

# ***Magic beyond the method call:zA communication framework for application developers using ‘friendly’ machine learning***

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

---

## Confirmation of Candidature Report

### **Supervisory Panel:**

Prof. Rajesh Vasa <[rajesh.vasa@deakin.edu.au](mailto:rajesh.vasa@deakin.edu.au)>

Prof. John Grundy <[john.grundy@monash.edu](mailto:john.grundy@monash.edu)>

A/Prof. Mohamed Abdelrazek <[mohamed.abdelrazek@deakin.edu.au](mailto:mohamed.abdelrazek@deakin.edu.au)>

A/Prof. Andrew Cain <[andrew.cain@deakin.edu.au](mailto:andrew.cain@deakin.edu.au)>

---



Applied Artificial Intelligence Institute  
Deakin University  
Melbourne, Australia

February 1, 2019



# 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.



# Contents

<b>Abstract</b>	<b>iii</b>
<b>Contents</b>	<b>iii</b>
<b>List of Abbreviations</b>	<b>vi</b>
<b>List of Figures</b>	<b>viii</b>
<b>List of Tables</b>	<b>ix</b>
<b>List of Listings</b>	<b>xi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation: Current Developer Mindsets’ . . . . .	5
1.1.1 The Impact on Software Quality . . . . .	6
1.1.2 Motivating Scenarios . . . . .	7
1.2 Research Outcomes . . . . .	12
1.3 Concluding Remarks . . . . .	15
<b>2 Background</b>	<b>17</b>
2.1 Software Quality . . . . .	18
2.1.1 Validation and Verification . . . . .	19
2.1.2 Quality Attributes and Models . . . . .	22
2.1.3 Reliability in Computer Vision . . . . .	24
2.2 Probabilistic and Nondeterministic Systems . . . . .	26
2.2.1 Interpreting the Uninterpretable . . . . .	26
2.2.2 Explanation and Communication . . . . .	28
2.2.3 Mechanics of Model Interpretation . . . . .	28
2.3 Application Programming Interfaces . . . . .	29
2.3.1 API Usability . . . . .	30

<b>3 Research Methodology</b>	<b>33</b>
3.1 Research Questions Revisited . . . . .	33
3.1.1 Knowledge Questions . . . . .	34
3.1.2 Design Questions . . . . .	35
3.2 Philosophical Stances . . . . .	35
3.3 Research Design . . . . .	36
3.3.1 Review of Relevant Research Methods . . . . .	36
3.3.2 Review of Data Collection Techniques for Field Studies . . . . .	39
3.4 Proposed Experiments . . . . .	39
3.4.1 Experiment I: Develop Initial Framework . . . . .	39
3.4.2 Developing the Initial Framework . . . . .	42
3.4.3 Experiment II: Validate Initial Framework . . . . .	42
3.5 Empirical Validity . . . . .	44
3.5.1 Threats to Internal Validity . . . . .	44
3.5.2 Threats to External Validity . . . . .	46
3.5.3 Threats to Construct Validity . . . . .	46
<b>4 Project Status</b>	<b>49</b>
4.1 Project Timeline . . . . .	49
4.2 Completed Work . . . . .	51
4.3 Proposed Chapters . . . . .	51
<b>References</b>	<b>74</b>
<b>A Additional Material</b>	<b>75</b>
A.1 The Development, Documentation and Usage of Web APIs . . . . .	77
<b>B Technical Report Manuscript</b>	<b>85</b>
<b>C Ethics Application Draft</b>	<b>97</b>

# List of Abbreviations

**A<sup>2</sup>I<sup>2</sup>** Applied Artificial Intelligence Institute. 38, 42, 45, 46

**AI** Artificial Intelligence. iii, 1, 2, 26–28, 81

**API** Application Programming Interface. 2–15, 17, 20, 21, 29–31, 33, 34, 36, 38–43, 46, 49, 77, 79

**AWS** Amazon Web Services. 1

**BYOML** Build Your Own Machine Learning. 2, 4

**CDSS** Clinical Decision Support System. 7, 10

**CIS** Cloud Intelligence Service. 3, 5–9, 11–14, 17–21, 23, 24, 26, 29, 31, 33–43, 46, 49

**CNN** Convolutional Neural Network. 10, 11, 24

**CRUD** Create, read, update, and delete. 79

**cvCIS** Computer Vision Cloud Intelligence Service. 5–9, 12, 13, 15, 17, 19, 20, 24, 30, 39, 43, 51

**DCE** Distributed Computing Environment. 77

**HITL** Human-in-the-loop. 11

**HTTP** Hypertext Transfer Protocol. 4, 77–80

**IDE** Integrated Development Environment. 1

**IDL** Interface Definition Language. 77, 79

**JSON** JavaScript Object Notation. 3

**ML** Machine Learning. iii, 1–4, 10, 21, 27

**NN** Neural Network. 25–27, 29

**QoS** Quality of Service. 77

**RAML** RESTful API Modeling Language. 79

**REST** REpresentational State Transfer. 3, 78–80

**ROI** Region of Interest. 10, 11

**RPC** Remote Procedure Call. 77

**SE** Software Engineer. 28

**SOA** Service-Level Agreement. 77

**SOA** Service-Oriented Architecture. 77

**SOAP** Simple Object Access Protocol. 3, 77–79

**SQuaRE** Systems and software Quality Requirements and Evaluation. 23

**SVM** Support Vector Machine. 26, 29

**URI** Uniform Resource Identifier. 79

**V&V** Verification & Validation. 17–22

**WADL** Web Application Description Language. 79

**WS** Web Service. 77

**WSDL** Web Services Description Language. 77

**XML** eXtendable Markup Language. 3, 77, 78

# List of Figures

1.1	Data-driven cloud services are not the same as rule-driven ones . . . . .	2
1.2	Examples of the machine learning spectrum in computer vision . . . . .	3
1.3	Overview of cloud intelligence services . . . . .	5
1.4	CancerAssist Context Diagram . . . . .	11
2.1	Mindset clashes within the development, use and nature of a CIS . . . . .	18
2.2	Leakage of internal and external quality in CISs . . . . .	21
2.3	The brief overview of the development of software quality models since 1977.	22
2.4	Sample adversarial examples . . . . .	25
2.5	Deterministic versus nondeterministic systems . . . . .	26
2.6	Theory of AI communication . . . . .	29
3.1	High-level overview of the proposed experiments . . . . .	40
4.1	PhD project timeline . . . . .	50
A.1	SOAP versus REST search interest over time . . . . .	78
A.2	Categorisation of AI-based products and services . . . . .	81
A.3	Increasing interest in the developer community of computer vision APIs . . .	82
A.4	Review of field study techniques . . . . .	83



# List of Tables

1.1	Differing characteristics of cloud services . . . . .	4
1.2	Comparison of the machine learning spectrum . . . . .	4
1.3	Varying confidence changes over time between 3 CV APIs . . . . .	8
1.4	Tautological definitions of confidence found in API documentation . . . . .	9
4.1	Mapping of project tasks to research questions . . . . .	52
4.2	Potential publications and venues . . . . .	52



# List of Listings

A.1	An example SOAP request . . . . .	78
A.2	An example SOAP response . . . . .	78
A.3	An example RESTful request . . . . .	80
A.4	An example RESTful response . . . . .	80



# Chapter 1

## Introduction

Within the last half-decade, we have seen an explosion of cloud-based services typically marketed under an Artificial Intelligence (AI) banner. Vendors are rapidly pushing out AI-based solutions, technologies and products that encapsulate half a century worth of machine-learning research: a 2016 report by market research company Forrester captured such growth into four key areas [149] as replicated in Figure A.2. Application developers are eager to develop the next generation of ‘AI-first’ software, that will reason, sense, think, act, listen, speak and execute every whim in our web browser or smartphone app.

A contrast is how these systems shift away from the traditional software engineering paradigm that application developers are comfortable with. Typical systems built by application developers are *rule-driven* (or algorithm-driven), deterministically human engineered using source code to drive each step behind the application. These rule-driven systems typically consume, utilise, and integrate libraries and frameworks, IDEs and other tooling, and cloud-based services such as Amazon Web Services (AWS) [3]. The AI-first software is, however, not rule-driven but *data-driven*, fuelled with large datasets used to train Machine Learning (ML) prediction algorithms and classifiers that present a typically probabilistic and nondeterministic behaviour.

So, how does the application developer approach her AI-first application?

1. She writes her own ML classifier from scratch and trains it from her own curated dataset.  
This approach is laborious in time and demands formal training in ML and mathematical knowledge, but the tradeoff is that she has full autonomy over the models she creates.
2. She downloads a pre-trained model and ‘plugs’ it into an existing ML framework, such as Tensorflow [25]. While this approach is less demanding in time, it still requires an understanding of how to ‘glue’ the ML framework code<sup>1</sup> with her own application code.

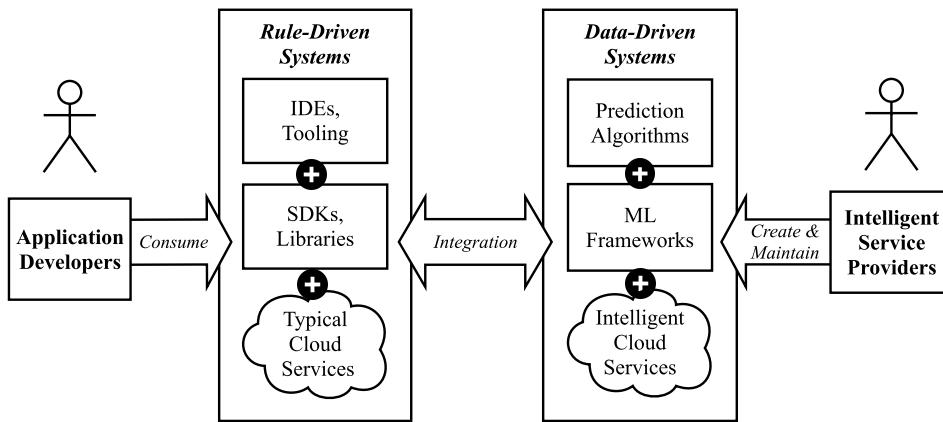
---

<sup>1</sup>Thus introducing a verbose list of ML terminology to her already-required developer vocabulary. See a list of 328 terms provided by Google here: <https://developers.google.com/machine-learning/glossary/>.

3. She uploads her data to a pre-existing cloud-based service: a *Friendly Machine Learning* service. She doesn't need to know anything behind the underlying 'intelligence' and how it works, is fast to integrate into her application and all abstracted behind a web-based Application Programming Interface (API) call.

At first sight, she perceives the data-driven cloud service as 'just another' cloud service offered in her toolchain. Her perception is that just because this is another cloud service, it should act and behave as any other typical service would. But internally, she isn't aware that this particular cloud service isn't a typical one; the data-driven service does not adapt to her rule-driven application in the way that she thinks (Figure 1.1). This is because the nature of typical cloud systems and data-driven ones are not exactly the same (Table 1.1).

**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.

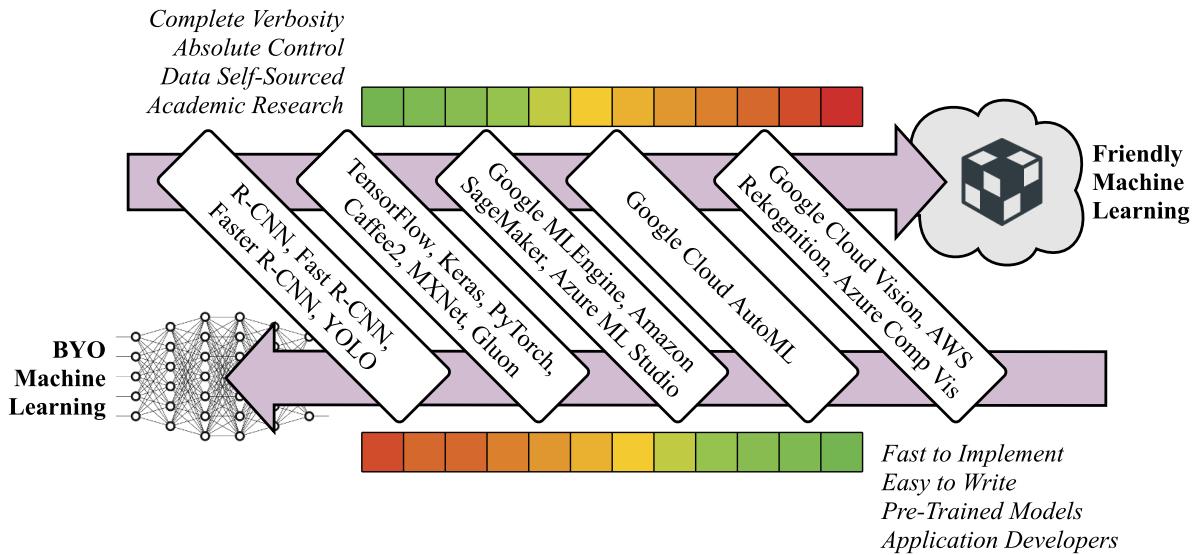


This 'range' of AI-first integration techniques partially reflects Google AI's<sup>2</sup> *Machine Learning Spectrum* [171, 138, 21], which encompasses the variety of skill, effort, users and types of outputs of integration techniques. On one extreme is the research of developing algorithms to achieve intelligence, produced chiefly in academia—coined as Build Your Own Machine Learning (BYOML) [171, 21, 18]. On the other, such intelligence becomes heavily abstracted as easy-to-use APIs, targeted mainly towards developers as 'friendly' ML. In the middle lies a broad mix of combining both cloud and locally-hosted solutions (with varying levels of automation to assist in development) that turn custom datasets into some form of predictive intelligence.

<sup>2</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 can be seen through Google's 2018 rebranding of *Google Research* to *Google AI* [107] or how AI is leveraged *at scale* within Facebook's infrastructure and platforms to serve its users with an AI-first attitude [175].

All techniques in this spectrum are data-driven, and we illustrate their slightly varied characteristics further in Table 1.2 and examples of the computer vision spectrum in Figure 1.2.

**Figure 1.2:** Examples within the machine learning spectrum of computer vision. Benefits and drawbacks of each end of the spectrum are indicated with the colour scales.



In this study, we advocate that the (i) integration, (ii) documentation, and (iii) quality attributes of data-driven cloud services juxtaposes the rule-driven nature of end-applications as these intelligent cloud services are vastly different to their traditional counterparts, and great care must therefore be considered by application developers.

These cloud services have begun to gain traction within developer circles: Figure A.3 shows the increasing trend of posts since 2014 on Stack Overflow that categorise popular computer vision cloud APIs.<sup>3</sup> In academia, these ‘off-the-shelf’ and pre-packaged ML solutions present a varied nomenclature such as *Cognitive Applications* and *Machine Learning Services* [110] or *Machine Learning as a Service* [190]. Some services provide the infrastructure to rapidly begin training from custom datasets (Google’s AutoML<sup>4</sup> is one such example) while others provide pre-trained datasets ‘ready-for-use’ in production without the need to train data. We refer to these latter services under the broader term *Cloud Intelligence Service (CIS)*, and diagrammatically express their usage within Figure 1.3.

The general workflow of a CIS is simple: a developer accesses a CIS component via REST/SOAP API(s). For their given input, they receive an intelligent-like response typically serialised as JSON/XML. We note the intelligence component masks its ‘intelligence’

<sup>3</sup>Query run on 12 October 2018 using StackExchange Data Explorer. Refer to <https://data.stackexchange.com/stackoverflow/query/910188> for full query.

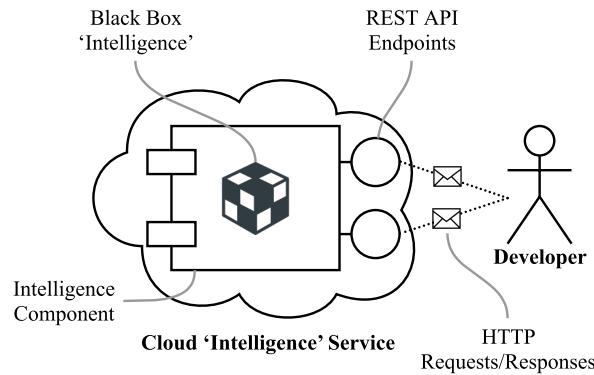
<sup>4</sup><https://cloud.google.com/automl/> last accessed 7 December 2018.

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

Intelligent Cloud Services	Typical Cloud Services
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				✓	✓

**Figure 1.3:** Overview of Cloud Intelligence Services.

through a black-box: in recent years, there is a rise in providing human-level intelligence via crowdsourcing Internet marketplaces such as Amazon Mechanical Turk [20] or ScaleAPI [23]. Thus, a CIS may be powered by varying degrees of intelligence: human intelligence, machine learning, data mining or even intelligence by brute-force.

While there are different types of CISs evident (such as OCR transcription, text-to-speech and speech-to-text, object categorisation, object comparison, natural language processing etc.), we scope the work investigated in this study to Computer Vision Cloud Intelligence Services (cvCISs) [15, 8, 3, 22, 16, 10, 9, 12, 17, 24, 19, 11, 4]. The ubiquity of cvCISs is exemplified through evermore growing applications that use these APIs: aiding the vision-impaired [188, 67], accounting [151], data analytics [116], and student education [72]. Moreover, we refer to its growing adoption in developer circles within Figure A.3.

## 1.1 Motivation: Current Developer Mindsets'

Figure A.3 shows an increasing trend to the adoption and discussion of CISs with developers. These services are accessible through APIs and consist of an ‘intelligence’ black box (Figure 1.3). When a term ‘black box’ is used, the input (or stimulus) is transformed to its outputs (or response) without any understanding of the internal architecture by which this transformation occurs, a theory arising from the electronic sciences and adapted to wider applications since the 1950s–60s [30, 51] to describe “systems whose internal mechanisms are not fully open to inspection” [30].

In the world of machine learning and data mining, where we develop algorithms to make predictions in our datasets or discover patterns within them, these black boxes are inherently probabilistic and stochastic; there is little room for certainty in these results as such insight is purely statistical and associational [179] against its training dataset. As an example, a

cvCIS returns the *probability* that a particular object (the response) exists in the raw pixels (the stimulus), and thus for a more certain (though not fully certain) distribution of overall confidence returned from the service, a developer must treat the problem stochastically by testing this case hundreds if not thousands of times to find a richer interpretation of the inference made. Developers (at present) do not need to treat their programs in any such stochastic way given their rule-driven mindset that computers will always make certain outcomes.

There are thus therefore three key factors to consider when implementing, testing and developing with a CIS: (i) the API usability, (ii) the nature of nondeterministic and probabilistic systems, and (iii) how both impact on software quality.

### 1.1.1 The Impact on Software Quality

Do traditional techniques for documenting deterministic APIs also apply to non-deterministic systems? As APIs reflect a set of design choices made by their providers intended for use by the developer, does the mindset between the machine learning architect and the novice programmer match? Evaluations of API usability advocate for the accuracy, consistency and completeness of APIs and their documentation [182, 195] written by providers, while providers should consider mismatches between the developer’s conceptual knowledge of the API its implementation [132]. However, consistency cannot be guaranteed in probabilistic systems, and the conceptual knowledge of such systems are still treated like black boxes. It is therefore imperative that CIS providers consider the impact of their API usability; if not, poor API usability hinders on the internal quality of development practices, slowing developers down to produce the software they need to create.

Moreover, CIS APIs are inherently non-deterministic in nature, but developers are still taught with the deterministic mindset that all API calls are the same. Simple arithmetic representations (e.g.,  $2 + 2 = 4$ ) will *always* result in 4; but a multi-layer perceptron neural network performing similar arithmetic representation [43] gives the probability where the target output (*exactly* 4) and the output inferred (*possibly* 4) matches as a percentage (or as an error where it does not match). That is, instead of an exact output, there is instead a *probabilistic* result:  $2 + 2$  *may* equal 4 with a confidence of  $n$ . External quality must therefore be considered in the outcome of these systems, such as in the case of thresholding values, to consider whether or not the inference has a high enough confidence to justify its result to end-users.

In order to fully understand this problem, there are multiple dimensions one must consider: the impact of software quality; the fact that these systems underneath are probabilistic and are stochastic; the cognitive biases of determinism in developers; the issue of consistency in API

usage. While existing literature does extensively explore software quality and API usability, these studies have only had emphasis on deterministic systems and thus little work to date has investigated such factors on probabilistic systems that make up the core of cvCISs. We explore more of these facets in the motivating scenarios below.

### 1.1.2 Motivating Scenarios

The market for intelligent services is increasing (Figure A.2) and as is developer uptake and enthusiasm in the software engineering community (Figure A.3). We investigate the impact of the mismatch between the developer’s mindset and the service provider’s mindset as little work has been presented in literature. How do developers work with a CIS, how usable are these cloud APIs, and how well do developers understand the non-deterministic and stochastic nature of a services backed by machine-learnt models?

To illustrate the context of use, we present the two scenarios of varying risk: (i) a fictional software developer named Tom who wishes to develop an inherently low-risk photo detection application for his friends and family; and (ii) a high-risk cancer Clinical Decision Support System (CDSS) that uses patient scans to recommend to surgeons if the patient should be sent to surgery.

#### Motivating Scenario I: Tom’s *PhotoSharer* App

Tom wants to develop a social media photo-sharing app on iOS and Android, *PhotoSharer*, that analyses photos taken on smartphones as they are taken. Tom wants the app to categorise photos into scenes (e.g., day vs. night, landscape vs. indoors), generate brief descriptions of each photo, and catalogue photos of his friends as well as common objects (e.g., all photos with his Border Collie dog, all photos taken on a beach on a sunny day). His app will then share all of this analysed intelligence of his photos with his friends on a social-media-like platform, where his friends can search and view the photos.

Rather than building a computer vision engine from scratch, which would take far too much time and effort, Tom thinks he can achieve this using one of the common cvCISs. Tom comes from a typical software engineering background and has insufficient knowledge of key computer vision terminology and no understanding of its underlying techniques. However, inspired by easily accessible cloud APIs that offer computer vision analysis, he chooses to use these. Built upon his experience of using other similar cloud services, he decides on one of the cvCIS APIs, and expects a static result always and consistency between similar APIs. Analogously, when Tom invokes the iOS Swift substring method `"doggy".prefix(3)`, he rightfully expects it to be consistent with the Android Java equivalent `"doggy".substring(0, 2)`.

Consistent, here, means two things: (i) that ‘dog’ will *always* be returned every time he invokes the method in either language (i.e., a static response); and (ii) that ‘dog’ will *always* be returned regardless of what programming language or string library is used, given the deterministic nature of the ‘substring’ construct (i.e., results for substring are API-agnostic).

**Table 1.3:** First six responses of image analysis for a Border Collie sent to three computer vision CIS APIs 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.

<b>Label</b>	<b>Provider A</b>		<b>Provider B</b>		<b>Provider C</b>	
	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>

More concretely, in Table 1.3, we illustrate how three cvCIS providers (anonymised) fail to provide similar consistency to that of the substring example above. If Tom uploads a photo of a border collie<sup>5</sup> to three different providers in August 2018 and January 2019, he would find that each provider is different in both the vocabulary used between. The confidence values and labels within the *same* provider also vary within a matter of five months. The evolution of the confidence changes is not explicitly documented by the providers (i.e., when the models change) nor do they document what confidence even means. Their current tautological nature of the definition of these changing confidence values (as presented in the API documentation) provides no insight for Tom to understand why there was a change in confidence, which we show in Table 1.4, unless he *knows* that the underlying models change with them. Thus, the

<sup>5</sup>The image used for these results can be found at <https://www.akc.org/dog-breeds/border-collie/>.

deterministic problem of a substring compared to the nondeterministic nature of the CIS is not so simple to comprehend unless he is explicitly told.

**Table 1.4:** Tautological definitions of ‘confidence’ found in the API documentation of three common cvCIS 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).” [13]
	“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].” [14]
	“The overall score of the result. Range [0, 1]” [14]
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.” [6]
	“The level of confidence the service has in the caption.” [7]
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.” [1]
	“[Provider C] also provide[s] a percentage score for how much confidence [Provider C] has in the accuracy of each detected label.” [2]

To make an assessment of these APIs, he tries his best to read through the documentation of some cvCIS APIs, but he has no guiding framework to help him choose the right one. Some of the questions that come to mind include:

- What does confidence mean?
- Are these APIs consistent in how they respond?
- Will he need a combination of multiple cvCIS APIs to solve this task?
- How does he know when there is a defect in the response? How can he report it?
- How does he know what labels the API can pick up, and what labels it can’t?
- How does it describe his photos and detect the faces?

- How can he interpret the results if he disagrees with it to help improve his app?
- Does he understand that the API uses a machine learnt model? Does he know what a ML model even is?
- If so, does he know when the models update? What is the release cycle?

Dazzled by this, he does some brief reading on Wikipedia but is confused by the immense technical detail to take in. He would like some form of guiding communication framework to assist him and in software engineering terms aligned to his background.

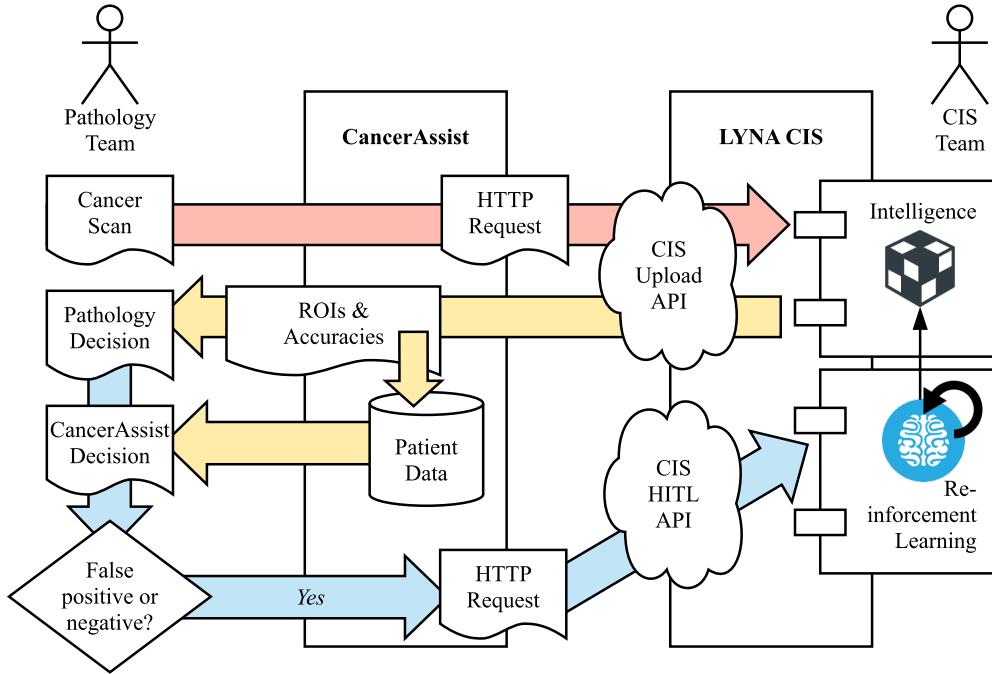
Although Tom generally anticipates some imperfections, he has no prior benchmark to guide him on what to expect. He understands that the app is not always going to be perfect: perhaps some photos of his dog may be missed because the dog is in the background and not the foreground, or his friends can't find the photos of their recent trip to the beach because it wasn't sunny enough for the beach to be recognised. These imperfections appear to be low-risk, but may become socially awkward when in use; for instance, if some of Tom's friends have low self-esteem and use the app, they may be sensitive to being misidentified or even mislabelled. Privacy issues also come into play especially if certain friends have only access to certain photos that they are (supposedly) in; e.g., photos from a holiday with Tom and his partner, however if the API identifies Tom's partner as a work colleague, then Tom's partner's privacy is at risk.

## Motivating Scenario II: Cancer Detection CDSS

Recent works in the oncology domain have used deep-learning Convolutional Neural Networks (CNNs) to detect Region of Interests (ROIs) in image scans of tissue (e.g., [148, 100, 81]), flagging these regions for doctors to review. Trials of such algorithms have been able to accurately detect cancer at higher rates than humans, and thus incorporating such capabilities into a CDSS is closer within reach. Some studies have suggested that practitioner over-reliance may erode independent decision-making [118, 59]; therefore the risks in developing CDSSs powered by intelligent services become paramount.

In Figure 1.4 we present a context diagram for a fictional CDSS named *CancerAssist*. *CancerAssist* is used by a team of busy pathologists who review patient lymph node scans and discuss and recommend, on consensus, if the patient should or should not be sent to surgery. When consensus is made, the lead pathologist enters the verdict into *CancerAssist*—running passively in the background—to ensure no oversight has been made in the team's discussions. When a conflict exists between the team's verdict and *CancerAssist*'s verdict, the system produces the scan with ROIs it thinks the team should review. Where the team

override the output of CancerAssist, this helps to reinforce CancerAssist's internal model as a Human-in-the-loop (HITL) learning process.



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

Powering CancerAssist is Google AI's Lymph Node Assistant (LYNA) [148], a CNN based on the Inception-v3 model [227, 135]. To provide intelligence to CancerAssist, LYNA is hosted on a CIS, and thus the developers of CancerAssist call the relevant CIS API endpoints, in conjunction with extra information such as patient data and medical history, to produce the verdict. In the case of a positive verdict, the relevant ROIs CancerAssist has found are highlighted with their respective bounding boxes and their respective cancer detection accuracies.

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

Certain questions in the production of CancerAssist and its use of a CIS may come into

mind:

- When is the model updated and how do the CIS team communicate that the model is updated?
- What benchmark test set of data do I use to ensure that the changed model doesn't affect other results?
- How do we know that the assumptions made by the CIS team who train model are correct?

Thus, improved documentation and additional metadata may be needed to better improve communication between developers and CIS providers. Such claims are further expanded upon in the following section.

## 1.2 Research Outcomes

*In this work, we present a framework to improve the documentation quality of Computer Vision Cloud Intelligence Services (cvCISs) and their APIs. We demonstrate that developers currently lack the understanding of how these services function and our framework mitigates this by presenting a solution to improve the documentation quality of the APIs and improve the existing techniques used to integrate these services into developer's end-applications.*

The goals of this study aim to provide a snapshot of current developer practices towards the usage of CISs. This allows us to develop a guiding framework and recommendations for application developers and CISs providers alike. Our anchoring perspective is software quality—specifically, validation and verification—with such systems and what best practices within the field of software engineering can be applied to assist in consumption of CISs. Based on the motivating case studies in Section 1.1, we articulate three Research Hypotheses (RH1–3) below and seven Research Questions (RQs) based on both empirical and non-empirical software engineering methodology [212, 213].

**RH1:** *Existing CISs present insufficient API documentation for general use.*

**Research Hypothesis** API documentation of intelligent services are inadequate and insufficient given the disparity of mindsets between the application developers and CIS providers. Chiefly, application developers have limited general understanding of the ‘magic’ that occurs behind these probabilistic ‘intelligent’ APIs. We do not know what

**RH1:** *Existing CISs present insufficient API documentation for general use. (cont)*

key aspects of the documentation matter to them, nor what they do or do not understand of the existing documentation.

**Research Goal** To improve the effectiveness of the documentation in existing CIS providers, specifically of cvCIS APIs.

**Research Questions**

**RQ1.1.** What practices are in use for intelligent services' API documentation?

**RQ1.2.** How do developers currently understand and interpret the documentation given a lack of formal training in artificial intelligence? That is, what do they understand and not understand, and what key aspects of the API documentation matter do developers as they see it?

**RQ1.3.** What additional information or attributes would developers prefer to be included in the API documentation?

**Research Contribution** An intelligent service API documentation quality assessment framework to evaluate how well the service has been documented for software engineers to use.

**RH2:** *Existing CISs present insufficient metadata for context-specificity.*

**Research Hypothesis** Intelligent service APIs respond with insufficient information for developers to operationalise the service into a business-driven application and, thus, additional metadata is needed to assist developers. Such metadata is likely to be needed as part of the response objects of the API.

**Research Goal** To improve the quality of *context-specific response data* from the API endpoints of intelligent services.

**Research Questions**

**RQ2.1.** What are current problems due to lack of return metadata?

**RQ2.2.** What additional metadata do developers desire to achieve implementing context-specific applications?

**RH2: Existing CISs present insufficient metadata for context-specificity. (cont)**

**Research Contributions** A list of metadata key-value-pairs that assist developers in using these APIs during the development of software that consume these services. In essence, improvements to the framework of Research Outcome 1: “*An intelligent service API documentation and metadata quality assessment framework*”.

**RH3: RH1 and RH2 improve quality, productivity or developer informativeness.**

**Research Hypothesis** The implication of hypotheses 1 and 2 suggest that improving both the documentation and providing further metadata will improve product quality (internal or external), and/or developer productivity and/or developer education in developing software with intelligent components.

**Research Goal** To confirm if improvements to API documentation and response metadata are reflected as improvements to product quality, developer productivity and/or developer education.

### Research Questions

**RQ3.1.** Does an improvement of documentation or metadata correlate to an improvement in software quality, developer productivity and/or developer informativeness?

**RQ3.2.** With respect to RQ3.1, the three aspects are explored:

- (a) Does the improvement cause increased product quality, as measured through improved external quality metrics?
- (b) Does the improvement cause increased developer productivity, as measured through improved internal quality metrics?
- (c) Does the improvement cause increased developer informativeness or increased confidence in developing CIS-powered applications?

**Research Contribution** A concrete sample solution or framework that improves such metrics, thereby confirming that our documentation and metadata quality assessment framework improves these facets.

### 1.3 Concluding Remarks

Ultimately, we seek to understand the conceptual understanding of software engineers who operationalise stochastic and probabilistic systems, and furthermore understand knowledge representation with these systems' API documentation. Our motivation is to provide insight into current practices and compare the best practices with actual practise. We strive for this to provide developers with a guiding framework on how to best operationalise these systems via the form of some checklist or tool they can use to ensure optimal software quality.

It is anticipated that the findings from this study in the cvCISs space will be generalisable to other areas, such as time-series information, natural language processing and others.



# Chapter 2

## Background

In Chapter 1, we defined a common set of (artificial) intelligence-based cloud services that we label Cloud Intelligence Services (CISs). Specifically, we scope the primary body of this study’s work on Computer Vision Cloud Intelligence Services (cvCISs) (e.g., Google Cloud Vision [15], AWS Rekognition [3], Azure Computer Vision [8], Watson Visual Recognition [16] etc.). We claim developers have a distinctly deterministic mindset (*2 + 2 always equals 4*) whereas a CIS’s ‘intelligence’ component (a black box) may return probabilistic results (*2 + 2 might equal 4 with a confidence of 95%*). Thus, there is a mindset mismatch between probabilistic results (from the API provider) and results interpreted with certainty (from the API consumer).

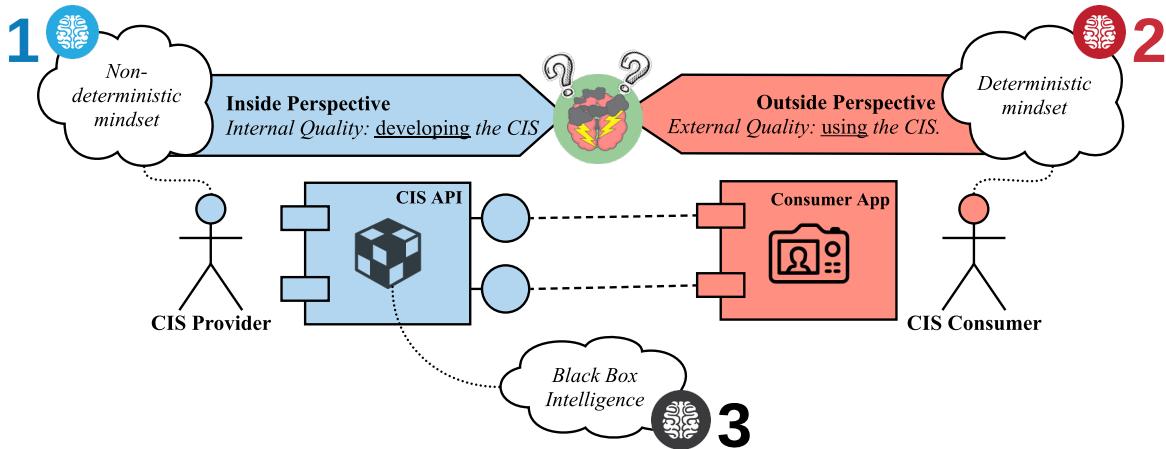
What affect does this mindset mismatch have on the developer’s approach towards building probabilistic software? What can we learn from common software engineering practices (e.g., [184, 218]) that apply to resolve this mismatch and thereby improve quality, such as Verification & Validation (V&V)? Chiefly, we anchor this question around three lenses of software engineering: creating a CIS, using a CIS, and the nature of CISs themselves.

Our chief concern lies with interaction and integration between CIS providers and consumers, the nature of applications built using a CIS, and the impact this has on software quality. We triangulate this around three pillars, which we diagrammatically represent in Figure 2.1.

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

software in the ways they expect the CIS to work.

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



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

Does a clash of deterministic consumer mindsets who use a CIS and the non-deterministic provider mindsets who develop them exist? And what impact does this have on the inside and outside perspective? Throughout this chapter, we will review these core principles due to such mindset mismatch from the anchoring perspective of software quality, particularly around V&V.

## 2.1 Software Quality

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

ROBERT PIRSIG, 1974 [183]

The philosophical viewpoint of ‘quality’ remains highly debated and there are multiple facets to perceive this complex concept [95]. Transcendentally, a viewpoint like that of Pirsig’s above shows that quality is not tangible but still recognisable; it’s hard to explicitly define but you know when it’s missing. The International Organization for Standardization provides a breakdown of seven universally-applicable principles that defines quality for organisations, developers, customers and training providers [114]. More pertinently, the 1986 ISO standard

for quality was simply “the totality of characteristics of an entity that bear on its ability to satisfy stated or implied needs” [113].

Using this sentence, what characteristics exist for non-deterministic CISs like that of a cvCIS? How do we know when the system has satisfied its ‘stated or implied needs’ when the system can only give us uncertain probabilities in its outputs? Such answers can be derived from related definitions—such as ‘conformance to specification or requirements’ [96, 66], ‘meeting or exceeding customer expectation’ [174], or ‘fitness for use’ [124]—but these then still depend on the solution description or requirements specification, and thus the same questions still apply.

*Software* quality is somewhat more concrete. Pressman [184] adapted the manufacturing-oriented view of quality from [42] and phrased software quality under three core pillars:

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

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

### 2.1.1 Validation and Verification

To explain V&V, we analogously recount a tale given by Pham [181] on his works on reliability. A high-school student sat a standardised test that was sent to 350,0000 students [228]. A multiple-choice algebraic equation problem used a variable,  $a$ , and intended that students *assume* that the variable was non-negative. Without making this assumption explicit, there

were two correct answers to the multiple choice answer. Up to 45,000 students had their scores retrospectively boosted by up to 30 points for those who ‘incorrectly’ answered, however, outcomes of a student’s higher education were, thereby, affected by this one oversight in quality assessment. The examiners wrote a poor question due to poor process standards to check if their ‘correct’ answers were actually correct. The examiners “didn’t build the right product” nor did they “build the product right” by writing an poor question and failing to ensure quality standards, in the phrases Boehm [45] coined.

This story describes the issues with the cost of quality [44] and the importance of V&V: just as the poorly written exam question had such a high toll the 45,000 unlucky students, so does poorly written software in production. As summarised by Pressman [184], data sourced from Digital Inc. [62] in a large-scale application showed that the difference in cost to fix a bug in development versus system testing is \$6,159 per error. In safety-critical systems, such as self-driving cars or clinical decision support systems, this cost skyrockets due to the extreme discipline needed to minimise error [230].

Formally, we refer to the IEEE Standard Glossary of Software Engineering Terminology [111] for to define V&V:

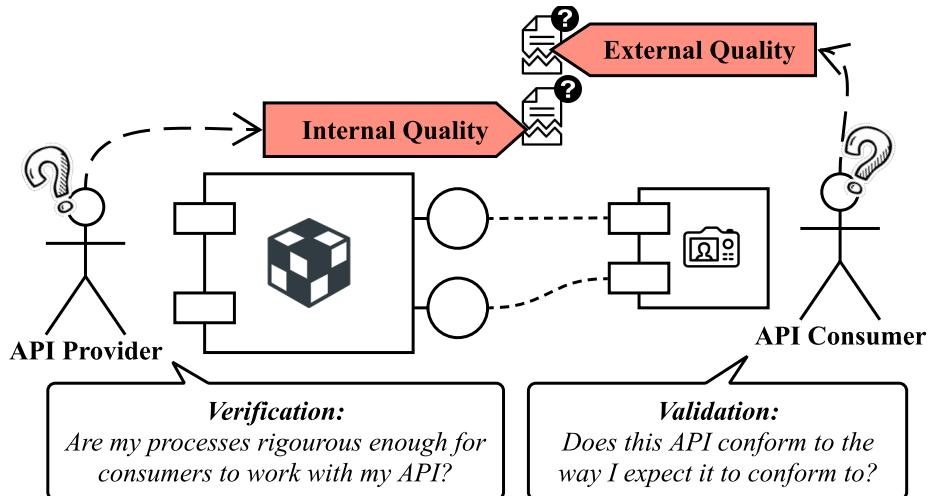
<b>verification</b>	The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.
<b>validation</b>	The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements.

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

The verification process of API providers ‘leak’ out to the context of the developer’s project dependent on the CIS. Poor verification in the *internal quality* of the CIS will entail poor process standards, such as poor definitions and terminology used, support tooling and description of documentations [218]. Though it is commonplace for providers to have a ‘ship-first-fix-later’ mentality of ‘good-enough’ software [234], the consequence of doing so leads to consumers absorbing the cost. Thus API providers must ensure that their verification strategies are rigorous enough for the consumers in the myriad contexts they wish to use it in. Studies have considered V&V in the context of web services on the cloud [163, 165, 88, 103, 55, 54, 247, 205], though little have recently considered how adding ‘intelligence’ to these services affects existing proposed frameworks and solutions. For a cvCIS, what might this entail? Which assurances are given to the consumers, and how is that information communicated? To

verify if the service is working correctly, does that mean that we need to deploy the system first to get a wider range of data, given the stochastic nature of the black box?

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



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

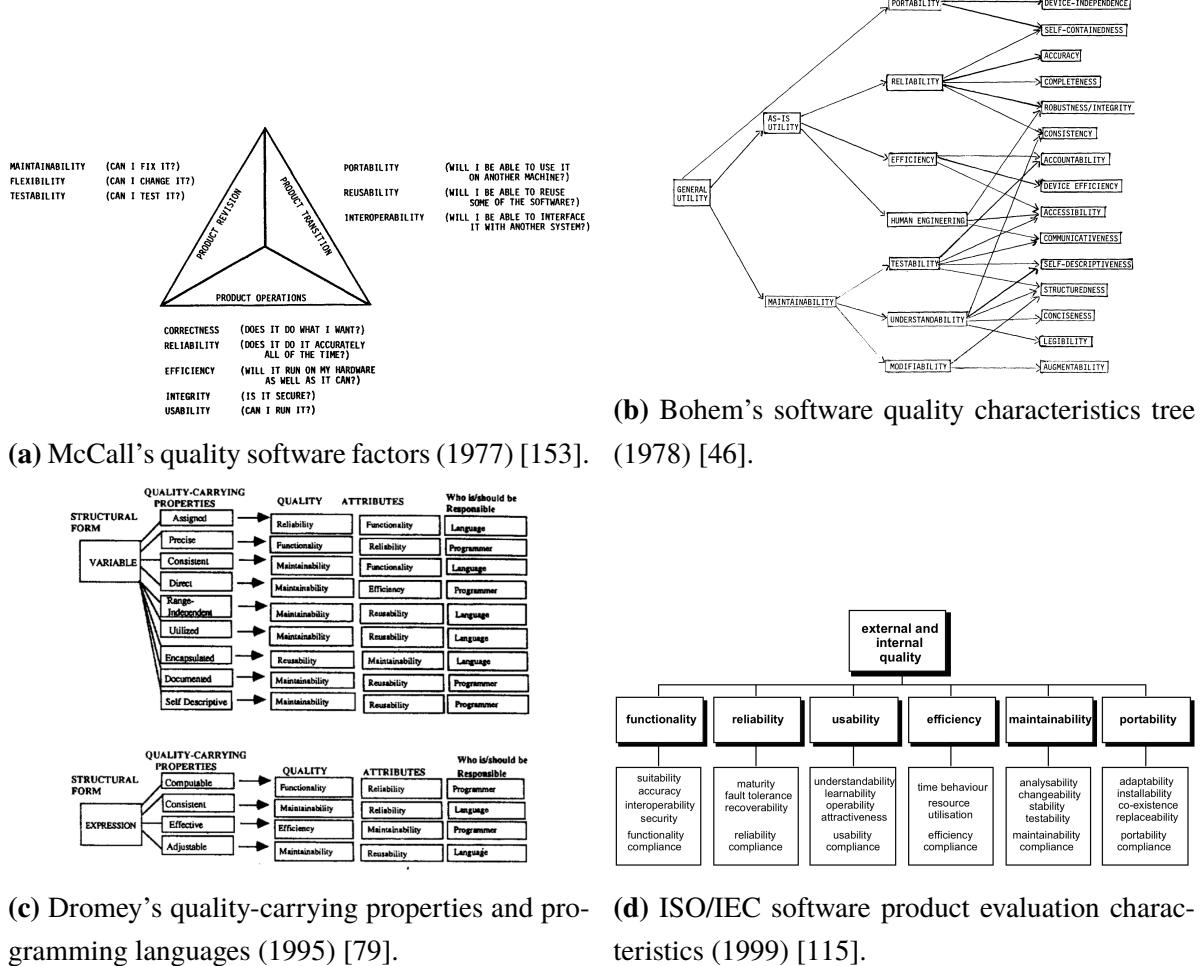
Therefore, just as the test answers with were both correct and incorrect at the same time, so is the same with CISs returning a probabilistic result: no result is certain. While V&V has been investigated in the area of mathematical and earth sciences for numerical probabilistic models and natural systems [170, 204], from the software engineering literature, little work has been achieved to look at the surrounding area of probabilistic systems hidden behind API calls.

Now that a developer is using a probabilistic system behind a deterministic API call, what does it mean in the context of V&V? Do current verification approaches and tools suffice, and if not, how do we fix it? From a validation perspective of ML and end-users, after a model is trained and an inference is given and if the output data point is incorrect, how will end users report a defect in the system? Compared to deterministic systems where such tooling as defect reporting forms are filled out (i.e., given input data in a given situation and the output data was X), how can we achieve similar outputs when the system is not non-deterministic? A key

problem with the probabilistic mindset is that once a model is ‘fixed’ by retraining it, while one data-point may be fixed, others may now have been effected, thereby not ensuring 100% validation. Thus, due to the unpredictable and blurry nature of probabilistic systems, V&V must be re-thought out extensively.

### 2.1.2 Quality Attributes and Models

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



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

Quality attribute assessment models (like those shown in Figure 2.3) are an early concept in software engineering, and systematically evaluating software quality appears as early as 1968 [203]. Rubey and Hartwick’s 1968 study introduced the phrase ‘attributes’ as a “prose expression of the particular quality of desired software” (as worded by Boehm et al. [46]) and

‘metrics’ as mathematical parameters on a scale of 0 to 100. Early attempts to categorise wider factors under a framework was proposed by McCall, Richards, and Walters in the late 1970s [153, 58]. This model described quality from the three perspectives of product revision (*how can we keep the system operational?*), transition (*how can we migrate the system as needed?*) and operation (*how effective is the system at achieving its tasks?*) (Figure 2.3a). The model also introduced 11 attributes alongside numerous direct and indirect measures to help quantify quality. This model was further developed by Boehm et al. [46] who independently developed a similar model, starting with an initial set of 11 software characteristics. It further defined candidate measurements of Fortran code to such characteristics, taking shape in a tree-like structure as in Figure 2.3b. In the mid-1990s, Dromey’s interpretation [79] defined a set of quality-carrying properties with structural forms associated to specific programming languages and conventions (Figure 2.3c). The model also supported quality defect identification and proposed an improved auditing method to automate defect detection for code editors in IDEs. As the need for quality models became prevalent, the International Organization for Standardization standardised software quality under ISO/IEC-9126 [115] (the Software Product Evaluation Characteristics, Figure 2.3d), which has since recently been revised to ISO/IEC-25010 with the introduction of the Systems and software Quality Requirements and Evaluation (SQuaRE) model [112], separating quality into *Product Quality* (consisting of eight quality characteristics and 31 sub-characteristics) and *Quality In Use* (consisting of five quality characteristics and 9 sub-characteristics). An extensive review on the development of quality models in software engineering is given in [26].

Of all the models described, there is one quality attribute that relates most with our narrative of CIS quality: reliability. The definition of reliability is similar among all quality models:

- McCall et al.** Extent to which a program can be expected to perform its intended function with required precision [154].
- Boehm et al.** Code possesses the characteristic *reliability* to the extent that it can be expected to perform its intended functions satisfactorily [46].
- Dromey** Functionality implies reliability. The reliability of software is therefore dependent on the same properties as functionality, that is, the correctness properties of a program [79].
- ISO/IEC-9126** The capability of the software product to maintain a specified level of performance when used under specified conditions [115].

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

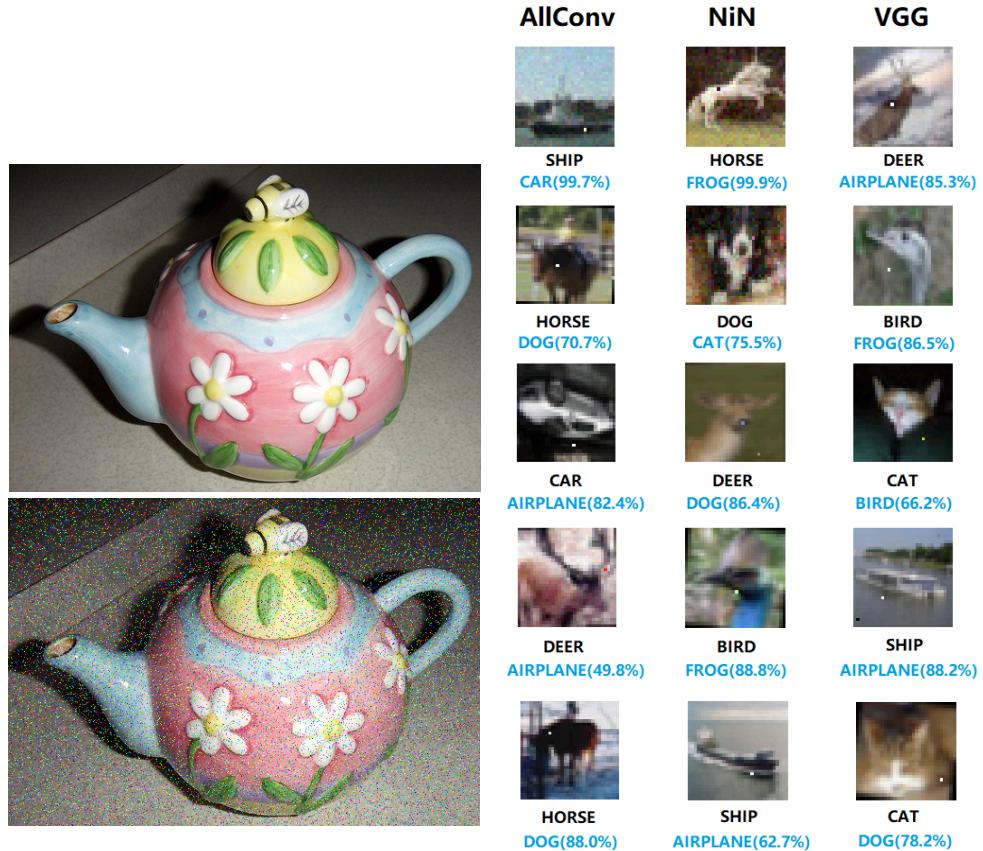
Moreover, defining these systems in themselves is challenging when requirements specifications and solution descriptions are dependent on nondeterministic and probabilistic algorithms. We discuss this further in Section 2.2.

### 2.1.3 Reliability in Computer Vision

Testing computer vision deep-learning reliability is an area explored typically through the use of adversarial examples [226]. These input examples are where images are slightly perturbed to maximise prediction error but are still interpretable to humans. Refer to Figure 2.4.

Google Cloud Vision, for instance, fails to correctly classify adversarial examples when noise is added to the original images [106]. Rosenfeld et al. [201] illustrated that inserting synthetic foreign objects to input images (e.g., a cartoon elephant) can alter classification output. Wang et al. [237] performed similar attacks on a transfer-learning approach of facial recognition by modifying pixels of a celebrity’s face to be recognised as a different celebrity, all while still retaining the same human-interpretable original celebrity. Su et al. [220] used the ImageNet database to show that 41.22% of images drop in confidence when just a *single pixel* is changed in the input image; and similarly, Eykholt et al. [84] recently showed similar results that made a CNN interpret a stop road-sign (with mimiccid graffiti) as a 45mph speed limit sign.

Thus, the state-of-the-art computer vision techniques may not be reliable enough for safety critical applications (such as self-driving cars) as they do not handle intentional or unintentional adversarial attacks. Moreover, as such adversarial examples exist in the physical world [137, 83], “the real world may be adversarial enough” [161] to fool such software.

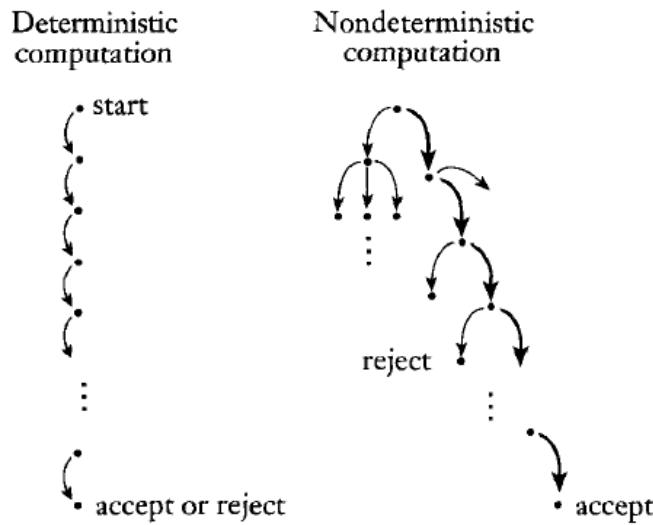


**(c)** Adversarial examples to trick face recognition from the source to target images [237].

**Figure 2.4:** Sample adversarial examples.

## 2.2 Probabilistic and Nondeterministic Systems

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



**Figure 2.5:** A deterministic system (left) always returns the same result in the same amount of steps. A nondeterministic system does not guarantee the same outcome, even with the same input data. Source: [87].

### 2.2.1 Interpreting the Uninterpretable

As the rise of applied AI increases, the need for engineering interpretability around models becomes paramount, chiefly from an external quality perspective that the *reliability* of the system can be inspected by end-users. Model interpretability has been stressed since early machine learning research in the late 1980s and 1990s (such as Quinlan [186] and Michie [158]), and although there has since been a significant body of work in the area [215, 32, 189, 52, 202, 146, 47, 122, 31, 94, 69, 233, 40, 85, 144, 152, 177, 235], it is evident that ‘accuracy’ or model ‘confidence’ is still used as a primary criterion for AI evaluation [108, 117, 217]. Much research into Neural Network (NN) or Support Vector Machine (SVM) development stresses that ‘good’ models are those with high accuracy. However, is accuracy enough to justify a model’s quality?

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

quantified in the AI testing stage, whereby the algorithm is tested against cases known by humans to have ground truth but such cases are unknown by the algorithm. In production, however, all cases are unknown by both the algorithm *and* the humans behind it, and therefore a single value of quality is “not reliable if the future dataset has a probability distribution significantly different from past data” [90], a problem commonly referred to as the *datashift* problem [223]. Analogously, Freitas [90] provides the following description of the problem:

*The military trained [a NN] to classify images of tanks into enemy and friendly tanks. However, when the [NN] was deployed in the field (corresponding to “future data”), it had a poor accuracy rate. Later, users noted that all photos of friendly (enemy) tanks were taken on a sunny (overcast) day. I.e., the [NN] learned to discriminate between the colors of the sky in sunny vs. overcast days! If the [NN] had output a comprehensible model (explaining that it was discriminating between colors at the top of the images), such a trivial mistake would immediately be noted.* [90]

So, why must we interpret models? While the formal definition of what it means to be *interpretable* is still somewhat disparate (though some suggestions have been proposed [146]), what is known is (i) there exists a critical trade-off between accuracy and interpretability [89, 121, 127, 98, 74, 249], and (ii) a single quantifiable value cannot satisfy the subjective needs of end-users [90]. As ever-growing domains ML become widespread<sup>1</sup>, these applications engage end-users for real-world goals, unlike the aims in early ML research where the aim was to get AI working in the first place. In safety-critical systems where AI provide informativeness to humans to make the final call (see [56, 128, 109]), there is often a mismatch between the formal objectives of the model (e.g., to minimise error) and complex real-world goals, where other considerations (such as the human factors and cognitive science behind explanations<sup>2</sup>) are not realised: model optimisation is only worthwhile if they “actually solve the original [human-centred] task of providing explanation” [164] to end-users. **Therefore, when human-decision makers must be interpretable themselves [192], any AI they depend on must also be interpretable.**

Recently, discussion behind such a notion to provide legal implications of interpretability is topical. Doshi-Velez et al. [77] discuss when explanations are not provided from a legal stance—for instance, those affected by algorithmic-based decisions have a ‘right to explanation’

---

<sup>1</sup>In areas such as medicine [39, 141, 178, 191, 251, 233, 122, 82, 246, 118, 52], bioinformatics [91, 225, 126, 73, 119], finance [32, 109, 71] and customer analytics [235, 144].

<sup>2</sup>*Interpretations and explanations* are often used interchangeably.

[97, 236] under the European Union’s GDPR<sup>3</sup>. But, explanations are not the only way to ensure AI accountability: theoretical guarantees (mathematical proofs) or statistical evidence can also serve as guarantees [77], however, in terms of explanations, what form they take and how they are proven correct are still open questions [146].

### 2.2.2 Explanation and Communication

From a software engineering perspective, explanations and interpretability are, by definition, inherently communication issues: what lacks here is a consistent interface between the AI system and the person using it. The ability to encode ‘common sense reasoning’ [155] into programs today has been achieved, but *decoding* that information is what still remains problematic. At a high level, Shannon and Weaver’s theory of communication [210] applies, just as others have done with similar issues in the Software Engineer (SE) realm [160, 241] (albeit to the domain of visual notations). Humans map the world in higher-level concepts easily when compared to AI systems: while we think of a tree first (not the photons of light or atoms that make up the tree), an algorithm simply sees pixels, and not the concrete object [77] and the AI interprets the tree inversely to humans. Therefore, the interpretation or explanation is done inversely: humans do not explain the individual neurons fired to explain their predictions, and therefore the algorithmic transparent explanations of AI algorithms (“*which neurons were fired to make this AI think this tree is a tree?*”) do not work here.

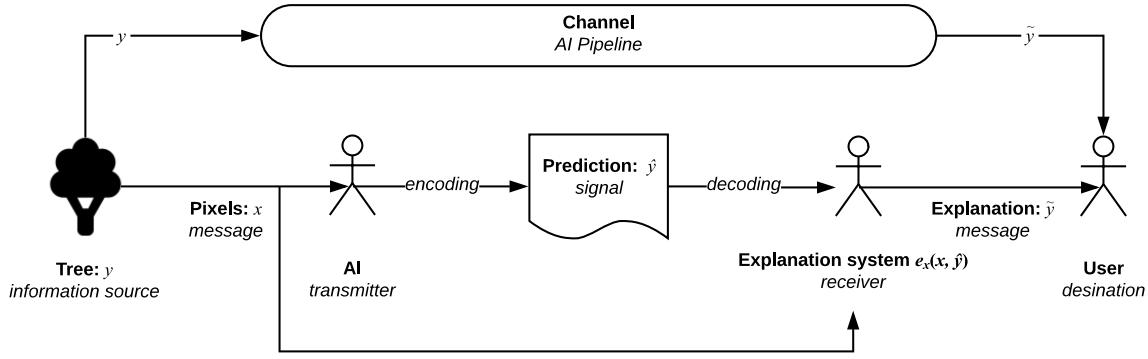
Therefore, to the user (as mapped using Shannon and Weaver’s theory), an AI pipeline (the communication *channel*) begins with a real-world concept,  $y$ , that acts as an *information source*. This information source is fed in as a *message*,  $x$ , (as pixels) to an AI system (the *transmitter*). The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

### 2.2.3 Mechanics of Model Interpretation

How do we interpret models? Methods for developing interpretation models include: decision trees [50, 101, 64, 185, 198], decision tables [144, 33] and decision sets [139, 164]; input gradients, gradient vectors or sensitivity analysis [208, 189, 142, 202, 32]; exemplars [129, 92];

---

<sup>3</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  $\tilde{y}$ .

generalised additive models [56]; classification (*if-then*) rules [232, 48, 63, 172, 243] and falling rule lists [215]; nearest neighbours [152, 209, 224, 240, 250] and Naïve Bayes analysis [39, 141, 134, 251, 157, 93, 60, 104].

Cross-domain studies have assessed the interpretability of these techniques against end-users, measuring response time, accuracy in model response and user confidence [109, 102, 27, 221, 206, 91, 152, 235], although it is generally agreed that decision rules and decision tables provide the most interpretation in non-linear models such as SVMs or NNs [91, 152, 235]. For an extensive survey of the benefits and fallbacks of these techniques, we refer to Freitas [90], Doshi-Velez et al. [77] and Doshi-Velez and Kim [76].

## 2.3 Application Programming Interfaces

Application Programming Interfaces (APIs) are the interface between a developer needs and the software components at their disposal [29] by abstracting the underlying component behind a subroutine, protocol or specific tool. Therefore, it is natural to assess internal quality (and external quality if the software is in itself a service to be used by other developers—in this case a CIS) is therefore directly related to the quality the API offers [133].

Good APIs are known to be intuitive and require less documentation browsing [182], thereby increasing developer productivity. Conversely, poor APIs are those that are hard to interpret, thereby reducing developer productivity and product quality. The consequences of this have shown a higher demand of technical support (as measured in [105]) that, ultimately, causes the maintenance to be far more expensive, a phenomenon widely known in software engineering economics (see Section 2.1.1).

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

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

### 2.3.1 API Usability

If a developer doesn't understand the overarching concepts of the context behind the API they wish to use, then they cannot formulate what gaps in their knowledge is missing. For example, a developer that knows nothing about machine learning techniques in computer vision cannot effectively formulate queries to help bridge those gaps in their understanding to figure out more about the cvCIS they wish to use.

Balancing the understanding of the information need (both conscious and unconscious), how to phrase that need and how to query it in an information retrieval system is concept long studied in the information sciences [231]. In API design, the most common form to convey knowledge to developers is through annotated code examples and overviews to a platform's architectural and design decisions [162, 196, 75, 49] though these studies have not effectively communicated *why* these artefacts are important. What makes the developer *conceptually understand* these artefacts?

Robillard and Deline [196] conducted a multi-phase, mixed-method approach to create knowledge grounded in the professional experience of 440 software engineers at Microsoft of varying experience to determine what makes APIs hard to learn, the results of which previously published in an earlier report [195]. Their results demonstrate that 'documentation-related obstacles' are the biggest hurdle in learning new APIs. One of these implications are the *intent documentation* of an API (i.e., *what is the intent for using a particular API?*) and such documentation is required only where correct API usage is not self-evident, where advanced uses of the API are documented (but not the intent), and where performance aspects of the API impact the application developed using it. They conclude that professional developers do not struggle with learning the *mechanics* of the API, but in the *understanding* of how the API fits in upwards to its problem domain and downward to its implementation:

*In the upwards direction, the study found that developers need help mapping desired scenarios in the problem domain to the content of the API, and in understanding what scenarios or usage patterns the API provider intends and does not intend to support. In the downwards direction, developers want to understand*

*how the API’s implementation consumes resources, reports errors and has side effects.* [196]

These results particularly corroborate to that of previous studies where developers quote that they feel that existing learning content currently focuses on “*how* to do things, not necessarily *why*” [168]. This thereby reiterates the conceptual understanding of an API as paramount.

A later study by Ko and Riche [132] assessed the importance of a programmer’s conceptual understanding of the background behind the task before implementing the task itself, a notion that we find most relevant for users of CIS APIs. While the study did not focus on developing web APIs (rather implementing a Bluetooth application using platform-agnostic terminology), the study demonstrated how developers show little confidence in their own metacognitive judgements to understand and assess the feasibility of the intent of the API and understand the vocabulary and concepts within the domain (i.e., wireless connectivity). This indecision over what search results were relevant in their searches ultimately hindered their progress implementing the functionality, again decreasing productivity. Ko and Riche suggest to improve API usability by introducing the background of the API and its relevant concepts using glossaries linked to tutorials to each of the major concepts, and then relate it back to how to implement the particular functionality.

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



# Chapter 3

## Research Methodology

Investigating software engineering practices is often a complex task as it is imperative to understand the social and cognitive processes around software engineers and not just the tools and processes used [80]. This chapter explores our research methodology by exploring five key elements of empirical software engineering research: firstly, (i) we provide an extended focus to the study by reviewing our research questions (see Section 1.2) anchored under the context of an existing classification taxonomy, (ii) characterise our research goals through an explicit philosophical stance, (iii) explain how the stance selected impacts our selection of research methods and data collection techniques (by dissecting our choice of methods used to reach these research goals), (iv) discuss a set of criteria for assessing the validity of our study design and the findings of our research, and lastly (v) discuss the practical considerations of our chosen methods.

The foundations for developing this research methodology has been expanded from that proposed by Easterbrook et al. [80], Wohlin and Aurum [244], Wohlin et al. [245] and Shaw [211].

### 3.1 Research Questions Revisited

In Section 1.2, we introduce three hypothesis of this study (RH1–RH3), namely: (i) existing CIS APIs are poorly documented for general use (RH1); (ii) existing CIS APIs do not provide sufficient metadata when used in context-specific use cases (RH2); and (iii) the combination of improving documentation and metadata will ultimately improve one of software quality, developer productivity and/or developer understanding (RH3).

To discuss our research strategy, we revisit our research questions through the classification technique discussed by Easterbrook et al. [80], a technique originally proposed in the field of psychology by Meltzoff [156] but adapted to software engineering. Our research study

involves a mix of five *knowledge questions*, that focus on existing practices and the ways in which they work, and two *design questions*, that focuses on designing better ways to approach software engineering tasks [213]. Both classes of questions are respectively concerned with empirical and non-empirical software engineering that, in practice, are best combined in long-term software engineering research studies (such as this one) as they assist in tackling the investigation of a specific problem, approaches to solve that problem and finding what solutions work best [242].

### 3.1.1 Knowledge Questions

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

We begin by formulating two *exploratory questions* to attempt to better understand the phenomena of poor API documentation and metadata; both RQ1.1 and RQ2.1 respectively describe and classify what practices are in use for existing CIS API documentation and what problems currently exist when no metadata is returned. Answering these two questions assists in refining preciser terms of the phenomena, ways in which we find evidence for them and ensuring the data found is valid.

By answering these questions, we have a clearer understanding of the phenomena; we then follow up by posing an additional *base-rate question* that helps provide a basis to confirm that the phenomena occurring is normal (or unusual) behaviour by investigating the patterns of phenomena's occurrence. RQ1.2 is a descriptive-process question to help us understand how the developer currently understands existing CIS API documentation, given their lack of formal extended training in artificial intelligence. This achieves us an insight into the developer's mindset and regular thought patterns toward these APIs.

Lastly, we investigate the relationship between the improved documentation and improvements to other aspects of the software development process. Chiefly, RQ3.1 is concerned with whether any improvements to metadata or documentation correlate to improvements in software quality, developer productivity, or developer education (and is a *relationship establishment question*). If we establish such a relationship, we refine the question and investigate the specific causes using three *causality questions* defined under RQ3.2, namely by associating three classes of measurable metrics (internal quality metrics, external quality metrics, developer education insight metrics) to the improved documentation.

### 3.1.2 Design Questions

RQ1.3 and RQ2.2 are both *design questions*; they are concerned with ways in which we can improve a CIS by investigating what additional attributes are needed in both the documentation and metadata that assist developers to achieve their goals. They are not classified as knowledge questions as we investigate what *will be* and not *what is*. By understanding the process by which developers desire additional attributes of metadata and documentation, we can help shape improvements to the existing design of a CIS.

## 3.2 Philosophical Stances

Philosophical stances guide the researcher's action by fortifying what constitutes 'valid truth' against a fundamental set of core beliefs [193]. In software engineering, four dominant philosophical stances are commonly characterised [65, 180]: positivism (or post-positivism), constructivism (or interpretivism), pragmatism, and critical theory (or advocacy/participatory). To construct such a 'validity of truth', we will review these four philosophical stances in this section, and state the stance that we explicitly adopt and our reasoning for this.

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

**Constructivism** Constructivists see knowledge embedded within the human context; truth is the *interpretive* observation by understanding the differences in human thought between meaning and action [131]. That is, the interpretation of the theory is just as important to the empirical observation itself. We partially adopt a constructivist stance as we attempt to model the developer's mindset, being an approach that is rich in qualitative data on human activity.

**Pragmatism** Pragmatism is a less dogmatic approach that encourages the incomplete and approximate nature of knowledge and is dependent on the methods in which the knowledge was extracted. The utility of consensually agreed knowledge is the key outcome, and is therefore relative to those who seek utility in the knowledge—what is the useful for one person is not

so for the other. While we value the utility of knowledge, it is difficult to obtain consensus especially on an ill-researched topic such as ours, and therefore we do not adopt this stance.

**Critical Theory** This study chiefly adopts the philosophy of critical theory [53]. A key outcome of the study is to shift the developer's restrictive deterministic mindset and shed light on developing a new framework actively with the developer community that seeks to improve the process of using such APIs. In software engineering, critical theory is used to “actively [seek] to challenge existing perceptions about software practice” [80], and this study utilises such an approach to shift the mindset of CIS consumers and providers alike on how the documentation and metadata should not be written with the ‘traditional’ deterministic mindset at heart. Thus, our key philosophical approach is critical theory to seek out *what-can-be* using partial constructivism to model the current *what-is*.

### 3.3 Research Design

Research methods are “a set of organising principles around which empirical data is collection and analysed” [80]. Creswell and Creswell [65] suggest that strong research design is reflected when the weaknesses of multiple methods complement each other. Using a mixed-methods approach is therefore commonplace in software engineering research, typically due to the human-oriented nature investigating how software engineers work both individually (where methods from psychology may be employed) and together (where methods from sociology may be employed).

Therefore, studies in software engineering are typically performed as field studies where researchers and developers (or the artefacts they produce) are analysed either directly or indirectly [214]. The mixed-methods approach combines five classes of field study methods (or empirical strategies/studies) most relevant in empirical software engineering research [80, 245, 125]: controlled experiments, case studies, survey research, ethnographies, and action research. We chiefly adopt a mixed-methods approach to our work using the *concurrent triangulation* mixed-methods strategy [120] as it best compensates for weaknesses that exist in all research methods, and employs the best strengths of others.

#### 3.3.1 Review of Relevant Research Methods

Below we review some of the research methods most relevant to our research questions as refined in Section 3.1 as presented by Easterbrook et al. [80].

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

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

**Case Studies** Case studies investigate phenomena in their real-life context and are well-suited when the boundary between context and phenomena is unknown [248]. They offer understanding of how and why certain phenomena occur, thereby investigating ways cause-effect relationships can occur. They can be used to test existing theories (*confirmatory case studies*) by refuting theories in real-world contexts instead of under laboratory conditions or to generate new hypotheses and build theories during the initial investigation of some phenomena (*exploratory case studies*).

Case studies are well-suited where the context of a situation plays a role in the phenomenon being studied, which we specifically relate back to RH2 (RQ2.1 and RQ2.2) in exploring whether the context of an application using a CIS requires the CIS context-specific or of context-agnostic. They also lend themselves to purposive sampling rather than random sampling, and thus we can selectively choose cases that benefit the research goal of RH2 and (using our critical theorist stance) select cases that will actively benefit our participant software engineering audience most to draw attention to situations regarded as most problematic.

**Survey Research** Survey research identifies characteristics of a broad population of individuals through direct data collection techniques such as interviews and questionnaires or

independent techniques such as data logging. Defining that well-defined population is critical, and selecting a representative sample from it to generalise the data gathered usually assists in answering base-rate questions.

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

**Ethnographies** Ethnographies investigates the understanding of social interaction within community through field observation [197]. Resulting ethnographies help understand how software engineering technical communities build practices, communication strategies and perform technical work collaboratively.

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

**Action Research** Action researchers simultaneously solve real-world problems while studying the experience of solving the problem [68] by actively seeking to intervene in the situation for the purpose of improving it. A precondition is to engage with a *problem owner* who is willing to collaborate in identifying and solving the problem faced. The problem must be authentic (a problem worth solving) and must have new knowledge outcomes for those involved. It is also characterised as an iterative approach to problem solving, where the knowledge gained from solving the problem has a desirable solution that empowers the problem owner and researcher.

This research is most associated to our adopted philosophical stance of critical theory. As this project is being conducted under the Applied Artificial Intelligence Institute ( $A^2I^2$ )

collaboratively with engaged industry clients, we have identified a need for solving an authentic problem that industry faces. The desired outcome of this project is to facilitate wider change in the usage and development of cvCISs; thus, engaging action research as a primary method throughout the mixed-methods approach used in this research.

### 3.3.2 Review of Data Collection Techniques for Field Studies

Singer et al. developed a taxonomy [214, 143] showcasing data collection techniques in field studies that are used in conjunction with a variety of methods based on the level of interaction between researcher and software engineer, if any. This taxonomy is reproduced in Figure A.4.

## 3.4 Proposed Experiments

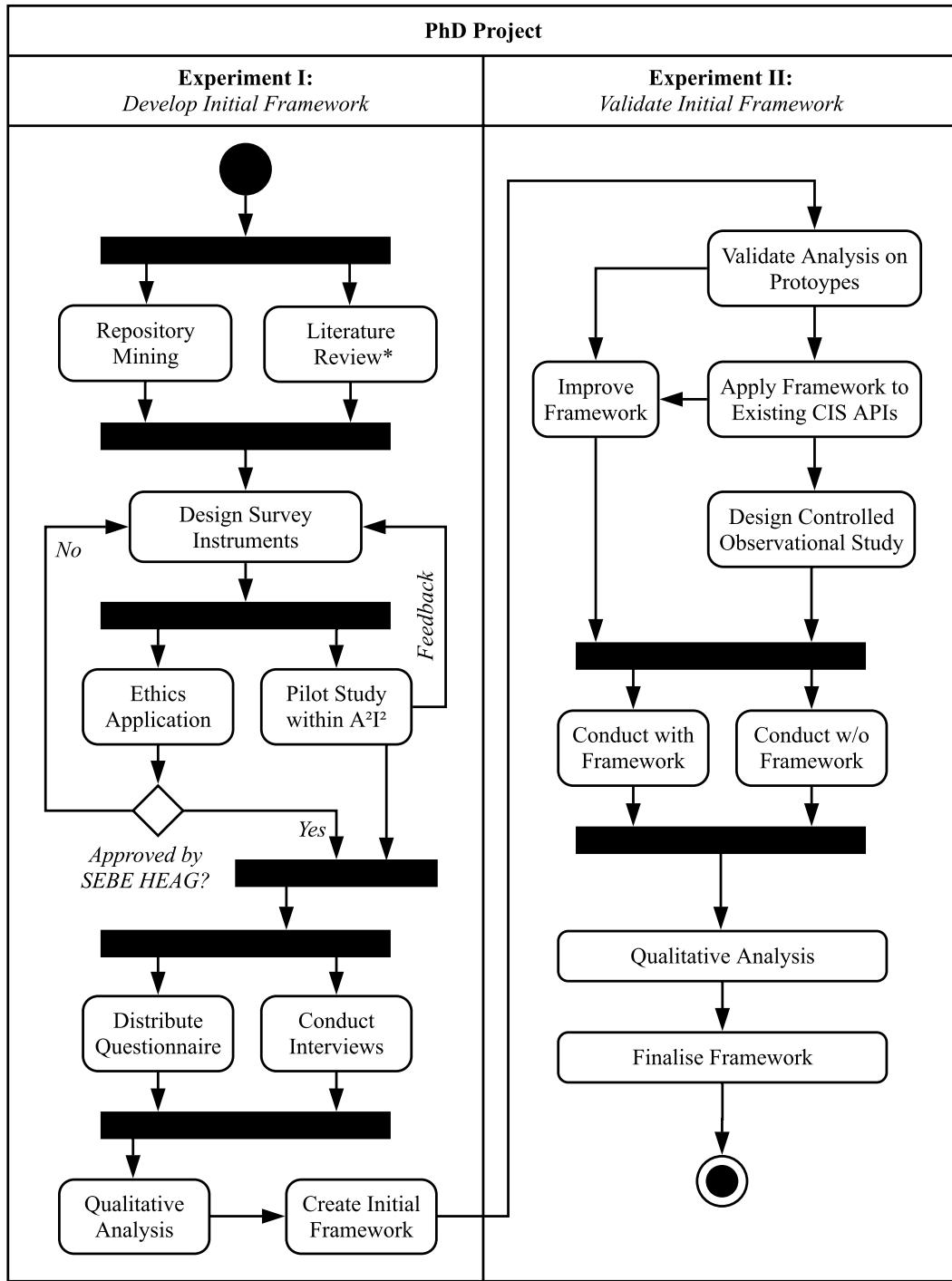
This section discusses two proposed experiments that we conduct in this study. For each experiment, we describe an overview of the experiment grounded known methods and techniques (Sections 3.3.1 and 3.3.2), our approach to analysing the data, as well as linking the experiment back to a research hypothesis and question (Section 1.2). A high-level overview of the proposed experiments and the major bodies of work they encompass is presented in Figure 3.1.

### 3.4.1 Experiment I: Develop Initial Framework

Experiment I shapes a context-agnostic approach to understand current usage patterns of CIS APIs and the ways by which developers interpret them. Briefly, this experiment is comprised under two phases of field survey research: (i) repository mining developer discussion forums (i.e., analysis of databases and documentation analysis) to understand what developers currently complain about on these forums and where their mismatch in understanding lies; (ii) conducting unstructured interviews and distributing a questionnaire to gather personal opinion based on individual developer's anecdotal remarks.

#### Relevance and Motivation

Experiment I aims to better understand the existing mindsets that developers have when approaching to use Computer Vision Cloud Intelligence Services (cvCISs). This work therefore ties in to RH1; by understanding the developer mindset in how they interpret cvCIS APIs, we are better informed to produce a framework that increases the effectiveness of the documentation of those existing cvCIS providers.



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

RH1 postulates that the software engineering community do not fully understand the ‘magic’ behind CIS APIs. As described in Section 1.2, they face a gap in their understanding around the underlying architecture of pre-built, machine learnt APIs (RQ1.2). Software developers are not well-supported by the CIS providers, and therefore do not have a consistent set of common best practices when approaching to use these APIs (RQ1.1). It is therefore neces-

sary that CIS providers provide additional information to gap this mismatched understanding (RQ1.3).

## Data Collection & Analysis

**Phase 1: Repository Mining** Developers typically congregate in search of discourses on issues they face in online forums, such as Stack Overflow and Quora, as well as writing their experiences in personal blogs such as Medium. The simplest of these platforms is Stack Overflow (a sub-community of the Stack Exchange family of targeted communities) that specifically targets developer issues on using a simple Q&A interface, where developers can discuss technical aspects and general software development topics. Moreover, Stack Overflow is often acknowledged as *the ‘go-to’ place* for developers to find high-quality code snippets that assist in their problems [222].

Thus, to begin validating CIS API usage and misunderstanding in a generalised context (i.e., context-agnostic to the project at hand), we propose using repository mining on Stack Overflow to help answer our research questions. Specifically, we select Stack Overflow due to its targeted community of developers<sup>1</sup> and the availability of its publicly available dataset released as ‘data dumps’ on the Stack Exchange Data Explorer<sup>2</sup> and Google BigQuery<sup>3</sup>. Studies conducted have also used Stack Overflow to mine developer discourse [61, 216, 167, 199, 173, 34, 145, 238, 36, 187, 28, 229].

Due to the enormity of the data produced, we will use qualitative analysis on the questions mined using assistive tools such as NVivo. For this, we will conduct a thematic analysis on the themes of each question mined, the relevance of the question to our research topic, and ensuring strict coding schemes (that reflect our research goals) are adhered to. We refer to Singer et al. [214] and Miles et al. [159] on coding and analysing this qualitative data gathered.

**Phase 2: Personal Opinion Surveys** We follow the triangulation approach proposed by Jick [120] to corroborate the qualitative data of developers’ discussion of Stack Overflow with secondary survey research, thereby validating what people say on Stack Overflow with what is said and done in real life. Kitchenham and Pfleeger [130] provide an introduction on methods used to conduct personal opinion surveys which we adopt as an initial reference in (i) shaping

---

<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 Stack Overflow’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.

our survey objectives around our research goals, (ii) designing a cross-sectional survey, (iii) developing and evaluating our two survey instruments (consisting of a structured questionnaire and semi-structured interview), (iv) evaluating our instruments, (v) obtaining the data and (vi) analysing the data.

As is good practice in developing questionnaire instruments to evaluate their reliability and validity [147], we evaluate our instrument design by asking colleagues to critique it via pilot studies within A<sup>2</sup>I<sup>2</sup>. This assists in identifying any problems with the questionnaire itself and with any issues that may occur with the response rate and follow-up procedures. We follow a similar approach by practicing the interview instrument on colleagues within A<sup>2</sup>I<sup>2</sup>.

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

### 3.4.2 Developing the Initial Framework

Our initial framework is developed using the qualitative and quantitative analyses from the findings of Experiment I. As this is a creative phase in which we are developing a new framework, the exact process by which we develop the initial framework will come to light once more insight is determined. However it is anticipated discussion with other researchers and engineers at A<sup>2</sup>I<sup>2</sup> about the analyses of the findings (i.e., white-boarding sessions of potential ideas from the findings) will help develop our initial documentation framework. This framework will take the shape of a checklist or table, typical of information systems studies (e.g., [140]), that indicate what attributes should be best suited for what needs.

### 3.4.3 Experiment II: Validate Initial Framework

Experiment II extends the *generalised* context of Experiment I by evaluating how the findings of Experiment I translates to context-specific applications. We confirm that the generalised findings are (indeed) genuine by conducting action research in combination with an observational study on software engineers. This experiment is also compromised of: (i) development of prototypes using CIS APIs of differing contexts; (ii) presenting a solution framework to

developers to interpret the improvement of their understanding when using a CIS.

### Relevance and Motivation

Experiment II aims to contextualise the findings from Experiment I; that is, if we add *varying contexts* to the applications we write using CIS APIs, what is needed to extend the *context-agnostic* framework developed in Experiment I? This work relates back to RH2; adding context-specific metadata to the endpoints of these APIs, we can highlight what issues exist when such metadata is not present (RQ2.1) and what types of metadata developers seek (RQ2.2).

Moreover, the implication of the first two hypotheses suggest that applying an API documentation and metadata quality assessment framework may have an effect on other aspects within the software engineering process (RH3). Thus, this experiment also confirms if our framework makes an improvement to software quality, developer productivity and/or developer informativeness (RQ3.1 and RQ3.2).

### Data Collection & Analysis

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

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

This outcome will also help us design the next stage of the experiment, consisting of a comparative controlled study [207] to capture firsthand behaviours and interactions toward how software engineers approach using a CIS with and without our framework applied. We will provide improved documentation and metadata responses of a set of popular cvCISs that is documented with the additional metadata and whose information is organised using our framework.

We then recruit 20 developers of varying experience (from beginner programmer to principal engineer) to complete five tasks under an observational, comparative controlled study, 10 of which will (a) develop with the *new* framework, and the other 10 will (b) develop with the *as-is/existing* documentation. From this, we compare if the framework makes improvements

by capturing metrics and recording the observational sessions for qualitative analysis. We use visual modelling to analyse the qualitative data using matrices [70], maps and networks [159] as these help illustrate any causal, temporal or contextual relationships that may exist to map out the developer’s mindset and difference in approaching the two sets of designs of the same tasks.

## 3.5 Empirical Validity

In Section 3.2, we state that this study primarily adopts a critical theorist stance. Critical theorists assess research quality by the utility of the knowledge gained [80]. Lau [140] established criteria on validating information systems research specifically for action research unifying four dimensions of the research (conceptual foundation, study design, research process, and role expectations) against 22 varying criteria. We also partially adopt constructivism as we attempt to model the developer mindset rich in qualitative data, to which eight strategies identified by Creswell and Creswell [65] cover.

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

### 3.5.1 Threats to Internal Validity

#### Hawthorne Effect

Observational field techniques involving participants often run a risk producing misaligned results from laboratory versus environmental (practical) conditions. This is commonly known as the Hawthorne effect [194, 78] and careful consideration of this effect must be reflected when designing our controlled observation (Section 3.4.3). We aim to carefully explain the purpose and protocol to research participants, encouraging them to act as much as possible as in their practical conditions. We also encourage the ‘think-aloud’ to participants protocol to reinforce this. By highlighting the Hawthorne effect to them, we anticipate that participants will be aware of the condition, and therefore avoid doing things that do not reflect real-world action.

#### Misleading Statements in Interviews

Similarly, threats to the interview survey instrument exist where participants do not often report differences in behaviour from what they actually do in practice [214]. We anticipate that

conducting interviews in a semi-structured manner may assist in following up with unexpected statements (as opposed to structured interviews) and additionally corroborate findings using Jick's concurrent triangulation method [120] to verify potentially misleading statements from participants with questionnaire results and observation findings.

### **Participant Observation Accuracy**

Conducting participant observations is a skill that requires training. While every effort will be instilled to ensure all relevant observations are noted, it is impossible for a single observer to note every possible interaction that occurs in all observations made. Therefore, to validate the consistency of data collected, we may require rater agreement exercises [123] and we will likely use a form of recording device (with participant consent) to ensure all information is transcribed correctly in the interview.

### **Unintended Interviewee Bias**

Interviewers should introduce the research by which participants are involved in by describing an expiation of the research being conducted. However, the amount of information described may impact the bias instilled on the interviewee. For example, if the participant does not understand the goals of the study or feel that they are of the ‘right target’, then it is likely that they may choose not to be involved in the study or give misleading answers. On the other hand, if interviewees are told too much information, then they may filter responses and leave out vital data that the interviewer may be interested in. To mitigate this, varying levels of information will be ‘tested’ against colleagues to determine the right level of how much information is divulged at the beginning of the interview.

### **Poor Questionnaire Responses**

Unless significant inducements are offered, Singer et al. [214] report that a consistent response rate of 5% has been found in software engineering questionnaires distributed and in information systems the median response rates for surveys are 60% [37]. We observe that low response rates may adversely effect the findings of our survey, typically as software engineers find little time to do them. Tackling this issue can be resolved by carefully designing succinct, unambiguous and well-worded questions that we will verify against our colleagues and within the pilot study in A<sup>2</sup>I<sup>2</sup>, wherein any adjustments made from the pilot study due to unexpected poor quality of the questionnaire will be reported and explained. We also adopt research conducted in the field of questionnaire design, such as ensuring all scales are worded with labels [136] or using a summating rating scale [219] to address a specific topic of interest if

people are to make mistakes in their response or answer in different ways at different times. Similar effects exist to that above where what occurs in reality is not what is reflected in our results; we refer to our concurrent triangulation approach to gap this risk.

### 3.5.2 Threats to External Validity

#### Representative Sample Size

Our results must generalise by ensuring a representative subset of the target population is found. If results do not generalise, then all that is found is potentially of little more value than personal anecdote [130]. Therefore, designing a well-defined sample frame to determine which developers we wish to target is empirical. For this, we refer to Kitchenham and Pfleeger [130].

#### Student Cohorts

External validity is typically undermined when students are recruited in software engineering research, which is common practice [80]. Analytical argument is required to describe why results on students are reflective of results found on developers in industry. Therefore, we anticipate that—through industry contacts at A<sup>2</sup>I<sup>2</sup>—we will be able to contact developers in industry, thereby minimising our reliance to use students as participants.

#### Concurrent Triangulation Strategy

A drawback with the concurrent triangulation strategy is that multiple sources of data are concurrently collected within the same time. Collecting and analysing data *sequentially*, instead of concurrently, allows for time to analyse data between studies, thus allowing each analysis to adapt as more emerging results are explored. Easterbrook et al. [80] states that the challenge in this approach is that it may be difficult for researchers to compare results of multiple analyses or resolve contradictions that begin to arise when this is performed concurrently. A mitigation strategy, should this occur, would be to seek out further sources of evidence, or even re-conduct a follow-up study if time permits.

### 3.5.3 Threats to Construct Validity

#### Developer Informativeness

RH3 describes that if we improve the documentation of CIS APIs, then developers are more informed/educated in what they do. This therefore increases their productivity and the quality of the applications they build. However, the construct of ‘informativeness’ is difficult to

capture with standalone metrics, and using simple quantitative metrics such as time taken to complete a task or lines of code to implement it may not reflect that a developer is more ‘informed’. Therefore, we propose further investigation into understanding how to measure informativeness of software engineers to ensure that this construct validity does not impact our results too greatly.



# Chapter 4

## Project Status

### 4.1 Project Timeline

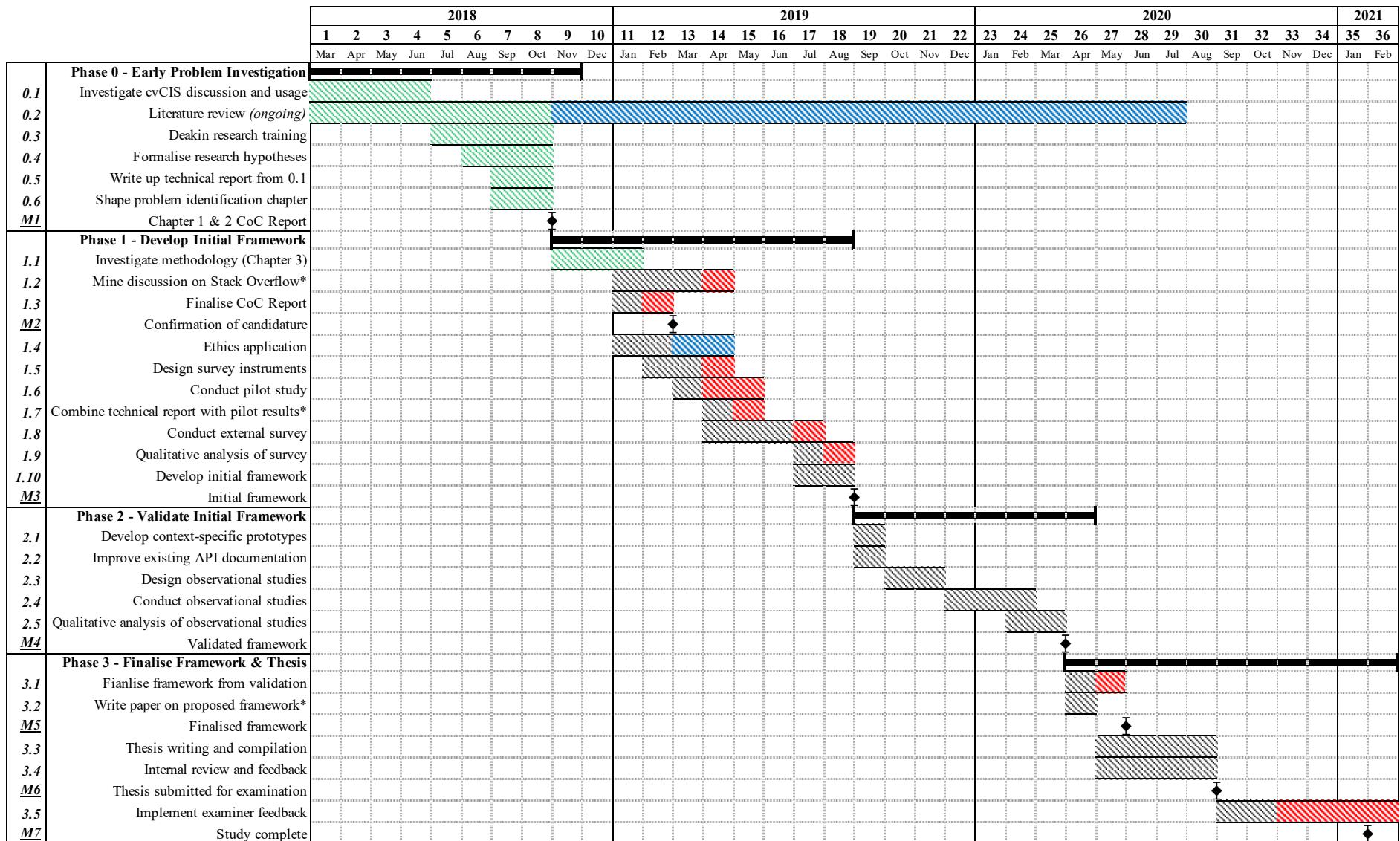
A project timeline is given in Figure 4.1. The project timeline consists of four phases (one preliminary) and seven milestones:

**Phase 0. Early problem investigation.** Early investigation into the problem area, exploring potential aspects for research hypotheses to develop. The outcome is the first two chapters of this report (Milestone 1) and the early stages of a technical report attached in Appendix B.

**Phase 1. Develop initial framework.** Process of developing initial framework by mining developer forums (Stack Overflow) and designing and conduct our surveys. This maps to Experiment I (see Section 3.4.3). The outcome is an initial context-agnostic CIS API documentation quality assessment framework (Milestone 3) and confirmation of candidature complete (Milestone 2). As ethics approval is required, the current draft Ethics Application is attached in Appendix C.

**Phase 2. Validate initial framework.** Validating that the framework we develop in Phase 1 correctly reflects the needs of developers by trialing it in controlled observational studies. This maps to Experiment II (Section 3.4.3). The outcome is a framework that has been validated against developers (Milestone 4).

**Phase 3. Finalise framework and thesis.** Implement any additional findings from the validation of Phase 2 into the framework, thereby finalising the framework (Milestone 5). Original contribution is the framework and therefore the thesis can be submitted (Milestone 6) and reviewed before the study is complete (Milestone 7).



**Figure 4.1:** Project timeline.

**Key:** Green Bar = Completed Task; Grey Bar = Planned Task; Red Bar = Slack Time; Blue Bar = Ongoing Task; Thick Black Line = Phase Duration; Diamond = Milestone; Asterisk = Potential publications.

## 4.2 Completed Work

- **Ethics Application:** The progress of the ethics application (Appendix C) is well underway with the Low Risk Application Form filled out. Partial sections of the Plain Language Statement (PLS) is also complete but unattached with this document. We plan to submit ethics to Deakin SEBE HEAG within the next two weeks.
- **Technical Report:** Initial findings from our preliminary investigations of cvCISs are attached within Appendix B. We plan on triangulating these findings with further supporting results from Stack Overflow mining and survey research from our pilot study.
- **Progress on Thesis:** We have approached this report as the initial writeup of our thesis. We have substantially introduced our problem domain in Chapter 1, a broad spectrum of literature in Chapter 2, and our approach to the overall study in Chapter 3. Whilst we aim to refine and enhance this work, this document will serve as a skeleton to the final thesis as we indicate below.
- **Stack Overflow Mining:** We have begun initial stages of our mining of Stack Overflow with preliminary results to be imported to qualitative analysis software such as NVivo.

## 4.3 Proposed Chapters

- **Chapter 4 - Developer Opinions:** This chapter maps to Experiment I (Section 3.4.1) by reporting directly related work regarding this experiment (i.e., surveys and interviews on nondeterministic systems, mining Stack Overflow for nondeterministic systems etc.), expanding on our methodology, and reporting our findings.
- **Chapter 5/6 - Proposed Framework:** This chapter will introduce our proposed framework (initially the initial framework will be presented but the updated framework coming from our evaluation will be discussed). It will also describe the methodology by which we converted our qualitative and quantitative analyses to the framework. The order of the chapter is to be determined.
- **Chapter 5/6 - Evaluation of Framework:** This chapter maps to Experiment II (Section 3.4.3) and reports work directly related to evaluation of frameworks, expanding our methodology proposed and reporting our findings from the evaluation. The order of this chapter is to be determined.
- **Chapter 7 - Conclusions:** This chapter will present concluding remarks of the study.

**Table 4.1:** Mapping of project tasks (see Figure 4.1) that address each of the research questions.

Research Question	Task #
RQ1.1	0.1, 0.2, 1.2
RQ1.2	1.2, 1.6, 1.8, 1.10
RQ1.3	1.6, 1.8, 1.10
RQ2.1	2.1
RQ2.2	2.4, 2.5
RQ3.1	2.5
RQ3.2	2.5

**Table 4.2:** Potential publications and respective targeted venues from outcomes of major bodies of work (see Figure 4.1).

Venue	Ranking	Task #	Location	Submissions Due	Date
ESEC/FSE	A2	1.2	Tallinn, Estonia	20 Feb 2019	26–30 Aug 2019
ASE	A1	1.2, 1.7	San Diego, USA	6 May 2019	11–15 Nov 2019
CHI	A1	1.2, 1.7	Honolulu, USA	13 Sep 2019	25–30 Apr 2020
ICSE	A1	1.7	Seoul, South Korea	TBA	23–29 May 2020
MSR	A1	1.2, 1.7	Seoul, South Korea	TBA	23–29 May 2020
ICML	A1	1.7	Vienna, Austria	TBA	12–17 Jul 2020
ICMLA	B2	1.7, 3.2	Copenhagen, Denmark	15 Nov 2020	11–12 Jun 2021
TOSEM	N/A	1.2, 1.7; 3.2	N/A	N/A	N/A
IEEE Software	N/A	1.2, 1.7; 3.2	N/A	N/A	N/A
TSE	N/A	1.2, 1.7; 3.2	N/A	N/A	N/A

# References

- [1] “Detecting research-methodology in an image,” <https://docs.aws.amazon.com/rekognition/latest/dg/research-methodology-detect-research-methodology-image.html>, accessed: 28 August 2018.
- [2] “Detecting objects and scenes,” <https://docs.aws.amazon.com/rekognition/latest/dg/research-methodology.html>, accessed: 28 August 2018.
- [3] “Amazon Rekognition,” <https://aws.amazon.com/rekognition>, accessed: 13 September 2018.
- [4] “Home - affectiva : Affectiva,” <https://www.affectiva.com>, accessed: 15 October 2018.
- [5] “SOAP, Representational state transfer - Explore - Google Trends.”
- [6] “How to call the Computer Vision API,” <https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/vision-api-how-to-topics/howtocallvisionapi>, accessed: 28 August 2018.
- [7] “azure-sdk-for-java/ImageTag.java,” <https://github.com/Azure/azure-sdk-for-java/blob/8d70f41fbdb88b3a9297af5ba24551cf26f40ad4/cognitiveservices/data-plane/vision/computervision/src/main/java/com/microsoft/azure/cognitiveservices/vision/computervision/models/ImageTag.java#L24>, accessed: 28 August 2018.
- [8] “Image Processing with the Computer Vision API | Microsoft Azure,” <https://azure.microsoft.com/en-au/services/cognitive-services/computer-vision/>, accessed: 13 September 2018.
- [9] “Clarifai,” <https://www.clarifai.com>, accessed: 13 September 2018.
- [10] “Image Recognition API & Visual Search Results,” <https://docs.aws.amazon.com/rekognition/latest/dg/research-methodology-detect-research-methodology-image.html>, accessed: 13 September 2018.

- [11] “The face recognition company - cognitec,” <http://www.cognitec.com>, accessed: 15 October 2018.
- [12] “Image recognition api | deepai,” <https://deepai.org/ai-image-processing>, accessed: 26 September 2018.
- [13] “Detect research-methodology | Google Cloud Vision API Documentation | Google Cloud,” <https://cloud.google.com/vision/docs/research-methodology>, accessed: 28 August 2018.
- [14] “Class EntityAnnotation | Google.Cloud.Vision.V1,” <https://googlecloudplatform.github.io/google-cloud-dotnet/docs/Google.Cloud.Vision.V1/api/Google.Cloud.Vision.V1.EntityAnnotation.html>, accessed: 28 August 2018.
- [15] “Vision API - Image Content Analysis | Cloud Vision API | Google Cloud,” <https://cloud.google.com/vision/>, accessed: 13 September 2018.
- [16] “Watson visual recognition,” <https://www.ibm.com/watson/services/visual-recognition/>, accessed: 13 September 2018.
- [17] “Imagga - powerful image recognition apis for automated categorization & tagging in the cloud and on-premises,” <https://imagga.com>, accessed: 13 September 2018.
- [18] *Pivotal Cloud Foundry, Google ML, and Spring*, Dec. 2017.
- [19] “Kairos: Serving businesses with face recognition,” <https://www.kairos.com>, accessed: 15 October 2018.
- [20] “Amazon Mechanical Turk,” <https://www.mturk.com>, accessed: 15 October 2018.
- [21] *Machine learning with Google APIs*, Jan. 2019.
- [22] “Detecting research-methodology in an image,” <https://docs.aws.amazon.com/rekognition/latest/dg/research-methodology-detect-research-methodology-image.html>, accessed: 13 September 2018.
- [23] “Scale: API for Training Data,” <https://www.scaleapi.com>, accessed: 15 October 2018.
- [24] “Image recognition - talkwalker,” <https://www.talkwalker.com/image-recognition>, accessed: 13 September 2018.

- [25] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org. [Online]. Available: <https://www.tensorflow.org/>
- [26] R. E. Al-Qutaish, “Quality models in software engineering literature: an analytical and comparative study,” *Journal of American Science*, vol. 6, no. 3, pp. 166–175, 2010.
- [27] H. Allahyari and N. Lavesson, “User-oriented assessment of classification model understandability,” in *11th scandinavian conference on Artificial intelligence*. IOS Press, 2011.
- [28] M. Allamanis and C. Sutton, “Why, when, and what: analyzing stack overflow questions by topic, type, and code,” in *Proceedings of the 10th Working Conference on Mining Software Repositories*. IEEE Press, 2013, pp. 53–56.
- [29] K. Arnold, “Programmers are people, too,” *ACM Queue*, vol. 3, no. 5, pp. 54–59, 2005.
- [30] W. R. Ashby and J. R. Pierce, “An Introduction to Cybernetics,” *Physics Today*, vol. 10, no. 7, pp. 34–36, Jul. 1957.
- [31] M. G. Augasta and T. Kathirvalavakumar, “Reverse engineering the neural networks for rule extraction in classification problems,” *Neural processing letters*, vol. 35, no. 2, pp. 131–150, 2012.
- [32] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, and K.-R. MÃžller, “How to explain individual classification decisions,” *Journal of Machine Learning Research*, vol. 11, no. Jun, pp. 1803–1831, 2010.
- [33] B. Baesens, C. Mues, M. De Backer, J. Vanthienen, and R. Setiono, “Building Intelligent Credit Scoring Systems Using Decision Tables.” *ICEIS*, 2003.
- [34] K. Bajaj, K. Pattabiraman, and A. Mesbah, “Mining questions asked by web developers,” in *Proceedings of the 11th Working Conference on Mining Software Repositories*. ACM, 2014, pp. 112–121.

- [35] K. Ballinger. (2014, Dec.) **Simplicity and Utility, or, Why SOAP Lost.** [Online]. Available: <http://keithba.net/simplicity-and-utility-or-why-soap-lost>
- [36] 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.
- [37] Y. Baruch, “Response rate in academic studies—A comparative analysis,” *Human relations*, vol. 52, no. 4, pp. 421–438, 1999.
- [38] L. Bass, P. Clements, and R. Kazman, *Software architecture in practice*. Addison-Wesley Professional, 2003.
- [39] 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.
- [40] A. Ben-David, “Monotonicity maintenance in information-theoretic machine learning algorithms,” *Machine Learning*, vol. 19, no. 1, pp. 29–43, 1995.
- [41] T. Berners-Lee, R. Fielding, and L. Masinter, “Uniform resource identifier (URI): Generic syntax,” Tech. Rep., 2004.
- [42] J. Bessin, “The Business Value of Quality,” *IBM developerWorks, June*, vol. 15, 2004.
- [43] J. J. Blake, L. P. Maguire, B. Roche, T. M. McGinnity, and L. J. McDaid, “The Implementation of Fuzzy Systems, Neural Networks and Fuzzy Neural Networks using FPGAs.” *Inf. Sci.*, 1998.
- [44] B. Boehm and V. R. Basili, “Software defect reduction top 10 list,” *Foundations of empirical software engineering: the legacy of Victor R. Basili*, vol. 426, no. 37, 2005.
- [45] B. W. Boehm, *Software engineering economics*. Prentice-hall Englewood Cliffs (NJ), 1981, vol. 197.
- [46] B. W. Boehm, J. R. Brown, and H. Kaspar, “Characteristics of software quality,” 1978.
- [47] O. Boz, “Extracting decision trees from trained neural networks,” in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2002, pp. 456–461.
- [48] M. Bramer, *Principles of data mining*. Springer, 2007, vol. 180.

- [49] 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 Systems*. ACM, 2009, pp. 1589–1598.
- [50] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. CRC press, 1984.
- [51] M. Bunge, “A General Black Box Theory,” *Philosophy of Science*, vol. 30, no. 4, pp. 346–358, Oct. 1963.
- [52] A. Bussone, S. Stumpf, and D. O’Sullivan, “The Role of Explanations on Trust and Reliance in Clinical Decision Support Systems.” *ICHI*, 2015.
- [53] C. Calhoun, *Critical social theory: Culture, history, and the challenge of difference*. Wiley-Blackwell, 1995.
- [54] G. Canfora, “User-side testing of web services,” in *Software Maintenance and Reengineering, 2005. CSMR 2005. Ninth European Conference on*. IEEE, 2005, p. 301.
- [55] 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.
- [56] 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.” *KDD*, pp. 1721–1730, 2015.
- [57] F. Casati, H. Kuno, G. Alonso, and V. Machiraju, “Web Services-Concepts, Architectures and Applications,” 2003.
- [58] J. P. Cavano, J. A. McCall, J. P. Cavano, J. A. McCall, J. P. Cavano, and J. A. McCall, “A framework for the measurement of software quality,” *ACM SIGSOFT Software Engineering Notes*, vol. 3, no. 5, pp. 133–139, Nov. 1978.
- [59] 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*, vol. 4, no. 1, pp. 19–26, 1991.
- [60] J. Cheng and R. Greiner, “Learning bayesian belief network classifiers: Algorithms and system,” in *Conference of the Canadian Society for Computational Studies of Intelligence*. Springer, 2001, pp. 141–151.

- [61] E. Choi, N. Yoshida, R. G. Kula, and K. Inoue, “What do practitioners ask about code clone? a preliminary investigation of stack overflow.” in *IWSC*, 2015, pp. 49–50.
- [62] Digital Inc., “Case study: Finding defects earlier yields enormous savings,” 2003.
- [63] P. Clark and R. Boswell, “Rule induction with CN2: Some recent improvements,” in *European Working Session on Learning*. Springer, 1991, pp. 151–163.
- [64] M. Craven and J. W. Shavlik, “Extracting Tree-Structured Representations of Trained Networks.” *NIPS*, 1995.
- [65] J. W. Creswell and J. D. Creswell, *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications, 2017.
- [66] P. B. Crosby, “Quality is free: The art of making quality free,” *New York*, 1979.
- [67] 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 and Management*, vol. 2, no. 2, pp. 1–3, 2017.
- [68] R. Davison, M. G. Martinsons, and N. Kock, “Principles of canonical action research,” *Information systems journal*, vol. 14, no. 1, pp. 65–86, 2004.
- [69] 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.
- [70] I. Dey, *Qualitative data analysis: A user friendly guide for social scientists*. Routledge, 2003.
- [71] 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. 251–280, 2000.
- [72] V. C. Dibia, M. Ashoori, A. Cox, and J. D. Weisz, “TJBot,” in *the 2017 CHI Conference Extended Abstracts*. New York, New York, USA: ACM Press, 2017, pp. 381–384.
- [73] 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.

- [74] P. Domingos, “Occam’s two razors: The sharp and the blunt,” in *KDD*, 1998, pp. 37–43.
- [75] B. Dorn and M. Guzdial, “Learning on the job: characterizing the programming knowledge and learning strategies of web designers,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2010, pp. 703–712.
- [76] F. Doshi-Velez and B. Kim, “Towards a rigorous science of interpretable machine learning,” 2017.
- [77] F. Doshi-Velez, M. Kortz, R. Budish, C. Bavitz, S. Gershman, D. O’Brien, S. Schieber, J. Waldo, D. Weinberger, and A. Wood, “Accountability of AI Under the Law: The Role of Explanation,” *arXiv.org*, Nov. 2017.
- [78] S. W. Draper. The Hawthorne, Pygmalion, Placebo and other effects of expectation: some notes.
- [79] R. G. Dromey, “A model for software product quality,” *IEEE Transactions on Software Engineering*, vol. 21, no. 2, pp. 146–162, 1995.
- [80] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, “Selecting empirical methods for software engineering research,” in *Guide to Advanced Empirical Software Engineering*. Springer Science & Business Media, Nov. 2007, pp. 285–311.
- [81] B. Ehteshami Bejnordi, M. Veta, P. Johannes van Diest, B. van Ginneken, N. Karssemeijer, G. Litjens, J. A. W. M. van der Laak, and the CAMELYON16 Consortium, M. Hermsen, Q. F. Manson, M. Balkenhol, O. Geessink, N. Stathonikos, M. C. 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. Vylegzhanin, 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. Ruusuvuori, K. Liimatainen, S. Albarqouni, B. Mungal, A. George, S. Demirci, N. Navab, S. Watanabe, S. Seno, Y. Takenaka, H. Matsuda, H. Ahmady 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,” *JAMA*, vol. 318, no. 22, pp. 2199–12, Dec. 2017.
- [82] W. Elazmeh, W. Matwin, D. O’Sullivan, W. Michalowski, and W. Farion, “Insights from predicting pediatric asthma exacerbations from retrospective clinical data,” in

- Evaluation Methods for Machine Learning II—Papers from 2007 AAAI Workshop*, 2007, pp. 10–15.
- [83] 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 IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1625–1634.
- [84] ———, “Robust Physical-World Attacks on Deep Learning Visual Classification,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1625–1634.
- [85] A. J. Feelders, “Prior knowledge in economic applications of data mining,” in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2000, pp. 395–400.
- [86] R. T. Fielding, “*Architectural Styles and the Design of Network-based Software Architectures*,” Ph.D. dissertation, University of California, Irvine.
- [87] I. Finalyson, “Nondeterministic Finite Automata,” 2018.
- [88] H. Foster, S. Uchitel, J. Magee, and J. Kramer, “Model-based verification of web service compositions,” in *Automated Software Engineering, 2003. Proceedings. 18th IEEE International Conference on*. IEEE, 2003, pp. 152–161.
- [89] A. A. Freitas, “A critical review of multi-objective optimization in data mining: a position paper,” *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 2, pp. 77–86, 2004.
- [90] ———, “Comprehensible classification models,” *ACM SIGKDD Explorations Newsletter*, vol. 15, no. 1, pp. 1–10, Mar. 2014.
- [91] A. A. Freitas, D. C. Wieser, and R. Apweiler, “On the importance of comprehensible classification models for protein function prediction,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 7, no. 1, pp. 172–182, 2010.
- [92] B. J. Frey and D. Dueck, “Clustering by Passing Messages Between Data Points,” *Science*, vol. 315, no. 5814, pp. 972–976, Feb. 2007.
- [93] N. Friedman, D. Geiger, and M. Goldszmidt, “Bayesian network classifiers,” *Machine Learning*, vol. 29, no. 2-3, pp. 131–163, 1997.

- [94] G. Fung, S. Sandilya, and R. B. Rao, "Rule extraction from linear support vector machines," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining.* ACM, 2005, pp. 32–40.
- [95] D. A. Garvin and W. D. P. Quality, "Really Mean," *Sloan management review*, vol. 25, 1984.
- [96] H. L. Gilmore, "Product conformance cost," *Quality progress*, vol. 7, no. 5, pp. 16–19, 1974.
- [97] B. Goodman and S. R. Flaxman, "EU regulations on algorithmic decision-making and a "right to explanation". ." *CoRR*, 2016.
- [98] P. D. Grünwald, *The minimum description length principle.* MIT press, 2007.
- [99] M. J. Hadley, "Web application description language (WADL)," 2006.
- [100] H. A. Haensle, C. Fink, R. Schneiderbauer, F. Toberer, T. Buhl, A. Blum, A. Kalloo, A. B. H. Hassen, L. Thomas, A. Enk, L. Uhlmann, Reader study level-I and level-II Groups, C. Alt, M. Arenbergerova, R. Bakos, A. Baltzer, I. Bertlich, A. Blum, T. Bokor-Billmann, J. Bowling, N. Braghiroli, R. Braun, K. Buder-Bakhaya, T. Buhl, H. Cabo, L. Cabrijan, N. Covic, 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. Wölbing, 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.
- [101] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*, ser. Data Mining, Inference, and Prediction. Springer Science & Business Media, Jan. 2001.
- [102] B. Hayete and J. R. Bienkowska, "GOTrees - Predicting GO Associations from Protein Domain Composition Using Decision Trees." *Pacific Symposium on Biocomputing*, pp. 127–138, 2005.

- [103] R. Heckel and M. Lohmann, “Towards contract-based testing of web services,” *Electronic Notes in Theoretical Computer Science*, vol. 116, pp. 145–156, 2005.
- [104] 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. Oct, pp. 49–75, 2000.
- [105] M. Henning, “API design matters,” *Commun. ACM*, vol. 52, no. 5, pp. 46–56, 2009.
- [106] H. Hosseini, B. Xiao, and R. Poovendran, “Google’s Cloud Vision API is Not Robust to Noise,” in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, Jan. 2018, pp. 101–105.
- [107] C. Howard. (2018, May) Introducing Google AI. [Online]. Available: <https://ai.googleblog.com/2018/05/introducing-google-ai.html>
- [108] J. Huang and C. X. Ling, “Using AUC and accuracy in evaluating learning algorithms,” *IEEE Transactions on knowledge and Data Engineering*, vol. 17, no. 3, pp. 299–310, 2005.
- [109] J. Huysmans, K. Dejaeger, C. Mues, J. Vanthienen, and B. Baesens, “An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models,” *Decision Support Systems*, vol. 51, no. 1, pp. 141–154, Apr. 2011.
- [110] K. Hwang, *Cloud Computing for Machine Learning and Cognitive Applications: A Machine Learning Approach*. MIT Press, 2017.
- [111] IEEE, “IEEE Standard Glossary of Software Engineering Terminology,” 1990.
- [112] International Organization for Standardization, “Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models,” 2011.
- [113] ——, “Quality – Vocabulary,” ISO/IEC, 1986.
- [114] ——, “Quality management systems – Fundamentals and vocabulary,” ISO/IEC, 2015.
- [115] ——, “Information technology – Software product quality,” Nov. 1999.
- [116] A. Iyengar, “Supporting Data Analytics Applications Which Utilize Cognitive Services,” in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, May 2017, pp. 1856–1864.

- [117] N. Japkowicz and M. Shah, *Evaluating learning algorithms: a classification perspective*. Cambridge University Press, 2011.
- [118] M. W. M. Jaspers, M. Smeulers, H. Vermeulen, and L. W. Peute, “Effects of clinical decision-support systems on practitioner performance and patient outcomes: a synthesis of high-quality systematic review findings,” *Journal of the American Medical Informatics Association*, vol. 18, no. 3, pp. 327–334, 2011.
- [119] T. Jiang and A. E. Keating, “AVID: an integrative framework for discovering functional relationships among proteins,” *BMC bioinformatics*, vol. 6, no. 1, p. 136, 2005.
- [120] T. Jick, *Mixing qualitative and quantitative methods*, ser. triangulation in action, 1979.
- [121] Y. Jin, *Multi-objective machine learning*. Springer Science & Business Media, 2006, vol. 16.
- [122] U. Johansson and L. Niklasson, “Evolving decision trees using oracle guides,” in *IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*. IEEE, 2009.
- [123] C. M. Judd, E. R. Smith, and L. H. Kidder, “Research Methods in Social Relations, Fort Worth: Holt, Rinehart and Winston,” 1991.
- [124] J. M. Juran, *Juran on planning for quality*. Collier Macmillan, 1988.
- [125] N. Juristo and A. M. Moreno, *Basics of Software Engineering Experimentation*. Springer Science & Business Media, Mar. 2013.
- [126] A. Karwath and R. D. King, “Homology induction: the use of machine learning to improve sequence similarity searches,” *BMC bioinformatics*, vol. 3, no. 1, p. 11, 2002.
- [127] K. A. Kaufman and R. S. Michalski, “Learning from inconsistent and noisy data: the AQ18 approach,” in *International Symposium on Methodologies for Intelligent Systems*. Springer, 1999, pp. 411–419.
- [128] B. Kim, *Interactive and Interpretable Machine Learning Models for Human Machine Collaboration*. Massachusetts Institute of Technology, 2015.
- [129] B. Kim, C. Rudin, and J. A. Shah, “The Bayesian Case Model - A Generative Approach for Case-Based Reasoning and Prototype Classification.” *NIPS*, 2014.
- [130] B. A. Kitchenham and S. L. Pfleeger, “Personal opinion surveys,” in *Guide to Advanced Empirical Software Engineering*. Springer Science & Business Media, Nov. 2007, pp. 63–92.

- [131] H. K. Klein and M. D. Myers, “A set of principles for conducting and evaluating interpretive field studies in information systems,” *MIS quarterly*, pp. 67–93, 1999.
- [132] 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.
- [133] A. J. Ko, B. A. Myers, and H. H. Aung, “Six learning barriers in end-user programming systems,” in *2004 IEEE Symposium on Visual Languages-Human Centric Computing*. IEEE, 2004, pp. 199–206.
- [134] I. Kononenko, “Inductive and Bayesian learning in medical diagnosis,” *Applied Artificial Intelligence an International Journal*, vol. 7, no. 4, pp. 317–337, 1993.
- [135] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks.” 2012.
- [136] J. A. Krosnick, “Survey research,” *Annual Review of Psychology*, vol. 50, no. 1, pp. 537–567, 1999.
- [137] A. Kurakin, I. Goodfellow, and S. Bengio, “Adversarial examples in the physical world,” *arXiv.org*, Jul. 2016.
- [138] G. Laforge, “Machine Intelligence at Google Scale,” in *QCon*, Jun. 2018, pp. 1–58.
- [139] H. Lakkaraju, S. H. Bach, and J. Leskovec, “Interpretable Decision Sets - A Joint Framework for Description and Prediction.” *KDD*, pp. 1675–1684, 2016.
- [140] F. Lau, “Toward a framework for action research in information systems studies,” *Information Technology & People*, vol. 12, no. 2, pp. 148–176, 1999.
- [141] N. Lavrač, “Selected techniques for data mining in medicine,” *Artificial intelligence in medicine*, vol. 16, no. 1, pp. 3–23, 1999.
- [142] T. Lei, R. Barzilay, and T. Jaakkola, “Rationalizing Neural Predictions,” *arXiv.org*, Jun. 2016.
- [143] T. C. Lethbridge, S. E. Sim, and J. Singer, “Studying Software Engineers: Data Collection Techniques for Software Field Studies,” *Empirical Software Engineering*, vol. 10, no. 3, pp. 311–341, Jul. 2005.

- [144] E. Lima, C. Mues, and B. Baesens, “Domain knowledge integration in data mining using decision tables: Case studies in churn prediction,” *Journal of the Operational Research Society*, vol. 60, no. 8, pp. 1096–1106, 2009.
- [145] M. Linares-Vásquez, G. Bavota, M. Di Penta, R. Oliveto, and D. Poshyvanyk, “How do api changes trigger stack overflow discussions? a study on the android sdk,” in *proceedings of the 22nd International Conference on Program Comprehension*. ACM, 2014, pp. 83–94.
- [146] Z. C. Lipton, “The Mythos of Model Interpretability.” *CoRR*, 2016.
- [147] M. S. Litwin and A. Fink, *How to measure survey reliability and validity*. Sage, 1995, vol. 7.
- [148] Y. Liu, T. Kohlberger, M. Norouzi, G. E. Dahl, J. L. Smith, A. Mohtashamian, N. Olson, L. H. Peng, J. D. Hipp, and M. C. Stumpe, “Artificial Intelligence-Based Breast Cancer Nodal Metastasis Detection,” *Archives of Pathology & Laboratory Medicine*, pp. arpa.2018–0147–OA–11, Oct. 2018.
- [149] D. Lo Giudice, C. Mines, A. LeClair, R. Curran, and A. Homan, “How AI Will Change Software Development And Applications,” Tech. Rep., Nov. 2016.
- [150] L. Mandel. (2008, May) Describe REST Web services with WSDL 2.0. [Online]. Available: <https://www.ibm.com/developerworks/webservices/library/ws-restwsdl/>
- [151] T. E. Marshall and S. L. Lambert, “Cloud-based intelligent accounting applications: accounting task automation using IBM watson cognitive computing,” *Journal of Emerging Technologies in Accounting*, vol. 15, no. 1, pp. 199–215, 2018.
- [152] D. Martens, J. Vanthienen, W. Verbeke, and B. Baesens, “Performance of classification models from a user perspective,” *Decision Support Systems*, vol. 51, no. 4, pp. 782–793, 2011.
- [153] J. A. McCall, “Factors in software quality,” *US Rome Air development center reports*, 1977.
- [154] J. A. McCall, P. K. Richards, and G. F. Walters, “Factors in software quality. volume i. concepts and definitions of software quality,” Tech. Rep., 1977.
- [155] J. McCarthy, “Programs with Common Sense,” Cambridge, MA, USA, Tech. Rep., 1960.

- [156] J. Meltzoff, *Critical thinking about research: Psychology and related fields.* American psychological association, 1998.
- [157] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, “Machine Learning and Statistical Classification of Artificial intelligence,” 1994.
- [158] D. Michie, “Machine Learning in the Next Five Years.” *EWSL*, 1988.
- [159] M. B. Miles, A. M. Huberman, M. A. Huberman, and M. Huberman, *Qualitative data analysis: An expanded sourcebook.* sage, 1994.
- [160] D. L. Moody, “The ‘Physics’ of Notations - Toward a Scientific Basis for Constructing Visual Notations in Software Engineering.” *IEEE Trans. Software Eng.*, 2009.
- [161] B. A. Myers, S. Y. Jeong, J. Stylos, R. Ehret, A. Efeoglu, Y. Xie, J. Beaton, J. Karstens, and D. K. Busse, “Studying the Documentation of an API for Enterprise Service-Oriented Architecture,” *services.igi-global.com*, no. chapter 4, pp. 1–8, Jan. 1.
- [162] B. A. Myers, S. Y. Jeong, Y. Xie, J. Beaton, J. Stylos, R. Ehret, J. Karstens, A. Efeoglu, and D. K. Busse, “Studying the documentation of an API for enterprise Service-Oriented Architecture,” *Journal of Organizational and End User Computing (JOEUC)*, vol. 22, no. 1, pp. 23–51, 2010.
- [163] S. Nakajima, “Model-checking verification for reliable web service,” in *OOPSLA 2002 Workshop on Object-Oriented Web Services, Seattle, Washington*, 2002.
- [164] M. Narayanan, E. Chen, J. He, B. Kim, S. Gershman, and F. Doshi-Velez, “How do Humans Understand Explanations from Machine Learning Systems? An Evaluation of the Human-Interpretability of Explanation.” *CoRR*, 2018.
- [165] S. Narayanan and S. A. McIlraith, “Simulation, verification and automated composition of web services,” in *Proceedings of the 11th international conference on World Wide Web*. ACM, 2002, pp. 77–88.
- [166] B. J. Nelson, “Remote procedure call,” Ph.D. dissertation, Carnegie Mellon University, 1981.
- [167] N. Novielli, F. Calefato, and F. Lanubile, “The challenges of sentiment detection in the social programmer ecosystem,” in *Proceedings of the 7th International Workshop on Social Software Engineering*. ACM, 2015, pp. 33–40.

- [168] 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.
- [169] Open Software Foundation, “Part 3: DCE Remote Procedure Call (RPC),” in *OSF DCE application development guide: revision 1.0*. Prentice Hall, Dec. 1991.
- [170] N. Oreskes, K. Shrader-Frechette, and K. Belitz, “Verification, Validation, and Confirmation of Numerical Models in the Earth Sciences,” *Science*, vol. 263, no. 5147, pp. 641–646, 1994.
- [171] A. L. M. Ortiz, “Curating Content with Google Machine Learning Application Programming Interfaces,” in *EIAPortugal*, Jul. 2017.
- [172] F. E. Otero and A. A. Freitas, “Improving the interpretability of classification rules discovered by an ant colony algorithm,” in *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, 2013, pp. 73–80.
- [173] A. Pal, S. Chang, and J. A. Konstan, “Evolution of experts in question answering communities.” in *ICWSM*, 2012.
- [174] A. Parasuraman, V. A. Zeithaml, and L. L. Berry, “Servqual: A multiple-item scale for measuring consumer perceptions of service quality,” *Journal of retailing*, vol. 64, no. 1, pp. 12–29, 1988.
- [175] R. Parekh, “Designing AI at Scale to Power Everyday Life,” in *the 23rd ACM SIGKDD International Conference*. New York, New York, USA: ACM Press, 2017, pp. 27–27.
- [176] C. Pautasso, O. Zimmermann, and F. Leymann, “**RESTful Web Services vs. “Big” Web Services: Making the Right Architectural Decision** ,” in *Proceedings of the 17th international conference on World Wide Web*. ACM, 2008, pp. 805–814.
- [177] M. Pazzani, “Comprehensible knowledge discovery: gaining insight from data,” in *First Federal Data Mining Conference and Exposition*, 1997, pp. 73–82.
- [178] 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. 05, pp. 380–385, 2001.
- [179] J. Pearl, “The Seven Tools of Causal Inference with Reflections on Machine Learning,” 2018.

- [180] K. Petersen and C. Gencel, “Worldviews, Research Methods, and their Relationship to Validity in Empirical Software Engineering Research,” in *2013 Joint Conference of the 23nd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement (IWSM-MENSURA)*. IEEE, Jan. 2019, pp. 81–89.
- [181] H. Pham, *Software reliability*. Springer Science & Business Media, 2000.
- [182] M. Piccioni, C. A. Furia, and B. Meyer, “An Empirical Study of API Usability,” in *2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*. IEEE, 2013, pp. 5–14.
- [183] R. M. Pirsig, *Zen and the art of motorcycle maintenance: An inquiry into values*. William Morrow and Company, 1974.
- [184] R. S. Pressman, *Software engineering: a practitioner’s approach*. Palgrave Macmillan, 2005.
- [185] J. R. Quinlan, “C4. 5: Programming for machine learning,” *Morgan Kauffmann*, vol. 38, p. 48, 1993.
- [186] ——, “Some elements of machine learning,” in *International Conference on Inductive Logic Programming*. Springer, 1999, pp. 15–18.
- [187] M. Rebouças, G. Pinto, F. Ebert, W. Torres, A. Serebrenik, and F. Castor, “An empirical study on the usage of the swift programming language,” in *Software Analysis, Evolution, and Reengineering (SANER), 2016 IEEE 23rd International Conference on*. IEEE, 2016, pp. 634–638.
- [188] 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.” *WorldCIST*, vol. 746, no. 12, pp. 174–184, 2018.
- [189] M. T. Ribeiro, S. Singh, and C. Guestrin, “"Why Should I Trust You?",” in *the 22nd ACM SIGKDD International Conference*. New York, New York, USA: ACM Press, 2016, pp. 1135–1144.
- [190] M. Ribeiro, K. Grolinger, and M. A. M. Capretz, “MLaaS: Machine Learning as a Service,” in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. IEEE, Dec. 2015, pp. 896–902.

- [191] 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.
- [192] G. Ridgeway, D. Madigan, T. Richardson, and J. O’Kane, “Interpretable Boosted Naïve Bayes Classification.” *KDD*, 1998.
- [193] G. Ritzer and E. Guba, “The Paradigm Dialog,” *Canadian Journal of Sociology / Cahiers canadiens de sociologie*, vol. 16, no. 4, p. 446, 1991.
- [194] S. P. Robbins and T. Judge, *Essentials of organizational behavior*. Pearson,, 2014.
- [195] M. P. Robillard, “What makes APIs hard to learn? Answers from developers,” *IEEE Software*, vol. 26, no. 6, pp. 27–34, 2009.
- [196] M. P. Robillard and R. Deline, “A field study of API learning obstacles,” *Empirical Software Engineering*, vol. 16, no. 6, pp. 703–732, 2011.
- [197] H. Robinson, J. Segal, and H. Sharp, “Ethnographically-informed empirical studies of software practice,” *Information and Software Technology*, vol. 49, no. 6, pp. 540–551, 2007.
- [198] L. Rokach and O. Z. Maimon, *Data mining with decision trees: theory and applications*. World scientific, 2008, vol. 69.
- [199] C. Rosen and E. Shihab, “What are mobile developers asking about? a large scale study using stack overflow,” *Empirical Software Engineering*, vol. 21, no. 3, pp. 1192–1223, 2016.
- [200] W. Rosenberry, D. Kenney, and G. Fisher, *Understanding DCE*. O'Reilly & Associates, Inc., 1992.
- [201] A. Rosenfeld, R. Zemel, and J. K. Tsotsos, “The Elephant in the Room.” *CoRR*, 2018.
- [202] A. S. Ross, M. C. Hughes, and F. Doshi-Velez, “Right for the Right Reasons: Training Differentiable Models by Constraining their Explanations,” *arXiv.org*, Mar. 2017.
- [203] R. J. Rubey and R. D. Hartwick, “Quantitative measurement of program quality,” in *Proceedings of the 1968 23rd ACM national conference*. New York, New York, USA: ACM, 1968, pp. 671–677.

- [204] J. Rutten, M. Kwiatkowska, G. Norman, and D. Parker, *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems*, P. Panangaden and F. van Breugel (eds.), ser. CRM Monograph Series. American Mathematical Society, 2004, vol. 23.
- [205] H. W. Schmidt, I. Crnkovic, G. T. Heineman, and J. A. Stafford, Eds., *A Framework for Contract-Based Collaborative Verification and Validation of Web Services*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.
- [206] M. Schwabacher, P. Langley, and P. Norvig, “Discovering communicable scientific knowledge from spatio-temporal data,” *ICML*, 2001.
- [207] C. B. Seaman, “Qualitative methods,” in *Guide to Advanced Empirical Software Engineering*. Springer Science & Business Media, Nov. 2007, pp. 35–62.
- [208] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization,” in *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 618–626.
- [209] S. Sen and L. Knight, “A genetic prototype learner,” in *IJCAI*. Citeseer, 1995, pp. 725–733.
- [210] C. E. Shannon and W. Weaver, *The mathematical theory of communication*. Urbana, IL: The University of Illinois Press, 1963.
- [211] M. Shaw, “Writing good software engineering research papers,” in *25th International Conference on Software Engineering, 2003. Proceedings*. IEEE, 2003, pp. 726–736.
- [212] Shull, Forrest, Singer, Janice, and Sjøberg, Dag I K, *Guide to Advanced Empirical Software Engineering*, F. Shull, J. Singer, and D. I. K. Sjøberg, Eds. Springer Science & Business Media, Nov. 2007.
- [213] H. A. Simon, *The sciences of the artificial*. MIT press, 1996.
- [214] J. Singer, S. E. Sim, and T. C. Lethbridge, “Software engineering data collection for field studies,” in *Guide to Advanced Empirical Software Engineering*. Springer Science & Business Media, Nov. 2007, pp. 9–34.
- [215] S. Singh, M. T. Ribeiro, and C. Guestrin, “Programs as Black-Box Explanations,” *arXiv.org*, Nov. 2016.

- [216] V. S. Sinha, S. Mani, and M. Gupta, “Exploring activeness of users in QA forums,” in *Proceedings of the 10th Working Conference on Mining Software Repositories*. IEEE Press, 2013, pp. 77–80.
- [217] M. Sokolova and G. Lapalme, “A systematic analysis of performance measures for classification tasks,” *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, 2009.
- [218] I. Sommerville, *Software Engineering*, 9th ed. Addison-Wesley, 2011.
- [219] P. E. Spector, *Summated rating scale construction: An introduction*. Sage, 1992.
- [220] J. Su, D. V. Vargas, and K. Sakurai, “One pixel attack for fooling deep neural networks.” *CoRR*, 2017.
- [221] G. H. Subramanian, J. Nosek, S. P. Raghunathan, and S. S. Kanitkar, “A comparison of the decision table and tree,” *Communications of the ACM*, vol. 35, no. 1, pp. 89–94, 1992.
- [222] S. Subramanian, L. Inozemtseva, and R. Holmes, “Live API documentation,” in *the 36th International Conference*. New York, New York, USA: ACM Press, 2014, pp. 643–652.
- [223] M. Sugiyama, N. D. Lawrence, and A. Schwaighofer, *Dataset shift in machine learning*. The MIT Press, 2017.
- [224] N. R. Suri, V. S. Srinivas, and M. N. Murty, “A cooperative game theoretic approach to prototype selection,” in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2007, pp. 556–564.
- [225] D. Szafron, P. Lu, R. Greiner, D. S. Wishart, B. Poulin, R. Eisner, Z. Lu, J. Anvik, C. Macdonell, and A. Fyshe, “Proteome Analyst: custom predictions with explanations in a web-based tool for high-throughput proteome annotations,” *Nucleic acids research*, vol. 32, no. 2, pp. W365–W371, 2004.
- [226] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” *arXiv preprint arXiv:1312.6199*, 2013.
- [227] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision.” *CVPR*, 2016.

- [228] M. B. W. Tabor, “**Student Proves That S.A.T. Can Be: (D) Wrong.**,” *New York Times*, Feb. 1997.
- [229] A. Tahir, A. Yamashita, S. Licorish, J. Dietrich, and S. Counsell, “Can you tell me if it smells?” in *the 22nd International Conference*. New York, New York, USA: ACM Press, 2018, pp. 68–78.
- [230] G. Tassey, *The Economic Impacts of Inadequate Infrastructure for Software Testing*. National Institute of Standards and Technology, Sep. 2002.
- [231] R. S. Taylor, “Question-negotiation and information-seeking in libraries (Vol. 29): College and Research Libraries,” 1968.
- [232] S. Thrun, “Is Learning The n-th Thing Any Easier Than Learning The First?” p. 7, 1996.
- [233] A. Van Assche and H. Blockeel, “Seeing the forest through the trees: Learning a comprehensible model from an ensemble,” in *European conference on machine learning*. Springer, 2007, pp. 418–429.
- [234] B. Venners, “Design by Contract: A Conversation with Bertrand Meyer,” *Artima Developer*, 2003.
- [235] W. Verbeke, D. Martens, C. Mues, and B. Baesens, “Building comprehensible customer churn prediction models with advanced rule induction techniques,” *Expert Systems with Applications*, vol. 38, no. 3, pp. 2354–2364, 2011.
- [236] S. Wachter, B. Mittelstadt, and L. Floridi, “Why a Right to Explanation of Automated Decision-Making Does Not Exist in the General Data Protection Regulation,” *International Data Privacy Law*, vol. 7, no. 2, pp. 76–99, Jun. 2017.
- [237] B. Wang, Y. Yao, B. Viswanath, H. Zheng, and B. Y. Zhao, “With Great Training Comes Great Vulnerability - Practical Attacks against Transfer Learning.” *USENIX Security Symposium*, 2018.
- [238] 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*. ACM, 2013, pp. 1019–1024.
- [239] S. Weerawarana, *Web Services Platform Architecture*, ser. SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More. Prentice-Hall PTR, 2005.

- [240] 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.
- [241] H. Wickham, “A Layered Grammar of Graphics,” *Journal of Computational and Graphical Statistics*, vol. 19, no. 1, pp. 3–28, Jan. 2010.
- [242] R. J. Wieringa and J. M. 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.
- [243] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.
- [244] 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 2014.
- [245] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.
- [246] M. L. Wong and K. S. Leung, *Data mining using grammar based genetic programming and applications*. Springer Science & Business Media, 2006, vol. 3.
- [247] X. Yi and K. J. Kochut, “A cp-nets-based design and verification framework for web services composition,” in *Web Services, 2004. Proceedings. IEEE International Conference on*. IEEE, 2004, pp. 756–760.
- [248] R. K. Yin, *Case study research and applications: Design and methods*. Sage publications, 2017.
- [249] J. Zahálka and F. Železný, “An experimental test of Occam’s razor in classification,” *Machine Learning*, vol. 82, no. 3, pp. 475–481, 2011.
- [250] J. Zhang and R. Kasturi, “Extraction of Text Objects in Video Documents: Recent Progress,” in *2008 The Eighth IAPR International Workshop on Document Analysis Systems (DAS)*. IEEE, 2008, pp. 5–17.
- [251] 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.

- [252] M. zur Muehlen, J. V. Nickerson, and K. D. Swenson, “Developing web services choreography standards—the case of REST vs. SOAP,” *Decision Support Systems*, vol. 40, no. 1, pp. 9–29, Jul. 2005.

# **Appendix A**

## **Additional Material**



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

The development of web 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 [200]. This framework used the synchronous communication paradigm Remote Procedure Calls (RPCs) first introduced by Nelson [166] that allows procedures to be called in a remote address space as if it were local. Its communication paradigm, DCE/RPC [169], 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 [169]). 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) [57] 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 (SOA) whereby the consumer and provider agree upon a set level of service, which in some cases are legally binding [38]. 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 [110, 239]). These attributes are included within SOAP headers; thus, QoS aspects are independent from the transport layer and instead exist at the application layer [176]. 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 [252, 176] 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



**Figure A.1:** Worldwide search interest for SOAP (blue) and REST (red) since 2004.  
Source: [5]

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).

**Listing A.1:** A SOAP HTTP POST consumer request to retrieve customer record #43456 from a web service provider. Source: [35].

```

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: [35].

```

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 [176]. That is, the HTTP is considered as a transport protocol solely. In 2000, Fielding [86] 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 [41].
- **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) [150]). Therefore, no RESTful service uses a standardised response document or invocation syntax. While there are proposals, such as WADL [99], 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.

---

<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.

**Listing A.3:** An equivalent HTTP consumer request to that of Listing A.1, but using REST. Source: [35].

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

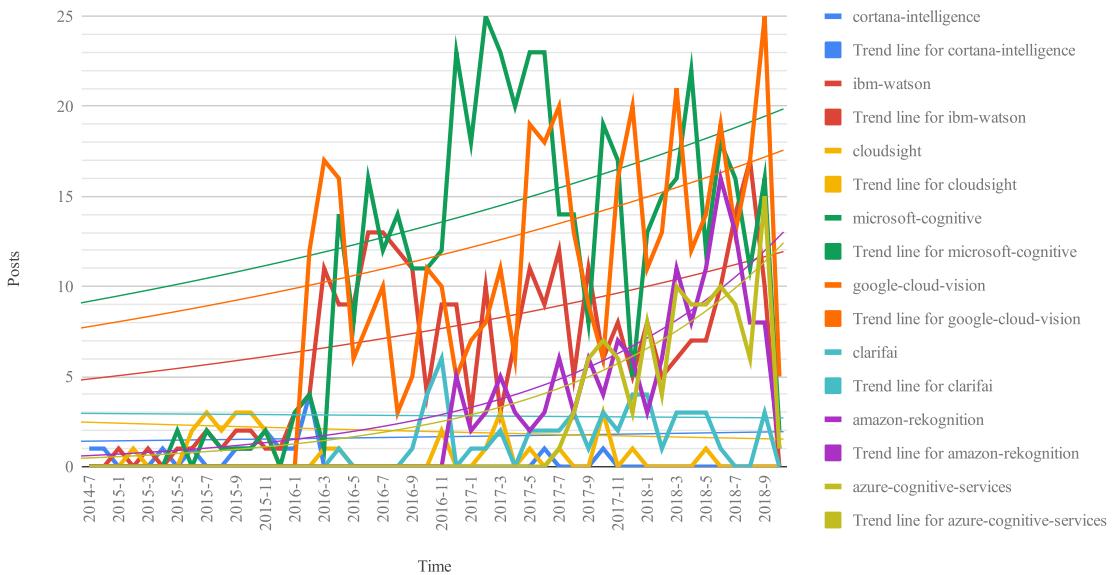
**Listing A.4:** The REST HTTP service provider response for Listing A.3. Source: [35].

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

**Figure A.2:** A Broad Range of AI-Based Products And Services Is Already Visible. (From [149].)

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: Synthesis</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 these intelligent cloud services.



**Figure A.4:** Questions asked by software engineering researchers (column 2) that can be answered by field study techniques. (From [214].)

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**

### **Technical Report Manuscript**



# Losing trust: Are computer vision APIs are insufficient for developers?

Alex Cummaudo<sup>1</sup>, Rajesh Vasa<sup>1</sup>, John Grundy<sup>2</sup>

<sup>1</sup> Applied Artificial Intelligence Institute, Deakin University, Geelong, Australia

<sup>2</sup> Faculty of Information Technology, Monash University, Clayton, Australia

{ ca, rajesh.vasa }@deakin.edu.au, john.grundy@monash.edu

**Abstract**—Recent advances in computer vision are available as cloud APIs and their accessibility and simplicity is compelling. Multiple vendors now offer such software as a service and developers want to leverage these advances to provide value to end-users. However different services provide different performance and hence there is a need to make an evaluation of disparate services before selection. There currently exists no evaluation framework in assisting developers to make this choice, and the services’ existing documentation and usage patterns hide information needed to make informed decisions. We evaluated the responses of three different computer vision cloud APIs over time using a static input dataset, verifying responses against the respective documentation. We found that there are: (1) inconsistencies in the vocabulary labels of these responses; (2) little ontology to describe the relationships between such labels; (3) a lack of definitions in terminologies used to document the APIs; (4) a lack of communication protocols to identify changes in responses over time. We propose several recommendations to both developers and API providers to improve the evaluation guidelines of such APIs.

## I. INTRODUCTION

The availability of Machine Learning as a Service (MLaaS) [1] has made AI tooling accessible to software developers and the entry barrier is lowered in both time and skill. Training AI-based computer vision analysers requires manually curating a dataset and writing a deep-learning classifier from scratch; laborious in time and requires machine-learning expertise. Contrast this to an MLaaS-powered computer vision analyser (e.g., [2–11]); the process is abstracted behind an API call, only requiring knowledge on how to use a RESTful architecture [cite:Restful thesis]. The ubiquity of MLaaS is exemplified through evermore growing applications that use these APIs: aiding the vision-impaired [12, 13], accounting [14], data analytics [15], and student education [16].

APIs reflect a set of design choices made by their providers. Evaluations of API usability show that developers must be wary of the accuracy and completeness of API documentation [17] written by providers, while providers should consider mismatches between the developer’s conceptual knowledge of the API its implementation [18]. It is therefore imperative that developers of MLaaS APIs consider the impact of API usability.

Moreover, developers need to be wary of unintended side effects. Computer vision techniques may involve geometric manipulation or image transformation, or encapsulate deep-

learning strategies or stochastic methods.<sup>1</sup> Where the latter is used, these APIs become inherently non-deterministic in nature, but developers are still taught with a deterministic mindset. Simple arithmetic representations (e.g.,  $2 + 2 = 4$ ) will *always* result in 4; but a multi-layer perceptron neural network performing similar arithmetic representation [19] gives a probability<sup>2</sup> where the target output (exactly 4) and the output inferred (possibly 4) matches as a percentage (or as an error where it does not match). That is, instead of an exact output, there is instead a *stochastic* result:  $2 + 2$  *may* equal 4 with a confidence of  $n$ .

Is such information sufficiently conveyed to developers? If MLaaS APIs are to be used in production services, do they follow rigours software engineering practices [20, 21] to communicate and test this? It is agreed that software developed with rigour undergoes quality assessment using a Software Quality Assurance (SQA) framework [22]. Developers, therefore, follow such frameworks to mitigate risk impacting internal and external quality; vital for safety-critical systems. Such quality frameworks advocate best practices for reviews and audits, defect detection and analysis, standardisation of usage-patterns, change logs and usage of consistent vocabulary. Do such practices apply to computer vision APIs?

To our knowledge, few studies have been conducted to evaluate this claim. This study assesses the understanding, documentation, and variability of these APIs by comparing three publicly available computer vision APIs provided by prominent vendors to identify recommendations for improvement. We aide this paper with a motivating example in section II, discussing related work and our methodology in sections III and IV. Our findings are discussed in section V, from which we draw several recommendations to both developers and API providers in section VI.

## II. MOTIVATING EXAMPLE

Consider a software developer named Pam. Pam wants to develop a social media photo-sharing mobile app that analyses her and her friends photos. Pam wants the app to categorise photos into scenes (e.g., day vs. night, landscape vs. indoors),

<sup>1</sup>We observe that Google Cloud Vision, for example, markets itself as using “powerful machine learning models” to perform image analysis.

<sup>2</sup>Varied terminology used here. Probability, confidence, accuracy and score can all be used interchangeably here.

generate brief descriptions of each photo, and catalogue photos of her friends as well as common objects (e.g., all photos with a dog, all photos with on the beach).

Rather than building a computer vision engine from scratch, Pam thinks she can achieve this using one of APIs A, B or C. However, Pam comes from a typical software engineering background and has insufficient knowledge of key computer vision terminology and no understanding of the processes behind deep-learning. She ultimately believes all are APIs alike and internalises a deterministic mindset of them; when she decides on one of the three APIs, she expects a static result always, just as she expects  $2 + 2$  to always be 4.

To make an assessment of these APIs, she tries her best to read through the documentation of some computer vision APIs, but she has no guiding framework to help her choose the right one. Many questions come to mind:

- What does confidence mean?
- Will she need a combination of many computer vision APIs to solve this task?
- How does she know when there is a defect in the response?
- How does she know what labels the API can pick up, and what labels it can't?
- How does it describe her photos and detect the faces?
- How can she interpret the results if she disagrees with it to help improve her app?

Dazzled by this, she does some brief reading on Wikipedia but is confused by the immense technical detail to take in. She would like some form of guiding framework to assist her.

### III. RELATED WORK

If we were to view computer vision software through the lenses of an SQA framework, robustness often features as a quality attribute in myriad software quality models (e.g., [23–28]). Such models attempt to standardise these quality attributes against measurable metrics [29]. (We refer to the well-documented literature evaluating these models [30–33].) It is well-known that software quality exists in the evaluation of the end-product (external quality) and/or the assurances in the development processes (internal quality) [20], and thus we summate AI quality literature from both these viewpoints in the following two subsections.

#### A. External Quality: Robustness

A majority of recent work investigates the robustness of the deep-learning within computer vision technique implementation, thereby assessing the quality of the end-product. This is typically achieved using adversarial examples [34], where input images are slightly perturbed to maximise prediction error but are still interpretable to humans.

Google Cloud Vision, for instance, fails to correctly classify adversarial examples when noise is added to the original images [35]. Rosenfeld et al. [36] illustrated that inserting synthetic foreign objects to input images (e.g., a cartoon elephant) can completely alter classification output. Wang et al. [37] performed similar attacks on a transfer-learning approach

of facial recognition by modifying pixels of a celebrity's face to be recognised as a completely different celebrity, all while still retaining the same human-interpretable original celebrity. Su et al. [38] used the ImageNet database to show that 41.22% of images drop in confidence when just a *single pixel* is changed in the input image; and similarly, Eykholt et al. [39] recently showed similar results that made a CNN interpret a stop road-sign (with mimiccid graffiti) as a 45mph speed limit sign.

Thus, the state-of-the-art computer vision techniques may not be robust enough for safety critical applications (such as self-driving cars) as they do not handle intentional or unintentional adversarial attacks. Moreover, as such adversarial examples exist in the physical world [40, 41], “the real world may be adversarial enough” [42] to fool AI software.

#### B. Internal Quality: API and Services

From the *developer's* perspective, little has been achieved to assess API quality or assure quality of these computer vision services. APIs are the interface between developers' needs and the software components [43]; therefore, assessing such computer vision services from the quality of their APIs is thereby directly related to the development quality [29]. Good APIs should be intuitive and require less documentation browsing [17], thereby increasing productivity. Conversely, poor APIs that are hard to understand and work with reduce developer productivity, reducing product quality. The consequences of this have shown a higher demand of technical support (as measured in [44]) that, ultimately, causes the maintenance to be far more expensive, a phenomenon widely known in software engineering economics [45–47].

Ko and Riche [18] assessed the importance of a programmer's conceptual understanding of the background behind the task before implementing the task itself. The study demonstrated how developers had little confidence in their own metacognitive judgements to understand and assess the feasibility of using a particular Bluetooth API to implement a task that requires Bluetooth. While code examples helped (as suggested by the early attempts of API usability [29]), they failed to understand the vocabulary and concepts of wireless connectivity. This indecision over what search results were relevant in their searches ultimately hindered their progress implementing the functionality, again decreasing productivity. Ko and Riche suggest to improve API usability by introducing the background of the API and its relevant concepts using glossaries linked to tutorials to each of the major concepts, and then relate it back to how to implement the particular functionality.

Lastly, computer vision services are based on cloud computing fundamentals under a subset of the Platform as a Service (PaaS) model [48]. There has been work in the evaluation of such services in terms of quality attributes [49–54]: these attributes are exposed using Service Level Agreements (SLA) between vendors and customers, and customers denote their demanded Quality of Service (QoS) to ensure the cloud ser-

vices adhere to measurable KPI metrics [55]. Again, however, such metrics are not available in computer vision services.

#### IV. METHOD

This study organically evolved by observing phenomena surrounding computer vision API internal quality, chiefly their documentation and responses. We adopted a mixed methods approach, performing both qualitative and quantitative data collection on these two key aspects by using documentary research methods for inspecting the API documentation and structured observations to quantitatively analyse the results over time. This, ultimately, helped us shape four key research questions which this paper addresses:

- RQ1.** How well do computer vision API providers *conceptually describe* their API documentation in the techniques and terminology?
- RQ2.** Given RQ1, how are these techniques and terminologies expressed in the application guidelines they provide?
- RQ3.** How do responses differ between various API providers given a single source dataset?
- RQ4.** Given RQ2, how do the responses differ over time?

We conducted two experiments to address these questions against three popular computer vision services: AWS Rekognition [4], Google Cloud Vision [2], Azure Computer Vision [3]. These particular cloud APIs were chosen given the ubiquity of the hosting cloud platforms (AWS, Google Cloud and Azure), although we acknowledge similar services [5–11] also exist. For the remainder of this paper, we (i) refer to the API creator as *providers* and API users (developers) as *consumers* and (ii) de-identify our selected APIs by labelling them as APIs A, B and C but do not reveal mapping to prevent any implicit bias. We describe both experiments below.

##### A. Experiment I: API Documentation Evaluation

We observed how providers *conceptually* communicate computer vision techniques to consumers by reviewing the documentation in each of the provider's websites. We categorised varying computer vision techniques offered by the three providers and made a union of all overlaps using a consistent vocabulary (Table III). We note that we only use providers' home websites [2–4] as our primary source in order to assess the 'official' API documentation, and not from third party sources such as StackOverflow. From this, we make a qualitative interpretation on (1) the terminologies by which these techniques are communicated (Section V-A1) and (2) the way these terms are used in the guidelines of the APIs (Section V-A2), reviewing disparity between the terms and the suggestions made on their guidelines.

The second aspect to this experiment was to focus on the disparity of the terminology on ML 'confidence' (Section V-A3). Therefore, we evaluated how this term is used in SDK and API documentation. In addition to searching through official provider documentation, we captured any empirical evidence of the developer community having any confusion

on mixed terminology by searching StackOverflow and Github issues. Both StackOverflow and GitHub issues were chosen as they provide insight into the discussion within the software engineering community. We made a StackOverflow search using text and tag matching:

```

1 | confidence score accuracy api
2 | [api] or [image] or [documentation] or
3 | [google-cloud-vision] or
4 | [google-cloud-platform] or
5 | [microsoft-cognitive] or
6 | [azure-cognitive-services] or
7 | [amazon-rekognition] or
8 | [amazon-web-services] or [vision-api]
```

Similarly, we performed a similar match using equivalent GitHub syntax. These searches were conducted on August 16, 2018.

##### B. Experiment II: API Response Evaluation

*< TODO: JG: need to do more images / get a bigger set of outcomes to compare - is there some 'standard' benchmark for testing image APIs??? > < TODO: AC: @JG yes there is but they are not for testing image APIs; they are typically used for testing the AI. >*

This second experiment involved sending a test image (of a Border Collie<sup>3</sup>) to all three services to evaluate the consistency of the responses between providers. This is highlighted in table I.

To assess evolution risk, we ran a five month experiment (April to August 2018) where we sent 30 images<sup>4</sup> on nine occasions. In total, we made 840 requests with a matching number of responses. For each response, we recorded the timestamp, the description and URL of the source image, the labels the service identified in those images and the accuracy of those labels. A total of 9576 labels and accuracies were identified.

We then performed quantitative analysis on each response by calculating the maximum confidences of each label over the period (that is, to get the labels of each image that were of the highest confidence consistently over the period). We also made note of the confidence changes over time by calculating the delta between the highest confidence value and the lowest confidence value per label, image and service. We selected the top three labels for each image over each service. These are aggregated in table II.

*< TODO: JG: User Evaluation - do experiment with users of the APIs to get feedback to triangulate with the other findings.. > < TODO: AC: @JG; I agree, but would need to get ethics for this to do so. >*

## V. FINDINGS

##### A. Experiment I: API Documentation Evaluation

<sup>3</sup>Image of the Border Collie sourced from: <https://www.akc.org/dog-breeds/border-collie/>.

<sup>4</sup>For reproducibility, the original photos are available at: <https://github.com/alexcu/cognitive-api-confidence-logger/tree/master/images>

1) *Techniques Analysis*: Of all the services provided, we note that API A offers the most computer vision techniques (as tabulated in table III).

We note the disparity between the terms ‘detection’, ‘recognition’, ‘localisation’ and ‘analysis’. This applies chiefly to object- and facial-related techniques. Detection applies to facial detection, which gives bounding box coordinates around all faces in an image. Similarly, localisation applies the same methodology to disparate objects in an image and labels them. However, table III shows that localisation technique only exists in API A (in beta); all services provide object ‘recognition’, whereby a single primary object in labelled in an image. In the context of facial ‘recognition’, this term implies that a face is *recognised* against a known set of faces. Lastly, ‘analysis’ applies in the context of facial analysis (gender, eye colour, expression etc.); there does not exist a similar analysis technique on objects.

Thus, the terms ‘localisation’, ‘detection’, ‘recognition’ and ‘analysis’ are largely disparate for objects versus faces. It can be quite easy for one to make a quick confusion on the terms, thereby miscommunicating conceptual understanding to technical understanding. For example, Pam may easily confuse object ‘recognition’ as a means to locate the bounding boxes of all different objects in an image, but this intended functionality from the developer only exists under object ‘localisation’.

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

2) *Safety-Critical Guidelines*: Of all three services, API C provides the most extensive advice on using their service for safety-critical applications. As their service is the only one of the three assessed that allows for facial recognition within a collection of faces (see section V-A1), they list several guidelines in how to use their software to ensure quality in the wider usage of the application. These guidelines can be categorised into three risk categories dependent on the type of application using the service:

- **high risk**: for high risk applications (e.g., crime detection), do not to use facial recognition software to make autonomous decisions,
- **medium risk**: for medium risk applications (e.g., building authentication), to ensure a facial similarity match threshold of at least 99%,
- **low risk**: for low risk applications (e.g., finding missing persons), a benefit of using a similarity match threshold lower than 99%.

API A obfuscates restrictions in its legal Terms of Service that developers “will not... use the Services for High Risk Activities<sup>5</sup>... [and] the services [are not guaranteed to be] error-free or interrupted. Neither the software nor the services are designed, manufactured, or intended for high risk activities.” [cite:Link to <https://cloud.google.com/terms/>]. API B does not make mention of high risk applications in its documentation.

3) *Inconsistency of the term ‘confidence’*: Self-redundancy in the term ‘confidence’ is tautologic across all three providers. Below, we detail references from the primary sources in addition to secondary sources found in StackOverflow and GitHub issues.

a) *API A*: API A provides a tautologic definition of ‘confidence’, and instead uses the term ‘score’ instead for the same concept. Usage in the how-to documentation [56] describes this as:

*“Score is the confidence score, which ranges from 0 (no confidence) to 1 (very high confidence).”*

Once again, inspecting the .NET SDK documentation [57] shows a similar definition for ‘confidence’:

*“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].”*

Note that the definition of the new ‘score’ term is, put bluntly, defined as the “overall score of the result. Range [0, 1]” [57].

b) *API B*: API B defines confidence in its how-to guide [58] as:

*“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.”*

Similarly, the Java SDK documentation [59] provides a similar lack of definition:

*“The level of confidence the service has in the caption.”*

c) *API C*: In the documentation of the DetectLabels operation [60] (used to detect ‘labels’ in an image) the following description is provided for confidence via an example:

*“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.”*

However, a tautologic definition is provided within the “Detecting Objects and Scenes” guide [61] to define what is meant by ‘confidence’ by a-likening it to a ‘percentage score’:

<sup>5</sup>Where ‘High Risk Activities’ is defined as: “the use or failure of the Services could lead to death, personal injury, or environmental damage.”

*[API C] also provide[s] a percentage score for how much confidence [API C] has in the accuracy of each detected label.”*

Thus, in both these examples, the term ‘confidence’ is still not well described; what is meant by ‘confidence’ and interchanging the term ‘confidence’ with ‘percentage score’ may be confusing to developers. Investigating this with respect to API C’s JavaDoc [62] or SDK documentation [63], the definition of ‘confidence’ again shows no further insight of the term:

*“Level of confidence. Type: Float Valid Range: Minimum value of 0. Maximum value of 100.”*

*d) Other Sources:* In addition to the above documentation lacking precise definitions of ‘confidence’, we found users on StackOverflow find this lack of definition confusing. Using our search described in section IV, five of a total of eight search results indicate issues from users. Two of these issues relate to similar AI-based services on API A, with the rest on API B-based AI services. We summarise these issues below:

- [64]: Confusion on API A that the ‘score’ and ‘topicality’ values are always the same. User was also confused by meaning of ‘score’.
- [65]: Unsure what the default thresholds were for facial recognition in API B.
- [66]: User unsure how to interpret ‘score’ for a related API of API B.
- [67]: User unsure what web entity ‘score’ meant for API A web search.
- [68]: User confused on API B facial recognition if two confidences equal to 1 meant that faces were exact match.

Additionally, we found similar confusion within GitHub issues of API A and API B, where developers are still unsure of best practices and calculation of ‘score’:

- [69]: In an issue thread on GitHub for API A, developers were unsure what is meant from the ‘score’ property and how it was calculated.
- [70]: In an issue thread on GitHub for API B, developers are unsure what best practices give optimal results in an image.

## B. Experiment II: API Response Evaluation

1) *Vocabulary of Labels:* Upon reading through all the documentation of each provider, we note that boundary sets of what labels the computer vision can categorise and what it cannot. The disparity of this is emphasised within table II, where a subset of our images (provided in fig. 2) have completely disparate results labelled for the one conceptual image. We reinforce this in table I: the same image of a Border Collie was provided to all three services. The only consistency in the services was ‘Dog’ for all three APIs, and ‘Animal’ and ‘Mammal’ for APIs B and C. ‘Border Collie’, ‘Dog’ and ‘Collie’ are the most conceptually-specific labels across all three services for APIs.

Thus, taxonomy of specificity is unknown; if ‘Border Collie’ is detected as a dog breed, does this imply the hypernym ‘Dog’

TABLE I: First six responses of image analysis for a Border Collie sent to each API’s demonstration website. The specificity (to 3 s.f.) and vocabulary of each label in the response varies between all services.

Label	A	B	C
Dog	0.990	0.999	0.992
Dog Like Mammal	0.960		
Dog Breed	0.940		
Border Collie	0.850		
Dog Breed Group	0.810		
Carnivoran	0.810		
Black		0.992	
Indoor		0.965	
Standing		0.792	
Animal		0.932	0.992
Canine			0.992
Collie			0.992
Mammal		0.929	0.992
Pet			0.992

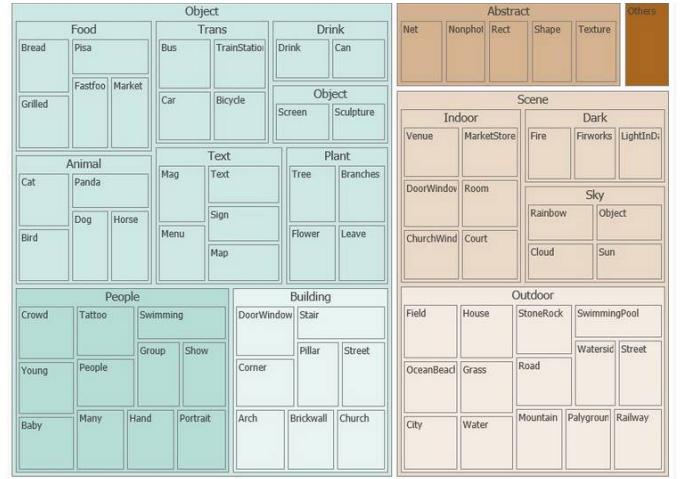


Fig. 1: Visualisation of Azure’s 86-category [71]. The taxonomy provides a layer of specificity to objects. Users can search within a domain of a specific category if the object they are analysing is within a known domain.

is detected, and then ‘Mammal’, then ‘Animal’, then ‘Object’? Only API B provides a taxonomy for capturing what level of scope is desired, providing what it calls the ‘86-category’ concept as found in its how-to guide:

*“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” [71].*

Figure 1 shows an illustration of the breakdown of this taxonomy.

2) *Confidence Variability:* We identified that the confidence of labels are variable without notification by negligible amounts in both API A and API C. API B’s models did not

TABLE II: First three responses comparing all labels against all APIs classified from the same source images identified in Figure 2. Note that API B’s response for Image 5 only returned two responses, hence the missing third response.

Image #	API A			API B			API C		
	Resp. 1	Resp. 2	Resp. 3	Resp. 1	Resp. 2	Resp. 3	Resp. 1	Resp. 2	Resp. 3
1	motor vehicle	car	vehicle	car	tree	outdoor	parking	parking lot	windshield
2	black	pink	mammal	indoors	bed	person	slide	toy	adorable
3	plant	walkway	flowerpot	outdoor	plant	stone	flora	jar	plant
4	countertop	kitchen	room	indoors	wall	floor	indoors	interior design	kitchen
5	vehicle	asphalt	automotive exterior	outdoor	ground	N/A	sunlight	flora	forest
6	wall	metal	public utility	building	ground	outdoor	rust	brick	face
7	road	lane	asphalt	tree	road	sky	freeway	road	building
8	rock	grass	walkway	ground	outdoor	sidewalk	rock	asphalt	tarmac
9	car	transport	town	outdoor	road	street	canopy	umbrella	automobile

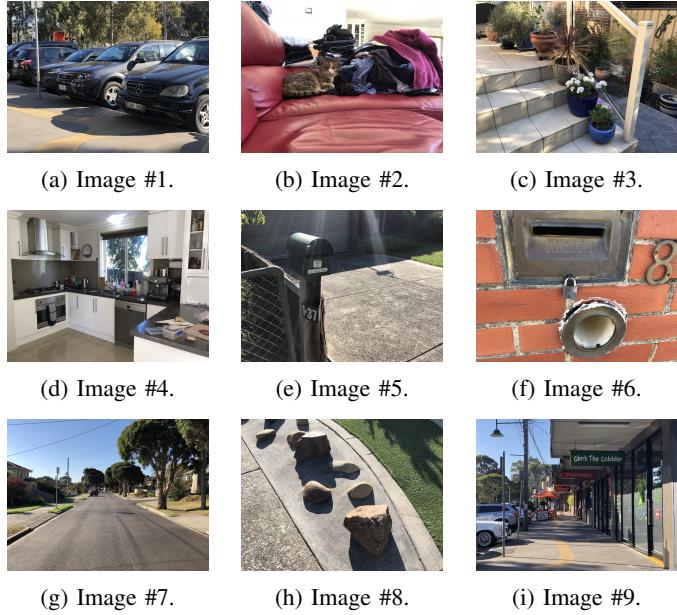


Fig. 2: Images used for vocabulary of labels, as identified in Table 1.

change during our assessment period. This, in turn, causes some variability in the label thresholds that are returned. It should be noted that some labels drop entirely.

Upon investigation of any change log to communicate for these confidence changes, the only ‘versioning’ found was schema differences in the response provided.

⟨ **TODO: JG: The above would really benefit from more examples tried out...** ⟩ ⟨ **TODO: AC: @JG yes I agree.** ⟩

## VI. DISCUSSION

### A. Recommendations for API Providers

1) *Standardise Techniques Vocabulary:* There lacks a standardised vocabulary of both describing these APIs and the labels in their responses. As found in previous studies [72], associating API features to their correct names when the naming is inconsistent can be difficult when the API is unfamiliar. Pam has no idea what ‘tagging’ versus ‘labelling’

means and thus may be confused by these unfamiliar APIs terms to describe what is conceptually the same technique. We suggest providers to shift away from using disparate terms for the same concept (e.g., ‘image tagging’ versus ‘image categorisation’ versus ‘image labelling’) and a taxonomy of these terms should be developed. Information may also be better communicated through meta-models to developers, and would be conceptually easier to digest when the terminology remains consistent.

2) *Make labels more transparent:* Pam does not know that API B can only provide back ‘Dog’ for the same image that API A provides back with ‘Border Collie’; how can she toggle the level of specificity in her results? She also is frustrated that API C does detect the Border Collie, but labels this as just ‘Collie’: how is she meant to implement such pattern matching in a generic sense? Thus, we recommend vendors to improve the documentation of APIs by making known publicly available the boundary set of the training data used for the algorithms. This way, developers would be able to review the API’s specificity for their intended use case (e.g., maybe Pam is satisfied her app can catalogue ‘dogs’ together, and in fact does not want specific dogs breeds catalogued). We also recommend that vendors publish usage guidelines should that include details of priors and how to evaluate the specific API results. This could also be included as metadata in the response object.

3) *Improve Metadata in Response:* Much of the information in these services is reduced to a single confidence value within the response object, and the details about training data and the internal AI architecture remains unknown; little metadata provided back to developers that encompass such detail. Furthermore, inconsistencies in the vocabulary used in the documentation around the AI makes using them even more conceptually challenging to developers. Providing as much metadata in the response object may resolve this.

4) *Improve Versioning:* Lastly, we recommend introducing a versioning system so that a model can be used from a specific date in production systems: when Pam tests her app today, she would like the API to remain the same for when her app is deployed in production tomorrow. Thus, in a request made to the vendor, Pam could specify what date she ran her app’s

QA testing on so that she knows that henceforth these model changes will not affect her app.

### B. Recommendations for Developers

1) *Be wary of API inconsistency:* There is inherent inconsistency between these APIs: if evaluating these APIs, be aware that there is no standardised vocabulary in the labels they return. Therefore, developers need to stick with a specific vendor they feel that is right for them and not stray away. Issues such as the ‘Border Collie’ vs. ‘Collie’ terms can therefore be avoided.

2) *Use case should inform API choice:* Developers must also be decisive in which API is right for their use case. There is no ‘one-size-fits-all’ approach to these visualisation services, and therefore inherit risks are apparent. For example, API C is able to recognise faces unlike the other services, but they do advise humans to be the final decision-makers in making decisions from this data.

3) *Avoid use in safety-critical systems:* Lastly, these computer vision services are in a nascent stage; we recommend developers to avoid their use in safety critical systems due to their lack of stability and visibility of changes. To ensure results are correct over time, we recommend developers to create a representative dataset of the intended dataset and evaluate these changes against the API frequently. This will help identify when changes, if any, have occurred when vendors do not provide a line of communication when this occurs.

## VII. CONCLUSIONS

We analysed three popular computer vision APIs for internal quality in a software engineering perspective. We identify that the definition of the term ‘confidence’ is an issue by the tautologic nature of its definition by the API providers, and by evidence supplied from developer community. We also identify that these services use inconsistent vocabulary both in describing their APIs and in the responses they give. Moreover, no training set provided to users to see what the algorithms were trained on, and thus developers remain unaware of what labels are and are not recognisable. We also found that APIs A and C change their models over time without any notification to developers: thus, we have compiled several recommendations for both developers consuming these APIs and the vendors who provide them.

While we have investigated this area in the area of computer vision APIs, we suspect similar patterns may occur in the documentation of other MLaaS-based services, such natural language processing or audio transcription. We remain this open for future work.

## REFERENCES

- [1] M. Ribeiro, K. Grolinger, and M. A. M. Capretz, “MLaaS: Machine Learning as a Service,” in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. IEEE, Dec. 2015, pp. 896–902.
- [2] “Vision API - Image Content Analysis — Cloud Vision API — Google Cloud,” <https://cloud.google.com/vision/>, accessed: 13 September 2018.
- [3] “Image Processing with the Computer Vision API — Microsoft Azure,” <https://azure.microsoft.com/en-au/services/cognitive-services/computer-vision/>, accessed: 13 September 2018.
- [4] “Amazon Rekognition,” <https://aws.amazon.com/rekognition>, accessed: 13 September 2018.
- [5] “Detecting labels in an image,” <https://docs.aws.amazon.com/rekognition/latest/dg/labels-detect-labels-image.html>, accessed: 13 September 2018.
- [6] “Watson visual recognition,” <https://www.ibm.com/watson/services/visual-recognition/>, accessed: 13 September 2018.
- [7] “Image Recognition API & Visual Search Results,” <https://docs.aws.amazon.com/rekognition/latest/dg/labels-detect-labels-image.html>, accessed: 13 September 2018.
- [8] “Clarifai,” <https://www.clarifai.com>, accessed: 13 September 2018.
- [9] “Image recognition api — deepai,” <https://deepai.org/ai-image-processing>, accessed: 26 September 2018.
- [10] “Imagga - powerful image recognition apis for automated categorization & tagging in the cloud and on-premises,” <https://imagga.com>, accessed: 13 September 2018.
- [11] “Image recognition - talkwalker,” <https://www.talkwalker.com/image-recognition>, accessed: 13 September 2018.
- [12] 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.” *WorldCIST*, vol. 746, no. 12, pp. 174–184, 2018.
- [13] 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 and Management*, vol. 2, no. 2, pp. 1–3, 2017.
- [14] T. E. Marshall and S. L. Lambert, “Cloud-based intelligent accounting applications: accounting task automation using IBM watson cognitive computing,” *Journal of Emerging Technologies in Accounting*, vol. 15, no. 1, pp. 199–215, 2018.
- [15] A. Iyengar, “Supporting Data Analytics Applications Which Utilize Cognitive Services,” in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, May 2017, pp. 1856–1864.
- [16] V. C. Dibia, M. Ashoori, A. Cox, and J. D. Weisz, “TJBot,” in *the 2017 CHI Conference Extended Abstracts*. New York, New York, USA: ACM Press, 2017, pp. 381–384.
- [17] M. Piccioni, C. A. Furia, and B. Meyer, “An Empirical Study of API Usability,” in *2013 ACM / IEEE International Symposium on Empirical Software Engineering*

- and Measurement.* IEEE, 2013, pp. 5–14.
- [18] 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.
- [19] J. J. Blake, L. P. Maguire, B. Roche, T. M. McGinnity, and L. J. McDaid, “The Implementation of Fuzzy Systems, Neural Networks and Fuzzy Neural Networks using FPGAs.” *Inf. Sci.*, 1998.
- [20] R. S. Pressman, *Software engineering: a practitioner’s approach*. Palgrave Macmillan, 2005.
- [21] I. Sommerville, *Software Engineering*, 9th ed. Addison-Wesley, 2011.
- [22] J. W. Horch, *Practical guide to software quality management*. Artech House, 2003.
- [23] International Organization for Standardization and I. E. Commission, *Software Engineering—Product Quality: Quality model*. ISO/IEC, 2001, vol. 1.
- [24] D. Garvin, “Competing on the eight dimensions of quality,” *Harv. Bus. Rev.*, pp. 101–109, 1987.
- [25] J. A. McCall, P. K. Richards, and G. F. Walters, “Factors in software quality. volume i. concepts and definitions of software quality,” Tech. Rep., 1977.
- [26] R. B. Grady, “Measuring and managing software maintenance,” *IEEE Software*, vol. 4, no. 5, pp. 35–45, 1987.
- [27] R. G. Dromey, “A model for software product quality,” *IEEE Transactions on Software Engineering*, vol. 21, no. 2, pp. 146–162, 1995.
- [28] B. W. Boehm, “A spiral model of software development and enhancement,” *Computer*, vol. 21, no. 5, pp. 61–72, 1988.
- [29] A. J. Ko, B. A. Myers, and H. H. Aung, “Six learning barriers in end-user programming systems,” in *2004 IEEE Symposium on Visual Languages-Human Centric Computing*. IEEE, 2004, pp. 199–206.
- [30] R. E. Al-Qutaish, “Quality models in software engineering literature: an analytical and comparative study,” *Journal of American Science*, vol. 6, no. 3, pp. 166–175, 2010.
- [31] T. M. Khoshgoftaar and N. Seliya, “Comparative assessment of software quality classification techniques: An empirical case study,” *Empirical Software Engineering*, vol. 9, no. 3, pp. 229–257, 2004.
- [32] J. Ghayathri and E. M. Priya, “Software quality models: A comparative study,” *International Journal of Advanced Research in Computer Science and Electronics Engineering (IJARCSEE)*, vol. 2, no. 1, pp. pp–042, 2013.
- [33] S. H. Kan, *Metrics and models in software quality engineering*. Addison-Wesley Longman Publishing Co., Inc., 2002.
- [34] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” *arXiv preprint arXiv:1312.6199*, 2013.
- [35] H. Hosseini, B. Xiao, and R. Poovendran, “Google’s Cloud Vision API is Not Robust to Noise,” in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, Jan. 2018, pp. 101–105.
- [36] A. Rosenfeld, R. Zemel, and J. K. Tsotsos, “The Elephant in the Room.” *CoRR*, 2018.
- [37] B. Wang, Y. Yao, B. Viswanath, H. Zheng, and B. Y. Zhao, “With Great Training Comes Great Vulnerability - Practical Attacks against Transfer Learning.” *USENIX Security Symposium*, 2018.
- [38] J. Su, D. V. Vargas, and K. Sakurai, “One pixel attack for fooling deep neural networks.” *CoRR*, 2017.
- [39] 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 IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1625–1634.
- [40] A. Kurakin, I. Goodfellow, and S. Bengio, “Adversarial examples in the physical world,” *arXiv.org*, Jul. 2016.
- [41] 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 IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1625–1634.
- [42] 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,” *IEEE International Symposium on Safety, Security, and Rescue Robotics SSRR*, pp. 1–8, Apr. 2018.
- [43] K. Arnold, “Programmers are people, too,” *ACM Queue*, vol. 3, no. 5, pp. 54–59, 2005.
- [44] M. Henning, “API design matters,” *Commun. ACM*, vol. 52, no. 5, pp. 46–56, 2009.
- [45] B. W. Boehm, *Software engineering economics*. Prentice-hall Englewood Cliffs (NJ), 1981, vol. 197.
- [46] B. Boehm and V. R. Basili, “Software defect reduction top 10 list,” *Foundations of empirical software engineering: the legacy of Victor R. Basili*, vol. 426, no. 37, 2005.
- [47] Digital, “Case study: Finding defects earlier yields enormous savings,” 2003.
- [48] G. Lawton, “Developing software online with platform-as-a-service technology,” *Computer*, vol. 41, no. 6, 2008.
- [49] S. K. Garg, S. Versteeg, and R. Buyya, “SMICloud: A Framework for Comparing and Ranking Cloud Services,” in *2011 IEEE 4th International Conference on Utility and Cloud Computing (UCC 2011)*. IEEE, Nov. 2011, pp. 210–218.
- [50] —, “A framework for ranking of cloud computing services,” *Future Generation Computer Systems*, vol. 29, no. 4, pp. 1012–1023, Jun. 2013.
- [51] Z. Zheng, Y. Zhang, and M. R. lyu, “CloudRank: A QoS-Driven Component Ranking Framework for Cloud Computing,” in *2010 IEEE International Symposium on Reliable Distributed Systems (SRDS)*. IEEE, Oct. 2010,

- pp. 184–193.
- [52] L. Sun, H. Dong, F. K. Hussain, O. K. Hussain, and E. Chang, “Cloud service selection: State-of-the-art and future research directions,” *Journal of Network and Computer Applications*, vol. 45, pp. 134–150, Oct. 2014.
- [53] L. Qu, Y. Wang, and M. A. Orgun, “Cloud Service Selection Based on the Aggregation of User Feedback and Quantitative Performance Assessment,” in *2013 IEEE International Conference on Services Computing (SCC)*. IEEE, Jun. 2013, pp. 152–159.
- [54] S. Sundareswaran, A. Squicciarini, and D. Lin, “A Brokerage-Based Approach for Cloud Service Selection,” in *2012 IEEE 5th International Conference on Cloud Computing (CLOUD)*. IEEE, Jun. 2012, pp. 558–565.
- [55] J. Siegel and J. Perdue, “Cloud services measures for global use: the service measurement index (SMI),” in *SRII Global Conference (SRII), 2012 Annual*. IEEE, 2012, pp. 411–415.
- [56] “Detect Labels — Google Cloud Vision API Documentation — Google Cloud,” <https://cloud.google.com/vision/docs/labels>, accessed: 28 August 2018.
- [57] “Class EntityAnnotation — Google.Cloud.Vision.V1,” <https://googlecloudplatform.github.io/google-cloud-dotnet/docs/Google.Cloud.Vision.V1/api/Google.Cloud.Vision.V1.EntityAnnotation.html>, accessed: 28 August 2018.
- [58] “How to call the Computer Vision API,” <https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/vision-api-how-to-topics/howtocallvisionapi>, accessed: 28 August 2018.
- [59] “azure-sdk-for-java/ImageTag.java,” <https://github.com/Azure/azure-sdk-for-java/blob/8d70f41fbdb88b3a9297af5ba24551cf26f40ad4/cognitiveservices/data-plane/vision/computervision/src/main/java/com/microsoft/azure/cognitiveservices/vision/computervision/models/ImageTag.java#L24>, accessed: 28 August 2018.
- [60] “Detecting labels in an image,” <https://docs.aws.amazon.com/rekognition/latest/dg/labels-detect-labels-image.html>, accessed: 28 August 2018.
- [61] “Detecting objects and scenes,” <https://docs.aws.amazon.com/rekognition/latest/dg/labels.html>, accessed: 28 August 2018.
- [62] “Label (AWS SDK for Java - 1.11.401),” <https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/amazonaws/services/rekognition/model/Label.html>, accessed: 28 August 2018.
- [63] “Label - Amazon Rekognition,” [https://docs.aws.amazon.com/rekognition/latest/dg/API\\_Label.html](https://docs.aws.amazon.com/rekognition/latest/dg/API_Label.html), accessed: 28 August 2018.
- [64] “StackOverflow: Vision API topicality and score always the same,” <https://stackoverflow.com/q/51273104>, accessed: 28 August 2018.
- [65] “StackOverflow: What is default threshold for Microsoft face api?” <https://stackoverflow.com/q/41199672>, accessed: 28 August 2018.
- [66] “StackOverflow: Meaning of score in Microsoft Cognitive Service’s Entity Linking API,” <https://stackoverflow.com/q/43249555>, accessed: 28 August 2018.
- [67] “StackOverflow: Google vision api: what do web detection scores indicate?” <https://stackoverflow.com/q/49436244>, accessed: 28 August 2018.
- [68] “StackOverflow: Validation of adding face to person group,” <https://stackoverflow.com/q/48761206>, accessed: 28 August 2018.
- [69] “GitHub Issues: Vision api imageProperties dominant color score meaning,” <https://github.com/GoogleCloudPlatform/google-cloud-node/issues/1324>, accessed: 28 August 2018.
- [70] “GitHub Issues: Please add Min density, resolution, etc for good OCR for each applicable API,” <https://github.com/MicrosoftDocs/azure-docs/issues/10228>, accessed: 28 August 2018.
- [71] “What is Computer Vision?” <https://docs.microsoft.com/en-gb/azure/cognitive-services/computer-vision/home#tagging-images>, accessed: 28 August 2018.
- [72] E. Duala-Ekoko and M. P. Robillard, “Asking and answering questions about unfamiliar APIs: An exploratory study,” in *Software Engineering (ICSE), 2012 34th International Conference on*. IEEE, 2012, pp. 266–276.

TABLE III: Techniques of the three assessed cognitive vision APIs.

Technique	A	B	C
Object Recognition	•	•	•
Object Localisation	(beta)		
Scene Recognition	•	•	•
Unsafe Content Recognition	•	•	•
Text Extraction (OCR)	•	•	•
Handwriting Extraction	(beta)	•	
Face Detection	•	•	•
Face Analysis	•	•	•
Face Recognition		(partial)	•
News-Related Events	•		
Product/Logo Recognition	•		
Landmark Detection	•	•	
Image Analysis	•	•	•
Domain-Specific Analysis		•	
Image Description		•	
Image Type Identification		•	



## **Appendix C**

### **Ethics Application Draft**



**Office use only**

Reference number:

Quiz results received for all applicants? Y/N

**DEAKIN UNIVERSITY HUMAN ETHICS ADVISORY GROUP  
LOW-RISK APPLICATION FORM**

The [National Statement on Ethical Conduct in Human Research](#) (2007) defines low risk research as:  
'Research in which the only foreseeable risk is one of discomfort. Research in which the risk for participants is more serious than discomfort is not low risk'.

**Project Title:** *Magic beyond the method call: A communication framework for application developers using 'friendly' machine learning*

**Proposed Start Date:** February 2019      **Proposed end date:** February 2021

**Principal Investigator/s:** Rajesh Vasa

**Student Investigator/s (if applicable):** Alex Cummaudo    **Student id:** 217092024

**Degree/s for which student/s enrolled:** Doctor of Philosophy

**School:** Applied Artificial Intelligence Institute (A<sup>2</sup>I<sup>2</sup>)    **Faculty:** SEBE

**Contact Telephone No:** 0420 553 685

**Email:** ca@deakin.edu.au

**Contact details of all researchers involved in the project:**

Name	Role	Email	Phone
Alex Cummaudo	Student Investigator	ca@deakin.edu.au	0420 553 685
Rajesh Vasa	Principal Supervisor	rajesh.vasa@deakin.edu.au	03 9246 8060
John Grundy	External Supervisor	john.grundy@monash.edu	03 9905 0511
Mohamed Abdelrazek	Associate Supervisor	mohamed.abdelrazek@deakin.edu.au	03 9244 6140
Andrew Cain	Associate Supervisor	andrew.cain@deakin.edu.au	03 9246 8655
Scott Barnett	Collaborator	scott.barnett@deakin.edu.au	N/A

**Please note:** There has been an update by Microsoft which blocks links to single sign-on web pages.

To access the Deakin Human Ethics Guidelines referred to in this form, cut and paste the following address to your browser:

<http://www.deakin.edu.au/students/research/research-support-and-scholarships/integrity-secure/human-ethics/dheg>

Links to external sites in this form will still work.

## PART A: Excluded Categories (see *National Statement Chapter 5.1.6*)

### 1 Does your project involve any of the following?

- Yes  No  Aboriginal or Torres Strait Islander Peoples or issues
- Yes  No  Research involving pregnant women or the human fetus
- Yes  No  People highly dependent on medical care who may be unable to give consent
- Yes  No  People with a cognitive impairment, an intellectual disability, or mental illness
- Yes  No  People who may be involved in illegal activities
- Yes  No  Interventions and therapies, including clinical trials and non-clinical trials and innovations that involve blinding of participants (i.e. participants are ‘blind’ to which participant group they have been assigned)
- Yes  No  Human genetics
- Yes  No  Human stem cells
- Yes  No  Projects involving ionising radiation
- Yes  No  Travel to regions classified as Level 2, 3 or 4 (see [DFAT](#) and Section 35 of the Deakin Guidelines)
- Yes  No  Projects involving active concealment or planned deception of participants
- Yes  No  Collection of identifiable personal information, without permission from the person identified
- Yes  No  Risk of harm to participants (more serious than discomfort, *National Statement Chapter 2.1.6*)
- Yes  No  Opt-out consent in relation to health or sensitive data (*National Statement, Chapter 2.3*) Sensitive data includes personal information about an individual’s health (including predictive genetic information), racial or ethnic origin, political opinions, membership of a political association, professional or trade association or trade union, religious beliefs or affiliations, philosophical beliefs, sexual orientation or practices, criminal record, biometric information that is to be used for certain purposes and biometric templates. Please note if you are intending to use opt-out consent then you must complete a waiver of consent, using the [Victorian Privacy Supplement](#). Please complete a higher than low risk form and submit to DUHREC (together with the Victorian Privacy Supplement).
- Yes  No  Waiver of consent using the [Victorian Privacy Supplement](#). Please complete a higher than low risk form and submit to DUHREC (together with the Victorian Privacy Supplement).

If you answered yes to ANY of these elements, your project is not eligible for low-risk review. You should complete a higher than low risk application form for full ethical review by DUHREC (see Section 6.5 of the Deakin Guidelines)

### 2 Does your project involve ethical review by another Australian organisation?

- Yes  No  If yes, your project is not eligible for review by a HEAG. You should consult Section 6 of the Deakin Guidelines regarding the processes which apply to applications previously approved by another organisation.

### 3 Does your project involve ONLY use of existing collections of non-identifiable data?

Data are non-identifiable when they do not identify the people to whom the information relates – identifiers should never have been collected, or should have been permanently removed from the data set before you received it.

- Yes  No  If yes, you should complete the application form for Exemption from Ethical Review (see Section 6 of the Deakin Guidelines).

## PART B: Checklist

This checklist will help you decide whether your research may be submitted for review by your Faculty HEAG. Research is eligible for low-risk review if the foreseeable risk level is no more than discomfort.

If you answer 'YES' to any items on the checklist **your project is not eligible for low risk review unless** you can explain how this potential risk will be managed or minimised to ensure that the project remains low risk. This should be explained in the special case assessment section (Section 6) below.

**It is your responsibility to assess the level of risk associated with your project. If your project is not considered low risk by the HEAG, you will be required to complete a high-risk application for submission to DUHREC.**

*Please ensure you include all signatures before submitting the application  
as approval cannot be granted until they are received.*

### 1 Are any of the following topics to be covered in part or in whole?

Parenting	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Sensitive personal issues	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Sensitive cultural issues	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Grief, death or serious/traumatic loss	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Gambling	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Eating disorders	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Illicit drug taking	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Substance abuse	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Self-report of criminal behaviour	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Any psychological disorder, depression, mood states and/or anxiety	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Suicide	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Sexuality, sexual behaviour or gender identity	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Race or ethnic identity	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Any disease or health problem	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Fertility	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Termination of pregnancy	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO

### 2 Are any of the following procedures to be employed?

Use of personal data obtained from Commonwealth or State Government Department/Agency	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Concealing the purposes of the research	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Covert observation	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Audio or visual recording without consent	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Recruitment via a third party or agency	<input type="checkbox"/> YES	<input type="checkbox"/> NO
Withholding from one group specific treatments or methods of learning, from which they may 'benefit' (e.g. in medicine or teaching)	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Psychological interventions or treatments	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Administration of physical stimulation	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Invasive physical procedures	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO

Infliction of pain	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Administration of drugs or placebos	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Administration of other substances	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Use of medical records where participants can be identified or linked	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO

### 3 PARTICIPANT VULNERABILITY ASSESSMENT

**Does the research specifically target participants from any of the following groups?**

Children or young people under 18 years	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
People with a physical disability or vulnerability	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
People whose ability to give consent is impaired	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Residents of a custodial institution	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
People unable to give free informed consent because of difficulties in understanding the Plain Language Statement or Information Sheet (e.g. language difficulties)	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Members of a socially identifiable group with special cultural or religious needs or political vulnerabilities	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
People in dependent or unequal relationship with the researchers (e.g. lecturer/student, doctor/patient, teacher/pupil, professional/client)	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
People with existing relationships with the researcher (e.g. relative, friend, co-worker)	<input checked="" type="checkbox"/> YES	<input type="checkbox"/> NO
People in a workplace setting with the potential for coercion or problems of confidentiality (e.g. employer/employee)	<input checked="" type="checkbox"/> YES	<input type="checkbox"/> NO
Participants able to be identified in any final report when specific consent for this has not been given	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Persons not usually considered vulnerable but would be thought so in the context of the project	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO

### 4. RESEARCH IN OVERSEAS SETTINGS ASSESSMENT

**Does the research involve any of the following?**

Research being undertaken in a politically unstable area	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Research involving sensitive cultural issues	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Research in countries where criticism of government and institutions might put participants and/or researchers at risk	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
*Please indicate which DFAT level of advice applies to the region you intend visiting (see <a href="#">DFAT</a> and Section 35 of the Deakin Guidelines)	<input checked="" type="checkbox"/> 1	<input type="checkbox"/> 2
	<input type="checkbox"/> 3	<input type="checkbox"/> 4
<i>*Note: Travel to regions classified as Level 2 (or above) is considered higher than low risk and not eligible for review by a HEAG.</i>		

### 5. OTHER RISKS

Are there any risks to the researcher, (e.g. research undertaken in unsafe environments or trouble spots)?	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO
Are there any other risks not covered in this assessment that you consider may be relevant?	<input type="checkbox"/> YES	<input checked="" type="checkbox"/> NO

## **6. SPECIAL CASE ASSESSMENT**

If you have answered 'YES' to an item in the checklist but you still believe that because of the particular nature of the project and the participants your project may still be eligible for low risk review. Please provide details below, or attach an additional sheet.

### **SPECIAL CASE DETAILS:**

#### **People with existing relationships with the researcher (e.g., relative, friend, co-worker):**

We intend to run a 'pilot' study of our survey and interview with participants being fellow co-workers A<sup>2</sup>I<sup>2</sup>. This is later explained in Part C Section 9. Participants will not be required to provide any information that is identifiable within our results. Participation is entirely voluntary, and participants will be free to withdraw from the study at any point. This is communicated verbally to participants and by way of a PLS.

#### **People in a workplace setting with the potential for coercion or problems of confidentiality (e.g., employer/employee):**

Recruitment of employees from within the A<sup>2</sup>I<sup>2</sup> group as participants of the pilot study will act as a recruitment method (see Part C, Section 9). The researchers will ensure that A<sup>2</sup>I<sup>2</sup> employees are only to participate in the study in an entirely voluntary basis and that there is no requirement or compulsion to participate in the project. All participants are free to withdraw their consent to participate before they submit the study and will not be pressured to participate. Any revocation of consent will not affect their relationship with A<sup>2</sup>I<sup>2</sup>.

The principal investigators will not be present during the usability study and this experiment will only be undertaken by the student investigator. The principal investigators will not know which A<sup>2</sup>I<sup>2</sup> employees are participants. There will be no unequal relationship present with participants as indicated previously.

## PART C: Project

### 1. Aims of the project

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 services are typically provided as cloud services to developers on the web, and we henceforth refer to them as Cloud Intelligent Services (CISs).

These CISs 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 CISs do not present sufficient detail in their Application Programming Interface (API) 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. We specifically target the area of Computer Vision CISs (cvCISs).

Our goal is to develop a framework to better document a cvCIS that improves both the quality of the software produced and the developer productivity behind it.

We propose **three hypotheses** and **seven research questions** in this study:

#### **Hypothesis 1: Existing CISs present insufficient API documentation for general use.**

**Description:** API documentation of intelligent services are inadequate and insufficient given the disparity of mindsets between the application developers and CIS providers. Chiefly, application developers have limited general understanding of the ‘magic’ that occurs behind these probabilistic ‘intelligent’ APIs. We do not know what key aspects of the documentation matter to them, nor what they do or do not understand of the existing documentation.

**Goal:** To improve the effectiveness of the documentation in existing CIS providers, specifically of cvCIS APIs.

#### **Research Questions:**

- RQ1.1. What practices are in use for CIS documentation?
- RQ1.2. How do developers currently perceive and interpret the API documentation?
- RQ1.3. What additional attributes would they prefer to be included in the documentation?

#### **Hypothesis 2: Existing CISs present insufficient metadata for context-specificity.**

**Description:** cvCISs respond with insufficient information for developers to operationalise the service into a business-driven application (as no context can be specified) and, thus, additional metadata is needed to assist developers. Such metadata is likely to be needed as part of the response objects of the API.

**Goal:** To improve the quality of *context-specific response data* from the API endpoints of cvCISs.

#### **Research Questions:**

- RQ2.1. What current problems exist due to the lack of context-specific metadata?
- RQ2.2. What additional metadata do developers desire to achieve building their applications?

## **Hypothesis 2: Improving documentation and metadata will improve quality, productivity or developer informativeness.**

**Description:** The implication of hypotheses 1 and 2 suggest that improving both the documentation and providing further metadata will improve product quality (internal or external), and/or developer productivity and/or developer education in developing software with intelligent components.

**Goal:** To confirm if improvements to API documentation and response metadata are reflected as improvements to product quality, developer productivity and/or developer education.

### **Research Questions:**

RQ3.1. Does an improvement of documentation or metadata correlate to an improvement in software quality, developer productivity and/or developer informativeness?

RQ3.2. With respect to RQ3.1, the three aspects are explored:

a) Does the improvement cause increased product quality, as measured through improved external quality metrics?

b) Does the improvement cause increased developer productivity, as measured through improved internal quality metrics?

c) Does the improvement cause increased developer informativeness or increased confidence in developing CIS-powered applications?

## **2. Research design and methods**

Give a concise and simple description of the proposed research design and the methods to be used. Please include all data collection procedures and all groups of participants.

We propose two experiments involving humans for this study:

Experiment 1: Developing an initial framework from opinion mining of developers

Experiment 2: Validating the developed framework using a controlled observational study

Both experiments are outlined below.

### **Experiment 1**

**Motivation:** We wish to corroborate qualitative data found by mining developer discussion forums such as Stack Overflow against developers in real-life. The data from Stack Overflow will highlight what types of questions developers are asking with regards to CIS usage, and we will investigate the same.

Please note that this experiment is to be conducted in a context-agnostic way by asking generalised questions towards different types of CISs (i.e., not targeting a specific type of CIS in a specific scenario).

**Method:** We wish to use survey research to develop a cross-sectional survey consisting of TWO survey instruments: (1) a structured questionnaire and (2) semi-structured interview. We adopt the methodology of shaping our survey objectives and around our research goals provided by Kitchenham and Pfleeger<sup>1</sup>.

We plan to conduct an initial pilot study within A<sup>2</sup>I<sup>2</sup> that will help us refine and shape any reliability or validity issues of our survey instruments. We will anticipate colleagues to critique our initial questionnaire and interview outline. This assists us with identifying any

<sup>1</sup>B A Kitchenham, "Kitchenham, B.: Guidelines for Performing Systematic Literature Reviews in Software Engineering. EBSE Technical Report EBSE-2007-01," n.d.A Kitchenham, "Kitchenham, B.: Guidelines for Performing Systematic Literature Reviews in Software Engineering. EBSE Technical Report EBSE-2007-01."

problems with these instruments such as potential lack of response rate, determining the exact average time the instruments take to complete, and what follow-up procedures best work. Once this pilot study is complete, we will then distribute it to industry.

**Collection:** *Online questionnaire:* An online survey collection tooling such as Qualtrics, Google Forms or SurveyMonkey (this will be decided once all three have been evaluated) will be used. This data will act chiefly as quantitative data on developer opinions towards CIS documentation and usage.

*Semi-structured interview:* This will consist of 15-30 minute sessions (timing TBC after initial pilot studies) where a basic outline of interview questions will be adhered to, allowing us to ask follow-up questions and gather qualitative data on developer opinions towards CIS documentation and usage. We will opt to record these sessions using an audio-recording device and will ensure that participants have agreed to have their voice recorded during the interview. This will allow us to transcribe the audio and feed it into a qualitative assessment program such as NVivo.

**RQs answered:** This assists us to answer RQ1.2 and RQ1.3 and allows us to begin developing our initial framework.

## **Experiment 2**

**Motivation:** We wish to evaluate how our findings from Experiment 1 translates to context-specific situations. That is, we wish to add context to application developer's usage of our framework and investigate if the generalised findings of Experiment 1 are genuine by conducting action research (embedding the prototype internally within internal tooling used at A<sup>2</sup>I<sup>2</sup>) in combination with an observational study on software developers using our framework on a specific task.

**Method:** We wish to recruit an even number of developers (e.g., 20—the exact number will be determined) of varying levels of experience (from beginner programmer to principal engineer) to complete five tasks (consisting of implementing certain applications on a computer using a CIS) under an observational comparative controlled study. Half of these developers will complete 5 tasks with the new framework and without the new framework (i.e., using the as-is/existing documentation). From this we compare if the framework makes improvements by evaluating differences in the way developers approach the tasks.

**Collection:** We will use the ‘think-aloud’ protocol to ask developers to describe their mental model in order to assess how differently they approach completing the task. We will note down observations made during the study to map their mental model to the actions they perform with and without our framework.

Similar to Experiment 1, we will opt to record these sessions using an audio-recording AND video-recording device and will ensure that participants have agreed to have their voice and actions on the computer recorded during the interview. This will allow us to transcribe the audio and feed it into a qualitative assessment program such as NVivo.

**RQs answered:** This assists us to answer RQs 2.1, 2.2, 3.1, 3.2 by validating our framework in a context-specific environment.

### **3. Use of existing stored data**

Please list any existing stored data that you plan to use as part of the project e.g. health or employment records used for recruitment, or comparison. Please include in your answer:

- The type and number of records being accessed
- Whether the records identify individual people
- How you will obtain permission to use them (consent from individuals or permission from custodians of non-identifiable data).

(See the *National Statement* [Chapter 3.1](#) and [Chapter 3.2](#); and Section 25 of the Deakin Guidelines for more information.)

N/A

### **4. Risks and benefits**

#### **Give a summary of the expected benefits of this project**

This may include benefits to the broader community, the participants, people with whom the participants identify or the researcher (see the *National Statement* on [benefits](#) for more information.)

The benefits of this project benefits the participants—i.e., software engineers of all levels of seniority—by contributing the following facets for both application developer consumers and CIS providers:

1. **Contribution I:** An intelligent service API documentation quality assessment framework to evaluate how well the intelligent service has been documented for software engineers to use.
2. **Contribution II:** A concrete sample solution or framework that improves such metrics, thereby confirming that our documentation and metadata quality assessment framework improves these facets.

The wider implications of these contributions suggest that the quality and robustness of applications that are built dependent on intelligent services will be improved AND the productivity of application developers using these services will be improved. A by-product of these two is that the uptake in intelligent services by application developers will be increased as it will make it easier for them to implement their applications.

#### **Give a summary of the expected risks of this project and how they will be managed**

This should include any risks to participants, researchers, to the environment or to Deakin or other organisations (see the *National Statement* on [assessment of risk](#) for more information).

Participants undertaking the online questionnaire and/or the observational study may suffer from eye strain when looking at a computer for long periods of time. While we do not anticipate these tasks to take more than an hour, some participants, particularly those who suffer from poor vision and eye strain, may find it difficult to read on a computer screen for long periods of time. We advise participants to take frequent breaks as needed and will prompt them to do so.

### **5. Monitoring**

As the researcher, how will you monitor the progress of the research?

You should include details of planned communication between members of the research team (e.g. face to face meetings, email, telephone or Skype) (see the *National Statement* [Chapter 5.5.3](#) for more information).

The research team is to conduct weekly meetings face-to-face within A<sup>2</sup>I<sup>2</sup> of all those who are partaking in the project. The student and primary investigators are to meet weekly to discuss the nature of the project, any progress and challenges.

### **6. Resources**

Please explain the amount and source of funding (sponsorship, tender, grant etc.). If there are specific resources required for the project how will they be provided?

Deakin University School of IT provides students with \$700 reimbursement of expenses for HDR students. These funds may be used as reimbursement to participants by way of gift cards.

## **7. Conflict of interest**

Do any of the researchers or others involved in this project have any conflict of interest in relation to it? If so, please explain how this will be managed (see the *National Statement* on [Conflict of Interest](#) for more information).

N/A

## **PARTICIPANTS**

### **8. Describe your participant group/s**

Please include the following information for each participant group how many participants you plan to recruit:

- a justification for the number of participants chosen for each participant group
- the inclusion and exclusion criteria.

(See the *National Statement* [Section 4](#) for more information.)

**Target Population:** Application software developers with varying degrees of experience (including those who and who have not used CISS or related tools before) and varying understanding of fundamental machine learning concepts.

**Justification:** We wish to sample if developers with more experience in programming makes an impact on the results we find. Similarly, those who are less-experienced (i.e., less seniority) but perhaps have had more familiarity with using machine learning frameworks and toolkits in the past, or even CISS, we would like to see if this makes a difference also.

**Inclusion:** Must have worked in software engineering industry or is pursuing a degree in a software engineering related field.

**Exclusion:** Cannot be a trained data scientist/engineer or academic researcher who has implemented/used machine learning techniques previously and is already overly confident in working with machine learning concepts.

### **9. Explain your recruitment process**

Please include the following information for each participant group:

- How will you locate the participants that you plan to recruit? If through existing records or contact lists, please explain how this will be done in a way that does not infringe privacy requirements.
- How will initial contact be made?
- If you plan to use a document or spoken statement e.g. flyer, letter, advertisement, phone call, please attach a copy of the document or script to this application.
- All advertisements (both written and spoken) must include the following statement: "This study has received Deakin University ethics approval (reference number: insert reference number here)."
- Will the participants be screened?
- If there is a screening tool, please attach a copy.

(See Section 8 of the Deakin Guidelines for more information.)

#### **Pilot Study**

**Locating:** The pilot study is conducted under A<sup>2</sup>I<sup>2</sup>; as this is the internal research group the student and principal investigators work under, then locating them does not require existing records or contact lists.

**Contact:** Initial contact will be made by word-of-mouth and by distribution of messages posted on Slack, the group's internal messaging system.

**Statement:** N/A

**Screening:** N/A

## **External Study**

**Locating:** The external study will have its participants recruited via several forms: (1) through word of mouth to colleagues of friends and family of the student investigator; (2) through word of mouth to engineers of clients of A<sup>2</sup>I<sup>2</sup>; (3) via online forums such as Reddit where developers congregate for discussion.

**Contact:** As above.

**Statement:** N/A

**Screening:** N/A

## **CONSENT**

### **10. Describe the consent process**

There are a variety of ways in which consent can be established, most commonly by giving participants a Plain Language Statement and Consent Form (PLSCF) or by return of survey. You may wish to consult Section 9 of the Deakin Guidelines for more information. Please include details such as:

- how and when you will provide consent materials to your potential participants
- how, when and to whom participants will indicate their consent.

(See the *National Statement* [Chapter 2.2](#) and [Chapter 2.3](#) for more information.)

A Plain Language Statement (PLS) will be provided in person to participants prior to the study from beginning by the student investigator, except for participants undertaking the online questionnaire where the PLS will be distributed online.

Participants will assent their consent to the student investigator after reading the PLS, wherein participants will acknowledge: (1) that they are of 18 years of age; (2) are able to fluently speak English; (2) are located within Australia; (3) are free to withdraw from the study at any time. They will be asked to sign a consent form, and for the online questionnaire, this acknowledgement will be provided by way of an acceptance form before the questionnaire begins.

### **11. Will there be reimbursement of expenses or incentives to participate?**

Where expenses will be reimbursed please state:

- the nature of the expenses incurred by participants
- the maximum value of any intended reimbursement.

Where incentives to participate are offered, please explain:

- why you consider that the proposed incentive will not encourage participants to take risks they would not otherwise take. In doing so, please consider both the risks associated with participation and the value of the incentive, relative to your participant group.

(See the *National Statement* [Chapter 2.2.10-2.2.11](#); and Section 8 of the Deakin Guidelines for more information.)

There will be no reimbursement to participants.

If this results in a very low participation rate, then gift cards of \$50 will be purchased by way of incentivising the study.

### **12. Pre-existing or unequal relationships**

Do any of the proposed participants have existing relationships with the researchers, each other or with any other organisation involved in the research? Please explain the relationships, and how you will make sure that participants do not feel pressured to take part.

(See the *National Statement* [Chapter 4.3](#); and the Section 22 of the Deakin Guidelines for more information.)

The internal pilot study involves co-workers in A<sup>2</sup>I<sup>2</sup>. Participants will not be required to enter any information that is identifiable. Participation is entirely voluntary and participants will be free to withdraw from the study at any point. This will be communicated verbally to A<sup>2</sup>I<sup>2</sup> staff. Recruitment of up to 10

employees from within the A<sup>2</sup>I<sup>2</sup> group as participants of the pilot study will act as the recruitment method. The researchers will ensure that A<sup>2</sup>I<sup>2</sup> employees are only to participate in the study in an entirely voluntary basis and that there is no requirement or compulsion to participate in the project. These participants will be chosen by the student investigator and will be not known to the primary investigators to limit coercion. All participants are free to withdraw their consent to participate before they submit the study, and will not be pressured to participate. Any revocation of consent will not affect their relationship with A<sup>2</sup>I<sup>2</sup>.

Furthermore, some A<sup>2</sup>I<sup>2</sup> employees will provide technical feedback regarding the framework developed and results to help improve the project's findings. It is within the research's best interest that all participants provide honest and frank feedback about this project and the study, so therefore any recommendations provided verbally from participants about the project or the framework is within the project's best interests; there is therefore no chance for A<sup>2</sup>I<sup>2</sup> to pressure employees to present a particular outcome on the study.

### **13. Does your project include children or young people under 18 years?**

If your project involves people under the age of 18, please answer the following questions.

- What age group is involved?
- Will parental/guardian consent be obtained? If the young people will consent on their own behalf, how their capacity to do this will be judged?
- Is it necessary to involve people under 18? Could your projects be undertaken with adult participants?
- Is the methodology appropriate for children/young people?
- Is there any reason to consider that participation in the research is not in the best interests of the children/young people?

(For further information, consult the *National Statement* [Chapter 4.2](#); and Section 19 of the Deakin Guidelines.)

No — all participants must be 18 years or older to participate.

### **14. Language and communication issues**

Will your project involve people who cannot communicate easily in English? (e.g. people who are not confident English speakers, or who have a disability, such as a hearing impairment that requires special arrangements for participation). If so, please explain how translation/interpretation issues will be managed.

(For further information consult Section 24 of the Deakin Guidelines.)

No — all participants must be able to speak English fluently.

### **15. People in other countries**

If you are planning to undertake research in other countries, please answer the following questions. What are the legal and ethical requirements for conducting research in the designated country?

- What arrangements will be in place for a local, readily accessible contact to receive responses, questions and complaints about the research? (*National Statement* [Chapter 4.8.16](#))
- How will the research be monitored on site?
- Are there cultural sensitivities relating to the research? How will these be managed?

If the research is to be conducted in a language other than English, please ensure that you have covered all relevant language issues under question 14.

(For further information consult the *National Statement*, [Chapter 4.8](#); and Section 35 of the Deakin Guidelines.)

No — all participants must be located within Australia.

## **CONFIDENTIALITY / PRIVACY**

### **16. Will you be collecting data in identified form?**

Data are generally divided into:

- **identifiable** (also called personal): the person to whom the data relates can be established from the data – either because they are named, or information that identifies them is included (e.g. position in an organisation at the time)
- **re-identifiable** (also called coded): the identifiers have been removed from the information and replaced with a code.
- **non-identifiable**: the data were collected anonymously, or all identifiers have been permanently removed.

Please explain the form in which the data will be collected. If you plan to collect it in identified form and later remove the identifiers, please explain how and when.

(See Section 10 of the Deakin Guidelines for more information.)

Non-identifiable. Notes from interviews and observational studies will not be associated to any specific member. Results from questionnaire will not collect any identifiable data.

### **17. Storage of data**

Data storage should meet the requirements of the Research Conduct Policy which can be found in the [Deakin Legislation and Policy Library](#). In most cases data should be stored securely at Deakin, for a period of at least five years after the final publication of the research outcomes. If the data will be stored in another location, please explain this, and how data security will be maintained. You should include:

- whether the data will be identified/re-identifiable/non-identifiable
- how security will be maintained (locked storage, secure server, etc.)
- how long the data will be stored
- if and when the data will be disposed of and how security will be maintained.

(See Section 10.8 of the Deakin Guidelines for more information.)

Data will be entirely non-identifiable. Data provided all tasks will be stored in computer file formats on a secured server that is located within the Google Drive platform in addition to a secure PC within DSTIL. Swipe-card access is required to enter DSTIL. In addition, this PC will be encrypted and password protected. Data will be stored for the duration of the project and may be securely disposed from the PC by December 2021. It will not be disposed from Google Drive for archival purposes.

### **18. Publication of results**

(See Section 4 of the Australian Code for the Responsible Conduct of Research for more information.)

Whose responsibility will it be to notify participants of the outcome of the research?

The student investigator.

How will you notify participants of the outcome of the research?

No identifiable data will be collected, and because participants will not be asked for any communication methods to correspond back to them, there is no way to inform them of publication of results. However, a link to the project website will be provided to participants on the PLS, which they can then periodically check. When the research is published, a post on the update of the progress of the research will be made (i.e., by linking to the Deakin University Library publication page).

How will your research be reported/published?

We aim to publish to several venues, specifically those outlined below. Journals are shown under the double line; conferences are shown above the line. Task # refer to:

- 1.2: Discussion mining from websites (*not relevant for ethics application*)
- 1.7: Survey research from findings from Experiment 1
- 3.2: Outcomes from findings of observations made in Experiment 2

<b>Venue</b>	<b>Ranking</b>	<b>Task #</b>	<b>Location</b>	<b>Submissions Due</b>	<b>Date</b>
ESEC/FSE	A2	1.2	Tallinn, Estonia	20 Feb 2019	26–30 Aug 2019
ASE	A1	1.2, 1.7	San Diego, USA	6 May 2019	11–15 Nov 2019
CHI	A1	1.2, 1.7	Honolulu, USA	13 Sep 2019	25–30 Apr 2020
ICSE	A1	1.7	Seoul, South Korea	TBA	23–29 May 2020
MSR	A1	1.2, 1.7	Seoul, South Korea	TBA	23–29 May 2020
ICML	A1	1.7	Vienna, Austria	TBA	12–17 Jul 2020
ICMLA	B2	1.7, 3.2	Copenhagen, Denmark	15 Nov 2020	11–12 Jun 2021
TOSEM	N/A	1.2, 1.7; 3.2	N/A	N/A	N/A
IEEE Software	N/A	1.2, 1.7; 3.2	N/A	N/A	N/A
TSE	N/A	1.2, 1.7; 3.2	N/A	N/A	N/A

How will you manage participant confidentiality?

Participants will be completely anonymised and will not be asked to give any identifiable data.

## PART D: Declarations

1 I/We, the undersigned declare that the information supplied in this application is true and accurate to the best of my/our knowledge.

I / We the undersigned have read the *National Statement on Ethical Conduct in Human Research* and accept responsibility for the conduct of the project detailed in this application in accordance with the principles contained in the Statement and any other conditions laid down by Deakin University or the Human Ethics Advisory Group.

I/We the undersigned, declare that where the research project may involve contact with a child or young person under the age of 18, I/we have a current Working with Children Check.

Where the project involves a student researcher, as the supervisor I accept responsibility for ensuring that ethics approval is obtained prior to commencing the research and for overseeing the ethical conduct of the project as detailed in the ethics application.

### Signatures:

Principal Investigator/s Date:

Associate Investigator/s Date:

Student Investigator/s Date:

Student Investigator/s Date:

Student Investigator/s Date:

## 2 ACKNOWLEDGEMENT OF HEAD OF SCHOOL\*/DIRECTOR OF RESEARCH OR THEIR NOMINEE

I the undersigned acknowledge that the Faculty has considered and approved the academic worth of the project described in this application.

Name:

Title:

Signature: Date:

\*If the Head of School (or similar) is also a member of the research or supervisory team, a more senior member of University staff e.g. Dean or Associate Dean (Research) must sign the project as authorising officer.

## Part E: Attachments

Have you attached the following?

Yes  No  N/A

A copy of the email or certificate confirming successful completion of the online human ethics quiz (first time applicants) or project id of an ethics application/s on which you are listed. For more information on the quiz, copy and paste the following link into your browser: <http://www.deakin.edu.au/students/research/research-support-and-scholarships/integrity-secure/human-ethics/human-research-ethics-training>

Yes  No  N/A

A copy of any advertisements/flyers or other recruitment materials. All advertisements (both written and spoken) must include the following statement: "This study has received Deakin University ethics approval (reference number: insert reference number here)."

Yes  No  N/A

A copy of the Plain Language Statement and Consent Form (PLSCF) or other consent materials to be used in the project

Yes  No  N/A

A copy of any survey, list of questions/topics for interviews, or other materials to be used in this project. Please note Deakin University's preferred online survey platform is Qualtrics. More information on Qualtrics can be found [here](#) or for technical assistance contact [eresearch@deakin.edu.au](mailto:eresearch@deakin.edu.au)

Yes  No  N/A

Any other documents to be supplied to the participants or used in the conduct of the project

Yes  No  N/A

A letter of support from the organisation/s involved or an organisational PLSC if you are proposing to recruit participants through an external organisation/s

Yes  No  N/A

A completed Organisational Consent Form Coversheet (available on the [Application Forms page](#)) if you are recruiting Deakin staff and students

Please submit all documents via email to your faculty's HEAG:

Faculty of Arts and Education

[artsed-ethics@deakin.edu.au](mailto:artsed-ethics@deakin.edu.au)

Faculty of Business and Law

[blethics@deakin.edu.au](mailto:blethics@deakin.edu.au)

Faculty of Health

[health-ethics@deakin.edu.au](mailto:health-ethics@deakin.edu.au)

Faculty of Science, Engineering and

[sciethic@deakin.edu.au](mailto:sciethic@deakin.edu.au)

Built Environment

Please note: if the hyperlinks in this form result in an error message, return to the form and:

1. right click on the hyperlink
2. click on Edit Hyperlink
3. copy the URL to your browser.

Deakin University is collecting your personal information on this form for the primary purpose of processing your human research ethics application. It will also use this information for monitoring your compliance with the approved protocol. For these purposes Deakin may also provide this information to potential research participants, past or current research participants, or other interested parties in your research. You are not required to provide the information requested, however if the information is not provided, Deakin may not be able to process your ethics application. Deakin manages personal information it holds, including requests by individuals for access to their personal information, in accordance with the Privacy and Data Protection Act 2014 (Vic). Deakin's Privacy Policy may be viewed on Deakin's [Policy Library](#). Information on privacy at Deakin is available at <http://www.deakin.edu.au/footer/privacy>. Questions about privacy may be directed to the Privacy Officer on (03) 5227 8524 or by email to [privacy@deakin.edu.au](mailto:privacy@deakin.edu.au).

A Kitchenham, B. "Kitchenham, B.: Guidelines for Performing Systematic Literature Reviews in Software Engineering. EBSE Technical Report EBSE-2007-01," n.d.



### DEAKIN UNIVERSITY ORGANISATIONAL CONSENT FORM

Where researchers want to conduct research with Deakin staff and students as participants, the research will be approved by:

- the Head of Academic Unit for staff and students enrolled in their Academic Unit (including participants in combined courses where the research relates to their enrolment in that Academic Unit).
- the Faculty Executive Dean or Institute Director for staff and students in more than one Academic Unit (ie more than one School or Department).
- the Dean of Students when recruiting students in two or more Faculties or Institutes.

**PLEASE NOTE:** [Deakin's Research Conduct Policy](#) states that **approval should only be given where staff and student participation is appropriate and necessary for the research and other options are not available**. Such research will minimise the number of staff and students involved. Where the research is by an external party it must also be of benefit to the University and relevant results provided to staff or students involved.

Approval should be obtained **prior to seeking human ethics approval**.

#### Project and research team details:

<b>Project Title:</b>	Magic beyond the method call: A communication framework for application developers using 'friendly' machine learning
<b>Principal Investigator:</b>	Alex Cummaudo (Student); Rajesh Vasa (Principal Supervisor)
<b>Other Investigator/s:</b>	John Grundy, Mohamed Abdelrazek, Andrew Cain, Scott Barnett
<b>School:</b>	Applied Artificial Intelligence Institute
<b>Faculty:</b>	SEBE
<b>Campus:</b>	Burwood
<b>Contact Telephone No:</b>	0420 553 685
<b>Email:</b>	ca@deakin.edu.au

**Has the project been approved by an external Human Research Ethics Committee (HREC)?**

Yes  No

If yes, please include name of HREC:

#### Part A: Recruitment details

1. Please provide the sample size (number of participants) you wish to recruit: 10
2. Please state the name of the course/s being undertaken by these participants: N/A
3. In which academic unit is the course/s conducted?: N/A
4. In which School/s and/or Faculties are these participants from?: Applied Artificial Intelligence Institute

**Part B: Signatures**

**Unit Chair(s): (If recruiting staff and students from individual units)**

Name:

Signature:

Date:

**Course Coordinator: (Please include if recruiting staff and students from course/s)**

Name:

Signature:

Date:

**Head of School/Department (or nominee): (If recruiting staff or students from one School/Department)**

Name: SIMON PARKER???

Signature:

Date:

**Pro Vice Chancellor (PVC) or Associate Dean of Research (ADR): (If recruiting staff and students from two or more Schools in the one Faculty. If recruiting staff from more than one faculty, please provide a signature for each faculty PVC or ADR.)**

Name:

Signature:

Date:

**Dean of Students: (If you are recruiting students from 2 or more Faculties or Institutes)**

Name:

Signature:

Date:

**Important:** When submitting your form for signing by the appropriate authority above, you must attach the full ethics application including all recruitment materials, PLSs, and survey questionnaires (including demographic questionnaires) or interview/focus group topics/questions.

Once signed, please submit this form with your ethics application via email to:

**DUHREC** (for higher than low risk ethics applications): [research-ethics@deakin.edu.au](mailto:research-ethics@deakin.edu.au)

For low risk ethics applications, please submit to your **Faculty Human Ethics Advisory Group (HEAG)**:

Arts and Education HEAG:	<a href="mailto:aeeethics@deakin.edu.au">aeeethics@deakin.edu.au</a>
Business & Law HEAG:	<a href="mailto:b lethics@deakin.edu.au">b lethics@deakin.edu.au</a>
Health HEAG:	<a href="mailto:health-ethics@deakin.edu.au">health-ethics@deakin.edu.au</a>
Science, Engineering & Built Environment HEAG:	<a href="mailto:sciethic@deakin.edu.au">sciethic@deakin.edu.au</a>

**For more information please contact [research-ethics@deakin.edu.au](mailto:research-ethics@deakin.edu.au) or call (03) 9251 7123 or (03) 5227 3012.**

Please note: [Deakin's Research Conduct Policy](#) states that where more than 200 students are to be surveyed, researcher/s will also consult with the Strategic Intelligence and Planning Unit (SIPU) prior to scheduling the research. Please email [dusurvey@deakin.edu.au](mailto:dusurvey@deakin.edu.au) with details of the proposed scheduling of your research. If you have any questions, please call +61 3 5227 8029.

Please note: if the hyperlinks in this form result in an error message, return to the form and:

1. *right click on the hyperlink*
2. *click on Edit Hyperlink*
3. *copy the URL to your browser.*

Deakin University is collecting your personal information on this form for the primary purpose of processing your human research ethics application. It will also use this information for monitoring your compliance with the approved protocol. For these purposes, Deakin may also provide this information to potential research participants, past or current research participants, or other interested parties in your research. You are not required to provide the information requested, however if the information is not provided, Deakin may not be able to process your ethics application. Deakin manages personal information it holds, including requests by individuals for access to their personal information, in accordance with the Privacy and Data Protection Act 2014 (Vic). Deakin's Privacy Policy may be viewed on Deakin's [Policy Library](#). Information on privacy at Deakin is available at <http://www.deakin.edu.au/footer/privacy>. Questions about privacy may be directed to the Privacy Officer on (03) 5227 8524 or by email to [privacy@deakin.edu.au](mailto:privacy@deakin.edu.au).