

Taming the Evolving Black Box:
Towards Improved Integration and Documentation of
Intelligent Web Services

Alex Cummaudo
BSc Swinburne, BIT(Hons)
<ca@deakin.edu.au>

A thesis submitted in partial fulfilment of the requirements for the
Doctor of Philosophy



Applied Artificial Intelligence Institute
Deakin University
Melbourne, Australia

February 12, 2020

Abstract

Application developers are eager to integrate machine learning (ML) into their software, with a plethora of vendors providing pre-packaged components—typically under the ‘AI’ banner—to entice them. Such components are marketed as developer ‘friendly’ ML and easy for them to integrate (being ‘just another’ component added to their toolchain). These components are, however, non-trivial: in particular, developers unknowingly add the risk of mixing nondeterministic ML behaviour into their applications that, in turn, impact the quality of their software. Prior research advocates that a developer’s conceptual understanding is critical to effective interpretation of reusable components. However, these ready-made AI components do not present sufficient detail to allow developers to acquire this conceptual understanding. In this study, by use of a mixed-methods approach of survey and action research, we investigate if the application developers’ deterministic approach to software development clashes with the mindset needed to incorporate probabilistic components. Our goal is to develop a framework to better document such AI components that improves both the quality of the software produced and the developer productivity behind it.

Declarations

I certify that the thesis entitled “*Taming the Evolving Black Box: Towards Improved Integration and Documentation of Intelligent Web Services*” submitted for the degree of Doctor of Philosophy complies with all statements below.

- (i) I am the creator of all or part of the whole work(s) (including content and layout) and that where reference is made to the work of others, due acknowledgement is given.
- (ii) The work(s) are not in any way a violation or infringement of any copyright, trademark, patent, or other rights whatsoever of any person.
- (iii) That if the work(s) have been commissioned, sponsored or supported by any organisation, I have fulfilled all of the obligations required by such contract or agreement.
- (iv) That any material in the thesis which has been accepted for a degree or diploma by any university or institution is identified in the text.
- (v) All research integrity requirements have been complied with.

Alex Cummaudo
BSc Swinburne, BIT(Hons)
February 12, 2020

To my family, friends, and teachers — all this worthless and impossible without you.

Acknowledgements

A long journey of 20 years education has led me to this thesis and upon reflection, there are so many people that have helped get me to this point. To start, I must thank my family for your support. Each of you have always been there for me in times good and bad. I'm especially grateful to my mother, Rosa, and father, Paul, for your love and support, and to my nieces Nina and Lucy—though too young to read this now—for bringing us all so much joy in the last three years. I thank all my teachers, particularly Natalie Heath, whose hard efforts paid off in my tertiary education, and, of course, all those who assisted me along and help shape this journey. Firstly, to Professor Rajesh Vasa, thank you for your many revisions, patience, ideas and efforts to shape this work: your years of phenomenal guidance—both as a supervisor and as a mentor—has helped rewire my thinking and worldview to far wider perspectives and approach thought and problem-solving in a remarkably new light. Secondly, I thank Professor John Grundy for your efforts and for being such an approachable and hard-working supervisor, always willing to provide feedback and guidance, and help me get over the finish line. I also thank Dr Scott Barnett for the many fruitful discussions shared, your interest in my work, and the joint collaborations we achieved in the last two years; Associate Professor Mohamed Abdelrazek for your help in shaping many of the works within this thesis; and, lastly, Associate Professor Andrew Cain, who not only taught me the realm of programming back in undergrad days, but who gave me the support that a PhD was within my reach, of which I had never fathomed. I must thank everyone at the Applied Artificial Intelligence Institute for creating such an enjoyable environment to work in, especially Jake Renzella, Rodney Pilgrim, Mahdi Babaei, and Reuben Wilson for their friendship over these years and for all the coffee runs, conversations, and ideas shared. And, lastly, to Tom Fellowes: thank you for being by my side throughout this journey.

This chapter is now over, the next chapter awaits...

— Alex Cummaudo
July 2020

Contents

Abstract	iii
Declaration	v
Acknowledgements	ix
Contents	ix
List of Publications	xv
List of Abbreviations	xvii
List of Figures	xxi
List of Tables	xxiv
List of Listings	xxv
I Preface	1
1 Introduction	3
1.1 Research Context	5
1.2 Motivating Scenarios	7
1.3 Research Motivation	12
1.4 Research Goals	14
1.5 Research Methodology	16
1.6 Thesis Organisation	17
1.7 Research Contributions	20
1.7.1 Contribution 1: Landscape Analysis & Preliminary Solutions	21
1.7.2 Contribution 2: Improving Documentation Attributes	22

1.7.3 Contribution 3: Service Integration Architecture	23
2 Background	25
2.1 Software Quality	26
2.1.1 Validation and Verification	27
2.1.2 Quality Attributes and Models	29
2.1.3 Reliability in Computer Vision	31
2.2 Probabilistic and Nondeterministic Systems	33
2.2.1 Interpreting the Uninterpretable	34
2.2.2 Explanation and Communication	35
2.2.3 Mechanics of Model Interpretation	35
2.3 Application Programming Interfaces	36
2.3.1 API Usability	37
3 Research Methodology	39
3.1 Research Questions Revisited	39
3.1.1 Empirical Research Questions	40
3.1.2 Non-Empirical Research Questions	41
3.2 Philosophical Stances	41
3.3 Research Methods	42
3.3.1 Review of Relevant Research Methods	43
3.3.2 Review of Data Collection Techniques for Field Studies	45
3.4 Research Design	45
3.4.1 Landscape Analysis of Computer Vision Services	45
3.4.2 Utility of API Documentation in Computer Vision Services	45
3.4.3 Developer Issues concerning Computer Vision Services	47
3.4.4 Designing Improved Integration Strategies	47
II Publications	49
4 Identifying Evolution in Computer Vision Services	51
4.1 Introduction	51
4.2 Motivating Example	53
4.3 Related Work	54
4.3.1 External Quality	54
4.3.2 Internal Quality	55
4.4 Method	56
4.5 Findings	59
4.5.1 Consistency of top labels	59
4.5.2 Consistency of confidence	61
4.5.3 Evolution risk	63
4.6 Recommendations	64
4.6.1 Recommendations for IWS users	64
4.6.2 Recommendations for IWS providers	65
4.7 Threats to Validity	66

4.7.1	Internal Validity	66
4.7.2	External Validity	67
4.7.3	Construct Validity	67
4.8	Conclusions & Future Work	67
5	Systematic Mapping Study of API Documentation Knowledge	69
5.1	Introduction	69
5.2	Related Work	71
5.3	Method	72
5.3.1	Systematic Mapping Study	72
5.3.2	Development of the Taxonomy	75
5.4	Taxonomy Validation	77
5.4.1	Survey Study	78
5.4.2	Empirical Application against Computer Vision Services	78
5.5	Taxonomy	78
5.6	Threats to Validity	78
5.7	Conclusions & Future Work	79
6	Interpreting Pain-Points in Computer Vision Services	83
6.1	Introduction	83
6.2	Motivation	85
6.3	Background	87
6.4	Method	88
6.4.1	Data Extraction	88
6.4.2	Data Filtering	90
6.4.3	Data Analysis	93
6.5	Findings	93
6.5.1	Post classification and reliability analysis	94
6.5.2	Developer Frustrations	94
6.5.3	Statistical Distribution Analysis	96
6.6	Discussion	97
6.6.1	Answers to Research Questions	97
6.6.2	The Developer's Learning Approach	98
6.6.3	Implications	101
6.7	Threats to Validity	103
6.7.1	Internal Validity	103
6.7.2	External Validity	103
6.7.3	Construct Validity	104
6.8	Conclusions	104
7	Ranking Computer Vision Service Issues using Emotion	105
8	Using a Facade Pattern to combine Computer Vision Services	107
8.1	Introduction	107
8.1.1	Motivating Scenario: Intelligent vs Traditional Web Services	108
8.1.2	Research Motivation	109

8.2	Merging API Responses	109
8.2.1	API Facade Pattern	110
8.2.2	Merge Operations	110
8.2.3	Merging Operators for Labels	111
8.3	Graph of Labels	112
8.3.1	Labels and synsets	112
8.3.2	Connected Components	112
8.4	API Results Merging Algorithm	115
8.4.1	Mapping Labels to Synsets	115
8.4.2	Deciding Total Number of Labels	115
8.4.3	Allocating Number of Labels to Connected Components	116
8.4.4	Selecting Labels from Connected Components	117
8.4.5	Conformance to properties	117
8.5	Evaluation	117
8.5.1	Evaluation Method	117
8.5.2	Naive Operators	118
8.5.3	Traditional Proportional Representation Operators	120
8.5.4	New Proposed Label Merge Technique	120
8.5.5	Performance	120
8.6	Conclusions and Future Work	121
9	Supporting Safe Usage of Intelligent Web Services	123
9.1	Introduction	123
9.2	Motivating Example	125
9.3	Threshy	127
9.4	Related work	128
9.5	Conclusions & Future Work	129
10	FSE Paper	131
III	Postface	133
11	Conclusions & Future Work	135
References		154
List of Online Artefacts		155
IV	Appendices	159
A	Additional Materials	161
A.1	On the Development, Documentation and Usage of Web APIs	163
A.2	Additional Figures	166
A.3	Reference Architecture Source Code	169

B Authorship Statements	171
C Ethics Clearance	185
D Primary Sources from Systematic Mapping Study	189

List of Publications

Below lists publications arising from work completed in this PhD.

1. A. Cummaudo, R. Vasa, J. Grundy, M. Abdelrazek, and A. Cain, “Losing Confidence in Quality: Unspoken Evolution of Computer Vision Services,” in *Proceedings of the 35th IEEE International Conference on Software Maintenance and Evolution*. Cleveland, OH, USA: IEEE, December 2019. DOI 10.1109/ICSME.2019.00051. ISBN 978-1-72-813094-1 pp. 333–342
2. A. Cummaudo, R. Vasa, and J. Grundy, “What should I document? A preliminary systematic mapping study into API documentation knowledge,” in *Proceedings of the 13th International Symposium on Empirical Software Engineering and Measurement*. Porto de Galinhas, Recife, Brazil: IEEE, October 2019. DOI 10.1109/ESEM.2019.8870148. ISBN 978-1-72-812968-6. ISSN 1949-3789 pp. 1–6
3. A. Cummaudo, R. Vasa, S. Barnett, J. Grundy, and M. Abdelrazek, “Interpreting Cloud Computer Vision Pain-Points: A Mining Study of Stack Overflow,” in *Proceedings of the 42nd International Conference on Software Engineering*. Seoul, Republic of Korea: IEEE, May 2020, In Press
4. A. Cummaudo, S. Barnett, R. Vasa, and J. Grundy, “Threshy: Supporting Safe Usage of Intelligent Web Services,” Seoul, Republic of Korea, 2020, Unpublished
5. A. Cummaudo, R. Vasa, and J. Grundy, “Assessing the efficacy of API documentation knowledge against practitioners and computer vision services,” 2020, Unpublished
6. A. Cummaudo, “ESEC/FSE Paper,” Unpublished
7. T. Ohtake, A. Cummaudo, M. Abdelrazek, R. Vasa, and J. Grundy, “Merging intelligent API responses using a proportional representation approach,” in *Proceedings of the 19th International Conference on Web Engineering*. Daejeon, Republic of Korea: Springer, June 2019. DOI 10.1007/978-3-030-19274-7_28. ISBN 978-3-03-019273-0. ISSN 1611-3349 pp. 391–406
8. M. K. Curumsing, A. Cummaudo, U. M. Graestch, S. Barnett, and R. Vasa, “Ranking Computer Vision Service Issues using Emotion,” in *Proceedings of the 5th International Workshop on Emotion Awareness in Software Engineering*, Seoul, Republic of Korea, May 2020

List of Abbreviations

- A²I²** Applied Artificial Intelligence Institute. 45, 47
- AI** artificial intelligence. iii, 3–5, 8, 12, 14, 34, 35, 51, 52, 55, 56, 65, 66, 68, 70, 109, 110, 167
- API** application programming interface. xxiv, 4–16, 18, 22, 23, 25, 26, 28, 29, 36–38, 40–42, 44, 45, 47, 52, 53, 56, 57, 65, 66, 68–76, 78–82, 107–110, 112, 114, 115, 117, 118, 121, 163, 165
- AWS** Amazon web services. 57, 60
- BYOML** Build Your Own Machine Learning. 5, 6
- CC** connected component. 112, 115–117, 121
- CDSS** clinical decision support system. 8, 10, 11
- CNN** convolutional neural network. 10, 11, 33, 54
- CRUD** Create, read, update, and delete. 165
- CV** computer vision. 5, 7, 23, 32, 37, 51, 53, 54, 66–68
- CVS** computer vision service. 7–10, 12, 14–21, 23, 25, 27, 28, 31, 37, 40–42, 44, 45, 47, 51–57, 60, 65–67, 70, 71, 107, 109, 110, 112, 115, 117, 121, 168
- DCE** distributed computing environment. 163
- HITL** human-in-the-loop. 11
- HTTP** Hypertext Transfer Protocol. 6, 163–165
- IDL** interface definition language. 163, 165

IWS intelligent web service. 5–7, 9, 11, 12, 14, 15, 17, 18, 25–28, 31, 33, 36–38, 51–53, 55, 56, 64, 66–68, 70, 71, 77, 107–109, 121

JSON JavaScript Object Notation. 7

KA knowledge area. 72, 76

ML machine learning. iii, 3–6, 8, 9, 12–14, 18, 21, 29, 34, 37, 51, 52, 55, 56, 65, 68, 107, 108, 121

NN neural network. 13, 32, 34, 36

PaaS Platform as a Service. 7, 11, 55

QoS quality of service. 55, 56, 163

RAML RESTful API Modeling Language. 165

REST REpresentational State Transfer. 7, 52, 108, 164, 165

ROI region of interest. 10, 11

RPC remote procedure call. 163

SDK software development kit. 72

SE software engineering. 14, 15, 17, 18, 35, 52, 55, 71–76

SLA service-level agreement. 55, 163

SMS systematic mapping study. 18, 20, 22, 70–72, 79

SO Stack Overflow. 5, 15, 18, 19, 22, 23, 40, 41, 44, 47, 56, 57, 168

SOA service-oriented architecture. 163

SOAP Simple Object Access Protocol. 7, 163–165

SQA service quality assurance. 53, 54

SQuaRE Systems and software Quality Requirements and Evaluation. 30

SUS System Usability Scale. 18, 71

SVM support vector machine. 34, 36

SWEBOK Software Engineering Body of Knowledge. 72, 73, 76

URI uniform resource identifier. 165

V&V verification & validation. 25–29

WADL Web Application Description Language. 165

WS web service. 6, 28, 56, 108, 109, 163, 165

WSDL Web Services Description Language. 163

XML eXtendable markup language. 7, 163

List of Figures

1.1	Differences between data- and rule-driven cloud services	4
1.2	The spectrum of machine learning	5
1.3	Overview of intelligent web services	7
1.4	CancerAssist Context Diagram	11
1.5	Overview publication coherency	22
2.1	Mindset clashes within the development, use and nature of a IWS . .	26
2.2	Leakage of internal and external quality in	29
2.3	Overview of software quality models	30
2.4	Adversarial examples in computer vision	32
2.5	Deterministic versus nondeterministic systems	33
2.6	Theory of AI communication	36
3.1	Review of field study techniques	46
4.1	Consistency of labels in CV services is rare	59
4.2	Top labels for images between CV services do not intersect	60
4.3	CV services can return multiple top labels	61
4.4	Cumulative distribution of top label confidences	62
4.5	Cumulative distribution of intersecting top label confidences . . .	62
4.6	Agreement of labels between multiple CV services do not share similar confidences	63
5.1	SMS search results, by years	74
5.2	A systematic map of API documentation knowledge studies	77
6.1	Traits of intelligent services compared to DIY ML	86
6.2	Trend of Stack Overflow posts discussing computer vision services .	87
6.3	Comparing documentation-specific and generalised classifications of SO posts	94

6.4 Alignment of Bloom and SOLO taxonomies against computer vision issues	100
8.1 Overview of the proposed facade	110
8.2 Graph of associated synsets against two different endpoints	113
8.3 Label counts per API assessed	114
8.4 Connected components vs. images	114
8.5 Allocation to connected components	116
8.6 F-measure comparison	120
9.1 Example case study of evaluating model performance in two different models	124
9.2 Example pipeline of a computer vision system	126
9.3 UI workflow of Threshy	127
9.4 Architecture of Threshy	129
A.1 SOAP versus REST search interest over time	164
A.2 Categorisation of AI-based products and services	167
A.3 Increasing interest in the developer community of computer vision services	168

List of Tables

1.1	Differing characteristics of cloud services	6
1.2	Comparison of the machine learning spectrum	6
1.3	Varying confidence changes over time between three computer vision services	9
1.4	Definitions of ‘confidence’ in CV documentation	10
1.5	List of publications resulting from this thesis	21
3.1	Classification of research questions in this thesis	40
4.1	Characteristics of data in CV evolution assessment	58
4.2	Ratio of consistent labels in CV services	59
4.3	Evolution of top labels and confidence values	63
5.1	Summary of search results in API documentation knowledge	73
5.2	Data extraction in API documentation knowledge study	76
5.3	Taxonomy proposed in API documentation knowledge study	80
6.1	Taxonomies used in our Stack Overflow mining study	92
6.2	Example posts aligning to Bloom’s and the SOLO taxonomy	99
8.1	Statistics for the number of labels	112
8.2	First allocation iteration	117
8.3	Second allocation iteration	117
8.4	Third allocation iteration	117
8.5	Fourth allocation iteration	117
8.6	Matching to human-verified labels	119
8.7	Evaluation results of the facade	119
8.8	Average of evaluation result of the facade	119

List of Listings

A.1	An example SOAP request	163
A.2	An example SOAP response	164
A.3	An example RESTful request	165
A.4	An example RESTful response	165

Part I

Preface

CHAPTER 1

3

4

5

Introduction

6

7 Within the last half-decade, we have seen an explosion of cloud-based services
8 typically marketed under an artificial intelligence (AI) banner. Vendors are rapidly
9 pushing out AI-based solutions, technologies and products encapsulating half a
10 century worth of machine-learning research.¹ Application developers are eager to
11 develop the next generation of ‘AI-first’ software, that will reason, sense, think, act,
12 listen, speak and execute every whim in our web browser or smartphone app.

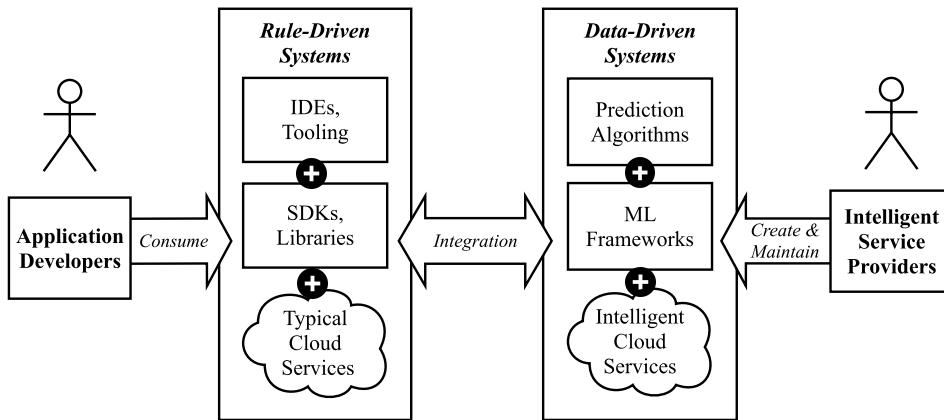
13 However, application developers, accustomed to traditional software engineering
14 paradigms, may not be aware of AI-first’s consequences. Application developers
15 build *rule-driven* applications, where every line of source code evaluates to produce
16 deterministic outcomes. AI-first software is, however, not rule-driven but *data-
driven*. Large datasets train machine learning (ML) prediction classifiers that result
17 in probabilistic confidences of results and nondeterministic behaviour if it continually
18 learns more data with time. Furthermore, developing AI-first applications requires
19 both code *and data*, and an application developer can approach developing from
20 three (non-traditional) perspectives, further expanded in Section 1.1:

- 22 1. The application developer writes an ML classifier from scratch and trains it
23 from a handcrafted and curated dataset. This approach is laborious in time and
24 demands formal training in ML and mathematical knowledge, but the tradeoff
25 is that they have full autonomy in the models they creates.
- 26 2. The application developer downloads a pre-trained model and ‘plugs’ it into
27 an existing ML framework, such as Tensorflow [1]. While this approach is
28 less demanding in time, it requires them to revise and understand how to ‘glue’
29 components of the ML framework together² into their application’s code.

¹A 2016 report by market research company Forrester captured such growth into four key areas [176], as reproduced in Figure A.2.

²Thus introducing a verbose list of ML terminology to her developer vocabulary. See a list of 328 terms provided by Google here: <https://developers.google.com/machine-learning/glossary/>. Last accessed 7 December 2018.

Figure 1.1: The application developer’s rule-driven toolchain is distinct from data-driven toolchain. A developer must consume a typical, data-driven cloud service in a different way than an intelligent data-driven cloud service as they are not the same type of system.



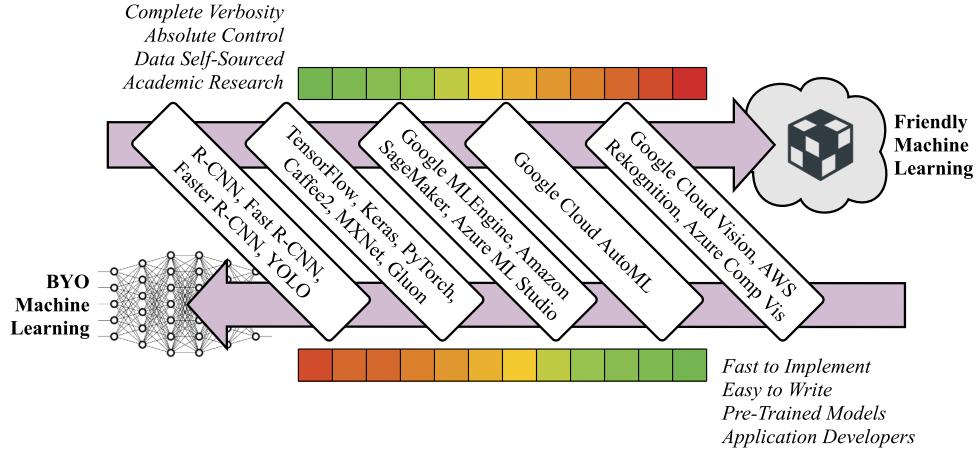
30 3. The application developer uses a data-driven and cloud-based service. They
 31 don’t need to know anything behind the underlying ‘intelligence’ and how it
 32 functions. It is fast to integrate into their applications, and the application pro-
 33 gramming interfaces (APIs) offered abstracts the technical know-how behind
 34 a web call.

35 The documentation of the service alludes that the data-driven service is as similar to
 36 other cloud services offered by the provider. Because this is ‘another’ cloud service,
 37 the application developer *assumes* it would act and behave as any other typical
 38 service would. But does this assumption—and a lack of appreciation of ML—lead
 39 to developer pain-points and miscomprehension? If so, how can the service providers
 40 improve their documentation to alleviate this? Do these data-driven services share
 41 similarities to the runtime behaviour of traditional cloud services? And if not,
 42 how best can the application developer integrate the data-driven service into their a
 43 rule-driven application to produce AI-first software?

44 Figure 1.1 provides an illustrative overview between the context clashing of rule-
 45 driven applications and data-driven cloud services, and we contrast characteristics
 46 of typical cloud systems and data-driven ones in Table 1.1.

In this thesis, we advocate that the integration and developer comprehen-
 sion of data-driven cloud services differ from the rule-driven nature of
 end-applications. As ‘intelligent’ components these contrast to traditional
 counterparts, and application developers need to take into account a greater
 appreciation of these factors.

Figure 1.2: Examples within the machine learning spectrum of computer vision. Colour scales indicates the benefits (green) and drawbacks (red) of each end of the spectrum.



47 1.1 Research Context

48 As described, the application developer has three key approaches in producing AI-
 49 first software. This ‘range’ of AI-first integration techniques partially reflects Google
 50 AI’s³ *machine learning Spectrum* [163, 188, 215], which encompasses the variety
 51 of skill, effort, users and types of outputs of integration techniques. One extreme
 52 involves the academic research of developing algorithms and self-sourcing data
 53 to achieve intelligence—coined as Build Your Own Machine Learning (BYOML)
 54 [141, 188, 215]. The other extreme involves off-the-shelf, ‘friendlier’ (abstracted)
 55 intelligence with easy-to-use APIs targeted towards applications developers. The
 56 middle-ground involves a mix of the two, with varying levels of automation to assist
 57 in development, that turns custom datasets into predictive intelligence. We illustrate
 58 the slightly varied characteristics within this spectrum in Table 1.2 and Figure 1.2.

59 These data-driven ‘friendly’ services are gaining traction within developer circles:
 60 we show an increasing trend of Stack Overflow posts mentioning a mix of
 61 intelligent computer vision (CV) services in Figure A.3.⁴ Academia provides var-
 62 ied nomenclature for these services, such as *Cognitive Applications* and *Machine*
 63 *Learning Services* [290] or *Machine Learning as a Service* [237]. For the context
 64 of this thesis, we will refer to such services under broader term of **intelligent web**
 65 **services (IWSs)**, and diagrammatically express their usage within Figure 1.3.

66 While there are many types of IWSs available to software developers,⁵ the

³Google AI was recently rebranded from Google Research, further highlighting how the ‘AI-first’ philosophy is increasingly becoming embedded in companies’ product lines and research and development teams. Spearheaded through work achieved at Google, Microsoft and Facebook, the emphasis on an AI-first attitude we see through Google’s 2018 rebranding of *Google Research* to *Google AI* [129] is evident. A further example includes how Facebook leverage AI *at scale* within their infrastructure and platforms [218].

⁴Query run on 12 October 2018 using StackExchange Data Explorer. Refer to <https://data.stackexchange.com/stackoverflow/query/910188> for full query.

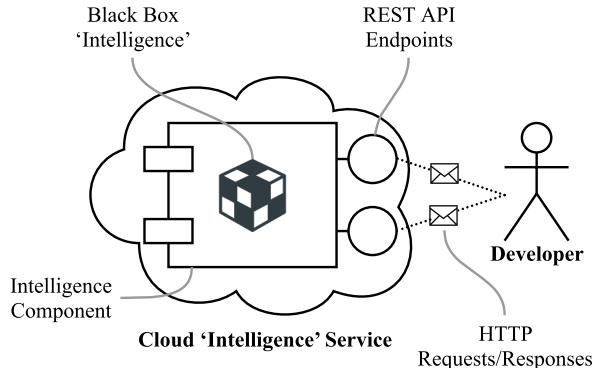
⁵Such as optical character recognition, text-to-speech and speech-to-text transcription, object

Table 1.1: Differing characteristics of intelligent and typical web services.

intelligent web service	Typical web service
Probabilistic	Deterministic
Machine Learnt	Human Engineered
Data-Driven	Rule-Driven
Black-Box	Mostly Transparent

Table 1.2: Comparison of the machine learning spectrum.

Comparator	BYOML	ML F'work	Cloud ML	Auto-Cloud ML	Cloud API
Hosting					
Locally	✓	✓			
Output					
Custom Model	✓	✓	✓	✓	
HTTP Response					✓
Autonomy					
Low					✓
Medium				✓	
High		✓	✓		
Highest	✓				
Time To Market					
Medium	✓	✓			
High			✓	✓	
Highest					✓
Data					
Self-Sourced	✓	✓	✓	✓	
Pre-Trained		✓			✓
Intended User					
Academics	✓	✓			
Data Scientist	✓	✓	✓	✓	
Developers				✓	✓

Figure 1.3: Overview of IWSs.

67 general workflow of using an IWS is more-or-less the same: a developer accesses
 68 an IWS component via REST/SOAP API(s), which is (typically) available as a
 69 cloud-based Platform as a Service (PaaS).^{6,7} For a given input, developers receive
 70 an ‘intelligent’ response and an associated confidence value that represents the
 71 likelihood of that result. This is typically serialised as a JSON/XML response
 72 object.

 Within this thesis, we scope our investigation to a mature subset of IWSs that provide computer vision intelligence [310, 313, 316, 317, 318, 324, 327, 330, 331, 333, 335, 372, 373]. For the context of this thesis, we will refer to such services as **computer vision services (CVSs)**.

73 1.2 Motivating Scenarios

74 < *todo: Could move this into the appendix to keep introduction chapter tighter* >
 75 < *todo: JG: or could use one of these at start of section 1.2 to help clarify/motivate...* >
 76 < *todo: yes too long in this chapter - use one, simplify a bit. Could put other one in*
 77 *appendix...* > < *todo: MA: i guess the discussion related to Table1.3 and Table1.4*
 78 *could be moved to 1.2 as one of the challenges* > < *todo: MA: after reading the*
 79 *section, it looks like you could move it before 1.2* >

80 The market for computer vision services (CVSs) is increasing (Figure A.2)
 81 and as is developer uptake and enthusiasm in the software engineering community
 82 (Figure A.3). However, the impact to software quality (internal and external) due

categorisation, facial analysis and recognition, natural language processing etc.

⁶We note, however, that a development team may use a similar approach *internally* within a product line or service that may not necessarily reflect a PaaS model.

⁷A number of services provide the platform infrastructure to rapidly begin training from custom datasets, such as Google’s AutoML (<https://cloud.google.com/automl/>, last accessed 7 December 2018). Others provide pre-trained datasets ‘ready-for-use’ in production without the need to train data.

83 to a mismatch of the application developer's deterministic mindset and the service
 84 provider's nondeterministic mindset is of concern.

85 To illustrate the context of use, we present the two scenarios of varying risk: (i) a
 86 fictional software developer, named Tom, who wishes to develop an inherently low-
 87 risk photo detection application for his friends and family; and (ii) a high-risk cancer
 88 clinical decision support system (CDSS) that uses patient scans to recommend if
 89 surgeons should send their patients to surgery. Both describe scenarios where AI-
 90 first components has substantiative impact to end-users when the software engineers
 91 developing with them misunderstand the nuances of ML, ultimately adversely affecting
 92 external quality. Moreover, due to lack of comprehension, this hinders developer
 93 experience, productivity, and understanding/appreciation of AI-based components.

94 Motivating Scenario I: Tom's *PhotoSharer* App

95 Tom wants to develop a social media photo-sharing app on iOS and Android, *Photo-*
 96 *Sharer*, that analyses photos taken on smartphones. Tom wants the app to categorise
 97 photos into scenes (e.g., day vs. night, landscape vs. indoors), generate brief de-
 98 scriptions of each photo, and catalogue photos of his friends and common objects
 99 (e.g., photos with his Border Collie dog, photos taken on a beach on a sunny day with
 100 his partner). His app will shares this analysed photo intelligence with his friends on
 101 a social-media platform, where his friends can search and view the photos.

102 Instead of building a computer vision engine from scratch, which takes too much
 103 time and effort, Tom thinks he can achieve this using one of the common CVSs. Tom
 104 comes from a typical software engineering background and has insufficient knowl-
 105 edge of key computer vision terminology and no understanding of its underlying
 106 techniques. However, inspired by easily accessible cloud APIs that offer computer
 107 vision analysis, he chooses to use these. Built upon his experience of using other
 108 similar cloud services, he decides on one of the CVS APIs, and expects a static result
 109 always and consistency between similar APIs. Analogously, when Tom invokes the
 110 iOS Swift substring method "doggy".prefix(3), he expects it to be consistent
 111 with the Android Java equivalent "doggy".substring(0, 2). Consistent, here,
 112 means two things: (i) that calling `substring` or `prefix` on 'dog' will *always*
 113 return in the same way every time he invokes the method; and (ii) that the result is
 114 *always* 'dog' regardless of the programming language or string library used, given
 115 the deterministic nature of the 'substring' construct (i.e., results for `substring` are
 116 API-agnostic).

117 More concretely, in Table 1.3, we illustrate how three (anonymised) CVS
 118 providers fail to provide similar consistency to that of the substring example above.
 119 If Tom uploads a photo of a border collie⁸ to three different providers in August
 120 2018 and January 2019, he would find that each provider is different in both the vo-
 121 cabulary used between. The confidence values and labels within the *same* provider
 122 varies within a matter of five months. The evolution of the confidence changes is
 123 not explicitly documented by the providers (i.e., when the models change) nor do
 124 they document what confidence means. Service providers use a tautological nature

⁸The image used for these results is <https://www.akc.org/dog-breeds/border-collie/>.

Table 1.3: First six responses of image analysis for a Border Collie sent to three CVS providers five months apart. The specificity (to 3 s.f.) and vocabulary of each label in the response varies between all services, and—except for Provider B—changes over time. Any confidence changes greater than 1 per cent are highlighted in red.

Label	Provider A		Provider B		Provider C	
	Aug 2018	Jan 2019	Aug 2018	Jan 2019	Aug 2018	Jan 2019
Dog	0.990	0.986	0.999	0.999	0.992	0.970
Dog Like Mammal	0.960	0.962	-	-	-	-
Dog Breed	0.940	0.943	-	-	-	-
Border Collie	0.850	0.852	-	-	-	-
Dog Breed Group	0.810	0.811	-	-	-	-
Carnivoran	0.810	0.680	-	-	-	-
Black	-	-	0.992	0.992	-	-
Indoor	-	-	0.965	0.965	-	-
Standing	-	-	0.792	0.792	-	-
Mammal	-	-	0.929	0.929	0.992	0.970
Animal	-	-	0.932	0.932	0.992	0.970
Canine	-	-	-	-	0.992	0.970
Collie	-	-	-	-	0.992	0.970
Pet	-	-	-	-	0.992	0.970

when defining what the confidence values are (as presented in the API documentation) provides no insight for Tom to understand why there was a change in confidence, which we show in Table 1.4, unless he *knows* that the underlying models change with them. Furthermore, they do not provide detailed understanding on how to select a threshold cut-off for a confidence value. Therefore, he's left with no understanding on how best to tune for image classification in this instance. The deterministic problem of a substring compared to the nondeterministic nature of the IWS is, therefore, non-trivial.

To make an assessment of these APIs, he tries his best to read through the documentation of different CVS APIs, but he has no guiding framework to help him choose the right one. A number of questions come to mind:

- What does ‘confidence’ mean?
- Which confidence is acceptable in this scenario?
- Are these APIs consistent in how they respond?
- Are the responses in APIs static and deterministic?
- Would a combination of multiple CVS APIs improve the response?
- How does he know when there is a defect in the response? How can he report it?
- How does he know what labels the API knows, and what labels it doesn’t?
- How does it describe his photos and detect the faces?
- Does he understand that the API uses a machine learnt model? Does he know what a ML model is?
- Does he know when models update? What is the release cycle?

Table 1.4: Tautological definitions of ‘confidence’ found in the API documentation of three common CVS providers.

API Provider	Definition(s) of Confidence
Provider A	“Score is the confidence score, which ranges from 0 (no confidence) to 1 (very high confidence).” [325]
	“Deprecated. Use score instead. The accuracy of the entity detection in an image. For example, for an image in which the ‘Eiffel Tower’ entity is detected, this field represents the confidence that there is a tower in the query image. Range [0, 1].” [326]
	“The overall score of the result. Range [0, 1]” [326]
Provider B	“Confidence score, between 0 and 1... if there insufficient confidence in the ability to produce a caption, the tags maybe [sic] the only information available to the caller.” [336]
Provider C	“The level of confidence the service has in the caption.” [334]
Provider C	“The response shows that the operation detected five labels (that is, beacon, building, lighthouse, rock, and sea). Each label has an associated level of confidence. For example, the detection algorithm is 98.4629% confident that the image contains a building.” [311]
	“[Provider C] also provide[s] a percentage score for how much confidence [Provider C] has in the accuracy of each detected label.” [312]

148 Although Tom generally anticipates these CVSs to not be perfect, he has no
 149 prior benchmark to guide him on what to expect. The imperfections appear to be
 150 low-risk, but may become socially awkward when in use; for instance, if Tom’s
 151 friends have low self-esteem and use the app, they may be sensitive to the app not
 152 identifying them or mislabelling them. Privacy issues come into play especially
 153 if certain friends have access to certain photos that they are (supposedly) in; e.g.,
 154 photos from a holiday with Tom and his partner, however if the API identifies Tom’s
 155 partner as a work colleague, Tom’s partner’s privacy is at risk.

156 Therefore, the level of risk and the determination of what constitutes an ‘error’ is
 157 dependent on the situation. In the following example, an error caused by the service
 158 may be more dangerous.

159 Motivating Scenario II: Cancer Detection CDSS

160 Recent studies in the oncology domain have used deep-learning convolutional neural
 161 networks (CNNs) to detect region of interests (ROIs) in image scans of tissue (e.g.,
 162 [24, 116, 175]), flagging these regions for doctors to review. Trials of such algorithms
 163 have been able to accurately detect cancer at higher rates than humans, and thus
 164 incorporating such capabilities into a CDSS is closer within reach. Studies have
 165 suggested these systems may erode a practitioner’s independent decision-making

[56, 139] due to over-reliance; therefore the risks in developing CDSSs powered by IWSs become paramount.

In Figure 1.4 we present a context diagram for a fictional CDSS named *CancerAssist*. A team of busy pathologists utilise CancerAssist to review patient lymph node scans and discuss and recommend, on consensus, if the patient requires an operation. When the team makes a consensus, the lead pathologist enters the verdict into CancerAssist—running passively in the background—to ensure there is no oversight in the team’s discussions. When a conflict exists between the team’s verdict and CancerAssist’s verdict, the system produces the scan with ROIs it thinks the team should review. Where the team overrides the output of CancerAssist, this reinforces CancerAssist’s internal model as a human-in-the-loop (HITL) learning process.

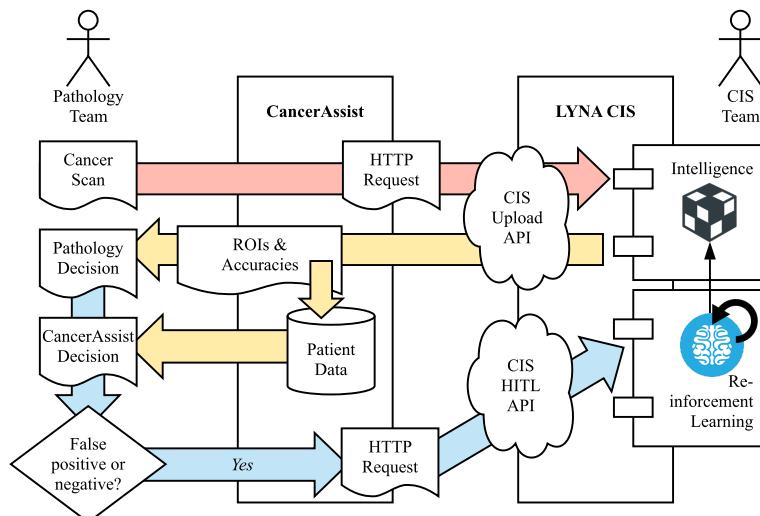


Figure 1.4: CancerAssist Context Diagram. *Key:* Red Arrows = Scan Input; Yellow Arrows = Decision Output; Blue Arrows = HITL Feedback Input.

Powering CancerAssist is Google AI’s Lymph Node Assistant (LYNA) [175], a CNN based on the Inception-v3 model [161, 274]. To provide intelligence to CancerAssist, the development team decide to host LYNA as an IWS using a cloud-based PaaS solution. Thus, CancerAssist provides API endpoints integrated with patient data and medical history, which produces the verdict. In the case of a positive verdict, CancerAssist highlights the relevant ROIs found with their respective bounding boxes and their respective cancer detection accuracies.

The developer of CancerAssist has no interaction with the Data Science team maintaining the LYNA IWS. As a result, they are unaware when updates to the model occur, nor do they know what training data they provide to test their system. The default assumptions are that the training data used to power the intelligence is near-perfect for universal situations; i.e., the algorithm chosen is the correct one for every assessable ontology tests in the given use case of CancerAssist. Thus, unlike deterministic systems—where the developer can manually test and validate the outcomes of the APIs—this is impossible for non-deterministic systems such

193 as CancerAssist and its underlying IWS. The ramifications of not being able to test
 194 such a system and putting it out into production may prove fatal to patients.

195 Certain questions in the production of CancerAssist and its use of an IWS may
 196 come into mind:

- 197 • When is the model updated and how do the IWS team communicate these
 198 updates?
- 199 • What benchmark test set of data ensures that the changed model doesn't affect
 200 other results?
- 201 • Are assumptions made by the IWS team who train the model correct?

202 Thus, to improve communication between developers and IWS providers, develop-
 203 ers require enhanced documentation, additional metadata, and guidance tooling.

204 1.3 Research Motivation

205 ⟨ *todo: MA: It is useful to highlight the takeaways of each paragraph - e.g. proba-*

206 *bilistic nature of ML, Choosing a threshold problem, lack of understanding of how*

207 *ml works, documentation style of ml models. then at the end of the section, clearly*

208 *state the problem/gap considered* ⟩ ⟨ *todo: AC: Have re-done a substantial part of*

209 *this section* ⟩

210 Evermore applications are using IWSs as demonstrated by ubiquitous examples:
 211 aiding the vision-impaired [74, 235], accounting [183], data analytics [137], and
 212 student education [79]. As our motivating examples have illustrated, these AI-based
 213 components—specifically CVSSs—are accessible through APIs consisting of ‘black
 214 box’ intelligence (Figure 1.3).⁹ Data science teams produce ML algorithms to make
 215 predictions in our datasets and discover patterns within them. As these algorithms
 216 are data-dependent, they are therefore inherently probabilistic and stochastic, which
 217 results in four critical issues that motivate our thesis: (i) certainty in results, (ii)
 218 evolution of datasets, (iii) selecting appropriate decision boundaries, and (iv) the
 219 clarity of ML documentation that address items i–iii.

220 There is little room for certainty in these results as the insight is purely statistical
 221 and associational [223] against its training dataset. Developers who build these
 222 applications do not treat their programs with a stochastic or probabilistic mindset,
 223 given that they are trained with a rule-driven mindset that computers make certain
 224 outcomes. However, CVSSs are data-driven, and therefore return the *probability* that
 225 a particular object exists in an input images’ pixels via confidence values. As an
 226 example, consider simple arithmetic representations (e.g., $2 + 2 = 4$). The deter-
 227 ministic (rule-driven) mindset suggests that the result will *always* be 4. However,
 228 the non-deterministic (data-driven) mindset suggests that results are probable: target
 229 output (*exactly* 4) and the output inferred (*a likelihood of* 4) matches as a probable

⁹The ‘black box’ refers to a system that transforms input (or stimulus) to outputs (or response) without any understanding of the internal architecture by which this transformation occurs. This arises from a theory in the electronic sciences and adapted to wider applications since the 1950s–60s [12, 48] to describe “systems whose internal mechanisms are not fully open to inspection” [12].

230 percentage (or as an error where it does not match).¹⁰ Instead of an exact output,
231 there is a *probabilistic* result: $2 + 2$ *may* equal 4 to a confidence of n . Thus, for a
232 more certain (though not fully certain) distribution of overall confidence returned
233 from the service, a developer must treat the problem stochastically by testing this
234 case hundreds if not thousands of times to find a richer interpretation of the inference
235 made and ensure reliability in its outcome.

236 Traditional software engineering principles advocate for software systems to be
237 versioned upon substantial change, but unfortunately we find that the most prominent
238 cloud vendors providing these intelligent services (e.g., Microsoft Azure, Google
239 Cloud and Amazon Web Services) do not release new versioned endpoints of the
240 APIs when the *internal model* changes [69]. In the context of computer vision, new
241 labels may be introduced or dropped, confidence values may differ, entire ontologies
242 or specific training parameters may change, but we hypothesise that is not effectively
243 communicated to developers. Broadly speaking, this can be attributed to a dichotomy
244 of release cycles from the data science and software engineering communities: the
245 data science iterations and work by which new models are trained and released runs
246 at a faster cycle than the maintenance cycle of traditional software engineering. Thus
247 we see cloud vendors integrating model changes without the *need* to update the API
248 version unless substantial code or schema changes are also introduced—the nuance
249 changes in the internal model does not warrant a shift in the API itself, and therefore
250 the version shift in a new model does not always propagate to a version shift in the
251 API endpoint. As demonstrated in Table 1.3, whatever input is uploaded at one time
252 may not necessarily be the same when uploaded at a later time. This again contrasts
253 the rule-driven mindset, where $2 + 2$ *always* equals 4. Therefore, in addition to the
254 certainty of a result in a single instance, the certainty of a result in *multiple instances*
255 may differ with time, which again impacts on the developers notion of reliable
256 software. Currently, it is impossible to invoke requests specific to a particular model
257 that was trained at a particular date in time, and therefore developers need to consider
258 how evolutionary changes of the services may impact their solutions *in production*.
259 Again, whether there is any noticeable behavioural changes from these changes is
260 dependent on the context of the problem domain—unless developers benchmark
261 these changes against their own domain-specific dataset and frequently check their
262 selected service against such a dataset, there is no way of knowing if substantive
263 errors have been introduced.

264 Further, the only response in these computer vision classifiers are a label and
265 confidence value; the decision boundaries need to be appropriately considered for
266 each use case and each model selected. The external quality of such software needs to
267 consider reliability in the case of thresholding confidence values—that is whether the
268 inference has an appropriate level of confidence to justify a predicted (and reliable)
269 result to end-users. Selecting this confidence threshold is non-trivial; a ML course
270 from Google suggests that “it is tempting to assume that [a] classification threshold
271 should always be 0.5, but thresholds are problem-dependent, and are therefore values
272 that you must tune.” [113]. Approaches to turning these values are considered for

10Blake et al. [33] produces a multi-layer perceptron neural network performing arithmetic representation.

²⁷³ data scientists, but are not yet well-understood for application developers with little
²⁷⁴ appreciation of the nuances of ML.

²⁷⁵ Similarly, developers should consider the internal quality of building AI-first
²⁷⁶ software. Reliable API usability and documentation advocate for the accuracy,
²⁷⁷ consistency and completeness of APIs and their documentation [228, 242] and
²⁷⁸ providers should consider mismatches between a developer’s conceptual knowledge
²⁷⁹ of the API its implementation [157]. Unreliable APIs ultimately hinder developer
²⁸⁰ performance and thus reduces productivity, in addition to producing potentially
²⁸¹ unreliable software where documentation is not well-understood (or clear to the
²⁸² developer).

²⁸³ Ultimately, these four issues present major threats to software reliability if left
²⁸⁴ unresolved. Given that such substantiative software engineering principles on re-
²⁸⁵ liability, versioning and quality are under-investigated within the context of IWSs,
²⁸⁶ we aim to explore guidance from the software engineering literature to investigate
²⁸⁷ what aspects in the development lifecycle could aide in mitigating these issues when
²⁸⁸ developing using AI-based components. This serves as our core motivation for this
²⁸⁹ work.

²⁹⁰ 1.4 Research Goals

²⁹¹ *(todo: JG: is it JUST CVS, or they one exemplar of intelligent APIs/services?? AC:
²⁹² Re-wrote... Tried to clarify that in this first sentence.)*

²⁹³ This thesis aims to investigate and better understand the nature of cloud-based
²⁹⁴ computer vision services (CVSs)¹¹ as a concrete exemplar of intelligent web services
²⁹⁵ (IWSs). We identify the maturity, viability and risks of CVSs through the anchoring
²⁹⁶ perspective of *reliability* that affects the internal and external quality of software.
²⁹⁷ We adopt the McCall [186] and Boehm [35] interpretations of reliability via the sub-
²⁹⁸ characteristics of a service’s *consistency* and *robustness* (or fault/error tolerance), and
²⁹⁹ the *completeness*¹² of its documentation. (A detailed discussion is further provided
³⁰⁰ in Section 2.1.) This thesis explores and contributes towards *four* key facets regarding
³⁰¹ reliability in CVS usage and the completeness of its associated documentation. We
³⁰² formulate four primary research questions (RQs) with seven sub-RQs, based on
³⁰³ both empirical and non-empirical software engineering methodology [194], further
³⁰⁴ discussed in Chapter 3.

³⁰⁵ Firstly, we investigate adverse implications that arise when using CVSs that
³⁰⁶ affects consistency and robustness (**Chapter 4**). We show how CVSs have a non-
³⁰⁷ deterministic runtime behaviour and evolve with unintended and non-trivial con-
³⁰⁸ sequences to developers. We demonstrate that these services have inconsistent
³⁰⁹ behaviour despite offering the same functionality and pose evolution risk that ef-
³¹⁰ fects robustness of consuming applications when responses change given the same
³¹¹ (consistent) inputs. Thus, we conclude how the nature of these services (at present)

¹¹As these services are proprietary, we are unable to conduct source code or model analysis, and hence are not used in the investigation of this thesis.

¹²We treat the API documentation of a CVS as a first-class citizen.

312 are not fully robust, consistent, and thus not reliable. Formally, we structure the
313 following RQs:

?

RQ1. What is the nature of cloud-based CVSs?

- RQ1.1.* What is their runtime behaviour?
- RQ1.2.* What is their evolution profile?

314 Secondly, we investigate the reliability of the documentation these services of-
315 fer through the lenses of its completeness. We collate prior knowledge of good
316 API documentation and assess the efficacy of such knowledge against practition-
317 ers (**Chapter 5**). We show that these service's behaviour and evolution is not
318 reliably documented adequately against this knowledge. Formally, we develop the
319 following RQs:

?

RQ2. Are CVS APIs sufficiently documented?

- RQ2.1.* What are the dimensions of a '*complete*' API document, ac-
cording to both literature and practitioners?
- RQ2.2.* What additional information or attributes do application de-
velopers need in CVS API documentation to make it more
complete?

320 Thirdly, we investigate how software developers approach using these services
321 and directly assess developer pain-points resulting from the nature of CVSs and
322 their documentation (**Chapter 6**). We show that there is a statistically significant
323 difference in these complaints when contrasted against more established software
324 engineering domains (such as web or mobile development) as expressed as ques-
325 tions asked on Stack Overflow. We provide a number of exploratory avenues for
326 researchers, educators, software engineers and IWS providers to alleviate these com-
327 plaints based on this analysis. Further, using a data set consisting of 1,245 Stack
328 Overflow questions, we explore the emotional state of developers to understand
329 which aspects (i.e., pain-points) developers are most frustrated with (**Chapter 7**).
330 We formulate the following RQs:

?

RQ3. Are CVSs more misunderstood than conventional software en- gineering domains?

- RQ3.1.* What types of issues do application developers face most when
using CVSs, as expressed as questions on Stack Overflow?
- RQ3.2.* Which of these issues are application developers most frus-
trated with?
- RQ3.3.* Is the distribution CVS pain-points different to established
software engineering domains, such as mobile or web devel-
opment?

331 Lastly, we explore several strategies to help improve CVSs reliability. Firstly,
332 we investigate whether merging the responses of *multiple* CVSs can improve their
333 reliability and propose a novel algorithm—based on the proportional representation
334 method used in electoral systems—to merge labels and associated confidence values
335 from three providers (**Chapter 8**). Secondly, we develop an integration architecture
336 style (or facade) to guard against CVS evolution, and synthesise an integration
337 workflow that addresses the concerns raised by developers in addition to embedding
338 ‘complete’ documentation artefacts into the workflow’s design (**Chapters 9 and 10**).
339 Our final RQ is:

340 **② RQ4. What strategies can developers employ to integrate their applications with CVSs while preserving robustness and reliability?**

1.5 Research Methodology

341 This thesis employs a mixed-methods approach using the concurrent triangulation
342 strategy [43, 185]. The research presented consists of both empirical and non-
343 empirical research design. This section provides a high-level overview of the re-
344 search methodology within this thesis. Further details are provided in Section 1.7
345 and Chapter 3.

346 Firstly, RQ1–RQ3 are all empirical, knowledge-based questions [87, 191] that
347 aim to provide the software engineering community with a greater understanding
348 of the phenomena surrounding CVSs from three perspectives: the nature of the ser-
349 vices themselves, how developers perceive these services and how service providers
350 can improve these services. We answer RQ1 using a longitudinal experiment that
351 assesses both the services’ responses and associated documentation (complement-
352 ing RQ2.2). We adopt qualitative and quantitative data collection; specifically (i)
353 structured observations to quantitatively analyse the results over time, and (ii) docu-
354 mentary research methods to inspect service documentation. Secondly, we perform
355 systematic mapping study following the guidelines of Kitchenham and Charters
356 [153] and Petersen et al. [225] to better understand how API documentation of these
357 services can be improved (i.e., more complete), which targets Item RQ2. Based on
358 the findings from this study, we use a systematic taxonomy development methodol-
359 ogy specifically targeted toward software engineering [284] that structures scattered
360 API documentation knowledge into a taxonomy. We then validate this taxonomy
361 against practitioners using survey research, adopting Brooke well-established Sys-
362 tematic Usability Score [47] surveying instrument and contextualising it within API
363 documentation utility, which answers RQ3.3. To answer RQ2.2, we perform an
364 empirical application of the taxonomy to three CVSs, and therefore assess where
365 improvements can be made. Thirdly, we adopt field survey research using repository
366 mining of developer discussion forums (i.e., Stack Overflow) to answer RQ3, and
367 classify these using both manual and automated techniques.

368 The second aspect of our research design involves non-empirical research, which
369 explores a design-based question [194] to answer RQ4. As the answers to our

370 first three RQs establish a greater understanding of the nature behind CVSSs from
371 various perspectives, the strategies we design in RQ4 aims at designing more reliable
372 integration methods so that developers can better use these cloud-based services in
373 their applications.

374 1.6 Thesis Organisation

375 We organise the thesis into four parts. **Part I (The Preface)** includes introductory
376 background and methodology chapters. This is a *PhD by Publication*, and
377 **Part II (Publications)** comprises of seven publications resulting from this work
378 over Chapters 4 to 10; publications are included verbatim except for terminology
379 and formatting changes to better fit the suitability of a coherent thesis. **Part III (The**
380 **Postface)** includes the conclusion and future works chapter, as well as a list of aca-
381 demic studies and online artefacts referenced within the thesis. **Part IV (Appendices)**
382 includes all supplementary material, including mandatory authorship statements and
383 ethics approval. Details of each chapter following this introductory chapter are pro-
384 vided in the following section.

385 Part I: Preface

386 **Chapter 2: Background** This chapter provides an overview of prior studies
387 broadly around three key pillars: the development of an IWS, the usage of an IWS,
388 and the nature of an IWS. We use the three perspectives of software quality (partic-
389 ularly, reliability), probabilistic and non-deterministic systems, and explanation and
390 communication theory to describe prior work.

391 **Chapter 3: Research Methodology** This chapter provides a summative review
392 of research methods and philosophical stances relevant to software engineering. We
393 illustrate that the methods used within our publications are sound via an analysis of
394 the methodologies used in seminal works referenced in this thesis.

395 Part II: Publications

396 **Chapter 4: Exploring the nature of CVSSs** This chapter was presented at the 2019
397 International Conference on Software Maintenance and Evolution (ICSME) [69].
398 We describe an 11-month longitudinal experiment assessing the behavioural (run-
399 time) issues of three popular CVSSs: Google Cloud Vision [327], Amazon Rekogni-
400 tion [313] and Azure Computer Vision [335]. By using three different data sets—two
401 of which we curate as additional contributions—we demonstrate how the services
402 are inconsistent amongst each other and within themselves. This study provides a
403 detailed answer to RQ1: Despite presenting conceptually-similar functionality, each
404 service behaves and produces slightly varied (inconsistent) results and demonstrates
405 non-deterministic runtime behaviour. We discuss potential evolution risks to con-
406 sumers of such services as the services provide non-static outputs for the same inputs,

407 thereby having significant impact to the robustness of consuming applications. Fur-
408 ther details in the study include a brief assessment into the lack of sufficient detail
409 of these concerns in their documentation.

410 **Chapter 5: Investigating improvements to CVS API documentation** This chap-
411 ter was originally a short paper presented at the 2019 International Symposium on
412 Empirical Software Engineering and Measurement (ESEM) [71]. To understand
413 where to improve CVS documentation, we first need to investigate *what* makes a
414 good API document. This short paper initially answered one aspect of RQ2.1: what
415 *academic literature* suggests a good (complete) API document should comprise of.
416 By conducting an systematic mapping study resulting in 21 primary studies, we
417 systematically develop a taxonomy that combines the recommendations of scattered
418 work into a structured framework of 5 dimensions and 34 weighted categorisa-
419 tions. We then extend this work¹³ by triangulating the taxonomy with opinions from
420 developers using the System Usability Scale to assess the efficacy of these recom-
421 mendations (thereby answering the second aspect of RQ2.1). From this, we assess
422 the how well CVS providers document their APIs via a heuristic validation of the
423 taxonomy, using the three services from the ICSME publication to make recom-
424 mendations where documentation should be more complete, thereby answering RQ2.2
425 (and thus RQ2).

426 **Chapter 6: Understanding developer struggles when using CVSs** This chapter
427 has been accepted for presentation at the 2020 International Conference on Software
428 Engineering (ICSE) [71]. We conduct a mining study of 1,425 Stack Overflow
429 questions that provide indications of the types frustrations that developers face when
430 integrating CVSs into their applications. To gather what their pain-points are,
431 we use two classification taxonomies that also use Stack Overflow to understand
432 generalised and documentation-specific pain-points in mature software engineering
433 (SE) domains. This study answers RQ3 in detail and provides a validation to
434 our motivation of RQ2: we validate that the *completeness* of current CVS API
435 documentation is a main concern for developers and there is insufficient explanation
436 into the errors and limitations of the service. We find that the documentation does
437 not adequately cover all aspects of the technical domain. In terms of integrating with
438 the service, developers struggle most with simple errors and ways in which to use the
439 APIs; this is in stark contrast to mature software domains. Our interpretation is that
440 developers fail to understand the IWS lifecycle and the ‘whole’ system that wraps
441 such services. We also interpret that developers have a shallower understanding
442 of the core issues within CVSs (likely due to the nuances of ML as suggested in
443 a discussion in the paper), which warrants an avenue for future work in software
444 engineering education.

445 **Chapter 7: Ranking CVS pain-points by frustration** This chapter has been
446 submitted to the the 2020 International Workshop on Emotion Awareness in Software

¹³The extended version of this chapter has been submitted to ([TODO: Revise the Journal of Systems and Software](#)) in [72] and is currently within review.

447 Engineering (SEmotion) [73]. In this work, we use our dataset consisting of the 1,425
448 Stack Overflow (SO) questions from [71] to interpret the breakdown of emotions
449 developers express per classification of pain-points conducted in Chapter 6. We find
450 that the distribution of various emotions differ per question type, and developers are
451 most frustrated when the expectations of a CVS does not match the reality of what
452 these services actually provide, which shapes our answer for RQ3.2 and thus RQ3.

453 **Chapter 8: Merging responses of multiple CVSs** This chapter was presented
454 at the 2019 International Conference on Web Engineering (ICWE) [212]. Early
455 exploration of CVSs showed that multiple services use vastly different ontologies
456 for the same input. As an initial strategy to improve the reliability of these services,
457 we explored if merging multiple responses using WordNet [196] and a novel label
458 merging algorithm based on the proportional representation approach used in polit-
459 ical voting could make any improvements. While this approach resulted in a modest
460 improvement to reliability, it did not consider to the evolution issues or developer
461 pain-points we later identified.

462 **Chapter 9: Developing a confidence thresholding tool** This chapter has been
463 submitted to the demonstrations track at ICSE 2020 [70]. When integrating with a
464 CVS, developers need to select an appropriate confidence threshold suited to their
465 use case and determine whether a decision should be made. An issue, however, is
466 that these CVSs are not calibrated to the specific problem-domain datasets and it is
467 difficult for software developers to determine an appropriate confidence threshold
468 on their problem domain. This tool presents a workflow and supporting tool for
469 application developers to select decision thresholds suited to their domain that—
470 unlike existing tooling—is designed to be used in pre-development, pre-release and
471 production. This tooling forms part of a solution to RQ4 for developers to maintain
472 robustness and reliability in their systems.

473 **Chapter 10: Developing a CVS integration architecture** This chapter has been
474 submitted to the 2020 Joint European Software Engineering Conference and Sympo-
475 sium on the Foundations of Software Engineering [67]. *([todo: Discuss findings.](#))*

476 **Part III: Postface**

477 **Chapter 11 - Conclusions & Future Work** In this chapter, we review the con-
478 tributions made in this thesis and the relevance and significance to identifying and
479 resolving key issues when application developers integrate with CVS. We evaluate
480 these outcomes with reference to the research goals, and discuss threats to validity
481 of the work. Lastly, we discuss the various avenues of research arising from this
482 work.

483 Part IV: Appendices

484 Appendix A provides additional material referenced within this thesis but not pro-
 485 vided in the body. We provide mandatory coauthor declaration forms describing
 486 the contribution breakdown for each publication within Appendix B. Appendix C
 487 contains copies of the ethics clearance for various experiments within this thesis.
 488 We describe the list of primary sources arising in the systematic mapping study we
 489 conduct in Chapter 5 within Appendix D.

490 1.7 Research Contributions

491 *< todo: would be good to have more detail HERE on the key contributions made in this
 492 work; Please include more details of the contributions - e.g. which research question
 493 is addressed in this contribution, key findings, you could also list the publications
 494 associated with each one. Also in this case you do not need section 1.4.2 >* < todo:
 495 AC: Have merged 1.4.2 into this; now the coherency of publications + contributions
 496 made are one, which makes sense given thesis by pub. > The outcomes of answering
 497 the four primary research questions elaborated in Section 1.4 shapes three primary
 498 contributions this thesis offers to software engineering knowledge:

- 499 • An improved understanding in the landscape of CVSs, with respect to their
 500 runtime behaviour and evolutionary profiles.
- 501 • A service integration architecture that helps developers with integrating their
 502 applications with CVSs.
- 503 • A list of attributes that should be documented to assist CVS providers to better
 504 document their services.

505 In this section, we detail how each publication forms a coherent body of work
 506 and how each publication relates to the primary contributions made. After our
 507 exploratory analysis on the nature of CVSs (Chapter 4), we proposed two sets of
 508 recommendations targeted towards two stakeholders: (i) the service *consumers* (i.e.,
 509 application developers) and (ii) the service *providers*. Our subsequent publications
 510 arose as a two-fold investigation to develop two strategies in which developers and
 511 providers can, respectively, (i) better integrate these intelligent components into
 512 their applications, and (ii) how these services can be better documented. Table 1.5
 513 provides a tabulated form of the publications and research questions addressed within
 514 this thesis; for ease of reference, we refer to the publications in within this section
 515 in their abbreviated form as listed in Table 1.5. We also provide abbreviations for
 516 easier reference in this section. A high-level overview of the cohesiveness of our
 517 publications is provided in Figure 1.5.

¹⁵Conference publications ranking measured using the CORE Conference Ranks (<http://www.core.edu.au/conference-portal>) and Journal publications rankings using the Scimago Ranking (<https://www.scimagojr.com/>). Rankings retrieved January 2020.

¹⁶Date of publication, if applicable.

¹⁷The extended version of this conference proceeding is provided in Chapter 5.

¹⁸We abbreviate this with an added ‘d’ (for the demonstrations track) to distinguish this paper from our full ICSE 2020 paper.

Table 1.5: List of publications resulting from this thesis, separated by phenomena exploration (above) and solution design (below).

Ref.	Venue	Acronym	Rank ¹⁴	Published ¹⁵	Chapter	RQs
[69]	35 th International Conference on Software Maintenance and Evolution	ICSME	A	05 Dec 2019	Chapter 4	RQ1
[68]	13 th International Symposium on Empirical Software Engineering and Measurement	ESEM	A	17 Oct 2019	Excluded ¹⁶	RQ2.1
[72]	Journal of Systems and Software	JSS	Q1	<i>In Press</i>	Chapter 5	RQ2
[71]	42 nd International Conference on Software Engineering	ICSE	A*	<i>In Press</i>	Chapter 6	RQ3
[73]	5 th International Workshop on Emotion Awareness in Software Engineering ¹⁷	SEmotion	A*	<i>In Press</i>	Chapter 7	RQ3.2
[212]	13 th International Conference on Web Engineering	ICWE	B	26 Apr 2019	Chapter 8	RQ4
[70]	42 nd International Conference on Software Engineering	ICSE(d) ¹⁸	A*	<i>In Press</i>	Chapter 9	RQ4
[67]	28 th Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering	ESEC/FSE	A*	<i>In Press</i>	Chapter 10	RQ4

518 1.7.1 Contribution 1: Landscape Analysis & Preliminary Solutions

519 The first two bodies of work in this paper are the ICSME and ICWE papers. These
 520 two works investigated a landscape analysis CSVs from two perspectives: firstly, we
 521 conducted a longitudinal study to better understand the attributes associated with
 522 these services (ICSME)—particularly their evolution and behavioural profiles, and
 523 their potential impacts to software reliability—and tackled a preliminary solution
 524 facade to ‘merge’ responses of the services together (ICWE).

525 The ICSME paper confirmed our hypotheses that the services have a non-
 526 deterministic behavioural profile, and that the evolution occurring within the ML
 527 models powering these services are not sufficiently communicated to software en-
 528 gineers. This therefore led to follow up investigation into how developers perceive
 529 these services, and thereby determine if they are frustrated due to this lack of com-
 530 munication.

531 Our ICWE paper explored one aspect identified from the ICSME paper that
 532 we identified early on: that different services use different vocabularies to describe
 533 semantically similar objects but in different ways (e.g., ‘border collie’ vs. ‘collie’),
 534 despite offering functionally similar capabilities. We attempted to merge the re-
 535 sponse labels from these services using a proportional representation approach, and
 536 upon comparison with more naive merge approaches, we improved label-merge per-
 537 formance by an F-measure of 0.015. However, while this was an interesting outcome
 538 for a preliminary solution design, investigation from our following work suggested
 539 that standardising ontologies between service providers becomes challenging and
 540 normalising the entire ontological hierarchy of response labels would need to fall
 541 under the responsibility of a certain body (that does not exist). Further, we did
 542 not find sufficient evidence that developers would frequently switch between service

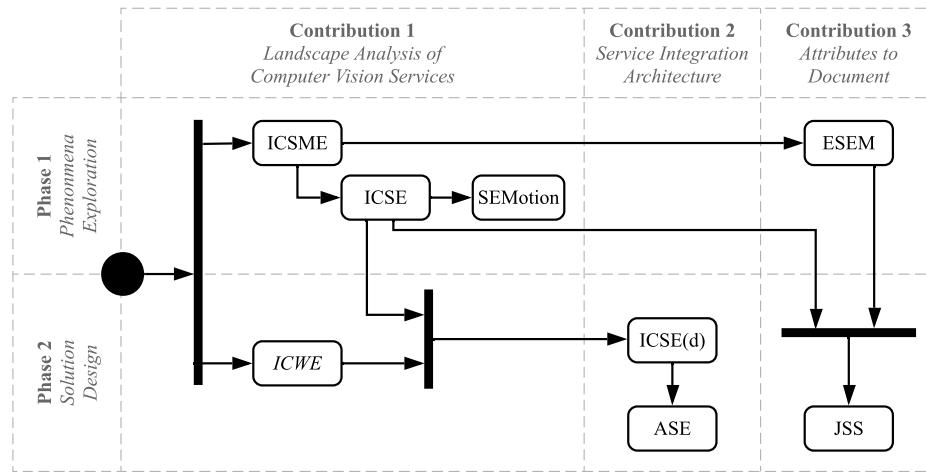


Figure 1.5: Activity diagram of the coherency of our publications, how our research was conducted, and relevant connections between publications. Our two-phase structure initial phenomena exploration and a proposed solutions to issues identified from the exploration. We map the contributions within each publication to the three primary contributions of the thesis.

543 providers. Therefore, we opted for a shielded relay architecture in our later design
 544 work.

545 1.7.2 Contribution 2: Improving Documentation Attributes

546 As mentioned, our ICSME paper found that evolutionary and non-deterministic
 547 behavioural profile of are not adequately documented in the service’s APIs docu-
 548 mentation. A recommendation concluding from this work was that service providers
 549 should improve their documentation, however there lacked a strategy by which they
 550 could do this, and our hypotheses that developers were actually frustrated by this
 551 lack of communication was yet to be tested. This led to two follow-up further
 552 investigations as presented in our ICSE and ESEM papers.

553 One aspect of our ICSE paper was to confirm whether developers are actually
 554 frustrated with the service’s limited API documentation. By mining Stack Overflow
 555 posts with reference to documentation issues, we adopted a 2019 documentation-
 556 related taxonomy by Aghajani et al. [2] to classify posts, and found that 47.87%
 557 of posts classified fell under the ‘completeness’ dimension of Aghajani et al.’s
 558 taxonomy. This interpretation, therefore, warranted the recommendation proposed
 559 in the ICSME paper to improve service documentation.

560 However, though improvements to more complete documentation was justified
 561 from the ICSE paper, we needed to explore exactly *what* makes a ‘complete’ API
 562 document. By conducting a systematic mapping study resulting in 4,501 results, we
 563 curated 21 primary studies that outline the facets of API documentation knowledge.
 564 From these studies, we distilled a documentation framework describing a prioritised

order of the documentation assets API’s should document that is described in our ESEM short paper. After receiving community feedback, we extended this short paper with a follow-up experiment submitted to JSS. By conducting a survey with developers, we assessed our API documentation taxonomy’s efficacy with practitioner opinions, thereby producing a weighted taxonomy against *both* literature and developer sources. Lastly, we triangulated both weightings against a heuristic evaluation against common CVS providers’ documentation. This allowed us to deduce which specific areas in existing CVS providers’ API documentation needed improvement, which was a primary contribution from our JSS article.

1.7.3 Contribution 3: Service Integration Architecture

Two recommendations from our ICSME study encouraged developers to test their applications with a representative ontology for their problem domain and to incorporate a specialised testing and monitoring techniques into their workflow. Strategies on *how* to achieve this were explored in later studies. Following a similar approach to our solution of improved API documentation, we validated the substantiveness of our recommendations using our mining study of Stack Overflow (our ICSE paper) to help inform us of generalised issues developers face whilst integrating CVSs into their applications. To achieve this, we used a Stack Overflow post classification taxonomy proposed by Beyer et al. [31] into seven categories, where 28.9% and 20.37% of posts asked issues regarding how to use the CVS API and conceptual issues behind CVSs, respectively. Developers presented an insufficient understanding of the non-deterministic runtime behaviour, functional capability, and limitations of these services and are not aware of key computer vision terminology. When contrasted to more conventional domains such as mobile-app development, the spread of these issues vary substantially.

We proposed two technical solutions in ICSE(d) and ESEC/FSE, respectively, to help alleviate this issue. *(todo: Revise this... needs to be fleshed out)* Firstly, our ICSE(d) paper provides a workflow for developers to better select an appropriate confidence threshold, and thus decision boundary, calibrated for their particular use case. In our ESEC/FSE paper, we provide a reference architecture for developers to guard against the non-deterministic issues that may ‘leak’ into their applications. This architecture is a facade style, similar to the style proposed in our ICWE paper, however, unlike the ICWE paper that uses proportional representation approach to modify multiple sources, our ESEC/FSE paper proposes a guarded relay, whereby a single service is used, and the facade should maintain a lifecycle to monitor evolution issues identified in ICSME and should be benchmarked against the developer’s dataset (i.e., against the particular application domain) as suggested in ICSE(d). These two primary contributions further serve as an answer to RQ4.

CHAPTER 2

603

604

605

Background

606

607 In Chapter 1, we defined a common set of (artificial) intelligence-based cloud ser-
608 vices that we label intelligent web services (IWSs). Specifically, we scope the
609 primary body of this study’s work on computer vision services (CVSs) (e.g., Google
610 Cloud Vision [327], AWS Rekognition [313], Azure Computer Vision [335], Watson
611 Visual Recognition [331] etc.). We claim developers have a distinctly determinis-
612 tic mindset ($2 + 2$ always equals 4) whereas an IWS’s ‘intelligence’ component (a
613 black box) may return probabilistic results ($2 + 2$ might equal 4 with a confidence
614 of 95%). Thus, there is a mindset mismatch between probabilistic results (from the
615 API provider) and results interpreted with certainty (from the API consumer).

616 What affect does this mindset mismatch have on the developer’s approach to-
617 wards building probabilistic software? What can we learn from common software
618 engineering practices (e.g., [230, 266]) that apply to resolve this mismatch and
619 thereby improve quality, such as verification & validation (V&V)? Chiefly, we an-
620 chor this question around three lenses of software engineering: creating an IWS,
621 using an IWS, and the nature of IWSs themselves.

622 Our chief concern lies with interaction and integration between IWS providers
623 and consumers, the nature of applications built using an IWS, and the impact this
624 has on software quality. We triangulate this around three pillars, which we diagram-
625 matically represent in Figure 2.1.

- 626 **(1) The development of the IWS.** We investigate the internal quality attributes
627 of creating an IWS from the IWS *provider’s* perspective. That is, we ask if
628 existing verification techniques are sufficient enough to ensure that the IWS
629 being developed actually satisfies the IWS consumer’s needs and if the internal
630 perspective of creating the system with a non-deterministic mindset clashes
631 with the outside perspective (i.e., pillar 2).
- 632 **(2) The usage of the IWS.** We investigate the external quality attributes of using
633 an IWS from the IWS *consumer’s* perspective. That is, we ask if existing
634 validation techniques are sufficient enough to ensure that the end-users can

635 actually use an IWS to build their software in the ways they expect the IWS to
 636 work.

637 **(3) The nature of an IWS.** We investigate what standard software engineering
 638 practices apply when developing non-deterministic systems. That is, we
 639 tackle what best practices exist when developing systems that are inherently
 640 stochastic and probabilistic, i.e., the ‘black box’ intelligence itself.

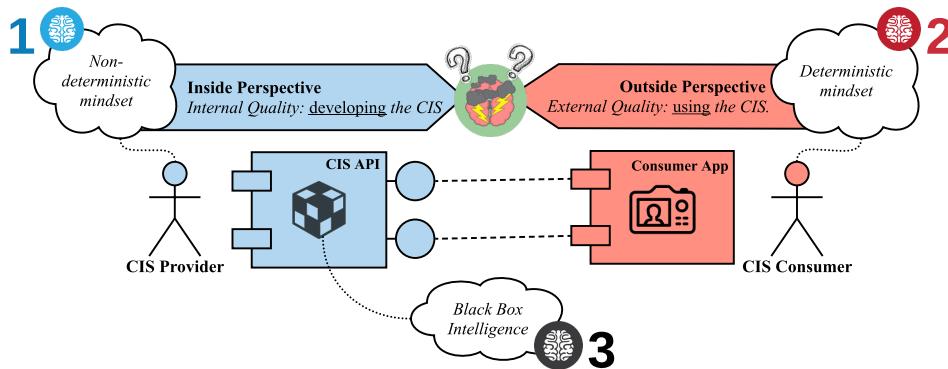


Figure 2.1: The three pillars by which we anchor the background: (1) developing an IWS with a non-deterministic mindset by the IWS provider; (2) the use of a IWS with a deterministic mindset by the IWS consumer; (3) the nature of a IWS itself.

641 Does a clash of deterministic consumer mindsets who use a IWS and the non-
 642 deterministic provider mindsets who develop them exist? And what impact does
 643 this have on the inside and outside perspective? Throughout this chapter, we will
 644 review these three core pillars due to such mindset mismatch from the anchoring per-
 645 spective of software quality, particularly around V&V and related quality attributes,
 646 probabilistic and nondeterministic software and the nature of APIs.

647 2.1 Software Quality

648 *Quality... you know what it is, yet you don't know what it is.*
 ROBERT PIRSIG, 1974 [229]

649 The philosophical viewpoint of ‘quality’ remains highly debated and there are mul-
 650 tiple facets to perceive this complex concept [105]. Transcendentally, a viewpoint
 651 like that of Pirsig’s above shows that quality is not tangible but still recognisable; it’s
 652 hard to explicitly define but you know when it’s missing. The International Orga-
 653 nization for Standardization provides a breakdown of seven universally-applicable
 654 principles that defines quality for organisations, developers, customers and training
 655 providers [135]. More pertinently, the 1986 ISO standard for quality was simply
 656 “the totality of characteristics of an entity that bear on its ability to satisfy stated or
 657 implied needs” [134].

Using this sentence, what characteristics exist for non-deterministic IWSs like that of a CVS? How do we know when the system has satisfied its ‘stated or implied needs’ when the system can only give us uncertain probabilities in its outputs? Such answers can be derived from related definitions—such as ‘conformance to specification or requirements’ [66, 109], ‘meeting or exceeding customer expectation’ [28], or ‘fitness for use’ [145]—but these then still depend on the solution description or requirements specification, and thus the same questions still apply.

Software quality is somewhat more concrete. Pressman [230] adapted the manufacturing-oriented view of quality from [29] and phrased software quality under three core pillars:

- **effective software processes**, where the infrastructure that supports the creation of quality software needs is effective, i.e., poor checks and balances, poor change management and a lack of technical reviews (all that lie in the *process* of building software, rather than the software itself) will inevitably lead to a poor quality product and vice-versa;
- **building useful software**, where quality software has fully satisfied the end-goals and requirements of all stakeholders in the software (be it explicit or implicit requirements) *in addition to* delivering these requirements in reliable and error-free ways; and lastly
- **adding value to both the producer and user**, where quality software provides a tangible value to the community or organisation using it to expedite a business process (increasing profitability or availability of information) *and* provides value to the software producers creating it whereby customer support, maintenance effort, and bug fixes are all reduced in production.

In the context of a non-deterministic IWS, however, are any of the above actually guaranteed? Given that the core of a system built using an IWS is fully dependent on the *probability* that an outcome is true, what assurances must be put in place to provide developers with the checks and balances needed to ensure that their software is built with quality? For this answer, we re-explore the concept of verification & validation (V&V).

2.1.1 Validation and Verification

To explain V&V, we analogously recount a tale given by Pham [227] on his works on reliability. A high-school student sat a standardised test that was sent to 350,0000 students [275]. A multiple-choice algebraic equation problem used a variable, a , and intended that students *assume* that the variable was non-negative. Without making this assumption explicit, there were two correct answers to the multiple choice answer. Up to 45,000 students had their scores retrospectively boosted by up to 30 points for those who ‘incorrectly’ answered, however, outcomes of a student’s higher education were, thereby, affected by this one oversight in quality assessment. The examiners wrote a poor question due to poor process standards to check if their ‘correct’ answers were actually correct. The examiners “didn’t build the right product” nor did they “build the product right” by writing an poor question and

700 failing to ensure quality standards, in the phrases Boehm [37] coined.

701 This story describes the issues with the cost of quality [36] and the importance
702 of V&V: just as the poorly written exam question had such a high toll the 45,000
703 unlucky students, so does poorly written software in production. As summarised by
704 Pressman [230], data sourced from Digital [60] in a large-scale application showed
705 that the difference in cost to fix a bug in development versus system testing is
706 \$6,159 per error. In safety-critical systems, such as self-driving cars or clinical
707 decision support systems, this cost skyrockets due to the extreme discipline needed
708 to minimise error [277].

709 Formally, we refer to the IEEE Standard Glossary of Software Engineering
710 Terminology [132] for to define V&V:

711 **verification** The process of evaluating a system or component to determine
712 whether the products of a given development phase satisfy the
713 conditions imposed at the start of that phase.

714 **validation** The process of evaluating a system or component during or at the
715 end of the development process to determine whether it satisfies
716 specified requirements.

717 Thus, in the context of an IWS, we have two perspectives on V&V: that of the API
718 provider and consumer (Figure 2.2).

719 The verification process of API providers ‘leak’ out to the context of the de-
720 veloper’s project dependent on the IWS. Poor verification in the *internal quality*
721 of the IWS will entail poor process standards, such as poor definitions and termi-
722 nology used, support tooling and description of documentations [266]. Though
723 it is commonplace for providers to have a ‘ship-first-fix-later’ mentality of ‘good-
724 enough’ software [286], the consequence of doing so leads to consumers absorb-
725 ing the cost. Thus API providers must ensure that their verification strategies
726 are rigorous enough for the consumers in the myriad contexts they wish to use
727 it in. Studies have considered V&V in the context of web services on the cloud
728 [15, 51, 52, 94, 121, 203, 205, 304], though little have recently considered how
729 adding ‘intelligence’ to these services affects existing proposed frameworks and
730 solutions. For a CVS, what might this entail? Which assurances are given to the
731 consumers, and how is that information communicated? To verify if the service is
732 working correctly, does that mean that we need to deploy the system first to get a
733 wider range of data, given the stochastic nature of the black box?

734 Likewise, the validation perspective comes from that of the consumer. While the
735 former perspective is of creation, this perspective comes from end-user (developer)
736 expectation. As described in Chapter 1, a developer calls the IWS component using
737 an API endpoint. Again, the mindset problem arises; does the developer know what
738 to expect in the output? What are their expectations for their specific context? In
739 the area of non-deterministic systems of probabilistic output, can the developer be
740 assured that what they enter in a testing phase outcome the same result when in
741 production?

742 Therefore, just as the test answers with were both correct and incorrect at the
743 same time, so is the same with IWSs returning a probabilistic result: no result is

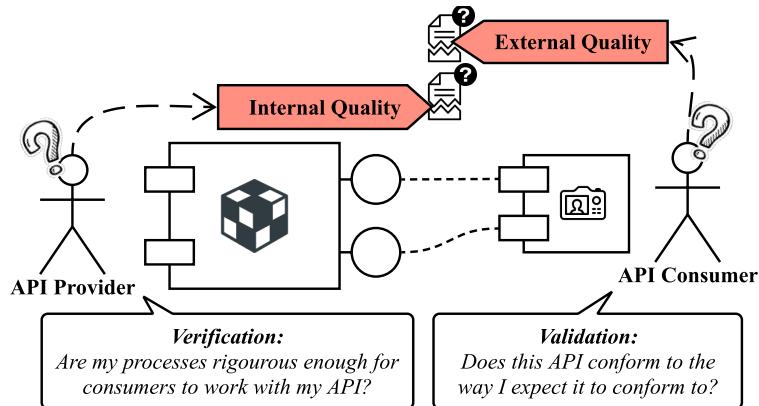


Figure 2.2: The ‘leakage’ of internal quality into the API consumer’s product and external quality imposing on the API provider.

744 certain. While V&V has been investigated in the area of mathematical and earth
 745 sciences for numerical probabilistic models and natural systems [214, 250], from
 746 the software engineering literature, little work has been achieved to look at the
 747 surrounding area of probabilistic systems hidden behind API calls.

748 Now that a developer is using a probabilistic system behind a deterministic API
 749 call, what does it mean in the context of V&V? Do current verification approaches
 750 and tools suffice, and if not, how do we fix it? From a validation perspective of
 751 ML and end-users, after a model is trained and an inference is given and if the
 752 output data point is incorrect, how will end users report a defect in the system?
 753 Compared to deterministic systems where such tooling as defect reporting forms are
 754 filled out (i.e., given input data in a given situation and the output data was X), how
 755 can we achieve similar outputs when the system is not non-deterministic? A key
 756 problem with the probabilistic mindset is that once a model is ‘fixed’ by retraining
 757 it, while one data-point may be fixed, others may now have been effected, thereby
 758 not ensuring 100% validation. Thus, due to the unpredictable and blurry nature of
 759 probabilistic systems, V&V must be re-thought out extensively.

760 2.1.2 Quality Attributes and Models

761 Similarly, quality models are used to capture internal and external quality attributes
 762 via measurable metrics. Is a similar issue reflected from that of V&V due to
 763 nondeterministic systems? As there is no ‘one’ definition of quality, there have been
 764 differing perspectives with literature placing varying value on disparate attributes.

765 Quality attribute assessment models (like those shown in Figure 2.3) are an early
 766 concept in software engineering, and systematically evaluating software quality
 767 appears as early as 1968 [249]. Rubey and Hartwick’s 1968 study introduced the
 768 phrase ‘attributes’ as a “prose expression of the particular quality of desired software”
 769 (as worded by Boehm et al. [35]) and ‘metrics’ as mathematical parameters on a
 770 scale of 0 to 100. Early attempts to categorise wider factors under a framework was
 771 proposed by McCall, Richards, and Walters in the late 1970s [55, 186]. This model

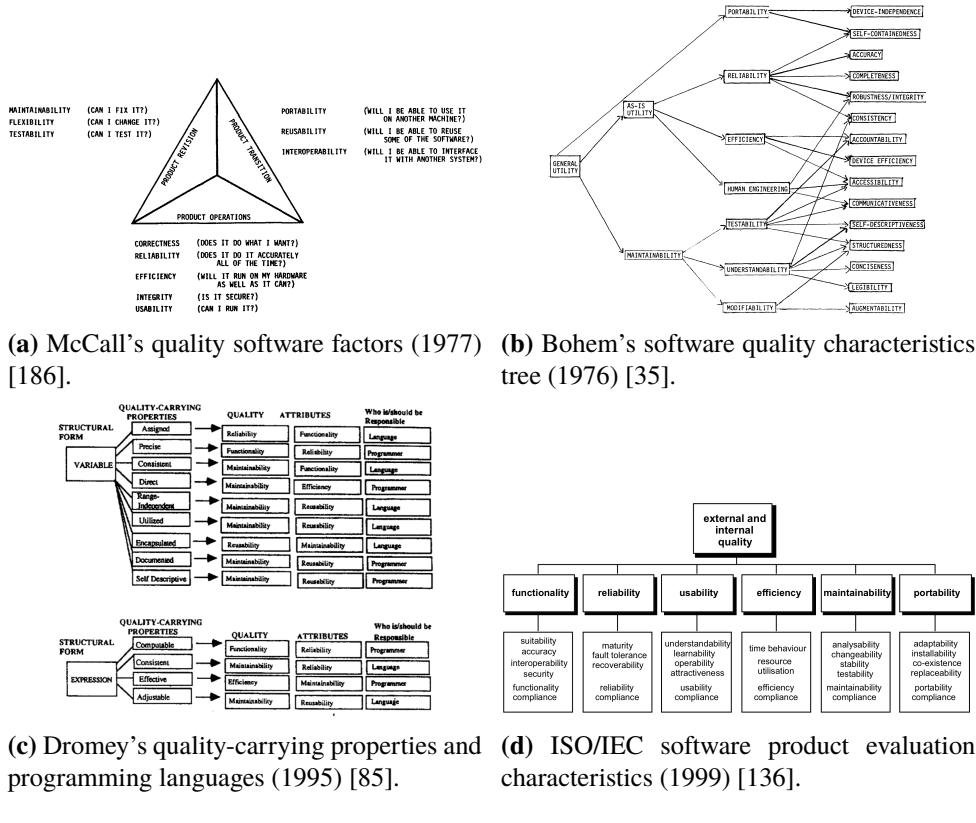


Figure 2.3: A brief overview of the development of software quality models since 1977.

described quality from the three perspectives of product revision (*how can we keep the system operational?*), transition (*how can we migrate the system as needed?*) and operation (*how effective is the system at achieving its tasks?*) (Figure 2.3a). The model also introduced 11 attributes alongside numerous direct and indirect measures to help quantify quality. This model was further developed by Boehm et al. [35] who independently developed a similar model, starting with an initial set of 11 software characteristics. It further defined candidate measurements of Fortran code to such characteristics, taking shape in a tree-like structure as in Figure 2.3b. In the mid-1990s, Dromey's interpretation [85] defined a set of quality-carrying properties with structural forms associated to specific programming languages and conventions (Figure 2.3c). The model also supported quality defect identification and proposed an improved auditing method to automate defect detection for code editors in IDEs. As the need for quality models became prevalent, the International Organization for Standardization standardised software quality under ISO/IEC-9126 [136] (the Software Product Evaluation Characteristics, Figure 2.3d), which has since recently been revised to ISO/IEC-25010 with the introduction of the Systems and software Quality Requirements and Evaluation (SQuaRE) model [133], separating quality into *Product Quality* (consisting of eight quality characteristics and 31 sub-characteristics) and *Quality In Use* (consisting of five quality characteristics and 9 sub-characteristics). An extensive review on the development of quality models in

792 software engineering is given in [5].

793 Of all the models described, there is one quality attribute that relates most
 794 with our narrative of IWS quality: reliability. Reliability is the primary quality
 795 factor investigated within this thesis (see Section 1.4). Both McCall and Boehm’s
 796 quality models have sub-characteristics of reliability relating to the primary research
 797 questions that investigate the *robustness*, *consistency* and *completeness*¹ of CVSs
 798 and its associated documentation. Moreover, the definition of reliability is similar
 799 among all quality models:

800 **McCall et al.** Extent to which a program can be expected to perform its in-
 801 tended function with required precision [186].

802 **Boehm et al.** Code possesses the characteristic *reliability* to the extent that
 803 it can be expected to perform its intended functions satisfac-
 804 torily [35].

805 **Dromey** Functionality implies reliability. The reliability of software is
 806 therefore dependent on the same properties as functionality, that
 807 is, the correctness properties of a program [85].

808 **ISO/IEC-9126** The capability of the software product to maintain a specified
 809 level of performance when used under specified conditions [136].

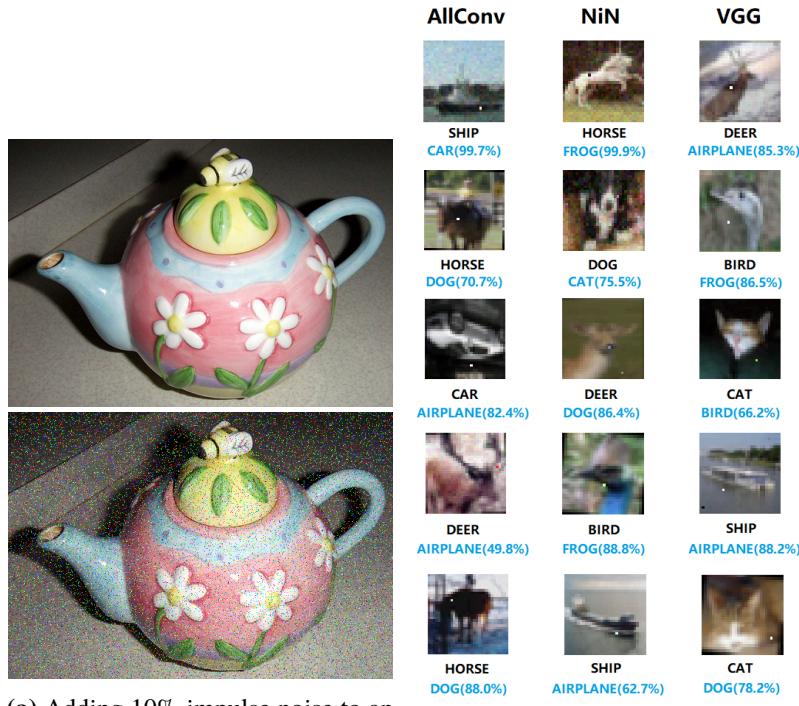
810 These definitions strongly relate to the system’s solution description in that
 811 reliability is the ability to maintain its *functionality* under given conditions. But what
 812 defines reliability when the nature of an IWS in itself is inherently unpredictable
 813 due to its probabilistic implementation? Can a non-deterministic system ever be
 814 considered reliable when the output of the system is uncertain? How do developers
 815 perceive these quality aspects of reliability in the context of such systems? A system
 816 cannot be perceived as ‘reliable’ if the system cannot reproduce the same results due
 817 to a probabilistic nature. Therefore, we believe the literature of quality models does
 818 not suffice in the context of IWS reliability; a CVS can interpret an image of a dog
 819 as a ‘Dog’ one day, but what if the next it interprets such image more specifically to
 820 the breed, such as ‘Border Collie’? Does this now mean the system is unreliable?

821 Moreover, defining these systems in themselves is challenging when require-
 822 ments specifications and solution descriptions are dependent on nondeterministic
 823 and probabilistic algorithms. We discuss this further in Section 2.2.

824 **2.1.3 Reliability in Computer Vision**

825 Testing computer vision deep-learning reliability is an area explored typically
 826 through the use of adversarial examples [273]. These input examples are where
 827 images are slightly perturbed to maximise prediction error but are still interpretable
 828 to humans. Refer to Figure 2.4.

¹In McCall’s model, completeness is a sub-characteristic of the ‘correctness’ quality factor; however in Boehm’s model it is a sub-characteristic of reliability. For consistency in this thesis, *completeness* is referred in the Boehm interpretation.



(c) Adversarial examples to trick face recognition from the source to target images [289].

Figure 2.4: Sample adversarial examples in state-of-the-art CV studies.

829 Google Cloud Vision, for instance, fails to correctly classify adversarial examples
 830 when noise is added to the original images [127]. Rosenfeld et al. [247] illustrated
 831 that inserting synthetic foreign objects to input images (e.g., a cartoon elephant)
 832 can alter classification output. Wang et al. [289] performed similar attacks on a
 833 transfer-learning approach of facial recognition by modifying pixels of a celebrity's
 834 face to be recognised as a different celebrity, all while still retaining the same human-
 835 interpretable original celebrity. Su et al. [268] used the ImageNet database to show
 836 that 41.22% of images drop in confidence when just a *single pixel* is changed in the
 837 input image; and similarly, Eykholt et al. [89] recently showed similar results that
 838 made a CNN interpret a stop road-sign (with mimicked graffiti) as a 45mph speed
 839 limit sign.

840 Thus, the state-of-the-art computer vision techniques may not be reliable enough
 841 for safety critical applications (such as self-driving cars) as they do not handle inten-
 842 tional or unintentional adversarial attacks. Moreover, as such adversarial examples
 843 exist in the physical world [89, 162], “the real world may be adversarial enough”
 844 [226] to fool such software.

845 2.2 Probabilistic and Nondeterministic Systems

846 Probabilistic and nondeterministic systems are those by which, for the same given
 847 input, different outcomes may result. The underlying models that power an IWS
 848 are treated as though they are nondeterministic; Chapter 2 introduces IWSs as
 849 essentially black-box behaviour that can change over time. As such, we adopt the
 850 nondeterministic behaviour that they present.

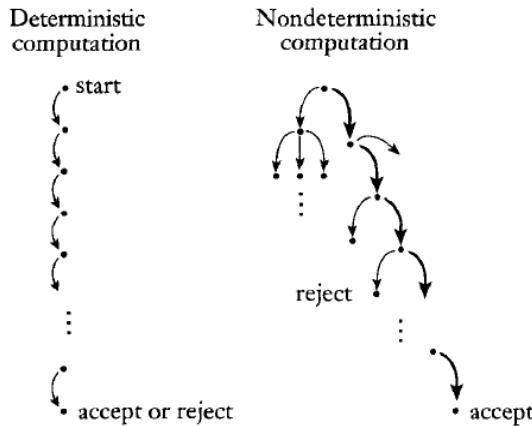


Figure 2.5: A deterministic system (left) always returns the same result in the same amount of steps. A nondeterministic system does not guarantee the same outcome, even with the same input data. Source: [93].

851 2.2.1 Interpreting the Uninterpretable

852 As the rise of applied AI increases, the need for engineering interpretability around
 853 models becomes paramount, chiefly from an external quality perspective that the
 854 *reliability* of the system can be inspected by end-users. Model interpretability has
 855 been stressed since early machine learning research in the late 1980s and 1990s (such
 856 as Quinlan [231] and Michie [195]), and although there has since been a significant
 857 body of work in the area [13, 26, 39, 50, 77, 91, 100, 108, 143, 170, 173, 184, 221,
 858 236, 248, 263, 285, 287], it is evident that ‘accuracy’ or model ‘confidence’ is still
 859 used as a primary criterion for AI evaluation [130, 138, 265]. Much research into
 860 neural network (NN) or support vector machine (SVM) development stresses that
 861 ‘good’ models are those with high accuracy. However, is accuracy enough to justify
 862 a model’s quality?

863 To answer this, we revisit what it means for a model to be accurate. Accuracy
 864 is an indicator for estimating how well a model’s algorithm will work with future
 865 or unforeseen data. It is quantified in the AI testing stage, whereby the algorithm
 866 is tested against cases known by humans to have ground truth but such cases are
 867 unknown by the algorithm. In production, however, all cases are unknown by both
 868 the algorithm *and* the humans behind it, and therefore a single value of quality is
 869 “not reliable if the future dataset has a probability distribution significantly different
 870 from past data” [96], a problem commonly referred to as the *datashift* problem [252].
 871 Analogously, Freitas [96] provides the following description of the problem:

872 *The military trained [a NN] to classify images of tanks into enemy
 873 and friendly tanks. However, when the [NN] was deployed in the field
 874 (corresponding to “future data”), it had a poor accuracy rate. Later,
 875 users noted that all photos of friendly (enemy) tanks were taken on a
 876 sunny (overcast) day. I.e., the [NN] learned to discriminate between
 877 the colors of the sky in sunny vs. overcast days! If the [NN] had
 878 output a comprehensible model (explaining that it was discriminating
 879 between colors at the top of the images), such a trivial mistake would
 880 immediately be noted.* [96]

881 So, why must we interpret models? While the formal definition of what it means
 882 to be *interpretable* is still somewhat disparate (though some suggestions have been
 883 proposed [173]), what is known is (i) there exists a critical trade-off between accuracy
 884 and interpretability [81, 95, 114, 142, 149, 306], and (ii) a single quantifiable value
 885 cannot satisfy the subjective needs of end-users [96]. As ever-growing domains
 886 ML become widespread², these applications engage end-users for real-world goals,
 887 unlike the aims in early ML research where the aim was to get AI working in the
 888 first place. In safety-critical systems where AI provide informativeness to humans
 889 to make the final call (see [53, 131, 151]), there is often a mismatch between the
 890 formal objectives of the model (e.g., to minimise error) and complex real-world
 891 goals, where other considerations (such as the human factors and cognitive science

²In areas such as medicine [25, 50, 88, 139, 143, 166, 222, 238, 285, 303, 308], bioinformatics [80, 97, 140, 148, 272], finance [13, 78, 131] and customer analytics [170, 287].

behind explanations³) are not realised: model optimisation is only worthwhile if they “actually solve the original [human-centred] task of providing explanation” [204] to end-users. **Therefore, when human-decision makers must be interpretable themselves [239], any AI they depend on must also be interpretable.**

Recently, discussion behind such a notion to provide legal implications of interpretability is topical. Doshi-Velez et al. [84] discuss when explanations are not provided from a legal stance—for instance, those affected by algorithmic-based decisions have a ‘right to explanation’ [181, 288] under the European Union’s GDPR⁴. But, explanations are not the only way to ensure AI accountability: theoretical guarantees (mathematical proofs) or statistical evidence can also serve as guarantees [84], however, in terms of explanations, what form they take and how they are proven correct are still open questions [173].

2.2.2 Explanation and Communication

From a software engineering perspective, explanations and interpretability are, by definition, inherently communication issues: what lacks here is a consistent interface between the AI system and the person using it. The ability to encode ‘common sense reasoning’ [187] into programs today has been achieved, but *decoding* that information is what still remains problematic. At a high level, Shannon and Weaver’s theory of communication [257] applies, just as others have done with similar issues in the SE realm [198, 298] (albeit to the domain of visual notations). Humans map the world in higher-level concepts easily when compared to AI systems: while we think of a tree first (not the photons of light or atoms that make up the tree), an algorithm simply sees pixels, and not the concrete object [84] and the AI interprets the tree inversely to humans. Therefore, the interpretation or explanation is done inversely: humans do not explain the individual neurons fired to explain their predictions, and therefore the algorithmic transparent explanations of AI algorithms (“*which neurons were fired to make this AI think this tree is a tree?*”) do not work here.

Therefore, to the user (as mapped using Shannon and Weaver’s theory), an AI pipeline (the communication *channel*) begins with a real-world concept, y , that acts as an *information source*. This information source is fed in as a *message*, x , (as pixels) to an AI system (the *transmitter*). The transmitter encodes the pixels to a prediction, \hat{y} , the *signal* of the message. This signal is decoded by the *receiver*, an explanation system, $e_x(x, \hat{y})$, that tailors the prediction with the given input data to the intended end user (the *destination*) as an explanation, \tilde{y} , another type of *message*. Therefore, the user only sees the channel as an input/output pipeline of real-world objects, y , and explanations, \tilde{y} , tailored to *them*, without needing to see the inner-mechanics of a prediction \hat{y} . We present this diagrammatically in Figure 2.6.

2.2.3 Mechanics of Model Interpretation

How do we interpret models? Methods for developing interpretation models include: decision trees [45, 64, 119, 177, 232], decision tables [14, 170] and decision sets

³Interpretations and explanations are often used interchangeably.

⁴<https://www.eugdpr.org> last accessed 13 August 2018.

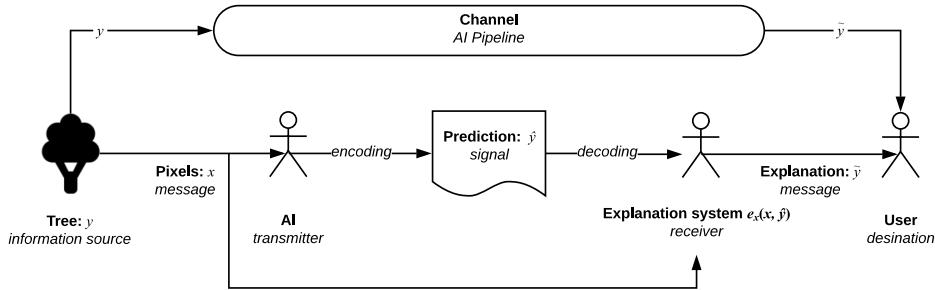


Figure 2.6: Theory of AI communication from information source, y , to intended user as explanations \tilde{y} .

[164, 204]; input gradients, gradient vectors or sensitivity analysis [13, 167, 236, 248, 255]; exemplars [98, 152]; generalised additive models [53]; classification (*if-then*) rules [41, 61, 216, 280, 300] and falling rule lists [263]; nearest neighbours [184, 233, 256, 296, 307] and Naïve Bayes analysis [25, 57, 90, 99, 122, 159, 166, 308].

Cross-domain studies have assessed the interpretability of these techniques against end-users, measuring response time, accuracy in model response and user confidence [6, 97, 120, 131, 184, 251, 269, 287], although it is generally agreed that decision rules and decision tables provide the most interpretation in non-linear models such as SVMs or NNs [97, 184, 287]. For an extensive survey of the benefits and fallbacks of these techniques, we refer to Freitas [96], Doshi-Velez et al. [84] and Doshi-Velez and Kim [83].

2.3 Application Programming Interfaces

Application programming interfaces (APIs) are the interface between a developer needs and the software components at their disposal [10] by abstracting the underlying component behind a subroutine, protocol or specific tool. Therefore, it is natural to assess internal quality (and external quality if the software is in itself a service to be used by other developers—in this case an IWS) is therefore directly related to the quality the API offers [158].

Good APIs are known to be intuitive and require less documentation browsing [228], thereby increasing developer productivity. Conversely, poor APIs are those that are hard to interpret, thereby reducing developer productivity and product quality. The consequences of this have shown a higher demand of technical support (as measured in [123]) that, ultimately, causes the maintenance to be far more expensive, a phenomenon widely known in software engineering economics (see Section 2.1.1).

While there are different types of APIs, such as software library/framework APIs for building desktop software, operating system APIs for interacting with the operating system, remote APIs for communication of varying technologies through common protocols, we focus on web APIs for communication of resources over

961 the web (being the common architecture of cloud-based services). Further information
962 on the development, usage and documentation of web APIs is provided in
963 Appendix A.1.

964 2.3.1 API Usability

965 If a developer doesn't understand the overarching concepts of the context behind the
966 API they wish to use, then they cannot formulate what gaps in their knowledge is
967 missing. For example, a developer that knows nothing about ML techniques in CV
968 cannot effectively formulate queries to help bridge those gaps in their understanding
969 to figure out more about the CVS they wish to use.

970 Balancing the understanding of the information need (both conscious and unconscious), how to phrase that need and how to query it in an information retrieval
971 system is concept long studied in the information sciences [278]. In API design,
972 the most common form to convey knowledge to developers is through annotated
973 code examples and overviews to a platform's architectural and design decisions
974 [42, 82, 201, 243] though these studies have not effectively communicated *why* these
975 artefacts are important. What makes the developer *conceptually understand* these
976 artefacts?

977 Robillard and Deline [243] conducted a multi-phase, mixed-method approach to
978 create knowledge grounded in the professional experience of 440 software engineers
979 at Microsoft of varying experience to determine what makes APIs hard to learn,
980 the results of which previously published in an earlier report [242]. Their results
981 demonstrate that 'documentation-related obstacles' are the biggest hurdle in learning
982 new APIs. One of these implications are the *intent documentation* of an API (i.e.,
983 *what is the intent for using a particular API?*) and such documentation is required
984 only where correct API usage is not self-evident, where advanced uses of the API are
985 documented (but not the intent), and where performance aspects of the API impact
986 the application developed using it. They conclude that professional developers do
987 not struggle with learning the *mechanics* of the API, but in the *understanding* of how
988 the API fits in upwards to its problem domain and downward to its implementation:

990 *In the upwards direction, the study found that developers need help
991 mapping desired scenarios in the problem domain to the content of the
992 API, and in understanding what scenarios or usage patterns the API
993 provider intends and does not intend to support. In the downwards
994 direction, developers want to understand how the API's implementation
995 consumes resources, reports errors and has side effects. [243]*

996 These results particularly corroborate to that of previous studies where developers
997 quote that they feel that existing learning content currently focuses on "how
998 to do things, not necessarily why" [211]. This thereby reiterates the conceptual
999 understanding of an API as paramount.

1000 A later study by Ko and Riche [157] assessed the importance of a programmer's
1001 conceptual understanding of the background behind the task before implementing the
1002 task itself, a notion that we find most relevant for users of IWS APIs. While the study

1003 did not focus on developing web APIs (rather implementing a Bluetooth application
1004 using platform-agnostic terminology), the study demonstrated how developers show
1005 little confidence in their own metacognitive judgements to understand and assess the
1006 feasibility of the intent of the API and understand the vocabulary and concepts within
1007 the domain (i.e., wireless connectivity). This indecision over what search results
1008 were relevant in their searches ultimately hindered their progress implementing the
1009 functionality, again decreasing productivity. Ko and Riche suggest to improve API
1010 usability by introducing the background of the API and its relevant concepts using
1011 glossaries linked to tutorials to each of the major concepts, and then relate it back to
1012 how to implement the particular functionality.

1013 Thus, an analysis of the conceptual understanding of IWS APIs by a range of
1014 developers (from beginner to professional) is critical to best understand any differ-
1015 ences between existing studies and those that are nondeterministic. Our proposal is
1016 to perform similar survey research (see Chapter 3) in the search for further insight
1017 into the developer's approach toward existing IWS APIs.

1018

CHAPTER 3

1019

1020

Research Methodology

1021

1022 Investigating software engineering practices is often a complex task as it is imper-
1023 ative to understand the social and cognitive processes around software engineers
1024 and not just the tools and processes used [87]. This chapter explores our research
1025 methodology by exploring five key elements of empirical software engineering re-
1026 search: firstly, (i) we provide an extended focus to the study by reviewing our research
1027 questions (see Section 1.4) anchored under the context of an existing research ques-
1028 tion classification taxonomy, (ii) characterise our research goals through an explicit
1029 philosophical stance, (iii) explain how the stance selected impacts our selection of
1030 research methods and data collection techniques (by dissecting our choice of meth-
1031 ods used to reach these research goals), (iv) discuss a set of criteria for assessing the
1032 validity of our study design and the findings of our research, and lastly (v) discuss
1033 the practical considerations of our chosen methods.

1034 The foundations for developing this research methodology has been expanded
1035 from that proposed by Easterbrook et al. [87], Wohlin and Aurum [301], Wohlin
1036 et al. [302] and Shaw [258].

1037

3.1 Research Questions Revisited

1038 To discuss our research strategy, we revisit our four primary and seven secondary
1039 research questions (RQs) through the classification technique discussed by East-
1040 erbrook et al. [87], a technique originally proposed in the field of psychology by
1041 Meltzoff and Cooper [191] but adapted to software engineering. A summary of the
1042 classifications made to our research questions are presented in Table 3.1.

1043 Our research study involves a mix of nine *empirical*¹ RQs, that focus on observ-
1044 ing and analysing existing phenomena, and two *non-empirical* RQs, that focuses
1045 on designing better approaches to solve software engineering tasks [194]. The use

¹Or ‘knowledge’ questions, that extend our *knowledge* on certain phenomena.

1046 of empirical *and* non-empirical RQs are best combined in long-term software engineering research studies where the phenomena are under-explored, as is the case 1047 with CVSs. Further, these approaches help propose solutions to issues found in the 1048 phenomena studied [299]. We discuss both our empirical and non-empirical RQs in 1049 Sections 3.1.1 and 3.1.2 below. 1050

Table 3.1: A summary of our research questions classified using the strategies presented by Easterbrook et al. [87] and Meltzoff and Cooper [191].

#	RQ	Primary/ Secondary	RQ Classification
RQ1	What is the nature of cloud-based CVSs?	Primary	EMPIRICAL ↔ Exploratory ↔ Description/Classification
	RQ1.1 What is their runtime behaviour?	Secondary	EMPIRICAL ↔ Exploratory ↔ Description/Classification
	RQ1.2 What is their evolution profile?	Secondary	EMPIRICAL ↔ Exploratory ↔ Description/Classification
RQ2	Are CVS APIs sufficiently documented?	Primary	EMPIRICAL ↔ Exploratory ↔ Existence
	RQ2.1 What are the dimensions of a ‘complete’ API document, according to both literature and practitioners?	Secondary	EMPIRICAL ↔ Exploratory ↔ Composition
	RQ2.2 What additional information or attributes do application developers need in CVS API documentation to make it more complete?	Secondary	NON-EMPIRICAL ↔ Design
RQ3	Are CVSs more misunderstood than conventional software engineering domains?	Primary	EMPIRICAL ↔ Exploratory ↔ Descriptive-Comparative
	RQ3.1 What types of issues do application developers face most when using CVSs, as expressed as questions on Stack Overflow?	Secondary	EMPIRICAL ↔ Base-Rate ↔ Frequency/Distribution
	RQ3.2 Which of these issues are application developers most frustrated with?	Secondary	EMPIRICAL ↔ Exploratory ↔ Description/Classification
	RQ3.3 Is the distribution CVS pain-points different to established software engineering domains, such as mobile or web development?	Secondary	EMPIRICAL ↔ Base-Rate ↔ Frequency/Distribution
RQ4	What strategies can developers employ to integrate their applications with CVSs while preserving robustness and reliability?	Primary	NON-EMPIRICAL ↔ Design

1051 3.1.1 Empirical Research Questions

1052 In total, nine empirically-based RQs are posed in this study to help us understand the 1053 way developers currently interact and work with web services that provide computer 1054 vision. The majority of these questions are *exploratory* questions that contribute to 1055 a landscape analysis of these services (RQ1, RQ1.1 and RQ1.2), how well they are 1056 documented (RQ2), and the issues developers currently face when using them (RQ3).

1057 Our other exploratory questions complement the answers to these questions. For
1058 instance, to understand if CVSs are sufficiently documented (an *existence* exploratory
1059 question posed in RQ2), we need to understand the components of a ‘sufficient’ or
1060 ‘complete’ API document via RQ2.1 as proposed in both the literature and by
1061 software developers. While RQ2.1 does not directly relate to CVSs, answering it
1062 gives us an understanding the components of complete API documentation, and
1063 therefore, we can assess what aspects they are missing and where improvements
1064 can be made (RQ2.2). These questions are *descriptive and classification* questions
1065 that help describe and classify what practices are in use for existing CVS API
1066 documentation and the nature behind these services. Answering these exploratory
1067 questions assists in refining preciser terms of the phenomena, ways in which we find
1068 evidence for them and ensuring the data found is valid.

1069 By answering these questions, we have a clearer understanding of the phenomena;
1070 we then follow up by posing two additional *base-rate questions* that helps
1071 provide a basis to confirm that the phenomena occurring is normal (or unusual)
1072 behaviour by investigating the patterns of phenomena’s occurrence against other
1073 phenomena. RQ3.1 is a *frequency and distribution* question to help us understand
1074 what types of issues developers often encounter most, given a lack of formal extended
1075 training in artificial intelligence. This achieves us an insight into the developer’s
1076 mindset and regular thought patterns toward these APIs. We can then contrast
1077 this distribution using our second base-rate question (RQ3.3), that assesses the
1078 distributional differences between these intelligent components and non-intelligent
1079 (conventional) software components. Combined, these two questions can help us
1080 answer how the issues raised against CVSs are different to normal Stack Overflow
1081 issues—our *descriptive-comparative* question posed in RQ3—and, similarly, we can
1082 classify and rank which issues developers find most frustrating (RQ3.2).

1083 3.1.2 Non-Empirical Research Questions

1084 RQ2.2 and RQ4 are both non-empirically-based *design questions*; they are con-
1085 cerned with ways in which we can improve a CVS by investigating what additional
1086 attributes are needed in both the documentation of CVSs and in the integration
1087 architectures developers can employ to improve reliability and robustness in their
1088 applications. They are not classified as empirical questions as we investigate what
1089 *will be* and not *what is*. By understanding the process by which developers desire
1090 additional attributes of documentation and integration strategies, we can help shape
1091 improvements to the existing designs of using CVSs.

1092 3.2 Philosophical Stances

1093 Philosophical stances guide the researcher’s action by fortifying what constitutes
1094 ‘valid truth’ against a fundamental set of core beliefs [241]. In software engineer-
1095 ing, four dominant philosophical stances are commonly characterised [65, 224]:
1096 positivism (or post-positivism), constructivism (or interpretivism), pragmatism, and
1097 critical theory (or advocacy/participatory). To construct such a ‘validity of truth’,

1098 we will review these four philosophical stances in this section, and state the stance
1099 that we explicitly adopt and our reasoning for this.

1100 **Positivism** Positivists claim truth to be all observable facts, reduced piece-by-
1101 piece to smaller components which is incrementally verifiable to form truth. We
1102 do not base our work on the positivistic stance as the theories governing verifiable
1103 hypothesis must be precise from the start of the research. Moreover, due to its
1104 reductionist approach, it is difficult to isolate these hypotheses and study them in
1105 isolation from context. As our hypotheses are not context-agnostic, we steer clear
1106 from this stance.

1107 **Constructivism** Constructivists see knowledge embedded within the human con-
1108 text; truth is the *interpretive* observation by understanding the differences in human
1109 thought between meaning and action [156]. That is, the interpretation of the theory
1110 is just as important to the empirical observation itself. We partially adopt a con-
1111 structivist stance as we attempt to model the developer’s mindset, being an approach
1112 that is rich in qualitative data on human activity.

1113 **Pragmatism** Pragmatism is a less dogmatic approach that encourages the incom-
1114 plete and approximate nature of knowledge and is dependent on the methods in which
1115 the knowledge was extracted. The utility of consensually agreed knowledge is the
1116 key outcome, and is therefore relative to those who seek utility in the knowledge—
1117 what is the useful for one person is not so for the other. While we value the utility
1118 of knowledge, it is difficult to obtain consensus especially on an ill-researched topic
1119 such as ours, and therefore we do not adopt this stance.

1120 **Critical Theory** This study chiefly adopts the philosophy of critical theory [8]. A
1121 key outcome of the study is to shift the developer’s restrictive deterministic mindset
1122 and shed light on developing a new framework actively with the developer community
1123 that seeks to improve the process of using such APIs. In software engineering,
1124 critical theory is used to “actively [seek] to challenge existing perceptions about
1125 software practice” [87], and this study utilises such an approach to shift the mindset
1126 of CVS consumers and providers alike on how the documentation and metadata
1127 should not be written with the ‘traditional’ deterministic mindset at heart. Thus, our
1128 key philosophical approach is critical theory to seek out *what-can-be* using partial
1129 constructivism to model the current *what-is*.

1130 3.3 Research Methods

1131 Research methods are “a set of organising principles around which empirical data
1132 is collection and analysed” [87]. Creswell [65] suggests that strong research design
1133 is reflected when the weaknesses of multiple methods complement each other. Us-
1134 ing a mixed-methods approach is therefore commonplace in software engineering

1135 research, typically due to the human-oriented nature investigating how software en-
1136 gineers work both individually (where methods from psychology may be employed)
1137 and together (where methods from sociology may be employed).

1138 Therefore, studies in software engineering are typically performed as field studies
1139 where researchers and developers (or the artefacts they produce) are analysed either
1140 directly or indirectly [262]. The mixed-methods approach combines five classes
1141 of field study methods (or empirical strategies/studies) most relevant in empirical
1142 software engineering research [87, 147, 302]: controlled experiments, case studies,
1143 survey research, ethnographies, and action research. We chiefly adopt a mixed-
1144 methods approach to our work using the *concurrent triangulation* mixed-methods
1145 strategy [185] as it best compensates for weaknesses that exist in all research methods,
1146 and employs the best strengths of others [65].

1147 3.3.1 Review of Relevant Research Methods

1148 Below we review some of the research methods most relevant to our research ques-
1149 tions as refined in Section 3.1 as presented by Easterbrook et al. [87].

1150 **Controlled Experiments** A controlled experiment is an investigation of a clear,
1151 testable hypothesis that guides the researcher to decide and precisely measure how
1152 at least one independent variable can be manipulated and effect at least one other
1153 dependent variable. They determine if the two variables are related and if a cause-
1154 effect relationship exists between them. The combination of independent variable
1155 values is a *treatment*. It is common to recruit human subjects to perform a task and
1156 measure the effect of a randomly assigned treatment on the subjects, though it is
1157 not always possible to achieve full randomisation in real-life software engineering
1158 contexts, in which case a *quasi-experiment* may be employed where subjects are not
1159 randomly assigned to treatments.

1160 While we have well-defined RQs, refining them into precise, *measurable* vari-
1161 ables is challenging due to the qualitative nature they present. A well-defined
1162 population is also critical and must be easily accessible; the varied range of beginner
1163 to expert software engineers with varied understanding of artificial intelligence
1164 concepts is required to perform controlled experiments, and thus recruitment may
1165 prove challenging. Lastly, the controlled experiment is essentially reductionist by
1166 affecting a small amount of variables of interest and controlling all others. This
1167 approach is too clinical for the practical outcomes by which our research goals aim
1168 for, and is therefore closely tied to the positivist stance.

1169 **Case Studies** Case studies investigate phenomena in their real-life context and are
1170 well-suited when the boundary between context and phenomena is unknown [305].
1171 They offer understanding of how and why certain phenomena occur, thereby inves-
1172 tigating ways cause-effect relationships can occur. They can be used to test existing
1173 theories (*confirmatory case studies*) by refuting theories in real-world contexts in-
1174 stead of under laboratory conditions or to generate new hypotheses and build theories
1175 during the initial investigation of some phenomena (*exploratory case studies*).

1176 Case studies are well-suited where the context of a situation plays a role in
1177 the phenomenon being studied. They also lend themselves to purposive sampling
1178 rather than random sampling, and thus it is possible to selectively choose cases that
1179 benefit our research goals and (using our critical theorist stance) select cases that
1180 will actively benefit our participant software engineering audience most to draw
1181 attention to situations regarded as problematic in CVS.

1182 **Survey Research** Survey research identifies characteristics of a broad population
1183 of individuals through direct data collection techniques such as interviews and ques-
1184 tionnaires or independent techniques such as data logging. Defining that well-defined
1185 population is critical, and selecting a representative sample from it to generalise the
1186 data gathered usually assists in answering base-rate questions.

1187 By identifying representative sample of the population, from beginner to ex-
1188 perienced developers with varying understanding of CVS APIs, we can use survey
1189 research to assist in answering our exploratory and base-rate RQs (see Section 3.1.1)
1190 in determining the qualitative aspects of how individual developers perceive and
1191 work with the existing APIs, either by directly asking them, or by mining third-party
1192 discussion websites such as Stack Overflow (SO). Similarly, we can use this strategy
1193 to assess the developer’s understanding on what makes API documentation sufficient
1194 by assessing whether specific factors suggested from literature are useful according
1195 to developers. However, with direct survey research techniques, low response rates
1196 may prove challenging, especially if no inducements can be offered for participation.

1197 **Ethnographies** Ethnographies investigates the understanding of social interac-
1198 tion within community through field observation [244]. Resulting ethnographies
1199 help understand how software engineering technical communities build practices,
1200 communication strategies and perform technical work collaboratively.

1201 Ethnographies require the researcher to be highly trained in observational and
1202 qualitative data analysis, especially if the form of ethnography is participant observa-
1203 tion, whereby the researcher is embedded of the technical community for observation.
1204 This may require the longevity of the study to be far greater than a couple of weeks,
1205 and the researcher must remain part of the project for its duration to develop enough
1206 local theories about how the community functions. While it assists in revealing
1207 subtle but important aspects of work practices within software teams, this study
1208 does not focus on the study of teams, and is therefore not a research method relevant
1209 to this project.

1210 **Action Research** Action researchers simultaneously solve real-world problems
1211 while studying the experience of solving the problem [75] by actively seeking to
1212 intervene in the situation for the purpose of improving it. A precondition is to engage
1213 with a *problem owner* who is willing to collaborate in identifying and solving the
1214 problem faced. The problem must be authentic (a problem worth solving) and must
1215 have new knowledge outcomes for those involved. It is also characterised as an
1216 iterative approach to problem solving, where the knowledge gained from solving the
1217 problem has a desirable solution that empowers the problem owner and researcher.

1218 This research is most associated to our adopted philosophical stance of critical
1219 theory. As this project is being conducted under the Applied Artificial Intelligence
1220 Institute (A^2I^2) collaboratively with engaged industry clients, we have identified a
1221 need for solving an authentic problem that industry faces. The desired outcome
1222 of this project is to facilitate wider change in the usage and development of CVSSs;
1223 thus, engaging action research as a potential method throughout the mixed-methods
1224 approach used in this research.

1225 **3.3.2 Review of Data Collection Techniques for Field Studies**

1226 Singer et al. developed a taxonomy [168, 262] showcasing data collection techniques
1227 in field studies that are used in conjunction with a variety of methods based on the
1228 level of interaction between researcher and software engineer, if any. This taxonomy
1229 is reproduced in Figure 3.1.

1230 **3.4 Research Design**

1231 This section discusses an overview of the design of methods used within the experi-
1232 ments conducted under this thesis. For each experiment, we describe an overview of
1233 the experiment grounded known methods and techniques (Sections 3.3.1 and 3.3.2)
1234 and our approach to analysing the data, as well as relating the selecting method back
1235 to a specific RQ. Details of each experiment presented in this thesis, the coherency
1236 between them, and where they can be found are given in Sections 1.6 and 1.7.

1237 **3.4.1 Landscape Analysis of Computer Vision Services**

1238 To understand the behavioural and evolutionary profiles of CVSSs (i.e., RQ1), we
1239 employ a longitudinal study based around a dynamic system analysis [262]. Specific-
1240 ally, we employ structured observations of three services using the same dataset to
1241 understand how the responses from these services change with time. Lastly, we em-
1242 ploy documentation analysis to assess the overall ‘picture’ of how these services are
1243 documented. Further details on this experiment is given in **Chapter 4, Section 4.4**.

1244 **3.4.2 Utility of API Documentation in Computer Vision Services**

1245 To assess whether these services are sufficiently documented (i.e., RQ2), we conduct
1246 a systematic mapping study [153, 225] of the various academic sources detailing API
1247 documentation knowledge. We then consolidate this information into a structured
1248 taxonomy following a systematic taxonomy development method specific to software
1249 engineering studies [284].

1250 We then follow the triangulation approach proposed by Mayring [185] to validate
1251 the taxonomy by use of a personal opinion survey. Kitchenham and Pfleeger [154]
1252 provide an introduction on methods used to conduct personal opinion surveys which
1253 we adopt as an initial reference in (i) shaping our survey objectives around our
1254 research goals, (ii) designing a cross-sectional survey, (iii) developing and evaluating
1255 our survey instrument, (iv) evaluating our instruments, (v) obtaining the data and

Figure 3.1: Questions asked by software engineering researchers (column 2) that can be answered by field study techniques. (From [262].)

Technique	Used by researchers when their goal is to understand:	Volume of data	Also used by software engineers for
Direct techniques			
Brainstorming and focus groups	Ideas and general background about the process and product, general opinions (also useful to enhance participant rapport)	Small	Requirements gathering, project planning
Interviews and questionnaires	General information (including opinions) about process, product, personal knowledge etc.	Small to large	Requirements and evaluation
Conceptual modeling	Mental models of product or process	Small	Requirements
Work diaries	Time spent or frequency of certain tasks (rough approximation, over days or weeks)	Medium	Time sheets
Think-aloud sessions	Mental models, goals, rationale and patterns of activities	Medium to large	UI evaluation
Shadowing and observation	Time spent or frequency of tasks (intermittent over relatively short periods), patterns of activities, some goals and rationale	Small	Advanced approaches to use case or task analysis
Participant observation (joining the team)	Deep understanding, goals and rationale for actions, time spent or frequency over a long period	Medium to large	
Indirect techniques			
Instrumenting systems	Software usage over a long period, for many participants	Large	Software usage analysis
Fly on the wall	Time spent intermittently in one location, patterns of activities (particularly collaboration)	Medium	
Independent techniques			
Analysis of work databases	Long-term patterns relating to software evolution, faults etc.	Large	Metrics gathering
Analysis of tool use logs	Details of tool usage	Large	
Documentation analysis	Design and documentation practices, general understanding	Medium	Reverse engineering
Static and dynamic analysis	Design and programming practices, general understanding	Large	Program comprehension, metrics, testing, etc.

1256 (vi) analysing the data. We adapt Brooke's systematic usability scale [47] technique
1257 by basing our research questions against a known surveying instrument.

1258 As is good practice in developing questionnaire instruments to evaluate their
1259 reliability and validity [174], we evaluate our instrument design by asking colleagues
1260 to critique it via pilot studies within A²I². This assists in identifying any problems
1261 with the questionnaire itself and with any issues that may occur with the response
1262 rate and follow-up procedures.

1263 Findings from the pilot study helps inform us for a widely distributed question-
1264 naire using snow-balling sampling. Ethics approval from the Faculty of Science,
1265 Engineering and Built Environment Human Ethics Advisory Group (SEBE HEAG)
1266 has been approved to externally conducting this survey research (see Appendix C).
1267 Further details on API these methods are detailed within ??, ??.

1268 3.4.3 Developer Issues concerning Computer Vision Services

1269 Developers typically congregate in search of discourses on issues they face in online
1270 forums, such as Stack Overflow (SO) and Quora, as well as writing their experiences
1271 in personal blogs such as Medium. The simplest of these platforms is SO (a sub-
1272 community of the Stack Exchange family of targeted communities) that specifically
1273 targets developer issues on using a simple Q&A interface, where developers can
1274 discuss technical aspects and general software development topics. Moreover, SO
1275 is often acknowledged as *the ‘go-to’ place* for developers to find high-quality code
1276 snippets that assist in their problems [270].

1277 Thus, to begin understanding the issues developers face when using CVSs and
1278 whether there is a substantial difference to conventional domains (i.e., RQ3), we
1279 propose using repository mining on SO to help answer our research questions.
1280 Specifically, we select SO due to its targeted community of developers² and the
1281 availability of its publicly available dataset released as ‘data dumps’ on the Stack
1282 Exchange Data Explorer³ and Google BigQuery⁴. Studies conducted have also used
1283 SO to mine developer discourse [7, 16, 21, 59, 172, 209, 217, 234, 245, 264, 276,
1284 291]. Further details on how we approached the design for this study can be found
1285 in **Chapters 6 and 7, Section 6.4 and ??**.

1286 3.4.4 Designing Improved Integration Strategies

1287 Our improved integration strategies (i.e., RQ4) evolved organically over the dura-
1288 tion of this research project through the use of industry case studies and action
1289 research. We develop several iterative prototypes and use a mix of statistical and
1290 technical-expert assessment to analyse whether our improved integration striges

²We also acknowledge that there are other targeted software engineering Stack Exchange communities such as Stack Exchange Software Engineering (<https://softwareengineering.stackexchange.com>), though (as of January 2019) this much smaller community consists of only 52,000 questions versus SO’s 17 million.

³<https://data.stackexchange.com/stackoverflow> last accessed 17 January 2017.

⁴<https://console.cloud.google.com/marketplace/details/stack-exchange/stack-overflow> last accessed 17 January 2017.

¹²⁹¹ can prove useful to developers. Further details about these approaches are detailed
¹²⁹² in **Chapters 8 to 10, Section 8.5.1 and ????**. ⟨ *TODO: Add more detail later* ⟩

1293

Part II

1294

Publications

CHAPTER 4

1295

1296

1297

1298

Identifying Evolution in Computer Vision Services[†]

1299 **Abstract** Recent advances in artificial intelligence (AI) and machine learning (ML), such
1300 as computer vision (CV), are now available as intelligent web services (IWSs) and their
1301 accessibility and simplicity is compelling. Multiple vendors now offer this technology as
1302 cloud services and developers want to leverage these advances to provide value to end-users.
1303 However, there is no firm investigation into the maintenance and evolution risks arising from
1304 use of these IWSs; in particular, their behavioural consistency and transparency of their
1305 functionality. We evaluated the responses of three different IWSs (specifically CV) over 11
1306 months using 3 different data sets, verifying responses against the respective documentation
1307 and assessing evolution risk. We found that there are: (1) inconsistencies in how these
1308 services behave; (2) evolution risk in the responses; and (3) a lack of clear communication
1309 that documents these risks and inconsistencies. We propose a set of recommendations to
1310 both developers and IWS providers to inform risk and assist maintainability.

4.1 Introduction

1311 The availability of intelligent web services (IWSs) has made artificial intelligence
1312 (AI) tooling accessible to software developers and promises a lower entry barrier for
1313 their utilisation. Consider state-of-the-art computer vision (CV) analysers, which
1314 require either manually training a deep-learning classifier, or selecting a pre-trained
1315 model and deploying these into an appropriate infrastructure. Either are laborious
1316 in time, and require non-trivial expertise along with a large data set when training
1317 or customisation is needed. In contrast, IWSs providing CV (i.e., computer vision
1318 services or CVSs such as [313, 315, 316, 317, 324, 327, 329, 330, 331, 335, 338,

[†]This chapter is originally based on A. Cummaudo, R. Vasa, J. Grundy, M. Abdelrazek, and A. Cain, “Losing Confidence in Quality: Unspoken Evolution of Computer Vision Services,” in *Proceedings of the 35th IEEE International Conference on Software Maintenance and Evolution*. Cleveland, OH, USA: IEEE, December 2019. DOI 10.1109/ICSME.2019.00051. ISBN 978-1-72-813094-1 pp. 333–342. Terminology has been updated to fit this thesis.

1320 339, 372, 373]) abstract these complexities behind a web application programming
1321 interface (API) call. This removes the need to understand the complexities required
1322 of machine learning (ML), and requires little more than the knowledge on how to
1323 use RESTful endpoints. The ubiquity of these services is exemplified through their
1324 rapid uptake in applications such as aiding the vision-impaired [74, 235].

1325 While IWSs have seen quick adoption in industry, there has been little work
1326 that has considered the software quality perspective of the risks and impacts posed
1327 by using such services. In relation to this, there are three main challenges: (1)
1328 incorporating stochastic algorithms into software that has traditionally been deter-
1329 ministic; (2) the general lack of transparency associated with the ML models; and
1330 (3) communicating to application developers.

1331 ML typically involves use of statistical techniques that yield components with
1332 a non-deterministic external behaviour; that is, for the same given input, different
1333 outcomes may result. However, developers, in general, are used to libraries and small
1334 components behaving predictably, while systems that rely on ML techniques work
1335 on confidence intervals¹ and probabilities. For example, the developer’s mindset
1336 suggests that an image of a border collie—if sent to three intelligent computer vision
1337 services (CVSs)—would return the label ‘dog’ consistently with time regardless
1338 of which service is used. However, one service may yield the specific dog breed,
1339 ‘border collie’, another service may yield a permutation of that breed, ‘collie’, and
1340 another may yield broader results, such as ‘animal’; each with results of varying
1341 confidence values.² Furthermore, the third service may evolve with time, and
1342 thus learn that the ‘animal’ is actually a ‘dog’ or even a ‘collie’. The outcomes
1343 are thus behaviourally inconsistent between services providing conceptually similar
1344 functionality. As a thought exercise, consider if the sub-string function were created
1345 using ML techniques—it would perform its operation with a confidence where the
1346 expected outcome and the AI inferred output match as a *probability*, rather than a
1347 deterministic (constant) outcome. How would this affect the developers’ approach
1348 to using such a function? Would they actively take into consideration the non-
1349 deterministic nature of the result?

1350 Myriad software quality models and software engineering (SE) practices advo-
1351 cate maintainability and reliability as primary characteristics; stability, testability,
1352 fault tolerance, changeability and maturity are all concerns for quality in software
1353 components [126, 230, 266] and one must factor these in with consideration to
1354 software evolution challenges [111, 112, 192, 193, 279]. However, the effect this
1355 non-deterministic behaviour has on quality when masked behind an IWS is still
1356 under-explored to date in SE literature, to our knowledge. Where software depends
1357 on IWSs to achieve functionality, these quality characteristics may not be achieved,
1358 and developers need to be wary of the unintended side effects and inconsistency that
1359 exists when using non-deterministic components. A CVS may encapsulate deep-
1360 learning strategies or stochastic methods to perform image analysis, but developers

¹Varied terminology used here. Probability, confidence, accuracy and score may all be used interchangeably.

²Indeed, we have observed this phenomenon using a picture of a border collie sent to various CVSs.

1361 are more likely to approach IWSs with a mindset that anticipates consistency. Al-
1362 though the documentation does hint at this non-deterministic behaviour (i.e., the
1363 descriptions of ‘confidence’ in various CVSs suggest they are not always confi-
1364 dent, and thus not deterministic [311, 325, 336]), the integration mechanisms offered
1365 by popular vendors do not seem to fully expose the nuances, and developers are not
1366 yet familiar with the trade-offs.

1367 Do popular CVSs, as they currently stand, offer consistent behaviour, and if not,
1368 how is this conveyed to developers (if it is at all)? If CVSs are to be used in production
1369 services, do they ensure quality under rigorous service quality assurance (SQA)
1370 frameworks [126]? What evolution risk [111, 112, 192, 193] do they pose if these
1371 services change? To our knowledge, few studies have been conducted to investigate
1372 these claims. This paper assesses the consistency, evolution risk and consequent
1373 maintenance issues that may arise when developers use IWSs. We introduce a
1374 motivating example in Section 4.2, discussing related work and our methodology
1375 in Sections 4.3 and 4.4. We present and interpret our findings in Section 4.5. We
1376 argue with quantified evidence that these IWSs can only be considered with a mature
1377 appreciation of risks, and we make a set of recommendations in Section 4.6.

1378 4.2 Motivating Example

1379 Consider Rosa, a software developer, who wants to develop a social media photo-
1380 sharing mobile app that analyses her and her friends photos on Android and iOS.
1381 Rosa wants the app to categorise photos into scenes (e.g., day vs. night, outdoors
1382 vs. indoors), generate brief descriptions of each photo, and catalogue photos of her
1383 friends as well as common objects (e.g., all photos with a dog, all photos on the
1384 beach).

1385 Rather than building a CV engine from scratch, Rosa thinks she can achieve this
1386 using one of the popular CVSs (e.g., [313, 315, 316, 317, 324, 327, 329, 330, 331,
1387 335, 338, 339, 372, 373]). However, Rosa comes from a typical software engineering
1388 background with limited knowledge of the underlying deep-learning techniques
1389 and implementations as currently used in CV. Not unexpectedly, she internalises a
1390 mindset of how such services work and behave based on her experience of using
1391 software libraries offered by various SDKs. This mindset assumes that different
1392 cloud vendor image processing APIs more-or-less provide similar functionality,
1393 with only minor variations. For example, cloud object storage for Amazon S3 is
1394 both conceptually and behaviourally very similar to that of Google Cloud Storage
1395 or Azure Storage. Rosa assumes the CVSs of these platforms will, therefore, likely
1396 be very similar. Similarly, consider the string libraries Rosa will use for the app.
1397 The conceptual and behavioural similarities are consistent; a string library in Java
1398 (Android) is conceptually very similar to the string library she will use in Swift
1399 (iOS), and likewise both behave similarly by providing the same results for their
1400 respective sub-string functionality. However, **unlike the cloud storage and string**
1401 **libraries, different CVSs often present conceptually similar functionality but**
1402 **are behaviourally very different.** IWS vendors also hide the depth of knowledge
1403 needed to use these effectively—for instance, the training data set and ontologies

1404 used to create these services are hidden in the documentation. Thus, Rosa isn't even
 1405 exposed to this knowledge as she reads through the documentation of the providers
 1406 and, thus, Rosa makes the following assumptions:

- 1407 • **"I think the responses will be consistent amongst these CVSs."** When Rosa
 1408 uploads a photo of a dog, she would expect them all to respond with 'dog'. If
 1409 Rosa decides to switch which service she is using, she expects the ontologies
 1410 to be compatible (all CVSs *surely* return dog for the same image) and therefore
 1411 she can expect to plug-in a different service should she feel like it making only
 1412 minor code modifications such as which endpoints she is relying on.
- 1413 • **"I think the responses will be constant with time."** When Rosa uploads the
 1414 photo of a dog for testing, she expects the response to be the same in 10 weeks
 1415 time once her app is in production. Hence, in 10 weeks, the same photo of the
 1416 dog should return the same label.

1417 4.3 Related Work

1418 If we were to view CVSs through the lenses of an SQA framework, robustness,
 1419 consistency, and maintainability often feature as quality attributes in myriad soft-
 1420 ware quality models (e.g., [136]). Software quality is determined from two key
 1421 dimensions: (1) in the evaluation of the end-product (external quality) and (2) the
 1422 assurances in the development processes (internal quality) [230]. We discuss both
 1423 perspectives of quality within the context of our work in this section.

1424 4.3.1 External Quality

1425 **Robustness for safety-critical applications** A typical focus of recent work has
 1426 been to investigate the robustness of deep-learning within CV technique imple-
 1427 mentation, thereby informing the effectiveness in the context of the end-product.
 1428 The common method for this has been via the use of adversarial examples [273],
 1429 where input images are slightly perturbed to maximise prediction error but are still
 1430 interpretable to humans.

1431 Google Cloud Vision, for instance, fails to correctly classify adversarial examples
 1432 when noise is added to the original images [127]. Rosenfeld et al. [247] illustrated
 1433 that inserting synthetic foreign objects to input images (e.g., a cartoon elephant)
 1434 can completely alter classification output. Wang et al. [289] performed similar
 1435 attacks on a transfer-learning approach of facial recognition by modifying pixels of
 1436 a celebrity's face to be recognised as a completely different celebrity, all while still
 1437 retaining the same human-interpretable original celebrity. Su et al. [268] used the
 1438 ImageNet database to show that 41.22% of images drop in confidence when just a
 1439 *single pixel* is changed in the input image; and similarly, Eykholt et al. [89] recently
 1440 showed similar results that made a convolutional neural network (CNN) interpret a
 1441 stop road-sign (with mimicked graffiti) as a 45mph speed limit sign.

1442 The results suggest that current state-of-the-art CV techniques may not be robust
 1443 enough for safety critical applications as they do not handle intentional or unin-
 1444 tentional adversarial attacks. Moreover, as such adversarial examples exist in the

1445 physical world [89, 162], “the natural world may be adversarial enough” [226] to
1446 fool AI software. Though some limitations and guidelines have been explored in this
1447 area, the perspective of *Intelligent Web Services* is yet to be considered and specific
1448 guidelines do not yet exist when using CVSSs.

1449 **Testing strategies in ML applications** Although much work applies ML tech-
1450 niques to automate testing strategies, there is only a growing emphasis that considers
1451 this in the opposite sense; that is, testing to ensure the ML product works correctly.
1452 There are few reliable test oracles that ensure if an ML has been implemented to
1453 serve its algorithm and use case purposefully; indeed, “the non-deterministic nature
1454 of many training algorithms makes testing of models even more challenging” [11].
1455 Murphy et al. [200] proposed a SE-based testing approach on ML ranking algorithms
1456 to evaluate the ‘correctness’ of the implementation on a real-world data set and prob-
1457 lem domain, whereby discrepancies were found from the formal mathematical proofs
1458 of the ML algorithm and the implementation.

1459 Recently, Braiek and Khomh [40] conducted a comprehensive review of testing
1460 strategies in ML software, proposing several research directions and recommenda-
1461 tions in how best to apply SE testing practices in ML programs. However, much
1462 of the area of this work specifically targets ML engineers, and not application de-
1463 velopers. Little has been investigated on how application developers perceive and
1464 understand ML concepts, given a lack of formal training; we note that other testing
1465 strategies and frameworks proposed (e.g., [44, 199, 208]) are targeted chiefly to the
1466 ML engineer, and not the application developer.

1467 However, Arpteg et al. [11] recently demonstrated (using real-world ML projects)
1468 the developmental challenges posed to developers, particularly those that arise when
1469 there is a lack of transparency on the models used and how to troubleshoot ML
1470 frameworks using traditional SE debugging tools. This said, there is no further in-
1471 vestigations into challenges when using the higher, ‘ML friendly’ layers (e.g., IWSs)
1472 of the ‘machine learning spectrum’ [215], rather than the ‘lower layers’ consisting
1473 of existing ML frameworks and algorithms targeted toward the ML community.

1474 4.3.2 Internal Quality

1475 **Quality metrics for cloud services** CVSSs are based on cloud computing funda-
1476 mentals under a subset of the Platform as a Service (PaaS) model. There has been
1477 work in the evaluation of PaaS in terms of quality attributes [102]: these attributes
1478 are exposed using service-level agreements (SLAs) between vendors and customers,
1479 and customers denote their demanded quality of service (QoS) to ensure the cloud
1480 services adhere to measurable KPI attributes.

1481 Although, popular services, such as cloud object storage, come with strong QoS
1482 agreement, to date IWSs do not come with deep assurances around their performance
1483 and responses, but do offer uptime guarantees. For example, how can Rosa demand
1484 a QoS that ensures all photos of dogs uploaded to her app guarantee the specific dog
1485 breeds are returned so that users can look up their other friend’s ‘border collie’s?
1486 If dog breeds are returned, what ontologies exist for breeds? Are they consistent

¹⁴⁸⁷ with each other, or shortened? ('Collie' versus 'border collie'; 'staffy' versus
¹⁴⁸⁸ 'staffordshire bull terrier')? For some applications, these unstated QoS metrics
¹⁴⁸⁹ specific to the ML service may have significant legal ramifications.

¹⁴⁹⁰ **Web service documentation and documenting ML** From the *developer's* per-
¹⁴⁹¹ spective, little has been achieved to assess IWS quality or assure quality of these
¹⁴⁹² CVSSs. web service and their APIs are the bridge between developers' needs and
¹⁴⁹³ the software components [10]; therefore, assessing such CVSSs from the quality
¹⁴⁹⁴ of their APIs is thereby directly related to the development quality [158]. Good
¹⁴⁹⁵ APIs should be intuitive and require less documentation browsing [228], thereby
¹⁴⁹⁶ increasing productivity. Conversely, poor APIs that are hard to understand and work
¹⁴⁹⁷ with reduce developer productivity, thereby reducing product quality. This typically
¹⁴⁹⁸ leads to developers congregating on forums such as Stack Overflow, leading to a
¹⁴⁹⁹ repository of unstructured knowledge likely to concern API design [293]. The con-
¹⁵⁰⁰ sequences of addressing these concerns in development leads to a higher demand
¹⁵⁰¹ in technical support (as measured in [123]) that, ultimately, causes the maintenance
¹⁵⁰² to be far more expensive, a phenomenon widely known in software engineering
¹⁵⁰³ economics [37]. Rosa, for instance, isn't aware of technical ML concepts; if she
¹⁵⁰⁴ cannot reason about what search results are relevant when browsing the service and
¹⁵⁰⁵ understanding functionality, her productivity is significantly decreased. Conceptual
¹⁵⁰⁶ understanding is critical for using APIs, as demonstrated by Ko and Riche, and the
¹⁵⁰⁷ effects of maintenance this may have in the future of her application is unknown.

¹⁵⁰⁸ Recent attempts to document attributes and characteristics on ML models have
¹⁵⁰⁹ been proposed. Model cards were introduced by Mitchell et al. [197] to describe how
¹⁵¹⁰ particular models were trained and benchmarked, thereby assisting users to reason
¹⁵¹¹ if the model is right for their purposes and if it can achieve its stated outcomes.
¹⁵¹² Gebru et al. [106] also proposed datasheets, a standardised documentation format to
¹⁵¹³ describe the need for a particular data set, the information contained within it and
¹⁵¹⁴ what scenarios it should be used for, including legal or ethical concerns.

¹⁵¹⁵ However, while target audiences for these documents may be of a more technical
¹⁵¹⁶ AI level (i.e., the ML engineer), there is still no standardised communication format
¹⁵¹⁷ for application developers to reason about using particular IWSs, and the ramifica-
¹⁵¹⁸ tions this may have on the applications they write is not fully conveyed. Hence, our
¹⁵¹⁹ work is focused on the application developer perspective.

¹⁵²⁰ 4.4 Method

¹⁵²¹ This study organically evolved by observing phenomena surrounding CVSSs by as-
¹⁵²² sessing both their documentation and responses. We adopted a mixed methods
¹⁵²³ approach, performing both qualitative and quantitative data collection on these two
¹⁵²⁴ key aspects by using documentary research methods for inspecting the documen-
¹⁵²⁵ tation and structured observations to quantitatively analyse the results over time.
¹⁵²⁶ This, ultimately, helped us shape the following research hypotheses which this paper
¹⁵²⁷ addresses:

1528 [RH1] CVSS do not respond with consistent outputs between services, given the
1529 same input image.

1530 [RH2] The responses from CVSS are non-deterministic and evolving, and the same
1531 service can change its top-most response over time given the same input
1532 image.

1533 [RH3] CVSS do not effectively communicate this evolution and instability, intro-
1534 ducing risk into engineering these systems.

1535 We conducted two experiments to address these hypotheses against three popular
1536 CVSSs: AWS Rekognition [313], Google Cloud Vision [327], Azure Computer
1537 Vision [335]. Specifically, we targeted the AWS DetectLabels endpoint [311],
1538 the Google Cloud Vision annotate:images endpoint [325] and Azure's analyze
1539 endpoint [336]. For the remainder of this paper, we de-identify our selected CVSSs
1540 by labelling them as services A, B and C but do not reveal mapping to prevent
1541 any implicit bias. Our selection criteria for using these particular three services
1542 are based on the weight behind each service provider given their prominence in
1543 the industry (Amazon, Google and Microsoft), the ubiquity of their hosting cloud
1544 platforms as industry leaders of cloud computing (i.e., AWS, Google Cloud and
1545 Azure), being in the top three most adopted cloud vendors in enterprise applications
1546 in 2018 [240] and the consistent popularity of discussion amongst developers in
1547 developer communities such as Stack Overflow. While we choose these particular
1548 cloud CVSSs, we acknowledge that similar services [316, 317, 324, 330, 331, 372, 373]
1549 also exist, including other popular services used in Asia [315, 329, 338, 339] (some
1550 offering 3D image analysis [314]). We reflect on the impacts this has to our study
1551 design in Section 4.7.

1552 Our study involved an 11-month longitudinal study which consisted of two 13
1553 week and 17 week experiments from April to August 2018 and November 2018 to
1554 March 2019, respectively. Our investigation into documentation occurred on August
1555 28 2018. In total, we assessed the services with three data sets; we first ran a pilot
1556 study using a smaller pool of 30 images to confirm the end-points remain stable,
1557 re-running the study with a larger pool of images of 1,650 and 5,000 images. Our
1558 selection criteria for these three data sets were that the images had to have varying
1559 objects, taken in various scenes and various times. Images also needed to contain
1560 disparate objects. Our small data set was sourced by the first author by taking photos
1561 of random scenes in an afternoon, whilst our second data set was sourced from
1562 various members of our research group from their personal photo libraries. We also
1563 wanted to include a data set that was publicly available prior to running our study,
1564 so for this data set we chose the COCO 2017 validation data set [171]. We have
1565 made our other two data sets available online ([319]). We collected results and their
1566 responses from each service's API endpoint using a python script [323] that sent
1567 requests to each service periodically via cron jobs. Table 4.1 summarises various
1568 characteristics about the data sets used in these experiments.

1569 We then performed quantitative analyses on each response's labels, ensuring all
1570 labels were lowercased as case changed for services A and C over the evaluation
1571 period. To derive at the consistency of responses for each image, we considered only
1572 the 'top' labels per image for each service and data set. That is, for the same image i

Table 4.1: Characteristics of our datasets and responses.

Data set	Small	Large	COCOVal17
# Images/data set	30	1,650	5000
# Unique labels found	307	3506	4507
Number of snapshots	9	22	22
Avg. days b/n requests	12 Days	8 Days	8 Days

1573 over all images in data set D where $i \in D$ and over the three services, the top labels
 1574 per image (T_i) of all labels per image L_i (i.e., $T_i \subseteq L_i$) is that where the respective
 1575 label's confidences are consistently the highest of all labels returned. Typically, the
 1576 top labels returned is a set containing only one element—that is, only one unique
 1577 label consistently returned with the highest label ($|T_i| = 1$)—however there are cases
 1578 where the top labels contains multiple elements as their respective confidences are
 1579 *equal* ($|T_i| > 1$).

1580 We measure response consistency under 6 aspects:

- 1581 **(1) Consistency of the top label between each service.** Where the same image of,
 1582 for example, a dog is sent to the three services, the top label for service A may
 1583 be ‘animal’, B ‘canine’ and C ‘animal’. Therefore, service B is inconsistent.
- 1584 **(2) Semantic consistency of the top labels.** Where a service has returned multi-
 1585 ple top labels ($|T_i| > 1$), there may lie semantic differences in what the service
 1586 thinks the image best represents. Therefore, there is conceptual inconsistency
 1587 in the top labels for a service even when the confidences are equal.
- 1588 **(3) Consistency of the top label’s confidence per service.** The top label for
 1589 an image does not guarantee a high confidence. Therefore, there may be
 1590 inconsistencies in how confident the top labels for all images in a service is.
- 1591 **(4) Consistency of confidence in the intersecting top label between each ser-**
 1592 **vice.** The spread of a top intersecting label, e.g., ‘cat’, may not have the same
 1593 confidences per service even when all three services agree that ‘cat’ is the top
 1594 label. Therefore, there is inconsistency in the confidences of a top label even
 1595 where all three services agree.
- 1596 **(5) Consistency of the top label over time.** Given an image, the top label in one
 1597 week may differ from the top label the following week. Therefore, there is
 1598 inconsistency in the top label itself due to model evolution.
- 1599 **(6) Consistency of the top label’s confidence over time.** The top label of an
 1600 image may remain static from one week to the next for the same service, but
 1601 its confidence values may change with time. Therefore, there is inconsistency
 1602 in the top label’s confidence due to model evolution.

1603 For the above aspects of consistency, we calculated the spread of variation for the
 1604 top label’s confidences of each service for every 1 percent point; that is, the frequency
 1605 of top label confidences within 100–99%, 99–98% etc. The consistency of top label’s
 1606 and their confidences between each service was determined by intersecting the labels
 1607 of each service per image and grouping the intersecting label’s confidences together.



Figure 4.1: The only consistent label for the above image is ‘people’ for services C and B. The top label for A is ‘conversation’ and this label is not registered amongst the other two services.

Table 4.2: Ratio of the top labels (to images) that intersect in each data set for each permutation of service.

Service	Small	Large	COCOVal17	μ	σ
$A \cap B \cap C$	3.33%	2.73%	4.68%	2.75%	0.0100
$A \cap B$	6.67%	11.27%	12.26%	10.07%	0.0299
$A \cap C$	20.00%	13.94%	17.28%	17.07%	0.0304
$B \cap C$	6.67%	12.97%	20.90%	13.51%	0.0713

1608 This allowed us to determine relevant probability distributions. For reproducibility,
 1609 all quantitative analysis is available online [320].

1610 4.5 Findings

1611 4.5.1 Consistency of top labels

1612 **Consistency across services** Table 4.2 presents the consistency of the top labels
 1613 between data sets, as measured by the cardinality of the intersection of all three ser-
 1614 vices’ set of top labels divided by the number of images per data set. A combination
 1615 of services present varied overlaps in their top labels; services A and C provide the
 1616 best overlap for all three data sets, however the intersection of all three irrespective
 1617 of data sets is low.

1618 The implication here is that, without semantic comparison (see Section 4.7),
 1619 service vendors are not ‘plug-and-play’. If Rosa uploaded the sample images in
 1620 this paper to her application to all services, she would find that only Figure 4.1
 1621 responds with ‘person’ for services B and C in their respective set of top labels.
 1622 However, if she decides to then adopt service A, then Figure 4.1’s top label becomes
 1623 ‘conversation’; the ‘person’ label does not appear within the top 15 labels for service
 1624 A and, conversely, the ‘conversation’ label does not appear in the other services top
 1625 15.

1626 Should she decide if the performance of a particular service isn’t to her needs,
 1627 then the vocabulary used for these labels becomes inconsistent for all other images;

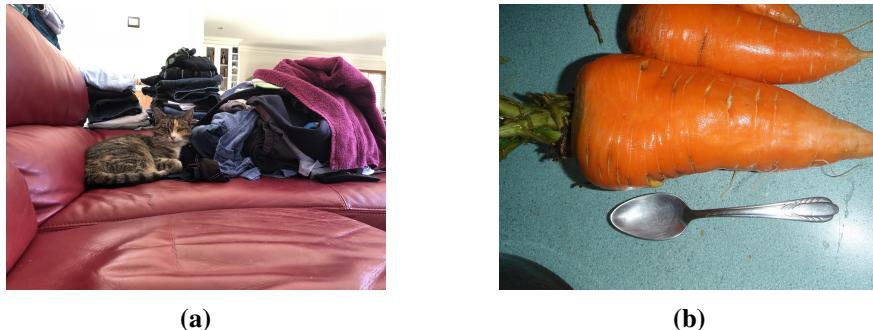


Figure 4.2: *Left:* The top labels for each service do not intersect, with each having a varied ontology: $T_i = \{ A = \{ \text{'black'} \}, B = \{ \text{'indoor'} \}, C = \{ \text{'slide'}, \text{'toy'} \} \}$. (Service C returns both ‘slide’ and ‘toy’ with equal confidence.) *Right:* The top labels for each service focus on disparate subjects in the image: $T_i = \{ A = \{ \text{'carrot'} \}, B = \{ \text{'indoor'} \}, C = \{ \text{'spoon'} \} \}$.

that is, the top label sets per service for Figure 4.2a shows no intersection at all. Furthermore, the part of the image each service focuses on may not be consistent for their top labels; in Figure 4.2b, service A’s top label focuses on the vegetable (‘carrot’), service C focuses on the ‘spoon’, while service B’s focus is that the image is ‘indoor’s. It is interesting to note that service B focuses on the scene matter (indoors) rather than the subject matter. (Furthermore, we do not actually know if the image in Figure 4.2b was taken indoors.)

Hence, developers should ensure that the vocabulary used by a particular service is right for them before implementation. As each service does not work to the same standardised model, trained with disparate training data, and tuned differently, results will differ despite the same input. This is unlike deterministic systems: for example, switching from AWS Object Storage to Google Cloud Object storage will conceptually provide the same output (storing files) for the same input (uploading files). However, CVSs do not agree on the top label for images, and therefore developers are likely to be vendor locked, making changes between services non-trivial.

Semantic consistency where $|T_i| > 1$ Service C returns two top labels for Figure 4.2a; ‘slide’ and ‘toy’. More than one top label is typically returned in service C (80.00%, 56.97%, and 81.66% of all images for all three data sets, respectively) though this also occurs in B in the large (4.97% of all images) and COCOVal17 data sets (2.38%). Semantic inconsistencies of what this label conceptually represents becomes a concern as these labels have confidences of *equal highest* consistency. Thus, some services are inconsistent in themselves and cannot give a guaranteed answer of what exists in an image; services C and B have multiple top labels, but the respective services cannot ‘agree’ on what the top label actually is. In Figure 4.3a, service C presents a reasonably high confidence for the set of 7 top labels it returns, however there is too much diversity ranging from a ‘hot chocolate’ to the hypernym ‘food’. Both are technically correct, but it is up to the developer to decide the level of hypernymy to label the image as. We also observe a similar effect in Figure 4.3b,



Figure 4.3: *Left:* Service C is 98.49% confident of the following labels: { ‘beverage’, ‘chocolate’, ‘cup’, ‘dessert’, ‘drink’, ‘food’, ‘hot chocolate’ }. However, it is up to the developer to decide which label to persist with as all are returned. *Right:* Service B persistently returns a top label set of { ‘book’, ‘several’ }. Both are semantically correct for the image, but disparate in what the label is to describe.

1657 where the image is labelled with both the subject matter and the number of subjects
 1658 per image.

1659 Thus, a taxonomy of ontologies is unknown; if a ‘border collie’ is detected in
 1660 an image, does this imply the hypernym ‘dog’ is detected, and then ‘mammal’, then
 1661 ‘animal’, then ‘object’? Only service B documents a taxonomy for capturing what
 1662 level of scope is desired, providing what it calls the ‘86-category’ concept as found
 1663 in its how-to guide:

1664 “*Identify and categorize an entire image, using a category taxonomy*
 1665 *with parent/child hereditary hierarchies. Categories can be used alone,*
 1666 *or with our new tagging models.*” [337]

1667 Thus, even if Rosa implemented conceptual similarity analysis for the image, the
 1668 top label set may not provide sufficient information to derive at a conclusive answer,
 1669 and if simply relying on only one label in this set, information such as the duplicity
 1670 of objects (e.g., ‘several’ in Figure 4.3b) may be missed.

1671 4.5.2 Consistency of confidence

1672 **Consistency of top label’s confidence** In Figure 4.4, we see that there is high
 1673 probability that top labels have high confidences for all services. In summary, one in
 1674 nine images uploaded to any service will return a top label confident to at least 97%.
 1675 However, there is higher probability for service A returning a lower confidence,
 1676 followed by B. The best performing service is C, with 90% of requests having a top
 1677 label confident to $\gtrapprox 95\%$, when compared to $\gtrapprox 87\%$ and $\gtrapprox 93\%$ for services A and
 1678 B, respectively.

1679 Therefore, Rosa could generally expect that the top labels she receives in her
 1680 images do have high confidence. That is, each service will return a top label that
 1681 they are confident about. This result is expected, considering that the ‘top’ label
 1682 is measured by the highest confidence, though it is interesting to note that some
 1683 services are generally more confident than others in what they present back to users.

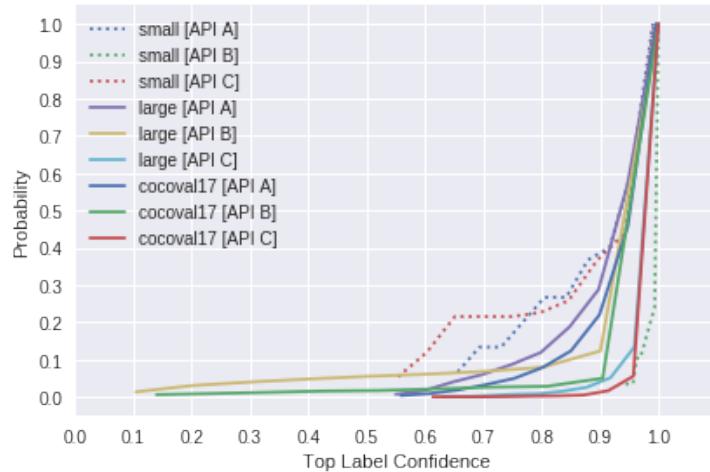


Figure 4.4: Cumulative distribution of the top labels' confidences. One in nine images return a top label(s) confident to $\gtrsim 97\%$, though there is a wider distribution for service A.

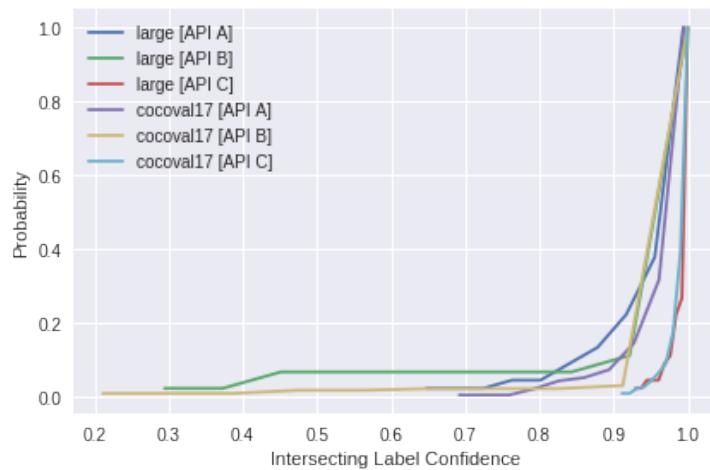


Figure 4.5: Cumulative distribution of intersecting top labels' confidences. The small data set is intentionally removed due to low intersections of labels (see Table 4.2).

Table 4.3: Ratio of the top labels (to images) that remained the top label but changed confidence values between intervals.

Service	Small	Large	COCOVal17	$\mu(\delta_c)$	$\sigma(\delta_c)$	Median(δ_c)	Range(δ_c)
A	53.33%	59.19%	44.92%	9.62e-8	6.84e-8	5.96e-8	[5.96e-8, 6.56e-7]
B	0.00%	0.00%	0.02%	-	-	-	-
C	33.33%	41.36%	15.60%	5.35e-7	8.76e-7	3.05e-7	[1.27e-7, 1.13e-5]



Figure 4.6: All three services agree the top label for the above image is ‘food’, but the confidences to which they agree by vary significantly. Service C is most confident to 94.93% (in addition with the label ‘bread’); service A is the second most confident to 84.32%; service B is the least confident with 41.39%.

1684 **Consistency of intersecting top label’s confidence** Even where all three services
 1685 do agree on a set of top labels, the disparity of how much they agree by is still of
 1686 importance. Just because three services agree that an image contains consistent top
 1687 labels, they do not always have a small spread of confidence. In Figure 4.6, the three
 1688 services agree with $\sigma = 0.277$, significantly larger than that of all images in general
 1689 $\sigma = 0.0831$. Figure 4.5 displays the cumulative distribution of all intersecting top
 1690 labels’ confidence values, presenting slightly similar results to that of Figure 4.4.

1691 4.5.3 Evolution risk

1692 **Label Stability** Generally, the top label(s) did not evolve in the evaluation period.
 1693 16.19% and 5.85% of images did change their top label(s) in the Large and COCO-
 1694 Val17 data sets in service A. Thus, top labels are stable but not guaranteed to be
 1695 constant.

1696 **Confidence Stability** Similarly, where the top label(s) remained the same from
 1697 one interval to the next, the confidence values were stable. Table 4.3 displays the
 1698 proportion of images that changed their top label’s confidence values with various
 1699 statistics on the confidence deltas between snapshots (δ_c). However, this delta is so
 1700 minuscule that we attribute such changes to statistical noise.

1701 4.6 Recommendations

1702 4.6.1 Recommendations for IWS users

1703 **Test with a representative ontology for the particular use case** Rosa should
1704 ensure that in her testing strategies for the app she develops, there is an ontology
1705 focus for the types of vocabulary that are returned. Additionally, we noted that there
1706 was a sudden change in case for services A and C; for all comparative purposes of
1707 labels, each label should be lower-cased.

1708 **Incorporate a specialised IWS testing methodology into the development life-
1709 cycle** Rosa can utilise the different aspects of consistency as outlined in this paper
1710 as part of her quality strategy. To ensure results are correct over time, we recom-
1711 mend developers create a representative data set of the intended application's data
1712 set and evaluate these changes against their chosen service frequently. This will
1713 help identify when changes, if any, have occurred if vendors do not provide a line of
1714 communication when this occurs.

1715 **IWSs are not ‘plug-and-play’** Rosa will be locked into whichever vendor she
1716 chooses as there is inherent inconsistency between these services in both the vocab-
1717 ularly and ontologies that they use. We have demonstrated that very few services
1718 overlap in their vocabularies, chiefly because they are still in early development and
1719 there is yet to be an established, standardised vocabulary that can be shared amongst
1720 the different vendors. Issues such as those shown in Section 4.5.1 can therefore be
1721 avoided.

1722 Throughout this work, we observed that the terminologies used by the vari-
1723 ous vendors are different. Documentation was studied, and we note that there is
1724 inconsistency between the ways techniques are described to users. We note the
1725 disparity between the terms ‘detection’, ‘recognition’, ‘localisation’ and ‘analysis’.
1726 This applies chiefly to object- and facial-related techniques. Detection applies to
1727 facial detection, which gives bounding box coordinates around all faces in an image.
1728 Similarly, localisation applies the same methodology to disparate objects in an image
1729 and labels them. In the context of facial ‘recognition’, this term implies that a face
1730 is *recognised* against a known set of faces. Lastly, ‘analysis’ applies in the context
1731 of facial analysis (gender, eye colour, expression etc.); there does not exist a similar
1732 analysis technique on objects.

1733 We notice similar patterns with object ‘tagging’, ‘detection’ and ‘labelling’.
1734 Service A uses ‘Entity Detection’ for object categorisation, service B uses ‘Image
1735 Tagging’, and service C uses the term ‘Detect Labels’: conceptually, these provide
1736 the same functionality but the lack of consistency used between all three providers is
1737 concerning and leaves room for confusion with developers during any comparative
1738 analyses. Rosa may find that she wants to label her images into day/night scenes, but
1739 this in turn means the ‘labelling’ of varying objects. There is therefore no consistent
1740 standards to use the same terminology for the same concepts, as there are in other
1741 developer areas (such as Web Development).

1742 **Avoid use in safety-critical systems** We have demonstrated in this paper that both
1743 labels and confidences are stable but not constant; there is still an evolution risk posed
1744 to developers that may cause unknown consequences in applications dependent on
1745 these CVSs. Developers should avoid their use in safety critical systems due to the
1746 lack of visible changes.

1747 **4.6.2 Recommendations for IWS providers**

1748 **Improve the documentation** Rosa does not know that service A returns back
1749 ‘carrot’ for its top response, with service C returning ‘spoon’ (Figure 4.2b). She
1750 is unable to tell the service’s API where to focus on the image. Moreover, how
1751 can she toggle the level of specificity in her results? She is frustrated that service
1752 C can detect ‘chocolate’, ‘food’ and also ‘beverage’ all as the same top label in
1753 Figure 4.3a: what label is she to choose when the service is meant to do so for her,
1754 and how does she get around this? Thus, we recommend vendors to improve the
1755 documentation of services by making known the boundary set of the training data
1756 used for the algorithms. By making such information publicly available, developers
1757 would be able to review the service’s specificity for their intended use case (e.g.,
1758 maybe Rosa is satisfied her app can catalogue ‘food’ together, and in fact does not
1759 want specific types of foods (‘hot chocolate’) catalogued). We also recommend that
1760 vendors publish usage guidelines should that include details of priors and how to
1761 evaluate the specific service results.

1762 Furthermore, we did not observe that the vendors documented how some images
1763 may respond with multiple labels of the exact same confidence value. It is not clear
1764 from the documentation that response objects can have duplicate top values, and
1765 tutorials and examples provided by the vendors do not consider this possibility. It
1766 is therefore left to the developer to decide which label from this top set of labels
1767 best suits for their particular use case; the documentation should describe that a rule
1768 engine may need to be added in the developer’s application to verify responses. The
1769 implications this would have on maintenance would be significant.

1770 **Improve versioning** We recommend introducing a versioning system so that a
1771 model can be used from a specific date in production systems: when Rosa tests her
1772 app today, she would like the service to remain *static* the same for when her app is
1773 deployed in production tomorrow. Thus, in a request made to the vendor, Rosa could
1774 specify what date she ran her app’s QA testing on so that she knows that henceforth
1775 these model changes will not affect her app.

1776 **Improve Metadata in Response** Much of the information in these services is
1777 reduced to a single confidence value within the response object, and the details about
1778 training data and the internal AI architecture remains unknown; little metadata is
1779 provided back to developers that encompass such detail. Early work into model
1780 cards and datasheets [106, 197] suggests more can be done to document attributes
1781 about ML systems, however at a minimum from our work, we recommend including
1782 a reference point via the form of an additional identifier. This identifier must

1783 also permit the developers to submit the identifier to another API endpoint should
1784 the developer wish to find further characteristics about the AI empowering the IWS,
1785 reinforcing the need for those presented in model cards and datasheets. For example,
1786 if Rosa sends this identifier she receives in the response object to the IWS descriptor
1787 API, she could find out additional information such as the version number or date
1788 when the model was trained, thereby resolving potential evolution risk, and/or the
1789 ontology of labels.

1790 **Apply constraints for predictions on all inputs** In this study, we used some
1791 images with intentionally disparate, and noisy objects. If services are not fully
1792 confident in the responses they give back, a form of customised error message
1793 should be returned. For example, if Rosa uploads an image of 10 various objects
1794 on a table, rather than returning a list of top labels with varying confidences, it
1795 may be best to return a ‘too many objects’ exception. Similarly, if Rosa uploads a
1796 photo that the model has had no priors on, it might be useful to return an ‘unknown
1797 object’ exception than to return a label it has no confidence of. We do however
1798 acknowledge that current state of the art CV techniques may have limits in what they
1799 can and cannot detect, but this limitation can be exposed in the documentation to the
1800 developers.

1801 A further example is sending a one pixel image to the service, analogous to
1802 sending an empty file. When we uploaded a single pixel white image to service A,
1803 we received responses such as ‘microwave oven’, ‘text’, ‘sky’, ‘white’ and ‘black’
1804 with confidences ranging from 51–95%. Prior checks should be performed on all
1805 input data, returning an ‘insufficient information’ error where any input data is below
1806 the information of its training data.

1807 4.7 Threats to Validity

1808 4.7.1 Internal Validity

1809 Not all CVSs were assessed. As suggested in Section 4.4, we note that there are
1810 other CVSs such as IBM Watson. Many services from Asia were also not considered
1811 due to language barriers (of the authors) in assessing these services. We limited our
1812 study to the most popular three providers (outside of Asia) to maintain focus in this
1813 body of work.

1814 A custom confidence threshold was not set. All responses returned from each of
1815 the services were included for analysis; where confidences were low, they were still
1816 included for analysis. This is because we used the default thresholds of each API to
1817 hint at what real-world applications may be like when testing and evaluating these
1818 services.

1819 The label string returned from each service was only considered. It is common
1820 for some labels to respond back that are conceptually similar (e.g., ‘car’ vs. ‘automobile’)
1821 or grammatically different (e.g., ‘clothes’ vs. ‘clothing’). While we could have
1822 employed more conceptual comparison or grammatical fixes in this study, we chose

1823 only to compare lowercased labels and as returned. We leave semantic comparison
1824 open to future work.

1825 Only introductory analysis has been applied in assessing the documentation of
1826 these services. Further detailed analysis of documentation quality against a rigorous
1827 documentation quality framework would be needed to fortify our analysis of the
1828 evolution of these services' documentation.

1829 **4.7.2 External Validity**

1830 The documentation and services do change over time and evolve, with many allowing
1831 for contributions from the developer community via GitHub. We note that our
1832 evaluation of the documentation was conducted on a single date (see Section 4.4)
1833 and acknowledge that the documentation may have changed from the evaluation date
1834 to the time of this publication. We also acknowledge that the responses and labelling
1835 may have evolved too since the evaluation period described and the date of this
1836 publication. Thus, this may have an impact on the results we have produced in this
1837 paper compared to current, real-world results. To mitigate this, we have supplied the
1838 raw responses available online [321].

1839 Moreover, in this paper we have investigated *computer vision* services. Thus,
1840 the significance of our results to other domains such as natural language processing
1841 or audio transcription is, therefore, unknown. Future studies may wish to repeat our
1842 methodology on other domains to validate if similar patterns occur; we remain this
1843 open for future work.

1844 **4.7.3 Construct Validity**

1845 It is not clear if all the recommendations proposed in Section 4.6 are feasible
1846 or implementable in practice. Construct validity defines how well an experiment
1847 measures up to its claims; the experiments proposed in this paper support our three
1848 hypotheses but these have been conducted in a clinical condition. Real-world case
1849 studies and feedback from developers and providers in industry would remove the
1850 controlled nature of our work.

1851 **4.8 Conclusions & Future Work**

1852 This study explored three popular CVSS over an 11 month longitudinal experiment
1853 to determine if these services pose any evolution risk or inconsistency. We find that
1854 these services are generally stable but behave inconsistently; responses from these
1855 services do change with time and this is not visible to the developers who use them.
1856 Furthermore, the limitations of these systems are not properly conveyed by vendors.
1857 From our analysis, we present a set of recommendations for both IWS vendors and
1858 developers.

1859 Standardised software quality models (e.g., [136]) target maintainability and
1860 reliability as primary characteristics. Quality software is stable, testable, fault
1861 tolerant, easy to change and mature. These CVSS are, however, in a nascent stage,

1862 difficult to evaluate, and currently are not easily interchangeable. Effectively, the
1863 IWS response objects are shifting in material ways to developers, albeit slowly, and
1864 vendors do not communicate this evolution or modify API endpoints; the endpoint
1865 remains static but the content returned does not despite the same input.

1866 There are many potential directions stemming from this work. To start, we plan
1867 to focus on preparing a more comprehensive datasheet specifically targeted at what
1868 should be documented to application developers, and not data scientists. Reapplying
1869 this work in real-world contexts, that is, to get real developer opinions and study
1870 production grade systems, would also be beneficial to understand these phenomena
1871 in-context. This will help us clarify if such changes are a real concern for developers
1872 (i.e., if they really need to change between services, or the service evolution has real
1873 impact on their applications). We also wish to refine and systematise the method
1874 used in this study and develop change detectors that can be used to identify evolution
1875 in these services that can be applied to specific ML domains (i.e., not just CV),
1876 data sets, and API endpoints, thereby assisting application developers in their testing
1877 strategies. Moreover, future studies may wish to expand the methodology applied by
1878 refining how the responses are compared. As there does not yet exist a standardised
1879 list of terms available between services, labels could be *semantically* compared
1880 instead of using exact matches (e.g., by using stem words and synonyms to compare
1881 similar meanings of these labels), similar to previous studies [212].

1882 This paper has highlighted only some high-level issues that may be involved
1883 in using these evolving services. The laws of software evolution suggest that for
1884 software to be useful, it must evolve [193, 279]. There is, therefore, a trade-off, as
1885 we have shown, between consistency and evolution in this space. For a component
1886 to be stable, any changes to dependencies it relies on must be communicated. We
1887 are yet to see this maturity of communication from IWS providers. Thus, developers
1888 must be cautious between integrating intelligent components into their applications
1889 at the expense of stability; as the field of AI is moving quickly, we are more likely to
1890 see further instability and evolution in IWSs as a consequence.

1891 CHAPTER 5

1892
1893 Systematic Mapping Study of API Documentation Knowledge[†]
1894

1895 **Abstract** Good application programming interface (API) documentation facilities the de-
1896 velopment process, improving productivity and quality. While the topic of API documen-
1897 tation quality has been of interest for the last two decades, there have been few studies
1898 to map the specific constructs needed to create a good document. In effect, we still need
1899 a structured taxonomy against which to capture knowledge. This study reports emerging
1900 results of a systematic mapping study. We capture key conclusions from previous studies
1901 that assess API documentation quality, and synthesise the results into a single framework.
1902 By conducting a systematic review of 21 key works, we have developed a five dimensional
1903 taxonomy based on 34 categorised weighted recommendations. All studies utilise field study
1904 techniques to arrive at their recommendations, with seven studies employing some form of
1905 interview and questionnaire, and four conducting documentation analysis. The taxonomy we
1906 synthesise reinforces that usage description details (code snippets, tutorials, and reference
1907 documents) are generally highly weighted as helpful in API documentation, in addition to
1908 design rationale and presentation. We propose extensions to this study aligned to developer's
1909 utility for each of the taxonomy's categories.

1910 **5.1 Introduction**

1911 Improving the quality of application programming interface (API) documentation is
1912 highly valuable to the software development process; good documentation facilitates
1913 productivity and thus quality is better engineered into the system [190]. Where an
1914 application developer integrates new pieces of functionality (via APIs) into a system,
1915 their productivity is affected either by inadequate skills (“*I've never used an API like
1916 this, so must learn from scratch*”) or, where their skills are adequate, an imbalanced
1917 cognitive load that causes excessive context switching (“*I have the skills for this,*

[†]This chapter is originally based on A. Cummaudo, R. Vasa, and J. Grundy, “Assessing the efficacy of API documentation knowledge against practitioners and computer vision services,” 2020, Unpublished. Terminology has been updated to fit this thesis.

1918 *but am confused or misunderstand*"). In the latter case, what causes this confusion
1919 and how to mitigate it via improved API documentation is an area that has been
1920 explored; prior studies have provided recommendations based on both qualitative
1921 and quantitative analysis of developer's opinions. These recommendations and
1922 guidelines propose ways by which developers, managers and solution architects can
1923 construct systems better.

1924 However, to date there has been little attempt to systematically capture this
1925 knowledge about API documentation from various studies into a readily accessible,
1926 consolidated format, that assists API designers to prepare better documentation.
1927 While previous works have covered certain aspects of API usage, many have lacked
1928 a systematic review of literature and do not offer a taxonomy to consolidate these
1929 guidelines together. For example, some studies have considered the technical imple-
1930 mentation improving API usability or tools to generate (or validate) API documen-
1931 tation from its source code (e.g., [179, 210, 294]); there still lacks a consolidated
1932 effort to capture the knowledge and artefacts best suited to *manually write* API
1933 documentation.

1934 *⟨ todo: Introduce intelligent services documentation ⟩* The need for these insights
1935 to be well-captured is evermore present with the introduction of intelligent web
1936 services (IWSs), in which an AI-based component produces a non-deterministic
1937 result based on a machine-learnt data-driven algorithm, rather than a predictable,
1938 rule-driven one [69]. A popular subset of such components include computer vision
1939 services (CVSs), which use machine intelligence to make predictions on images
1940 such as object categorisation or facial recognition [313, 314, 315, 316, 317, 324,
1941 327, 329, 330, 331, 335, 338, 339, 372, 373]. The longer-term impacts of *poor*
1942 *documentation* for CVSs is largely under-explored and may have significant impacts
1943 of AI-first systems if the application developers who use them do not think in the
1944 non-deterministic mental model of the designers who create CVSs.

1945 This paper presents outcomes from a preliminary work to address this gap and
1946 offers four key contributions:

- 1947 • a systematic mapping study (SMS) consisting of 21 studies that capture what
1948 knowledge or artefacts should be contained within API documentation; and,
- 1949 • a structured taxonomy based on the consolidated recommendations of these
1950 21 studies.
- 1951 • *⟨ todo: Relevance as per devs ⟩* an assessment of the efficacy of these rec-
1952 commendations via a 'relevance ranking' for each of the 34 categories that
1953 empirically reflects what is important to document from a *practitioner* point
1954 of view;
- 1955 • *⟨ todo: Assessment against docs ⟩* a heuristic validation of the taxonomy
1956 against CVSs to assess where existing CVSs documentation needs improve-
1957 ment.

1958 After performing our SMS on what API knowledge should be captured in
1959 documentation—to assist API designers—we propose a five dimensional taxonomy
1960 consisting of: (1) Usage Description; (2) Design Rationale; (3) Domain Concepts;
1961 (4) Support Artefacts; and (5) Documentation Presentation. *⟨ todo: Describe that*

1962 *this was weighted against devs opinions*) This taxonomy was then surveyed against
1963 X developers to assess the relevance of these weightings from the practitioner’s
1964 viewpoint. ⟨ todo: *Describe our application of the taxonomy to CVS* ⟩ We then
1965 assess our taxonomy against a subset of IWSs—three popular CVS: Google Cloud
1966 Vision [327], AWS Rekognition [313] and Azure Computer Vision [335]. We an-
1967 ticipate that future API designers may use this resource as a means to improve the
1968 ways in which they communicate their mental models and patterns of thinking while
1969 developing these APIs.

1970 This paper is structured as thus: Section 5.2 presents related work in the area;
1971 Section 5.3 is divided into two subsections, the first describing how primary sources
1972 were selected in a SMS with the second describing the development of our taxonomy
1973 from these sources; ⟨ todo: *Introduce section where we validated study* ⟩ Section 5.4
1974 describes how we adopted the System Usability Scale (SUS) to develop a survey
1975 instrument of 52 questions to validate the taxonomy and assess its efficacy against
1976 the three popular CVS selected; Section 5.5 presents the findings from our validation
1977 and the proposed taxonomy; Section 5.6 describes the threats to validity of this work
1978 and Section 5.7 provides concluding remarks and the future directions of this study.

1979 5.2 Related Work

1980 SMSs have previously been explored in the area of API usability and developer
1981 experience. Nybom et al. [210] recently performed a SMS on 36 API documentation
1982 generation tools and approaches. Presented is an analysis of state-of-the-art of
1983 the tools developed, what kind of documentation is generated by them, and the
1984 dependencies they require to generate this documentation. Their findings highlight
1985 a recent effort on the development of API documentation by producing example
1986 code snippets and/or templates on how to use the API or bootstrap developers to
1987 begin using the APIs. A secondary focus is closely followed by tools that produce
1988 natural language descriptions that can be produced within developer documentation.
1989 However, Nybom et al. produce a SMS on the types of *tooling* that exists to assist
1990 in producing and validating API documentation. While this is a systematic study
1991 with key insights into the types of tooling produced, there is still a gap for a SMS in
1992 what *guidelines* have been produced by the literature in developing natural-language
1993 documentation itself, which our work has addressed.

1994 Watson [294] performed a heuristic assessment of 11 high-level universal design
1995 elements of API documentation against 35 popular APIs. He demonstrated that many
1996 of these popular APIs fail to grasp even the basic of these elements; for example,
1997 25% of the documentation sets did not provide any basic overview documentation.
1998 However, the heuristic used within this study consists of just 11 elements and is based
1999 on only three seminal works. Our work extends these heuristics and structures them
2000 into a consolidated, hierarchical taxonomy using a systematic taxonomy development
2001 method for software engineering (SE).

2002 A taxonomy of knowledge patterns within API reference documentation by
2003 Maalej and Robillard [179] classified 12 distinct knowledge types. Evaluation of the
2004 taxonomy against JDK 6 and .NET 4.0 showed that, while functionality and structure

of the API is well-communicated, core concepts and rationale about the API are quite rare to find. Moreover, they demonstrated that low-value ‘non-information’ (documentation that provides uninformative boilerplate text with no insight into the API at all) is substantially present in the documentation of methods and fields in these APIs. Their findings recommend that developers factor their 12 distinct knowledge types into the process of code documentation and prevent documenting low-value documentation. The development of their taxonomy consisted of questions to model knowledge and information, thereby capturing the reason about disparate information units independent to context; a key difference to this paper is the systematic taxonomy approach utilised.

⟨ *todo: AC: Paragraph on the SUS; paragraph on ICVS* ⟩

2016 5.3 Method

2017 Our taxonomy development consisted of two phases. Firstly, we conducted an
2018 SMS to identify and analyse API documentation studies, following the guidelines
2019 of Kitchenham and Charters [153] and Petersen et al. [225]. Following this, we
2020 followed the SE taxonomy development method devised by Usman et al. [284] on
2021 our findings from the SMS.

2022 5.3.1 Systematic Mapping Study

2023 **Research Questions (RQs)** To guide our SMS, we developed the following RQs:

RQ1 What knowledge do API documentation studies contribute?

RQ2 How is API documentation studied?

2024 The intent behind RQ1 is to collate as much of the insight provided by the
2025 literature on how API providers should best document their work. This helped us
2026 shape and form the taxonomy provided in Section 5.5. RQ2 addresses methodologies
2027 by which these studies come to these conclusions to identify gaps in literature where
2028 future studies can potentially focus.

2029 **Automatic Filtering** Informed by similar previous studies in SE [104, 110, 284],
2030 we begin by defining the SWEBOK [132] knowledge areas (KAs) to assist in the
2031 search and mapping process of an SMS. Our search query was built using related
2032 KAs, relevant synonyms, and the term ‘software engineering’ (for comprehensiveness)
2033 all joined with the OR operator. Due to the lack of a standard definition of an
2034 API, we include the terms: ‘API’ and its expanded term; software library, compo-
2035 nent and framework; and lastly SDK and its expanded term. These too were joined
2036 with the OR operator, appended with an AND. Lastly, the term ‘documentation’ was
2037 appended with an AND. Our final search string was:

Table 5.1: Summary of our search results and publication types

Publication type	WoS	C/I	Scopus	Total
Conference Paper	27	442	2353	2822
Journal Article	41	127	1236	1404
Book	23	17	224	264
Other	0	5	6	11
Total	91	591	3819	4501

(“software design” OR “software architecture” OR “software construction” OR “software development” OR “software maintenance” OR “software engineering process” OR “software process” OR “software lifecycle” OR “software methods” OR “software quality” OR “software engineering professional practice” OR “software engineering”) AND (API OR “application programming interface” OR “software library” OR “software component” OR “software framework” OR sdk OR “software development kit”) AND (documentation)

2038 The query was then executed on all available metadata (title, abstract and key-
 2039 words) on three primary sources to search for relevant studies in May 2019. Web of
 2040 Science¹ (WoS), Compendex/Inspec² (C/I) and Scopus³ were chosen due to their rel-
 2041 evance in SE literature (containing the IEEE, ACM, Springer and Elsevier databases)
 2042 and their ability to support advanced queries [46, 153]. A total 4,501 results⁴ were
 2043 found, with 549 being duplicates. Table 5.1 displays our results in further detail (du-
 2044 plicates not omitted); Figure 5.1 shows an exponential trend of API documentation
 2045 publications produced within the last two decades.

2046 **Manual Filtering** A follow-up manual filtering to select primary studies was per-
 2047 formed on the 4,501 results using the following inclusion criteria (IC) and exclusion
 2048 criteria (EC):

2049 **IC1** Studies must be relevant to API documentation: specifically, we
 2050 exclude studies that deal with improving the technical API usability
 2051 (e.g., improved usage patterns);

2052 **IC2** Studies must propose new knowledge or recommendations to docu-
 2053 ment APIs;

2054 **IC3** Studies must be relevant to SE as defined in SWEBOK;

2055 **EC1** Studies where full-text is not accessible through standard institutional
 2056 databases;

2057 **EC2** Studies that do not propose or extend how to improve the official,
 2058 natural language documentation of an API;

¹<http://apps.webofknowledge.com> last accessed 23 May 2019.

²<http://www.engineeringvillage.com> last accessed 23 May 2019.

³<http://www.scopus.com> last accessed 23 May 2019.

⁴Raw results can be located at <http://bit.ly/2KxBLs4>

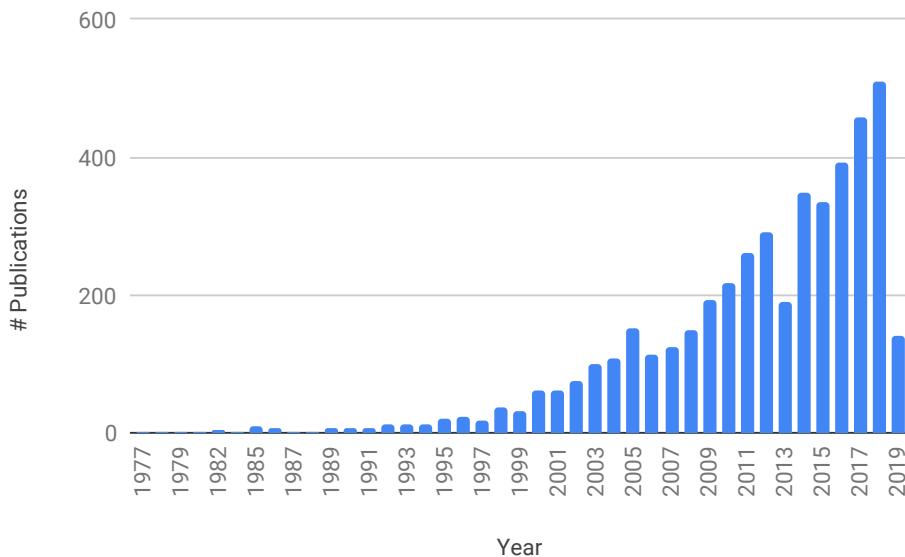


Figure 5.1: Histogram of the search results and years published. (As this search was conducted in May 2019, results taper in 2019.)

2059 **EC3** Studies proposing a third-party tool to enhance existing documentation or generate new documentation using data mining (i.e., not
 2060 proposing strategies to improve official documentation);
 2061

2062 **EC4** Studies not written in English;

2063 **EC5** Studies not peer-reviewed.

2064 After exporting metadata of search results to a spreadsheet, a three-phase curation
 2065 process was conducted. The first author read the publication source (to omit non-
 2066 SE papers quickly), author keywords and title of all 4,501 studies (514 that were
 2067 duplicates), and abstract. As we considered multiple databases, some studies were
 2068 repeated. However, the DOIs and titles were sorted and reviewed, retaining only
 2069 one copy of the paper from a single database. Moreover, as there was no limit
 2070 to the year range in our query, some studies were republished in various venues.
 2071 These, too, were handled with title similarity matching, wherein only the first paper
 2072 was considered. Where the inclusion or exclusion criteria could not be determined
 2073 from the abstract alone, the paper was automatically shortlisted. Any doubt in a
 2074 study automatically included it into the second phase. This resulted in 133 studies
 2075 being shortlisted to the second phase. We rejected 427 studies that were unrelated
 2076 to SE, 3,235 were not directly related to documenting APIs (e.g., to enhance coding
 2077 techniques that improve the overall developer usability of the API), 182 proposed new
 2078 tools to enhance API documentation or used machine learning to mine developer's
 2079 discussion of APIs, and 10 were not in English.

2080 The shortlisted studies were then re-evaluated by re-reading the abstract, the
 2081 introduction and conclusion. Performing this second phase removed a further 64

2082 studies that were on API usability or non API-related documentation (i.e., code
2083 commenting); we further refined our exclusion criteria to better match the research
2084 outcomes of this goal (chiefly including the word ‘natural language’ documentation
2085 in EC2) which removed studies focused to improve technical documentation of
2086 APIs such as data types and communication schemas. Additionally, 26 studies were
2087 removed as they were related to introducing new tools (EC3), 3 were focused on tools
2088 to mine API documentation, 7 studies where no recommendations were provided,
2089 2 further duplicate studies, and a further 10 studies where the full text was not
2090 available, not peer reviewed or in English. Books are commonly not peer-reviewed
2091 (EC5), however no books were shortlisted within these results. This resulted in 21
2092 primary studies for further analysis. The mapping of primary study identifiers to
2093 references S1–21 can be found in Appendix D.

2094 Intra-rater reliability of our 133 shortlisted papers was tested using the test-retest
2095 approach [153] by re-evaluating a random sample of 10% (13 total) of the studies
2096 shortlisted above a week after initial studies were shortlisted. Using the Cohen’s
2097 kappa coefficient as a metric for reliability, $\kappa = 0.7547$, indicating substantial
2098 agreement [165].

2099 **Data Extraction** Of the 21 primary studies, we conducted abstract key-wording
2100 adhering to Petersen et al.’s guidelines [225] to develop a classification scheme. An
2101 initial set of keywords were applied for each paper in terms of their methodologies and
2102 research approaches (RQ2), based on an existing classification schema by Wieringa
2103 and Heerkens [299]: evaluation, validation, personal experience and philosophical
2104 papers.

2105 After all primary studies had been assigned keywords, we noticed that all papers
2106 used field study techniques, and thus we consolidated these keywords using Singer
2107 et al.’s framework of SE field study techniques [262]. Singer et al. captures both
2108 study techniques *and* methods to collect data within the one framework, namely:
2109 *direct techniques*, including brainstorming and focus groups, interviews and ques-
2110 tionnaires, conceptual modelling, work diaries, think-aloud sessions, shadowing and
2111 observation, participant observation; *indirect techniques*, including instrumenting
2112 systems, fly-on-the-wall; and *independent techniques*, including analysis of work
2113 databases, tool use logs, documentation analysis, and static and dynamic analysis.

2114 Table 5.2 describes our data extraction form, which was used to collect relevant
2115 data from each paper. Figure 5.2 maps each study to one (or more, if applicable) of
2116 methodologies plotted against Wieringa and Heerkens’s research approaches.

2117 **5.3.2 Development of the Taxonomy**

2118 Usman et al. concludes that a majority of SE taxonomies are developed in an ad-hoc
2119 way [284], and proposes a systematic approach to develop taxonomies in SE that
2120 extends previous efforts by including lessons learned from more mature fields. In
2121 this subsection, we outline the 4 phases and 13 steps taken to develop our taxonomy
2122 based on Usman et al.’s technique.

Table 5.2: Data extraction form

Data item(s)	Description
Citation metadata	Title, author(s), years, publication venue, publication type
Key recommendation(s)	As per IC2, the study must propose at least one recommendation on what should be captured in API documentation
Evaluation method	Did the authors evaluate their recommendations? If so, how?
Primary technique	The primary technique used to devise the recommendation(s)
Secondary technique	As above, if a second study was conducted
Tertiary technique	As above, if a third study was conducted
Research type	The research type employed in the study as defined by Wieringa and Heerkens's taxonomy

2123 **Planning phase** The planning phase of the technique involves the following six
 2124 steps:

- 2125 (1) *defining the SE KA*: The software engineering KA, as defined by the SWEBOK,
 2126 is software construction;
- 2127 (2) *defining the objective*: The main objective of the proposed taxonomy is to
 2128 define a set of categories that enables to classify different facets of natural-
 2129 language API *documentation* knowledge (not API *usability* knowledge) as
 2130 reported in existing literature;
- 2131 (3) *defining the subject matter*: The subject matter of our proposed taxonomy is
 2132 documentation artefacts of APIs;
- 2133 (4) *defining the classification structure*: The classification structure of our pro-
 2134 posed taxonomy is *hierarchical*;
- 2135 (5) *defining the classification procedure*: The procedure used to classify the
 2136 documentation artefacts is *qualitative*;
- 2137 (6) *defining the data sources*: The basis of the taxonomy is derived from field
 2138 study techniques (see Section 5.3.1).

2139 **Identification and extraction phase** The second phase of the taxonomy devel-
 2140 opment involves (7) *extracting all terms and concepts* from relevant literature, as
 2141 selected from our 21 primary studies. These terms are then consolidated by (8) *per-*
 2142 *forming terminology control*, as some terms may refer to different concepts and
 2143 vice-versa.

2144 **Design phase** The design phase identified the core dimensions and categories
 2145 within the extracted data items. The first step is to (9) *identify and define taxonomy*
 2146 *dimensions*; for this study we utilised a bottom-up approach to identify the dimen-
 2147 sions, i.e., extracting the categories first and then nominating which dimensions

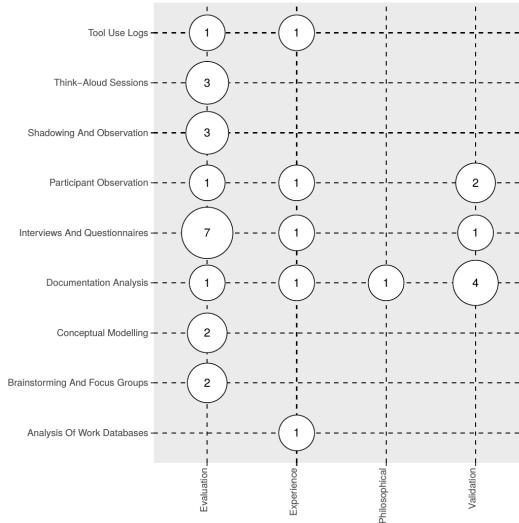


Figure 5.2: Systematic map: field study technique vs research type

2148 these categories fit into using an iterative approach. As a bottom-up approach was
 2149 utilised, step (9) also encompassed the second stage of the design phase, which is to
 2150 **(10)** *identify and describe the categories* of each dimension. Thirdly, we **(11)** *iden-*
 2151 *tify and describe relationships* between dimensions and categories, which can be
 2152 skipped if the relationships are too close together, as is the case of our grouping
 2153 technique which allows for new dimensions and categories to be added. The last
 2154 step in this phase is to **(12)** *define guidelines for using and updating the taxonomy*,
 2155 however as this taxonomy still an emerging result, guidelines to update and use the
 2156 taxonomy are anticipated future work.

2157 **Validation phase** *< todo: Revise this paragraph >* In the final phase of taxonomy
 2158 development, taxonomy designers must **(13)** *validate the taxonomy* to assess its use-
 2159 fulness. Usman et al. [284] describe three approaches to validate taxonomies: (i)
 2160 orthogonal demonstration, in which the taxonomy's orthogonality is demonstrated
 2161 against the dimensions and categories, (ii) benchmarking the taxonomy against sim-
 2162 ilar classification schemes, or (iii) utility demonstration by applying the taxonomy
 2163 heuristically against subject-matter examples. In our study, we adopt utility demon-
 2164 stration by use of a rigorous experiment and heuristic application against real-world
 2165 case-studies (i.e., within the domain of IWSs). This is is discussed in greater detail
 2166 within Section 5.4.

2167 **5.4 Taxonomy Validation**

2168 *< todo: Describe validation >* To validate our taxonomy, we conducted a two-fold
 2169 experiment. Firstly, we

2170 **5.4.1 Survey Study**

2171 **5.4.2 Empirical Application against Computer Vision Services**

2172 **5.5 Taxonomy**

2173 Our taxonomy consists of five dimensions (labelled A–E) that respectively cover:

- 2174 • **[A] Usage Description** on *how* to use the API for the developer's intended
use case;
- 2176 • **[B] Design Rationale** on *when* the developer should choose this API for a
particular use case;
- 2178 • **[C] Domain Concepts** of the domain behind the API to understand *why* this
API should be chosen for this domain;
- 2180 • **[D] Support Artefacts** that describe *what* additional documentation the API
provides; and
- 2182 • **[E] Documentation Presentation** to help organise the *visualisation* of the
above information.

2184 Further descriptions of the categories encompassing each dimension are given within
2185 Table 5.3, coded as $[Xi]$, where i is the category identifier within a dimension, X ,
2186 where $X \in \{A, B, C, D, E\}$.

2187 We expand these five dimensions into 34 categories (sub-dimensions) and Ta-
2188 ble 5.3 provides a weighting of these categories in the rightmost column as calculated
2189 as a percentage of the number of primary studies per category divided by the total
2190 of primary studies. The top five weighted categories (bolded in Table 5.3) highlight
2191 what most studies recommend documenting in API documentation, with the top
2192 three falling under the Usage Description dimension.

2193 The majority (71%) of studies advocate for **code snippets** as a necessary piece
2194 in the API documentation puzzle [A5]. While code snippets generally only reflect
2195 small portions of API functionality (limited to 15–30 LoC), this is complimented by
2196 **step-by-step tutorials** (57% of studies) that tie in multiple (disparate) components
2197 of API functionality, generally with some form of screenshots, demonstrating the
2198 development of a non-trivial application using the API step-by-step [A6]. The third
2199 highest category weighted was also under the Usage Description dimension, being
2200 **low-level reference documentation** at 52% [A2]. These three categories were the
2201 only categories to be weighted as majority categories (i.e., their weighting was above
2202 50%).

2203 The fourth and fifth highest weights are **an entry-level purpose/overview of**
2204 **the API** (48%) that gives a brief motivation as to why a developer should choose
2205 a particular API over another [B1] and **consistency in the look and feel** of the
2206 documentation throughout all of the API's official documentation (43%) [E6].

2207 **5.6 Threats to Validity**

2208 Threats to *internal validity* concern factors internal to our study that may affect
2209 results. Guidelines on producing systematic reviews [153] suggest that single re-

2210 researchers conducting their reviews should discuss the review protocol, inclusion
2211 decisions, data extraction with a third party. In this paper, we have presented the
2212 early outcomes of our systematic review, which has utilised the test-retest method-
2213 ology as a measure of reliability. MacDonell et al. [180] states that a defining
2214 characteristic of any SMS is to test the reliability of the review and extraction pro-
2215 cesses. We plan to mitigate this threat by conducting *inter*-relater reliability with
2216 the continuation of this work, using independent analysis and conflict resolution as
2217 per guidelines suggested by Garousi and Felderer [103]. Similarly, the development
2218 of our taxonomy would benefit from an inter-rater reliability categorisation of a
2219 sample of papers to both ensure that our weightings of categories are reliable and
2220 that the categories and dimensions fit the objectives of the taxonomy. Furthermore,
2221 a future user study (see Section 5.7) will be needed to assess whether the extracted
2222 information from API documentation actually impacts on developer productivity,
2223 and the usefulness of such a taxonomy should be evaluated.

2224 Threats to *external validity* represent the generalisation of the observations we
2225 have found in this study. While we have used a broad range of literature that
2226 encompasses API documentation guidelines, we acknowledge that not all papers
2227 contributing to API documentation may have been captured in the taxonomy. All
2228 efforts were made to include as many papers as possible given our filtering technique,
2229 though it is likely that some papers filtered out (e.g., papers not in English) may
2230 alter our conclusions, introducing conflicting recommendations. However, given the
2231 consistency of these trends within the studies that were sourced, we consider this a
2232 low likelihood.

2233 Threats to *construct validity* relates to the degree by which the data extrapolated
2234 in this study sufficiently measures its intended goals. Automatic searching was
2235 conducted in the SMS by choice of three popular databases (see Section 5.3.1).
2236 As a consequence of selecting multiple databases, duplicates were returned. This
2237 was mitigated by manually curating out all duplicate results from the set of studies
2238 returned. Additionally, we acknowledge that the lack manual searching of papers
2239 within particular venues may be an additional threat due to the misalignment of
2240 search query keywords to intended papers of inclusion. Thus, our conclusions are
2241 only applicable to the information we were able to extract and summarise, given the
2242 primary sources selected.

2243 5.7 Conclusions & Future Work

2244 API documentation is an aspect of quality of software, as it facilitates the developer’s
2245 productivity and assists with evolution. Improving the quality of the documentation
2246 of third party APIs improves the quality of software using them.

2247 To date, we did not find a systematic literature review that offers a consolidated
2248 taxonomy of key recommendations. Moreover, there has been little work on map-
2249 ping the research produced in this space against the techniques used to arrive at
2250 the recommendations. Starting with 4,501 papers potentially relating to API doc-
2251 umentation, we identified 21 key relevant studies, and synthesise a taxonomy of
2252 the various documentation aspects that should improve API documentation quality.

Furthermore, we also capture the most commonly used analysis techniques used in the academic literature. Figure 5.2 highlights that a majority of these studies employ interviews and questionnaires, and only some undertake structured documentation analysis.

In future revisions of this work, we intend use our results as the input to a restricted systematic literature review in API documentation artefacts. In doing so, we will consider conducting the following:

- improving reliability metrics of our study (see Section 5.6) with an inter-rater reliability method;
- reviewing the techniques and evaluation of our selected studies to extract the effectiveness of the various approaches used in the conclusions

We believe the results of this preliminary empirical work may provide further insight for future follow-up user studies with developers. Whilst our aim is to eventually improve the quality of API documentation, the ultimate goal is improving the developer's experience when producing systems and, therefore, improving the efficacy and productivity at which software is produced within industry. We hope that API designers will utilise the taxonomy produced in this paper as a weighted checklist for what should be considered in their own APIs.

Table 5.3: An overview of the 5 dimensions and categories (sub-dimensions) within our proposed taxonomy.

Key	Description	Primary Sources	Total (%)
A1	Quick-start guides to rapidly get started using the API in a specific programming language.	S4, S9, S10	3/21 (14%)
A2	Low-level reference manual documenting all API components to review fine-grade detail.	S1, S3, S4, S8, S9, S10, S11, S12, S15, S16, S17	11/21 (52%)
A3	Explanations of the API's high-level architecture to better understand intent and context.	S1, S2, S4, S11, S14, S16, S19, S20	8/21 (38%)
A4	Source code implementation and code comments (where applicable) to understand the API author's mindset.	S1, S4, S7, S12, S13, S17, S20	7/21 (33%)
A5	Code snippets (with comments) of no more than 30 LoC to understand a basic component functionality within the API.	S1, S2, S4, S5, S6, S7, S9, S10, S11, S14, S15, S16, S18, S20, S21	15/21 (71%)
A6	Step-by-step tutorials, with screenshots to understand how to build a non-trivial piece of functionality with multiple components of the API.	S1, S2, S4, S5, S7, S9, S10, S15, S16, S18, S20, S21	12/21 (57%)

Continued on next page...

An overview of the 5 dimensions and categories (sub-dimensions) within our proposed taxonomy (*Continued*).

Key	Description	Primary Sources	Total (%)
A7	Downloadable source code of production-ready applications that use the API to understand implementation in a large-scale solution.	S1, S2, S5, S9, S15	5/21 (24%)
A8	Best-practices of implementation to assist with debugging and efficient use of the API.	S1, S2, S4, S5, S7, S8, S9, S14	8/21 (38%)
A9	An exhaustive list of all major components that exist within the API.	S4, S16, S19	3/21 (14%)
A10	Minimum system requirements and dependencies to use the API.	S4, S7, S13, S17, S19	5/21 (24%)
A11	Instructions to install or begin using the API and details on its release cycle and updating it.	S4, S7, S8, S9, S11, S13, S16, S19	8/21 (38%)
A12	Error definitions that describe how to address a specific problem.	S1, S2, S4, S5, S9, S11, S13	7/21 (33%)
B1	A brief description of the purpose or overview of the API as a low barrier to entry.	S1, S2, S4, S5, S6, S8, S10, S11, S15, S16	10/21 (48%)
B2	Descriptions of the types of applications the API can develop.	S2, S4, S9, S11, S15, S18	6/21 (29%)
B3	Descriptions of the types of users who should use the API.	S4, S9	2/21 (10%)
B4	Descriptions of the types of users who will use the product the API creates.	S4	1/21 (5%)
B5	Success stories about the API used in production.	S4	1/21 (5%)
B6	Documentation to compare similar APIs within the context to this API.	S2, S6, S13, S18	4/21 (19%)
B7	Limitations on what the API can and cannot provide.	S4, S5, S8, S9, S14, S16	6/21 (29%)
C1	Descriptions of the relationship between API components and domain concepts.	S3, S10	2/21 (10%)
C2	Definitions of domain-terminology and concepts, with synonyms if applicable.	S2, S3, S4, S6, S7, S10, S14, S16	8/21 (38%)
C3	Generalised documentation for non-technical audiences regarding the API and its domain.	S4, S8, S16	3/21 (14%)
D1	A list of FAQs.	S4, S7	2/21 (10%)
D2	Troubleshooting suggestions.	S4, S8	2/21 (10%)
D3	Diagrammatically representing API components using visual architectural representations.	S6, S13, S20	3/21 (14%)
D4	Contact information for technical support.	S4, S8, S19	3/21 (14%)
D5	A printed/printable resource for assistance.	S4, S6, S7, S9, S16	5/21 (24%)

Continued on next page...

An overview of the 5 dimensions and categories (sub-dimensions) within our proposed taxonomy (*Continued*).

Key	Description	Primary Sources	Total (%)
D6	Licensing information.	S7	1/21 (5%)
E1	Searchable knowledge base.	S3, S4, S6, S10, S14, S17, S18	7/21 (33%)
E2	Context-specific discussion forum.	S4, S10, S11	3/21 (14%)
E3	Quick-links to other relevant documentation frequently viewed by developers.	S6, S16, S20	3/21 (14%)
E4	Structured navigational style (e.g., breadcrumbs).	S6, S10, S20	3/21 (14%)
E5	Visualised map of navigational paths to certain API components in the website.	S6, S14, S20	3/21 (14%)
E6	Consistent look and feel of documentation.	S1, S2, S3, S5, S6, S8, S10, S15, S20	9/21 (43%)

CHAPTER 6

2271

2272

Interpreting Pain-Points in Computer Vision Services[†]

2274

2275 **Abstract** Intelligent services are becoming increasingly more pervasive; application de-
2276 velopers want to leverage the latest advances in areas such as computer vision to provide new
2277 services and products to users, and large technology firms enable this via RESTful APIs.
2278 While such APIs promise an easy-to-integrate on-demand machine intelligence, their cur-
2279 rent design, documentation and developer interface hides much of the underlying machine
2280 learning techniques that power them. Such APIs look and feel like conventional APIs but
2281 abstract away data-driven probabilistic behaviour—the implications of a developer treating
2282 these APIs in the same way as other, traditional cloud services, such as cloud storage, is
2283 of concern. The objective of this study is to determine the various pain-points developers
2284 face when implementing systems that rely on the most mature of these intelligent services,
2285 specifically those that provide computer vision. We use Stack Overflow to mine indications
2286 of the frustrations that developers appear to face when using computer vision services, clas-
2287 sifying their questions against two recent classification taxonomies (documentation-related
2288 and general questions). We find that, unlike mature fields like mobile development, there
2289 is a contrast in the types of questions asked by developers. These indicate a shallow un-
2290 derstanding of the underlying technology that empower such systems. We discuss several
2291 implications of these findings via the lens of learning taxonomies to suggest how the software
2292 engineering community can improve these services and comment on the nature by which
2293 developers use them.

6.1 Introduction

2295 The availability of recent advances in artificial intelligence (AI) over simple REST-
2296 ful end-points offers application developers new opportunities. These new intel-

[†]This chapter is originally based on A. Cummaudo, R. Vasa, S. Barnett, J. Grundy, and M. Abd elrazek, "Interpreting Cloud Computer Vision Pain-Points: A Mining Study of Stack Overflow," in *Proceedings of the 42nd International Conference on Software Engineering*. Seoul, Republic of Korea: IEEE, May 2020. In Press. Terminology has been updated to fit this thesis.

2297 ligent services (AI components) abstract complex machine learning (ML) and AI
2298 techniques behind simpler API calls. In particular, they hide (either explicitly or
2299 implicitly) any data-driven and non-deterministic properties inherent to the process
2300 of their construction. The promise is that software engineers can incorporate com-
2301 plex machine learnt capabilities, such as computer vision, by simply calling an API
2302 end-point.

2303 The expectation is that application developers can use these AI-powered services
2304 like they use other conventional software components and cloud services (e.g., object
2305 storage like AWS S3). Furthermore, the documentation of these AI components is
2306 still anchored to the traditional approach of briefly explaining the end-points with
2307 some information about the expected inputs and responses. The presupposition
2308 is that developers can reason and work with this high level information. These
2309 services are also marketed to suggest that application developers do not need to fully
2310 understand how these components were created (i.e., assumptions in training data
2311 and training algorithms), the ways in which the components can fail, and when such
2312 components should and should not be used.

2313 The nuances of ML and AI powering intelligent services have to be appre-
2314 ciated, as there are real-world consequences to software quality for applications
2315 that depend on them if they are ignored [69]. This is especially true when ML
2316 and AI are abstracted and masked behind a conventional-looking API call, yet the
2317 mechanisms behind the API are data-dependent, probabilistic and potentially non-
2318 deterministic [212]. We are yet to discover what long-term impacts exist during
2319 development and production due to poor documentation that do not capture these
2320 traits, nor do we know the depth of understanding application developers have for
2321 these components. Given the way AI-powered services are currently presented, de-
2322 velopers are also likely to reason about these new services much like a string library
2323 or a cloud data storage service. That is, they may not fully consider the implica-
2324 tions of the underlying statistical nature of these new abstractions or the consequent
2325 impacts on productivity and quality.

2326 Typically, when developers are unable to correctly align to the mindset of the
2327 API designer, they attempt to resolve issues by (re-)reading the API documentation.
2328 If they are still unable to resolve these issues on their own after some internet
2329 searching, they consider online discussion platforms (e.g., Stack Overflow, GitHub
2330 Issues, Mailing Lists) where they seek technological advice from their peers [3].
2331 Capturing what developers discuss on these platforms offers an insight into the
2332 frustrations developers face when using different software components as shown
2333 by recent works [30, 150, 245, 267, 292]. However, to our knowledge, no studies
2334 have yet analysed what developers struggle with when using the new generation of
2335 *intelligent* services. Given the re-emergent interest in AI and the anticipated value
2336 from this technology [176], a better understanding of issues faced by developers
2337 will help us improve the quality of services. Our hypothesis is that application
2338 developers do not fully appreciate the probabilistic nature of these services, nor do
2339 they have sufficient appreciation of necessary background knowledge—however, we
2340 do not know the specific areas of concern. The motivation for our study is to inform
2341 API designers on which aspects to focus in their documentation, education, and

2342 potentially refine the design of the end-points.

2343 This study involves an investigation of 1,825 Stack Overflow (SO) posts regarding
2344 one of the most mature types of intelligent services—computer vision services—
2345 dating from November 2012 to June 2019. We adapt existing methodologies of prior
2346 SO analyses [30, 276] to extract posts related to computer vision services. We then
2347 apply two existing SO question classification schemes presented at ICPC and ICSE
2348 in 2018 and 2019 [3, 31]. These previous studies focused on mobile apps and web
2349 applications. Although not a direct motivation, our work also serves as a validation
2350 of the applicability of these two issue classification taxonomies [3, 31] in the context
2351 of intelligent services (hence potential for generalisation). Additionally our work is
2352 the first—to our knowledge—to *test* the applicability of these taxonomies in a new
2353 study.

2354 The taxonomies in previous works focus on the specific aspects from the domain
2355 (e.g. API usage, specificity within the documentation etc.) and as such do not
2356 deeply consider the learning gap of an application developer. To explore the API
2357 learning implications raised by our SO analysis, we applied an additional lens of
2358 two taxonomies from the field of pedagogy. This was motivated by the need to offer
2359 an insight into the work needed to help developers learn how to use these relatively
2360 new services.

2361 The key findings of our study are:

- 2362 • The primary areas that developers raise as issues reflect a relatively primitive
2363 understanding of the underlying concepts of data-driven ML approaches used.
2364 We note this via the issues raised due to conceptual misunderstanding and
2365 confusion in interpreting errors,
- 2366 • Developers predominantly encounter a different distribution of issue types than
2367 were reported in previous studies, indicating the complexity of the technical
2368 domain has a non-trivial influence on intelligent API usage; and
- 2369 • Most of these issues can be resolved with better documentation, based on our
2370 analysis.

2371 The paper also offers a data-set as an additional contribution to the research
2372 community and to permit replication [322]. The paper structure is as follows:
2373 Section 6.2 provides motivational examples to highlight the core focus of our study;
2374 Section 6.3 provides a background on prior studies that have mined SO to gather
2375 insight into the SE community; Section 6.4 describes our study design in detail;
2376 Section 6.5 presents the findings from the SO extraction; Section 6.6 offers an
2377 interpretation of the results in addition to potential implications that arise from our
2378 work; Section 6.7 outlines the limitations of our study; concluding remarks are given
2379 in Section 6.8.

2380 **6.2 Motivation**

2381 “Intelligent” services are often available as a cloud end-point and provide developers
2382 a friendly approach to access recent AI/ML advances without being experts in the
2383 underlying processes. Figure 6.1 highlights how these services abstract away much

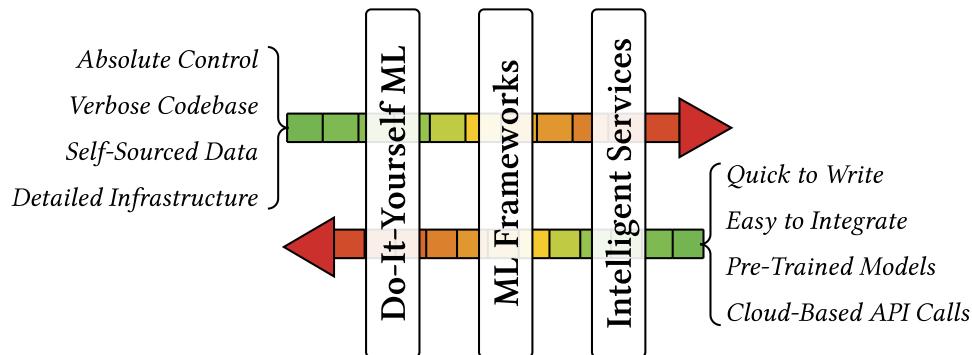


Figure 6.1: Some traits of Intelligent Services vs. ‘Do-It-Yourself’ ML. Green-to-red arrows indicate the presence of these traits. *Adapted from Ortiz [215]*.

of the technical know-how needed to create and operationalise these intelligent services [215]. In particular, they hide information about the training algorithm and data-sets used in training, the evaluation procedures, the optimisations undertaken, and—surprisingly—they often do not offer a properly versioned end-point [69, 212]. That is, the cloud vendors may change the behaviour of the services without sufficient transparency.

The trade-off towards ease of use for application developers, coupled with the current state of documentation (and assumed developer background) has a cost as reflected in the increasing discussions on developer communities such as SO (see Figure 6.2). To illustrate the key concerns, we list below a few up-voted questions:

- **unsure of ML specific vocabulary:** “*Though it’s now not so clear to me what ‘score’ actually means.*” [349]; “*I’m trying out the [intelligent service], and there’s a score field that returns that I’m not sure how to interpret [it].*” [363]
- **frustrated about non-deterministic results:** “*Often the API has troubles in recognizing single digits... At other times Vision confuses digits with letters.*” [362]; “*Is there a way to help the program recognize numbers better, for example limit the results to a specific format, or to numbers only?*” [359]
- **unaware of the limitations behind the services:** “*Is there any API available where we can recognize human other body parts (Chest, hand, legs and other parts of the body), because as per the Google vision API it’s only able to detect face of the human not other parts.*” [343]
- **seeking further documentation:** “*Does anybody know if Google has published their full list of labels ([‘produce’, ‘meal’, ...]) and where I could find that? Are those labels structured in any way? - e.g. is it known that ‘food’ is a superset of ‘produce’, for example.*” [346]

The objective of our study is to better understand the nature of the questions that developers raise when using intelligent services, in order to inform the service designers and documenters. In particular, the knowledge we identify can be used to improve the documentation, educational material and (potentially) the information contained in the services’ response objects—these are the main avenues developers

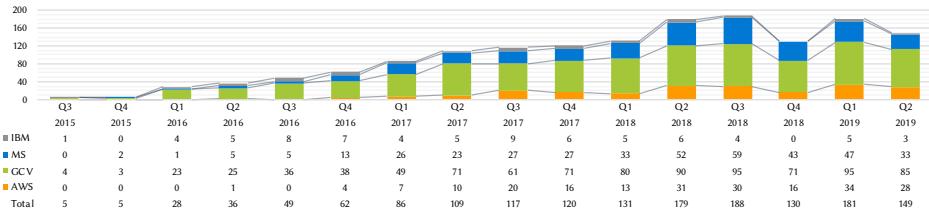


Figure 6.2: Trend of posts, where IBM = IBM Watson Visual Recognition, MS = Azure Computer Vision, AWS = AWS Rekognition and GCV = Google Cloud Vision. Three MS posts from Q4 2012, Q3 2013 and Q4 2013 have been removed for graph clarity.

have to learn and reason about when using these services. There is previous work that has investigated issues raised by developers [3, 31, 276]. We build on top of this work by adapting the study methodology and apply the taxonomies offered to identify the nature of the issues and this results in the following research questions in this paper:

RQ1. RQ1. How do developers mis-comprehend intelligent services as presented within Stack Overflow pain-points? While the AI community is well aware in the the nuances that empower intelligent services, such services are being released for application developers who may not be aware of their limitations or how they work. This is especially the case when machine intelligence is accessed via web-based APIs where such details are not fully exposed.

RQ2. Are the distribution of issues similar to prior studies? We compare how the distributions of previous studies' of posts about conventional, deterministic API services differ from those of intelligent services. By assessing the distribution of intelligent services' issues against similar studies that focus on mobile and web development, we identify whether a new taxonomy is needed specific to AI-based services, and if gaps specific to AI knowledge exist that need to be captured in these taxonomies.

6.3 Background

The primary goal of analysing issues is to better understand the root causes. Hence, a good issue classification taxonomy should ideally capture the underlying causal aspects (instead of pure functional groupings) [58]. Although this idea (of cause related classification) is not new (Chillarege advocated for it in this TSE paper in 1992), this is not a universally followed approach when studying online discussions and some recent works have largely classified issues into the “*what is*” and not “*how to fix it*” [21, 30, 282]. They typically (manually) classify discussion into either *functional areas* (e.g., Website Design/CSS, Mobile App Development, .NET Framework, Java [21]) or *descriptive areas* (e.g., Coding Style/Practice, Problem/-Solution, Design, QA [21, 282]). As a result, many of these studies do not give us a prioritised means of targeted attack on how to *resolve* these issues with, for

example, improved documentation. Interestingly, recent taxonomies that studied SO data (Aghajani et al. [3] and Beyer et al. [31]) were causal in nature and developed to understand discussions related to mobile and web applications. However, issues that arise when developers use intelligent services have not been studied, nor do we know if existing issue classification taxonomies are sufficient in this domain.

Researchers studying APIs have also attempted to understand developer's opinions towards APIs [282], categorise the questions they ask about these APIs [21, 22, 31, 245], and understand API related documentation and usage issues [3, 4, 7, 21, 128, 276]. These studies often employ automation to assist in the data analysis stages of their research. Latent Dirichlet Allocation [7, 21, 245, 282] is applied for topic modelling and other ML techniques such as Random Forests [31], Conditional Random Fields [4] or Support Vector Machines [31, 128] are also used.

However, automatic techniques are tuned to classify into *descriptive* categories, that is, they help paint a landscape of *what is*, but generally do not address the causal factors to address the issues in great detail. For example, functional areas such as 'Website Design' [21], 'User Interface' [30] or 'Design' [283] result from such analyses. These automatic approaches are generally non-causal, making it hard to address reasons for *why* developers are asking such questions. However, not all studies in the space use automatic techniques; other studies employ manual thematic analysis [3, 22, 276] (e.g., card sorting) or a combination of both [30, 31, 245, 281]. Our work uses a manual approach for classification, and we use taxonomies that are more causally aligned allowing our findings to be directly useful in terms of addressing the issues.

Evidence-based SE [155] has helped shape the last 15 years worth of research, but the reliability of such evidence has been questioned [144, 146, 259]. Replication studies, especially in empirical works, can give us the confidence that existing results are adaptable to new domains; in this context, we extend (to intelligent services) and work with study methods developed in previous works.

6.4 Method

6.4.1 Data Extraction

This study initially attempted to capture SO posts on a broad range of many intelligent services by identifying issues related to four popular intelligent service cloud providers: Google Cloud [327], AWS [313], Azure [335] and IBM Cloud [331]. We based our selection criteria on the prominence of the providers in industry (Google, Amazon, Microsoft, IBM) and their ubiquity in cloud platform services. Additionally, in 2018, these services were considered the most adopted cloud vendors for enterprise applications [240].

However, during the filtering stage (see Section 6.4.2), we decided to focus on a subset of these services, computer vision, as these are one of the more mature and stable ML/AI-based services with widespread and increasing adoption in the developer community (see Figure 6.2). We acknowledge other services beyond the four analysed provide similar capabilities [316, 317, 324, 330, 372, 373] and only

2487 English-speaking services have been selected, excluding popular services from Asia
2488 (e.g., [314, 315, 329, 338, 339])—see Section 6.7. For comprehensiveness, we
2489 explain below our initial attempts to extract *all* intelligent services.

2490 **Defining a list of intelligent services** As there exists no global ‘list’ of intelligent
2491 services to search on, we needed to derive a *corpus of initial terms* to allow us
2492 to know *what* to search for on the Stack Exchange Data Explorer¹ (SEDE). We
2493 began by looking at different brand names of cloud services and their permutations
2494 (e.g., Google Cloud Services and GCS) as well as various ML-related products
2495 (e.g., Google Cloud ML). To do this, we performed extensive Google searches² in
2496 addition to manually reviewing six ‘overview’ pages of the relevant cloud platforms.
2497 We identified 91 initial intelligent services to incorporate into our search terms³.

2498 **Manual search for relevant, related terms** We then ran a manual search² on
2499 each term to determine if these terms were relevant. We did this by querying each
2500 term within SO’s search feature, reviewing the titles and body post previews of
2501 the first three pages of results (we did not review the answers, only the questions).
2502 We also noted down the user-defined *Tags* of each post (up to five per question);
2503 by clicking into each tag, we could review similar tags (e.g., ‘project-oxford’ for
2504 ‘azure-cognitive-services’) and check if the tag had synonyms (e.g., ‘aws-lex’ and
2505 ‘amazon-lex’). We then compiled a *corpus of tags* consisting of 31 terms.

2506 **Developing a search query** We recognise that searching SEDE via *Tags* exclu-
2507 sively can be ineffective (see [21, 276]). To mitigate this, we produced a *corpus of*
2508 *title and body terms*. Such terms are those that exist within the title and body of
2509 the posts to reflect the ways in which individual developers commonly use to refer
2510 to different intelligent services. To derive at such a list, we performed a search^{2,3}
2511 of the 31 tags above in SEDE, filtering out posts that were not answers (i.e., ques-
2512 tions only) as we wanted to see how developers *phrase* their questions. For each
2513 search, we extracted a random sample of 100 questions (400 total for each service)
2514 and reviewed each question. We noted many patterns in the permutations of how
2515 developers refer to these services, such as: common misspellings (‘bind’ vs. ‘bing’);
2516 brand misunderstanding (‘Microsoft computer vision’ vs. ‘Azure computer vision’);
2517 hyphenation (‘Auto-ML’ vs. ‘Auto ML’); UK and US English (‘Watson Analyser’
2518 vs. ‘Watson Analyzer’); and, the use of apostrophes, plurals, and abbreviations
2519 (‘Microsoft’s Computer Vision API’, ‘Microsoft Computer Vision Services’, ‘GCV’
2520 vs. ‘Google Cloud Vision’). We arrived at a final list of 229 terms compromising
2521 all of the intelligent services provided by Google, Amazon, Microsoft and IBM as
2522 of January 2019³.

2523 **Executing our search query** Our next step was to perform a case-insensitive
2524 search of all 229 terms within the body or title of posts. We used Google BigQuery’s

¹<http://data.stackexchange.com/stackoverflow>

²This search was conducted on 17 January 2019

³For reproducibility, this is available at <http://bit.ly/2ZcwNJO>.

2525 public data-set of SO posts⁴ to overcome SEDE’s 50,000 row limit and to conduct
 2526 a case-insensitive search. This search was conducted on 10 May 2019, where we
 2527 extracted 21,226 results. We then performed several filtering steps to cleanse our
 2528 extracted data, as explained below.

2529 6.4.2 Data Filtering

2530 **Refining our inclusion/exclusion criteria** We performed an initial manual filter-
 2531 ing of the 50 most recent posts (sorted by descending *CreationDate* values) of the
 2532 21,226 posts above, assessing the suitability of the results and to help further refine
 2533 our inclusion and exclusion criteria. We did note that some abbreviations used in the
 2534 search terms (e.g., ‘GCV’, ‘WCS’⁵), resulting in irrelevant questions in our result
 2535 set. We therefore removed abbreviations from our search query and consolidated all
 2536 overlapping terms (e.g., ‘Google Vision **API**’ was collapsed into ‘Google Vision’).

2537 We also recognised that 21,226 results would be non-trivial to analyse without
 2538 automated techniques. As we wanted to do manual qualitative analysis, we reduced
 2539 our search space to 27 search terms of just the *computer vision services* within
 2540 the original corpus of 229 terms. These were Google Cloud Vision [327], AWS
 2541 Rekognition [313], Azure Computer Vision [335], and IBM Watson Visual Recog-
 2542 nition [331]. This resulted in 1,425 results that were extracted on 21 June 2019. The
 2543 query used and raw results are available online in our supplementary materials [322].

2544 **Duplicates** Within 1,425 results, no duplicate questions were noted, as determined
 2545 by unique post ID, title or timestamp.

2546 **Automated and manual filtering** To assess the suitability and nature of the 1,425
 2547 questions extracted, the first author began with a manual check on a randomised
 2548 sample of 50 questions. As the questions were exported in a raw CSV format (with
 2549 HTML tags included in the post’s body), we parsed the questions through an ERB
 2550 templating engine script⁶ in which the ID, title, body, tags, created date, and view,
 2551 answer and comment counts were rendered for each post in an easily-readable format.
 2552 Additionally, SQL matches in the extraction process were also highlighted in yellow
 2553 (i.e., in the body of the post) and listed at the top of each post. These visual cues
 2554 helped to identify 3 false positive matches where library imports or stack traces
 2555 included terms within our corpus of 26 computer vision service terms. For example,
 2556 `aws-java-sdk-rekognition:jar` is falsely matched as a dependency within an
 2557 unrelated question. As such exact matches would be hard to remove without the use
 2558 of regular expressions, and due to the low likelihood (6%) of their appearance, we
 2559 did not perform any followup automatic filtering.

2560 **Classification** Our 1,425 posts were then split into 4 additional random samples
 2561 (in addition to the random sample of 50 above). 475 posts were classified by the first

⁴<http://bit.ly/2LrN7OA>

⁵Watson Cognitive Services

⁶We make this available for future use at: <http://bit.ly/2NqBB70>

2562 author and three other research assistants, software engineers with at least 2 years
2563 industry experience, assisted to classify the remaining 900. This left a total of 1,375
2564 classifications made by four people plus an additional 450 classifications made from
2565 reliability analysis, in which the remaining 50 posts were classified nine times (as
2566 detailed in Section 6.4.3). Thus, a total of 1,825 classifications were made from the
2567 original 1,425 posts extracted.

2568 Whilst we could have chosen to employ topic modelling, these are too descriptive
2569 in nature (as discussed in Section 6.3). Moreover, we wanted to see if prior
2570 taxonomies can be applied to intelligent services (as opposed to creating a new
2571 one) and compare if their distributions are similar. Therefore, we applied the two
2572 existing taxonomies described in Section 6.3 to each post; (i) a documentation-
2573 specific taxonomy that addresses issues directly resulting from documentation, and
2574 (ii) a generalised taxonomy that covers a broad range of SO issues in a well-defined
2575 SE area (specifically mobile app development). Aghajani et al.’s documentation-
2576 specific taxonomy (Taxonomy A) is multi-layered consisting of four dimensions and
2577 16 sub-categories [3]. Similarly, Beyer’s SO generalised post classification taxon-
2578 omy (Taxonomy B) consists of seven dimensions [31]. We code each dimension
2579 with a number, X , and each sub-category with a letter y : (Xy). We describe both
2580 taxonomies in detail within Table 6.1. Where a post was included in our results
2581 but not applicable to intelligent services (see Section 6.4.2) or not applicable to
2582 a taxonomy dimension/category, then the post was flagged for removal in further
2583 analysis. Table 6.1 presents *our understanding* of the respective taxonomies; our
2584 intent is not to methodologically replicate Aghajani et al. or Beyer et al.’s studies in
2585 the intelligent service domain, rather to acknowledge related work in the area of SO
2586 classification and reduce the need to synthesise a new taxonomy. We baseline all
2587 coding against *our interpretation only*. Our classifications are therefore independent
2588 of the previous authors’ findings.

Table 6.1: Descriptions of dimensions (■) and sub-categories (↔) from both taxonomies used.

A Documentation-specific classification (Aghajani et al. [3])	
A-1	■ Information Content (What)
A-1a	↔ <i>Correctness</i>
A-1b	↔ <i>Completeness</i>
A-1c	↔ <i>Up-to-dateness</i>
A-2	■ Information Content (How)
A-2a	↔ <i>Maintainability</i>
A-2b	↔ <i>Readability</i>
A-2c	↔ <i>Usability</i>
A-2d	↔ <i>Usefulness</i>
A-3	■ Process-Related
A-3a	↔ <i>Internationalisation</i>
A-3b	↔ <i>Contribution-Related</i>
A-3c	↔ <i>Configuration-Related</i>
A-3d	↔ <i>Implementation-Related</i>
A-3e	↔ <i>Traceability</i>
A-4	■ Tool-Related
A-4a	↔ <i>Tooling Bugs</i>
A-3b	↔ <i>Tooling Discrepancy</i>
A-3c	↔ <i>Tooling Help Required</i>
A-3d	↔ <i>Tooling Migration</i>
B Generalised classification (Beyer et al. [31])	
B-1	■ API usage
B-2	■ Discrepancy
B-3	■ Errors
B-4	■ Review
B-5	■ Conceptual
B-6	■ API change
B-7	■ Learning

2589 6.4.3 Data Analysis

2590 **Reliability of Classification** To measure consistency of the categories assigned
2591 by each rater to each post, we utilised both intra- and inter-rater reliability [189].
2592 As verbatim descriptions from dimensions and sub-categories were considered quite
2593 lengthy from their original sources, all raters met to agree on a shared interpretation of
2594 the descriptions, which were then paraphrased as discussed in the previous subsection
2595 and tabulated in Table 6.1. To perform statistical calculations of reliability, each
2596 category was assigned a nominal value and a random sample of 50 posts were
2597 extracted. Two-phase reliability analysis followed.

2598 Firstly, intra-rater agreement by the first author was conducted twice on 28 June
2599 2019 and 9 August 2019. Secondly, inter-rater agreement was conducted with the
2600 remaining four co-authors in addition to three research assistants within our research
2601 group in mid-August 2019. Thus, the 50 posts were classified an additional nine
2602 times, resulting in 450 classifications for reliability analysis. We include these
2603 classifications in our overall analysis.

2604 At first, we followed methods of reliability analysis similar to previous SO
2605 studies (e.g., [276]) using the percentage agreement metric that divides the number
2606 of agreed categories assigned per post by the total number of raters [189]. However,
2607 percentage agreement is generally rejected as an inadequate measure of reliability
2608 analysis [62, 117, 160] in statistical communities. As we used more than 2 coders
2609 and our reliability analysis was conducted under the same random sample of 50
2610 posts, we applied *Light's Kappa* [169] to our ratings, which indicates an overall
2611 index of agreement. This was done using the `irr` computational R package [101]
2612 as suggested in [117].

2613 **Distribution Analysis** In order to compare the distribution of categories from our
2614 study with previous studies we carried out a χ^2 test. We selected a χ^2 test as the
2615 following assumptions [260] are satisfied: (i) the data is categorical, (ii) all counts are
2616 greater than 5, and (iii) we can assume simple random sampling. The null hypothesis
2617 describes the case where each population has the same proportion of observations
2618 and the alternative hypothesis is where at least one of the null hypothesis statements
2619 is false. We chose a significance value, α , of 0.05 following a standard rule of
2620 thumb. As to the best of our knowledge this is the first statistical comparison using
2621 Taxonomy A and B on SO posts. To report the effect size we selected Cramer's Phi,
2622 ϕ_c which is well suited for use on nominal data [260].

2623 6.5 Findings

2624 We present our findings from classifying a total of 1,825 SO posts aimed at answering
2625 RQs 1 and 2. 450 posts were classified using Taxonomies A and B for reliability
2626 analysis as described in Section 6.4.3 and the remaining 1,375 posts were classified
2627 as per Section 6.4.2. A summary of our classification using Taxonomies A and B is
2628 shown in Figure 6.3.

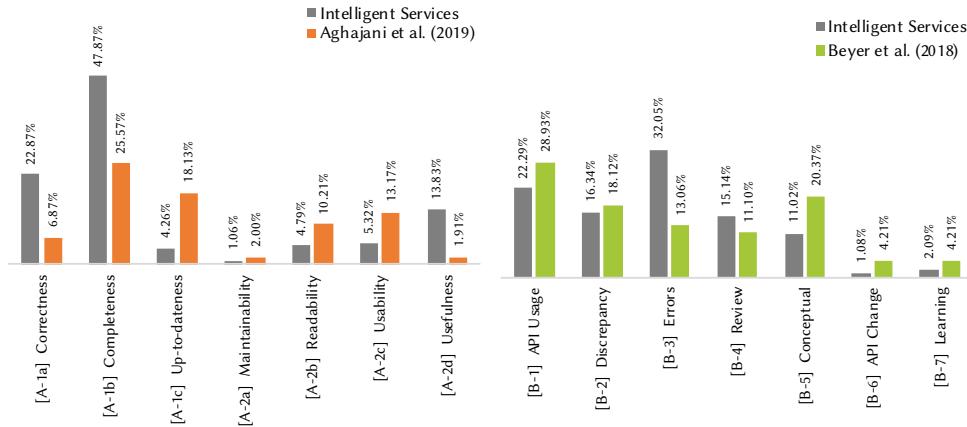


Figure 6.3: *Left:* Documentation-specific classification taxonomy results highlights a mostly similar distribution to that of Aghajani et al.’s findings [3]. *Right:* Generalised classification taxonomy results highlight differences from more mature fields (i.e., Android APIs in Beyer et al. [31]) to less mature fields (i.e., intelligent services).

2629 6.5.1 Post classification and reliability analysis

2630 When undertaking the classification, we found that 238 issues (13.04%) did not
 2631 relate to intelligent services directly. For example, library dependencies were still
 2632 included in a number of results (see Section 6.4.2), and we found there to be many
 2633 posts discussing Android’s Mobile Vision API as Google (Cloud) Vision. These
 2634 issues were flagged and ignored for further analysis (see Section 6.4.2).

2635 For our reliability analysis, we classified a total of 450 posts of which 70 posts
 2636 were flagged as irrelevant. Landis and Koch [165] provide guidelines to interpret
 2637 kappa reliability statistics, where $0.00 \leq \kappa \leq 0.20$ indicates *slight* agreement and
 2638 $0.21 \leq \kappa \leq 0.40$ indicates *fair* agreement. Despite all raters meeting to agree on a
 2639 shared interpretation of the taxonomies (see Section 6.4.3) our inter-rater measures
 2640 aligned *slightly* (0.148) for Taxonomy A and *fairly* (0.295) for Taxonomy B. We
 2641 report further in Section 6.7.

2642 6.5.2 Developer Frustrations

2643 We found Beyer et al.’s high-level abstraction taxonomy (Taxonomy B) was able to
 2644 classify 86.52% of posts. 10.30% posts were assigned exclusively under Aghajani
 2645 et al.’s documentation-specific taxonomy (Taxonomy A). We found that developers
 2646 do not generally ask questions exclusive to documentation, and typically either
 2647 pair documentation-related issues to their own code or context. The following two
 2648 subsections further explain results from both Taxonomy A and B’s perspective.

2649 **Results from Aghajani et al.’s taxonomy** Results for Aghajani et al.’s low-level
 2650 documentation taxonomy (Taxonomy A), indicates that most discussion on SO does
 2651 not directly relate to documentation about an intelligent service. We did not find
 2652 any process-related (A-3) or tool-related (A-4) questions as, understandably, the

2653 developers who write the documentation of the intelligent services would not be
2654 posting questions of such nature on SO. One can *infer* documentation-related issues
2655 from posts (i.e., parts of the documentation *lacking* that may cause the issue posted).
2656 However, there are few questions that *directly* relate to documentation of intelligent
2657 services.

2658 Few developers question or ask questions directly about the API documentation,
2659 but some (47.87%) posts ask for additional information to understand the
2660 API (**A-1b**), for example: “*Is there a full list of potential labels*
2661 *that Google’s Vision API will return?*” [346]; “*There seems to be very little to no*
2662 *documentation for AWS iOS text recognition inside an image?*” [344].

2663 22.87% of posts question the **accuracy (A-1a)** of certain parts of the cloud docu-
2664 mentation, especially in relation to incorrect quotas and limitations: “*Are the Cloud*
2665 *Vision API limits in documentation correct?*” [357], “*According to the Google Vision*
2666 *documentation, the maximum number of image files per request is 16. Elsewhere,*
2667 *however, I’m finding that the maximum number of requests per minute is as high as*
2668 *1800.*” [342].

2669 There are also many references (23.94%) addressing the confusing nature of
2670 some documentation, indicating that the **readability, usability and usefulness of**
2671 **the documentation (A-2b, A-2c and A-2d)** could be improved. For example, “*Am*
2672 *I encoding it correctly? The docs are quite vague.*” [340], “*The aws docs for this*
2673 *are really confusing.*” [369].

2674 **Results from Beyer et al.’s taxonomy** We found that a majority (32.05%) of
2675 posts are primarily **error-related questions (B-3)**, including a dump of the stack
2676 trace or exception message from the service’s programming-language SDK (usually
2677 Java, Python or C#) that relates to a specific error. For example: “*I can’t fix an*
2678 *error that’s causing us to fall behind.*” [366]; “*I’m using the Java Google Vision*
2679 *API to run through a batch of images... I’m now getting a channel closed and*
2680 *ClosedChannelException error on the request.*” [360].

2681 **API usage questions (B-1)** were the second highest category at 22.29% of
2682 posts. Reading the questions revealed that many developers present an insufficient
2683 understanding of the behaviour, functional capability and limitation of these services
2684 and the need for further data processing. For example, while Azure provides an image
2685 captioning service, this is not universal to all computer vision services: “*In Amazon*
2686 *Rekognition for image processing how do I get the caption for an image?*” [351].
2687 Similarly, OCR-related and label-related questions often indicate interest in cross-
2688 language translation, where a separate translation service would be required: “*Can*
2689 *Google Cloud Vision generate labels in Spanish via its API?*” [365]; “[*How can*
2690 *I] specify language for response in Google Cloud Vision API*” [352]; “*When I*
2691 *request a text detection of an image, it gives only English Alphabet characters*
2692 *(characters without accents) which is not enough for me. How can I get the UTF-32*
2693 *characters?*” [347].

2694 It was commonplace to see questions that demonstrate a lack of depth in under-
2695 standing and appreciating how these services work, instead posting simple debugging
2696 questions. For instance, in the 11.02% of **conceptual-related questions (B-5)** that

we categorised, we noticed causal links to a misunderstanding (or lack of awareness) of the vocabulary used within computer vision. For example: “*The problem is that I need to know not only what is on the image but also the position of that object. Some of those APIs have such feature but only for face detection.*” [358]; “*I want to know if the new image has a face similar to the original image.... [the service] can identify faces, but can I use it to get similar faces to the identified face in other images?*” [350]. It is evident that some application developers are not aware of conceptual differences in computer vision such as object/face *detection* versus *localisation* versus .

In the 16.34% of **discrepancy-related questions (B-2)**, we see further unawareness from developers in how the underlying systems work. In OCR-related questions, developers do not understand the pre-processing steps required before an OCR is performed. In instances where text is separated into multiple columns, for example, text is read top-down rather than left-to-right and segmentation would be required to achieve the expected results. For example, “*it appears that the API is using some kind of logic that makes it scan top to bottom on the left side and moving to right side and doing a top to bottom scan.*” [364]; “*this method returns scanned text in wrong sequence... please tell me how to get text in proper sequence.*” [370].

A number of **review-related questions (B-4)** (15.14%) seem to provide some further depth in understanding the context to which these systems work, where training data (or training stages) are needed to understand how inferences are made: “*How can we find an exhaustive list (or graph) of all logos which are effectively recognized using Google Vision logo detection feature?*” [368]; “*when object banana is detected with accuracy greater than certain value, then next action will be dispatched... how can I confidently define and validate the threshold value for each item?*” [354].

API change (B-6) was shown in 1.08% of posts, with evolution of the services occurring (e.g., due to new training data) but not necessarily documented “*Recently something about the Google Vision API changed... Suddenly, the API started to respond differently to my requests. I sent the same picture to the API today, and I got a different response (from the past).*” [367].

6.5.3 Statistical Distribution Analysis

We obtained the following results $\chi^2 = 131.86$, $\alpha = 0.05$, $p \text{ value} = 2.2 \times 10^{-16}$ and $\phi_c = 0.362$ from our distribution analysis with Taxonomy A to compare our study with that of Aghajani et al. [3]. Comparing our study to Beyer et al. [31] produced the following results $\chi^2 = 145.58$, $\alpha = 0.05$, $p \text{ value} = 2.2 \times 10^{-16}$ and $\phi_c = 0.252$. These results show that we are able to reject the null hypothesis that the distribution of posts using each taxonomy was the same as the comparison study. While there are limited guidelines for interpreting ϕ_c when there is no prior information for effect size [271], Sun et al. suggests the following: $0.07 \leq \phi_c \leq 0.20$ indicates a *small* effect, $0.21 \leq \phi_c \leq 0.35$ indicates a *medium* effect, and $0.35 > \phi_c$ indicates a *large* effect. Based on this criteria we obtained a *large* effect size for the documentation-specific classification (Taxonomy A) and a *medium* effect size for the generalised classification (Taxonomy B).

2740 **6.6 Discussion**

2741 **6.6.1 Answers to Research Questions**

2742 **RQ1. How do developers mis-comprehend intelligent services as presented**
2743 **within Stack Overflow pain-points?** Upon meeting to discuss the discrepancies
2744 between our categorisation of intelligent service usage SO posts, we found that
2745 our interpretations of the *posts themselves* were largely subjective. For example,
2746 many posts presented multi-faceted dimensions for Taxonomy B; Beyer et al. [31]
2747 argue that a post can have more than one question category and therefore multi-label
2748 classification is appropriate at times. We highlight this further in the threats to
2749 validity (Section 6.7).

2750 We have to define the context of intelligent services to address RQ1. We use
2751 the concept of a “technical domain” [18] to define this context. A technical domain
2752 captures the domain-specific concerns that influence the non-functional requirements
2753 of a system [18]. In the context of intelligent services, the technical domain includes
2754 exploration, data engineering, distributed infrastructure, training data, and model
2755 characteristics as first class citizens [18]. We would then expect to see posts on SO
2756 related to these core concerns.

2757 In Figure 6.3, for the documentation-specific classification, the majority of posts
2758 were classified as **Completeness (A1-b)** related (47.87%). An interpretation for this
2759 is that the documentation does not adequately cover the technical domain concerns.
2760 Comments by developers such as “*I'm searching for a list of all the possible image*
2761 *labels that the Google Cloud Vision API can return?*” [345] indicates the documen-
2762 *tation does not adequately describe the training data for the API—developers do*
2763 *not know the required usage assumptions.* Another quote from a developer, “*Can*
2764 *Google Cloud Vision generate labels in Spanish via its API? ... [Does the API]*
2765 *allow to select which language to return the labels in?*” [365] points to a lack of
2766 details relating to the characteristics of the models used by the API. It would seem
2767 that developers are unaware of aspects of the technical domain concerns.

2768 The next most frequent category is **Correctness (A-1a)** with 22.87% of posts. In
2769 the context of the technical domain there are many limits that developers need to be
2770 aware of: range and increments of a model score [69]; required data pre-processing
2771 steps for optimal performance; and features provided by the models (as explained
2772 in Section 6.5.2). Considering the relation between technical concerns and software
2773 quality, developers are right to question providers on correctness; “*Are the Cloud*
2774 *Vision API limits in documentation correct?*” [357].

2775 **RQ2. Are the distribution of issues similar to prior studies?** Visual inspection of
2776 Figure 6.3 shows that the distributions for the documentation-specific classification
2777 and the generalised classification are different (compared to prior studies). As a
2778 sanity check we conducted a χ^2 test and calculated the effect size ϕ_c . We were able
2779 to reject the null hypothesis for both classification schemes, that the distribution of
2780 issues were the same as the previous studies (see Section 6.5). We now discuss the
2781 most prominent differences between our study and the previous studies.

2782 In the context of intelligent service SO posts, Taxonomy B suggests that Errors
2783 (B-3) are discussed most amongst developers. These results are in contrast to similar

2784 studies made in more *mature* API domains, such as Mobile Development [19, 20,
2785 30, 31, 245] and Web Development [281]. Here, API Usage (B-1) is much more
2786 frequently discussed, followed by Conceptual (B-5), Discrepancy (B-2) and Errors
2787 (B-3). We argue in the following section that an improved developer understanding
2788 can be achieved by educating them about the intelligent service lifecycle and the
2789 ‘whole’ system that wraps such services.

2790 In the Android study API usage questions (B-1) were the highest category
2791 (28.93% compared to 22.29% in our study). As stated in the analysis of the Error
2792 questions this discrepancy could be due to the maturity of the domain. However,
2793 another explanation could be the scope of the two individual studies. Beyer et al. [31]
2794 used a broad search strategy consisting of posts tagged Android. This search term
2795 fetches issues related to the entire Android platform which is significantly larger
2796 than searching for computer vision APIs using 229 search terms. As a consequence
2797 of more posts and more APIs there would be use cases resulting in additional posts
2798 related to API Usage (B-1).

2799 Applying existing SO taxonomies allowed us to better understand the distribution
2800 of the issues across different domains. In particular, the issues raised around
2801 intelligent services appear to be primarily due to poor documentation, or insufficient
2802 explanation around errors and limitations. Hence, many of the concerns could be
2803 addressed by adding more details to the end-point descriptions, and by providing
2804 additional information around how these services are designed to work.

2805 6.6.2 The Developer’s Learning Approach

2806 In this subsection, we offer an explanation as to why developers are complaining
2807 about certain things when trying to use intelligent services on SO (RQ1), as char-
2808 acterised through the use of prior SO classification frameworks (RQ2). This is
2809 described through the theoretical lenses of two learning taxonomies: Bloom’s con-
2810 text complexity and intellectual ability taxonomy, and the SOLO taxonomy (i.e., the
2811 nature by which developer’s learn). We argue that the issues with using intelligent
2812 services relating to the lower-levels of these learning taxonomies are easily solvable
2813 by slight fixes and improvements to the documentation of these services. However,
2814 the higher dimensions of these taxonomies demand far more rigorous mitigation
2815 strategies than documentation alone (potentially more structured education). Thus,
2816 many of the questions posted are from developers who are *learning to understand*
2817 the domain of intelligent services and AI, and (hence) both SOLO and Bloom’s tax-
2818 onomies are applicable for this discussion—as described below within the context
2819 of our domain—as pedagogical aides.

2820 **Bloom’s Taxonomy** The cognitive domain under Bloom’s taxonomy [34] consists
2821 of six objectives. Within the context of intelligent services, developers are likely
2822 to ask questions due to causal links that exist in the following layers of Bloom’s
2823 taxonomy: (i) *knowledge*, where the developer does not remember or know of the
2824 basic concepts of computer vision and AI (in essence, they may think that AI is as
2825 smart as a human); (ii) *comprehension*, where the developer does not understand

2826 how to interpret basic concepts, or they are mis-understanding how they are used
 2827 in context; (iii) *application*, where the developer is struggling to apply existing
 2828 concepts within the context of their own situation; (iv) *analysis*, where the developer
 2829 is unable to analyse the results from intelligent services (i.e., understand response
 2830 objects); (v) *evaluation*, where the developer is unable to evaluate issues and make
 2831 use of best-practices when using intelligent services; and (vi) *synthesise*, where
 2832 the developer is posing creative questions to ask if new concepts are possible with
 2833 computer vision services.

2834 **SOLO Taxonomy** The SOLO taxonomy [32] consists of five levels of under-
 2835 standing. The causal links behind the SO questions we have found relate to the
 2836 following layers of the SOLO taxonomy: (i) *pre-structural*, where the developer has
 2837 a question indicating incompetence or has little understanding of computer vision;
 2838 (ii) *uni-structural*, where the developer is struggling with one key aspect (i.e., a
 2839 simple question about computer vision); (iii) *multi-structural*, where the developer
 2840 is questioning multiple concepts (independently) to understand how to build their
 2841 system (e.g., system integration with the intelligent service); (iv) *relational*, where
 2842 the developer is comparing and contrasting the best ways to achieve something with
 2843 intelligent services; and (v) *extended abstract*, where the developer poses a question
 2844 theorising, formulating or postulating a new concept within intelligent services.

Table 6.2: Example Alignments of Stack Overflow posts to Bloom’s and the SOLO taxonomy.

Issue Quote	Bloom	SOLO
“I’m using Microsoft Face API for a small project and I was trying to detect a face inside a .jpg file in the local system (say, stored in a directory D:\Image\abc.jpg)... but it does not work.” [361]	Knowledge	Pre-Structural
“The problem is that the response JSON is rather big and confusing. It says a lot about the picture but doesn’t say what the whole picture is of (food or something like that).” [341]	Comprehension	Uni-Structural
“The bounding box around individual characters is sometimes accurate and sometimes not, often within the same image. Is this a normal side-effect of a probabilistic nature of the vision algorithm, a bug in the Vision API, or of course an issue with how I’m interpreting the response?” [348]	Comprehension	Multi-Structural
“I’m working on image processing. So far Google Cloud Vision and Clarifai are the best API’s to detect objects from images and videos, but both API’s doesn’t support object detection from 360 degree images and videos. Is there any solution for this problem?” [355]	Application	Uni-Structural
“Before I train Watson, I can delete pictures that may throw things off. Should I delete pictures of: Multiple dogs, A dog with another animal, A dog with a person, A partially obscured dog, A dog wearing glasses, Also, would dogs on a white background make for better training samples? Watson also takes negative examples. Would cats and other small animals be good negative examples?” [353]	Analysis	Relational

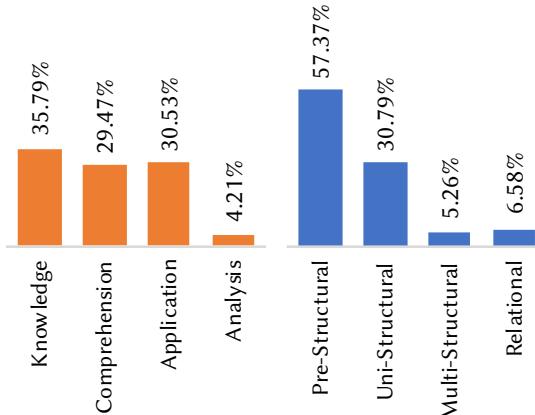


Figure 6.4: Alignment of Bloom (Orange) and SOLO (Blue) taxonomies against Taxonomy A and B dimensions against all 213 classifications made in the random sample of 50 posts.

2845 **Aligning SO taxonomies to Bloom’s and SOLO taxonomies** To understand our
 2846 findings with the lenses of pedagogical aids, we aligned Taxonomies A and B to
 2847 Bloom’s and the SOLO taxonomies for a random sample of 50 issues described in
 2848 Section 6.4.3. To do this, we reviewed all 50 of these SO posted questions and
 2849 applied both the Bloom and SOLO taxonomies. The primary author assigned each
 2850 of the 50 questions a level within the Bloom and SOLO taxonomies, removed out
 2851 noise (i.e., false positive posts of no relevance to intelligent services) and unassigned
 2852 dimensions from reliability agreement, and then compared the relevant dimensions of
 2853 Taxonomy A and B dimensions (not sub-categories). The comparison of alignments
 2854 of posts to the five SOLO dimensions and six Bloom dimensions are shown in
 2855 Figure 6.4. We acknowledge that this is only an approximation of the current state of
 2856 the developer’s understanding of intelligent services. This early model will require
 2857 further studies to perform a more thorough analysis, but we offer this interpretation
 2858 for early discussion.

2859 As shown in Figure 6.4, the bulk of the posts fall in the lower constructs of
 2860 Bloom’s and the SOLO taxonomy. This indicates that modification to certain doc-
 2861 umentation aspects can address many of these issues. For example, many issues
 2862 can be ratified with better descriptions of response data and error messages: “*I was*
 2863 *exploring google vision and in the specific function ‘detectCrops’, gives me the crop*
 2864 *hints. what does this means exactly?*” [356]; “*I am a making a very simple API call*
 2865 *to the Google Vision API, but all the time it’s giving me error that ‘google.oauth2’*
 2866 *module not found.*” [371]

2867 However, and more importantly, the higher-construct questions ranging from
 2868 the middle of the third dimensions on are not as easily solvable through improved
 2869 documentation (i.e., apply and multi-structural) which leaves 34.74% (Bloom’s)
 2870 and 11.84% (SOLO) unaccounted for, resolvable only through improved education
 2871 practices.

2872 6.6.3 Implications

2873 **Researchers** *(i) Investigate the evolution of post classification:* Analysing how the
2874 distribution of the reported issues changes over time would be an important study.
2875 This study could answer questions such as ‘*Does the evolution of intelligent services*
2876 *follow the same pattern as previous software engineering trends such as mobile app*
2877 *or web development?*’ As with any new emerging field, it is key to analyse how
2878 developers perceive such issues over time. For instance, early issues with web or
2879 mobile app development matured as their respective domain matured, and we would
2880 expect similar results to occur in the intelligent services space. Future researchers
2881 could plan for a longitudinal study, such as a long-term survey with developers to
2882 gather their insights in this evolving domain, reviewing case studies of projects that
2883 use intelligent web services from now into the future, or re-mining SO at a later date
2884 and comparing the results to this study. This will help assess evolving trends and
2885 characteristics, and determine how and if the nature of the developer’s experience
2886 with intelligent services (and AI in general) changes with time. *(ii) Investigate the*
2887 *impact of technical challenges on API usage:* As discussed above, intelligent services
2888 have characteristics that may influence API usage patterns and should be investigated
2889 as a further avenue of research. Further mining of open source software repositories
2890 that make use of intelligent services could be assessed, thereby investigating if API
2891 patterns evolve with the rise of AI-based applications.

2892 **Educators** *(i) Education on high-level aspects of intelligent services:* As demon-
2893 strated in our analysis of their SO posts, many developers appear to be unaware of
2894 the higher-level concepts that exist within the AI and ML realm. This includes the
2895 need to pre- and post-process data, the data dependency and instability that exists in
2896 these services, and the specific algorithms that empower the underlying intelligence
2897 and hence their limitations and characteristics. However, most developers don’t
2898 seem to complain about these factors due to the lack of documentation (i.e., via
2899 Taxonomy A). Rather, they are unaware that such information should be documen-
2900 tation and instead ask generalised and open questions (i.e., via Taxonomy B). Thus,
2901 documentation improvements alone may not be enough to solve these issues. This
2902 results in uncertainty during the preparation and operation (usage) of such services.
2903 Such high-level conceptual information is currently largely missing in developer
2904 documentation for intelligent services. Furthermore, many of the background ML
2905 and AI algorithm information needed to understand and use intelligent systems in
2906 context are built within data science (not SE) communities. A possible road-map to
2907 mitigate this issue would be the development of a software engineer’s ‘crash-course’
2908 in ML and AI. The aim of such a course would encourage software engineers to
2909 develop an appreciation of the nuances and the inherent risks and implications that
2910 comes with using intelligent services. This could be taught at an undergraduate
2911 level to prepare the next generation of developers of a ‘programming 2.0’ era. How-
2912 ever, the key aspects and implications that are presented with AI would need to be
2913 well-understood before such a course is developed, and determining the best strategy
2914 to curate the content to developers would be best left to the SE education domain.

2915 Further investigation in applying educational taxonomies in the area (such as our
2916 attempts to interpret our findings using Bloom's and the SOLO taxonomies) would
2917 need to be thoroughly explored beforehand.

2918 **Software Engineers** *(i) Better understanding of intelligent API contextual usage:*
2919 Our results show that developers are still learning to use these APIs. We applied
2920 two learning perspectives to interpret our results. In applying the two pedagogical
2921 taxonomies to our findings, we see that most issues seem to fall into the pre-structural
2922 and knowledge-based categories; little is asked of higher level concepts and a ma-
2923 jority of issues do not offer complex analysis from developers. This suggests that
2924 developers are struggling as they are unaware of the vocabulary needed to actually
2925 use such APIs, further reinforcing the need for API providers to write overview
2926 documentation (as noted in prior work [68]) and not just simple endpoint documen-
2927 tation. This said, improved documentation isn't always enough—as suggested by
2928 our discussion in Section 6.6.2, software engineers should explore further education
2929 to attain a greater appreciation of the nuances of ML when attempting to use these
2930 services.

2931 **Intelligent Service Providers** *(i) Clarify use cases for intelligent services:* In-
2932 specting SO posts revealed that there is a level of confusion around the capabilities
2933 of different intelligent services. This needs to be clarified in associated API doc-
2934 umentation. The complication with this comes with targeting the documentation
2935 such that software developers (who are untrained in the nuances of AI and ML as
2936 per Section 6.6.3) can to digest it and apply it in-context to application develop-
2937 ment. *(ii) Technical domain matters:* More needs to be provided than a simple
2938 endpoint description as conventional APIs offer by describing the whole framework
2939 by which the endpoint sits, giving further context. This said, compared to traditional
2940 APIs, we find that developers complain less about the documentation and more
2941 about shallower issues. All expected pre-processing and post-processing needs to be
2942 clearly explained. A possible mitigation to this could be an interactive tutorial that
2943 helps developers fully understand the technical domain using a hands-on approach.
2944 For example, websites offer interactive Git tutorials⁷ to help developers understand
2945 and explore the technical domain matters under version control in their own pace.
2946 *(iii) Clarify limitations:* API developers need to add clear limitations of the existing
2947 APIs. Limitations include list of objects that can be returned from an endpoint. We
2948 found that the cognitive anchors of how existing, conventional API documentation
2949 is written has become ‘ported’ to the computer vision realm, however a lot more
2950 overview documentation than what is given at present (i.e., better descriptions of
2951 errors, improved context of how these systems work in etc.) needs to be given. Such
2952 documentation could be provided using interactive tutorials.

⁷For example, <https://learngitbranching.js.org>.

2953 6.7 Threats to Validity**2954 6.7.1 Internal Validity**

2955 As detailed in Section 6.4.3, Taxonomies A and B present slight and fair agreement,
2956 respectively, when inter-rater reliability was applied. The nature of our disagree-
2957 ments largely fell due to the subjectivity in applying either taxonomies to posts.
2958 Despite all coders agreeing to the shared interpretation of both taxonomies, both
2959 taxonomies are subjective in their application, which was not reported by either
2960 Aghajani et al. or Beyer et al.. In many cases, multi-label classification seemed ap-
2961 propriate, however both taxonomies use single-label mapping which we find results
2962 in too much subjectivity. This subjectivity, therefore, ultimately adversely affects
2963 IRR analysis. Thus, a future mitigation strategy for similar work should explore
2964 multi-label classification to avoid this issue; Beyer et al., for example, plan for multi-
2965 label classification as future work. However, these studies would need to consider
2966 the statistical challenges in calculating multi-rater, multi-label IRR for thorough re-
2967 liability analysis in addressing subjectivity. The selection of SO posts used for our
2968 labelling, chiefly in the subjectivity of our classifications, is of concern. We mitigate
2969 this by an extensive review process assessing the reliability of our results as per Sec-
2970 tion 6.4.3. The classification of our posts into the SOLO and Bloom’s taxonomies
2971 was performed by the primary author only, and therefore no inter-rater reliability
2972 statistics were performed. However, we used these pedagogy related taxonomies as
2973 a lens to gain an additional perspective to interpret our results. Future studies should
2974 attempt a more rigorous analysis of SO posts using Bloom’s and SOLO taxonomies.
2975 We only aligned posts to one category for each taxonomy and did not align these
2976 using multi-label classification. This brings more complexity to the analysis, and
2977 our attempts to repeat prior studies’ methodologies (see Section 6.3). Multi-label
2978 classification for intelligent services SO posts is an avenue for future research.

2979 6.7.2 External Validity

2980 While every effort was made to select posts from SO relevant to computer vision
2981 services, there are some cases where we may have missed some posts. This is
2982 especially due to the case where some developers mis-reference certain intelligent
2983 services under different names (see Section 6.4.2). Our SOLO and Bloom’s taxon-
2984 omy analysis has only been investigated through the lenses of intelligent services,
2985 and not in terms of conventional APIs (e.g., Andriod APIs). Therefore, we are not
2986 fully certain how these results found would compare to other types of APIs. Two
2987 *existing* SO classification taxonomies were used rather than developing our own.
2988 We wanted to see if previous SO taxonomies could be applied to intelligent services
2989 before developing a new, specific taxonomy, and these taxonomies were applied
2990 based on our interpretation (see Section 6.4.2) and may not necessarily reflect the
2991 interpretation of the original authors. Moreover, automated techniques such as topic
2992 modelling were not utilised as we found these produce descriptive classifications
2993 only (see Section 6.3). Hence, manual analysis was performed by humans to ensure
2994 categories could be aligned back to causal factors. Only English-speaking intelligent

2995 services were selected; the applicability of our analysis to other, non-English speaking
2996 services may affect results. Use of computer vision in this study is an illustrative
2997 example to focus on one area of the intelligent services spectrum. While our narrow
2998 scope helps us obtain more concrete findings, we suggest that wider issues exist in
2999 other intelligent service domains may affect the generalisability of this study, and
3000 suggest future work be explored in this space.

3001 **6.7.3 Construct Validity**

3002 Some questions extracted from SO produced false positives, as mentioned in Section
3003 6.4.2 and Section 6.5. However, all non-relevant posts were marked as noise
3004 for our study, and thus did not affect our findings. Moreover, SO is known to have
3005 issues where developers simply ask basic questions without looking at the actual
3006 documentation where the answer exists. Such questions, although down-voted, were
3007 still included in our data-set analysis, but as these were so few, it does not have a
3008 substantial impact on categorised posts.

3009 **6.8 Conclusions**

3010 Intelligent services, such as computer vision services, offer powerful capabilities
3011 that can be added into the developer's toolkit via simple RESTful APIs. However,
3012 certain technical nuances of computer vision become abstracted away. We note that
3013 this abstraction comes at the expense of a full appreciation of the technical domain,
3014 context and proper usage of these systems. We applied two recent existing SO
3015 classification taxonomies (from 2018 and 2019) to see if existing taxonomies are
3016 able to fully categorise the types of complaints developers have. Intelligent services
3017 have a diverging distribution of the types of issues developers ask when compared
3018 to more mature domains (i.e., mobile app development and web development).
3019 Developers are more likely to complain about shallower, simple debugging issues
3020 without a distinct understanding of the AI algorithms that actually empower the APIs
3021 they use. Moreover, developers are more likely to complain about the completeness
3022 and correctness of existing intelligent service documentation, thereby suggesting
3023 that the documentation approach for these services should be reconsidered. Greater
3024 attention to education in the use of AI-powered APIs and their limitations is needed,
3025 and our discussion offered in Section 6.6.2 motivates future work in resolving these
3026 issues in the SE education space.

3027 CHAPTER 7

3028

3029 Ranking Computer Vision Service Issues using Emotion[†]

3030

[†]This chapter is originally based on M. K. Curumsing, A. Cummaudo, U. M. Graestch, S. Barnett, and R. Vasa, “Ranking Computer Vision Service Issues using Emotion,” in *Proceedings of the 5th International Workshop on Emotion Awareness in Software Engineering*, Seoul, Republic of Korea, May 2020. Terminology has been updated to fit this thesis.

3031 CHAPTER 8

3032

3033 Using a Facade Pattern to combine Computer Vision Services[†]

3034

3035 **Abstract** Intelligent computer vision services, such as Google Cloud Vision or Amazon
3036 Rekognition, are becoming evermore pervasive and easily accessible to developers to build
3037 applications. Because of the stochastic nature that ML entails and disparate datasets used in
3038 their training, the outputs from different computer vision services varies with time, resulting
3039 in low reliability—for some cases—when compared against each other. Merging multiple
3040 unreliable API responses from multiple vendors may increase the reliability of the overall
3041 response, and thus the reliability of the intelligent end-product. We introduce a novel
3042 methodology—inspired by the proportional representation used in electoral systems—to
3043 merge outputs of different intelligent computer vision API provided by multiple vendors.
3044 Experiments show that our method outperforms both naive merge methods and traditional
3045 proportional representation methods by 0.015 F-measure.

3046 **8.1 Introduction**

3047 With the introduction of intelligent web services (IWSs) that make machine learning
3048 (ML) more accessible to developers [237, 290], we have seen a large growth of
3049 intelligent applications dependent on such services [49, 107]. For example, consider
3050 the advances made in computer vision, where objects are localised within an image
3051 and labelled with associated categories. Cloud-based computer vision services
3052 (CVSs)—e.g., [313, 316, 324, 327, 330, 331, 335, 373]—are a subset of IWSs.
3053 They utilise ML techniques to achieve image recognition via a remote black-box
3054 approach, thereby reducing the overhead for application developers to understand
3055 how to implement intelligent systems from scratch. Furthermore, as the processing

[†]This chapter is originally based on T. Ohtake, A. Cummaudo, M. Abdelrazek, R. Vasa, and J. Grundy, “Merging intelligent API responses using a proportional representation approach,” in *Proceedings of the 19th International Conference on Web Engineering*. Daejeon, Republic of Korea: Springer, June 2019. DOI 10.1007/978-3-030-19274-7_28. ISBN 978-3-03-019273-0. ISSN 1611-3349 pp. 391–406. Terminology has been updated to fit this thesis.

3056 and training of the machine-learnt algorithms is offloaded to the cloud, developers
3057 simply send RESTful API requests to do the recognition. There are, however, inherit
3058 differences and drawbacks between traditional web services and IWSs, which we
3059 describe with the motivating scenario below.

3060 **8.1.1 Motivating Scenario: Intelligent vs Traditional Web Services**

3061 An application developer, Tom, wishes to develop a social media Android and iOS
3062 app that catalogues photos of him and his friends, common objects in the photo,
3063 and generates brief descriptions in the photo (e.g., all photos with his husky dog,
3064 all photos on a sunny day etc.). Tom comes from a typical software engineering
3065 background with little knowledge of computer vision and its underlying concepts.
3066 He knows that intelligent computer vision web APIs are far more accessible than
3067 building a computer vision engine from scratch, and opts for building his app using
3068 these cloud services instead.

3069 Based on his experiences using similar cloud services, Tom would expect consistency
3070 of the results from the same API and different APIs that provide the same (or similar)
3071 functionality. As an analogy, when Tom writes the Java substring method
3072 "doggy".substring(0, 2), he expects it to be the same result as the Swift equivalent
3073 "doggy".prefix(3). Each and every time he interacts with the substring
3074 method using either API, he gets "dog" as the response. This is because Tom is
3075 used to deterministic, rule-driven APIs that drive the implementation behind the
3076 substring method.

3077 Tom's deterministic mindset results in three key differentials between a traditional
3078 web service and an IWS:

3079 **(1) Given similar input, results differ between similar IWSs.** When Tom
3080 interacts with the API of an IWS, he is not aware that each API provider trains
3081 their own, unique ML model, both with disparate methods and datasets. These
3082 IWSs are, therefore, nondeterministic and data-driven; input images—even
3083 if they contain the same conceptual objects—often output different results.
3084 Contrast this to the substring method of traditional APIs; regardless of what
3085 programming language or string library is used, the same response is expected
3086 by developers.

3087 **(2) Intelligent responses are not certain.** When Tom interprets the response
3088 object of an IWS, he finds that there is a ‘confidence’ value or ‘score’. This
3089 is because the ML models that power IWSs are inherently probabilistic and
3090 stochastic; any insight they produce is purely statistical and associational [223].
3091 Unlike the substring example, where the rule-driven implementation provides
3092 certainty to the results, this is not guaranteed for IWSs. For example, a picture
3093 of a husky breed of dog is misclassified as a wolf. This could be due to
3094 adversarial examples [273] that ‘trick’ the model into misclassifying images
3095 when they are fully decipherable to humans. It is well-studied that such
3096 adversarial examples exist in the real world unintentionally [89, 162, 226].

3097 **(3) Intelligent APIs evolve over time.** Tom may find that responses to processing
3098 an image may change over time; the labels he processes in testing may evolve

3099 and therefore differ to when in production. In traditional web services, evo-
3100 lution in responses is slower, generally well-communicated, and usually rare
3101 (Tom would always expect "dog" to be returned in the substring example).
3102 This has many implications on software systems that depend on these APIs,
3103 such as confidence in the output and portability of the solution. Currently, if
3104 Tom switches from one API provider to another, or if he doesn't regularly test
3105 his app in production, he may begin to see a very different set of labels and
3106 confidence levels.

3107 **8.1.2 Research Motivation**

3108 These drawbacks bring difficulties to the intended API users like Tom. We identify a
3109 gap in the software engineering literature regarding such drawbacks, including: lack
3110 of best practices in using IWSs; assessing and improving the reliability of APIs for
3111 their use in end-products; evaluating which API is suitable for different developer
3112 and application needs; and how to mitigate risk associated with these APIs. We
3113 focus on improving reliability of CVSs for use in end-products. The key research
3114 questions in this paper are:

3115 **RQ1:** Is it possible to improve reliability by merging multiple CVS results?

3116 **RQ2:** Are there better algorithms for merging these results than currently in
3117 use?

3118 Previous attempts at overcoming low reliability include triple-modular redun-
3119 dancy [178]. This method uses three modules and decides output using majority
3120 rule. However, in CVSs, it is difficult to apply majority rule: these APIs respond with
3121 a list of labels and corresponding scores. Moreover, disparate APIs ordinarily output
3122 different results. These differences make it hard to apply majority rule because the
3123 type of outputs are complex and disparate APIs output different results for the same
3124 input. Merging search results is another technique to improve reliability [261]. It
3125 normalises scores of different databases using a centralised sample database. Nor-
3126 malising scores makes it possible to merge search results into a single ranked list.
3127 However, search responses are disjoint, whereas they are not in the context of most
3128 CVSs.

3129 In this paper, we introduce a novel method to merge responses of CVSs, using
3130 image recognition APIs endpoints as our motivating example. Section 8.2 describes
3131 naive merging methods and requirements. Section 8.3 gives insights into the struc-
3132 ture of labels. Section 8.4 introduces our method of merging computer vision labels.
3133 Section 8.5 compares precision and recall for each method. Section 8.6 presents
3134 conclusions and future work.

3135 **8.2 Merging API Responses**

3136 Image recognition APIs have similar interfaces: they receive a single input (image)
3137 and respond with a list of labels and associated confidence scores. Similarly, other
3138 supervised-AI-based APIs do the same (e.g., detecting emotions from text and

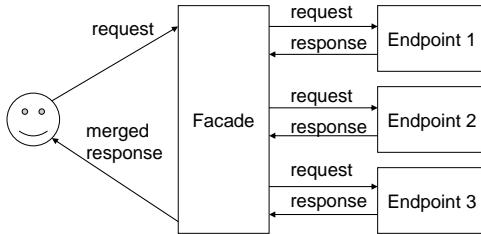


Figure 8.1: The user sends a request to the facade; this request is propagated to the relevant APIs. Responses are merged by the facade and returned back to the user.

3139 natural language processing [332, 374]). It is difficult to apply majority rule on such
 3140 disparate, complex outputs. While the outputs by *multiple* AI-based API endpoints
 3141 is different and complex, the general format of the output is the same: a list of labels
 3142 and associated scores.

3143 8.2.1 API Facade Pattern

3144 To merge responses from multiple APIs, we introduce the notion of an API facade.
 3145 It is similar to a metasearch engine, but differs in their external endpoints. The
 3146 facade accepts the input from one API endpoint (the facade endpoint), propagates
 3147 that input to all user-registered concrete (external) API endpoints simultaneously,
 3148 then ‘merges’ outputs from these concrete endpoints before sending this merged
 3149 response to the API user. We demonstrate this process in Figure 8.1.

3150 Although the model introduces more time and cost overhead, both can be miti-
 3151 gated by caching results. On the other hand, the facade pattern provides the following
 3152 benefits:

- 3153 • **Easy to modify:** It requires only small modifications to applications, e.g.,
 3154 changing each concrete endpoint URL.
- 3155 • **Easy to customise:** It merges results from disparate and concrete APIs ac-
 3156 cording to the user’s preference.
- 3157 • **Improves reliability:** It enhances reliability of the overall returned result by
 3158 merging results from different endpoints.

3159 8.2.2 Merge Operations

3160 The API facade is applicable to many use cases. However, this paper focuses on
 3161 APIs that output a list of labels and scores, as is the case for CVSs. Merge operations
 3162 involve the mapping of multiple lists and associated scores, produced by multiple
 3163 APIs, to just one list. For instance, a CVS receives a bowl of fruit as the input image
 3164 and outputs the following:

3165 `[[‘apple’, 0.9], [‘banana’, 0.8]]`

3166 where the first item is the label and the second item is the score. Similarly, another
 3167 computer vision API outputs the following for the same image:

3168 $[[\text{'apple'}, 0.7], [\text{'cherry'}, 0.8]]$.

3169 Merge operations can, therefore, merge these two responses into just one response.
 3170 Naive ways of merging results could make use of *max*, *min*, and *average* operations
 3171 on the confidence scores. For example, *max* merges results to:

3172 $[[\text{'apple'}, 0.9], [\text{'banana'}, 0.8], [\text{'cherry'}, 0.8]]$;

3173 *min* merges results to:

3174 $[[\text{'apple'}, 0.7]]$;

3175 and *average* merges results to:

3176 $[[\text{'apple'}, 0.8], [\text{'banana'}, 0.4], [\text{'cherry'}, 0.4]]$.

3177 However, as the object's labels in each result are natural language, the operations
 3178 do not exploit the label's semantics when conducting label merging. To improve
 3179 the quality of the merged results, we consider the ontologies of these labels, as we
 3180 describe below.

3181 8.2.3 Merging Operators for Labels

3182 Merge operations on labels are *n*-ary operations that map R^n to R , where $R_i =$
 3183 $\{(l_{ij}, s_{ij})\}$ is a response from endpoint i and contains pairs of labels (l_{ij}) and scores
 3184 (s_{ij}). Merge operations on labels have the following properties:

- 3185 • *identity* defines that merging a single response should output same response
 (i.e., $R = \text{merge}(R)$ is always true);
- 3187 • *commutativity* defines that the order of operands should not change the result
 (i.e., $\text{merge}(R_1, R_2) = \text{merge}(R_2, R_1)$ is always true);
- 3189 • *reflexivity* defines that merging multiple same responses should output same
 response (i.e., $R = \text{merge}(R, R)$ is always true); and,
- 3191 • *additivity* defines that, for a specific label, the merged response should have
 higher or equal score for the label if a concrete endpoint has a higher score.
 Let $R = \text{merge}(R_1, R_2)$ and $R' = \text{merge}(R'_1, R_2)$ be merged responses. R_1 and
 R'_1 are same, except R'_1 has a higher score for label l_x than R_1 . The additive
 score property requires that R' score for l_x should be greater than or equal to
 R score for l_x .

3197 The *max*, *min*, and *average* operations in Section 8.2.2 follow each of these rules
 3198 as all operations calculate the score by applying these operations on each score.

Table 8.1: Statistics for the number of labels, on average, per service identified.

Endpoint	Average number of labels	Has synset	No synset
Amazon Rekognition	11.42 ± 7.52	10.74 ± 7.10 (94.0%)	0.66 ± 0.87
Google Cloud Vision	8.77 ± 2.15	6.36 ± 2.22 (72.5%)	2.41 ± 1.93
Azure Computer Vision	5.39 ± 3.29	5.26 ± 3.32 (97.6%)	0.14 ± 0.37

3199 8.3 Graph of Labels

3200 CVSSs typically return lists of labels and their associated scores. In most cases, the
 3201 label can be a singular word (e.g., ‘husky’) or multiple words (e.g., ‘dog breed’).
 3202 Lexical databases, such as WordNet [196], can therefore be used to describe the
 3203 ontology behind these labels’ meanings. Figure 8.2 is an example of a graph of
 3204 labels and synsets. A synset is a grouped set of synonyms for a word. In this image,
 3205 we consider two fictional endpoints, endpoints 1–2. We label red nodes as labels
 3206 from endpoint 1, yellow nodes as labels from endpoint 2, and blue nodes as synsets
 3207 for the associated labels from both endpoints. As actual graphs are usually more
 3208 complex, Figure 8.2 is a simplified graph to illustrate the usage of associating labels
 3209 from two concrete sources to synsets.

3210 8.3.1 Labels and synsets

3211 The number of labels depends on input images and concrete API endpoints used.
 3212 Table 8.1 and Figure 8.3 show how many labels are returned, on average per image,
 3213 from Google Cloud Vision [327], Amazon Rekognition [313] and Azure Computer
 3214 Vision [335] image recognition APIs. These statistics were calculated using 1,000
 3215 images from Open Images Dataset V4 [328] Image-Level Labels set.

3216 Labels from Amazon and Microsoft tend to have corresponding synsets, and
 3217 therefore these endpoints return common words that are found in WordNet. On the
 3218 other hand, Google’s labels have less corresponding synsets: for example, labels
 3219 without corresponding synsets are car models and dog breeds.¹

3220 8.3.2 Connected Components

3221 A connected component (CC) is a subgraph in which there are paths between any
 3222 two nodes. In graphs of labels and synsets, CCs are clusters of labels and synsets
 3223 with similar semantic meaning. For instance, there are two CCs in Figure 8.2. CC 1
 3224 in Figure 8.2 has ‘beverage’, ‘dessert’, ‘chocolate’, ‘hot chocolate’,
 3225 ‘drink’, and ‘food’ labels from the red first endpoint and ‘coffee’, ‘hot
 3226 chocolate’, ‘drink’, ‘caffeine’, and ‘tea’ labels from the yellow second
 3227 endpoint. Therefore, these labels are related to ‘drink’. On the other hand, CC 2
 3228 in Figure 8.2 has ‘cup’ and ‘coffee cup’ labels from the first red endpoint and
 3229 ‘cup’, ‘coffee cup’, and ‘tableware’ labels from the yellow second endpoint.
 3230 These labels are, therefore, related to ‘cup’.

¹We noticed from our upload of 1,000 images that Google tries to identify objects in greater detail.

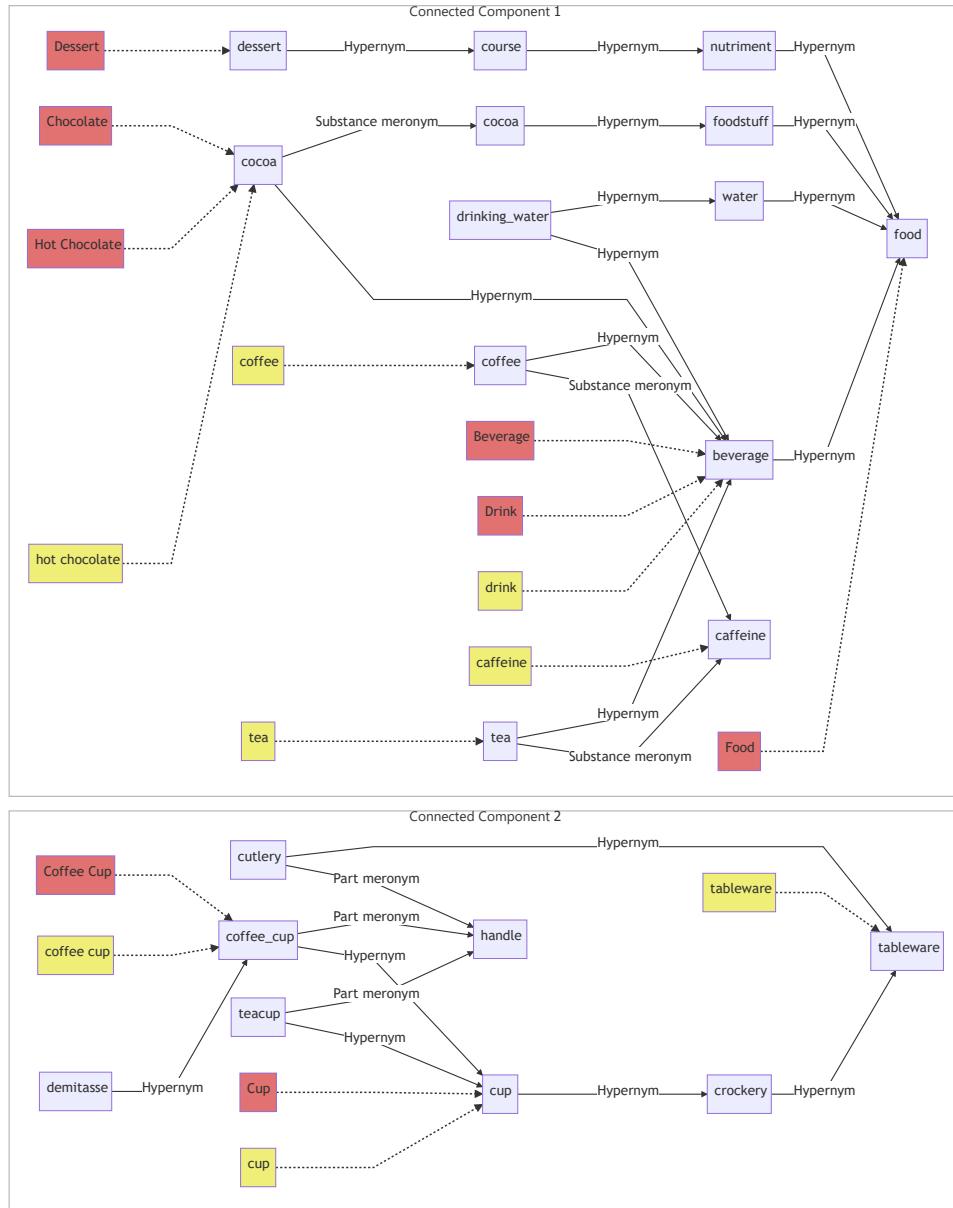


Figure 8.2: Graph of labels from two concrete endpoints (red and yellow) and their associated synsets related to both words (blue).

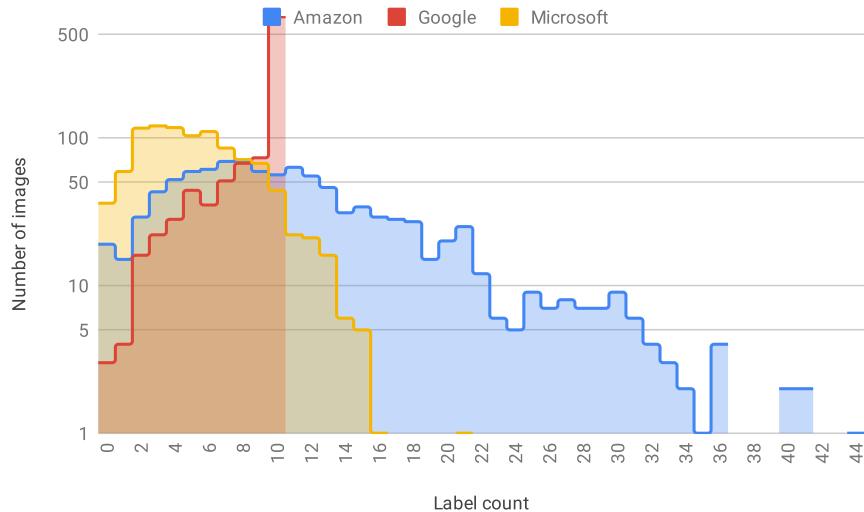


Figure 8.3: Number of labels responded from our input dataset to three concrete APIs assessed.

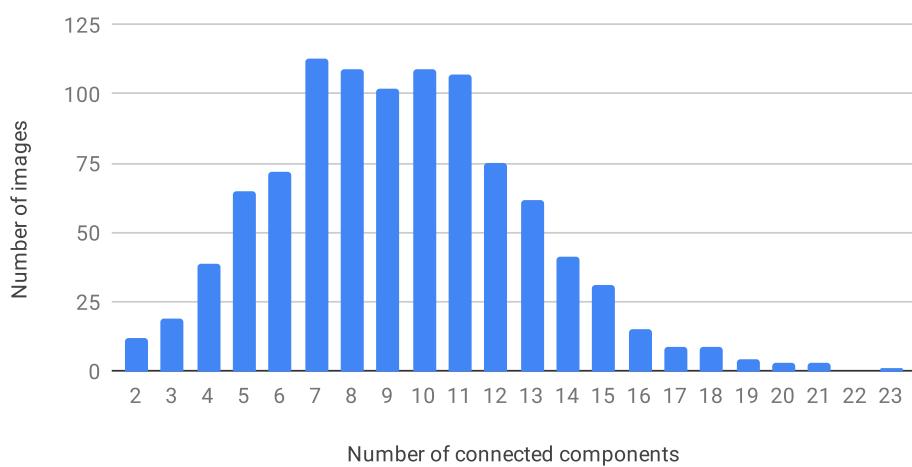


Figure 8.4: Number of connected components compared to the number of images.

3231 Figure 8.4 shows a distribution of number of CCs for the 1,000-image label
 3232 detections on Amazon Rekognition, Google Cloud Vision, and Azure Computer
 3233 Vision APIs. The average number of CCs is 9.36 ± 3.49 . The smaller number of
 3234 CCs means that most of labels have similar meanings, while a larger value means
 3235 that the labels are more disparate.

3236 **8.4 API Results Merging Algorithm**

3237 Our proposed algorithm to merge labels consists of four parts: (1) mapping labels to
 3238 synsets, (2) deciding the total number of labels, (3) allocating the number of labels
 3239 to CCs, and (4) selecting labels from CCs.

3240 **8.4.1 Mapping Labels to Synsets**

3241 Labels returned in CVS responses are words (in natural language) that do not always
 3242 identify their intended meanings. For instance, a label *orange* may represent the
 3243 fruit, the colour, or the name of the longest river in South Africa. To identify the
 3244 actual meanings behind a label, our facade enumerates all synsets corresponding to
 3245 labels. It then finds the most likely synsets for labels by traversing WordNet links.
 3246 For instance, if an API endpoint outputs the ‘orange’ and ‘lemon’ labels, the
 3247 facade regards ‘orange’ as a related synset word of ‘fruit’. If an API endpoint
 3248 outputs ‘orange’ and ‘water’ labels, the facade regards ‘orange’ as a ‘river’.

3249 **8.4.2 Deciding Total Number of Labels**

3250 The number of labels in responses from endpoints vary as described in Section 8.3.1.
 3251 The facade decides the number of merged labels using the numbers of labels from
 3252 each endpoint. We formulate the following equation to calculate the number of
 3253 labels:

$$\min_i(|R_i|) \leq \frac{\sum_i|R_i|}{n} \leq \max_i(|R_i|) \leq \sum_i|R_i|$$

3254 where $|R|$ is number of labels and scores in response, and n is number of endpoints.
 3255 In case of naive operations in Section 8.2.2, the following is true:

$$\begin{aligned} |\text{merge}_{\max}(R_1, \dots, R_n)| &\leq \min_i(|R_i|) \\ \max_i(|R_i|) &\leq |\text{merge}_{\min}(R_1, \dots, R_n)| \leq \sum_i|R_i| \\ \max_i(|R_i|) &\leq |\text{merge}_{\text{average}}(R_1, \dots, R_n)| \leq \sum_i|R_i|. \end{aligned}$$

3256 The proposal uses $\lfloor \sum_i|R_i|/n \rfloor$ to conform to the necessary condition described in
 3257 Section 8.4.3.

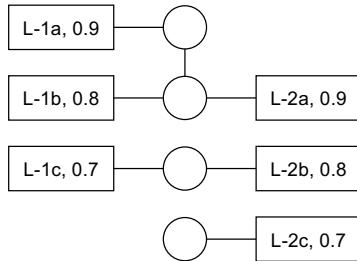


Figure 8.5: Allocation to connected components.

3258 8.4.3 Allocating Number of Labels to Connected Components

3259 The graph of labels and synsets is then divided into several CCs. The facade decides
 3260 how many labels are allocated for each CC. For example, in Figure 8.5, there are
 3261 three CCs, where square-shaped nodes are labels in responses from endpoints. Text
 3262 within these label nodes describe which endpoint outputs the label and score, for
 3263 instance, “L-1a, 0.9” is label *a* from endpoint *1* with a score *0.9*. Circle-shaped nodes
 3264 represent synsets, where the edges between the label and synset nodes indicate the
 3265 relationships between them. Edges between synsets are links in WordNet.

3266 Allegorically, allocating the number of labels to CCs is similar to proportional
 3267 representation in a political voting system, where CCs are the political parties and
 3268 labels are the votes to a party. Several allocation algorithms are introduced in
 3269 proportional representation, for instance, the D’Hondt and Hare-Niemeyer methods
 3270 [207]. However, there are differences from proportional representation in the politi-
 3271 cal context. For label merging, labels have scores and origin endpoints and such
 3272 information may improve the allocation algorithm. For instance, CCs supported
 3273 with more endpoints should have a higher allocation than CCs with fewer endpoints,
 3274 and CCs with higher scores should have a higher allocation than CCs with lower
 3275 scores. We introduce an algorithm to allocate the number of labels to CCs. This
 3276 allocates more to a CC with more supporting endpoints and higher scores. The steps
 3277 of the algorithm are:

- 3278 **Step I.** Sort scores separately for each endpoint.
- 3279 **Step II.** If all CCs have an empty score array or more, remove one, and go to Step
 II.
- 3281 **Step III.** Select the highest score for each endpoint and calculate product of highest
 scores.
- 3283 **Step IV.** A CC with the highest product score receives an allocation. This CC
 removes every first element from the score array.
- 3285 **Step V.** If the requested number of allocations is complete, then stop allocation.
 Otherwise, go to Step II.

3287 Tables 8.2 to 8.5 are examples of allocation iterations. In Table 8.2, the facade
 3288 sorts scores separately for each endpoint. For instance, the first CC in Figure 8.5
 3289 has scores of 0.9 and 0.8 from endpoint 1 and 0.9 from endpoint 2. All CCs have a

Table 8.2: Allocation iteration 1.

Scores	Highest	Product	Allocated
[0.9, 0.8], [0.9]	[0.9, 0.9]	0.81	0+1
[0.7], [0.8]	[0.7, 0.8]	0.56	0
[], [0.7]	[N/A, 0.7]	N/A	0

Table 8.4: Allocation iteration 3.

Scores	Highest	Product	Allocated
[0.8], []	—	—	1
[], []	—	—	1
[], [0.7]	—	—	0

Table 8.3: Allocation iteration 2.

Scores	Highest	Product	Allocated
[0.8], []	[0.8, N/A]	N/A	1
[0.7], [0.8]	[0.7, 0.8]	0.56	0+1
[], [0.7]	[N/A, 0.7]	N/A	0

Table 8.5: Allocation iteration 4.

Scores	Highest	Product	Allocated
[0.8]	[0.8]	0.8	1+1
[]	[N/A]	N/A	1
[0.7]	[0.7]	0.7	0

3290 non-empty score array or more, so the facade skips Step II. The facade then picks
 3291 the highest scores for each endpoint and CC. CC 1 has the largest product of highest
 3292 scores and receives an allocation. In Table 8.3, the first CC removes every first score
 3293 in its array as it received an allocation in Table 8.2. In this iteration, the second CC
 3294 has largest product of scores and receives an allocation. In Table 8.4, the second CC
 3295 removes every first score in its array. At Step II, all the three CCs have an empty
 3296 array. The facade removes one empty array from each CC. In Table 8.5, the first CC
 3297 receives an allocation. The algorithm is applicable if total number of allocation is
 3298 less than or equal to $\max_i(|R_i|)$ as scores are removed in Step II. The condition is a
 3299 necessary condition.

3300 8.4.4 Selecting Labels from Connected Components

3301 For each CC, the facade applies the *average* operator from Section 8.2.2 and takes
 3302 labels with n -highest scores up to allocation, as per Section 8.4.3.

3303 8.4.5 Conformance to properties

3304 Section 8.2.3 defines four properties: identity, commutativity, reflexivity, and addi-
 3305 tivity. Our proposed method conforms to these properties:

- 3306 • *identity*: the method outputs same result if there is one response;
- 3307 • *commutativity*: the method does not care about ordering of operands;
- 3308 • *reflexivity*: the allocations to CCs are same to number of labels in CCs; and
- 3309 • *additivity*: increases in score increases or does not change the allocation to
 3310 the corresponding CC.

3311 8.5 Evaluation

3312 8.5.1 Evaluation Method

3313 To evaluate the merge methods, we merged CVS results from three representative
 3314 image analysis API endpoints and compared these merged results against human-

3315 verified labels. Images and human-verified labels are sourced from 1,000 randomly-
 3316 sampled images from the Open Images Dataset V4 [328] Image-Level Labels test
 3317 set.

3318 The first three rows in Table 8.7 are the evaluation of original responses from
 3319 each API endpoint. Precision, recall, and F-measure in Table 8.7 do not reflect
 3320 actual values: for instance, it appears that Google performs best at first glance, but
 3321 this is mainly because Google’s labels are similar to that of the Open Images label
 3322 set.

3323 The Open Images Dataset uses 19,995 classes for labelling. The human-verified
 3324 labels for the 1,000 images contain 8,878 of these classes. Table 8.6 shows the
 3325 correspondence between each service’s labels and the Open Images Dataset classes.
 3326 For instance, Amazon Rekognition outputs 11,416 labels in total for 1,000 images.
 3327 There are 1,409 unique labels in 11,416 labels. 1,111 labels out of 1,409 can be
 3328 found in Open Images Dataset classes. Rekognition’s labels matches to Open Images
 3329 Dataset classes at 78.9% ratio, while Google has an outstanding matched percentage
 3330 of 94.1%. This high match is likely due to Google providing both Google Cloud
 3331 Vision and the Open Images Dataset—it is likely that they are trained on the same
 3332 data and labels. An endpoint with higher matched percentage has a more similar
 3333 label set to the Open Images Dataset classes. However, a higher matched percentage
 3334 does not mean imply *better quality* of an API endpoint; it will increase apparent
 3335 precision, recall, and F-measure only.

3336 The true and false positive (TP/FP) label averages and the TP/FP ratio is shown
 3337 in Table 8.7. Where the TP/FP ratio is larger, the scores are more reliable, however
 3338 it is possible to increase the TP/FP ratio by adding more false labels with low scores.
 3339 On the other hand, it is impossible to increase F-measure intentionally, because
 3340 increasing precision will decrease recall, and vice versa. Hence, the importance of
 3341 the F-measure statistic is critical for our analysis.

3342 Let R_A , R_G , and R_M be responses from Amazon Rekognition, Google Cloud
 3343 Vision, and Microsoft’s Azure Computer Vision, respectively. There are four sets of
 3344 operands, i.e., (R_A, R_G) , (R_G, R_M) , (R_M, R_A) , and (R_A, R_G, R_M) . Table 8.7 shows
 3345 the evaluation of each operands set, Table 8.8 shows the averages of the four operands
 3346 sets, and Figure 8.6 shows the comparison of F-measure for each methods.

3347 8.5.2 Naive Operators

3348 Results of *min*, *max*, and *average* operators are shown in Tables 8.7 and 8.8 and Fig-
 3349 ure 8.6. The *min* operator is similar to *union* operator of set operation, and outputs
 3350 all labels of operands. The precision of the *min* operator is always greater than any
 3351 precision of operands, and the recall is always lesser than any precision of operands.
 3352 *Max* and *average* operators are similar to *intersection* operator of set operations.
 3353 Both operators output intersection of labels of operands and there is no clear relation
 3354 to the precision and recall of operands. Since both operators have the same preci-
 3355 sion, recall, and F-measure, Figure 8.6 groups them into one. The *average* operator
 3356 performs well on the TP/FP ratio, where most of the same labels from multiple
 3357 endpoints are TPs. In many cases of the four operand sets, all naive operators’

Table 8.6: Matching to human-verified labels.

Endpoint	Total	Unique	Matched	Matched %
Amazon Rekognition	11,416	1,409	1,111	78.9
Google Cloud Vision	8,766	2,644	2,487	94.1
Azure Computer Vision	5,392	746	470	63.0

Table 8.7: Evaluation results. A = Amazon Rekognition, G = Google Cloud Vision, M = Microsoft’s Azure Computer Vision.

Operands	Operator	Precision	Recall	F-measure	TP average	FP average	TP/FP ratio
A		0.217	0.282	0.246	0.848 ± 0.165	0.695 ± 0.185	1.220
G		0.474	0.465	0.469	0.834 ± 0.121	0.741 ± 0.132	1.126
M		0.263	0.164	0.202	0.858 ± 0.217	0.716 ± 0.306	1.198
A, G	Min	0.771	0.194	0.310	0.805 ± 0.142	0.673 ± 0.141	1.197
A, G	Max	0.280	0.572	0.376	0.850 ± 0.136	0.712 ± 0.171	1.193
A, G	Average	0.280	0.572	0.376	0.546 ± 0.225	0.368 ± 0.114	1.485
A, G	D’Hondt	0.350	0.389	0.369	0.713 ± 0.249	0.518 ± 0.202	1.377
A, G	Hare-Niemeyer	0.344	0.384	0.363	0.723 ± 0.242	0.527 ± 0.199	1.371
A, G	Proposal	0.380	0.423	0.401	0.706 ± 0.239	0.559 ± 0.190	1.262
G, M	Min	0.789	0.142	0.240	0.794 ± 0.209	0.726 ± 0.210	1.093
G, M	Max	0.357	0.521	0.424	0.749 ± 0.135	0.729 ± 0.231	1.165
G, M	Average	0.357	0.521	0.424	0.504 ± 0.201	0.375 ± 0.141	1.342
G, M	D’Hondt	0.444	0.344	0.388	0.696 ± 0.250	0.551 ± 0.254	1.262
G, M	Hare-Niemeyer	0.477	0.375	0.420	0.696 ± 0.242	0.591 ± 0.226	1.179
G, M	Proposal	0.414	0.424	0.419	0.682 ± 0.238	0.597 ± 0.209	1.143
M, A	Min	0.693	0.143	0.237	0.822 ± 0.201	0.664 ± 0.242	1.239
M, A	Max	0.185	0.318	0.234	0.863 ± 0.178	0.703 ± 0.229	1.228
M, A	Average	0.185	0.318	0.234	0.589 ± 0.262	0.364 ± 0.144	1.616
M, A	D’Hondt	0.271	0.254	0.262	0.737 ± 0.261	0.527 ± 0.223	1.397
M, A	Hare-Niemeyer	0.260	0.245	0.253	0.755 ± 0.251	0.538 ± 0.218	1.402
M, A	Proposal	0.257	0.242	0.250	0.769 ± 0.244	0.571 ± 0.205	1.337
A, G, M	Min	0.866	0.126	0.220	0.774 ± 0.196	0.644 ± 0.219	1.202
A, G, M	Max	0.241	0.587	0.342	0.857 ± 0.142	0.714 ± 0.210	1.201
A, G, M	Average	0.241	0.587	0.342	0.432 ± 0.233	0.253 ± 0.106	1.712
A, G, M	D’Hondt	0.375	0.352	0.363	0.678 ± 0.266	0.455 ± 0.208	1.492
A, G, M	Hare-Niemeyer	0.362	0.340	0.351	0.693 ± 0.260	0.444 ± 0.216	1.559
A, G, M	Proposal	0.380	0.357	0.368	0.684 ± 0.259	0.484 ± 0.200	1.414

Table 8.8: Average of the evaluation result.

Operator	Precision	Recall	F-measure	TP/FP ratio
Min	0.780	0.151	0.252	1.183
Max	0.266	0.500	0.344	1.197
Average	0.266	0.500	0.344	1.539
D’Hondt	0.361	0.335	0.346	1.382
Hare-Niemeyer	0.361	0.336	0.347	1.378
Proposal	0.358	0.362	0.360	1.289

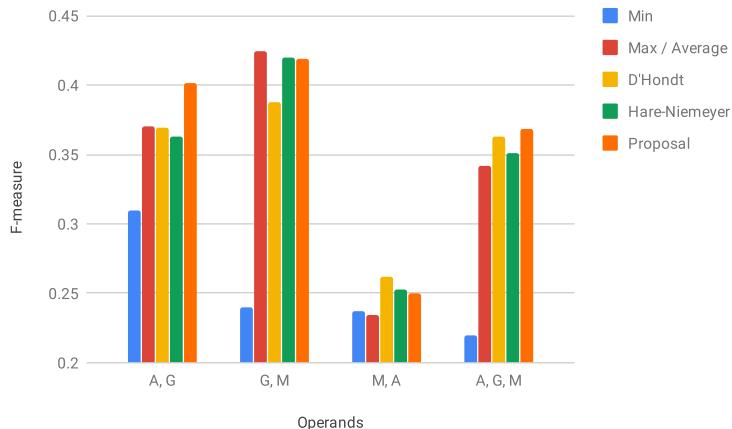


Figure 8.6: F-measure comparison.

3358 F-measures are between F-measures of operands. None of naive operators therefore
 3359 improve results by merging responses from multiple endpoints.

3360 **8.5.3 Traditional Proportional Representation Operators**

3361 There are many existing allocation algorithms in proportional representation, e.g.,
 3362 the Niemeyer and Niemeyer method [207]. These methods may be replacements of
 3363 those in Section 8.4.3. Other steps, i.e., Sections 8.4.1, 8.4.2 and 8.4.4, are the same
 3364 as for our proposed technique. Tables 8.7 and 8.8 and Figure 8.6 show the result of
 3365 these traditional proportional representation algorithms. Averages of F-measures by
 3366 traditional proportional representation operators are almost equal to that of the *max*
 3367 and *average* operators. It is worth noting that merging *M* and *A* responses results in
 3368 a better F-measure than each F-measure of *M* and *A* individually. As these are not
 3369 biased to human-verified labels, situations in the real-world usage should, therefore,
 3370 be similar to the case of *M* and *A*. Hence, RQ1 is true.

3371 **8.5.4 New Proposed Label Merge Technique**

3372 As shown in Table 8.8, our proposed new method performs best in F-measure.
 3373 Instead, the TP/FP ratio is less than *average*, the D'Hondt method, and Hare-
 3374 Niemeyer method. As described in Section 8.5.1, we argue that F-measure is a
 3375 more important measure than the TP/FP ratio (in this case). Therefore, RQ2 is
 3376 true. Shown in Table 8.7, our proposed new method improves the results when
 3377 merging *M* and *A* in non-biased endpoints. It is similar to traditional proportional
 3378 representation operators, but does not perform as well. However, it performs better
 3379 on other operand sets, and performs best overall as shown in Figure 8.6.

3380 **8.5.5 Performance**

3381 We used AWS EC2 m5.large instance (2 vCPUs, 2.5 GHz Intel Xeon, 8 GiB RAM);
 3382 Amazon Linux 2 AMI (HVM), SSD Volume Type; Node.js 8.12.0. It takes 0.370

seconds to merge responses from three endpoints. Computational complexity of the algorithm in Section 8.4.3 is $O(n^2)$, where n is total number of labels in responses. (The estimation assumes that the number of endpoints is a constant.) Complexity of Step I in Section 8.4.3 is $O(n \log n)$, as the worst case is that all n labels are from one single endpoint and all n labels are in one CC. Complexity of Step II to Step V is $O(n^2)$, as the number of CCs is less than or equal to n and number of iterations are less than or equal to n . As Table 8.1 shows, the averaged total number of three endpoints is 25.58. Most of time for merging is consumed by looking up WordNet synsets (Section 8.4.1). The API facade calls each APIs on actual endpoints in parallel. It takes about 5 seconds, which is much longer than 0.370 seconds taken for the merging of responses.

8.6 Conclusions and Future Work

In this paper, we propose a method to merge responses from CVSs. Our method merges API responses better than naive operators and other proportional representation methods (i.e., D’Hondt and Hare-Niemeyer). The average of F-measure of our method marks 0.360; the next best method, Hare-Niemeyer, marks 0.347. Our method and other proportional representation methods are able to improve the F-measure from original responses in some cases. Merging non-biased responses results in an F-measure of 0.250, while original responses have an F-measure between 0.246 and 0.242. Therefore, users can improve their applications’ precision with small modification, i.e., by switching from a singular URL endpoint to a facade-based architecture. The performance impact by applying facades is small, because overhead in facades is much smaller than API invocation. Our proposal method conforms identity, commutativity, reflexivity, and additivity properties and these properties are advisable for integrating multiple responses.

Our idea of a proportional representation approach can be applied to other IWSs. If the response of such a service is list consisting of an entity and score, and if there is a way to group entities, a proposal algorithm can be applied. The opposite approach is to improve results by inferring labels. Our current approach picks some of the labels returned by endpoints. IWSs are not only based on supervised ML—thus to cover a wide range of IWSs, it is necessary to classify and analyse each APIs and establish a method to improve results by merging. Currently graph structures of labels and synsets (Figure 8.2) are not considered when merging results. Propagating scores from labels could be used, losing the additivity property but improving results for users. There are many ways to propagate scores. For instance, setting propagation factors for each link type would improve merging and could be customised for users’ preferences. It would be possible to generate an API facade automatically. APIs with the same functionality have same or similar signatures. Machine-readable API documentation, for instance, OpenAPI Specification, could help a generator to build an API facade.

3423 CHAPTER 9

3424

3425 Threshy: Supporting Safe Usage of Intelligent Web Services[†]

3426

3427 **Abstract** Increased popularity of ‘intelligent’ web services provides end-users with machine-
3428 learnt functionality at little effort to developers. However, these services require a decision
3429 threshold to be set which is dependent on problem-specific data. Developers lack a systematic
3430 approach for evaluating intelligent services and existing evaluation tools are predominantly
3431 targeted at data scientists for pre-development evaluation. This paper presents a workflow
3432 and supporting tool, Threshy, to help *software developers* select a decision threshold suited
3433 to their problem domain. Unlike existing tools, Threshy is designed to operate in multiple
3434 workflows including pre-development, pre-release, and support. Threshold configuration
3435 files exported by Threshy can be integrated into client applications and monitoring infras-
3436 tructure. Demo: <https://bit.ly/2YKeYhE>.

3437

9.1 Introduction

3438 Machine learning algorithm adoption is increasing in modern software. End users
3439 routinely benefit from machine-learnt functionality through personalised recom-
3440 mendations [63], voice-user interfaces [202], and intelligent digital assistants [38].
3441 The easy accessibility and availability of intelligent web services¹ is contributing to
3442 their adoption. These intelligent web services simplify the development of machine
3443 learning solutions as they (i) do not require specialised machine learning expertise
3444 to build and maintain, (ii) abstract away infrastructure related issues associated with
3445 machine learning [11, 254], and (iii) provide web APIs for ease of integration.

3446 However, unlike traditional web services, the functionality of these *intelligent*

[†]This chapter is originally based on A. Cummaudo, S. Barnett, R. Vasa, and J. Grundy, “Threshy: Supporting Safe Usage of Intelligent Web Services,” Seoul, Republic of Korea, 2020, Unpublished. Terminology has been updated to fit this thesis.

¹Such as Azure Computer Vision (<https://azure.microsoft.com/en-au/services/cognitive-services/computer-vision/>), Google Cloud Vision (<https://cloud.google.com/vision/>), or Amazon Rekognition (<https://aws.amazon.com/rekognition/>).

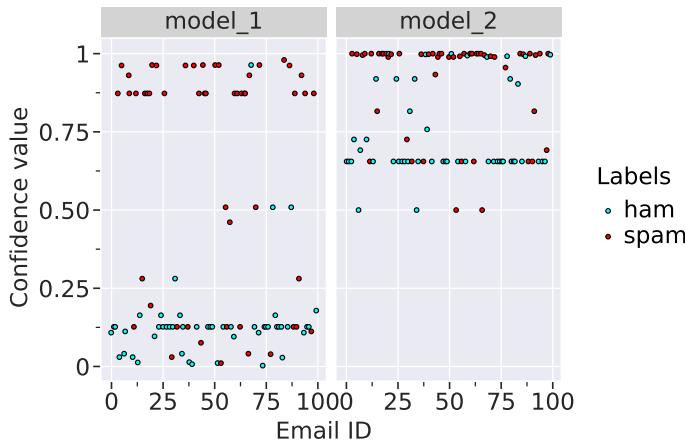


Figure 9.1: Predictions for 100 emails from two spam classifiers. Decision thresholds are classifier-dependent: a single threshold for both classifiers is *not* appropriate as ham emails are clustered at 0.12 (model_1) and at 0.65 (model_2). Developers must evaluate performance for *both* thresholds.

3447 *services* is dependent on a set of assumptions unique to machine learning [69].
 3448 These assumptions are based on the data used to train machine learning algorithms,
 3449 the choice of algorithm, and the choice of data processing steps—most of which
 3450 are not documented. For developers, these assumptions mean that the performance
 3451 characteristics of an intelligent service in any particular application problem domain
 3452 is not fully knowable. Intelligent services represent this uncertainty through a
 3453 confidence value associated with their predictions. Thus an evaluation procedure
 3454 must be followed as a part of using an intelligent service for an application.

3455 A typical evaluation process would involve a test data set (curated by the devel-
 3456 opers using the intelligent service) that is used to determine an appropriate threshold.
 3457 Choice of a decision threshold is a critical element of the evaluation procedure [118].
 3458 This is especially true for classification problems such as detecting if an image con-
 3459 tains cancer or identifying all of the topics in a document. Simple approaches
 3460 to selecting a threshold are often insufficient, as highlighted in Google’s machine
 3461 learning course: “*It is tempting to assume that [a] classification threshold should
 3462 always be 0.5, but thresholds are problem-dependent, and are therefore values that
 3463 you must tune.*”² As an example consider the predictions from two email spam
 3464 classifiers shown in Figure 9.1. The predicted safe emails, ‘ham’, are in two separate
 3465 clusters (a simple threshold set to approx. 0.2 for model 1 and 0.65 for model 2),
 3466 indicating that different decision thresholds may be required depending on the clas-
 3467 sifier. Also note that some emails have been misclassified; how many depends on
 3468 the choice of decision threshold. An appropriate threshold considers factors outside
 3469 algorithmic performance, such as financial cost and impact of wrong decisions. To
 3470 select an appropriate decision threshold, developers using intelligent services need
 3471 approaches to reason about and consider trade-offs between competing *cost fac-
 3472 tors*. These include impact, financial costs, and maintenance implications. Without

²See <https://bit.ly/36oMgWb>.

3473 considering these trade-offs, sub-optimal decision thresholds will be selected.

3474 The standard approach for tuning thresholds in classification problems involve
3475 making trade-offs between the number of false positives and false negatives using
3476 the receiver operating characteristic (ROC) curve. However, developers (i) need
3477 to realise that this trade-off between false positives and false negatives is a data
3478 dependent optimisation process [253], (ii) often need to develop custom scripts
3479 and follow a trial-and-error based approach to determine a threshold, (iii) must
3480 have appropriate statistical training and expertise, and (iv) be aware that multi-
3481 label classification require more complex optimisation methods when setting label
3482 specific costs. However, current intelligent services do not sufficiently guide or
3483 support software engineers through the evaluation process, nor do they make this
3484 need clear in the documentation.

3485 In this paper we present **Threshy**³, a tool to assist developers in selecting decision
3486 thresholds when using intelligent services. The motivation for developing Threshy
3487 arose from our consultancy work with industry. Unlike existing tooling (see Sec-
3488 tion 9.4), **Threshy serves as a means to up-skill and educate software engineers**
3489 **in selecting machine-learnt decision thresholds**, for example, on aspects such as
3490 confusion matrices. Threshy provides a visually interactive interface for developers
3491 to fine-tune thresholds and explore trade-offs of prediction hits/misses. This exposes
3492 the need for optimisation of thresholds, which is dependent on particular use cases.

3493 Threshy improves developer productivity through automation of the threshold
3494 selection process by leveraging an optimisation algorithm to propose thresholds.
3495 The algorithm considers different cost factors providing developers with summary
3496 information so they can make more informed trade-offs. Developers also benefit
3497 from the workflow implemented in Threshy by providing a reproducible procedure
3498 for testing and tuning thresholds for any category of classification problem (binary,
3499 multi-class, and multi-label). Threshy has also been designed to work for different
3500 input data types including images, text and categorical values. The output, is a
3501 text file and can be integrated into client applications ensuring that the thresholds
3502 can be updated without code changes (if needed), and continuously monitored in a
3503 production setting.

3504 9.2 Motivating Example

3505 As a motivating example consider Nina, a fictitious developer, who has been em-
3506 ployed by Lucy’s Tomato Farm to automate the picking of tomatoes from their vines
3507 (when ripe) using computer vision and a harvesting robot. Lucy’s Farm grow five
3508 types of tomatoes (roma, cherry, plum, green, and yellow tomatoes). Nina’s robot—
3509 using an attached webcam—will crawl and take a photo of each vine to assess it
3510 for harvesting. Nina’s automated harvester needs to sort picked tomatoes into a
3511 respective container, and thus several business rules need to be encoded into the
3512 prediction logic to sort each tomato detected based on its *ripeness* (ripe or not ripe)
3513 and *type of tomato* (as above).

³Threshy is available for use at <http://bit.ly/a2i2threshy>.

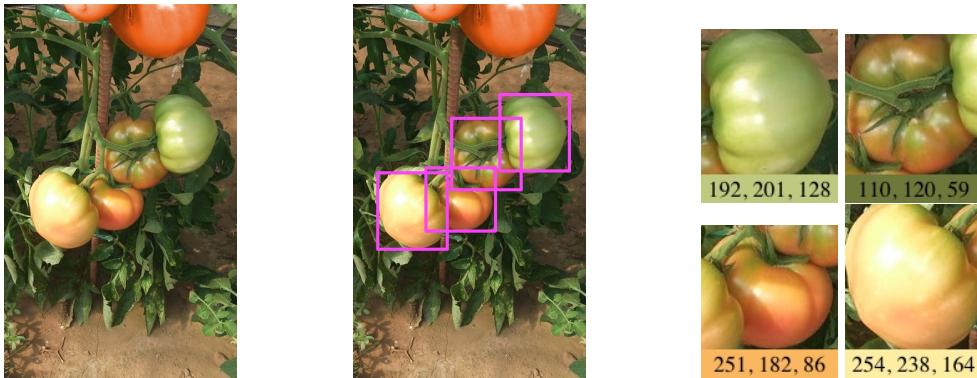


Figure 9.2: Pipeline of Nina’s harvesting robot. *Left:* Photo from harvesting robot’s webcam. *Centre:* Classification detecting different types of tomatoes. *Right:* Binary classification for ripeness (ripe/unripe) based on (R, G, B values).

3514 Nina uses a two-stage pipeline consisting of a multi-class and a binary classi-
 3515 fication model. She has decided to evaluate the viability of cloud based intelligent
 3516 services and use them if operationally effective. Figure 9.2 illustrates an example of
 3517 the the pipeline as listed below:

- 3518 1. **Classify tomato ‘type’.** This stage uses an object localisation service to detect
 3519 all tomato-like objects in the frame and classifies each tomato into one of the
 3520 following labels: [‘roma’, ‘cherry’, ‘plum’, ‘green’, ‘yellow’].
- 3521 2. **Assess tomato ‘ripeness’.** This stage uses a crop of the localised tomatoes
 3522 from the original frame to assess the crop’s colour properties (i.e., average
 3523 colour must have $R > 200$ and $G < 240$). This produces a binary classification
 3524 to deduce whether the tomato is ripe or not.

3525 Nina only has a minimal appreciation of the evaluation method to use for off-
 3526 the-shelf computer vision (classification) services. She also needs to consider the
 3527 financial costs of mis-classifying either the tomato type or the ripeness. Missing a
 3528 few ripe tomatoes isn’t a problem as the robot travels the field twice a week during
 3529 harvest season. However, picking an unripe tomato is expensive as Lucy cannot sell
 3530 them. Therefore, Nina needs a better (automated) way to assess the performance
 3531 of the service and set optimal thresholds for her picking robot, thereby maximising
 3532 profit.

3533 To assist in developing Nina’s pipeline, Lucy sampled a section of 1000 tomatoes
 3534 by taking a photo of each tomato, labelling its type, and assessing whether the vine
 3535 was ‘ripe’ or ‘not_ripe’. Nina ran the labelled images through an intelligent
 3536 service, with each image having a predicted type (multi-class) and ripeness (binary),
 3537 with respective confidence values.

3538 Nina combined the predictions, their respective confidence values, and Lucy’s
 3539 labelled ground truths into a CSV file which was then uploaded to Threshy. Nina
 3540 asked Lucy to assist in setting relevant costs for correct predictions and false predic-
 3541 tions. Threshy then recommended a choice of decision threshold which Nina then
 3542 fine tuned while considering the performance and cost implications.

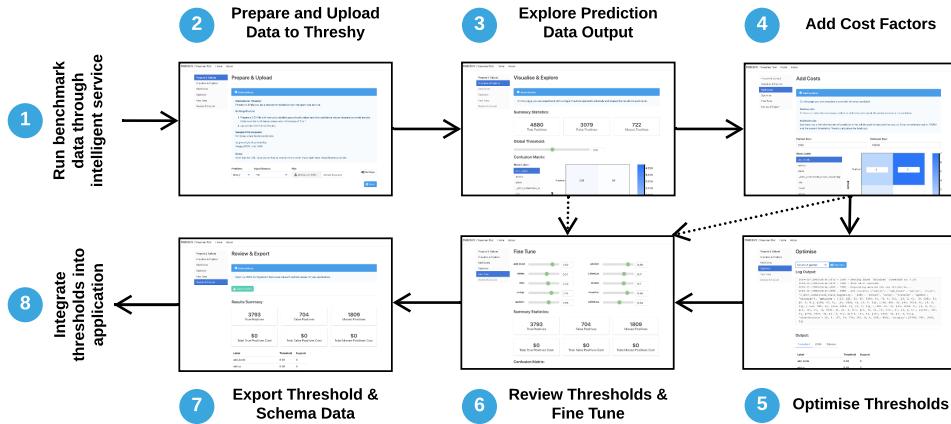


Figure 9.3: UI workflow for interacting with Threshy to optimise the thresholds for classification problem.

9.3 Threshy

Threshy is a tool to assist software engineers with setting decision thresholds when integrating machine-learnt components in a system. Our tool also serves as a method to inform and educate engineers about the nuances to consider. The novel features of Threshy are:

- Automating threshold selection using an optimisation algorithm (NSGA-II [76]), optimising the results for each label.
- Support for additional user defined weights when optimising thresholds such as financial costs and impact to society (different type of cost). This allows decision thresholds to be set within a business context as they differ from application to application [86].
- Handles nuances of classification problems such as dealing with multi-objective optimisation, and metric selection—reducing errors of omission.
- Support key classification problems including binary (e.g. email is either spam or ham), multi-class (e.g. predicting the colour of a car), and multi-label (e.g. assign multiple topics to a document). Existing tools ignore multi-label classification.

Setting thresholds in Threshy is an eight step process as shown in Figure 9.3. Software engineers ① run a benchmark dataset through the machine-learnt component to create a CSV file with true labels and predicted labels along with the predicted confidence values. The CSV file is then ② uploaded for initial exploration where engineers can ③ experiment with modifying a single global threshold for the dataset. Developers may choose to exit at this point (as indicated by dotted arrows in Figure 9.3). Optionally, the engineer ④ defines costs for missed predictions followed by selecting optimisation settings. The optional optimisation step of Threshy ⑤ considers the performance and costs when deriving the thresholds. Finally, the engineer can ⑥ review and fine tune the calculated thresholds, associated

3570 costs, and (7) download generated threshold meta-data to be (8) integrated into their
3571 application.

3572 Threshy runs a client/server architecture with a thin-client (see Figure 9.4). The
3573 web-based application consists of an interactive front-end where developers upload
3574 benchmark results—consisting of both human annotated labels (ground truths) and
3575 machine predictions (from the intelligent service)—and use threshold tuners (via
3576 sliders) to present a data summary of the uploaded CSV. Predicted performances
3577 and costs are entered manually into the web interface by the developer. The back-end
3578 of Threshy asynchronously runs a data analyser, cost processor and metrics calculator
3579 when relevant changes are made to the front-end’s tuning sliders. Separating the
3580 two concerns allows for high intensity processing to be done on the server and not
3581 the front end.

3582 The data analyser provides a comprehensive overview of confusion matrices
3583 compatible for multi-label multi-class classification problems. When representing
3584 the confusion matrix, it is trivial to represent instances where multi-label multi-
3585 classification is not considered. For example, in the simplest case, a single row in
3586 the matrix represents a single label out of two classes, or each row has one label but
3587 it has multiple classes. However, a more challenging case to visualise the confusion
3588 arises when you have n labels and n classes; the true/false matches become too
3589 excessive to visualise as it is disproportionate to the true results. To deal with this
3590 issue, we condense the summary statistics down to three constructs: (i) number of
3591 true positives, (ii) false positives, (iii) missed positives. This therefore allows us to
3592 optimise against the true positives and minimise the other two constructs.

3593 Threshy is a fully self-contained repository containing implementation of the
3594 tool, scripting and exploratory notebooks, which we make available at <https://github.com/a2i2/threshy>.

3596 9.4 Related work

3597 Optimal machine-learnt decision boundaries depend on identifying the operating
3598 conditions of the problem domain. A systematic study by Drummond and Holte
3599 [86] classifies four such operating conditions to determine a decision threshold: (i)
3600 the operating condition is known and thus the model trained matches perfectly; (ii)
3601 where the operating conditions are known but change with time, and thus the model
3602 must be adaptable to such changes; (iii) where there is uncertainty in the knowledge
3603 of the operating conditions certain changes in the operating condition are more likely
3604 than others; (iv) where there is no knowledge of the operating conditions and the
3605 conditions may change from the model in any possible way. Various approaches
3606 to determine appropriate thresholds exist for all four of these cases, such as cost-
3607 sensitive learning, ROC analysis, cost curves, and Brier scores.

3608 However, an *automated* attempt to calibrate decision threshold boundaries is
3609 not considered, and is largely pitched at a non-software engineering audience. A
3610 more recent study touches on this in model management for large-scale adversarial
3611 instances in Google’s advertising system [253], however this is only a single com-
3612 ponent within the entire architecture, and is not a tool that is useful for developer’s

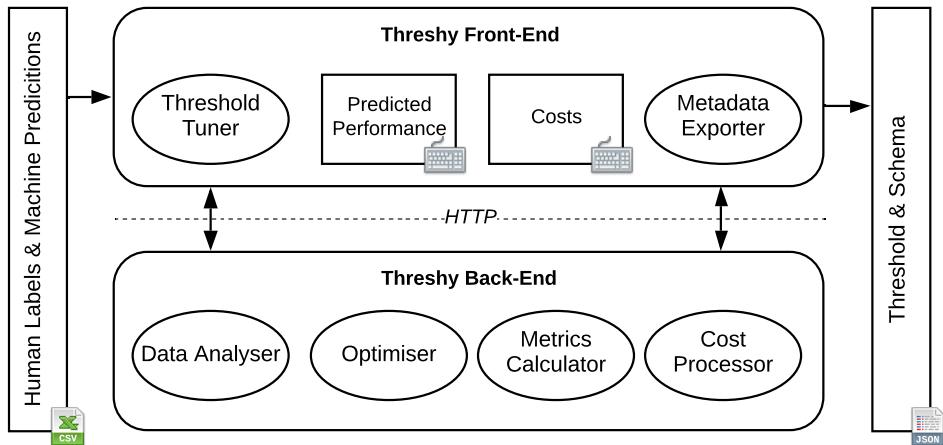


Figure 9.4: Architecture of Threshy.

3613 in varying contexts. Unlike this study, our work presents a ‘plug-and-play’ style
 3614 calibration method where any context/domain can have thresholds automatically
 3615 calibrated (in-context) *and* optimised for engineers; Threshy’s architecture and
 3616 design facilitates operating in a headless mode enabling use in monitoring and support
 3617 workflows.

3618 Support tools for ML frameworks generally fall into two categories; the first
 3619 attempts to illuminate the ‘black box’ by offering ways in which developers can better
 3620 understand the internals of the model to improve its performance. (For extensive
 3621 analyses and surveys into this area, see [125, 219].) However, a recent emphasis to
 3622 probe only inputs and outputs of a model has been explored, exploring off-the-shelf
 3623 models without knowledge of its unknowns (see Figure 9.1) to reflect the nature
 3624 of real-world development. Google’s *What-If Tool* [297] for Tensorflow provides a
 3625 means for data scientists to visualise, measure and assess model performance and
 3626 fairness with various hypothetical scenarios and data features; similarly, Microsoft’s
 3627 *Gamut* tool [124] provides an interface to test hypotheticals (although only on
 3628 Generalized Additive Models) and their *ModelTracker* tool [9] collates summary
 3629 statistics on a set of sample data to enable rich visualisation of model behaviour and
 3630 access to key performance metrics.

3631 However, these tools are largely focused toward pre-development model eval-
 3632 uation and are not designed for the software engineering workflow. They are also
 3633 targeted to data scientists and not engineers, and certain tools are tied to specific
 3634 machine learning frameworks (e.g., What-If and Tensorflow). Our work attempts to
 3635 bridge these gaps through a structured workflow with an automated tool targeted to
 3636 software developers. We also consider the need to have a consistent tool that works
 3637 across development, test, and production environments.

3638 **9.5 Conclusions & Future Work**

3639 Primary contributions of this work include Threshy, a tool for automating threshold
 3640 selection, and the overall meta-workflow proposed in Threshy that developers can

³⁶⁴¹ use as a point of reference for calibrating thresholds. In future work, we plan to
³⁶⁴² evaluate Threshy with software engineers to identify additional insights required to
³⁶⁴³ make decision thresholds in practice and add code synthesis for monitoring concept
³⁶⁴⁴ drift and for implementing decision thresholds.

CHAPTER 10

3645

3646

3647

3648

FSE Paper[†]

[†]This chapter is originally based on A. Cummaudo, “ESEC/FSE Paper,” Unpublished. Terminology has been updated to fit this thesis.

3649

Part III

3650

Postface

CHAPTER 11

3651

3652

3653

3654

Conclusions & Future Work

References

- [1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: A system for large-scale machine learning,” in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation*. Savannah, GA, USA: ACM, 2016. ISBN 978-1-93-197133-1 pp. 265–283.
- [2] E. Aghajani, C. Nagy, G. Bavota, and M. Lanza, “A Large-scale empirical study on linguistic antipatterns affecting apis,” in *Proceedings of the 34th International Conference on Software Maintenance and Evolution*. Madrid, Spain: IEEE, September 2018. DOI 10.1109/IC-SME.2018.00012. ISBN 978-1-53-867870-1 pp. 25–35.
- [3] E. Aghajani, C. Nagy, O. L. Vega-Marquez, M. Linares-Vasquez, L. Moreno, G. Bavota, and M. Lanza, “Software Documentation Issues Unveiled,” in *Proceedings of the 41st International Conference on Software Engineering*. Montreal, QC, Canada: IEEE, May 2019. DOI 10.1109/ICSE.2019.00122. ISBN 978-1-72-810869-8. ISSN 0270-5257 pp. 1199–1210.
- [4] M. Ahazanuzzaman, M. Asaduzzaman, C. K. Roy, and K. A. Schneider, “Classifying stack overflow posts on API issues,” in *Proceedings of the 25th International Conference on Software Analysis, Evolution and Reengineering*. Campobasso, Italy: IEEE, March 2018. DOI 10.1109/SANER.2018.8330213. ISBN 978-1-53-864969-5 pp. 244–254.
- [5] R. E. Al-Qutaish, “Quality Models in Software Engineering Literature: An Analytical and Comparative Study,” *Journal of American Science*, vol. 6, no. 3, pp. 166–175, 2010.
- [6] H. Allahyari and N. Lavesson, “User-oriented assessment of classification model understandability,” in *Proceedings of the 11th Scandinavian Conference on Artificial Intelligence*, vol. 227. Trondheim, Norway: IOS Press, May 2011. DOI 10.3233/978-1-60750-754-3-11. ISBN 978-1-60-750753-6. ISSN 0922-6389 pp. 11–19.
- [7] M. Allamanis and C. Sutton, “Why, when, and what: Analyzing stack overflow questions by topic, type, and code,” in *Proceedings of the 10th IEEE International Working Conference on Mining Software Repositories*. San Francisco, CA, USA: IEEE, May 2013. DOI 10.1109/MSR.2013.6624004. ISBN 978-1-46-732936-1. ISSN 2160-1852 pp. 53–56.
- [8] J. Alway and C. Calhoun, *Critical Social Theory: Culture, History, and the Challenge of Difference*. American Sociological Association, 1997, vol. 26, no. 1, DOI 10.2307/2076647.
- [9] S. Amershi, M. Chickering, S. M. Drucker, B. Lee, P. Simard, and J. Suh, “Modeltracker: Redesigning performance analysis tools for machine learning,” in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. Seoul, Republic of Korea: ACM, April 2015. DOI 10.1145/2702123.2702509. ISBN 978-1-45-033145-6 pp. 337–346.
- [10] K. Arnold, “Programmers are People, Too,” *ACM Queue*, vol. 3, no. 5, pp. 54–59, 2005, DOI 10.1145/1071713.1071731. ISSN 1542-7749

- [11] A. Arpteg, B. Brinne, L. Crnkovic-Friis, and J. Bosch, "Software engineering challenges of deep learning," in *Proceedings of the 44th Euromicro Conference on Software Engineering and Advanced Applications*. Prague, Czech Republic: IEEE, August 2018. DOI 10.1109/SEAA.2018.00018. ISBN 978-1-53-867382-9 pp. 50–59.
- [12] W. R. Ashby and J. R. Pierce, "An Introduction to Cybernetics," *Physics Today*, vol. 10, no. 7, pp. 34–36, July 1957.
- [13] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, and K. R. Müller, "How to explain individual classification decisions," *Journal of Machine Learning Research*, vol. 11, pp. 1803–1831, 2010. ISSN 1532-4435
- [14] B. Baesens, C. Mues, M. De Backer, J. Vanthienen, and R. Setiono, "Building intelligent credit scoring systems using decision tables," in *Proceedings of the 5th International Conference on Enterprise Information Systems*, vol. 2. Angers, France: IEEE, April 2003. DOI 10.1007/1-4020-2673-0_15. ISBN 9-72-988161-8 pp. 19–25.
- [15] X. Bai, Y. Wang, G. Dai, W. T. Tsai, and Y. Chen, "A framework for contract-based collaborative verification and validation of Web services," in *Proceedings of the 10th International Symposium of Component-Based Software Engineering*. Medford, MA, USA: Springer, July 2007. DOI 10.1007/978-3-540-73551-9_18. ISBN 978-3-54-073550-2. ISSN 0302-9743 pp. 258–273.
- [16] K. Bajaj, K. Pattabiraman, and A. Mesbah, "Mining questions asked by web developers," in *Proceedings of the 11th Working Conference on Mining Software Repositories*. Hyderabad, India: ACM, May 2014. DOI 10.1145/2597073.2597083. ISBN 978-1-45-032863-0 pp. 112–121.
- [17] K. Ballinger, "Simplicity and Utility, or, Why SOAP Lost," [Online] Available: <http://bit.ly/37vLms0>, December 2014, Accessed: 28 August 2018.
- [18] S. Barnett, "Extracting technical domain knowledge to improve software architecture," Ph.D. dissertation, Swinburne University of Technology, Hawthorn, Australia, 2018.
- [19] S. Barnett, R. Vasa, and J. Grundy, "Bootstrapping Mobile App Development," in *Proceedings of the 37th International Conference on Software Engineering*. Florence, Italy: IEEE, May 2015. DOI 10.1109/ICSE.2015.216. ISBN 978-1-47-991934-5. ISSN 0270-5257 pp. 657–660.
- [20] S. Barnett, R. Vasa, and A. Tang, "A Conceptual Model for Architecting Mobile Applications," in *Proceedings of the 12th Working IEEE/IFIP Conference on Software Architecture*. Montreal, QC, Canada: IEEE, May 2015. DOI 10.1109/WICSA.2015.28. ISBN 978-1-47-991922-2 pp. 105–114.
- [21] A. Barua, S. W. Thomas, and A. E. Hassan, "What are developers talking about? An analysis of topics and trends in Stack Overflow," *Empirical Software Engineering*, vol. 19, no. 3, pp. 619–654, 2014, DOI 10.1007/s10664-012-9231-y. ISSN 1573-7616
- [22] O. Barzilay, C. Treude, and A. Zagalsky, "Facilitating crowd sourced software engineering via stack overflow," in *Finding Source Code on the Web for Remix and Reuse*, 2014, no. 4, pp. 289–308. ISBN 978-1-46-146596-6
- [23] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 2nd ed. Addison-Wesley, 2003. ISBN 0-32-115495-9
- [24] B. E. Bejnordi, M. Veta, P. J. Van Diest, B. Van Ginneken, N. Karssemeijer, G. Litjens, J. A. W. M. Van Der Laak, M. Hermsen, Q. F. Manson, M. Balkenhol, O. Geessink, N. Stathonikos, M. C. R. F. Van Dijk, P. Bult, F. Beca, A. H. Beck, D. Wang, A. Khosla, R. Gargeya, H. Irshad, A. Zhong, Q. Dou, Q. Li, H. Chen, H. J. Lin, P. A. Heng, C. Haß, E. Bruni, Q. Wong, U. Halici, M. Ü. Öner, R. Cetin-Atalay, M. Berseth, V. Khvatkov, A. Vylegzhannin, O. Kraus, M. Shaban, N. Rajpoot, R. Awan, K. Sirinukunwattana, T. Qaiser, Y. W. Tsang, D. Tellez, J. Annuscheit, P. Hufnagl, M. Valkonen, K. Kartasalo, L. Latonen, P. Ruusuviuri, K. Liimatainen, S. Albarqouni, B. Mungal, A. George, S. Demirci, N. Navab, S. Watanabe, S. Seno, Y. Takenaka, H. Matsuda, H. A. Phoulady, V. Kovalev, A. Kalinovsky, V. Liauchuk, G. Bueno, M. M. Fernandez-Carrobles, I. Serrano, O. Deniz, D. Racoceanu, and R. Venâncio, "Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer," *Journal of the American Medical Association*, vol. 318, no. 22, pp. 2199–2210, December 2017, DOI 10.1001/jama.2017.14585. ISSN 1538-3598

- 3749 [25] R. Bellazzi and B. Zupan, "Predictive data mining in clinical medicine: Current issues and
3750 guidelines," *International Journal of Medical Informatics*, vol. 77, no. 2, pp. 81–97, 2008,
3751 DOI 10.1016/j.ijmedinf.2006.11.006. ISSN 1386-5056
- 3752 [26] A. Ben-David, "Monotonicity Maintenance in Information-Theoretic Machine Learning Algo-
3753 rithms," *Machine Learning*, vol. 19, no. 1, pp. 29–43, 1995, DOI 10.1023/A:1022655006810.
3754 ISSN 1573-0565
- 3755 [27] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform resource identifier (URI): Generic
3756 syntax," Tech. Rep., 2004.
- 3757 [28] L. L. Berry, A. Parasuraman, and V. A. Zeithaml, "SERVQUAL: A multiple-item scale for
3758 measuring consumer perceptions of service quality," *Journal of Retailing*, vol. 64, no. 1, pp.
3759 12–40, 1988, DOI 10.1016/S0148-2963(99)00084-3. ISBN 00224359. ISSN 0022-4359
- 3760 [29] J. Bessin, "The Business Value of Quality," [Online] Available: <https://ibm.co/2u0UDK0>, June
3761 2004.
- 3762 [30] S. Beyer and M. Pinzger, "A manual categorization of android app development issues on stack
3763 overflow," in *Proceedings of the 30th International Conference on Software Maintenance and
3764 Evolution*. Victoria, BC, Canada: IEEE, September 2014. DOI 10.1109/ICSME.2014.88.
3765 ISBN 978-0-76-955303-0 pp. 531–535.
- 3766 [31] S. Beyer, C. MacHo, M. Pinzger, and M. Di Penta, "Automatically classifying posts into question
3767 categories on stack overflow," in *Proceedings of the 26th International Conference on Program
3768 Comprehension*. Gothenburg, Sweden: ACM, May 2018. DOI 10.1145/3196321.3196333.
3769 ISBN 978-1-45-035714-2. ISSN 0270-5257 pp. 211–221.
- 3770 [32] J. Biggs and K. Collis, "Evaluating the Quality of Learning: The SOLO Taxonomy (Structure
3771 of the Observed Learning Outcome)," *Management in Education*, vol. 1, no. 4, p. 20, 1987,
3772 DOI 10.1177/089202068700100412. ISBN 0-12-097551-1. ISSN 0892-0206
- 3773 [33] J. J. Blake, L. P. Maguire, T. M. McGinnity, B. Roche, and L. J. McDaid, "The implementation of
3774 fuzzy systems, neural networks and fuzzy neural networks using FPGAs," *Information Sciences*,
3775 vol. 112, no. 1-4, pp. 151–168, 1998, DOI 10.1016/S0020-0255(98)10029-4. ISSN 0020-0255
- 3776 [34] B. S. Bloom, *Taxonomy of Educational Objectives, Handbook 1: Cognitive Domain*, 2nd ed.
3777 Addison-Wesley Longman, 1956. ISBN 978-0-58-228010-6
- 3778 [35] B. W. Boehm, J. R. Brown, and M. Lipow, "Quantitative evaluation of software quality," in
3779 *Proceedings of the 2nd International Conference on Software Engineering*. San Francisco,
3780 California, USA: IEEE, October 1976. ISSN 0270-5257 pp. 592–605.
- 3781 [36] B. Boehm and V. R. Basili, "Software defect reduction top 10 list," *Software Management*, pp.
3782 419–421, 2007, DOI 10.1109/9780470049167.ch12. ISBN 978-0-47-004916-7
- 3783 [37] B. W. Boehm, *Software engineering economics*. Englewood Cliffs, NJ, USA: Prentice-Hall,
3784 1981. ISBN 0-13-822122-7
- 3785 [38] M. Boyd and N. Wilson, "Just ask Siri? A pilot study comparing smartphone digital assistants
3786 and laptop Google searches for smoking cessation advice," *PLoS ONE*, vol. 13, no. 3, 2018,
3787 DOI 10.1371/journal.pone.0194811. ISSN 1932-6203
- 3788 [39] O. Boz, "Extracting decision trees from trained neural networks," in *Proceedings of the 8th ACM
3789 SIGKDD International Conference on Knowledge Discovery and Data Mining*. Edmonton,
3790 AB, Canada: ACM, July 2002. DOI 10.1145/775107.775113, pp. 456–461.
- 3791 [40] H. B. Braiek and F. Khomh, "On Testing Machine Learning Programs," *arXiv preprint
3792 arXiv:1812.02257*, December 2018.
- 3793 [41] M. Bramer, *Principles of Data Mining*, ser. Undergraduate Topics in Computer Science. Lon-
3794 don, England, UK: Springer, 2016, vol. 180, DOI 10.1007/978-1-4471-7307-6. ISBN 978-1-
3795 44-717306-9
- 3796 [42] J. Brandt, P. J. Guo, J. Lewenstein, M. Dontcheva, and S. R. Klemmer, "Two studies of oppor-
3797 tunistic programming: Interleaving web foraging, learning, and writing code," in *Proceedings
3798 of the SIGCHI Conference on Human Factors in Computing System*. Boston, MA, USA: ACM,
3799 April 2009. DOI 10.1145/1518701.1518944. ISBN 978-1-60-558247-4 pp. 1589–1598.
- 3800 [43] L. Brathall and M. Jørgensen, "Can you trust a single data source exploratory software engi-
3801 neering case study?" *Empirical Software Engineering*, 2002, DOI 10.1023/A:1014866909191.
3802 ISSN 13823256

- 3803 [44] E. Breck, S. Cai, E. Nielsen, M. Salib, and D. Sculley, "What's your ML Test Score? A rubric for
3804 ML production systems," in *Proceedings of the 30th Annual Conference on Neural Information
3805 Processing Systems*. Barcelona, Spain: Curran Associates Inc., December 2016.
- 3806 [45] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*.
3807 New York, NY, USA: CRC press, 1984. DOI 10.1201/9781315139470. ISBN 978-1-35-
3808 146049-1
- 3809 [46] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, "Lessons from applying
3810 the systematic literature review process within the software engineering domain," *Journal of
3811 Systems and Software*, vol. 80, no. 4, pp. 571–583, April 2007, DOI 10.1016/j.jss.2006.07.009.
3812 ISSN 0164-1212
- 3813 [47] J. Brooke, "SUS-A quick and dirty usability scale," *Usability Evaluation in Industry*, pp.
3814 189–194, 1996. ISBN 978-0-74-840460-5
- 3815 [48] M. Bunge, "A General Black Box Theory," *Philosophy of Science*, vol. 30, no. 4, pp. 346–358,
3816 October 1963, DOI 10.1086/287954. ISSN 0031-8248
- 3817 [49] BusinessWire, "FileShadow Delivers Machine Learning to End Users with Google Vision API
3818 | Business Wire." [Online] Available: <https://bwnews.pr/2O5qv78>, July 2018, Accessed: 25
3819 January 2019.
- 3820 [50] A. Bussone, S. Stumpf, and D. O'Sullivan, "The role of explanations on trust and reliance in
3821 clinical decision support systems," in *Proceedings of the 2015 IEEE International Conference on
3822 Healthcare Informatics*. Dallas, TX, USA: IEEE, October 2015. DOI 10.1109/IChI.2015.26.
3823 ISBN 978-1-46-739548-9 pp. 160–169.
- 3824 [51] G. Canfora, "User-side testing of Web Services," in *Proceedings of the 9th European Conference
3825 on Software Maintenance and Reengineering*. Manchester, England, UK: IEEE, March 2005.
3826 DOI 10.1109/csmr.2005.57. ISSN 1534-5351 p. 301.
- 3827 [52] G. Canfora and M. Di Penta, "Testing services and service-centric systems: Challenges and
3828 opportunities," *IT Professional*, vol. 8, no. 2, pp. 10–17, 2006, DOI 10.1109/MITP.2006.51.
3829 ISSN 1520-9202
- 3830 [53] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad, "Intelligible models for
3831 healthcare: Predicting pneumonia risk and hospital 30-day readmission," in *Proceedings of
3832 the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*,
3833 vol. 2015-Augus. Sydney, Australia: ACM, August 2015. DOI 10.1145/2783258.2788613.
3834 ISBN 978-1-45-033664-2 pp. 1721–1730.
- 3835 [54] F. Casati, H. Kuno, G. Alonso, and V. Machiraju, *Web Services-Concepts, Architectures and
3836 Applications*, 2004. ISBN 978-3-64-207888-0
- 3837 [55] J. P. Cavano and J. A. McCall, "A framework for the measurement of software quality," in
3838 *Proceedings of the Software Quality Assurance Workshop on Functional and Performance
3839 Issues*, vol. 3, no. 5, November 1978, DOI 10.1145/800283.811113, pp. 133–139.
- 3840 [56] C. V. Chambers, D. J. Balaban, B. L. Carlson, and D. M. Grasberger, "The effect of
3841 microcomputer-generated reminders on influenza vaccination rates in a university-based family
3842 practice center." *The Journal of the American Board of Family Practice / American Board of
3843 Family Practice*, vol. 4, no. 1, pp. 19–26, 1991, DOI 10.3122/jabfm.4.1.19. ISSN 0893-8652
- 3844 [57] J. Cheng and R. Greiner, "Learning bayesian belief network classifiers: Algorithms and system,"
3845 in *Proceedings of the 14th Biennial Conference of the Canadian Society for Computational
3846 Studies of Intelligence*, vol. 2056. Ottawa, ON, Canada: Springer, June 2001. DOI 10.1007/3-
3847 540-45153-6_14. ISBN 3-54-042144-0. ISSN 1611-3349 pp. 141–151.
- 3848 [58] R. Chillarege, I. S. Bhandari, J. K. Chaar, M. J. Halliday, B. K. Ray, and D. S. Moebus, "Or-
3849 thogonal Defect Classification—A Concept for In-Process Measurements," *IEEE Transactions
3850 on Software Engineering*, vol. 18, no. 11, pp. 943–956, 1992, DOI 10.1109/32.177364. ISSN
3851 0098-5589
- 3852 [59] E. Choi, N. Yoshida, R. G. Kula, and K. Inoue, "What do practitioners ask about code clone? a
3853 preliminary investigation of stack overflow," in *Proceedings of the 9th International Workshop
3854 on Software Clones*, Montreal, QC, Canada, March 2015, DOI 10.1109/IWSC.2015.7069890.
3855 ISBN 978-1-46-736914-5 pp. 49–50.
- 3856 [60] Digital, "Case Study: Finding defects earlier yields enormous savings," [Online] Available:
3857 <http://bit.ly/36II2cE>, 2003.

- 3858 [61] P. Clark and R. Boswell, "Rule induction with CN2: Some recent improvements," in *Proceedings of the 1991 European Working Session on Learning*. Porto, Portugal: Springer, March
3859 1991. DOI 10.1007/BFb0017011. ISBN 978-3-54-053816-5. ISSN 1611-3349 pp. 151–163.
- 3860 [62] J. Cohen, "A Coefficient of Agreement for Nominal Scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, 1960, DOI 10.1177/001316446002000104. ISSN
3861 1552-3888
- 3862 [63] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations,"
3863 in *Proceedings of the 10th ACM Conference on Recommender Systems*. Boston, MA, USA:
3864 ACM, September 2016. DOI 10.1145/2959100.2959190. ISBN 978-1-45-034035-9 pp. 191–
3865 198.
- 3866 [64] M. W. Craven and J. W. Shavlik, "Extracting tree-structured representations of trained neural
3867 networks," in *Proceedings of the 8th International Conference on Neural Information Processing
3868 Systems*, vol. 8. Denver, CO, USA: MIT Press, December 1996. ISBN 978-0-26-220107-0 pp.
3869 24–30.
- 3870 [65] J. W. Creswell, *Research design: Qualitative, quantitative, and mixed methods approaches*,
3871 4th ed. SAGE, 2017. ISBN 860-1-40-429618-5
- 3872 [66] P. B. Crosby, *Quality is free: The art of making quality certain*. McGraw-Hill, 1979. ISBN
3873 978-0-07-014512-2
- 3874 [67] A. Cummaudo, "ESEC/FSE Paper," Unpublished.
- 3875 [68] A. Cummaudo, R. Vasa, and J. Grundy, "What should I document? A preliminary systematic
3876 mapping study into API documentation knowledge," in *Proceedings of the 13th International
3877 Symposium on Empirical Software Engineering and Measurement*. Porto de Galinhas, Recife,
3878 Brazil: IEEE, October 2019. DOI 10.1109/ESEM.2019.8870148. ISBN 978-1-72-812968-6.
3879 ISSN 1949-3789 pp. 1–6.
- 3880 [69] A. Cummaudo, R. Vasa, J. Grundy, M. Abdelrazek, and A. Cain, "Losing Confidence in
3881 Quality: Unspoken Evolution of Computer Vision Services," in *Proceedings of the 35th IEEE
3882 International Conference on Software Maintenance and Evolution*. Cleveland, OH, USA:
3883 IEEE, December 2019. DOI 10.1109/ICSME.2019.00051. ISBN 978-1-72-813094-1 pp.
3884 333–342.
- 3885 [70] A. Cummaudo, S. Barnett, R. Vasa, and J. Grundy, "Threshy: Supporting Safe Usage of
3886 Intelligent Web Services," Seoul, Republic of Korea, 2020, Unpublished.
- 3887 [71] A. Cummaudo, R. Vasa, S. Barnett, J. Grundy, and M. Abdelrazek, "Interpreting Cloud Com-
3888 puter Vision Pain-Points: A Mining Study of Stack Overflow," in *Proceedings of the 42nd
3889 International Conference on Software Engineering*. Seoul, Republic of Korea: IEEE, May
3890 2020, In Press.
- 3891 [72] A. Cummaudo, R. Vasa, and J. Grundy, "Assessing the efficacy of API documentation knowl-
3892 edge against practitioners and computer vision services," 2020, Unpublished.
- 3893 [73] M. K. Curumsing, A. Cummaudo, U. M. Graestch, S. Barnett, and R. Vasa, "Ranking Computer
3894 Vision Service Issues using Emotion," in *Proceedings of the 5th International Workshop on
3895 Emotion Awareness in Software Engineering*, Seoul, Republic of Korea, May 2020.
- 3896 [74] H. da Mota Silveira and L. C. Martini, "How the New Approaches on Cloud Computer Vision
3897 can Contribute to Growth of Assistive Technologies to Visually Impaired in the Following
3898 Years?" *Journal of Information Systems Engineering & Management*, vol. 2, no. 2, pp. 1–3,
3899 2017, DOI 10.20897/jisem.201709. ISSN 2468-4376
- 3900 [75] R. M. Davison, M. G. Martinsons, and N. Kock, "Principles of canonical action re-
3901 search," *Information Systems Journal*, vol. 14, no. 1, pp. 65–86, 2004, DOI 10.1111/j.1365-
3902 2575.2004.00162.x. ISSN 1350-1917
- 3903 [76] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic
3904 algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp.
3905 182–197, April 2002, DOI 10.1109/4235.996017. ISSN 1089778X
- 3906 [77] K. Dejaeger, F. Goethals, A. Giangreco, L. Mola, and B. Baesens, "Gaining insight into student
3907 satisfaction using comprehensible data mining techniques," *European Journal of Operational
3908 Research*, vol. 218, no. 2, pp. 548–562, 2012, DOI 10.1016/j.ejor.2011.11.022. ISSN 0377-2217
- 3909 [78] V. Dhar, D. Chou, and F. Provost, "Discovering interesting patterns for investment decision
3910 making with GLOWER - A genetic learner overlaid with entropy reduction," *Data Mining and
3911*

- 3913 *Knowledge Discovery*, vol. 4, no. 4, pp. 69–80, 2000, DOI 10.1023/A:1009848126475. ISSN
3914 1384-5810
- 3915 [79] V. Dibia, A. Cox, and J. Weisz, “Designing for Democratization: Introducing Novices to
3916 Artificial Intelligence Via Maker Kits,” in *Proceedings of the 2017 CHI Conference Extended
3917 Abstracts on Human Factors in Computing Systems*. Denver, CO, USA: ACM, May 2017, pp.
3918 381–384.
- 3919 [80] M. Doderer, K. Yoon, J. Salinas, and S. Kwek, “Protein subcellular localization prediction
3920 using a hybrid of similarity search and Error-Correcting Output Code techniques that produces
3921 interpretable results,” *In Silico Biology*, vol. 6, no. 5, pp. 419–433, 2006. ISSN 1386-6338
- 3922 [81] P. Domingos, “Occam’s Two Razors: The Sharp and the Blunt,” in *Proceedings of the 4th
3923 International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA:
3924 AAAI, August 1998. DOI 10.1.1.40.3278, pp. 37–43.
- 3925 [82] B. Dorn and M. Guzdial, “Learning on the job: Characterizing the programming knowl-
3926 edge and learning strategies of web designers,” in *Proceedings of the 28th ACM Conference
3927 on Human Factors in Computing Systems*, vol. 2. Atlanta, GA, USA: ACM, April 2010.
3928 DOI 10.1145/1753326.1753430. ISBN 978-1-60-558929-9 pp. 703–712.
- 3929 [83] F. Doshi-Velez and B. Kim, “Towards A Rigorous Science of Interpretable Machine Learning,”
3930 *arXiv preprint arXiv:1702.08608*, 2017.
- 3931 [84] F. Doshi-Velez, M. Kortz, R. Budish, C. Bavitz, S. J. Gershman, D. O’Brien, S. Shieber,
3932 J. Waldo, D. Weinberger, and A. Wood, “Accountability of AI Under the Law: The Role of
3933 Explanation,” *SSRN Electronic Journal*, November 2017, In Press, DOI 10.2139/ssrn.3064761.
- 3934 [85] R. G. Dromey, “A model for software product quality,” *IEEE Transactions on Software Engi-
3935 neering*, vol. 21, no. 2, pp. 146–162, 1995, DOI 10.1109/32.345830. ISBN 978-1-11-815666-7.
3936 ISSN 0098-5589
- 3937 [86] C. Drummond and R. C. Holte, “Cost curves: An improved method for visualizing classifier per-
3938 formance,” *Machine Learning*, vol. 65, no. 1, pp. 95–130, October 2006, DOI 10.1007/s10994-
3939 006-8199-5. ISSN 0885-6125
- 3940 [87] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, “Selecting empirical methods for
3941 software engineering research,” in *Guide to Advanced Empirical Software Engineering*, F. Shull,
3942 J. Singer, and D. I. K. Sjøberg, Eds. Springer, November 2007, ch. 11, pp. 285–311. ISBN
3943 978-1-84-800043-8
- 3944 [88] W. Elazmeh, S. Matwin, D. O’Sullivan, W. Michalowski, and K. Farion, “Insights from pre-
3945 dicting pediatric asthma exacerbations from retrospective clinical data,” in *Proceedings of the
3946 22nd Conference on Artificial Intelligence*, vol. WS-07-05. Vancouver, BC, Canada: AAAI,
3947 July 2007. ISBN 978-1-57-735332-4 pp. 10–15.
- 3948 [89] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno,
3949 and D. Song, “Robust Physical-World Attacks on Deep Learning Visual Classification,” in
3950 *Proceedings of the 2017 IEEE Computer Society Conference on Computer Vision and Pattern
3951 Recognition*, Honolulu, HI, USA, July 2018, DOI 10.1109/CVPR.2018.00175. ISBN 978-1-
3952 53-866420-9. ISSN 1063-6919 pp. 1625–1634.
- 3953 [90] F. Elder, D. Michie, D. J. Spiegelhalter, and C. C. Taylor, “Machine Learning, Neural, and
3954 Statistical Classification.” *Journal of the American Statistical Association*, vol. 91, no. 433, pp.
3955 436–438, 1996, DOI 10.2307/2291432. ISBN 978-0-13-106360-0. ISSN 0162-1459
- 3956 [91] A. J. Feelders, “Prior knowledge in economic applications of data mining,” in *Proceedings of
3957 the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, vol.
3958 1910. Lyon, France: Springer, September 2000. DOI 10.1007/3-540-45372-5_42. ISBN
3959 978-3-54-041066-9. ISSN 1611-3349 pp. 395–400.
- 3960 [92] R. T. Fielding, “Architectural Styles and the Design of Network-based Software Architectures,”
3961 Ph.D. dissertation, University of California, Irvine, 2000.
- 3962 [93] I. Finalyson, “Nondeterministic Finite Automata,” [Online] Available: <http://bit.ly/319GOF9>,
3963 Fredericksburg, VA, USA, 2018.
- 3964 [94] H. Foster, S. Uchitel, J. Magee, and J. Kramer, “Model-based verification of Web service
3965 compositions,” in *Proceedings of the 18th International Conference on Automated Software
3966 Engineering*. Linz, Austria: IEEE, September 2004. DOI 10.1109/ase.2003.1240303, pp.
3967 152–161.

- 3968 [95] A. A. Freitas, "A critical review of multi-objective optimization in data mining," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 2, p. 77, 2004, DOI 10.1145/1046456.1046467. ISSN 1931-0145
- 3969 [96] ———, "Comprehensible classification models," *ACM SIGKDD Explorations Newsletter*, vol. 15, no. 1, pp. 1–10, March 2014, DOI 10.1145/2594473.2594475. ISSN 1931-0145
- 3970 [97] A. A. Freitas, D. C. Wieser, and R. Apweiler, "On the importance of comprehensible classification models for protein function prediction," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 7, no. 1, pp. 172–182, 2010, DOI 10.1109/TCBB.2008.47. ISSN 1545-5963
- 3971 [98] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, February 2007, DOI 10.1126/science.1136800. ISSN 0036-8075
- 3972 [99] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian Network Classifiers," *Machine Learning*, vol. 29, no. 2-3, pp. 131–163, 1997, DOI 10.1002/9780470400531.eorms0099. ISSN 0885-6125
- 3973 [100] G. Fung, S. Sandilya, and R. B. Rao, "Rule extraction from linear support vector machines," *Studies in Computational Intelligence*, vol. 80, no. 1, pp. 83–107, 2009, DOI 10.1007/978-3-540-75390-2_4.
- 3974 [101] M. Gamer, J. Lemon, I. Fellows, and P. Singh, "Irr: various coefficients of interrater reliability," *R package version 0.83*, 2010.
- 3975 [102] S. K. Garg, S. Versteeg, and R. Buyya, "SMICloud: A framework for comparing and ranking cloud services," in *Proceedings of the 4th IEEE International Conference on Utility and Cloud Computing*. Melbourne, Australia: IEEE, December 2011. DOI 10.1109/UCC.2011.36. ISBN 978-0-76954592-9 pp. 210–218.
- 3976 [103] V. Garousi and M. Felderer, "Experience-based guidelines for effective and efficient data extraction in systematic reviews in software engineering," in *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, vol. Part F1286. Karlshamn, Sweden: ACM, June 2017. DOI 10.1145/3084226.3084238. ISBN 978-1-45034804-1 pp. 170–179.
- 3977 [104] V. Garousi, M. Felderer, and M. V. Mäntylä, "Guidelines for including grey literature and conducting multivocal literature reviews in software engineering," *Information and Software Technology*, vol. 106, pp. 101–121, 2019, DOI 10.1016/j.infsof.2018.09.006. ISSN 0950-5849
- 3978 [105] D. A. Garvin, "What Does 'Product Quality' Really Mean?" *MIT Sloan Management Review*, vol. 26, no. 1, pp. 25–43, 1984. ISSN 0019-848X
- 3979 [106] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. Daumeé, and K. Crawford, "Datasheets for Datasets," *arXiv preprint arXiv:1803.09010*, 2018.
- 3980 [107] GeoSpatial World, "Mapillary and Amazon Rekognition collaborate to build a parking solution for US cities through computer vision," [Online] Available: <http://bit.ly/36AdRmS>, September 2018, Accessed: 25 January 2019.
- 3981 [108] M. Gethsiyal Augusta and T. Kathirvalavakumar, "Reverse engineering the neural networks for rule extraction in classification problems," *Neural Processing Letters*, vol. 35, no. 2, pp. 131–150, 2012, DOI 10.1007/s11063-011-9207-8. ISSN 1370-4621
- 3982 [109] H. L. Gilmore, "Product conformance cost," *Quality progress*, vol. 7, no. 5, pp. 16–19, 1974.
- 3983 [110] R. L. Glass, I. Vessey, and V. Ramesh, "RESRES: The story behind the paper "Research in software engineering: An analysis of the literature","" *Information and Software Technology*, vol. 51, no. 1, pp. 68–70, 2009, DOI 10.1016/j.infsof.2008.09.015. ISSN 0950-5849
- 3984 [111] M. W. Godfrey and D. M. German, "The past, present, and future of software evolution," in *Proceedings of the 2008 Frontiers of Software Maintenance*, Beijing, China, October 2008, DOI 10.1109/FOSM.2008.4659256. ISBN 978-1-42442655-3 pp. 129–138.
- 3985 [112] M. W. Godfrey and Q. Tu, "Evolution in open source software: a case study," in *Conference on Software Maintenance*. San Jose, CA, USA: IEEE, August 2000. DOI 10.1109/icsm.2000.883030, pp. 131–142.
- 3986 [113] Google LLC, "Classification: Thresholding | Machine Learning Crash Course," [Online] Available: <http://bit.ly/36oMgWb>, 2019, Accessed: 5 February 2020.
- 3987 [114] P. D. Grünwald, *The Minimum Description Length Principle*. MIT press, 2019. DOI 10.7551/mitpress/4643.001.0001.

- 4023 [115] M. J. Hadley and H. Marc, "Web Application Description Language," [Online] Available:
4024 <http://bit.ly/2RXRhQ1>, August 2009.
- 4025 [116] H. A. Haenssle, C. Fink, R. Schneiderbauer, F. Toberer, T. Buhl, A. Blum, A. Kalloo, A. Ben
4026 Hadj Hassen, L. Thomas, A. Enk, L. Uhlmann, C. Alt, M. Arenbergerova, R. Bakos, A. Baltzer,
4027 I. Bertlich, A. Blum, T. Bokor-Billmann, J. Bowling, N. Braghierioli, R. Braun, K. Buder-
4028 Bakhaya, T. Buhl, H. Cabo, L. Cabrijan, N. Cevic, A. Classen, D. Deltgen, C. Fink, I. Georgieva,
4029 L. E. Hakim-Meibodi, S. Hanner, F. Hartmann, J. Hartmann, G. Haus, E. Hoxha, R. Karls,
4030 H. Koga, J. Kreusch, A. Lallas, P. Majenka, A. Marghoob, C. Massone, L. Mekokishvili,
4031 D. Mestel, V. Meyer, A. Neuberger, K. Nielsen, M. Oliviero, R. Pampena, J. Paoli, E. Pawlik,
4032 B. Rao, A. Rendon, T. Russo, A. Sadek, K. Samhaber, R. Schneiderbauer, A. Schweizer,
4033 F. Toberer, L. Trennheuser, L. Vlahova, A. Wald, J. Winkler, P. Wōlbing, and I. Zalaudek, "Man
4034 against Machine: Diagnostic performance of a deep learning convolutional neural network for
4035 dermoscopic melanoma recognition in comparison to 58 dermatologists," *Annals of Oncology*,
4036 vol. 29, no. 8, pp. 1836–1842, May 2018, DOI 10.1093/annonc/mdy166. ISSN 1569-8041
- 4037 [117] K. A. Hallgren, "Computing Inter-Rater Reliability for Observational Data: An Overview and
4038 Tutorial," *Tutorials in Quantitative Methods for Psychology*, vol. 8, no. 1, pp. 23–34, February
4039 2012, DOI 10.20982/tqmp.08.1.p023. ISSN 1913-4126
- 4040 [118] M. Hardt, E. Price, and N. Srebro, "Equality of opportunity in supervised learning," in *Pro-
4041 ceedings of the 30th International Conference on Neural Information Processing Systems*.
4042 Barcelona, Spain: Curran Associates Inc., December 2016. DOI 978-1-51-083881-9. ISSN
4043 1049-5258 pp. 3323–3331.
- 4044 [119] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*, 2nd ed., ser.
4045 Data Mining, Inference, and Prediction. Springer, January 2001.
- 4046 [120] B. Hayete and J. R. Bienkowska, "Gotrees: Predicting go associations from protein domain
4047 composition using decision trees," in *Proceedings of the Pacific Symposium on Biocomputing
4048 2005, PSB 2005*. Hawaii, USA: World Scientific Publishing Company, January 2005.
4049 DOI 10.1142/9789812702456_0013. ISBN 9-81-256046-7 pp. 127–138.
- 4050 [121] R. Heckel and M. Lohmann, "Towards Contract-based Testing of Web Services," *Elec-
4051 tronic Notes in Theoretical Computer Science*, vol. 116, pp. 145–156, January 2005,
4052 DOI 10.1016/j.entcs.2004.02.073. ISSN 1571-0661
- 4053 [122] D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie, "Dependency
4054 networks for inference, collaborative filtering, and data visualization," *Journal of Machine
4055 Learning Research*, vol. 1, no. 1, pp. 49–75, 2001, DOI 10.1162/153244301753344614. ISSN
4056 1532-4435
- 4057 [123] M. Henning, "API design matters," *Communications of the ACM*, vol. 52, no. 5, pp. 46–56,
4058 2009, DOI 10.1145/1506409.1506424. ISSN 0001-0782
- 4059 [124] F. Hohman, A. Head, R. Caruana, R. DeLine, and S. M. Drucker, "Gamut: A design probe
4060 to understand how data scientists understand machine learning models," in *Proceedings of the
4061 2019 CHI Conference on Human Factors in Computing Systems*. Glasgow, Scotland, UK:
4062 ACM, May 2019. DOI 10.1145/3290605.3300809. ISBN 978-1-45-035970-2
- 4063 [125] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau, "Visual Analytics in Deep Learning: An
4064 Interrogative Survey for the Next Frontiers," *IEEE Transactions on Visualization and Computer
4065 Graphics*, vol. 25, no. 8, pp. 2674–2693, 2019, DOI 10.1109/TVCG.2018.2843369. ISSN
4066 1941-0506
- 4067 [126] J. W. Horch, *Practical Guide To Software Quality Management*. Artech House, 2003. ISBN
4068 978-1-58-053604-2
- 4069 [127] H. Hosseini, B. Xiao, and R. Poovendran, "Google's cloud vision API is not robust to noise," in
4070 *Proceedings of the 16th IEEE International Conference on Machine Learning and Applications*,
4071 vol. 2017-Decem. Cancun, Mexico: IEEE, December 2017. DOI 10.1109/ICMLA.2017.0-
4072 172. ISBN 978-1-53-861417-4 pp. 101–105.
- 4073 [128] D. Hou and L. Mo, "Content categorization of API discussions," in *Proceedings of the 29th In-
4074 ternational Conference on Software Maintenance*. Eindhoven, Netherlands: IEEE, September
4075 2013. DOI 10.1109/ICSM.2013.17, pp. 60–69.
- 4076 [129] C. Howard, "Introducing Google AI," [Online] Available: <http://bit.ly/2uI6vAr>, May 2018,
4077 Accessed: 28 August 2018.

- 4078 [130] J. Huang and C. X. Ling, "Using AUC and accuracy in evaluating learning algorithms,"
4079 *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 3, pp. 299–310, 2005,
4080 DOI 10.1109/TKDE.2005.50. ISSN 1041-4347
- 4081 [131] J. Huysmans, K. Dejaeger, C. Mues, J. Vanthienen, and B. Baesens, "An empirical evaluation
4082 of the comprehensibility of decision table, tree and rule based predictive models," *Decision
4083 Support Systems*, vol. 51, no. 1, pp. 141–154, April 2011, DOI 10.1016/j.dss.2010.12.003.
4084 ISSN 0167-9236
- 4085 [132] IEEE, "IEEE Standard Glossary of Software Engineering Terminology," 1990.
- 4086 [133] International Organization for Standardization, "ISO/IEC 25010:2011 Systems and software
4087 engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System
4088 and software quality models," [Online] Available: <http://bit.ly/2S4yzGs>, 2011.
- 4089 [134] ———, "ISO 8402:1986 Information Technology - Software Product Evaluation - Quality Char-
4090 acteristics and Guidelines for Their Use," [Online] Available: <http://bit.ly/37SK4HP>, 1986.
- 4091 [135] ———, "ISO 9000:2015 Quality management systems – Fundamentals and vocabulary," [Online]
4092 Available: <http://bit.ly/37O4oKo>, 2015.
- 4093 [136] ———, "ISO/IEC 9126 Information Technology - Software Product Evaluation - Quality Char-
4094 acteristics and Guidelines for Their Use," [Online] Available: <http://bit.ly/2tgMHUE>, November
4095 1999.
- 4096 [137] A. Iyengar, "Supporting Data Analytics Applications Which Utilize Cognitive Services," in
4097 *Proceedings of the 37th International Conference on Distributed Computing Systems*. Atlanta,
4098 GA, USA: IEEE, June 2017. DOI 10.1109/ICDCS.2017.172. ISBN 978-1-53-861791-5 pp.
4099 1856–1864.
- 4100 [138] N. Japkowicz and M. Shah, *Evaluating learning algorithms: A classification perspective*.
4101 Cambridge University Press, 2011, vol. 9780521196, DOI 10.1017/CBO9780511921803. ISBN
4102 978-0-51-192180-3
- 4103 [139] M. W. M. Jaspers, M. Smeulders, H. Vermeulen, and L. W. Peute, "Effects of clinical decision-
4104 support systems on practitioner performance and patient outcomes: A synthesis of high-quality
4105 systematic review findings," *Journal of the American Medical Informatics Association*, vol. 18,
4106 no. 3, pp. 327–334, 2011, DOI 10.1136/amiainjnl-2011-000094. ISSN 1067-5027
- 4107 [140] T. Jiang and A. E. Keating, "AVID: An integrative framework for discovering functional rela-
4108 tionship among proteins," *BMC Bioinformatics*, vol. 6, no. 1, p. 136, 2005, DOI 10.1186/1471-
4109 2105-6-136. ISSN 1471-2105
- 4110 [141] B. Jimerson and B. Gregory, "Pivotal Cloud Foundry, Google ML, and Spring," [Online]
4111 Available: <http://bit.ly/2RUBIIIL>, San Francisco, CA, USA, December 2017.
- 4112 [142] Y. Jin, *Multi-Objective Machine Learning*, ser. Studies in Computational Intelligence. Berlin,
4113 Heidelberg: Springer, 2006. DOI 10.1007/3-540-33019-4. ISBN 978-3-54-030676-4
- 4114 [143] U. Johansson and L. Niklasson, "Evolving decision trees using oracle guides," in *Proceedings
4115 of the 2009 IEEE Symposium on Computational Intelligence and Data Mining*. Nashville,
4116 TN, USA: IEEE, May 2009. DOI 10.1109/CIDM.2009.4938655. ISBN 978-1-42-442765-9
4117 pp. 238–244.
- 4118 [144] M. Jørgensen, T. Dybå, K. Liestøl, and D. I. K. Sjøberg, "Incorrect results in software engineer-
4119 ing experiments: How to improve research practices," *Journal of Systems and Software*, vol.
4120 116, pp. 133–145, 2016, DOI 10.1016/j.jss.2015.03.065. ISSN 0164-1212
- 4121 [145] J. M. Juran, *Juran on Planning for Quality*. New York, NY, USA: The Free Press, 1988. ISBN
4122 978-0-02-916681-9
- 4123 [146] N. Juristo and O. S. Gómez, "Replication of software engineering experiments," in *Proceedings
4124 of the LASER Summer School on Software Engineering*. Elba Island, Italy: Springer, 2011.
4125 DOI 10.1007/978-3-642-25231-0_2. ISBN 978-3-64-225230-3. ISSN 0302-9743 pp. 60–88.
- 4126 [147] N. Juristo and A. M. Moreno, *Basics of Software Engineering Experimentation*. Boston, MA,
4127 USA: Springer, March 2001. DOI 10.1007/978-1-4757-3304-4.
- 4128 [148] A. Karwath and R. D. King, "Homology induction: The use of machine learning to improve se-
4129 quence similarity searches," *BMC Bioinformatics*, vol. 3, no. 1, p. 11, 2002, DOI 10.1186/1471-
4130 2105-3-11. ISSN 1471-2105
- 4131 [149] K. A. Kaufman and R. S. Michalski, "Learning from inconsistent and noisy data: The AQ18
4132 approach," in *Proceedings of the 11th European Conference on Principles and Practice of*

- 4133 *Knowledge Discovery in Databases*, vol. 1609. Warsaw, Poland: Springer, September 1999.
4134 DOI 10.1007/BFb0095128. ISBN 3-540-65965-X. ISSN 1611-3349 pp. 411–419.
- 4135 [150] D. Kavaler, D. Posnett, C. Gibler, H. Chen, P. Devanbu, and V. Filkov, “Using and asking:
4136 APIs used in the Android market and asked about in StackOverflow,” in *Proceedings of the
4137 5th International Conference on Social Informatics*. Kyoto, Japan: Springer, November 2013.
4138 DOI 10.1007/978-3-319-03260-3_35. ISBN 978-3-319-03259-7. ISSN 0302-9743 pp. 405–
4139 418.
- 4140 [151] B. Kim, “Interactive and Interpretable Machine Learning Models for Human Machine Collab-
4141 oration,” Ph.D. dissertation, Massachusetts Institute of Technology, 2015.
- 4142 [152] B. Kim, C. Rudin, and J. Shah, “The Bayesian case model: A generative approach for case-
4143 based reasoning and prototype classification,” in *Proceedings of the 28th Conference on Neural
4144 Information Processing Systems*, Montreal, QC, Canada, December 2014. ISSN 1049-5258 pp.
4145 1952–1960.
- 4146 [153] B. Kitchenham and S. Charters, “Guidelines for performing Systematic Literature Reviews
4147 in Software Engineering,” Software Engineering Group, Keele University and Department of
4148 Computer Science, University of Durham, Keele, UK, Tech. Rep., 2007.
- 4149 [154] B. A. Kitchenham and S. L. Pfleeger, “Personal opinion surveys,” in *Guide to Advanced
4150 Empirical Software Engineering*, F. Shull, J. Singer, and D. I. K. Sjøberg, Eds. Springer,
4151 November 2007, ch. 3, pp. 63–92. ISBN 978-1-84-800043-8
- 4152 [155] B. A. Kitchenham, T. Dybå, and M. Jorgensen, “Evidence-Based Software Engineering,” in
4153 *Proceedings of the 26th International Conference on Software Engineering*. Edinburgh,
4154 Scotland, UK: IEEE, May 2004. ISBN 978-0-76-952163-3 pp. 273–281.
- 4155 [156] H. K. Klein and M. D. Myers, “A set of principles for conducting and evaluating interpretive
4156 field studies in information systems,” *MIS Quarterly: Management Information Systems*, vol. 23,
4157 no. 1, pp. 67–94, 1999, DOI 10.2307/249410. ISSN 0276-7783
- 4158 [157] A. J. Ko and Y. Riche, “The role of conceptual knowledge in API usability,” in *Proceedings of
4159 the 2011 IEEE Symposium on Visual Languages and Human Centric Computing*. Pittsburg,
4160 PA, USA: IEEE, September 2011. DOI 10.1109/VLHCC.2011.6070395. ISBN 978-1-45-
4161 771245-6 pp. 173–176.
- 4162 [158] A. J. Ko, B. A. Myers, and H. H. Aung, “Six learning barriers in end-user programming
4163 systems,” in *Proceedings of the 2004 IEEE Symposium on Visual Languages and Human
4164 Centric Computing*. Rome, Italy: IEEE, September 2004. DOI 10.1109/vlhcc.2004.47.
4165 ISBN 0-78-038696-5 pp. 199–206.
- 4166 [159] I. Kononenko, “Inductive and bayesian learning in medical diagnosis,” *Applied Artificial Intel-
4167 ligence*, vol. 7, no. 4, pp. 317–337, 1993, DOI 10.1080/08839519308949993. ISSN 1087-6545
- 4168 [160] K. Krippendorff, *Content Analysis*, ser. An Introduction to Its Methodology. SAGE, 1980.
4169 ISBN 978-1-50-639566-1
- 4170 [161] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convo-
4171 lutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017,
4172 DOI 10.1145/3065386. ISSN 1557-7317
- 4173 [162] A. Kurakin, I. J. Goodfellow, and S. Bengio, “Adversarial examples in the physical world,” in
4174 *Proceedings of the 5th International Conference on Learning Representations*, Toulon, France,
4175 April 2017.
- 4176 [163] G. Laforge, “Machine Intelligence at Google Scale,” in *QCon*, London, England, UK, June
4177 2018.
- 4178 [164] H. Lakkaraju, S. H. Bach, and J. Leskovec, “Interpretable decision sets: A joint framework for
4179 description and prediction,” in *Proceedings of the 22nd ACM SIGKDD International Conference
4180 on Knowledge Discovery and Data Mining*. San Francisco, CA, USA: ACM, August 2016.
4181 DOI 10.1145/2939672.2939874. ISBN 978-1-45-034232-2 pp. 1675–1684.
- 4182 [165] J. R. Landis and G. G. Koch, “The Measurement of Observer Agreement for Categorical Data,”
4183 *Biometrics*, vol. 33, no. 1, p. 159, March 1977, DOI 10.2307/2529310. ISSN 0006341X
- 4184 [166] N. Lavrač, “Selected techniques for data mining in medicine,” *Artificial Intelligence in Medicine*,
4185 vol. 16, no. 1, pp. 3–23, 1999, DOI 10.1016/S0933-3657(98)00062-1. ISSN 0933-3657
- 4186 [167] T. Lei, R. Barzilay, and T. Jaakkola, “Rationalizing neural predictions,” in *Proceedings of the 9th
4187 International Joint Conference on Natural Language Processing and Conference on Empirical
4188 Methods in Natural Language Processing*. Austin, TX, USA: Association for Computational

- 4189 Linguistics, November 2016. DOI 10.18653/v1/d16-1011. ISBN 978-1-94-562625-8 pp.
4190 107–117.
- 4191 [168] T. C. Lethbridge, S. E. Sim, and J. Singer, “Studying software engineers: Data collection
4192 techniques for software field studies,” *Empirical Software Engineering*, vol. 10, no. 3, pp.
4193 311–341, July 2005, DOI 10.1007/s10664-005-1290-x. ISSN 1382-3256
- 4194 [169] R. J. Light, “Measures of response agreement for qualitative data: Some generalizations and
4195 alternatives,” *Psychological Bulletin*, vol. 76, no. 5, pp. 365–377, 1971, DOI 10.1037/h0031643.
4196 ISSN 0033-2909
- 4197 [170] E. Lima, C. Mues, and B. Baesens, “Domain knowledge integration in data mining using
4198 decision tables: Case studies in churn prediction,” *Journal of the Operational Research Society*,
4199 vol. 60, no. 8, pp. 1096–1106, 2009, DOI 10.1057/jors.2008.161. ISSN 0160-5682
- 4200 [171] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L.
4201 Zitnick, “Microsoft COCO: Common objects in context,” in *Proceedings of the 13th European
4202 Conference on Computer Vision*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., vol.
4203 8693 LNCS, no. PART 5. Zurich, Germany: Springer, September 2014. DOI 10.1007/978-
4204 3-319-10602-1_48. ISSN 1611-3349 pp. 740–755.
- 4205 [172] M. Linares-Vásquez, G. Bavota, M. Di Penta, R. Oliveto, and D. Poshyvanyk, “How do API
4206 changes trigger stack overflow discussions? A study on the android SDK,” in *Proceedings of the 22nd International Conference on Program Comprehension*. Hyderabad, India: ACM,
4207 June 2014. DOI 10.1145/2597008.2597155. ISBN 978-1-45-032879-1 pp. 83–94.
- 4208 [173] Z. C. Lipton, “The mythos of model interpretability,” *Communications of the ACM*, vol. 61,
4209 no. 10, pp. 35–43, 2018, DOI 10.1145/3233231. ISSN 1557-7317
- 4210 [174] M. Litwin, *How to Measure Survey Reliability and Validity*. Thousand Oaks, CA, USA: SAGE,
4211 1995, vol. 7, DOI 10.4135/9781483348957. ISBN 978-0-80-395704-6
- 4212 [175] Y. Liu, T. Kohlberger, M. Norouzi, G. E. Dahl, J. L. Smith, A. Mohtashamian, N. Olson,
4213 L. H. Peng, J. D. Hipp, and M. C. Stumpe, “Artificial Intelligence-Based Breast Cancer Nodal
4214 Metastasis Detection.” *Archives of Pathology & Laboratory Medicine*, vol. 143, no. 7, pp.
4215 859–868, July 2017, DOI 10.5858/arpa.2018-0147-OA. ISSN 1543-2165
- 4216 [176] D. Lo Giudice, C. Mines, A. LeClair, R. Curran, and A. Homan, “How AI Will Change
4217 Software Development And Applications,” [Online] Available: <http://bit.ly/38RiAIN>, Forrester
4218 Research, Inc., Tech. Rep., November 2016.
- 4219 [177] R. Lori and M. Oded, *Data mining with decision trees*. World Scientific Publishing Company,
4220 2008, vol. 69. ISBN 978-9-81-277171-1
- 4221 [178] R. E. Lyons and W. Vanderkulk, “The Use of Triple-Modular Redundancy to Improve Computer
4222 Reliability,” *IBM Journal of Research and Development*, vol. 6, no. 2, pp. 200–209, April 2010,
4223 DOI 10.1147/rd.62.0200. ISSN 0018-8646
- 4224 [179] W. Maalej and M. P. Robillard, “Patterns of knowledge in API reference documentation,” *IEEE
4225 Transactions on Software Engineering*, 2013, DOI 10.1109/TSE.2013.12. ISSN 0098-5589
- 4226 [180] S. MacDonell, M. Shepperd, B. Kitchenham, and E. Mendes, “How reliable are systematic
4227 reviews in empirical software engineering?” *IEEE Transactions on Software Engineering*,
4228 vol. 36, no. 5, pp. 676–687, September 2010, DOI 10.1109/TSE.2010.28. ISSN 0098-5589
- 4229 [181] G. Malgieri and G. Comandé, “Why a right to legibility of automated decision-making exists
4230 in the general data protection regulation,” *International Data Privacy Law*, vol. 7, no. 4, pp.
4231 243–265, June 2017, DOI 10.1093/idpl/ixp019. ISSN 2044-4001
- 4232 [182] L. Mandel, “Describe REST Web services with WSDL 2.0,” [Online] Available: <https://ibm.co/313RoNV>, May 2008, Accessed: 28 August 2018.
- 4233 [183] T. E. Marshall and S. L. Lambert, “Cloud-based intelligent accounting applications: Accounting
4234 task automation using IBM watson cognitive computing,” *Journal of Emerging Technologies
4235 in Accounting*, vol. 15, no. 1, pp. 199–215, 2018, DOI 10.2308/jeta-52095. ISSN 1558-7940
- 4236 [184] D. Martens, J. Vanthienen, W. Verbeke, and B. Baesens, “Performance of classification mod-
4237 els from a user perspective,” *Decision Support Systems*, vol. 51, no. 4, pp. 782–793, 2011,
4238 DOI 10.1016/j.dss.2011.01.013. ISSN 0167-9236
- 4239 [185] P. Mayring, “Mixing Qualitative and Quantitative Methods,” in *Mixed Methodology in Psycho-
4240 logical Research*. Sense Publishers, 2007, ch. 6, pp. 27–36. ISBN 978-9-07-787473-8
- 4241 [186] P. Mayring, “Qualitative and Quantitative Methods,” in *Handbook of Qualitative and Quantitative
4242 Methods in Psychology*. Springer, 2018, ch. 1, pp. 1–14. ISBN 978-3-030-00001-1

- 4243 [186] J. A. McCall, P. K. Richards, and G. F. Walters, "Factors in Software Quality: Concept and
4244 Definitions of Software Quality," General Electric Company, Griffiss Air Force Base, NY, USA,
4245 Tech. Rep. RADC-TR-77-369, November 1977.
- 4246 [187] J. McCarthy, "Programs with common sense," in *Proceedings of the Symposium on the Mechanization of Thought Processes*, Cambridge, MA, USA, 1963, pp. 1–15.
- 4248 [188] B. McGowen, "Machine learning with Google APIs," [Online] Available: <http://bit.ly/3aUQpo2>, January 2019.
- 4250 [189] M. L. McHugh, "Interrater reliability: The kappa statistic," *Biochemia Medica*, vol. 22, no. 3,
4251 pp. 276–282, 2012, DOI 10.11613/bm.2012.031. ISSN 1330-0962
- 4252 [190] L. McLeod and S. G. MacDonell, "Factors that affect software systems development project
4253 outcomes: A survey of research," *ACM Computing Surveys*, vol. 43, no. 4, p. 24, 2011,
4254 DOI 10.1145/1978802.1978803. ISSN 0360-0300
- 4255 [191] J. Meltzoff and H. Cooper, *Critical thinking about research: Psychology and related fields*,
4256 2nd ed. American Psychological Association, 2018. DOI 10.1037/0000052-000.
- 4257 [192] T. Mens and S. Demeyer, *Software Evolution*. Berlin, Heidelberg: Springer, 2008.
4258 DOI 10.1007/978-3-540-76440-3. ISBN 978-3-54-076439-7
- 4259 [193] T. Mens, S. Demeyer, M. Wermelinger, R. Hirschfeld, S. Ducasse, and M. Jazayeri, "Chal-
4260 lenges in software evolution," in *Proceedings of the 8th International Workshop on Principles
4261 of Software Evolution*, vol. 2005. Lisbon, Portugal: IEEE, September 2005. DOI 10.1109/I-
4262 WPSE.2005.7. ISBN 0-76-952349-8. ISSN 1550-4077 pp. 13–22.
- 4263 [194] A. C. Michalos and H. A. Simon, *The Sciences of the Artificial*. MIT press, 1970, vol. 11,
4264 no. 1, DOI 10.2307/3102825.
- 4265 [195] D. Michie, "Machine learning in the next five years," in *Proceedings of the 3rd European
4266 Conference on European Working Session on Learning*. Glasgow, Scotland, UK: Pitman
4267 Publishing, Inc., October 1988. ISBN 978-0-27-308800-4 pp. 107–122.
- 4268 [196] G. A. Miller, "WordNet: A Lexical Database for English," *Communications of the ACM*, vol. 38,
4269 no. 11, pp. 39–41, November 1995, DOI 10.1145/219717.219748. ISSN 1557-7317
- 4270 [197] M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D.
4271 Raji, and T. Gebru, "Model cards for model reporting," in *Proceedings of the 2nd Conference
4272 on Fairness, Accountability, and Transparency*. Atlanta, GA, USA: ACM, January 2019.
4273 DOI 10.1145/3287560.3287596. ISBN 978-1-45-036125-5 pp. 220–229.
- 4274 [198] D. Moody, "The physics of notations: Toward a scientific basis for constructing visual notations
4275 in software engineering," *IEEE Transactions on Software Engineering*, vol. 35, no. 6, pp.
4276 756–779, 2009, DOI 10.1109/TSE.2009.67. ISSN 0098-5589
- 4277 [199] C. Murphy and G. Kaiser, "Improving the Dependability of Machine Learning Applications,"
4278 Department of Computer Science, Columbia University, New York, NY, USA, Tech. Rep. MI,
4279 2008.
- 4280 [200] C. Murphy, G. Kaiser, and M. Arias, "An approach to software testing of machine learning
4281 applications," in *Proceedings of the 19th International Conference on Software Engineering and
4282 Knowledge Engineering*, Boston, MA, USA, July 2007. ISBN 978-1-62-748661-3 pp. 167–172.
- 4283 [201] B. A. Myers, S. Y. Jeong, Y. Xie, J. Beaton, J. Stylos, R. Ehret, J. Karstens, A. Efeoglu, and D. K.
4284 Busse, "Studying the Documentation of an API for Enterprise Service-Oriented Architecture,"
4285 *Journal of Organizational and End User Computing*, vol. 22, no. 1, pp. 23–51, January 2010,
4286 DOI 10.4018/joeuc.2010101903. ISSN 1546-2234
- 4287 [202] C. Myers, A. Furqan, J. Nebolsky, K. Caro, and J. Zhu, "Patterns for how users overcome
4288 obstacles in Voice User Interfaces," in *Proceedings of the 2018 CHI Conference on Human
4289 Factors in Computing Systems*, vol. 2018-April. Montreal, QC, Canada: ACM, April 2018.
4290 DOI 10.1145/3173574.3173580. ISBN 978-1-45-035620-6 p. 6.
- 4291 [203] S. Nakajima, "Model-Checking Verification for Reliable Web Service," in *Proceedings of the
4292 First International Symposium on Cyber World*. Montreal, QC, Canada: IEEE, November
4293 2002. ISBN 978-0-76-951862-6 pp. 378–385.
- 4294 [204] M. Narayanan, E. Chen, J. He, B. Kim, S. Gershman, and F. Doshi-Velez, "How do Humans
4295 Understand Explanations from Machine Learning Systems? An Evaluation of the Human-
4296 Interpretability of Explanation," *IEEE Transactions on Evolutionary Computation*, 2018, In
4297 Press.

- 4298 [205] S. Narayanan and S. A. McIlraith, "Simulation, verification and automated composition of web
4299 services," in *Proceedings of the 11th International Conference on World Wide Web*. Honolulu,
4300 HI, USA: ACM, May 2002. DOI 10.1145/511446.511457. ISBN 1-58-113449-5 pp. 77–88.
- 4301 [206] B. J. Nelson, "Remote Procedure Call," Ph.D. dissertation, Carnegie Mellon University, 1981.
- 4302 [207] H. F. Niemeyer and A. C. Niemeyer, "Apportionment methods," *Mathematical Social Sciences*,
4303 vol. 56, no. 2, pp. 240–253, 2008. ISSN 0165-4896
- 4304 [208] Y. Nishi, S. Masuda, H. Ogawa, and K. Uetsuki, "A test architecture for machine learning
4305 product," in *Proceedings of the 11th International Conference on Software Testing,
4306 Verification and Validation Workshops*. Västerås, Sweden: IEEE, April 2018.
4307 DOI 10.1109/ICSTW.2018.00060. ISBN 978-1-53-866352-3 pp. 273–278.
- 4308 [209] N. Novielli, F. Calefato, and F. Lanobile, "The challenges of sentiment detection in the social
4309 programmer ecosystem," in *Proceedings of the 7th International Workshop on Social Software
4310 Engineering*, Bergamo, Italy, August 2015, DOI 10.1145/2804381.2804387. ISBN 978-1-45-
4311 033818-9 pp. 33–40.
- 4312 [210] K. Nybom, A. Ashraf, and I. Porres, "A systematic mapping study on API documentation
4313 generation approaches," in *Proceedings of the 44th Euromicro Conference on Software
4314 Engineering and Advanced Applications*. Prague, Czech Republic: IEEE, August 2018.
4315 DOI 10.1109/SEAA.2018.00081. ISBN 978-1-53-867382-9 pp. 462–469.
- 4316 [211] J. Nykaza, R. Messinger, F. Boehme, C. L. Norman, M. Mace, and M. Gordon, "What program-
4317 mers really want: Results of a needs assessment for SDK documentation," in *Proceedings of the
4318 20th Annual International Conference on Computer Documentation*. Toronto, ON, Canada:
4319 ACM, October 2002. DOI 10.1145/584955.584976, pp. 133–141.
- 4320 [212] T. Ohtake, A. Cummaudo, M. Abdelrazek, R. Vasa, and J. Grundy, "Merging intelligent API
4321 responses using a proportional representation approach," in *Proceedings of the 19th Interna-
4322 tional Conference on Web Engineering*. Daejeon, Republic of Korea: Springer, June 2019.
4323 DOI 10.1007/978-3-030-19274-7_28. ISBN 978-3-03-019273-0. ISSN 1611-3349 pp. 391–
4324 406.
- 4325 [213] Open Software Foundation, "Part 3: DCE Remote Procedure Call (RPC)," in *OSF DCE
4326 application development guide: revision 1.0*. Prentice Hall, December 1991.
- 4327 [214] N. Oreskes, K. Shrader-Frechette, and K. Belitz, "Verification, validation, and confirmation
4328 of numerical models in the earth sciences," *Science*, vol. 263, no. 5147, pp. 641–646, 1994,
4329 DOI 10.1126/science.263.5147.641. ISSN 0036-8075
- 4330 [215] A. L. M. Ortiz, "Curating Content with Google Machine Learning Application Programming
4331 Interfaces," in *EIAPortugal*, July 2017.
- 4332 [216] F. E. B. Otero and A. A. Freitas, "Improving the interpretability of classification rules discovered
4333 by an ant colony algorithm: Extended results," in *Evolutionary Computation*, vol. 24, no. 3.
4334 ACM, 2016. DOI 10.1162/EVCO_a_00155. ISSN 1530-9304 pp. 385–409.
- 4335 [217] A. Pal, S. Chang, and J. A. Konstan, "Evolution of experts in question answering communities,"
4336 in *Proceedings of the 6th International AAAI Conference on Weblogs and Social Media*. Dublin,
4337 Ireland: AAAI, June 2012. ISBN 978-1-57-735556-4 pp. 274–281.
- 4338 [218] R. Parekh, "Designing AI at Scale to Power Everyday Life," in *Proceedings of the 23rd ACM
4339 SIGKDD International Conference on Knowledge Discovery and Data Mining*. Halifax, NS,
4340 Canada: ACM, August 2017. DOI 10.1145/3097983.3105815, p. 27.
- 4341 [219] K. Patel, J. Fogarty, J. A. Landay, and B. Harrison, "Investigating statistical machine learn-
4342 ing as a tool for software development," in *Proceedings of the 26th SIGCHI Conference on
4343 Human Factors in Computing Systems*, ser. CHI '08. Florence, Italy: ACM, April 2008.
4344 DOI 10.1145/1357054.1357160. ISBN 978-1-60-558011-1 pp. 667–676.
- 4345 [220] C. Pautasso, O. Zimmermann, and F. Leymann, "RESTful web services vs. "Big" web services:
4346 Making the right architectural decision," in *Proceedings of the 17th International Conference
4347 on World Wide Web*. Beijing, China: ACM, April 2008. DOI 10.1145/1367497.1367606.
4348 ISBN 978-1-60-558085-2
- 4349 [221] M. Pazzani, "Comprehensible knowledge discovery: gaining insight from data," in *Proceedings
4350 of the First Federal Data Mining Conference and Exposition*, Washington, DC, USA, 1997, pp.
4351 73–82.

- 4352 [222] M. J. Pazzani, S. Mani, and W. R. Shankle, “Acceptance of rules generated by machine learning
4353 among medical experts,” *Methods of Information in Medicine*, vol. 40, no. 5, pp. 380–385,
4354 2001, DOI 10.1055/s-0038-1634196. ISSN 0026-1270
- 4355 [223] J. Pearl, “The seven tools of causal inference, with reflections on machine learning,” *Communi-
4356 cations of the ACM*, vol. 62, no. 3, pp. 54–60, 2019, DOI 10.1145/3241036. ISSN 1557-7317
- 4357 [224] K. Petersen and C. Gencel, “Worldviews, research methods, and their relationship to validity
4358 in empirical software engineering research,” in *Proceedings of the Joint Conference of the
4359 23rd International Workshop on Software Measurement and the 8th International Conference
4360 on Software Process and Product Measurement*. Ankara, Turkey: IEEE, October 2013.
4361 DOI 10.1109/IWSM-Mensura.2013.22. ISBN 978-0-76-955078-7 pp. 81–89.
- 4362 [225] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, “Systematic mapping studies in software en-
4363 gineering,” in *Proceedings of the 12th International Conference on Evaluation and Assessment
4364 in Software Engineering, EASE 2008*, 2008, DOI 10.14236/ewic/ease2008.8, pp. 68–77.
- 4365 [226] Z. Pezzementi, T. Tabor, S. Yim, J. K. Chang, B. Drozd, D. Guttendorf, M. Wagner, and
4366 P. Koopman, “Putting Image Manipulations in Context: Robustness Testing for Safe Perception,”
4367 in *Proceedings of the 15th IEEE International Symposium on Safety, Security, and Rescue
4368 Robotics*. Philadelphia, PA, USA: IEEE, August 2018. DOI 10.1109/SSRR.2018.8468619.
4369 ISBN 978-1-53-865572-6 pp. 1–8.
- 4370 [227] H. Pham, *System Software Reliability*, 1st ed. Springer, 2000. ISBN 978-1-84-628295-9
- 4371 [228] M. Piccioni, C. A. Furia, and B. Meyer, “An empirical study of API usability,” in *Proceedings
4372 of the 13th International Symposium on Empirical Software Engineering and Measurement*.
4373 Baltimore, MD, USA: IEEE, October 2013. DOI 10.1109/ESEM.2013.14. ISSN 1949-3770
4374 pp. 5–14.
- 4375 [229] R. M. Pirsig, *Zen and the art of motorcycle maintenance: An inquiry into values*, 1st ed.
4376 HarperTorch, 1974. ISBN 9-780-06-058946-2
- 4377 [230] R. S. Pressman, *Software Engineering: A Practitioner’s Approach*, 8th ed. McGraw-Hill,
4378 2005. ISBN 978-0-07-802212-8
- 4379 [231] J. R. Quinlan, “Some elements of machine learning,” in *Proceedings of the 9th International
4380 Workshop on Inductive Logic Programming*, vol. 1634. Bled, Slovenia: Springer, June 1999.
4381 DOI 10.1007/3-540-48751-4_3. ISBN 3-54-066109-3. ISSN 1611-3349 pp. 15–18.
- 4382 [232] ——, *C4.5: Programs for machine learning*. San Francisco, CA, USA: Morgan Kauffmann,
4383 1993. ISBN 978-1-55-860238-0
- 4384 [233] N. Rama Suri, V. S. Srinivas, and M. Narasimha Murty, “A cooperative game theoretic approach
4385 to prototype selection,” in *Proceedings of the 11th European Conference on Principles and
4386 Practice of Knowledge Discovery in Databases*. Warsaw, Poland: Springer, September 2007.
4387 DOI 10.1007/978-3-540-74976-9_58. ISBN 978-3-54-074975-2. ISSN 0302-9743 pp. 556–
4388 564.
- 4389 [234] M. Reboucas, G. Pinto, F. Ebert, W. Torres, A. Serebrenik, and F. Castor, “An Empirical Study
4390 on the Usage of the Swift Programming Language,” in *Proceedings of the 23rd International
4391 Conference on Software Analysis, Evolution, and Reengineering*. Suita, Japan: IEEE, March
4392 2016. DOI 10.1109/saner.2016.66, pp. 634–638.
- 4393 [235] A. Reis, D. Paulino, V. Filipe, and J. Barroso, “Using online artificial vision services to assist
4394 the blind - An assessment of Microsoft Cognitive Services and Google Cloud Vision,” *Advances
4395 in Intelligent Systems and Computing*, vol. 746, no. 12, pp. 174–184, 2018, DOI 10.1007/978-
4396 3-319-77712-2_17. ISBN 978-3-31-977711-5. ISSN 2194-5357
- 4397 [236] M. T. Ribeiro, S. Singh, and C. Guestrin, “‘Why Should I Trust You?’: Explaining the Predic-
4398 tions of Any Classifier,” in *Proceedings of the 22nd ACM SIGKDD International Conference
4399 on Knowledge Discovery and Data Mining*. San Francisco, CA, USA: ACM, August 2016.
4400 DOI 2939672.2939778, pp. 1135–1144.
- 4401 [237] M. Ribeiro, K. Grolinger, and M. A. M. Capretz, “MLaaS: Machine learning as a service,”
4402 in *Proceedings of the 14th International Conference on Machine Learning and Applications*.
4403 Miami, FL, USA: IEEE, December 2015. DOI 10.1109/ICMLA.2015.152. ISBN 978-1-50-
4404 900287-0 pp. 896–902.
- 4405 [238] G. Richards, V. J. Rayward-Smith, P. H. Sönksen, S. Carey, and C. Weng, “Data mining for
4406 indicators of early mortality in a database of clinical records,” *Artificial Intelligence in Medicine*,
4407 vol. 22, no. 3, pp. 215–231, 2001, DOI 10.1016/S0933-3657(00)00110-X. ISSN 0933-3657

- 4408 [239] G. Ridgeway, D. Madigan, T. Richardson, and J. O’Kane, “Interpretable Boosted Naïve Bayes
4409 Classification,” in *Proceedings of the 4th International Conference on Knowledge Discovery
4410 and Data Mining*. New York, NY, USA: AAAI, 1998, pp. 101–104.
- 4411 [240] RightScale Inc., “State of the Cloud Report: DevOps Trends,” Tech. Rep., 2016.
- 4412 [241] G. Ritzer and E. Guba, “The Paradigm Dialog,” *Canadian Journal of Sociology*, vol. 16, no. 4,
4413 p. 446, 1991, DOI 10.2307/3340973. ISSN 0318-6431
- 4414 [242] M. P. Robillard, “What makes APIs hard to learn? Answers from developers,” *IEEE Software*,
4415 vol. 26, no. 6, pp. 27–34, 2009, DOI 10.1109/MS.2009.193. ISSN 0740-7459
- 4416 [243] M. P. Robillard and R. Deline, “A field study of API learning obstacles,” *Empirical Software
4417 Engineering*, vol. 16, no. 6, pp. 703–732, 2011, DOI 10.1007/s10664-010-9150-8. ISSN 1382-
4418 3256
- 4419 [244] H. Robinson, J. Segal, and H. Sharp, “Ethnographically-informed empirical studies of soft-
4420 ware practice,” *Information and Software Technology*, vol. 49, no. 6, pp. 540–551, 2007,
4421 DOI 10.1016/j.infsof.2007.02.007. ISSN 0950-5849
- 4422 [245] C. Rosen and E. Shihab, “What are mobile developers asking about? A large scale study
4423 using stack overflow,” *Empirical Software Engineering*, vol. 21, no. 3, pp. 1192–1223, 2016,
4424 DOI 10.1007/s10664-015-9379-3. ISSN 1573-7616
- 4425 [246] W. Rosenberry, D. Kenney, and G. Fisher, *Understanding DCE*. O’Reilly & Associates, Inc.,
4426 1992. ISBN 978-1-56-592005-7
- 4427 [247] A. Rosenfeld, R. Zemel, and J. K. Tsotsos, “The Elephant in the Room,” *arXiv preprint
4428 arXiv:1808.03305*, 2018.
- 4429 [248] A. S. Ross, M. C. Hughes, and F. Doshi-Velez, “Right for the right reasons: Training differentiable
4430 models by constraining their explanations,” in *Proceedings of the 26th International Joint
4431 Conferences on Artificial Intelligence*, Melbourne, Australia, August 2017, DOI 10.24963/ij-
4432 cai.2017/371. ISBN 978-0-99-924110-3. ISSN 1045-0823 pp. 2662–2670.
- 4433 [249] R. J. Rubey and R. D. Hartwick, “Quantitative measurement of program quality,” in *Proceedings
4434 of the 1968 23rd ACM National Conference*. Las Vegas, NV, USA: ACM, August 1968.
4435 DOI 10.1145/800186.810631. ISBN 978-1-45-037486-6 pp. 671–677.
- 4436 [250] J. Rutten, M. Kwiatkowska, G. Norman, and D. Parker, *Mathematical techniques for analyzing
4437 concurrent and probabilistic systems*, ser. CRM Monograph Series, P. Panangaden and F. van
4438 Breugel, Eds. American Mathematical Society, 2004, vol. 23.
- 4439 [251] M. Schwabacher and P. Langley, “Discovering communicable scientific knowledge from spatio-
4440 temporal data,” in *Proceedings of the 18th International Conference on Machine Learning*.
4441 Williamstown, MA, USA: Morgan Kaufmann, June 2001. ISBN 978-1-55-860778-1 pp. 489–
4442 496.
- 4443 [252] A. Schwaighofer and N. D. Lawrence, *Dataset shift in machine learning*, J. Quiñonero-Candela
4444 and M. Sugiyama, Eds. Cambridge, MA, USA: The MIT Press, 2008. ISBN 978-0-26-217005-
4445 5
- 4446 [253] D. Sculley, M. E. Otey, M. Pohl, B. Spitznagel, J. Hainsworth, and Y. Zhou, “Detecting
4447 adversarial advertisements in the wild,” in *Proceedings of the 17th ACM SIGKDD International
4448 Conference on Knowledge Discovery and Data Mining*, ACM. San Diego, CA, USA: ACM,
4449 August 2011. DOI 10.1145/2020408.2020455, pp. 274–282.
- 4450 [254] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J. F.
4451 Crespo, and D. Dennison, “Hidden technical debt in machine learning systems,” in *Proceedings
4452 of the 29th Conference on Neural Information Processing Systems*, Montreal, QC, Canada,
4453 December 2015. ISBN 0262017091, 9780262017091. ISSN 1049-5258 pp. 2503–2511.
- 4454 [255] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-CAM:
4455 Visual Explanations from Deep Networks via Gradient-Based Localization,” *International
4456 Journal of Computer Vision*, pp. 618–626, 2019, DOI 10.1007/s11263-019-01228-7. ISSN
4457 1573-1405
- 4458 [256] S. Sen and L. Knight, “A genetic prototype learner,” in *Proceedings of the International Joint
4459 Conference on Artificial Intelligence*. Montreal, QC, Canada: Morgan Kaufmann, August
4460 1995, pp. 725–733.
- 4461 [257] C. E. Shannon and W. Weaver, “The mathematical theory of communication,” *The Bell
4462 System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948, DOI 10.1002/j.1538-
4463 7305.1948.tb01338.x.

- 4464 [258] M. Shaw, "Writing good software engineering research papers," in *Proceedings of the 25th*
4465 *International Conference on Software Engineering*. Portland, OR, USA: IEEE, May 2003.
4466 ISBN 978-0-76-951877-0 pp. 726–736.
- 4467 [259] M. Shepperd, "Replication studies considered harmful," in *Proceedings of the 40th Inter-*
4468 *national Conference on Software Engineering*. Gothenburg, Sweden: ACM, May 2018.
4469 DOI 10.1145/3183399.3183423. ISBN 978-1-45-035662-6. ISSN 0270-5257 pp. 73–76.
- 4470 [260] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*. New York,
4471 NY, USA: Chapman and Hall/CRC, 2004. DOI 10.4324/9780203489536.
- 4472 [261] L. Si and J. Callan, "A semisupervised learning method to merge search engine results,"
4473 *ACM Transactions on Information Systems*, vol. 21, no. 4, pp. 457–491, October 2003,
4474 DOI 10.1145/944012.944017. ISSN 1046-8188
- 4475 [262] J. Singer, S. E. Sim, and T. C. Lethbridge, "Software engineering data collection for field
4476 studies," in *Guide to Advanced Empirical Software Engineering*, F. Shull, J. Singer, and D. I. K.
4477 Sjøberg, Eds. Springer, November 2007, ch. 1, pp. 9–34. ISBN 978-1-84-800043-8
- 4478 [263] S. Singh, M. T. Ribeiro, and C. Guestrin, "Programs as Black-Box Explanations," *arXiv preprint*
4479 *arXiv:1611.07579*, November 2016.
- 4480 [264] V. S. Sinha, S. Mani, and M. Gupta, "Exploring activeness of users in QA forums," in *Proceed-*
4481 *ings of the 10th Working Conference on Mining Software Repositories*. San Francisco, CA,
4482 USA: IEEE, May 2013. DOI 10.1109/MSR.2013.6624010. ISBN 978-1-46-732936-1. ISSN
4483 2160-1852 pp. 77–80.
- 4484 [265] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classifi-
4485 cation tasks," *Information Processing and Management*, vol. 45, no. 4, pp. 427–437, 2009,
4486 DOI 10.1016/j.ipm.2009.03.002. ISSN 0306-4573
- 4487 [266] I. Sommerville, *Software Engineering*, 9th ed. Boston, MA, USA: Addison-Wesley, 2011.
4488 ISBN 978-0-13-703515-1
- 4489 [267] R. Stevens, J. Ganz, V. Filkov, P. Devanbu, and H. Chen, "Asking for (and about) permissions
4490 used by Android apps," in *Proceedings of the 10th Working Conference on Mining Software*
4491 *Repositories*. San Francisco, CA, USA: IEEE, May 2013. ISBN 978-1-46-732936-1 pp. 31–40.
- 4492 [268] J. Su, D. V. Vargas, and K. Sakurai, "One Pixel Attack for Fooling Deep Neural Net-
4493 works," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019,
4494 DOI 10.1109/TEVC.2019.2890858. ISSN 1941-0026
- 4495 [269] G. H. Subramanian, J. Nosek, S. P. Raghunathan, and S. S. Kanitkar, "A comparison of
4496 the decision table and tree," *Communications of the ACM*, vol. 35, no. 1, pp. 89–94, 1992,
4497 DOI 10.1145/129617.129621. ISSN 1557-7317
- 4498 [270] S. Subramanian, L. Inozemtseva, and R. Holmes, "Live API documentation," in *Proceedings*
4499 *of the 36th International Conference on Software Engineering*. Hyderabad, India: ACM, May
4500 2014. DOI 10.1145/2568225.2568313. ISSN 0270-5257 pp. 643–652.
- 4501 [271] S. Sun, W. Pan, and L. L. Wang, "A Comprehensive Review of Effect Size Reporting and Inter-
4502 pretting Practices in Academic Journals in Education and Psychology," *Journal of Educational*
4503 *Psychology*, vol. 102, no. 4, pp. 989–1004, 2010, DOI 10.1037/a0019507. ISSN 0022-0663
- 4504 [272] D. Szafron, P. Lu, R. Greiner, D. S. Wishart, B. Poulin, R. Eisner, Z. Lu, J. Anvik, C. Macdonell,
4505 A. Fyshe, and D. Meeuwis, "Proteome Analyst: Custom predictions with explanations in a web-
4506 based tool for high-throughput proteome annotations," *Nucleic Acids Research*, vol. 32, 2004,
4507 DOI 10.1093/nar/gkh485. ISSN 0305-1048
- 4508 [273] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus,
4509 "Intriguing properties of neural networks," in *Proceedings of the 2nd International Conference*
4510 *on Learning Representations*. Banff, AB, Canada: ACM, April 2014.
- 4511 [274] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Archi-
4512 tecture for Computer Vision," in *Proceedings of the 2016 IEEE Computer Society Conference*
4513 *on Computer Vision and Pattern Recognition*. Las Vegas, NV, USA: IEEE, June 2016.
4514 DOI 10.1109/CVPR.2016.308. ISBN 978-1-46-738850-4. ISSN 1063-6919 pp. 2818–2826.
- 4515 [275] M. B. W. Tabor, "Student Proves That S.A.T. Can Be: (D) Wrong," [Online] Available: <https://nyti.ms/2UiKrrd>, New York, NY, USA, February 1997.
- 4516 [276] A. Tahir, A. Yamashita, S. Licorish, J. Dietrich, and S. Counsell, "Can you tell me if it
4517 smells? A study on how developers discuss code smells and anti-patterns in Stack Overflow,"
4518 in *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software*
- 4519

- 4520 *Engineering*. Christchurch, New Zealand: ACM, June 2018. DOI 10.1145/3210459.3210466.
4521 ISBN 978-1-45-036403-4 pp. 68–78.
- 4522 [277] G. Tassey, *The economic impacts of inadequate infrastructure for software testing*. National
4523 Institute of Standards and Technology, September 2002. DOI 10.1080/10438590500197315.
4524 ISBN 978-0-75-672618-8
- 4525 [278] R. S. Taylor, “Question-Negotiation and Information Seeking in Libraries,” *College and Re-*
4526 *search Libraries*, vol. 29, no. 3, 1968, DOI 10.5860/crl_29_03_178.
- 4527 [279] S. W. Thomas, B. Adams, A. E. Hassan, and D. Blostein, “Studying software evolu-
4528 tion using topic models,” *Science of Computer Programming*, vol. 80, pp. 457–479, 2014,
4529 DOI 10.1016/j.scico.2012.08.003. ISSN 0167-6423
- 4530 [280] S. Thrun, “Is Learning The n-th Thing Any Easier Than Learning The First?” in *Proceedings*
4531 *of the 8th International Conference on Neural Information Processing Systems*. Denver, CO,
4532 USA: MIT Press, November 1996. ISSN 1049-5258 p. 7.
- 4533 [281] C. Treude, O. Barzilay, and M. A. Storey, “How do programmers ask and answer questions
4534 on the web?” in *Proceedings of the 33rd International Conference on Software Engineering*.
4535 Honolulu, HI, USA: ACM, May 2011. DOI 10.1145/1985793.1985907. ISBN 978-1-45-
4536 030445-0. ISSN 0270-5257 pp. 804–807.
- 4537 [282] G. Uddin and F. Khomh, “Automatic Mining of Opinions Expressed About APIs in
4538 Stack Overflow,” *IEEE Transactions on Software Engineering*, February 2019, In Press,
4539 DOI 10.1109/TSE.2019.2900245. ISSN 1939-3520
- 4540 [283] G. Uddin and M. P. Robillard, “How API Documentation Fails,” *IEEE Software*, vol. 32, no. 4,
4541 pp. 68–75, June 2015, DOI 10.1109/MS.2014.80. ISSN 0740-7459
- 4542 [284] M. Usman, R. Britto, J. Börstler, and E. Mendes, “Taxonomies in software engineering: A
4543 Systematic mapping study and a revised taxonomy development method,” *Information and*
4544 *Software Technology*, vol. 85, pp. 43–59, May 2017, DOI 10.1016/j.infsof.2017.01.006. ISSN
4545 0950-5849
- 4546 [285] A. Van Assche and H. Blockeel, “Seeing the forest through the trees learning a comprehensible
4547 model from a first order ensemble,” in *Proceedings of the 17th International Conference on*
4548 *Inductive Logic Programming*. Corvallis, OR, USA: Springer, June 2007. DOI 10.1007/978-
4549 3-540-78469-2_26. ISBN 3-540-078468-3. ISSN 0302-9743 pp. 269–279.
- 4550 [286] B. Venners, “Design by Contract: A Conversation with Bertrand Meyer,” *Artima Developer*,
4551 2003.
- 4552 [287] W. Verbeke, D. Martens, C. Mues, and B. Baesens, “Building comprehensible customer churn
4553 prediction models with advanced rule induction techniques,” *Expert Systems with Applications*,
4554 vol. 38, no. 3, pp. 2354–2364, 2011, DOI 10.1016/j.eswa.2010.08.023. ISSN 0957-4174
- 4555 [288] F. Wachter, Mitterstadt, “EU regulations on algorithmic decision-making and a “right to expla-
4556 nation”,” in *Proceedings of the 2016 ICML Workshop on Human Interpretability in Machine*
4557 *Learning*, New York, NY, USA, June 2016, pp. 26–30.
- 4558 [289] B. Wang, Y. Yao, B. Viswanath, H. Zheng, and B. Y. Zhao, “With great training comes great
4559 vulnerability: Practical attacks against transfer learning,” in *Proceedings of the 27th USENIX*
4560 *Security Symposium*. Baltimore, MD, USA: USENIX Association, July 2018. ISBN 978-1-
4561 93-913304-5 pp. 1281–1297.
- 4562 [290] K. Wang, *Cloud Computing for Machine Learning and Cognitive Applications: A Machine*
4563 *Learning Approach*. Cambridge, MA, USA: MIT Press, 2017. ISBN 978-0-26-203641-2
- 4564 [291] S. Wang, D. Lo, and L. Jiang, “An empirical study on developer interactions in StackOverflow,”
4565 in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. Coimbra, Portugal:
4566 ACM, March 2013. DOI 10.1145/2480362.2480557, pp. 1019–1024.
- 4567 [292] W. Wang and M. W. Godfrey, “Detecting API usage obstacles: A study of iOS and android
4568 developer questions,” in *Proceedings of the 10th Working Conference on Mining Software*
4569 *Repositories*. San Francisco, CA, USA: IEEE, May 2013. DOI 10.1109/MSR.2013.6624006.
4570 ISBN 978-1-46-732936-1. ISSN 2160-1852 pp. 61–64.
- 4571 [293] W. Wang, H. Malik, and M. W. Godfrey, “Recommending Posts concerning API Issues in
4572 Developer Q&A Sites,” in *Proceedings of the 12th Working Conference on Mining Software*
4573 *Repositories*. Florence, Italy: IEEE, May 2015. DOI 10.1109/MSR.2015.28. ISBN 978-0-
4574 7695-5594-2. ISSN 2160-1860 pp. 224–234.

- 4575 [294] R. Watson, "Development and application of a heuristic to assess trends in API documentation," in *Proceedings of the 30th ACM International Conference on Design of Communication*.
4576 Seattle, WA, USA: ACM, October 2012. DOI 10.1145/2379057.2379112. ISBN 978-1-45-
4577 031497-8 pp. 295–302.
- 4579 [295] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D. F. Ferguson, *Web Services Platform*
4580 *Architecture*. Crawfordsville, IN, USA: Prentice-Hall, 2005. ISBN 0-13-148874-0
- 4581 [296] D. Wettschereck, D. W. Aha, and T. Mohri, "A Review and Empirical Evaluation of Feature
4582 Weighting Methods for a Class of Lazy Learning Algorithms," *Artificial Intelligence Review*,
4583 vol. 11, no. 1-5, pp. 273–314, 1997, DOI 10.1007/978-94-017-2053-3_11. ISSN 0269-2821
- 4584 [297] J. Wexler, M. Pushkarna, T. Bolukbasi, M. Wattenberg, F. Viegas, and J. Wilson, "The What-If
4585 Tool: Interactive Probing of Machine Learning Models," *IEEE Transactions on Visualization*
4586 and *Computer Graphics*, vol. 26, no. 1, pp. 56–65, 2019, DOI 10.1109/tvcg.2019.2934619.
4587 ISSN 1077-2626
- 4588 [298] H. Wickham, "A Layered grammar of graphics," *Journal of Computational and Graphical*
4589 *Statistics*, vol. 19, no. 1, pp. 3–28, January 2010, DOI 10.1198/jcgs.2009.07098. ISSN 1061-
4590 8600
- 4591 [299] R. J. Wieringa and J. M. G. Heerkens, "The methodological soundness of requirements en-
4592 gineering papers: A conceptual framework and two case studies," *Requirements Engineering*,
4593 vol. 11, no. 4, pp. 295–307, 2006, DOI 10.1007/s00766-006-0037-6. ISSN 0947-3602
- 4594 [300] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning*
4595 *Tools and Techniques*. Morgan Kaufmann, 2016. DOI 10.1016/c2009-0-19715-5. ISBN
4596 978-0-12-804291-5
- 4597 [301] C. Wohlin and A. Aurum, "Towards a decision-making structure for selecting a research design
4598 in empirical software engineering," *Empirical Software Engineering*, vol. 20, no. 6, pp. 1427–
4599 1455, May 2015, DOI 10.1007/s10664-014-9319-7. ISSN 1573-7616
- 4600 [302] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation*
4601 *in Software Engineering*. Berlin, Heidelberg: Springer, 2012. DOI 10.1007/978-3-642-
4602 29044-2. ISBN 978-3-642-229044-2
- 4603 [303] M. L. Wong and K. S. Leung, *Data Mining Using Grammar Based Genetic Programming and*
4604 *Applications*. Springer, 2002. DOI 10.1007/b116131. ISBN 978-0-79-237746-7
- 4605 [304] X. Yi and K. J. Kochut, "A CP-nets-based design and verification framework for web services
4606 composition," in *Proceedings of the 2004 IEEE International Conference on Web Services*. San
4607 Diego, CA, USA: IEEE, July 2004. DOI 10.1109/icws.2004.1314810. ISBN 0-76-952167-3
4608 pp. 756–760.
- 4609 [305] R. K. Yin, *Case study research and applications: Design and methods*, 6th ed. Los Angeles,
4610 CA, USA: SAGE, 2017. ISBN 978-1-50-633616-9
- 4611 [306] J. Zahálka and F. Železný, "An experimental test of Occam's razor in classification," *Machine*
4612 *Learning*, vol. 82, no. 3, pp. 475–481, 2011, DOI 10.1007/s10994-010-5227-2. ISSN 0885-6125
- 4613 [307] J. Zhang and R. Kasturi, "Extraction of Text Objects in Video Documents: Recent Progress," in
4614 *Proceedings of the 8th International Workshop on Document Analysis Systems*. Nara, Japan:
4615 IEEE, September 2008. DOI 10.1109/das.2008.49, pp. 5–17.
- 4616 [308] B. Zupan, J. Demšar, M. W. Kattan, J. R. Beck, and I. Bratko, "Machine learning for survival
4617 analysis: a case study on recurrence of prostate cancer," *Artificial intelligence in medicine*,
4618 vol. 20, no. 1, pp. 59–75, 2000.
- 4619 [309] M. Zur Muehlen, J. V. Nickerson, and K. D. Swenson, "Developing web services choreography
4620 standards - The case of REST vs. SOAP," *Decision Support Systems*, vol. 40, no. 1, pp. 9–29,
4621 July 2005, DOI 10.1016/j.dss.2004.04.008. ISSN 0167-9236

4622

4623

List of Online Artefacts

4624

4625 The online artefacts listed below have been downloaded and stored on the Deakin
4626 Research Data Store (RDS) for archival purposes at the following location:

4627 **RDS29448-Alex-Cummaudo-PhD/datasets/webrefs**

- 4628 [310] Affectiva, Inc., “Home - Affectiva : Affectiva,” <http://bit.ly/36sgbMM>, 2018, accessed: 15
4629 October 2018.
- 4630 [311] Amazon Web Services, Inc., “Detecting Labels in an Image,” <https://amzn.to/2TBNTa>, 2018,
4631 accessed: 28 August 2018.
- 4632 [312] ——, “Detecting Objects and Scenes,” <https://amzn.to/2TDed5V>, 2018, accessed: 28 August
4633 2018.
- 4634 [313] ——, “Amazon Rekognition,” <https://amzn.to/2TyT2BL>, 2018, accessed: 13 September 2018.
- 4635 [314] Beijing Geling Shentong Information Technology Co., Ltd., “DeepGlint,” <http://bit.ly/2uIHdPS>, 2018, accessed: 3 April 2019.
- 4636 [315] Beijing Kuangshi Technology Co., Ltd., “Megvii,” <http://bit.ly/2WJYFzk>, 2018, accessed: 3
4637 April 2019.
- 4638 [316] Clarifai, Inc., “Enterprise AI Powered Computer Vision Solutions | Clarifai,” <http://bit.ly/2TB3kSa>, 2018, accessed: 13 September 2018.
- 4639 [317] CloudSight, Inc., “Image Recognition API & Visual Search Results | CloudSight AI,” <http://bit.ly/2UmNPCw>, 2018, accessed: 13 September 2018.
- 4640 [318] Cognitec Systems GmbH, “The face recognition company - Cognitec,” <http://bit.ly/38VguBB>,
4641 2018, accessed: 15 October 2018.
- 4642 [319] A. Cummaudo, <http://bit.ly/2KlyhcD>, 2019, accessed: 27 March 2019.
- 4643 [320] ——, <http://bit.ly/2G7saFJ>, 2019, accessed: 27 March 2019.
- 4644 [321] ——, <http://bit.ly/2G5ZEEe>, 2019, accessed: 27 March 2019.
- 4645 [322] ——, “ICSE 2020 Submission #564 Supplementary Materials,” <http://bit.ly/2Z8zOKW>, 2019.
- 4646 [323] ——, <http://bit.ly/2G6ZOeC>, 2019, accessed: 27 March 2019.
- 4647 [324] Deep AI, Inc., “DeepAI: The front page of A.I. | DeepAI,” <http://bit.ly/2TBNYgf>, 2018, ac-
4648 cessed: 26 September 2018.
- 4649 [325] Google LLC, “Detect Labels | Google Cloud Vision API Documentation | Google Cloud,”
4650 <http://bit.ly/2TD5kcy>, 2018, accessed: 28 August 2018.
- 4651 [326] ——, “Class EntityAnnotation | Google.Cloud.Vision.V1,” <http://bit.ly/2TD5fpg>, 2018, ac-
4652 cessed: 28 August 2018.
- 4653 [327] ——, “Vision API - Image Content Analysis | Cloud Vision API | Google Cloud,” <http://bit.ly/2TD9mBs>, 2018, accessed: 13 September 2018.

- 4658 [328] ——, “Open Images Dataset V4,” <http://bit.ly/2Ry2vvF>, 2019, accessed: 9 November 2018.
- 4659 [329] Guangzhou Tup Network Technology, “TupuTech,” <http://bit.ly/2uF4IsN>, 2018, accessed: 3 April 2019.
- 4660 [330] Imagga Technologies, “Imagga - powerful image recognition APIs for automated categorization & tagging in the cloud and on-premises,” <http://bit.ly/2TxsyRe>, 2018, accessed: 13 September 2018.
- 4661 [331] International Business Machines Corporation, “Watson Visual Recognition - Overview | IBM,” <https://ibm.co/2TBNIO4>, 2018, accessed: 13 September 2018.
- 4662 [332] ——, “Watson Tone Analyzer,” <https://ibm.co/37w3y4A>, 2019, accessed: 25 January 2019.
- 4663 [333] Kairos AR, Inc., “Kairos: Serving Businesses with Face Recognition,” <http://bit.ly/30WHGNs>, 2018, accessed: 15 October 2018.
- 4664 [334] Microsoft Corporation, “azure-sdk-for-java/ImageTag.java,” <http://bit.ly/38IDPWU>, 2018, accessed: 28 August 2018.
- 4665 [335] ——, “Image Processing with the Computer Vision API | Microsoft Azure,” <http://bit.ly/2YqhkS6>, 2018, accessed: 13 September 2018.
- 4666 [336] ——, “How to call the Computer Vision API,” <http://bit.ly/2TD5oJk>, 2018, accessed: 28 August 2018.
- 4667 [337] ——, “What is Computer Vision?” <http://bit.ly/2TDgUnU>, 2018, accessed: 28 August 2018.
- 4668 [338] SenseTime, “SenseTime,” <http://bit.ly/2WH6Rjf>, 2018, accessed: 3 April 2019.
- 4669 [339] Shanghai Yitu Technology Co., Ltd., “Yitu Technology,” <http://bit.ly/2uGvxgf>, 2018, accessed: 3 April 2019.
- 4670 [340] Stack Overflow User #1008563 ‘samiles’, “AWS Rekognition PHP SDK gives invalid image encoding error,” <http://bit.ly/31Sgpec>, 2019, accessed: 22 June 2019.
- 4671 [341] Stack Overflow User #10318601 ‘reza naderii’, “google cloud vision category detecting,” <http://bit.ly/31Uf32t>, 2019, accessed: 22 June 2019.
- 4672 [342] Stack Overflow User #10729564 ‘gabgob’, “Multiple Google Vision OCR requests at once?” <http://bit.ly/31P09dU>, 2019, accessed: 22 June 2019.
- 4673 [343] Stack Overflow User #1453704 ‘deeptimancode’, “Human body part detection in Android,” <http://bit.ly/31T5pxd>, 2019, accessed: 22 June 2019.
- 4674 [344] Stack Overflow User #174602 ‘geekyaleks’, “aws Rekognition not initializing on iOS,” <http://bit.ly/31UeqG9>, 2019, accessed: 22 June 2019.
- 4675 [345] Stack Overflow User #2251258 ‘James Dorfman’, “All GoogleVision label possibilities?” <http://bit.ly/31R4FZi>, 2019, accessed: 22 June 2019.
- 4676 [346] Stack Overflow User #2521469 ‘Hillary Sanders’, “Is there a full list of potential labels that Google’s Vision API will return?” <http://bit.ly/2KNnJSB>, 2019, accessed: 22 June 2019.
- 4677 [347] Stack Overflow User #2604150 ‘user2604150’, “Google Vision Accent Character Set NodeJs,” <http://bit.ly/31TsVdp>, 2019, accessed: 22 June 2019.
- 4678 [348] Stack Overflow User #3092947 ‘Mark Bench’, “Google Cloud Vision OCR API returning incorrect values for bounding box/vertices,” <http://bit.ly/31UeZjf>, 2019, accessed: 22 June 2019.
- 4679 [349] Stack Overflow User #3565255 ‘CSquare’, “Vision API topicality and score always the same,” <http://bit.ly/2TD5As2>, 2019, accessed: 22 June 2019.
- 4680 [350] Stack Overflow User #4748115 ‘Latifa Al-jifry’, “similar face recognition using google cloud vision API in android studio,” <http://bit.ly/31WhMZy>, 2019, accessed: 22 June 2019.
- 4681 [351] Stack Overflow User #4852910 ‘Gaurav Mathur’, “Amazon Rekognition Image caption,” <http://bit.ly/31P08qm>, 2019, accessed: 22 June 2019.
- 4682 [352] Stack Overflow User #5294761 ‘Eury Pérez Beltré’, “Specify language for response in Google Cloud Vision API,” <http://bit.ly/31SsUGG>, 2019, accessed: 22 June 2019.
- 4683 [353] Stack Overflow User #549312 ‘GroovyDotCom’, “Image Selection for Training Visual Recognition,” <http://bit.ly/31W8lcw>, 2019, accessed: 22 June 2019.
- 4684 [354] Stack Overflow User #5809351 ‘J.Doe’, “How to confidently validate object detection results returned from Google Cloud Vision,” <http://bit.ly/31UcCNy>, 2019, accessed: 22 June 2019.
- 4685 [355] Stack Overflow User #5844927 ‘Amit Pawar’, “Google cloud Vision and Clarifai doesn’t Support tagging for 360 degree images and videos,” <http://bit.ly/31StuEm>, 2019, accessed: 22 June 2019.
- 4686 [356] Stack Overflow User #5924523 ‘Akash Dathan’, “Can i give aspect ratio in Google Vision api?” <http://bit.ly/2KSJwsp>, 2019, accessed: 22 June 2019.

- 4714 [357] Stack Overflow User #6210900 ‘Mike Grommet’, “Are the Cloud Vision API limits in documentation correct?” <http://bit.ly/31SsNLg>, 2019, accessed: 22 June 2019.
- 4715 [358] Stack Overflow User #6649145 ‘I. Sokolyk’, “How to get a position of custom object on image using vision recognition api,” <http://bit.ly/3210Q49>, 2019, accessed: 22 June 2019.
- 4716 [359] Stack Overflow User #6841211 ‘NigelJL’, “Google Cloud Vision - Numbers and Numerals OCR,” <http://bit.ly/31P07mi>, 2019, accessed: 22 June 2019.
- 4717 [360] Stack Overflow User #7064840 ‘Josh’, “Google Cloud Vision fails at batch annotate images. Getting Netty Shaded ClosedChannelException error,” <http://bit.ly/31UrBH9>, 2019, accessed: 22 June 2019.
- 4718 [361] Stack Overflow User #7187987 ‘tuanars10’, “Adding a local path to Microsoft Face API by Python,” <http://bit.ly/2KLeMt3>, 2019, accessed: 22 June 2019.
- 4719 [362] Stack Overflow User #7219743 ‘Davide Biraghi’, “Google Vision API does not recognize single digits,” <http://bit.ly/31Ws1Nj>, 2019, accessed: 22 June 2019.
- 4720 [363] Stack Overflow User #738248 ‘lavuy’, “Meaning of score in Microsoft Cognitive Service’s Entity Linking API,” <http://bit.ly/2TD9vVw>, 2019, accessed: 22 June 2019.
- 4721 [364] Stack Overflow User #7604576 ‘Alagappan Narayanan’, “Text extraction - line-by-line,” <http://bit.ly/31Yc21s>, 2019, accessed: 22 June 2019.
- 4722 [365] Stack Overflow User #7692297 ‘lucas’, “Can Google Cloud Vision generate labels in Spanish via its API?” <http://bit.ly/31UcBsY>, 2019, accessed: 22 June 2019.
- 4723 [366] Stack Overflow User #7896427 ‘David mark’, “Google Api Vision, ““before_request”” error,” <http://bit.ly/31Z27Zt>, 2019, accessed: 22 June 2019.
- 4724 [367] Stack Overflow User #8210103 ‘Cosmin-Ioan Leferman’, “Google Vision API text detection strange behaviour - Javascript,” <http://bit.ly/31Ucyxi>, 2019, accessed: 22 June 2019.
- 4725 [368] Stack Overflow User #8411506 ‘AsSportac’, “How can we find an exhaustive list (or graph) of all logos which are effectively recognized using Google Vision logo detection feature?” <http://bit.ly/31Z27IX>, 2019, accessed: 22 June 2019.
- 4726 [369] Stack Overflow User #8594124 ‘God Himself’, “How to set up AWS mobile SDK in iOS project in Xcode,” <http://bit.ly/31St2pE>, 2019, accessed: 22 June 2019.
- 4727 [370] Stack Overflow User #9006896 ‘Dexter Intelligence’, “Getting wrong text sequence when image scanned by offline google mobile vision API,” <http://bit.ly/31Sgr5O>, 2019, accessed: 22 June 2019.
- 4728 [371] Stack Overflow User #9913535 ‘Sahil Mehra’, “Google Vision API: ModuleNotFoundError: module not found ‘google.oauth2’,” <http://bit.ly/31VIZfU>, 2019, accessed: 22 June 2019.
- 4729 [372] Symisc Systems, S.U.A.R.L, “Computer Vision & Media Processing APIs | PixLab,” <http://bit.ly/2UlkW9K>, 2018, accessed: 13 September 2018.
- 4730 [373] Talkwalker Inc., “Image Recognition - Talkwalker,” <http://bit.ly/2TyT7W5>, 2018, accessed: 13 September 2018.
- 4731 [374] TheySay Limited, “Sentiment Analysis API | TheySay,” <http://bit.ly/37AzTHI>, 2019, accessed: 25 January 2019.

Part IV

Appendices

APPENDIX A

Additional Materials

A.1 On the Development, Documentation and Usage of Web APIs

The development of web application programming interfaces (APIs) (commonly referred to as a *web service*) and web APIs traces its roots back to the early 1990s, where the Open Software Foundation’s distributed computing environment (DCE) introduced a collection of services and tools for developing and maintaining distributed systems using a client/server architecture [246]. This framework used the synchronous communication paradigm remote procedure calls (RPCs) first introduced by Nelson [206] that allows procedures to be called in a remote address space as if it were local. Its communication paradigm, DCE/RPC [213], enables developers to write distributed software with underlying network code abstracted away. To bridge remote DCE/RPCs over components of different operating systems and languages, an interface definition language (IDL) document served as the common service contract or *service interface* for software components.

This important leap toward language-agnostic distributed programming paved way for XML-RPC, enabling RPCs over HTTP (and thus the Web) encoded using XML (instead of octet streams [213]). As new functionality was introduced, this lead to the natural development of the Simple Object Access Protocol (SOAP), the backbone messaging connector for web service (WS) applications, a realisation of the service-oriented architecture (SOA) [54] pattern. The SOA pattern prescribes that services are offered by service providers and consumed by service consumers in a platform- and language-agnostic manner and are used in large-scale enterprise systems (e.g., banking, health). Key to the SOA pattern is that a service’s quality attributes (see Section 2.1) can be specified and guaranteed using a service-level agreement (SLA) whereby the consumer and provider agree upon a set level of service, which in some cases are legally binding [23]. This agreement can be measured using quality of service (QoS) parameters met by the service provider during the transportation layer (e.g., response time, cost of leasing resources, reliability guarantees, system availability and trust/security assurance [290, 295]). These attributes are included within SOAP headers; thus, QoS aspects are independent from the transport layer and instead exist at the application layer [220]. The IDL of SOAP is Web Services Description Language (WSDL), providing a description of how the web service is invoked, what parameters to expect, and what data structures are returned.

While it is rich in metadata and verbosity, discussions on whether this was a benefit or drawback came about the mid-2000s [220, 309] whether the amount of data transfer paid off (especially for mobile clients where data usage was scarce). Developer usability for debugging the SOAP ‘envelopes’ (messages POSTed over HTTP to the service provider component) was difficult, both due to the nature of XML’s wordiness and difficulty to test (by sending POST requests) in-browser. As a simple example, 25 lines (794 bytes) of HTTP communication is transferred to request a customer’s name from a record using SOAP (Listings A.1 and A.2).

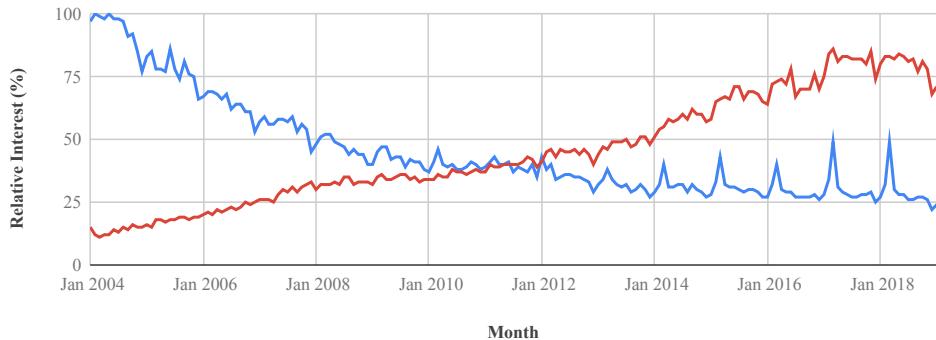


Figure A.1: Worldwide search interest for SOAP (blue) and REST (red) since 2004. Source: Google Trends.

Listing A.1: A SOAP HTTP POST consumer request to retrieve customer record #43456 from a web service provider. Source: [17].

```

1 | POST /customers HTTP/1.1
2 | Host: www.example.org
3 | Content-Type: application/soap+xml; charset=utf-8
4 |
5 | <?xml version="1.0"?>
6 | <soap:Envelope
7 |   xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
8 |   <soap:Body>
9 |     <m:GetCustomer
10 |       xmlns:m="http://www.example.org/customers">
11 |         <m:CustomerId>43456</m:CustomerId>
12 |       </m:GetCustomer>
13 |     </soap:Body>
14 |   </soap:Envelope>
```

Listing A.2: The SOAP HTTP service provider response for Listing A.1. Source: [17].

```

1 | HTTP/1.1 200 OK
2 | Content-Type: application/soap+xml; charset=utf-8
3 |
4 | <?xml version='1.0' ?>
5 | <env:Envelope
6 |   xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
7 |   <env:Body>
8 |     <m:GetCustomerResponse
9 |       xmlns:m="http://www.example.org/customers">
10 |         <m:Customer>Foobar Quux, inc</m:Customer>
11 |       </m:GetCustomerResponse>
12 |     </env:Body>
13 |   </env:Envelope>
```

SOAP uses the architectural principle that web services (or the applications they provide) should remain *outside* the web, using HTTP only as a tunnelling protocol to enable remote communication [220]. That is, the HTTP is considered as a transport protocol solely. In 2000, Fielding [92] introduced REpresentational State Transfer (REST), which instead approaches the web as a medium to publish data (i.e., HTTP is part of the *application* layer instead). Hence, applications become amalgamated into of the Web. Fielding bases REST on four key principles:

- **URIs identify resources.** Resources and services have a consistent global address space that aides in their discovery via URIs [27].
- **HTTP verbs manipulate those resources.** Resources are manipulated using the four consistent CRUD verbs provided by HTTP: POST, GET, PUT, DELETE.
- **Self-descriptive messages.** Each request provides enough description and context for the server to process that message.
- **Resources are stateless.** Every interaction with a resource is stateless.

Consider the equivalent example of Listings A.1 and A.2 but in a RESTful architecture (Listings A.3 and A.4) and it is clear why this style has grown more popular with developers (as we highlight in Figure A.1). Developers have since embraced RESTful API development, though the major drawback of RESTful services is its lack of a uniform IDL to facilitate development (though it is possible to achieve this using Web Application Description Language (WADL) [182]). Therefore, no RESTful service uses a standardised response document or invocation syntax. While there are proposals, such as WADL [115], RAML¹, API Blueprint², and the OpenAPI³ specification (initially based on Swagger⁴), there is still no consensus as there was for SOAP and convergence of these IDLs is still underway.

Listing A.3: An equivalent HTTP consumer request to that of Listing A.1, but using REST. Source: [17].

```
1 | GET /customers/43456 HTTP/1.1
2 | Host: www.example.org
```

Listing A.4: The REST HTTP service provider response for Listing A.3.

```
1 | HTTP/1.1 200 OK
2 | Content-Type: application/json; charset=utf-8
3 |
4 | {"Customer": "Foobar Quux, inc"}
```

¹<https://raml.org> last accessed 25 January 2019.

²<https://apiblueprint.org> last accessed 25 January 2019.

³<https://www.openapis.org> last accessed 25 January 2019.

⁴<https://swagger.io> last accessed 25 January 2019.

A.2 Additional Figures

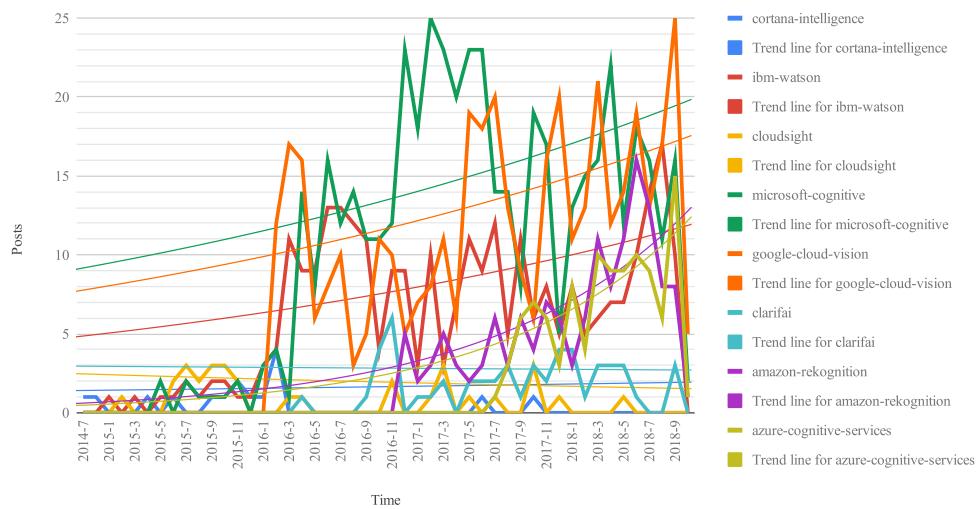
The following figures are listed in this section:

- List
- of
- every
- additional
- figure (and page ref)
- and
- why
- it
- is
- relevant.

*< todo: Include causal model: draft1 & 4 >
< todo: Include technical domain model >
< todo: Include threshy: decision boundary > < todo: Include threshy: domain
model > < todo: Include threshy: sequence diagrams >
< todo: Include ICSE workflow >
< todo: Include architectural model: new brc > < todo: Include architectural
model: evolution > < todo: Include architectural model: creation of request >
< todo: Include architectural model: evolution > < todo: Include architectural
model: class diagram > < todo: Include architectural model: evolution > < todo:
Include architectural model: overall state diagram > < todo: Include architectural
model: page6 > < todo: Include architectural model: page7 >*

Figure A.2: A Broad Range of artificial intelligence (AI)-Based Products And Services Is Already Visible. (From [176].)

Category	Sample vendors and products	Typical use cases
Embedded AI Expert assistants leverage AI technology embedded in platforms and solutions.	<ul style="list-style-type: none"> Amazon: Alexa Apple: Siri Facebook: Messenger Google: Google Assistant (and more) Microsoft: Cortana Salesforce: MetaMind (acquisition) 	<ul style="list-style-type: none"> Personal assistants for search, simple inquiry, and growing as expert assistance (composed problems, not just search) Available on mobile platforms, devices, the internet of things Voice, image recognition, various levels of NLP sophistication Bots, agents
AI point solutions Point solutions provide specialized capabilities for NLP, vision, speech, and reasoning.	<ul style="list-style-type: none"> 24[7]: 24[7] Admantx: Admantx Affectiva: Affdex Assist: AssistDigital Automated Insights: Wordsmith Beyond Verbal: Beyond Verbal Expert System: Cogito HPE: Haven OnDemand IBM: Watson Analytics, Explorer, Advisor Narrative Science: Quill Nuance: Dragon Salesforce: MetaMind (acquisition) Wise.io: Wise Support 	<ul style="list-style-type: none"> Semantic text, facial/visual recognition, voice intonation, intelligent narratives Various levels of NLP from brief text messaging, chat/conversational messaging, full complex text understanding Machine learning, predictive analytics, text analytics/mining Knowledge management and search Expert advisors, reasoning tools Customer service, support APIs
AI platforms Platforms that offer various AI tech, including (deep) machine learning, as tools, APIs, or services to build solutions.	<ul style="list-style-type: none"> CognitiveScale: Engage, Amplify Digital Reasoning: Synthesys Google: Google Cloud Machine Learning IBM: Watson Developers, Watson Knowledge Studio Intel: Saffron Natural Intelligence IPsoft: Amelia, Apollo, IP Center Microsoft: Cortana Intelligence Suite Nuance: 360 platform Salesforce: Einstein Wipro: Holmes 	<ul style="list-style-type: none"> APIs, cloud services, on-premises for developers to build AI solutions Insights/advice building Rule-based reasoning Vertical domain advisors (e.g., fraud detection in banking, financial advisors, healthcare) Cognitive services and bots
Deep learning Platforms, advanced projects, and algorithms for deep learning.	<ul style="list-style-type: none"> Amazon: FireFly Google: TensorFlow/DeepMind LoopAI Labs: LoopAI Numenta: Grok Vicarious: Vicarious 	<ul style="list-style-type: none"> Deep learning neural networks for categorization, clustering, search, image recognition, NLP, and more Location pattern recognition Brain neocortex simulation

Figure A.3: Increasing interest on Stack Overflow for CVSS.

A.3 Reference Architecture Source Code

(todo: ICVS benchmarker code)

APPENDIX B

Authorship Statements

Deakin University Authorship Procedure

Schedule A: Authorship Statement

1. Details of the publication and executive author

Title of publication	Losing Confidence in Quality: Unspoken Evolution of Computer Vision Services
Publication details	Presented at the 35th IEEE International Conference on Software Maintenance and Evolution, Cleveland, USA, 2019
Name of executive author	Alex Cummaudo
School/Institute/Division if at Deakin Organisation and address if non-Deakin	Applied Artificial Intelligence Institute
Email or phone	ca@deakin.edu.au

2. Inclusion of publication in a thesis

Is it intended to include this publication in a higher degree by research (HDR) thesis?
*If Yes, please complete Section 3
 If No, go straight to Section 4.*

3. HDR thesis author's declaration

Name of HDR thesis author if different from above. <i>(If the same, write "as above")</i>	As above
School/Institute/Division if at Deakin	Applied Artificial Intelligence Institute
Thesis title	Provenance in Large-Scale Artificial Intelligence Solutions
If there are multiple authors, give a full description of HDR thesis author's contribution to the publication.	See page 2

I declare that the above is an accurate description of my contribution to this paper, and the contributions of other authors are as described below.

Signed:



Dated: 22 July 2019

4. Description of all author contributions

Name and affiliation of author 1	Alex Cummaudo Applied Artificial Intelligence Institute Deakin University
Contribution of author 1	Alex Cummaudo initiated the conception of the project. Additionally, he designed a detailed methodology, conducted all data collection via a data-collection instrument he designed and implemented and performed a majority of data analysis. He drafted the full manuscript and made further revisions, modifications and prepared the camera ready version for publication in the conference proceedings.
Name and affiliation of author 2	Rajesh Vasa Applied Artificial Intelligence Institute Deakin University
Contribution of author 2	Rajesh Vasa contributed to the initial conception of this project by providing high-level guidance over overview of what the project and its experiments should comprise of. Rajesh also contributed to detailed revisions of the initial manuscripts, and assisted in advising Alex Cummaudo on improved analytical insight into the collected results. Rajesh Vasa also assisted in shaping the paper to specifically target the conference audience. Rajesh Vasa is the primary supervisor of Alex Cummaudo.
Name and affiliation of author 3	John Grundy Faculty of Information Technology Monash University
Contribution of author 3	John Grundy provided high-level oversight of the project. He contributed to detailed reviews of the methodology and manuscript. John Grundy is the external supervisor of Alex Cummaudo.
Name and affiliation of author 4	Mohamed Abdelrazek School of Information Technology Deakin University
Contribution of author 4	Mohamed Abdelrazek made final edits and suggestions to the final draft of the manuscript before submitting for peer review. Mohamed Abdelrazek is an associate supervisor of Alex Cummaudo.
Name and affiliation of author 5	Andrew Cain School of Information Technology Deakin University
Contribution of author 5	Andrew Cain made edits and suggestions to the abstract and introduction paragraphs of the manuscript. Andrew Cain is an associate supervisor of Alex Cummaudo.

5. Author declarations

I agree to be named as one of the authors of this work, and confirm:

- i. that I have met the authorship criteria set out in the Deakin University Research Conduct Policy,
- ii. that there are no other authors according to these criteria,
- iii. that the description in Section 4 of my contribution(s) to this publication is accurate,
- iv. that the data on which these findings are based are stored as set out in Section 7 below.

If this work is to form part of an HDR thesis as described in Sections 2 and 3, I further

- v. consent to the incorporation of the publication into the candidate's HDR thesis submitted to Deakin University and, if the higher degree is awarded, the subsequent publication of the thesis by the university (subject to relevant Copyright provisions).

Author 1

Alex Cummaudo

Signed: 
Dated: 22 July 2019

Author 2

Rajesh Vasa

Signed: 
Dated: 22 July 2019

Author 3

John Grundy

Signed: 
Dated: 22 July 2019

Author 4

Mohamed Abdelrazek

Signed: 
Dated: 22 July 2019

Author 5

Andrew Cain

Signed: 
Dated: 22 July 2019

6. Other contributor declarations

There are no other contributors for this publication to declare.

7. Data storage

The original data for this project are stored in the following locations. (The locations must be within an appropriate institutional setting. If the executive author is a Deakin staff member and data are stored outside Deakin University, permission for this must be given by the Head of Academic Unit within which the executive author is based.)

Data format	Comma separated values (CSV), iPython Notebook
Storage location	Deakin University Research Data Store (RDS) Location: RDS29448-Alex-Cummaudo-PhD/results/icsme19

8. Additional notices

This form must be retained by the executive author, within the school or institute in which they are based.

If the publication is to be included as part of an HDR thesis, a copy of this form must be included in the thesis with the publication.

Deakin University Authorship Procedure

Schedule A: Authorship Statement

1. Details of the publication and executive author

Title of publication	What should I document? A preliminary systematic mapping study into API documentation knowledge
Publication details	Presented at the 13th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), Porto de Galinhas, Brazil, 2019
Name of executive author	Alex Cummaudo
School/Institute/Division if at Deakin	Applied Artificial Intelligence Institute
Organisation and address if non-Deakin	
Email or phone	ca@deakin.edu.au

2. Inclusion of publication in a thesis

Is it intended to include this publication in a higher degree by research (HDR) thesis? Yes
*If Yes, please complete Section 3
If No, go straight to Section 4.*

3. HDR thesis author's declaration

Name of HDR thesis author if different from above. <i>(If the same, write "as above")</i>	As above
School/Institute/Division if at Deakin	Applied Artificial Intelligence Institute
Thesis title	Provenance in Large-Scale Artificial Intelligence Solutions
If there are multiple authors, give a full description of HDR thesis author's contribution to the publication.	See page 2

I declare that the above is an accurate description of my contribution to this paper, and the contributions of other authors are as described below.

Signed:



Dated: 22 July 2019

4. Description of all author contributions

Name and affiliation of author 1	Alex Cummaudo Applied Artificial Intelligence Institute Deakin University
Contribution of author 1	Alex Cummaudo devised the conception of this project and the intended objectives and hypotheses. Additionally, he designed a detailed methodology, conducted data collection with a custom tool he wrote himself and performed analysis. He drafted the manuscript and made further revisions, modifications and prepared the camera ready version for publication in the conference proceedings.
Name and affiliation of author 2	Rajesh Vasa Applied Artificial Intelligence Institute Deakin University
Contribution of author 2	Rajesh Vasa contributed to the initial conception of this project by providing high-level guidance over overview of what the project and its experiments should comprise of. Rajesh also contributed to detailed revisions of the initial manuscripts, and assisted in advising Alex Cummaudo on improved analytical insight into the collected results. Rajesh Vasa also assisted in shaping the paper to specifically target the conference audience. Rajesh Vasa is the primary supervisor of Alex Cummaudo.
Name and affiliation of author 3	John Grundy Faculty of Information Technology Monash University
Contribution of author 3	John Grundy provided high-level oversight of the project. He contributed to detailed reviews of the methodology and manuscript. John Grundy is the external supervisor of Alex Cummaudo.

5. Author declarations

I agree to be named as one of the authors of this work, and confirm:

- i. that I have met the authorship criteria set out in the Deakin University Research Conduct Policy,
- ii. that there are no other authors according to these criteria,
- iii. that the description in Section 4 of my contribution(s) to this publication is accurate,
- iv. that the data on which these findings are based are stored as set out in Section 7 below.

If this work is to form part of an HDR thesis as described in Sections 2 and 3, I further

- v. consent to the incorporation of the publication into the candidate's HDR thesis submitted to Deakin University and, if the higher degree is awarded, the subsequent publication of the thesis by the university (subject to relevant Copyright provisions).

Author 1

Alex Cummaudo

Signed: 
Dated: 22 July 2019

Author 2

Rajesh Vasa

Signed: 
Dated: 22 July 2019

Author 3

John Grundy

Signed: 
Dated: 22 July 2019

6. Other contributor declarations

There are no other contributors for this publication to declare.

7. Data storage

The original data for this project are stored in the following locations. (The locations must be within an appropriate institutional setting. If the executive author is a Deakin staff member and data are stored outside Deakin University, permission for this must be given by the Head of Academic Unit within which the executive author is based.)

Data format	Comma separated values (CSV), Portable Document Format (PDF)
Storage location	Deakin University Research Data Store (RDS) Location: RDS29448-Alex-Cummaudo-PhD/results/esem19

8. Additional notices

This form must be retained by the executive author, within the school or institute in which they are based.

If the publication is to be included as part of an HDR thesis, a copy of this form must be included in the thesis with the publication.

Deakin University Authorship Procedure

Schedule A: Authorship Statement

1. Details of the publication and executive author

Title of publication	Merging Intelligent API Responses Using a Proportional Representation Approach
Publication details	Presented at the 19th International Conference on Web Engineering (ICWE), Daejeon, South Korea, 2019
Name of executive author	Tomohiro Otake
School/Institute/Division if at Deakin Organisation and address if non-Deakin	Faculty of Science, Engineering and Built Environment
Email or phone	tomohiro.otake@deakin.edu.au

2. Inclusion of publication in a thesis

Is it intended to include this publication in a higher degree by research (HDR) thesis?
*If Yes, please complete Section 3
 If No, go straight to Section 4.*

3. HDR thesis author's declaration

Name of HDR thesis author if different from above. <i>(If the same, write "as above")</i>	As above
School/Institute/Division if at Deakin	Applied Artificial Intelligence Institute
Thesis title	Provenance in Large-Scale Artificial Intelligence Solutions
If there are multiple authors, give a full description of HDR thesis author's contribution to the publication.	See page 2

I declare that the above is an accurate description of my contribution to this paper, and the contributions of other authors are as described below.

Signed:



Dated: 2 August 2019

4. Description of all author contributions

Name and affiliation of author 1 Tomohiro Otake
Faculty of Science, Engineering and Built Environment
Deakin University

Contribution of author 1 Tomohiro Otake designed a detailed methodology for data collection in the primary experiment of this work. He conducted all data collection via a data-collection instrument he designed and implemented and performed a majority of data analysis. He drafted the full manuscript and made further revisions, modifications and prepared the camera ready version for publication in the conference proceedings.

Name and affiliation of author 2 Alex Cummaudo
Applied Artificial Intelligence Institute
Deakin University

Contribution of author 2 Alex Cummaudo's primary contribution to this work was the conception and writing up of the motivating sections in the manuscript. He additionally contributed to detailed editing of the manuscripting to make further revisions and modifications and implemented reviewer feedback.

Name and affiliation of author 3 Mohamed Abdelrazek
Faculty of Science, Engineering and Built Environment
Deakin University

Contribution of author 3 Mohamed Abdelrazek contributed to the initial conception of this project by providing high-level guidance over overview of what the project and its experiments should comprise of. Mohamed also contributed to detailed revisions of the initial manuscripts, and assisted in advising Tomohiro Otake on improved analytical insight into the collected results, and implementing reviewer feedback.

Name and affiliation of author 4 Rajesh Vasa
Faculty of Science, Engineering and Built Environment
Deakin University

Contribution of author 4 Rajesh Vasa provided high-level oversight of the project. He contributed to detailed reviews of the methodology and manuscript.

Name and affiliation of author 5 John Grundy
Faculty of Information Technology
Monash University

Contribution of author 5 John Grundy provided high-level oversight of the project. He contributed to detailed reviews of the methodology and manuscript.

5. Author declarations

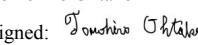
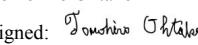
I agree to be named as one of the authors of this work, and confirm:

- i. that I have met the authorship criteria set out in the Deakin University Research Conduct Policy,
- ii. that there are no other authors according to these criteria,
- iii. that the description in Section 4 of my contribution(s) to this publication is accurate,
- iv. that the data on which these findings are based are stored as set out in Section 7 below.

If this work is to form part of an HDR thesis as described in Sections 2 and 3, I further

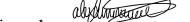
- v. consent to the incorporation of the publication into the candidate's HDR thesis submitted to Deakin University and, if the higher degree is awarded, the subsequent publication of the thesis by the university (subject to relevant Copyright provisions).

Author 1

Tomohiro Ohtake

 Signed: 
 Dated: 2 August 2019

Author 2

Alex Cummaudo

 Signed: 
 Dated: 2 August 2019

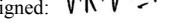
Author 3

Mohamed Abdelrarez

 Signed: 
 Dated: 2 August 2019

Author 4

Rajesh Vasa

 Signed: 
 Dated: 2 August 2019

Author 5

John Grundy

 Signed: 
 Dated: 2 August 2019

6. Other contributor declarations

There are no other contributors for this publication to declare.

7. Data storage

The original data for this project are stored in the following locations. (The locations must be within an appropriate institutional setting. If the executive author is a Deakin staff member and data are stored outside Deakin University, permission for this must be given by the Head of Academic Unit within which the executive author is based.)

Data format	Comma separated values (CSV)
Storage location	Deakin University Research Data Store (RDS) Location: RDS29448-Alex-Cummaudo-PhD/results/icwe19

8. Additional notices

This form must be retained by the executive author, within the school or institute in which they are based.

If the publication is to be included as part of an HDR thesis, a copy of this form must be included in the thesis with the publication.

APPENDIX C

Ethics Clearance



Rajesh Vasa and Alex Cummaudo
Applied Artificial Intelligence Institute (A²I²)
C.c Mohamed Abdelrazek, Andrew Cain

2 May 2019

Dear Rajesh and Alex

STEC-11-2019-CUMMAUDO titled "*Developer opinions towards the importance of web API documentation recommendations*"

Thank you for submitting the above project for consideration by the Faculty Human Ethics Advisory Group (HEAG). The HEAG recognised that the project complies with the National Statement on Ethical Conduct in Human Research (2007) and has approved it. You may commence the project upon receipt of this communication.

The approval period is for three years until **02/05/22**. It is your responsibility to contact the Faculty HEAG immediately should any of the following occur:

- Serious or unexpected adverse effects on the participants
- Any proposed changes in the protocol, including extensions of time
- Any changes to the research team or changes to contact details
- Any events which might affect the continuing ethical acceptability of the project
- The project is discontinued before the expected date of completion.

You will be required to submit an annual report giving details of the progress of your research. Please forward your first annual report on **02/05/20**. Failure to do so may result in the termination of the project. Once the project is completed, you will be required to submit a final report informing the HEAG of its completion.

Please ensure that the Deakin logo is on the Plain Language Statement and Consent Forms. You should also ensure that the project ID is inserted in the complaints clause on the Plain Language Statement, and be reminded that the project number must always be quoted in any communication with the HEAG to avoid delays. All communication should be directed to sciethic@deakin.edu.au

The Faculty HEAG and/or Deakin University Human Research Ethics Committee (HREC) may need to audit this project as part of the requirements for monitoring set out in the National Statement on Ethical Conduct in Human Research (2007).

If you have any queries in the future, please do not hesitate to contact me.

We wish you well with your research.

Kind regards

A handwritten signature in blue ink that reads "Teresa Treffry".

Teresa Treffry
Secretary, Human Ethics Advisory Group (HEAG)
Faculty of Science Engineering & Built Environment



Rajesh Vasa, Mohamed Abdelrazeq, Andrew Cain, Scott Barnett, Alex Cummaudo
Applied Artificial Intelligence Institute (A²I²) (G)

23rd July 2019

Dear Rajesh and research team

STEC-39-2019-CUMMAUDO titled "*Factors that impact the learnability, interpretability and adoption of intelligent services*".

Thank you for submitting the above project for consideration by the Faculty Human Ethics Advisory Group (HEAG). The HEAG recognised that the project complies with the National Statement on Ethical Conduct in Human Research (2007) and has approved it. You may commence the project upon receipt of this communication.

The approval period is for three years until 23/07/22. It is your responsibility to contact the Faculty HEAG immediately should any of the following occur:

- Serious or unexpected adverse effects on the participants
- Any proposed changes in the protocol, including extensions of time
- Any changes to the research team or changes to contact details
- Any events which might affect the continuing ethical acceptability of the project
- The project is discontinued before the expected date of completion.

You will be required to submit an annual report giving details of the progress of your research. Please forward your first annual report on 23/07/20. Failure to do so may result in the termination of the project. Once the project is completed, you will be required to submit a final report informing the HEAG of its completion.

Please ensure that the project number must always be quoted in any communication with the HEAG to avoid delays. All communication should be directed to sciethic@deakin.edu.au.

The Faculty HEAG and/or Deakin University Human Research Ethics Committee (HREC) may need to audit this project as part of the requirements for monitoring set out in the National Statement on Ethical Conduct in Human Research (2007).

If you have any queries in the future, please do not hesitate to contact me.

We wish you well with your research.

Kind regards

Rickie Morey

Rickie Morey
Senior Research Administration Officer
Representing the Human Ethics Advisory Group (HEAG)
Faculty of Science Engineering & Built Environment

APPENDIX D

Primary Sources from Systematic Mapping Study

References

- [1] M. P. Robillard, "What makes APIs hard to learn? Answers from developers," *IEEE Software*, vol. 26, no. 6, pp. 27–34, 2009.
- [2] M. P. Robillard and R. Deline, "A field study of API learning obstacles," *Empirical Software Engineering*, vol. 16, no. 6, pp. 703–732, 2011.
- [3] A. J. Ko and Y. Riche, "The role of conceptual knowledge in API usability," in *2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2011, pp. 173–176.
- [4] J. Nykaza, R. Messinger, F. Boehme, C. L. Norman, M. Mace, and M. Gordon, "What programmers really want - results of a needs assessment for SDK documentation." *SIGDOC*, 2002.
- [5] R. Watson, M. Mark Stamnes, J. Jeannot-Schroeder, and J. H. Spyridakis, "API documentation and software community values," in *the 31st ACM international conference*. New York, New York, USA: ACM Press, 2013, pp. 165–10.
- [6] S. Y. Jeong, Y. Xie, J. Beaton, B. A. Myers, J. Stylos, R. Ehret, J. Karstens, A. Efeoglu, and D. K. Busse, "Improving documentation for eSOA APIs through user studies," in *International Symposium on End User Development*. Springer, 2009, pp. 86–105.
- [7] E. Aghajani, C. Nagy, O. L. Vega-Márquez, M. Linares-Vásquez, L. Moreno, G. Bavota, and M. Lanza, "Software Documentation Issues Unveiled," *ICSE*, vol. 2, no. 3, pp. 127–139, 2019.
- [8] S. Haselbock, R. Weinreich, G. Buchgeher, and T. Kriegbaum, "Microservice Design Space Analysis and Decision Documentation: A Case Study on API Management," *2018 IEEE 11th Conference on Service-Oriented Computing and Applications (SOCA)*, pp. 1–8, Nov. 2018.
- [9] S. Inzunza, R. Juárez-Ramírez, and S. Jiménez, "API Documentation," in *Trends and Advances in Information Systems and Technologies*. Cham: Springer, Cham, Mar. 2018, pp. 229–239.
- [10] M. Meng, S. Steinhardt, and A. Schubert, "Application Programming Interface Documentation: What Do Software Developers Want?" *Journal of Technical Writing and Communication*, vol. 48, no. 3, pp. 295–330, Aug. 2017.
- [11] R. S. Geiger, N. Varoquaux, C. Mazel-Cabasse, and C. Holdgraf, "The Types, Roles, and Practices of Documentation in Data Analytics Open Source Software Libraries," *Computer Supported Cooperative Work (CSCW)*, vol. 27, no. 3-6, pp. 767–802, May 2018.
- [12] A. Head, C. Sadowski, E. Murphy-Hill, and A. Knight, *When not to comment: questions and tradeoffs with API documentation for C++ projects*, ser. questions and tradeoffs with API documentation for C++ projects. New York, New York, USA: ACM, May 2018.

- [13] L. Aversano, D. Guardabascio, and M. Tortorella, “Analysis of the Documentation of ERP Software Projects,” *Procedia Computer Science*, vol. 121, pp. 423–430, Jan. 2017.
- [14] M. P. Robillard, A. Marcus, C. Treude, G. Bavota, O. Chaparro, N. Ernst, M. A. Gerosa, M. Godfrey, M. Lanza, M. Linares-Vásquez, G. C. Murphy, L. Moreno, D. Shepherd, and E. Wong, “On-demand Developer Documentation,” in *2017 IEEE International Conference on Software Maintenance and Evolution (IC-SME)*. IEEE, 2017, pp. 479–483.
- [15] R. B. Watson, “Development and application of a heuristic to assess trends in API documentation.” *SIGDOC*, 2012.
- [16] W. Maalej and M. P. Robillard, “Patterns of Knowledge in API Reference Documentation.” *IEEE Trans. Software Eng.*, 2013.
- [17] D. L. Parnas and S. A. Vilkomir, “Precise Documentation of Critical Software,” in *10th IEEE High Assurance Systems Engineering Symposium (HASE'07)*. IEEE, Sep. 2007, pp. 237–244.
- [18] C. Bottomley, “What part writer? What part programmer? A survey of practices and knowledge used in programmer writing.” *IPCC 2005. Proceedings. International Professional Communication Conference*, 2005., pp. 802–812, Jan. 2005.
- [19] A. Taulavuori, E. Niemelä, and P. Kallio, “Component documentation—a key issue in software product lines,” *Information and Software Technology*, vol. 46, no. 8, pp. 535–546, Jun. 2004.
- [20] J. Kotula, “Using Patterns To Create Component Documentation.” *IEEE Software*, 1998.
- [21] S. G. McLellan, A. W. Roesler, J. T. Tempest, and C. Spinuzzi, “Building More Usable APIs.” *IEEE Software*, 1998.