restricted to accept a specific data format or structure text. Other enhancement might relate to enforcing relationships and dependencies among attributes

and groups of uncertainties. But for the purpose of this thesis the framework achieves its initial design motivation.

In terms of the evaluation case studies, all three case studies generate knowledge and documentation about the architecture. But only the first case study was specifically designed for this purpose. The other two case studies demonstrate the uncertainty management potential of the framework. First, IMPALA platform architecture analysis demonstrated that framework uncertainty data can be analysed to generate insight into the software architecture. In addition, the IoV candidate selection case studies demonstrated that the uncertainty data can be processed using algorithms to contribute to software architecture decision problems. The results in all these cases studies, were positive.

The major next phase of this work is to evaluate the framework with practitioners so that the framework can be evaluated in practice. However, this will require tools and notations to support the application of the framework. In this regard, we have proposed the workbench infrastructure, which defines a basis for building a workbench of tools to consider uncertainty in software architecture. Overall, as a proof of concept, the work has been successful and generated interesting results.

12.2.7 Final thoughts on the main contributions

The central hypothesis of this work is that a generic and useful conceptual framework for considering uncertainty in software systems can be defined. We have observed that uncertainty is characterised with distinct attributes in different contexts, potentially omitting useful details. We have consolidated and extended attributes from various software systems domains to propose a generic, foundational, and customisable framework for uncertainty in the broader context of software systems. The use of the framework is evaluation using three studies from three different application domains. In addition, a concept of workbench infrastructure is developed on the basis of the framework. Together, the contributes address the two research questions - first, the definition of a generic uncertainty framework for software systems. Secondly, the application and evaluation of the framework in the contexts of software architecture.

12.3 Future work

In terms of future work, throughout the thesis, we have highlighted potential future work directions for the consideration of uncertainty in software architecture and systems. In this section, we briefly present some specific future works.

12.3.1 Continuous uncertainty management

Uncertainty management should be a continuous process. With time, more uncertainties may be discovered; therefore, the number of uncertainties can increase. Similarly, some of the uncertainties can be reduced so that they become certainties, thus decreasing the number of uncertainties. Therefore, software architecture tools must be developed to actively manage uncertainty in architecture and systems. The framework presents a platform for building such tools.

12.3.2 Uncertainty and Agile software development

Another area we envision future work in the relationship between the uncertainty framework and Agile software development. In an Agile process, where a repository of uncertainties could be maintained and continuously updated to track the uncertainty status of the system, the uncertainty framework can be used to create a tracker to monitor such aspects like uncertainty mitigation implementations and risk statuses.

This can build on the extensibility of the uncertainty framework with attributes such as a *Status* attribute for tracking whether an uncertainty is still *open* or *closed* can, for instance, be used for such a purpose. Additionally, *Timestamp* attributes could be added to support monitoring of changes in uncertainty status.

However, the framework is not limited to specific development methodologies, system artefacts, notations, or levels of abstraction. It is intended to be applied to different artefacts and stages of the software lifecycle.

12.3.3 Uncertainty framework automation

However, manually compiling all uncertainties associated with any non-trivial system would be a time-consuming activity, in addition to being error-prone. Since the framework provides a structured and systematic basis for capturing uncertainty data, automation may be a feasible alternative to applying the framework on a larger scale. With automation, rules for associating uncertainties with specific patterns of structure and interactions in the system can be specified and used to generate an initial set of uncertainty data that can be manually augmented by stakeholders.

12.3.4 Uncertainty framework customisation

While we have provided a definition of each attribute in the framework, we have deliberately avoided stipulating any processes or guidelines to populate these attributes. Therefore, the

granularity of the data could depend on the context of the specific system, the stage of the lifecycle at which uncertainty is considered, and the aims of the users of the framework.

It is likely that values of attributes with well-defined options such as that of nature, which can be either aleatory or epistemic, will be uniform across systems. However, details such as description, dependencies, and outcomes might vary from the examples provided and across different systems.

12.3.5 Final thought on future work

The thesis has empirically demonstrated the feasibility of defining a generic and extensible framework from individual sources in existing literature. The framework is not designed to be exhaustive or mandated on specific methodologies, but can be customised to specific use cases. A key part of future work will require that user evaluation studies be conducted with the framework to evaluate its use in practice. This also includes understanding the use cases that can be generated for the use of the framework data.

12.4 Conclusions

The work presented in this thesis is the beginning of what we believe to be an exciting and important avenue of future research. As the framework is applied to different facets of software systems, its effectiveness and viability will be continuously evaluated and refined. As far as we are aware, no work currently exists that expresses uncertainty attributes in a generalised format for software systems and architecture.

Uncertainty is a common concept in many contexts. In casual discussions, we can interpret uncertainty in an ad-hoc manner from the context. However, in more formal or consequential circumstances, such as during the life of critical software systems, a more precise specification of uncertainty is likely to be required. We have defined a conceptual generic customisable framework to considering uncertainty as a first-class concern in software systems. Several attributes are included to capture the details of uncertainties, which can be extended as necessary.

REFERENCES

- [1] M. Smithson, *Ignorance and uncertainty: emerging paradigms*. Springer Science & Business Media, 1989.
- [2] H. McManus and D. Hastings, “3.4. 1 A Framework for Understanding Uncertainty and its Mitigation and Exploitation in Complex Systems,” in *INCOSE international symposium*, vol. 15, pp. 484–503, Wiley Online Library, 2005.
- [3] M. Zhang, B. Selic, S. Ali, T. Yue, O. Okariz, and R. Norgren, “Understanding uncertainty in cyber-physical systems: a conceptual model,” in *European conference on modelling foundations and applications*, pp. 247–264, Springer, 2016.
- [4] G. A. Moreno, J. Cámara, D. Garlan, and M. Klein, “Uncertainty reduction in self-adaptive systems,” in *2018 IEEE/ACM 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pp. 51–57, IEEE, 2018.
- [5] J. L. Sanchez-Lopez, R. A. S. Fernández, H. Bavle, C. Sampedro, M. Molina, J. Pestana, and P. Campoy, “Aerostack: An architecture and open-source software framework for aerial robotics,” in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 332–341, IEEE, 2016.
- [6] L.-M. Ang, K. P. Seng, G. K. Ijamaru, and A. M. Zungeru, “Deployment of iov for smart cities: Applications, architecture, and challenges,” *IEEE access*, vol. 7, pp. 6473–6492, 2018.
- [7] J. Contreras-Castillo, S. Zeadally, and J. A. Guerrero-Ibañez, “Internet of vehicles: architecture, protocols, and security,” *IEEE internet of things Journal*, vol. 5, no. 5, pp. 3701–3709, 2017.
- [8] M. M. Lehman, “Programs, Life Cycles, and Laws of Software Evolution,” *Proceedings of the IEEE*, vol. 68, no. 9, 1980.

- [9] O. L. de Weck, C. Eckert, and J. Clarkson, “A classification of uncertainty for early product and system design,” in *International Conference on Engineering Design*, Massachusetts Institute of Technology. Engineering Systems Division, 2007.
- [10] E. Letier, D. Stefan, and E. T. Barr, “Uncertainty, risk, and information value in software requirements and architecture,” in *Proceedings of the 36th International Conference on Software Engineering*, pp. 883–894, ACM, 2014.
- [11] D. E. Perry and A. L. Wolf, “Foundations for the study of software architecture,” *Computer*, vol. 17, no. 5, pp. 40–52, 1992.
- [12] C. Lupafya, “A framework for managing uncertainty in software architecture,” in *Proceedings of the 13th European Conference on Software Architecture-Volume 2*, pp. 71–74, ACM, 2019.
- [13] P. A. Laplante, “The Heisenberg uncertainty principle and the halting problem,” *ACM SIGACT News*, vol. 22, no. 3, pp. 63–65, 1991.
- [14] A. J. Ramirez, A. C. Jensen, and B. H. C. Cheng, “A taxonomy of uncertainty for dynamically adaptive systems,” in *Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pp. 99–108, IEEE Press, 2012.
- [15] W. E. Walker, P. Harremoës, J. Rotmans, J. P. Van Der Sluijs, M. B. A. Van Asselt, P. Janssen, and M. P. von Krauss, “Defining uncertainty: a conceptual basis for uncertainty management in model-based decision support,” *Integrated assessment*, vol. 4, no. 1, pp. 5–17, 2003.
- [16] Y. Hichri, S. Dahi, and H. Fathallah, “Candidate architectures for emerging iov: a survey and comparative study,” *Design Automation for Embedded Systems*, vol. 25, no. 4, pp. 237–263, 2021.
- [17] Y. S. Chan, “A visual representation of uncertainty in software systems (unpublished master’s thesis),” msc dissertation, School of Computer Science, University of St Andrews, School of Computer Science, University of St Andrews, Jack Cole Building, North Haugh, St Andrews , KY16 9SX, aug 2022.
- [18] M. M. Lehman and J. F. Ramil, “Software uncertainty,” in *Soft-Ware 2002: Computing in an Imperfect World* (D. Bustard, W. Liu, and R. Sterritt, eds.), (Berlin, Heidelberg), pp. 174–190, Springer Berlin Heidelberg, 2002.

- [19] D. Garlan, “Software engineering in an uncertain world,” *Proceedings of the FSE/SDP workshop on Future of software engineering research SE - FoSER '10*, 2010.
- [20] H. Ziv, D. Richardson, and R. Klösch, “The uncertainty principle in software engineering,” in *Proceedings of the 19th International Conference on Software Engineering (ICSE'97)*, 1997.
- [21] S. Mahdavi-Hezavehi, P. Avgeriou, and D. Weyns, “A classification framework of uncertainty in architecture-based self-adaptive systems with multiple quality requirements,” in *Managing Trade-Offs in Adaptable Software Architectures*, pp. 45–77, Elsevier, 2017.
- [22] D. Perez-Palacin and R. Mirandola, “Uncertainties in the modeling of self-adaptive systems: a taxonomy and an example of availability evaluation,” in *Proceedings of the 5th ACM/SPEC international conference on Performance engineering*, pp. 3–14, ACM, 2014.
- [23] J. B. Kadane, *Principles of uncertainty*. CRC Press, 2011.
- [24] M. Autili, V. Cortellessa, D. Di Ruscio, P. Inverardi, P. Pelliccione, and M. Tivoli, “Eagle: Engineering software in the ubiquitous globe by leveraging uncertainty,” in *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pp. 488–491, ACM, 2011.
- [25] J. Whittle, P. Sawyer, N. Bencomo, B. H. Chengy, J. M. Bruelz, B. H. C. Cheng, and J.-M. Bruel, “RELAX: Incorporating uncertainty into the specification of self-adaptive systems,” *Proceedings of the IEEE International Conference on Requirements Engineering*, pp. 79–88, 2009.
- [26] N. Esfahani, K. Razavi, and S. Malek, “Dealing with uncertainty in early software architecture,” in *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, p. 21, ACM, 2012.
- [27] N. Esfahani, S. Malek, and K. Razavi, “Guidearch: guiding the exploration of architectural solution space under uncertainty,” in *Software Engineering (ICSE), 2013 35th International Conference on*, pp. 43–52, IEEE, 2013.
- [28] A. L. Wolf, “Succeedings of the Second International Software Architecture Workshop,” *SIGSOFT Softw. Eng. Notes*, vol. 22, no. 1, pp. 42–56, 1997.
- [29] D. E. Perry and A. L. Wolf, “Foundations for the study of software architecture,” *ACM SIGSOFT Software Engineering Notes*, vol. 17, no. 4, pp. 40–52, 1992.

- [30] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*. Addison-Wesley, 2 ed ed., 2003.
- [31] R. N. Taylor, N. Medvidovic, and E. M. Dashofy, *Software Architecture: Foundations, Theory, and Practice*. Wiley Publishing, 2009.
- [32] C. Dhaya and G. Zayaraz, “Combined architectural framework for the selection of architectures using ATAM, FAHP and CBAM,” *International Journal of Computer Applications in Technology*, vol. 54, no. 4, pp. 350–361, 2016.
- [33] C. Trubiani, I. Meedeniya, V. Cortellessa, A. Aleti, and L. Grunske, “Model-based performance analysis of software architectures under uncertainty,” in *Proceedings of the 9th international ACM Sigsoft conference on Quality of software architectures*, pp. 69–78, ACM, 2013.
- [34] A. Sedaghatbaf and M. A. Azgomi, “Reliability evaluation of UML/DAM software architectures under parameter uncertainty,” *IET Software*, vol. 12, no. 3, 2018.
- [35] C. Miksovic and O. Zimmermann, “Architecturally significant requirements, reference architecture, and metamodel for knowledge management in information technology services,” *Proceedings - 9th Working IEEE/IFIP Conference on Software Architecture, WICSA 2011*, pp. 270–279, 2011.
- [36] A.-j. Stoica, “Facets of the Software Development Represented by Model Systems : Analysis and Enhancement,” *14th International Forum on COCOMO/SCM*, pp. 1–25, 1999.
- [37] P. Kruchten, “Architectural blueprints—the” 4+ 1” view model of software architecture,” *IEEE Software*, vol. 12, no. November, pp. 42–50, 1995.
- [38] I. Gorton and J. Haack, “Architecting in the face of uncertainty: An experience report,” in *Proceedings - International Conference on Software Engineering*, vol. 26, 2004.
- [39] R. Raman and M. D’Souza, “Knowledge based decision framework for architecting complex systems,” in *Proceedings of the ACM Symposium on Applied Computing*, vol. Part F1280, pp. 1147–1153, 2017.
- [40] D. Sobhy, R. Bahsoon, L. Minku, and R. Kazman, “Evaluation of software architectures under uncertainty: a systematic literature review,” *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 30, no. 4, pp. 1–50, 2021.

- [41] S. M. Hezavehi, D. Weyns, P. Avgeriou, R. Calinescu, R. Mirandola, and D. Perez-Palacin, "Uncertainty in self-adaptive systems: A research community perspective," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 15, no. 4, pp. 1–36, 2021.
- [42] I. Lytra, C. Carrillo, R. Capilla, and U. Zdun, "Quality attributes use in architecture design decision methods: research and practice," *Computing*, vol. 102, no. 2, 2020.
- [43] V. Kumar, L. Singh, and A. K. Tripathi, "Reliability analysis of safety-critical and control systems: A state-of-the-art review," 2018.
- [44] J. Kramer and J. Magee, "Analysing dynamic change in distributed software architectures1," *IEE Proceedings: Software*, vol. 145, no. 5, 1998.
- [45] P. Oreizy and R. N. Taylor, "On the role of software architectures in runtime system reconfiguration," *IEE Proceedings: Software*, vol. 145, no. 5, 1998.
- [46] L. Williams and C. Smith, "Performance evaluation of software architectures," 1998.
- [47] R. Bahsoon and W. Emmerich, "ArchOptions: A real options-based model for predicting the stability of software architectures," in *Proceedings of the 5th Workshop on Economics-Driven Software Research (EDSER-5)*, (Portland, USA), pp. 35–40, 2003.
- [48] R. L. Nord, M. R. Barbacci, P. Clements, R. Kazman, and M. Klein, "Integrating the Architecture Tradeoff Analysis Method (ATAM) with the cost benefit analysis method (CBAM)," tech. rep., Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 2003.
- [49] C. Hofmeister, P. Kruchten, R. L. Nord, H. Obbink, A. Ran, and P. America, "A general model of software architecture design derived from five industrial approaches," *Journal of Systems and Software*, vol. 80, no. 1, pp. 106–126, 2007.
- [50] R. C. Nickerson, U. Varshney, and J. Muntermann, "A method for taxonomy development and its application in information systems," *European Journal of Information Systems*, vol. 22, no. 3, pp. 336–359, 2013.
- [51] V. Grassi and R. Mirandola, "The tao way to anti-fragile software architectures: the case of mobile applications," in *2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C)*, pp. 86–89, IEEE, 2021.
- [52] F. Niederman, J. C. Brancheau, and J. C. Wetherbe, "Information Systems Management Issues for the 1990s," vol. 15, no. 4, pp. 475–500, 1991.

- [53] J. Axelsson, *On how to deal with uncertainty when architecting embedded software and systems*, vol. 6903 LNCS. 2011.
- [54] F. Oquendo, “Dealing with Uncertainty in Software Architecture on the Internet-of-Things with Digital Twins,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11619 LNCS, 2019.
- [55] F. Oquendo, “Fuzzy Architecture Description for Handling Uncertainty in IoT Systems-of-Systems,” in *SOSE 2020 - IEEE 15th International Conference of System of Systems Engineering, Proceedings*, 2020.
- [56] F. Oquendo, “Case study on the fuzzy architecture description of cyber-physical sos under uncertainty,” in *2021 16th International Conference of System of Systems Engineering (SoSE)*, pp. 61–68, IEEE, 2021.
- [57] F. Oquendo, “Formally describing the software architecture of Systems-of-Systems with SosADL,” in *2016 11th Systems of Systems Engineering Conference, SoSE 2016*, 2016.
- [58] M. Guessi, F. Oquendo, and E. Y. Nakagawa, “Ark: a constraint-based method for architectural synthesis of smart systems,” *Software and Systems Modeling*, vol. 19, no. 3, 2020.
- [59] I. Lytra and U. Zdun, “Supporting architectural decision making for systems-of-systems design under uncertainty,” in *1st ACM SIGSOFT/SIGPLAN International Workshop on Software Engineering for Systems-of-Systems, SESoS 2013 Proceedings*, 2013.
- [60] F. Alkhabbas, I. Murturi, R. Spalazzese, P. Davidsson, and S. Dustdar, “A goal-driven approach for deploying self-adaptive IoT systems,” in *Proceedings - IEEE 17th International Conference on Software Architecture, ICSA 2020*, pp. 146–156, 2020.
- [61] K. Shumaiev and M. Bhat, “Automatic uncertainty detection in software architecture documentation,” in *Proceedings - 2017 IEEE International Conference on Software Architecture Workshops, ICSAW 2017: Side Track Proceedings*, pp. 216–219, 2017.
- [62] M. Waterman, “Agility, risk, and uncertainty, part 2: How risk impacts agile architecture,” *IEEE Software*, vol. 35, no. 3, 2018.
- [63] R. Pincioli and C. Trubiani, “Model-based performance analysis for architecting cyber-physical dynamic spaces,” in *2021 IEEE 18th International Conference on Software Architecture (ICSA)*, pp. 104–114, IEEE, 2021.

- [64] C. Trubiani, A. Ghabi, and A. Egyed, “Exploiting traceability uncertainty between software architectural models and performance analysis results,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9278, 2015.
- [65] L. Cheung, L. Golubchik, N. Medvidovic, and G. Sukhatme, “Identifying and addressing uncertainty in architecture-level software reliability modeling,” in *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pp. 1–6, IEEE, 2007.
- [66] L. Cheung, R. Roshandel, N. Medvidovic, and L. Golubchik, “Early prediction of software component reliability,” in *Proceedings - International Conference on Software Engineering*, pp. 111–120, 2008.
- [67] I. Meedeniya, I. Moser, A. Aleti, and L. Grunske, “Architecture-based reliability evaluation under uncertainty,” in *CompArch’11 - Proceedings of the 2011 Federated Events on Component-Based Software Engineering and Software Architecture - QoSA+ISARCS’11*, 2011.
- [68] L. Fiondella and S. S. Gokhale, “Software reliability with architectural uncertainties,” in *IPDPS Miami 2008 - Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium, Program and CD-ROM*, 2008.
- [69] R. Bahsoon, W. Emmerich, and J. Macke, “Using real options to select stable middleware-induced software architectures,” *IEE Proceedings: Software*, vol. 152, no. 4, 2005.
- [70] W. Cui and T. Sherwood, “Estimating and understanding architectural risk,” in *Proceedings of the Annual International Symposium on Microarchitecture, MICRO*, vol. Part F1312, 2017.
- [71] K. Shumaiev, M. Bhat, O. Klymenko, F. Matthes, A. Biesdorf, and U. Hohenstein, “Uncertainty expressions in software architecture group decision making: Explorative study,” in *ACM International Conference Proceeding Series*, 2018.
- [72] D. Tofan, M. Galster, P. Avgeriou, and W. Schuitema, “Past and future of software architectural decisions - A systematic mapping study,” 2014.
- [73] S. A. Busari, “Towards search-based modelling and analysis of requirements and architecture decisions,” in *ASE 2017 - Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, 2017.
- [74] K. D. Evensen, “Reducing uncertainty in architectural decisions with AADL,” *Proceedings of the Annual Hawaii International Conference on System Sciences*, pp. 1–9, 2011.

- [75] A. El Malki and U. Zdun, “Guiding architectural decision making on service mesh based microservice architectures,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11681 LNCS, 2019.
- [76] J. Cámara, M. Silva, D. Garlan, and B. Schmerl, “Explaining architectural design tradeoff spaces: A machine learning approach,” in *European Conference on Software Architecture*, pp. 49–65, Springer, 2021.
- [77] M. Saadatmand and S. Tahvili, “A Fuzzy Decision Support Approach for Model-Based Tradeoff Analysis of Non-functional Requirements,” in *Proceedings - 12th International Conference on Information Technology: New Generations, ITNG 2015*, 2015.
- [78] J. Cámara, D. Garlan, and B. Schmerl, “Synthesizing tradeoff spaces with quantitative guarantees for families of software systems,” *Journal of Systems and Software*, vol. 152, 2019.
- [79] M. Salama and R. Bahsoon, “Analysing and modelling runtime architectural stability for self-adaptive software,” *Journal of Systems and Software*, vol. 133, 2017.
- [80] S. Hahner, “Dealing with uncertainty in architectural confidentiality analysis,” in *Software Engineering (Satellite Events)*, 2021.
- [81] D. Sobhy, L. Minku, R. Bahsoon, T. Chen, and R. Kazman, “Run-time evaluation of architectures: A case study of diversification in IoT,” *Journal of Systems and Software*, vol. 159, 2020.
- [82] K. Watanabe, N. Ubayashi, T. Fukamachi, S. Nakamura, H. Muraoka, and Y. Kamei, “IArch-U: Interface-Centric Integrated Uncertainty-Aware Development Environment,” in *Proceedings - 2017 IEEE/ACM 9th International Workshop on Modelling in Software Engineering, MiSE 2017*, 2017.
- [83] M. Famelis, N. Ben-David, A. Di Sandro, R. Salay, and M. Chechik, “MU-MMINT: An IDE for Model Uncertainty,” in *Proceedings - International Conference on Software Engineering*, vol. 2, 2015.
- [84] A. Serban, E. Poll, and J. Visser, “Towards using probabilistic models to design software systems with inherent uncertainty,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12292 LNCS, 2020.

- [85] M. Galster, A. Eberlein, and M. Moussavi, "Systematic selection of software architecture styles," *IET Software*, vol. 4, no. 5, p. 349, 2010.
- [86] S. Moaven and J. Habibi, "A fuzzy-AHP-based approach to select software architecture based on quality attributes (FASSA)," *Knowledge and Information Systems*, vol. 62, no. 12, 2020.
- [87] H. P. Breivold, I. Crnkovic, and M. Larsson, "A systematic review of software architecture evolution research," *Information and Software Technology*, vol. 54, no. 1, pp. 16–40, 2012.
- [88] H. P. Breivold, I. Crnkovic, R. Land, and S. Larsson, "Using dependency model to support software architecture evolution," *Automated Software Engineering - Workshops, 2008. ASE Workshops 2008. 23rd IEEE/ACM International Conference on*, pp. 82–91, 2008.
- [89] S. Bonfanti, E. Riccobene, and P. Scandurra, "A runtime safety enforcement approach by monitoring and adaptation," in *European Conference on Software Architecture*, pp. 20–36, Springer, 2021.
- [90] S.-W. Cheng and D. Garlan, "Handling uncertainty in autonomic systems," *Int'l Wrkshp. on Living with Uncertainty*, 2007.
- [91] S. W. Cheng and D. Garlan, "Stitch: A language for architecture-based self-adaptation," *Journal of Systems and Software*, vol. 85, no. 12, pp. 2860–2875, 2012.
- [92] H. Wang and Z. Zheng, "Self-adaptive method based on software architecture by inspecting uncertainty," in *Proceedings - International Conference on Artificial Intelligence and Computational Intelligence, AICI 2010*, vol. 3, 2010.
- [93] C. L. McGhan, R. M. Murray, R. Serra, M. D. Ingham, M. Ono, T. Estlin, and B. C. Williams, "A risk-aware architecture for resilient spacecraft operations," in *IEEE Aerospace Conference Proceedings*, vol. 2015-June, 2015.
- [94] D. Garlan, S.-W. Cheng, A.-C. Huang, B. Schmerl, and P. Steenkiste, "Rainbow: Architecture-based self-adaptation with reusable infrastructure," *Computer*, vol. 37, no. 10, pp. 46–54, 2004.
- [95] N. Esfahani, "A framework for managing uncertainty in self-adaptive software systems," in *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering*, pp. 646–650, IEEE Computer Society, 2011.

- [96] D. Han, Q. Yang, J. Xing, J. Li, and H. Wang, “FAME: A UML-based framework for modeling fuzzy self-adaptive software,” *Information and Software Technology*, vol. 76, pp. 118–134, 2016.
- [97] T. Bureš, P. Hnětynka, F. Plášil, D. Škoda, J. Kofroň, R. Al Ali, and I. Gerostathopoulos, “Targeting uncertainty in smart cps by confidence-based logic,” *Journal of Systems and Software*, vol. 181, p. 111065, 2021.
- [98] I. Gerostathopoulos, D. Skoda, F. Plasil, T. Bures, and A. Knauss, “Tuning self-adaptation in cyber-physical systems through architectural homeostasis,” *Journal of Systems and Software*, vol. 148, 2019.
- [99] N. Bencomo and L. H. Garcia Paucar, “RaM: Causally-Connected and Requirements-Aware Runtime Models using Bayesian Learning,” in *Proceedings - 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems, MODELS 2019*, 2019.
- [100] H. Muccini and K. Vaidhyanathan, “ArchLearner: Leveraging machine-learning techniques for proactive architectural adaptation,” in *ACM International Conference Proceeding Series*, vol. 2, 2019.
- [101] P. B. Kruchten, “The 4+ 1 view model of architecture,” *IEEE software*, vol. 12, no. 6, pp. 42–50, 1995.
- [102] D. Mishra, A. Mishra, and A. Yazici, “Successful requirement elicitation by combining requirement engineering techniques,” *1st International Conference on the Applications of Digital Information and Web Technologies, ICADIWT 2008*, pp. 258–263, 2008.
- [103] E. Letier, D. Stefan, and E. T. Barr, “Uncertainty, Risk, and Information Value in Software Requirements and Architecture,” *Proceedings of the 36th International Conference on Software Engineering*, pp. 883–894, 2014.
- [104] S. Aaramaa, S. Dasanayake, M. Oivo, J. Markkula, and S. Saukkonen, “Requirements volatility in software architecture design: An exploratory case study,” in *ACM International Conference Proceeding Series*, vol. Part F1287, pp. 40–49, 2017.
- [105] A. Ghabi and A. Egyed, “Exploiting traceability uncertainty between architectural models and code,” in *Proceedings of the 2012 Joint Working Conference on Software Architecture and 6th European Conference on Software Architecture, WICSA/ECSA 2012*, 2012.

- [106] A. Sedaghatbaf and M. A. Azgomi, "SQME: a framework for modeling and evaluation of software architecture quality attributes," *Software and Systems Modeling*, vol. 18, no. 4, 2019.
- [107] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, and A. Tang, "What industry needs from architectural languages: A survey," *IEEE Transactions on Software Engineering*, vol. 39, no. 6, pp. 869–891, 2013.
- [108] G. Fairbanks, "The risk-driven model," *CrossTalk*, p. 9, 2010.
- [109] D. Dubois and H. Prade, "Possibility theory," *Scholarpedia*, vol. 2, no. 10, p. 2074, 2007.
- [110] M. Zhang, S. Ali, T. Yue, R. Norgren, and O. Okariz, "Uncertainty-wise cyber-physical system test modeling," *Software & Systems Modeling*, vol. 18, no. 2, pp. 1379–1418, 2019.
- [111] A. Musil, J. Musil, D. Weyns, T. Bures, H. Muccini, and M. Sharaf, "Patterns for self-adaptation in cyber-physical systems," in *Multi-disciplinary engineering for cyber-physical production systems*, pp. 331–368, Springer, 2017.
- [112] N. Esfahani and S. Malek, "Uncertainty in self-adaptive software systems," in *Software Engineering for Self-Adaptive Systems II*, pp. 214–238, Springer, 2013.
- [113] J. Voros, "A primer on futures studies, foresight and the use of scenarios," *Prospect: The Foresight Bulletin*, vol. 6, no. 1, 2001.
- [114] R. Kazman, M. Klein, and P. Clements, "Atam: Method for architecture evaluation," tech. rep., Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst, 2000.
- [115] P. Traverso, "Planning under uncertainty and its applications," in *Reasoning, Action and Interaction in AI Theories and Systems*, pp. 213–228, Springer, 2006.
- [116] M. Schmitt Laser, N. Medvidovic, D. M. Le, and J. Garcia, "Arcade: an extensible workbench for architecture recovery, change, and decay evaluation," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1546–1550, 2020.
- [117] L. O'Brien, "Architecture reconstruction to support a product line effort: Case study," tech. rep., CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 2001.

- [118] C. Lupafya and D. Balasubramaniam, “A framework for considering uncertainty in software systems,” in *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 1519–1524, IEEE, 2022.
- [119] R. Kazman, M. Klein, and P. Clements, “Atam: Method for architecture evaluation,” tech. rep., Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst, 2000.
- [120] E. Ntontos, U. Zdun, K. Plakidas, S. Meixner, and S. Geiger, “Assessing architecture conformance to coupling-related patterns and practices in microservices,” in *European Conference on Software Architecture*, pp. 3–20, Springer, 2020.
- [121] N. Esfahani, E. Kouroshfar, and S. Malek, “Taming uncertainty in self-adaptive software,” in *SIGSOFT/FSE 2011 - Proceedings of the 19th ACM SIGSOFT Symposium on Foundations of Software Engineering*, pp. 234–244, 2011.



APPENDIX A

APPENDIX-A - ARCHITECTURE SURVEY LITERATURE SOURCES

Part 2 of 5: Research focus of literature source: aspects

Reference	Uncertainty	Definitions and concepts											Activities							Artefacts				Tools	Notations		
		Classification	General	Cd-Sys			SE Methodology	Sys-Q			Communication	Decisions & Tradeoff	SA-LC Stages					Self-Adaptation	Description	Documentation	Requirements	Traceability					
				Systems of systems	IoT	Knowledge		Performance	Reliability	Stability			Risk	Analysis	Deployment	Design	Evaluation						Evolution			Run-time	
Analysing dynamic change in distributed software architectures [44]	E												✓					✓									
Architecting in the face of uncertainty: An experience report [38]	E														✓							✓					
Architecturally significant requirements, reference architecture, and metamodel for knowledge management in information technology services [35]	E						✓					✓			✓										✓	✓	
Architecture-based reliability evaluation under uncertainty [67]	E								✓						✓												
ArchLearner: Leveraging machine-learning techniques for proactive architectural adaptation [100]	B																	✓	✓								
Ark: a constraint-based method for architectural synthesis of smart systems [58]	B			✓											✓					✓					✓	✓	
Automatic uncertainty detection in software architecture documentation [61]	E																				✓						
Case Study on the Fuzzy Architecture Description of Cyber-Physical SoS under Uncertainty [56]	B			✓											✓					✓					✓	✓	
Combined architectural framework for the selection of architectures using ATAM, FAHP and CBAM [32]	E															✓											
Dealing with uncertainty in architectural confidentiality analysis [80]	B												✓														
Dealing with uncertainty in early software architecture [26]	E														✓	✓											
Dealing with Uncertainty in Software Architecture on the Internet-of-Things with Digital Twins [54]	E			✓											✓			✓							✓	✓	
Eagle: Engineering software in the ubiquitous globe by leveraging uncertainty [24]	E				✓		✓																				
Early prediction of software component reliability [66]	E								✓						✓												
Estimating and understanding architectural risk [70]	B	✓								✓			✓		✓												
Evaluation of software architectures under uncertainty: a systematic literature review [40]	B														✓		✓										

Reference	Uncertainty	Classification	Definitions and concepts										Activities										Artefacts				
			General	Systems of systems		Knowledge	SE Methodology	Sys-Q			Risk	Communication	Decisions & Tradeoff		SA-LC Stages						Self-Adaptation	Description	Documentation	Requirements	Traceability	Tools	Notations
				IoT			Performance	Reliability	Stability				Analysis	Deployment	Design	Evaluation	Evolution	Run-time									
Explaining Architectural Design Tradeoff Spaces: A Machine Learning Approach [76]	B												✓		✓			✓									
Exploiting traceability uncertainty between architectural models and code [105]	E														✓						✓			✓	✓	✓	
Exploiting traceability uncertainty between software architectural models and performance analysis [64]	E						✓						✓										✓	✓	✓		
FAME: A UML-based framework for modeling fuzzy self-adaptive software. Information and Software Technology [96]	E												✓		✓				✓	✓		✓		✓	✓		
Formally describing the software architecture of Systems-of-Systems with SosADL [57]	E			✓											✓					✓				✓	✓		
Foundations for the study of software architecture [29]			✓																								
Fuzzy Architecture Description for Handling Uncertainty in IoT Systems-of-Systems [55]	E			✓	✓															✓				✓			
Guidearch: guiding the exploration of architectural solution space under uncertainty [27]	E												✓				✓							✓	✓		
Guiding architectural decision making on service mesh based microservice architectures [75]	E												✓		✓												
Handling uncertainty in autonomic systems. [90]	B																	✓	✓								
IArch-U: Interface-Centric Integrated Uncertainty-Aware Development Environment [82]	E														✓									✓	✓		
Identifying and addressing uncertainty in architecture-level software reliability modeling [65]	E							✓									✓										
Information Systems Management Issues for the 1990s [52]			✓																								
Integrating the Architecture Tradeoff Analysis Method (ATAM) with the cost benefit analysis method (CBAM). [48]	E												✓			✓											
Knowledge based decision framework for architecting complex systems [39]	E					✓							✓		✓						✓	✓	✓				
Model-based Performance Analysis for Architecting Cyber-Physical Dynamic Spaces [63]	B						✓						✓														
Model-based performance analysis of software architectures under uncertainty [33]	E						✓						✓											✓	✓		

Part 4 of 5: Research focus of literature source: aspects

Reference	Uncertainty	Classification	Definitions and concepts										Activities								Artefacts				Tools	Notations	
			General	Cd-Sys			Sys-Q				Communication	Decisions & Tradeoff	SA-LC Stages						Self-Adaptation	Description	Documentation	Requirements	Traceability				
				Systems of systems	IoT	Knowledge	SE Methodology	Performance	Reliability	Stability			Risk	Analysis	Deployment	Design	Evaluation	Evolution						Run-time			
MU-MMINT: An IDE for Model Uncertainty [83]	E															✓					✓					✓	✓
On how to deal with uncertainty when architecting embedded software and systems [53]	B	✓	✓													✓											
On the role of software architectures in runtime system reconfiguration [45]	B													✓					✓								✓
Past and future of software architectural decisions - A systematic mapping [72]	E		✓			✓							✓									✓					
Performance evaluation of software architectures. [46]	E							✓						✓			✓										
Quality attributes use in architecture design decision methods: research and practice [42]	E												✓														
Rainbow: Architecture-based self-adaptation with reusable infrastructure [94]	B																		✓	✓							
RaM: Causally-Connected and Requirements-Aware Runtime Models using Bayesian Learning [99]	B																		✓	✓						✓	✓
Reducing uncertainty in architectural decisions with AADL [74]	E												✓			✓										✓	✓
RELAX: Incorporating uncertainty into the specification of self-adaptive systems [25]	B																			✓			✓			✓	✓
Reliability evaluation of UML/DAM software architectures under parameter uncertainty [34]	E								✓								✓									✓	✓
Requirements volatility in software architecture design: An exploratory case study [104]	E																						✓				
Run-time evaluation of architectures: A case study of diversification in IoT [81]	E																✓		✓								
Self-adaptive method based on software architecture by inspecting uncertainty [92]	B															✓			✓	✓	✓					✓	✓
Software engineering in an uncertain world. [19]	B		✓																								
Software reliability with architectural uncertainties [68]	E								✓					✓			✓										
SQME: a framework for modeling and evaluation of software architecture quality attributes. [106]	E																✓										
Stitch: A language for architecture-based self-adaptation [91]	B																		✓	✓						✓	✓

Part 5 of 5: Research focus of literature source: aspects

Reference	Uncertainty	Definitions and concepts										Activities								Artefacts				Tools	Notations		
		Classification	General	Cd-Sys		Knowledge	SE Methodology	Sys-Q			Communication	Decisions & Tradeoff	SA-LC Stages					Self-Adaptation	Description	Documentation	Requirements	Traceability					
				Systems of systems	IoT			Performance	Reliability	Stability			Risk	Analysis	Deployment	Design	Evaluation						Evolution			Run-time	
Successful requirement elicitation by combining requirement engineering techniques [102]	E																										
Supporting architectural decision making for systems-of-systems design under uncertainty [59]	E			✓											✓					✓	✓			✓			
Synthesizing tradeoff spaces with quantitative guarantees for families of software systems [78]	A												✓		✓			✓									
Systematic selection of software architecture styles [85]	E																✓										
Taming uncertainty in self-adaptive software [121]	E																	✓	✓								
Targeting uncertainty in smart CPS by confidence-based logic [97]	B																	✓	✓								
The uncertainty principle in software engineering [20]	E		✓																								
The Tao way to anti-fragile software architectures: the case of mobile applications [51]	B		✓																								
Towards search-based modelling and analysis of requirements and architecture decisions [73]	E											✓	✓		✓							✓			✓	✓	
Towards using probabilistic models to design software systems with inherent uncertainty [84]	B														✓	✓			✓						✓	✓	
Tuning self-adaptation in cyber-physical systems through architectural homeostasis [98]	B																	✓	✓								
Uncertainties in the modeling of self-adaptive systems: a taxonomy and an example of availability evaluation [22]	B	✓																	✓								
Uncertainty expressions in software architecture group decision making: Explorative study [71]	E					✓					✓										✓						
Uncertainty in Self-adaptive Systems: A Research Community Perspective [41]	B																	✓									
Uncertainty, Risk, and Information Value in Software Requirements and Architecture. [103]	E										✓		✓		✓							✓					
Using real options to select stable middleware induced software architectures. [69]	E											✓				✓	✓										

APPENDIX-B

UNCERTAINTY

FRAMEWORK DATA AND

META-DATA

B.1 Uncertainty Attributes - JSON OBJECT template

```
{
  "id":1642559068879,
  "facet":"",
  "manifestation":"",
  "monitor":"",
  "bound":"",
  "nature":{
    "options":[
      "Aleatory",
      "Epistemic"
    ],
    "value":""
  },
  "perspective":{
    "options":[
      "Subjective",
      "Objective"
    ],
    "value":""
  },
  "source":{
```

```

    "options":[
      "External",
      "Internal"
    ],
    "value":""
  },
  "viewpoint":{
    "options":[
      "Logical",
      "Physical",
      "Process"
    ],
    "value":""
  },
  "description":"",
  "evidence":"",
  "relationship":"",
  "SourceDescription":"",
  "cause":"",
  "relatedUncertainties":{

},
  "location":"",
  "level":{
    "options":[
      "High",
      "Medium",
      "Low"
    ],
    "value":""
  },
  "awareness":{
    "options":[
      "Known Unknown",
      "Unknown unknown"
    ],
    "value":""
  },
  "emergingTime":{
    "options":[
      "Requirement",
      "Development",
      "Runtime"
    ],
    "value":""
  },
  "lifetime":"",
  "change":{
    "options":[
      "Static",
      "Dynamic"
    ],
    "value":""
  },
},

```

```
"pattern":{
  "options":[
    "Periodic",
    "Persistence ",
    "Transient",
    "Sporadic"
  ],
  "value":""
},
"measure":{
  "options":[
    "Probability",
    "Persistence ",
    "Fuzziness",
    "Temporal logic",
    "Non-specificity"
  ],
  "value":""
},
"dependency": "",
"risk": "",
"opportunity": "",
"mitigation": "",
"exploitation": "",
"outcome": ""
}
```

Listing B.1: JSON data

B.2 XML data for screenshot

```

<mxGraphModel dx="2229" dy="780" grid="1" gridSize="10" guides="1" tooltips="1"
  connect="1" arrows="1" fold="1" page="1" pageScale="1" pageWidth="827"
  pageHeight="1169" math="0" shadow="0">
<root>
  <mxCell id="0" />
  <mxCell id="1" parent="0" />
  <mxCell id="Lx7MsKEuWNhGFfsztCJ6-1" value="Logical architecture" parent="0"
    />
  <mxCell id="Lx7MsKEuWNhGFfsztCJ6-30" value="Block" style="verticalAlign=top;
    align=left;spacingTop=8;spacingLeft=2;spacingRight=12;shape=cube;size
    =10;direction=south;fontStyle=4;html=1;" vertex="1" parent="
    Lx7MsKEuWNhGFfsztCJ6-1">
    <mxGeometry x="40" y="80" width="785" height="560" as="geometry" />
  </mxCell>
  <mxCell id="Lx7MsKEuWNhGFfsztCJ6-32" value="MEME Network" style="text;html
    =1;align=center;verticalAlign=middle;whiteSpace=wrap;rounded=0;
    strokeWidth=4;strokeColor=#000000;fontStyle=1" vertex="1" parent="
    Lx7MsKEuWNhGFfsztCJ6-1">
    <mxGeometry x="40" y="90" width="777.5" height="30" as="geometry" />
  </mxCell>
  <mxCell id="Lx7MsKEuWNhGFfsztCJ6-33" value="Governance and collaboration"
    style="text;html=1;align=center;verticalAlign=middle;whiteSpace=wrap;
    rounded=0;strokeWidth=4;strokeColor=#000000;fontStyle=1" vertex="1"
    parent="Lx7MsKEuWNhGFfsztCJ6-1">
    <mxGeometry x="170" y="162" width="640" height="20" as="geometry" />
  </mxCell>
  <mxCell id="Lx7MsKEuWNhGFfsztCJ6-35" value="" style="rounded=0;whiteSpace=
    wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1" parent="
    Lx7MsKEuWNhGFfsztCJ6-1">
    <mxGeometry x="170" y="520" width="630" height="105" as="geometry" />
  </mxCell>
  <mxCell id="Lx7MsKEuWNhGFfsztCJ6-47" value="Manage" style="text;html=1;
    strokeColor=none;fillColor=none;align=center;verticalAlign=middle;
    whiteSpace=wrap;rounded=0;fontStyle=1" vertex="1" parent="
    Lx7MsKEuWNhGFfsztCJ6-1">
    <mxGeometry x="180" y="526" width="40" height="20" as="geometry" />
  </mxCell>
  <mxCell id="Lx7MsKEuWNhGFfsztCJ6-48" value="Metadata" style="rounded=0;
    whiteSpace=wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1"
    parent="Lx7MsKEuWNhGFfsztCJ6-1">
    <mxGeometry x="690" y="570" width="100" height="36" as="geometry" />
  </mxCell>
  <mxCell id="Lx7MsKEuWNhGFfsztCJ6-50" value="&lt;div&gt;ETL &amp; Data&lt
    ;/div&gt;&lt;div&gt;Preparation&lt;br&gt;&lt;/div&gt;" style="rounded=0;
    whiteSpace=wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1"
    parent="Lx7MsKEuWNhGFfsztCJ6-1">
    <mxGeometry x="690" y="530" width="100" height="36" as="geometry" />
  </mxCell>
  <mxCell id="Lx7MsKEuWNhGFfsztCJ6-51" value="Data Privacy" style="rounded=0;
    whiteSpace=wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1"

```

```

        parent="Lx7MsKEuWNhGFfsztCJ6-1">
        <mxGeometry x="560" y="570" width="120" height="36" as="geometry" />
    </mxCell>
    <mxCell id="Lx7MsKEuWNhGFfsztCJ6-52" value="Tagging" style="rounded=0;
        whiteSpace=wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1"
        parent="Lx7MsKEuWNhGFfsztCJ6-1">
        <mxGeometry x="430" y="530" width="114" height="36" as="geometry" />
    </mxCell>
    <mxCell id="Lx7MsKEuWNhGFfsztCJ6-53" value="Data Quality" style="rounded=0;
        whiteSpace=wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1"
        parent="Lx7MsKEuWNhGFfsztCJ6-1">
        <mxGeometry x="560" y="530" width="120" height="36" as="geometry" />
    </mxCell>
    <mxCell id="Lx7MsKEuWNhGFfsztCJ6-54" value="Data Security" style="rounded=0;
        whiteSpace=wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1"
        parent="Lx7MsKEuWNhGFfsztCJ6-1">
        <mxGeometry x="430" y="570" width="115" height="36" as="geometry" />
    </mxCell>
    <mxCell id="Lx7MsKEuWNhGFfsztCJ6-55" value="" style="group" vertex="1"
        connectable="0" parent="Lx7MsKEuWNhGFfsztCJ6-1">
        <mxGeometry x="560" y="190" width="120" height="320" as="geometry" />
    </mxCell>
    <mxCell id="Lx7MsKEuWNhGFfsztCJ6-39" value="" style="rounded=0;whiteSpace=
        wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1" parent="
        Lx7MsKEuWNhGFfsztCJ6-55">
        <mxGeometry width="120" height="320" as="geometry" />
    </mxCell>
    <mxCell id="Lx7MsKEuWNhGFfsztCJ6-45" value="Analyse" style="text;html=1;
        strokeColor=none;fillColor=none;align=center;verticalAlign=middle;
        whiteSpace=wrap;rounded=0;fontStyle=1" vertex="1" parent="
        Lx7MsKEuWNhGFfsztCJ6-55">
        <mxGeometry x="9.230769230769232" y="8" width="36.92307692307693" height="
        20" as="geometry" />
    </mxCell>
    <mxCell id="Lx7MsKEuWNhGFfsztCJ6-83" value="&lt;div&gt;Data Request&lt;/div&
        gt;&lt;div&gt;Fulfilment&lt;br&gt;&lt;/div&gt;" style="rounded=0;
        whiteSpace=wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1"
        parent="Lx7MsKEuWNhGFfsztCJ6-55">
        <mxGeometry x="5" y="60" width="110" height="36" as="geometry" />
    </mxCell>
    <mxCell id="Lx7MsKEuWNhGFfsztCJ6-85" value="&lt;div&gt;Occ. Health&lt;/div&
        gt;&lt;div&gt;Surveillance Analysis&lt;br&gt;&lt;/div&gt;" style="
        rounded=0;whiteSpace=wrap;html=1;strokeColor=#000000;strokeWidth=2;"
        vertex="1" parent="Lx7MsKEuWNhGFfsztCJ6-55">
        <mxGeometry x="5" y="100" width="110" height="50" as="geometry" />
    </mxCell>
    <mxCell id="Lx7MsKEuWNhGFfsztCJ6-84" value="&lt;div&gt;Exploratory&lt;/div&
        gt;&lt;div&gt;Analysis&lt;br&gt;&lt;/div&gt;" style="rounded=0;
        whiteSpace=wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1"
        parent="Lx7MsKEuWNhGFfsztCJ6-55">
        <mxGeometry x="5" y="160" width="120" height="36" as="geometry" />
    </mxCell>
    <mxCell id="Lx7MsKEuWNhGFfsztCJ6-86" value="&lt;div&gt;Search, Mine &amp;
        &lt;/div&gt;&lt;div&gt;Ad-hoc Query&lt;/div&gt;" style="rounded=0;

```



```

        whiteSpace=wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1"
        parent="Lx7MsKEuWNhGFfsztCJ6-55">
    <mxGeometry x="5" y="200" width="110" height="36" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-87" value="&lt;div&gt;Reports &amp;&amp;&lt;
;/div&gt;&lt;div&gt;Dashboards&lt;br&gt;&lt;/div&gt;" style="rounded=0;
whiteSpace=wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1"
parent="Lx7MsKEuWNhGFfsztCJ6-55">
    <mxGeometry x="5" y="240" width="110" height="36" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-88" style="edgeStyle=orthogonalEdgeStyle;
rounded=0;orthogonalLoop=1;jettySize=auto;html=1;exitX=0.5;exitY=1;
exitDx=0;exitDy=0;" edge="1" parent="Lx7MsKEuWNhGFfsztCJ6-55" source="
Lx7MsKEuWNhGFfsztCJ6-84" target="Lx7MsKEuWNhGFfsztCJ6-84">
    <mxGeometry relative="1" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-89" value="&lt;div&gt;Analysis &amp;&amp;&lt;
;/div&gt;&lt;div&gt;Visualization&lt;br&gt;&lt;/div&gt;" style="rounded
=0;whiteSpace=wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1"
parent="Lx7MsKEuWNhGFfsztCJ6-55">
    <mxGeometry x="5" y="280" width="110" height="36" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-56" value="" style="group" vertex="1"
connectable="0" parent="Lx7MsKEuWNhGFfsztCJ6-1">
    <mxGeometry x="690" y="190" width="110" height="320" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-40" value="" style="rounded=0;whiteSpace=
wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1" parent="
Lx7MsKEuWNhGFfsztCJ6-56">
    <mxGeometry width="110" height="320" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-46" value="Distribute" style="text;html=1;
strokeColor=none;fillColor=none;align=center;verticalAlign=middle;
whiteSpace=wrap;rounded=0;fontStyle=1" vertex="1" parent="
Lx7MsKEuWNhGFfsztCJ6-56">
    <mxGeometry x="20.00153846153846" y="10" width="33.84615384615385" height=
"20" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-76" value="&lt;div&gt;&lt;br&gt;&lt;/div&gt;
&lt;div&gt;Reusable&lt;/div&gt;&lt;div&gt;Datasets&lt;br&gt;&lt;/div&gt;
;" style="strokeWidth=2;html=1;shape=mxgraph.flowchart.database;
whiteSpace=wrap;" vertex="1" parent="Lx7MsKEuWNhGFfsztCJ6-56">
    <mxGeometry x="10" y="50" width="80" height="60" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-77" value="&lt;div&gt;Packaged&lt;/div&gt;&
lt;div&gt;Datasets&lt;br&gt;&lt;/div&gt;" style="shape=cube;whiteSpace=
wrap;html=1;boundedLbl=1;backgroundOutline=1;darkOpacity=0.05;
darkOpacity2=0.1;strokeColor=#000000;strokeWidth=2;" vertex="1" parent="
Lx7MsKEuWNhGFfsztCJ6-56">
    <mxGeometry x="13.75" y="210" width="82.5" height="60" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-80" value="Files" style="strokeWidth=2;html
=1;shape=mxgraph.flowchart.multi-document;whiteSpace=wrap;fillColor=#
ffffff;direction=south;" vertex="1" parent="Lx7MsKEuWNhGFfsztCJ6-56">

```

```

    <mxGeometry x="30" y="160" width="50" height="62" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-57" value="" style="group" vertex="1"
    connectable="0" parent="Lx7MsKEuWNhGFfsztCJ6-1">
    <mxGeometry x="430" y="190" width="114" height="320" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-38" value="" style="rounded=0;whiteSpace=
    wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1" parent="
    Lx7MsKEuWNhGFfsztCJ6-57">
    <mxGeometry width="114" height="320" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-44" value="Store" style="text;html=1;
    strokeColor=none;fillColor=none;align=center;verticalAlign=middle;
    whiteSpace=wrap;rounded=0;fontStyle=1" vertex="1" parent="
    Lx7MsKEuWNhGFfsztCJ6-57">
    <mxGeometry x="8.76923076923077" y="8" width="35.07692307692308" height="
    20" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-90" value="" style="rounded=0;whiteSpace=
    wrap;html=1;strokeColor=#000000;strokeWidth=2;fillColor=#ffffff;dashed
    =1;" vertex="1" parent="Lx7MsKEuWNhGFfsztCJ6-57">
    <mxGeometry x="15" y="190" width="84" height="110" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-91" value="" style="rounded=0;whiteSpace=
    wrap;html=1;strokeColor=#000000;strokeWidth=2;fillColor=#ffffff;dashed
    =1;" vertex="1" parent="Lx7MsKEuWNhGFfsztCJ6-57">
    <mxGeometry x="15" y="30" width="84" height="130" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-92" value="&lt;div&gt;Ops&lt;/div&gt;&lt;
    div&gt;Data&lt;br&gt;&lt;/div&gt;" style="strokeWidth=2;html=1;shape=
    mxgraph.flowchart.database;whiteSpace=wrap;fillColor=#ffffff;" vertex="1"
    parent="Lx7MsKEuWNhGFfsztCJ6-57">
    <mxGeometry x="22.5" y="60" width="39" height="40" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-95" value="&lt;div&gt;Active&lt;/div&gt;&lt;
    div&gt;Data&lt;br&gt;&lt;/div&gt;" style="strokeWidth=2;html=1;shape=
    mxgraph.flowchart.database;whiteSpace=wrap;fillColor=#ffffff;" vertex="1"
    parent="Lx7MsKEuWNhGFfsztCJ6-57">
    <mxGeometry x="52.5" y="60" width="39" height="40" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-94" value="&lt;div&gt;Master&lt;/div&gt;&lt;
    div&gt;Data&lt;br&gt;&lt;/div&gt;" style="strokeWidth=2;html=1;shape=
    mxgraph.flowchart.database;whiteSpace=wrap;fillColor=#ffffff;" vertex="1"
    parent="Lx7MsKEuWNhGFfsztCJ6-57">
    <mxGeometry x="22.5" y="120" width="40" height="30" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-93" value="&lt;div&gt;Ref&lt;/div&gt;&lt;
    div&gt;Data&lt;br&gt;&lt;/div&gt;" style="strokeWidth=2;html=1;shape=
    mxgraph.flowchart.database;whiteSpace=wrap;fillColor=#ffffff;" vertex="1"
    parent="Lx7MsKEuWNhGFfsztCJ6-57">
    <mxGeometry x="52.5" y="110" width="39" height="40" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-101" value="Trusted Data" style="text;html
    =1;strokeColor=none;fillColor=none;align=center;verticalAlign=middle;

```

```

        whiteSpace=wrap;rounded=0;fontStyle=1" vertex="1" parent="
        Lx7MsKEuWNhGFfsztCJ6-57">
    <mxGeometry x="15" y="30" width="80" height="20" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-100" value="&lt;div&gt;Raw Data&lt;/div&gt;"
    style="text;html=1;strokeColor=none;fillColor=none;align=center;
    verticalAlign=middle;whiteSpace=wrap;rounded=0;fontStyle=1" vertex="1"
    parent="Lx7MsKEuWNhGFfsztCJ6-57">
    <mxGeometry x="11.5" y="190" width="80" height="20" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-96" value="EMR" style="strokeWidth=2;html
    =1;shape=mxgraph.flowchart.database;whiteSpace=wrap;fillColor=#ffffff;"
    vertex="1" parent="Lx7MsKEuWNhGFfsztCJ6-57">
    <mxGeometry x="23.5" y="210" width="39" height="30" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-97" value="LSAH" style="strokeWidth=2;html
    =1;shape=mxgraph.flowchart.database;whiteSpace=wrap;fillColor=#ffffff;"
    vertex="1" parent="Lx7MsKEuWNhGFfsztCJ6-57">
    <mxGeometry x="43.85" y="230" width="39" height="30" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-98" value="MEDB" style="strokeWidth=2;html
    =1;shape=mxgraph.flowchart.database;whiteSpace=wrap;fillColor=#ffffff;"
    vertex="1" parent="Lx7MsKEuWNhGFfsztCJ6-57">
    <mxGeometry x="60" y="250" width="39" height="40" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-58" value="" style="group;fontStyle=1"
    vertex="1" connectable="0" parent="Lx7MsKEuWNhGFfsztCJ6-1">
    <mxGeometry x="300" y="190" width="114" height="320" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-37" value="" style="rounded=0;whiteSpace=
    wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1" parent="
    Lx7MsKEuWNhGFfsztCJ6-58">
    <mxGeometry width="114" height="320" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-43" value="Refine" style="text;html=1;
    strokeColor=none;fillColor=none;align=center;verticalAlign=middle;
    whiteSpace=wrap;rounded=0;fontStyle=1" vertex="1" parent="
    Lx7MsKEuWNhGFfsztCJ6-58">
    <mxGeometry x="8.76923076923077" y="11" width="35.07692307692308" height="
    20" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-103" value="Data Cataloging" style="rounded
    =0;whiteSpace=wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1"
    parent="Lx7MsKEuWNhGFfsztCJ6-58">
    <mxGeometry x="5" y="40" width="104" height="30" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-106" value="&lt;div&gt;Error &amp;&lt;/
    div&gt;&lt;div&gt;Exception&lt;/div&gt;&lt;div&gt;Management&lt;br&gt;&
    lt;/div&gt;" style="rounded=0;whiteSpace=wrap;html=1;strokeColor
    =#000000;strokeWidth=2;" vertex="1" parent="Lx7MsKEuWNhGFfsztCJ6-58">
    <mxGeometry x="5" y="80" width="104" height="60" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-107" value="Data Profiling" style="rounded
    =0;whiteSpace=wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1"

```

```

        parent="Lx7MsKEuWNhGFfsztCJ6-58">
        <mxGeometry x="5" y="150" width="104" height="30" as="geometry" />
    </mxCell>
    <mxCell id="Lx7MsKEuWNhGFfsztCJ6-108" value="Data Edits" style="rounded=0;
        whiteSpace=wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1"
        parent="Lx7MsKEuWNhGFfsztCJ6-58">
        <mxGeometry x="5" y="190" width="104" height="30" as="geometry" />
    </mxCell>
    <mxCell id="Lx7MsKEuWNhGFfsztCJ6-60" value="" style="group" vertex="1"
        connectable="0" parent="Lx7MsKEuWNhGFfsztCJ6-1">
        <mxGeometry x="170" y="190" width="115" height="320" as="geometry" />
    </mxCell>
    <mxCell id="Lx7MsKEuWNhGFfsztCJ6-41" value="" style="rounded=0;whiteSpace=
        wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1" parent="
        Lx7MsKEuWNhGFfsztCJ6-60">
        <mxGeometry width="115" height="320" as="geometry" />
    </mxCell>
    <mxCell id="Lx7MsKEuWNhGFfsztCJ6-42" value="Transport" style="text;html=1;
        strokeColor=none;fillColor=none;align=center;verticalAlign=middle;
        whiteSpace=wrap;rounded=0;fontStyle=1" vertex="1" parent="
        Lx7MsKEuWNhGFfsztCJ6-60">
        <mxGeometry x="19.996153846153845" y="10" width="35.38461538461539" height
            ="20" as="geometry" />
    </mxCell>
    <mxCell id="Lx7MsKEuWNhGFfsztCJ6-116" value="&lt;div&gt;Connect to Data&lt;/div&gt;
        &lt;div&gt;&lt;div&gt;Source&lt;br&gt;&lt;/div&gt;" style="rounded=0;
        whiteSpace=wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1"
        parent="Lx7MsKEuWNhGFfsztCJ6-60">
        <mxGeometry x="5.5" y="40" width="104" height="30" as="geometry" />
    </mxCell>
    <mxCell id="Lx7MsKEuWNhGFfsztCJ6-118" value="&lt;div&gt;Ingestion at&lt;/div&gt;
        &lt;div&gt;&lt;div&gt;scale&lt;br&gt;&lt;/div&gt;" style="rounded=0;whiteSpace=
        wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1" parent="
        Lx7MsKEuWNhGFfsztCJ6-60">
        <mxGeometry x="5.5" y="90" width="104" height="30" as="geometry" />
    </mxCell>
    <mxCell id="Lx7MsKEuWNhGFfsztCJ6-119" value="&lt;div&gt;Rules-based&lt;/div&gt;
        &lt;div&gt;&lt;div&gt;Cleaning&lt;br&gt;&lt;/div&gt;" style="rounded=0;
        whiteSpace=wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1"
        parent="Lx7MsKEuWNhGFfsztCJ6-60">
        <mxGeometry x="5.5" y="130" width="104" height="30" as="geometry" />
    </mxCell>
    <mxCell id="Lx7MsKEuWNhGFfsztCJ6-81" value="Access Control" style="rounded
        =0;whiteSpace=wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1"
        parent="Lx7MsKEuWNhGFfsztCJ6-1">
        <mxGeometry x="300" y="570" width="114" height="36" as="geometry" />
    </mxCell>
    <mxCell id="Lx7MsKEuWNhGFfsztCJ6-82" value="Data Cataloging" style="rounded
        =0;whiteSpace=wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1"
        parent="Lx7MsKEuWNhGFfsztCJ6-1">
        <mxGeometry x="300" y="530" width="114" height="36" as="geometry" />
    </mxCell>
    <mxCell id="Lx7MsKEuWNhGFfsztCJ6-120" value="" style="group" vertex="1"
        connectable="0" parent="Lx7MsKEuWNhGFfsztCJ6-1">

```

```

    <mxGeometry x="40" y="120" width="110" height="520" as="geometry" />
  </mxCell>
  <mxCell id="Lx7MsKEuWNhGFfsztCJ6-36" value="" style="rounded=0;whiteSpace=
    wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1" parent="
    Lx7MsKEuWNhGFfsztCJ6-120">
    <mxGeometry width="110" height="519.9999999999999" as="geometry" />
  </mxCell>
  <mxCell id="Lx7MsKEuWNhGFfsztCJ6-66" value="" style="group" vertex="1"
    connectable="0" parent="Lx7MsKEuWNhGFfsztCJ6-120">
    <mxGeometry y="31.108235294117648" width="110" height="366.53882352941173"
      as="geometry" />
  </mxCell>
  <mxCell id="Lx7MsKEuWNhGFfsztCJ6-67" value="" style="rounded=0;whiteSpace=
    wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1" parent="
    Lx7MsKEuWNhGFfsztCJ6-66">
    <mxGeometry y="21.02195592621081" width="110" height="345.5168676032009"
      as="geometry" />
  </mxCell>
  <mxCell id="Lx7MsKEuWNhGFfsztCJ6-68" value="Capture" style="text;html=1;
    strokeColor=none;fillColor=none;align=center;verticalAlign=middle;
    whiteSpace=wrap;rounded=0;fontStyle=1" vertex="1" parent="
    Lx7MsKEuWNhGFfsztCJ6-66">
    <mxGeometry x="19.995217391304347" y="21.019260803656167" width="
      38.26086956521739" height="22.908541714460497" as="geometry" />
  </mxCell>
  <mxCell id="Lx7MsKEuWNhGFfsztCJ6-69" value="RDBMS" style="strokeWidth=2;html
    =1;shape=mxgraph.flowchart.database;whiteSpace=wrap;" vertex="1" parent=
    "Lx7MsKEuWNhGFfsztCJ6-66">
    <mxGeometry x="25" y="45.817083428920995" width="60" height="
      45.817083428920995" as="geometry" />
  </mxCell>
  <mxCell id="Lx7MsKEuWNhGFfsztCJ6-70" value="NoSQL" style="strokeWidth=2;html
    =1;shape=mxgraph.flowchart.database;whiteSpace=wrap;" vertex="1" parent=
    "Lx7MsKEuWNhGFfsztCJ6-66">
    <mxGeometry x="25" y="103.08843771507225" width="60" height="
      45.817083428920995" as="geometry" />
  </mxCell>
  <mxCell id="Lx7MsKEuWNhGFfsztCJ6-71" value="Cloud" style="ellipse;shape=
    cloud;whiteSpace=wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex=
    "1" parent="Lx7MsKEuWNhGFfsztCJ6-66">
    <mxGeometry x="5" y="240.53968800183523" width="85" height="
      45.817083428920995" as="geometry" />
  </mxCell>
  <mxCell id="Lx7MsKEuWNhGFfsztCJ6-73" value="Files" style="strokeWidth=2;html
    =1;shape=mxgraph.flowchart.multi-document;whiteSpace=wrap;" vertex="1"
    parent="Lx7MsKEuWNhGFfsztCJ6-66">
    <mxGeometry x="20" y="171.81406285845372" width="70" height="
      57.271354286151244" as="geometry" />
  </mxCell>
  <mxCell id="Lx7MsKEuWNhGFfsztCJ6-72" value="<div>File <br></div><br><br>
    <div><br><br>Shares<br><br></div>" style="shape=card;
    whiteSpace=wrap;html=1;strokeColor=#000000;strokeWidth=2;" vertex="1"
    parent="Lx7MsKEuWNhGFfsztCJ6-66">

```



```

    <mxGeometry x="20" y="297.8110422879865" width="55" height="
      57.271354286151244" as="geometry" />
  </mxCell>
  <mxCell id="Lx7MsKEuWNhGFfsztCJ6-122" value="" style="html=1;shadow=0;dashed
    =0;align=center;verticalAlign=middle;shape=mxgraph.arrows2.arrow;dy=0.6;
    dx=40;notch=0;strokeColor=#000000;strokeWidth=2;fillColor=#ffffff;"
    vertex="1" parent="Lx7MsKEuWNhGFfsztCJ6-120">
    <mxGeometry x="-100" y="10" width="200" height="30" as="geometry" />
  </mxCell>
  <mxCell id="Lx7MsKEuWNhGFfsztCJ6-121" value="IMPALA System" style="text;html
    =1;align=center;verticalAlign=middle;whiteSpace=wrap;rounded=0;
    strokeWidth=4;strokeColor=#000000;fontStyle=1" vertex="1" parent="
    Lx7MsKEuWNhGFfsztCJ6-1">
    <mxGeometry x="170" y="130" width="640" height="20" as="geometry" />
  </mxCell>
  <object label="HH&P IT" uncertaintyDB="{&quot;1623347515019&quot;;:&quot;
    ;id&quot;;:1623347515019,&quot;;system&quot;;:&quot;;System name&quot;;,&quot;
    ;manifestation&quot;;:&quot;;&quot;;,&quot;;environmentMonitor&quot;;:&quot;;&
    quot;;,&quot;;nature&quot;;:&quot;;options&quot;;:[&quot;;Aleatory&quot;;,&
    quot;;Epistemic&quot;;],&quot;;value&quot;;:&quot;;&quot;;,&quot;;perspective&
    quot;;:&quot;;options&quot;;:[&quot;;Subjective&quot;;,&quot;;Objective&quot;
    ;],&quot;;value&quot;;:&quot;;&quot;;,&quot;;source&quot;;:&quot;;options&
    quot;;:[&quot;;External&quot;;,&quot;;Internal&quot;;],&quot;;value&quot;;:&
    quot;;&quot;;,&quot;;viewpoint&quot;;:&quot;;options&quot;;:[&quot;;Logical&
    quot;;,&quot;;Physical&quot;;,&quot;;Process&quot;;],&quot;;value&quot;;:&quot;
    &quot;;&quot;;,&quot;;description&quot;;:&quot;;Description&quot;;,&quot;;evidence&
    quot;;:&quot;;Evidence&quot;;,&quot;;SourceDescription&quot;;:&quot;;&quot;;,&
    quot;;relatedUncertainties&quot;;:&quot;;,&quot;;location&quot;;:&quot;;&quot;;,&
    quot;;level&quot;;:&quot;;options&quot;;:[&quot;;Known Unknown&quot;;,&quot;
    Unknown unknown&quot;;,&quot;;Statistical&quot;;],&quot;;value&quot;;:&quot;;&
    quot;;,&quot;;awareness&quot;;:&quot;;options&quot;;:[&quot;;Known Unknown&
    quot;;,&quot;;Unknown unknown&quot;;],&quot;;value&quot;;:&quot;;&quot;;,&quot;
    ;emergingTime&quot;;:&quot;;options&quot;;:[&quot;;Requirement&quot;;,&quot;
    Development&quot;;,&quot;;Runtime&quot;;],&quot;;value&quot;;:&quot;;&quot;;,&
    quot;;lifetime&quot;;:&quot;;&quot;;,&quot;;pattern&quot;;:&quot;;options&quot;
    ;:[&quot;;Periodic&quot;;,&quot;;Persistence &quot;;,&quot;;Transient&quot;;,&
    quot;;Sporadic&quot;;],&quot;;value&quot;;:&quot;;&quot;;,&quot;;measure&quot;
    ;:&quot;;options&quot;;:[&quot;;Probability&quot;;,&quot;;Persistence &quot;
    ;,&quot;;Fuzziness&quot;;,&quot;;Temporal logic&quot;;,&quot;;Non-specificity
    &quot;;],&quot;;value&quot;;:&quot;;&quot;;,&quot;;related&quot;;:&quot;;&quot;
    ;,&quot;;operator&quot;;:&quot;;options&quot;;:[&quot;;Modal&quot;;,&quot;
    Temporal&quot;;,&quot;;Ordinal&quot;;],&quot;;value&quot;;:&quot;;Modal&quot;
    ;,&quot;;modal&quot;;:&quot;;options&quot;;:[&quot;;MAY&quot;;,&quot;;SHALL&
    quot;;],&quot;;value&quot;;:&quot;;&quot;;,&quot;;ordinal&quot;;:&quot;;
    options&quot;;:[&quot;;AS CLOSE AS POSSIBLE TO &#39;?&#39;&quot;;,&quot;;AS
    MANY POSSIBLE TO &#39;?&#39;&quot;;,&quot;;AS FEW AS POSSIBLE TO
    &#39;?&#39;&quot;;],&quot;;value&quot;;:&quot;;&quot;;,&quot;;temporal&quot;
    ;:&quot;;options&quot;;:[&quot;;EVENTUALLY&quot;;,&quot;;UNTIL&quot;;,&quot;
    BEFORE&quot;;,&quot;;AFTER&quot;;,&quot;;AS EARLY AS POSSIBLE&quot;;,&quot;;AS
    LATE AS POSSIBLE&quot;;,&quot;;AS CLOSE AS POSSIBLE TO &#39;?&#39;&quot;
    ;],&quot;;value&quot;;:&quot;;&quot;;}}}" id="Lx7MsKEuWNhGFfsztCJ6-6">
  <mxCell style="swimlane;fontStyle=0;childLayout=stackLayout;horizontal=1;
    startSize=26;horizontalStack=0;resizeParent=1;resizeParentMax=0;
    resizeLast=0;collapsible=1;marginBottom=0;align=center;fontSize=14;"

```

```

        vertex="1" parent="Lx7MsKEuWNhGFfsztCJ6-1">
        <mxGeometry x="-220" y="308" width="160" height="104" as="geometry" />
    </mxCell>
</object>
<object label="System of Records" uncertaintyDB="{&quot;1623347531099&quot;
;:{&quot;id&quot;:1623347531099,&quot;system&quot;:&quot;System name&
quot;,&quot;manifestation&quot;:&quot;&quot;,&quot;environmentMonitor&
quot;:&quot;&quot;,&quot;nature&quot;:{&quot;options&quot;:[&quot;
Aleatory&quot;,&quot;Epistemic&quot;],&quot;value&quot;:&quot;&quot;},&
quot;perspective&quot;:{&quot;options&quot;:[&quot;Subjective&quot;,&
quot;Objective&quot;],&quot;value&quot;:&quot;&quot;},&quot;source&quot;
;:{&quot;options&quot;:[&quot;External&quot;,&quot;Internal&quot;],&quot;
value&quot;:&quot;&quot;},&quot;viewpoint&quot;:{&quot;options&quot;:[&
quot;Logical&quot;,&quot;Physical&quot;,&quot;Process&quot;],&quot;value
&quot;:&quot;&quot;},&quot;description&quot;:&quot;Description&quot;,&
quot;evidence&quot;:&quot;Evidence&quot;,&quot;SourceDescription&quot;:&
quot;&quot;,&quot;relatedUncertainties&quot;:{},&quot;location&quot;:&
quot;&quot;,&quot;level&quot;:{&quot;options&quot;:[&quot;Known Unknown&
quot;,&quot;Unknown unknown&quot;,&quot;Statistical&quot;],&quot;value&
quot;:&quot;&quot;},&quot;awareness&quot;:{&quot;options&quot;:[&quot;
Known Unknown&quot;,&quot;Unknown unknown&quot;],&quot;value&quot;:&quot;
&quot;},&quot;emergingTime&quot;:{&quot;options&quot;:[&quot;
Requirement&quot;,&quot;Development&quot;,&quot;Runtime&quot;],&quot;
value&quot;:&quot;&quot;},&quot;lifetime&quot;:&quot;&quot;,&quot;
pattern&quot;:{&quot;options&quot;:[&quot;Periodic&quot;,&quot;
Persistence &quot;,&quot;Transient&quot;,&quot;Sporadic&quot;],&quot;
value&quot;:&quot;&quot;},&quot;measure&quot;:{&quot;options&quot;:[&
quot;Probability&quot;,&quot;Persistence &quot;,&quot;Fuzziness&quot;,&
quot;Temporal logic&quot;,&quot;Non-specificity&quot;],&quot;value&quot;
:&quot;&quot;},&quot;related&quot;:&quot;&quot;,&quot;operator&quot;:{&
quot;options&quot;:[&quot;Modal&quot;,&quot;Temporal&quot;,&quot;Ordinal
&quot;],&quot;value&quot;:&quot;Modal&quot;},&quot;modal&quot;:{&quot;
options&quot;:[&quot;MAY&quot;,&quot;SHALL&quot;],&quot;value&quot;:&
quot;&quot;},&quot;ordinal&quot;:{&quot;options&quot;:[&quot;AS CLOSE AS
POSSIBLE TO &#39;&#39;&quot;,&quot;AS MANY POSSIBLE TO &#39;&#39;&
quot;,&quot;AS FEW AS POSSIBLE TO &#39;&#39;&quot;],&quot;value&quot;:&
quot;&quot;},&quot;temporal&quot;:{&quot;options&quot;:[&quot;EVENTUALLY
&quot;,&quot;UNTIL&quot;,&quot;BEFORE&quot;,&quot;AFTER&quot;,&quot;AS
EARLY AS POSSIBLE&quot;,&quot;AS LATE AS POSSIBLE&quot;,&quot;AS CLOSE
AS POSSIBLE TO &#39;&#39;&quot;],&quot;value&quot;:&quot;&quot;}}}" id=
"Lx7MsKEuWNhGFfsztCJ6-7">
<mxCell style="text;strokeColor=none;fillColor=none;spacingLeft=4;
spacingRight=4;overflow=hidden;rotatable=0;points=[[0,0.5],[1,0.5]];
portConstraint=eastwest;fontSize=12;" vertex="1" parent="
Lx7MsKEuWNhGFfsztCJ6-6">
    <mxGeometry y="26" width="160" height="26" as="geometry" />
</mxCell>
</object>
<object label="Build ETL Process" uncertaintyDB="{&quot;1623347529998&quot;
;:{&quot;id&quot;:1623347529998,&quot;system&quot;:&quot;System name&
quot;,&quot;manifestation&quot;:&quot;&quot;,&quot;environmentMonitor&
quot;:&quot;&quot;,&quot;nature&quot;:{&quot;options&quot;:[&quot;
Aleatory&quot;,&quot;Epistemic&quot;],&quot;value&quot;:&quot;&quot;},&
quot;perspective&quot;:{&quot;options&quot;:[&quot;Subjective&quot;,&

```

```

    "Objective"],"value":"","source"
    :{"options":["External","Internal"],"
    value":"","viewpoint":{"options":["
    Logical","Physical","Process"],"value
    ":"","description":"Description","
    "evidence":"Evidence","SourceDescription":"
    ","relatedUncertainties":{},"location":"
    ","level":{"options":["Known Unknown"
    ","Unknown unknown","Statistical"],"value"
    ":"","awareness":{"options":["
    Known Unknown","Unknown unknown"],"value":"
    ","emergingTime":{"options":["
    Requirement","Development","Runtime"],"
    value":"","lifetime":"","
    pattern":{"options":["Periodic","
    Persistence","Transient","Sporadic"],"
    value":"","measure":{"options":["
    Probability","Persistence","Fuzziness","
    Temporal logic","Non-specificity"],"value"
    :"","related":"","operator":{"
    options":["Modal","Temporal","Ordinal
    "],"value":"","modal":{"
    options":["MAY","SHALL"],"value":"
    ","ordinal":{"options":["AS CLOSE AS
    POSSIBLE TO &#39;&#39;","AS MANY POSSIBLE TO &#39;&#39;&
    ","AS FEW AS POSSIBLE TO &#39;&#39;","value":"
    ","temporal":{"options":["EVENTUALLY
    ","UNTIL","BEFORE","AFTER","AS
    EARLY AS POSSIBLE","AS LATE AS POSSIBLE","AS CLOSE
    AS POSSIBLE TO &#39;&#39;"],"value":""}}}" id=
    "Lx7MsKEuWNhGFfsztCJ6-8">
<mxCell style="text;strokeColor=none;fillColor=none;spacingLeft=4;
    spacingRight=4;overflow=hidden;rotatable=0;points=[[0,0.5],[1,0.5]];
    portConstraint=eastwest;fontSize=12;" vertex="1" parent="
    Lx7MsKEuWNhGFfsztCJ6-6">
    <mxGeometry y="52" width="160" height="26" as="geometry" />
</mxCell>
</object>
<object label="Manage Interfaces" uncertaintyDB="{"1623347564024"
    :{"id":"1623347564024","system":"System name"
    ","manifestation":"","environmentMonitor"
    ":"","nature":{"options":["
    Aleatory","Epistemic"],"value":"","
    perspective":{"options":["Subjective","
    Objective"],"value":"","source"
    :{"options":["External","Internal"],"
    value":"","viewpoint":{"options":["
    Logical","Physical","Process"],"value
    ":"","description":"Description","
    "evidence":"Evidence","SourceDescription":"
    ","relatedUncertainties":{},"location":"
    ","level":{"options":["Known Unknown"
    ","Unknown unknown","Statistical"],"value"
    ":"","awareness":{"options":["

```



```

Known Unknown","Unknown unknown"],"value":"
",","emergingTime":"options":"["
Requirement","Development","Runtime"],"
value":",","lifetime":",","
pattern":"options":"Periodic","
Persistence ","Transient","Sporadic"],"
value":",","measure":"options":"["
"Probability","Persistence ","Fuzziness","
"Temporal logic","Non-specificity"],"value"
:",",","related":",","operator":"{"
"options":"Modal","Temporal","Ordinal
","value":"Modal","modal":"{"
"options":"MAY","SHALL"],"value":"
",","ordinal":"options":"AS CLOSE AS
POSSIBLE TO &#39;&#39;","AS MANY POSSIBLE TO &#39;&#39;&
","AS FEW AS POSSIBLE TO &#39;&#39;","value":"
",","temporal":"options":"["EVENTUALLY
","UNTIL","BEFORE","AFTER","AS
EARLY AS POSSIBLE","AS LATE AS POSSIBLE","AS CLOSE
AS POSSIBLE TO &#39;&#39;","value":",","}" id=
"Lx7MsKEuWNhGFFsztCJ6-9">
<mxCell style="text;strokeColor=none;fillColor=none;spacingLeft=4;
spacingRight=4;overflow=hidden;rotatable=0;points=[[0,0.5],[1,0.5]];
portConstraint=eastwest;fontSize=12;" vertex="1" parent="
Lx7MsKEuWNhGFFsztCJ6-6">
<mxGeometry y="78" width="160" height="26" as="geometry" />
</mxCell>
</object>
<object label="Governance" uncertaintyDB="{"1623347515019":"
id":"1623347515019,"system":"System name","
manifestation":","environmentMonitor":"
","nature":"options":"Aleatory","
"Epistemic"],"value":","perspective"
:"options":"Subjective","Objective"
],"value":","source":"options"
:"External","Internal"],"value":"
",","viewpoint":"options":"Logical"
,"Physical","Process"],"value":"
",","description":"Description","evidence"
:"Evidence","SourceDescription":","
"relatedUncertainties":","location":","
"level":"options":"Known Unknown","
Unknown unknown","Statistical"],"value":"
","awareness":"options":"Known Unknown"
,"Unknown unknown"],"value":","
"emergingTime":"options":"Requirement","
Development","Runtime"],"value":","
"lifetime":",","pattern":"options"
:"Periodic","Persistence ","Transient","
"Sporadic"],"value":",","measure"
:"options":"Probability","Persistence "
,"Fuzziness","Temporal logic","Non-specificity
","value":",",","related":","
","operator":"options":"Modal","

```

```

Temporal&quot;;&quot;Ordinal&quot;],&quot;value&quot;:&quot;Modal&quot;
;,&quot;modal&quot;:&quot;options&quot;:[&quot;MAY&quot;;&quot;SHALL&quot;
&quot;],&quot;value&quot;:&quot;&quot;,&quot;ordinal&quot;:&quot;
options&quot;:[&quot;AS CLOSE AS POSSIBLE TO &#39;?&#39;&quot;;&quot;AS
MANY POSSIBLE TO &#39;?&#39;&quot;;&quot;AS FEW AS POSSIBLE TO
&#39;?&#39;&quot;],&quot;value&quot;:&quot;&quot;,&quot;temporal&quot;
:&quot;options&quot;:[&quot;EVENTUALLY&quot;;&quot;UNTIL&quot;;&quot;
BEFORE&quot;;&quot;AFTER&quot;;&quot;AS EARLY AS POSSIBLE&quot;;&quot;
LATE AS POSSIBLE&quot;;&quot;AS CLOSE AS POSSIBLE TO &#39;?&#39;&quot;
;,&quot;value&quot;:&quot;&quot;,&quot;&quot;}}" id="Lx7MsKEuWNhGFfsztCJ6-2">
<mxCell style="swimlane;fontStyle=0;childLayout=stackLayout;horizontal=1;
startSize=26;horizontalStack=0;resizeParent=1;resizeParentMax=0;
resizeLast=0;collapsible=1;marginBottom=0;align=center;fontSize=14;"
vertex="1" parent="Lx7MsKEuWNhGFfsztCJ6-1">
<mxGeometry x="-220" y="100" width="160" height="104" as="geometry" />
</mxCell>
</object>
<object label="Data Sharing Agreements" uncertaintyDB="{&quot;1623347531099&
quot;:&quot;id&quot;:&quot;1623347531099,&quot;system&quot;:&quot;System name
&quot;;&quot;manifestation&quot;:&quot;&quot;;&quot;environmentMonitor&
quot;:&quot;&quot;;&quot;nature&quot;:&quot;options&quot;:&quot;
Aleatory&quot;;&quot;Epistemic&quot;],&quot;value&quot;:&quot;&quot;,&quot;
quot;perspective&quot;:&quot;options&quot;:[&quot;Subjective&quot;;&
quot;Objective&quot;],&quot;value&quot;:&quot;&quot;,&quot;source&quot;
:&quot;options&quot;:[&quot;External&quot;;&quot;Internal&quot;],&quot;
value&quot;:&quot;&quot;,&quot;viewpoint&quot;:&quot;options&quot;:[&
quot;Logical&quot;;&quot;Physical&quot;;&quot;Process&quot;],&quot;value
&quot;:&quot;&quot;,&quot;description&quot;:&quot;Description&quot;;&
quot;evidence&quot;:&quot;Evidence&quot;;&quot;SourceDescription&quot;:&
quot;&quot;;&quot;relatedUncertainties&quot;:{},&quot;location&quot;:&
quot;&quot;;&quot;level&quot;:&quot;options&quot;:[&quot;Known Unknown&
quot;;&quot;Unknown unknown&quot;;&quot;Statistical&quot;],&quot;value&
quot;:&quot;&quot;,&quot;awareness&quot;:&quot;options&quot;:[&quot;
Known Unknown&quot;;&quot;Unknown unknown&quot;],&quot;value&quot;:&quot;
&quot;,&quot;emergingTime&quot;:&quot;options&quot;:[&quot;
Requirement&quot;;&quot;Development&quot;;&quot;Runtime&quot;],&quot;
value&quot;:&quot;&quot;,&quot;lifetime&quot;:&quot;&quot;;&quot;
pattern&quot;:&quot;options&quot;:[&quot;Periodic&quot;;&quot;
Persistence &quot;;&quot;Transient&quot;;&quot;Sporadic&quot;],&quot;
value&quot;:&quot;&quot;,&quot;measure&quot;:&quot;options&quot;:[&
quot;Probability&quot;;&quot;Persistence &quot;;&quot;Fuzziness&quot;;&
quot;Temporal logic&quot;;&quot;Non-specificity&quot;],&quot;value&quot;
:&quot;&quot;,&quot;related&quot;:&quot;&quot;;&quot;operator&quot;:&quot;
options&quot;:[&quot;Modal&quot;;&quot;Temporal&quot;;&quot;Ordinal
&quot;],&quot;value&quot;:&quot;Modal&quot;,&quot;modal&quot;:&quot;
options&quot;:[&quot;MAY&quot;;&quot;SHALL&quot;],&quot;value&quot;:&
quot;&quot;,&quot;ordinal&quot;:&quot;options&quot;:[&quot;AS CLOSE AS
POSSIBLE TO &#39;?&#39;&quot;;&quot;AS MANY POSSIBLE TO &#39;?&#39;&
quot;;&quot;AS FEW AS POSSIBLE TO &#39;?&#39;&quot;],&quot;value&quot;:&
quot;&quot;,&quot;temporal&quot;:&quot;options&quot;:[&quot;EVENTUALLY
&quot;;&quot;UNTIL&quot;;&quot;BEFORE&quot;;&quot;AFTER&quot;;&quot;AS
EARLY AS POSSIBLE&quot;;&quot;AS LATE AS POSSIBLE&quot;;&quot;AS CLOSE
AS POSSIBLE TO &#39;?&#39;&quot;],&quot;value&quot;:&quot;&quot;,&quot;&quot;}}" id=
"Lx7MsKEuWNhGFfsztCJ6-3">

```

```

<mxCell style="text;strokeColor=none;fillColor=none;spacingLeft=4;
    spacingRight=4;overflow=hidden;rotatable=0;points=[[0,0.5],[1,0.5]];
    portConstraint=eastwest;fontSize=12;" vertex="1" parent="
    Lx7MsKEuWNhGFfsztCJ6-2">
    <mxGeometry y="26" width="160" height="26" as="geometry" />
</mxCell>
</object>
<object label="Data Usage & Consent" uncertaintyDB="{&quot;
    ;1623347529998&quot;;:&quot;id&quot;;:1623347529998,&quot;;system&quot;;:&
    quot;;System name&quot;;,&quot;;manifestation&quot;;:&quot;;&quot;;,&quot;;
    environmentMonitor&quot;;:&quot;;&quot;;,&quot;;nature&quot;;:&quot;;options&
    quot;;:&quot;;Aleatory&quot;;,&quot;;Epistemic&quot;;,&quot;;value&quot;;:&
    quot;;&quot;;,&quot;;perspective&quot;;:&quot;;options&quot;;:&quot;;
    Subjective&quot;;,&quot;;Objective&quot;;,&quot;;value&quot;;:&quot;;&quot;
    ;,&quot;;source&quot;;:&quot;;options&quot;;:&quot;;External&quot;;,&quot;;
    Internal&quot;;,&quot;;value&quot;;:&quot;;&quot;;,&quot;;viewpoint&quot;;:&
    quot;;options&quot;;:&quot;;Logical&quot;;,&quot;;Physical&quot;;,&quot;;
    Process&quot;;,&quot;;value&quot;;:&quot;;&quot;;,&quot;;description&quot;;:&
    quot;;Description&quot;;,&quot;;evidence&quot;;:&quot;;Evidence&quot;;,&quot;;
    SourceDescription&quot;;:&quot;;&quot;;,&quot;;relatedUncertainties&quot;
    ;:&quot;;location&quot;;:&quot;;&quot;;,&quot;;level&quot;;:&quot;;options&
    quot;;:&quot;;Known Unknown&quot;;,&quot;;Unknown unknown&quot;;,&quot;;
    Statistical&quot;;,&quot;;value&quot;;:&quot;;&quot;;,&quot;;awareness&quot;
    ;:&quot;;options&quot;;:&quot;;Known Unknown&quot;;,&quot;;Unknown unknown&
    quot;;,&quot;;value&quot;;:&quot;;&quot;;,&quot;;emergingTime&quot;;:&quot;;
    options&quot;;:&quot;;Requirement&quot;;,&quot;;Development&quot;;,&quot;;
    Runtime&quot;;,&quot;;value&quot;;:&quot;;&quot;;,&quot;;lifetime&quot;;:&
    quot;;&quot;;,&quot;;pattern&quot;;:&quot;;options&quot;;:&quot;;Periodic&
    quot;;,&quot;;Persistence &quot;;,&quot;;Transient&quot;;,&quot;;Sporadic&quot;
    ;,&quot;;value&quot;;:&quot;;&quot;;,&quot;;measure&quot;;:&quot;;options&
    quot;;:&quot;;Probability&quot;;,&quot;;Persistence &quot;;,&quot;;Fuzziness&
    quot;;,&quot;;Temporal logic&quot;;,&quot;;Non-specificity&quot;;,&quot;;
    value&quot;;:&quot;;&quot;;,&quot;;related&quot;;:&quot;;&quot;;,&quot;;
    operator&quot;;:&quot;;options&quot;;:&quot;;Modal&quot;;,&quot;;Temporal&
    quot;;,&quot;;Ordinal&quot;;,&quot;;value&quot;;:&quot;;Modal&quot;;,&quot;;
    modal&quot;;:&quot;;options&quot;;:&quot;;MAY&quot;;,&quot;;SHALL&quot;;,&quot;;
    quot;;value&quot;;:&quot;;&quot;;,&quot;;ordinal&quot;;:&quot;;options&quot;
    ;:&quot;;AS CLOSE AS POSSIBLE TO &#39;?&#39;&quot;;,&quot;;AS MANY
    POSSIBLE TO &#39;?&#39;&quot;;,&quot;;AS FEW AS POSSIBLE TO &#39;?&#39;&
    quot;;,&quot;;value&quot;;:&quot;;&quot;;,&quot;;temporal&quot;;:&quot;;
    options&quot;;:&quot;;EVENTUALLY&quot;;,&quot;;UNTIL&quot;;,&quot;;BEFORE&
    quot;;,&quot;;AFTER&quot;;,&quot;;AS EARLY AS POSSIBLE&quot;;,&quot;;AS LATE
    AS POSSIBLE&quot;;,&quot;;AS CLOSE AS POSSIBLE TO &#39;?&#39;&quot;;,&quot;;
    value&quot;;:&quot;;&quot;;}}" id="Lx7MsKEuWNhGFfsztCJ6-4">
<mxCell style="text;strokeColor=none;fillColor=none;spacingLeft=4;
    spacingRight=4;overflow=hidden;rotatable=0;points=[[0,0.5],[1,0.5]];
    portConstraint=eastwest;fontSize=12;" vertex="1" parent="
    Lx7MsKEuWNhGFfsztCJ6-2">
    <mxGeometry y="52" width="160" height="26" as="geometry" />
</mxCell>
</object>
<object label="3rd Party Engagement Rules" uncertaintyDB="{&quot;
    ;1623347564024&quot;;:&quot;id&quot;;:1623347564024,&quot;;system&quot;;:&
    quot;;System name&quot;;,&quot;;manifestation&quot;;:&quot;;&quot;;,&quot;;

```

```

environmentMonitor&quot;;&quot;&quot;;&quot;nature&quot;:{&quot;options&
quot;:[&quot;Aleatory&quot;;&quot;Epistemic&quot;],&quot;value&quot;:&
quot;&quot;},&quot;perspective&quot;:{&quot;options&quot;:[&quot;
Subjective&quot;;&quot;Objective&quot;],&quot;value&quot;:&quot;&quot;
};&quot;source&quot;:{&quot;options&quot;:[&quot;External&quot;;&quot;
Internal&quot;],&quot;value&quot;:&quot;&quot;},&quot;viewpoint&quot;:{&
quot;options&quot;:[&quot;Logical&quot;;&quot;Physical&quot;;&quot;
Process&quot;],&quot;value&quot;:&quot;&quot;},&quot;description&quot;:&
quot;Description&quot;;&quot;evidence&quot;:&quot;Evidence&quot;;&quot;
SourceDescription&quot;:&quot;&quot;;&quot;relatedUncertainties&quot;
;:{},&quot;location&quot;:&quot;&quot;;&quot;level&quot;:{&quot;options&
quot;:[&quot;Known Unknown&quot;;&quot;Unknown unknown&quot;;&quot;
Statistical&quot;],&quot;value&quot;:&quot;&quot;},&quot;awareness&quot;
;:{&quot;options&quot;:[&quot;Known Unknown&quot;;&quot;Unknown unknown&
quot;],&quot;value&quot;:&quot;&quot;},&quot;emergingTime&quot;:{&quot;
options&quot;:[&quot;Requirement&quot;;&quot;Development&quot;;&quot;
Runtime&quot;],&quot;value&quot;:&quot;&quot;},&quot;lifetime&quot;:&
quot;&quot;;&quot;pattern&quot;:{&quot;options&quot;:[&quot;Periodic&
quot;;&quot;Persistence &quot;;&quot;Transient&quot;;&quot;Sporadic&quot;
;],&quot;value&quot;:&quot;&quot;},&quot;measure&quot;:{&quot;options&
quot;:[&quot;Probability&quot;;&quot;Persistence &quot;;&quot;Fuzziness&
quot;;&quot;Temporal logic&quot;;&quot;Non-specificity&quot;],&quot;
value&quot;:&quot;&quot;},&quot;related&quot;:&quot;&quot;;&quot;
operator&quot;:{&quot;options&quot;:[&quot;Modal&quot;;&quot;Temporal&
quot;;&quot;Ordinal&quot;],&quot;value&quot;:&quot;Modal&quot;},&quot;
modal&quot;:{&quot;options&quot;:[&quot;MAY&quot;;&quot;SHALL&quot;],&
quot;value&quot;:&quot;&quot;},&quot;ordinal&quot;:{&quot;options&quot;
:[&quot;AS CLOSE AS POSSIBLE TO &#39;?&#39;&quot;;&quot;AS MANY
POSSIBLE TO &#39;?&#39;&quot;;&quot;AS FEW AS POSSIBLE TO &#39;?&#39;&
quot;],&quot;value&quot;:&quot;&quot;},&quot;temporal&quot;:{&quot;
options&quot;:[&quot;EVENTUALLY&quot;;&quot;UNTIL&quot;;&quot;BEFORE&
quot;;&quot;AFTER&quot;;&quot;AS EARLY AS POSSIBLE&quot;;&quot;AS LATE
AS POSSIBLE&quot;;&quot;AS CLOSE AS POSSIBLE TO &#39;?&#39;&quot;],&quot;
value&quot;:&quot;&quot;}}}" id="Lx7MsKEuWNhGFfsztCJ6-5">
<mxCell style="text;strokeColor=none;fillColor=none;spacingLeft=4;
spacingRight=4;overflow=hidden;rotatable=0;points=[[0,0.5],[1,0.5]];
portConstraint=eastwest;fontSize=12;" vertex="1" parent="
Lx7MsKEuWNhGFfsztCJ6-2">
<mxGeometry y="78" width="160" height="26" as="geometry" />
</mxCell>
</object>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-123" value="" style="html=1;shadow=0;dashed
=0;align=center;verticalAlign=middle;shape=mxgraph.arrows2.arrow;dy=0.6;
dx=40;notch=0;strokeColor=#000000;strokeWidth=2;fillColor=#ffffff;"
vertex="1" parent="Lx7MsKEuWNhGFfsztCJ6-1">
<mxGeometry x="-60" y="546" width="200" height="30" as="geometry" />
</mxCell>
<object label="Dat Stewards" uncertaintyDB="{&quot;1623347515019&quot;:{&
quot;id&quot;:1623347515019,&quot;system&quot;:&quot;System name&quot;;&
quot;manifestation&quot;:&quot;&quot;;&quot;environmentMonitor&quot;:&
quot;&quot;;&quot;nature&quot;:{&quot;options&quot;:[&quot;Aleatory&quot;
,&quot;Epistemic&quot;],&quot;value&quot;:&quot;&quot;},&quot;
perspective&quot;:{&quot;options&quot;:[&quot;Subjective&quot;;&quot;
Objective&quot;],&quot;value&quot;:&quot;&quot;},&quot;source&quot;:{&

```

```

    "options": [{"External": true, "Internal": true}], "value": [{"Logical": true, "Physical": true, "Process": true}], "description": "Evidence", "relatedUncertainties": {}, "location": {"level": [{"Known": true, "Unknown": true}], "Statistical": true}, "awareness": [{"Known": true, "Unknown": true}], "emergingTime": [{"Requirement": true, "Development": true, "Runtime": true}], "lifetime": {"pattern": [{"Periodic": true}], "Persistence": [{"Transient": true, "Sporadic": true}], "measure": [{"Probability": true, "Persistence": true, "Fuzziness": true, "Temporal logic": true, "Non-specificity": true}], "operator": [{"options": [{"Modal": true, "Temporal": true, "Ordinal": true}], "value": [{"MAY": true, "SHALL": true}], "ordinal": true}], "options": [{"AS POSSIBLE TO": true, "AS MANY POSSIBLE TO": true, "AS FEW AS POSSIBLE TO": true, "UNTIL": true, "BEFORE": true, "AFTER": true, "AS EARLY AS POSSIBLE": true, "AS LATE AS POSSIBLE": true, "AS CLOSE AS POSSIBLE TO": true}], "id": "Lx7MsKEuWNhGFfsztCJ6-10">
<mxCell style="swimlane;fontStyle=0;childLayout=stackLayout;horizontal=1;
    startSize=26;horizontalStack=0;resizeParent=1;resizeParentMax=0;
    resizeLast=0;collapsible=1;marginBottom=0;align=center;fontSize=14;"
    vertex="1" parent="Lx7MsKEuWNhGFfsztCJ6-1">
    <mxGeometry x="-220" y="502" width="160" height="104" as="geometry" />
</mxCell>
</object>
<object label="Data Catalog" uncertaintyDB="{"1623347531099":{"id":1623347531099,"system":"System name","manifestation":"environmentMonitor","nature":{"options":["Aleatory","Epistemic"]},"value":"perspective":{"options":["Subjective","Objective"]},"source":{"options":["External","Internal"]},"viewpoint":{"options":["Logical","Physical","Process"]},"value":"description":"Evidence","SourceDescription":"relatedUncertainties":{},"location":"level":{"options":["Known","Unknown"]},"Statistical":"value":"awareness":{"options":["Known","Unknown"]},"emergingTime":"Requirement","Development","Runtime"},"lifetime":{"pattern":["Periodic"],"Persistence":["Transient","Sporadic"],"measure":["Probability","Persistence","Fuzziness","Temporal logic","Non-specificity"]},"operator":["options":["Modal","Temporal","Ordinal"],"value":["MAY","SHALL"],"ordinal"],"options":["AS POSSIBLE TO","AS MANY POSSIBLE TO","AS FEW AS POSSIBLE TO","UNTIL","BEFORE","AFTER","AS EARLY AS POSSIBLE","AS LATE AS POSSIBLE","AS CLOSE AS POSSIBLE TO"]}}}" id=

```

```

    &quot;,&quot;emergingTime&quot;:{&quot;options&quot;:[&quot;
Requirement&quot;,&quot;Development&quot;,&quot;Runtime&quot;],&quot;
value&quot;:&quot;&quot;,&quot;lifetime&quot;:&quot;&quot;,&quot;
pattern&quot;:{&quot;options&quot;:[&quot;Periodic&quot;,&quot;
Persistence &quot;,&quot;Transient&quot;,&quot;Sporadic&quot;],&quot;
value&quot;:&quot;&quot;,&quot;measure&quot;:{&quot;options&quot;:[&
quot;Probability&quot;,&quot;Persistence &quot;,&quot;Fuzziness&quot;,&
quot;Temporal logic&quot;,&quot;Non-specificity&quot;],&quot;value&quot;
:&quot;&quot;,&quot;related&quot;:&quot;&quot;,&quot;operator&quot;:{&
quot;options&quot;:[&quot;Modal&quot;,&quot;Temporal&quot;,&quot;Ordinal
&quot;],&quot;value&quot;:&quot;&quot;,&quot;modal&quot;:{&quot;
options&quot;:[&quot;MAY&quot;,&quot;SHALL&quot;],&quot;value&quot;:&
quot;&quot;,&quot;ordinal&quot;:{&quot;options&quot;:[&quot;AS CLOSE AS
POSSIBLE TO &#39;?&#39;&quot;,&quot;AS MANY POSSIBLE TO &#39;?&#39;&
quot;,&quot;AS FEW AS POSSIBLE TO &#39;?&#39;&quot;],&quot;value&quot;:&quot;
&quot;,&quot;temporal&quot;:{&quot;options&quot;:[&quot;EVENTUALLY
&quot;,&quot;UNTIL&quot;,&quot;BEFORE&quot;,&quot;AFTER&quot;,&quot;AS
EARLY AS POSSIBLE&quot;,&quot;AS LATE AS POSSIBLE&quot;,&quot;AS CLOSE
AS POSSIBLE TO &#39;?&#39;&quot;],&quot;value&quot;:&quot;&quot;}}}" id=
"Lx7MsKEuWNhGFfsztCJ6-11">
<mxCell style="text;strokeColor=none;fillColor=none;spacingLeft=4;
    spacingRight=4;overflow=hidden;rotatable=0;points=[[0,0.5],[1,0.5]];
    portConstraint=eastwest;fontSize=12;" vertex="1" parent="
    Lx7MsKEuWNhGFfsztCJ6-10">
    <mxGeometry y="26" width="160" height="26" as="geometry" />
</mxCell>
</object>
<object label="Curate Data" uncertaintyDB="{&quot;1623347529998&quot;:{&quot;
id&quot;:1623347529998,&quot;system&quot;:&quot;System name&quot;,&quot;
manifestation&quot;:&quot;&quot;,&quot;environmentMonitor&quot;:&quot;&
quot;,&quot;nature&quot;:{&quot;options&quot;:[&quot;Aleatory&quot;,&
quot;Epistemic&quot;],&quot;value&quot;:&quot;&quot;,&quot;perspective&
quot;:{&quot;options&quot;:[&quot;Subjective&quot;,&quot;Objective&quot;
],&quot;value&quot;:&quot;&quot;,&quot;source&quot;:{&quot;options&
quot;:[&quot;External&quot;,&quot;Internal&quot;],&quot;value&quot;:&
quot;&quot;,&quot;viewpoint&quot;:{&quot;options&quot;:[&quot;Logical&
quot;,&quot;Physical&quot;,&quot;Process&quot;],&quot;value&quot;:&quot;
&quot;,&quot;description&quot;:&quot;Description&quot;,&quot;evidence&
quot;:&quot;Evidence&quot;,&quot;SourceDescription&quot;:&quot;&quot;,&
quot;relatedUncertainties&quot;:{&quot;,&quot;location&quot;:&quot;&quot;,&
quot;level&quot;:{&quot;options&quot;:[&quot;Known Unknown&quot;,&quot;
Unknown unknown&quot;,&quot;Statistical&quot;],&quot;value&quot;:&quot;&
quot;,&quot;awareness&quot;:{&quot;options&quot;:[&quot;Known Unknown&
quot;,&quot;Unknown unknown&quot;],&quot;value&quot;:&quot;&quot;,&quot;
emergingTime&quot;:{&quot;options&quot;:[&quot;Requirement&quot;,&quot;
Development&quot;,&quot;Runtime&quot;],&quot;value&quot;:&quot;&quot;,&quot;
lifetime&quot;:&quot;&quot;,&quot;pattern&quot;:{&quot;options&quot;
:[&quot;Periodic&quot;,&quot;Persistence &quot;,&quot;Transient&quot;,&
quot;Sporadic&quot;],&quot;value&quot;:&quot;&quot;,&quot;measure&quot;
:&quot;options&quot;:[&quot;Probability&quot;,&quot;Persistence &quot;
,&quot;Fuzziness&quot;,&quot;Temporal logic&quot;,&quot;Non-specificity
&quot;],&quot;value&quot;:&quot;&quot;,&quot;related&quot;:&quot;&quot;
,&quot;operator&quot;:{&quot;options&quot;:[&quot;Modal&quot;,&quot;
Temporal&quot;,&quot;Ordinal&quot;],&quot;value&quot;:&quot;Modal&quot;

```



```

;],&quot;modal&quot;;{&quot;options&quot;;[&quot;MAY&quot;;&quot;SHALL&
quot;],&quot;value&quot;;&quot;&quot;}&quot;;&quot;ordinal&quot;;{&quot;
options&quot;;[&quot;AS CLOSE AS POSSIBLE TO &#39;?&#39;&quot;;&quot;AS
MANY POSSIBLE TO &#39;?&#39;&quot;;&quot;AS FEW AS POSSIBLE TO
&#39;?&#39;&quot;],&quot;value&quot;;&quot;&quot;}&quot;;&quot;temporal&quot;
;:{&quot;options&quot;;[&quot;EVENTUALLY&quot;;&quot;UNTIL&quot;;&quot;
BEFORE&quot;;&quot;AFTER&quot;;&quot;AS EARLY AS POSSIBLE&quot;;&quot;AS
LATE AS POSSIBLE&quot;;&quot;AS CLOSE AS POSSIBLE TO &#39;?&#39;&quot;
;],&quot;value&quot;;&quot;&quot;}}}" id="Lx7MsKEuWNhGFfsztCJ6-12">
<mxCell style="text;strokeColor=none;fillColor=none;spacingLeft=4;
spacingRight=4;overflow=hidden;rotatable=0;points=[[0,0.5],[1,0.5]];
portConstraint=eastwest;fontSize=12;" vertex="1" parent="
Lx7MsKEuWNhGFfsztCJ6-10">
  <mxGeometry y="52" width="160" height="26" as="geometry" />
</mxCell>
</object>
<object label="Manage Data" uncertaintyDB="{&quot;1623347564024&quot;;{&quot;
id&quot;;1623347564024,&quot;system&quot;;&quot;System name&quot;;&quot;
manifestation&quot;;&quot;&quot;;&quot;environmentMonitor&quot;;&quot;&
quot;;&quot;nature&quot;;{&quot;options&quot;;[&quot;Aleatory&quot;;&
quot;Epistemic&quot;],&quot;value&quot;;&quot;&quot;}&quot;;&quot;perspective&
quot;;{&quot;options&quot;;[&quot;Subjective&quot;;&quot;Objective&quot;
;],&quot;value&quot;;&quot;&quot;}&quot;;&quot;source&quot;;{&quot;options&
quot;;[&quot;External&quot;;&quot;Internal&quot;],&quot;value&quot;;&
quot;&quot;}&quot;;&quot;viewpoint&quot;;{&quot;options&quot;;[&quot;Logical&
quot;;&quot;Physical&quot;;&quot;Process&quot;],&quot;value&quot;;&quot;
&quot;}&quot;;&quot;description&quot;;&quot;Description&quot;;&quot;evidence&
quot;;&quot;Evidence&quot;;&quot;SourceDescription&quot;;&quot;&quot;;&
quot;relatedUncertainties&quot;;{}&quot;;&quot;location&quot;;&quot;&quot;;&
quot;level&quot;;{&quot;options&quot;;[&quot;Known Unknown&quot;;&quot;
Unknown unknown&quot;;&quot;Statistical&quot;],&quot;value&quot;;&quot;&
quot;}&quot;;&quot;awareness&quot;;{&quot;options&quot;;[&quot;Known Unknown&
quot;;&quot;Unknown unknown&quot;],&quot;value&quot;;&quot;&quot;}&quot;;&quot;
emergingTime&quot;;{&quot;options&quot;;[&quot;Requirement&quot;;&quot;
Development&quot;;&quot;Runtime&quot;],&quot;value&quot;;&quot;&quot;}&quot;;&
quot;lifetime&quot;;&quot;&quot;;&quot;pattern&quot;;{&quot;options&quot;
;[&quot;Periodic&quot;;&quot;Persistence &quot;;&quot;Transient&quot;;&
quot;Sporadic&quot;],&quot;value&quot;;&quot;&quot;}&quot;;&quot;measure&quot;
;{&quot;options&quot;;[&quot;Probability&quot;;&quot;Persistence &quot;
;&quot;Fuzziness&quot;;&quot;Temporal logic&quot;;&quot;Non-specificity
&quot;],&quot;value&quot;;&quot;&quot;}&quot;;&quot;related&quot;;&quot;&quot;
;&quot;operator&quot;;{&quot;options&quot;;[&quot;Modal&quot;;&quot;
Temporal&quot;;&quot;Ordinal&quot;],&quot;value&quot;;&quot;Modal&quot;
;}&quot;;&quot;modal&quot;;{&quot;options&quot;;[&quot;MAY&quot;;&quot;SHALL&
quot;],&quot;value&quot;;&quot;&quot;}&quot;;&quot;ordinal&quot;;{&quot;
options&quot;;[&quot;AS CLOSE AS POSSIBLE TO &#39;?&#39;&quot;;&quot;AS
MANY POSSIBLE TO &#39;?&#39;&quot;;&quot;AS FEW AS POSSIBLE TO
&#39;?&#39;&quot;],&quot;value&quot;;&quot;&quot;}&quot;;&quot;temporal&quot;
;:{&quot;options&quot;;[&quot;EVENTUALLY&quot;;&quot;UNTIL&quot;;&quot;
BEFORE&quot;;&quot;AFTER&quot;;&quot;AS EARLY AS POSSIBLE&quot;;&quot;AS
LATE AS POSSIBLE&quot;;&quot;AS CLOSE AS POSSIBLE TO &#39;?&#39;&quot;
;],&quot;value&quot;;&quot;&quot;}}}" id="Lx7MsKEuWNhGFfsztCJ6-13">
<mxCell style="text;strokeColor=none;fillColor=none;spacingLeft=4;
spacingRight=4;overflow=hidden;rotatable=0;points=[[0,0.5],[1,0.5]];

```

```

        portConstraint=eastwest;fontSize=12;" vertex="1" parent="
        Lx7MsKEuWNhGFfsztCJ6-10">
    <mxGeometry y="78" width="160" height="26" as="geometry" />
</mxCell>
</object>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-124" value="" style="html=1;shadow=0;dashed
=0;align=center;verticalAlign=middle;shape=mxgraph.arrows2.arrow;dy=0.6;
dx=40;notch=0;strokeColor=#000000;strokeWidth=2;fillColor=#ffffff;"
vertex="1" parent="Lx7MsKEuWNhGFfsztCJ6-1">
    <mxGeometry x="-60" y="353" width="70" height="30" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-125" value="" style="html=1;shadow=0;dashed
=0;align=center;verticalAlign=middle;shape=mxgraph.arrows2.arrow;dy=0.6;
dx=40;flipH=1;notch=0;strokeColor=#000000;strokeWidth=2;fillColor=#
ffffff;" vertex="1" parent="Lx7MsKEuWNhGFfsztCJ6-1">
    <mxGeometry x="810" y="155.5" width="110" height="33" as="geometry" />
</mxCell>
<object label="Data Analysis" uncertaintyDB="{&quot;1623347515019&quot;;:&{&
quot;id&quot;;:1623347515019,&quot;;system&quot;;:&quot;;System name&quot;;,&
quot;;manifestation&quot;;:&quot;;&quot;;,&quot;;environmentMonitor&quot;;:&
quot;;&quot;;,&quot;;nature&quot;;:&{&quot;;options&quot;;:[&quot;;Aleatory&quot;
;&quot;;Epistemic&quot;;],&quot;;value&quot;;:&quot;;&quot;;,&quot;;perspective&quot;;:&{&quot;;options&quot;;:[&quot;;Subjective&quot;;,&quot;;
Objective&quot;;],&quot;;value&quot;;:&quot;;&quot;;,&quot;;source&quot;;:&{&
quot;;options&quot;;:[&quot;;External&quot;;,&quot;;Internal&quot;;],&quot;;
value&quot;;:&quot;;&quot;;,&quot;;viewpoint&quot;;:&{&quot;;options&quot;;:[&
quot;;Logical&quot;;,&quot;;Physical&quot;;,&quot;;Process&quot;;],&quot;;value
&quot;;:&quot;;&quot;;,&quot;;description&quot;;:&quot;;Description&quot;;,&
quot;;evidence&quot;;:&quot;;Evidence&quot;;,&quot;;SourceDescription&quot;;:&
quot;;&quot;;,&quot;;relatedUncertainties&quot;;:&{&quot;;location&quot;;:&
quot;;&quot;;,&quot;;level&quot;;:&{&quot;;options&quot;;:[&quot;;Known Unknown&
quot;;,&quot;;Unknown unknown&quot;;,&quot;;Statistical&quot;;],&quot;;value&
quot;;:&quot;;&quot;;,&quot;;awareness&quot;;:&{&quot;;options&quot;;:[&quot;;
Known Unknown&quot;;,&quot;;Unknown unknown&quot;;],&quot;;value&quot;;:&quot;
&quot;;,&quot;;emergingTime&quot;;:&{&quot;;options&quot;;:[&quot;;
Requirement&quot;;,&quot;;Development&quot;;,&quot;;Runtime&quot;;],&quot;;
value&quot;;:&quot;;&quot;;,&quot;;lifetime&quot;;:&quot;;&quot;;,&quot;;
pattern&quot;;:&{&quot;;options&quot;;:[&quot;;Periodic&quot;;,&quot;;
Persistence &quot;;,&quot;;Transient&quot;;,&quot;;Sporadic&quot;;],&quot;;
value&quot;;:&quot;;&quot;;,&quot;;measure&quot;;:&{&quot;;options&quot;;:[&
quot;;Probability&quot;;,&quot;;Persistence &quot;;,&quot;;Fuzziness&quot;;,&
quot;;Temporal logic&quot;;,&quot;;Non-specificity&quot;;],&quot;;value&quot;
:&quot;;&quot;;,&quot;;related&quot;;:&quot;;&quot;;,&quot;;operator&quot;;:&{&
quot;;options&quot;;:[&quot;;Modal&quot;;,&quot;;Temporal&quot;;,&quot;;Ordinal
&quot;;],&quot;;value&quot;;:&quot;;Modal&quot;;,&quot;;modal&quot;;:&{&quot;;
options&quot;;:[&quot;;MAY&quot;;,&quot;;SHALL&quot;;],&quot;;value&quot;;:&
quot;;&quot;;,&quot;;ordinal&quot;;:&{&quot;;options&quot;;:[&quot;;AS CLOSE AS
POSSIBLE TO &#39;?&#39;&quot;;,&quot;;AS MANY POSSIBLE TO &#39;?&#39;&
quot;;,&quot;;AS FEW AS POSSIBLE TO &#39;?&#39;&quot;;],&quot;;value&quot;;:&
quot;;&quot;;,&quot;;temporal&quot;;:&{&quot;;options&quot;;:[&quot;;EVENTUALLY
&quot;;,&quot;;UNTIL&quot;;,&quot;;BEFORE&quot;;,&quot;;AFTER&quot;;,&quot;;AS
EARLY AS POSSIBLE&quot;;,&quot;;AS LATE AS POSSIBLE&quot;;,&quot;;AS CLOSE
AS POSSIBLE TO &#39;?&#39;&quot;;],&quot;;value&quot;;:&quot;;&quot;;}}}" id=
"Lx7MsKEuWNhGFfsztCJ6-14">

```



```

<mxCell style="swimlane;fontStyle=0;childLayout=stackLayout;horizontal=1;
    startSize=26;horizontalStack=0;resizeParent=1;resizeParentMax=0;
    resizeLast=0;collapsible=1;marginBottom=0;align=center;fontSize=14;"
    vertex="1" parent="Lx7MsKEuWNhGFfsztCJ6-1">
    <mxGeometry x="920" y="120" width="160" height="104" as="geometry" />
</mxCell>
</object>
<object label="Data Consumers" uncertaintyDB="{&quot;1623347531099&quot;;{&
    quot;id&quot;;:1623347531099,&quot;;system&quot;;:&quot;;System name&quot;;,&
    quot;;manifestation&quot;;:&quot;;&quot;;,&quot;;environmentMonitor&quot;;:&
    quot;;&quot;;,&quot;;nature&quot;;:{&quot;;options&quot;;:[&quot;;Aleatory&quot;
    ;,&quot;;Epistemic&quot;;],&quot;;value&quot;;:&quot;;&quot;;}&quot;;,&quot;;
    perspective&quot;;:{&quot;;options&quot;;:[&quot;;Subjective&quot;;,&quot;;
    Objective&quot;;],&quot;;value&quot;;:&quot;;&quot;;}&quot;;,&quot;;source&quot;;:{&
    quot;;options&quot;;:[&quot;;External&quot;;,&quot;;Internal&quot;;],&quot;;
    value&quot;;:&quot;;&quot;;}&quot;;,&quot;;viewpoint&quot;;:{&quot;;options&quot;;:[&
    quot;;Logical&quot;;,&quot;;Physical&quot;;,&quot;;Process&quot;;],&quot;;value
    &quot;;:&quot;;&quot;;}&quot;;,&quot;;description&quot;;:&quot;;Description&quot;;,&
    quot;;evidence&quot;;:&quot;;Evidence&quot;;,&quot;;SourceDescription&quot;;:&
    quot;;&quot;;,&quot;;relatedUncertainties&quot;;:{&quot;;,&quot;;location&quot;;:&
    quot;;&quot;;,&quot;;level&quot;;:{&quot;;options&quot;;:[&quot;;Known Unknown&
    quot;;,&quot;;Unknown unknown&quot;;,&quot;;Statistical&quot;;],&quot;;value&
    quot;;:&quot;;&quot;;}&quot;;,&quot;;awareness&quot;;:{&quot;;options&quot;;:[&quot;;
    Known Unknown&quot;;,&quot;;Unknown unknown&quot;;],&quot;;value&quot;;:&quot;;
    &quot;;}&quot;;,&quot;;emergingTime&quot;;:{&quot;;options&quot;;:[&quot;;
    Requirement&quot;;,&quot;;Development&quot;;,&quot;;Runtime&quot;;],&quot;;
    value&quot;;:&quot;;&quot;;}&quot;;,&quot;;lifetime&quot;;:&quot;;&quot;;,&quot;;
    pattern&quot;;:{&quot;;options&quot;;:[&quot;;Periodic&quot;;,&quot;;
    Persistence &quot;;,&quot;;Transient&quot;;,&quot;;Sporadic&quot;;],&quot;;
    value&quot;;:&quot;;&quot;;}&quot;;,&quot;;measure&quot;;:{&quot;;options&quot;;:[&
    quot;;Probability&quot;;,&quot;;Persistence &quot;;,&quot;;Fuzziness&quot;;,&
    quot;;Temporal logic&quot;;,&quot;;Non-specificity&quot;;],&quot;;value&quot;
    ;:&quot;;&quot;;}&quot;;,&quot;;related&quot;;:&quot;;&quot;;,&quot;;operator&quot;;:{&
    quot;;options&quot;;:[&quot;;Modal&quot;;,&quot;;Temporal&quot;;,&quot;;Ordinal
    &quot;;],&quot;;value&quot;;:&quot;;Modal&quot;;}&quot;;,&quot;;modal&quot;;:{&quot;;
    options&quot;;:[&quot;;MAY&quot;;,&quot;;SHALL&quot;;],&quot;;value&quot;;:&
    quot;;&quot;;}&quot;;,&quot;;ordinal&quot;;:{&quot;;options&quot;;:[&quot;;AS CLOSE AS
    POSSIBLE TO &#39;?&#39;&quot;;,&quot;;AS MANY POSSIBLE TO &#39;?&#39;&
    quot;;,&quot;;AS FEW AS POSSIBLE TO &#39;?&#39;&quot;;],&quot;;value&quot;;:&
    quot;;&quot;;}&quot;;,&quot;;temporal&quot;;:{&quot;;options&quot;;:[&quot;;EVENTUALLY
    &quot;;,&quot;;UNTIL&quot;;,&quot;;BEFORE&quot;;,&quot;;AFTER&quot;;,&quot;;AS
    EARLY AS POSSIBLE&quot;;,&quot;;AS LATE AS POSSIBLE&quot;;,&quot;;AS CLOSE
    AS POSSIBLE TO &#39;?&#39;&quot;;],&quot;;value&quot;;:&quot;;&quot;;}}}" id=
    "Lx7MsKEuWNhGFfsztCJ6-15">
<mxCell style="text;strokeColor=none;fillColor=none;spacingLeft=4;
    spacingRight=4;overflow=hidden;rotatable=0;points=[[0,0.5],[1,0.5]];
    portConstraint=eastwest;fontSize=12;" vertex="1" parent="
    Lx7MsKEuWNhGFfsztCJ6-14">
    <mxGeometry y="26" width="160" height="26" as="geometry" />
</mxCell>
</object>
<object label="Deliver Insights" uncertaintyDB="{&quot;1623347529998&quot;
    ;:&quot;;id&quot;;:1623347529998,&quot;;system&quot;;:&quot;;System name&
    quot;;,&quot;;manifestation&quot;;:&quot;;&quot;;,&quot;;environmentMonitor&

```

```

    quote;:&quot;&quot;&quot;,&quot;nature&quot;:&quot;options&quot;:&quot;[&quot;
    Aleatory&quot;,&quot;Epistemic&quot;],&quot;value&quot;:&quot;&quot;,&quot;
    quote;perspective&quot;:&quot;options&quot;:&quot;[&quot;Subjective&quot;,&
    quote;Objective&quot;],&quot;value&quot;:&quot;&quot;,&quot;source&quot;
    ;:&quot;options&quot;:&quot;[&quot;External&quot;,&quot;Internal&quot;],&quot;
    ;value&quot;:&quot;&quot;,&quot;viewpoint&quot;:&quot;options&quot;:&quot;[&
    quote;Logical&quot;,&quot;Physical&quot;,&quot;Process&quot;],&quot;value
    &quot;:&quot;&quot;,&quot;description&quot;:&quot;Description&quot;,&
    quote;evidence&quot;:&quot;Evidence&quot;,&quot;SourceDescription&quot;:&
    quote;&quot;,&quot;relatedUncertainties&quot;:&quot;{}&quot;,&quot;location&quot;:&
    quote;&quot;,&quot;level&quot;:&quot;options&quot;:&quot;[&quot;Known Unknown&
    quote;,&quot;Unknown unknown&quot;,&quot;Statistical&quot;],&quot;value&
    quote;:&quot;&quot;,&quot;awareness&quot;:&quot;options&quot;:&quot;[&quot;
    Known Unknown&quot;,&quot;Unknown unknown&quot;],&quot;value&quot;:&quot;
    &quot;,&quot;emergingTime&quot;:&quot;options&quot;:&quot;[&quot;
    Requirement&quot;,&quot;Development&quot;,&quot;Runtime&quot;],&quot;
    value&quot;:&quot;&quot;,&quot;lifetime&quot;:&quot;&quot;,&quot;
    pattern&quot;:&quot;options&quot;:&quot;[&quot;Periodic&quot;,&quot;
    Persistence &quot;,&quot;Transient&quot;,&quot;Sporadic&quot;],&quot;
    value&quot;:&quot;&quot;,&quot;measure&quot;:&quot;options&quot;:&quot;[&
    quote;Probability&quot;,&quot;Persistence &quot;,&quot;Fuzziness&quot;,&
    quote;Temporal logic&quot;,&quot;Non-specificity&quot;],&quot;value&quot;
    ;:&quot;&quot;,&quot;related&quot;:&quot;&quot;,&quot;operator&quot;:&quot;{&
    quote;options&quot;:&quot;[&quot;Modal&quot;,&quot;Temporal&quot;,&quot;Ordinal
    &quot;],&quot;value&quot;:&quot;Modal&quot;,&quot;modal&quot;:&quot;
    options&quot;:&quot;[&quot;MAY&quot;,&quot;SHALL&quot;],&quot;value&quot;:&
    quote;&quot;,&quot;ordinal&quot;:&quot;options&quot;:&quot;[&quot;AS CLOSE AS
    POSSIBLE TO &#39;&#39;&quot;,&quot;AS MANY POSSIBLE TO &#39;&#39;&
    quote;,&quot;AS FEW AS POSSIBLE TO &#39;&#39;&quot;],&quot;value&quot;:&
    quote;&quot;,&quot;temporal&quot;:&quot;options&quot;:&quot;[&quot;EVENTUALLY
    &quot;,&quot;UNTIL&quot;,&quot;BEFORE&quot;,&quot;AFTER&quot;,&quot;AS
    EARLY AS POSSIBLE&quot;,&quot;AS LATE AS POSSIBLE&quot;,&quot;AS CLOSE
    AS POSSIBLE TO &#39;&#39;&quot;],&quot;value&quot;:&quot;&quot;}}}" id=
    "Lx7MsKEuWNhGFfsztCJ6-16">
<mxCell style="text;strokeColor=none;fillColor=none;spacingLeft=4;
    spacingRight=4;overflow=hidden;rotatable=0;points=[[0,0.5],[1,0.5]];
    portConstraint=eastwest;fontSize=12;" vertex="1" parent="
    Lx7MsKEuWNhGFfsztCJ6-14">
    <mxGeometry y="52" width="160" height="26" as="geometry" />
</mxCell>
</object>
<object label="Recommended Actions" uncertaintyDB="{&quot;1623347564024&quot;
    ;:&quot;id&quot;:&quot;1623347564024,&quot;system&quot;:&quot;System name&
    quote;,&quot;manifestation&quot;:&quot;&quot;,&quot;environmentMonitor&
    quote;:&quot;&quot;,&quot;nature&quot;:&quot;options&quot;:&quot;[&quot;
    Aleatory&quot;,&quot;Epistemic&quot;],&quot;value&quot;:&quot;&quot;,&quot;
    quote;perspective&quot;:&quot;options&quot;:&quot;[&quot;Subjective&quot;,&
    quote;Objective&quot;],&quot;value&quot;:&quot;&quot;,&quot;source&quot;
    ;:&quot;options&quot;:&quot;[&quot;External&quot;,&quot;Internal&quot;],&quot;
    ;value&quot;:&quot;&quot;,&quot;viewpoint&quot;:&quot;options&quot;:&quot;[&
    quote;Logical&quot;,&quot;Physical&quot;,&quot;Process&quot;],&quot;value
    &quot;:&quot;&quot;,&quot;description&quot;:&quot;Description&quot;,&
    quote;evidence&quot;:&quot;Evidence&quot;,&quot;SourceDescription&quot;:&
    quote;&quot;,&quot;relatedUncertainties&quot;:&quot;{}&quot;,&quot;location&quot;:&

```

```

    "known": "Known", "unknown": "Unknown", "level": "Level", "options": "Options", "value": "Value", "awareness": "Awareness", "emergingTime": "Emerging Time", "requirement": "Requirement", "development": "Development", "runtime": "Runtime", "lifetime": "Lifetime", "pattern": "Pattern", "persistence": "Persistence", "transient": "Transient", "sporadic": "Sporadic", "measure": "Measure", "probability": "Probability", "persistence": "Persistence", "fuzziness": "Fuzziness", "temporalLogic": "Temporal Logic", "nonSpecificity": "Non-specificity", "related": "Related", "operator": "Operator", "modal": "Modal", "temporal": "Temporal", "ordinal": "Ordinal", "may": "MAY", "shall": "SHALL", "asCloseAsPossibleTo": "AS CLOSE AS POSSIBLE TO", "asManyAsPossibleTo": "AS MANY AS POSSIBLE TO", "asFewAsPossibleTo": "AS FEW AS POSSIBLE TO", "until": "UNTIL", "before": "BEFORE", "after": "AFTER", "asEarlyAsPossible": "AS EARLY AS POSSIBLE", "asLateAsPossible": "AS LATE AS POSSIBLE", "asCloseAsPossibleTo": "AS CLOSE AS POSSIBLE TO"}" id="Lx7MsKEuWNhGffszCJ6-17">
<mxCell style="text;strokeColor=none;fillColor=none;spacingLeft=4;spacingRight=4;overflow=hidden;rotatable=0;points=[[0,0.5],[1,0.5]];portConstraint=eastwest;fontSize=12;" vertex="1" parent="Lx7MsKEuWNhGffszCJ6-14">
<mxGeometry y="78" width="160" height="26" as="geometry" />
</mxCell>
</object>
<object label="Data Scientists" uncertaintyDB="{"1623347515019":{"id":1623347515019,"system":"System name","manifestation":"","environmentMonitor":"","nature":{"options":["Aleatory","Epistemic"]},"value":"","perspective":{"options":["Subjective","Objective"]},"value":"","source":{"options":["External","Internal"]},"value":"","viewpoint":{"options":["Logical","Physical","Process"]},"value":"","description":"","evidence":"","Evidence":"","SourceDescription":"","relatedUncertainties":{},"location":"","level":{"options":["Known Unknown","Unknown unknown","Statistical"]},"value":"","awareness":{"options":["Known Unknown","Unknown unknown"]},"value":"","emergingTime":{"options":["Requirement","Development","Runtime"]},"value":"","lifetime":"","pattern":{"options":["Periodic","Persistence","Transient","Sporadic"]},"value":"","measure":{"options":["Probability","Persistence","Fuzziness","Temporal logic","Non-specificity"]},"value":"","related":"","operator":{"options":["Modal","Temporal","Ordinal"]},"value":"","modal":{"options":["MAY","SHALL"]},"value":"","ordinal":{"options":["AS CLOSE AS POSSIBLE TO","AS MANY AS POSSIBLE TO","AS FEW AS POSSIBLE TO"]},"value":"","temporal":{"options":["EVENTUALLY","UNTIL","BEFORE","AFTER","AS EARLY AS POSSIBLE","AS LATE AS POSSIBLE","AS CLOSE AS POSSIBLE TO"]}}" id="Lx7MsKEuWNhGffszCJ6-17">

```

```

        quot;Probability&quot;;,&quot;Persistence &quot;;,&quot;Fuzziness&quot;;,&
        quot;Temporal logic&quot;;,&quot;Non-specificity&quot;],&quot;value&quot;
        ;:&quot;&quot;,&quot;related&quot;:&quot;&quot;;,&quot;operator&quot;:&quot;{&
        quot;options&quot;:&quot;Modal&quot;;,&quot;Temporal&quot;;,&quot;Ordinal
        &quot;],&quot;value&quot;:&quot;Modal&quot;},&quot;modal&quot;:&quot;{&quot;
        options&quot;:&quot;MAY&quot;;,&quot;SHALL&quot;],&quot;value&quot;:&
        quot;&quot;,&quot;ordinal&quot;:&quot;{&quot;options&quot;:&quot;AS CLOSE AS
        POSSIBLE TO &#39;?&#39;&quot;;,&quot;AS MANY POSSIBLE TO &#39;?&#39;&
        quot;;,&quot;AS FEW AS POSSIBLE TO &#39;?&#39;&quot;],&quot;value&quot;:&
        quot;&quot;,&quot;temporal&quot;:&quot;{&quot;options&quot;:&quot;EVENTUALLY
        &quot;;,&quot;UNTIL&quot;;,&quot;BEFORE&quot;;,&quot;AFTER&quot;;,&quot;AS
        EARLY AS POSSIBLE&quot;;,&quot;AS LATE AS POSSIBLE&quot;;,&quot;AS CLOSE
        AS POSSIBLE TO &#39;?&#39;&quot;],&quot;value&quot;:&quot;&quot;}}}" id=
        "Lx7MsKEuWNhGffszCJ6-18">
<mxCell style="swimlane;fontStyle=0;childLayout=stackLayout;horizontal=1;
        startSize=26;horizontalStack=0;resizeParent=1;resizeParentMax=0;
        resizeLast=0;collapsible=1;marginBottom=0;align=center;fontSize=14;"
        vertex="1" parent="Lx7MsKEuWNhGffszCJ6-1">
        <mxGeometry x="920" y="298" width="160" height="104" as="geometry" />
</mxCell>
</object>
<object label="Manage Data Relationships" uncertaintyDB="{&quot;
        ;1623347531099&quot;:&quot;id&quot;:&quot;1623347531099,&quot;system&quot;:&
        quot;System name&quot;;,&quot;manifestation&quot;:&quot;&quot;;,&quot;
        environmentMonitor&quot;:&quot;&quot;;,&quot;nature&quot;:&quot;{&quot;options&
        quot;:&quot;Aleatory&quot;;,&quot;Epistemic&quot;],&quot;value&quot;:&
        quot;&quot;,&quot;perspective&quot;:&quot;{&quot;options&quot;:&quot;{&quot;
        Subjective&quot;;,&quot;Objective&quot;],&quot;value&quot;:&quot;&quot;
        ;,&quot;source&quot;:&quot;{&quot;options&quot;:&quot;External&quot;;,&quot;
        Internal&quot;],&quot;value&quot;:&quot;&quot;,&quot;viewpoint&quot;:&quot;{&
        quot;options&quot;:&quot;Logical&quot;;,&quot;Physical&quot;;,&quot;
        Process&quot;],&quot;value&quot;:&quot;&quot;,&quot;description&quot;:&
        quot;Description&quot;;,&quot;evidence&quot;:&quot;Evidence&quot;;,&quot;
        SourceDescription&quot;:&quot;&quot;;,&quot;relatedUncertainties&quot;
        ;:&quot;,&quot;location&quot;:&quot;&quot;;,&quot;level&quot;:&quot;{&quot;options&
        quot;:&quot;Known Unknown&quot;;,&quot;Unknown unknown&quot;;,&quot;
        Statistical&quot;],&quot;value&quot;:&quot;&quot;,&quot;awareness&quot;
        ;:&quot;options&quot;:&quot;Known Unknown&quot;;,&quot;Unknown unknown&
        quot;],&quot;value&quot;:&quot;&quot;,&quot;emergingTime&quot;:&quot;{&quot;
        options&quot;:&quot;Requirement&quot;;,&quot;Development&quot;;,&quot;
        Runtime&quot;],&quot;value&quot;:&quot;&quot;,&quot;lifetime&quot;:&
        quot;&quot;;,&quot;pattern&quot;:&quot;{&quot;options&quot;:&quot;Periodic&
        quot;;,&quot;Persistence &quot;;,&quot;Transient&quot;;,&quot;Sporadic&quot;
        ;],&quot;value&quot;:&quot;&quot;,&quot;measure&quot;:&quot;{&quot;options&
        quot;:&quot;Probability&quot;;,&quot;Persistence &quot;;,&quot;Fuzziness&
        quot;;,&quot;Temporal logic&quot;;,&quot;Non-specificity&quot;],&quot;
        value&quot;:&quot;&quot;,&quot;related&quot;:&quot;&quot;;,&quot;
        operator&quot;:&quot;{&quot;options&quot;:&quot;Modal&quot;;,&quot;Temporal&
        quot;;,&quot;Ordinal&quot;],&quot;value&quot;:&quot;Modal&quot;},&quot;
        modal&quot;:&quot;{&quot;options&quot;:&quot;MAY&quot;;,&quot;SHALL&quot;],&
        quot;value&quot;:&quot;&quot;,&quot;ordinal&quot;:&quot;{&quot;options&quot;
        ;:&quot;AS CLOSE AS POSSIBLE TO &#39;?&#39;&quot;;,&quot;AS MANY
        POSSIBLE TO &#39;?&#39;&quot;;,&quot;AS FEW AS POSSIBLE TO &#39;?&#39;&
        quot;],&quot;value&quot;:&quot;&quot;,&quot;temporal&quot;:&quot;{&quot;

```

```

options&quot;;[&quot;EVENTUALLY&quot;;&quot;UNTIL&quot;;&quot;BEFORE&
quot;;&quot;AFTER&quot;;&quot;AS EARLY AS POSSIBLE&quot;;&quot;AS LATE
AS POSSIBLE&quot;;&quot;AS CLOSE AS POSSIBLE TO &#39;?&#39;&quot;;]&quot;
;value&quot;;&quot;&quot;}}" id="Lx7MsKEuWNhGFfsztCJ6-19">
<mxCell style="text;strokeColor=none;fillColor=none;spacingLeft=4;
spacingRight=4;overflow=hidden;rotatable=0;points=[[0,0.5],[1,0.5]];
portConstraint=eastwest;fontSize=12;" vertex="1" parent="
Lx7MsKEuWNhGFfsztCJ6-18">
<mxGeometry y="26" width="160" height="26" as="geometry" />
</mxCell>
</object>
<object label="Manual Data Edits" uncertaintyDB="{&quot;1623347529998&quot;
;:&quot;id&quot;;:1623347529998,&quot;system&quot;;:&quot;System name&
quot;;&quot;manifestation&quot;;&quot;&quot;;&quot;environmentMonitor&
quot;;&quot;&quot;;&quot;nature&quot;;:&quot;options&quot;;:&quot;[&quot;
Aleatory&quot;;&quot;Epistemic&quot;]&quot;;&quot;value&quot;;:&quot;&quot;;&
quot;perspective&quot;;:&quot;options&quot;;:&quot;Subjective&quot;;&
quot;Objective&quot;]&quot;;&quot;value&quot;;:&quot;&quot;;&quot;source&quot;
;:&quot;options&quot;;:&quot;External&quot;;&quot;Internal&quot;]&quot;;&quot;
value&quot;;:&quot;&quot;;&quot;viewpoint&quot;;:&quot;options&quot;;:&quot;[&
quot;Logical&quot;;&quot;Physical&quot;;&quot;Process&quot;]&quot;;&quot;value
&quot;;&quot;&quot;;&quot;description&quot;;:&quot;Description&quot;;&
quot;evidence&quot;;&quot;Evidence&quot;;&quot;SourceDescription&quot;;&
quot;&quot;;&quot;relatedUncertainties&quot;;:{&quot;location&quot;;:&
quot;&quot;;&quot;level&quot;;:&quot;options&quot;;:&quot;Known Unknown&
quot;;&quot;Unknown unknown&quot;;&quot;Statistical&quot;]&quot;;&quot;value&
quot;;&quot;&quot;;&quot;awareness&quot;;:&quot;options&quot;;:&quot;Known
Unknown&quot;;&quot;Unknown unknown&quot;]&quot;;&quot;value&quot;;:&quot;
&quot;;&quot;emergingTime&quot;;:&quot;options&quot;;:&quot;
Requirement&quot;;&quot;Development&quot;;&quot;Runtime&quot;]&quot;;
value&quot;;:&quot;&quot;;&quot;lifetime&quot;;:&quot;&quot;;&quot;
pattern&quot;;:&quot;options&quot;;:&quot;Periodic&quot;;&quot;
Persistence &quot;;&quot;Transient&quot;;&quot;Sporadic&quot;]&quot;;
value&quot;;:&quot;&quot;;&quot;measure&quot;;:&quot;options&quot;;:&quot;[&
quot;Probability&quot;;&quot;Persistence &quot;;&quot;Fuzziness&quot;;&
quot;Temporal logic&quot;;&quot;Non-specificity&quot;]&quot;;&quot;value&quot;
:&quot;&quot;;&quot;related&quot;;:&quot;&quot;;&quot;operator&quot;;:&quot;
options&quot;;:&quot;Modal&quot;;&quot;Temporal&quot;;&quot;Ordinal
&quot;]&quot;;&quot;value&quot;;:&quot;Modal&quot;]&quot;;&quot;modal&quot;;:&quot;
options&quot;;:&quot;MAY&quot;;&quot;SHALL&quot;]&quot;;&quot;value&quot;;:&
quot;&quot;;&quot;ordinal&quot;;:&quot;options&quot;;:&quot;AS CLOSE AS
POSSIBLE TO &#39;?&#39;&quot;;&quot;AS MANY POSSIBLE TO &#39;?&#39;&
quot;;&quot;AS FEW AS POSSIBLE TO &#39;?&#39;&quot;]&quot;;&quot;value&quot;;:&
quot;&quot;;&quot;temporal&quot;;:&quot;options&quot;;:&quot;EVENTUALLY
&quot;;&quot;UNTIL&quot;;&quot;BEFORE&quot;;&quot;AFTER&quot;;&quot;AS
EARLY AS POSSIBLE&quot;;&quot;AS LATE AS POSSIBLE&quot;;&quot;AS CLOSE
AS POSSIBLE TO &#39;?&#39;&quot;]&quot;;&quot;value&quot;;:&quot;&quot;}}" id=
"Lx7MsKEuWNhGFfsztCJ6-20">
<mxCell style="text;strokeColor=none;fillColor=none;spacingLeft=4;
spacingRight=4;overflow=hidden;rotatable=0;points=[[0,0.5],[1,0.5]];
portConstraint=eastwest;fontSize=12;" vertex="1" parent="
Lx7MsKEuWNhGFfsztCJ6-18">
<mxGeometry y="52" width="160" height="26" as="geometry" />
</mxCell>

```

```

</object>
<object label="Create & Manage Datasets" uncertaintyDB="{&quot;
;1623347564024&quot;;:&quot;id&quot;;:1623347564024,&quot;;system&quot;;:&
quot;;System name&quot;;,&quot;;manifestation&quot;;:&quot;;&quot;;,&quot;;
environmentMonitor&quot;;:&quot;;&quot;;,&quot;;nature&quot;;:&quot;;options&
quot;;:[&quot;;Aleatory&quot;;,&quot;;Epistemic&quot;;],&quot;;value&quot;;:&
quot;;&quot;;,&quot;;perspective&quot;;:&quot;;options&quot;;:[&quot;;
Subjective&quot;;,&quot;;Objective&quot;;],&quot;;value&quot;;:&quot;;&quot;;
;,&quot;;source&quot;;:&quot;;options&quot;;:[&quot;;External&quot;;,&quot;;
Internal&quot;;],&quot;;value&quot;;:&quot;;&quot;;,&quot;;viewpoint&quot;;:&
quot;;options&quot;;:[&quot;;Logical&quot;;,&quot;;Physical&quot;;,&quot;;
Process&quot;;],&quot;;value&quot;;:&quot;;&quot;;,&quot;;description&quot;;:&
quot;;Description&quot;;,&quot;;evidence&quot;;:&quot;;Evidence&quot;;,&quot;;
SourceDescription&quot;;:&quot;;&quot;;,&quot;;relatedUncertainties&quot;
;:{},&quot;;location&quot;;:&quot;;&quot;;,&quot;;level&quot;;:&quot;;options&
quot;;:[&quot;;Known Unknown&quot;;,&quot;;Unknown unknown&quot;;,&quot;;
Statistical&quot;;],&quot;;value&quot;;:&quot;;&quot;;,&quot;;awareness&quot;
;:{&quot;;options&quot;;:[&quot;;Known Unknown&quot;;,&quot;;Unknown unknown&
quot;;],&quot;;value&quot;;:&quot;;&quot;;,&quot;;emergingTime&quot;;:&quot;;
options&quot;;:[&quot;;Requirement&quot;;,&quot;;Development&quot;;,&quot;;
Runtime&quot;;],&quot;;value&quot;;:&quot;;&quot;;,&quot;;lifetime&quot;;:&
quot;;&quot;;,&quot;;pattern&quot;;:&quot;;options&quot;;:[&quot;;Periodic&
quot;;,&quot;;Persistence &quot;;,&quot;;Transient&quot;;,&quot;;Sporadic&quot;
;],&quot;;value&quot;;:&quot;;&quot;;,&quot;;measure&quot;;:&quot;;options&
quot;;:[&quot;;Probability&quot;;,&quot;;Persistence &quot;;,&quot;;Fuzziness&
quot;;,&quot;;Temporal logic&quot;;,&quot;;Non-specificity&quot;;],&quot;;
value&quot;;:&quot;;&quot;;,&quot;;related&quot;;:&quot;;&quot;;,&quot;;
operator&quot;;:&quot;;options&quot;;:[&quot;;Modal&quot;;,&quot;;Temporal&
quot;;,&quot;;Ordinal&quot;;],&quot;;value&quot;;:&quot;;Modal&quot;;,&quot;;
modal&quot;;:&quot;;options&quot;;:[&quot;;MAY&quot;;,&quot;;SHALL&quot;;],&
quot;;value&quot;;:&quot;;&quot;;,&quot;;ordinal&quot;;:&quot;;options&quot;
;:[&quot;;AS CLOSE AS POSSIBLE TO &#39;?&#39;&quot;;,&quot;;AS MANY
POSSIBLE TO &#39;?&#39;&quot;;,&quot;;AS FEW AS POSSIBLE TO &#39;?&#39;&
quot;;],&quot;;value&quot;;:&quot;;&quot;;,&quot;;temporal&quot;;:&quot;;
options&quot;;:[&quot;;EVENTUALLY&quot;;,&quot;;UNTIL&quot;;,&quot;;BEFORE&
quot;;,&quot;;AFTER&quot;;,&quot;;AS EARLY AS POSSIBLE&quot;;,&quot;;AS LATE
AS POSSIBLE&quot;;,&quot;;AS CLOSE AS POSSIBLE TO &#39;?&#39;&quot;;],&quot;
;value&quot;;:&quot;;&quot;;}}}" id="Lx7MsKEuWNhGFfsztCJ6-21">
<mxCell style="text;strokeColor=none;fillColor=none;spacingLeft=4;
spacingRight=4;overflow=hidden;rotatable=0;points=[[0,0.5],[1,0.5]];
portConstraint=eastwest;fontSize=12;" vertex="1" parent="
Lx7MsKEuWNhGFfsztCJ6-18">
<mxGeometry y="78" width="160" height="26" as="geometry" />
</mxCell>
</object>
<object label="Administrators" uncertaintyDB="{&quot;;1623347515019&quot;;:&
quot;;id&quot;;:1623347515019,&quot;;system&quot;;:&quot;;System name&quot;;,&
quot;;manifestation&quot;;:&quot;;&quot;;,&quot;;environmentMonitor&quot;;:&
quot;;&quot;;,&quot;;nature&quot;;:&quot;;options&quot;;:[&quot;;Aleatory&quot;
;,&quot;;Epistemic&quot;;],&quot;;value&quot;;:&quot;;&quot;;,&quot;;
perspective&quot;;:&quot;;options&quot;;:[&quot;;Subjective&quot;;,&quot;;
Objective&quot;;],&quot;;value&quot;;:&quot;;&quot;;,&quot;;source&quot;;:&
quot;;options&quot;;:[&quot;;External&quot;;,&quot;;Internal&quot;;],&quot;;
value&quot;;:&quot;;&quot;;,&quot;;viewpoint&quot;;:&quot;;options&quot;;:[&

```



```

    "Logical", "Physical", "Process"], "value": "description", "description": "Evidence", "evidence": "Evidence", "sourceDescription": "relatedUncertainties": {}, "location": "level": {"options": ["Known", "Unknown", "Unknown", "unknown"], "Statistical": "value": "awareness": {"options": ["Known", "Unknown", "Unknown", "unknown"], "value": "emergingTime": {"options": ["Requirement", "Development", "Runtime"], "value": "lifetime": "pattern": {"options": ["Periodic"], "Persistence": "Transient", "Sporadic"], "value": "measure": {"options": ["Probability", "Persistence", "Fuzziness", "Temporal logic", "Non-specificity"], "value": "related": "operator": {"options": ["Modal", "Temporal", "Ordinal"], "value": "modal": {"options": ["MAY", "SHALL"], "value": "ordinal": {"options": ["POSSIBLE TO", "AS FEW AS POSSIBLE TO", "AS MANY POSSIBLE TO", "AS FEW AS POSSIBLE TO", "UNTIL", "BEFORE", "AFTER", "AS EARLY AS POSSIBLE", "AS LATE AS POSSIBLE", "AS POSSIBLE TO"], "value": "id="
    "Lx7MsKEuWNhGFFsztCJ6-22">
<mxCell style="swimlane;fontStyle=0;childLayout=stackLayout;horizontal=1;
    startSize=26;horizontalStack=0;resizeParent=1;resizeParentMax=0;
    resizeLast=0;collapsible=1;marginBottom=0;align=center;fontSize=14;"
    vertex="1" parent="Lx7MsKEuWNhGFFsztCJ6-1">
    <mxGeometry x="920" y="484" width="160" height="104" as="geometry" />
</mxCell>
</object>
<object label="Data Sharing Agreements" uncertaintyDB="{&quot;1623347531099&
    quot;:&quot;id&quot;:1623347531099,&quot;system&quot;:&quot;System name
    &quot;,&quot;manifestation&quot;:&quot;,&quot;environmentMonitor&
    quot;:&quot;,&quot;nature&quot;:&quot;options&quot;:&quot;Aleatory&quot;,&quot;Epistemic&quot;,&quot;value&quot;:&quot;,&
    quot;perspective&quot;:&quot;options&quot;:&quot;Subjective&quot;,&
    quot;Objective&quot;,&quot;value&quot;:&quot;,&quot;source&quot;
    :&quot;options&quot;:&quot;External&quot;,&quot;Internal&quot;,&quot;value&quot;:&quot;,&quot;viewpoint&quot;:&quot;options&quot;:&quot;[&
    quot;Logical&quot;,&quot;Physical&quot;,&quot;Process&quot;],&quot;value
    &quot;:&quot;,&quot;description&quot;:&quot;Description&quot;,&
    quot;evidence&quot;:&quot;Evidence&quot;,&quot;SourceDescription&quot;:&
    quot;,&quot;relatedUncertainties&quot;:{},&quot;location&quot;:&
    quot;,&quot;level&quot;:&quot;options&quot;:&quot;Known Unknown&
    quot;,&quot;Unknown unknown&quot;,&quot;Statistical&quot;,&quot;value&
    quot;:&quot;,&quot;awareness&quot;:&quot;options&quot;:&quot;Known
    Unknown&quot;,&quot;Unknown unknown&quot;,&quot;value&quot;:&quot;
    &quot;,&quot;emergingTime&quot;:&quot;options&quot;:&quot;Requirement&quot;,&quot;Development&quot;,&quot;Runtime&quot;],&quot;

```

```

value&quot;;&quot;&quot;},&quot;lifetime&quot;:&quot;&quot;,&quot;
pattern&quot;:{&quot;options&quot;:[&quot;Periodic&quot;,&quot;
Persistence &quot;,&quot;Transient&quot;,&quot;Sporadic&quot;],&quot;
value&quot;:&quot;&quot;},&quot;measure&quot;:{&quot;options&quot;:[&
quot;Probability&quot;,&quot;Persistence &quot;,&quot;Fuzziness&quot;,&
quot;Temporal logic&quot;,&quot;Non-specificity&quot;],&quot;value&quot;
:&quot;&quot;},&quot;related&quot;:&quot;&quot;,&quot;operator&quot;:{&
quot;options&quot;:[&quot;Modal&quot;,&quot;Temporal&quot;,&quot;Ordinal
&quot;],&quot;value&quot;:&quot;Modal&quot;},&quot;modal&quot;:{&quot;
options&quot;:[&quot;MAY&quot;,&quot;SHALL&quot;],&quot;value&quot;:&
quot;&quot;},&quot;ordinal&quot;:{&quot;options&quot;:[&quot;AS CLOSE AS
POSSIBLE TO &#39;?&#39;&quot;,&quot;AS MANY POSSIBLE TO &#39;?&#39;&
quot;,&quot;AS FEW AS POSSIBLE TO &#39;?&#39;&quot;],&quot;value&quot;:&
quot;&quot;},&quot;temporal&quot;:{&quot;options&quot;:[&quot;EVENTUALLY
&quot;,&quot;UNTIL&quot;,&quot;BEFORE&quot;,&quot;AFTER&quot;,&quot;AS
EARLY AS POSSIBLE&quot;,&quot;AS LATE AS POSSIBLE&quot;,&quot;AS CLOSE
AS POSSIBLE TO &#39;?&#39;&quot;],&quot;value&quot;:&quot;&quot;}}}" id=
"Lx7MsKEuWNhGFfsztCJ6-23">
<mxCell style="text;strokeColor=none;fillColor=none;spacingLeft=4;
spacingRight=4;overflow=hidden;rotatable=0;points=[[0,0.5],[1,0.5]];
portConstraint=eastwest;fontSize=12;" vertex="1" parent="
Lx7MsKEuWNhGFfsztCJ6-22">
<mxGeometry y="26" width="160" height="26" as="geometry" />
</mxCell>
</object>
<object label="Data Usage & Consent" uncertaintyDB="{&quot;
;1623347529998&quot;:{&quot;id&quot;:1623347529998,&quot;system&quot;:&
quot;System name&quot;,&quot;manifestation&quot;:&quot;&quot;,&quot;
environmentMonitor&quot;:&quot;&quot;,&quot;nature&quot;:{&quot;options&
quot;:[&quot;Aleatory&quot;,&quot;Epistemic&quot;],&quot;value&quot;:&
quot;&quot;},&quot;perspective&quot;:{&quot;options&quot;:[&quot;
Subjective&quot;,&quot;Objective&quot;],&quot;value&quot;:&quot;&quot;
},&quot;source&quot;:{&quot;options&quot;:[&quot;External&quot;,&quot;
Internal&quot;],&quot;value&quot;:&quot;&quot;},&quot;viewpoint&quot;:{&
quot;options&quot;:[&quot;Logical&quot;,&quot;Physical&quot;,&quot;
Process&quot;],&quot;value&quot;:&quot;&quot;},&quot;description&quot;:&
quot;Description&quot;,&quot;evidence&quot;:&quot;Evidence&quot;,&quot;
SourceDescription&quot;:&quot;&quot;,&quot;relatedUncertainties&quot;
:,&quot;location&quot;:&quot;&quot;,&quot;level&quot;:{&quot;options&
quot;:[&quot;Known Unknown&quot;,&quot;Unknown unknown&quot;,&quot;
Statistical&quot;],&quot;value&quot;:&quot;&quot;},&quot;awareness&quot;
:,&quot;options&quot;:[&quot;Known Unknown&quot;,&quot;Unknown unknown&
quot;],&quot;value&quot;:&quot;&quot;},&quot;emergingTime&quot;:{&quot;
options&quot;:[&quot;Requirement&quot;,&quot;Development&quot;,&quot;
Runtime&quot;],&quot;value&quot;:&quot;&quot;},&quot;lifetime&quot;:&
quot;&quot;,&quot;pattern&quot;:{&quot;options&quot;:[&quot;Periodic&
quot;,&quot;Persistence &quot;,&quot;Transient&quot;,&quot;Sporadic&quot;
],&quot;value&quot;:&quot;&quot;},&quot;measure&quot;:{&quot;options&
quot;:[&quot;Probability&quot;,&quot;Persistence &quot;,&quot;Fuzziness&
quot;,&quot;Temporal logic&quot;,&quot;Non-specificity&quot;],&quot;
value&quot;:&quot;&quot;},&quot;related&quot;:&quot;&quot;,&quot;
operator&quot;:{&quot;options&quot;:[&quot;Modal&quot;,&quot;Temporal&
quot;,&quot;Ordinal&quot;],&quot;value&quot;:&quot;Modal&quot;},&quot;
modal&quot;:{&quot;options&quot;:[&quot;MAY&quot;,&quot;SHALL&quot;],&

```



```

quot;value&quot;;&quot;&quot;},&quot;ordinal&quot;:{&quot;options&quot;
;:[&quot;AS CLOSE AS POSSIBLE TO &#39;?&#39;&quot;;&quot;AS MANY
POSSIBLE TO &#39;?&#39;&quot;;&quot;AS FEW AS POSSIBLE TO &#39;?&#39;&
quot;],&quot;value&quot;:&quot;&quot;},&quot;temporal&quot;:{&quot;
options&quot;:[&quot;EVENTUALLY&quot;;&quot;UNTIL&quot;;&quot;BEFORE&
quot;;&quot;AFTER&quot;;&quot;AS EARLY AS POSSIBLE&quot;;&quot;AS LATE
AS POSSIBLE&quot;;&quot;AS CLOSE AS POSSIBLE TO &#39;?&#39;&quot;],&quot;
value&quot;:&quot;&quot;}}" id="Lx7MsKEuWNhGFfsztCJ6-24">
<mxCell style="text;strokeColor=none;fillColor=none;spacingLeft=4;
spacingRight=4;overflow=hidden;rotatable=0;points=[[0,0.5],[1,0.5]];
portConstraint=eastwest;fontSize=12;" vertex="1" parent="
Lx7MsKEuWNhGFfsztCJ6-22">
<mxGeometry y="52" width="160" height="26" as="geometry" />
</mxCell>
</object>
<object label="3rd Party Engagement Rules" uncertaintyDB="{&quot;
;1623347564024&quot;;&quot;id&quot;;:1623347564024,&quot;system&quot;:&
quot;System name&quot;;&quot;manifestation&quot;:&quot;&quot;;&quot;
environmentMonitor&quot;:&quot;&quot;;&quot;nature&quot;:{&quot;options&
quot;:[&quot;Aleatory&quot;;&quot;Epistemic&quot;],&quot;value&quot;:&
quot;&quot;},&quot;perspective&quot;:{&quot;options&quot;:[&quot;
Subjective&quot;;&quot;Objective&quot;],&quot;value&quot;:&quot;&quot;
},&quot;source&quot;:{&quot;options&quot;:[&quot;External&quot;;&quot;
Internal&quot;],&quot;value&quot;:&quot;&quot;},&quot;viewpoint&quot;:{&
quot;options&quot;:[&quot;Logical&quot;;&quot;Physical&quot;;&quot;
Process&quot;],&quot;value&quot;:&quot;&quot;},&quot;description&quot;:&
quot;Description&quot;;&quot;evidence&quot;:&quot;Evidence&quot;;&quot;
SourceDescription&quot;:&quot;&quot;;&quot;relatedUncertainties&quot;
;:{},&quot;location&quot;:&quot;&quot;;&quot;level&quot;:{&quot;options&
quot;:[&quot;Known Unknown&quot;;&quot;Unknown unknown&quot;;&quot;
Statistical&quot;],&quot;value&quot;:&quot;&quot;},&quot;awareness&quot;
;:{&quot;options&quot;:[&quot;Known Unknown&quot;;&quot;Unknown unknown&
quot;],&quot;value&quot;:&quot;&quot;},&quot;emergingTime&quot;:{&quot;
options&quot;:[&quot;Requirement&quot;;&quot;Development&quot;;&quot;
Runtime&quot;],&quot;value&quot;:&quot;&quot;},&quot;lifetime&quot;:&
quot;&quot;;&quot;pattern&quot;:{&quot;options&quot;:[&quot;Periodic&
quot;;&quot;Persistence &quot;;&quot;Transient&quot;;&quot;Sporadic&quot;
],&quot;value&quot;:&quot;&quot;},&quot;measure&quot;:{&quot;options&
quot;:[&quot;Probability&quot;;&quot;Persistence &quot;;&quot;Fuzziness&
quot;;&quot;Temporal logic&quot;;&quot;Non-specificity&quot;],&quot;
value&quot;:&quot;&quot;},&quot;related&quot;:&quot;&quot;;&quot;
operator&quot;:{&quot;options&quot;:[&quot;Modal&quot;;&quot;Temporal&
quot;;&quot;Ordinal&quot;],&quot;value&quot;:&quot;Modal&quot;},&quot;
modal&quot;:{&quot;options&quot;:[&quot;MAY&quot;;&quot;SHALL&quot;],&
quot;value&quot;:&quot;&quot;},&quot;ordinal&quot;:{&quot;options&quot;
;:[&quot;AS CLOSE AS POSSIBLE TO &#39;?&#39;&quot;;&quot;AS MANY
POSSIBLE TO &#39;?&#39;&quot;;&quot;AS FEW AS POSSIBLE TO &#39;?&#39;&
quot;],&quot;value&quot;:&quot;&quot;},&quot;temporal&quot;:{&quot;
options&quot;:[&quot;EVENTUALLY&quot;;&quot;UNTIL&quot;;&quot;BEFORE&
quot;;&quot;AFTER&quot;;&quot;AS EARLY AS POSSIBLE&quot;;&quot;AS LATE
AS POSSIBLE&quot;;&quot;AS CLOSE AS POSSIBLE TO &#39;?&#39;&quot;],&quot;
value&quot;:&quot;&quot;}}" id="Lx7MsKEuWNhGFfsztCJ6-25">
<mxCell style="text;strokeColor=none;fillColor=none;spacingLeft=4;
spacingRight=4;overflow=hidden;rotatable=0;points=[[0,0.5],[1,0.5]];

```

```

        portConstraint=eastwest;fontSize=12;" vertex="1" parent="
        Lx7MsKEuWNhGFfsztCJ6-22">
    <mxGeometry y="78" width="160" height="26" as="geometry" />
</mxCell>
</object>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-126" value="" style="html=1;shadow=0;dashed
=0;align=center;verticalAlign=middle;shape=mxgraph.arrows2.arrow;dy=0.6;
dx=40;flipH=1;notch=0;strokeColor=#000000;strokeWidth=2;fillColor=#
ffffff;" vertex="1" parent="Lx7MsKEuWNhGFfsztCJ6-1">
    <mxGeometry x="820" y="343.5" width="100" height="33" as="geometry" />
</mxCell>
<mxCell id="Lx7MsKEuWNhGFfsztCJ6-127" value="" style="html=1;shadow=0;dashed
=0;align=center;verticalAlign=middle;shape=mxgraph.arrows2.arrow;dy=0.6;
dx=40;flipH=1;notch=0;strokeColor=#000000;strokeWidth=2;fillColor=#
ffffff;" vertex="1" parent="Lx7MsKEuWNhGFfsztCJ6-1">
    <mxGeometry x="810" y="520" width="110" height="33" as="geometry" />
</mxCell>
</root>
</mxGraphModel>

```

Listing B.2: XML data for screenshot

APPENDIX C

APPENDIX-C: CASE STUDY 3 UNCERTAINTY FRAMEWORK DATA

Table C.1: Internet of Vehicles common uncertainties data

Attribute	Uncertainties					
Uncertainty ID	1	2	3	4	5	6
Description	IoV network bandwidth	IoV computation capability	IoV storage capacity	IoV network failure	IoV coordination failure	IoV vehicle, data, and other components growth
Nature	Aleatory	Aleatory	Aleatory	Aleatory	Aleatory	Epistemic
Bound	0 to 100% capability	Maximum available processor capacity				
Perspective	Objective	Objective	Subjective	Objective	Subjective	Objective
Awareness	Known unknown	Known unknown	Known unknown	Known unknown	Known unknown	Known unknown
Level	Medium	Medium	High	Medium	Medium	Medium
Source type	Exogenous	Exogenous	Exogenous or Endogenous	Exogenous	Exogenous or Endogenous	Exogenous or Endogenous
Cause	Bandwidth fluctuations	Limited computing process power	Growth in Big Data from cameras, sensors, and internal automotive actuators (brakes, accelerator, etc.)	Hardware (such as transmitter, receiver and relay) issues	Disruptions due to dynamicity and the mobility of a vehicle	Unpredictable growth in vehicles, data, application etc
Viewpoints	Logical	Logical	Logical	Logical	Logical	Logical
Facets	Architecture	Architecture	Architecture	Architecture	Architecture	Architecture
Location	Network	Computation	Data store	Network	IoV nodes, components and communication channels	IoV components, infrastructure and Data
Manifestation	Bandwidth changes	Processing speed	Storage capacity	Connectivity loss	Lack of coordination among nodes and components and other IoV elements	Growth quantities, magnitudes and numbers
Measure	Probability	Fluctuations	Storage capacity proportion	Probability		
Monitor	Network connection	Processor monitor	Storage capacity monitor	Connection status	Coordination harmony	Various growth indicators
Evidence	Bandwidth fluctuation tends	Processing trends	Free space	Connectivity data	Seamless cooperation and sharing	Growth data
Relationship				Ping command feedback		
Emerging time	Run-time	Run-time	Run-time	Run-time	Run-time	Run-time
Lifetime	Perpetuity	Perpetuity	Perpetuity	Perpetuity	Perpetuity	Perpetuity
Change	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic
Pattern	Aperiodic- sporadic	Aperiodic- sporadic	Aperiodic- sporadic	Aperiodic- sporadic	Aperiodic-sporadic	Aperiodic-sporadic
Dependencies	Network, data traffic and connectivity uncertainties	Real-time processing uncertainties	Data sources uncertainties	IoV components include vehicles, infrastructure and others	Communication challenges and network uncertainties	IoV uncertainties
Risk or Opportunities	Risk: Quality of Service degeneration	Risk: System slow, real-time failure, potential accidents	Risk: Data loss and data inaccessibility	Risk: IoV failure, data loss, accidents and others	Risk: traffic congestion, Insufficient information between IoV components, accidents and other coordination dependent aspects	Risk: Capability limits, poor performance, data loss
Mitigation or Exploitation	Mitigation: High bandwidth network such as 5G and use of fog or edge computing	Mitigation: Us clouding computing and fog computing	Mitigation: cloud computing and storage	Mitigation: Software Define Networks (SDN) to abstract hardware issues	Mitigation: high capacity networks and computation	Mitigation: IoV Scalability design include 5G, cloud and fog computing
Outcome (with operators)	MAXIMISE network bandwidth and network reliability	MAXIMISE IoV computation power and speed	MAXIMISE IoV Storage capacity	MAXIMISE network availability, reliability and robustness	MAXIMISE coordination and data sharing for optimal IoV operation	Support AS MUCH unforeseen growth AS POSSIBLE
Influence measure	High	Medium	High	Medium	Medium	High

Table C.2: Internet of Vehicles common uncertainties

Attribute	Uncertainties					
Uncertainty ID	7	8	9	10	11	12
Description	IoV future services	IoV protocols variations and heterogeneous	IoV range of communication or inter-connection channels	IoV real-time operation	IoV security	IoV mapping localisation
Nature	Epistemic	Epistemic	Epistemic	Aleatory	Epistemic	Aleatory
Bound						
Perspective	Subjective	Objective	Objective	Subjective	Subjective	Objective
Awareness	Known unknown	Known unknown	Known unknown	Known unknown	Known unknown	Known unknown
Level	Medium	medium	Low	Low	Medium	Medium
Source type	Exogenous or Endogenous	Exogenous	Exogenous or Endogenous	Endogenous		
Cause	Technology, market, regulation and business innovations	IoV message variations due to factors such as vehicle variety	Available communication channels options	Not always available due to various issues	Security attacks from malicious sources, cyber attacks etc.	"Global Positioning System (GPS) and sensors issues"
Viewpoints	Logical	Logical	Logical	Logical	Logical	Logical
Facets	Architecture	Architecture	Architecture	Architecture	Architecture	Architecture
Location	IoV services	Network and Data	Network	IoV services	IoV software and hardware infrastructure	Network and IoV services
Manifestation	Upgrades and new services	Message format for vehicles, IoV infrastructure and humans	New communication channel	Unavailability of real-time operations	Detected security vulnerability and exploitation	Misreading directions and mapping
Measure					Probability	Approximation
Monitor	Changes and update requests	Interoperability monitor		Real-time feedback monitor	Security threats and attacks monitor	Location and mapping monitor
Evidence	Upgrades and updates	Interoperability issues	New communication channel request	Real-time feedback	Attack and vulnerability data	Pattern predictions and data
Relationship					Attack results	
Emerging time	Run-time & Deployment	Run-time	Run-time	Run-time	Run-time	Run-time
Lifetime	Perpetuity	Perpetuity	Limited	Perpetuity	Perpetuity	Perpetuity
Change	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic
Pattern	Systematic-persistent	Systematic-persistent	Systematic-limited	Aperiodic-sporadic	Aperiodic-sporadic	Aperiodic-sporadic
Dependencies	IoV uncertainties	IoV communication and services uncertainties	Network uncertainties	IoV uncertainties	IoV uncertainties	Coordination uncertainties
Risk or Opportunities	Opportunities: new Innovations	Risk: Communication and corporation failure	Opportunity: IoV growth with new connections	Risk: handle emergencies such as collision warning	Risk: Data loss, accidents, traffic disruptions and others	Risk: Data loss, coordination failure, IoV disruption and accidents
Mitigation or Exploitation	Exploitation: market potential and footprint	Mitigation: Standardisation of IoV enabling technologies	Mitigation: Software Defined Networks and Interoperability support	Mitigation: Fog or edge computing	Mitigation: Architecture security design consideration	Mitigation: Context Awareness
Outcome (with operators)	IoV with BEST AVAILABLE services for comfortable, efficient and safe driving	MAXIMISE interoperability among IoV components and elements	MAXIMISE possible communication channels and options or IoV architecture should connect to AS MANY components AS POSSIBLE	Continuous access and exchanging reliable data in real-time	MAXIMISE safe driving and ENSURE security	Ensure accurate mapping and location AS POSSIBLE
Influence measure	Medium	Medium	Medium	Very high	High	Medium