

***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 4, 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 4, 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

---

<b>Abstract</b>	<b>iii</b>
<b>Declaration</b>	<b>v</b>
<b>Acknowledgements</b>	<b>ix</b>
<b>Contents</b>	<b>ix</b>
<b>List of Publications</b>	<b>xv</b>
<b>List of Abbreviations</b>	<b>xvii</b>
<b>List of Figures</b>	<b>xxi</b>
<b>List of Tables</b>	<b>xxiv</b>
<b>List of Listings</b>	<b>xxv</b>
<b>I Preface</b>	<b>1</b>
<b>1 Introduction</b>	<b>3</b>
1.1 Research Context . . . . .	4
1.2 Research Motivation . . . . .	7
1.2.1 Motivating Scenarios . . . . .	9
1.3 Research Goals . . . . .	13
1.3.1 Research Overview . . . . .	13
1.3.2 Research Questions . . . . .	14
1.4 Research Methodology . . . . .	15
1.5 Thesis Organisation . . . . .	16
1.5.1 Overview of Chapters . . . . .	16

1.5.2	Coherency of Publications . . . . .	19
1.6	Research Contributions . . . . .	22
<b>2</b>	<b>Background</b>	<b>25</b>
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 . . . . .	32
2.2	Probabilistic and Nondeterministic Systems . . . . .	32
2.2.1	Interpreting the Uninterpretable . . . . .	32
2.2.2	Explanation and Communication . . . . .	35
2.2.3	Mechanics of Model Interpretation . . . . .	36
2.3	Application Programming Interfaces . . . . .	36
2.3.1	API Usability . . . . .	37
<b>3</b>	<b>Research Methodology</b>	<b>39</b>
3.1	Research Questions Revisited . . . . .	39
3.1.1	Knowledge Questions . . . . .	40
3.1.2	Design Questions . . . . .	40
3.2	Philosophical Stances . . . . .	41
3.3	Research Design . . . . .	42
3.3.1	Review of Relevant Research Methods . . . . .	42
3.3.2	Review of Data Collection Techniques for Field Studies . . . . .	44
3.4	Proposed Experiments . . . . .	44
3.4.1	Experiment I: Develop Initial Framework . . . . .	44
3.4.2	Developing the Initial Framework . . . . .	47
3.4.3	Experiment II: Validate Initial Framework . . . . .	47
3.5	Empirical Validity . . . . .	48
3.5.1	Threats to Internal Validity . . . . .	49
3.5.2	Threats to External Validity . . . . .	50
3.5.3	Threats to Construct Validity . . . . .	51
<b>II</b>	<b>Publications</b>	<b>53</b>
<b>4</b>	<b>Identifying Evolution in Computer Vision Services</b>	<b>55</b>
4.1	Introduction . . . . .	55
4.2	Motivating Example . . . . .	57
4.3	Related Work . . . . .	58
4.3.1	External Quality . . . . .	58
4.3.2	Internal Quality . . . . .	59
4.4	Method . . . . .	60
4.5	Findings . . . . .	63
4.5.1	Consistency of top labels . . . . .	63
4.5.2	Consistency of confidence . . . . .	65
4.5.3	Evolution risk . . . . .	67

4.6	Recommendations . . . . .	68
4.6.1	Recommendations for IWS users . . . . .	68
4.6.2	Recommendations for IWS providers . . . . .	69
4.7	Threats to Validity . . . . .	70
4.7.1	Internal Validity . . . . .	70
4.7.2	External Validity . . . . .	71
4.7.3	Construct Validity . . . . .	71
4.8	Conclusions & Future Work . . . . .	71
<b>5</b>	<b>Systematic Mapping Study of API Documentation Knowledge</b>	<b>73</b>
5.1	Introduction . . . . .	73
5.2	Related Work . . . . .	75
5.3	Method . . . . .	76
5.3.1	Systematic Mapping Study . . . . .	76
5.3.2	Development of the Taxonomy . . . . .	79
5.4	Taxonomy Validation . . . . .	81
5.4.1	Survey Study . . . . .	82
5.4.2	Empirical Application against Computer Vision Services . . . . .	82
5.5	Taxonomy . . . . .	82
5.6	Threats to Validity . . . . .	82
5.7	Conclusions & Future Work . . . . .	83
<b>6</b>	<b>Interpreting Pain-Points in Computer Vision Services</b>	<b>87</b>
6.1	Introduction . . . . .	87
6.2	Motivation . . . . .	89
6.3	Background . . . . .	91
6.4	Method . . . . .	92
6.4.1	Data Extraction . . . . .	92
6.4.2	Data Filtering . . . . .	94
6.4.3	Data Analysis . . . . .	97
6.5	Findings . . . . .	97
6.5.1	Post classification and reliability analysis . . . . .	98
6.5.2	Developer Frustrations . . . . .	98
6.5.3	Statistical Distribution Analysis . . . . .	100
6.6	Discussion . . . . .	101
6.6.1	Answers to Research Questions . . . . .	101
6.6.2	The Developer's Learning Approach . . . . .	102
6.6.3	Implications . . . . .	105
6.7	Threats to Validity . . . . .	107
6.8	Conclusions . . . . .	108
<b>7</b>	<b>Ranking Computer Vision Service Issues using Emotion</b>	<b>109</b>

<b>8 Using a Facade Pattern to combine Computer Vision Services</b>	<b>111</b>
8.1 Introduction . . . . .	111
8.1.1 Motivating Scenario: Intelligent vs Traditional Web Services	112
8.1.2 Research Motivation . . . . .	113
8.2 Merging API Responses . . . . .	113
8.2.1 API Facade Pattern . . . . .	114
8.2.2 Merge Operations . . . . .	114
8.2.3 Merging Operators for Labels . . . . .	115
8.3 Graph of Labels . . . . .	116
8.3.1 Labels and synsets . . . . .	116
8.3.2 Connected Components . . . . .	116
8.4 API Results Merging Algorithm . . . . .	119
8.4.1 Mapping Labels to Synsets . . . . .	119
8.4.2 Deciding Total Number of Labels . . . . .	119
8.4.3 Allocating Number of Labels to Connected Components . .	120
8.4.4 Selecting Labels from Connected Components . . . . .	121
8.4.5 Conformance to properties . . . . .	121
8.5 Evaluation . . . . .	121
8.5.1 Evaluation Method . . . . .	121
8.5.2 Naive Operators . . . . .	122
8.5.3 Traditional Proportional Representation Operators . . .	124
8.5.4 New Proposed Label Merge Technique . . . . .	124
8.5.5 Performance . . . . .	124
8.6 Conclusions and Future Work . . . . .	125
<b>9 Supporting Safe Usage of Intelligent Web Services</b>	<b>127</b>
9.1 Introduction . . . . .	127
9.2 Motivating Example . . . . .	129
9.3 Threshy . . . . .	131
9.4 Related work . . . . .	132
9.5 Conclusions & Future Work . . . . .	133
<b>10 FSE Paper</b>	<b>135</b>
<b>III Postface</b>	<b>137</b>
<b>11 Conclusions &amp; Future Work</b>	<b>139</b>
<b>References</b>	<b>159</b>
<b>List of Online Artefacts</b>	<b>161</b>

<b>IV Appendices</b>	<b>165</b>
<b>A Additional Materials</b>	<b>167</b>
A.1 The Development, Documentation and Usage of Web APIs . . . . .	169
<b>B Authorship Statements</b>	<b>175</b>
<b>C Ethics Clearance</b>	<b>189</b>
<b>D Primary Sources from Systematic Mapping Study</b>	<b>193</b>



---

## 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

---

**A<sup>2</sup>I<sup>2</sup>** Applied Artificial Intelligence Institute. 44, 47, 50

**AI** artificial intelligence. iii, 3, 4, 8, 9, 32, 34, 35, 55, 56, 59, 60, 69, 70, 72, 74, 113, 114, 172

**API** application programming interface. xxiv, 4–14, 16–18, 21, 22, 25, 26, 28, 29, 36–41, 43–48, 51, 56, 57, 60, 61, 69, 70, 72–80, 82–86, 111–114, 116, 118, 119, 121, 122, 125, 169, 171

**AWS** Amazon web services. 61, 64

**BYOML** Build Your Own Machine Learning. 5, 6

**CC** connected component. 116, 119–121, 125

**CDSS** clinical decision support system. 9, 12

**CNN** convolutional neural network. 12, 32, 58

**CRUD** Create, read, update, and delete. 171

**CV** computer vision. 5, 7, 22, 33, 37, 55, 57, 58, 70–72

**CVS** computer vision service. 7–11, 13–20, 22, 23, 25, 27, 28, 31, 37, 44, 45, 48, 55–61, 64, 69–71, 74, 75, 111, 113, 114, 116, 119, 121, 125, 173

**DCE** distributed computing environment. 169

**HITL** human-in-the-loop. 12, 13

**HTTP** Hypertext Transfer Protocol. 6, 169–171

**IDL** interface definition language. 169, 171

**IWS** intelligent web service. 6–10, 12–14, 16, 18, 25–29, 31, 32, 36, 38–41, 43–48, 51, 55–57, 59, 60, 68, 70–72, 74, 75, 81, 111–113, 125

**JSON** JavaScript Object Notation. 7

**KA** knowledge area. 76, 80

**ML** machine learning. iii, 3, 4, 6, 8, 9, 11, 18, 20, 29, 34, 37, 55, 56, 59, 60, 69, 72, 111, 112, 125

**NN** neural network. 8, 32–34, 36

**PaaS** Platform as a Service. 7, 12, 59

**QoS** quality of service. 59, 60, 169

**RAML** RESTful API Modeling Language. 171

**REST** REpresentational State Transfer. 7, 56, 112, 170, 171

**ROI** region of interest. 12

**RPC** remote procedure call. 169

**SDK** software development kit. 76

**SE** software engineering. 14–16, 18, 35, 56, 59, 75–80

**SLA** service-level agreement. 59, 169

**SMS** systematic mapping study. 17, 19, 21, 74–76, 83

**SO** Stack Overflow. 5, 14, 15, 17, 18, 21, 22, 43, 46, 60, 61, 173

**SOA** service-oriented architecture. 169

**SOAP** Simple Object Access Protocol. 7, 169–171

**SQA** service quality assurance. 57, 58

**SQuaRE** Systems and software Quality Requirements and Evaluation. 31

**SUS** System Usability Scale. 17, 75

**SVM** support vector machine. 32, 36

**SWEBOK** Software Engineering Body of Knowledge. 76, 77, 80

**URI** uniform resource identifier. 171

**V&V** verification & validation. 25–29

**WADL** Web Application Description Language. 171

**WS** web service. 6, 28, 60, 112, 113, 169, 171

**WSDL** Web Services Description Language. 169

**XML** eXtendable markup language. 7, 169



---

## List of Figures

---

1.1	Differences between data- and rule-driven cloud services . . . . .	5
1.2	The spectrum of machine learning . . . . .	5
1.3	Overview of intelligent web services . . . . .	7
1.4	CancerAssist Context Diagram . . . . .	13
1.5	Overview publication coherency . . . . .	21
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 . . . . .	33
2.5	Deterministic versus nondeterministic systems . . . . .	34
2.6	Theory of AI communication . . . . .	36
3.1	High-level overview of the proposed experiments . . . . .	45
4.1	Consistency of labels in CV services is rare . . . . .	63
4.2	Top labels for images between CV services do not intersect . . . .	64
4.3	CV services can return multiple top labels . . . . .	65
4.4	Cumulative distribution of top label confidences . . . . .	66
4.5	Cumulative distribution of intersecting top label confidences . .	66
4.6	Agreement of labels between multiple CV services do not share similar confidences . . . . .	67
5.1	SMS search results, by years . . . . .	78
5.2	A systematic map of API documentation knowledge studies . . . .	81
6.1	Traits of intelligent services compared to DIY ML . . . . .	90
6.2	Trend of Stack Overflow posts discussing computer vision services .	91
6.3	Comparing documentation-specific and generalised classifications of SO posts . . . . .	98

6.4 Alignment of Bloom and SOLO taxonomies against computer vision issues . . . . .	104
8.1 Overview of the proposed facade . . . . .	114
8.2 Graph of associated synsets against two different endpoints . . . . .	117
8.3 Label counts per API assessed . . . . .	118
8.4 Connected components vs. images . . . . .	118
8.5 Allocation to connected components . . . . .	120
8.6 F-measure comparison . . . . .	124
9.1 Example case study of evaluating model performance in two different models . . . . .	128
9.2 Example pipeline of a computer vision system . . . . .	130
9.3 UI workflow of Threshy . . . . .	131
9.4 Architecture of Threshy . . . . .	133
A.1 SOAP versus REST search interest over time . . . . .	170
A.2 Categorisation of AI-based products and services . . . . .	172
A.3 Increasing interest in the developer community of computer vision services . . . . .	173
A.4 Review of field study techniques . . . . .	174

---

## 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 . . . . .	10
1.4	Definitions of ‘confidence’ in CV documentation . . . . .	11
1.5	List of publications resulting from this thesis . . . . .	20
4.1	Characteristics of data in CV evolution assessment . . . . .	62
4.2	Ratio of consistent labels in CV services . . . . .	63
4.3	Evolution of top labels and confidence values . . . . .	67
5.1	Summary of search results in API documentation knowledge . . . . .	77
5.2	Data extraction in API documentation knowledge study . . . . .	80
5.3	Taxonomy proposed in API documentation knowledge study . . . . .	84
6.1	Taxonomies used in our Stack Overflow mining study . . . . .	96
6.2	Example posts aligning to Bloom’s and the SOLO taxonomy . . . . .	103
8.1	Statistics for the number of labels . . . . .	116
8.2	First allocation iteration . . . . .	121
8.3	Second allocation iteration . . . . .	121
8.4	Third allocation iteration . . . . .	121
8.5	Fourth allocation iteration . . . . .	121
8.6	Matching to human-verified labels . . . . .	123
8.7	Evaluation results of the facade . . . . .	123
8.8	Average of evaluation result of the facade . . . . .	123



---

## List of Listings

---

A.1	An example SOAP request . . . . .	169
A.2	An example SOAP response . . . . .	170
A.3	An example RESTful request . . . . .	171
A.4	An example RESTful response . . . . .	171



**Part I**

**Preface**



# CHAPTER 1

3

4

---

5

## Introduction

6

---

7   ⟨ *todo: Introduce a broad statement of the problem. AI-first software is data-driven, but documentation and nature of services that provide these services make it look like they are deterministic* ⟩

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

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

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

---

<sup>1</sup>A 2016 report by market research company Forrester captured such growth into four key areas [178], as reproduced in Figure A.2.

<sup>32</sup> components of the ML framework together<sup>2</sup> into their application’s code.

<sup>33</sup> 3. The application developer uses a data-driven and cloud-based service. They  
<sup>34</sup> don’t need to know anything behind the underlying ‘intelligence’ and how it  
<sup>35</sup> functions. It is fast to integrate into their applications, and the application pro-  
<sup>36</sup> gramming interfaces (APIs) offered abstracts the technical know-how behind  
<sup>37</sup> a web call.

<sup>38</sup> The documentation of the service alludes that the data-driven service is as similar to  
<sup>39</sup> other cloud services offered by the provider. Because this is ‘another’ cloud service,  
<sup>40</sup> the application developer *assumes* it would act and behave as any other typical  
<sup>41</sup> service would. But does this assumption—and a lack of appreciation of ML—lead  
<sup>42</sup> to developer pain-points and miscomprehension? If so, how can the service providers  
<sup>43</sup> improve their documentation to alleviate this? Do these data-driven services share  
<sup>44</sup> similarities to the runtime behaviour of traditional cloud services? And if not,  
<sup>45</sup> how best can the application developer integrate the data-driven service into their a  
<sup>46</sup> rule-driven application to produce AI-first software?

<sup>47</sup> Figure 1.1 provides an illustrative overview between the context clashing of rule-  
<sup>48</sup> driven applications and data-driven cloud services, and we contrast characteristics  
<sup>49</sup> 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.

## <sup>50</sup> 1.1 Research Context

<sup>51</sup> ⟨ *TODO: Unpack the ‘spectrum’ a little, and make it clear that this thesis focuses on*  
<sup>52</sup> *computer vision services.* ⟩

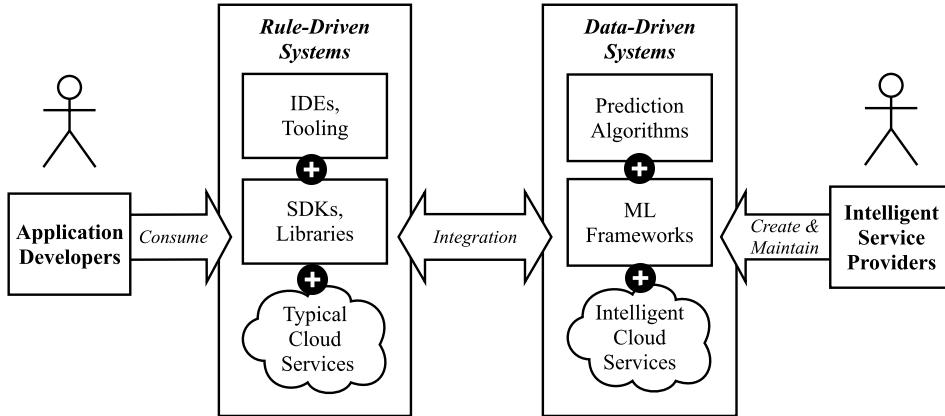
<sup>53</sup> As described, the application developer has three key approaches in producing  
<sup>54</sup> AI-first software. This ‘range’ of AI-first integration techniques partially reflects  
<sup>55</sup> Google AI’s<sup>3</sup> *machine learning Spectrum* [164, 189, 216], which encompasses the  
<sup>56</sup> variety of skill, effort, users and types of outputs of integration techniques. One ex-  
<sup>57</sup> treme involves the academic research of developing algorithms and self-sourcing data

---

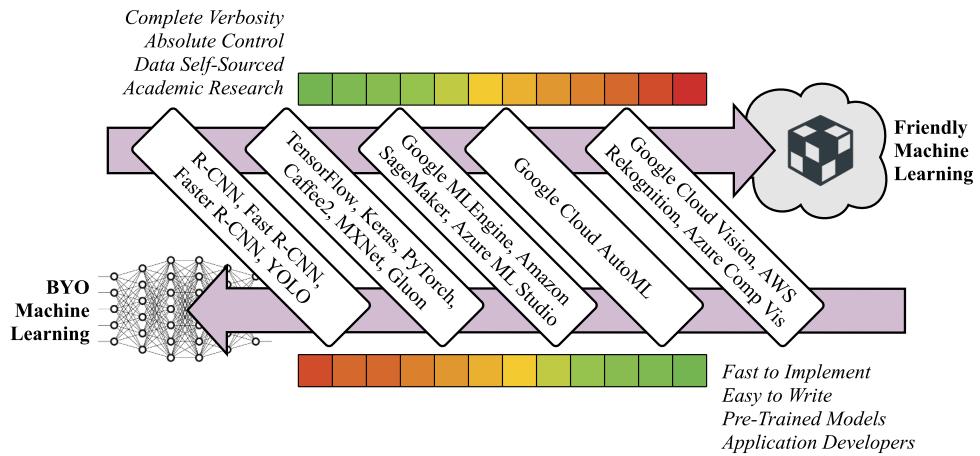
<sup>2</sup>Thus introducing a verbose list of ML terminology to her developer vocabulary. See a list of 328 terms provided by Google here: <https://developers.google.com/machine-learning/glossary/>. Last accessed 7 December 2018.

<sup>3</sup>Google 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* [130] is evident. A further example includes how Facebook leverage AI *at scale* within their infrastructure and platforms [219].

**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.



**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.



<sup>58</sup> to achieve intelligence—coined as Build Your Own Machine Learning (BYOML)  
<sup>59</sup> [140, 189, 216]. The other extreme involves off-the-shelf, ‘friendlier’ (abstracted)  
<sup>60</sup> intelligence with easy-to-use APIs targeted towards applications developers. The  
<sup>61</sup> middle-ground involves a mix of the two, with varying levels of automation to assist  
<sup>62</sup> in development, that turns custom datasets into predictive intelligence. We illustrate  
<sup>63</sup> the slightly varied characteristics within this spectrum in Table 1.2 and Figure 1.2.

<sup>64</sup> These data-driven ‘friendly’ services are gaining traction within developer circles:  
<sup>65</sup> we show an increasing trend of Stack Overflow posts mentioning a mix of  
<sup>66</sup> intelligent computer vision (CV) services in Figure A.3.<sup>4</sup> Academia provides var-  
<sup>67</sup> ied nomenclature for these services, such as *Cognitive Applications* and *Machine*  
<sup>68</sup> *Learning Services* [295] or *Machine Learning as a Service* [238]. For the context

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

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

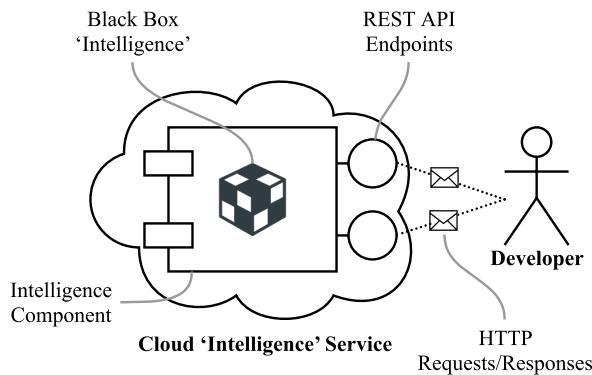
intelligent web service	Typical web service
Probabilistic	Deterministic
Machine Learnt	Human Engineered
Data-Driven	Rule-Driven
Black-Box	Mostly Transparent

**Table 1.2:** Comparison of the machine learning spectrum.

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

<sup>69</sup> of this thesis, we will refer to such services under broader term of **intelligent web services (IWSs)**, and diagrammatically express their usage within Figure 1.3.

**Figure 1.3:** Overview of IWSs.



<sup>71</sup> While there are many types of IWSs available to software developers,<sup>5</sup> the  
<sup>72</sup> general workflow of using an IWS is more-or-less the same: a developer accesses  
<sup>73</sup> an IWS component via REST/SOAP API(s), which is (typically) available as a  
<sup>74</sup> cloud-based Platform as a Service (PaaS).<sup>6,7</sup> For a given input, developers receive  
<sup>75</sup> an ‘intelligent’ response and an associated confidence value that represents the  
<sup>76</sup> likelihood of that result. This is typically serialised as a JSON/XML response  
<sup>77</sup> object.

☞ *Within this thesis, we scope our investigation to a mature subset of IWSs that provide computer vision intelligence [315, 318, 321, 322, 323, 329, 331, 334, 335, 337, 339, 376, 377]. For the context of this thesis, we will refer to such services as **computer vision services (CVSs)**.*

## 1.2 Research Motivation

<sup>79</sup> *< todo: This section describes deterministic mindset clashing against probabilistic behaviour >*

<sup>81</sup> *< todo: MA: It is useful to highlight the takeaways of each paragraph - e.g. probabilistic nature of ML, Choosing a threshold problem, lack of understanding*

<sup>5</sup>Such as optical character recognition, text-to-speech and speech-to-text transcription, object categorisation, facial analysis and recognition, natural language processing etc.

<sup>6</sup>We 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.

<sup>7</sup>A 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 of how ml works, documentation style of ml models. then at the end of the section,  
84 clearly state the problem/gap considered >*

*85 IWSs are accessible through APIs consisting of ‘black box’ intelligence (Figure  
86 1.3).<sup>8</sup> Data scientists produce ML algorithms to make predictions in our datasets  
87 and discover patterns within them. These black boxes are inherently probabilistic  
88 and stochastic; there is little room for certainty in these results, as the insight is  
89 purely statistical and associational [224] against its training dataset.*

*90 Evermore applications are using CVSs as demonstrated by ubiquitous examples:  
91 aiding the vision-impaired [74, 236], accounting [184], data analytics [136], and  
92 student education [80]. However, developers who build these applications do not  
93 need to treat their programs in a stochastic or probabilistic mindset manner, given  
94 a rule-driven mindset that computers make certain outcomes. A CVS, for example,  
95 may return the *probability* that a particular object exists in an input images’ pixels, and  
96 thus for a more certain (though not fully certain) distribution of overall confidence  
97 returned from the service, a developer must treat the problem stochastically by  
98 testing this case hundreds if not thousands of times to find a richer interpretation of  
99 the inference made.*

*100 Unless a change in API version occurs, educators teach application developers  
101 a deterministic mindset that an API call returns the same outputs for the same  
102 inputs. As an example, consider simple arithmetic representations (e.g.,  $2 + 2 =$   
103 4). The deterministic (rule-driven) mindset suggests that the result will *always* be  
104 4. However, the non-deterministic (data-driven) mindset suggests that results are  
105 probable: target output (*exactly* 4) and the output inferred (*a likelihood of* 4) matches  
106 as a probable percentage (or as an error where it does not match).<sup>9</sup> Instead of an  
107 exact output, there is a *probabilistic* result:  $2 + 2$  *may* equal 4 to a confidence of  $n$ .*

*108 Developers must appreciate such a critical principle of ML software when building  
109 quality AI-first applications, particularly reliability. The external quality of such  
110 software needs to consider reliability in the case of thresholding confidence values,  
111 that is whether the inference has an appropriate level of confidence to justify a  
112 predicted (and reliable) result to end-users. Selecting this confidence threshold is  
113 non-trivial; a ML course from Google suggests that “it is tempting to assume that [a]  
114 classification threshold should always be 0.5, but thresholds are problem-dependent,  
115 and are therefore values that you must tune.” [?CITE? ]. Approaches to turning  
116 these values are considered for data scientists, but are not yet well-understood for  
117 application developers with little appreciation of the nuances of ML. Similarly, de-  
118 velopers should consider the internal quality of building AI-first software. Reliable  
119 API usability and documentation advocate for the accuracy, consistency and com-  
120 pleteness of APIs and their documentation [229, 244] and providers should consider  
121 mismatches between a developer’s conceptual knowledge of the API its implemen-  
122 tation [157]. Unreliable APIs ultimately hinder developer performance and thus*

---

<sup>8</sup>The ‘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].

<sup>9</sup>Blake et al. [33] produces a multi-layer perceptron neural network performing arithmetic representation.

123 reduces productivity. Section 1.2.1 provides two motivating scenarios describing  
124 these concerns in greater detail.

125 **1.2.1 Motivating Scenarios**

126 *< todo: Could move this into the appendix to keep introduction chapter tighter >*  
127    *< todo: JG: or could use one of these at start of section 1.2 to help clarify/moti-*  
128    *vate... >*  
129    *< todo: yes too long in this chapter - use one, simplify a bit. Could put other one*  
130    *in appendix... >*  
131    *< todo: MA: i guess the discussion related to Table1.3 and Table1.4 could be*  
132    *moved to 1.2 as one of the challenges > < todo: MA: after reading the section, it looks*  
133    *like you could move it before 1.2 >*

134 The market for IWSs is increasing (Figure A.2) and as is developer uptake and  
135 enthusiasm in the software engineering community (Figure A.3). However, the  
136 impact to software quality (internal and external) due to a mismatch of the applica-  
137 tion developer's deterministic mindset and the service provider's nondeterministic  
138 mindset is of concern.

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

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

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

156 Instead of building a computer vision engine from scratch, which takes too much  
157 time and effort, Tom thinks he can achieve this using one of the common CVSs. Tom  
158 comes from a typical software engineering background and has insufficient knowl-  
159 edge of key computer vision terminology and no understanding of its underlying  
160 techniques. However, inspired by easily accessible cloud APIs that offer computer  
161 vision analysis, he chooses to use these. Built upon his experience of using other  
162 similar cloud services, he decides on one of the CVS APIs, and expects a static result  
163 always and consistency between similar APIs. Analogously, when Tom invokes the  
164 iOS Swift substring method "doggy".prefix(3), he expects it to be consistent

165 with the Android Java equivalent "doggy".`substring(0, 2)`. Consistent, here,  
166 means two things: (i) that calling `substring` or `prefix` on 'dog' will *always*  
167 return in the same way every time he invokes the method; and (ii) that the result is  
168 *always* 'dog' regardless of the programming language or string library used, given  
169 the deterministic nature of the '`substring`' construct (i.e., results for `substring` are  
170 API-agnostic).

**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	<b>0.986</b>	0.999	0.999	0.992	<b>0.970</b>
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	<b>0.680</b>	-	-	-	-
Black	-	-	0.992	0.992	-	-
Indoor	-	-	0.965	0.965	-	-
Standing	-	-	0.792	0.792	-	-
Mammal	-	-	0.929	0.929	0.992	<b>0.970</b>
Animal	-	-	0.932	0.932	0.992	<b>0.970</b>
Canine	-	-	-	-	0.992	<b>0.970</b>
Collie	-	-	-	-	0.992	<b>0.970</b>
Pet	-	-	-	-	0.992	<b>0.970</b>

171 More concretely, in Table 1.3, we illustrate how three (anonymised) CVS  
172 providers fail to provide similar consistency to that of the substring example above.  
173 If Tom uploads a photo of a border collie<sup>10</sup> to three different providers in August  
174 2018 and January 2019, he would find that each provider is different in both the vo-  
175 cabulary used between. The confidence values and labels within the *same* provider  
176 varies within a matter of five months. The evolution of the confidence changes is  
177 not explicitly documented by the providers (i.e., when the models change) nor do  
178 they document what confidence means. Service providers use a tautological nature  
179 when defining what the confidence confidence values are (as presented in the API  
180 documentation) provides no insight for Tom to understand why there was a change  
181 in confidence, which we show in Table 1.4, unless he *knows* that the underlying  
182 models change with them. Furthermore, they do not provide detailed understanding  
183 on how to select a threshold cut-off for a confidence value. Therefore, he's left with  
184 no understanding on how best to tune for image classification in this instance. The  
185 deterministic problem of a substring compared to the nondeterministic nature of the  
186 IWS is, therefore, non-trivial.

187 To make an assessment of these APIs, he tries his best to read through the  
188 documentation of different CVS APIs, but he has no guiding framework to help him

<sup>10</sup>The image used for these results is <https://www.akc.org/dog-breeds/border-collie/>.

**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).” [330]
	“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].” [?CITE? ]
	“The overall score of the result. Range [0, 1]” [?CITE? ]
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.” [340]
	“The level of confidence the service has in the caption.” [338]
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.” [316]
	“[Provider C] also provide[s] a percentage score for how much confidence [Provider C] has in the accuracy of each detected label.” [317]

189 choose the right one. A number of questions come to mind:

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

200        Although Tom generally anticipates these CVSs to not be perfect, he has no prior benchmark to guide him on what to expect. The imperfections appear to be low-risk, but may become socially awkward when in use; for instance, if Tom’s friends have low self-esteem and use the app, they may be sensitive to the app not

206 identifying them or mislabelling them. Privacy issues come into play especially  
207 if certain friends have access to certain photos that they are (supposedly) in; e.g.,  
208 photos from a holiday with Tom and his partner, however if the API identifies Tom’s  
209 partner as a work colleague, Tom’s partner’s privacy is at risk.

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

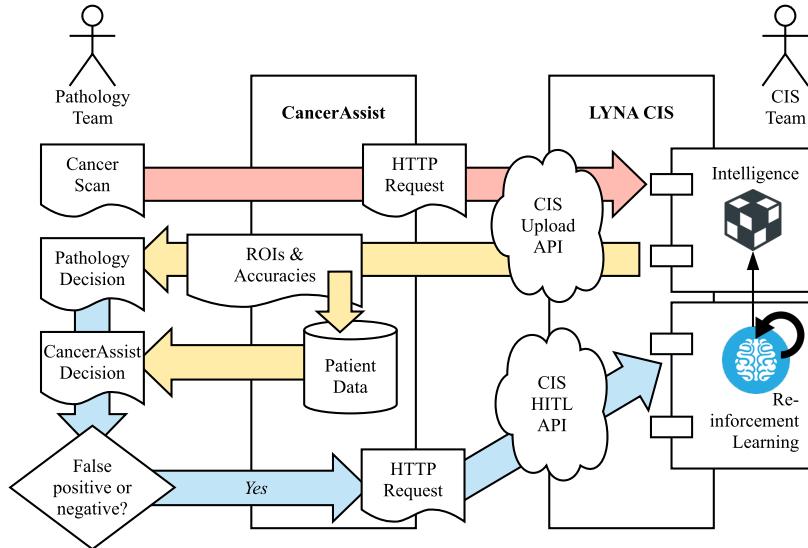
213 **Motivating Scenario II: Cancer Detection CDSS**

214 Recent studies in the oncology domain have used deep-learning convolutional neural  
215 networks (CNNs) to detect region of interests (ROIs) in image scans of tissue (e.g.,  
216 [24, 117, 177]), flagging these regions for doctors to review. Trials of such algorithms  
217 have been able to accurately detect cancer at higher rates than humans, and thus  
218 incorporating such capabilities into a CDSS is closer within reach. Studies have  
219 suggested these systems may erode a practitioner’s independent decision-making  
220 [56, 138] due to over-reliance; therefore the risks in developing CDSSs powered by  
221 IWSs become paramount.

222 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  
223 node scans and discuss and recommend, on consensus, if the patient requires an  
224 operation. When the team makes a consensus, the lead pathologist enters the ver-  
225 dict into CancerAssist—running passively in the background—to ensure there is  
226 no oversight in the team’s discussions. When a conflict exists between the team’s  
227 verdict and CancerAssist’s verdict, the system produces the scan with ROIs it thinks  
228 the team should review. Where the team overrides the output of CancerAssist, this  
229 reinforces CancerAssist’s internal model as a human-in-the-loop (HITL) learning  
230 process.

232 Powering CancerAssist is Google AI’s Lymph Node Assistant (LYNA) [177],  
233 a CNN based on the Inception-v3 model [161, 279]. To provide intelligence to  
234 CancerAssist, the development team decide to host LYNA as an IWS using a cloud-  
235 based PaaS solution. Thus, CancerAssist provides API endpoints integrated with  
236 patient data and medical history, which produces the verdict. In the case of a positive  
237 verdict, CancerAssist highlights the relevant ROIs found are with their respective  
238 bounding boxes and their respective cancer detection accuracies.

239 The developer of CancerAssist has no interaction with the Data Science team  
240 maintaining the LYNA IWS. As a result, they are unaware when updates to the  
241 model occur, nor do they know what training data they provide to test their system.  
242 The default assumptions are that the training data used to power the intelligence  
243 is near-perfect for universal situations; i.e., the algorithm chosen is the correct one  
244 for every assessable ontology tests in the given use case of CancerAssist. Thus,  
245 unlike deterministic systems—where the developer can manually test and validate  
246 the outcomes of the APIs—this is impossible for non-deterministic systems such  
247 as CancerAssist and its underlying IWS. The ramifications of not being able to test  
248 such a system and putting it out into production may prove fatal to patients.



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

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

- 251     • When is the model updated and how do the IWS team communicate these  
252         updates?
- 253     • What benchmark test set of data ensures that the changed model doesn't affect  
254         other results?
- 255     • Are assumptions made by the IWS team who train the model correct?

256     Thus, to improve communication between developers and IWS providers, devel-  
257         opers require enhanced documentation, additional metadata, and guidance tooling.

### 258     1.3 Research Goals

#### 259     1.3.1 Research Overview

260     ⟨ *TODO: is it JUST CVS, or they one exemplar of intelligent APIs/services??* ⟩ This  
261     thesis aims to investigate and better understand the nature of cloud-based computer  
262     vision services (CVSs).<sup>11</sup> We identify the maturity, viability and risks of CVSs  
263     through the anchoring perspective of *reliability* that affects the internal and external  
264     quality of software. We adopt the McCall [187] and Boehm [35] interpretations  
265     of reliability via the sub-characteristics of a service's *consistency* and *robustness*  
266     (or fault/error tolerance), and the *completeness*<sup>12</sup> of its documentation. (A detailed  
267     discussion is further provided in Section 2.1.) This thesis explores and contributes

<sup>11</sup>As 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.

<sup>12</sup>We treat the API documentation of a CVS as a first-class citizen.

<sup>268</sup> towards *four* key facets regarding reliability in CVS usage and the completeness  
<sup>269</sup> of its associated documentation. We formulate four primary research questions  
<sup>270</sup> (RQs) with seven sub-RQs, based on both empirical and non-empirical software  
<sup>271</sup> engineering methodology [195], further discussed in Chapter 3.

### <sup>272</sup> 1.3.2 Research Questions

<sup>273</sup> Firstly, we investigate adverse implications that arise when using CVSs that af-  
<sup>274</sup> fects consistency and robustness (**Chapter 4**). We show how CVSs have a non-  
<sup>275</sup> deterministic runtime behaviour and evolve with unintended and non-trivial con-  
<sup>276</sup> sequences to developers. We demonstrate that these services have inconsistent  
<sup>277</sup> behaviour despite offering the same functionality and pose evolution risk that ef-  
<sup>278</sup> fects robustness of consuming applications when responses change given the same  
<sup>279</sup> (consistent) inputs. Thus, we conclude how the nature of these services (at present)  
<sup>280</sup> are not fully robust, consistent, and thus not reliable. Formally, we structure the  
<sup>281</sup> following RQs:

#### ⌚ RQ1. What is the nature of cloud-based CVS?

*RQ1.1.* What is their runtime behaviour?

*RQ1.2.* What is their evolution profile?

<sup>282</sup> Secondly, we investigate the reliability of the documentation these services of-  
<sup>283</sup> fer through the lenses of its completeness. We collate prior knowledge of good  
<sup>284</sup> API documentation and assess the efficacy of such knowledge against practition-  
<sup>285</sup> ers (**Chapter 5**). We show that these service's behaviour and evolution is not  
<sup>286</sup> reliably documented adequately against this knowledge. Formally, we develop the  
<sup>287</sup> following RQs:

#### ⌚ RQ2. How complete is current CVS APIs documentation?

*RQ2.1.* What are the dimensions of '*complete*' API documentation,  
 according to both literature and practitioners?

*RQ2.2.* What additional information or attributes do developers need  
 in CVS API documentation to make it more complete?

<sup>288</sup> Thirdly, we investigate how software developers approach using these services  
<sup>289</sup> and directly assess developer pain-points resulting from the nature of CVSs and  
<sup>290</sup> their documentation (**Chapter 6**). We show that there is a statistically significant  
<sup>291</sup> difference in these complaints when contrasted against more established software  
<sup>292</sup> engineering domains (such as web or mobile development) as expressed as ques-  
<sup>293</sup> tions asked on Stack Overflow. We provide a number of exploratory avenues for  
<sup>294</sup> researchers, educators, software engineers and IWS providers to alleviate these com-  
<sup>295</sup> plaints based on this analysis. Further, using a data set consisting of 1,245 Stack  
<sup>296</sup> Overflow questions, we explore the emotional state of developers to understand

<sup>297</sup> which aspects (i.e., pain-points) developers are most frustrated with (**Chapter 7**).  
<sup>298</sup> We formulate the following RQs:

**?** **RQ3. How does the developer's comprehension of CVSs differ to conventional software engineering domains?**

- RQ3.1.* What aspects of CVSs and its documentation do developers struggle with, as expressed as pain-points within Stack Overflow questions?
- RQ3.2.* Is the distribution of these pain-points different to established computer vision service fields?
- RQ3.3.* Which of these pain-points express the most frustration within developer communities?

<sup>299</sup> Lastly, we explore several strategies to help improve CVSs reliability. Firstly,  
<sup>300</sup> we investigate whether merging the responses of *multiple* CVSs can improve their  
<sup>301</sup> reliability and propose a novel algorithm—based on the proportional representation  
<sup>302</sup> method used in electoral systems—to merge labels and associated confidence values  
<sup>303</sup> from three providers (**Chapter 8**). Secondly, we develop an integration architecture  
<sup>304</sup> style (or facade) to guard against CVS evolution, and synthesise an integration  
<sup>305</sup> workflow that addresses the concerns raised by developers in addition to embedding  
<sup>306</sup> ‘complete’ documentation artefacts into the workflow’s design (**Chapters 9 and 10**).  
<sup>307</sup> Our final RQ is:

**?** **RQ4. What strategies can developers use to integrate with and use CVSs, while preserving robustness and reliability?**

<sup>308</sup> **1.4 Research Methodology**

<sup>309</sup> This thesis employs a mixed-methods approach using the concurrent triangulation  
<sup>310</sup> strategy [43, 186]. The research presented consists of both empirical and non-  
<sup>311</sup> empirical research design. This section provides a high-level overview of the re-  
<sup>312</sup> search methodology within this thesis. Further details are provided in Section 1.5.2  
<sup>313</sup> and Chapter 3.

<sup>314</sup> Firstly, RQ1–RQ3 are all empirical, knowledge-based questions [89, 192] that  
<sup>315</sup> aim to provide the software engineering community with a greater understanding  
<sup>316</sup> of the phenomena surrounding CVSs from three perspectives: the nature of the ser-  
<sup>317</sup> vices themselves, how developers perceive these services and how service providers  
<sup>318</sup> can improve these services. We answer RQ1 using a longitudinal experiment that  
<sup>319</sup> assesses both the services’ responses and associated documentation (complement-  
<sup>320</sup> ing RQ2.2). We adopt qualitative and quantitative data collection; specifically (i)  
<sup>321</sup> structured observations to quantitatively analyse the results over time, and (ii) docu-  
<sup>322</sup> mentary research methods to inspect service documentation. Secondly, we perform  
<sup>323</sup> systematic mapping study following the guidelines of Kitchenham and Charters

[153] and Petersen et al. [226] to better understand how API documentation of these services can be improved (i.e., more complete), which targets Item RQ2. Based on the findings from this study, we use a systematic taxonomy development methodology specifically targeted toward software engineering [289] that structures scattered API documentation knowledge into a taxonomy. We then validate this taxonomy against practitioners using survey research, adopting Brooke well-established Systematic Usability Score [47] surveying instrument and contextualising it within API documentation utility, which answers RQ3.2. To answer RQ2.2, we perform an empirical application of the taxonomy to three CVSs, and therefore assess where improvements can be made. Thirdly, we adopt field survey research using repository mining of developer discussion forums (i.e., Stack Overflow) to answer RQ3, and classify these using both manual and automated techniques.

The second aspect of our research design involves non-empirical research, which explores a design-based question [195] to answer RQ4. As the answers to our first three RQs establish a greater understanding of the nature behind CVSs from various perspectives, the strategies we design in RQ4 aims at designing more reliable integration methods so that developers can better use these cloud-based services in their applications.

## 1.5 Thesis Organisation

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

### 1.5.1 Overview of Chapters

#### Part I: Preface

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

**Chapter 3: Research Methodology** This chapter provides a summative review of research methods and philosophical stances relevant to software engineering. We

362 illustrate that the methods used within our publications are sound via an analysis of  
363 the methodologies used in seminal works referenced in this thesis.

364 **Part II: Publications**

365 **Chapter 4: Exploring the nature of CVSs** This chapter was presented at the 2019  
366 International Conference on Software Maintenance and Evolution (ICSME) [69].  
367 We describe an 11-month longitudinal experiment assessing the behavioural (run-  
368 time) issues of three popular CVSs: Google Cloud Vision [331], Amazon Rekogni-  
369 tion [318] and Azure Computer Vision [339]. By using three different data sets—two  
370 of which we curate as additional contributions—we demonstrate how the services  
371 are inconsistent amongst each other and within themselves. This study provides a  
372 detailed answer to RQ1: Despite presenting conceptually-similar functionality, each  
373 service behaves and produces slightly varied (inconsistent) results and demonstrates  
374 non-deterministic runtime behaviour. We discuss potential evolution risks to con-  
375 sumers of such services as the services provide non-static outputs for the same inputs,  
376 thereby having significant impact to the robustness of consuming applications. Fur-  
377 ther details in the study include a brief assessment into the lack of sufficient detail  
378 of these concerns in their documentation.

379 **Chapter 5: Investigating improvements to CVS API documentation** This chap-  
380 ter was originally a short paper presented at the 2019 International Symposium on  
381 Empirical Software Engineering and Measurement (ESEM) [71]. To understand  
382 where to improve CVS documentation, we first need to investigate *what* makes a  
383 good API document. This short paper initially answered one aspect of RQ2.1: what  
384 *academic literature* suggests a good (complete) API document should comprise of.  
385 By conducting an systematic mapping study resulting in 21 primary studies, we  
386 systematically develop a taxonomy that combines the recommendations of scattered  
387 work into a structured framework of 5 dimensions and 34 weighted categorisa-  
388 tions. We then extend this work<sup>13</sup> by triangulating the taxonomy with opinions from  
389 developers using the System Usability Scale to assess the efficacy of these recom-  
390 mendations (thereby answering the second aspect of RQ2.1). From this, we assess  
391 the how well CVS providers document their APIs via a heuristic validation of the  
392 taxonomy, using the three services from the ICSME publication to make recom-  
393 mendations where documentation should be more complete, thereby answering RQ2.2  
394 (and thus RQ2).

395 **Chapter 6: Understanding developer struggles when using CVSs** This chapter  
396 has been accepted for presentation at the 2020 International Conference on Software  
397 Engineering (ICSE) [71]. We conduct a mining study of 1,425 Stack Overflow  
398 questions that provide indications of the types frustrations that developers face when  
399 integrating CVSs into their applications. To gather what their pain-points are,  
400 we use two classification taxonomies that also use Stack Overflow to understand

---

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

<sup>401</sup> generalised and documentation-specific pain-points in mature software engineering  
<sup>402</sup> (SE) domains. This study answers RQ3 in detail and provides a validation to  
<sup>403</sup> our motivation of RQ2: we validate that the *completeness* of current CVS API  
<sup>404</sup> documentation is a main concern for developers and there is insufficient explanation  
<sup>405</sup> into the errors and limitations of the service. We find that the documentation does  
<sup>406</sup> not adequately cover all aspects of the technical domain. In terms of integrating with  
<sup>407</sup> the service, developers struggle most with simple errors and ways in which to use the  
<sup>408</sup> APIs; this is in stark contrast to mature software domains. Our interpretation is that  
<sup>409</sup> developers fail to understand the IWS lifecycle and the ‘whole’ system that wraps  
<sup>410</sup> such services. We also interpret that developers have a shallower understanding  
<sup>411</sup> of the core issues within CVSs (likely due to the nuances of ML as suggested in  
<sup>412</sup> a discussion in the paper), which warrants an avenue for future work in software  
<sup>413</sup> engineering education.

<sup>414</sup> **Chapter 7: Ranking CVS pain-points by frustration** This chapter has been  
<sup>415</sup> submitted to the the 2020 International Workshop on Emotion Awareness in Software  
<sup>416</sup> Engineering (SEmotion) [73]. In this work, we use our dataset consisting of the 1,425  
<sup>417</sup> Stack Overflow (SO) questions from [71] to interpret the breakdown of emotions  
<sup>418</sup> developers express per classification of pain-points conducted in Chapter 6. We find  
<sup>419</sup> that the distribution of various emotions differ per question type, and developers are  
<sup>420</sup> most frustrated when the expectations of a CVS does not match the reality of what  
<sup>421</sup> these services actually provide, which shapes our answer for RQ3.3 and thus RQ3.

<sup>422</sup> **Chapter 8: Merging responses of multiple CVSs** This chapter was presented  
<sup>423</sup> at the 2019 International Conference on Web Engineering (ICWE) [213]. Early  
<sup>424</sup> exploration of CVSs showed that multiple services use vastly different ontologies  
<sup>425</sup> for the same input. As an initial strategy to improve the reliability of these services,  
<sup>426</sup> we explored if merging multiple responses using WordNet [197] and a novel label  
<sup>427</sup> merging algorithm based on the proportional representation approach used in polit-  
<sup>428</sup> ical voting could make any improvements. While this approach resulted in a modest  
<sup>429</sup> improvement to reliability, it did not consider to the evolution issues or developer  
<sup>430</sup> pain-points we later identified.

<sup>431</sup> **Chapter 9: Developing a confidence thresholding tool** This chapter has been  
<sup>432</sup> submitted to the demonstrations track at ICSE 2020 [70]. When integrating with a  
<sup>433</sup> CVS, developers need to select an appropriate confidence threshold suited to their  
<sup>434</sup> use case and determine whether a decision should be made. An issue, however, is  
<sup>435</sup> that these CVSs are not calibrated to the specific problem-domain datasets and it is  
<sup>436</sup> difficult for software developers to determine an appropriate confidence threshold  
<sup>437</sup> on their problem domain. This tool presents a workflow and supporting tool for  
<sup>438</sup> application developers to select decision thresholds suited to their domain that—  
<sup>439</sup> unlike existing tooling—is designed to be used in pre-development, pre-release and  
<sup>440</sup> production. This tooling forms part of a solution to RQ4 for developers to maintain  
<sup>441</sup> robustness and reliability in their systems.

<sup>442</sup> **Chapter 10: Developing a CVS integration architecture** This chapter has been  
<sup>443</sup> submitted to the 2020 Joint European Software Engineering Conference and Symposi-  
<sup>444</sup> um on the Foundations of Software Engineering [67]. ⟨ todo: *Discuss findings.* ⟩

<sup>445</sup> **Part III: Postface**

<sup>446</sup> **Chapter 11 - Conclusions & Future Work** In this chapter, we review the con-  
<sup>447</sup> tributions made in this thesis and the relevance and significance to identifying and  
<sup>448</sup> resolving key issues when application developers integrate with CVS. We evaluate  
<sup>449</sup> these outcomes with reference to the research goals, and discuss threats to validity  
<sup>450</sup> of the work. Lastly, we discuss the various avenues of research arising from this  
<sup>451</sup> work.

<sup>452</sup> **Part IV: Appendices**

<sup>453</sup> Appendix A provides additional material referenced within this thesis but not pro-  
<sup>454</sup> vided in the body. We provide mandatory coauthor declaration forms describing  
<sup>455</sup> the contribution breakdown for each publication within Appendix B. Appendix C  
<sup>456</sup> contains copies of the ethics clearance for various experiments within this thesis.  
<sup>457</sup> We describe the list of primary sources arising in the systematic mapping study we  
<sup>458</sup> conduct in Chapter 5 within Appendix D.

<sup>459</sup> **1.5.2 Coherency of Publications**

<sup>460</sup> In this section, we detail how each publication forms a coherent body of work. After  
<sup>461</sup> our exploratory analysis on the nature of CVSs (Chapter 4), we proposed two sets of  
<sup>462</sup> recommendations targeted towards two stakeholders: (i) the service *consumers* (i.e.,  
<sup>463</sup> application developers) and (ii) the service *providers*. Our subsequent publications  
<sup>464</sup> arose as a two-fold investigation to develop two strategies in which developers and  
<sup>465</sup> providers can, respectively, (i) better integrate these intelligent components into  
<sup>466</sup> their applications, and (ii) how these services can be better documented. Table 1.5  
<sup>467</sup> provides a tabulated form of the publications and research questions addressed within  
<sup>468</sup> this thesis; for ease of reference, we refer to the publications in within this section  
<sup>469</sup> in their abbreviated form as listed in Table 1.5. We also provide abbreviations for  
<sup>470</sup> easier reference in this section. A high-level overview of the cohesiveness of our  
<sup>471</sup> publications is provided in Figure 1.5.

---

<sup>15</sup>Conference 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.

<sup>16</sup>Date of publication, if applicable.

<sup>17</sup>The extended version of this conference proceeding is provided in Chapter 5.

<sup>18</sup>We 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).

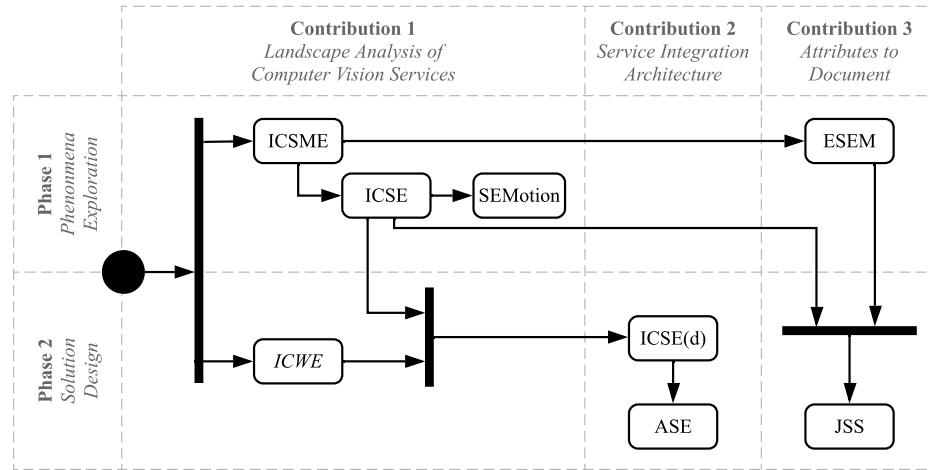
Ref.	Venue	Acronym	Rank <sup>14</sup>	Published <sup>15</sup>	Chapter	RQs
[69]	35 <sup>th</sup> International Conference on Software Maintenance and Evolution	ICSME	A	05 Dec 2019	Chapter 4	RQ1
[68]	13 <sup>th</sup> International Symposium on Empirical Software Engineering and Measurement	ESEM	A	17 Oct 2019	Excluded <sup>16</sup>	RQ2.1
[72]	Journal of Systems and Software	JSS	Q1	<i>In Press</i>	Chapter 5	RQ2
[71]	42 <sup>nd</sup> International Conference on Software Engineering	ICSE	A*	<i>In Press</i>	Chapter 6	RQ3
[73]	5 <sup>th</sup> International Workshop on Emotion Awareness in Software Engineering <sup>17</sup>	SEmotion	A*	<i>In Press</i>	Chapter 7	RQ3.3
[213]	13 <sup>th</sup> International Conference on Web Engineering	ICWE	B	26 Apr 2019	Chapter 8	RQ4
[70]	42 <sup>nd</sup> International Conference on Software Engineering	ICSE(d) <sup>18</sup>	A*	<i>In Press</i>	Chapter 9	RQ4
[67]	28 <sup>th</sup> Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering	ESEC/FSE	A*	<i>In Press</i>	Chapter 10	RQ4

## 472 Landscape Analysis and Preliminary Solution Design

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

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

485 Our ICWE paper explored one aspect identified from the ICSME paper that  
 486 we identified early on: that different services use different vocabularies to describe  
 487 semantically similar objects but in different ways (e.g., ‘border collie’ vs. ‘collie’),  
 488 despite offering functionally similar capabilities. We attempted to merge the re-  
 489 sponse labels from these services using a proportional representation approach, and  
 490 upon comparison with more naive merge approaches, we improved label-merge per-  
 491 formance by an F-measure of 0.015. However, while this was an interesting outcome  
 492 for a preliminary solution design, investigation from our following work suggested  
 493 that standardising ontologies between service providers becomes challenging and  
 494 normalising the entire ontological hierarchy of response labels would need to fall  
 495 under the responsibility of a certain body (that does not exist). Further, we did  
 496 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.

<sup>497</sup> providers. Therefore, we opted for a shielded relay architecture in our later design  
<sup>498</sup> work.

#### <sup>499</sup> Improving Documentation Attributes

<sup>500</sup> As mentioned, our ICSME paper found that evolutionary and non-deterministic  
<sup>501</sup> behavioural profile of are not adequately documented in the service’s APIs docu-  
<sup>502</sup> mentation. A recommendation concluding from this work was that service providers  
<sup>503</sup> should improve their documentation, however there lacked a strategy by which they  
<sup>504</sup> could do this, and our hypotheses that developers were actually frustrated by this  
<sup>505</sup> lack of communication was yet to be tested. This led to two follow-up further  
<sup>506</sup> investigations as presented in our ICSE and ESEM papers.

<sup>507</sup> One aspect of our ICSE paper was to confirm whether developers are actually  
<sup>508</sup> frustrated with the service’s limited API documentation. By mining Stack Overflow  
<sup>509</sup> posts with reference to documentation issues, we adopted a 2019 documentation-  
<sup>510</sup> related taxonomy by Aghajani et al. [2] to classify posts, and found that 47.87%  
<sup>511</sup> of posts classified fell under the ‘completeness’ dimension of Aghajani et al.’s  
<sup>512</sup> taxonomy. This interpretation, therefore, warranted the recommendation proposed  
<sup>513</sup> in the ICSME paper to improve service documentation.

<sup>514</sup> However, though improvements to more complete documentation was justified  
<sup>515</sup> from the ICSE paper, we needed to explore exactly *what* makes a ‘complete’ API  
<sup>516</sup> document. By conducting a systematic mapping study resulting in 4,501 results, we  
<sup>517</sup> curated 21 primary studies that outline the facets of API documentation knowledge.  
<sup>518</sup> 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.

## 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.

## 1.6 Research Contributions

⟨ *todo: would be good to have more detail HERE on the key contributions made in this work* ⟩

560      *< todo: Please include more details of the contributions - e.g. which research  
561      question is addressed in this contribution, key findings, you could also list the  
562      publications associated with each one. Also in this case you do not need section  
563      1.4.2 >*

564      The outcomes of answering these four primary research questions helps shape  
565      the three primary contributions this thesis offers to software engineering knowledge:

- 566      • An improved understanding in the landscape of CVSs, with respect to their  
567      runtime behaviour and evolutionary profiles.
- 568      • A service integration architecture that helps developers with integrating their  
569      applications with CVSs.
- 570      • A list of attributes that should be documented to assist CVS providers to better  
571      document their services.

572      These contributions are detailed further within Chapter 11. *< todo: I wonder if  
573      putting summary of key thesis contributions HERE to end the chapter might be  
574      best???* >



## CHAPTER 2

575

576

---

577

### Background

578

---

579 In Chapter 1, we defined a common set of (artificial) intelligence-based cloud ser-  
580 vices that we label intelligent web services (IWSs). Specifically, we scope the  
581 primary body of this study’s work on computer vision services (CVSs) (e.g., Google  
582 Cloud Vision [331], AWS Rekognition [318], Azure Computer Vision [339], Watson  
583 Visual Recognition [335] etc.). We claim developers have a distinctly determinis-  
584 tic mindset ( $2 + 2$  always equals 4) whereas an IWS’s ‘intelligence’ component (a  
585 black box) may return probabilistic results ( $2 + 2$  might equal 4 with a confidence  
586 of 95%). Thus, there is a mindset mismatch between probabilistic results (from the  
587 API provider) and results interpreted with certainty (from the API consumer).

588 What affect does this mindset mismatch have on the developer’s approach to-  
589 wards building probabilistic software? What can we learn from common software  
590 engineering practices (e.g., [231, 270]) that apply to resolve this mismatch and  
591 thereby improve quality, such as verification & validation (V&V)? Chiefly, we an-  
592 chor this question around three lenses of software engineering: creating an IWS,  
593 using an IWS, and the nature of IWSs themselves.

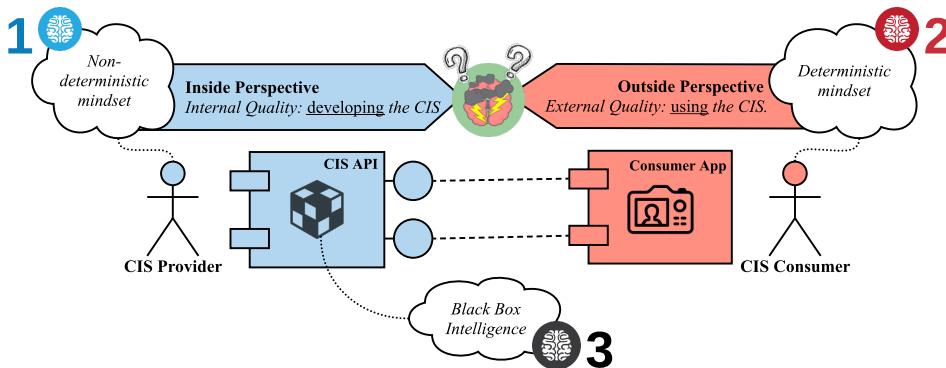
594 Our chief concern lies with interaction and integration between IWS providers  
595 and consumers, the nature of applications built using an IWS, and the impact this  
596 has on software quality. We triangulate this around three pillars, which we diagram-  
597 matically represent in Figure 2.1.

- 598 (1) **The development of the IWS.** We investigate the internal quality attributes  
599 of creating an IWS from the IWS *provider’s* perspective. That is, we ask if  
600 existing verification techniques are sufficient enough to ensure that the IWS  
601 being developed actually satisfies the IWS consumer’s needs and if the internal  
602 perspective of creating the system with a non-deterministic mindset clashes  
603 with the outside perspective (i.e., pillar 2).
- 604 (2) **The usage of the IWS.** We investigate the external quality attributes of using  
605 an IWS from the IWS *consumer’s* perspective. That is, we ask if existing  
606 validation techniques are sufficient enough to ensure that the end-users can

Pillar/Perspective	<i>Development of an IWS</i>	<i>Usage of an IWS</i>	<i>Nature of an IWS</i>
<i>Software Quality</i>	X	X	X
<i>Probabilistic &amp; Non-Deterministic Systems</i>	X	X	X
<i>APIs &amp; their usability</i>	X	X	X

607 actually use an IWS to build their software in the ways they expect the IWS to  
 608 work.

609 (3) **The nature of an IWS.** We investigate what standard software engineering  
 610 practices apply when developing non-deterministic systems. That is, we  
 611 tackle what best practices exist when developing systems that are inherently  
 612 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.

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

## 619 2.1 Software Quality

620 *Quality... you know what it is, yet you don't know what it is.*

ROBERT PIRSIG, 1974 [230]

621 The philosophical viewpoint of ‘quality’ remains highly debated and there are mul-  
 622 tiple facets to perceive this complex concept [107]. Transcendentally, a viewpoint  
 623 like that of Pirsig’s above shows that quality is not tangible but still recognisable; it’s  
 624 hard to explicitly define but you know when it’s missing. The ?CITE? provides a

625 breakdown of seven universally-applicable principles that defines quality for organi-  
626 sations, developers, customers and training providers [?CITE? ]. More pertinently,  
627 the ?CITE? ISO standard for quality was simply “the totality of characteristics of  
628 an entity that bear on its ability to satisfy stated or implied needs” [?CITE? ].

629 Using this sentence, what characteristics exist for non-deterministic IWSs like  
630 that of a CVS? How do we know when the system has satisfied its ‘stated or implied  
631 needs’ when the system can only give us uncertain probabilities in its outputs? Such  
632 answers can be derived from related definitions—such as ‘conformance to specifica-  
633 tion or requirements’ [66, 111], ‘meeting or exceeding customer expectation’ [28],  
634 or ‘fitness for use’ [145]—but these then still depend on the solution description or  
635 requirements specification, and thus the same questions still apply.

636 *Software* quality is somewhat more concrete. Pressman [231] adapted the  
637 manufacturing-oriented view of quality from [29] and phrased software quality  
638 under three core pillars:

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

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

### 659 2.1.1 Validation and Verification

660 To explain V&V, we analogously recount a tale given by Pham [228] on his works  
661 on reliability. A high-school student sat a standardised test that was sent to 350,0000  
662 students [280]. A multiple-choice algebraic equation problem used a variable, *a*,  
663 and intended that students *assume* that the variable was non-negative. Without  
664 making this assumption explicit, there were two correct answers to the multiple  
665 choice answer. Up to 45,000 students had their scores retrospectively boosted by up  
666 to 30 points for those who ‘incorrectly’ answered, however, outcomes of a student’s

667 higher education were, thereby, affected by this one oversight in quality assessment.  
668 The examiners wrote a poor question due to poor process standards to check if  
669 their ‘correct’ answers were actually correct. The examiners “didn’t build the right  
670 product” nor did they “build the product right” by writing an poor question and  
671 failing to ensure quality standards, in the phrases Boehm [37] coined.

672 This story describes the issues with the cost of quality [36] and the importance  
673 of V&V: just as the poorly written exam question had such a high toll the 45,000  
674 unlucky students, so does poorly written software in production. As summarised by  
675 Pressman [231], data sourced from Cigital [60] in a large-scale application showed  
676 that the difference in cost to fix a bug in development versus system testing is  
677 \$6,159 per error. In safety-critical systems, such as self-driving cars or clinical  
678 decision support systems, this cost skyrockets due to the extreme discipline needed  
679 to minimise error [282].

680 Formally, we refer to the IEEE Standard Glossary of Software Engineering  
681 Terminology [133] for to define V&V:

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

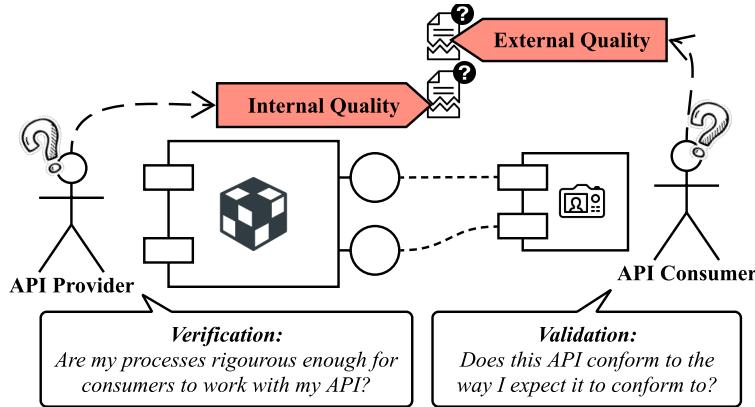
685 **validation** The process of evaluating a system or component during or at the  
686 end of the development process to determine whether it satisfies  
687 specified requirements.

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

690 The verification process of API providers ‘leak’ out to the context of the de-  
691 veloper’s project dependent on the IWS. Poor verification in the *internal quality*  
692 of the IWS will entail poor process standards, such as poor definitions and termi-  
693 nology used, support tooling and description of documentations [270]. Though  
694 it is commonplace for providers to have a ‘ship-first-fix-later’ mentality of ‘good-  
695 enough’ software [291], the consequence of doing so leads to consumers absorbing  
696 the cost. Thus API providers must ensure that their verification strategies  
697 are rigorous enough for the consumers in the myriad contexts they wish to use  
698 it in. Studies have considered V&V in the context of web services on the cloud  
699 [15, 51, 52, 96, 122, 204, 206, 309], though little have recently considered how  
700 adding ‘intelligence’ to these services affects existing proposed frameworks and  
701 solutions. For a CVS, what might this entail? Which assurances are given to the  
702 consumers, and how is that information communicated? To verify if the service is  
703 working correctly, does that mean that we need to deploy the system first to get a  
704 wider range of data, given the stochastic nature of the black box?

705 Likewise, the validation perspective comes from that of the consumer. While the  
706 former perspective is of creation, this perspective comes from end-user (developer)  
707 expectation. As described in Chapter 1, a developer calls the IWS component using  
708 an API endpoint. Again, the mindset problem arises; does the developer know what  
709 to expect in the output? What are their expectations for their specific context? In

710 the area of non-deterministic systems of probabilistic output, can the developer be  
711 assured that what they enter in a testing phase outcome the same result when in  
712 production?



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

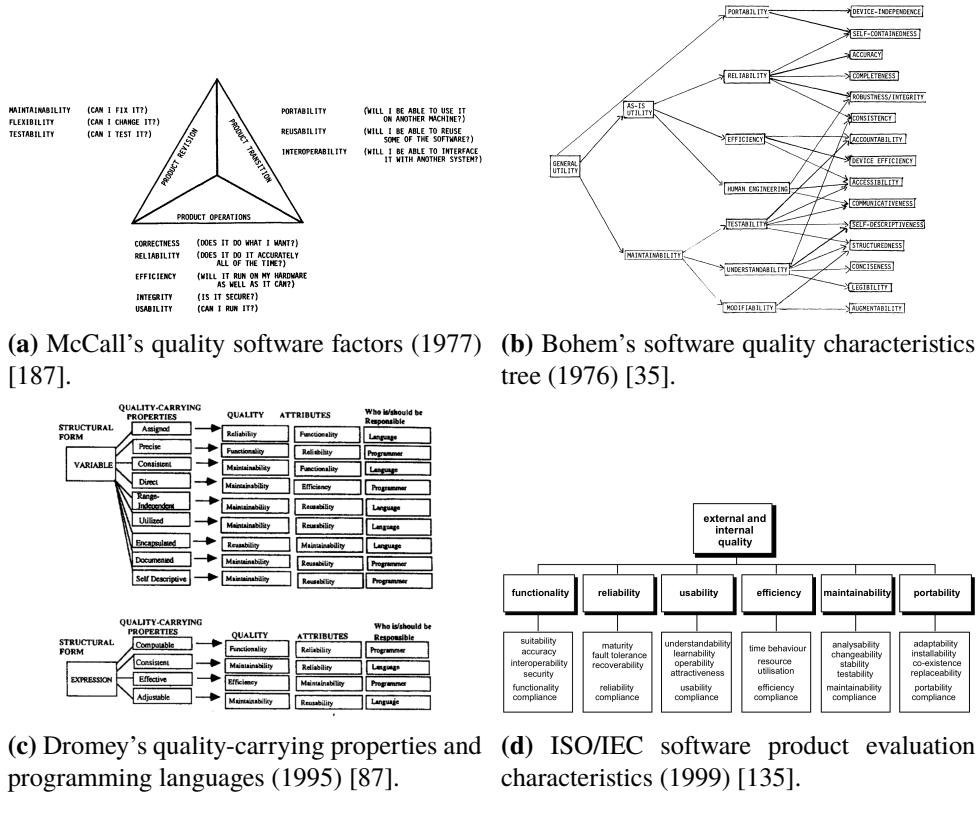
713 Therefore, just as the test answers were both correct and incorrect at the  
714 same time, so is the same with IWSs returning a probabilistic result: no result is  
715 certain. While V&V has been investigated in the area of mathematical and earth  
716 sciences for numerical probabilistic models and natural systems [215, 252], from  
717 the software engineering literature, little work has been achieved to look at the  
718 surrounding area of probabilistic systems hidden behind API calls.

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

### 731 2.1.2 Quality Attributes and Models

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

736 Quality attribute assessment models (like those shown in Figure 2.3) are an early  
737 concept in software engineering, and systematically evaluating software quality



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

738 appears as early as 1968 [251]. Rubey and Hartwick's 1968 study introduced the  
 739 phrase 'attributes' as a "prose expression of the particular quality of desired software"  
 740 (as worded by Boehm et al. [35]) and 'metrics' as mathematical parameters on a  
 741 scale of 0 to 100. Early attempts to categorise wider factors under a framework was  
 742 proposed by McCall, Richards, and Walters in the late 1970s [55, 187]. This model  
 743 described quality from the three perspectives of product revision (*how can we keep*  
 744 *the system operational?*), transition (*how can we migrate the system as needed?*)  
 745 and operation (*how effective is the system at achieving its tasks?*) (Figure 2.3a).  
 746 The model also introduced 11 attributes alongside numerous direct and indirect  
 747 measures to help quantify quality. This model was further developed by Boehm  
 748 et al. [35] who independently developed a similar model, starting with an initial set  
 749 of 11 software characteristics. It further defined candidate measurements of Fortran  
 750 code to such characteristics, taking shape in a tree-like structure as in Figure 2.3b.  
 751 In the mid-1990s, Dromey's interpretation [87] defined a set of quality-carrying  
 752 properties with structural forms associated to specific programming languages and  
 753 conventions (Figure 2.3c). The model also supported quality defect identification  
 754 and proposed an improved auditing method to automate defect detection for code  
 755 editors in IDEs. As the need for quality models became prevalent, the International  
 756 Organization for Standardization standardised software quality under ISO/IEC-9126  
 757 [135] (the Software Product Evaluation Characteristics, Figure 2.3d), which has since

758 recently been revised to ISO/IEC-25010 with the introduction of the Systems and  
759 software Quality Requirements and Evaluation (SQuaRE) model [134], separating  
760 quality into *Product Quality* (consisting of eight quality characteristics and 31 sub-  
761 characteristics) and *Quality In Use* (consisting of five quality characteristics and 9  
762 sub-characteristics). An extensive review on the development of quality models in  
763 software engineering is given in [5].

764 Of all the models described, there is one quality attribute that relates most  
765 with our narrative of IWS quality: reliability. Reliability is the primary quality  
766 factor investigated within this thesis (see Section 1.3). Both McCall and Boehm's  
767 quality models have sub-characteristics of reliability relating to the primary research  
768 questions that investigate the *robustness*, *consistency* and *completeness*<sup>1</sup> of CVSs  
769 and its associated documentation. Moreover, the definition of reliability is similar  
770 among all quality models:

771 **McCall et al.** Extent to which a program can be expected to perform its in-  
772 tended function with required precision [187].

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

776 **Dromey** Functionality implies reliability. The reliability of software is  
777 therefore dependent on the same properties as functionality, that  
778 is, the correctness properties of a program [87].

779 **ISO/IEC-9126** The capability of the software product to maintain a specified  
780 level of performance when used under specified conditions [135].

781 These definitions strongly relate to the system's solution description in that  
782 reliability is the ability to maintain its *functionality* under given conditions. But what  
783 defines reliability when the nature of an IWS in itself is inherently unpredictable  
784 due to its probabilistic implementation? Can a non-deterministic system ever be  
785 considered reliable when the output of the system is uncertain? How do developers  
786 perceive these quality aspects of reliability in the context of such systems? A system  
787 cannot be perceived as 'reliable' if the system cannot reproduce the same results due  
788 to a probabilistic nature. Therefore, we believe the literature of quality models does  
789 not suffice in the context of IWS reliability; a CVS can interpret an image of a dog  
790 as a 'Dog' one day, but what if the next it interprets such image more specifically to  
791 the breed, such as 'Border Collie'? Does this now mean the system is unreliable?

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

---

<sup>1</sup>In 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.

### 795 2.1.3 Reliability in Computer Vision

796 Testing computer vision deep-learning reliability is an area explored typically  
797 through the use of adversarial examples [278]. These input examples are where  
798 images are slightly perturbed to maximise prediction error but are still interpretable  
799 to humans. Refer to Figure 2.4.

800 Google Cloud Vision, for instance, fails to correctly classify adversarial examples  
801 when noise is added to the original images [128]. Rosenfeld et al. [249] illustrated  
802 that inserting synthetic foreign objects to input images (e.g., a cartoon elephant)  
803 can alter classification output. Wang et al. [294] performed similar attacks on a  
804 transfer-learning approach of facial recognition by modifying pixels of a celebrity’s  
805 face to be recognised as a different celebrity, all while still retaining the same human-  
806 interpretable original celebrity. Su et al. [273] used the ImageNet database to show  
807 that 41.22% of images drop in confidence when just a *single pixel* is changed in the  
808 input image; and similarly, Eykholt et al. [91] recently showed similar results that  
809 made a CNN interpret a stop road-sign (with mimiccid graffiti) as a 45mph speed  
810 limit sign.

811 Thus, the state-of-the-art computer vision techniques may not be reliable enough  
812 for safety critical applications (such as self-driving cars) as they do not handle inten-  
813 tional or unintentional adversarial attacks. Moreover, as such adversarial examples  
814 exist in the physical world [91, 163], “the real world may be adversarial enough”  
815 [227] to fool such software.

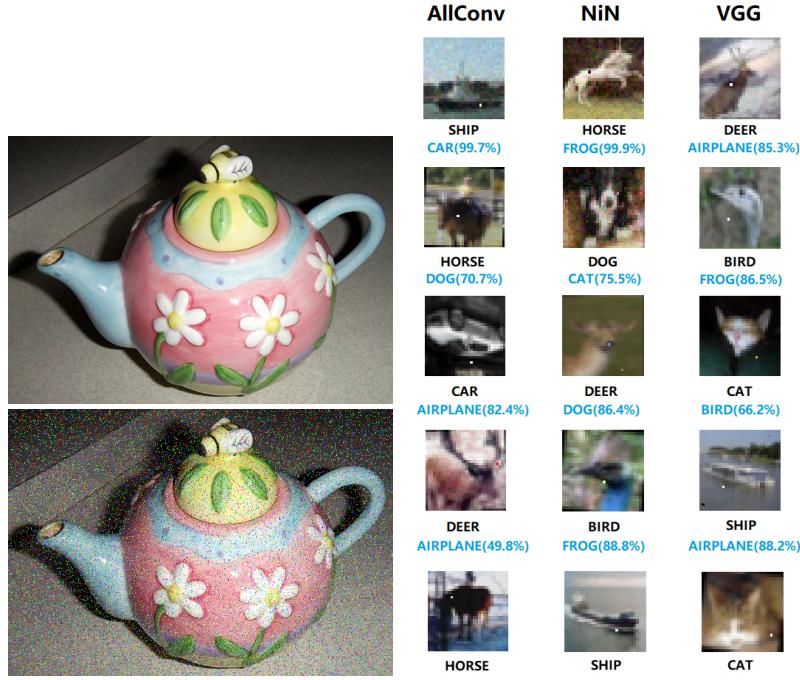
## 816 2.2 Probabilistic and Nondeterministic Systems

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

### 822 2.2.1 Interpreting the Uninterpretable

823 As the rise of applied AI increases, the need for engineering interpretability around  
824 models becomes paramount, chiefly from an external quality perspective that the  
825 *reliability* of the system can be inspected by end-users. Model interpretability has  
826 been stressed since early machine learning research in the late 1980s and 1990s (such  
827 as Quinlan [232] and Michie [196]), and although there has since been a significant  
828 body of work in the area [13, 26, 39, 50, 77, 93, 102, 110, 142, 172, 175, 185, 222,  
829 237, 250, 267, 290, 292], it is evident that ‘accuracy’ or model ‘confidence’ is still  
830 used as a primary criterion for AI evaluation [131, 137, 269]. Much research into  
831 neural network (NN) or support vector machine (SVM) development stresses that  
832 ‘good’ models are those with high accuracy. However, is accuracy enough to justify  
833 a model’s quality?

834 To answer this, we revisit what it means for a model to be accurate. Accuracy  
835 is an indicator for estimating how well a model’s algorithm will work with future



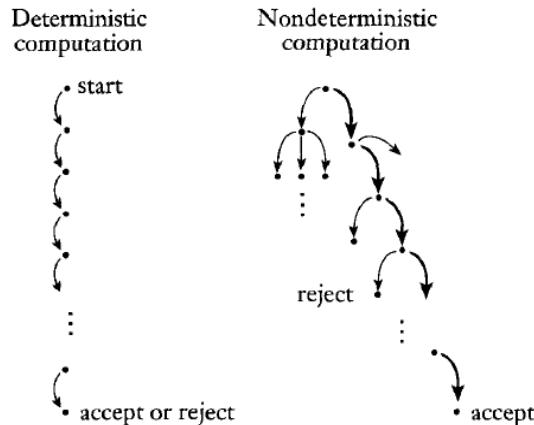
(a) Adding 10% impulse noise to an image of a teapot changes Google Cloud Vision's label from *teapot* (above) to *biology* (below) [128].

(b) One-pixel attacks applied to three neural network (NN): AllConv, NiN and VGG [273].



(c) Adversarial examples to trick face recognition from the source to target images [294].

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



**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: [95].

836 or unforeseen data. It is quantified in the AI testing stage, whereby the algorithm  
 837 is tested against cases known by humans to have ground truth but such cases are  
 838 unknown by the algorithm. In production, however, all cases are unknown by both  
 839 the algorithm *and* the humans behind it, and therefore a single value of quality is  
 840 “not reliable if the future dataset has a probability distribution significantly different  
 841 from past data” [98], a problem commonly referred to as the *datashift* problem [254].  
 842 Analogously, Freitas [98] provides the following description of the problem:

843 *The military trained [a NN] to classify images of tanks into enemy  
 844 and friendly tanks. However, when the [NN] was deployed in the field  
 845 (corresponding to “future data”), it had a poor accuracy rate. Later,  
 846 users noted that all photos of friendly (enemy) tanks were taken on a  
 847 sunny (overcast) day. I.e., the [NN] learned to discriminate between  
 848 the colors of the sky in sunny vs. overcast days! If the [NN] had  
 849 output a comprehensible model (explaining that it was discriminating  
 850 between colors at the top of the images), such a trivial mistake would  
 851 immediately be noted.* [98]

852 So, why must we interpret models? While the formal definition of what it means  
 853 to be *interpretable* is still somewhat disparate (though some suggestions have been  
 854 proposed [175]), what is known is (i) there exists a critical trade-off between accuracy  
 855 and interpretability [82, 97, 115, 141, 149, 311], and (ii) a single quantifiable value  
 856 cannot satisfy the subjective needs of end-users [98]. As ever-growing domains  
 857 ML become widespread<sup>2</sup>, these applications engage end-users for real-world goals,  
 858 unlike the aims in early ML research where the aim was to get AI working in the  
 859 first place. In safety-critical systems where AI provide informativeness to humans

<sup>2</sup>In areas such as medicine [25, 50, 90, 138, 142, 168, 223, 239, 290, 308, 313], bioinformatics [81, 99, 139, 148, 277], finance [13, 79, 132] and customer analytics [172, 292].

860 to make the final call (see [53, 132, 151]), there is often a mismatch between the  
861 formal objectives of the model (e.g., to minimise error) and complex real-world  
862 goals, where other considerations (such as the human factors and cognitive science  
863 behind explanations<sup>3</sup>) are not realised: model optimisation is only worthwhile if they  
864 “actually solve the original [human-centred] task of providing explanation” [205]  
865 to end-users. **Therefore, when human-decision makers must be interpretable**  
866 **themselves [240], any AI they depend on must also be interpretable.**

867 Recently, discussion behind such a notion to provide legal implications of in-  
868 terpretability is topical. Doshi-Velez et al. [85] discuss when explanations are not  
869 provided from a legal stance—for instance, those affected by algorithmic-based de-  
870 cisions have a ‘right to explanation’ [183, 293] under the European Union’s GDPR<sup>4</sup>.  
871 But, explanations are not the only way to ensure AI accountability: theoretical  
872 guarantees (mathematical proofs) or statistical evidence can also serve as guarantees  
873 [85], however, in terms of explanations, what form they take and how they are proven  
874 correct are still open questions [175].

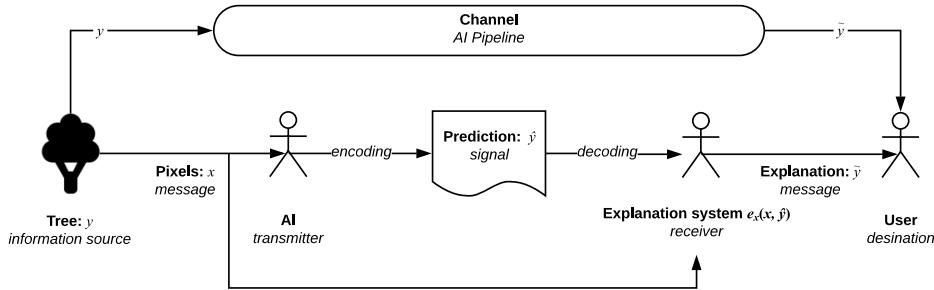
### 875 2.2.2 Explanation and Communication

876 From a software engineering perspective, explanations and interpretability are, by  
877 definition, inherently communication issues: what lacks here is a consistent interface  
878 between the AI system and the person using it. The ability to encode ‘common  
879 sense reasoning’ [188] into programs today has been achieved, but *decoding* that  
880 information is what still remains problematic. At a high level, Shannon and Weaver’s  
881 theory of communication [261] applies, just as others have done with similar issues in  
882 the SE realm [199, 303] (albeit to the domain of visual notations). Humans map the  
883 world in higher-level concepts easily when compared to AI systems: while we think  
884 of a tree first (not the photons of light or atoms that make up the tree), an algorithm  
885 simply sees pixels, and not the concrete object [85] and the AI interprets the tree  
886 inversely to humans. Therefore, the interpretation or explanation is done inversely:  
887 humans do not explain the individual neurons fired to explain their predictions, and  
888 therefore the algorithmic transparent explanations of AI algorithms (“*which neurons*  
889 *were fired to make this AI think this tree is a tree?*”) do not work here.

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

<sup>3</sup>Interpretations and explanations are often used interchangeably.

<sup>4</sup><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  $\tilde{y}$ .

### 900 2.2.3 Mechanics of Model Interpretation

901 How do we interpret models? Methods for developing interpretation models include:  
902 decision trees [45, 64, 120, 179, 233], decision tables [14, 172] and decision sets  
903 [165, 205]; input gradients, gradient vectors or sensitivity analysis [13, 169, 237,  
904 250, 259]; exemplars [100, 152]; generalised additive models [53]; classification  
905 (*if-then*) rules [41, 61, 217, 285, 305] and falling rule lists [267]; nearest neighbours  
906 [185, 234, 260, 301, 312] and Naïve Bayes analysis [25, 57, 92, 101, 123, 159, 168,  
907 313].

908 Cross-domain studies have assessed the interpretability of these techniques  
909 against end-users, measuring response time, accuracy in model response and user  
910 confidence [6, 99, 121, 132, 185, 253, 274, 292], although it is generally agreed  
911 that decision rules and decision tables provide the most interpretation in non-linear  
912 models such as SVMs or NNs [99, 185, 292]. For an extensive survey of the benefits  
913 and fallbacks of these techniques, we refer to Freitas [98], Doshi-Velez et al. [85]  
914 and Doshi-Velez and Kim [84].

## 915 2.3 Application Programming Interfaces

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

922 Good APIs are known to be intuitive and require less documentation browsing  
923 [229], thereby increasing developer productivity. Conversely, poor APIs are those  
924 that are hard to interpret, thereby reducing developer productivity and product quality.  
925 The consequences of this have shown a higher demand of technical support (as  
926 measured in [124]) that, ultimately, causes the maintenance to be far more expensive,  
927 a phenomenon widely known in software engineering economics (see Section 2.1.1).

928 While there are different types of APIs, such as software library/framework

929 APIs for building desktop software, operating system APIs for interacting with the  
930 operating system, remote APIs for communication of varying technologies through  
931 common protocols, we focus on web APIs for communication of resources over  
932 the web (being the common architecture of cloud-based services). Further infor-  
933 mation on the development, usage and documentation of web APIs is provided in  
934 Appendix A.1.

### 935 2.3.1 API Usability

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

941 Balancing the understanding of the information need (both conscious and un-  
942 conscious), how to phrase that need and how to query it in an information retrieval  
943 system is concept long studied in the information sciences [283]. In API design,  
944 the most common form to convey knowledge to developers is through annotated  
945 code examples and overviews to a platform's architectural and design decisions  
946 [42, 83, 202, 245] though these studies have not effectively communicated *why* these  
947 artefacts are important. What makes the developer *conceptually understand* these  
948 artefacts?

949 Robillard and Deline [245] conducted a multi-phase, mixed-method approach to  
950 create knowledge grounded in the professional experience of 440 software engineers  
951 at Microsoft of varying experience to determine what makes APIs hard to learn,  
952 the results of which previously published in an earlier report [244]. Their results  
953 demonstrate that 'documentation-related obstacles' are the biggest hurdle in learning  
954 new APIs. One of these implications are the *intent documentation* of an API (i.e.,  
955 *what is the intent for using a particular API?*) and such documentation is required  
956 only where correct API usage is not self-evident, where advanced uses of the API are  
957 documented (but not the intent), and where performance aspects of the API impact  
958 the application developed using it. They conclude that professional developers do  
959 not struggle with learning the *mechanics* of the API, but in the *understanding* of how  
960 the API fits in upwards to its problem domain and downward to its implementation:

961 *In the upwards direction, the study found that developers need help*  
962 *mapping desired scenarios in the problem domain to the content of the*  
963 *API, and in understanding what scenarios or usage patterns the API*  
964 *provider intends and does not intend to support. In the downwards*  
965 *direction, developers want to understand how the API's implementation*  
966 *consumes resources, reports errors and has side effects.* [245]

967 These results particularly corroborate to that of previous studies where devel-  
968 opers quote that they feel that existing learning content currently focuses on "how  
969 to do things, not necessarily why" [212]. This thereby reiterates the conceptual  
970 understanding of an API as paramount.

<sup>971</sup> A later study by Ko and Riche [157] assessed the importance of a programmer’s  
<sup>972</sup> conceptual understanding of the background behind the task before implementing the  
<sup>973</sup> task itself, a notion that we find most relevant for users of IWS APIs. While the study  
<sup>974</sup> did not focus on developing web APIs (rather implementing a Bluetooth application  
<sup>975</sup> using platform-agnostic terminology), the study demonstrated how developers show  
<sup>976</sup> little confidence in their own metacognitive judgements to understand and assess the  
<sup>977</sup> feasibility of the intent of the API and understand the vocabulary and concepts within  
<sup>978</sup> the domain (i.e., wireless connectivity). This indecision over what search results  
<sup>979</sup> were relevant in their searches ultimately hindered their progress implementing the  
<sup>980</sup> functionality, again decreasing productivity. Ko and Riche suggest to improve API  
<sup>981</sup> usability by introducing the background of the API and its relevant concepts using  
<sup>982</sup> glossaries linked to tutorials to each of the major concepts, and then relate it back to  
<sup>983</sup> how to implement the particular functionality.

<sup>984</sup> Thus, an analysis of the conceptual understanding of IWS APIs by a range of  
<sup>985</sup> developers (from beginner to professional) is critical to best understand any differ-  
<sup>986</sup> ences between existing studies and those that are nondeterministic. Our proposal is  
<sup>987</sup> to perform similar survey research (see Chapter 3) in the search for further insight  
<sup>988</sup> into the developer’s approach toward existing IWS APIs.

# CHAPTER 3

989

990

---

991

## Research Methodology

992

---

993 Investigating software engineering practices is often a complex task as it is imperative  
994 to understand the social and cognitive processes around software engineers and not  
995 just the tools and processes used [89]. This chapter explores our research method-  
996 ology by exploring five key elements of empirical software engineering research:  
997 firstly, (i) we provide an extended focus to the study by reviewing our research  
998 questions (see Section 1.3) anchored under the context of an existing classifica-  
999 tion taxonomy, (ii) characterise our research goals through an explicit philosophical  
1000 stance, (iii) explain how the stance selected impacts our selection of research meth-  
1001 ods and data collection techniques (by dissecting our choice of methods used to  
1002 reach these research goals), (iv) discuss a set of criteria for assessing the validity of  
1003 our study design and the findings of our research, and lastly (v) discuss the practical  
1004 considerations of our chosen methods.

1005 The foundations for developing this research methodology has been expanded  
1006 from that proposed by Easterbrook et al. [89], Wohlin and Aurum [306], Wohlin  
1007 et al. [307] and Shaw [262].

### 1008 3.1 Research Questions Revisited

1009 In Section 1.3, we introduce three hypothesis of this study (RH1–RH3), namely: (i)  
1010 existing IWS APIs are poorly documented for general use (RH1); (ii) existing IWS  
1011 APIs do not provide sufficient metadata when used in context-specific use cases  
1012 (RH2); and (iii) the combination of improving documentation and metadata will  
1013 ultimately improve one of software quality, developer productivity and/or developer  
1014 understanding (RH3).

1015 To discuss our research strategy, we revisit our research questions through the  
1016 classification technique discussed by Easterbrook et al. [89], a technique originally  
1017 proposed in the field of psychology by Meltzoff and Cooper [192] but adapted to  
1018 software engineering. Our research study involves a mix of five *knowledge questions*,

1019 that focus on existing practices and the ways in which they work, and two *design*  
1020 *questions*, that focuses on designing better ways to approach software engineering  
1021 tasks [195]. Both classes of questions are respectively concerned with empirical and  
1022 non-empirical software engineering that, in practice, are best combined in long-term  
1023 software engineering research studies (such as this one) as they assist in tackling the  
1024 investigation of a specific problem, approaches to solve that problem and finding  
1025 what solutions work best [304].

### 1026 3.1.1 Knowledge Questions

1027 In total, five knowledge questions are posed in this study to help us understand the  
1028 way developers currently interact and work with an IWS API; two exploratory, one  
1029 base-rate, and two relationship and causality questions.

1030 We begin by formulating two *exploratory questions* to attempt to better under-  
1031 stand the phenomena of poor API documentation and metadata; both ?REF? and  
1032 ?REF? respectively describe and classify what practices are in use for existing IWS  
1033 API documentation and what problems currently exist when no metadata is returned.  
1034 Answering these two questions assists in refining preciser terms of the phenomena,  
1035 ways in which we find evidence for them and ensuring the data found is valid.

1036 By answering these questions, we have a clearer understanding of the phenom-  
1037 ena; we then follow up by posing an additional *base-rate question* that helps provide  
1038 a basis to confirm that the phenomena occurring is normal (or unusual) behaviour  
1039 by investigating the patterns of phenomena's occurrence. ?REF? is a descriptive-  
1040 process question to help us understand how the developer currently understands  
1041 existing IWS API documentation, given their lack of formal extended training in  
1042 artificial intelligence. This achieves us an insight into the developer's mindset and  
1043 regular thought patterns toward these APIs.

1044 Lastly, we investigate the relationship between the improved documentation  
1045 and improvements to other aspects of the software development process. Chiefly,  
1046 ?REF? is concerned with whether any improvements to metadata or documentation  
1047 correlate to improvements in software quality, developer productivity, or developer  
1048 education (and is a *relationship establishment question*). If we establish such a  
1049 relationship, we refine the question and investigate the specific causes using three  
1050 *causality questions* defined under ?REF?, namely by associating three classes of  
1051 measurable metrics (internal quality metrics, external quality metrics, developer  
1052 education insight metrics) to the improved documentation.

### 1053 3.1.2 Design Questions

1054 ?REF? and ?REF? are both *design questions*; they are concerned with ways in  
1055 which we can improve an IWS by investigating what additional attributes are needed  
1056 in both the documentation and metadata that assist developers to achieve their goals.  
1057 They are not classified as knowledge questions as we investigate what *will be* and  
1058 not *what is*. By understanding the process by which developers desire additional  
1059 attributes of metadata and documentation, we can help shape improvements to the  
1060 existing design of an IWS.

## 1061 3.2 Philosophical Stances

1062 Philosophical stances guide the researcher's action by fortifying what constitutes  
1063 'valid truth' against a fundamental set of core beliefs [242]. In software engineering,  
1064 four dominant philosophical stances are commonly characterised [65, 225]:  
1065 positivism (or post-positivism), constructivism (or interpretivism), pragmatism, and  
1066 critical theory (or advocacy/participatory). To construct such a 'validity of truth',  
1067 we will review these four philosophical stances in this section, and state the stance  
1068 that we explicitly adopt and our reasoning for this.

1069 **Positivism** Positivists claim truth to be all observable facts, reduced piece-by-  
1070 piece to smaller components which is incrementally verifiable to form truth. We  
1071 do not base our work on the positivistic stance as the theories governing verifiable  
1072 hypothesis must be precise from the start of the research. Moreover, due to its  
1073 reductionist approach, it is difficult to isolate these hypotheses and study them in  
1074 isolation from context. As our hypotheses are not context-agnostic, we steer clear  
1075 from this stance.

1076 **Constructivism** Constructivists see knowledge embedded within the human context;  
1077 truth is the *interpretive* observation by understanding the differences in human  
1078 thought between meaning and action [156]. That is, the interpretation of the theory  
1079 is just as important to the empirical observation itself. We partially adopt a con-  
1080 structivist stance as we attempt to model the developer's mindset, being an approach  
1081 that is rich in qualitative data on human activity.

1082 **Pragmatism** Pragmatism is a less dogmatic approach that encourages the incom-  
1083 plete and approximate nature of knowledge and is dependent on the methods in which  
1084 the knowledge was extracted. The utility of consensually agreed knowledge is the  
1085 key outcome, and is therefore relative to those who seek utility in the knowledge—  
1086 what is the useful for one person is not so for the other. While we value the utility  
1087 of knowledge, it is difficult to obtain consensus especially on an ill-researched topic  
1088 such as ours, and therefore we do not adopt this stance.

1089 **Critical Theory** This study chiefly adopts the philosophy of critical theory [8]. A  
1090 key outcome of the study is to shift the developer's restrictive deterministic mindset  
1091 and shed light on developing a new framework actively with the developer community  
1092 that seeks to improve the process of using such APIs. In software engineering,  
1093 critical theory is used to "actively [seek] to challenge existing perceptions about  
1094 software practice" [89], and this study utilises such an approach to shift the mindset  
1095 of IWS consumers and providers alike on how the documentation and metadata  
1096 should not be written with the 'traditional' deterministic mindset at heart. Thus, our  
1097 key philosophical approach is critical theory to seek out *what-can-be* using partial  
1098 constructivism to model the current *what-is*.

### 1099 3.3 Research Design

1100 Research methods are “a set of organising principles around which empirical data  
1101 is collection and analysed” [89]. Creswell [65] suggests that strong research design  
1102 is reflected when the weaknesses of multiple methods complement each other. Us-  
1103 ing a mixed-methods approach is therefore commonplace in software engineering  
1104 research, typically due to the human-oriented nature investigating how software en-  
1105 gineers work both individually (where methods from psychology may be employed)  
1106 and together (where methods from sociology may be employed).

1107 Therefore, studies in software engineering are typically performed as field studies  
1108 where researchers and developers (or the artefacts they produce) are analysed either  
1109 directly or indirectly [266]. The mixed-methods approach combines five classes  
1110 of field study methods (or empirical strategies/studies) most relevant in empirical  
1111 software engineering research [89, 147, 307]: controlled experiments, case studies,  
1112 survey research, ethnographies, and action research. We chiefly adopt a mixed-  
1113 methods approach to our work using the *concurrent triangulation* mixed-methods  
1114 strategy [186] as it best compensates for weaknesses that exist in all research methods,  
1115 and employs the best strengths of others.

#### 1116 3.3.1 Review of Relevant Research Methods

1117 Below we review some of the research methods most relevant to our research ques-  
1118 tions as refined in Section 3.1 as presented by Easterbrook et al. [89].

1119 **Controlled Experiments** A controlled experiment is an investigation of a clear,  
1120 testable hypothesis that guides the researcher to decide and precisely measure how  
1121 at least one independent variable can be manipulated and effect at least one other  
1122 dependent variable. They determine if the two variables are related and if a cause-  
1123 effect relationship exists between them. The combination of independent variable  
1124 values is a *treatment*. It is common to recruit human subjects to perform a task and  
1125 measure the effect of a randomly assigned treatment on the subjects, though it is  
1126 not always possible to achieve full randomisation in real-life software engineering  
1127 contexts, in which case a *quasi-experiment* may be employed where subjects are not  
1128 randomly assigned to treatments.

1129 While we have defined hypotheses (RH1–RH3), refining them into precise,  
1130 measurable variables is challenging due to the qualitative nature they present. A  
1131 well-defined population is also critical and must be easily accessible; the varied  
1132 range of beginner to expert software engineers with varied understanding of artifi-  
1133 cial intelligence concepts is required to perform controlled experiments, and thus  
1134 recruitment may prove challenging. Lastly, the controlled experiment is essentially  
1135 reductionist by affecting a small amount of variables of interest and controlling all  
1136 others. This approach is too clinical for the practical outcomes by which our research  
1137 goals aim for, and is therefore closely tied to the positivist stance.

1138 **Case Studies** Case studies investigate phenomena in their real-life context and are  
1139 well-suited when the boundary between context and phenomena is unknown [310].  
1140 They offer understanding of how and why certain phenomena occur, thereby investi-  
1141 gating ways cause-effect relationships can occur. They can be used to test existing  
1142 theories (*confirmatory case studies*) by refuting theories in real-world contexts in-  
1143 stead of under laboratory conditions or to generate new hypotheses and build theories  
1144 during the initial investigation of some phenomena (*exploratory case studies*).

1145 Case studies are well-suited where the context of a situation plays a role in the  
1146 phenomenon being studied, which we specifically relate back to RH2 (?REF? and  
1147 ?REF?) in exploring whether the context of an application using an IWS requires the  
1148 IWS context-specific or of context-agnostic. They also lend themselves to purposive  
1149 sampling rather than random sampling, and thus we can selectively choose cases that  
1150 benefit the research goal of RH2 and (using our critical theorist stance) select cases  
1151 that will actively benefit our participant software engineering audience most to draw  
1152 attention to situations regarded as most problematic.

1153 **Survey Research** Survey research identifies characteristics of a broad population  
1154 of individuals through direct data collection techniques such as interviews and ques-  
1155 tionnaires or independent techniques such as data logging. Defining that well-defined  
1156 population is critical, and selecting a representative sample from it to generalise the  
1157 data gathered usually assists in answering base-rate questions.

1158 By identifying representative sample of the population, from beginner to ex-  
1159 perienced developers with varying understanding of IWS APIs, we can use survey  
1160 research to assist in answering our exploratory and base-rate research questions un-  
1161 der RH1 and RH2 (see Section 3.1.1) in determining the qualitative aspects of how  
1162 individual developers perceive and work with the existing APIs, either by directly  
1163 asking them or by mining third-party discussion websites such as Stack Overflow  
1164 (SO). However, with direct survey research techniques, low response rates may prove  
1165 challenging, especially if no inducements can be offered for participation.

1166 **Ethnographies** Ethnographies investigates the understanding of social interac-  
1167 tion within community through field observation [246]. Resulting ethnographies  
1168 help understand how software engineering technical communities build practices,  
1169 communication strategies and perform technical work collaboratively.

1170 Ethnographies require the researcher to be highly trained in observational and  
1171 qualitative data analysis, especially if the form of ethnography is participant observa-  
1172 tion, whereby the researcher is embedded in the technical community for observation.  
1173 This may require the longevity of the study to be far greater than a couple of weeks,  
1174 and the researcher must remain part of the project for its duration to develop enough  
1175 local theories about how the community functions. While it assists in revealing  
1176 subtle but important aspects of work practices within software teams, this study  
1177 does not focus on the study of teams, and is therefore not a research method relevant  
1178 to this project.

<sup>1179</sup> **Action Research** Action researchers simultaneously solve real-world problems  
<sup>1180</sup> while studying the experience of solving the problem [75] by actively seeking to  
<sup>1181</sup> intervene in the situation for the purpose of improving it. A precondition is to engage  
<sup>1182</sup> with a *problem owner* who is willing to collaborate in identifying and solving the  
<sup>1183</sup> problem faced. The problem must be authentic (a problem worth solving) and must  
<sup>1184</sup> have new knowledge outcomes for those involved. It is also characterised as an  
<sup>1185</sup> iterative approach to problem solving, where the knowledge gained from solving the  
<sup>1186</sup> problem has a desirable solution that empowers the problem owner and researcher.

<sup>1187</sup> This research is most associated to our adopted philosophical stance of critical  
<sup>1188</sup> theory. As this project is being conducted under the Applied Artificial Intelligence  
<sup>1189</sup> Institute ( $A^2I^2$ ) collaboratively with engaged industry clients, we have identified a  
<sup>1190</sup> need for solving an authentic problem that industry faces. The desired outcome  
<sup>1191</sup> of this project is to facilitate wider change in the usage and development of CVSs;  
<sup>1192</sup> thus, engaging action research as a primary method throughout the mixed-methods  
<sup>1193</sup> approach used in this research.

### <sup>1194</sup> 3.3.2 Review of Data Collection Techniques for Field Studies

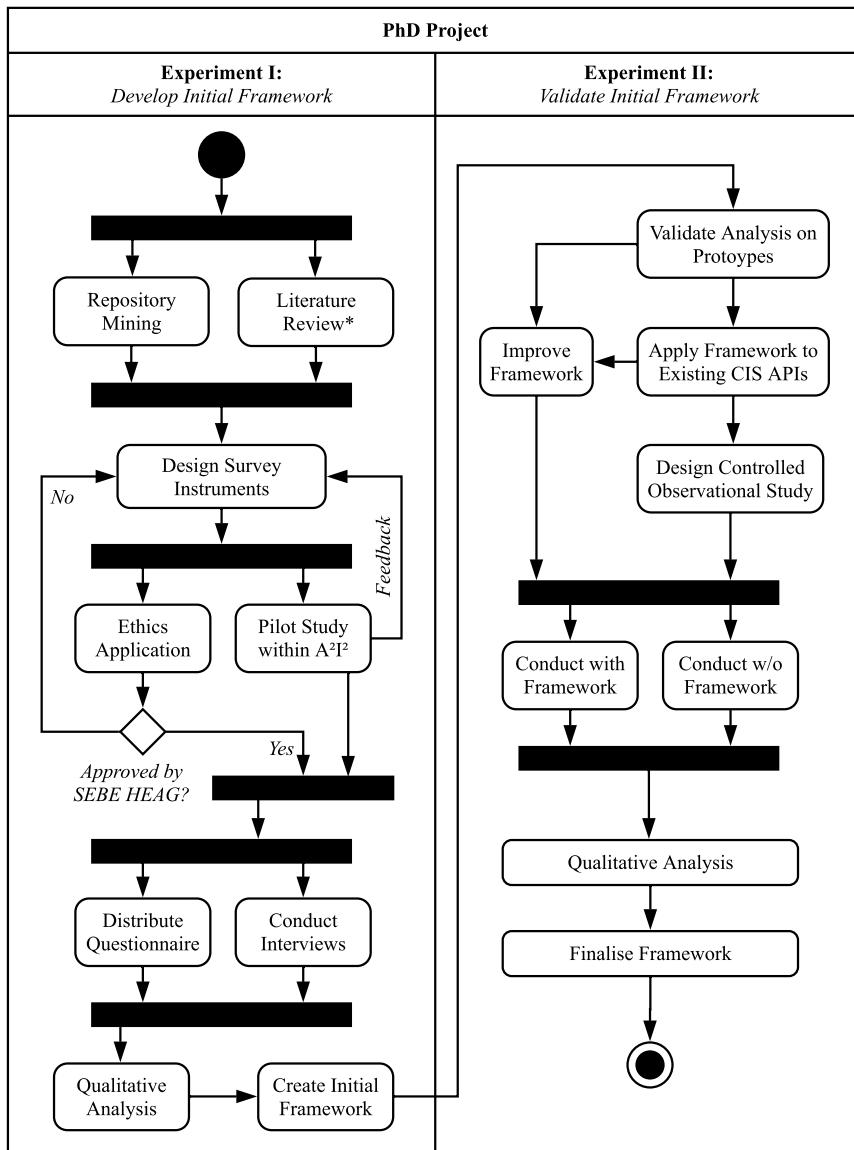
<sup>1195</sup> Singer et al. developed a taxonomy [170, 266] showcasing data collection techniques  
<sup>1196</sup> in field studies that are used in conjunction with a variety of methods based on the  
<sup>1197</sup> level of interaction between researcher and software engineer, if any. This taxonomy  
<sup>1198</sup> is reproduced in Figure A.4.

## <sup>1199</sup> 3.4 Proposed Experiments

<sup>1200</sup> This section discusses two proposed experiments that we conduct in this study.  
<sup>1201</sup> For each experiment, we describe an overview of the experiment grounded known  
<sup>1202</sup> methods and techniques (Sections 3.3.1 and 3.3.2), our approach to analysing the  
<sup>1203</sup> data, as well as linking the experiment back to a research hypothesis and question  
<sup>1204</sup> (Section 1.3). A high-level overview of the proposed experiments and the major  
<sup>1205</sup> bodies of work they encompass is presented in Figure 3.1.

### <sup>1206</sup> 3.4.1 Experiment I: Develop Initial Framework

<sup>1207</sup> Experiment I shapes a context-agnostic approach to understand current usage pat-  
<sup>1208</sup> terns of IWS APIs and the ways by which developers interpret them. Briefly, this  
<sup>1209</sup> experiment is comprised under two phases of field survey research: (i) repository  
<sup>1210</sup> mining developer discussion forums (i.e., analysis of databases and documentation  
<sup>1211</sup> analysis) to understand what developers currently complain about on these forums  
<sup>1212</sup> and where their mismatch in understanding lies; (ii) conducting unstructured inter-  
<sup>1213</sup> views and distributing a questionnaire to gather personal opinion based on individual  
<sup>1214</sup> developer's anecdotal remarks.



**Figure 3.1:** High-level activity diagram of the proposed experiments in this study. Literature review is ongoing.

### 1215 Relevance and Motivation

1216 Experiment I aims to better understand the existing mindsets that developers have  
 1217 when approaching to use computer vision services (CVSs). This work therefore ties in to RH1; by understanding the developer mindset in how they interpret CVS APIs,  
 1218 we are better informed to produce a framework that increases the effectiveness of  
 1219 the documentation of those existing CVS providers.

1220 RH1 postulates that the software engineering community do not fully understand  
 1221 the ‘magic’ behind IWS APIs. As described in Section 1.3, they face a gap in their

understanding around the underlying architecture of pre-built, machine learnt APIs (?REF?). Software developers are not well-supported by the IWS providers, and therefore do not have a consistent set of common best practices when approaching to use these APIs (?REF?). It is therefore necessary that IWS providers provide additional information to gap this mismatched understanding (?REF?).

## 1228 Data Collection & Analysis

1229 **Phase 1: Repository Mining** Developers typically congregate in search of dis-  
 1230 courses on issues they face in online forums, such as Stack Overflow (SO) and  
 1231 Quora, as well as writing their experiences in personal blogs such as Medium. The  
 1232 simplest of these platforms is SO (a sub-community of the Stack Exchange family  
 1233 of targeted communities) that specifically targets developer issues on using a simple  
 1234 Q&A interface, where developers can discuss technical aspects and general software  
 1235 development topics. Moreover, SO is often acknowledged as *the ‘go-to’ place* for  
 1236 developers to find high-quality code snippets that assist in their problems [275].

1237 Thus, to begin validating IWS API usage and misunderstanding in a generalised  
 1238 context (i.e., context-agnostic to the project at hand), we propose using repository  
 1239 mining on SO to help answer our research questions. Specifically, we select SO  
 1240 due to its targeted community of developers<sup>1</sup> and the availability of its publicly  
 1241 available dataset released as ‘data dumps’ on the Stack Exchange Data Explorer<sup>2</sup>  
 1242 and Google BigQuery<sup>3</sup>. Studies conducted have also used SO to mine developer  
 1243 discourse [7, 16, 20, 59, 174, 210, 218, 235, 247, 268, 281, 296].

1244 Due to the enormity of the data produced, we will use qualitative analysis on  
 1245 the questions mined using assistive tools such as NVivo. For this, we will conduct a  
 1246 thematic analysis on the themes of each question mined, the relevance of the question  
 1247 to our research topic, and ensuring strict coding schemes (that reflect our research  
 1248 goals) are adhered to. We refer to Singer et al. [266] and Schwandt [255] on coding  
 1249 and analysing this qualitative data gathered.

1250 **Phase 2: Personal Opinion Surveys** We follow the triangulation approach pro-  
 1251 posed by Mayring [186] to corroborate the qualitative data of developers’ discussion  
 1252 of SO with secondary survey research, thereby validating what people say on git  
 1253 with what is said and done in real life. Kitchenham and Pfleeger [154] provide an  
 1254 introduction on methods used to conduct personal opinion surveys which we adopt  
 1255 as an initial reference in (i) shaping our survey objectives around our research goals,  
 1256 (ii) designing a cross-sectional survey, (iii) developing and evaluating our two survey  
 1257 instruments (consisting of a structured questionnaire and semi-structured interview),  
 1258 (iv) evaluating our instruments, (v) obtaining the data and (vi) analysing the data.

<sup>1</sup>We 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.

<sup>2</sup><https://data.stackexchange.com/stackoverflow> last accessed 17 January 2017.

<sup>3</sup><https://console.cloud.google.com/marketplace/details/stack-exchange/stack-overflow> last accessed 17 January 2017.

1259 As is good practice in developing questionnaire instruments to evaluate their  
1260 reliability and validity [176], we evaluate our instrument design by asking colleagues  
1261 to critique it via pilot studies within A<sup>2</sup>I<sup>2</sup>. This assists in identifying any problems  
1262 with the questionnaire itself and with any issues that may occur with the response rate  
1263 and follow-up procedures. We follow a similar approach by practicing the interview  
1264 instrument on colleagues within A<sup>2</sup>I<sup>2</sup>.

1265 Findings from the pilot study helps inform us for a widely distributed question-  
1266 naire and conducting interviews out in the field, where we recruit external software  
1267 engineers in industry through the industry contacts of A<sup>2</sup>I<sup>2</sup>. Ethics approval from  
1268 the Faculty of Science, Engineering and Built Environment Human Ethics Advisory  
1269 Group (SEBE HEAG) will be required prior to externally conducting this survey  
1270 research (see Appendix C). The quantitative (survey) and qualitative (interview)  
1271 analysis allows us to shape the research outcome of RH1—an API documentation  
1272 quality assessment framework—and assists in stabilising our general understanding  
1273 of how developers use these existing APIs.

### 1274 **3.4.2 Developing the Initial Framework**

1275 Our initial framework is developed using the qualitative and quantitative analyses  
1276 from the findings of Experiment I. As this is a creative phase in which we are  
1277 developing a new framework, the exact process by which we develop the initial  
1278 framework will come to light once more insight is determined. However it is  
1279 anticipated discussion with other researchers and engineers at A<sup>2</sup>I<sup>2</sup> about the analyses  
1280 of the findings (i.e., white-boarding sessions of potential ideas from the findings)  
1281 will help develop our initial documentation framework. This framework will take  
1282 the shape of a checklist or table, typical of information systems studies (e.g., [167]),  
1283 that indicate what attributes should be best suited for what needs.

### 1284 **3.4.3 Experiment II: Validate Initial Framework**

1285 Experiment II extends the *generalised* context of Experiment I by evaluating how  
1286 the findings of Experiment I translates to context-specific applications. We confirm  
1287 that the generalised findings are (indeed) genuine by conducting action research in  
1288 combination with an observational study on software engineers. This experiment  
1289 is also compromised of: (i) development of prototypes using IWS APIs of dif-  
1290 fering contexts; (ii) presenting a solution framework to developers to interpret the  
1291 improvement of their understanding when using an IWS.

### 1292 **Relevance and Motivation**

1293 Experiment II aims to contextualise the findings from Experiment I; that is, if we  
1294 add *varying contexts* to the applications we write using IWS APIs, what is needed to  
1295 extend the *context-agnostic* framework developed in Experiment I? This work relates  
1296 back to RH2; adding context-specific metadata to the endpoints of these APIs, we  
1297 can highlight what issues exist when such metadata is not present (?REF?) and  
1298 what types of metadata developers seek (?REF?).

1299 Moreover, the implication of the first two hypotheses suggest that applying an API  
1300 documentation and metadata quality assessment framework may have an effect on  
1301 other aspects within the software engineering process (RH3). Thus, this experiment  
1302 also confirms if our framework makes an improvement to software quality, developer  
1303 productivity and/or developer informativeness (?REF? and ?REF?).

#### 1304 Data Collection & Analysis

1305 To confirm findings of the method within RH1 is genuine, we shift from reviewing  
1306 the documentation from a general stance to a specialised (context-specific) stance in  
1307 the use of these APIs.

1308 This is firstly achieved by using existing IWS APIs to develop basic ‘prototypes’,  
1309 each having differing contexts. The number of prototypes to develop and the use  
1310 cases they have will be informed by the results of Experiment I, and therefore cannot  
1311 yet be described at this stage. Our action research in developing the prototypes will  
1312 help inform any potential gaps that exist in the findings of RH1, especially with  
1313 regards to context-specificity, and therefore improves the metadata component of our  
1314 framework (as per the outcome of RH2).

1315 This outcome will also help us design the next stage of the experiment, con-  
1316 sisting of a comparative controlled study [258] to capture firsthand behaviours and  
1317 interactions toward how software engineers approach using an IWS with and with-  
1318 out our framework applied. We will provide improved documentation and metadata  
1319 responses of a set of popular CVSs that is documented with the additional metadata  
1320 and whose information is organised using our framework.

1321 We then recruit 20 developers of varying experience (from beginner programmer  
1322 to principal engineer) to complete five tasks under an observational, comparative  
1323 controlled study, 10 of which will (a) develop with the *new* framework, and the  
1324 other 10 will (b) develop with the *as-is/existing* documentation. From this, we  
1325 compare if the framework makes improvements by capturing metrics and recording  
1326 the observational sessions for qualitative analysis. We use visual modelling to  
1327 analyse the qualitative data using matrices [78], maps and networks [255] as these  
1328 help illustrate any causal, temporal or contextual relationships that may exist to map  
1329 out the developer’s mindset and difference in approaching the two sets of designs of  
1330 the same tasks.

### 1331 3.5 Empirical Validity

1332 In Section 3.2, we state that this study primarily adopts a critical theorist stance.  
1333 Critical theorists assess research quality by the utility of the knowledge gained [89].  
1334 Lau [167] established criteria on validating information systems research specifically  
1335 for action research unifying four dimensions of the research (conceptual foundation,  
1336 study design, research process, and role expectations) against 22 varying criteria.  
1337 We also partially adopt constructivism as we attempt to model the developer mindset  
1338 rich in qualitative data, to which eight strategies identified by Creswell [65] cover.

1339 We identify possible threats to internal- (study design), external- (generality of  
1340 results), and construct-validity (theoretical understanding) in the following sections,  
1341 and describe how we mitigate these threats.

### 1342 3.5.1 Threats to Internal Validity

#### 1343 Hawthorne Effect

1344 Observational field techniques involving participants often run a risk producing mis-  
1345 aligned results from laboratory versus environmental (practical) conditions. This is  
1346 commonly known as the Hawthorne effect [86, 243] and careful consideration of this  
1347 effect must be reflected when designing our controlled observation (Section 3.4.3).  
1348 We aim to carefully explain the purpose and protocol to research participants, en-  
1349 couraging them to act as much as possible as in their practical conditions. We also  
1350 encourage the ‘think-aloud’ to participants protocol to reinforce this. By highlight-  
1351 ing the Hawthorne effect to them, we anticipate that participants will be aware of the  
1352 condition, and therefore avoid doing things that do not reflect real-world action.

#### 1353 Misleading Statements in Interviews

1354 Similarly, threats to the interview survey instrument exist where participants do not  
1355 often report differences in behaviour from what they actually do in practice [266].  
1356 We anticipate that conducting interviews in a semi-structured manner may assist in  
1357 following up with unexpected statements (as opposed to structured interviews) and  
1358 additionally corroborate findings using Mayring’s concurrent triangulation method  
1359 [186] to verify potentially misleading statements from participants with question-  
1360naire results and observation findings.

#### 1361 Participant Observation Accuracy

1362 Conducting participant observations is a skill that requires training. While every  
1363 effort will be instilled to ensure all relevant observations are noted, it is impossible  
1364 for a single observer to note every possible interaction that occurs in all observations  
1365 made. Therefore, to validate the consistency of data collected, we may require  
1366 rater agreement exercises [144] and we will likely use a form of recording device  
1367 (with participant consent) to ensure all information is transcribed correctly in the  
1368 interview.

#### 1369 Unintended Interviewee Bias

1370 Interviewers should introduce the research by which participants are involved in by  
1371 describing an expiation of the research being conducted. However, the amount of  
1372 information described may impact the bias instilled on the interviewee. For example,  
1373 if the participant does not understand the goals of the study or feel that they are of  
1374 the ‘right target’, then it is likely that they may choose not to be involved in the  
1375 study or give misleading answers. On the other hand, if interviewees are told too  
1376 much information, then they may filter responses and leave out vital data that the

<sup>1377</sup> interviewer may be interested in. To mitigate this, varying levels of information will  
<sup>1378</sup> be ‘tested’ against colleagues to determine the right level of how much information  
<sup>1379</sup> is divulged at the beginning of the interview.

### <sup>1380</sup> **Poor Questionnaire Responses**

<sup>1381</sup> Unless significant inducements are offered, Singer et al. [266] report that a con-  
<sup>1382</sup> sistent response rate of 5% has been found in software engineering questionnaires  
<sup>1383</sup> distributed and in information systems the median response rates for surveys are 60%  
<sup>1384</sup> [21]. We observe that low response rates may adversely effect the findings of our  
<sup>1385</sup> survey, typically as software engineers find little time to do them. Tackling this is-  
<sup>1386</sup> sue can be resolved by carefully designing succinct, unambiguous and well-worded  
<sup>1387</sup> questions that we will verify against our colleagues and within the pilot study in  
<sup>1388</sup> A<sup>2</sup>I<sup>2</sup>, wherein any adjustments made from the pilot study due to unexpected poor  
<sup>1389</sup> quality of the questionnaire will be reported and explained. We also adopt research  
<sup>1390</sup> conducted in the field of questionnaire design, such as ensuring all scales are worded  
<sup>1391</sup> with labels [162] or using a summating rating scale [271] to address a specific topic  
<sup>1392</sup> of interest if people are to make mistakes in their response or answer in different  
<sup>1393</sup> ways at different times. Similar effects exist to that above where what occurs in  
<sup>1394</sup> reality is not what is reflected in our results; we refer to our concurrent triangulation  
<sup>1395</sup> approach to gap this risk.

### <sup>1396</sup> **3.5.2 Threats to External Validity**

#### <sup>1397</sup> **Representative Sample Size**

<sup>1398</sup> Our results must generalise by ensuring a representative subset of the target popu-  
<sup>1399</sup> lation is found. If results do not generalise, then all that is found is potentially of  
<sup>1400</sup> little more value than personal anecdote [154]. Therefore, designing a well-defined  
<sup>1401</sup> sample frame to determine which developers we wish to target is empirical. For this,  
<sup>1402</sup> we refer to Kitchenham and Pfleeger [154].

#### <sup>1403</sup> **Student Cohorts**

<sup>1404</sup> External validity is typically undermined when students are recruited in software en-  
<sup>1405</sup> gineering research, which is common practice [89]. Analytical argument is required  
<sup>1406</sup> to describe why results on students are reflective of results found on developers in  
<sup>1407</sup> industry. Therefore, we anticipate that—through industry contacts at A<sup>2</sup>I<sup>2</sup>—we will  
<sup>1408</sup> be able to contact developers in industry, thereby minimising our reliance to use  
<sup>1409</sup> students as participants.

#### <sup>1410</sup> **Concurrent Triangulation Strategy**

<sup>1411</sup> A drawback with the concurrent triangulation strategy is that multiple sources of  
<sup>1412</sup> data are concurrently collected within the same time. Collecting and analysing data  
<sup>1413</sup> *sequentially*, instead of concurrently, allows for time to analyse data between studies,

1414 thus allowing each analysis to adapt as more emerging results are explored. Easter-  
1415 brook et al. [89] states that the challenge in this approach is that it may be difficult  
1416 for researchers to compare results of multiple analyses or resolve contradictions that  
1417 begin to arise when this is performed concurrently. A mitigation strategy, should  
1418 this occur, would be to seek out further sources of evidence, or even re-conduct a  
1419 follow-up study if time permits.

1420 **3.5.3 Threats to Construct Validity**

1421 **Developer Informativeness**

1422 RH3 describes that if we improve the documentation of IWS APIs, then developers  
1423 are more informed/educated in what they do. This therefore increases their pro-  
1424 ductivity and the quality of the applications they build. However, the construct of  
1425 ‘informativeness’ is difficult to capture with standalone metrics, and using simple  
1426 quantitative metrics such as time taken to complete a task or lines of code to imple-  
1427 ment it may not reflect that a developer is more ‘informed’. Therefore, we propose  
1428 further investigation into understanding how to measure informativeness of software  
1429 engineers to ensure that this construct validity does not impact our results too greatly.



1430

## **Part II**

1431

# **Publications**



# CHAPTER 4

1432

1433

1434

## Identifying Evolution in Computer Vision Services<sup>†</sup>

1435

1436 **Abstract** Recent advances in artificial intelligence (AI) and machine learning (ML), such  
1437 as computer vision (CV), are now available as intelligent web services (IWSs) and their  
1438 accessibility and simplicity is compelling. Multiple vendors now offer this technology as  
1439 cloud services and developers want to leverage these advances to provide value to end-users.  
1440 However, there is no firm investigation into the maintenance and evolution risks arising from  
1441 use of these IWSs; in particular, their behavioural consistency and transparency of their  
1442 functionality. We evaluated the responses of three different IWSs (specifically CV) over 11  
1443 months using 3 different data sets, verifying responses against the respective documentation  
1444 and assessing evolution risk. We found that there are: (1) inconsistencies in how these  
1445 services behave; (2) evolution risk in the responses; and (3) a lack of clear communication  
1446 that documents these risks and inconsistencies. We propose a set of recommendations to  
1447 both developers and IWS providers to inform risk and assist maintainability.

### 4.1 Introduction

1448 The availability of intelligent web services (IWSs) has made artificial intelligence  
1449 (AI) tooling accessible to software developers and promises a lower entry barrier for  
1450 their utilisation. Consider state-of-the-art computer vision (CV) analysers, which  
1451 require either manually training a deep-learning classifier, or selecting a pre-trained  
1452 model and deploying these into an appropriate infrastructure. Either are laborious  
1453 in time, and require non-trivial expertise along with a large data set when training  
1454 or customisation is needed. In contrast, IWSs providing CV (i.e., computer vision  
1455 services or CVSs such as [318, 320, 321, 322, 329, 331, 333, 334, 335, 339, 342,

<sup>†</sup>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.

<sup>1457</sup> 343, 376, 377]) abstract these complexities behind a web application programming  
<sup>1458</sup> interface (API) call. This removes the need to understand the complexities required  
<sup>1459</sup> of machine learning (ML), and requires little more than the knowledge on how to  
<sup>1460</sup> use RESTful endpoints. The ubiquity of these services is exemplified through their  
<sup>1461</sup> rapid uptake in applications such as aiding the vision-impaired [74, 236].

<sup>1462</sup> While IWSs have seen quick adoption in industry, there has been little work  
<sup>1463</sup> that has considered the software quality perspective of the risks and impacts posed  
<sup>1464</sup> by using such services. In relation to this, there are three main challenges: (1)  
<sup>1465</sup> incorporating stochastic algorithms into software that has traditionally been deter-  
<sup>1466</sup> ministic; (2) the general lack of transparency associated with the ML models; and  
<sup>1467</sup> (3) communicating to application developers.

<sup>1468</sup> ML typically involves use of statistical techniques that yield components with  
<sup>1469</sup> a non-deterministic external behaviour; that is, for the same given input, different  
<sup>1470</sup> outcomes may result. However, developers, in general, are used to libraries and small  
<sup>1471</sup> components behaving predictably, while systems that rely on ML techniques work  
<sup>1472</sup> on confidence intervals<sup>1</sup> and probabilities. For example, the developer’s mindset  
<sup>1473</sup> suggests that an image of a border collie—if sent to three intelligent computer vision  
<sup>1474</sup> services (CVSs)—would return the label ‘dog’ consistently with time regardless  
<sup>1475</sup> of which service is used. However, one service may yield the specific dog breed,  
<sup>1476</sup> ‘border collie’, another service may yield a permutation of that breed, ‘collie’, and  
<sup>1477</sup> another may yield broader results, such as ‘animal’; each with results of varying  
<sup>1478</sup> confidence values.<sup>2</sup> Furthermore, the third service may evolve with time, and  
<sup>1479</sup> thus learn that the ‘animal’ is actually a ‘dog’ or even a ‘collie’. The outcomes  
<sup>1480</sup> are thus behaviourally inconsistent between services providing conceptually similar  
<sup>1481</sup> functionality. As a thought exercise, consider if the sub-string function were created  
<sup>1482</sup> using ML techniques—it would perform its operation with a confidence where the  
<sup>1483</sup> expected outcome and the AI inferred output match as a *probability*, rather than a  
<sup>1484</sup> deterministic (constant) outcome. How would this affect the developers’ approach  
<sup>1485</sup> to using such a function? Would they actively take into consideration the non-  
<sup>1486</sup> deterministic nature of the result?

<sup>1487</sup> Myriad software quality models and software engineering (SE) practices advo-  
<sup>1488</sup> cate maintainability and reliability as primary characteristics; stability, testability,  
<sup>1489</sup> fault tolerance, changeability and maturity are all concerns for quality in software  
<sup>1490</sup> components [127, 231, 270] and one must factor these in with consideration to  
<sup>1491</sup> software evolution challenges [113, 114, 193, 194, 284]. However, the effect this  
<sup>1492</sup> non-deterministic behaviour has on quality when masked behind an IWS is still  
<sup>1493</sup> under-explored to date in SE literature, to our knowledge. Where software depends  
<sup>1494</sup> on IWSs to achieve functionality, these quality characteristics may not be achieved,  
<sup>1495</sup> and developers need to be wary of the unintended side effects and inconsistency that  
<sup>1496</sup> exists when using non-deterministic components. A CVS may encapsulate deep-  
<sup>1497</sup> learning strategies or stochastic methods to perform image analysis, but developers

---

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

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

1498 are more likely to approach IWSs with a mindset that anticipates consistency. Al-  
1499 though the documentation does hint at this non-deterministic behaviour (i.e., the  
1500 descriptions of ‘confidence’ in various CVSs suggest they are not always confi-  
1501 dent, and thus not deterministic [316, 330, 340]), the integration mechanisms offered  
1502 by popular vendors do not seem to fully expose the nuances, and developers are not  
1503 yet familiar with the trade-offs.

1504 Do popular CVSs, as they currently stand, offer consistent behaviour, and if not,  
1505 how is this conveyed to developers (if it is at all)? If CVSs are to be used in production  
1506 services, do they ensure quality under rigorous service quality assurance (SQA)  
1507 frameworks [127]? What evolution risk [113, 114, 193, 194] do they pose if these  
1508 services change? To our knowledge, few studies have been conducted to investigate  
1509 these claims. This paper assesses the consistency, evolution risk and consequent  
1510 maintenance issues that may arise when developers use IWSs. We introduce a  
1511 motivating example in Section 4.2, discussing related work and our methodology  
1512 in Sections 4.3 and 4.4. We present and interpret our findings in Section 4.5. We  
1513 argue with quantified evidence that these IWSs can only be considered with a mature  
1514 appreciation of risks, and we make a set of recommendations in Section 4.6.

## 1515 4.2 Motivating Example

1516 Consider Rosa, a software developer, who wants to develop a social media photo-  
1517 sharing mobile app that analyses her and her friends photos on Android and iOS.  
1518 Rosa wants the app to categorise photos into scenes (e.g., day vs. night, outdoors  
1519 vs. indoors), generate brief descriptions of each photo, and catalogue photos of her  
1520 friends as well as common objects (e.g., all photos with a dog, all photos on the  
1521 beach).

1522 Rather than building a CV engine from scratch, Rosa thinks she can achieve this  
1523 using one of the popular CVSs (e.g., [318, 320, 321, 322, 329, 331, 333, 334, 335,  
1524 339, 342, 343, 376, 377]). However, Rosa comes from a typical software engineering  
1525 background with limited knowledge of the underlying deep-learning techniques  
1526 and implementations as currently used in CV. Not unexpectedly, she internalises a  
1527 mindset of how such services work and behave based on her experience of using  
1528 software libraries offered by various SDKs. This mindset assumes that different  
1529 cloud vendor image processing APIs more-or-less provide similar functionality,  
1530 with only minor variations. For example, cloud object storage for Amazon S3 is  
1531 both conceptually and behaviourally very similar to that of Google Cloud Storage  
1532 or Azure Storage. Rosa assumes the CVSs of these platforms will, therefore, likely  
1533 be very similar. Similarly, consider the string libraries Rosa will use for the app.  
1534 The conceptual and behavioural similarities are consistent; a string library in Java  
1535 (Android) is conceptually very similar to the string library she will use in Swift  
1536 (iOS), and likewise both behave similarly by providing the same results for their  
1537 respective sub-string functionality. However, **unlike the cloud storage and string**  
1538 **libraries, different CVSs often present conceptually similar functionality but**  
1539 **are behaviourally very different.** IWS vendors also hide the depth of knowledge  
1540 needed to use these effectively—for instance, the training data set and ontologies

1541 used to create these services are hidden in the documentation. Thus, Rosa isn't even  
1542 exposed to this knowledge as she reads through the documentation of the providers  
1543 and, thus, Rosa makes the following assumptions:

- 1544 • **"I think the responses will be consistent amongst these CVSs."** When Rosa  
1545 uploads a photo of a dog, she would expect them all to respond with 'dog'. If  
1546 Rosa decides to switch which service she is using, she expects the ontologies  
1547 to be compatible (all CVSs *surely* return dog for the same image) and therefore  
1548 she can expect to plug-in a different service should she feel like it making only  
1549 minor code modifications such as which endpoints she is relying on.
- 1550 • **"I think the responses will be constant with time."** When Rosa uploads the  
1551 photo of a dog for testing, she expects the response to be the same in 10 weeks  
1552 time once her app is in production. Hence, in 10 weeks, the same photo of the  
1553 dog should return the same label.

### 1554 4.3 Related Work

1555 If we were to view CVSs through the lenses of an SQA framework, robustness,  
1556 consistency, and maintainability often feature as quality attributes in myriad soft-  
1557 ware quality models (e.g., [135]). Software quality is determined from two key  
1558 dimensions: (1) in the evaluation of the end-product (external quality) and (2) the  
1559 assurances in the development processes (internal quality) [231]. We discuss both  
1560 perspectives of quality within the context of our work in this section.

#### 1561 4.3.1 External Quality

1562 **Robustness for safety-critical applications** A typical focus of recent work has  
1563 been to investigate the robustness of deep-learning within CV technique imple-  
1564 mentation, thereby informing the effectiveness in the context of the end-product.  
1565 The common method for this has been via the use of adversarial examples [278],  
1566 where input images are slightly perturbed to maximise prediction error but are still  
1567 interpretable to humans.

1568 Google Cloud Vision, for instance, fails to correctly classify adversarial examples  
1569 when noise is added to the original images [128]. Rosenfeld et al. [249] illustrated  
1570 that inserting synthetic foreign objects to input images (e.g., a cartoon elephant)  
1571 can completely alter classification output. Wang et al. [294] performed similar  
1572 attacks on a transfer-learning approach of facial recognition by modifying pixels of  
1573 a celebrity's face to be recognised as a completely different celebrity, all while still  
1574 retaining the same human-interpretable original celebrity. Su et al. [273] used the  
1575 ImageNet database to show that 41.22% of images drop in confidence when just a  
1576 *single pixel* is changed in the input image; and similarly, Eykholt et al. [91] recently  
1577 showed similar results that made a convolutional neural network (CNN) interpret a  
1578 stop road-sign (with mimicked graffiti) as a 45mph speed limit sign.

1579 The results suggest that current state-of-the-art CV techniques may not be robust  
1580 enough for safety critical applications as they do not handle intentional or unin-  
1581tentional adversarial attacks. Moreover, as such adversarial examples exist in the

1582 physical world [91, 163], “the natural world may be adversarial enough” [227] to  
1583 fool AI software. Though some limitations and guidelines have been explored in this  
1584 area, the perspective of *Intelligent Web Services* is yet to be considered and specific  
1585 guidelines do not yet exist when using CVSs.

1586 **Testing strategies in ML applications** Although much work applies ML tech-  
1587 niques to automate testing strategies, there is only a growing emphasis that considers  
1588 this in the opposite sense; that is, testing to ensure the ML product works correctly.  
1589 There are few reliable test oracles that ensure if an ML has been implemented to  
1590 serve its algorithm and use case purposefully; indeed, “the non-deterministic nature  
1591 of many training algorithms makes testing of models even more challenging” [11].  
1592 Murphy et al. [201] proposed a SE-based testing approach on ML ranking algorithms  
1593 to evaluate the ‘correctness’ of the implementation on a real-world data set and prob-  
1594 lem domain, whereby discrepancies were found from the formal mathematical proofs  
1595 of the ML algorithm and the implementation.

1596 Recently, Braiek and Khomh [40] conducted a comprehensive review of testing  
1597 strategies in ML software, proposing several research directions and recommenda-  
1598 tions in how best to apply SE testing practices in ML programs. However, much  
1599 of the area of this work specifically targets ML engineers, and not application de-  
1600 velopers. Little has been investigated on how application developers perceive and  
1601 understand ML concepts, given a lack of formal training; we note that other testing  
1602 strategies and frameworks proposed (e.g., [44, 200, 209]) are targeted chiefly to the  
1603 ML engineer, and not the application developer.

1604 However, Arpteg et al. [11] recently demonstrated (using real-world ML projects)  
1605 the developmental challenges posed to developers, particularly those that arise when  
1606 there is a lack of transparency on the models used and how to troubleshoot ML  
1607 frameworks using traditional SE debugging tools. This said, there is no further in-  
1608 vestigations into challenges when using the higher, ‘ML friendly’ layers (e.g., IWSs)  
1609 of the ‘machine learning spectrum’ [216], rather than the ‘lower layers’ consisting  
1610 of existing ML frameworks and algorithms targeted toward the ML community.

### 1611 4.3.2 Internal Quality

1612 **Quality metrics for cloud services** CVSs are based on cloud computing funda-  
1613 mentals under a subset of the Platform as a Service (PaaS) model. There has been  
1614 work in the evaluation of PaaS in terms of quality attributes [104]: these attributes  
1615 are exposed using service-level agreements (SLAs) between vendors and customers,  
1616 and customers denote their demanded quality of service (QoS) to ensure the cloud  
1617 services adhere to measurable KPI attributes.

1618 Although, popular services, such as cloud object storage, come with strong QoS  
1619 agreement, to date IWSs do not come with deep assurances around their performance  
1620 and responses, but do offer uptime guarantees. For example, how can Rosa demand  
1621 a QoS that ensures all photos of dogs uploaded to her app guarantee the specific dog  
1622 breeds are returned so that users can look up their other friend’s ‘border collie’s?  
1623 If dog breeds are returned, what ontologies exist for breeds? Are they consistent

1624 with each other, or shortened? ('Collie' versus 'border collie'; 'staffy' versus  
1625 'staffordshire bull terrier'?). For some applications, these unstated QoS metrics  
1626 specific to the ML service may have significant legal ramifications.

1627 **Web service documentation and documenting ML** From the *developer's* per-  
1628 spective, little has been achieved to assess IWS quality or assure quality of these  
1629 CVSSs. web service and their APIs are the bridge between developers' needs and  
1630 the software components [10]; therefore, assessing such CVSSs from the quality  
1631 of their APIs is thereby directly related to the development quality [158]. Good  
1632 APIs should be intuitive and require less documentation browsing [229], thereby  
1633 increasing productivity. Conversely, poor APIs that are hard to understand and work  
1634 with reduce developer productivity, thereby reducing product quality. This typically  
1635 leads to developers congregating on forums such as Stack Overflow, leading to a  
1636 repository of unstructured knowledge likely to concern API design [298]. The con-  
1637 sequences of addressing these concerns in development leads to a higher demand  
1638 in technical support (as measured in [124]) that, ultimately, causes the maintenance  
1639 to be far more expensive, a phenomenon widely known in software engineering  
1640 economics [37]. Rosa, for instance, isn't aware of technical ML concepts; if she  
1641 cannot reason about what search results are relevant when browsing the service and  
1642 understanding functionality, her productivity is significantly decreased. Conceptual  
1643 understanding is critical for using APIs, as demonstrated by Ko and Riche, and the  
1644 effects of maintenance this may have in the future of her application is unknown.

1645 Recent attempts to document attributes and characteristics on ML models have  
1646 been proposed. Model cards were introduced by Mitchell et al. [198] to describe how  
1647 particular models were trained and benchmarked, thereby assisting users to reason  
1648 if the model is right for their purposes and if it can achieve its stated outcomes.  
1649 Gebru et al. [108] also proposed datasheets, a standardised documentation format to  
1650 describe the need for a particular data set, the information contained within it and  
1651 what scenarios it should be used for, including legal or ethical concerns.

1652 However, while target audiences for these documents may be of a more technical  
1653 AI level (i.e., the ML engineer), there is still no standardised communication format  
1654 for application developers to reason about using particular IWSs, and the ramifica-  
1655 tions this may have on the applications they write is not fully conveyed. Hence, our  
1656 work is focused on the application developer perspective.

#### 1657 4.4 Method

1658 This study organically evolved by observing phenomena surrounding CVSSs by as-  
1659 sessing both their documentation and responses. We adopted a mixed methods  
1660 approach, performing both qualitative and quantitative data collection on these two  
1661 key aspects by using documentary research methods for inspecting the documen-  
1662 tation and structured observations to quantitatively analyse the results over time.  
1663 This, ultimately, helped us shape the following research hypotheses which this paper  
1664 addresses:

1665 [RH1] CVSs do not respond with consistent outputs between services, given the  
1666 same input image.

1667 [RH2] The responses from CVSs are non-deterministic and evolving, and the same  
1668 service can change its top-most response over time given the same input  
1669 image.

1670 [RH3] CVSs do not effectively communicate this evolution and instability, intro-  
1671 ducing risk into engineering these systems.

1672 We conducted two experiments to address these hypotheses against three popular  
1673 CVSs: AWS Rekognition [318], Google Cloud Vision [331], Azure Computer  
1674 Vision [339]. Specifically, we targeted the AWS DetectLabels endpoint [316],  
1675 the Google Cloud Vision annotate:images endpoint [330] and Azure's analyze  
1676 endpoint [340]. For the remainder of this paper, we de-identify our selected CVSs  
1677 by labelling them as services A, B and C but do not reveal mapping to prevent  
1678 any implicit bias. Our selection criteria for using these particular three services  
1679 are based on the weight behind each service provider given their prominence in  
1680 the industry (Amazon, Google and Microsoft), the ubiquity of their hosting cloud  
1681 platforms as industry leaders of cloud computing (i.e., AWS, Google Cloud and  
1682 Azure), being in the top three most adopted cloud vendors in enterprise applications  
1683 in 2018 [241] and the consistent popularity of discussion amongst developers in  
1684 developer communities such as Stack Overflow. While we choose these particular  
1685 cloud CVSs, we acknowledge that similar services [321, 322, 329, 334, 335, 376, 377]  
1686 also exist, including other popular services used in Asia [320, 333, 342, 343] (some  
1687 offering 3D image analysis [319]). We reflect on the impacts this has to our study  
1688 design in Section 4.7.

1689 Our study involved an 11-month longitudinal study which consisted of two 13  
1690 week and 17 week experiments from April to August 2018 and November 2018 to  
1691 March 2019, respectively. Our investigation into documentation occurred on August  
1692 28 2018. In total, we assessed the services with three data sets; we first ran a pilot  
1693 study using a smaller pool of 30 images to confirm the end-points remain stable,  
1694 re-running the study with a larger pool of images of 1,650 and 5,000 images. Our  
1695 selection criteria for these three data sets were that the images had to have varying  
1696 objects, taken in various scenes and various times. Images also needed to contain  
1697 disparate objects. Our small data set was sourced by the first author by taking photos  
1698 of random scenes in an afternoon, whilst our second data set was sourced from  
1699 various members of our research group from their personal photo libraries. We also  
1700 wanted to include a data set that was publicly available prior to running our study,  
1701 so for this data set we chose the COCO 2017 validation data set [173]. We have  
1702 made our other two data sets available online ([324]). We collected results and their  
1703 responses from each service's API endpoint using a python script [328] that sent  
1704 requests to each service periodically via cron jobs. Table 4.1 summarises various  
1705 characteristics about the data sets used in these experiments.

1706 We then performed quantitative analyses on each response's labels, ensuring all  
1707 labels were lowercased as case changed for services A and C over the evaluation  
1708 period. To derive at the consistency of responses for each image, we considered only  
1709 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

1710 over all images in data set  $D$  where  $i \in D$  and over the three services, the top labels  
 1711 per image ( $T_i$ ) of all labels per image  $L_i$  (i.e.,  $T_i \subseteq L_i$ ) is that where the respective  
 1712 label's confidences are consistently the highest of all labels returned. Typically, the  
 1713 top labels returned is a set containing only one element—that is, only one unique  
 1714 label consistently returned with the highest label ( $|T_i| = 1$ )—however there are cases  
 1715 where the top labels contains multiple elements as their respective confidences are  
 1716 *equal* ( $|T_i| > 1$ ).

1717 We measure response consistency under 6 aspects:

- 1718 (1) **Consistency of the top label between each service.** Where the same image of,  
 1719 for example, a dog is sent to the three services, the top label for service A may  
 1720 be ‘animal’, B ‘canine’ and C ‘animal’. Therefore, service B is inconsistent.
- 1721 (2) **Semantic consistency of the top labels.** Where a service has returned multi-  
 1722 ple top labels ( $|T_i| > 1$ ), there may lie semantic differences in what the service  
 1723 thinks the image best represents. Therefore, there is conceptual inconsistency  
 1724 in the top labels for a service even when the confidences are equal.
- 1725 (3) **Consistency of the top label’s confidence per service.** The top label for  
 1726 an image does not guarantee a high confidence. Therefore, there may be  
 1727 inconsistencies in how confident the top labels for all images in a service is.
- 1728 (4) **Consistency of confidence in the intersecting top label between each ser-  
 1729 vice.** The spread of a top intersecting label, e.g., ‘cat’, may not have the same  
 1730 confidences per service even when all three services agree that ‘cat’ is the top  
 1731 label. Therefore, there is inconsistency in the confidences of a top label even  
 1732 where all three services agree.
- 1733 (5) **Consistency of the top label over time.** Given an image, the top label in one  
 1734 week may differ from the top label the following week. Therefore, there is  
 1735 inconsistency in the top label itself due to model evolution.
- 1736 (6) **Consistency of the top label’s confidence over time.** The top label of an  
 1737 image may remain static from one week to the next for the same service, but  
 1738 its confidence values may change with time. Therefore, there is inconsistency  
 1739 in the top label’s confidence due to model evolution.

1740 For the above aspects of consistency, we calculated the spread of variation for the  
 1741 top label’s confidences of each service for every 1 percent point; that is, the frequency  
 1742 of top label confidences within 100–99%, 99–98% etc. The consistency of top label’s  
 1743 and their confidences between each service was determined by intersecting the labels  
 1744 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	$\mu$	$\sigma$
$A \cap B \cap C$	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

1745 This allowed us to determine relevant probability distributions. For reproducibility,  
1746 all quantitative analysis is available online [325].

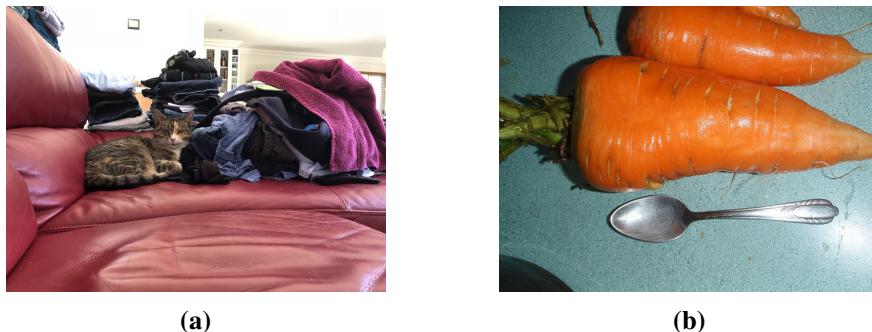
## 1747 4.5 Findings

### 1748 4.5.1 Consistency of top labels

1749 **Consistency across services** Table 4.2 presents the consistency of the top labels  
1750 between data sets, as measured by the cardinality of the intersection of all three ser-  
1751 vices’ set of top labels divided by the number of images per data set. A combina-  
1752 tion of services present varied overlaps in their top labels; services A and C provide the  
1753 best overlap for all three data sets, however the intersection of all three irrespective  
1754 of data sets is low.

1755 The implication here is that, without semantic comparison (see Section 4.7),  
1756 service vendors are not ‘plug-and-play’. If Rosa uploaded the sample images in  
1757 this paper to her application to all services, she would find that only Figure 4.1  
1758 responds with ‘person’ for services B and C in their respective set of top labels.  
1759 However, if she decides to then adopt service A, then Figure 4.1’s top label becomes  
1760 ‘conversation’; the ‘person’ label does not appear within the top 15 labels for service  
1761 A and, conversely, the ‘conversation’ label does not appear in the other services top  
1762 15.

1763 Should she decide if the performance of a particular service isn’t to her needs,  
1764 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.

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

1796 Thus, a taxonomy of ontologies is unknown; if a ‘border collie’ is detected in  
1797 an image, does this imply the hypernym ‘dog’ is detected, and then ‘mammal’, then  
1798 ‘animal’, then ‘object’? Only service B documents a taxonomy for capturing what  
1799 level of scope is desired, providing what it calls the ‘86-category’ concept as found  
1800 in its how-to guide:

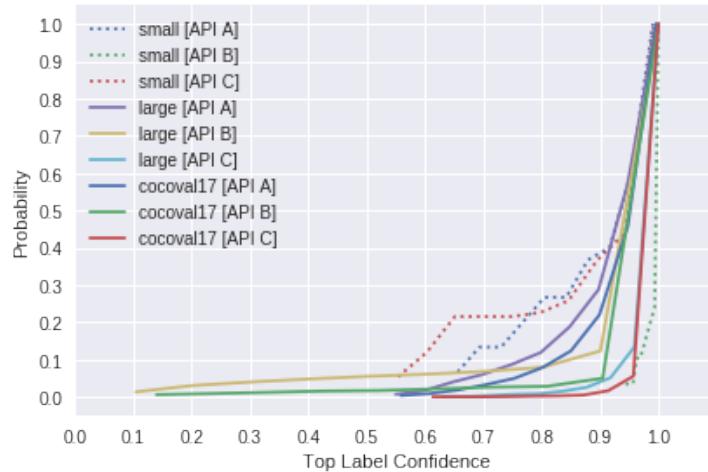
**1801** “Identify and categorize an entire image, using a category taxonomy with parent/child hereditary hierarchies. Categories can be used alone, or with our new tagging models.” [341]

<sup>1804</sup> Thus, even if Rosa implemented conceptual similarity analysis for the image, the  
<sup>1805</sup> top label set may not provide sufficient information to derive at a conclusive answer,  
<sup>1806</sup> and if simply relying on only one label in this set, information such as the duplicity  
<sup>1807</sup> of objects (e.g., ‘several’ in Figure 4.3b) may be missed.

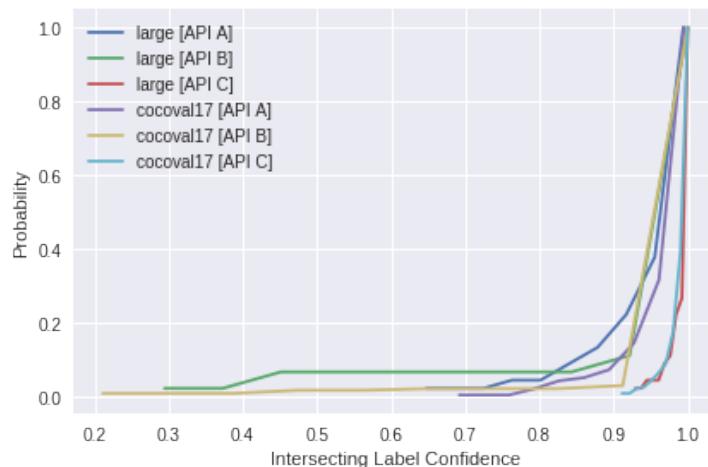
## 4.5.2 Consistency of confidence

**1809 Consistency of top label's confidence** In Figure 4.4, we see that there is high  
1810 probability that top labels have high confidences for all services. In summary, one in  
1811 nine images uploaded to any service will return a top label confident to at least 97%.  
1812 However, there is higher probability for service A returning a lower confidence,  
1813 followed by B. The best performing service is C, with 90% of requests having a top  
1814 label confident to  $\gtrapprox 95\%$ , when compared to  $\gtrapprox 87\%$  and  $\gtrapprox 93\%$  for services A and  
1815 B, respectively.

1816 Therefore, Rosa could generally expect that the top labels she receives in her  
1817 images do have high confidence. That is, each service will return a top label that  
1818 they are confident about. This result is expected, considering that the ‘top’ label  
1819 is measured by the highest confidence, though it is interesting to note that some  
1820 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( $\delta_c$ )	Range( $\delta_c$ )
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%.

1821 **Consistency of intersecting top label’s confidence** Even where all three services  
1822 do agree on a set of top labels, the disparity of how much they agree by is still of  
1823 importance. Just because three services agree that an image contains consistent top  
1824 labels, they do not always have a small spread of confidence. In Figure 4.6, the three  
1825 services agree with  $\sigma = 0.277$ , significantly larger than that of all images in general  
1826  $\sigma = 0.0831$ . Figure 4.5 displays the cumulative distribution of all intersecting top  
1827 labels’ confidence values, presenting slightly similar results to that of Figure 4.4.

### 1828 4.5.3 Evolution risk

1829 **Label Stability** Generally, the top label(s) did not evolve in the evaluation period.  
1830 16.19% and 5.85% of images did change their top label(s) in the Large and COCO-  
1831 Val17 data sets in service A. Thus, top labels are stable but not guaranteed to be  
1832 constant.

1833 **Confidence Stability** Similarly, where the top label(s) remained the same from  
1834 one interval to the next, the confidence values were stable. Table 4.3 displays the  
1835 proportion of images that changed their top label’s confidence values with various  
1836 statistics on the confidence deltas between snapshots ( $\delta_c$ ). However, this delta is so  
1837 minuscule that we attribute such changes to statistical noise.

## 1838 4.6 Recommendations

### 1839 4.6.1 Recommendations for IWS users

1840 **Test with a representative ontology for the particular use case** Rosa should  
1841 ensure that in her testing strategies for the app she develops, there is an ontology  
1842 focus for the types of vocabulary that are returned. Additionally, we noted that there  
1843 was a sudden change in case for services A and C; for all comparative purposes of  
1844 labels, each label should be lower-cased.

1845 **Incorporate a specialised IWS testing methodology into the development life-  
1846 cycle** Rosa can utilise the different aspects of consistency as outlined in this paper  
1847 as part of her quality strategy. To ensure results are correct over time, we recom-  
1848 mend developers create a representative data set of the intended application's data  
1849 set and evaluate these changes against their chosen service frequently. This will  
1850 help identify when changes, if any, have occurred if vendors do not provide a line of  
1851 communication when this occurs.

1852 **IWSs are not ‘plug-and-play’** Rosa will be locked into whichever vendor she  
1853 chooses as there is inherent inconsistency between these services in both the vocab-  
1854 ularly and ontologies that they use. We have demonstrated that very few services  
1855 overlap in their vocabularies, chiefly because they are still in early development and  
1856 there is yet to be an established, standardised vocabulary that can be shared amongst  
1857 the different vendors. Issues such as those shown in Section 4.5.1 can therefore be  
1858 avoided.

1859 Throughout this work, we observed that the terminologies used by the vari-  
1860 ous vendors are different. Documentation was studied, and we note that there is  
1861 inconsistency between the ways techniques are described to users. We note the  
1862 disparity between the terms ‘detection’, ‘recognition’, ‘localisation’ and ‘analysis’.  
1863 This applies chiefly to object- and facial-related techniques. Detection applies to  
1864 facial detection, which gives bounding box coordinates around all faces in an image.  
1865 Similarly, localisation applies the same methodology to disparate objects in an image  
1866 and labels them. In the context of facial ‘recognition’, this term implies that a face  
1867 is *recognised* against a known set of faces. Lastly, ‘analysis’ applies in the context  
1868 of facial analysis (gender, eye colour, expression etc.); there does not exist a similar  
1869 analysis technique on objects.

1870 We notice similar patterns with object ‘tagging’, ‘detection’ and ‘labelling’.  
1871 Service A uses ‘Entity Detection’ for object categorisation, service B uses ‘Image  
1872 Tagging’, and service C uses the term ‘Detect Labels’: conceptually, these provide  
1873 the same functionality but the lack of consistency used between all three providers is  
1874 concerning and leaves room for confusion with developers during any comparative  
1875 analyses. Rosa may find that she wants to label her images into day/night scenes, but  
1876 this in turn means the ‘labelling’ of varying objects. There is therefore no consistent  
1877 standards to use the same terminology for the same concepts, as there are in other  
1878 developer areas (such as Web Development).

1879 **Avoid use in safety-critical systems** We have demonstrated in this paper that both  
1880 labels and confidences are stable but not constant; there is still an evolution risk posed  
1881 to developers that may cause unknown consequences in applications dependent on  
1882 these CVSs. Developers should avoid their use in safety critical systems due to the  
1883 lack of visible changes.

#### 1884 4.6.2 Recommendations for IWS providers

1885 **Improve the documentation** Rosa does not know that service A returns back  
1886 ‘carrot’ for its top response, with service C returning ‘spoon’ (Figure 4.2b). She  
1887 is unable to tell the service’s API where to focus on the image. Moreover, how  
1888 can she toggle the level of specificity in her results? She is frustrated that service  
1889 C can detect ‘chocolate’, ‘food’ and also ‘beverage’ all as the same top label in  
1890 Figure 4.3a: what label is she to choose when the service is meant to do so for her,  
1891 and how does she get around this? Thus, we recommend vendors to improve the  
1892 documentation of services by making known the boundary set of the training data  
1893 used for the algorithms. By making such information publicly available, developers  
1894 would be able to review the service’s specificity for their intended use case (e.g.,  
1895 maybe Rosa is satisfied her app can catalogue ‘food’ together, and in fact does not  
1896 want specific types of foods (‘hot chocolate’) catalogued). We also recommend that  
1897 vendors publish usage guidelines should that include details of priors and how to  
1898 evaluate the specific service results.

1899 Furthermore, we did not observe that the vendors documented how some images  
1900 may respond with multiple labels of the exact same confidence value. It is not clear  
1901 from the documentation that response objects can have duplicate top values, and  
1902 tutorials and examples provided by the vendors do not consider this possibility. It  
1903 is therefore left to the developer to decide which label from this top set of labels  
1904 best suits for their particular use case; the documentation should describe that a rule  
1905 engine may need to be added in the developer’s application to verify responses. The  
1906 implications this would have on maintenance would be significant.

1907 **Improve versioning** We recommend introducing a versioning system so that a  
1908 model can be used from a specific date in production systems: when Rosa tests her  
1909 app today, she would like the service to remain *static* the same for when her app is  
1910 deployed in production tomorrow. Thus, in a request made to the vendor, Rosa could  
1911 specify what date she ran her app’s QA testing on so that she knows that henceforth  
1912 these model changes will not affect her app.

1913 **Improve Metadata in Response** Much of the information in these services is  
1914 reduced to a single confidence value within the response object, and the details about  
1915 training data and the internal AI architecture remains unknown; little metadata is  
1916 provided back to developers that encompass such detail. Early work into model  
1917 cards and datasheets [108, 198] suggests more can be done to document attributes  
1918 about ML systems, however at a minimum from our work, we recommend including  
1919 a reference point via the form of an additional identifier. This identifier must

1920 also permit the developers to submit the identifier to another API endpoint should  
1921 the developer wish to find further characteristics about the AI empowering the IWS,  
1922 reinforcing the need for those presented in model cards and datasheets. For example,  
1923 if Rosa sends this identifier she receives in the response object to the IWS descriptor  
1924 API, she could find out additional information such as the version number or date  
1925 when the model was trained, thereby resolving potential evolution risk, and/or the  
1926 ontology of labels.

1927 **Apply constraints for predictions on all inputs** In this study, we used some  
1928 images with intentionally disparate, and noisy objects. If services are not fully  
1929 confident in the responses they give back, a form of customised error message  
1930 should be returned. For example, if Rosa uploads an image of 10 various objects  
1931 on a table, rather than returning a list of top labels with varying confidences, it  
1932 may be best to return a ‘too many objects’ exception. Similarly, if Rosa uploads a  
1933 photo that the model has had no priors on, it might be useful to return an ‘unknown  
1934 object’ exception than to return a label it has no confidence of. We do however  
1935 acknowledge that current state of the art CV techniques may have limits in what they  
1936 can and cannot detect, but this limitation can be exposed in the documentation to the  
1937 developers.

1938 A further example is sending a one pixel image to the service, analogous to  
1939 sending an empty file. When we uploaded a single pixel white image to service A,  
1940 we received responses such as ‘microwave oven’, ‘text’, ‘sky’, ‘white’ and ‘black’  
1941 with confidences ranging from 51–95%. Prior checks should be performed on all  
1942 input data, returning an ‘insufficient information’ error where any input data is below  
1943 the information of its training data.

## 1944 4.7 Threats to Validity

### 1945 4.7.1 Internal Validity

1946 Not all CVSs were assessed. As suggested in Section 4.4, we note that there are  
1947 other CVSs such as IBM Watson. Many services from Asia were also not considered  
1948 due to language barriers (of the authors) in assessing these services. We limited our  
1949 study to the most popular three providers (outside of Asia) to maintain focus in this  
1950 body of work.

1951 A custom confidence threshold was not set. All responses returned from each of  
1952 the services were included for analysis; where confidences were low, they were still  
1953 included for analysis. This is because we used the default thresholds of each API to  
1954 hint at what real-world applications may be like when testing and evaluating these  
1955 services.

1956 The label string returned from each service was only considered. It is common  
1957 for some labels to respond back that are conceptually similar (e.g., ‘car’ vs. ‘automobile’)  
1958 or grammatically different (e.g., ‘clothes’ vs. ‘clothing’). While we could have  
1959 employed more conceptual comparison or grammatical fixes in this study, we chose

1960 only to compare lowercased labels and as returned. We leave semantic comparison  
1961 open to future work.

1962 Only introductory analysis has been applied in assessing the documentation of  
1963 these services. Further detailed analysis of documentation quality against a rigorous  
1964 documentation quality framework would be needed to fortify our analysis of the  
1965 evolution of these services' documentation.

#### 1966 4.7.2 External Validity

1967 The documentation and services do change over time and evolve, with many allowing  
1968 for contributions from the developer community via GitHub. We note that our  
1969 evaluation of the documentation was conducted on a single date (see Section 4.4)  
1970 and acknowledge that the documentation may have changed from the evaluation date  
1971 to the time of this publication. We also acknowledge that the responses and labelling  
1972 may have evolved too since the evaluation period described and the date of this  
1973 publication. Thus, this may have an impact on the results we have produced in this  
1974 paper compared to current, real-world results. To mitigate this, we have supplied the  
1975 raw responses available online [326].

1976 Moreover, in this paper we have investigated *computer vision* services. Thus,  
1977 the significance of our results to other domains such as natural language processing  
1978 or audio transcription is, therefore, unknown. Future studies may wish to repeat our  
1979 methodology on other domains to validate if similar patterns occur; we remain this  
1980 open for future work.

#### 1981 4.7.3 Construct Validity

1982 It is not clear if all the recommendations proposed in Section 4.6 are feasible  
1983 or implementable in practice. Construct validity defines how well an experiment  
1984 measures up to its claims; the experiments proposed in this paper support our three  
1985 hypotheses but these have been conducted in a clinical condition. Real-world case  
1986 studies and feedback from developers and providers in industry would remove the  
1987 controlled nature of our work.

### 1988 4.8 Conclusions & Future Work

1989 This study explored three popular CVSS over an 11 month longitudinal experiment  
1990 to determine if these services pose any evolution risk or inconsistency. We find that  
1991 these services are generally stable but behave inconsistently; responses from these  
1992 services do change with time and this is not visible to the developers who use them.  
1993 Furthermore, the limitations of these systems are not properly conveyed by vendors.  
1994 From our analysis, we present a set of recommendations for both IWS vendors and  
1995 developers.

1996 Standardised software quality models (e.g., [135]) target maintainability and  
1997 reliability as primary characteristics. Quality software is stable, testable, fault  
1998 tolerant, easy to change and mature. These CVSS are, however, in a nascent stage,

1999 difficult to evaluate, and currently are not easily interchangeable. Effectively, the  
2000 IWS response objects are shifting in material ways to developers, albeit slowly, and  
2001 vendors do not communicate this evolution or modify API endpoints; the endpoint  
2002 remains static but the content returned does not despite the same input.

2003 There are many potential directions stemming from this work. To start, we plan  
2004 to focus on preparing a more comprehensive datasheet specifically targeted at what  
2005 should be documented to application developers, and not data scientists. Reapplying  
2006 this work in real-world contexts, that is, to get real developer opinions and study  
2007 production grade systems, would also be beneficial to understand these phenomena  
2008 in-context. This will help us clarify if such changes are a real concern for developers  
2009 (i.e., if they really need to change between services, or the service evolution has real  
2010 impact on their applications). We also wish to refine and systematise the method  
2011 used in this study and develop change detectors that can be used to identify evolution  
2012 in these services that can be applied to specific ML domains (i.e., not just CV),  
2013 data sets, and API endpoints, thereby assisting application developers in their testing  
2014 strategies. Moreover, future studies may wish to expand the methodology applied by  
2015 refining how the responses are compared. As there does not yet exist a standardised  
2016 list of terms available between services, labels could be *semantically* compared  
2017 instead of using exact matches (e.g., by using stem words and synonyms to compare  
2018 similar meanings of these labels), similar to previous studies [213].

2019 This paper has highlighted only some high-level issues that may be involved  
2020 in using these evolving services. The laws of software evolution suggest that for  
2021 software to be useful, it must evolve [194, 284]. There is, therefore, a trade-off, as  
2022 we have shown, between consistency and evolution in this space. For a component  
2023 to be stable, any changes to dependencies it relies on must be communicated. We  
2024 are yet to see this maturity of communication from IWS providers. Thus, developers  
2025 must be cautious between integrating intelligent components into their applications  
2026 at the expense of stability; as the field of AI is moving quickly, we are more likely to  
2027 see further instability and evolution in IWSs as a consequence.

# CHAPTER 5

2028

2029

---

## 2030 Systematic Mapping Study of API Documentation Knowledge<sup>†</sup>

---

2031

2032 **Abstract** Good application programming interface (API) documentation facilities the development process, improving productivity and quality. While the topic of API documentation quality has been of interest for the last two decades, there have been few studies to map the specific constructs needed to create a good document. In effect, we still need a structured taxonomy against which to capture knowledge. This study reports emerging results of a systematic mapping study. We capture key conclusions from previous studies that assess API documentation quality, and synthesise the results into a single framework. By conducting a systematic review of 21 key works, we have developed a five dimensional taxonomy based on 34 categorised weighted recommendations. All studies utilise field study techniques to arrive at their recommendations, with seven studies employing some form of interview and questionnaire, and four conducting documentation analysis. The taxonomy we synthesise reinforces that usage description details (code snippets, tutorials, and reference documents) are generally highly weighted as helpful in API documentation, in addition to design rationale and presentation. We propose extensions to this study aligned to developer's utility for each of the taxonomy's categories.

### 2047 5.1 Introduction

2048 Improving the quality of application programming interface (API) documentation is  
2049 highly valuable to the software development process; good documentation facilitates  
2050 productivity and thus quality is better engineered into the system [191]. Where an  
2051 application developer integrates new pieces of functionality (via APIs) into a system,  
2052 their productivity is affected either by inadequate skills (“*I've never used an API like  
2053 this, so must learn from scratch*”) or, where their skills are adequate, an imbalanced  
2054 cognitive load that causes excessive context switching (“*I have the skills for this,*

---

<sup>†</sup>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.

2055 *but am confused or misunderstand*"). In the latter case, what causes this confusion  
2056 and how to mitigate it via improved API documentation is an area that has been  
2057 explored; prior studies have provided recommendations based on both qualitative  
2058 and quantitative analysis of developer's opinions. These recommendations and  
2059 guidelines propose ways by which developers, managers and solution architects can  
2060 construct systems better.

2061 However, to date there has been little attempt to systematically capture this  
2062 knowledge about API documentation from various studies into a readily accessible,  
2063 consolidated format, that assists API designers to prepare better documentation.  
2064 While previous works have covered certain aspects of API usage, many have lacked  
2065 a systematic review of literature and do not offer a taxonomy to consolidate these  
2066 guidelines together. For example, some studies have considered the technical imple-  
2067 mentation improving API usability or tools to generate (or validate) API documen-  
2068 tation from its source code (e.g., [181, 211, 299]); there still lacks a consolidated  
2069 effort to capture the knowledge and artefacts best suited to *manually write* API  
2070 documentation.

2071 *⟨ todo: Introduce intelligent services documentation ⟩* The need for these insights  
2072 to be well-captured is evermore present with the introduction of intelligent web  
2073 services (IWSs), in which an AI-based component produces a non-deterministic  
2074 result based on a machine-learnt data-driven algorithm, rather than a predictable,  
2075 rule-driven one [69]. A popular subset of such components include computer vision  
2076 services (CVSs), which use machine intelligence to make predictions on images  
2077 such as object categorisation or facial recognition [318, 319, 320, 321, 322, 329,  
2078 331, 333, 334, 335, 339, 342, 343, 376, 377]. The longer-term impacts of *poor*  
2079 *documentation* for CVSs is largely under-explored and may have significant impacts  
2080 of AI-first systems if the application developers who use them do not think in the  
2081 non-deterministic mental model of the designers who create CVSs.

2082 This paper presents outcomes from a preliminary work to address this gap and  
2083 offers four key contributions:

- 2084 • a systematic mapping study (SMS) consisting of 21 studies that capture what  
2085 knowledge or artefacts should be contained within API documentation; and,
- 2086 • a structured taxonomy based on the consolidated recommendations of these  
2087 21 studies.
- 2088 • *⟨ todo: Relevance as per devs ⟩* an assessment of the efficacy of these rec-  
2089 commendations via a 'relevance ranking' for each of the 34 categories that  
2090 empirically reflects what is important to document from a *practitioner* point  
2091 of view;
- 2092 • *⟨ todo: Assessment against docs ⟩* a heuristic validation of the taxonomy  
2093 against CVSs to assess where existing CVSs documentation needs improve-  
2094 ment.

2095 After performing our SMS on what API knowledge should be captured in  
2096 documentation—to assist API designers—we propose a five dimensional taxonomy  
2097 consisting of: (1) Usage Description; (2) Design Rationale; (3) Domain Concepts;  
2098 (4) Support Artefacts; and (5) Documentation Presentation. *⟨ todo: Describe that*

2099     *this was weighted against devs opinions* ) This taxonomy was then surveyed against  
2100 X developers to assess the relevance of these weightings from the practitioner’s  
2101 viewpoint. ⟨ todo: *Describe our application of the taxonomy to CVS* ⟩ We then  
2102 assess our taxonomy against a subset of IWSs—three popular CVS: Google Cloud  
2103 Vision [331], AWS Rekognition [318] and Azure Computer Vision [339]. We an-  
2104 ticipate that future API designers may use this resource as a means to improve the  
2105 ways in which they communicate their mental models and patterns of thinking while  
2106 developing these APIs.

2107     This paper is structured as thus: Section 5.2 presents related work in the area;  
2108 Section 5.3 is divided into two subsections, the first describing how primary sources  
2109 were selected in a SMS with the second describing the development of our taxonomy  
2110 from these sources; ⟨ todo: *Introduce section where we validated study* ⟩ Section 5.4  
2111 describes how we adopted the System Usability Scale (SUS) to develop a survey  
2112 instrument of 52 questions to validate the taxonomy and assess its efficacy against  
2113 the three popular CVS selected; Section 5.5 presents the findings from our validation  
2114 and the proposed taxonomy; Section 5.6 describes the threats to validity of this work  
2115 and Section 5.7 provides concluding remarks and the future directions of this study.

## 2116     5.2 Related Work

2117     SMSs have previously been explored in the area of API usability and developer  
2118 experience. Nybom et al. [211] recently performed a SMS on 36 API documentation  
2119 generation tools and approaches. Presented is an analysis of state-of-the-art of  
2120 the tools developed, what kind of documentation is generated by them, and the  
2121 dependencies they require to generate this documentation. Their findings highlight  
2122 a recent effort on the development of API documentation by producing example  
2123 code snippets and/or templates on how to use the API or bootstrap developers to  
2124 begin using the APIs. A secondary focus is closely followed by tools that produce  
2125 natural language descriptions that can be produced within developer documentation.  
2126 However, Nybom et al. produce a SMS on the types of *tooling* that exists to assist  
2127 in producing and validating API documentation. While this is a systematic study  
2128 with key insights into the types of tooling produced, there is still a gap for a SMS in  
2129 what *guidelines* have been produced by the literature in developing natural-language  
2130 documentation itself, which our work has addressed.

2131     Watson [299] performed a heuristic assessment of 11 high-level universal design  
2132 elements of API documentation against 35 popular APIs. He demonstrated that many  
2133 of these popular APIs fail to grasp even the basic of these elements; for example,  
2134 25% of the documentation sets did not provide any basic overview documentation.  
2135 However, the heuristic used within this study consists of just 11 elements and is based  
2136 on only three seminal works. Our work extends these heuristics and structures them  
2137 into a consolidated, hierarchical taxonomy using a systematic taxonomy development  
2138 method for software engineering (SE).

2139     A taxonomy of knowledge patterns within API reference documentation by  
2140 Maalej and Robillard [181] classified 12 distinct knowledge types. Evaluation of the  
2141 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* ⟩

### 5.3 Method

Our taxonomy development consisted of two phases. Firstly, we conducted an SMS to identify and analyse API documentation studies, following the guidelines of Kitchenham and Charters [153] and Petersen et al. [226]. Following this, we followed the SE taxonomy development method devised by Usman et al. [289] on our findings from the SMS.

#### 5.3.1 Systematic Mapping Study

**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?

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

**Automatic Filtering** Informed by similar previous studies in SE [106, 112, 289], we begin by defining the SWEBOK [133] knowledge areas (KAs) to assist in the search and mapping process of an SMS. Our search query was built using related KAs, relevant synonyms, and the term ‘software engineering’ (for comprehensiveness) all joined with the OR operator. Due to the lack of a standard definition of an API, we include the terms: ‘API’ and its expanded term; software library, component and framework; and lastly SDK and its expanded term. These too were joined with the OR operator, appended with an AND. Lastly, the term ‘documentation’ was 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
<b>Total</b>	<b>91</b>	<b>591</b>	<b>3819</b>	<b>4501</b>

( “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 )

<sup>2175</sup> The query was then executed on all available metadata (title, abstract and key-  
<sup>2176</sup> words) on three primary sources to search for relevant studies in May 2019. Web of  
<sup>2177</sup> Science<sup>1</sup> (WoS), Compendex/Inspec<sup>2</sup> (C/I) and Scopus<sup>3</sup> were chosen due to their rel-  
<sup>2178</sup> evance in SE literature (containing the IEEE, ACM, Springer and Elsevier databases)  
<sup>2179</sup> and their ability to support advanced queries [46, 153]. A total 4,501 results<sup>4</sup> were  
<sup>2180</sup> found, with 549 being duplicates. Table 5.1 displays our results in further detail (du-  
<sup>2181</sup> plicates not omitted); Figure 5.1 shows an exponential trend of API documentation  
<sup>2182</sup> publications produced within the last two decades.

<sup>2183</sup> **Manual Filtering** A follow-up manual filtering to select primary studies was per-  
<sup>2184</sup> formed on the 4,501 results using the following inclusion criteria (IC) and exclusion  
<sup>2185</sup> criteria (EC):

<sup>2186</sup> **IC1** Studies must be relevant to API documentation: specifically, we  
<sup>2187</sup> exclude studies that deal with improving the technical API usability  
<sup>2188</sup> (e.g., improved usage patterns);

<sup>2189</sup> **IC2** Studies must propose new knowledge or recommendations to docu-  
<sup>2190</sup> ment APIs;

<sup>2191</sup> **IC3** Studies must be relevant to SE as defined in SWEBOK;

<sup>2192</sup> **EC1** Studies where full-text is not accessible through standard institutional  
<sup>2193</sup> databases;

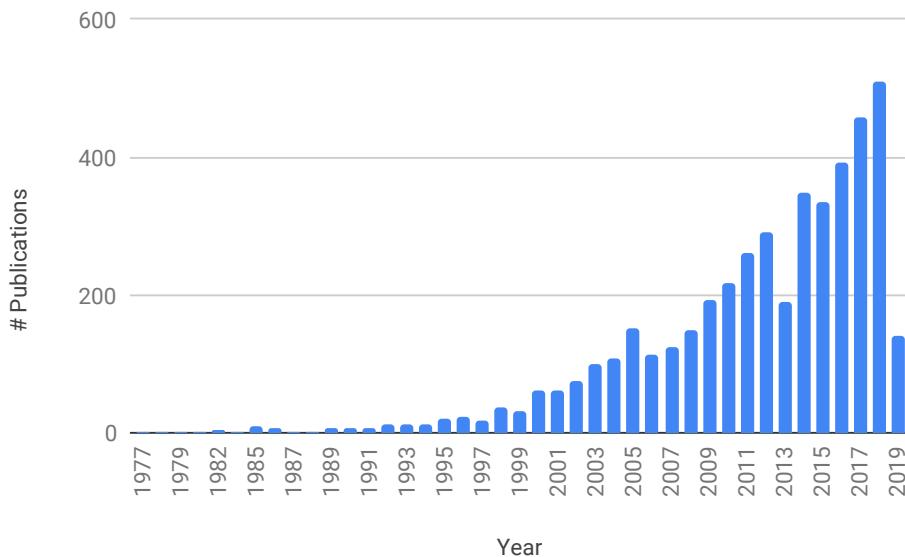
<sup>2194</sup> **EC2** Studies that do not propose or extend how to improve the official,  
<sup>2195</sup> natural language documentation of an API;

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

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

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

<sup>4</sup>Raw 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.)

- <sup>2196</sup> **EC3** Studies proposing a third-party tool to enhance existing documentation or generate new documentation using data mining (i.e., not proposing strategies to improve official documentation);
- <sup>2197</sup> **EC4** Studies not written in English;
- <sup>2198</sup> **EC5** Studies not peer-reviewed.

<sup>2201</sup> After exporting metadata of search results to a spreadsheet, a three-phase curation  
<sup>2202</sup> process was conducted. The first author read the publication source (to omit non-  
<sup>2203</sup> SE papers quickly), author keywords and title of all 4,501 studies (514 that were  
<sup>2204</sup> duplicates), and abstract. As we considered multiple databases, some studies were  
<sup>2205</sup> repeated. However, the DOIs and titles were sorted and reviewed, retaining only  
<sup>2206</sup> one copy of the paper from a single database. Moreover, as there was no limit  
<sup>2207</sup> to the year range in our query, some studies were republished in various venues.  
<sup>2208</sup> These, too, were handled with title similarity matching, wherein only the first paper  
<sup>2209</sup> was considered. Where the inclusion or exclusion criteria could not be determined  
<sup>2210</sup> from the abstract alone, the paper was automatically shortlisted. Any doubt in a  
<sup>2211</sup> study automatically included it into the second phase. This resulted in 133 studies  
<sup>2212</sup> being shortlisted to the second phase. We rejected 427 studies that were unrelated  
<sup>2213</sup> to SE, 3,235 were not directly related to documenting APIs (e.g., to enhance coding  
<sup>2214</sup> techniques that improve the overall developer usability of the API), 182 proposed new  
<sup>2215</sup> tools to enhance API documentation or used machine learning to mine developer's  
<sup>2216</sup> discussion of APIs, and 10 were not in English.

<sup>2217</sup> The shortlisted studies were then re-evaluated by re-reading the abstract, the  
<sup>2218</sup> introduction and conclusion. Performing this second phase removed a further 64

2219 studies that were on API usability or non API-related documentation (i.e., code  
2220 commenting); we further refined our exclusion criteria to better match the research  
2221 outcomes of this goal (chiefly including the word ‘natural language’ documentation  
2222 in EC2) which removed studies focused to improve technical documentation of  
2223 APIs such as data types and communication schemas. Additionally, 26 studies were  
2224 removed as they were related to introducing new tools (EC3), 3 were focused on tools  
2225 to mine API documentation, 7 studies where no recommendations were provided,  
2226 2 further duplicate studies, and a further 10 studies where the full text was not  
2227 available, not peer reviewed or in English. Books are commonly not peer-reviewed  
2228 (EC5), however no books were shortlisted within these results. This resulted in 21  
2229 primary studies for further analysis. The mapping of primary study identifiers to  
2230 references S1–21 can be found in Appendix D.

2231 Intra-rater reliability of our 133 shortlisted papers was tested using the test-retest  
2232 approach [153] by re-evaluating a random sample of 10% (13 total) of the studies  
2233 shortlisted above a week after initial studies were shortlisted. Using the Cohen’s  
2234 kappa coefficient as a metric for reliability,  $\kappa = 0.7547$ , indicating substantial  
2235 agreement [166].

2236 **Data Extraction** Of the 21 primary studies, we conducted abstract key-wording  
2237 adhering to Petersen et al.’s guidelines [226] to develop a classification scheme. An  
2238 initial set of keywords were applied for each paper in terms of their methodologies and  
2239 research approaches (RQ2), based on an existing classification schema by Wieringa  
2240 and Heerkens [304]: evaluation, validation, personal experience and philosophical  
2241 papers.

2242 After all primary studies had been assigned keywords, we noticed that all papers  
2243 used field study techniques, and thus we consolidated these keywords using Singer  
2244 et al.’s framework of SE field study techniques [266]. Singer et al. captures both  
2245 study techniques *and* methods to collect data within the one framework, namely:  
2246 *direct techniques*, including brainstorming and focus groups, interviews and questionnaires,  
2247 conceptual modelling, work diaries, think-aloud sessions, shadowing and observation,  
2248 participant observation; *indirect techniques*, including instrumenting systems, fly-on-the-wall;  
2249 and *independent techniques*, including analysis of work databases, tool use logs, documentation analysis, and static and dynamic analysis.

2251 Table 5.2 describes our data extraction form, which was used to collect relevant  
2252 data from each paper. Figure 5.2 maps each study to one (or more, if applicable) of  
2253 methodologies plotted against Wieringa and Heerkens’s research approaches.

### 2254 **5.3.2 Development of the Taxonomy**

2255 Usman et al. concludes that a majority of SE taxonomies are developed in an ad-hoc  
2256 way [289], and proposes a systematic approach to develop taxonomies in SE that  
2257 extends previous efforts by including lessons learned from more mature fields. In  
2258 this subsection, we outline the 4 phases and 13 steps taken to develop our taxonomy  
2259 based on Usman et al.’s technique.

**Table 5.2:** Data extraction form

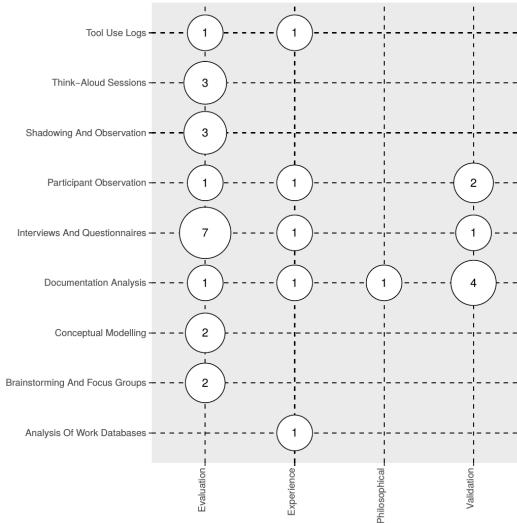
Data item(s)	Description
Citation metadata	Title, author(s), years, publication venue, publication type
Key recommendation(s)	As per IC2, the study must propose at least one recommendation on what should be captured in API documentation
Evaluation method	Did the authors evaluate their recommendations? 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

2260 **Planning phase** The planning phase of the technique involves the following six  
2261 steps:

- 2262 **(1) defining the SE KA:** The software engineering KA, as defined by the SWEBOK,  
2263 is software construction;
- 2264 **(2) defining the objective:** The main objective of the proposed taxonomy is to  
2265 define a set of categories that enables to classify different facets of natural-  
2266 language API *documentation* knowledge (not API *usability* knowledge) as  
2267 reported in existing literature;
- 2268 **(3) defining the subject matter:** The subject matter of our proposed taxonomy is  
2269 documentation artefacts of APIs;
- 2270 **(4) defining the classification structure:** The classification structure of our pro-  
2271 posed taxonomy is *hierarchical*;
- 2272 **(5) defining the classification procedure:** The procedure used to classify the  
2273 documentation artefacts is *qualitative*;
- 2274 **(6) defining the data sources:** The basis of the taxonomy is derived from field  
2275 study techniques (see Section 5.3.1).

2276 **Identification and extraction phase** The second phase of the taxonomy devel-  
2277 opment involves **(7) extracting all terms and concepts** from relevant literature, as  
2278 selected from our 21 primary studies. These terms are then consolidated by **(8) per-**  
2279 forming terminology control, as some terms may refer to different concepts and  
2280 vice-versa.

2281 **Design phase** The design phase identified the core dimensions and categories  
2282 within the extracted data items. The first step is to **(9) identify and define taxonomy**  
2283 *dimensions*; for this study we utilised a bottom-up approach to identify the dimen-  
2284 sions, i.e., extracting the categories first and then nominating which dimensions



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

2285 these categories fit into using an iterative approach. As a bottom-up approach was  
 2286 utilised, step (9) also encompassed the second stage of the design phase, which is to  
 2287 **(10)** *identify and describe the categories* of each dimension. Thirdly, we **(11)** *iden-*  
 2288 *tify and describe relationships* between dimensions and categories, which can be  
 2289 skipped if the relationships are too close together, as is the case of our grouping  
 2290 technique which allows for new dimensions and categories to be added. The last  
 2291 step in this phase is to **(12)** *define guidelines for using and updating the taxonomy*,  
 2292 however as this taxonomy still an emerging result, guidelines to update and use the  
 2293 taxonomy are anticipated future work.

2294 **Validation phase** < *todo: Revise this paragraph* > In the final phase of taxonomy  
 2295 development, taxonomy designers must **(13)** *validate the taxonomy* to assess its use-  
 2296 fulness. Usman et al. [289] describe three approaches to validate taxonomies: (i)  
 2297 orthogonal demonstration, in which the taxonomy's orthogonality is demonstrated  
 2298 against the dimensions and categories, (ii) benchmarking the taxonomy against sim-  
 2299 ilar classification schemes, or (iii) utility demonstration by applying the taxonomy  
 2300 heuristically against subject-matter examples. In our study, we adopt utility demon-  
 2301 stration by use of a rigorous experiment and heuristic application against real-world  
 2302 case-studies (i.e., within the domain of IWSs). This is is discussed in greater detail  
 2303 within Section 5.4.

## 2304 **5.4 Taxonomy Validation**

2305 < *todo: Describe validation* > To validate our taxonomy, we conducted a two-fold  
 2306 experiment. Firstly, we

### 2307 **5.4.1 Survey Study**

### 2308 **5.4.2 Empirical Application against Computer Vision Services**

## 2309 **5.5 Taxonomy**

2310 Our taxonomy consists of five dimensions (labelled A–E) that respectively cover:

- 2311 • **[A] Usage Description** on *how* to use the API for the developer’s intended  
use case;
- 2313 • **[B] Design Rationale** on *when* the developer should choose this API for a  
particular use case;
- 2315 • **[C] Domain Concepts** of the domain behind the API to understand *why* this  
API should be chosen for this domain;
- 2317 • **[D] Support Artefacts** that describe *what* additional documentation the API  
provides; and
- 2319 • **[E] Documentation Presentation** to help organise the *visualisation* of the  
above information.

2321 Further descriptions of the categories encompassing each dimension are given within  
2322 Table 5.3, coded as  $[Xi]$ , where  $i$  is the category identifier within a dimension,  $X$ ,  
2323 where  $X \in \{A, B, C, D, E\}$ .

2324 We expand these five dimensions into 34 categories (sub-dimensions) and Ta-  
2325 ble 5.3 provides a weighting of these categories in the rightmost column as calculated  
2326 as a percentage of the number of primary studies per category divided by the total  
2327 of primary studies. The top five weighted categories (bolded in Table 5.3) highlight  
2328 what most studies recommend documenting in API documentation, with the top  
2329 three falling under the Usage Description dimension.

2330 The majority (71%) of studies advocate for **code snippets** as a necessary piece  
2331 in the API documentation puzzle [A5]. While code snippets generally only reflect  
2332 small portions of API functionality (limited to 15–30 LoC), this is complimented by  
2333 **step-by-step tutorials** (57% of studies) that tie in multiple (disparate) components  
2334 of API functionality, generally with some form of screenshots, demonstrating the  
2335 development of a non-trivial application using the API step-by-step [A6]. The third  
2336 highest category weighted was also under the Usage Description dimension, being  
2337 **low-level reference documentation** at 52% [A2]. These three categories were the  
2338 only categories to be weighted as majority categories (i.e., their weighting was above  
2339 50%).

2340 The fourth and fifth highest weights are **an entry-level purpose/overview of**  
2341 **the API** (48%) that gives a brief motivation as to why a developer should choose  
2342 a particular API over another [B1] and **consistency in the look and feel** of the  
2343 documentation throughout all of the API’s official documentation (43%) [E6].

## 2344 **5.6 Threats to Validity**

2345 Threats to *internal validity* concern factors internal to our study that may affect  
2346 results. Guidelines on producing systematic reviews [153] suggest that single re-

2347 researchers conducting their reviews should discuss the review protocol, inclusion  
2348 decisions, data extraction with a third party. In this paper, we have presented the  
2349 early outcomes of our systematic review, which has utilised the test-retest method-  
2350 ology as a measure of reliability. MacDonell et al. [182] states that a defining  
2351 characteristic of any SMS is to test the reliability of the review and extraction pro-  
2352 cesses. We plan to mitigate this threat by conducting *inter*-relater reliability with  
2353 the continuation of this work, using independent analysis and conflict resolution as  
2354 per guidelines suggested by Garousi and Felderer [105]. Similarly, the development  
2355 of our taxonomy would benefit from an inter-rater reliability categorisation of a  
2356 sample of papers to both ensure that our weightings of categories are reliable and  
2357 that the categories and dimensions fit the objectives of the taxonomy. Furthermore,  
2358 a future user study (see Section 5.7) will be needed to assess whether the extracted  
2359 information from API documentation actually impacts on developer productivity,  
2360 and the usefulness of such a taxonomy should be evaluated.

2361 Threats to *external validity* represent the generalisation of the observations we  
2362 have found in this study. While we have used a broad range of literature that  
2363 encompasses API documentation guidelines, we acknowledge that not all papers  
2364 contributing to API documentation may have been captured in the taxonomy. All  
2365 efforts were made to include as many papers as possible given our filtering technique,  
2366 though it is likely that some papers filtered out (e.g., papers not in English) may  
2367 alter our conclusions, introducing conflicting recommendations. However, given the  
2368 consistency of these trends within the studies that were sourced, we consider this a  
2369 low likelihood.

2370 Threats to *construct validity* relates to the degree by which the data extrapolated  
2371 in this study sufficiently measures its intended goals. Automatic searching was  
2372 conducted in the SMS by choice of three popular databases (see Section 5.3.1).  
2373 As a consequence of selecting multiple databases, duplicates were returned. This  
2374 was mitigated by manually curating out all duplicate results from the set of studies  
2375 returned. Additionally, we acknowledge that the lack manual searching of papers  
2376 within particular venues may be an additional threat due to the misalignment of  
2377 search query keywords to intended papers of inclusion. Thus, our conclusions are  
2378 only applicable to the information we were able to extract and summarise, given the  
2379 primary sources selected.

## 2380 5.7 Conclusions & Future Work

2381 API documentation is an aspect of quality of software, as it facilitates the developer’s  
2382 productivity and assists with evolution. Improving the quality of the documentation  
2383 of third party APIs improves the quality of software using them.

2384 To date, we did not find a systematic literature review that offers a consolidated  
2385 taxonomy of key recommendations. Moreover, there has been little work on map-  
2386 ping the research produced in this space against the techniques used to arrive at  
2387 the recommendations. Starting with 4,501 papers potentially relating to API doc-  
2388 umentation, we identified 21 key relevant studies, and synthesise a taxonomy of  
2389 the various documentation aspects that should improve API documentation quality.

<sup>2390</sup> Furthermore, we also capture the most commonly used analysis techniques used in  
<sup>2391</sup> the academic literature. Figure 5.2 highlights that a majority of these studies employ  
<sup>2392</sup> interviews and questionnaires, and only some undertake structured documentation  
<sup>2393</sup> analysis.

<sup>2394</sup> In future revisions of this work, we intend use our results as the input to a  
<sup>2395</sup> restricted systematic literature review in API documentation artefacts. In doing so,  
<sup>2396</sup> we will consider conducting the following:

- <sup>2397</sup> • improving reliability metrics of our study (see Section 5.6) with an inter-rater  
<sup>2398</sup> reliability method;
- <sup>2399</sup> • reviewing the techniques and evaluation of our selected studies to extract the  
<sup>2400</sup> effectiveness of the various approaches used in the conclusions

<sup>2401</sup> We believe the results of this preliminary empirical work may provide further  
<sup>2402</sup> insight for future follow-up user studies with developers. Whilst our aim is to  
<sup>2403</sup> eventually improve the quality of API documentation, the ultimate goal is improving  
<sup>2404</sup> the developer's experience when producing systems and, therefore, improving the  
<sup>2405</sup> efficacy and productivity at which software is produced within industry. We hope  
<sup>2406</sup> that API designers will utilise the taxonomy produced in this paper as a weighted  
<sup>2407</sup> 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*).

<b>Key</b>	<b>Description</b>	<b>Primary Sources</b>	<b>Total (%)</b>
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

2408

2409

2410

## Interpreting Pain-Points in Computer Vision Services<sup>†</sup>

2411

2412 **Abstract** Intelligent services are becoming increasingly more pervasive; application de-  
2413 velopers want to leverage the latest advances in areas such as computer vision to provide new  
2414 services and products to users, and large technology firms enable this via RESTful APIs.  
2415 While such APIs promise an easy-to-integrate on-demand machine intelligence, their current  
2416 design, documentation and developer interface hides much of the underlying machine  
2417 learning techniques that power them. Such APIs look and feel like conventional APIs but  
2418 abstract away data-driven probabilistic behaviour—the implications of a developer treating  
2419 these APIs in the same way as other, traditional cloud services, such as cloud storage, is  
2420 of concern. The objective of this study is to determine the various pain-points developers  
2421 face when implementing systems that rely on the most mature of these intelligent services,  
2422 specifically those that provide computer vision. We use Stack Overflow to mine indications  
2423 of the frustrations that developers appear to face when using computer vision services, clas-  
2424 sifying their questions against two recent classification taxonomies (documentation-related  
2425 and general questions). We find that, unlike mature fields like mobile development, there  
2426 is a contrast in the types of questions asked by developers. These indicate a shallow un-  
2427 derstanding of the underlying technology that empower such systems. We discuss several  
2428 implications of these findings via the lens of learning taxonomies to suggest how the software  
2429 engineering community can improve these services and comment on the nature by which  
2430 developers use them.

### 6.1 Introduction

2431 The availability of recent advances in artificial intelligence (AI) over simple REST-  
2432 ful end-points offers application developers new opportunities. These new intel-

<sup>†</sup>This chapter is originally based on A. Cummaudo, R. Vasa, S. Barnett, J. Grundy, and M. Abdellazek, “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.

2434    ligent services (AI components) abstract complex machine learning (ML) and AI  
2435    techniques behind simpler API calls. In particular, they hide (either explicitly or  
2436    implicitly) any data-driven and non-deterministic properties inherent to the process  
2437    of their construction. The promise is that software engineers can incorporate com-  
2438    plex machine learnt capabilities, such as computer vision, by simply calling an API  
2439    end-point.

2440    The expectation is that application developers can use these AI-powered services  
2441    like they use other conventional software components and cloud services (e.g., object  
2442    storage like AWS S3). Furthermore, the documentation of these AI components is  
2443    still anchored to the traditional approach of briefly explaining the end-points with  
2444    some information about the expected inputs and responses. The presupposition  
2445    is that developers can reason and work with this high level information. These  
2446    services are also marketed to suggest that application developers do not need to fully  
2447    understand how these components were created (i.e., assumptions in training data  
2448    and training algorithms), the ways in which the components can fail, and when such  
2449    components should and should not be used.

2450    The nuances of ML and AI powering intelligent services have to be appre-  
2451    ciated, as there are real-world consequences to software quality for applications  
2452    that depend on them if they are ignored [69]. This is especially true when ML  
2453    and AI are abstracted and masked behind a conventional-looking API call, yet the  
2454    mechanisms behind the API are data-dependent, probabilistic and potentially non-  
2455    deterministic [213]. We are yet to discover what long-term impacts exist during  
2456    development and production due to poor documentation that do not capture these  
2457    traits, nor do we know the depth of understanding application developers have for  
2458    these components. Given the way AI-powered services are currently presented, de-  
2459    velopers are also likely to reason about these new services much like a string library  
2460    or a cloud data storage service. That is, they may not fully consider the implica-  
2461    tions of the underlying statistical nature of these new abstractions or the consequent  
2462    impacts on productivity and quality.

2463    Typically, when developers are unable to correctly align to the mindset of the  
2464    API designer, they attempt to resolve issues by (re-)reading the API documentation.  
2465    If they are still unable to resolve these issues on their own after some internet  
2466    searching, they consider online discussion platforms (e.g., Stack Overflow, GitHub  
2467    Issues, Mailing Lists) where they seek technological advice from their peers [3].  
2468    Capturing what developers discuss on these platforms offers an insight into the  
2469    frustrations developers face when using different software components as shown  
2470    by recent works [30, 150, 247, 272, 297]. However, to our knowledge, no studies  
2471    have yet analysed what developers struggle with when using the new generation of  
2472    *intelligent* services. Given the re-emergent interest in AI and the anticipated value  
2473    from this technology [178], a better understanding of issues faced by developers  
2474    will help us improve the quality of services. Our hypothesis is that application  
2475    developers do not fully appreciate the probabilistic nature of these services, nor do  
2476    they have sufficient appreciation of necessary background knowledge—however, we  
2477    do not know the specific areas of concern. The motivation for our study is to inform  
2478    API designers on which aspects to focus in their documentation, education, and

2479 potentially refine the design of the end-points.

2480 This study involves an investigation of 1,825 Stack Overflow (SO) posts regarding  
2481 one of the most mature types of intelligent services—computer vision services—  
2482 dating from November 2012 to June 2019. We adapt existing methodologies of prior  
2483 SO analyses [30, 281] to extract posts related to computer vision services. We then  
2484 apply two existing SO question classification schemes presented at ICPC and ICSE  
2485 in 2018 and 2019 [3, 31]. These previous studies focused on mobile apps and web  
2486 applications. Although not a direct motivation, our work also serves as a validation  
2487 of the applicability of these two issue classification taxonomies [3, 31] in the context  
2488 of intelligent services (hence potential for generalisation). Additionally our work is  
2489 the first—to our knowledge—to *test* the applicability of these taxonomies in a new  
2490 study.

2491 The taxonomies in previous works focus on the specific aspects from the domain  
2492 (e.g. API usage, specificity within the documentation etc.) and as such do not  
2493 deeply consider the learning gap of an application developer. To explore the API  
2494 learning implications raised by our SO analysis, we applied an additional lens of  
2495 two taxonomies from the field of pedagogy. This was motivated by the need to offer  
2496 an insight into the work needed to help developers learn how to use these relatively  
2497 new services.

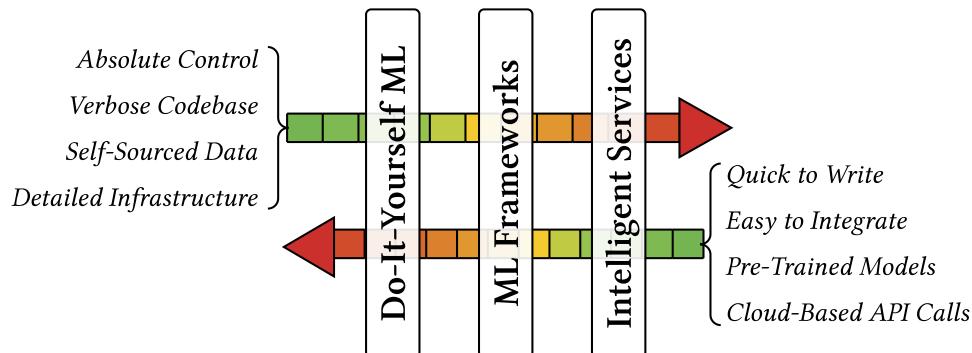
2498 The key findings of our study are:

- 2499 • The primary areas that developers raise as issues reflect a relatively primitive  
2500 understanding of the underlying concepts of data-driven ML approaches used.  
2501 We note this via the issues raised due to conceptual misunderstanding and  
2502 confusion in interpreting errors,
- 2503 • Developers predominantly encounter a different distribution of issue types than  
2504 were reported in previous studies, indicating the complexity of the technical  
2505 domain has a non-trivial influence on intelligent API usage; and
- 2506 • Most of these issues can be resolved with better documentation, based on our  
2507 analysis.

2508 The paper also offers a data-set as an additional contribution to the research  
2509 community and to permit replication [327]. The paper structure is as follows:  
2510 Section 6.2 provides motivational examples to highlight the core focus of our study;  
2511 Section 6.3 provides a background on prior studies that have mined SO to gather  
2512 insight into the SE community; Section 6.4 describes our study design in detail;  
2513 Section 6.5 presents the findings from the SO extraction; Section 6.6 offers an  
2514 interpretation of the results in addition to potential implications that arise from our  
2515 work; Section 6.7 outlines the limitations of our study; concluding remarks are given  
2516 in Section 6.8.

## 2517 6.2 Motivation

2518 “Intelligent” services are often available as a cloud end-point and provide developers  
2519 a friendly approach to access recent AI/ML advances without being experts in the  
2520 underlying processes. Figure 6.1 highlights how these services abstract away much



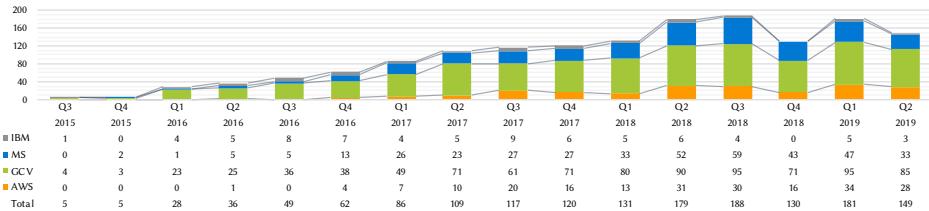
**Figure 6.1:** Some traits of Intelligent Services vs. ‘Do-It-Yourself’ ML. Green-to-red arrows indicate the presence of these traits. *Adapted from Ortiz [216].*

of the technical know-how needed to create and operationalise these intelligent services [216]. 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, 213]. 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.*” [353]; “*I’m trying out the [intelligent service], and there’s a score field that returns that I’m not sure how to interpret [it].*” [367]
- **frustrated about non-deterministic results:** “*Often the API has troubles in recognizing single digits... At other times Vision confuses digits with letters.*” [366]; “*Is there a way to help the program recognize numbers better, for example limit the results to a specific format, or to numbers only?*” [363]
- **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.*” [347]
- **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.*” [350]

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, 281]. 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*” [20, 30, 287]. They typically (manually) classify discussion into either *functional areas* (e.g., Website Design/CSS, Mobile App Development, .NET Framework, Java [20]) or *descriptive areas* (e.g., Coding Style/Practice, Problem/-Solution, Design, QA [20, 287]). 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 [287], categorise the questions they ask about these APIs [20, 22, 31, 247], and understand API related documentation and usage issues [3, 4, 7, 20, 129, 281]. These studies often employ automation to assist in the data analysis stages of their research. Latent Dirichlet Allocation [7, 20, 247, 287] is applied for topic modelling and other ML techniques such as Random Forests [31], Conditional Random Fields [4] or Support Vector Machines [31, 129] 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' [20], 'User Interface' [30] or 'Design' [288] 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, 281] (e.g., card sorting) or a combination of both [30, 31, 247, 286]. 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 [143, 146, 263]. 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 [331], AWS [318], Azure [339] and IBM Cloud [335]. 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 [241].

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 [321, 322, 329, 334, 376, 377] and only

2624 English-speaking services have been selected, excluding popular services from Asia  
2625 (e.g., [319, 320, 333, 342, 343])—see Section 6.7. For comprehensiveness, we  
2626 explain below our initial attempts to extract *all* intelligent services.

2627 **Defining a list of intelligent services** As there exists no global ‘list’ of intelligent  
2628 services to search on, we needed to derive a *corpus of initial terms* to allow us  
2629 to know *what* to search for on the Stack Exchange Data Explorer<sup>1</sup> (SEDE). We  
2630 began by looking at different brand names of cloud services and their permutations  
2631 (e.g., Google Cloud Services and GCS) as well as various ML-related products  
2632 (e.g., Google Cloud ML). To do this, we performed extensive Google searches<sup>2</sup> in  
2633 addition to manually reviewing six ‘overview’ pages of the relevant cloud platforms.  
2634 We identified 91 initial intelligent services to incorporate into our search terms<sup>3</sup>.

2635 **Manual search for relevant, related terms** We then ran a manual search<sup>2</sup> on  
2636 each term to determine if these terms were relevant. We did this by querying each  
2637 term within SO’s search feature, reviewing the titles and body post previews of  
2638 the first three pages of results (we did not review the answers, only the questions).  
2639 We also noted down the user-defined *Tags* of each post (up to five per question);  
2640 by clicking into each tag, we could review similar tags (e.g., ‘project-oxford’ for  
2641 ‘azure-cognitive-services’) and check if the tag had synonyms (e.g., ‘aws-lex’ and  
2642 ‘amazon-lex’). We then compiled a *corpus of tags* consisting of 31 terms.

2643 **Developing a search query** We recognise that searching SEDE via *Tags* exclu-  
2644 sively can be ineffective (see [20, 281]). To mitigate this, we produced a *corpus of*  
2645 *title and body terms*. Such terms are those that exist within the title and body of  
2646 the posts to reflect the ways in which individual developers commonly use to refer  
2647 to different intelligent services. To derive at such a list, we performed a search<sup>2,3</sup>  
2648 of the 31 tags above in SEDE, filtering out posts that were not answers (i.e., ques-  
2649 tions only) as we wanted to see how developers *phrase* their questions. For each  
2650 search, we extracted a random sample of 100 questions (400 total for each service)  
2651 and reviewed each question. We noted many patterns in the permutations of how  
2652 developers refer to these services, such as: common misspellings (‘bind’ vs. ‘bing’);  
2653 brand misunderstanding (‘Microsoft computer vision’ vs. ‘Azure computer vision’);  
2654 hyphenation (‘Auto-ML’ vs. ‘Auto ML’); UK and US English (‘Watson Analyser’  
2655 vs. ‘Watson Analyzer’); and, the use of apostrophes, plurals, and abbreviations  
2656 (‘Microsoft’s Computer Vision API’, ‘Microsoft Computer Vision Services’, ‘GCV’  
2657 vs. ‘Google Cloud Vision’). We arrived at a final list of 229 terms compromising  
2658 all of the intelligent services provided by Google, Amazon, Microsoft and IBM as  
2659 of January 2019<sup>3</sup>.

2660 **Executing our search query** Our next step was to perform a case-insensitive  
2661 search of all 229 terms within the body or title of posts. We used Google BigQuery’s

---

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

<sup>2</sup>This search was conducted on 17 January 2019

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

2662 public data-set of SO posts<sup>4</sup> to overcome SEDE’s 50,000 row limit and to conduct  
2663 a case-insensitive search. This search was conducted on 10 May 2019, where we  
2664 extracted 21,226 results. We then performed several filtering steps to cleanse our  
2665 extracted data, as explained below.

#### 2666 6.4.2 Data Filtering

2667 **Refining our inclusion/exclusion criteria** We performed an initial manual filter-  
2668 ing of the 50 most recent posts (sorted by descending *CreationDate* values) of the  
2669 21,226 posts above, assessing the suitability of the results and to help further refine  
2670 our inclusion and exclusion criteria. We did note that some abbreviations used in the  
2671 search terms (e.g., ‘GCV’, ‘WCS’<sup>5</sup>), resulting in irrelevant questions in our result  
2672 set. We therefore removed abbreviations from our search query and consolidated all  
2673 overlapping terms (e.g., ‘Google Vision **API**’ was collapsed into ‘Google Vision’).

2674 We also recognised that 21,226 results would be non-trivial to analyse without  
2675 automated techniques. As we wanted to do manual qualitative analysis, we reduced  
2676 our search space to 27 search terms of just the *computer vision services* within  
2677 the original corpus of 229 terms. These were Google Cloud Vision [331], AWS  
2678 Rekognition [318], Azure Computer Vision [339], and IBM Watson Visual Recog-  
2679 nition [335]. This resulted in 1,425 results that were extracted on 21 June 2019. The  
2680 query used and raw results are available online in our supplementary materials [327].

2681 **Duplicates** Within 1,425 results, no duplicate questions were noted, as determined  
2682 by unique post ID, title or timestamp.

2683 **Automated and manual filtering** To assess the suitability and nature of the 1,425  
2684 questions extracted, the first author began with a manual check on a randomised  
2685 sample of 50 questions. As the questions were exported in a raw CSV format (with  
2686 HTML tags included in the post’s body), we parsed the questions through an ERB  
2687 templating engine script<sup>6</sup> in which the ID, title, body, tags, created date, and view,  
2688 answer and comment counts were rendered for each post in an easily-readable format.  
2689 Additionally, SQL matches in the extraction process were also highlighted in yellow  
2690 (i.e., in the body of the post) and listed at the top of each post. These visual cues  
2691 helped to identify 3 false positive matches where library imports or stack traces  
2692 included terms within our corpus of 26 computer vision service terms. For example,  
2693 `aws-java-sdk-rekognition:jar` is falsely matched as a dependency within an  
2694 unrelated question. As such exact matches would be hard to remove without the use  
2695 of regular expressions, and due to the low likelihood (6%) of their appearance, we  
2696 did not perform any followup automatic filtering.

2697 **Classification** Our 1,425 posts were then split into 4 additional random samples  
2698 (in addition to the random sample of 50 above). 475 posts were classified by the first

---

<sup>4</sup><http://bit.ly/2LrN7OA>

<sup>5</sup>Watson Cognitive Services

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

2699 author and three other research assistants, software engineers with at least 2 years  
2700 industry experience, assisted to classify the remaining 900. This left a total of 1,375  
2701 classifications made by four people plus an additional 450 classifications made from  
2702 reliability analysis, in which the remaining 50 posts were classified nine times (as  
2703 detailed in Section 6.4.3). Thus, a total of 1,825 classifications were made from the  
2704 original 1,425 posts extracted.

2705 Whilst we could have chosen to employ topic modelling, these are too descriptive  
2706 in nature (as discussed in Section 6.3). Moreover, we wanted to see if prior  
2707 taxonomies can be applied to intelligent services (as opposed to creating a new  
2708 one) and compare if their distributions are similar. Therefore, we applied the two  
2709 existing taxonomies described in Section 6.3 to each post; (i) a documentation-  
2710 specific taxonomy that addresses issues directly resulting from documentation, and  
2711 (ii) a generalised taxonomy that covers a broad range of SO issues in a well-defined  
2712 SE area (specifically mobile app development). Aghajani et al.’s documentation-  
2713 specific taxonomy (Taxonomy A) is multi-layered consisting of four dimensions and  
2714 16 sub-categories [3]. Similarly, Beyer’s SO generalised post classification taxon-  
2715 omy (Taxonomy B) consists of seven dimensions [31]. We code each dimension  
2716 with a number,  $X$ , and each sub-category with a letter  $y$ : ( $Xy$ ). We describe both  
2717 taxonomies in detail within Table 6.1. Where a post was included in our results  
2718 but not applicable to intelligent services (see Section 6.4.2) or not applicable to  
2719 a taxonomy dimension/category, then the post was flagged for removal in further  
2720 analysis. Table 6.1 presents *our understanding* of the respective taxonomies; our  
2721 intent is not to methodologically replicate Aghajani et al. or Beyer et al.’s studies in  
2722 the intelligent service domain, rather to acknowledge related work in the area of SO  
2723 classification and reduce the need to synthesise a new taxonomy. We baseline all  
2724 coding against *our interpretation only*. Our classifications are therefore independent  
2725 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	■ <b>Information Content (What)</b> .....
A-1a	↔ <i>Correctness</i> .....
A-1b	↔ <i>Completeness</i> .....
A-1c	↔ <i>Up-to-dateness</i> .....
A-2	■ <b>Information Content (How)</b> .....
A-2a	↔ <i>Maintainability</i> .....
A-2b	↔ <i>Readability</i> .....
A-2c	↔ <i>Usability</i> .....
A-2d	↔ <i>Usefulness</i> .....
A-3	■ <b>Process-Related</b> .....
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	■ <b>Tool-Related</b> .....
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	■ <b>API usage</b> .....
B-2	■ <b>Discrepancy</b> .....
B-3	■ <b>Errors</b> .....
B-4	■ <b>Review</b> .....
B-5	■ <b>Conceptual</b> .....
B-6	■ <b>API change</b> .....
B-7	■ <b>Learning</b> .....

**2726 6.4.3 Data Analysis**

2727 **Reliability of Classification** To measure consistency of the categories assigned  
2728 by each rater to each post, we utilised both intra- and inter-rater reliability [190].  
2729 As verbatim descriptions from dimensions and sub-categories were considered quite  
2730 lengthy from their original sources, all raters met to agree on a shared interpretation of  
2731 the descriptions, which were then paraphrased as discussed in the previous subsection  
2732 and tabulated in Table 6.1. To perform statistical calculations of reliability, each  
2733 category was assigned a nominal value and a random sample of 50 posts were  
2734 extracted. Two-phase reliability analysis followed.

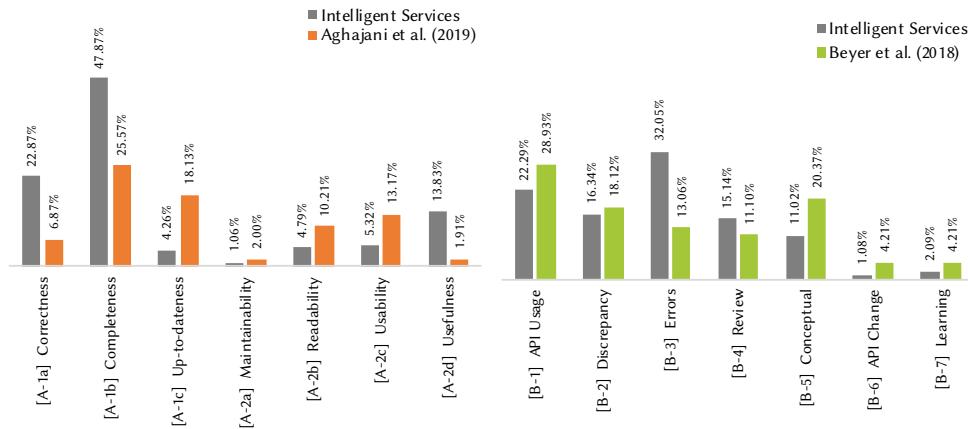
2735 Firstly, intra-rater agreement by the first author was conducted twice on 28 June  
2736 2019 and 9 August 2019. Secondly, inter-rater agreement was conducted with the  
2737 remaining four co-authors in addition to three research assistants within our research  
2738 group in mid-August 2019. Thus, the 50 posts were classified an additional nine  
2739 times, resulting in 450 classifications for reliability analysis. We include these  
2740 classifications in our overall analysis.

2741 At first, we followed methods of reliability analysis similar to previous SO  
2742 studies (e.g., [281]) using the percentage agreement metric that divides the number  
2743 of agreed categories assigned per post by the total number of raters [190]. However,  
2744 percentage agreement is generally rejected as an inadequate measure of reliability  
2745 analysis [62, 118, 160] in statistical communities. As we used more than 2 coders  
2746 and our reliability analysis was conducted under the same random sample of 50  
2747 posts, we applied *Light's Kappa* [171] to our ratings, which indicates an overall  
2748 index of agreement. This was done using the `irr` computational R package [103]  
2749 as suggested in [118].

2750 **Distribution Analysis** In order to compare the distribution of categories from our  
2751 study with previous studies we carried out a  $\chi^2$  test. We selected a  $\chi^2$  test as the  
2752 following assumptions [264] are satisfied: (i) the data is categorical, (ii) all counts are  
2753 greater than 5, and (iii) we can assume simple random sampling. The null hypothesis  
2754 describes the case where each population has the same proportion of observations  
2755 and the alternative hypothesis is where at least one of the null hypothesis statements  
2756 is false. We chose a significance value,  $\alpha$ , of 0.05 following a standard rule of  
2757 thumb. As to the best of our knowledge this is the first statistical comparison using  
2758 Taxonomy A and B on SO posts. To report the effect size we selected Cramer's Phi,  
2759  $\phi_c$  which is well suited for use on nominal data [264].

**2760 6.5 Findings**

2761 We present our findings from classifying a total of 1,825 SO posts aimed at answering  
2762 RQs 1 and 2. 450 posts were classified using Taxonomies A and B for reliability  
2763 analysis as described in Section 6.4.3 and the remaining 1,375 posts were classified  
2764 as per Section 6.4.2. A summary of our classification using Taxonomies A and B is  
2765 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).

### 2766 6.5.1 Post classification and reliability analysis

2767 When undertaking the classification, we found that 238 issues (13.04%) did not  
 2768 relate to intelligent services directly. For example, library dependencies were still  
 2769 included in a number of results (see Section 6.4.2), and we found there to be many  
 2770 posts discussing Android’s Mobile Vision API as Google (Cloud) Vision. These  
 2771 issues were flagged and ignored for further analysis (see Section 6.4.2).

2772 For our reliability analysis, we classified a total of 450 posts of which 70 posts  
 2773 were flagged as irrelevant. Landis and Koch [166] provide guidelines to interpret  
 2774 kappa reliability statistics, where  $0.00 \leq \kappa \leq 0.20$  indicates *slight* agreement and  
 2775  $0.21 \leq \kappa \leq 0.40$  indicates *fair* agreement. Despite all raters meeting to agree on a  
 2776 shared interpretation of the taxonomies (see Section 6.4.3) our inter-rater measures  
 2777 aligned *slightly* (0.148) for Taxonomy A and *fairly* (0.295) for Taxonomy B. We  
 2778 report further in Section 6.7.

### 2779 6.5.2 Developer Frustrations

2780 We found Beyer et al.’s high-level abstraction taxonomy (Taxonomy B) was able to  
 2781 classify 86.52% of posts. 10.30% posts were assigned exclusively under Aghajani  
 2782 et al.’s documentation-specific taxonomy (Taxonomy A). We found that developers  
 2783 do not generally ask questions exclusive to documentation, and typically either  
 2784 pair documentation-related issues to their own code or context. The following two  
 2785 subsections further explain results from both Taxonomy A and B’s perspective.

2786 **Results from Aghajani et al.’s taxonomy** Results for Aghajani et al.’s low-level  
 2787 documentation taxonomy (Taxonomy A), indicates that most discussion on SO does  
 2788 not directly relate to documentation about an intelligent service. We did not find  
 2789 any process-related (A-3) or tool-related (A-4) questions as, understandably, the

2790 developers who write the documentation of the intelligent services would not be  
2791 posting questions of such nature on SO. One can *infer* documentation-related issues  
2792 from posts (i.e., parts of the documentation *lacking* that may cause the issue posted).  
2793 However, there are few questions that *directly* relate to documentation of intelligent  
2794 services.

2795 Few developers question or ask questions directly about the API documentation  
2796 but some (47.87%) posts ask for additional information to understand the  
2797 API (**A-1b**), for example: “*Is there a full list of potential labels*  
2798 *that Google’s Vision API will return?*” [350]; “*There seems to be very little to no*  
2799 *documentation for AWS iOS text recognition inside an image?*” [348].

2800 22.87% of posts question the **accuracy (A-1a)** of certain parts of the cloud docu-  
2801 mentation, especially in relation to incorrect quotas and limitations: “*Are the Cloud*  
2802 *Vision API limits in documentation correct?*” [361], “*According to the Google Vision*  
2803 *documentation, the maximum number of image files per request is 16. Elsewhere,*  
2804 *however, I’m finding that the maximum number of requests per minute is as high as*  
2805 *1800.*” [346].

2806 There are also many references (23.94%) addressing the confusing nature of  
2807 some documentation, indicating that the **readability, usability and usefulness of**  
2808 **the documentation (A-2b, A-2c and A-2d)** could be improved. For example, “*Am*  
2809 *I encoding it correctly? The docs are quite vague.*” [344], “*The aws docs for this*  
2810 *are really confusing.*” [373].

2811 **Results from Beyer et al.’s taxonomy** We found that a majority (32.05%) of  
2812 posts are primarily **error-related questions (B-3)**, including a dump of the stack  
2813 trace or exception message from the service’s programming-language SDK (usually  
2814 Java, Python or C#) that relates to a specific error. For example: “*I can’t fix an*  
2815 *error that’s causing us to fall behind.*” [370]; “*I’m using the Java Google Vision*  
2816 *API to run through a batch of images... I’m now getting a channel closed and*  
2817 *ClosedChannelException error on the request.*” [364].

2818 **API usage questions (B-1)** were the second highest category at 22.29% of  
2819 posts. Reading the questions revealed that many developers present an insufficient  
2820 understanding of the behaviour, functional capability and limitation of these services  
2821 and the need for further data processing. For example, while Azure provides an image  
2822 captioning service, this is not universal to all computer vision services: “*In Amazon*  
2823 *Rekognition for image processing how do I get the caption for an image?*” [355].  
2824 Similarly, OCR-related and label-related questions often indicate interest in cross-  
2825 language translation, where a separate translation service would be required: “*Can*  
2826 *Google Cloud Vision generate labels in Spanish via its API?*” [369]; “[*How can*  
2827 *I] specify language for response in Google Cloud Vision API*” [356]; “*When I*  
2828 *request a text detection of an image, it gives only English Alphabet characters*  
2829 *(characters without accents) which is not enough for me. How can I get the UTF-32*  
2830 *characters?*” [351].

2831 It was commonplace to see questions that demonstrate a lack of depth in under-  
2832 standing and appreciating how these services work, instead posting simple debugging  
2833 questions. For instance, in the 11.02% of **conceptual-related questions (B-5)** that

<sup>2834</sup> we categorised, we noticed causal links to a misunderstanding (or lack of aware-  
<sup>2835</sup> ness) of the vocabulary used within computer vision. For example: “*The problem  
<sup>2836</sup> is that I need to know not only what is on the image but also the position of that  
<sup>2837</sup> object. Some of those APIs have such feature but only for face detection.*” [362];  
<sup>2838</sup> “*I want to know if the new image has a face similar to the original image.... [the  
<sup>2839</sup> service] can identify faces, but can I use it to get similar faces to the identified face  
<sup>2840</sup> in other images?*” [354]. It is evident that some application developers are not aware  
<sup>2841</sup> of conceptual differences in computer vision such as object/face *detection* versus  
<sup>2842</sup> *localisation* versus *recognition*.

<sup>2843</sup> In the 16.34% of **discrepancy-related questions (B-2)**, we see further unaware-  
<sup>2844</sup> ness from developers in how the underlying systems work. In OCR-related questions,  
<sup>2845</sup> developers do not understand the pre-processing steps required before an OCR is  
<sup>2846</sup> performed. In instances where text is separated into multiple columns, for example,  
<sup>2847</sup> text is read top-down rather than left-to-right and segmentation would be required  
<sup>2848</sup> to achieve the expected results. For example, “*it appears that the API is using some  
<sup>2849</sup> kind of logic that makes it scan top to bottom on the left side and moving to right  
<sup>2850</sup> side and doing a top to bottom scan.*” [368]; “*this method returns scanned text in  
<sup>2851</sup> wrong sequence... please tell me how to get text in proper sequence.*” [374].

<sup>2852</sup> A number of **review-related questions (B-4)** (15.14%) seem to provide some  
<sup>2853</sup> further depth in understanding the context to which these systems work, where train-  
<sup>2854</sup> ing data (or training stages) are needed to understand how inferences are made: “*How  
<sup>2855</sup> can we find an exhaustive list (or graph) of all logos which are effectively recognized  
<sup>2856</sup> using Google Vision logo detection feature?*” [372]; “*when object banana is detected  
<sup>2857</sup> with accuracy greater than certain value, then next action will be dispatched... how  
<sup>2858</sup> can I confidently define and validate the threshold value for each item?*” [358].

<sup>2859</sup> **API change (B-6)** was shown in 1.08% of posts, with evolution of the services  
<sup>2860</sup> occurring (e.g., due to new training data) but not necessarily documented “*Recently  
<sup>2861</sup> something about the Google Vision API changed... Suddenly, the API started to  
<sup>2862</sup> respond differently to my requests. I sent the same picture to the API today, and I  
<sup>2863</sup> got a different response (from the past).*” [371].

### <sup>2864</sup> 6.5.3 Statistical Distribution Analysis

<sup>2865</sup> We obtained the following results  $\chi^2 = 131.86$ ,  $\alpha = 0.05$ ,  $p \text{ value} = 2.2 \times 10^{-16}$  and  
<sup>2866</sup>  $\phi_c = 0.362$  from our distribution analysis with Taxonomy A to compare our study  
<sup>2867</sup> with that of Aghajani et al. [3]. Comparing our study to Beyer et al. [31] produced the  
<sup>2868</sup> following results  $\chi^2 = 145.58$ ,  $\alpha = 0.05$ ,  $p \text{ value} = 2.2 \times 10^{-16}$  and  $\phi_c = 0.252$ .  
<sup>2869</sup> These results show that we are able to reject the null hypothesis that the distribution  
<sup>2870</sup> of posts using each taxonomy was the same as the comparison study. While there are  
<sup>2871</sup> limited guidelines for interpreting  $\phi_c$  when there is no prior information for effect  
<sup>2872</sup> size [276], Sun et al. suggests the following:  $0.07 \leq \phi_c \leq 0.20$  indicates a *small*  
<sup>2873</sup> effect,  $0.21 \leq \phi_c \leq 0.35$  indicates a *medium* effect, and  $0.35 > \phi_c$  indicates a *large*  
<sup>2874</sup> effect. Based on this criteria we obtained a *large* effect size for the documentation-  
<sup>2875</sup> specific classification (Taxonomy A) and a *medium* effect size for the generalised  
<sup>2876</sup> classification (Taxonomy B).

**2877 6.6 Discussion****2878 6.6.1 Answers to Research Questions**

2879 **RQ1. How do developers mis-comprehend intelligent services as presented**  
2880 **within Stack Overflow pain-points?** Upon meeting to discuss the discrepancies  
2881 between our categorisation of intelligent service usage SO posts, we found that  
2882 our interpretations of the *posts themselves* were largely subjective. For example,  
2883 many posts presented multi-faceted dimensions for Taxonomy B; Beyer et al. [31]  
2884 argue that a post can have more than one question category and therefore multi-label  
2885 classification is appropriate at times. We highlight this further in the threats to  
2886 validity (Section 6.7).

2887 We have to define the context of intelligent services to address RQ1. We use  
2888 the concept of a “technical domain” [17] to define this context. A technical domain  
2889 captures the domain-specific concerns that influence the non-functional requirements  
2890 of a system [17]. In the context of intelligent services, the technical domain includes  
2891 exploration, data engineering, distributed infrastructure, training data, and model  
2892 characteristics as first class citizens [17]. We would then expect to see posts on SO  
2893 related to these core concerns.

2894 In Figure 6.3, for the documentation-specific classification, the majority of posts  
2895 were classified as **Completeness (A1-b)** related (47.87%). An interpretation for this  
2896 is that the documentation does not adequately cover the technical domain concerns.  
2897 Comments by developers such as “*I'm searching for a list of all the possible image*  
2898 *labels that the Google Cloud Vision API can return?*” [349] indicates the documen-  
2899 *tation does not adequately describe the training data for the API—developers do*  
2900 *not know the required usage assumptions.* Another quote from a developer, “*Can*  
2901 *Google Cloud Vision generate labels in Spanish via its API? ... [Does the API]*  
2902 *allow to select which language to return the labels in?*” [369] points to a lack of  
2903 details relating to the characteristics of the models used by the API. It would seem  
2904 that developers are unaware of aspects of the technical domain concerns.

2905 The next most frequent category is **Correctness (A-1a)** with 22.87% of posts. In  
2906 the context of the technical domain there are many limits that developers need to be  
2907 aware of: range and increments of a model score [69]; required data pre-processing  
2908 steps for optimal performance; and features provided by the models (as explained  
2909 in Section 6.5.2). Considering the relation between technical concerns and software  
2910 quality, developers are right to question providers on correctness; “*Are the Cloud*  
2911 *Vision API limits in documentation correct?*” [361].

2912 **RQ2. Are the distribution of issues similar to prior studies?** Visual inspection of  
2913 Figure 6.3 shows that the distributions for the documentation-specific classification  
2914 and the generalised classification are different (compared to prior studies). As a  
2915 sanity check we conducted a  $\chi^2$  test and calculated the effect size  $\phi_c$ . We were able  
2916 to reject the null hypothesis for both classification schemes, that the distribution of  
2917 issues were the same as the previous studies (see Section 6.5). We now discuss the  
2918 most prominent differences between our study and the previous studies.

2919 In the context of intelligent service SO posts, Taxonomy B suggests that Errors  
2920 (B-3) are discussed most amongst developers. These results are in contrast to similar

2921 studies made in more *mature* API domains, such as Mobile Development [18, 19,  
2922 30, 31, 247] and Web Development [286]. Here, API Usage (B-1) is much more  
2923 frequently discussed, followed by Conceptual (B-5), Discrepancy (B-2) and Errors  
2924 (B-3). We argue in the following section that an improved developer understanding  
2925 can be achieved by educating them about the intelligent service lifecycle and the  
2926 ‘whole’ system that wraps such services.

2927 In the Android study API usage questions (B-1) were the highest category  
2928 (28.93% compared to 22.29% in our study). As stated in the analysis of the Error  
2929 questions this discrepancy could be due to the maturity of the domain. However,  
2930 another explanation could be the scope of the two individual studies. Beyer et al. [31]  
2931 used a broad search strategy consisting of posts tagged Android. This search term  
2932 fetches issues related to the entire Android platform which is significantly larger  
2933 than searching for computer vision APIs using 229 search terms. As a consequence  
2934 of more posts and more APIs there would be use cases resulting in additional posts  
2935 related to API Usage (B-1).

2936 Applying existing SO taxonomies allowed us to better understand the distribution  
2937 of the issues across different domains. In particular, the issues raised around  
2938 intelligent services appear to be primarily due to poor documentation, or insufficient  
2939 explanation around errors and limitations. Hence, many of the concerns could be  
2940 addressed by adding more details to the end-point descriptions, and by providing  
2941 additional information around how these services are designed to work.

### 2942 **6.6.2 The Developer’s Learning Approach**

2943 In this subsection, we offer an explanation as to why developers are complaining  
2944 about certain things when trying to use intelligent services on SO (RQ1), as char-  
2945 acterised through the use of prior SO classification frameworks (RQ2). This is  
2946 described through the theoretical lenses of two learning taxonomies: Bloom’s con-  
2947 text complexity and intellectual ability taxonomy, and the SOLO taxonomy (i.e., the  
2948 nature by which developer’s learn). We argue that the issues with using intelligent  
2949 services relating to the lower-levels of these learning taxonomies are easily solvable  
2950 by slight fixes and improvements to the documentation of these services. However,  
2951 the higher dimensions of these taxonomies demand far more rigorous mitigation  
2952 strategies than documentation alone (potentially more structured education). Thus,  
2953 many of the questions posted are from developers who are *learning to understand*  
2954 the domain of intelligent services and AI, and (hence) both SOLO and Bloom’s tax-  
2955 onomies are applicable for this discussion—as described below within the context  
2956 of our domain—as pedagogical aides.

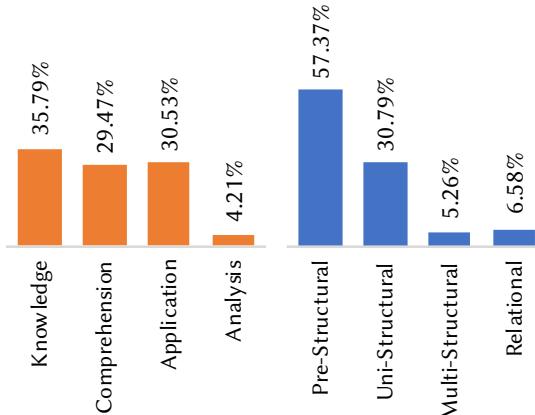
2957 **Bloom’s Taxonomy** The cognitive domain under Bloom’s taxonomy [34] consists  
2958 of six objectives. Within the context of intelligent services, developers are likely  
2959 to ask questions due to causal links that exist in the following layers of Bloom’s  
2960 taxonomy: (i) *knowledge*, where the developer does not remember or know of the  
2961 basic concepts of computer vision and AI (in essence, they may think that AI is as  
2962 smart as a human); (ii) *comprehension*, where the developer does not understand

2963 how to interpret basic concepts, or they are mis-understanding how they are used  
 2964 in context; (iii) *application*, where the developer is struggling to apply existing  
 2965 concepts within the context of their own situation; (iv) *analysis*, where the developer  
 2966 is unable to analyse the results from intelligent services (i.e., understand response  
 2967 objects); (v) *evaluation*, where the developer is unable to evaluate issues and make  
 2968 use of best-practices when using intelligent services; and (vi) *synthesise*, where  
 2969 the developer is posing creative questions to ask if new concepts are possible with  
 2970 computer vision services.

2971 **SOLO Taxonomy** The SOLO taxonomy [32] consists of five levels of under-  
 2972 standing. The causal links behind the SO questions we have found relate to the  
 2973 following layers of the SOLO taxonomy: (i) *pre-structural*, where the developer has  
 2974 a question indicating incompetence or has little understanding of computer vision;  
 2975 (ii) *uni-structural*, where the developer is struggling with one key aspect (i.e., a  
 2976 simple question about computer vision); (iii) *multi-structural*, where the developer  
 2977 is questioning multiple concepts (independently) to understand how to build their  
 2978 system (e.g., system integration with the intelligent service); (iv) *relational*, where  
 2979 the developer is comparing and contrasting the best ways to achieve something with  
 2980 intelligent services; and (v) *extended abstract*, where the developer poses a question  
 2981 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.” [365]	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).” [345]	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?” [352]	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?” [359]	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?” [357]	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.

2982 **Aligning SO taxonomies to Bloom’s and SOLO taxonomies** To understand our  
2983 findings with the lenses of pedagogical aids, we aligned Taxonomies A and B to  
2984 Bloom’s and the SOLO taxonomies for a random sample of 50 issues described in  
2985 Section 6.4.3. To do this, we reviewed all 50 of these SO posted questions and  
2986 applied both the Bloom and SOLO taxonomies. The primary author assigned each  
2987 of the 50 questions a level within the Bloom and SOLO taxonomies, removed out  
2988 noise (i.e., false positive posts of no relevance to intelligent services) and unassigned  
2989 dimensions from reliability agreement, and then compared the relevant dimensions of  
2990 Taxonomy A and B dimensions (not sub-categories). The comparison of alignments  
2991 of posts to the five SOLO dimensions and six Bloom dimensions are shown in  
2992 Figure 6.4. We acknowledge that this is only an approximation of the current state of  
2993 the developer’s understanding of intelligent services. This early model will require  
2994 further studies to perform a more thorough analysis, but we offer this interpretation  
2995 for early discussion.

2996 As shown in Figure 6.4, the bulk of the posts fall in the lower constructs of  
2997 Bloom’s and the SOLO taxonomy. This indicates that modification to certain doc-  
2998 umentation aspects can address many of these issues. For example, many issues  
2999 can be ratified with better descriptions of response data and error messages: “*I was*  
3000 *exploring google vision and in the specific function ‘detectCrops’, gives me the crop*  
3001 *hints. what does this means exactly?*” [360]; “*I am a making a very simple API call*  
3002 *to the Google Vision API, but all the time it’s giving me error that ‘google.oauth2’*  
3003 *module not found.*” [375]

3004 However, and more importantly, the higher-construct questions ranging from  
3005 the middle of the third dimensions on are not as easily solvable through improved  
3006 documentation (i.e., apply and multi-structural) which leaves 34.74% (Bloom’s)  
3007 and 11.84% (SOLO) unaccounted for, resolvable only through improved education  
3008 practices.

### 3009 6.6.3 Implications

3010 **Researchers** (i) *Investigate the evolution of post classification:* Analysing how the  
3011 distribution of the reported issues changes over time would be an important study.  
3012 This study could answer questions such as ‘*Does the evolution of intelligent services*  
3013 *follow the same pattern as previous software engineering trends such as mobile app*  
3014 *or web development?*’ As with any new emerging field, it is key to analyse how  
3015 developers perceive such issues over time. For instance, early issues with web or  
3016 mobile app development matured as their respective domain matured, and we would  
3017 expect similar results to occur in the intelligent services space. Future researchers  
3018 could plan for a longitudinal study, such as a long-term survey with developers to  
3019 gather their insights in this evolving domain, reviewing case studies of projects that  
3020 use intelligent web services from now into the future, or re-mining SO at a later date  
3021 and comparing the results to this study. This will help assess evolving trends and  
3022 characteristics, and determine how and if the nature of the developer’s experience  
3023 with intelligent services (and AI in general) changes with time. (ii) *Investigate the*  
3024 *impact of technical challenges on API usage:* As discussed above, intelligent services  
3025 have characteristics that may influence API usage patterns and should be investigated  
3026 as a further avenue of research. Further mining of open source software repositories  
3027 that make use of intelligent services could be assessed, thereby investigating if API  
3028 patterns evolve with the rise of AI-based applications.

3029 **Educators** (i) *Education on high-level aspects of intelligent services:* As demon-  
3030 strated in our analysis of their SO posts, many developers appear to be unaware of  
3031 the higher-level concepts that exist within the AI and ML realm. This includes the  
3032 need to pre- and post-process data, the data dependency and instability that exists in  
3033 these services, and the specific algorithms that empower the underlying intelligence  
3034 and hence their limitations and characteristics. However, most developers don’t  
3035 seem to complain about these factors due to the lack of documentation (i.e., via  
3036 Taxonomy A). Rather, they are unaware that such information should be documen-  
3037 tation and instead ask generalised and open questions (i.e., via Taxonomy B). Thus,  
3038 documentation improvements alone may not be enough to solve these issues. This  
3039 results in uncertainty during the preparation and operation (usage) of such services.  
3040 Such high-level conceptual information is currently largely missing in developer  
3041 documentation for intelligent services. Furthermore, many of the background ML  
3042 and AI algorithm information needed to understand and use intelligent systems in  
3043 context are built within data science (not SE) communities. A possible road-map to  
3044 mitigate this issue would be the development of a software engineer’s ‘crash-course’  
3045 in ML and AI. The aim of such a course would encourage software engineers to  
3046 develop an appreciation of the nuances and the inherent risks and implications that  
3047 comes with using intelligent services. This could be taught at an undergraduate  
3048 level to prepare the next generation of developers of a ‘programming 2.0’ era. How-  
3049 ever, the key aspects and implications that are presented with AI would need to be  
3050 well-understood before such a course is developed, and determining the best strategy  
3051 to curate the content to developers would be best left to the SE education domain.

3052 Further investigation in applying educational taxonomies in the area (such as our  
3053 attempts to interpret our findings using Bloom's and the SOLO taxonomies) would  
3054 need to be thoroughly explored beforehand.

3055 **Software Engineers** *(i) Better understanding of intelligent API contextual usage:*  
3056 Our results show that developers are still learning to use these APIs. We applied  
3057 two learning perspectives to interpret our results. In applying the two pedagogical  
3058 taxonomies to our findings, we see that most issues seem to fall into the pre-structural  
3059 and knowledge-based categories; little is asked of higher level concepts and a ma-  
3060 jority of issues do not offer complex analysis from developers. This suggests that  
3061 developers are struggling as they are unaware of the vocabulary needed to actually  
3062 use such APIs, further reinforcing the need for API providers to write overview  
3063 documentation (as noted in prior work [68]) and not just simple endpoint documen-  
3064 tation. This said, improved documentation isn't always enough—as suggested by  
3065 our discussion in Section 6.6.2, software engineers should explore further education  
3066 to attain a greater appreciation of the nuances of ML when attempting to use these  
3067 services.

3068 **Intelligent Service Providers** *(i) Clarify use cases for intelligent services:* In-  
3069 specting SO posts revealed that there is a level of confusion around the capabilities  
3070 of different intelligent services. This needs to be clarified in associated API doc-  
3071 umentation. The complication with this comes with targeting the documentation  
3072 such that software developers (who are untrained in the nuances of AI and ML as  
3073 per Section 6.6.3) can to digest it and apply it in-context to application develop-  
3074 ment. *(ii) Technical domain matters:* More needs to be provided than a simple  
3075 endpoint description as conventional APIs offer by describing the whole framework  
3076 by which the endpoint sits, giving further context. This said, compared to traditional  
3077 APIs, we find that developers complain less about the documentation and more  
3078 about shallower issues. All expected pre-processing and post-processing needs to be  
3079 clearly explained. A possible mitigation to this could be an interactive tutorial that  
3080 helps developers fully understand the technical domain using a hands-on approach.  
3081 For example, websites offer interactive Git tutorials<sup>7</sup> to help developers understand  
3082 and explore the technical domain matters under version control in their own pace.  
3083 *(iii) Clarify limitations:* API developers need to add clear limitations of the existing  
3084 APIs. Limitations include list of objects that can be returned from an endpoint. We  
3085 found that the cognitive anchors of how existing, conventional API documentation  
3086 is written has become ‘ported’ to the computer vision realm, however a lot more  
3087 overview documentation than what is given at present (i.e., better descriptions of  
3088 errors, improved context of how these systems work in etc.) needs to be given. Such  
3089 documentation could be provided using interactive tutorials.

---

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

## 3090 6.7 Threats to Validity

3091 *Construct validity:* Some questions extracted from SO produced false positives, as  
3092 mentioned in Section 6.4.2 and Section 6.5. However, all non-relevant posts were  
3093 marked as noise for our study, and thus did not affect our findings. Moreover, SO is  
3094 known to have issues where developers simply ask basic questions without looking  
3095 at the actual documentation where the answer exists. Such questions, although  
3096 down-voted, were still included in our data-set analysis, but as these were so few, it  
3097 does not have a substantial impact on categorised posts.

3098 *Internal validity:* As detailed in Section 6.4.3, Taxonomies A and B present  
3099 slight and fair agreement, respectively, when inter-rater reliability was applied. The  
3100 nature of our disagreements largely fell due to the subjectivity in applying either  
3101 taxonomies to posts. Despite all coders agreeing to the shared interpretation of  
3102 both taxonomies, both taxonomies are subjective in their application, which was  
3103 not reported by either Aghajani et al. or Beyer et al.. In many cases, multi-  
3104 label classification seemed appropriate, however both taxonomies use single-label  
3105 mapping which we find results in too much subjectivity. This subjectivity, therefore,  
3106 ultimately adversely affects IRR analysis. Thus, a future mitigation strategy for  
3107 similar work should explore multi-label classification to avoid this issue; Beyer  
3108 et al., for example, plan for multi-label classification as future work. However,  
3109 these studies would need to consider the statistical challenges in calculating multi-  
3110 rater, multi-label IRR for thorough reliability analysis in addressing subjectivity.  
3111 The selection of SO posts used for our labelling, chiefly in the subjectivity of our  
3112 classifications, is of concern. We mitigate this by an extensive review process  
3113 assessing the reliability of our results as per Section 6.4.3. The classification of our  
3114 posts into the SOLO and Bloom's taxonomies was performed by the primary author  
3115 only, and therefore no inter-rater reliability statistics were performed. However, we  
3116 used these pedagogy related taxonomies as a lens to gain an additional perspective to  
3117 interpret our results. Future studies should attempt a more rigorous analysis of SO  
3118 posts using Bloom's and SOLO taxonomies. We only aligned posts to one category  
3119 for each taxonomy and did not align these using multi-label classification. This  
3120 brings more complexity to the analysis, and our attempts to repeat prior studies'  
3121 methodologies (see Section 6.3). Multi-label classification for intelligent services  
3122 SO posts is an avenue for future research.

3123 *External validity:* While every effort was made to select posts from SO relevant  
3124 to computer vision services, there are some cases where we may have missed some  
3125 posts. This is especially due to the case where some developers mis-reference  
3126 certain intelligent services under different names (see Section 6.4.2). Our SOLO  
3127 and Bloom's taxonomy analysis has only been investigated through the lenses of  
3128 intelligent services, and not in terms of conventional APIs (e.g., Andriod APIs).  
3129 Therefore, we are not fully certain how these results found would compare to other  
3130 types of APIs. Two *existing* SO classification taxonomies were used rather than  
3131 developing our own. We wanted to see if previous SO taxonomies could be applied to  
3132 intelligent services before developing a new, specific taxonomy, and these taxonomies  
3133 were applied based on our interpretation (see Section 6.4.2) and may not necessarily

reflect the interpretation of the original authors. Moreover, automated techniques such as topic modelling were not utilised as we found these produce descriptive classifications only (see Section 6.3). Hence, manual analysis was performed by humans to ensure categories could be aligned back to causal factors. Only English-speaking intelligent services were selected; the applicability of our analysis to other, non-English speaking services may affect results. Use of computer vision in this study is an illustrative example to focus on one area of the intelligent services spectrum. While our narrow scope helps us obtain more concrete findings, we suggest that wider issues exist in other intelligent service domains may affect the generalisability of this study, and suggest future work be explored in this space.

## 6.8 Conclusions

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

# CHAPTER 7

3162

---

3163

3164

## Ranking Computer Vision Service Issues using Emotion<sup>†</sup>

3165

---

---

<sup>†</sup>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.



# CHAPTER 8

3166

3167

---

## 3168 Using a Facade Pattern to combine Computer Vision Services<sup>†</sup>

---

3169

3170 **Abstract** Intelligent computer vision services, such as Google Cloud Vision or Amazon  
3171 Rekognition, are becoming evermore pervasive and easily accessible to developers to build  
3172 applications. Because of the stochastic nature that ML entails and disparate datasets used in  
3173 their training, the outputs from different computer vision services varies with time, resulting  
3174 in low reliability—for some cases—when compared against each other. Merging multiple  
3175 unreliable API responses from multiple vendors may increase the reliability of the overall  
3176 response, and thus the reliability of the intelligent end-product. We introduce a novel  
3177 methodology—inspired by the proportional representation used in electoral systems—to  
3178 merge outputs of different intelligent computer vision API provided by multiple vendors.  
3179 Experiments show that our method outperforms both naive merge methods and traditional  
3180 proportional representation methods by 0.015 F-measure.

### 3181 8.1 Introduction

3182 With the introduction of intelligent web services (IWSs) that make machine learning  
3183 (ML) more accessible to developers [238, 295], we have seen a large growth of  
3184 intelligent applications dependent on such services [49, 109]. For example, consider  
3185 the advances made in computer vision, where objects are localised within an image  
3186 and labelled with associated categories. Cloud-based computer vision services  
3187 (CVSs)—e.g., [318, 321, 329, 331, 334, 335, 339, 377]—are a subset of IWSs.  
3188 They utilise ML techniques to achieve image recognition via a remote black-box  
3189 approach, thereby reducing the overhead for application developers to understand  
3190 how to implement intelligent systems from scratch. Furthermore, as the processing

---

<sup>†</sup>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.

3191 and training of the machine-learnt algorithms is offloaded to the cloud, developers  
3192 simply send RESTful API requests to do the recognition. There are, however, inherit  
3193 differences and drawbacks between traditional web services and IWSs, which we  
3194 describe with the motivating scenario below.

### 3195 8.1.1 Motivating Scenario: Intelligent vs Traditional Web Services

3196 An application developer, Tom, wishes to develop a social media Android and iOS  
3197 app that catalogues photos of him and his friends, common objects in the photo,  
3198 and generates brief descriptions in the photo (e.g., all photos with his husky dog,  
3199 all photos on a sunny day etc.). Tom comes from a typical software engineering  
3200 background with little knowledge of computer vision and its underlying concepts.  
3201 He knows that intelligent computer vision web APIs are far more accessible than  
3202 building a computer vision engine from scratch, and opts for building his app using  
3203 these cloud services instead.

3204 Based on his experiences using similar cloud services, Tom would expect consistency  
3205 of the results from the same API and different APIs that provide the same (or  
3206 similar) functionality. As an analogy, when Tom writes the Java substring method  
3207 "doggy".substring(0, 2), he expects it to be the same result as the Swift equivalent  
3208 "doggy".prefix(3). Each and every time he interacts with the substring  
3209 method using either API, he gets "dog" as the response. This is because Tom is  
3210 used to deterministic, rule-driven APIs that drive the implementation behind the  
3211 substring method.

3212 Tom's deterministic mindset results in three key differentials between a traditional  
3213 web service and an IWS:

3214 **(1) Given similar input, results differ between similar IWSs.** When Tom  
3215 interacts with the API of an IWS, he is not aware that each API provider trains  
3216 their own, unique ML model, both with disparate methods and datasets. These  
3217 IWSs are, therefore, nondeterministic and data-driven; input images—even  
3218 if they contain the same conceptual objects—often output different results.  
3219 Contrast this to the substring method of traditional APIs; regardless of what  
3220 programming language or string library is used, the same response is expected  
3221 by developers.

3222 **(2) Intelligent responses are not certain.** When Tom interprets the response  
3223 object of an IWS, he finds that there is a ‘confidence’ value or ‘score’. This  
3224 is because the ML models that power IWSs are inherently probabilistic and  
3225 stochastic; any insight they produce is purely statistical and associational [224].  
3226 Unlike the substring example, where the rule-driven implementation provides  
3227 certainty to the results, this is not guaranteed for IWSs. For example, a picture  
3228 of a husky breed of dog is misclassified as a wolf. This could be due to  
3229 adversarial examples [278] that ‘trick’ the model into misclassifying images  
3230 when they are fully decipherable to humans. It is well-studied that such  
3231 adversarial examples exist in the real world unintentionally [91, 163, 227].

3232 **(3) Intelligent APIs evolve over time.** Tom may find that responses to processing  
3233 an image may change over time; the labels he processes in testing may evolve

3234 and therefore differ to when in production. In traditional web services, evo-  
3235 lution in responses is slower, generally well-communicated, and usually rare  
3236 (Tom would always expect "dog" to be returned in the substring example).  
3237 This has many implications on software systems that depend on these APIs,  
3238 such as confidence in the output and portability of the solution. Currently, if  
3239 Tom switches from one API provider to another, or if he doesn't regularly test  
3240 his app in production, he may begin to see a very different set of labels and  
3241 confidence levels.

3242 **8.1.2 Research Motivation**

3243 These drawbacks bring difficulties to the intended API users like Tom. We identify a  
3244 gap in the software engineering literature regarding such drawbacks, including: lack  
3245 of best practices in using IWSs; assessing and improving the reliability of APIs for  
3246 their use in end-products; evaluating which API is suitable for different developer  
3247 and application needs; and how to mitigate risk associated with these APIs. We  
3248 focus on improving reliability of CVSs for use in end-products. The key research  
3249 questions in this paper are:

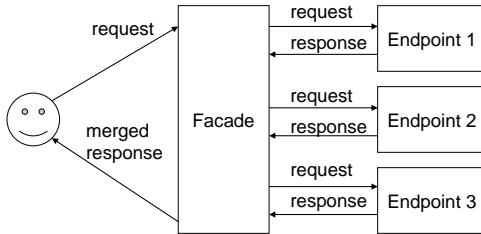
- 3250 **RQ1:** Is it possible to improve reliability by merging multiple CVS results?  
3251 **RQ2:** Are there better algorithms for merging these results than currently in  
3252 use?

3253 Previous attempts at overcoming low reliability include triple-modular redundan-  
3254 tancy [180]. This method uses three modules and decides output using majority  
3255 rule. However, in CVSs, it is difficult to apply majority rule: these APIs respond with  
3256 a list of labels and corresponding scores. Moreover, disparate APIs ordinarily output  
3257 different results. These differences make it hard to apply majority rule because the  
3258 type of outputs are complex and disparate APIs output different results for the same  
3259 input. Merging search results is another technique to improve reliability [265]. It  
3260 normalises scores of different databases using a centralised sample database. Nor-  
3261 malising scores makes it possible to merge search results into a single ranked list.  
3262 However, search responses are disjoint, whereas they are not in the context of most  
3263 CVSs.

3264 In this paper, we introduce a novel method to merge responses of CVSs, using  
3265 image recognition APIs endpoints as our motivating example. Section 8.2 describes  
3266 naive merging methods and requirements. Section 8.3 gives insights into the struc-  
3267 ture of labels. Section 8.4 introduces our method of merging computer vision labels.  
3268 Section 8.5 compares precision and recall for each method. Section 8.6 presents  
3269 conclusions and future work.

3270 **8.2 Merging API Responses**

3271 Image recognition APIs have similar interfaces: they receive a single input (image)  
3272 and respond with a list of labels and associated confidence scores. Similarly, other  
3273 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.

3274 natural language processing [336, 378]). It is difficult to apply majority rule on such  
3275 disparate, complex outputs. While the outputs by *multiple* AI-based API endpoints  
3276 is different and complex, the general format of the output is the same: a list of labels  
3277 and associated scores.

### 3278 8.2.1 API Facade Pattern

3279 To merge responses from multiple APIs, we introduce the notion of an API facade.  
3280 It is similar to a metasearch engine, but differs in their external endpoints. The  
3281 facade accepts the input from one API endpoint (the facade endpoint), propagates  
3282 that input to all user-registered concrete (external) API endpoints simultaneously,  
3283 then ‘merges’ outputs from these concrete endpoints before sending this merged  
3284 response to the API user. We demonstrate this process in Figure 8.1.

3285 Although the model introduces more time and cost overhead, both can be miti-  
3286 gated by caching results. On the other hand, the facade pattern provides the following  
3287 benefits:

- 3288 • **Easy to modify:** It requires only small modifications to applications, e.g.,  
3289 changing each concrete endpoint URL.
- 3290 • **Easy to customise:** It merges results from disparate and concrete APIs ac-  
3291 cording to the user’s preference.
- 3292 • **Improves reliability:** It enhances reliability of the overall returned result by  
3293 merging results from different endpoints.

### 3294 8.2.2 Merge Operations

3295 The API facade is applicable to many use cases. However, this paper focuses on  
3296 APIs that output a list of labels and scores, as is the case for CVSs. Merge operations  
3297 involve the mapping of multiple lists and associated scores, produced by multiple  
3298 APIs, to just one list. For instance, a CVS receives a bowl of fruit as the input image  
3299 and outputs the following:

3300 `[[‘apple’, 0.9], [‘banana’, 0.8]]`

3301 where the first item is the label and the second item is the score. Similarly, another  
3302 computer vision API outputs the following for the same image:

3303 `[[‘apple’, 0.7], [‘cherry’, 0.8]].`

3304 Merge operations can, therefore, merge these two responses into just one response.  
3305 Naive ways of merging results could make use of *max*, *min*, and *average* operations  
3306 on the confidence scores. For example, *max* merges results to:

3307 `[[‘apple’, 0.9], [‘banana’, 0.8], [‘cherry’, 0.8]];`

3308 *min* merges results to:

3309 `[[‘apple’, 0.7]];`

3310 and *average* merges results to:

3311 `[[‘apple’, 0.8], [‘banana’, 0.4], [‘cherry’, 0.4]].`

3312 However, as the object’s labels in each result are natural language, the operations  
3313 do not exploit the label’s semantics when conducting label merging. To improve  
3314 the quality of the merged results, we consider the ontologies of these labels, as we  
3315 describe below.

### 3316 8.2.3 Merging Operators for Labels

3317 Merge operations on labels are *n*-ary operations that map  $R^n$  to  $R$ , where  $R_i =$   
3318  $\{(l_{ij}, s_{ij})\}$  is a response from endpoint  $i$  and contains pairs of labels ( $l_{ij}$ ) and scores  
3319 ( $s_{ij}$ ). Merge operations on labels have the following properties:

- 3320 • *identity* defines that merging a single response should output same response  
3321 (i.e.,  $R = \text{merge}(R)$  is always true);
- 3322 • *commutativity* defines that the order of operands should not change the result  
3323 (i.e.,  $\text{merge}(R_1, R_2) = \text{merge}(R_2, R_1)$  is always true);
- 3324 • *reflexivity* defines that merging multiple same responses should output same  
3325 response (i.e.,  $R = \text{merge}(R, R)$  is always true); and,
- 3326 • *additivity* defines that, for a specific label, the merged response should have  
3327 higher or equal score for the label if a concrete endpoint has a higher score.  
3328 Let  $R = \text{merge}(R_1, R_2)$  and  $R' = \text{merge}(R'_1, R_2)$  be merged responses.  $R_1$  and  
3329  $R'_1$  are same, except  $R'_1$  has a higher score for label  $l_x$  than  $R_1$ . The additive  
3330 score property requires that  $R'$  score for  $l_x$  should be greater than or equal to  
3331  $R$  score for  $l_x$ .

3332 The *max*, *min*, and *average* operations in Section 8.2.2 follow each of these rules  
3333 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	$11.42 \pm 7.52$	$10.74 \pm 7.10$ (94.0%)	$0.66 \pm 0.87$
Google Cloud Vision	$8.77 \pm 2.15$	$6.36 \pm 2.22$ (72.5%)	$2.41 \pm 1.93$
Azure Computer Vision	$5.39 \pm 3.29$	$5.26 \pm 3.32$ (97.6%)	$0.14 \pm 0.37$

## 3334 8.3 Graph of Labels

3335 CVSs typically return lists of labels and their associated scores. In most cases, the  
 3336 label can be a singular word (e.g., ‘husky’) or multiple words (e.g., ‘dog breed’).  
 3337 Lexical databases, such as WordNet [197], can therefore be used to describe the  
 3338 ontology behind these labels’ meanings. Figure 8.2 is an example of a graph of  
 3339 labels and synsets. A synset is a grouped set of synonyms for a word. In this image,  
 3340 we consider two fictional endpoints, endpoints 1–2. We label red nodes as labels  
 3341 from endpoint 1, yellow nodes as labels from endpoint 2, and blue nodes as synsets  
 3342 for the associated labels from both endpoints. As actual graphs are usually more  
 3343 complex, Figure 8.2 is a simplified graph to illustrate the usage of associating labels  
 3344 from two concrete sources to synsets.

### 3345 8.3.1 Labels and synsets

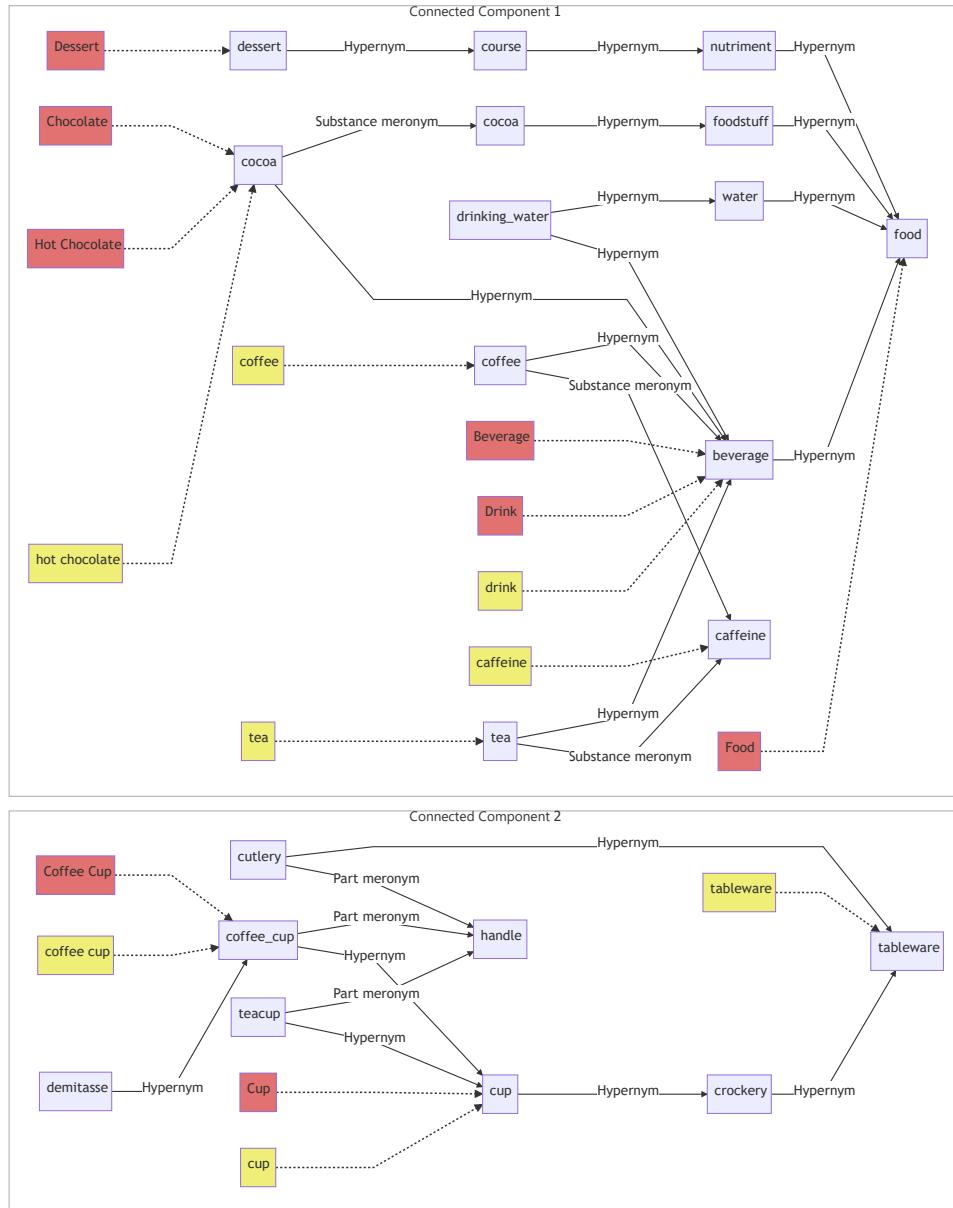
3346 The number of labels depends on input images and concrete API endpoints used.  
 3347 Table 8.1 and Figure 8.3 show how many labels are returned, on average per image,  
 3348 from Google Cloud Vision [331], Amazon Rekognition [318] and Azure Computer  
 3349 Vision [339] image recognition APIs. These statistics were calculated using 1,000  
 3350 images from Open Images Dataset V4 [332] Image-Level Labels set.

3351 Labels from Amazon and Microsoft tend to have corresponding synsets, and  
 3352 therefore these endpoints return common words that are found in WordNet. On the  
 3353 other hand, Google’s labels have less corresponding synsets: for example, labels  
 3354 without corresponding synsets are car models and dog breeds.<sup>1</sup>

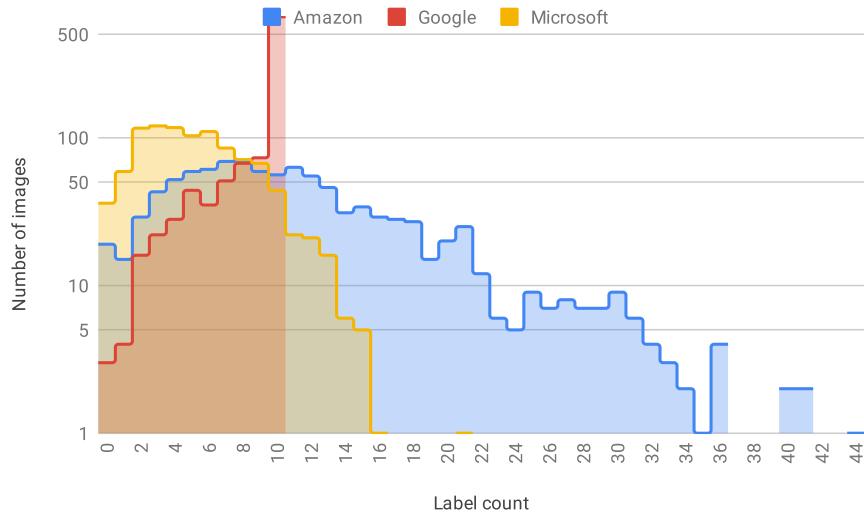
### 3355 8.3.2 Connected Components

3356 A connected component (CC) is a subgraph in which there are paths between any  
 3357 two nodes. In graphs of labels and synsets, CCs are clusters of labels and synsets  
 3358 with similar semantic meaning. For instance, there are two CCs in Figure 8.2. CC 1  
 3359 in Figure 8.2 has ‘beverage’, ‘dessert’, ‘chocolate’, ‘hot chocolate’,  
 3360 ‘drink’, and ‘food’ labels from the red first endpoint and ‘coffee’, ‘hot  
 3361 chocolate’, ‘drink’, ‘caffeine’, and ‘tea’ labels from the yellow second  
 3362 endpoint. Therefore, these labels are related to ‘drink’. On the other hand, CC 2  
 3363 in Figure 8.2 has ‘cup’ and ‘coffee cup’ labels from the first red endpoint and  
 3364 ‘cup’, ‘coffee cup’, and ‘tableware’ labels from the yellow second endpoint.  
 3365 These labels are, therefore, related to ‘cup’.

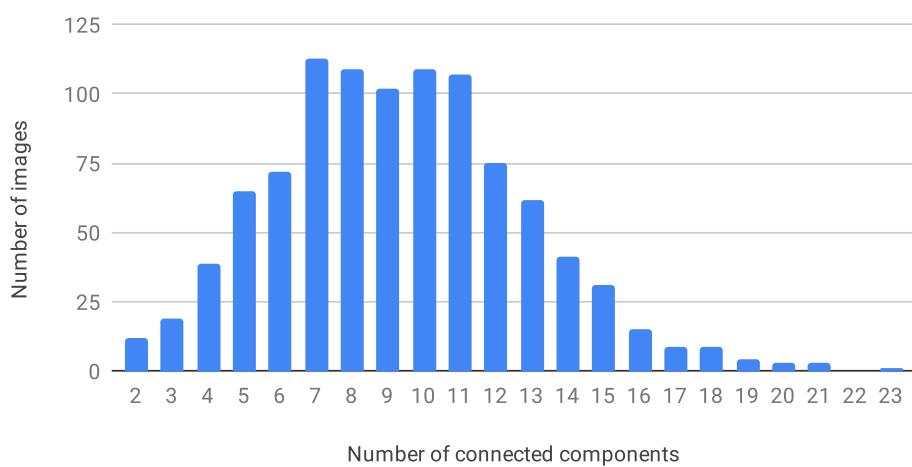
<sup>1</sup>We noticed from our upload of 1,000 images that Google tries to identify objects in greater detail.



**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.

3366 Figure 8.4 shows a distribution of number of CCs for the 1,000-image label  
3367 detections on Amazon Rekognition, Google Cloud Vision, and Azure Computer  
3368 Vision APIs. The average number of CCs is  $9.36 \pm 3.49$ . The smaller number of  
3369 CCs means that most of labels have similar meanings, while a larger value means  
3370 that the labels are more disparate.

## 3371 8.4 API Results Merging Algorithm

3372 Our proposed algorithm to merge labels consists of four parts: (1) mapping labels to  
3373 synsets, (2) deciding the total number of labels, (3) allocating the number of labels  
3374 to CCs, and (4) selecting labels from CCs.

### 3375 8.4.1 Mapping Labels to Synsets

3376 Labels returned in CVS responses are words (in natural language) that do not always  
3377 identify their intended meanings. For instance, a label *orange* may represent the  
3378 fruit, the colour, or the name of the longest river in South Africa. To identify the  
3379 actual meanings behind a label, our facade enumerates all synsets corresponding to  
3380 labels. It then finds the most likely synsets for labels by traversing WordNet links.  
3381 For instance, if an API endpoint outputs the ‘orange’ and ‘lemon’ labels, the  
3382 facade regards ‘orange’ as a related synset word of ‘fruit’. If an API endpoint  
3383 outputs ‘orange’ and ‘water’ labels, the facade regards ‘orange’ as a ‘river’.

### 3384 8.4.2 Deciding Total Number of Labels

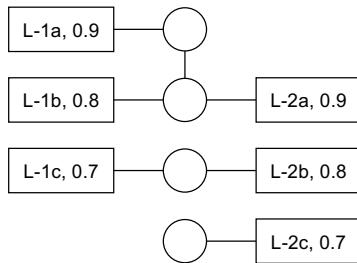
3385 The number of labels in responses from endpoints vary as described in Section 8.3.1.  
3386 The facade decides the number of merged labels using the numbers of labels from  
3387 each endpoint. We formulate the following equation to calculate the number of  
3388 labels:

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

3389 where  $|R|$  is number of labels and scores in response, and  $n$  is number of endpoints.  
3390 In case of naive operations in Section 8.2.2, the following is true:

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

3391 The proposal uses  $\lfloor \sum_i|R_i|/n \rfloor$  to conform to the necessary condition described in  
3392 Section 8.4.3.



**Figure 8.5:** Allocation to connected components.

### 8.4.3 Allocating Number of Labels to Connected Components

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

Allegorically, allocating the number of labels to CCs is similar to proportional representation in a political voting system, where CCs are the political parties and labels are the votes to a party. Several allocation algorithms are introduced in proportional representation, for instance, the D’Hondt and Hare-Niemeyer methods [208]. However, there are differences from proportional representation in the political context. For label merging, labels have scores and origin endpoints and such information may improve the allocation algorithm. For instance, CCs supported with more endpoints should have a higher allocation than CCs with fewer endpoints, and CCs with higher scores should have a higher allocation than CCs with lower scores. We introduce an algorithm to allocate the number of labels to CCs. This allocates more to a CC with more supporting endpoints and higher scores. The steps of the algorithm are:

- 3413     **Step I.** Sort scores separately for each endpoint.
- 3414     **Step II.** If all CCs have an empty score array or more, remove one, and go to Step II.
- 3416     **Step III.** Select the highest score for each endpoint and calculate product of highest scores.
- 3418     **Step IV.** A CC with the highest product score receives an allocation. This CC removes every first element from the score array.
- 3420     **Step V.** If the requested number of allocations is complete, then stop allocation. Otherwise, go to Step II.

Tables 8.2 to 8.5 are examples of allocation iterations. In Table 8.2, the facade sorts scores separately for each endpoint. For instance, the first CC in Figure 8.5 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

3425 non-empty score array or more, so the facade skips Step II. The facade then picks  
 3426 the highest scores for each endpoint and CC. CC 1 has the largest product of highest  
 3427 scores and receives an allocation. In Table 8.3, the first CC removes every first score  
 3428 in its array as it received an allocation in Table 8.2. In this iteration, the second CC  
 3429 has largest product of scores and receives an allocation. In Table 8.4, the second CC  
 3430 removes every first score in its array. At Step II, all the three CCs have an empty  
 3431 array. The facade removes one empty array from each CC. In Table 8.5, the first CC  
 3432 receives an allocation. The algorithm is applicable if total number of allocation is  
 3433 less than or equal to  $\max_i(|R_i|)$  as scores are removed in Step II. The condition is a  
 3434 necessary condition.

#### 3435 **8.4.4 Selecting Labels from Connected Components**

3436 For each CC, the facade applies the *average* operator from Section 8.2.2 and takes  
 3437 labels with  $n$ -highest scores up to allocation, as per Section 8.4.3.

#### 3438 **8.4.5 Conformance to properties**

3439 Section 8.2.3 defines four properties: identity, commutativity, reflexivity, and additivity.  
 3440 Our proposed method conforms to these properties:

- 3441 • *identity*: the method outputs same result if there is one response;
- 3442 • *commutativity*: the method does not care about ordering of operands;
- 3443 • *reflexivity*: the allocations to CCs are same to number of labels in CCs; and
- 3444 • *additivity*: increases in score increases or does not change the allocation to  
 3445 the corresponding CC.

## 3446 **8.5 Evaluation**

### 3447 **8.5.1 Evaluation Method**

3448 To evaluate the merge methods, we merged CVS results from three representative  
 3449 image analysis API endpoints and compared these merged results against human-

3450 verified labels. Images and human-verified labels are sourced from 1,000 randomly-  
3451 sampled images from the Open Images Dataset V4 [332] Image-Level Labels test  
3452 set.

3453 The first three rows in Table 8.7 are the evaluation of original responses from  
3454 each API endpoint. Precision, recall, and F-measure in Table 8.7 do not reflect  
3455 actual values: for instance, it appears that Google performs best at first glance, but  
3456 this is mainly because Google’s labels are similar to that of the Open Images label  
3457 set.

3458 The Open Images Dataset uses 19,995 classes for labelling. The human-verified  
3459 labels for the 1,000 images contain 8,878 of these classes. Table 8.6 shows the  
3460 correspondence between each service’s labels and the Open Images Dataset classes.  
3461 For instance, Amazon Rekognition outputs 11,416 labels in total for 1,000 images.  
3462 There are 1,409 unique labels in 11,416 labels. 1,111 labels out of 1,409 can be  
3463 found in Open Images Dataset classes. Rekognition’s labels matches to Open Images  
3464 Dataset classes at 78.9% ratio, while Google has an outstanding matched percentage  
3465 of 94.1%. This high match is likely due to Google providing both Google Cloud  
3466 Vision and the Open Images Dataset—it is likely that they are trained on the same  
3467 data and labels. An endpoint with higher matched percentage has a more similar  
3468 label set to the Open Images Dataset classes. However, a higher matched percentage  
3469 does not mean imply *better quality* of an API endpoint; it will increase apparent  
3470 precision, recall, and F-measure only.

3471 The true and false positive (TP/FP) label averages and the TP/FP ratio is shown  
3472 in Table 8.7. Where the TP/FP ratio is larger, the scores are more reliable, however  
3473 it is possible to increase the TP/FP ratio by adding more false labels with low scores.  
3474 On the other hand, it is impossible to increase F-measure intentionally, because  
3475 increasing precision will decrease recall, and vice versa. Hence, the importance of  
3476 the F-measure statistic is critical for our analysis.

3477 Let  $R_A$ ,  $R_G$ , and  $R_M$  be responses from Amazon Rekognition, Google Cloud  
3478 Vision, and Microsoft’s Azure Computer Vision, respectively. There are four sets of  
3479 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  
3480 the evaluation of each operands set, Table 8.8 shows the averages of the four operands  
3481 sets, and Figure 8.6 shows the comparison of F-measure for each methods.

### 3482 8.5.2 Naive Operators

3483 Results of *min*, *max*, and *average* operators are shown in Tables 8.7 and 8.8 and Fig-  
3484 ure 8.6. The *min* operator is similar to *union* operator of set operation, and outputs  
3485 all labels of operands. The precision of the *min* operator is always greater than any  
3486 precision of operands, and the recall is always lesser than any precision of operands.  
3487 *Max* and *average* operators are similar to *intersection* operator of set operations.  
3488 Both operators output intersection of labels of operands and there is no clear relation  
3489 to the precision and recall of operands. Since both operators have the same preci-  
3490 sion, recall, and F-measure, Figure 8.6 groups them into one. The *average* operator  
3491 performs well on the TP/FP ratio, where most of the same labels from multiple  
3492 endpoints are TPs. In many cases of the four operand sets, all naive operators’

**Table 8.6:** Matching to human-verified labels.

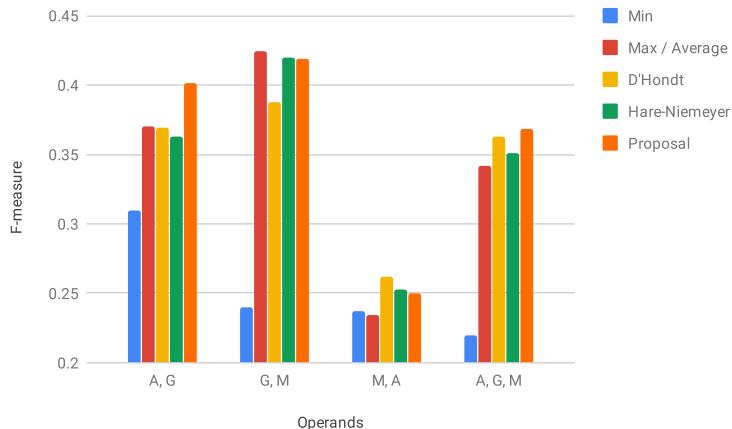
Endpoint	Total	Unique	Matched	Matched %
Amazon Rekognition	11,416	1,409	1,111	78.9
Google Cloud Vision	8,766	2,644	2,487	94.1
Azure Computer Vision	5,392	746	470	63.0

**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.

3493 F-measures are between F-measures of operands. None of naive operators therefore  
3494 improve results by merging responses from multiple endpoints.

### 3495 8.5.3 Traditional Proportional Representation Operators

3496 There are many existing allocation algorithms in proportional representation, e.g.,  
3497 the Niemeyer and Niemeyer method [208]. These methods may be replacements of  
3498 those in Section 8.4.3. Other steps, i.e., Sections 8.4.1, 8.4.2 and 8.4.4, are the same  
3499 as for our proposed technique. Tables 8.7 and 8.8 and Figure 8.6 show the result of  
3500 these traditional proportional representation algorithms. Averages of F-measures by  
3501 traditional proportional representation operators are almost equal to that of the *max*  
3502 and *average* operators. It is worth noting that merging *M* and *A* responses results in  
3503 a better F-measure than each F-measure of *M* and *A* individually. As these are not  
3504 biased to human-verified labels, situations in the real-world usage should, therefore,  
3505 be similar to the case of *M* and *A*. Hence, RQ1 is true.

### 3506 8.5.4 New Proposed Label Merge Technique

3507 As shown in Table 8.8, our proposed new method performs best in F-measure.  
3508 Instead, the TP/FP ratio is less than *average*, the D'Hondt method, and Hare-  
3509 Niemeyer method. As described in Section 8.5.1, we argue that F-measure is a  
3510 more important measure than the TP/FP ratio (in this case). Therefore, RQ2 is  
3511 true. Shown in Table 8.7, our proposed new method improves the results when  
3512 merging *M* and *A* in non-biased endpoints. It is similar to traditional proportional  
3513 representation operators, but does not perform as well. However, it performs better  
3514 on other operand sets, and performs best overall as shown in Figure 8.6.

### 3515 8.5.5 Performance

3516 We used AWS EC2 m5.large instance (2 vCPUs, 2.5 GHz Intel Xeon, 8 GiB RAM);  
3517 Amazon Linux 2 AMI (HVM), SSD Volume Type; Node.js 8.12.0. It takes 0.370

3518 seconds to merge responses from three endpoints. Computational complexity of the  
3519 algorithm in Section 8.4.3 is  $O(n^2)$ , where  $n$  is total number of labels in responses.  
3520 (The estimation assumes that the number of endpoints is a constant.) Complexity  
3521 of Step I in Section 8.4.3 is  $O(n \log n)$ , as the worst case is that all  $n$  labels are from  
3522 one single endpoint and all  $n$  labels are in one CC. Complexity of Step II to Step V  
3523 is  $O(n^2)$ , as the number of CCs is less than or equal to  $n$  and number of iterations  
3524 are less than or equal to  $n$ . As Table 8.1 shows, the averaged total number of three  
3525 endpoints is 25.58. Most of time for merging is consumed by looking up WordNet  
3526 synsets (Section 8.4.1). The API facade calls each APIs on actual endpoints in  
3527 parallel. It takes about 5 seconds, which is much longer than 0.370 seconds taken  
3528 for the merging of responses.

## 3529 8.6 Conclusions and Future Work

3530 In this paper, we propose a method to merge responses from CVSs. Our method  
3531 merges API responses better than naive operators and other proportional represen-  
3532 tation methods (i.e., D’Hondt and Hare-Niemeyer). The average of F-measure of  
3533 our method marks 0.360; the next best method, Hare-Niemeyer, marks 0.347. Our  
3534 method and other proportional representation methods are able to improve the F-  
3535 measure from original responses in some cases. Merging non-biased responses  
3536 results in an F-measure of 0.250, while original responses have an F-measure be-  
3537 tween 0.246 and 0.242. Therefore, users can improve their applications’ precision  
3538 with small modification, i.e., by switching from a singular URL endpoint to a facade-  
3539 based architecture. The performance impact by applying facades is small, because  
3540 overhead in facades is much smaller than API invocation. Our proposal method  
3541 conforms identity, commutativity, reflexivity, and additivity properties and these  
3542 properties are advisable for integrating multiple responses.

3543 Our idea of a proportional representation approach can be applied to other IWSs.  
3544 If the response of such a service is list consisting of an entity and score, and if there is a  
3545 way to group entities, a proposal algorithm can be applied. The opposite approach is  
3546 to improve results by inferring labels. Our current approach picks some of the labels  
3547 returned by endpoints. IWSs are not only based on supervised ML—thus to cover a  
3548 wide range of IWSs, it is necessary to classify and analyse each APIs and establish  
3549 a method to improve results by merging. Currently graph structures of labels and  
3550 synsets (Figure 8.2) are not considered when merging results. Propagating scores  
3551 from labels could be used, losing the additivity property but improving results for  
3552 users. There are many ways to propagate scores. For instance, setting propagation  
3553 factors for each link type would improve merging and could be customised for users’  
3554 preferences. It would be possible to generate an API facade automatically. APIs  
3555 with the same functionality have same or similar signatures. Machine-readable API  
3556 documentation, for instance, OpenAPI Specification, could help a generator to build  
3557 an API facade.



# CHAPTER 9

3558

3559

## 3560 Threshy: Supporting Safe Usage of Intelligent Web Services<sup>†</sup>

3561

3562 **Abstract** Increased popularity of ‘intelligent’ web services provides end-users with machine-  
3563 learnt functionality at little effort to developers. However, these services require a decision  
3564 threshold to be set which is dependent on problem-specific data. Developers lack a systematic  
3565 approach for evaluating intelligent services and existing evaluation tools are predominantly  
3566 targeted at data scientists for pre-development evaluation. This paper presents a workflow  
3567 and supporting tool, Threshy, to help *software developers* select a decision threshold suited  
3568 to their problem domain. Unlike existing tools, Threshy is designed to operate in multiple  
3569 workflows including pre-development, pre-release, and support. Threshold configuration  
3570 files exported by Threshy can be integrated into client applications and monitoring infrastruc-  
3571 ture. Demo: <https://bit.ly/2YKeYhE>.

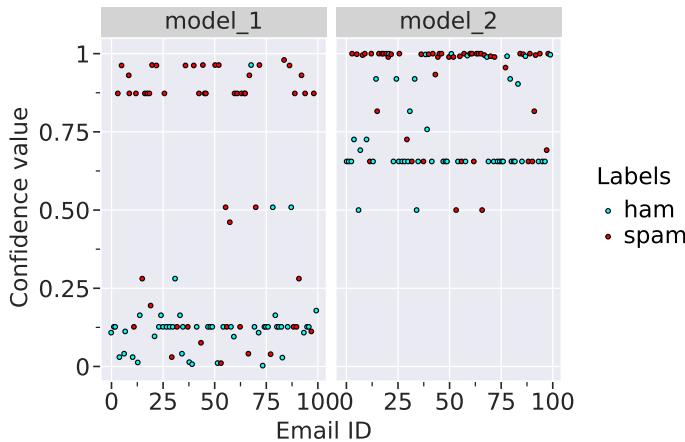
### 3572 9.1 Introduction

3573 Machine learning algorithm adoption is increasing in modern software. End users  
3574 routinely benefit from machine-learnt functionality through personalised recom-  
3575 mendations [63], voice-user interfaces [203], and intelligent digital assistants [38].  
3576 The easy accessibility and availability of intelligent web services<sup>1</sup> is contributing to  
3577 their adoption. These intelligent web services simplify the development of machine  
3578 learning solutions as they (i) do not require specialised machine learning expertise  
3579 to build and maintain, (ii) abstract away infrastructure related issues associated with  
3580 machine learning [11, 257], and (iii) provide web APIs for ease of integration.

3581 However, unlike traditional web services, the functionality of these *intelligent*

<sup>†</sup>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.

<sup>1</sup>Such 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.

3582 services is dependent on a set of assumptions unique to machine learning [69].  
 3583 These assumptions are based on the data used to train machine learning algorithms,  
 3584 the choice of algorithm, and the choice of data processing steps—most of which  
 3585 are not documented. For developers, these assumptions mean that the performance  
 3586 characteristics of an intelligent service in any particular application problem domain  
 3587 is not fully knowable. Intelligent services represent this uncertainty through a  
 3588 confidence value associated with their predictions. Thus an evaluation procedure  
 3589 must be followed as a part of using an intelligent service for an application.

3590 A typical evaluation process would involve a test data set (curated by the devel-  
 3591 opers using the intelligent service) that is used to determine an appropriate threshold.  
 3592 Choice of a decision threshold is a critical element of the evaluation procedure [119].  
 3593 This is especially true for classification problems such as detecting if an image con-  
 3594 tains cancer or identifying all of the topics in a document. Simple approaches  
 3595 to selecting a threshold are often insufficient, as highlighted in Google’s machine  
 3596 learning course: “*It is tempting to assume that [a] classification threshold should  
 3597 always be 0.5, but thresholds are problem-dependent, and are therefore values that  
 3598 you must tune.*”<sup>2</sup> As an example consider the predictions from two email spam  
 3599 classifiers shown in Figure 9.1. The predicted safe emails, ‘ham’, are in two separate  
 3600 clusters (a simple threshold set to approx. 0.2 for model 1 and 0.65 for model 2,  
 3601 indicating that different decision thresholds may be required depending on the clas-  
 3602 sifier. Also note that some emails have been misclassified; how many depends on  
 3603 the choice of decision threshold. An appropriate threshold considers factors outside  
 3604 algorithmic performance, such as financial cost and impact of wrong decisions. To  
 3605 select an appropriate decision threshold, developers using intelligent services need  
 3606 approaches to reason about and consider trade-offs between competing *cost fac-  
 3607 tors*. These include impact, financial costs, and maintenance implications. Without

<sup>2</sup>See <https://bit.ly/36oMgWb>.

3608 considering these trade-offs, sub-optimal decision thresholds will be selected.

3609 The standard approach for tuning thresholds in classification problems involve  
3610 making trade-offs between the number of false positives and false negatives using  
3611 the receiver operating characteristic (ROC) curve. However, developers (i) need  
3612 to realise that this trade-off between false positives and false negatives is a data  
3613 dependent optimisation process [256], (ii) often need to develop custom scripts  
3614 and follow a trial-and-error based approach to determine a threshold, (iii) must  
3615 have appropriate statistical training and expertise, and (iv) be aware that multi-  
3616 label classification require more complex optimisation methods when setting label  
3617 specific costs. However, current intelligent services do not sufficiently guide or  
3618 support software engineers through the evaluation process, nor do they make this  
3619 need clear in the documentation.

3620 In this paper we present **Threshy**<sup>3</sup>, a tool to assist developers in selecting decision  
3621 thresholds when using intelligent services. The motivation for developing Threshy  
3622 arose from our consultancy work with industry. Unlike existing tooling (see Section  
3623 9.4), **Threshy serves as a means to up-skill and educate software engineers**  
3624 **in selecting machine-learnt decision thresholds**, for example, on aspects such as  
3625 confusion matrices. Threshy provides a visually interactive interface for developers  
3626 to fine-tune thresholds and explore trade-offs of prediction hits/misses. This exposes  
3627 the need for optimisation of thresholds, which is dependent on particular use cases.

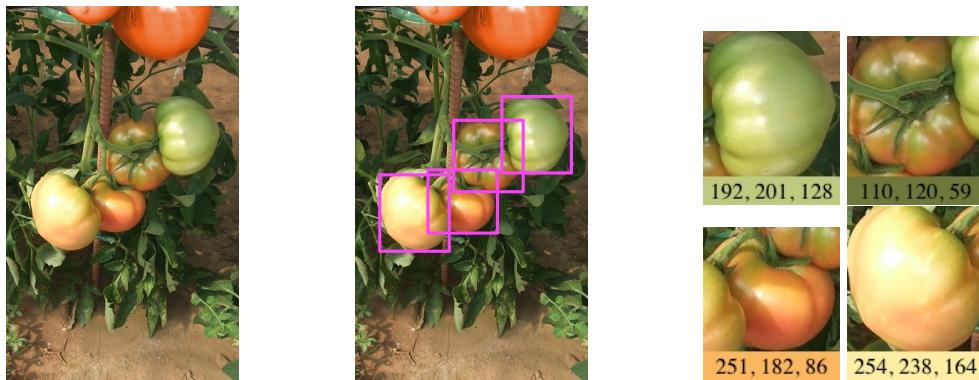
3628 Threshy improves developer productivity through automation of the threshold  
3629 selection process by leveraging an optimisation algorithm to propose thresholds.  
3630 The algorithm considers different cost factors providing developers with summary  
3631 information so they can make more informed trade-offs. Developers also benefit  
3632 from the workflow implemented in Threshy by providing a reproducible procedure  
3633 for testing and tuning thresholds for any category of classification problem (binary,  
3634 multi-class, and multi-label). Threshy has also been designed to work for different  
3635 input data types including images, text and categorical values. The output, is a  
3636 text file and can be integrated into client applications ensuring that the thresholds  
3637 can be updated without code changes (if needed), and continuously monitored in a  
3638 production setting.

## 3639 9.2 Motivating Example

3640 As a motivating example consider Nina, a fictitious developer, who has been em-  
3641 ployed by Lucy’s Tomato Farm to automate the picking of tomatoes from their vines  
3642 (when ripe) using computer vision and a harvesting robot. Lucy’s Farm grow five  
3643 types of tomatoes (roma, cherry, plum, green, and yellow tomatoes). Nina’s robot—  
3644 using an attached webcam—will crawl and take a photo of each vine to assess it  
3645 for harvesting. Nina’s automated harvester needs to sort picked tomatoes into a  
3646 respective container, and thus several business rules need to be encoded into the  
3647 prediction logic to sort each tomato detected based on its *ripeness* (ripe or not ripe)  
3648 and *type of tomato* (as above).

---

<sup>3</sup>Threshy 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).

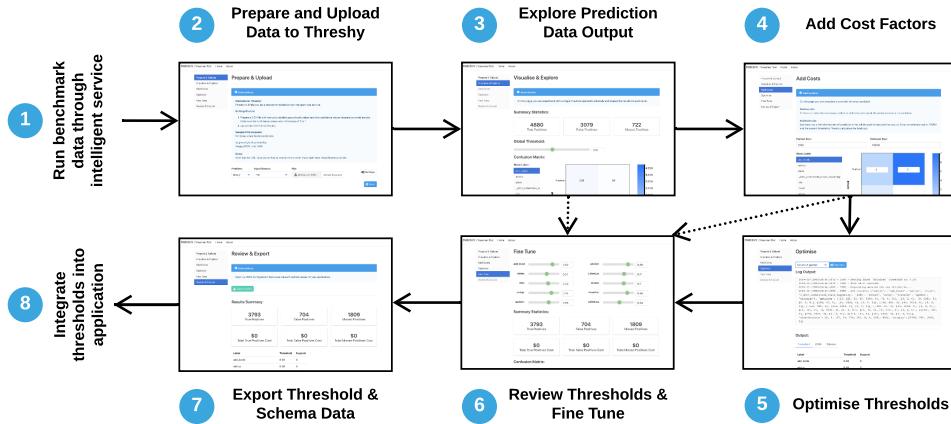
3649 Nina uses a two-stage pipeline consisting of a multi-class and a binary classi-  
 3650 fication model. She has decided to evaluate the viability of cloud based intelligent  
 3651 services and use them if operationally effective. Figure 9.2 illustrates an example of  
 3652 the the pipeline as listed below:

- 3653 1. **Classify tomato ‘type’.** This stage uses an object localisation service to detect  
 3654 all tomato-like objects in the frame and classifies each tomato into one of the  
 3655 following labels: [‘roma’, ‘cherry’, ‘plum’, ‘green’, ‘yellow’].
- 3656 2. **Assess tomato ‘ripeness’.** This stage uses a crop of the localised tomatoes  
 3657 from the original frame to assess the crop’s colour properties (i.e., average  
 3658 colour must have  $R > 200$  and  $G < 240$ ). This produces a binary classification  
 3659 to deduce whether the tomato is ripe or not.

3660 Nina only has a minimal appreciation of the evaluation method to use for off-  
 3661 the-shelf computer vision (classification) services. She also needs to consider the  
 3662 financial costs of mis-classifying either the tomato type or the ripeness. Missing a  
 3663 few ripe tomatoes isn’t a problem as the robot travels the field twice a week during  
 3664 harvest season. However, picking an unripe tomato is expensive as Lucy cannot sell  
 3665 them. Therefore, Nina needs a better (automated) way to assess the performance  
 3666 of the service and set optimal thresholds for her picking robot, thereby maximising  
 3667 profit.

3668 To assist in developing Nina’s pipeline, Lucy sampled a section of 1000 tomatoes  
 3669 by taking a photo of each tomato, labelling its type, and assessing whether the vine  
 3670 was ‘ripe’ or ‘not\_ripe’. Nina ran the labelled images through an intelligent  
 3671 service, with each image having a predicted type (multi-class) and ripeness (binary),  
 3672 with respective confidence values.

3673 Nina combined the predictions, their respective confidence values, and Lucy’s  
 3674 labelled ground truths into a CSV file which was then uploaded to Threshy. Nina  
 3675 asked Lucy to assist in setting relevant costs for correct predictions and false predic-  
 3676 tions. Threshy then recommended a choice of decision threshold which Nina then  
 3677 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

3679 Threshy is a tool to assist software engineers with setting decision thresholds when  
 3680 integrating machine-learnt components in a system. Our tool also serves as a method  
 3681 to inform and educate engineers about the nuances to consider. The novel features  
 3682 of Threshy are:

- 3683 • Automating threshold selection using an optimisation algorithm (NSGA-II  
 3684 [76]), optimising the results for each label.
- 3685 • Support for additional user defined weights when optimising thresholds such  
 3686 as financial costs and impact to society (different type of cost). This allows  
 3687 decision thresholds to be set within a business context as they differ from  
 3688 application to application [88].
- 3689 • Handles nuances of classification problems such as dealing with multi-objective  
 3690 optimisation, and metric selection—reducing errors of omission.
- 3691 • Support key classification problems including binary (e.g. email is either  
 3692 spam or ham), multi-class (e.g. predicting the colour of a car), and multi-label  
 3693 (e.g. assign multiple topics to a document). Existing tools ignore multi-label  
 3694 classification.

3695 Setting thresholds in Threshy is an eight step process as shown in Figure 9.3.  
 3696 Software engineers ① run a benchmark dataset through the machine-learnt com-  
 3697 ponent to create a CSV file with true labels and predicted labels along with the  
 3698 predicted confidence values. The CSV file is then ② uploaded for initial explo-  
 3699 ration where engineers can ③ experiment with modifying a single global threshold  
 3700 for the dataset. Developers may choose to exit at this point (as indicated by dotted  
 3701 arrows in Figure 9.3). Optionally, the engineer ④ defines costs for missed pre-  
 3702 dictions followed by selecting optimisation settings. The optional optimisation step of  
 3703 Threshy ⑤ considers the performance and costs when deriving the thresholds. Fi-  
 3704 nally, the engineer can ⑥ review and fine tune the calculated thresholds, associated

3705 costs, and ⑦ download generated threshold meta-data to be ⑧ integrated into their  
3706 application.

3707 Threshy runs a client/server architecture with a thin-client (see Figure 9.4). The  
3708 web-based application consists of an interactive front-end where developers upload  
3709 benchmark results—consisting of both human annotated labels (ground truths) and  
3710 machine predictions (from the intelligent service)—and use threshold tuners (via  
3711 sliders) to present a data summary of the uploaded CSV. Predicted performances  
3712 and costs are entered manually into the web interface by the developer. The back-end  
3713 of Threshy asynchronously runs a data analyser, cost processor and metrics calculator  
3714 when relevant changes are made to the front-end’s tuning sliders. Separating the  
3715 two concerns allows for high intensity processing to be done on the server and not  
3716 the front end.

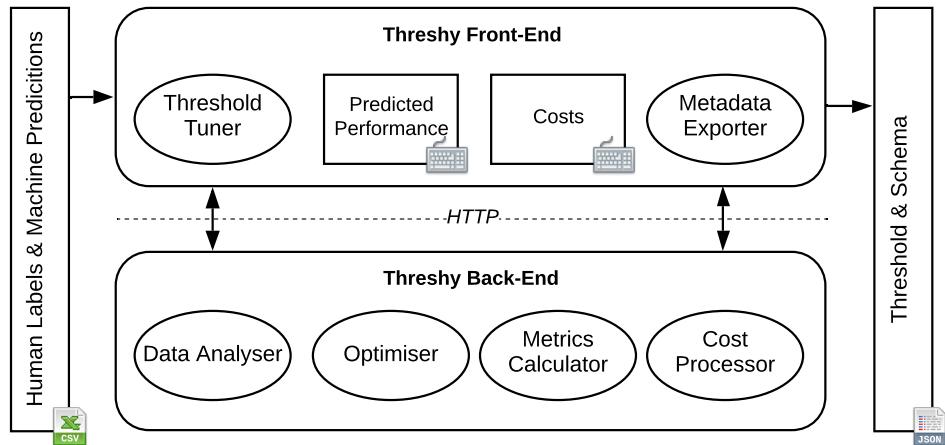
3717 The data analyser provides a comprehensive overview of confusion matrices  
3718 compatible for multi-label multi-class classification problems. When representing  
3719 the confusion matrix, it is trivial to represent instances where multi-label multi-  
3720 classification is not considered. For example, in the simplest case, a single row in  
3721 the matrix represents a single label out of two classes, or each row has one label but  
3722 it has multiple classes. However, a more challenging case to visualise the confusion  
3723 arises when you have  $n$  labels and  $n$  classes; the true/false matches become too  
3724 excessive to visualise as it is disproportionate to the true results. To deal with this  
3725 issue, we condense the summary statistics down to three constructs: (i) number of  
3726 true positives, (ii) false positives, (iii) missed positives. This therefore allows us to  
3727 optimise against the true positives and minimise the other two constructs.

3728 Threshy is a fully self-contained repository containing implementation of the  
3729 tool, scripting and exploratory notebooks, which we make available at <https://github.com/a2i2/threshy>.

## 3731 9.4 Related work

3732 Optimal machine-learnt decision boundaries depend on identifying the operating  
3733 conditions of the problem domain. A systematic study by Drummond and Holte  
3734 [88] classifies four such operating conditions to determine a decision threshold: (i)  
3735 the operating condition is known and thus the model trained matches perfectly; (ii)  
3736 where the operating conditions are known but change with time, and thus the model  
3737 must be adaptable to such changes; (iii) where there is uncertainty in the knowledge  
3738 of the operating conditions certain changes in the operating condition are more likely  
3739 than others; (iv) where there is no knowledge of the operating conditions and the  
3740 conditions may change from the model in any possible way. Various approaches  
3741 to determine appropriate thresholds exist for all four of these cases, such as cost-  
3742 sensitive learning, ROC analysis, cost curves, and Brier scores.

3743 However, an *automated* attempt to calibrate decision threshold boundaries is  
3744 not considered, and is largely pitched at a non-software engineering audience. A  
3745 more recent study touches on this in model management for large-scale adversarial  
3746 instances in Google’s advertising system [256], however this is only a single com-  
3747 ponent within the entire architecture, and is not a tool that is useful for developer’s



**Figure 9.4:** Architecture of Threshy.

3748 in varying contexts. Unlike this study, our work presents a ‘plug-and-play’ style  
 3749 calibration method where any context/domain can have thresholds automatically  
 3750 calibrated (in-context) *and* optimised for engineers; Threshy’s architecture and  
 3751 design facilitates operating in a headless mode enabling use in monitoring and support  
 3752 workflows.

3753 Support tools for ML frameworks generally fall into two categories; the first  
 3754 attempts to illuminate the ‘black box’ by offering ways in which developers can better  
 3755 understand the internals of the model to improve its performance. (For extensive  
 3756 analyses and surveys into this area, see [126, 220].) However, a recent emphasis to  
 3757 probe only inputs and outputs of a model has been explored, exploring off-the-shelf  
 3758 models without knowledge of its unknowns (see Figure 9.1) to reflect the nature  
 3759 of real-world development. Google’s *What-If Tool* [302] for Tensorflow provides a  
 3760 means for data scientists to visualise, measure and assess model performance and  
 3761 fairness with various hypothetical scenarios and data features; similarly, Microsoft’s  
 3762 *Gamut* tool [125] provides an interface to test hypotheticals (although only on  
 3763 Generalized Additive Models) and their *ModelTracker* tool [9] collates summary  
 3764 statistics on a set of sample data to enable rich visualisation of model behaviour and  
 3765 access to key performance metrics.

3766 However, these tools are largely focused toward pre-development model eval-  
 3767 uation and are not designed for the software engineering workflow. They are also  
 3768 targeted to data scientists and not engineers, and certain tools are tied to specific  
 3769 machine learning frameworks (e.g., What-If and Tensorflow). Our work attempts to  
 3770 bridge these gaps through a structured workflow with an automated tool targeted to  
 3771 software developers. We also consider the need to have a consistent tool that works  
 3772 across development, test, and production environments.

## 3773 **9.5 Conclusions & Future Work**

3774 Primary contributions of this work include Threshy, a tool for automating threshold  
 3775 selection, and the overall meta-workflow proposed in Threshy that developers can

<sup>3776</sup> use as a point of reference for calibrating thresholds. In future work, we plan to  
<sup>3777</sup> evaluate Threshy with software engineers to identify additional insights required to  
<sup>3778</sup> make decision thresholds in practice and add code synthesis for monitoring concept  
<sup>3779</sup> drift and for implementing decision thresholds.

# CHAPTER 10

3780

3781

3782

3783

FSE Paper<sup>†</sup>

---

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



3784

## **Part III**

3785

## **Postface**



## CHAPTER 11

3786

3787

3788

3789

### Conclusions & Future Work



3790

---

## References

---

3791

3792

- 3793 [1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving,  
3794 M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker,  
3795 V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: A system for large-  
3796 scale machine learning,” in *Proceedings of the 12th USENIX Symposium on Operating Systems  
3797 Design and Implementation*. Savannah, GA, USA: ACM, 2016. ISBN 978-1-93-197133-1 pp.  
3798 265–283.
- 3799 [2] E. Aghajani, C. Nagy, G. Bavota, and M. Lanza, “A Large-scale empirical study on linguistic  
3800 antipatterns affecting apis,” in *Proceedings of the 34th International Conference on Software  
3801 Maintenance and Evolution*. Madrid, Spain: IEEE, September 2018. DOI 10.1109/IC-  
3802 SME.2018.00012. ISBN 978-1-53-867870-1 pp. 25–35.
- 3803 [3] E. Aghajani, C. Nagy, O. L. Vega-Marquez, M. Linares-Vasquez, L. Moreno, G. Bavota,  
3804 and M. Lanza, “Software Documentation Issues Unveiled,” in *Proceedings of the 41st Interna-  
3805 tional Conference on Software Engineering*. Montreal, QC, Canada: IEEE, May 2019.  
3806 DOI 10.1109/ICSE.2019.00122. ISBN 978-1-72-810869-8. ISSN 0270-5257 pp. 1199–1210.
- 3807 [4] M. Ahazanuzzaman, M. Asaduzzaman, C. K. Roy, and K. A. Schneider, “Classifying stack  
3808 overflow posts on API issues,” in *Proceedings of the 25th International Conference on  
3809 Software Analysis, Evolution and Reengineering*. Campobasso, Italy: IEEE, March 2018.  
3810 DOI 10.1109/SANER.2018.8330213. ISBN 978-1-53-864969-5 pp. 244–254.
- 3811 [5] R. E. Al-Qutaish, “Quality Models in Software Engineering Literature: An Analytical and  
3812 Comparative Study,” *Journal of American Science*, vol. 6, no. 3, pp. 166–175, 2010.
- 3813 [6] H. Allahyari and N. Lavesson, “User-oriented assessment of classification model under-  
3814 standability,” in *Proceedings of the 11th Scandinavian Conference on Artificial Intelligence*, vol.  
3815 227. Trondheim, Norway: IOS Press, May 2011. DOI 10.3233/978-1-60750-754-3-11.  
3816 ISBN 978-1-60-750753-6. ISSN 0922-6389 pp. 11–19.
- 3817 [7] M. Allamanis and C. Sutton, “Why, when, and what: Analyzing stack overflow questions  
3818 by topic, type, and code,” in *Proceedings of the 10th IEEE International Working Con-  
3819 ference on Mining Software Repositories*. San Francisco, CA, USA: IEEE, May 2013.  
3820 DOI 10.1109/MSR.2013.6624004. ISBN 978-1-46-732936-1. ISSN 2160-1852 pp. 53–56.
- 3821 [8] J. Alway and C. Calhoun, *Critical Social Theory: Culture, History, and the Challenge of  
3822 Difference*. American Sociological Association, 1997, vol. 26, no. 1, DOI 10.2307/2076647.
- 3823 [9] S. Amershi, M. Chickering, S. M. Drucker, B. Lee, P. Simard, and J. Suh, “Modeltracker:  
3824 Redesigning performance analysis tools for machine learning,” in *Proceedings of the 33rd  
3825 Annual ACM Conference on Human Factors in Computing Systems*. Seoul, Republic of  
3826 Korea: ACM, April 2015. DOI 10.1145/2702123.2702509. ISBN 978-1-45-033145-6 pp.  
3827 337–346.
- 3828 [10] K. Arnold, “Programmers are People, Too,” *ACM Queue*, vol. 3, no. 5, pp. 54–59, 2005,  
3829 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] S. Barnett, "Extracting technical domain knowledge to improve software architecture," Ph.D. dissertation, Swinburne University of Technology, Hawthorn, Australia, 2018.
- [18] 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.
- [19] 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.
- [20] 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
- [21] Y. Baruch, "Response rate in academic studies - A comparative analysis," *Human Relations*, vol. 52, no. 4, pp. 421–438, 1999, DOI 10.1177/00182679905200401. ISSN 0018-7267
- [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

- [3884] [25] R. Bellazzi and B. Zupan, "Predictive data mining in clinical medicine: Current issues and guidelines," *International Journal of Medical Informatics*, vol. 77, no. 2, pp. 81–97, 2008, DOI 10.1016/j.ijmedinf.2006.11.006. ISSN 1386-5056
- [3885]
- [3886]
- [3887] [26] A. Ben-David, "Monotonicity Maintenance in Information-Theoretic Machine Learning Algorithms," *Machine Learning*, vol. 19, no. 1, pp. 29–43, 1995, DOI 10.1023/A:1022655006810. ISSN 1573-0565
- [3888]
- [3889]
- [3890] [27] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform resource identifier (URI): Generic syntax," Tech. Rep., 2004.
- [3891]
- [3892] [28] L. L. Berry, A. Parasuraman, and V. A. Zeithaml, "SERVQUAL: A multiple-item scale for measuring consumer perceptions of service quality," *Journal of Retailing*, vol. 64, no. 1, pp. 12–40, 1988, DOI 10.1016/S0148-2963(99)00084-3. ISBN 00224359. ISSN 0022-4359
- [3893]
- [3894]
- [3895] [29] J. Bessin, "The Business Value of Quality," [Online] Available: <https://ibm.co/2u0UDK0>, June 2004.
- [3896]
- [3897] [30] S. Beyer and M. Pinzger, "A manual categorization of android app development issues on stack overflow," in *Proceedings of the 30th International Conference on Software Maintenance and Evolution*. Victoria, BC, Canada: IEEE, September 2014. DOI 10.1109/ICSME.2014.88. ISBN 978-0-76-955303-0 pp. 531–535.
- [3898]
- [3899]
- [3900]
- [3901] [31] S. Beyer, C. MacHo, M. Pinzger, and M. Di Penta, "Automatically classifying posts into question categories on stack overflow," in *Proceedings of the 26th International Conference on Program Comprehension*. Gothenburg, Sweden: ACM, May 2018. DOI 10.1145/3196321.3196333. ISBN 978-1-45-035714-2. ISSN 0270-5257 pp. 211–221.
- [3902]
- [3903]
- [3904]
- [3905] [32] J. Biggs and K. Collis, "Evaluating the Quality of Learning: The SOLO Taxonomy (Structure of the Observed Learning Outcome)," *Management in Education*, vol. 1, no. 4, p. 20, 1987, DOI 10.1177/089202068700100412. ISBN 0-12-097551-1. ISSN 0892-0206
- [3906]
- [3907]
- [3908] [33] J. J. Blake, L. P. Maguire, T. M. McGinnity, B. Roche, and L. J. McDaid, "The implementation of fuzzy systems, neural networks and fuzzy neural networks using FPGAs," *Information Sciences*, vol. 112, no. 1-4, pp. 151–168, 1998, DOI 10.1016/S0020-0255(98)10029-4. ISSN 0020-0255
- [3909]
- [3910]
- [3911] [34] B. S. Bloom, *Taxonomy of Educational Objectives, Handbook 1: Cognitive Domain*, 2nd ed. Addison-Wesley Longman, 1956. ISBN 978-0-58-228010-6
- [3912]
- [3913] [35] B. W. Boehm, J. R. Brown, and M. Lipow, "Quantitative evaluation of software quality," in *Proceedings of the 2nd International Conference on Software Engineering*. San Francisco, California, USA: IEEE, October 1976. ISSN 0270-5257 pp. 592–605.
- [3914]
- [3915]
- [3916] [36] B. Boehm and V. R. Basili, "Software defect reduction top 10 list," *Software Management*, pp. 419–421, 2007, DOI 10.1109/9780470049167.ch12. ISBN 978-0-47-004916-7
- [3917]
- [3918] [37] B. W. Boehm, *Software engineering economics*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1981. ISBN 0-13-822122-7
- [3919]
- [3920] [38] M. Boyd and N. Wilson, "Just ask Siri? A pilot study comparing smartphone digital assistants and laptop Google searches for smoking cessation advice," *PLoS ONE*, vol. 13, no. 3, 2018, DOI 10.1371/journal.pone.0194811. ISSN 1932-6203
- [3921]
- [3922]
- [3923] [39] O. Boz, "Extracting decision trees from trained neural networks," in *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Edmonton, AB, Canada: ACM, July 2002. DOI 10.1145/775107.775113, pp. 456–461.
- [3924]
- [3925]
- [3926] [40] H. B. Braiek and F. Khomh, "On Testing Machine Learning Programs," *arXiv preprint arXiv:1812.02257*, December 2018.
- [3927]
- [3928] [41] M. Bramer, *Principles of Data Mining*, ser. Undergraduate Topics in Computer Science. London, England, UK: Springer, 2016, vol. 180, DOI 10.1007/978-1-4471-7307-6. ISBN 978-1-44-717306-9
- [3929]
- [3930]
- [3931] [42] J. Brandt, P. J. Guo, J. Lewenstein, M. Dontcheva, and S. R. Klemmer, "Two studies of opportunistic programming: Interleaving web foraging, learning, and writing code," in *Proceedings of the SIGCHI Conference on Human Factors in Computing System*. Boston, MA, USA: ACM, April 2009. DOI 10.1145/1518701.1518944. ISBN 978-1-60-558247-4 pp. 1589–1598.
- [3932]
- [3933]
- [3934]
- [3935] [43] L. Brathall and M. Jørgensen, "Can you trust a single data source exploratory software engineering case study?" *Empirical Software Engineering*, 2002, DOI 10.1023/A:1014866909191. ISSN 13823256
- [3936]
- [3937]

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

- [3993] [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 1991. DOI 10.1007/BFb0017011. ISBN 978-3-54-053816-5. ISSN 1611-3349 pp. 151–163.
- [3994]
- [3995]
- [3996] [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 1552-3888
- [3997]
- [3998]
- [3999] [63] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*. Boston, MA, USA: ACM, September 2016. DOI 10.1145/2959100.2959190. ISBN 978-1-45-034035-9 pp. 191–198.
- [4000]
- [4001]
- [4002]
- [4003] [64] M. W. Craven and J. W. Shavlik, "Extracting tree-structured representations of trained neural networks," in *Proceedings of the 8th International Conference on Neural Information Processing Systems*, vol. 8. Denver, CO, USA: MIT Press, December 1996. ISBN 978-0-26-220107-0 pp. 24–30.
- [4004]
- [4005]
- [4006]
- [4007] [65] J. W. Creswell, *Research design: Qualitative, quantitative, and mixed methods approaches*, 4th ed. SAGE, 2017. ISBN 860-1-40-429618-5
- [4008]
- [4009] [66] P. B. Crosby, *Quality is free: The art of making quality certain*. McGraw-Hill, 1979. ISBN 978-0-07-014512-2
- [4010]
- [4011] [67] A. Cummaudo, "ESEC/FSE Paper," Unpublished.
- [4012]
- [4013] [68] 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.
- [4014]
- [4015]
- [4016]
- [4017] [69] 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.
- [4018]
- [4019]
- [4020]
- [4021]
- [4022] [70] A. Cummaudo, S. Barnett, R. Vasa, and J. Grundy, "Threshy: Supporting Safe Usage of Intelligent Web Services," Seoul, Republic of Korea, 2020, Unpublished.
- [4023]
- [4024] [71] 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.
- [4025]
- [4026]
- [4027]
- [4028] [72] A. Cummaudo, R. Vasa, and J. Grundy, "Assessing the efficacy of API documentation knowledge against practitioners and computer vision services," 2020, Unpublished.
- [4029]
- [4030] [73] 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.
- [4031]
- [4032]
- [4033] [74] H. da Mota Silveira and L. C. Martini, "How the New Approaches on Cloud Computer Vision can Contribute to Growth of Assistive Technologies to Visually Impaired in the Following Years?" *Journal of Information Systems Engineering & Management*, vol. 2, no. 2, pp. 1–3, 2017, DOI 10.20897/jisem.201709. ISSN 2468-4376
- [4034]
- [4035]
- [4036]
- [4037] [75] R. M. Davison, M. G. Martinsons, and N. Kock, "Principles of canonical action research," *Information Systems Journal*, vol. 14, no. 1, pp. 65–86, 2004, DOI 10.1111/j.1365-2575.2004.00162.x. ISSN 1350-1917
- [4038]
- [4039]
- [4040] [76] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, April 2002, DOI 10.1109/4235.996017. ISSN 1089778X
- [4041]
- [4042]
- [4043] [77] K. Dejaeger, F. Goethals, A. Giangreco, L. Mola, and B. Baesens, "Gaining insight into student satisfaction using comprehensible data mining techniques," *European Journal of Operational Research*, vol. 218, no. 2, pp. 548–562, 2012, DOI 10.1016/j.ejor.2011.11.022. ISSN 0377-2217
- [4044]
- [4045]
- [4046] [78] I. Dey, *Qualitative Data Analysis: A User-Friendly Guide for Social Scientists*. New York, NY: Routledge, 1993. DOI 10.4324/9780203412497. ISBN 978-0-41-505852-0
- [4047]

- [79] V. Dhar, D. Chou, and F. Provost, "Discovering interesting patterns for investment decision making with GLOWER - A genetic learner overlaid with entropy reduction," *Data Mining and Knowledge Discovery*, vol. 4, no. 4, pp. 69–80, 2000, DOI 10.1023/A:1009848126475. ISSN 1384-5810
- [80] V. Dibia, A. Cox, and J. Weisz, "Designing for Democratization: Introducing Novices to Artificial Intelligence Via Maker Kits," in *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. Denver, CO, USA: ACM, May 2017, pp. 381–384.
- [81] M. Doderer, K. Yoon, J. Salinas, and S. Kwek, "Protein subcellular localization prediction using a hybrid of similarity search and Error-Correcting Output Code techniques that produces interpretable results," *In Silico Biology*, vol. 6, no. 5, pp. 419–433, 2006. ISSN 1386-6338
- [82] P. Domingos, "Occam's Two Razors: The Sharp and the Blunt," in *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: AAAI, August 1998. DOI 10.1.1.40.3278, pp. 37–43.
- [83] B. Dorn and M. Guzdial, "Learning on the job: Characterizing the programming knowledge and learning strategies of web designers," in *Proceedings of the 28th ACM Conference on Human Factors in Computing Systems*, vol. 2. Atlanta, GA, USA: ACM, April 2010. DOI 10.1145/1753326.1753430. ISBN 978-1-60-558929-9 pp. 703–712.
- [84] F. Doshi-Velez and B. Kim, "Towards A Rigorous Science of Interpretable Machine Learning," *arXiv preprint arXiv:1702.08608*, 2017.
- [85] F. Doshi-Velez, M. Kortz, R. Budish, C. Bavitz, S. J. Gershman, D. O'Brien, S. Shieber, J. Waldo, D. Weinberger, and A. Wood, "Accountability of AI Under the Law: The Role of Explanation," *SSRN Electronic Journal*, November 2017, In Press, DOI 10.2139/ssrn.3064761.
- [86] S. W. Draper, "The Hawthorne, Pygmalion, Placebo and other effects of expectation: some notes," [Online] Available: <http://bit.ly/2uO2Kth>, Glasgow, Scotland, UK, 2006.
- [87] R. G. Dromey, "A model for software product quality," *IEEE Transactions on Software Engineering*, vol. 21, no. 2, pp. 146–162, 1995, DOI 10.1109/32.345830. ISBN 978-1-11-815666-7. ISSN 0098-5589
- [88] C. Drummond and R. C. Holte, "Cost curves: An improved method for visualizing classifier performance," *Machine Learning*, vol. 65, no. 1, pp. 95–130, October 2006, DOI 10.1007/s10994-006-8199-5. ISSN 0885-6125
- [89] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, "Selecting empirical methods for software engineering research," in *Guide to Advanced Empirical Software Engineering*, F. Shull, J. Singer, and D. I. K. Sjøberg, Eds. Springer, November 2007, ch. 11, pp. 285–311. ISBN 978-1-84-800043-8
- [90] W. Elazmeh, S. Matwin, D. O'Sullivan, W. Michalowski, and K. Farion, "Insights from predicting pediatric asthma exacerbations from retrospective clinical data," in *Proceedings of the 22nd Conference on Artificial Intelligence*, vol. WS-07-05. Vancouver, BC, Canada: AAAI, July 2007. ISBN 978-1-57-735332-4 pp. 10–15.
- [91] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust Physical-World Attacks on Deep Learning Visual Classification," in *Proceedings of the 2017 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, July 2018, DOI 10.1109/CVPR.2018.00175. ISBN 978-1-53-866420-9. ISSN 1063-6919 pp. 1625–1634.
- [92] F. Elder, D. Michie, D. J. Spiegelhalter, and C. C. Taylor, "Machine Learning, Neural, and Statistical Classification." *Journal of the American Statistical Association*, vol. 91, no. 433, pp. 436–438, 1996, DOI 10.2307/2291432. ISBN 978-0-13-106360-0. ISSN 0162-1459
- [93] A. J. Feelders, "Prior knowledge in economic applications of data mining," in *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, vol. 1910. Lyon, France: Springer, September 2000. DOI 10.1007/3-540-45372-5\_42. ISBN 978-3-54-041066-9. ISSN 1611-3349 pp. 395–400.
- [94] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Ph.D. dissertation, University of California, Irvine, 2000.
- [95] I. Finalyson, "Nondeterministic Finite Automata," [Online] Available: <http://bit.ly/319GOF9>, Fredericksburg, VA, USA, 2018.

- 4103 [96] H. Foster, S. Uchitel, J. Magee, and J. Kramer, "Model-based verification of Web service  
4104 compositions," in *Proceedings of the 18th International Conference on Automated Software  
4105 Engineering*. Linz, Austria: IEEE, September 2004. DOI 10.1109/ase.2003.1240303, pp.  
4106 152–161.
- 4107 [97] A. A. Freitas, "A critical review of multi-objective optimization in data mining," *ACM SIGKDD  
4108 Explorations Newsletter*, vol. 6, no. 2, p. 77, 2004, DOI 10.1145/1046456.1046467. ISSN 1931-  
4109 0145
- 4110 [98] ——, "Comprehensible classification models," *ACM SIGKDD Explorations Newsletter*, vol. 15,  
4111 no. 1, pp. 1–10, March 2014, DOI 10.1145/2594473.2594475. ISSN 1931-0145
- 4112 [99] A. A. Freitas, D. C. Wieser, and R. Apweiler, "On the importance of comprehensible classi-  
4113 fication models for protein function prediction," *IEEE/ACM Transactions on Computational  
4114 Biology and Bioinformatics*, vol. 7, no. 1, pp. 172–182, 2010, DOI 10.1109/TCBB.2008.47.  
4115 ISSN 1545-5963
- 4116 [100] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol.  
4117 315, no. 5814, pp. 972–976, February 2007, DOI 10.1126/science.1136800. ISSN 0036-8075
- 4118 [101] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian Network Classifiers," *Machine Learn-  
4119 ing*, vol. 29, no. 2-3, pp. 131–163, 1997, DOI 10.1002/9780470400531.eorms0099. ISSN  
4120 0885-6125
- 4121 [102] G. Fung, S. Sandilya, and R. B. Rao, "Rule extraction from linear support vector machines,"  
4122 *Studies in Computational Intelligence*, vol. 80, no. 1, pp. 83–107, 2009, DOI 10.1007/978-3-  
4123 540-75390-2\_4.
- 4124 [103] M. Gamer, J. Lemon, I. Fellows, and P. Singh, "Irr: various coefficients of interrater reliability,"  
4125 *R package version 0.83*, 2010.
- 4126 [104] S. K. Garg, S. Versteeg, and R. Buyya, "SMICloud: A framework for comparing and ranking  
4127 cloud services," in *Proceedings of the 4th IEEE International Conference on Utility and Cloud  
4128 Computing*. Melbourne, Australia: IEEE, December 2011. DOI 10.1109/UCC.2011.36.  
4129 ISBN 978-0-76-954592-9 pp. 210–218.
- 4130 [105] V. Garousi and M. Felderer, "Experience-based guidelines for effective and efficient data ex-  
4131 traction in systematic reviews in software engineering," in *Proceedings of the 21st International  
4132 Conference on Evaluation and Assessment in Software Engineering*, vol. Part F1286. Karl-  
4133 skrona, Sweden: ACM, June 2017. DOI 10.1145/3084226.3084238. ISBN 978-1-45-034804-1  
4134 pp. 170–179.
- 4135 [106] V. Garousi, M. Felderer, and M. V. Mäntylä, "Guidelines for including grey literature and  
4136 conducting multivocal literature reviews in software engineering," *Information and Software  
4137 Technology*, vol. 106, pp. 101–121, 2019, DOI 10.1016/j.infsof.2018.09.006. ISSN 0950-5849
- 4138 [107] D. A. Garvin, "What Does 'Product Quality' Really Mean?" *MIT Sloan Management Review*,  
4139 vol. 26, no. 1, pp. 25–43, 1984. ISSN 0019-848X
- 4140 [108] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. Daumeé, and K. Craw-  
4141 ford, "Datasheets for Datasets," *arXiv preprint arXiv:1803.09010*, 2018.
- 4142 [109] GeoSpatial World, "Mapillary and Amazon Rekognition collaborate to build a parking solution  
4143 for US cities through computer vision," [Online] Available: <http://bit.ly/36AdRmS>, September  
4144 2018, Accessed: 25 January 2019.
- 4145 [110] M. Gethsiyal Augusta and T. Kathirvalavakumar, "Reverse engineering the neural networks  
4146 for rule extraction in classification problems," *Neural Processing Letters*, vol. 35, no. 2, pp.  
4147 131–150, 2012, DOI 10.1007/s11063-011-9207-8. ISSN 1370-4621
- 4148 [111] H. L. Gilmore, "Product conformance cost," *Quality progress*, vol. 7, no. 5, pp. 16–19, 1974.
- 4149 [112] R. L. Glass, I. Vessey, and V. Ramesh, "RESRES: The story behind the paper "Research in  
4150 software engineering: An analysis of the literature"," *Information and Software Technology*,  
4151 vol. 51, no. 1, pp. 68–70, 2009, DOI 10.1016/j.infsof.2008.09.015. ISSN 0950-5849
- 4152 [113] M. W. Godfrey and D. M. German, "The past, present, and future of software evolution," in  
4153 *Proceedings of the 2008 Frontiers of Software Maintenance*, Beijing, China, October 2008,  
4154 DOI 10.1109/FOSM.2008.4659256. ISBN 978-1-42-442655-3 pp. 129–138.
- 4155 [114] M. W. Godfrey and Q. Tu, "Evolution in open source software: a case study," in  
4156 *Conference on Software Maintenance*. San Jose, CA, USA: IEEE, August 2000.  
4157 DOI 10.1109/icsm.2000.883030, pp. 131–142.

- [4158] [115] P. D. Grünwald, *The Minimum Description Length Principle*. MIT press, 2019.  
 DOI 10.7551/mitpress/4643.001.0001.
- [4159] [116] M. J. Hadley and H. Marc, “Web Application Description Language,” [Online] Available: <http://bit.ly/2RXRhQ1>, August 2009.
- [4160] [117] H. A. Haenssle, C. Fink, R. Schneiderbauer, F. Toberer, T. Buhl, A. Blum, A. Kalloo, A. Ben Hadj Hassen, L. Thomas, A. Enk, L. Uhlmann, C. Alt, M. Arenbergerova, R. Bakos, A. Baltzer, I. Bertlich, A. Blum, T. Bokor-Billmann, J. Bowling, N. Braghiroli, R. Braun, K. Budermann, Bakhaya, T. Buhl, H. Cabo, L. Cabrijan, N. Cevic, A. Classen, D. Deltgen, C. Fink, I. Georgieva, L. E. Hakim-Meibodi, S. Hanner, F. Hartmann, J. Hartmann, G. Haus, E. Hoxha, R. Karls, H. Koga, J. Kreusch, A. Lallas, P. Majenka, A. Marghoob, C. Massone, L. Mekokishvili, D. Mestel, V. Meyer, A. Neuberger, K. Nielsen, M. Oliviero, R. Pampena, J. Paoli, E. Pawlik, B. Rao, A. Rendon, T. Russo, A. Sadek, K. Samhaber, R. Schneiderbauer, A. Schweizer, F. Toberer, L. Trennheuser, L. Vlahova, A. Wald, J. Winkler, P. Wołbing, and I. Zalaudek, “Man against Machine: Diagnostic performance of a deep learning convolutional neural network for dermoscopic melanoma recognition in comparison to 58 dermatologists,” *Annals of Oncology*, vol. 29, no. 8, pp. 1836–1842, May 2018, DOI 10.1093/annonc/mdy166. ISSN 1569-8041
- [4161] [118] K. A. Hallgren, “Computing Inter-Rater Reliability for Observational Data: An Overview and Tutorial,” *Tutorials in Quantitative Methods for Psychology*, vol. 8, no. 1, pp. 23–34, February 2012, DOI 10.20982/tqmp.08.1.p023. ISSN 1913-4126
- [4162] [119] M. Hardt, E. Price, and N. Srebro, “Equality of opportunity in supervised learning,” in *Proceedings of the 30th International Conference on Neural Information Processing Systems*. Barcelona, Spain: Curran Associates Inc., December 2016. DOI 978-1-51-083881-9. ISSN 1049-5258 pp. 3323–3331.
- [4163] [120] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*, 2nd ed., ser. Data Mining, Inference, and Prediction. Springer, January 2001.
- [4164] [121] B. Hayete and J. R. Bienkowska, “Gotrees: Predicting go associations from protein domain composition using decision trees,” in *Proceedings of the Pacific Symposium on Biocomputing 2005, PSB 2005*. Hawaii, USA: World Scientific Publishing Company, January 2005. DOI 10.1142/9789812702456\_0013. ISBN 9-81-256046-7 pp. 127–138.
- [4165] [122] R. Heckel and M. Lohmann, “Towards Contract-based Testing of Web Services,” *Electronic Notes in Theoretical Computer Science*, vol. 116, pp. 145–156, January 2005, DOI 10.1016/j.entcs.2004.02.073. ISSN 1571-0661
- [4166] [123] D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie, “Dependency networks for inference, collaborative filtering, and data visualization,” *Journal of Machine Learning Research*, vol. 1, no. 1, pp. 49–75, 2001, DOI 10.1162/153244301753344614. ISSN 1532-4435
- [4167] [124] M. Henning, “API design matters,” *Communications of the ACM*, vol. 52, no. 5, pp. 46–56, 2009, DOI 10.1145/1506409.1506424. ISSN 0001-0782
- [4168] [125] F. Hohman, A. Head, R. Caruana, R. DeLine, and S. M. Drucker, “Gamut: A design probe to understand how data scientists understand machine learning models,” in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. Glasgow, Scotland, UK: ACM, May 2019. DOI 10.1145/3290605.3300809. ISBN 978-1-45-035970-2
- [4169] [126] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau, “Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 8, pp. 2674–2693, 2019, DOI 10.1109/TVCG.2018.2843369. ISSN 1941-0506
- [4170] [127] J. W. Horch, *Practical Guide To Software Quality Management*. Artech House, 2003. ISBN 978-1-58-053604-2
- [4171] [128] H. Hosseini, B. Xiao, and R. Poovendran, “Google’s cloud vision API is not robust to noise,” in *Proceedings of the 16th IEEE International Conference on Machine Learning and Applications*, vol. 2017-Decem. Cancun, Mexico: IEEE, December 2017. DOI 10.1109/ICMLA.2017.0-172. ISBN 978-1-53-861417-4 pp. 101–105.
- [4172] [129] D. Hou and L. Mo, “Content categorization of API discussions,” in *Proceedings of the 29th International Conference on Software Maintenance*. Eindhoven, Netherlands: IEEE, September 2013. DOI 10.1109/ICSM.2013.17, pp. 60–69.

- 4213 [130] C. Howard, "Introducing Google AI," [Online] Available: <http://bit.ly/2uI6vAr>, May 2018,  
4214 Accessed: 28 August 2018.
- 4215 [131] J. Huang and C. X. Ling, "Using AUC and accuracy in evaluating learning algorithms,"  
4216 *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 3, pp. 299–310, 2005,  
4217 DOI 10.1109/TKDE.2005.50. ISSN 1041-4347
- 4218 [132] J. Huysmans, K. Dejaeger, C. Mues, J. Vanthienen, and B. Baesens, "An empirical evaluation  
4219 of the comprehensibility of decision table, tree and rule based predictive models," *Decision  
4220 Support Systems*, vol. 51, no. 1, pp. 141–154, April 2011, DOI 10.1016/j.dss.2010.12.003.  
4221 ISSN 0167-9236
- 4222 [133] IEEE, "IEEE Standard Glossary of Software Engineering Terminology," 1990.
- 4223 [134] International Organization for Standardization, "ISO25010:2011 - Systems and software engi-  
4224 neering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and  
4225 software quality models," 2011.
- 4226 [135] ———, "ISO/IEC 9126. Information technology – Software product quality," November 1999.
- 4227 [136] A. Iyengar, "Supporting Data Analytics Applications Which Utilize Cognitive Services," in  
4228 *Proceedings of the 37th International Conference on Distributed Computing Systems*. Atlanta,  
4229 GA, USA: IEEE, June 2017. DOI 10.1109/ICDCS.2017.172. ISBN 978-1-53-861791-5 pp.  
4230 1856–1864.
- 4231 [137] N. Japkowicz and M. Shah, *Evaluating learning algorithms: A classification perspective*.  
4232 Cambridge University Press, 2011, vol. 9780521196, DOI 10.1017/CBO9780511921803. ISBN  
4233 978-0-51-192180-3
- 4234 [138] M. W. M. Jaspers, M. Smeulers, H. Vermeulen, and L. W. Peute, "Effects of clinical decision-  
4235 support systems on practitioner performance and patient outcomes: A synthesis of high-quality  
4236 systematic review findings," *Journal of the American Medical Informatics Association*, vol. 18,  
4237 no. 3, pp. 327–334, 2011, DOI 10.1136/amiajnl-2011-000094. ISSN 1067-5027
- 4238 [139] T. Jiang and A. E. Keating, "AVID: An integrative framework for discovering functional rela-  
4239 tionship among proteins," *BMC Bioinformatics*, vol. 6, no. 1, p. 136, 2005, DOI 10.1186/1471-  
4240 2105-6-136. ISSN 1471-2105
- 4241 [140] B. Jimerson and B. Gregory, "Pivotal Cloud Foundry, Google ML, and Spring," [Online]  
4242 Available: <http://bit.ly/2RUBIIL>, San Francisco, CA, USA, December 2017.
- 4243 [141] Y. Jin, *Multi-Objective Machine Learning*, ser. Studies in Computational Intelligence. Berlin,  
4244 Heidelberg: Springer, 2006. DOI 10.1007/3-540-33019-4. ISBN 978-3-54-030676-4
- 4245 [142] U. Johansson and L. Niklasson, "Evolving decision trees using oracle guides," in *Proceedings  
4246 of the 2009 IEEE Symposium on Computational Intelligence and Data Mining*. Nashville,  
4247 TN, USA: IEEE, May 2009. DOI 10.1109/CIDM.2009.4938655. ISBN 978-1-42-442765-9  
4248 pp. 238–244.
- 4249 [143] M. Jørgensen, T. Dybå, K. Liestøl, and D. I. K. Sjøberg, "Incorrect results in software engineer-  
4250 ing experiments: How to improve research practices," *Journal of Systems and Software*, vol.  
4251 116, pp. 133–145, 2016, DOI 10.1016/j.jss.2015.03.065. ISSN 0164-1212
- 4252 [144] C. M. Judd, E. R. Smith, and L. H. Kidder, *Research Methods in Social Relations*, L. H. Kidder,  
4253 Ed. Holt, Rinehart, and Winston, 1991. ISBN 978-0-03-031149-9
- 4254 [145] J. M. Juran, *Juran on Planning for Quality*. New York, NY, USA: The Free Press, 1988. ISBN  
4255 978-0-02-916681-9
- 4256 [146] N. Juristo and O. S. Gómez, "Replication of software engineering experiments," in *Proceedings  
4257 of the LASER Summer School on Software Engineering*. Elba Island, Italy: Springer, 2011.  
4258 DOI 10.1007/978-3-642-25231-0\_2. ISBN 978-3-64-225230-3. ISSN 0302-9743 pp. 60–88.
- 4259 [147] N. Juristo and A. M. Moreno, *Basics of Software Engineering Experimentation*. Boston, MA,  
4260 USA: Springer, March 2001. DOI 10.1007/978-1-4757-3304-4.
- 4261 [148] A. Karwath and R. D. King, "Homology induction: The use of machine learning to improve se-  
4262 quence similarity searches," *BMC Bioinformatics*, vol. 3, no. 1, p. 11, 2002, DOI 10.1186/1471-  
4263 2105-3-11. ISSN 1471-2105
- 4264 [149] K. A. Kaufman and R. S. Michalski, "Learning from inconsistent and noisy data: The AQ18  
4265 approach," in *Proceedings of the 11th European Conference on Principles and Practice of  
4266 Knowledge Discovery in Databases*, vol. 1609. Warsaw, Poland: Springer, September 1999.  
4267 DOI 10.1007/BFb0095128. ISBN 3-540-65965-X. ISSN 1611-3349 pp. 411–419.

- [4268] [150] D. Kavaler, D. Posnett, C. Gibler, H. Chen, P. Devanbu, and V. Filkov, "Using and asking: APIs used in the Android market and asked about in StackOverflow," in *Proceedings of the 5th International Conference on Social Infomatics*. Kyoto, Japan: Springer, November 2013. DOI 10.1007/978-3-319-03260-3\_35. ISBN 978-3-319-03259-7. ISSN 0302-9743 pp. 405–418.
- [4269] [151] B. Kim, "Interactive and Interpretable Machine Learning Models for Human Machine Collaboration," Ph.D. dissertation, Massachusetts Institute of Technology, 2015.
- [4270] [152] B. Kim, C. Rudin, and J. Shah, "The Bayesian case model: A generative approach for case-based reasoning and prototype classification," in *Proceedings of the 28th Conference on Neural Information Processing Systems*, Montreal, QC, Canada, December 2014. ISSN 1049-5258 pp. 1952–1960.
- [4271] [153] B. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature Reviews in Software Engineering," Software Engineering Group, Keele University and Department of Computer Science, University of Durham, Keele, UK, Tech. Rep., 2007.
- [4272] [154] B. A. Kitchenham and S. L. Pfleeger, "Personal opinion surveys," in *Guide to Advanced Empirical Software Engineering*, F. Shull, J. Singer, and D. I. K. Sjøberg, Eds. Springer, November 2007, ch. 3, pp. 63–92. ISBN 978-1-84-800043-8
- [4273] [155] B. A. Kitchenham, T. Dybå, and M. Jorgensen, "Evidence-Based Software Engineering," in *Proceedings of the 26th International Conference on Software Engineering*. Edinburgh, Scotland, UK: IEEE, May 2004. ISBN 978-0-76-952163-3 pp. 273–281.
- [4274] [156] H. K. Klein and M. D. Myers, "A set of principles for conducting and evaluating interpretive field studies in information systems," *MIS Quarterly: Management Information Systems*, vol. 23, no. 1, pp. 67–94, 1999, DOI 10.2307/249410. ISSN 0276-7783
- [4275] [157] A. J. Ko and Y. Riche, "The role of conceptual knowledge in API usability," in *Proceedings of the 2011 IEEE Symposium on Visual Languages and Human Centric Computing*. Pittsburg, PA, USA: IEEE, September 2011. DOI 10.1109/VLHCC.2011.6070395. ISBN 978-1-45-771245-6 pp. 173–176.
- [4276] [158] A. J. Ko, B. A. Myers, and H. H. Aung, "Six learning barriers in end-user programming systems," in *Proceedings of the 2004 IEEE Symposium on Visual Languages and Human Centric Computing*. Rome, Italy: IEEE, September 2004. DOI 10.1109/vlhcc.2004.47. ISBN 0-78-038696-5 pp. 199–206.
- [4277] [159] I. Kononenko, "Inductive and bayesian learning in medical diagnosis," *Applied Artificial Intelligence*, vol. 7, no. 4, pp. 317–337, 1993, DOI 10.1080/08839519308949993. ISSN 1087-6545
- [4278] [160] K. Krippendorff, *Content Analysis*, ser. An Introduction to Its Methodology. SAGE, 1980. ISBN 978-1-50-639566-1
- [4279] [161] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017, DOI 10.1145/3065386. ISSN 1557-7317
- [4280] [162] J. A. Krosnick, "Survey Research," *Annual Review of Psychology*, vol. 50, no. 1, pp. 537–567, February 1999, DOI 10.1146/annurev.psych.50.1.537. ISSN 0066-4308
- [4281] [163] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Proceedings of the 5th International Conference on Learning Representations*, Toulon, France, April 2017.
- [4282] [164] G. Laforge, "Machine Intelligence at Google Scale," in *QCon*, London, England, UK, June 2018.
- [4283] [165] H. Lakkaraju, S. H. Bach, and J. Leskovec, "Interpretable decision sets: A joint framework for description and prediction," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco, CA, USA: ACM, August 2016. DOI 10.1145/2939672.2939874. ISBN 978-1-45-034232-2 pp. 1675–1684.
- [4284] [166] J. R. Landis and G. G. Koch, "The Measurement of Observer Agreement for Categorical Data," *Biometrics*, vol. 33, no. 1, p. 159, March 1977, DOI 10.2307/2529310. ISSN 0006341X
- [4285] [167] F. Lau, "Toward a framework for action research in information systems studies," *Information Technology & People*, vol. 12, no. 2, pp. 148–176, 1999, DOI 10.1108/09593849910267206. ISSN 0959-3845
- [4286] [168] N. Lavrač, "Selected techniques for data mining in medicine," *Artificial Intelligence in Medicine*, vol. 16, no. 1, pp. 3–23, 1999, DOI 10.1016/S0933-3657(98)00062-1. ISSN 0933-3657

- 4324 [169] T. Lei, R. Barzilay, and T. Jaakkola, "Rationalizing neural predictions," in *Proceedings of the 9th*  
4325 *International Joint Conference on Natural Language Processing and Conference on Empirical*  
4326 *Methods in Natural Language Processing*. Austin, TX, USA: Association for Computational  
4327 Linguistics, November 2016. DOI 10.18653/v1/d16-1011. ISBN 978-1-94-562625-8 pp.  
4328 107–117.
- 4329 [170] T. C. Lethbridge, S. E. Sim, and J. Singer, "Studying software engineers: Data collection  
4330 techniques for software field studies," *Empirical Software Engineering*, vol. 10, no. 3, pp.  
4331 311–341, July 2005, DOI 10.1007/s10664-005-1290-x. ISSN 1382-3256
- 4332 [171] R. J. Light, "Measures of response agreement for qualitative data: Some generalizations and  
4333 alternatives," *Psychological Bulletin*, vol. 76, no. 5, pp. 365–377, 1971, DOI 10.1037/h0031643.  
4334 ISSN 0033-2909
- 4335 [172] E. Lima, C. Mues, and B. Baesens, "Domain knowledge integration in data mining using  
4336 decision tables: Case studies in churn prediction," *Journal of the Operational Research Society*,  
4337 vol. 60, no. 8, pp. 1096–1106, 2009, DOI 10.1057/jors.2008.161. ISSN 0160-5682
- 4338 [173] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L.  
4339 Zitnick, "Microsoft COCO: Common objects in context," in *Proceedings of the 13th European*  
4340 *Conference on Computer Vision*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., vol.  
4341 8693 LNCS, no. PART 5. Zurich, Germany: Springer, September 2014. DOI 10.1007/978-  
4342 3-319-10602-1\_48. ISSN 1611-3349 pp. 740–755.
- 4343 [174] M. Linares-Vásquez, G. Bavota, M. Di Penta, R. Oliveto, and D. Poshyvanyk, "How do API  
4344 changes trigger stack overflow discussions? A study on the android SDK," in *Proceedings of*  
4345 *the 22nd International Conference on Program Comprehension*. Hyderabad, India: ACM,  
4346 June 2014. DOI 10.1145/2597008.2597155. ISBN 978-1-45-032879-1 pp. 83–94.
- 4347 [175] Z. C. Lipton, "The mythos of model interpretability," *Communications of the ACM*, vol. 61,  
4348 no. 10, pp. 35–43, 2018, DOI 10.1145/3233231. ISSN 1557-7317
- 4349 [176] M. Litwin, *How to Measure Survey Reliability and Validity*. Thousand Oaks, CA, USA: SAGE,  
4350 1995, vol. 7, DOI 10.4135/9781483348957. ISBN 978-0-80-395704-6
- 4351 [177] Y. Liu, T. Kohlberger, M. Norouzi, G. E. Dahl, J. L. Smith, A. Mohtashamian, N. Olson,  
4352 L. H. Peng, J. D. Hipp, and M. C. Stumpe, "Artificial Intelligence-Based Breast Cancer Nodal  
4353 Metastasis Detection." *Archives of Pathology & Laboratory Medicine*, vol. 143, no. 7, pp.  
4354 859–868, July 2017, DOI 10.5858/arpa.2018-0147-OA. ISSN 1543-2165
- 4355 [178] D. Lo Giudice, C. Mines, A. LeClair, R. Curran, and A. Homan, "How AI Will Change  
4356 Software Development And Applications," [Online] Available: <http://bit.ly/38RiAlN>, Forrester  
4357 Research, Inc., Tech. Rep., November 2016.
- 4358 [179] R. Lori and M. Oded, *Data mining with decision trees*. World Scientific Publishing Company,  
4359 2008, vol. 69. ISBN 978-9-81-277171-1
- 4360 [180] R. E. Lyons and W. Vanderkulk, "The Use of Triple-Modular Redundancy to Improve Computer  
4361 Reliability," *IBM Journal of Research and Development*, vol. 6, no. 2, pp. 200–209, April 2010,  
4362 DOI 10.1147/rd.62.0200. ISSN 0018-8646
- 4363 [181] W. Maalej and M. P. Robillard, "Patterns of knowledge in API reference documentation," *IEEE*  
4364 *Transactions on Software Engineering*, 2013, DOI 10.1109/TSE.2013.12. ISSN 0098-5589
- 4365 [182] S. MacDonell, M. Shepperd, B. Kitchenham, and E. Mendes, "How reliable are systematic  
4366 reviews in empirical software engineering?" *IEEE Transactions on Software Engineering*,  
4367 vol. 36, no. 5, pp. 676–687, September 2010, DOI 10.1109/TSE.2010.28. ISSN 0098-5589
- 4368 [183] G. Malgieri and G. Comandé, "Why a right to legibility of automated decision-making exists  
4369 in the general data protection regulation," *International Data Privacy Law*, vol. 7, no. 4, pp.  
4370 243–265, June 2017, DOI 10.1093/idpl/ixp019. ISSN 2044-4001
- 4371 [184] T. E. Marshall and S. L. Lambert, "Cloud-based intelligent accounting applications: Accounting  
4372 task automation using IBM watson cognitive computing," *Journal of Emerging Technologies*  
4373 *in Accounting*, vol. 15, no. 1, pp. 199–215, 2018, DOI 10.2308/jeta-52095. ISSN 1558-7940
- 4374 [185] D. Martens, J. Vanthienen, W. Verbeke, and B. Baesens, "Performance of classification mod-  
4375 els from a user perspective," *Decision Support Systems*, vol. 51, no. 4, pp. 782–793, 2011,  
4376 DOI 10.1016/j.dss.2011.01.013. ISSN 0167-9236
- 4377 [186] P. Mayring, "Mixing Qualitative and Quantitative Methods," in *Mixed Methodology in Psycho-  
4378 logical Research*. Sense Publishers, 2007, ch. 6, pp. 27–36. ISBN 978-9-07-787473-8

- 4379 [187] J. A. McCall, P. K. Richards, and G. F. Walters, "Factors in Software Quality: Concept and  
4380 Definitions of Software Quality," General Electric Company, Griffiss Air Force Base, NY, USA,  
4381 Tech. Rep. RADC-TR-77-369, November 1977.
- 4382 [188] J. McCarthy, "Programs with common sense," in *Proceedings of the Symposium on the Mechanization of Thought Processes*, Cambridge, MA, USA, 1963, pp. 1–15.
- 4384 [189] B. McGowen, "Machine learning with Google APIs," [Online] Available: <http://bit.ly/3aUQpo2>, January 2019.
- 4386 [190] M. L. McHugh, "Interrater reliability: The kappa statistic," *Biochemia Medica*, vol. 22, no. 3,  
4387 pp. 276–282, 2012, DOI 10.11613/bm.2012.031. ISSN 1330-0962
- 4388 [191] L. McLeod and S. G. MacDonell, "Factors that affect software systems development project  
4389 outcomes: A survey of research," *ACM Computing Surveys*, vol. 43, no. 4, p. 24, 2011,  
4390 DOI 10.1145/1978802.1978803. ISSN 0360-0300
- 4391 [192] J. Meltzoff and H. Cooper, *Critical thinking about research: Psychology and related fields*,  
4392 2nd ed. American Psychological Association, 2018. DOI 10.1037/0000052-000.
- 4393 [193] T. Mens and S. Demeyer, *Software Evolution*. Berlin, Heidelberg: Springer, 2008.  
4394 DOI 10.1007/978-3-540-76440-3. ISBN 978-3-54-076439-7
- 4395 [194] T. Mens, S. Demeyer, M. Wermelinger, R. Hirschfeld, S. Ducasse, and M. Jazayeri, "Chal-  
4396 lenges in software evolution," in *Proceedings of the 8th International Workshop on Principles  
4397 of Software Evolution*, vol. 2005. Lisbon, Portugal: IEEE, September 2005. DOI 10.1109/I-  
4398 WPSE.2005.7. ISBN 0-76-952349-8. ISSN 1550-4077 pp. 13–22.
- 4399 [195] A. C. Michalos and H. A. Simon, *The Sciences of the Artificial*. MIT press, 1970, vol. 11,  
4400 no. 1, DOI 10.2307/3102825.
- 4401 [196] D. Michie, "Machine learning in the next five years," in *Proceedings of the 3rd European  
4402 Conference on European Working Session on Learning*. Glasgow, Scotland, UK: Pitman  
4403 Publishing, Inc., October 1988. ISBN 978-0-27-308800-4 pp. 107–122.
- 4404 [197] G. A. Miller, "WordNet: A Lexical Database for English," *Communications of the ACM*, vol. 38,  
4405 no. 11, pp. 39–41, November 1995, DOI 10.1145/219717.219748. ISSN 1557-7317
- 4406 [198] M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D.  
4407 Raji, and T. Gebru, "Model cards for model reporting," in *Proceedings of the 2nd Conference  
4408 on Fairness, Accountability, and Transparency*. Atlanta, GA, USA: ACM, January 2019.  
4409 DOI 10.1145/3287560.3287596. ISBN 978-1-45-036125-5 pp. 220–229.
- 4410 [199] D. Moody, "The physics of notations: Toward a scientific basis for constructing visual notations  
4411 in software engineering," *IEEE Transactions on Software Engineering*, vol. 35, no. 6, pp.  
4412 756–779, 2009, DOI 10.1109/TSE.2009.67. ISSN 0098-5589
- 4413 [200] C. Murphy and G. Kaiser, "Improving the Dependability of Machine Learning Applications,"  
4414 Department of Computer Science, Columbia University, New York, NY, USA, Tech. Rep. MI,  
4415 2008.
- 4416 [201] C. Murphy, G. Kaiser, and M. Arias, "An approach to software testing of machine learning  
4417 applications," in *Proceedings of the 19th International Conference on Software Engineering and  
4418 Knowledge Engineering*, Boston, MA, USA, July 2007. ISBN 978-1-62-748661-3 pp. 167–172.
- 4419 [202] B. A. Myers, S. Y. Jeong, Y. Xie, J. Beaton, J. Stylos, R. Ehret, J. Karstens, A. Efeoglu, and D. K.  
4420 Busse, "Studying the Documentation of an API for Enterprise Service-Oriented Architecture,"  
4421 *Journal of Organizational and End User Computing*, vol. 22, no. 1, pp. 23–51, January 2010,  
4422 DOI 10.4018/joeuc.2010101903. ISSN 1546-2234
- 4423 [203] C. Myers, A. Furqan, J. Nebolsky, K. Caro, and J. Zhu, "Patterns for how users overcome  
4424 obstacles in Voice User Interfaces," in *Proceedings of the 2018 CHI Conference on Human  
4425 Factors in Computing Systems*, vol. 2018-April. Montreal, QC, Canada: ACM, April 2018.  
4426 DOI 10.1145/3173574.3173580. ISBN 978-1-45-035620-6 p. 6.
- 4427 [204] S. Nakajima, "Model-Checking Verification for Reliable Web Service," in *Proceedings of the  
4428 First International Symposium on Cyber World*. Montreal, QC, Canada: IEEE, November  
4429 2002. ISBN 978-0-76-951862-6 pp. 378–385.
- 4430 [205] M. Narayanan, E. Chen, J. He, B. Kim, S. Gershman, and F. Doshi-Velez, "How do Humans  
4431 Understand Explanations from Machine Learning Systems? An Evaluation of the Human-  
4432 Interpretability of Explanation," *IEEE Transactions on Evolutionary Computation*, 2018, In  
4433 Press.

- 4434 [206] S. Narayanan and S. A. McIlraith, "Simulation, verification and automated composition of web  
4435 services," in *Proceedings of the 11th International Conference on World Wide Web*. Honolulu,  
4436 HI, USA: ACM, May 2002. DOI 10.1145/511446.511457. ISBN 1-58-113449-5 pp. 77–88.
- 4437 [207] B. J. Nelson, "Remote Procedure Call," Ph.D. dissertation, Carnegie Mellon University, 1981.
- 4438 [208] H. F. Niemeyer and A. C. Niemeyer, "Apportionment methods," *Mathematical Social Sciences*,  
4439 vol. 56, no. 2, pp. 240–253, 2008. ISSN 0165-4896
- 4440 [209] Y. Nishi, S. Masuda, H. Ogawa, and K. Uetsuki, "A test architecture for machine learning  
4441 product," in *Proceedings of the 11th International Conference on Software Testing,  
4442 Verification and Validation Workshops*. Västerås, Sweden: IEEE, April 2018.  
4443 DOI 10.1109/ICSTW.2018.00060. ISBN 978-1-53-866352-3 pp. 273–278.
- 4444 [210] N. Novielli, F. Calefato, and F. Lanobile, "The challenges of sentiment detection in the social  
4445 programmer ecosystem," in *Proceedings of the 7th International Workshop on Social Software  
4446 Engineering*, Bergamo, Italy, August 2015, DOI 10.1145/2804381.2804387. ISBN 978-1-45-  
4447 03381-9 pp. 33–40.
- 4448 [211] K. Nybom, A. Ashraf, and I. Porres, "A systematic mapping study on API documentation  
4449 generation approaches," in *Proceedings of the 44th Euromicro Conference on Software  
4450 Engineering and Advanced Applications*. Prague, Czech Republic: IEEE, August 2018.  
4451 DOI 10.1109/SEAA.2018.00081. ISBN 978-1-53-867382-9 pp. 462–469.
- 4452 [212] J. Nykaza, R. Messinger, F. Boehme, C. L. Norman, M. Mace, and M. Gordon, "What program-  
4453 mers really want: Results of a needs assessment for SDK documentation," in *Proceedings of the  
4454 20th Annual International Conference on Computer Documentation*. Toronto, ON, Canada:  
4455 ACM, October 2002. DOI 10.1145/584955.584976, pp. 133–141.
- 4456 [213] T. Ohtake, A. Cummaudo, M. Abdelrazek, R. Vasa, and J. Grundy, "Merging intelligent API  
4457 responses using a proportional representation approach," in *Proceedings of the 19th Interna-  
4458 tional Conference on Web Engineering*. Daejeon, Republic of Korea: Springer, June 2019.  
4459 DOI 10.1007/978-3-030-19274-7\_28. ISBN 978-3-03-019273-0. ISSN 1611-3349 pp. 391–  
4460 406.
- 4461 [214] Open Software Foundation, "Part 3: DCE Remote Procedure Call (RPC)," in *OSF DCE  
4462 application development guide: revision 1.0*. Prentice Hall, December 1991.
- 4463 [215] N. Oreskes, K. Shrader-Frechette, and K. Belitz, "Verification, validation, and confirmation  
4464 of numerical models in the earth sciences," *Science*, vol. 263, no. 5147, pp. 641–646, 1994,  
4465 DOI 10.1126/science.263.5147.641. ISSN 0036-8075
- 4466 [216] A. L. M. Ortiz, "Curating Content with Google Machine Learning Application Programming  
4467 Interfaces," in *EIAPortugal*, July 2017.
- 4468 [217] F. E. B. Otero and A. A. Freitas, "Improving the interpretability of classification rules discovered  
4469 by an ant colony algorithm: Extended results," in *Evolutionary Computation*, vol. 24, no. 3.  
4470 ACM, 2016. DOI 10.1162/EVCO\_a\_00155. ISSN 1530-9304 pp. 385–409.
- 4471 [218] A. Pal, S. Chang, and J. A. Konstan, "Evolution of experts in question answering communities,"  
4472 in *Proceedings of the 6th International AAAI Conference on Weblogs and Social Media*. Dublin,  
4473 Ireland: AAAI, June 2012. ISBN 978-1-57-735556-4 pp. 274–281.
- 4474 [219] R. Parekh, "Designing AI at Scale to Power Everyday Life," in *Proceedings of the 23rd ACM  
4475 SIGKDD International Conference on Knowledge Discovery and Data Mining*. Halifax, NS,  
4476 Canada: ACM, August 2017. DOI 10.1145/3097983.3105815, p. 27.
- 4477 [220] K. Patel, J. Fogarty, J. A. Landay, and B. Harrison, "Investigating statistical machine learn-  
4478 ing as a tool for software development," in *Proceedings of the 26th SIGCHI Conference on  
4479 Human Factors in Computing Systems*, ser. CHI '08. Florence, Italy: ACM, April 2008.  
4480 DOI 10.1145/1357054.1357160. ISBN 978-1-60-558011-1 pp. 667–676.
- 4481 [221] C. Pautasso, O. Zimmermann, and F. Leymann, "RESTful web services vs. "Big" web services:  
4482 Making the right architectural decision," in *Proceedings of the 17th International Conference  
4483 on World Wide Web*. Beijing, China: ACM, April 2008. DOI 10.1145/1367497.1367606.  
4484 ISBN 978-1-60-558085-2
- 4485 [222] M. Pazzani, "Comprehensible knowledge discovery: gaining insight from data," in *Proceedings  
4486 of the First Federal Data Mining Conference and Exposition*, Washington, DC, USA, 1997, pp.  
4487 73–82.

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

- 4544 [240] G. Ridgeway, D. Madigan, T. Richardson, and J. O’Kane, “Interpretable Boosted Naïve Bayes  
4545 Classification,” in *Proceedings of the 4th International Conference on Knowledge Discovery  
4546 and Data Mining*. New York, NY, USA: AAAI, 1998, pp. 101–104.
- 4547 [241] RightScale Inc., “State of the Cloud Report: DevOps Trends,” Tech. Rep., 2016.
- 4548 [242] G. Ritzer and E. Guba, “The Paradigm Dialog,” *Canadian Journal of Sociology*, vol. 16, no. 4,  
4549 p. 446, 1991, DOI 10.2307/3340973. ISSN 0318-6431
- 4550 [243] S. P. Robbins, *Essentials of organizational behavior*, 14th ed. Pearson, 2017. ISBN 978-0-  
4551 13-452470-2
- 4552 [244] M. P. Robillard, “What makes APIs hard to learn? Answers from developers,” *IEEE Software*,  
4553 vol. 26, no. 6, pp. 27–34, 2009, DOI 10.1109/MS.2009.193. ISSN 0740-7459
- 4554 [245] M. P. Robillard and R. Deline, “A field study of API learning obstacles,” *Empirical Software  
4555 Engineering*, vol. 16, no. 6, pp. 703–732, 2011, DOI 10.1007/s10664-010-9150-8. ISSN 1382-  
4556 3256
- 4557 [246] H. Robinson, J. Segal, and H. Sharp, “Ethnographically-informed empirical studies of soft-  
4558 ware practice,” *Information and Software Technology*, vol. 49, no. 6, pp. 540–551, 2007,  
4559 DOI 10.1016/j.infsof.2007.02.007. ISSN 0950-5849
- 4560 [247] C. Rosen and E. Shihab, “What are mobile developers asking about? A large scale study  
4561 using stack overflow,” *Empirical Software Engineering*, vol. 21, no. 3, pp. 1192–1223, 2016,  
4562 DOI 10.1007/s10664-015-9379-3. ISSN 1573-7616
- 4563 [248] W. Rosenberry, D. Kenney, and G. Fisher, *Understanding DCE*. O’Reilly & Associates, Inc.,  
4564 1992. ISBN 978-1-56-592005-7
- 4565 [249] A. Rosenfeld, R. Zemel, and J. K. Tsotsos, “The Elephant in the Room,” *arXiv preprint  
4566 arXiv:1808.03305*, 2018.
- 4567 [250] A. S. Ross, M. C. Hughes, and F. Doshi-Velez, “Right for the right reasons: Training differenti-  
4568 able models by constraining their explanations,” in *Proceedings of the 26th International Joint  
4569 Conferences on Artificial Intelligence*, Melbourne, Australia, August 2017, DOI 10.24963/ij-  
4570 cai.2017/371. ISBN 978-0-99-924110-3. ISSN 1045-0823 pp. 2662–2670.
- 4571 [251] R. J. Rubey and R. D. Hartwick, “Quantitative measurement of program quality,” in *Proceedings  
4572 of the 1968 23rd ACM National Conference*. Las Vegas, NV, USA: ACM, August 1968.  
4573 DOI 10.1145/800186.810631. ISBN 978-1-45-037486-6 pp. 671–677.
- 4574 [252] J. Rutten, M. Kwiatkowska, G. Norman, and D. Parker, *Mathematical techniques for analyzing  
4575 concurrent and probabilistic systems*, ser. CRM Monograph Series, P. Panangaden and F. van  
4576 Breugel, Eds. American Mathematical Society, 2004, vol. 23.
- 4577 [253] M. Schwabacher and P. Langley, “Discovering communicable scientific knowledge from spatio-  
4578 temporal data,” in *Proceedings of the 18th International Conference on Machine Learning*.  
4579 Williamstown, MA, USA: Morgan Kaufmann, June 2001. ISBN 978-1-55-860778-1 pp. 489–  
4580 496.
- 4581 [254] A. Schwaighofer and N. D. Lawrence, *Dataset shift in machine learning*, J. Quiñonero-Candela  
4582 and M. Sugiyama, Eds. Cambridge, MA, USA: The MIT Press, 2008. ISBN 978-0-26-217005-  
4583 5
- 4584 [255] T. A. Schwandt, “Qualitative data analysis: An expanded sourcebook,” *Evaluation and Program  
4585 Planning*, vol. 19, no. 1, pp. 106–107, 1996, DOI 10.1016/0149-7189(96)88232-2. ISSN 0149-  
4586 7189
- 4587 [256] D. Sculley, M. E. Otey, M. Pohl, B. Spitznagel, J. Hainsworth, and Y. Zhou, “Detecting  
4588 adversarial advertisements in the wild,” in *Proceedings of the 17th ACM SIGKDD International  
4589 Conference on Knowledge Discovery and Data Mining*, ACM. San Diego, CA, USA: ACM,  
4590 August 2011. DOI 10.1145/2020408.2020455, pp. 274–282.
- 4591 [257] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J. F.  
4592 Crespo, and D. Dennison, “Hidden technical debt in machine learning systems,” in *Proceedings  
4593 of the 29th Conference on Neural Information Processing Systems*, Montreal, QC, Canada,  
4594 December 2015. ISBN 0262017091, 9780262017091. ISSN 1049-5258 pp. 2503–2511.
- 4595 [258] C. B. Seaman, “Qualitative methods,” in *Guide to Advanced Empirical Software Engineering*,  
4596 F. Shull, J. Singer, and D. I. K. Sjøberg, Eds. Springer, November 2007, ch. 2, pp. 35–62.  
4597 ISBN 978-1-84-800043-8
- 4598 [259] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-CAM:  
4599 Visual Explanations from Deep Networks via Gradient-Based Localization,” *International*

- 4600         *Journal of Computer Vision*, pp. 618–626, 2019, DOI 10.1007/s11263-019-01228-7. ISSN  
4601         1573-1405
- 4602         [260] S. Sen and L. Knight, “A genetic prototype learner,” in *Proceedings of the International Joint*  
4603         *Conference on Artificial Intelligence*. Montreal, QC, Canada: Morgan Kaufmann, August  
4604         1995, pp. 725–733.
- 4605         [261] C. E. Shannon and W. Weaver, “The mathematical theory of communication,” *The Bell*  
4606         *System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948, DOI 10.1002/j.1538-  
4607         7305.1948.tb01338.x.
- 4608         [262] M. Shaw, “Writing good software engineering research papers,” in *Proceedings of the 25th*  
4609         *International Conference on Software Engineering*. Portland, OR, USA: IEEE, May 2003.  
4610         ISBN 978-0-7695-1877-0 pp. 726–736.
- 4611         [263] M. Shepperd, “Replication studies considered harmful,” in *Proceedings of the 40th Interna-*  
4612         *tional Conference on Software Engineering*. Gothenburg, Sweden: ACM, May 2018.  
4613         DOI 10.1145/3183399.3183423. ISBN 978-1-4503-5662-6. ISSN 0270-5257 pp. 73–76.
- 4614         [264] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*. New York,  
4615         NY, USA: Chapman and Hall/CRC, 2004. DOI 10.4324/9780203489536.
- 4616         [265] L. Si and J. Callan, “A semisupervised learning method to merge search engine results,”  
4617         *ACM Transactions on Information Systems*, vol. 21, no. 4, pp. 457–491, October 2003,  
4618         DOI 10.1145/944012.944017. ISSN 1046-8188
- 4619         [266] J. Singer, S. E. Sim, and T. C. Lethbridge, “Software engineering data collection for field  
4620         studies,” in *Guide to Advanced Empirical Software Engineering*, F. Shull, J. Singer, and D. I. K.  
4621         Sjøberg, Eds. Springer, November 2007, ch. 1, pp. 9–34. ISBN 978-1-84800043-8
- 4622         [267] S. Singh, M. T. Ribeiro, and C. Guestrin, “Programs as Black-Box Explanations,” *arXiv preprint*  
4623         arXiv:1611.07579, November 2016.
- 4624         [268] V. S. Sinha, S. Mani, and M. Gupta, “Exploring activeness of users in QA forums,” in *Proced-*  
4625         *ings of the 10th Working Conference on Mining Software Repositories*. San Francisco, CA,  
4626         USA: IEEE, May 2013. DOI 10.1109/MSR.2013.6624010. ISBN 978-1-4673-2936-1. ISSN  
4627         2160-1852 pp. 77–80.
- 4628         [269] M. Sokolova and G. Lapalme, “A systematic analysis of performance measures for classifi-  
4629         cation tasks,” *Information Processing and Management*, vol. 45, no. 4, pp. 427–437, 2009,  
4630         DOI 10.1016/j.ipm.2009.03.002. ISSN 0306-4573
- 4631         [270] I. Sommerville, *Software Engineering*, 9th ed. Boston, MA, USA: Addison-Wesley, 2011.  
4632         ISBN 978-0-13-703515-1
- 4633         [271] P. Spector, *Summated Rating Scale Construction*. Newbury Park, CA, USA: SAGE, 1992.  
4634         DOI 10.4135/9781412986038. ISBN 978-0-80-394341-4
- 4635         [272] R. Stevens, J. Ganz, V. Filkov, P. Devanbu, and H. Chen, “Asking for (and about) permissions  
4636         used by Android apps,” in *Proceedings of the 10th Working Conference on Mining Software*  
4637         *Repositories*. San Francisco, CA, USA: IEEE, May 2013. ISBN 978-1-4673-2936-1 pp. 31–40.
- 4638         [273] J. Su, D. V. Vargas, and K. Sakurai, “One Pixel Attack for Fooling Deep Neural Net-  
4639         works,” *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019,  
4640         DOI 10.1109/TEVC.2019.2890858. ISSN 1941-0026
- 4641         [274] G. H. Subramanian, J. Nosek, S. P. Raghunathan, and S. S. Kanitkar, “A comparison of  
4642         the decision table and tree,” *Communications of the ACM*, vol. 35, no. 1, pp. 89–94, 1992,  
4643         DOI 10.1145/129617.129621. ISSN 1557-7317
- 4644         [275] S. Subramanian, L. Inozemtseva, and R. Holmes, “Live API documentation,” in *Proceedings*  
4645         *of the 36th International Conference on Software Engineering*. Hyderabad, India: ACM, May  
4646         2014. DOI 10.1145/2568225.2568313. ISSN 0270-5257 pp. 643–652.
- 4647         [276] S. Sun, W. Pan, and L. L. Wang, “A Comprehensive Review of Effect Size Reporting and Inter-  
4648         preting Practices in Academic Journals in Education and Psychology,” *Journal of Educational*  
4649         *Psychology*, vol. 102, no. 4, pp. 989–1004, 2010, DOI 10.1037/a0019507. ISSN 0022-0663
- 4650         [277] D. Szafron, P. Lu, R. Greiner, D. S. Wishart, B. Poulin, R. Eisner, Z. Lu, J. Anvik, C. Macdonell,  
4651         A. Fyshe, and D. Meeuwis, “Proteome Analyst: Custom predictions with explanations in a web-  
4652         based tool for high-throughput proteome annotations,” *Nucleic Acids Research*, vol. 32, 2004,  
4653         DOI 10.1093/nar/gkh485. ISSN 0305-1048

- 4654 [278] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus,  
4655 “Intriguing properties of neural networks,” in *Proceedings of the 2nd International Conference  
4656 on Learning Representations*. Banff, AB, Canada: ACM, April 2014.
- 4657 [279] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Archi-  
4658 tecture for Computer Vision,” in *Proceedings of the 2016 IEEE Computer Society Conference  
4659 on Computer Vision and Pattern Recognition*. Las Vegas, NV, USA: IEEE, June 2016.  
4660 DOI 10.1109/CVPR.2016.308. ISBN 978-1-46-738850-4. ISSN 1063-6919 pp. 2818–2826.
- 4661 [280] M. B. W. Tabor, “Student Proves That S.A.T. Can Be: (D) Wrong,” [Online] Available: <https://hyti.ms/2UiKrd>, New York, NY, USA, February 1997.
- 4663 [281] A. Tahir, A. Yamashita, S. Licorish, J. Dietrich, and S. Counsell, “Can you tell me if it  
4664 smells? A study on how developers discuss code smells and anti-patterns in Stack Overflow,”  
4665 in *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software  
4666 Engineering*. Christchurch, New Zealand: ACM, June 2018. DOI 10.1145/3210459.3210466.  
4667 ISBN 978-1-45-036403-4 pp. 68–78.
- 4668 [282] G. Tassey, *The economic impacts of inadequate infrastructure for software testing*. National  
4669 Institute of Standards and Technology, September 2002. DOI 10.1080/10438590500197315.  
4670 ISBN 978-0-75-672618-8
- 4671 [283] R. S. Taylor, “Question-Negotiation and Information Seeking in Libraries,” *College and Re-  
4672 search Libraries*, vol. 29, no. 3, 1968, DOI 10.5860/crl\_29\_03\_178.
- 4673 [284] S. W. Thomas, B. Adams, A. E. Hassan, and D. Blostein, “Studying software evolu-  
4674 tion using topic models,” *Science of Computer Programming*, vol. 80, pp. 457–479, 2014,  
4675 DOI 10.1016/j.scico.2012.08.003. ISSN 0167-6423
- 4676 [285] S. Thrun, “Is Learning The n-th Thing Any Easier Than Learning The First?” in *Proceedings  
4677 of the 8th International Conference on Neural Information Processing Systems*. Denver, CO,  
4678 USA: MIT Press, November 1996. ISSN 1049-5258 p. 7.
- 4679 [286] C. Treude, O. Barzilay, and M. A. Storey, “How do programmers ask and answer questions  
4680 on the web?” in *Proceedings of the 33rd International Conference on Software Engineering*.  
4681 Honolulu, HI, USA: ACM, May 2011. DOI 10.1145/1985793.1985907. ISBN 978-1-45-  
4682 030445-0. ISSN 0270-5257 pp. 804–807.
- 4683 [287] G. Uddin and F. Khomh, “Automatic Mining of Opinions Expressed About APIs in  
4684 Stack Overflow,” *IEEE Transactions on Software Engineering*, February 2019, In Press,  
4685 DOI 10.1109/TSE.2019.2900245. ISSN 1939-3520
- 4686 [288] G. Uddin and M. P. Robillard, “How API Documentation Fails,” *IEEE Software*, vol. 32, no. 4,  
4687 pp. 68–75, June 2015, DOI 10.1109/MS.2014.80. ISSN 0740-7459
- 4688 [289] M. Usman, R. Britto, J. Börstler, and E. Mendes, “Taxonomies in software engineering: A  
4689 Systematic mapping study and a revised taxonomy development method,” *Information and  
4690 Software Technology*, vol. 85, pp. 43–59, May 2017, DOI 10.1016/j.infsof.2017.01.006. ISSN  
4691 0950-5849
- 4692 [290] A. Van Assche and H. Blockeel, “Seeing the forest through the trees learning a comprehensible  
4693 model from a first order ensemble,” in *Proceedings of the 17th International Conference on  
4694 Inductive Logic Programming*. Corvallis, OR, USA: Springer, June 2007. DOI 10.1007/978-  
4695 3-540-78469-2\_26. ISBN 3-540-078468-3. ISSN 0302-9743 pp. 269–279.
- 4696 [291] B. Venners, “Design by Contract: A Conversation with Bertrand Meyer,” *Artima Developer*,  
4697 2003.
- 4698 [292] W. Verbeke, D. Martens, C. Mues, and B. Baesens, “Building comprehensible customer churn  
4699 prediction models with advanced rule induction techniques,” *Expert Systems with Applications*,  
4700 vol. 38, no. 3, pp. 2354–2364, 2011, DOI 10.1016/j.eswa.2010.08.023. ISSN 0957-4174
- 4701 [293] F. Wachter, Mitterstadt, “EU regulations on algorithmic decision-making and a “right to expla-  
4702 nation”,” in *Proceedings of the 2016 ICML Workshop on Human Interpretability in Machine  
4703 Learning*, New York, NY, USA, June 2016, pp. 26–30.
- 4704 [294] B. Wang, Y. Yao, B. Viswanath, H. Zheng, and B. Y. Zhao, “With great training comes great  
4705 vulnerability: Practical attacks against transfer learning,” in *Proceedings of the 27th USENIX  
4706 Security Symposium*. Baltimore, MD, USA: USENIX Association, July 2018. ISBN 978-1-  
4707 93-913304-5 pp. 1281–1297.
- 4708 [295] K. Wang, *Cloud Computing for Machine Learning and Cognitive Applications: A Machine  
4709 Learning Approach*. Cambridge, MA, USA: MIT Press, 2017. ISBN 978-0-26-203641-2

- [296] S. Wang, D. Lo, and L. Jiang, "An empirical study on developer interactions in StackOverflow," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. Coimbra, Portugal: ACM, March 2013. DOI 10.1145/2480362.2480557, pp. 1019–1024.
- [297] W. Wang and M. W. Godfrey, "Detecting API usage obstacles: A study of iOS and android developer questions," in *Proceedings of the 10th Working Conference on Mining Software Repositories*. San Francisco, CA, USA: IEEE, May 2013. DOI 10.1109/MSR.2013.6624006. ISBN 978-1-46-732936-1. ISSN 2160-1852 pp. 61–64.
- [298] W. Wang, H. Malik, and M. W. Godfrey, "Recommending Posts concerning API Issues in Developer Q&A Sites," in *Proceedings of the 12th Working Conference on Mining Software Repositories*. Florence, Italy: IEEE, May 2015. DOI 10.1109/MSR.2015.28. ISBN 978-0-7695-5594-2. ISSN 2160-1860 pp. 224–234.
- [299] 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*. Seattle, WA, USA: ACM, October 2012. DOI 10.1145/2379057.2379112. ISBN 978-1-45-031497-8 pp. 295–302.
- [300] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D. F. Ferguson, *Web Services Platform Architecture*. Crawfordsville, IN, USA: Prentice-Hall, 2005. ISBN 0-13-148874-0
- [301] D. Wettschereck, D. W. Aha, and T. Mohri, "A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms," *Artificial Intelligence Review*, vol. 11, no. 1-5, pp. 273–314, 1997, DOI 10.1007/978-94-017-2053-3\_11. ISSN 0269-2821
- [302] J. Wexler, M. Pushkarna, T. Bolukbasi, M. Wattenberg, F. Viegas, and J. Wilson, "The What-If Tool: Interactive Probing of Machine Learning Models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 56–65, 2019, DOI 10.1109/tvcg.2019.2934619. ISSN 1077-2626
- [303] H. Wickham, "A Layered grammar of graphics," *Journal of Computational and Graphical Statistics*, vol. 19, no. 1, pp. 3–28, January 2010, DOI 10.1198/jcgs.2009.07098. ISSN 1061-8600
- [304] R. J. Wieringa and J. M. G. Heerkens, "The methodological soundness of requirements engineering papers: A conceptual framework and two case studies," *Requirements Engineering*, vol. 11, no. 4, pp. 295–307, 2006, DOI 10.1007/s00766-006-0037-6. ISSN 0947-3602
- [305] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2016. DOI 10.1016/c2009-0-19715-5. ISBN 978-0-12-804291-5
- [306] C. Wohlin and A. Aurum, "Towards a decision-making structure for selecting a research design in empirical software engineering," *Empirical Software Engineering*, vol. 20, no. 6, pp. 1427–1455, May 2015, DOI 10.1007/s10664-014-9319-7. ISSN 1573-7616
- [307] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Berlin, Heidelberg: Springer, 2012. DOI 10.1007/978-3-642-29044-2. ISBN 978-3-64-229044-2
- [308] M. L. Wong and K. S. Leung, *Data Mining Using Grammar Based Genetic Programming and Applications*. Springer, 2002. DOI 10.1007/b116131. ISBN 978-0-79-237746-7
- [309] X. Yi and K. J. Kochut, "A CP-nets-based design and verification framework for web services composition," in *Proceedings of the 2004 IEEE International Conference on Web Services*. San Diego, CA, USA: IEEE, July 2004. DOI 10.1109/icws.2004.1314810. ISBN 0-76-952167-3 pp. 756–760.
- [310] R. K. Yin, *Case study research and applications: Design and methods*, 6th ed. Los Angeles, CA, USA: SAGE, 2017. ISBN 978-1-50-633616-9
- [311] J. Zahálka and F. Železný, "An experimental test of Occam's razor in classification," *Machine Learning*, vol. 82, no. 3, pp. 475–481, 2011, DOI 10.1007/s10994-010-5227-2. ISSN 0885-6125
- [312] J. Zhang and R. Kasturi, "Extraction of Text Objects in Video Documents: Recent Progress," in *Proceedings of the 8th International Workshop on Document Analysis Systems*. Nara, Japan: IEEE, September 2008. DOI 10.1109/das.2008.49, pp. 5–17.
- [313] B. Zupan, J. Demšar, M. W. Kattan, J. R. Beck, and I. Bratko, "Machine learning for survival analysis: a case study on recurrence of prostate cancer," *Artificial intelligence in medicine*, vol. 20, no. 1, pp. 59–75, 2000.

- <sup>4765</sup> [314] M. Zur Muehlen, J. V. Nickerson, and K. D. Swenson, “Developing web services choreography  
<sup>4766</sup> standards - The case of REST vs. SOAP,” *Decision Support Systems*, vol. 40, no. 1, pp. 9–29,  
<sup>4767</sup> July 2005, DOI 10.1016/j.dss.2004.04.008. ISSN 0167-9236



4768

---

4769

## List of Online Artefacts

4770

---

4771 The online artefacts listed below have been downloaded and stored on the Deakin  
4772 Research Data Store (RDS) for archival purposes at the following location:

4773 RDS29448-Alex-Cummaudo-PhD/datasets/webrefs

- 4774 [315] Affectiva, Inc., “Home - Affectiva : Affectiva,” <http://bit.ly/36sgbMM>, 2018, accessed: 15  
4775 October 2018.
- 4776 [316] Amazon Web Services, Inc., “Detecting Labels in an Image,” <https://amzn.to/2TBNtTa>, 2018,  
4777 accessed: 28 August 2018.
- 4778 [317] ——, “Detecting Objects and Scenes,” <https://amzn.to/2TDed5V>, 2018, accessed: 28 August  
4779 2018.
- 4780 [318] ——, “Amazon Rekognition,” <https://amzn.to/2TyT2BL>, 2018, accessed: 13 September 2018.
- 4781 [319] Beijing Geling Shentong Information Technology Co., Ltd., “DeepGlint,” <http://bit.ly/2uHHdPS>, 2018, accessed: 3 April 2019.
- 4782 [320] Beijing Kuangshi Technology Co., Ltd., “Megvii,” <http://bit.ly/2WJYFzk>, 2018, accessed: 3  
4783 April 2019.
- 4784 [321] Clarifai, Inc., “Enterprise AI Powered Computer Vision Solutions | Clarifai,” <http://bit.ly/2TB3kSa>, 2018, accessed: 13 September 2018.
- 4785 [322] CloudSight, Inc., “Image Recognition API & Visual Search Results | CloudSight AI,” <http://bit.ly/2UmNPCw>, 2018, accessed: 13 September 2018.
- 4786 [323] Cognitec Systems GmbH, “The face recognition company - Cognitec,” <http://bit.ly/38VguBB>,  
4787 2018, accessed: 15 October 2018.
- 4788 [324] A. Cummaudo, <http://bit.ly/2KlyhcD>, 2019, accessed: 27 March 2019.
- 4789 [325] ——, <http://bit.ly/2G7saFJ>, 2019, accessed: 27 March 2019.
- 4790 [326] ——, <http://bit.ly/2G5ZEEe>, 2019, accessed: 27 March 2019.
- 4791 [327] ——, “ICSE 2020 Submission #564 Supplementary Materials,” <http://bit.ly/2Z8zOKW>, 2019.
- 4792 [328] ——, <http://bit.ly/2G6ZOeC>, 2019, accessed: 27 March 2019.
- 4793 [329] Deep AI, Inc., “DeepAI: The front page of A.I. | DeepAI,” <http://bit.ly/2TBNYgf>, 2018, ac-  
4794 cessed: 26 September 2018.
- 4795 [330] Google LLC, “Detect Labels | Google Cloud Vision API Documentation | Google Cloud,”  
4796 <http://bit.ly/2TD5kcy>, 2018, accessed: 28 August 2018.
- 4797 [331] ——, “Vision API - Image Content Analysis | Cloud Vision API | Google Cloud,” <http://bit.ly/2TD9mBs>, 2018, accessed: 13 September 2018.
- 4798 [332] ——, “Open Images Dataset V4,” <http://bit.ly/2Ry2vvF>, 2019, accessed: 9 November 2018.

- 4803 [333] Guangzhou Tup Network Technology, “TupuTech,” <http://bit.ly/2uF4IsN>, 2018, accessed: 3 April 2019.
- 4804 [334] 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.
- 4805 [335] International Business Machines Corporation, “Watson Visual Recognition - Overview | IBM,” <https://ibm.co/2TBNIO4>, 2018, accessed: 13 September 2018.
- 4806 [336] ——, “Watson Tone Analyzer,” <https://ibm.co/37w3y4A>, 2019, accessed: 25 January 2019.
- 4807 [337] Kairos AR, Inc., “Kairos: Serving Businesses with Face Recognition,” <http://bit.ly/30WHGNs>, 2018, accessed: 15 October 2018.
- 4808 [338] Microsoft Corporation, “azure-sdk-for-java/ImageTag.java,” <http://bit.ly/38IDPWU>, 2018, accessed: 28 August 2018.
- 4809 [339] ——, “Image Processing with the Computer Vision API | Microsoft Azure,” <http://bit.ly/2YqhkS6>, 2018, accessed: 13 September 2018.
- 4810 [340] ——, “How to call the Computer Vision API,” <http://bit.ly/2TD5oJk>, 2018, accessed: 28 August 2018.
- 4811 [341] ——, “What is Computer Vision?” <http://bit.ly/2TDgUnU>, 2018, accessed: 28 August 2018.
- 4812 [342] SenseTime, “SenseTime,” <http://bit.ly/2WH6Rjf>, 2018, accessed: 3 April 2019.
- 4813 [343] Shanghai Yitu Technology Co., Ltd., “Yitu Technology,” <http://bit.ly/2uGvxgf>, 2018, accessed: 3 April 2019.
- 4814 [344] Stack Overflow User #1008563 ‘samiles’, “AWS Rekognition PHP SDK gives invalid image encoding error,” <http://bit.ly/31Sgpec>, 2019, accessed: 22 June 2019.
- 4815 [345] Stack Overflow User #10318601 ‘reza naderii’, “google cloud vision category detecting,” <http://bit.ly/31Uf32t>, 2019, accessed: 22 June 2019.
- 4816 [346] Stack Overflow User #10729564 ‘gabgob’, “Multiple Google Vision OCR requests at once?” <http://bit.ly/31P09dU>, 2019, accessed: 22 June 2019.
- 4817 [347] Stack Overflow User #1453704 ‘deoptimancode’, “Human body part detection in Android,” <http://bit.ly/31T5pxd>, 2019, accessed: 22 June 2019.
- 4818 [348] Stack Overflow User #174602 ‘geekyaleks’, “aws Rekognition not initializing on iOS,” <http://bit.ly/31UeqG9>, 2019, accessed: 22 June 2019.
- 4819 [349] Stack Overflow User #2251258 ‘James Dorfman’, “All GoogleVision label possibilities?” <http://bit.ly/31R4FZi>, 2019, accessed: 22 June 2019.
- 4820 [350] 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.
- 4821 [351] Stack Overflow User #2604150 ‘user2604150’, “Google Vision Accent Character Set NodeJs,” <http://bit.ly/31TsVdp>, 2019, accessed: 22 June 2019.
- 4822 [352] 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.
- 4823 [353] Stack Overflow User #3565255 ‘CSquare’, “Vision API topicality and score always the same,” <http://bit.ly/2TD5As2>, 2019, accessed: 22 June 2019.
- 4824 [354] Stack Overflow User #4748115 ‘Latifa Al-jiffry’, “similar face recognition using google cloud vision API in android studio,” <http://bit.ly/31WhMZy>, 2019, accessed: 22 June 2019.
- 4825 [355] Stack Overflow User #4852910 ‘Gaurav Mathur’, “Amazon Rekognition Image caption,” <http://bit.ly/31P08qm>, 2019, accessed: 22 June 2019.
- 4826 [356] 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.
- 4827 [357] Stack Overflow User #549312 ‘GroovyDotCom’, “Image Selection for Training Visual Recognition,” <http://bit.ly/31W8lcw>, 2019, accessed: 22 June 2019.
- 4828 [358] 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.
- 4829 [359] 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.
- 4830 [360] Stack Overflow User #5924523 ‘Akash Dathan’, “Can i give aspect ratio in Google Vision api?” <http://bit.ly/2KSJwsp>, 2019, accessed: 22 June 2019.

- 4858 [361] Stack Overflow User #6210900 ‘Mike Grommet’, “Are the Cloud Vision API limits in documentation correct?” <http://bit.ly/31SsNLg>, 2019, accessed: 22 June 2019.
- 4859 [362] 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.
- 4860 [363] Stack Overflow User #6841211 ‘NigelJL’, “Google Cloud Vision - Numbers and Numerals OCR,” <http://bit.ly/31P07mi>, 2019, accessed: 22 June 2019.
- 4861 [364] 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.
- 4862 [365] Stack Overflow User #7187987 ‘tuanars10’, “Adding a local path to Microsoft Face API by Python,” <http://bit.ly/2KLeMt3>, 2019, accessed: 22 June 2019.
- 4863 [366] Stack Overflow User #7219743 ‘Davide Biraghi’, “Google Vision API does not recognize single digits,” <http://bit.ly/31Ws1Nj>, 2019, accessed: 22 June 2019.
- 4864 [367] 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.
- 4865 [368] Stack Overflow User #7604576 ‘Alagappan Narayanan’, “Text extraction - line-by-line,” <http://bit.ly/31Yc21s>, 2019, accessed: 22 June 2019.
- 4866 [369] 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.
- 4867 [370] Stack Overflow User #7896427 ‘David mark’, “Google Api Vision, ““before\_request”” error,” <http://bit.ly/31Z27Zt>, 2019, accessed: 22 June 2019.
- 4868 [371] Stack Overflow User #8210103 ‘Cosmin-Ioan Leferman’, “Google Vision API text detection strange behaviour - Javascript,” <http://bit.ly/31Ucyxi>, 2019, accessed: 22 June 2019.
- 4869 [372] 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.
- 4870 [373] 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.
- 4871 [374] 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.
- 4872 [375] Stack Overflow User #9913535 ‘Sahil Mehra’, “Google Vision API: ModuleNotFoundError: module not found ‘google.oauth2’,” <http://bit.ly/31VIZfU>, 2019, accessed: 22 June 2019.
- 4873 [376] Symisc Systems, S.U.A.R.L, “Computer Vision & Media Processing APIs | PixLab,” <http://bit.ly/2UlkW9K>, 2018, accessed: 13 September 2018.
- 4874 [377] Talkwalker Inc., “Image Recognition - Talkwalker,” <http://bit.ly/2TyT7W5>, 2018, accessed: 13 September 2018.
- 4875 [378] TheySay Limited, “Sentiment Analysis API | TheySay,” <http://bit.ly/37AzTHI>, 2019, accessed: 25 January 2019.



## **Part IV**

# **Appendices**



## APPENDIX A

---

### Additional Materials

---

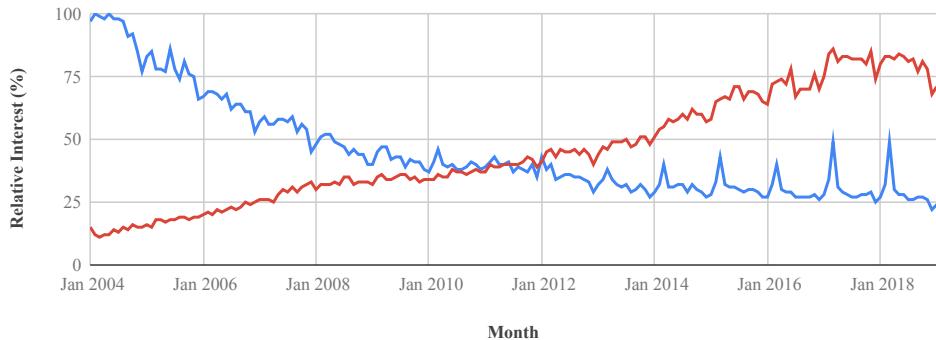


## A.1 The Development, Documentation and Usage of Web APIs

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 [248]. This framework used the synchronous communication paradigm remote procedure calls (RPCs) first introduced by Nelson [207] that allows procedures to be called in a remote address space as if it were local. Its communication paradigm, DCE/RPC [214], 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 [214]). 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 [295, 300]). These attributes are included within SOAP headers; thus, QoS aspects are independent from the transport layer and instead exist at the application layer [221]. 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 [221, 314] 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: [?CITE?].

```

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: [?CITE?].

```

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 [221]. That is, the HTTP is considered as a transport protocol solely. In 2000, Fielding [94] 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) [?CITE?]). Therefore, no RESTful service uses a standardised response document or invocation syntax. While there are proposals, such as WADL [116], RAML<sup>1</sup>, API Blueprint<sup>2</sup>, and the OpenAPI<sup>3</sup> specification (initially based on Swagger<sup>4</sup>), 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: [?CITE?].

```
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"}
```

---

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

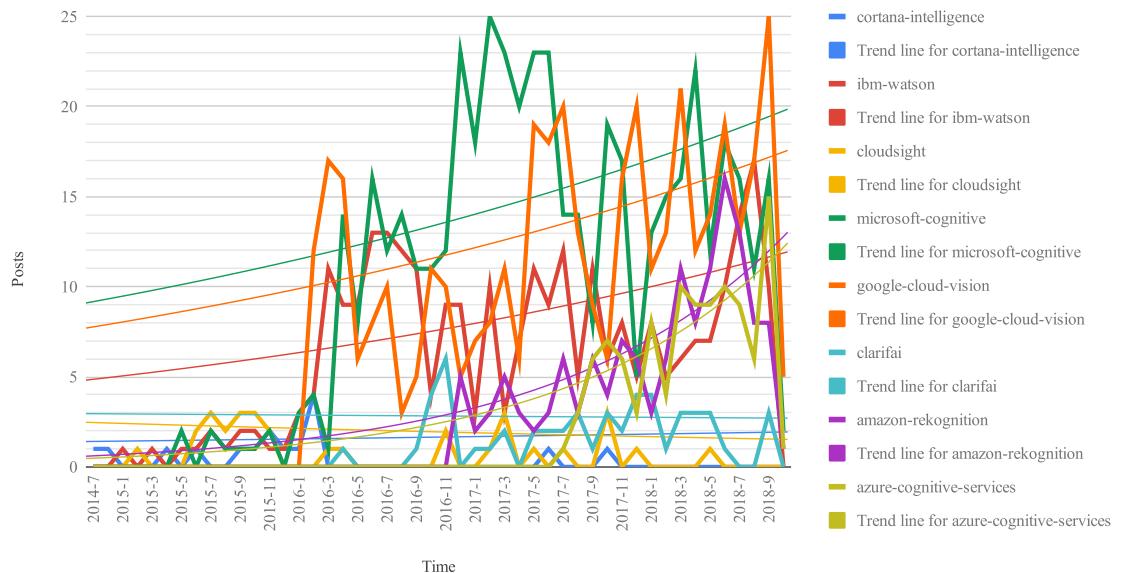
<sup>2</sup><https://apiblueprint.org> last accessed 25 January 2019.

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

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

**Figure A.2:** A Broad Range of artificial intelligence (AI)-Based Products And Services Is Already Visible. (From [178].)

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

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

**Figure A.4:** Questions asked by software engineering researchers (column 2) that can be answered by field study techniques. (From [266].)

Technique	Used by researchers when their goal is to understand:	Volume of data	Also used by software engineers for
<b>Direct techniques</b>			
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	
<b>Indirect techniques</b>			
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	
<b>Independent techniques</b>			
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.

## **APPENDIX B**

---

### **Authorship Statements**

---



## Deakin University Authorship Procedure

### Schedule A: Authorship Statement

#### **1. Details of the publication and executive author**

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

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

<b>Name and affiliation of author 1</b>	Alex Cummaudo Applied Artificial Intelligence Institute Deakin University
<b>Contribution of author 1</b>	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.
<b>Name and affiliation of author 2</b>	Rajesh Vasa Applied Artificial Intelligence Institute Deakin University
<b>Contribution of author 2</b>	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.
<b>Name and affiliation of author 3</b>	John Grundy Faculty of Information Technology Monash University
<b>Contribution of author 3</b>	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.
<b>Name and affiliation of author 4</b>	Mohamed Abdelrazek School of Information Technology Deakin University
<b>Contribution of author 4</b>	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.
<b>Name and affiliation of author 5</b>	Andrew Cain School of Information Technology Deakin University
<b>Contribution of author 5</b>	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

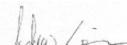
### Author 4

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.)

<b>Data format</b>	Comma separated values (CSV), iPython Notebook
<b>Storage location</b>	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**

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

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

<b>Name and affiliation of author 1</b>	Alex Cummaudo Applied Artificial Intelligence Institute Deakin University
<b>Contribution of author 1</b>	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.
<b>Name and affiliation of author 2</b>	Rajesh Vasa Applied Artificial Intelligence Institute Deakin University
<b>Contribution of author 2</b>	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.
<b>Name and affiliation of author 3</b>	John Grundy Faculty of Information Technology Monash University
<b>Contribution of author 3</b>	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.)

<b>Data format</b>	Comma separated values (CSV), Portable Document Format (PDF)
<b>Storage location</b>	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**

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

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

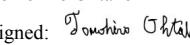
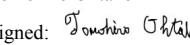
I agree to be named as one of the authors of this work, and confirm:

- 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.)

<b>Data format</b>	Comma separated values (CSV)
<b>Storage location</b>	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 (A<sup>2</sup>I<sup>2</sup>)  
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](mailto: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 (A<sup>2</sup>I<sup>2</sup>) (G)

23<sup>rd</sup> 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](mailto: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.