Requirements Elicitation in the Age of AI: A Tool's Multi-System Journey

Khlood Ahmad[1][0000−0002−7148−380X], Chetan Arora[2][0000−0003−1466−7386], Mohamed Abdelrazek[1][0000−0003−3812−9785], John Grundy[2][0000−0003−4928−7076], and Rajesh Vasa[1][0000−0003−4805−1467]

[1] Deakin University, Geelong, VIC, Australia
[2] Monash University, Clayton, VIC, Australia

Abstract. Traditional Requirements Engineering (RE) practices have introduced new tools to elicit and model requirements. Applying these tools to building AI software solutions has raised new issues and challenges. Also, most AI-based software solutions ignore human-centred values and focus on technical aspects. Existing tools and RE methods must extend to consider building more human-centred AI solutions. Recognizing this, we present a novel tool to support requirements elicitation and modelling for human-centred AI software. This paper details the tool's multi-system journey as it was applied across three diverse real-world projects: a mHealth app, a Virtual Reality 360 video enhancer, and a supermarket compliance app. The first two projects were in the later stages of software development, and the third was conducted during the early stages of building the software solution. The tool was used to elicit and model requirements for the three case studies in collaboration with eight experts. The tool helped to understand what requirements must be captured at the initial stages vs later stages in RE for AI (RE4AI).

Keywords: requirements engineering · software engineering · artificial intelligence · machine learning · human-centered · conceptual modeling

1 Introduction

Recent technological developments have shown a shift towards using Artificial Intelligence (AI) components in many software solutions. This technology shift has become possible because of increased processing power and data availability [33]. However, the prevalence of AI components in building new software systems has impacted the way we build software and many new issues have emerged in the process. Existing methods and tools used in Software Engineering (SE) are inadequate in building AI software [53, 23]. Furthermore, new requirements not considered in traditional SE have appeared, such as around ethics and data [43].

In traditional Requirements Engineering (RE) for software systems that do not have an AI component, the development process usually involves specifying requirements for systems that are deterministic and outputs are known early on. However, in RE for AI (RE4AI), it is more difficult to specify requirements as these systems tend to be undeterministic and black-box in nature, and outputs are usually unknown until the models are trained and tested with data [14, 43, 1, 35]. These changes have resulted in a gap in RE practises, tools and techniques for building AI-based software systems.

Another major issue when building AI-based software is that it is much more common to focus on the technical aspects of the AI components [42, 52] and overlook human-centred aspects such as age, gender, culture, emotions, ethnicity, and many others [31, 55]. Overlooking these human-centred aspects when building AI software can lead to biased systems, discrimination towards users, or non-inclusive [15, 8, 54]. Human-centred design resolves around building systems that meet human needs first, rather than building technically sound systems that do not address the users needs [42, 55]. Recently, more research has been invested in ways to build more human-centred AI software. Leading industry organizations such as Google, Microsoft, and Apple are proposing guidelines for building human-centred AI software [29, 44, 11]. Although this is a growing area of research, there is limited work on RE for human-centred AI, and most of the available work focuses on designing systems [6, 3].

When comparing the results from a recent SLR [6] with a user survey [4], we found that in-practice tools were used more than the ones reported in the literature. The survey identified 15 different tools and drawing editors used. Although more tools were used in practice, such as JIRA, Confluence, and Excel, most do not consider many aspects of RE4AI and could contribute to low-quality requirements in AI software.

In this study, we propose and evaluate a new tool to elicit and model requirements for human-centred AI software. The tool consists of a catalogue of requirements for human-centred AI and a modelling language to present these requirements visually. The catalogue contains requirements extracted from industrial human-centred AI guidelines and any requirements on RE4AI identified during the mapping study [5]. We apply and evaluate our proposed tool in three case studies. Our framework helps improve team awareness of what others are doing regarding AI-based software component requirements.

We published the preliminary result of the tool at the 18th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE) [5]. The study showcases and evaluates our framework and tool support in one case study. This chapter extends the evaluation to another case study, and we conduct interviews with seven experts from the three projects to get an insight into the benefits and limitations of our tool. The rest of the paper is structured as follows: Section 2 provides a brief background related work. Section 3 presents details of our toolkit. Section 4 reports on implementing the tool in a case study. Section 5 presents the interview with the experts to evaluate the tool. Section 6 discusses key results and summarizes emerging theories. Section 7 concludes the paper.

2 Background

Requirements elicitation is one of the most crucial parts of RE as it sets out to unravel and capture the need for the system from the stakeholders early on in the software development life cycle [20]. The process of eliciting requirements includes "a set of activities that must allow communication, prioritization, negotiation, and collaboration with all the relevant stakeholders" [60]. To capture the correct requirements, system boundaries need to be set and defined [47].

In Zowghi and Coulin [60], requirements elicitation undertake the following activities: 1) Understanding the problems (i.e. application domain) that must be solved. 2) Identifying the different sources when collecting requirements for a given software project. In this step, we need to identify the stakeholders, system users, systems to be replaced, business processes, documentation, etc. 3) Identifying and analysing relevant stakeholders, whether they are part of the organization or external stakeholders. This will include finding out the most important stakeholders (the customers and clients). 4) Deciding on what tools, techniques and methods to use when eliciting requirements for a given system. These will depend on the context of a given system and business rules. 5) Once the stakeholders, methods to use, business operations and rules are established, the last thing will be to elicit the requirements needed from the stakeholders.

There are several issues related to requirements elicitation techniques that include miscommunication and difficulties in transferring knowledge between the elicitor and stakeholders [26]. Some of these issues include eliciting: (1) known-unknowns which is the knowledge that the elicitor is aware of but not the stakeholder. (2) Unknown-known knowledge held by the stakeholder and cannot express to the elicitor. (3) Unknown-unknowns, which are the most challenging of all, and both the stakeholder and elicitor are unaware of the existing knowledge in this situation [58, 27]. We argue that there is a significant amount of requirements that are unknown-unknowns in RE4AI, as most of the time, it is difficult to set requirements early when building AI software, and in most situations, it is unknown what the outcomes of the system would be until the data is trained on a given model. Therefore, some requirements can only become known towards the end of the software life-cycle.

Requirements modelling languages are used to visually display requirements and identify the stakeholders' needs at a higher-level abstraction of the system [28]. Examples of RE modelling languages include Goal-Oriented Requirements Languages (GRL) [37] and the i* model [18]. In GRL, goals are used to model non-functional requirements and business rules [9]. However, the disadvantage of using GLR is that it is difficult to learn in non-software engineering. Modelling languages such as UML and SysML have been used to model requirements in RE4AI. Although these modelling languages are easier to use and learn than GRL, they have limitations in modelling NFRs and business rules [49, 30, 7]. Other studies proposed to use conceptual models to present requirements in RE4AI. In [46], the authors presented a conceptual framework to model requirements for ML systems from three views, including the business view, analytic design view, and the data preparation view. The three views are combined to show a holistic view of the system. More solutions are being proposed to model requirements in RE4AI; however, there is limited work in modelling human-centred RE4AI.

3 A Toolkit to Elicit and Model Requirements for Human-centered AI systems

There is a considerable amount of tools used in SE to manage software systems. However, most of these tools are not accustomed to building software with an AI component [53]. Although new approaches have been proposed to provide tailored tools in RE4AI, most of these new methods are still in the early phases of development with limited empirical work [6]. Although most of these tools are common and easy to use among diverse team members, they do not provide the support needed to manage requirements for AI software. Consequently, we propose a new tool based on our framework to elicit and model requirements for human-centred AI software.

3.1 Requirements Engineering for Human-centered AI Framework

Our proposed Requirements Engineering for Human-centered AI (RE4HCAI) framework consists of three layers as shown in Figure 1, and was inspired by Google PAIR's human-centred AI guidelines [29]. We used five of the areas mentioned in Google PAIR and added a new area "Model Needs". The six selected areas included: User Needs, Model Needs, Data Needs, Feedback & User Control, Explainability & Trust, and Errors & Failure. The three layers of the framework are explained below.

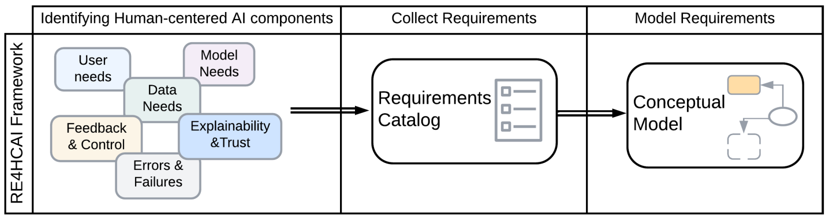


Fig. 1. Framework for eliciting and modelling requirements for human-centred AI software (from [5])

Identifying Human-centered AI Requirements: We combined all the human-centred guidelines from Google PAIR, Microsoft's guidelines for human-centred AI interaction [44], and the guidelines for Apple's human interface for developing ML applications [11] along with the Machine Learning Canvas [39]. Note that the ML Canvas is not aimed at including human-centred aspects and mostly focuses on technical features; however, we found that it provided a great collaboration platform and we found it would complement the available human-centred guidelines. The combined guidelines were then mapped against any existing human-centred studies found in our mapping study on RE4AI [3] and [59]. Also, we collected any requirements found in the mapping studies that could fit into any of our six selected areas. Figure 2 shows a high-level summary of the collected requirements for each area.

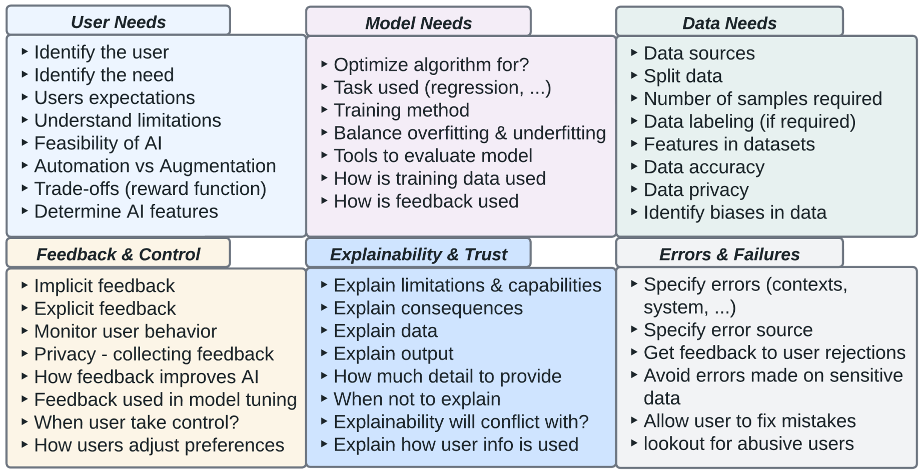

User Needs
‣ Identify the user
‣ Identify the need
‣ Users expectations
‣ Understand limitations
‣ Feasibility of AI
‣ Automation vs Augmentation
‣ Trade-offs (reward function)
‣ Determine AI features
Model Needs
‣ Optimize algorithm for?
‣ Task used (regression, ...)
‣ Training method
‣ Balance overfitting & underfitting
‣ Tools to evaluate model
‣ How is training data used
‣ How is feedback used
Data Needs
‣ Data sources
‣ Split data
‣ Number of samples required
‣ Data labeling (if required)
‣ Features in datasets
‣ Data accuracy
‣ Data privacy
‣ Identify biases in data
Feedback & Control
‣ Implicit feedback
‣ Explicit feedback
‣ Monitor user behavior
‣ Privacy - collecting feedback
‣ How feedback improves AI
‣ Feedback used in model tuning
‣ When user take control?
‣ How users adjust preferences
Explainability & Trust
‣ Explain limitations & capabilities
‣ Explain consequences
‣ Explain data
‣ Explain output
‣ How much detail to provide
‣ When not to explain
‣ Explainability will conflict with?
‣ Explain how user info is used
Errors & Failures
‣ Specify errors (contexts, system, ...)
‣ Specify error source
‣ Get feedback to user rejections
‣ Avoid errors made on sensitive data
‣ Allow user to fix mistakes
‣ lookout for abusive users

Fig. 2. Requirements for human-centred AI (from [5])

When mapping the human-centred guidelines against the requirements found in the literature, we found that there was a gap in two of the selected areas: (4) Feedback & User Control and (6) Errors & Failure, with not much literature found in those two areas. Additionally, the collected human-centred guidelines were targeted at the design phase of the software development process. Thus, we wanted to know which of the guideline needed to be addressed in RE. We asked participants in a user survey which of these guidelines they have either included when specifying requirements for AI software, or they thought should be included in RE4AI. The results showed that all the collected human-centred guidelines should be covered in RE. There was less emphasis on identifying errors and deciding on what feedback to use when building AI software, however, we believe that these two areas are overlooked in RE and need to be investigated further in research.

Catalogue of Requirements for Human-Centered AI: The collected human-centred requirements were listed in a tabular format. The catalogue included all the requirements that were mapped from the literature and the guidelines and had six sections. Each section was dedicated to displaying more detailed requirements for each of the areas shown in Figure 2. The objective was to use the catalogue as a means to elicit requirements from the stakeholders first, then model these requirements using our conceptual modelling language.

Conceptual Model: When conducting the mapping study, we looked at papers that used a modelling language or notation to present requirements, we found that most of the studies either preferred UML or used a Domain Specific

Language (DSL). The reason for UML being more popular than other used modelling languages was that it was easier to understand and use among different groups [41, 12]. However, the issue with UML was that it had some limitations when presenting requirements for AI, and it lacked support for modelling business rules and non-functional requirements. Also, we had concepts that were difficult to present such as needs, limitations, and trade-offs. Therefore, we created a domain-specific model to present our requirements visually.

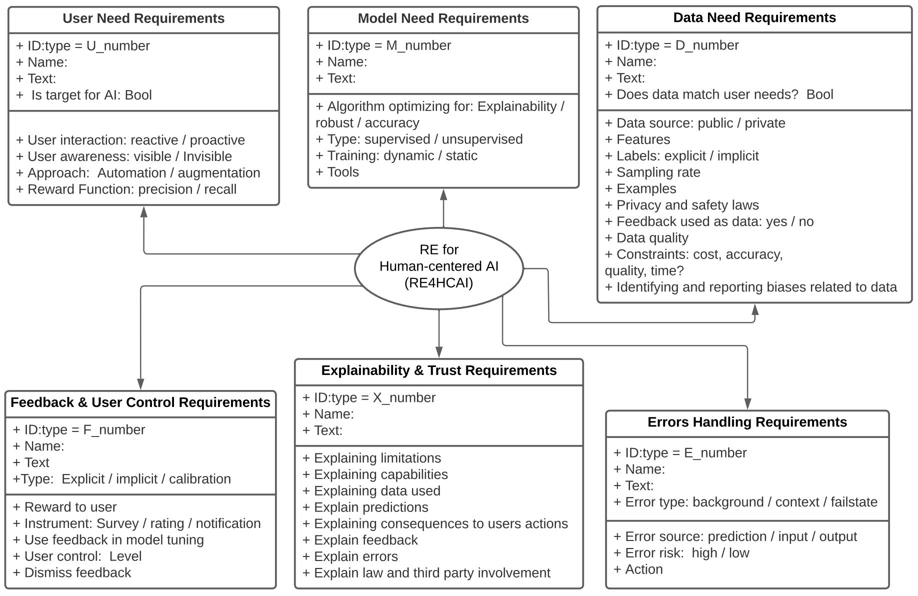

User Need Requirements
+ ID:type = U_number
+ Name:
+ Text:
+ Is target for AI: Bool
+ User interaction: reactive / proactive
+ User awareness: visible / Invisible
+ Approach: Automation / augmentation
+ Reward Function: precision / recall
Model Need Requirements
+ ID:type = M_number
+ Name:
+ Text:
+ Algorithm optimizing for: Explainability / robust / accuracy
+ Type: supervised / unsupervised
+ Training: dynamic / static
+ Tools
Data Need Requirements
+ ID:type = D_number
+ Name:
+ Text:
+ Does data match user needs? Bool
+ Data source: public / private
+ Features
+ Labels: explicit / implicit
+ Sampling rate
+ Examples
+ Privacy and safety laws
+ Feedback used as data: yes / no
+ Data quality
+ Constraints: cost, accuracy, quality, time?
+ Identifying and reporting biases related to data
RE for Human-centered AI (RE4HCAI)
Feedback & User Control Requirements
+ ID:type = F_number
+ Name:
+ Text
+Type: Explicit / implicit / calibration
+ Reward to user
+ Instrument: Survey / rating / notification
+ Use feedback in model tuning
+ User control: Level
+ Dismiss feedback
Explainability & Trust Requirements
+ ID:type = X_number
+ Name:
+ Text:
+ Explaining limitations
+ Explaining capabilities
+ Explaining data used
+ Explain predictions
+ Explaining consequences to users actions
+ Explain feedback
+ Explain errors
+ Explain law and third party involvement
Errors Handling Requirements
+ ID:type = E_number
+ Name:
+ Text:
+ Error type: background / context / failstate
+ Error source: prediction / input / output
+ Error risk: high / low
+ Action

Fig. 3. First level of modelling language: Six different areas at a high-level view

Our model consisted of two layers; the first layer provided a holistic view of each of our six areas as shown in Figure 3. In the first layer, we used UML class notations to showcase a more high-level presentation of the requirements needed for each area, and an oval shape to show the high-level goal of the system and connecting all the six presented areas. The second layer presented a separate model for each area, we used unified notations for all the areas as presented in Figure 4. When creating our modelling notations we tried to adhere as much as possible with the Physics of Notations [45]. We incorporated the use of different shapes, colours, and textures to reduce the cognitive overload of users and make the notations as easy to use and understand as possible. We did limit the use of colour to yellow for people who are colour-blind or vision impaired. Each notation is used to model a different concept from the requirements collected in the catalogue.

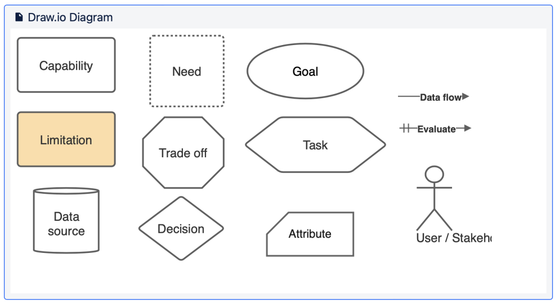


Fig. 4. Legend for the conceptual model to show more abstract requirements [5]

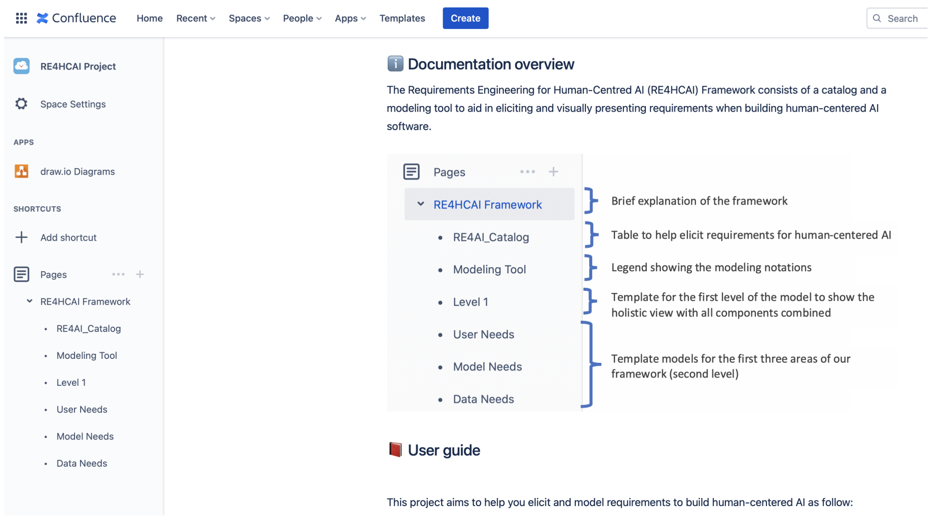
3.2 Tool and Platform Selection

Since building a software system with an AI component required different team roles to work together, we wanted to use platforms and tools that most people were familiar with. The idea was to utilize existing tools and collaboration platforms. Therefore, we decided to use some of the platforms identified in our user survey to evaluate our initial prototype. We used Confluence [56] which is an online collaboration platform where team members can work on projects, share, plan, and build ideas together. Confluence was used as means to present and distribute our framework among participants and allowed us to present our modelling tool using existing drawing applications such as Lucid Chart [40] and Draw.io [57].

A new project space was created and a brief overview of the tool was provided on the introduction page as shown in Figure 5. Also, the introduction page provided a short tutorial on how to use the tool. The first page included a table with the catalogue, which the participant could use to elicit the human-centred requirements for a specific AI software system. The second page was used to showcase the available modelling notations that are displayed in Figure 4. The next four pages contained templates to our the models from the first and second layers. We only provide models for the first three area's to include "User Needs", "Model Needs", and "Data Needs".

4 Application and Evaluation

We conducted three case studies to investigate how our framework and tool could contribute to engineering AI-based systems with a human-centred perspective, and if it would benefit the process of creating human-centred AI software. In this section we describe the process we used in designing, selecting, recurring, and conducting the case study research in order to evaluate our proposed tool and framework.



Documentation overview

The Requirements Engineering for Human-Centred AI (RE4HCAI) Framework consists of a catalog and a modeling tool to aid in eliciting and visually presenting requirements when building human-centered AI software.

- Pages
  - RE4HCAI Framework — Brief explanation of the framework
    - RE4AI_Catalog — Table to help elicit requirements for human-centered AI
    - Modeling Tool — Legend showing the modeling notations
    - Level 1 — Template for the first level of the model to show the holistic view with all components combined
    - User Needs
    - Model Needs
    - Data Needs — Template models for the first three areas of our framework (second level)

User guide

This project aims to help you elicit and model requirements to build human-centered AI as follow:

Fig. 5. The Confluence collaboration page used to conduct the case study (from [5])

The unit of analysis was the AI software projects that our participants were working on or have previously worked on. We wanted to analyze the actual process of building AI software, and how could these processes improve [13].

4.1 Case Study Design

The case study was designed following the guidelines presented in [21] and the steps provided in [36]. The study included two to three meetings intended to elicit requirements for an AI system from the participants and a final workshop or meeting to model the elicited requirements. Each meeting and workshop took between an hour to two hours of the participant's time. Once all the requirements were elicited and modelled we conducted a final interview to evaluate our tool, the interview questions included:

1. General information about the participants' role in the organization, types of projects they have worked on, and years of experience. No identifying information such as name, age or contact details was collected (unless the participant wanted to be acknowledged). These questions aimed at identifying what type of projects were used within a particular application domain and helped us find how the Requirements Engineering (RE) would differ depending on the application domain.
2. The second set of questions involved finding if the catalogue helped in identifying human-centred aspects of the project that the participants were not aware of. Also, these questions helped us evaluate the framework and modelling tool including identifying what tools were our participants currently using if they had any limitations, and if the proposed framework addressed

any of these limitations. Finally, we asked our participants how could we improve the framework.

4.2 Study selection & Recruitment

When selecting the pilot projects, we looked for software projects that were either fully established, towards the end of the software life cycle, or at the early stages of the development process. The reason for selecting more established projects is that we wanted to see how the original RE process differed from before applying our framework to after implementing the framework. It was also important to see which requirements were not considered, and how these requirements changed over time. The main selection criteria for the software project was that it had to include an AI component, as our framework only focused on eliciting and modelling requirements for the AI part of the software system. Initially, the search was open to any application domain, and no specific domain was targeted during recruitment.

Once the ethics approval was obtained, we recruited participants based on our inclusion criteria as follows:

1. Any practitioner or researcher working on building a software system with an AI component.
2. The participant had to have no prior knowledge about our framework.
3. The participant did not need to have any modelling knowledge, as we wanted to see how much time novice vs expert users needed to learn how to use our modelling tool.
4. The participant could be from any of the following disciplines: Software engineers, requirements engineers, data scientists, ML specialists, AI developers, IT professionals, project managers, academics, researchers, and research students.

Participants that agreed to undertake the evaluation study were given a plain language statement and consent form that explained our research aims, contributions and possible findings. Once we collected the participants' consents we started conducting the meetings, workshops and interviews. Each meeting and workshop took approximately an hour to complete. We used the participants' preferred communication method and the meetings were conducted using either MS Teams or in person at the participants' work locations.

4.3 The iMove Project

The iMove project [19] involves designing and building a real-time health application for people with type 2 diabetes (T2D). The application keeps track of the user's behaviour towards movement, measures the sedentary time, and provides notifications to remind the user to move. These notifications consist of predicted messages from the DL model to remind the person to stand up or walk after sitting for a prolonged time.

We recruited three people that worked on the iMove project with different roles including a project manager, a data scientist, and a software engineer. We created a dedicated project in Confluence for the iMove project, and access to the page was given to each participant. Participants could go back at any time and edit the elicited requirements and models. Due to their busy schedules, three different sessions were conducted using Microsoft Teams with each participant individually. Requirements were elicited using the catalogue from each of the three participants. Each session lasted between 45 minutes up to two hours. Once all the requirements were elicited, a holistic view of the iMove project was established. More details on the iMove project can be found in [5].

4.4 The VR 360°video enhancer project

The first project used Deep Learning (DL) to enhance the quality of 360° videos for VR platforms. The main aim of the AI-based software was to improve the Quality of Experience (QoE) and Quality of Service (QoS) for systems streaming and rendering 360° video content. Current solutions used in building 360°video content can result in a degradation in the final quality of the video due to stitching multiple videos, bandwidth constraints when streaming, and the quality of hardware used by the end-users [22, 10, 50]. The proposed solution was to enhance the quality of the final product using AI-based software. We recruited a data scientist working on the VR 360°video enhancer project and elicited and modelled requirements for the project. More detail on the VR 360°video enhancer project can be found in [2].

4.5 The Planogram Compliance Project

Planograms are used in retail to manage items on display. They are used as visual representations for where products should be located on store shelves to maximize sales [32]. Planogram compliance checks if the products displayed on shelves in retail stores are in the same order as the given planogram [48]. Different methods are used to ensure that retail stores comply with the planogram. These methods range from manually checking the store shelf to using Radio Frequency Identification (RFID) [24] and IoT systems [25]. However, the first approach can be costly in human resources, and the second can have limitations related to using sensors and installation costs [51]. More recent efforts have focused on using AI to provide more affordable solutions. One study used computer vision to identify recurring patterns of items that are not placed correctly on a self without the need of a template [38]. Other approaches experimented with different Convolutional Neural Network (CNN) DL models to find the most suitable planogram compliance outcomes for a given dataset [16].

For this case study, we elicit and model requirements for a project provided by a client to build a solution for a planogram compliance system using AI software. We recruited four people to work on that project. An initial workshop was set to elicit requirements, and each participant was given access to the space dedicated to the planogram compliance project in Confluence. The workshop took approximately two hours to complete. Once the requirements were obtained, we modelled them using the provided templates and shared them with each participant. We followed up with four separate interviews to get the participant's feedback. A higher-level view of the project is presented in Figure 6.

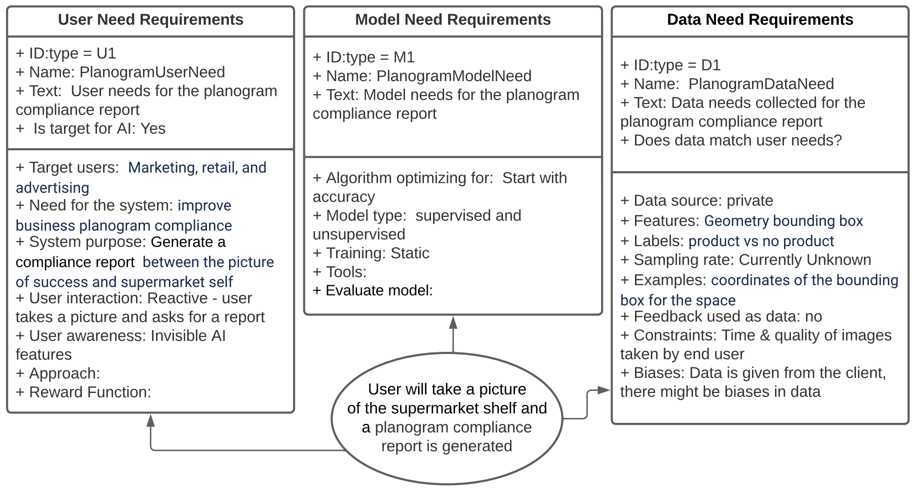

User Need Requirements
+ ID:type = U1
+ Name: PlanogramUserNeed
+ Text: User needs for the planogram compliance report
+ Is target for AI: Yes
+ Target users: Marketing, retail, and advertising
+ Need for the system: improve business planogram compliance
+ System purpose: Generate a compliance report between the picture of success and supermarket self
+ User interaction: Reactive - user takes a picture and asks for a report
+ User awareness: Invisible AI features
+ Approach:
+ Reward Function:
Model Need Requirements
+ ID:type = M1
+ Name: PlanogramModelNeed
+ Text: Model needs for the planogram compliance report
+ Algorithm optimizing for: Start with accuracy
+ Model type: supervised and unsupervised
+ Training: Static
+ Tools:
+ Evaluate model:
Data Need Requirements
+ ID:type = D1
+ Name: PlanogramDataNeed
+ Text: Data needs collected for the planogram compliance report
+ Does data match user needs?
+ Data source: private
+ Features: Geometry bounding box
+ Labels: product vs no product
+ Sampling rate: Currently Unknown
+ Examples: coordinates of the bounding box for the space
+ Feedback used as data: no
+ Constraints: Time & quality of images taken by end user
+ Biases: Data is given from the client, there might be biases in data
User will take a picture of the supermarket shelf and a planogram compliance report is generated

Fig. 6. A holistic view of elicited requirements for the planogram-compliance project

We note that this case study was conducted during the early stages of the development process. Therefore, there were a lot of unknown requirements. One of the major issues in requirements elicitation is unknown unknowns, which represents requirements neither the stakeholder nor the elicitor is aware of [58, 27]. When eliciting requirements for the planogram compliance project, we noticed that there were more unknown-unknown requirements in comparison to the previous two case studies.

User Needs for the Planogram Compliance Project. The need for this system was given to the team working on this project by an outside company. The client provided them with the data and asked them to solve an existing problem with planogram compliance. The system's primary purpose was to compare the picture of success (the given planogram) and a picture taken of the retail store self. Consequently, someone from the marketing and advertising department will take that picture, and the system will generate a compliance report. It was estimated for now that the system would be able to measure correctly to around 80% of its capacity.

The next step involved identifying the limitations and benefits of the system. The benefits included ensuring that the retail stores complied with requests from marketing and advertising to where items should be located on shelves. The other advantage allowed the marketing department to understand if a product would sell faster depending on its location on the self. There were a number of existing limitations that included the data, as we will explain in section 4.5. The other limitations were related to the performance of the final system, as the model would need to be compressed when used over the server, affecting its performance. Thus, the system would operate slower when used directly from the device. The system was designed to be reactive in the sense that the end user will ask for the system to check if the image of the shelf complies with the planogram provided by the retailer. The end user might not have been aware of the AI component.

When reviewing the choice between automation vs augmentation, the participants said it was too early to determine. However, after further discussions with the participants, we established that the choice would depend on a combination of system performance and end-user needs. The end user needs should be settled with the client, and if there is the need for a feedback loop, i.e., the person at the retail store taking the photo is the same person providing feedback, then we need to capture that and build the system to augment. Conversely, if there is a downstream process that later validates the outcome, and they don't need the end user to interact with the system, they could build towards automation. The system's performance is the other aspect that would influence the decision to build automation vs. augmentation. Automation could be considered if the system performs well and the end user needs minimal interaction with the system. However, if the model's performance is poor, the resulting system might need to incorporate more feedback from the end user, making augmentation a better approach.

Next, we wanted to identify the choice of the reward function or evaluation method. The participants could not answer this question since the project was still in the very early phases. However, we discussed the possible consequences of having an incorrect prediction. The first step was to identify what would happen if an FP vs. an FN existed. If an FP is present, then the compliance report would predict that the product is presumably in the correct location on the shelf when in reality, it is missing or misplaced. On the other hand, an FN indicates that the product is placed in its correct location, but the compliance report shows that the product is misplaced.

Having an FP or an FN will affect the generated compliance report. Having an FP would mean that a retailer being not compliant will go undetected. This will have a more significant loss towards the marketing department as a missing item is not reported in this situation. In the condition where an FN is present, the report will claim that the supermarket was not compliant when in fact, they were compliant, and the marketing or advertising department will be presuming false accusations against them.

In this particular situation, the different fallouts to account for should be discussed with the client to conclude which of the incorrect predictions will have higher stakes. Additionally, it was early to suggest how predictions were going to be used in making decisions. The participants explained that they had to deal with multiple predictions used in multiple models. Therefore, it was difficult to report how the predictions were going to be used per model and how it would influence the final output. Requirements for user needs are shown in Figure 7.

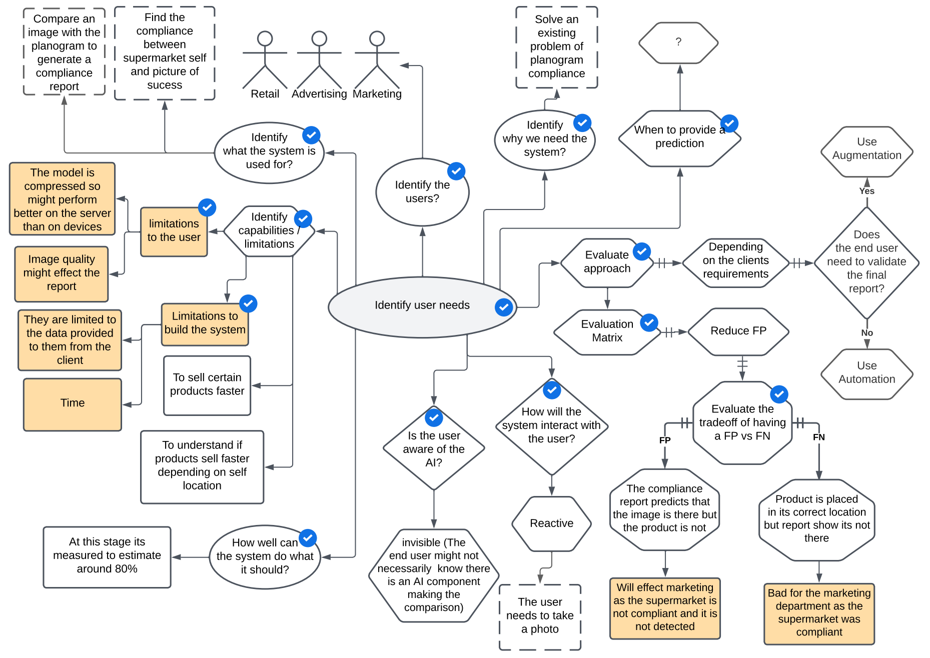

Compare an image with the planogram to generate a compliance report
Find the compliance between supermarket self and picture of sucess
Retail
Advertising
Marketing
Solve an existing problem of planogram compliance
?
Identify what the system is used for?
Identify the users?
Identify why we need the system?
When to provide a prediction
Use Augmentation
Yes
No
Use Automation
The model is compressed so might perform better on the server than on devices
limitations to the user
Identify capabilities / limitations
Image quality might effect the report
Identify user needs
Evaluate approach
Depending on the clients requirements
Does the end user need to validate the final report?
They are limited to the data provided to them from the client
Limitations to build the system
Evaluation Matrix
Reduce FP
Time
To sell certain products faster
To understand if products sell faster depending on self location
Is the user aware of the AI?
How will the system interact with the user?
Evaluate the tradeoff of having a FP vs FN
FP
FN
The compliance report predicts that the image is there but the product is not
Product is placed in its correct location but report show its not there
Reactive
At this stage its measured to estimate around 80%
How well can the system do what it should?
invisible (The end user might not necessarily know there is an AI component making the comparison)
The user needs to take a photo
Will effect marketing as the supermarket is not compliant and it is not detected
Bad for the marketing department as the supermarket was compliant

Fig. 7. Model with the User Needs requirements for the planogram compliance project

Model Needs for the Planogram Compliance project. Limited requirements were established for this area. There were more unknowns in the Model Needs to include a balance between overfitting and underfitting, how the quality of the model is going to be evaluated, and identifying tools post evaluation. Some requirements were not fully established as they couldn't predict how the model would behave on the provided training dataset. The type of model used for this task was a DL model with a combination of supervised and unsupervised learning. They planned to optimize for accuracy and then move towards robustness as the project progressed. A static offline method was used to train the model, and they did not account for user feedback in re-training the model. Nevertheless, the project was still in its early phases, so more requirements were still unknown at this stage, and more requirements would change or emerge as the project progressed.

Model tuning was done at the start of the project. The ML engineer explained that the standard way to build an ML project these days was to use a pre-trained model and fine-tune it to adjust to the specific task. Using an existing model also depended on the dataset. If the data contained images or text, it was more common to fine-tune an existing model. Also, pre-evaluation was performed using a Python library to generate an initial analysis of the data per branch and adjust the number of data for model training. Specifying for scalability issues was a bit more tricky, as initially, the system was customized to work on one platform only, which meant that such issues might not be apparent. However, if future plans include adding more platforms, this requirement should be investigated further. Requirements for model needs are shown in detail in Figure 8.

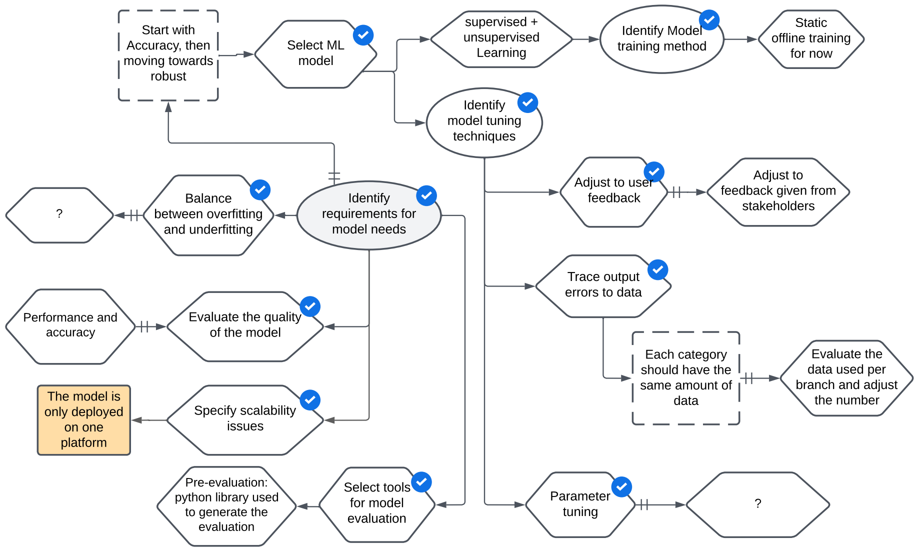

Start with Accuracy, then moving towards robust
Select ML model
supervised + unsupervised Learning
Identify Model training method
Static offline training for now
Identify model tuning techniques
?
Balance between overfitting and underfitting
Identify requirements for model needs
Adjust to user feedback
Adjust to feedback given from stakeholders
Trace output errors to data
Performance and accuracy
Evaluate the quality of the model
Each category should have the same amount of data
Evaluate the data used per branch and adjust the number
The model is only deployed on one platform
Specify scalability issues
Pre-evaluation: python library used to generate the evaluation
Select tools for model evaluation
Parameter tuning
?

Fig. 8. Model with the Model Needs requirements for planogram compliance project

Data Needs for the Planogram Compliance project. Data needs involved identifying requirements related to the data selection, including establishing features, labels, examples, and the sampling rate. The initial model used was to classify if the product was on the self or missing. Therefore, labelling the data was done implicitly by using an ML model to classify them as product vs no product. Features included adding a geometry bounding box around each product. Each row of data (examples) included the coordinates of the bounding box for each space. Finally, the sampling will depend on the evaluations to find the correct amount of data needed for the model training and testing.

Constraints related to the data included the quality of images and the time needed to clean and label the data. The quality constraint was a concern as it required the end user to take an image, which might compromise the compliance

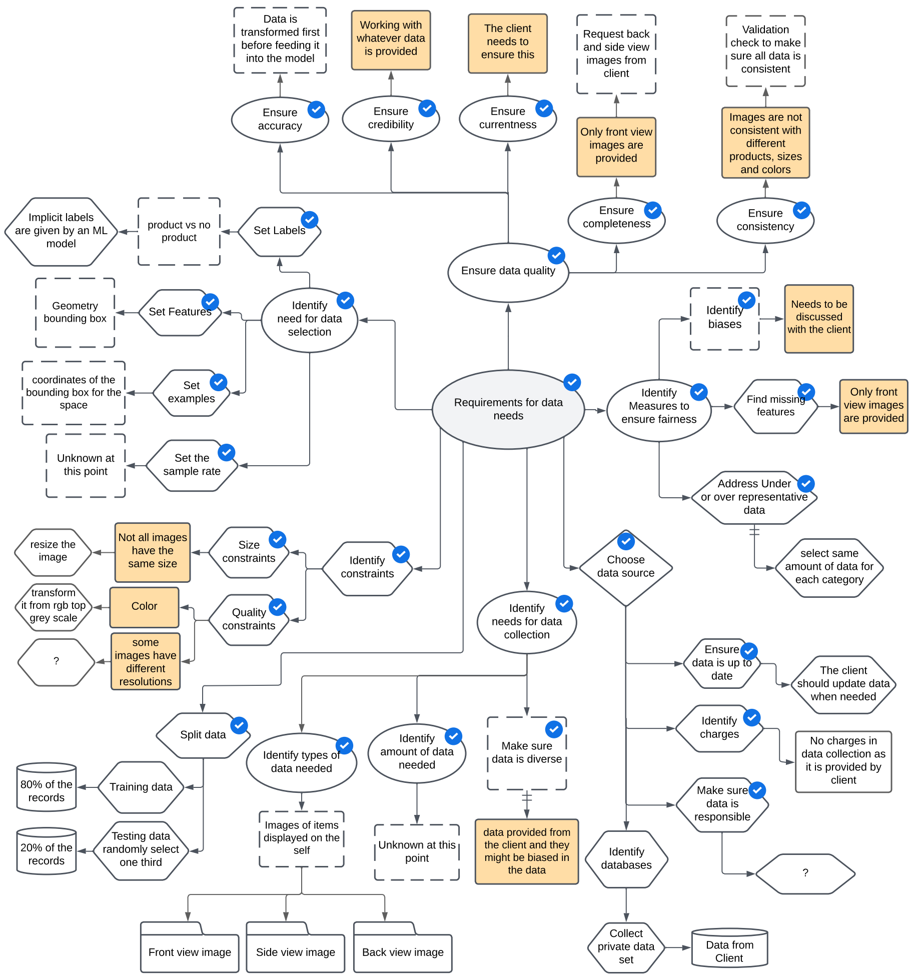

Requirements for data needs
Ensure data quality
Ensure accuracy
Data is transformed first before feeding it into the model
Ensure credibility
Working with whatever data is provided
Ensure currentness
The client needs to ensure this
Ensure completeness
Only front view images are provided
Request back and side view images from client
Ensure consistency
Images are not consistent with different products, sizes and colors
Validation check to make sure all data is consistent
Identify need for data selection
Set Labels
product vs no product
Implicit labels are given by an ML model
Set Features
Geometry bounding box
Set examples
coordinates of the bounding box for the space
Set the sample rate
Unknown at this point
Identify Measures to ensure fairness
Identify biases
Needs to be discussed with the client
Find missing features
Only front view images are provided
Address Under or over representative data
select same amount of data for each category
Identify constraints
Size constraints
Not all images have the same size
resize the image
Quality constraints
Color
transform it from rgb top grey scale
some images have different resolutions
?
Choose data source
Ensure data is up to date
The client should update data when needed
Identify charges
No charges in data collection as it is provided by client
Make sure data is responsible
?
Identify databases
Collect private data set
Data from Client
Identify needs for data collection
Split data
Training data
80% of the records
Testing data randomly select one third
20% of the records
Identify types of data needed
Images of items displayed on the self
Front view image
Side view image
Back view image
Identify amount of data needed
Unknown at this point
Make sure data is diverse
data provided from the client and they might be biased in the data

Fig. 9. Model with the Data Needs requirements for the planogram compliance project

report if the quality of the image was not adequate. Therefore, the development team had to consider the image quality when building the product. The second constraint was time, as the client had given them a deadline to deliver the product. Thus, they needed to provide high-quality data within the given time frame to support the training and testing of the model.

The data was a private dataset given to the development team by the client. Certain aspects, such as biases and if the data was responsibly sourced, were unknown to our participants. These biases could include more representations of certain products or missing products, which might affect the quality of the predictions when using new products. Moreover, the same amount of images from each category was used when training the model to avoid selection bias.

Ensuring the data quality meant providing accurate, complete, consistent, credible, and current data. The last two measures were unknown as they had to assume that the given data was authentic and collected within the correct time frame. Making the data accurate involved transforming each image to a similar scale before feeding it into the model. There were a few issues with data consistency, as the images needed to be more consistent. The dataset had images that were different in size, quality, and colour. Therefore, to make the data consistent, they resized all the images to the same size and transformed them from RGB to grayscale. Another presenting issue was regarding the completeness of the data, as the images provided by the client were only front-view images. This caused issues when training the model; if a product was displayed with the back or side view on the self, the model could not recognize it. Subsequently, they had to ask the client to provide more robust images with front and side views of the products to improve the model's performance. Requirements for data needs are shown in Figure 9.

5 Interview-based Evaluation

5.1 Study Design

We invited the eight participants we worked with during the case studies to reflect on what they thought of our tool and whether it effectively supported eliciting and modelling requirements for AI software. Seven people responded and agreed to participate. The data scientist from the iMove project was not available for the interview stage due to personal reasons. Consent was obtained from each participant. Interviews were carried out either via the online platform Microsoft Teams or in person at the participant's work location. The interviews took between 20 minutes to one hour to conduct, and they were all audio recorded and transcribed for later analysis. Different roles included: one software engineer, one project manager, two data scientists, and three ML experts. Participants had work experience ranging from 1.5 to 10 years, as shown in Table 1.

Table 1. Participants experience's

ID	**Role**	**Years of Experience**	**Application Domain**
P1	Software Engineer	2 years	Digital health
P2	Project Manager	9 years	Digital health
P3	Data Scientist	2 years	Multimedia
P4	ML Engineer	10 years	Marketing
P5	ML Engineer	3 years	Marketing
P6	ML Scientist	8 years	Marketing
P7	Data Scientist	1.5 years	Marketing

Each interview was transcribed and saved into a separate Word document for thematic analysis. Themes were created based on the research questions, and codes were extracted in the process. Since we had a smaller sample size, there was no need to use a qualitative software platform, and instead, we created a table to group the main themes with their corresponding codes. A new row was created for each theme, and emerging codes were created and grouped under that theme. Each code was linked to its definition and the relevant examples from the transcribed interviews. Once the coding process started, codes were highlighted and grouped into themes. Also, we looked for emerging new meaningful patterns found in the transcripts and grouped them into new themes [34, 17]. Through some of the new emerging codes, we could identify which parts of the human-centred requirements were ignored most and what methods some of our participants were adopting when writing requirements.

5.2 Advantages of the Framework and Tool

We had positive feedback when it came to using the tool. Only some of the participants were technical in AI development. P2 was not technical and only focused on the business side of the project, and P1 was not familiar with AI development and worked on engineering the sensors and data collection. We had the most positive results from P2, and coming from a non-technical background, the models made it easier to understand the requirements for data needs and model needs. The key advantages of using our tool from the participant's point of view are summarised as follows:

Systematic Method: The first visible advantage of our framework was that it provided a more systematic way to write and model requirements for AI software. P3 explained that some of the issues they faced when building AI solutions were the lack of a systematic approach and the difficulty of putting ideas together. Having the framework provided a more organized guideline to follow and made them see things differently, "if I did that from the beginning, I think a few things would have been different regarding the system".

Holistic View: P3 reported that before using the framework, it was difficult to see which parts of the system interacted with each other. However, after getting familiar with our tool, they said that having the holistic view could allow them to see how the different features and needs interrelated with each other and helped them think differently regarding the system's needs. P1 and P2 both note that the tool helps look at the entire system first before going into the feature level. P1 explains that as developers, they always think about the system's features rather than the entire system as a whole.

Catalog: The catalogue was helpful as it got the human-centred aspects that most participants have not thought about when building their software solutions. P5 found it beneficial to have such a catalogue, especially when starting a new project, and it could be used as a checklist. P7 said it helped gather client and user requirements at the start of the project. The benefit would be to establish expectations from the stakeholder's perspective. Also, it would be clear for someone coming new to the project, as we found that sometimes people leave and new people join halfway through the project.

Model: The advantages of the model included providing a visual representation of the system requirements. P6 explained that having such detail is important when working with ML models. If the accuracy is not working as expected, they can rely on the models to visualize what went wrong in order to improve the performance. Using yellow for limitations was favoured by three of the participants. P2 mentions that "limitations are negative and having it coloured yellow made it easier to distinguish". P2 mentioned that using notations such as icons, database notations, and decisions was convenient as they could easily identify them without checking the legend. P6 said that there were aspects of the model, such as data feature engineering, that they were already performing but never explicitly visualized and having the visual element would be useful.

Assigning tasks: One participant said that the tool would be suitable for assigning tasks to different team members. P1 explained that specific tasks could be given to each person working on the project. For example, when P1 was working on data collection, they needed to know how much data to collect and what was the data rate. Using the tool, the project manager or ML specialist could set a given task for the software engineer to collect the data.

Documentation for future use: Two participants mentioned that our tool would be helpful for documentation, and having the visual aspect will make it easier to go back and revise information for future use.

Collaboration: Having a platform where everyone can work on the same project is one of the positive outcomes of the project. P2 felt that it was a good choice to use confluence. They clarify that "using confluence compared to other platforms that I have used before such as Jira was easy to use." P2 also felt that such a platform could help shorten the study's duration and clarify everything.

5.3 Limitations of the Framework and Tool

We asked the participants if there were any missing elements from the framework and how we could improve our tool. Key suggested improvements included expanding the catalogue to include a wider range of requirements and reducing the complexity of the final models. Our framework mostly focused on AI solutions using classifications. Thus, when we extracted the requirements via the catalogue, we found that parts of the User Needs were not relevant in the first case study, and the reward function would have been a loss function instead. P3 and P6 said that we needed to include other AI applications to make it more inclusive, and the way the system expects to be evaluated needs to be broader. The second suggestion to improve the framework was to provide a system where a task could be assigned to a role. The idea was to include an option to provide information specific to each role and which parts of the project they needed to complete. The third suggestion was to include a timestamp to keep track of changes. P5 explains that some questions were too early to ask, and requirements that were changed later on needed a timestamp to know when these changes were added. The advantage of having the timestamp would also help trace back the changes made "every time we lock a requirement, we add a timestamp, and when we make a change, we can know that according to the timeframe".

Key suggested improvements to the catalogue and modelling language included:

Use real-world data: P7 suggested including requirements that ensure the data is from the real world and that the project does not use superficial data. They explained that sometimes people build ML models based on available data without knowing if they use real-world data. Therefore, we need to provide assumptions in relation to the data being real data.

Use practitioner terminology: P6 felt that the terminology used in the catalogue was a bit different than the terminology used in practice. P6 explained that the terminology we used was more literature based and suggested that the terminology aspects could be improved so it would be helpful for the industry.

Use examples to explain terminology: On the contrary, P2 coming from a non-technical background, found the terminologies more challenging to understand "I had some questions and aspects I did not understand as I am not from an engineering background, and you had to clarify those to me". They suggested we provide examples in the catalogue to explain the concepts and include a comment section to provide feedback when required.

Model complexity: The major limitation to the models was that it could get a bit complex at some point "given it's an A4 window, it's a bit packed," explained P1. For example, much information was presented in the Data Needs model, and keeping track of it was difficult. Suggestions to Reduce the complexity of the final models included:

- Divide the model into smaller sub-models.
- Categorize each part of the model.
- Highlight parts of the diagram based on how it's categorized.
- Have multiple branches where each stakeholder can focus on the zone that is allocated for them.
- Draw a borderline or different colour per branch.
- Label tasks with a bounding box so that the user can see separate tasks

Modelling language enhancements: Room for improvement to the modelling language included adding more colour. P2 said that they were visual learners, and colour helped them distinguish features in the model.

6 Discussion

We applied the tool to three case studies from three different application domains. For the first two case studies, the requirements were elicited and modeled toward the end of the project. Whereas for the third case study, we elicited requirements during the early phases of the development process. We found that there were more missing requirements in the third case study. The concern was that there were a lot of unanswered questions, especially the ones related to the human-centered aspects. Also, there was limited communication between the team and the stakeholders, and some team members were unaware of what others were doing.

6.1 Missing Human-centered Aspects

Before implementing the RE4HCAI framework, we found that many human-centered aspects were missing from the final product. Most of the development was based on the technical aspects of building the AI-based system. For example, classifying what data was needed was based on technical aspects, to include model and hardware needs, and the human perspective was ignored. Most of the elements in the user needs were missing. Our participants said they had to re-consider most of the user needs only after viewing the models.

The missing significant human-centered aspects that were not discussed with the stakeholder *at the start* of the project from all the three case studies included:

- Did not consider discussing the systems limitations and capabilities with the stakeholders.
- How the user was going to interact with the AI system, and if they needed to be aware of it or not.
- Deciding on automating the system or augmenting users tasks.
- Deciding on evaluation matrices used, what are potential benefits for each method used, and calculating trade-offs to using the selected method.
- Deciding on how predictions are going to be used in making decisions and the impact of using new predictions and feedback.
- How is the model going to be adjusted to user behavior.
- Choosing the tools used to evaluate the model
- Identifying biases in data and how to avoid under or over representing data, this was not considered even towards the end of the projects.
- Reporting and addressing biases in data

Some of our participants argued that it would have been too early to identify these requirements. Some of them were more obvious such as questions related to overfeeding, underfeeding, FP, and FN. Although our participants emphasize that these could only have become known once they had enough data to identify, we argue that these should have been planned earlier. Especially aspects such as calculating the trade-off of having an FP vs. an FN or providing augmentation vs. automation. P5 explains that they only think about the impact of having a FP vs a FN towards the end, "basically, at the start of any ML experiment, we cover a bunch of models, and we feed the data into those models and carry some evaluation metric and then from that moment we start calculating the FP and the FN". However, P7 and P2 both agreed that it was essential to calculate the trade-off between having a TP and FP before starting the project, as you need to know which one has a higher trade-off, especially if you are working in the medical domain. Thus, we believe that this is something that should be established early on with the stakeholder to ensure trust.

6.2 Requirements and Methods Currently Used

When talking to our participants, we could establish some of the current methods used in RE. We found that building AI software is mostly based on trial and error, with limited structure or planning. P6 explains that although ML is mainly based on trial and error, it doesn't mean that we have to follow that method blindly. He later explained that they run their projects based on some logical reasoning. They usually start with a specific set of high-level requirements that cover a broader idea of what needs to be done. Then, as the project progresses, they derive new requirements one by one.

Furthermore, these requirements usually depend on the model's accuracy and client requirements. For example, in the planogram compliance project, image similarity recognition might work perfectly fine when tested by the team. However, when the project is deployed in the real world, and they send people to the supermarket to take photos, they might not capture them as intended. The images might be blurry, low quality, or at an angle the ML model has not been trained to recognize. Therefore, they must train on a wider range of images to improve accuracy. In reality, having that much training data is not always possible. Thus, they have to come up with other solutions to generate different variations of images using AI and send them back into the pipeline to re-train the model for more robust results.

The iterative approach of training the system is something they had to learn. However, there is the need to establish the requirements that determine how many rounds of trial and error we can have and when to stop. Both P4 and P6 argued that the time needed to ensure desirable results were subjective and depended on several factors, including 1) Establishing an agreement with the client and explaining that the results will never be 100% accurate. 2) The time frame that is given to complete the project. 3) How well the model will perform on the given dataset. 4) The allocated budget. 5) Account for failure.

In transitional software engineering, we start with the requirements and identify the features that need to be included in the system. Furthermore, establishing if a given feature is worth investing in could be known early on. However, in AI, you cannot justify the requirements or if the given feature is worth the investment. P4 explains that "you think you can do the project, but the budget just burns down, and you still don't know". Thus, before starting a project in AI, you need to demonstrate if there is any value in a given requirement and establish expectations with the stakeholder, or except failure, "and people don't like failure" explains P4.

6.3 Known Unknowns in RE4AI

We found that this project presented more known unknown requirements, such as using automation vs. augmentation or when predictions should be provided. The reason for having these unknowns was linked to two possible causes. The first one was that most of the requirements presented in our catalog were not discussed with the stakeholder early on in the project's life cycle. The second reason was associated with the black-box nature of AI and the limited knowledge of how well the chosen model will perform on the available data. Regardless of the two reasons, we argue that these presenting known unknowns should be discussed with the stakeholder to avoid having high expectations from the stakeholder's side and establish trust.

7 Conclusion and Future Work

In this paper, we presented our framework and tool for eliciting and modelling requirements for human-centred AI. The tool is used to evaluate our proposed framework which is based on the industrial human-centred guidelines, RE4AI requirements obtained from the literature, and a user survey. These results were used as a baseline to build our RE for Human-centered AI (RE4HCAI) framework. The tool was evaluated using three real world case studies and interviews with seven experts. We found that human-centred aspects were largely ignored in all three case studies, and that more requirements were unknown during the early phases of the development process, and many requirements changed or evolved as the project advanced toward deployment.

We only evaluated the first three areas of the RE4HCAI framework, i.e., user, data, and model needs. One reason for selecting these three areas was because they were more favoured by our participants in the user survey. However, we feel that the other two areas, including feedback & user control and errors & failures, are under-represented by literature and practice. Regarding our fifth area, explainability & trust, we found that most of the existing work was theoretical, with limited empirical evaluations in research. Although this area was prevalent in our survey, we did not have the time to evaluate it during this project. Since the results from the survey emphasized that all six areas should be included in RE, we propose to evaluate them in the future.

Acknowledgements

The team gratefully thanks all project participants and industry partners. Ahmad was supported by a Deakin University PhD Scholarship. Grundy is supported by ARC Laureate Fellowship FL190100035.

References

1. Agarwal, M., Goel, S.: Expert system and it's requirement engineering process. In: International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014). pp. 1–4. IEEE (2014)
2. Ahmad, K., Abdelrazek, M., Arora, C., Baniya, A.A., Bano, M., Grundy, J.: Requirements engineering framework for human-centered artificial intelligence software systems. Applied Soft Computing **143**, 110455 (2023)
3. Ahmad, K., Abdelrazek, M., Arora, C., Bano, M., Grundy, J.: Requirements engineering for artificial intelligence systems: A systematic mapping study. Information and Software Technology p. 107176 (2023)
4. Ahmad, K., Abdelrazek, M., Arora, C., Bano, M., Grundy, J.: Requirements practices and gaps when engineering human-centered artificial intelligence systems. Applied Soft Computing **143**, 110421 (2023)
5. Ahmad, K., Abdelrazek, M., Arora, C., Grundy, J., Bano, M.: Requirements elicitation and modelling of artificial intelligence systems: An empirical study. In: Proceedings of the 18th International Conference on Evaluation of Novel Approaches to Software Engineering - ENASE. pp. 126–137. INSTICC (2023)

6. Ahmad, K., Bano, M., Abdelrazek, M., Arora, C., Grundy, J.: What's up with requirements engineering for artificial intelligence systems? In: 2021 IEEE 29th International Requirements Engineering Conference (RE). pp. 1–12. IEEE (2021)
7. Amaral, G., Guizzardi, R., Guizzardi, G., Mylopoulos, J.: Ontology-based modeling and analysis of trustworthiness requirements: Preliminary results. In: International Conference on Conceptual Modeling. pp. 342–352. Springer (2020)
8. Amershi, S., Cakmak, M., Knox, W.B., Kulesza, T.: Power to the people: The role of humans in interactive machine learning. Ai Magazine **35**(4), 105–120 (2014)
9. Amyot, D., Mussbacher, G.: User requirements notation: the first ten years, the next ten years. JSW **6**(5), 747–768 (2011)
10. Anwar, M.S., Wang, J., Khan, W., Ullah, A., Ahmad, S., Fei, Z.: Subjective qoe of 360-degree virtual reality videos and machine learning predictions. IEEE Access **8**, 148084–148099 (2020)
11. Apple Developer: Human interface guidelines, [online] https://developer.apple.com/design/human-interface-guidelines/machine-learning/overview/introduction/ [Accessed 1 May 2020]
12. Arora, C., Sabetzadeh, M., Briand, L., Zimmer, F.: Extracting domain models from natural-language requirements: approach and industrial evaluation. In: Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems. pp. 250–260 (2016)
13. Baxter, P., Jack, S., et al.: Qualitative case study methodology: Study design and implementation for novice researchers. The qualitative report **13**(4), 544–559 (2008)
14. Belani, H., Vukovic, M., Car, Ž.: Requirements engineering challenges in building AI-based complex systems. In: 2019 IEEE 27th International Requirements Engineering Conference Workshops (REW). pp. 252–255. IEEE (2019)
15. Calders, T., Žliobaitė, I.: Why unbiased computational processes can lead to discriminative decision procedures. In: Discrimination and privacy in the information society, pp. 43–57. Springer (2013)
16. Chong, T., Bustan, I., Wee, M.: Deep learning approach to planogram compliance in retail stores. Semantic Scholar pp. 1–6 (2016)
17. Clarke, V., Braun, V., Hayfield, N.: Thematic analysis. Qualitative psychology: A practical guide to research methods **222**(2015), 248 (2015)
18. Dalpiaz, F., Franch, X., Horkoff, J.: istar 2.0 language guide. arXiv preprint arXiv:1605.07767 (2016)
19. Daryabeygi-Khotbehsara, R., Shariful Islam, S.M., Dunstan, D.W., Abdelrazek, M., Markides, B., Pham, T., Maddison, R.: Development of an android mobile application for reducing sitting time and increasing walking time in people with type 2 diabetes. Electronics **11**(19), 3011 (2022)
20. Davey, B., Parker, K.R.: Requirements elicitation problems: a literature analysis. Issues in Informing Science and Information Technology **12**, 71–82 (2015)
21. Easterbrook, S., Singer, J., Storey, M.A., Damian, D.: Selecting empirical methods for software engineering research. In: Guide to advanced empirical software engineering, pp. 285–311. Springer (2008)
22. Fan, C.L., Lo, W.C., Pai, Y.T., Hsu, C.H.: A survey on 360 video streaming: Acquisition, transmission, and display. ACM Computing Surveys (CSUR) **52**(4), 1–36 (2019)
23. Feldt, R., de Oliveira Neto, F.G., Torkar, R.: Ways of applying artificial intelligence in software engineering. In: Proceedings of the 6th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering. pp. 35–41 (2018)

24. Ferreira Chaves, L.W., Buchmann, E., Böhm, K.: Finding misplaced items in retail by clustering rfid data. In: Proceedings of the 13th International Conference on Extending Database Technology. pp. 501–512 (2010)
25. Frontoni, E., Mancini, A., Zingaretti, P.: Embedded vision sensor network for planogram maintenance in retail environments. Sensors **15**(9), 21114–21133 (2015)
26. Fuentes-Fernández, R., Gómez-Sanz, J.J., Pavón, J.: Understanding the human context in requirements elicitation. Requirements engineering **15**(3), 267–283 (2010)
27. Gervasi, V., Gacitua, R., Rouncefield, M., Sawyer, P., Kof, L., Ma, L., Piwek, P., Roeck, A.d., Willis, A., Yang, H., et al.: Unpacking tacit knowledge for requirements engineering. In: Managing requirements knowledge, pp. 23–47. Springer (2013)
28. Gonçalves, E., de Oliveira, M.A., Monteiro, I., Castro, J., Araújo, J.: Understanding what is important in istar extension proposals: the viewpoint of researchers. Requirements Engineering **24**(1), 55–84 (2019)
29. Google Research: The people + AI guidebook (2019), [online] Available at: https://research.google/teams/brain/pair/ [Accessed 1 April 2020]
30. Gruber, K., Huemer, J., Zimmermann, A., Maschotta, R.: Integrated description of functional and non-functional requirements for automotive systems design using sysml. In: 2017 7th IEEE International Conference on System Engineering and Technology (ICSET). pp. 27–31. IEEE (2017)
31. Grundy, J.C.: Impact of end user human aspects on software engineering. In: ENASE. pp. 9–20 (2021)
32. Hansen, J.M., Raut, S., Swami, S.: Retail shelf allocation: A comparative analysis of heuristic and meta-heuristic approaches. Journal of Retailing **86**(1), 94–105 (2010)
33. Holmquist, L.E.: Intelligence on tap: artificial intelligence as a new design material. interactions **24**(4), 28–33 (2017)
34. Joffe, H.: Thematic analysis. Qualitative research methods in mental health and psychotherapy **1**, 210–223 (2012)
35. Khomh, F., Adams, B., Cheng, J., Fokaefs, M., Antoniol, G.: Software engineering for machine-learning applications: The road ahead. IEEE Software **35**(5), 81–84 (2018)
36. Kitchenham, B., Pickard, L., Pfleeger, S.L.: Case studies for method and tool evaluation. IEEE software **12**(4), 52–62 (1995)
37. Lapouchnian, A.: Goal-oriented requirements engineering: An overview of the current research. University of Toronto **32** (2005)
38. Liu, S., Tian, H.: Planogram compliance checking using recurring patterns. In: 2015 IEEE International Symposium on Multimedia (ISM). pp. 27–32. IEEE (2015)
39. Louis Dorard: The machine learning canvas, https://www.louisdorard.com/machine-learning-canvas. Accessed [March, 2020]
40. Lucid: Lucid chart, [online] Available at: https://www.lucidchart.com/pages/
41. Maciaszek, L.A.: Requirements analysis and system design: developing information systems with UML. Addison-Wesley Longman Ltd. (2001)
42. Maguire, M.: Methods to support human-centred design. International journal of human-computer studies **55**(4), 587–634 (2001)
43. Martínez-Fernández, S., Bogner, J., Franch, X., Oriol, M., Siebert, J., Trendowicz, A., Vollmer, A.M., Wagner, S.: Software engineering for ai-based systems: a survey. ACM Transactions on Software Engineering and Methodology (TOSEM) **31**(2), 1–59 (2022)

44. Microsoft: Guidelines for human-ai interaction, [online] https://www.microsoft.com/en-us/research/project/guidelines-for-human-ai-interaction/ [Accessed 1 Feb 2022]
45. Moody, D.: The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. IEEE Transactions on software engineering **35**(6), 756–779 (2009)
46. Nalchigar, S., Yu, E., Keshavjee, K.: Modeling machine learning requirements from three perspectives: a case report from the healthcare domain. Requirements Engineering **26**(2), 237–254 (2021)
47. Nuseibeh, B., Easterbrook, S.: Requirements engineering: a roadmap. In: Proceedings of the Conference on the Future of Software Engineering. pp. 35–46 (2000)
48. Parameswaran, L., Vaiapury, K., et al.: An automated vision based change detection method for planogram compliance in retail stores. In: Computational vision and bio inspired computing, pp. 399–411. Springer (2018)
49. Ries, B., Guelfi, N., Jahic, B.: An MDE method for improving deep learning dataset requirements engineering using alloy and UML. In: Proceedings of the 9th International Conference on Model-Driven Engineering and Software Development. pp. 41–52. SCITEPRESS (2021)
50. Roberto, G.d.A., Birkbeck, N., De Simone, F., Janatra, I., Adsumilli, B., Frossard, P.: Visual distortions in 360° videos. IEEE Transactions on Circuits and Systems for Video Technology **30**(8), 2524–2537 (2019)
51. Saran, A., Hassan, E., Maurya, A.K.: Robust visual analysis for planogram compliance problem. In: 2015 14th IAPR International Conference on Machine Vision Applications (MVA). pp. 576–579. IEEE (2015)
52. Schmidt, A.: Interactive human centered artificial intelligence: a definition and research challenges. In: Proceedings of the International Conference on Advanced Visual Interfaces. pp. 1–4 (2020)
53. Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.F., Dennison, D.: Hidden technical debt in machine learning systems. In: Advances in neural information processing systems. pp. 2503–2511 (2015)
54. Shneiderman, B.: Human-centered ai. Issues in Science and Technology **37**(2), 56–61 (2021)
55. Shneiderman, B.: Human-Centered AI. Oxford University Press (2022)
56. Software, C.: Atlassian solution, [online] Available at: https://www.atlassian.com/software/confluence [Accessed 20 December 2021]
57. Software, D.: [online] Available at: https://drawio-app.com [Accessed 20 December 2021]
58. Sutcliffe, A., Sawyer, P.: Requirements elicitation: Towards the unknown unknowns. In: 2013 21st IEEE International Requirements Engineering Conference (RE). pp. 92–104. IEEE (2013)
59. Villamizar, H., Escovedo, T., Kalinowski, M.: Requirements engineering for machine learning: A systematic mapping study. In: 2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). pp. 29–36. IEEE (2021)
60. Zowghi, D., Coulin, C.: Requirements elicitation: A survey of techniques, approaches, and tools. In: Engineering and managing software requirements, pp. 19–46. Springer (2005)