



QE + RE
Y

How Can ~~Software Engineering~~ solve the "AI Dilemma"?

Walid Maalej, 2023

About myself

Research Blueprint

Data- and Human-Centered Systems

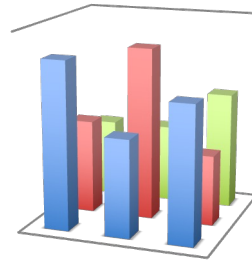
User- and value-driven software design



Software knowledge sharing



Empiricism, analytics and RecSys for SE



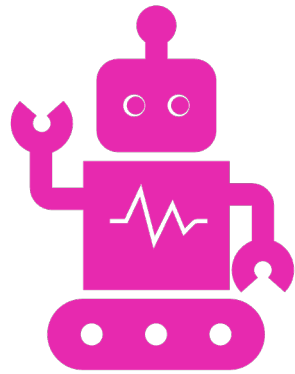
AI Engineering



Software and Requirements Engineering

A scenic landscape featuring rolling green hills under a cloudy sky. A stone path leads from the foreground towards a large, rounded hill in the distance. The foreground is filled with tall grasses and small white flowers. A person is visible sitting on the path in the middle ground.

How Can Software Engineering
solve the “AI Dilemma”?



**“Only 53% of AI prototypes actually make it into production.”
[Gartner]**

Google Translate

WIRED BACKCHANNEL

DETECT LANG

WILL KNIGHT BUSINESS AUG 10, 2022

Sloppy Use of a 'Reproducib

AI hype has researchers in field that they don't always unders

f t e

ENC

He is she i he is She's he is Why

cookies on Forbes

Get WIRED for just \$29.9

in Some hav enforcem

The BAILLIE GIFFORD PRIZE FOR NON-FICTION 2018

SHORTLISTED

How to Be Human

Hello

in the Age of

the Machine

'Stylish, thoughtful . . . One of the best books yet written on data and algorithms' *The Times*

world.

HANNAH FRY

SUBSCRIBE Sign In

g robot cars in y

f t e

9

als Search Login

la nce?

Privacy Settings

Many AI models fail after deployment

- Acceptance / trust issues
- Bad integration into user's workflows
- Accuracy issues because context or environment not controllable anymore
- Unrealistic assumptions about the domain
- Legal, technical or social requirements not met

A Human-Centered Evaluation of a Deep Learning System Deployed in Clinics for the Detection of Diabetic Retinopathy

Emma Beede
Google Health
Palo Alto, CA
embeede@google.com

Elizabeth Baylor
Google Health
Palo Alto, CA
ebaylor@google.com

Fred Hersch
Google Health
Singapore
fredhersch@google.com

Anna Iurchenko
Google Health
Palo Alto, CA
annaiu@google.com

Lauren Wilcox
Google Health
Palo Alto, CA
lwilcox@google.com

Paisan Ruamviboonsuk
Rajavithi Hospital
Bangkok, Thailand
paisan.tr@gmail.com

Laura M. Vardoulakis
Google Health
Palo Alto, CA
lauravar@google.com

ABSTRACT

Deep learning algorithms promise to improve clinician workflows and patient outcomes. However, these gains have yet to be fully demonstrated in real world clinical settings. In this paper, we describe a human-centered study of a deep learning system used in clinics for the detection of diabetic eye disease. From interviews and observation across eleven clinics in Thailand, we characterize current eye-screening workflows, user expectations for an AI-assisted screening process, and post-deployment experiences. Our findings indicate that several socio-environmental factors impact model performance, nursing workflows, and the patient experience. We draw on these findings to reflect on the value of conducting human-centered evaluative research alongside prospective evaluations of model accuracy.

Author Keywords

human-centered AI, health, deep learning, diabetes

CCS Concepts

•Human-centered computing → Empirical studies in HCI;

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

CHI'20, April 25–30, 2020, Honolulu, HI, USA

© 2020 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-6708-0/20/04.

DOI: <https://doi.org/10.1145/3313831.3376718>

INTRODUCTION

Diabetes is a growing problem around the world, including in Southeast Asia [39]. As of 2016, 9.6% of Thailand's population was living with diabetes, comparable to 9.1% of the population in the United States [41, 40]. With diabetes comes complications, including diabetic retinopathy (DR), a condition caused by chronically high blood sugar that damages blood vessels in the retina, the thin layer at the back of the eye responsible for sensing light and sending signals to the brain. These blood vessels can leak or hemorrhage, causing vision distortion or loss. DR is one of the leading causes of vision impairment in the world [29], and causes 5% of cases of blindness worldwide, excluding refractive errors [38]. In Thailand, 34% of patients with diabetes have low vision or blindness in either eye [23].

In early stages of DR, a patient often has no symptoms, making it important for people living with diabetes to be screened regularly, as this is the stage in which damage can be reversed—progression of DR can be stopped or significantly reduced by blood sugar control. Early detection is key to initiate timely treatment and mitigate the risk of blindness.

Since 2013, the Ministry of Health in Thailand has set a goal to screen 60% of its diabetic population for diabetic retinopathy (DR). However, reaching this goal is a challenge due to a shortage of clinical specialists. In Thailand, there are 1500 ophthalmologists, including 200 retinal specialists, who provide ophthalmic care to approximately 4.5 million patients with diabetes [23]—a ratio of about 1:3000, about double of what it is in the United States [3, 11]. The shortage of doctors limits the ability to screen patients and also creates a treatment backlog for those found to have DR. As a result, nurses

Relevant people



Francesco Poch...

@Fra_Pochetti

Follow


AWS ML Hero. MLE @boltapp.
Failed Chemist. Blogging about
my ML/DL journey. IceVision
core-dev.



Francesco Pochetti


@Fra_Pochetti

All ML projects which turned into a disaster in my career have a single common point:

 I didn't understand the business context first, got over-excited about the tech, and jumped into coding too early.

7:08 PM · Mar 12, 2022 · Twitter Web App

293 Retweets **37** Quote Tweets **1,769** Likes

A stack of several books with various colored covers (red, blue, green, brown) is positioned on the right side of the slide. The books are stacked vertically, with the spines facing left. The background is a solid teal color.

Some problems are
emerging simply
due to the lack of
RE/SE skills!

This is rather "easy" to solve!

SE & AI: Long co-existence & co-development

- 1960's: Symbolic AI / Reasoning / Logic Programming
- 1980's: Rule-based Systems / Blackboard Patterns
- 1990's Reasoning in RE / (Goal) Modeling
- Last decade: Mega Machine Learning (ML) trend through availability of data:
 - SE has "ridden" this mega trend
 - Open-source data → Mining Software Repos
 - RecSys in software engineering from 2010+
 - NLP for SE (now also Computer Vision for SE)
- 2022: CAIN Conference @ ICSE





While “surfing the ML wave”, we neglected our core strengths to help AI!

AI-based systems are different

- Design system (behavior) and “data scheme”

Users will “feed” system with data



- Here’s the data, build a system around it! It shall tolerate and adapt to user “errors”; but no bias please!

- Observable errors are debug'able through decision paths



- Black-boxed tacit errors
Unclear/ untraceable decision logic

- “I trust you know what’s reasonable/feasible for my requirements”



- “We need an AI for this.
Can’t we do all this with ChatGPT”

Then...

Now...

Selection of recent general studies about engineering ML systems

ICSE'19

Software Engineering for Machine Learning: A Case Study

Saleema Amershi
Microsoft Research
Redmond, WA USA
samershi@microsoft.com

Andrew Begel
Microsoft Research
Redmond, WA USA
andrew.begel@microsoft.com

Christian Bird
Microsoft Research
Redmond, WA USA
cbird@microsoft.com

Robert DeLine
Microsoft Research
Redmond, WA USA
rdeline@microsoft.com

Harald Gall
University of Zurich
Zurich, Switzerland
gall@ifi.uzh.ch

Ece Kamar
Microsoft Research
Redmond, WA USA
eckamar@microsoft.com

Nachiappan Nagappan
Microsoft Research
Redmond, WA USA
nachin@microsoft.com

Besmira Nushi
Microsoft Research
Redmond, WA USA
besmira.nushi@microsoft.com

Thomas Zimmermann
Microsoft Research
Redmond, WA USA
tzimmer@microsoft.com

TSE'19

How does Machine Learning Change Software Development Practices?

Zhiyuan Wan, Xin Xia, David Lo and Gail C. Murphy

Abstract—Adding an ability for a system to learn inherently adds uncertainty into the system. Given the rising popularity of incorporating machine learning into systems, we wondered how the addition alters software development practices. We performed a mixture of qualitative and quantitative studies with 14 interviewees and 342 survey respondents from 26 countries across four continents to elicit significant differences in software development practices. We performed a mixture of qualitative and quantitative studies with 14 interviewees and 342 survey respondents from 26 countries across four continents to elicit significant differences in software development practices. We performed a mixture of qualitative and quantitative studies with 14 interviewees and 342 survey respondents from 26 countries across four continents to elicit significant differences in software development practices.

ICSE'22

Collaboration Challenges in Building ML-Enabled Systems: Communication, Documentation, Engineering, and Process

Nadia Nahar
nadian@andrew.cmu.edu
Carnegie Mellon University
Pittsburgh, PA, USA

Shurui Zhou
University of Toronto
Toronto, Ontario, Canada

Grace Lewis
Carnegie Mellon Software Engineering Institute
Pittsburgh, PA, USA

Christian Kästner
Carnegie Mellon University
Pittsburgh, PA, USA

ABSTRACT
The introduction of machine learning (ML) components in software projects has created the need for software engineers to collaborate with data scientists and ML experts. This paper reports on the challenges of building ML-enabled systems, focusing on communication, documentation, engineering, and process. We conducted a series of focus group discussions and a survey with 342 software engineers and data scientists to understand these challenges. Our findings indicate that building ML-enabled systems is more challenging than building traditional software systems, and that collaboration between software engineers and data scientists is essential for success. We discuss the challenges of building ML-enabled systems and provide recommendations for software engineers and data scientists to overcome these challenges.

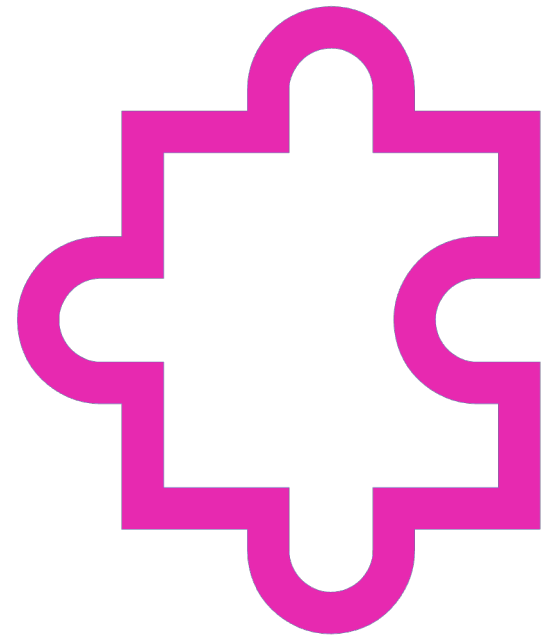
“The randomness... complicates **testing**: random data, random observation order, random initialization for weights, random batches... random optimizations in different ... libraries”

“**Requirements** are more uncertain for ML systems than non-ML systems”

“**test** case generation for ML systems is more challenging, compared to non-ML systems”

~“the product team must involve the data science team in the negotiation of **requirements** to avoid unrealistic expectations”

Tailoring Software and Requirement Engineering for AI





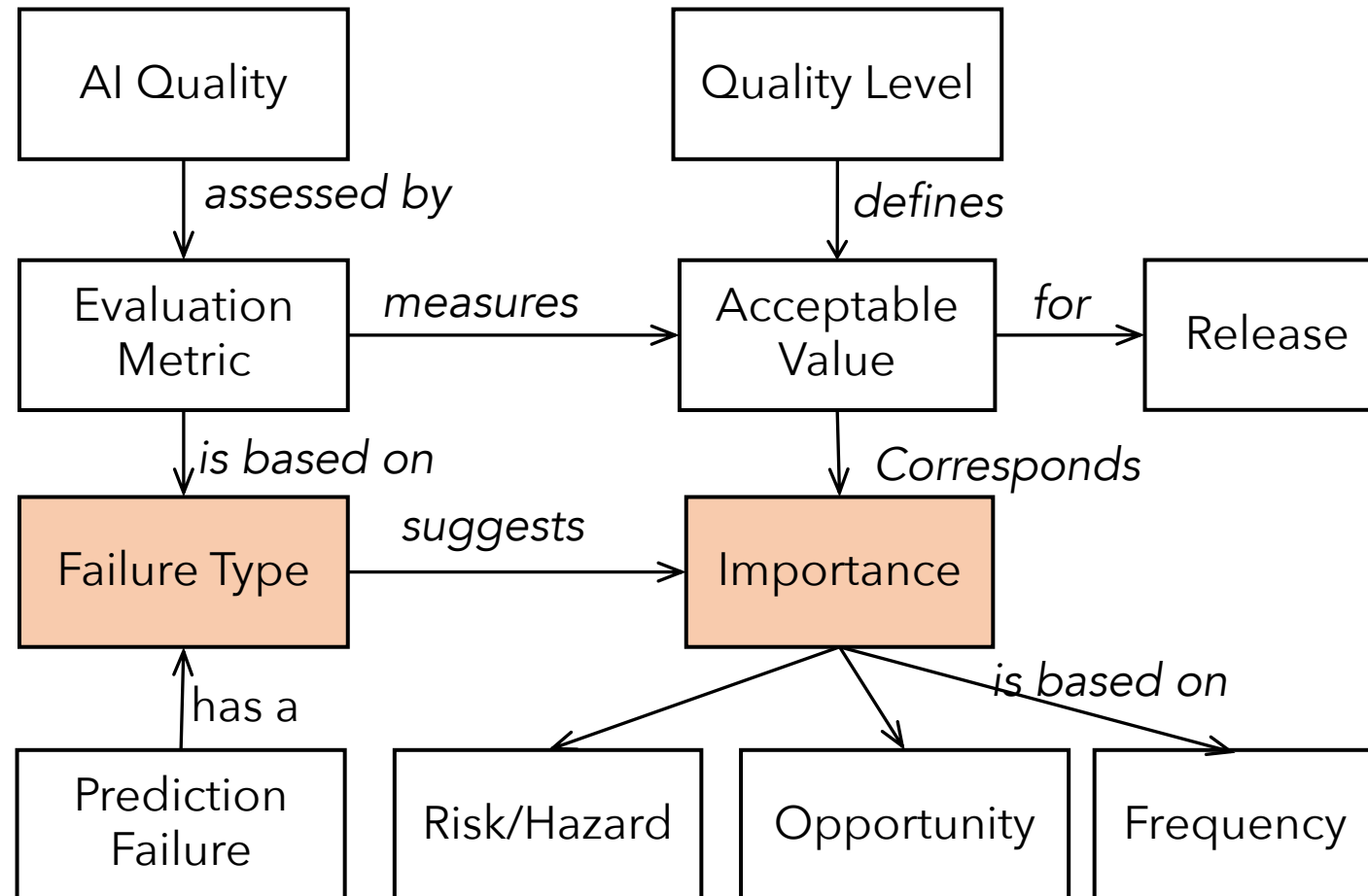
1. Move from
correct to
acceptable
requirements

Acceptable AI requirements



-
- Instead of "as high as possible"
 - Depending on domain, use case, requirements, release plan...
 - Qualitative and quantitative **failure analysis**
 - What types or failures/ rates are acceptable, when, why, for whom...?

Metamodel for discussing and specifying acceptable quality requirements





SUPPLIER'S NAME

MODEL IDENTIFIER



Negotiate and specify **standard** levels instead of values

-
- Deal with technical dept and **uncertainty**
 - Allow comparison
 - Manage **expectation**

$$F = G \frac{m_1 m_2}{d^2}$$

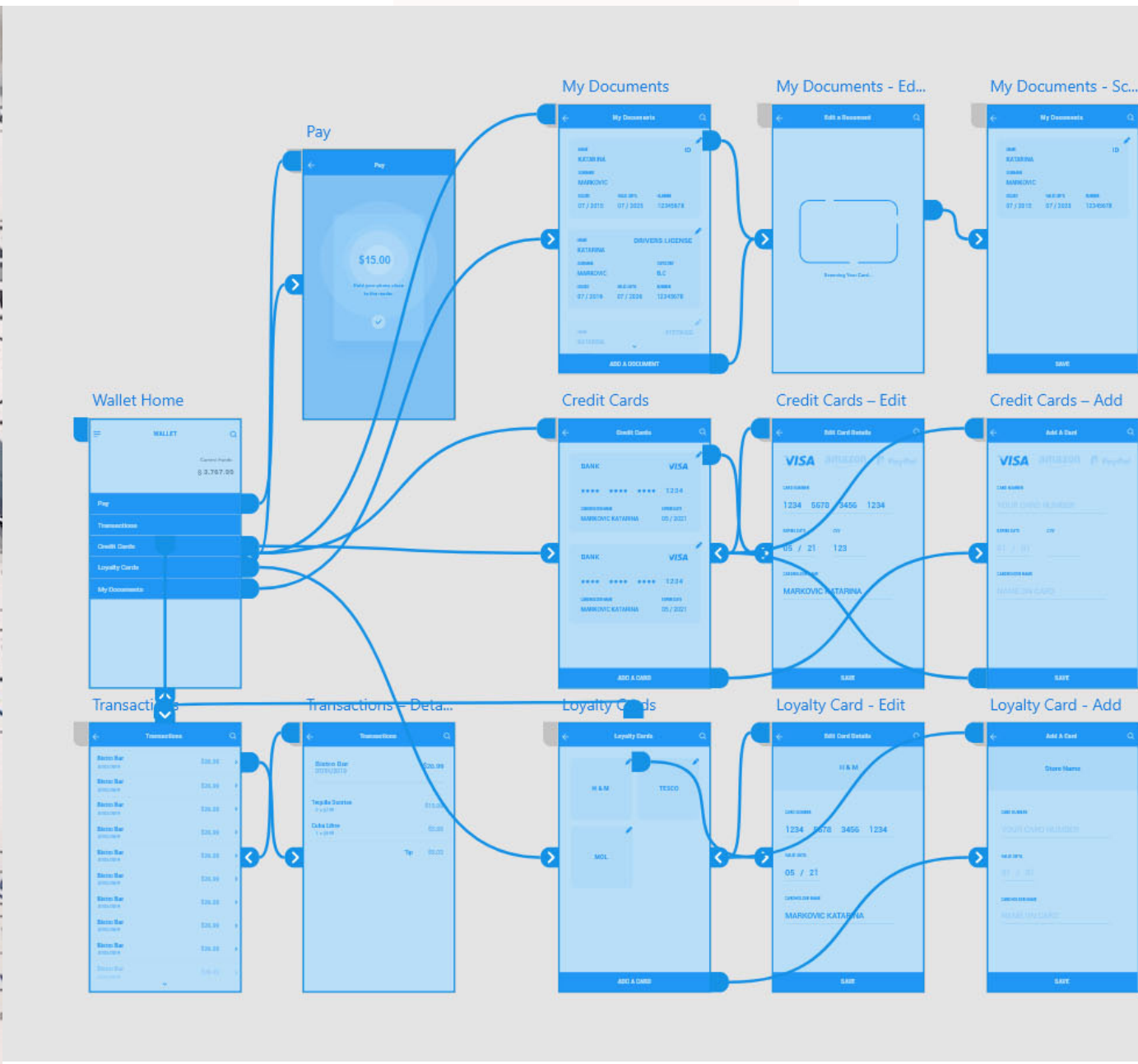
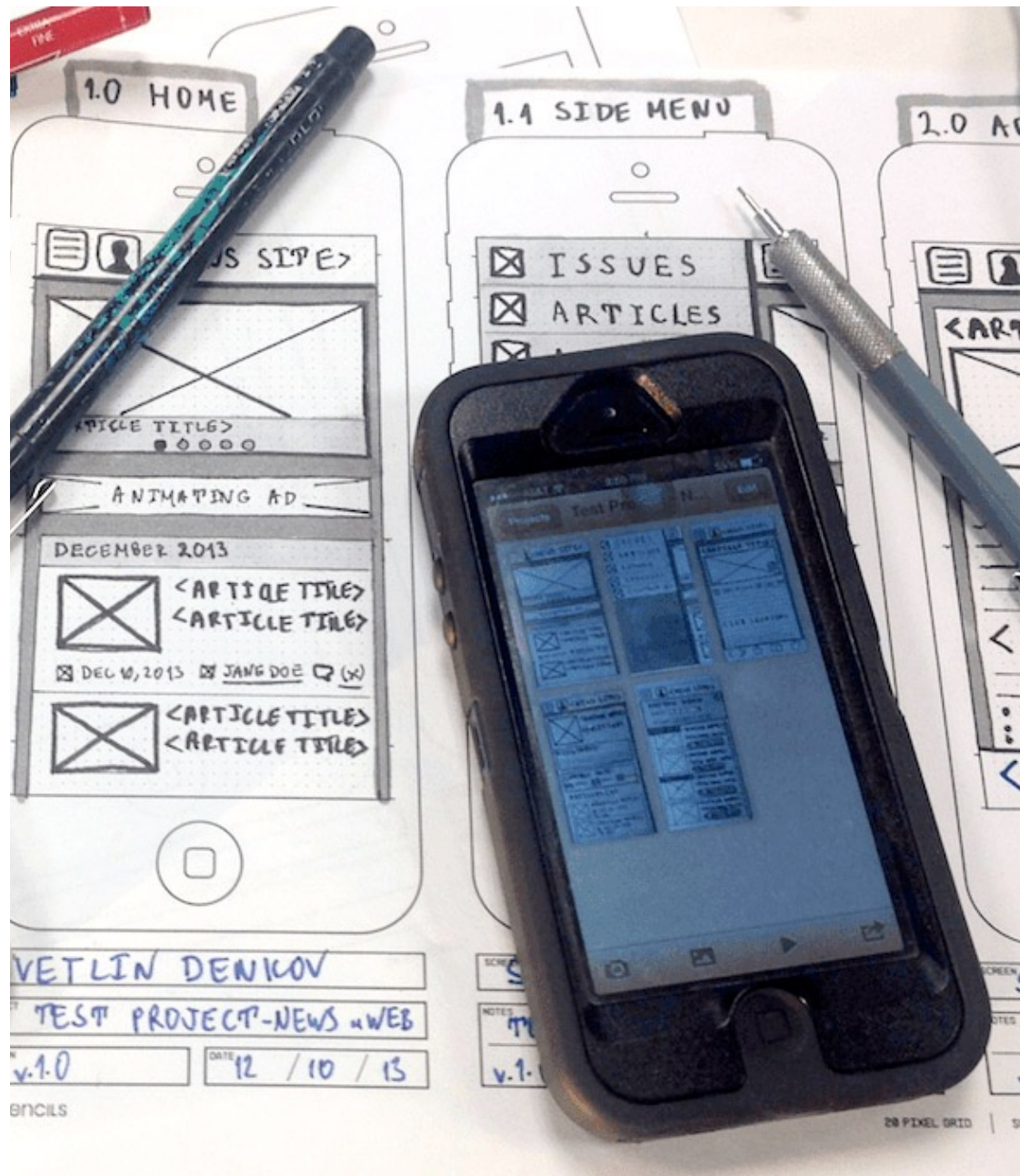
$$\phi(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$i\hbar \frac{\partial}{\partial t} \psi = \hat{H} \psi$$

2. Prototyping is different in AI

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}$$

$$\frac{df}{dt} = \lim_{h \rightarrow 0} \frac{f(t+h) - f(t)}{h}$$



Prototyping in AI \approx Computational Notebooks

The image displays two overlapping Jupyter Notebook windows. The foreground window, titled "Exploring the Lorenz System", shows the following content:

Exploring the Lorenz System

In this Notebook we explore the [Lorenz system](#) of differential equations:

$$\begin{aligned}\dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy\end{aligned}$$

This is one of the classic systems in non-linear differential equations. It exhibits a range of complex behaviors as the parameters (σ, β, ρ) are varied, including what are known as *chaotic solutions*. The system was originally developed as a simplified mathematical model for atmospheric convection in 1963.

```
In [7]: interact(Lorenz, N=fixed(10), angle=(0.,360.),
                sigma=(0.0,50.0), beta=(0.,5), rho=(0.0,50.0))
```

angle 308.2
max_time 12
 σ 10
 β 2.6
 ρ 28

The background window shows the "Welcome to the Jupyter Notebook Server" page, which includes a warning message: "WARNING: Don't rely on this server. Your server is hosted that..." and instructions on how to run Python code.



> Data- + User- Centered Prototyping

-
- Design AI with human (users) in the loop!
 - E.g. using prediction uncertainty

Example of Human (Users) in the Loop:

Moderated classifiers as deployment pattern

$$f_{mod}^{\omega}(x) := \begin{cases} f^{\omega}(x) \\ o_H(x) \end{cases}$$

Human oracle

if $u[y|x, \omega] \leq \vartheta_u$
else

Uncertainty
measure

Threshold based
on a moderation
strategy

3. RE += Data Requirements Engineering

The screenshot shows a YouTube video player. The video title is "Iguana Detection Example". The slide content includes a large image of two iguanas on the left and three smaller images on the right, each with a red bounding box around one of the iguanas. Below the large image is the text: "Labeling instruction: Use bounding boxes to indicate the position of iguanas". A pink thought bubble is overlaid on the right side of the slide, containing the text "Quality Data instead of Big Data". The video player interface shows a progress bar at 14:15 / 1:00:10 and a "Zoom" watermark.

From model-centric to data-centric AI

- Data processes
- Data smells
- Data labeling
- Quality of labels & labelers
- Quality of single labels vs the entire dataset



4. Embedding responsible AI **terminology** into the engineering workflows



INDEPENDENT
**HIGH-LEVEL EXPERT GROUP ON
ARTIFICIAL INTELLIGENCE**
SET UP BY THE EUROPEAN COMMISSION



**ETHICS GUIDELINES
FOR TRUSTWORTHY AI**



1. Human agency and oversight
2. Technical robustness and safety
3. Privacy and data governance transparency
4. Diversity, non-discrimination and fairness
5. Environmental and societal well-being and
6. Accountability



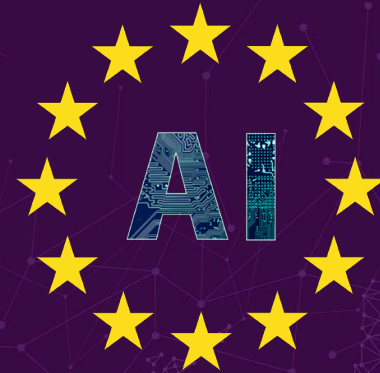
INDEPENDENT
**HIGH-LEVEL EXPERT GROUP ON
ARTIFICIAL INTELLIGENCE**
SET UP BY THE EUROPEAN COMMISSION



**ETHICS GUIDELINES
FOR TRUSTWORTHY AI**



INDEPENDENT
**HIGH-LEVEL EXPERT GROUP ON
ARTIFICIAL INTELLIGENCE**
SET UP BY THE EUROPEAN COMMISSION



THE ASSESSMENT LIST FOR
TRUSTWORTHY ARTIFICIAL
INTELLIGENCE (ALTAI)
for self assessment

The Role of Linguistic Relativity on the Identification of Sustainability Requirements: An Empirical Study

Yen Dieu Pham, Abir Bouraffa, Marleen Hillen and Walid Maalej
University of Hamburg
Hamburg, Germany
{yen.pham-1, abir.bouraffa, marleen.hillen, walid.maalej}@uni-hamburg.de

Abstract—Linguistic-Relativity-Theory states that language and its structure influence people’s world view and cognition. We investigate how this theory impacts the identification of requirements in practice. To this end, we conducted two controlled experiments with 101 participants. We randomly showed participants a set of requirements dimensions (i.e. a language structure) either with a focus on software quality or on sustainability and asked them to identify the requirements for a grocery shopping app according to these dimensions. Participants of the control group were not given any dimensions. The results show that the use of requirements dimensions significantly increases the number of identified requirements in comparison to the control group. Furthermore, participants who were given the sustainability dimensions identified more sustainability requirements. In follow up interviews with 16 practitioners, the interviewees reported benefits of the dimensions such as a holistic guidance but were also concerned about the customers acceptance. Furthermore, they stated challenges of implementing sustainability dimensions in the daily business but also suggested solutions like establishing sustainability as a common standard. Our study indicates that carefully structuring requirements engineering along sustainability dimensions can guide development teams towards considering and ensuring software sustainability.

Index Terms—software sustainability, requirements engineering, requirements dimension, interdisciplinary design

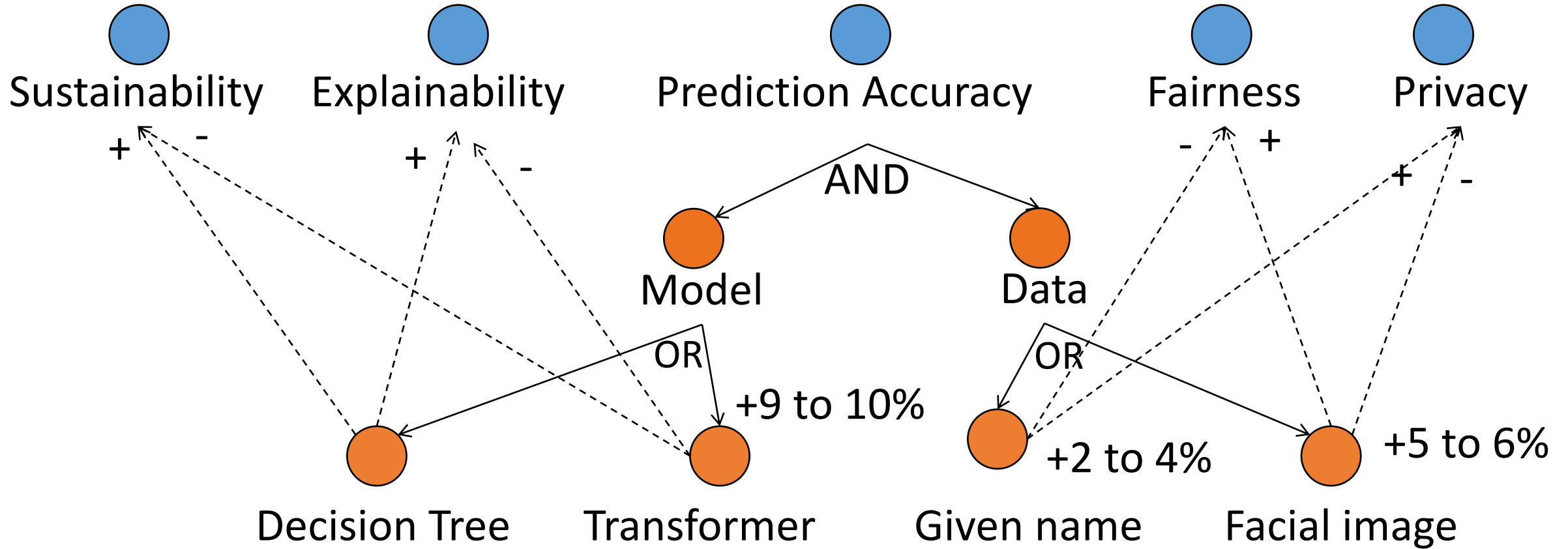
regard to sustainability requirements. The basic assumption is that requirements dimensions represent structures of the language used during the requirements identification and resulting requirements represent the “worldview and behaviour” of the stakeholders involved in this task.

We report on two online experiments, where we asked participants to identify requirements for a grocery shopping app according to a randomly assigned set of requirements dimensions or no dimensions. We focused on two sets of dimensions. The first set is derived from the ISO/IEC 25010:2011 standard on Systems and Software Quality [5]. However, this set does not take sustainability dimensions into account [6]. Therefore we derived a second set from the framework ShapeRE [7], which includes the Karlskrona Manifesto’s [8] sustainability dimensions such as social, economic, ecological, individual and technical. We analyzed the requirements identified by the participants to determine whether the dimensions lead i.) to more requirements in general, and ii) to more sustainability requirements. We define a sustainability requirement as a requirement that can be matched to at least one of the sustainability dimensions of the Karlskrona Manifesto. Since in practice Linguistic-Relativity might be just one of

A group of business professionals in a meeting. One person is pointing at a tablet displaying a chart. There are coffee cups on the table. The scene is brightly lit, likely in a modern office.

5. TRADEOFF ANALYSIS FOR RESPONSIBLE AI

Tradeoff analysis for AI systems

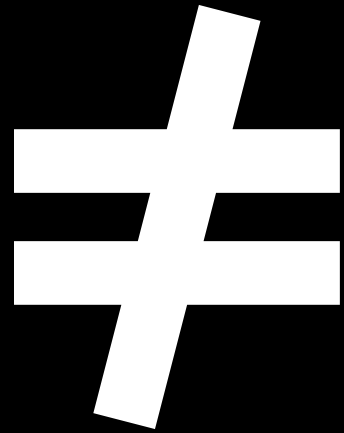


[Strubell et al ACL'19][Chazette et al. RE'21]

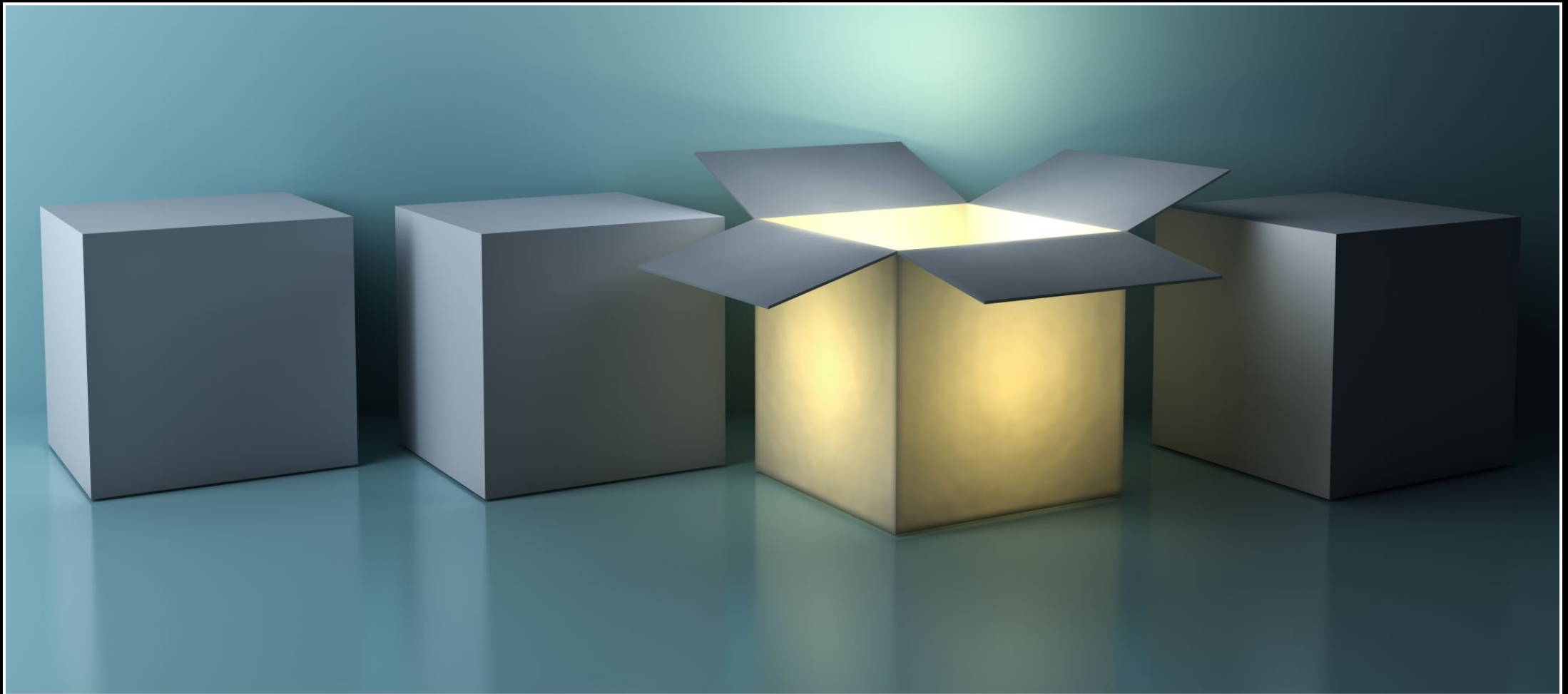
A close-up photograph of a laboratory setting. A pipette tip is positioned above a multi-well plate, dispensing a small amount of liquid. The background is blurred, showing a person in a lab coat and other laboratory equipment. The text is overlaid on an orange rectangular background.

6. Requirements as foundation for AI testing and quality assurance

**Software
Testing**



**Model
Testing**



The interplay between black-box and white-box testing

Two black-box testing ideas

ACL'20

Beyond Accuracy: Behavioral Testing of NLP Models with CHECKLIST

Marco Tulio Ribeiro¹ Tongshuang Wu² Carlos Guestrin² Sameer Singh³

¹Microsoft Research ²University of Washington ³University of California, Irvine
marcotcr@gmail.com {wtshuang, guestrin}@cs.uw.edu sameer@uci.edu

Abstract

Although measuring held-out accuracy has been the primary approach to evaluate generalization, it often overestimates the performance of NLP models, while alternative approaches for evaluating models either focus on individual tasks or on specific behaviors. Inspired by principles of behavioral testing in software engineering, we introduce CHECKLIST, a task-agnostic methodology for testing NLP models. CHECKLIST includes a matrix of general linguistic *capabilities* and *test types* that facilitate comprehensive test ideation, as well as a software tool to generate a large and diverse number of test cases quickly. We illustrate the utility of CHECKLIST with tests for three tasks, identifying critical failures in both commercial and state-of-art models. In a user study, a team responsible for a commercial sentiment analysis model found new and actionable bugs in an extensively tested model. In another user study, NLP practitioners with CHECKLIST created twice as many tests, and found almost three times as many bugs as users without it.

1 Introduction

One of the primary goals of training NLP models is generalization. Since testing “in the wild” is expensive and does not allow for fast iterations, the standard paradigm for evaluation is using train-validation-test splits to estimate the accuracy of the model, including the use of leader boards to track progress on a task (Rajpurkar et al., 2016). While performance on held-out data is a useful indicator, held-out datasets are often not comprehensive, and contain the same biases as the training data (Rajpurkar et al., 2018), such that real-world performance may be overestimated (Patel et al.,

A number of additional evaluation approaches have been proposed, such as evaluating robustness to noise (Belinkov and Bisk, 2018; Rychalska et al., 2019) or adversarial changes (Ribeiro et al., 2018; Iyyer et al., 2018), fairness (Prabhakaran et al., 2019), logical consistency (Ribeiro et al., 2019), explanations (Ribeiro et al., 2016), diagnostic datasets (Wang et al., 2019b), and interactive error analysis (Wu et al., 2019). However, these approaches focus either on individual tasks such as Question Answering or Natural Language Inference, or on a few capabilities (e.g. robustness), and thus do not provide comprehensive guidance on how to evaluate models. Software engineering research, on the other hand, has proposed a variety of paradigms and tools for *testing* complex software systems. In particular, “behavioral testing” (also known as black-box testing) is concerned with testing different capabilities of a system by validating the input-output behavior, without any knowledge of the internal structure (Beizer, 1995). While there are clear similarities, many insights from software engineering are yet to be applied to NLP models.

In this work, we propose CHECKLIST, a new evaluation methodology and accompanying tool¹ for comprehensive behavioral testing of NLP models. CHECKLIST guides users in what to test, by providing a list of linguistic *capabilities*, which are applicable to most tasks. To break down potential capability failures into specific behaviors, CHECKLIST introduces different *test types*, such as prediction invariance in the presence of certain perturbations, or performance on a set of “sanity checks.” Finally, our implementation of CHECKLIST includes multiple *abstractions* that help users generate large numbers of test cases easily, such as templates, lexi-

TSE'21

BiasFinder: Metamorphic Test Generation to Uncover Bias for Sentiment Analysis Systems

Muhammad Hilmi Asyrofi, Zhou Yang, Imam Nur Bani Yusuf, Hong Jin Kang, Ferdian Thung and David Lo

Abstract—Artificial Intelligence (AI) software systems, such as Sentiment Analysis (SA) systems, typically learn from large amounts of data that may reflect human biases. Consequently, the machine learning model in such software systems may exhibit unintended demographic bias based on specific characteristics (e.g., gender, occupation, country-of-origin, etc.). Such biases manifest in an SA system when it predicts a different sentiment for similar texts that differ only in the characteristic of individuals described. Existing studies on revealing bias in SA systems rely on the production of sentences from a small set of short, predefined templates. To address this limitation, we present BiasFinder, an approach to discover biased predictions in SA systems via metamorphic testing. A key feature of BiasFinder is the automatic curation of suitable templates based on the pieces of text from a large corpus, using various Natural Language Processing (NLP) techniques to identify words that describe demographic characteristics. Next, BiasFinder instantiates new text from these templates by filling in placeholders with words associated with a class of a characteristic (e.g., gender-specific words such as female names, “she”, “her”). These texts are used to tease out bias in an SA system. BiasFinder identifies a bias-uncovering test case (BTC) when it detects that the SA system exhibits demographic bias for a pair of texts, i.e., it predicts a different sentiment for texts that differ only in words associated with a different class (e.g., male vs. female) of a target characteristic (e.g., gender). Our empirical evaluation showed that BiasFinder can effectively create a larger number of fluent and diverse test cases that uncover various biases in an SA system.

Index Terms—sentiment analysis, test case generation, metamorphic testing, bias, fairness bug

1 INTRODUCTION

MANY modern software systems employ AI systems to make decisions. In AI systems, fairness is considered to be an important non-functional requirement; bias in AI systems, reflecting discriminatory behavior towards unprivileged groups, can lead to real-world harms. To address this requirement, software engineering research techniques, such as test generation, have been applied to detect bias [1]–[5]. While various techniques have been proposed for test generation of machine learning systems [1]–[4], there have been limited studies on detecting biases in text-based machine learning systems [5]. Text-based ML systems have numerous applications, for example, NLP techniques have been used for Sentiment Analysis (SA). It is, therefore, important that biases in these systems can be detected before these systems are deployed.

SA systems are used to measure the attitudes and affects in text reviews about an entity, such as a movie or a news article [6], [7]. In this work, we focus on uncovering bias in SA for three reasons:

Firstly, SA has widespread adoption in many domains [8], [9], including politics [10], [11], finance [12]–[15], business [16], education [17]–[19], and healthcare [20]–[22]. In the research community, SA continues to be widely studied [23]–[28]. In the industry, many companies, such

as Microsoft¹ and Google², have developed and provided APIs for software developers to access SA capabilities. This suggests the prevalence of SA in real-life applications. As a result, bias in SA systems can have a big impact on society.

Secondly, SA has generalizability to other areas of NLP. Some NLP researchers have considered SA to be “mini-NLP” [9], as research on SA techniques builds on top of a wide range of topics and tasks in the NLP domain. Cambria et al. [29] argues that SA is a problem with a composite nature, requiring 15 more fundamental NLP problems to be addressed at the same time. Therefore, we believe that tackling bias in SA is a suitable first step that could lead to a more general approach to detect bias in textual data.

Thirdly, due to the importance of SA systems, there are many recent research works [5], [30]–[34] that focus solely on the fairness issues in SA systems. Although these works are not guaranteed to be fully generalizable to all kinds of NLP systems, the importance and wide applicability of SA justify the need of fairness studies that focus on it.

Modern SA models have outstanding performance on benchmark datasets, which demonstrates their effectiveness. However, there has been a growing understanding in both the Software Engineering [3] and Artificial Intelligence [5] research communities that it is important to study non-functional requirements, such as fairness, which have been overlooked. AI systems learn from data generated by humans. In the case of SA, the training data is typically a dataset of human-written reviews. The training data may

arXiv:2102.01859v2 [cs.SE] 5 Oct 2021

¹ M.H. Asyrofi, Z. Yang, I.N.B. Yusuf, H.J. Kang, F. Thung, D. Lo are with the School of Computing and Intelligent Systems, Singapore

Checklist **capabilities** of models based on templates and human creativity

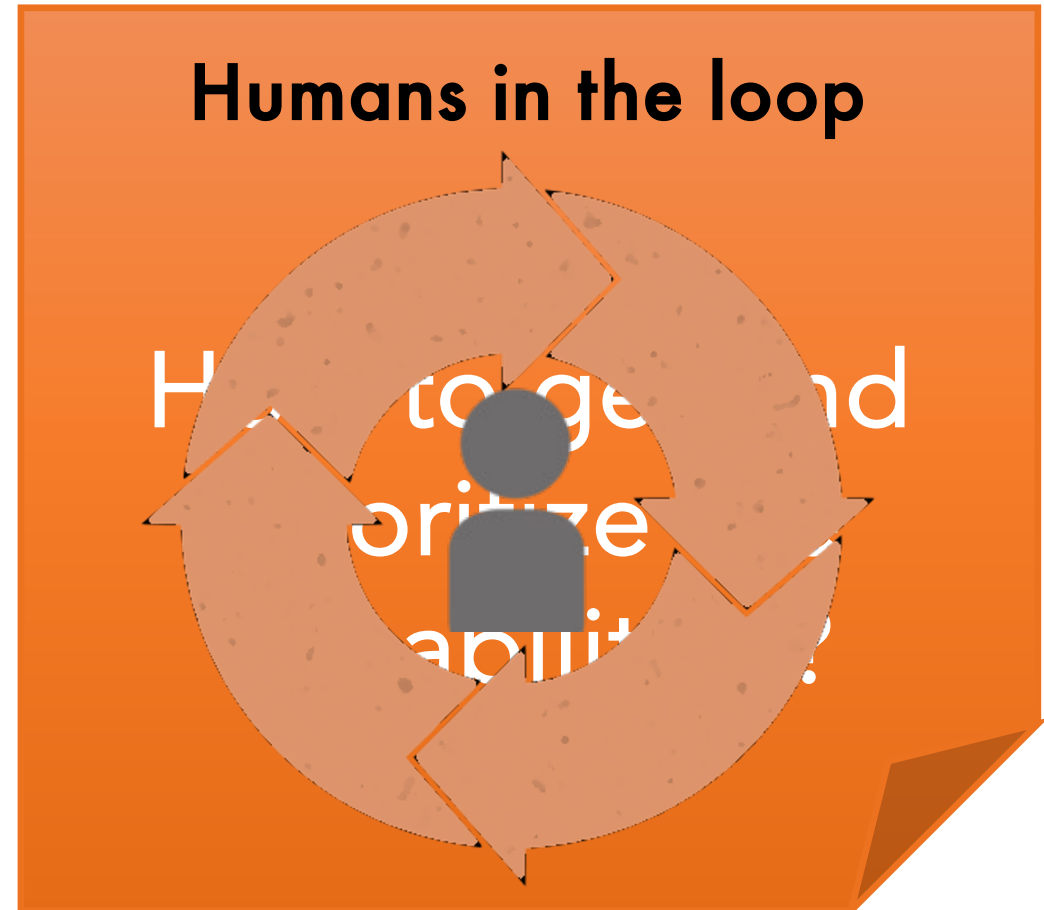
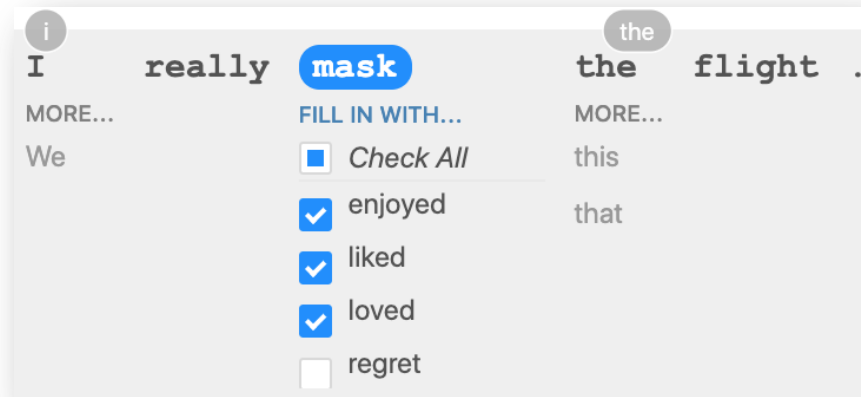
NER

INV: Switching locations should not change predictions

INV: Switching person names should not change predictions

Temporal

MFT: Sentiment change over time, present should prevail



Metamorphic testing for NLP models

Generate templates (and then mutants) for gender, occupation, and county-of-origin biases

Original Text

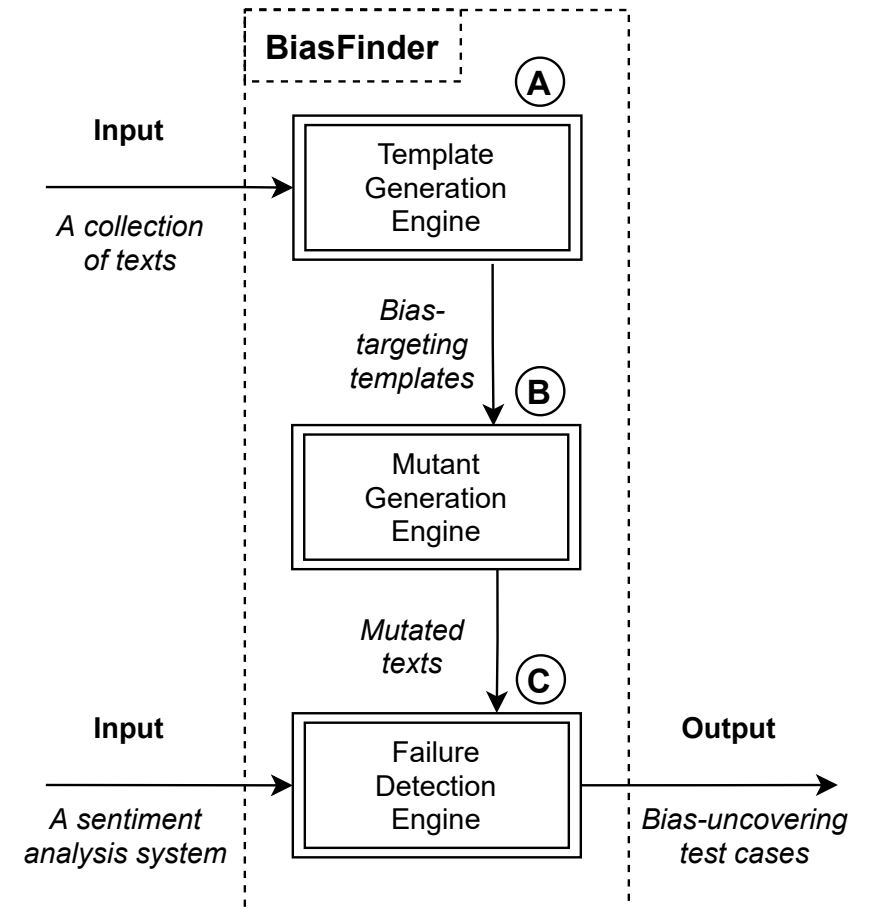
It seems that Jake with all his knowledge of the great outdoors didn't realize the danger! He enters a mine shaft that's leaking with dangerous gas!

Mutated Text

It seems that Julia with all her knowledge of the great outdoors didn't realize the danger! She enters a mine shaft that's leaking with dangerous gas!

[Hilmi Asyrofi et al., TSE'21]

When to generate test cases and when to specify with "human intelligence"?



Two white-box testing ideas

ASE'20

Cats Are Not Fish: Deep Learning Testing Calls for Out-Of-Distribution Awareness

David Berend
Nanyang Technological University,
Singapore

Xiaofei Xie*
Nanyang Technological University,
Singapore

Lei Ma
Kyushu University, Japan

Lingjun Zhou
Tianjin University, China

Yang Liu
Nanyang Technological University,
Zhejiang Sci-Tech University, China

Chi Xu
Singapore Institute of Manufacturing
Technology, A*Star

Jianjun Zhao
Kyushu University, Japan

ABSTRACT

As Deep Learning (DL) is continuously adopted in many industrial applications, its quality and reliability start to raise concerns. Similar to the traditional software development process, testing the DL software to uncover its defects at an early stage is an effective way to reduce risks after deployment. According to the fundamental assumption of deep learning, the DL software does not provide statistical guarantee and has limited capability in handling data that falls outside of its learned distribution, i.e., out-of-distribution (OOD) data. Although recent progress has been made in designing novel testing techniques for DL software, which can detect thousands of errors, the current state-of-the-art DL testing techniques usually do not take the distribution of generated test data into consideration. It is therefore hard to judge whether the “identified errors” are indeed meaningful errors to the DL application (i.e., due to quality issues of the model) or outliers that cannot be handled by the current model (i.e., due to the lack of training data). To fill this gap, we take the first step and conduct a large scale empirical study, with a total of 451 experiment configurations, 42 deep neural networks (DNNs) and 1.2 million test data instances, to investigate and characterize the impact of OOD-awareness on DL testing. We further analyze the consequences when DL systems go into production by evaluating the effectiveness of adversarial retraining with distribution-aware errors. The results confirm that introducing data distribution awareness in both testing and enhancement phases outperforms distribution unaware retraining by up to 21.5%.

CCS CONCEPTS

• Computing methodologies → Neural networks; • Software and its engineering → Software testing and debugging.

KEYWORDS

Deep learning testing, quality assurance, out of distribution

ACM Reference Format:

David Berend, Xiaofei Xie, Lei Ma, Lingjun Zhou, Yang Liu, Chi Xu, and Jianjun Zhao. 2020. Cats Are Not Fish: Deep Learning Testing Calls for Out-Of-Distribution Awareness. In *35th IEEE/ACM International Conference on Automated Software Engineering (ASE '20)*, September 21–25, 2020, Virtual Event, Australia. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3324884.3416609>

MSR'22

Beyond Duplicates: Towards Understanding and Predicting Link Types in Issue Tracking Systems

Clara Marie Lüders, Abir Bouraffa, Walid Maalej

ABSTRACT

Software projects use Issue Tracking Systems (ITS) like JIRA to track issues and organize the workflows around them. Issues are often inter-connected via different links such as the default JIRA link types *Duplicate*, *Relate*, *Block*, and *Subtask*. While previous research has focused on analyzing and predicting duplication links, this work aims at understanding the various other link types, their prevalence, and characteristics towards a more reliable link type prediction. For this, we studied 607,208 links connecting 698,790 issues in 15 public JIRA repositories. Besides the default types, the custom types *Depend*, *Incorporate*, *Split*, and *Cause* were also common. We manually grouped all 75 link types used in the repositories into five general categories: *General Relation*, *Duplication*, *Composition*, *Temporal / Causal*, and *Workflow*. Comparing the structures of the corresponding graphs, we observed several trends. For instance, as expected, *Duplication* links tend to represent simpler issue graphs often with two components and *Composition* links present the highest amount of hierarchical tree structures (97.7%). Surprisingly, *General Relation* links have a significantly higher transitivity score than *Duplication* and *Temporal / Causal* links.

Motivated by the differences between the types and by their popularity, we evaluated the robustness of two state-of-the-art duplicate detection approaches from the literature on our JIRA dataset. We found that current deep-learning approaches confuse between *Duplication* and other links in almost all repositories. On average, the classification accuracy dropped by 6% for one approach and 12% for the other. Extending the training sets with other link types seems to partly solve this issue. We discuss our findings and their implications for research and practice.

ACM Reference Format:

Clara Marie Lüders, Abir Bouraffa, Walid Maalej. 2022. Beyond Duplicates: Towards Understanding and Predicting Link Types in Issue Tracking Systems. In *Proceedings of MSR '22: Proceedings of the 19th International Conference on Mining Software Repositories (MSR 2022)*. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

1 INTRODUCTION

Development teams use Issue Tracking Systems (ITS) such as Bugzilla, Github Issues, or JIRA to track issues, including bugs to be fixed or features to be implemented. Over the years ITS have emerged as a central tool for planning and organizing development work [37],

and for communicating with users and other stakeholders [4]. Most ITS allow the creation of links between the issues to indicate technical or workflow dependencies. For instance, Bugzilla allows to set properties such as “depends on”, “blocks”, and “See also” for bug reports as well as to set the resolution status as “duplicate” with link to a duplicate report [1]. Similarly, JIRA users can choose between four possible default types in the *Issue Links* section: *Relate*, *Duplicate*, *Block*, and *Clone*¹. Additionally, *Subtasks* and *Epics* can be linked in separate sections.

Organizations can create and use additional link types to meet their specific needs. For instance, Qt² uses 6 link types including *Split* and *Replace*. Apache³ uses as many as 21 including the custom link types *Container* or *Breaks*. Each link type usually has an explicit definition in the ITS. Over time, stakeholders might also develop an “implicit” understanding of the connection represented by the link type. This might be either indeed unique to the community or simply a different name denoting the same connection labeled differently in another community. For instance, Apache uses both the link type *Depend* and *Block*, while all other repositories only use one of these types predominantly. However, in Bugzilla these two link types are equivalent⁴.

Studying the various link types and their usage patterns across the communities is essential to understanding collaboration in ITS and tightening automated tool support such as predicting missing links [27, 35]. In recent years, research has intensively studied the specific type *Duplicate*. Detecting those links would reduce the resolution time of duplicated issues and might reveal additional information included in one but not the other issue [5, 8]. Based on a Bugzilla dataset by Lazar et al. [21], researchers recently presented duplicate prediction approaches using state-of-the-art machine learning models with top performances of up to 97% [10, 16].

This work takes a holistic view on issue link types. We report on a study comparing the *various types* and their usage in 15 well-known public JIRA repositories [25]. By studying link types in JIRA, a widely used ITS in practice, we hope to create awareness about how the other types beyond duplicates are used and to inform a more generalizable and reliable link type predictions. Our work has three specific contributions. First, we manually reviewed and analyzed all types found in the ITS and categorized them into five general link categories. We report on the types, categories, and usage frequencies across the studied repositories. Second, we apply techniques from the field of graph theory to compare the complexity, shape, transitivity [29], and assortativity [26] of the different graphs corresponding to each link category. Our comparison reveals structural similarities across the repositories and several expected and unexpected trends. Third, we show that current link prediction models struggle to understand the existence of the links instead of

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, etc.

(Out-of-) Distribution aware Testing

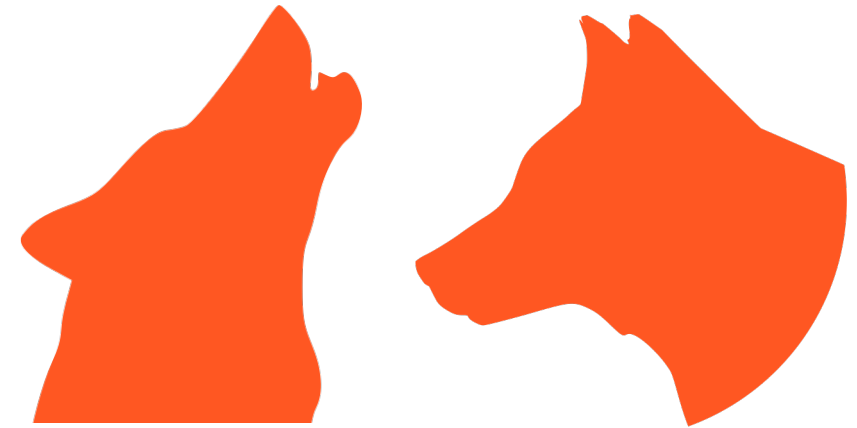
Capability in handling data that falls outside of its learned distribution (ID and OOD test case generation)



[Berend et al. ASE'20]

Taxonomy aware Testing

Capability of recognising similar but negative classes (e.g. canids vs. wolf vs. dogs)



[Lueders et al. MSR'22]



One last thing

**Can't we do all this with
ChatGPT?**



Dr. Milan Milanović 

@milan_milanovic



To replace programmers with AI, clients will need to accurately describe what they want.

We're safe.



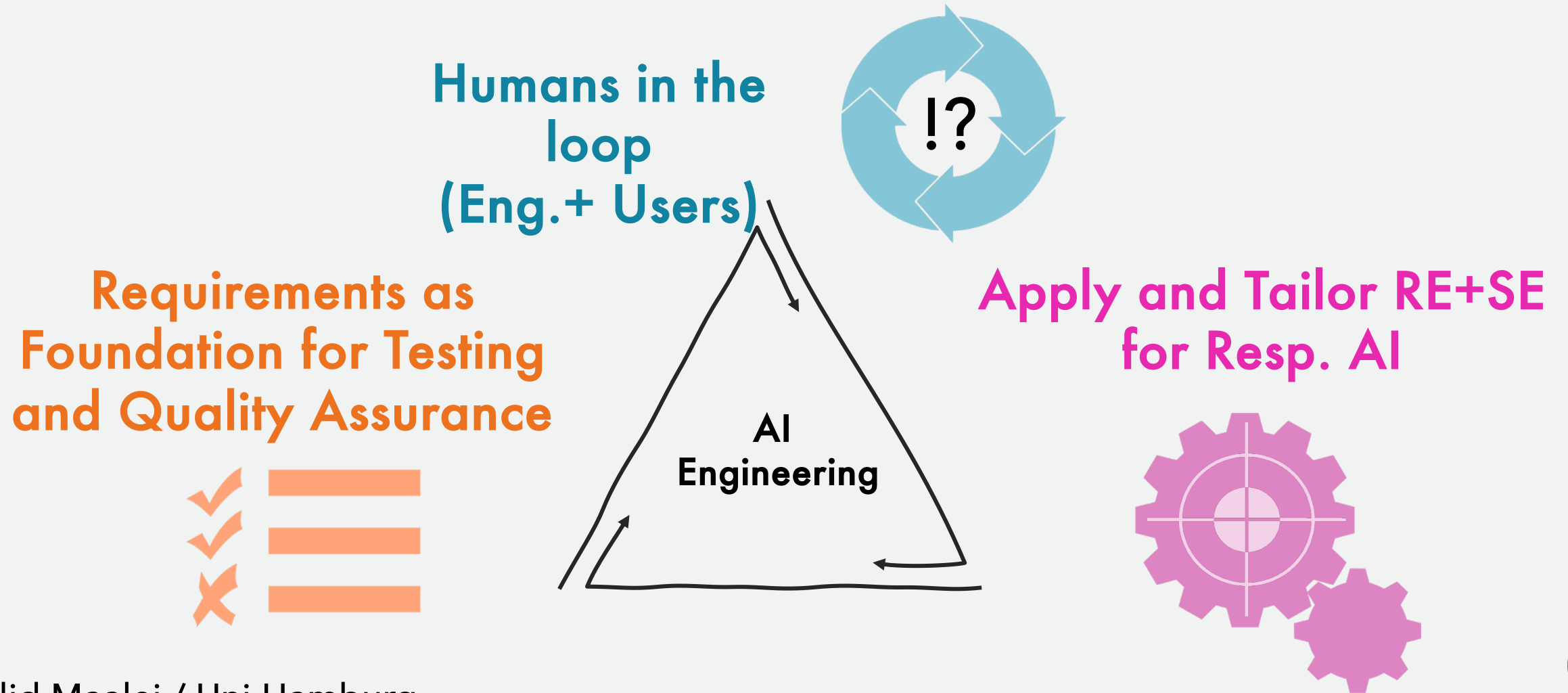
23,479

401 comments • 1,124 re



In summary

The AI Engineering Triangle



Computer

Tailoring Requirements Engineering for Responsible AI

Walid Maalej, Yen Dieu Pham and Larissa Chazette

Abstract—Requirements Engineering (RE) is the discipline for identifying, analyzing, as well as ensuring the implementation and delivery of user, technical, and societal requirements. Recently reported issues concerning the acceptance of Artificial Intelligence (AI) solutions after deployment, e.g. in the medical, automotive, or scientific domains, stress the importance of RE for designing and delivering Responsible AI systems. In this paper, we argue that RE should not only be carefully conducted but also tailored for Responsible AI. We outline related challenges for research and practice.

Index Terms—AI Engineering, Machine Learning Engineering, Quality Requirements, Data-Centric AI, Trustworthy AI, Human-in-the-Loop.

INTRODUCTION

A remarkably high number of AI solutions either do not make it to the production environment^[1] or fail after deployment. The reason is often the same: a missing or a bad

cases have in common, is that their AI models are designed and trained in a lab environment, representing a limited understanding and representation of the real world scenarios, and not accurately reflecting the context when making a decision. Technology-driven AI solutions tend to prioritize automation over stakeholder needs and to oversimplify rare but important scenarios and tradeoffs. Moreover, a lack of transparency and explainability of AI-based solutions often lead to mistrust and low acceptance by users [4].

While some might argue that these issues represent greater scientific, regulatory, and societal challenges, the good news for responsible AI is that there is already an established pragmatic engineering discipline which focuses on understanding stakeholder needs, specifying acceptable requirements, and ensuring the satisfaction of these requirements to a reasonable extent within the solution space. This discipline is called Requirements Engineering (RE) with a large body of knowl-

21 Feb 2023

