Eliciting Explainability Requirements for Safety-Critical Systems: A Nuclear Case Study

***Abstract*—Trustworthy, transparent, and explainable systems are in demand, as autonomy is introduced to safety-critical systems. The minimal crossover between methods in the fields of Explainable AI (XAI) and Requirements Engineering (RE) means that there is currently no well-defined approach to building explainable systems. The high level of reliability required by systems in safety-critical environments demands rigorous approaches to requirements elicitation and system design, however the question remains whether existing RE techniques will be sufficient in eliciting explainability requirements for this purpose. We address this question by outlining an elicitation approach combining: stakeholder analysis, semi-structured interviews, and a scenario-based approach. We apply this to a nuclear robotics case study, showing that these three RE techniques are sufficient for defining explainability requirements. We present a set of explainability requirements for robotic systems in nuclear navigation and task scheduling missions, demonstrating how to extract and formalise requirements from our approach. In addition, we present the emerging taxonomy for explainability requirements and outline lessons learned for the wider research community.**

I. Introduction

Robotic systems in the nuclear sector are used to reduce the risks to human workers. Current systems are usually controlled through teleoperation for tasks such as remote handling and inspection [20]. As control is increasingly given to the robotic system, and Artificial Intelligence (AI) is integrated, it is imperative that they are trustworthy; i.e., humans (both individuals and organisations) can trust that the system is designed and implemented to mitigate harmful outcomes and produce positive outcomes while operating autonomously. One factor that supports an autonomous system's trustworthiness is the ability for its actions to be justified and explained to a diverse set of users: operators, regulators, and other stakeholders [12].

Once a robotic system has been approved for use in a nuclear facility, it is not deployed without due supervision from an operator. The responsibility ultimately falls to the operator in this safety-critical environment. Therefore it is essential that the operator has confidence in the system and deems it to be trustworthy and somewhat predictable. Including end-users in the design and evaluation of AI systems is recommended by European Union AI policy [28], and echoed by the UK's nuclear regulator [37]. We propose that explainability can support operators (and regulators) in determining whether an autonomous system's decisions are worthy of their confidence. We suggest that designing explanations in collaboration with the system's stakeholders, and with the system's context in mind, is crucial to the success of explanations to support system transparency.

The design of explanations for an autonomous system tends to focus heavily around the algorithms to be explained and the data that can be used to form the explanation; there is little focus on the users who interpret the explanation and (hopefully) trust these systems. Current approaches to designing explainable systems also have a tendency to align with the RE process of validating requirements ([48], [9]) by evaluating the quality of the explanations based on how "useful", "natural" or "robust" users perceive them to be. Testing explanations with participants is of course beneficial, but without eliciting requirements and re-engineering the explanations based on the results of these tests, this post-hoc approach does not help to engender trust in the system through its explanations [13].

Few studies directly approach end-users to understand what information they would want an explanation to include in a specific scenario before designing and integrating the explanations into an AI system. One example however is Brandão et al. [7], who propose a preliminary taxonomy for explainable systems applied to motion planning, and recognise the need to extend their taxonomy via user studies. Our approach builds on this work, taking the view that including end-users is necessary to guide the design of explainable systems. Eliciting the explainability requirements before the design stage is something that we believe to be fundamental in gaining acceptance and approval for a deployable explainable system. Brunotte et al. [10] highlight the open research question of: "whether the existing RE and [Software Engineering] activities and methods are sufficient in the case of explainability."

In this paper, we seek to answer the following questions:

RQ1: Can a combination of existing requirements elicitation techniques (stakeholder analysis, semi-structured interviews, and a scenario-based approach) be sufficient for extracting explainability requirements?

RQ2: What are the requirements for explainable robotic systems in nuclear navigation and task scheduling missions?

To this end, we present our approach for the elicitation of a system's explainability requirements in the nuclear domain, and share our recommendations for applying this approach to other safety-critical scenarios. We combine three existing RE methods: stakeholder analysis, semi-structured interviews and a scenario-based approach to a nuclear case study. In our case study we present participants with two scenarios: Navigation and Task scheduling; to elicit their requirements for explanations. We also illustrate how to extract high-level requirements from the output of our elicitation process. Given the safety-critical nature of our case study, we have chosen to write the high-level requirements using NASA's Formal Requirements Elicitation Tool (FRET), in which requirements are written in a structured natural language called FRETISH and automatically translated into temporal logic [22]. Alternatives

to FRET include EARS [40], which also supports a structured natural language, and DOORS [30], which is widely used as a database environment for managing requirements.

Overall, we contribute: 1) A user-centred approach for eliciting explainability requirements, that we demonstrate on a nuclear case study, and 2) Guidance for applying our approach to other safety-critical sectors. We provide a route to human-centred design of explainable systems, which we suggest improves the likelihood of generating understandable and useful explanations, which is particularly important for explainable systems in safety-critical scenarios that involve a human-in-the-loop. Current XAI approaches often seek user validation after implementation; crucially, our approach elicits the user's requirements *before* engaging them in the design process.

II. Background

A. Explainability

The definition of an *explanation* is open to interpretation. For example, Social Science and Computer Science have different definitions. Explanations themselves are personal in the way that humans use, understand, and accept them. Some Social Science theories suggest that the factors we use to explain intentional actions are: reasons, causal history of reasons, and enabling factors [39], [38]. Others demonstrate the use of beliefs, desires, and intentions to understand and explain human actions [31]. Miller provides an excellent discussion of how the Social Science perspective can be combined with the design of XAI [42]. However, Computer Science often focusses on the specific system components to be explained, rather than the user's requirements for explanations. For example, XAI systems are categorised depending on whether they produce local or global explanations, or whether they are model-specific or model-agnostic. See [34] for a review of interpretable Machine Learning methods. While important design features, these do not focus on the user, who must interpret and trust the explanations. Explanations can be presented in many forms, including: visual, textual, speech-based or graphical, [3] (preference is unique to the interpreter).

Defining and reliably assessing explanations lacks consensus [51]. Explainability is seen as an emerging requirement as autonomy and machine learning systems become more prevalent in critical settings [13]. The format, construction and interpretation of an explanation proves to have endless possibilities, preference of which is unique to the individual receiving the explanation. Hence, defining a user requirement for explanations and explainable systems is challenging.

B. Requirements Elicitation Techniques

Elicitation is the first step in the requirements development process [32]. Four standard steps in requirements elicitation are: 1) identify requirements from customers and wider community, 2) write requirements, 3) organise requirements into a specification, and 4) get feedback on the requirements specification from a variety of audiences [1]. These steps, however, are not yet a common practice for XAI; and user evaluation tends to occur after the system has been designed and implemented. A deep review of requirements elicitation techniques, approaches, and tools can be found in [54].

A common mistake in requirements elicitation interviews is to ignore stakeholders that were not the user/operator (e.g. regulators/technical experts) [15]. To avoid this, we use stakeholder analysis to identify all stakeholders before the interviews. We use the stakeholder roles from the Agent Ecosystem Model [50], which was designed for machine learning systems but was applicable to our (technology-agnostic) Use Case.

Interviews are one of the most common requirements elicitation techniques. We choose semi-structured interviews to balance consistency between interviews, with the flexibility to investigate spontaneous strands of conversation that may lead to unconsidered requirements. This aims to elicit explainability requirements that structured interviews might not missed.

A *scenario* is a sequence of steps that a stakeholder performs during their daily operations [26]. Scenario-based elicitation [46], [14] uses scenarios to help elicit requirements, which makes it useful for exploring the interaction between users and the system. Since the options for an explanation are vast and contrastive, we use domain-specific scenarios to focus the requirements elicitation process. This approach usually focuses on goals that users have with regards to a system, however a key part of our approach is that we intentionally **do not** consider any specific Machine Learning algorithm or AI method. By abstracting from the system-specifics, we focus on scenario-specific goals. It seemed unnecessary to restrict our scenarios to specific types of AI, and there is no predominant type of AI currently used in the nuclear sector.

C. FRETISH Requirements

The structure of the FRETISH language supports users in writing more precise requirements than they could in natural-language. FRETISH requirements have the following structure:

scope `condition` component shall timing response

Of these five fields, `condition`, component, and response are compulsory; scope and timing are optional. For example, a valid FRETISH requirement for a boiler controller could be: `when pressure > threshold Boiler` shall `satisfy open_release_valve`

FRET parses FRETISH requirements and produces a diagramatic semantics, illustrating the time interval in which the requirement should hold. FRET also translates each requirement into temporal logic, which can be used to formally prove that a model of the system's design obeys its requirements. Both of these representations are useful for sanity-checking the requirements. FRET can also export requirements directly into a contract language for Simulink [6] and a runtime monitoring framework [17]. In this paper, we use FRET purely for writing and managing requirements; we do not use these extra features, but they will be examined in our future work.

III. Requirements Elicitation Approach

This section describes our two-phase requirements elicitation approach, in Fig. 1, that we use to investigate **RQ1**.

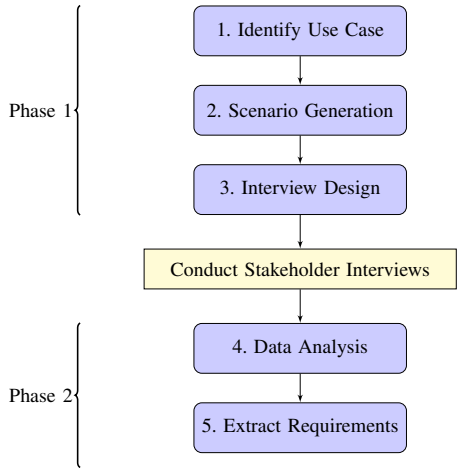

Fig. 1. Our requirements elicitation approach comprises two phases: we perform Phase 1 before the elicitation interviews and Phase 2 afterwards. Rounded-edge boxes denote steps.

Phase 1 prepares for the elicitation interviews with the system stakeholders, and Phase 2 consumes these interview results to produce high-level requirements. In Phase 1, we perform three steps: 1) identify the Use Case, 2) generate the scenarios, and 3) design the interview(s). Phase 2 comprises two steps: 4) analyse the interview data, and 5) identify the high-level requirements from the interview data. Explainability requirements emerge in Phase 2. The individual RE processes used in Fig. 1 are not unique. However, their combination has not been used in the elicitation of explainability requirements before. The steps in Fig. 1 mirror common RE processes, which means that our approach can be easily integrated into processes that elicit (for example) functional requirements, to form an approach eliciting multiple types of requirements at once. In this paper, we focus on how the approach elicits explainability requirements. We describe these steps in more detail below.

A. Step 1: Identify Use Case

Identification of the context of Use Cases includes defining the users, the task, the system, and its operating environment. As part of this step, we conduct a stakeholder analysis to identify the stakeholders, clarify their interests, and validate their position in the ecosystem. We assign the stakeholders to one of the six stakeholder roles identified by Tomsett et al. [50]: *creators*, *operators*, *executors*, *decision-subjects*, *data-subjects*, and *examiners*. This step enriches our understanding of the Use Case, providing detailed knowledge of the tasks and stakeholders. It also maps the stakeholders to one of the six stakeholder roles.

B. Step 2: Generate Scenarios

This step generates scenarios, starting by defining the problem context of a scenario to be explored. This is a natural-language description of the problem that we develop into a scenario suitable for stakeholders. From this, we extract the scenario's setting, actors, actions, events, and goals; using Rosson and Carroll's scenario analysis model [45], [11]. These elements define all the necessary aspects of the problem context, so we can derive an appropriate contextual description to present to interviewees. In our approach, we explore providing an example explanation to the participant after asking what information they want, to understand if this helps them express their needs. See §IV-B for an example of this in our case study.

C. Step 3: Design the Interview Guide

We produce the interview question guide to ensure consistency between interviews. The guide sets the scenarios from Step 2 (§III-B) within the traditional RE technique of a conversational semi-structured interview. The interview guide should be carefully followed, to give consistent data; and be used flexibly to elicit previously unconsidered requirements. This produces an interview design suitable for eliciting the explanation requirements for the system in the Use Case.

To design the interview questions, we reflect on the stakeholder analysis in §III-A. It is important at the beginning of the interview to identify the stakeholder settings, to reaffirm their position in the agent ecosystem model described previously. Following this, we suggest some high-level questions relating to explainability. As we consider safety-critical systems in this work, we propose a catalogue of questions about safety-related explainability. Finally, we suggest asking task-related explainability questions. The interview question guide that we use for our case study can be seen in Table IV.

D. Step 4: Data Analysis

After requirements elicitation interviews, we analyse the gathered data. We suggest using an inductive thematic analysis, a standard qualitative data analysis technique [8], [41], which reduces subjectivity. An important factor here is to distinguish between information and explanation; not all of the information that a stakeholder says that they want to know will become an explanation. For example, a stakeholder may request a 'continuous update on the location of the robot/system'. This may form a requirement in its own capacity but, we do not consider this to describe the outcome or decision making justification, therefore it is not an *explanation* and consequently will not form an *explainability requirement*.

Related work [49] explores whether explanations should be designed and constructed using concepts from human psychology. They compare their findings to the conceptual theory of how intentional and unintentional behaviours are explained [38]. Using this approach, requirements engineers work backwards from the information that people think is useful to garner from an autonomous system, align this with how it could form an explanation, and consider if it would be beneficial to form them in line with theory from human psychology. This perspective forms part of our data analysis.

This step is necessary but not unique to requirements elicitation, rather it is a data pre-processing step that condenses the large amounts of data gathered during interview.

E. Step 5: Extract Requirements

In this final step, we extract the explainability requirements from the data. These requirements can be used to guide the

design and verification of the explainable system. As previously mentioned, this step does not depend on any particular requirements specification language, but we demonstrate this step using FRETISH. The structure of FRETISH (see §II-C) gives enough detail for requirements specification, but enough flexibility for free-flowing conversations with domain experts. This reduces ambiguity in the resulting requirements set [19].

The requirements for the explanations that the system should provide should cover all stakeholder groups included in the interviews. Some groups may have differing or conflicting explainability requirements, which must all be supported by the final system to ensure that the explanations are useful and safe for each stakeholder group. Because RE is an iterative process, the requirements should be validated with stakeholders to ensure that they accurately represent stakeholder needs.

IV. Case study: Phase 1

This section describes how we applied our requirements elicitation approach (§III) to a nuclear industry case study.

A. Step 1: The Nuclear Use Case

As mentioned in §III-A, a key part of Step 1 in our approach (Fig. 1) is to develop an understanding of the Use Case's context. It is also necessary to define the stakeholders.

1) Setting the Context: Our case study is a robotic system inspecting parts of a civil nuclear facility where the radiation is hazardous to humans. This use is beneficial because it keeps humans away from the hazardous radiation, but the design and implementation of the system must also be able to demonstrate that it will make safe and reliable decisions.

The nuclear industry is, necessarily, heavily regulated and safety-orientated. This leads us to believe that a structured RE approach to designing explainable systems is necessary; for example, including stakeholders in the design process. Regulators, such as the UK's Office for Nuclear Regulation (ONR), must be able to check that the system is safe and reliable enough to be used on-site. Guidance for developing safe and trustworthy autonomous systems is steadily emerging. This includes within the European Union [27], [28]; the IEEE standards on transparency [52] and fail-safety [18]; from the ONR [37], or derived from that work [35]; a recent workshop [2] for the Robotics and AI in Nuclear (RAIN) research project[1]; or elsewhere in the literature [21]. We suggest that explainability is a route to demonstrate to operators, regulators like the ONR, and other stakeholders, that the system is reliable and safe. It also simplifies the task of determining what went wrong when the system fails.

2) Stakeholder Analysis: Different stakeholders may have conflicting explainability requirements. For example, a creator who is debugging the system may want more detailed explanations than an operator. For our study, we interviewed 16 stakeholders from the nuclear industry, who were experienced in the design or deployment of robotic systems for nuclear environments. We apply the ecosystem model from [50], to identify our stakeholders' roles (Table I). The ONR have the role of the examiners, and our other stakeholders are: creators (who create/implement), operators (who directly interact with) and executors (that make decisions based on the output) of a potentially explainable system. We validate our stakeholder analysis during interviews by including semi-structured questions that identify stakeholder settings (Table IV).

[1]RAIN Hub: https://rainhub.org.uk/

TABLE I
Summary of participant stakeholder roles

Role	ID	Participant Job Title
Creator	P2	Lead Software and Electronics Engineer
Creator	P8	Post Doctoral Researcher
Creator	P12	Research Fellow
Creator	P13	Post Doctoral Researcher
Creator	P14	Senior Robotics Researcher
Creator	P15	Electrical Engineer
Creator	P16	Post Doctoral Researcher
Operator	P4	Remote Handling Operations Engineer
Operator	P7	Remote Handling Operations Engineer
Operator	P11	Lead Technologist, Remote Handling Operations
Executor	P1	Professor/Project Leader
Executor	P3	Consultant Project Manager
Executor	P5	Head of Nuclear
Executor	P6	Managing Director of Robotics company
Executor	P9	Team Leader of Robotics and AI Research
Executor	P10	Head of Research

B. Step 2: Navigation and Task Scheduling Scenarios

Our study used two scenarios: Navigation and Task Scheduling. We purposely designed the scenarios to be open to interpretation on a number of levels, for example: how the system is controlled, its level of autonomy, etc. Conversely, we have well defined tasks for each scenario, which were provided to participants. These tasks, and the scenarios themselves, are representative of the proposed uses of these systems in the nuclear industry [20]. For each scenario, we outline its problem context, the derived scenario analysis model, and the contextual description (including an example explanation, see §III-B) that was shown to participants in the interviews.

1) Scenario 1 (Navigation): The first scenario sees an autonomous robot navigating inside a nuclear facility.

a) Problem context: Autonomous robots in the nuclear industry must behave safely and be trustworthy as they move around the environment. Robots may be sent into rooms that have not been entered for a long time, where it is unlikely that an accurate map of the environment exists. There may be unknown debris and obstacles that the robot must avoid, or unstable radiation sources.

b) Scenario Analysis Model: We define the scenario analysis model (Table II). Here, we outline the setting, actors, actions, events and goals for Scenario 1. The following contextual description was given to each participant.

c) Contextual Description: "An autonomous robot is being deployed into a nuclear plant room. Fig.2(a) shows the topological map of the room. The circles v1, v2, etc. represent the waypoints where the robot may have tasks to perform, and the blue lines represent the paths between the waypoints in which the robot can navigate."

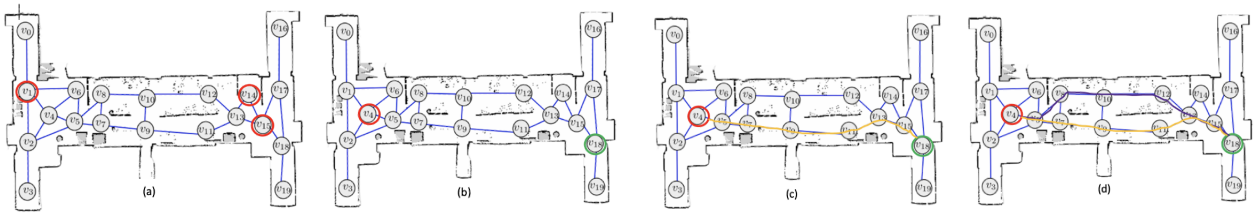

Fig. 2. Diagrams showing modified images of topological maps from [33] for illustration purposes. These were provided to participants for Scenario 1: Navigation. The red circles indicate the locations in which the robot will have to travel to, and the green circles represent the robots current location

TABLE II
SCENARIO ANALYSIS MODEL FOR SCENARIO 1: NAVIGATION

Element	Description
Setting	Plant room within a nuclear facility (see Fig. 2)
Actors	Autonomous robot, operator overseeing the robot
Actions	Navigate and perform tasks at specific locations
Events	Deviation of original plan or unexpected behaviour
Goals	Safely and successfully complete tasks and report progress to operator

TABLE III
SCENARIO ANALYSIS MODEL FOR SCENARIO 2: TASK SCHEDULING

Element	Description
Setting	Nuclear facility
Actors	Autonomous robot, operator overseeing robot
Actions	Autonomously schedule and execute a set of tasks provided by the operator
Events	Unexpected change to schedule
Goals	Report the schedule and progress to the operator

In each of the following situations, participants were asked what information they would want from the robot, when it:

- has tasks to perform at the three circled waypoints shown in Fig. 2(a).
- is located at the green circle, v18, and it needs to navigate to the red circle, v4 to perform a task, shown in Fig. 2(b).
- has predicted its optimal navigation route as the route marked in yellow in Fig. 2(c).
- has executed its navigation but takes a different route to what was predicted, marked in purple in Fig. 2(d).

Example explanation given by the robot to participants: When trying to execute the action to go to v11 from v13 an obstacle caused redirection to arrive at v12 instead. Given this navigation failure and that the optimal action from v12 is go to v10, a new optimal navigation trajectory was calculated and executed to reach the original goal.

2) Scenario 2 (Task Scheduling): An autonomous system that can schedule and execute its own tasks.

a) Problem context: Robots are introduced to perform specific tasks, such as visual inspections, taking sensor readings, and moving or removing objects. Often a human (or now a robot) must perform the same repeated tasks every day, week or month. It would be beneficial (and possibly safer) for the robot to perform mundane repetitive tasks autonomously. This *long-term autonomy* would be possible if the system could schedule and execute tasks independently [25].

b) Scenario Analysis Model: Following the same structure as the previous scenario, we define this scenario analysis model in Table III.

c) Contextual Description: "The robot has 4 more tasks to complete by 5pm today. It calculates that it has 90% probability of completing the next 2 tasks and 50% probability of completing all 4. The robot executes Task 1 and Task 2 successfully. The robot begins Task 3 and aborts the task before completion to begin Task 4."

Similar to Scenario 1, each participant was then asked what information they would want to know from the robot.

Example explanation given by the robot to participants: I began execution of Task 3 which was taking longer than expected. I calculated that if I completed Task 3, I would have a 20% probability of completing Task 4 by 5pm, however if I abort Task 3, the probability of completing Task 4 would rise to 75%. Task 4 is flagged as higher priority therefore I aborted Task 3.

C. Step 3: Interview Procedure and Question Guide

We used semi-structured interviews, where we suggest asked questions like *'what do you want to know?'* about scenario settings, rather than asking about explanations specifically. This approach avoids preconceptions biasing the interviewees responses, similar to the approach in [44].

In contrast to the detail-orientated questions relating to the scenarios, we design the remaining questions for the interviews with higher-level concepts. The question guide (Table IV) identifies stakeholder settings, and considers areas of importance to the nuclear industry, such as safety and task-related applications. Using high-level questions helped us to understand the general context of opinions about explainability; in contrast to the scenarios, which focused on specific circumstances where explanations could be used. By combining open, high-level questions with specific scenario related questions, we gather a rich set of requirements.

V. CASE STUDY: PHASE 2

A. Step 4: Analysing the Data

We gathered raw qualitative data in the form of audio recordings[2]. We split the data into the responses to: 1) the general interview questions, and 2) the scenarios. We used

[2]The data were transcribed and evaluated in NVivo 12.

TABLE IV
INTERVIEW QUESTION GUIDE

1. Identify Stakeholder Settings	
1.1	What is your current job?
1.2	How long have you worked with robotics or in the nuclear industry?
1.3	What experience do you have in the deployment of robotic systems into nuclear facilities?
1.4	What sort of information do you feel is useful or lacking when using the deployable robots?
2. Explainability Focused Questions	
2.1	Is there anything that comes to mind from your experience that you think explanations would be useful for in terms of the nuclear industry and the robots that are being deployed?
2.2	What are your thoughts on explainability being used within the nuclear industry for deployable robots? Advantages/Disadvantages?
2.3	What type of explanation do you think would be useful? i.e. policy info, visual explanation, human-like explanation?
2.4	Does your opinion change based on whether the outcome is expected or unexpected?
2.5	Over time, would the amount of information you want from an explanation change?
3. Safety Related Questions	
3.1	If the robot did something unexpected that caused a safety concern, would this change the amount/type of information you would want?
3.2	What impact do you think explainability would have on safety and safety regulations for nuclear deployment?
3.3	Are there any immediate safety concerns or benefits?
3.4	Do you think explainability will have impact on training operators to use robots?
4. Task Related Questions	
4.1	What situations could you see explanations being useful, if any?
4.2	Any specific types of application/robots?

inductive thematic analysis to understand whether the information that participants want aligns with the way that humans explain behaviours. The interviews were coded and analysed by the lead researcher and this was cross-validated with one further researcher to ensure reliability and reduce bias. We found that participants wanted to know *why* and *how* a decision was made, and *what* the decision it was based on. We classify our data into the following four themes: 1) *Application of explanations:* how explainable systems could fit into current projects and tasks, 2) *Impact of explanations:* the benefits and drawbacks of using explanations in autonomous systems, 3) *Explanation Content:* what information the explanations should contain, 4) *Explanation Format:* the explanation's level of detail, medium (e.g graphical or textual), timing constraints, and if operators can intervene if they disagree with a decision.

The thematic analysis and categorisation of data allowed for a more straightforward extraction of requirements. Raw data segments were then extracted where requirements were evident, and specified using FRET[3]. Our analysis also produced sub-themes for four themes mentioned above, but they are omitted for brevity.

B. Step 5: Extracting Requirements

This section describes how the requirements were extracted from the data analysis results and encoded into FRETISH (Table V). We present an example of how we extracted Natural-Language Requirements (NLR's) from a a raw interview quote, and the FRETISH version of the requirement.

[3]Raw data related to this publication cannot be openly released; although anonymised, the interview transcripts contain deployment information meaning participants may be identified by others in the industry.

Raw Quote (from Operator)

"I'd want to know at the point where the paths separate, that its changed [referring to the navigation scenario]. And that maybe when you get to that split point that you have a hold point that requires somebody to say yes. Okay. Carry on. But you want to know why the robot has decided to change the original plan."

Natural Language Requirements

REQ-1: When the system detects a route change, the autonomous system shall explain to the operator why the route has changed.
REQ-2: If the route changes, the autonomous system shall stop and seek approval of the new plan.

FRET Requirements

REQ-1: `if routeChange & infoRequest Robot` shall `at the next timepoint satisfy provideReason`
REQ-2: `Robot` shall `before proceed satisfy requestPermission`

This extract approach was repeated, to produce the full FRETISH requirement set in Table V.

VI. OUR TAXONOMY FOR EXPLAINABILITY REQUIREMENTS

We propose *explainability requirements* as a new category of non-functional requirements, to capture the complex preferences that come with explanations from autonomous systems. The standard categories of non-functional requirements are: Usability, Reliability, Performance, Scalability, Security, Compliance and Quality requirements [23]. Tackling ignored non-functional requirements is among the most costly and difficult activities in a software project [43]. Non-functional requirements are often ignored in requirements elicitation interviews, especially by interviewers lacking specific training [15]. Explainability requirements are more likely to be ignored, because it is an emerging requirement type.

During analysis of the elicited explainability requirements, further categories emerged, therefore a preliminary taxonomy for explainability requirements is presented:

1) Characteristics of the explanation itself
2) Timing-specific requirements
3) Abstraction-specific requirements
4) General content of an explanation
5) General circumstance when an explanation is required

Table V shows the categories each requirement belongs to.

VII. ANALYSIS

A FRETISH requirement's component defines the part of the system to which it relates. Our requirements set has two components: `Robot` and `Explanation`. The majority of the requirements relate to the `Robot`, however we also found requirements about the `Explanation` itself. For example, some requirements refer to the level of abstraction or the characteristics of an explanation (e.g. Category 1 and 3 in Table V). This is unusual for FRETISH requirements, because the components are usually within the system. To our knowledge this is the first time FRETISH has been used to capture explainability requirements.

TABLE V
FRETISH REQUIREMENTS

ID	Category	FRETISH Requirement
R.001	4	Robot shall before mission satisfy explainHealth
R.002	4	Robot shall before mission satisfy explainSuitable
R.003	4 / 5	Robot shall always satisfy monitorHealth & ((health<threshold) => explain)
R.004	4 / 5	in manipulation mode Robot shall immediately satisfy explainObject & Future(explainMaterial) & Future(explainCut)
R.005	4	Robot shall always satisfy explainLocation & provideReason
R.006	4	upon assessment Robot shall at the next timepoint satisfy provideReason
R.007	5	if areaAvoided Robot shall immediately satisfy explain
R.008	5	in pathPlanning mode Robot shall always satisfy justifyDecision
R.009	4	in manipulation mode if objectPlaced Robot shall at the next timepoint satisfy explainObjectLocation
R.010	5	if failureDetected Robot shall immediately satisfy diagnose & explain
R.011	1	Explanation shall always satisfy Truthful
R.012	1	Explanation shall always satisfy Factual
R.013	1	Explanation shall always satisfy Verifiable
R.014	1	if !clearance Explanation shall never satisfy shareSensitiveInformation
R.015	3 / 5	when timeCritical Explanation shall immediately satisfy concise
R.016	3 / 5	in training mode Explanation shall always satisfy detailed
R.017	3 / 5	in performanceReview mode Explanation shall always satisfy detailed
R.018	3 / 5	Robot shall always satisfy ((unexpectedBehaviour & detailRequested) => detailedExplain) | (!unexpectedBehaviour => explain)
R.019	3 / 5	if situationAssessment=>lowRisk Explanation shall satisfy detailed
R.020	5	when decisionMaking Robot shall eventually satisfy explain
R.021	3	in systemDiagnosis mode Explanation shall satisfy detailed
R.022	3 / 5	if safetyConcernDetected Robot shall at the next timepoint satisfy justifyDecision
R.023	5	if problemDetected Robot shall eventually satisfy explain
R.024	5	if priorities=>changePlan Robot shall at the next timepoint satisfy explain
R.025	4	Robot shall always satisfy explainTaskProbability
R.026	5	if unexpectedObstacle Robot shall always satisfy explain
R.027	5	if timeConstraints=>changePlan Robot shall at the next timepoint satisfy explain
R.028	4	Explanation shall satisfy robotLocation & taskCompletionStatus
R.029	4	in pathPlanning mode Robot shall always satisfy explainPath & justifyDecision
R.030	5	if taskCannotComplete Robot shall immediately satisfy explain
R.031	4	Robot shall always satisfy calculateTaskCompletionProbability & explain
R.032	5	if probabilityTaskCompletion < previousProbabilityTaskCompletion Robot shall immediately satisfy explainAlternatives
R.033	4 / 5	if taskAbort Robot shall immediately satisfy explainFailSafe
R.034	4	if selfRecoveryExists Robot shall eventually satisfy explainRecoveryStrategy
R.035	2 / 5	Robot shall before go satisfy explain
R.036	2 / 5	Robot shall before proceed satisfy requestPermission
R.037	2	if missioncomplete Robot shall at the next timepoint satisfy explain
R.038	5	if unexpectedCondition Robot shall immediately satisfy notifyUsers
R.039	2	Robot shall always satisfy failureDetected => failSafe
R.040	3	in developer mode Robot shall always satisfy produceDetailedExplanation
R.041	4 / 5	if safetyConcernDetected Robot shall at the next timepoint satisfy verifyAction
R.042	2 / 4 / 5	if routeChange & infoRequest Robot shall at the next timepoint satisfy provideReason
R.043	5	if planDeviation Robot shall immediately satisfy returnCurrentLocation
R.044	4	Robot shall always satisfy (explainCurrentPath => returnCurrentPath) & (explainIntendedPath => returnIntendedPath) & followPath & noDeviations
R.045	4	Robot shall always satisfy (explainVisited => returnVisited) & (explainMeasurements => returnMeasurementOrigins)
R.046	4 / 5	if notOptimal Robot shall immediately satisfy explain
R.047	4	if pathImpossible Robot shall immediately satisfy returnAllOptions
R.048	4	Robot shall always satisfy provideHealthUpdate & selfRecoveryStrategyConfidence > acceptableThreshold
R.049	4	Robot shall always satisfy avoidObstacles & minimiseRadRisk & sampleDoseRate
R.050	5	if abort Robot shall immediately satisfy explain
R.051	4	when probComplete >= p Robot shall immediately satisfy returnUncertaintyVariables & explain

A. How Do Stakeholder Responsibilities Drive Requirements?

Different stakeholders may have conflicting explainability requirements. For example, a creator who is debugging the system may want more detailed explanations than an operator. This difference was observed in the requirements. The following modes were identified in the requirements which represented requirements specific to stakeholder roles: in training mode, in performanceReview mode, in developer mode and in systemDiagnosis mode. These predominantly influenced the level of abstraction of the required explanation. For instance, R.016 and R.017 specify that detailed explanations are required in training mode and in performanceReview mode respectively. In summary, it was evident that concise explanations are required at most points, and in particular when an *operator* stakeholder is using the system, however in circumstances relating to the modes listed above, the requirements indicate more detail is required. We anticipate that the exact detail sought might also differ depending on the role of the person involved.

B. Cross-Theme Requirements

Some requirements crossed multiple categories. For example, R.003, R.004, R.033, R.041, R.042 and R.046 all crossed *Content* and *Circumstance* categories. This indicates a correlation between the circumstance and the required content of an

explanation. Further, a strong correlation exists between the circumstance and the level of abstraction of the explanation. Requirements R.015, R.016, R.017, R.018, R.019 and R.022 all crossed *Abstraction* and *Circumstance* categories. R.035 and R.036 cross *Characteristics* and *Circumstance* categories. Although R.036 is not necessarily an *explainability requirement*, it refers to the behaviour that the `Robot` should exhibit after providing an explanation and is therefore still included.

R.003 stated that 'the robot should continually assess its health and performance, and assess if it is able to undertake the task, if it decides it is not able to, it should explain why not'. This was translated into '`Robot` shall `always satisfy monitorHealth & ((health<threshold) => explain)`'. The threshold would need to be defined later in the development process before implementation.

Requirement R.004 is interesting because it requires Linear Temporal Logic (LTL). In FRET, only the scope, `condition` and timing terms affect the LTL, and in this case it is the `condition`. In R.004, it is always required that the robot will immediately explain the object, however the timing of when the material and cut is explained is ambiguous depending on when the task is undertaken. This is denoted by the term 'Future'. We use Future here since not all of the boolean responses happen necessarily at the same time.

R.015 stated the explanation should be 'direct and concise' however the term *direct* is not really definable. It is assumed that all explanations will be *direct* in nature, and we understood that the stakeholder was implying a *direct* so, therefore this was replaced with `immediately` in the formal requirement with a similar effect.

R.018 provides an interesting point for discussion as it differs slightly from the other requirements. This requirement was difficult to define from the NLR as it stated the requirement for the ability to 'dig down' into more detail of the explanation. This was denoted by applying an 'OR' condition in the response, to represent the distinction between when a lower level of abstraction is required.

C. Taxonomy Categories

1) Explanation Characteristic Requirements: The main emerging characteristic requirements for an explanation were that it is *truthful* (R.011), *factual* (R.012), *verifiable* (R.013) and the explanation should never share sensitive information to people who were not cleared to receive it (R.014). These requirements highlight the importance of safety and contribute overall design considerations for developing explainable systems. These are most likely applicable requirements for any explainable system, but they remain high level, so depending on the system these would need to be investigated further.

Related to trust and explainability are requirements in support of *transparency*. A more transparent system is likely more amenable to explainability. Hosseini et al. have deliniated four facets of transparency [29]including transparency stakeholders, transparency meaningfulness, transparency usefulness and information quality of transparency. Interestingly, in the category of transparency usefulness, information perception and understandability are listed as key steps. Perception and understandability also form part of explainability so transparency and explainability requirements are clearly related.

2) Timing-Specific Requirements: Explanations' timing constraints emerged as an important factor in our elicitation interviews. In most cases, the timing constraints were clear; however, some NLRs were ambiguous about timing, which presented a challenge for translation to FRETISH. Our requirements use the timing constraints: `before`, `at the next timepoint`, `always`, `immediately`, and `eventually`. We use the '`always`' constraint to indicate a capability or type of explanation that shall *always* be available, after the requirements triggering condition becomes true. It does not refer to there being a constant explanation output. Using this pattern to capture the requirements abstracts from the mechanism of the user asking for the explanation. The timing constraints for explanations are important, so they should be validated before specifying the system's design.

Some time-critical, requirements were identified. For example, R.042 specifies that a response should happen at the next time point. Du et al. [16] studied explanation impact on drivers' trust in autonomous vehicles. They found that explanations were most effective when given before an autonomous vehicle acted. This supports R.035 and R.036, where stakeholders required the `Robot` to stop, provide an explanation, and wait for confirmation or approval before proceeding. It is important to note that R.036 is not necessarily an *explainability requirement* by itself, however it was required in tandem with the explanation and thus is included.

Requirements emerged relating to the timing of an explanation that call for further analysis. For example, R.035 specifies that the robot should explain before continuing the mission, where R.037 states that the robot should explain at the end of the mission. These two requirements could potentially both be satisfied in the same system, however the stakeholder role influenced these requirements and therefore needs further investigation. It was predominantly found that 'operators' preferred explanations as specified in R.035 whereas the preferences varied in the other stakeholder roles.

3) Abstraction Specific Requirements: As mentioned in §VII-A, an explanation's level of abstraction is influenced by the role of the stakeholder who is asking for the explanation. In FRETISH, we used the scope field to describe these roles as different system modes. Some requirements were identified to be abstraction-specific. For example, R.040 relates to when the system is `in developer mode` because developers require more detail in explanations compared to system operators. The level of abstraction is also influenced by specific `condition`s. For instance, R.015 specifies `when timeCritical` a concise explanation is required, in contrast to R.019 which specifies that `if situationAssessment=>lowRisk`, a more detailed explanation is required. We found a clear need for varying levels of abstraction, influenced by multiple factors, which would be interesting to explore in future work.

4) Content-Specific Requirements: This category of requirements predominantly supports an explanations' con-

tent. For example, R.033 states that the robot should explain what fail safes are in place. Some of the requirements state a generic explanation response; for example, '`satisfy explain`' in R.046, where further detail would be provided later in the development process; whereas others specify the exact content required, for example R.034 states '`satisfy explainRecoveryStrategy`'. Other responses emerged relating to the explanation, for instance: `satisfy explainHealth`, `satisfy explainVisited` and `satisfy explainLocation`. These provide more context to the explanation, and requirements that only state '`satisfy explain`' would need further investigation in the next iteration of the requirements process. However, some explanation specifics can be inferred from the `condition` associated with it, for example, R.050 states `if abort Robot` shall `immediately satisfy explain`, which implies the explanation should contain the reason for aborting, An interesting content requirement was `satisfy provideReason`, which states that the explanation should contain the *reason* for the decision rather than the occurrence itself. There was also an interesting split between requirements that should `satisfy explain` a past action, and those that should `satisfy justifyDecision` to explain future actions. These potentially map to the *'why did you'* and *'why can't you'* questions, respectively; which enable humans to query why the robot performed an action (*'why did you take that route?'*) or what is preventing the robot from taking an action (*'why can't you move forward?'*) [36].

R.008 and R.029 provide an interesting comparison of 'explain' with 'justify'. R.008 states that the robot should justify its path choice. R.029 states that it should explain its path choice *and* justify why it is the optimal choice. Our analysis links these to Malle's folk theory of explanations [39], where R.008 requires a *Reason* to a *'why not?'* question and R.029 requires a *Causal History of Reason* explanation to satisfy a *'why?'* and *'why not?'* question. Our analysis also supports the findings in [49]. This relates to the abstraction of the explanation and its content. The timing of these are both marked as `always`, however this is stakeholder dependent.

5) Explanation Circumstance Requirements : This is the largest category in our requirements set. These requirements describe the circumstances where an explanation is required, at what time point, and at what level of detail. The requirements in this category are likely to be context-specific and may vary depending on the use case. In the scenarios presented during elicitation, it emerged that explanations were required when the robot satisfies `condition`s such as: `areaAvoided`, `problemDetected`, `unexpectedCondition` and `abort`.

There is an intriguing comparison to be made between R.010 and R.030. It is interpreted that R.030 refers to when a task is incomplete, and R.010 refers specifically to when a failure occurs. However, it is possible that the task cannot complete *because* of a failure, and thus, a connection between the requirements is probable. Furthermore, it is noted that R.010 requires `satisfy diagnose & explain`, while R.030 requires only `satisfy explain`. This shows that the indication of a failure requires more information. Hall et al. [24] suggest that safety-critical applications would have different explainability requirements to a non-safety critical application. Although both R.010 and R.030 refer to a safety-critical application, the potential increase in risk factor indicated between R.010 and R.030 correlates to Hall et al.'s suggestion.

VIII. Related Work

Factorial Survey Experiments (FSE) [4], [5] prompt participants to respond to *vignettes* describing a situation involving the elements of interest to the survey. By avoiding asking about the elements directly, FSE aims to side-step social desirability bias. This is similar to our work, where we do not directly reference explanations when presenting the scenarios to participants until the end where we show them an example explanation. We discuss this further in §IX.

Works such as [14] and [53] propose scenario-based requirements elicitation methods for XAI design, but do not elicit requirements. Scenario-based elicitation techniques are used in RE to identify stakeholder needs and are considered 'problem-centred'; a preferred method to avoid a solution-first approach [46]. Other explanation requirements, like medium (e.g. visual, textual, speech-based, graphical) also vary with the explanation's interpreter [3], so these requirements should also be elicited from the system's stakeholders.

Chazette et al. provide a systematic literature review and a definition of explainability [12]. This work focuses on the emergence of explainability as a non-functional requirement that can have a significant impact on system quality. They note that explainability needs further attention from the RE community. They also emphasise how explainability both positively and negatively impacts other quality aspects including verifiability, trustworthiness and correctness.

Recent work defines a quality framework for eliciting explainability requirements which links the dependencies, characteristics and evaluation methods for explainability requirements [13]. This work examines how to construct explanations to increase frequency of use, system acceptance and user satisfaction. The stakeholders in their case study have loosely defined roles, in contrast to our case study where stakeholders are clearly identifiable. It would be interesting to see the impact of their approach on our case study.

As previously mentioned, [15] highlights nine common mistakes when conducting requirements elicitation interviews, one of which is ignoring non-functional requirements. Our work specifically focusses on the non-functional requirement of explainability, and we avoid the other mistakes as much as possible during our interviews. A follow on study from [15] in [47] highlights that ambiguities in interview results often suggest tacit knowledge that was not transferred to the requirement specification. It is suggested, in [47], that identifying these ambiguities after the interview can be used to identify this tacit knowledge. Reflecting on our work, there could be ambiguities in our interview results that conceal tacit knowledge; this is an area we could improve in future studies.

IX. Lessons Learnt

We present five lessons for the reader when applying our approach to requirements elicitation.

Lesson 1: Framing of Research Questions During the presentation of scenarios, the use of broad open-ended questions enabled stakeholders to freely communicate all relevant information without limitations. This was later filtered to focus on explanation-related information. This proved to be a positive feature of the approach. Despite the increased data-processing effort, it served as an effective means to avoid unintentionally overlooking any explainability requirements. Consequently, it is recommended that readers adopt a similar approach when designing their scenarios.

Lesson 2: Give Example Explanations It was found that providing example explanations allowed stakeholders to express their requirements more easily. This was invaluable in giving context of what an explanation from an intelligent system might be and although it didn't directly impact the requirements, it helped stakeholders to validate their thoughts. This step is not essential to the elicitation approach, however we observed direct benefits for the stakeholders to be able to describe their needs through contrasting with the example.

Lesson 3: "Explain" is not always an Explanation Some stakeholders used the word 'explain' to express that they were interested in the value of a variable at a given point in time. It is important to distinguish between explanations and information; not all of the information that a stakeholder says that they want to know will become an explanation. Step 4 in our approach pre-processes the data, to separate explanations from information requests. The thematic analysis and categorisation of data allowed for a more straightforward extraction of requirements.

Lesson 4: Requirements *before* System Definition It was evident in the study that people often don't know enough about how explanations are generated to comment on what kind of explanation they want. By asking what they want to know about the system's behaviour, the information can be collected in a less biased way and connected to explanations later. Stakeholder expectations do not always align with realistic system explanations. Therefore, it is recommended, if possible, not to restrict requirements elicitation to a specific ML/AI algorithm, hence allowing expectations to be more achievable.

Lesson 5: Conflicting Requirements and Preferences It emerged that explanation preference is unique to the individual interpreting it, supporting theories in the social science literature. For example, explainability requirements were conflicted between creators and operators, most predominantly in the explanation format, e.g. the level of detail wanted from an explanation. We suggest this needs carefully considered during the design of explainable systems, to support the needs of all stakeholders. However, a coherence was found in the content of an explanation, aligning with the social science perspective of an explanation containing beliefs, desires and intentions. Resolving any conflicts should be approached carefully to support the needs of all stakeholders.

X. Limitations and Threats to Validity

Firstly, we did not have an equal number of participants for each of the stakeholder roles (Table I). There were more than double the number of 'Creator' stakeholders than 'Operators'. To reduce this threat to our approach in future work, we aim to validate our requirements with a more balanced set of participants. We also recognise that the interviewed stakeholders may not be representative of those in the nuclear sector at a national or international level, however we attempted to mitigate this concern by gathering a representative set from our available network. The participants included academic and industrial researchers, and employees from private nuclear operators, giving us a diverse pool of participants.

Our scenarios were motivated by areas of research from the RAIN project and serve as a small (but representative) subset of nuclear applications where explainable systems could be used. We focused on navigation and scheduling use cases, but there are other autonomous robotic systems in nuclear that could be analysed e.g. cutting open barrels or waste disposal that may present interesting explainability concerns and requirements that our scenarios did not encounter. The requirements of explanations for these systems may differ but we believe that the basic, high-level requirements for explainability are likely to be similar. Our approach should be validated on other scenarios. Investigating other such use cases is left as future work.

Notwithstanding these limitations, the approach outlined in §III provides evidence that existing RE techniques *are* sufficient in the elicitation of explainability requirements (**RQ1**).

XI. Conclusion

In this work, we present an approach to eliciting explainability requirements for safety-critical systems. We identify a Use Case within the nuclear industry, and identify the relevant stakeholders by applying an agent ecosystem model [50]. We formulate scenarios using Scenario-Based Design methods [45], [11], then design and conduct semi-structured interviews with 16 stakeholders. We provide an example of how we extract requirements from the data and formalise them in FRET. Our approach shows that combining existing requirements elicitation techniques are sufficient for extracting explainability requirements, which is our answer to **RQ1**. To the best of our knowledge, we also provide the first set of explainability requirements for nuclear systems, which addresses **RQ2**. We highlight our lessons learned for the community, and show that combining multiple RE techniques in tandem provides both broad high-level requirements for an explainable system, and more detailed requirements that are scenario specific, resulting in a comprehensive understanding of what stakeholders require. We hope that out work can have a positive impact on regulating autonomy as it is steadily introduced into the nuclear industry. Our work provides a template for explainability requirements before these systems are built and integrated into the workplace, and an approach for eliciting explainability requirements for specific Use Cases, stakeholders, and contexts.

References

[1] IEEE Guide for Developing System Requirements Specifications. Standard, Institute of Electrical and Electronics Engineers (IEEE), 1998.

[2] C. Anderson. Principles and safety cases for the use of autonomous systems in nuclear environments. https://autonomy-and-verification.github.io/events/principles-safety-cases-workshop.

[3] S. Anjomshoae, A. Najjar, D. Calvaresi, and K. Främling. Explainable agents and robots: Results from a systematic literature review. In *18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019*, pages 1078–1088. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

[4] C. Atzmüller and P. M. Steiner. Experimental Vignette Studies in Survey Research. *Methodology*, 6(3):128–138, Jan. 2010.

[5] K. Auspurg and T. Hinz. *Factorial Survey Experiments*, volume 175. Sage Publications, 2014.

[6] H. Bourbouh, P.-L. Garoche, T. Loquen, É. Noulard, and C. Pagetti. CoCoSim, a code generation framework for control/command applications An overview of CoCoSim for multi-periodic discrete Simulink models. In *European Congress on Embedded Real Time Software and Systems*, 2020.

[7] M. Brandão, G. Canal, S. Krivić, P. Luff, and A. Coles. How experts explain motion planner output: a preliminary user-study to inform the design of explainable planners. In *2021 30th IEEE International Conference on Robot Human Interactive Communication (RO-MAN)*, pages 299–306, 2021.

[8] V. Braun and V. Clarke. Using thematic analysis in psychology. *Qualitative research in psychology*, 3(2):77–101, 2006.

[9] J. Broekens, M. Harbers, K. Hindriks, K. v. d. Bosch, C. Jonker, and J.-J. Meyer. Do you get it? User-evaluated explainable BDI agents. In *German Conference on Multiagent System Technologies*, pages 28–39. Springer, 2010.

[10] W. Brunotte, L. Chazette, V. Klös, and T. Speith. Quo vadis, explainability? – a research roadmap for explainability engineering. In V. Gervasi and A. Vogelsang, editors, *Requirements Engineering: Foundation for Software Quality*, pages 26–32, Cham, 2022. Springer International Publishing.

[11] J. M. Carroll. *Making use: scenario-based design of human-computer interactions*. MIT press, 2003.

[12] L. Chazette, W. Brunotte, and T. Speith. Exploring explainability: a definition, a model, and a knowledge catalogue. In *2021 IEEE 29th international requirements engineering conference (RE)*, pages 197–208. IEEE, 2021.

[13] L. Chazette, V. Klös, F. Herzog, and K. Schneider. Requirements on explanations: a quality framework for explainability. In *2022 IEEE 30th International Requirements Engineering Conference (RE)*, pages 140–152. IEEE, 2022.

[14] D. Cirqueira, D. Nedbal, M. Helfert, and M. Bezbradica. Scenario-based requirements elicitation for user-centric explainable ai. In A. Holzinger, P. Kieseberg, A. M. Tjoa, and E. Weippl, editors, *Machine Learning and Knowledge Extraction*, pages 321–341, Cham, 2020. Springer International Publishing.

[15] B. Donati, A. Ferrari, P. Spoletini, and S. Gnesi. Common mistakes of student analysts in requirements elicitation interviews. In *Requirements Engineering: Foundation for Software Quality: 23rd International Working Conference, REFSQ 2017, Essen, Germany, February 27–March 2, 2017, Proceedings 23*, pages 148–164. Springer, 2017.

[16] N. Du, J. Haspiel, Q. Zhang, D. Tilbury, A. K. Pradhan, X. J. Yang, and L. P. Robert Jr. Look who's talking now: Implications of av's explanations on driver's trust, av preference, anxiety and mental workload. *Transportation research part C: emerging technologies*, 104:428–442, 2019.

[17] A. Dutle, C. Muñoz, E. Conrad, A. Goodloe, I. Perez, S. Balachandran, D. Giannakopoulou, A. Mavridou, T. Pressburger, et al. From Requirements to Autonomous Flight: An Overview of the Monitoring ICAROUS Project. In *Workshop on Formal Methods for Autonomous Systems*, pages 23–30. EPTCS, 2020.

[18] M. Farrell, M. Luckcuck, L. Pullum, M. Fisher, A. Hessami, D. Gal, Z. Murahwi, and K. Wallace. Evolution of the IEEE P7009 standard: Towards fail-safe design of autonomous systems. In *2021 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 401–406, 2021.

[19] M. Farrell, M. Luckcuck, O. Sheridan, and R. Monahan. Fretting about requirements: formalised requirements for an aircraft engine controller. In *Requirements Engineering: Foundation for Software Quality: 28th International Working Conference, REFSQ 2022, Birmingham, UK, March 21–24, 2022, Proceedings*, pages 96–111. Springer, 2022.

[20] M. Fisher, R. C. Cardoso, E. C. Collins, C. Dadswell, L. A. Dennis, C. Dixon, M. Farrell, A. Ferrando, X. Huang, M. Jump, et al. An overview of verification and validation challenges for inspection robots. *Robotics*, 10(2):67, 2021.

[21] M. Fisher, V. Mascardi, K. Y. Rozier, B.-H. Schlingloff, M. Winikoff, and N. Yorke-Smith. Towards a framework for certification of reliable autonomous systems. *Autonomous Agents and Multi-Agent Systems*, 35(1):8, Apr. 2021.

[22] D. Giannakopoulou, A. Mavridou, J. Rhein, T. Pressburger, J. Schumann, and N. Shi. Formal Requirements Elicitation with FRET. Pisa, Mar. 2020. NTRS Author Affiliations: NASA Ames Research Center, Stinger Ghaffarian Technologies Inc. (SGT Inc.), Technische Univ. NTRS Report/Patent Number: ARC-E-DAA-TN77785 NTRS Document ID: 20200001989 NTRS Research Center: Ames Research Center (ARC).

[23] M. Glinz. Rethinking the notion of non-functional requirements. In *Proc. Third World Congress for Software Quality*, volume 2, pages 55–64, 2005.

[24] M. Hall, D. Harborne, R. Tomsett, V. Galetic, S. Quintana-Amate, A. Nottle, and A. Preece. A systematic method to understand requirements for explainable ai (xai) systems. In Proceedings of the IJCAI Workshop on eXplainable Artificial Intelligence (XAI 2019), Macau, China (Vol. 11).

[25] N. Hawes, C. Burbridge, F. Jovan, L. Kunze, B. Lacerda, L. Mudrova, J. Young, J. Wyatt, D. Hebesberger, T. Kortner, et al. The strands project: Long-term autonomy in everyday environments. *IEEE Robotics & Automation Magazine*, pages 146–156, 2017.

[26] M. Hertzum. Making use of scenarios: A field study of conceptual design. *International Journal of Human-Computer Studies*, 58:215–239, 02 2003.

[27] High-Level Expert Group on AI (AI HLEG). Ethics guidelines for trustworthy AI . Technical report, European Commission, 2019.

[28] High-Level Expert Group on AI (AI HLEG). Policy and investment recommendations for trustworthy Artificial Intelligence. Technical report, European Commission, 2019.

[29] M. Hosseini, A. Shahri, K. Phalp, and R. Ali. Foundations for transparency requirements engineering. In *Requirements Engineering: Foundation for Software Quality: 22nd International Working Conference, REFSQ 2016, Gothenburg, Sweden, March 14-17, 2016, Proceedings 22*, pages 225–231. Springer, 2016.

[30] E. Hull, K. Jackson, J. Dick, E. Hull, K. Jackson, and J. Dick. Doors: a tool to manage requirements. *Requirements engineering*, pages 187–204, 2002.

[31] Y. Kashima, A. McKintyre, and P. Clifford. The category of the mind: Folk psychology of belief, desire, and intention. *Asian Journal of Social Psychology*, 1:289–313, 1998.

[32] G. Kotonya and I. Sommerville. *Requirements Engineering: Processes and Techniques*. John Wiley & Sons, Inc., 1998.

[33] B. Lacerda, D. Parker, and N. Hawes. Optimal and dynamic planning for markov decision processes with co-safe ltl specifications. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1511–1516, 2014.

[34] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis. Explainable ai: a review of machine learning interpretability methods. *Entropy*, 23(1):18, 2020.

[35] M. Luckcuck. Using Formal Methods for Autonomous Systems: Five Recipes for Formal Cerification. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, page 1748006X2110349, July 2021.

[36] M. Luckcuck, H. M. Taylor, and M. Farrell. An abstract architecture for explainable autonomy in hazardous environments. In *2022 IEEE 30th International Requirements Engineering Conference Workshops (REW)*. IEEE, Aug. 2022.

[37] Luckcuck, Matt, M. Fisher, L. Dennis, S. Frost, A. White, and D. Styles. Principles for the Development and Assurance of Autonomous Systems for Safe Use in Hazardous Environments. Technical report, Zenodo, June 2021.

[38] B. F. Malle. How people explain behavior: A new theoretical framework. *Personality and Social Psychology Review*, 3(1):23–48, 1999.

[39] B. F. Malle, J. Knobe, M. J. O'Laughlin, G. E. Pearce, and S. E. Nelson. Conceptual structure and social functions of behavior explanations: Beyond person–situation attributions. *Journal of Personality and Social Psychology*, 79(3):309–326, 2000.

[40] A. Mavin, P. Wilkinson, A. Harwood, and M. Novak. Easy approach to requirements syntax (ears). In *2009 17th IEEE International Requirements Engineering Conference*, pages 317–322, 2009.

[41] P. Mihas. Qualitative research methods: approaches to qualitative data analysis. In R. J. Tierney, F. Rizvi, and K. Ercikan, editors, *International Encyclopedia of Education (Fourth Edition)*, pages 302–313. Elsevier, Oxford, fourth edition edition, 2023.

[42] T. Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

[43] J. Mylopoulos, L. Chung, and B. Nixon. Representing and using nonfunctional requirements: A process-oriented approach. *IEEE Transactions on software engineering*, 18(6):483–497, 1992.

[44] C. Negri-Ribalta. A method to deal with social bias and desirability in ethical requirements. Joint Proceedings of REFSQ-2022 Workshops, Doctoral Symposium, and Poster & Tools Track, 2022.

[45] M. B. Rosson and J. M. Carroll. *Usability engineering: scenario-based development of human-computer interaction*. Morgan Kaufmann, 2002.

[46] M. B. Rosson and J. M. Carroll. Human-computer interaction: Scenario-based design, 2009.

[47] P. Spoletini, A. Ferrari, M. Bano, D. Zowghi, and S. Gnesi. Interview review: An empirical study on detecting ambiguities in requirements elicitation interviews. In *Requirements Engineering: Foundation for Software Quality: 24th International Working Conference, REFSQ 2018, Utrecht, The Netherlands, March 19-22, 2018, Proceedings 24*, pages 101–118. Springer, 2018.

[48] S. Stange and S. Kopp. *Effects of a Social Robot's Self-Explanations on How Humans Understand and Evaluate Its Behavior*, page 619–627. Association for Computing Machinery, New York, NY, USA, 2020.

[49] H. M. Taylor, C. Jay, B. Lennox, A. Cangelosi, and L. Dennis. Should ai systems in nuclear facilities explain decisions the way humans do? an interview study. In *2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 956–962, 2022.

[50] R. Tomsett, D. Braines, D. Harborne, A. Preece, and S. Chakraborty. Interpretable to whom? a role-based model for analyzing interpretable machine learning systems. In *Proceedings of the ICML Workshop on Human Interpretability in Machine Learning*, 2018.

[51] G. Vilone and L. Longo. Notions of explainability and evaluation approaches for explainable artificial intelligence. *Information Fusion*, 76:89–106, 2021.

[52] A. F. T. Winfield, S. Booth, L. A. Dennis, T. Egawa, H. Hastie, N. Jacobs, R. I. Muttram, J. I. Olszewska, F. Rajabiyazdi, A. Theodorou, M. A. Underwood, R. H. Wortham, and E. Watson. IEEE P7001: A Proposed Standard on Transparency. *Frontiers in Robotics and AI*, 8:665729, July 2021.

[53] C. T. Wolf. Explainability scenarios: Towards scenario-based xai design. IUI '19, New York, NY, USA, 2019. Association for Computing Machinery.

[54] D. Zowghi and C. Coulin. Requirements elicitation: A survey of techniques, approaches, and tools. *Engineering and managing software requirements*, 2005.