

# **Safety Assurance for Patient-Facing Clinical Conversational Agents (CAs)**

Haris Aftab

Doctor of Philosophy

University of York  
Computer Science

October 2025

# Abstract

Conversational Agents (CAs) are increasingly being utilised in healthcare to alleviate the burden on clinical resources. However, the integration of machine learning (ML)-based CAs into clinical settings introduces significant safety risks, as the non-deterministic behaviour of ML algorithms can create unpredictable failure modes that may compromise patient safety. Despite emerging literature on ML in healthcare, systematic safety assurance approaches specifically tailored for CAs remain underdeveloped.

This thesis presents a safety assurance methodology for clinical CAs. First, a structured failure mode taxonomy for CAs is developed, identifying both technical and socio-technical causes. This taxonomy is then used to support a structured safety assurance framework. This framework demonstrates how to systematically conduct hazard analysis, derive a multi-layered set of safety requirements (for the system, the ML model, and its data), and construct a safety case using Goal Structuring Notation (GSN).

The methodology is applied and evaluated through a series of progressively mature use cases, leading to a detailed case study of a real-world CA deployed in NHS hospitals. The research combines quantitative analysis of ML model performance with qualitative evaluation of the overall framework through stakeholder interviews. The findings suggest that the methodology helps identify context-dependent failures and improves traceability. Overall, the thesis contributes a structured methodology, evaluated through empirical case studies, for identifying hazards, analysing and managing risks, and implementing mitigations, thereby helping to bridge the gap between safety engineering and the practical deployment of ML-based clinical CAs.

*For my mother, mother, mother, and father . . .*

# Table of Contents

<b>List of Tables</b>	<b>11</b>
<b>List of Figures</b>	<b>13</b>
<b>Acknowledgements</b>	<b>15</b>
<b>Author’s Declaration</b>	<b>16</b>
<b>1 Introduction</b>	<b>17</b>
1.1 Research Questions and Thesis Contributions . . . . .	19
1.2 Thesis Overview . . . . .	20
<b>2 Literature Review</b>	<b>22</b>
2.1 Safety Assurance and Regulation in Healthcare . . . . .	23
2.1.1 The unique landscape of Healthcare Safety . . . . .	23
2.1.2 Patient Safety and Degrees of Harm . . . . .	25
2.1.3 Arguing Safety: The Safety Case and GSN . . . . .	27
2.1.4 Safety Analysis Methodologies . . . . .	30
2.1.4.1 Checklists . . . . .	30
2.1.4.2 Failure Mode and Effects Analysis (FMEA) . . . . .	31
2.1.4.3 Hierarchical Task Analysis (HTA) . . . . .	32
2.1.4.4 SHARD . . . . .	32
2.1.4.5 Hazard and Operability Study (HAZOP) . . . . .	33
2.1.4.6 HAZOP-UML . . . . .	34
2.1.5 The Regulatory Landscape for Health IT and Medical Devices . .	34
2.1.5.1 Medical Devices . . . . .	35
2.1.5.2 Health IT Systems . . . . .	39
2.2 Conversational Agents (CAs) . . . . .	40
2.2.1 Chatbots . . . . .	40

2.2.1.1	Rule-based Chatbot . . . . .	41
2.2.1.2	Corpus-based Chatbots . . . . .	41
2.2.1.3	Information Retrieval IR-based Chatbots . . . . .	42
2.2.2	Task-Oriented Dialogue Agents . . . . .	43
2.2.2.1	GUS - A Frame-Based Architecture . . . . .	43
2.2.2.2	The Dialogue State Architecture . . . . .	45
2.2.2.3	CA Evaluation Techniques and Domain-Specific Implications . . . . .	49
2.2.2.4	The Confusion Matrix . . . . .	49
2.2.2.5	Core Evaluation Metrics . . . . .	50
2.2.2.6	Performance Metrics . . . . .	50
2.2.2.7	Sensitivity and Impact in Healthcare Applications . . . . .	51
2.2.3	Conversational Agents (CAs) in Healthcare . . . . .	52
2.2.3.1	Safety Implications of Clinical CAs . . . . .	55
2.3	Machine Learning . . . . .	56
2.3.1	Machine Learning in CAs . . . . .	56
2.3.2	Machine Learning Challenges in CAs . . . . .	58
2.3.2.1	Data . . . . .	58
2.3.2.2	Uncertainty . . . . .	59
2.3.2.3	ML Robustness Techniques . . . . .	59
2.3.3	Frameworks for CA Development . . . . .	60
2.4	Methodological Rationale: The Case Study Approach . . . . .	64
2.4.1	The Suitability of Case Study Research for Socio-Technical Systems . . . . .	64
2.4.2	Case Study Designs Relevant to this Thesis . . . . .	64
2.4.3	Selection of Yin's Framework for Rigorous Inquiry . . . . .	65
2.5	Summary . . . . .	65
<b>3</b>	<b>Safety Assurance Methodology</b>	<b>67</b>
3.1	Introduction . . . . .	67
3.2	Overview of the Methodological Framework . . . . .	67
3.3	Phase 1: Foundational Taxonomy Development . . . . .	69
3.4	Phase 2: Systematic Hazard and Risk Analysis . . . . .	69
3.5	Phase 3: Deriving Safety Requirements . . . . .	70
3.6	Phase 4: Constructing the Safety Case with GSN . . . . .	70
3.7	Phase 5: Technical Validation and Stakeholder Evaluation . . . . .	71
3.8	Connecting the Thesis: From Theory to Practice . . . . .	71

---

3.9 Summary . . . . .	72
<b>4 A Failure Mode Taxonomy for Clinical Conversational Agents (CAs)</b>	<b>73</b>
4.1 Introduction . . . . .	73
4.2 Theoretical Foundations of Taxonomies in System Analysis . . . . .	75
4.3 A Safety Engineering Perspective on System Failure . . . . .	76
4.3.1 Defining Fault, Failure, and Failure Mode . . . . .	76
4.3.2 The Causal Chain from Fault to Harm . . . . .	77
4.4 A System-Theoretic Approach to Taxonomy Design . . . . .	79
4.4.1 Methodological Inspiration: The SHELL Model . . . . .	79
4.4.2 A Hierarchical Design for a Failure Mode Taxonomy . . . . .	81
4.5 A Failure Mode Taxonomy for Clinical CAs . . . . .	82
4.5.1 Software Failure Modes and Causal Factors . . . . .	82
4.5.1.1 Recognition (ASR) . . . . .	87
4.5.1.2 Understanding (NLU) . . . . .	87
4.5.1.3 Dialogue Manager (DM) & Dialogue Policy (DP) . . . . .	88
4.5.1.4 Response Generation (NLG) . . . . .	89
4.5.1.5 User Experience (UX) and Interface (UI) . . . . .	89
4.5.2 Machine Learning (ML) as a Root Cause of Software Failures . . . . .	90
4.5.2.1 Data-Induced Faults . . . . .	91
4.5.2.2 Model-Induced Faults . . . . .	92
4.5.3 Hardware Failure Modes and Causal Factors . . . . .	93
4.5.4 The Role of User and Environment as Causal Contributors . . . . .	94
4.5.4.1 User . . . . .	94
4.5.4.2 Environment . . . . .	95
4.6 Taxonomy Evolution and Validation Through Use Cases . . . . .	96
4.7 Analysis of Patient Safety Implications . . . . .	97
4.7.1 Mapping Failure Modes to Clinical Hazards . . . . .	97
4.7.1.1 Recognition and Understanding Failures . . . . .	97
4.7.1.2 Decision-Making and Design Failures . . . . .	98
4.7.1.3 Automation Bias and Complacency . . . . .	98
4.7.1.4 Clinical Deskillling . . . . .	98
4.7.1.5 Exacerbation of Health Disparities . . . . .	99
4.7.2 Table of Illustrative Scenarios . . . . .	99
4.8 Taxonomy Scope, Application, and Validity . . . . .	99

4.9	Summary . . . . .	102
<b>5</b>	<b>System Safety Analyses for CAs</b>	<b>103</b>
5.1	Use Case 1: Symptom Checking CA . . . . .	104
5.1.1	System Design and Implementation . . . . .	105
5.1.2	Research Methodology: A Hybrid Approach . . . . .	106
5.1.2.1	Phase 1: Technical Failure Analysis Methodology . . . . .	106
5.1.2.2	Phase 2: Exploratory Case Study Evaluation Methodology . . . . .	106
5.1.3	Findings Part 1: Technical Analysis of System Failures . . . . .	107
5.1.3.1	Quantitative Performance Analysis . . . . .	111
5.1.4	Findings Part 2: Thematic Analysis of Stakeholder Evaluation . . . . .	115
5.1.5	Discussion and Lessons Learned . . . . .	118
5.1.5.1	Answering RQ2a: Demonstrating the Analysis Methodology . . . . .	118
5.1.5.2	Deriving Mitigation Strategies from the Analysis . . . . .	119
5.2	Use Case 2: Uncertainty-Aware NLU . . . . .	120
5.2.1	Overview . . . . .	120
5.2.2	Clinical Relevance and Safety Concerns . . . . .	122
5.2.3	Model Design and Technical Approach . . . . .	124
5.2.3.1	MC Dropout as a Bayesian Approximation . . . . .	124
5.2.3.2	Quantifying Uncertainty with Shannon Entropy . . . . .	125
5.2.3.3	Model Architecture and Hyperparameter Justification . . . . .	125
5.2.4	Dataset and Experimental Setup . . . . .	127
5.2.4.1	In-Distribution (ID) Dataset . . . . .	127
5.2.4.2	Data Preprocessing and Partitioning . . . . .	128
5.2.4.3	Out-of-Distribution (OOD) Evaluation Dataset . . . . .	129
5.2.4.4	Experimental Protocol for Uncertainty Estimation . . . . .	129
5.2.5	Evaluation Results . . . . .	130
5.2.5.1	Core Predictive Performance on ID Data . . . . .	130
5.2.5.2	Uncertainty Estimation on ID vs. OOD Data . . . . .	130
5.2.6	Comparative Analysis of Uncertainty Distributions . . . . .	132
5.2.7	Hazard and Risk Analysis . . . . .	134
5.2.8	Safety Requirements and Monitor Design . . . . .	137
5.2.9	Summary . . . . .	137
5.3	Use Case 3: Gestational Diabetes Mellitus CA . . . . .	138
5.3.1	Overview and Architecture . . . . .	138

---

5.3.2	Background and Clinical Context . . . . .	140
5.3.3	Hazard Analysis and Risk Assessment . . . . .	141
5.3.4	Risk Assessment Prioritisation . . . . .	144
5.3.5	Linking HAZOP-UML Deviations to the Failure Mode Taxonomy . . . . .	146
5.3.6	Linking Analysis to Clinical Hazards . . . . .	148
5.3.7	Deriving Safety Requirements from the Analysis . . . . .	149
5.3.8	A Structured Safety Case for the GDM CA . . . . .	151
5.3.8.1	Top-Level Safety Argument . . . . .	152
5.3.8.2	Argumentation over Hazard H01 (Incorrect Dose) . . . . .	153
5.3.8.3	Argumentation over Hazard H02 (Failure to Provide a Dose) . . . . .	155
5.4	Summary . . . . .	156
<b>6</b>	<b>Case Study: Safety Analysis of a Real-World Clinically Deployed CA</b>	<b>158</b>
6.1	Background and Deployment Context . . . . .	158
6.2	System Architecture and Functional Overview . . . . .	159
6.3	Case Study Design and Methodology . . . . .	160
6.3.1	Research Approach: An Evaluative Case Study Framework . . . . .	160
6.3.2	Designing the Case Study: The Five Components of Research Design . . . . .	161
6.3.2.1	The Case Study's Questions . . . . .	161
6.3.2.2	The Case's Propositions . . . . .	161
6.3.2.3	The Case's Unit of Analysis . . . . .	163
6.3.2.4	Linking Data and Interpreting Findings . . . . .	163
6.3.3	Data Collection and the Role of the Protocol . . . . .	163
6.3.4	Data Analysis Strategy . . . . .	164
6.4	System Safety Analysis . . . . .	164
6.4.1	Hazard Analysis . . . . .	165
6.4.1.1	The SHARD Method: Guidewords and Process . . . . .	165
6.4.1.2	Data Flow Diagrams . . . . .	166
6.4.1.3	SHARD Analysis Results . . . . .	168
6.4.2	Identification of Failure Modes . . . . .	170
6.4.3	Hazards and Risk Assessment . . . . .	171
6.4.3.1	Assessing Severity: The Worst Plausible Outcome . . . . .	171
6.4.3.2	Assessing Likelihood: Considering System Controls and Behaviour . . . . .	172

6.4.3.3	Calculating the Final Risk Score . . . . .	173
6.4.4	Derivation of a Multi-Layered Safety Requirements Framework . .	173
6.4.4.1	Safety/Design Recommendations . . . . .	174
6.4.4.2	Hinata Safety Requirements . . . . .	175
6.4.4.3	NLU Model Requirements . . . . .	177
6.4.4.4	NLU Data Requirements . . . . .	178
6.4.5	Structure of the Safety Case . . . . .	179
6.4.6	Hinata Safety Case (GSN) . . . . .	180
6.4.6.1	Argumentation over Top Level Goal . . . . .	181
6.4.6.2	Argumentation over Hazard H-01 . . . . .	182
6.4.6.3	Argumentation over the NLU Component . . . . .	184
6.4.7	The Critical Role of Confidence Arguments (ACPs) . . . . .	191
6.4.7.1	ACP.H1.S1: Confidence in Hazard Mitigation through System Components . . . . .	191
6.4.7.2	ACP.H1.C1: Confidence in Eye Cataract Symptoms Data . . .	192
6.5	Evaluation of the Case Study Methodology . . . . .	192
6.5.1	Evaluation Methodology Recap . . . . .	192
6.5.2	Thematic Analysis of Stakeholder Feedback . . . . .	192
6.5.2.1	Theme 1: Convergence on the Realism of Hazard Identification	193
6.5.2.2	Theme 2: ML as Source of Modern Safety Concerns . . . . .	194
6.5.2.3	Theme 3: Safety Case as a Tool for Governance, Trust, and Communication . . . . .	195
6.5.2.4	Theme 4: The Foundational Importance of Data Quality in ML Safety . . . . .	196
6.5.2.5	Theme 5: Human Review as a Safety Control . . . . .	197
6.5.2.6	Theme 6: The Practical Value and Potential Generalisability of the Safety Case Framework . . . . .	198
6.5.3	Summary . . . . .	198
<b>7</b>	<b>Conclusion and Future Work</b>	<b>200</b>
7.1	Summary of Thesis Contributions . . . . .	200
7.1.1	Contribution 1: A Failure Mode Taxonomy for clinical CAs . . . . .	200
7.1.2	Contribution 2: Uncertainty-Aware NLU for Safe Decision Making	201
7.1.3	Contribution 3: A Methodology for Constructing and Evaluating a Safety Argument for an ML-based CA . . . . .	202

---

7.2 Research Limitations . . . . .	203
7.3 Future Work . . . . .	204
<b>Appendix A Failure Mode Taxonomy for CAs</b>	<b>206</b>
<b>Appendix B Exploratory Interview Questions - Use Case 1</b>	<b>210</b>
<b>Appendix C Evaluative Interview Questions - Case Study</b>	<b>211</b>
<b>Appendix D SHARD Analysis Tables - Case Study</b>	<b>213</b>
<b>Abbreviations</b>	<b>231</b>
<b>References</b>	<b>234</b>

# List of Tables

Table 1	Example DA for a CA providing care facilities . . . . .	46
Table 2	Machine learning confusion matrix . . . . .	49
Table 3	Overview of clinical CAs . . . . .	55
Table 4	Summary of safety issues in clinical CAs . . . . .	57
Table 5	Failure mode taxonomy for CAs . . . . .	83
Table 6	ML model and data faults contributing to CA software failures . . . . .	90
Table 7	Illustrative CA failures and impact on patient safety . . . . .	100
Table 8	Summary of themes from exploratory stakeholder interview . . . . .	116
Table 9	Derived mitigation strategies for identified failure modes . . . . .	120
Table 10	Use case 2 RNN architecture . . . . .	126
Table 11	Average evaluation metrics for medical symptoms . . . . .	133
Table 12	Sample predictions with entropy scores for test utterances . . . . .	133
Table 13	Uncertainty estimation for OOD utterances . . . . .	134
Table 14	Calculation of insulin dosing . . . . .	141
Table 15	GDM CA: HAZOP deviations . . . . .	145
Table 16	GDM CA: HAZOP causes and recommendations . . . . .	146
Table 17	GDM CA: clinical hazard log . . . . .	149
Table 18	GDM CA: derived safety requirements . . . . .	150
Table 19	Hinata: overview of the case study research design . . . . .	162
Table 20	Hinata: selected SHARD analyses . . . . .	169
Table 21	Hinata: identified causal failure modes . . . . .	170
Table 22	Hinata: clinical hazard log . . . . .	171
Table 23	Hinata: safety and design recommendations . . . . .	175
Table 24	Hinata: safety requirements with hazard traceability . . . . .	176
Table 25	Hinata: NLU model safety requirements . . . . .	178
Table 26	Hinata: NLU data safety requirements . . . . .	180
Table 27	Principal themes from stakeholder interviews . . . . .	193
Table 28	Consolidated failure mode taxonomy for CAs . . . . .	206

---

Table 29	SHARD flow: 1.1 response generation . . . . .	213
Table 30	SHARD flow: 1.1.1 NLG data . . . . .	215
Table 31	SHARD flow: 1.2 next action . . . . .	216
Table 32	SHARD flow: QC.1.2 next action . . . . .	218
Table 33	SHARD flow: 1.3 intent classification . . . . .	219
Table 34	SHARD flow: QC.1.3 intent classification . . . . .	220
Table 35	SHARD flow: 1.3.1 NLU data . . . . .	221
Table 36	SHARD flow: QC.1.3.1 NLU data . . . . .	222
Table 37	SHARD flow: 3.0 text-to-speech . . . . .	223
Table 38	SHARD flow: 5.0 speech-to-text . . . . .	224
Table 39	SHARD flow: QC.7.0 patient's call . . . . .	225
Table 40	SHARD flow: QC.8.0 supervisor input . . . . .	226
Table 41	SHARD flow: QC.9.0 store supervisor inputs . . . . .	228
Table 42	SHARD flow: QC.12.0 calculate safety check decision . . . . .	229

# List of Figures

Figure 1	Graphical notation of core GSN constructs and relationships . . . .	28
Figure 2	An example safety case constructed using GSN . . . . .	29
Figure 3	MHRA : device determination flow chart . . . . .	36
Figure 4	MHRA : medical device determination flow chart . . . . .	37
Figure 5	The GIP overall top level safety case argument . . . . .	38
Figure 6	The GIP medical device safety case sub-argument . . . . .	39
Figure 7	ELIZA chatbot . . . . .	41
Figure 8	Date type in GUS architecture . . . . .	44
Figure 9	Sample intent generation from the user's utterance . . . . .	44
Figure 10	CA pipeline architecture . . . . .	45
Figure 11	DST example for a restaurant recommendation system . . . . .	47
Figure 12	NLG template from a restaurant recommendation system . . . . .	48
Figure 13	ML in CA architecture . . . . .	58
Figure 14	Causal chain from fault to harm . . . . .	78
Figure 15	The SHELL Model . . . . .	80
Figure 16	NLU intent confusion matrix . . . . .	112
Figure 17	Dialogue model action confusion matrix . . . . .	114
Figure 18	NLU confidence histogram for symptom checking CA . . . . .	115
Figure 19	Use case 2 CA architecture . . . . .	121
Figure 20	Use case 2 CA safety architecture . . . . .	127
Figure 21	The distribution of medical symptoms in the dataset . . . . .	128
Figure 22	Confusion matrix of symptoms classification . . . . .	131
Figure 23	Entropy calculations for ID and OOD data . . . . .	132
Figure 24	3D view of confidence, uncertainty, and data type . . . . .	135
Figure 25	T-SNE visualisation of word embeddings by intent . . . . .	136
Figure 26	GDM CA architecture . . . . .	139
Figure 27	GDM CA detailed architecture . . . . .	140
Figure 28	GDM CA sequence diagram . . . . .	143

---

Figure 29	NHS NPSA 5x5 Risk classification matrix . . . . .	147
Figure 30	GDM CA safety case top level argument . . . . .	152
Figure 31	GDM CA safety case hazard 1 mitigation argument . . . . .	154
Figure 32	GDM CA safety case hazard 2 mitigation argument . . . . .	155
Figure 33	Hinata abstract architecture . . . . .	160
Figure 34	Hinata DFD context diagram . . . . .	167
Figure 35	Hinata DFD RASA conversation flow . . . . .	167
Figure 36	Hinata safety case top level argument . . . . .	181
Figure 37	Hinata safety case argument for H-01 mitigation . . . . .	183
Figure 38	Hinata safety case NLU model argument for H-01 mitigation . . . . .	187
Figure 39	Hinata safety case NLU data argument for H-01 mitigation . . . . .	188
Figure 40	Hinata safety case confidence arguments for H-01 mitigation . . . . .	191

# Acknowledgements

First and foremost, I extend my sincerest thanks to my supervisor, Professor Ibrahim Habli. His consistent guidance, unparalleled support, and unwavering motivation have been central to this research. His mentorship, particularly during challenging personal times, was invaluable and extended far beyond the academic realm.

This research was made possible by generous funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 812.788 (MSCA-ETN SAS). Project website: <http://etn-sas.eu/>.

I am also grateful to my co-supervisors, Professor Jérémie Guiochet (LAAS-CNRS) and Eoin O'Carroll (Portable Medical Technologies), for their instrumental expertise and support during my secondments. I am thankful to Dr. Jo Iacovides for her valuable feedback throughout this work. I would also like to thank Dr. Nick de Pennington, Mr. James Godwin, Dr. Aisling Higham, and Dr. Ernest Lim for their support on main case study of this research. My heartfelt thanks also go to my friends for their motivation and personal support: Mr. Osama Zia, Mr. Abdullah Khan Durani, Dr. Zaid Tahir, Dr. Enes Algul, and Dr. Yasir Saleem.

On a personal note, I am eternally indebted to my family. To my late mother, whose memory I hold dear, and to my father, for their unwavering support throughout every circumstance of my life. I profoundly wish my mother could have been here to see this achievement. I would also like to acknowledge the unparalleled support from my elder brother, Awais, during very tough personal and family circumstances. To my wife, Fatima, and my daughter, Hannah, I am immeasurably grateful for your emotional support, patience, and understanding.

I acknowledge that I received assistance from Grammarly, ChatGPT, and Gemini to proofread this thesis in line with the Policy on Transparency in Authorship in PGR Programmes.

# Author's Declaration

*I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for a degree or other qualification at this University or elsewhere. All sources are acknowledged as references.*

Some material presented in this thesis has been published in *peer-reviewed conference proceedings*. The published articles are:

- H. Aftab, S. H. H. Shah, and I. Habli, "Classification of failures in the perception of conversational agents (CAs) and their implications on patient safety," in *Public Health and Informatics*. IOS Press, 2021, pp. 659–663.
- H. Aftab, V. Gautam, R. Hawkins, R. Alexander, and I. Habli, "Robust intent classification using Bayesian LSTM for clinical conversational agents (CAs)," in *Proc. Int. Conf. Wireless Mobile Commun. Healthcare (MobiHealth)*. Cham, Switzerland: Springer, 2021, pp. 106–118.

Haris Aftab  
October 2025

# Chapter 1

## Introduction

There has been growing commercial and research interest in the use of automated dialogue systems in healthcare [1]. These systems, widely known as Conversational Agents (CAs), are designed to simulate a one-to-one conversation between a human and a computer for various health purposes [2]. Advances in Artificial Intelligence (AI) and Natural Language Processing (NLP) have transformed CAs, enabling more natural and intuitive interactions. As a result, CAs are beneficial for both healthcare providers and patients. For providers, they offer an alternative to already stretched healthcare resources by assisting with tasks like patient monitoring and follow-up calls. For patients, CAs are a cost-effective and accessible solution for services such as symptom checking, medication adherence, and managing chronic conditions [3, 4].

The effectiveness of modern CAs is largely due to advancements in Machine Learning (ML) techniques. To achieve natural conversation, CAs utilise multiple technologies, including automatic speech recognition (ASR), natural language understanding (NLU), dialogue management (DM), and natural language generation (NLG). These components are powered by a range of technologies, from rule-based systems to various forms of ML, some of which can function as black-box technologies that complicate system analysis and understanding.

Although CAs provide many benefits, their integration into a safety-critical domain like healthcare presents a number of serious challenges. Component-level failures may lead to an unexpected response, potentially critical in clinical contexts [5]. Safety challenges in CAs primarily arise from two sources: the nature of human-computer interaction and the complexity of the underlying technology [6]. The use of ML, for example, requires sufficient and accurate training data, which is often difficult to acquire in healthcare [7]. Even with correct data, ML models may not be optimally configured to make safe decisions, as they are not explicitly programmed. Whether

---

used in a supervised clinical setting or as a stand-alone application, a CA can introduce hazards that may harm the patient [8].

At present, the use of clinical CAs is typically confined to non-urgent or supervised contexts. As greater autonomy is explored in clinical settings, a robust safety methodology becomes paramount. The current landscape lacks established, domain-specific safety requirements and assurance methods for clinical CAs: while the literature reports failure patterns and safety risks in general-purpose assistants and healthcare chatbots, formal assurance perspectives remain scarce [5, 7, 8]. This gap reinforces the need to systematically identify, analyse, and mitigate risks grounded in realistic failure modes, uncertainty, and human-AI interaction before wider clinical deployment [6].

Large Language Models (LLMs) have fundamentally advanced conversational AI in healthcare by enabling more natural, context-aware, and patient-specific interactions, supporting tasks from triage to chronic disease management and patient education [9–11]. However, these models introduce new safety challenges, including hallucinations, inconsistent outputs, and vulnerability to adversarial prompts, which can result in the propagation of clinically unsafe or misleading information [11–13]. Despite their sophistication, LLMs remain black-box systems, often lacking transparency and interpretability, which complicates error detection and risk assessment in high-stakes clinical environments [13, 14].

Recent studies highlight that LLMs can generate plausible but incorrect or unsafe advice, and their performance is highly sensitive to input phrasing and data quality, underscoring the need for robust, domain-specific safety validation [11, 12, 15]. Consequently, established safety assurance methods such as failure-mode analysis, hazard/risk analysis, and structured assurance cases remain essential for clinical CAs, as they provide systematic frameworks to identify, analyse, and mitigate risks across the information flow (input, processing, output), regardless of underlying model complexity [11, 15, 13]. While this thesis focuses on ML-based pipeline CAs, some artefacts (taxonomy, hazard/risk analysis, and the GSN safety case) are defined at the information-flow level (input-processing-output) and may have partial relevance to LLM-based pipelines; their applicability is not established here.

This thesis focuses on patient-facing clinical CAs, meaning systems that interact directly with patients or members of the public in clinical or health-related contexts. This includes systems used as stand-alone applications as well as systems deployed within provider-managed or clinically supervised pathways. The central safety

concern is therefore the risk arising from direct CA-user interaction, including failures in recognising, understanding, or responding to patient input, and the potential for these failures to contribute to clinical hazards.

## Thesis Statement

The thesis addresses this gap by arguing that the safety of ML-based clinical CAs can be systematically assured through a holistic methodology. This thesis proposes that by establishing a structured failure mode taxonomy, and using it to drive a rigorous process of hazard analysis, requirements derivation, and formal safety case construction, the risks inherent in these complex systems can be effectively managed, enabling their safe and trustworthy deployment.

## 1.1 Research Questions and Thesis Contributions

To understand how clinical CAs fail and how these failures can be systematically analysed and mitigated, this thesis is structured around the following research questions:

**RQ1** What are the different failure modes in CAs that can lead to clinical hazards?

**RQ2** How do we systematically develop safety cases for clinical CAs?

**RQ2a** How do we analyse and mitigate potential hazards arising from failures within clinical CAs?

**RQ2b** How can we reduce the risks in clinical decision-making of CAs by robustly handling the inherent uncertainty in their understanding?

**RQ2c** How can we develop a safety argument for clinical CAs?

For the RQs presented earlier, the thesis contributes to the body of knowledge in the following specific ways:

1. **Failure-mode taxonomy.** It proposes a failure mode taxonomy for clinical CAs that systematically classifies potential failures and their causal factors across the entire socio-technical system (software, hardware, user, and environment). A key novelty is its detailed analysis of failures rooted in ML, distinguishing between faults induced by the ML model and its data. Grounded in literature and insights from applied use cases, the taxonomy directly addresses **RQ1**.

2. **Uncertainty-aware NLU.** It develops and evaluates an uncertainty-aware intent classification approach (Bayesian LSTM with Monte Carlo dropout), showing how uncertainty estimation and out-of-distribution (OOD) detection can trigger safe fallbacks (e.g. asking for clarification, politely declining as it is out of scope). The empirical results on in-distribution (ID) and OOD data provide evidence for safety controls referenced in Chapters 5 and 6, strengthening the mitigation of ML-related failure modes and supporting **RQ2b**.
3. **Safety-assurance methodology.** It introduces a systematic safety-assurance methodology for clinical CAs, showing how to progress from hazard identification and risk analysis to multi-layered safety requirements and a formal, evidence-based safety argument using Goal Structuring Notation (GSN). This methodological contribution details how to mitigate ML-specific risks within pipeline CAs and constitutes the core answer to **RQ2a**, and **RQ2c**.

## 1.2 Thesis Overview

This thesis is organised as follows:

- **Chapter 2** presents a detailed literature review covering system safety analysis, safety assurance practices in healthcare, machine learning challenges in conversational agents, and current CA deployments. It outlines key safety analysis techniques, regulatory frameworks, ML failure modes, and CA-specific evaluation methods.
- **Chapter 3** introduces the safety assurance methodology adopted in this research. It explains how various safety techniques are mapped to different CA components and how the methodology bridges theoretical safety frameworks with applied system design to answer **RQ2**.
- **Chapter 4** presents the first contribution of the thesis: a taxonomy of failure modes in clinical CAs that spans software architecture, hardware, user interaction, and environmental factors, and provides a structured response to **RQ1**. It also discusses safety implications to support hazard identification for later analyses.

- **Chapter 5** details a series of use cases, each investigating and illustrating different aspects of the safety assurance process. The first use case is an exploratory study of a symptom checker prototype, where a combination of technical failure analysis and a stakeholder interview were used to identify early user-facing risks (addresses **RQ2a** and demonstrates link to **RQ1**). The second use case focuses on ML robustness, evaluating uncertainty using Monte Carlo Dropout and identifying ML-specific hazards (addresses **RQ2b**). The third use case examines an end-to-end CA for gestational diabetes, applying HAZOP-UML to explore system-level hazards, risk ranking, and safety requirements (addresses **RQ2a** and **RQ2c**).
- **Chapter 6** presents the main case study: a detailed safety analysis of a real-world clinical CA deployed in the NHS. This chapter applies the full methodology, culminating in the development of a structured safety case, thereby answering **RQ2**.
- **Chapter 7** concludes the thesis by summarising the main contributions and outlining future research directions.
- **Appendix A** provides the failure mode taxonomy from Chapter 4 in a consolidated form for practical reference.
- **Appendix B** provides the exploratory interview questions for use case 1 (Chapter 5).
- **Appendix C** provides the evaluative interview questions for the case study (Chapter 6).
- **Appendix D** presents detailed SHARD analysis tables for the case study (Chapter 6).

# Chapter 2

## Literature Review

This chapter provides a broad overview of the multidisciplinary literature that forms the foundation for this thesis. It centres on safety engineering, healthcare regulation, and conversational AI technology. The objective is to establish the current state-of-the-art and, in doing so, identify the research gaps that this thesis aims to address.

The chapter begins by establishing the foundational principles of system safety engineering, defining core terminology such as hazards and risk, and reviewing established analysis techniques like Hazard and Operability Analysis (HAZOP), Failure Mode and Effects Analysis, and Software Hazard and Resolution in Design (SHARD). The review then contextualises these principles within the unique safety and regulatory landscape of healthcare, examining the role of the safety case, the Goal Structuring Notation (GSN), and key industry standards that govern medical devices and health IT systems. Subsequently, the chapter introduces the core technology of Conversational Agents (CAs), detailing their typical architecture, the Natural Language Processing (NLP) pipeline that enables them, and their growing applications across various clinical domains. Building on this foundation, the review synthesises existing research to identify the significant safety challenges that emerge at the intersection of CAs and healthcare, with a particular focus on the risks introduced by Machine Learning (ML) components.

By mapping the current state of knowledge across these interconnected fields, this chapter demonstrates the absence of a holistic, systematic safety assurance framework for ML-based clinical CAs. This identified gap provides the central justification for the contributions developed in the subsequent chapters of this thesis.

## **2.1 Safety Assurance and Regulation in Healthcare**

### **2.1.1 The unique landscape of Healthcare Safety**

The concept of safety-criticality in a system originates from two properties: mass and dread [16]. Events of mass casualty, in which a large number of people die, capture public attention in a manner that a larger cumulative number of deaths occurring over many small episodes does not. Healthcare risks that lead to injury or death often manifest as these smaller, distributed episodes and consequently do not draw the same level of attention. Another reason healthcare is treated differently is that many adverse events in other safety-critical industries are associated with feelings of dread, unwilling participation in risks, or a loss of control. In healthcare, most adverse events are indistinguishable from natural events. As Gaba et al. [17] mention, because every human is destined to die, there is a strong chance of this occurring within the vicinity of healthcare. As a result, deaths and disabilities are sometimes viewed as normal events, even when they are unexpected outcomes that would never be considered normal in industries like nuclear power or aviation.

Healthcare practices encompass a diverse set of activities and principles, ranging from highly hazardous surgery to relationship-centred primary care and the unpredictable, constantly changing demands of emergency medicine [18]. This diversity, which also includes hospital medicine, community care, and home care, makes healthcare a complex domain with few commonalities with other safety-critical industries, which tend to operate with more tightly standardised, protocol-driven workflows and a higher proportion of routine, well-specified tasks. To ensure reliability and safety in those domains, it is normal practice to minimise situations that fall outside validated procedures. The level of uncertainty in healthcare is quite large and depends on more than one factor. For example, a medicine's effect on a patient's condition is dependent on that patient's current health, the ability of their immune system to fight side effects, and the involvement of heredity and genetic effects. In other industries, system routines are typically monitored while the system is in operation, for instance, a running aeroplane or chemical plant. The work in healthcare, however, is hands-on and potentially prone to more errors [18]. People are in good health in other industries, while in healthcare they are sick, injured, or have disabilities, and are thus vulnerable to even very small errors in their care.

On an organisational level, there are also vast differences between healthcare and other industries. According to [19], most high-risk industries are well-structured with centralised control. Healthcare is quite decentralised in this regard; for example, the NHS in England. In other industries, there is a strong emphasis on standardising work and training processes [20]. In commercial aviation, pilots qualified on the same aircraft type within the same operator can often be paired across flights with limited handover, enabled by type ratings and operator standard operating procedures (SOPs); this interchangeability is constrained by currency and route/airport qualifications. In healthcare, a physician, even at the same level of qualification, cannot be substituted in this way without substantial handover and context, and attempting similar interchangeability among clinicians or nurses would likely increase error risk. The regulation in healthcare has not yet reached the level of aviation or other industries.

Humans have cognitive limits, and when these limits are exceeded, the probability of making mistakes increases. Consider, for example, fatigue, which is an important variable in human health but is a factor that has been recognised very slowly. In a study by Sexton et al. [21], only 30% of surgeons were willing to admit that their performance is worse without proper sleep. In contrast, aviation has governing policies for such issues: for example, pilots and controllers have mandatory retirement ages and time-on-duty limits that recognise the negative impact of fatigue, and they have recertification requirements [22]. Pilots must undergo a proficiency check every six months, while in the UK, doctors now undergo revalidation after five years [23].

Healthcare shares some surface traits with the nuclear industry as both are complex, tightly coupled, socio-technical systems but their uncertainty profiles differ. In contrast to nuclear power, healthcare has evolved organically over time. The nuclear industry, by comparison, is engineered, and each plant is designed and built to be part of an integrated system. Primary processes in healthcare are less well understood than the physical processes associated with nuclear power [24]. Therefore, healthcare has greater uncertainty and risk at the subsystem level (patient care) than nuclear power. Healthcare should not simply adopt solutions from other industries but should compare and contrast organisational attributes and take safety measures only where conditions are similar [25]. For example, from aviation, healthcare organisations might learn about crew resource management training in homogeneous teams. As pointed out by Macrae et al. [20], investigations

and monitoring in healthcare are not yet on a par with other high-risk industries. Healthcare should have an external investigation body comparable to, for example, the UK's Air Accidents Investigation Branch (AAIB), which investigates serious civil aircraft accidents.

While healthcare has already adopted many interventions from other industries, which include safety checklists [26] and emergency manuals [27], learning from other sectors is not a straightforward task [28]. There is room for improvement in healthcare by understanding the mechanisms, systems, attitudes, and values that underpin the successful application of these techniques in other domains. The prevention of harm to patients is at the forefront of providing safety in healthcare.

### 2.1.2 Patient Safety and Degrees of Harm

**Patient safety** is defined as *"The avoidance, prevention, and amelioration of adverse outcomes or injuries stemming from the process of healthcare"* [29]. Although it is related to the quality of care a patient receives in a hospital or clinic, these two concepts are not necessarily the same. Safety is a single dimension of quality, which is a broader term. The quality of healthcare, according to [18], is encapsulated by six dimensions which include safety, effectiveness, patient-centredness, time, efficiency, and equity. The Institute of Medicine report 'Crossing the Quality Chasm' provides a story about a working mother who suffered a preventable but long-lasting disability because of poor quality of care. From this story, it is evident that there is little separating safety and quality of care. She suffered from this harm not directly because of a drug or surgery, but due to inefficiency, delay, and a non-patient-centred approach. Safety is one dimension of quality of care which is the most critical and important to patients.

According to the World Health Organization (WHO), *"Patient safety is the absence of preventable harm to a patient during the process of healthcare and reduction of risk of unnecessary harm associated with healthcare to an acceptable minimum"* [30]. The "acceptable minimum" here refers to the context of given knowledge and resources, in which the quality of care is delivered against the risk involved in treatment or non-treatment. In the UK, the NHS defines it as *"Patient safety is about maximising the things that go right and minimising the things that go wrong for people experiencing healthcare"* [31]. This definition is more realistic because as

humans, there is still a chance of error, but this needs to be at a minimum level to provide better care for patients.

In the context of healthcare, a **hazard** is defined as “*a potential source of harm to a patient,*” while **clinical risk** is defined as the “*combination of the severity of harm to a patient and the likelihood of occurrence of that harm*” by the NHS [32]. **Harm** is defined as “*death, physical injury, psychological trauma and/or damage to the health or well-being of a patient*”. Risk is assessed as a combination of severity and likelihood. The severity is defined as “the measure of possible consequences of a hazard.” The severity of a hazard can be either quantitative (e.g. the number of deaths) or a qualitative classification (e.g. minor, significant, major, catastrophic). Similarly, the likelihood or probability can also be either a quantitative or a qualitative classification (e.g. very high, high, medium, low, or very low). To assure the safety of a system, it is necessary to ensure all potential hazards and risks are mitigated to an acceptable level before the operation of the system. The prevention of harm to patients is the main constituent of patient safety, as can be understood from the above definitions. Harms can be caused by healthcare hazards, which include medication error, wrong dosage, or infection from medical devices. The healthcare process to a certain extent contains elements that can cause hazards, and those hazards can ultimately harm patients.

**Degrees of Harm** The National Reporting and Learning System (NRLS) provides five degrees of harm experienced by patients [33].

- **No harm:** a situation described by having no harm incident or a prevented safety incident.
- **Low harm:** an incident needing minor care and causing minimal harm to a person.
- **Moderate harm:** any incident resulting in further treatment which might include surgical intervention and caused short-term harm to a person.
- **Severe harm:** an incident causing permanent or long-term harm.
- **Death:** an event that results in the death of a person.

### 2.1.3 Arguing Safety: The Safety Case and GSN

A safety case is a structured argument that is supported by evidence that a system is safe to use in a particular context [34]. The core of the safety case is a risk-based argument with evidence that demonstrates that all risks associated with the system have been identified and that appropriate risk controls have been put in place. Accordingly, it must be clear, concise, and present a compelling argument with supporting evidence. In the literature, the more generalised term “Assurance Case” [35] is also used interchangeably with the term safety case. In this thesis, however, the term safety case is our focus and is used throughout to define the safety of the systems.

A safety case typically contains three key elements: claims, arguments, and evidence [34]. The objective of the safety case is defined by the top claim; the argument explicitly shows how claims are satisfied by the evidence for the safety of the system; and the evidence describes the measures taken to support the claim. All these elements are important, as without a proper argument, evidence may not be understood well, and without solid evidence, the argument is meaningless.

**Safety Case Representation** Safety cases may be written in text form or represented by a graphical notation. The graphical form communicates more clearly and is easier to review than text. The earlier forms of safety cases were quite complex, with some requiring a large volume of text to describe the safety cases, which were often unmanageable. The **Goal Structuring Notation (GSN)**, developed at the University of York, identified these issues and provided a structured and graphical representation for the better management of safety cases [36]. GSN is now a widely used graphical representation that explicitly represents elements of a safety argument (requirements, claims, evidence, and context) and the relationships that exist between them. Some other graphical representations of safety case development, such as Claims, Arguments and Evidence (CAE) [37], are also used. This thesis focuses on GSN as the standard graphical representation for the development of safety cases.

There are four key elements of the GSN which are explained below:

- **Goal:** The overall top-level safety claim about a system is represented by a "Goal." For example, the system is acceptably safe to operate in a given

environment. Goals are further divided into sub-goals, either directly or indirectly, by using a strategy.

- **Strategy:** Strategies are used to split the top-level goals into more achievable sub-goals by providing a rationale.
- **Context:** These are the conditions that provide constraints to goals or strategies. There is no possible way a system can be safe in any available context. The context defines the bounding area of the system.
- **Solution:** The safety evidence for an end-level or leaf goal.

The four main constructs of GSN described above are combined with relationship elements forming a goal structure [36]. GSN supports two types of relationship elements, which are described below:

- **Supported by:** This relationship is indicated by drawing an arrow with a solid filled arrowhead and is used to document inferential or evidential relationships. Inferential relationships show an inference between goals, while evidential relationships state the link between a goal and the evidence used to support it.
- **In context of:** This relationship is specified by rendering an arrow with a hollow arrowhead and is used for contextual relationships.



Figure 1: Graphical notation of core GSN constructs and relationships [36]

Figure 1 shows the graphical notation of the goal structures of GSN and the GSN relationship elements. This figure also shows the notation of an undeveloped goal. This is created by drawing a hollow diamond beneath the rectangular symbol of a goal structure. An undeveloped goal represents a claim that is intentionally left undeveloped in the argument. The other two elements of GSN presented in this figure, justification and assumption, are not very commonly used. Sometimes claims or strategies need to be expressed in the context of some assumption. The GSN justification element is used when a claim or strategy requires more explanation as to why it is considered acceptable by the author.

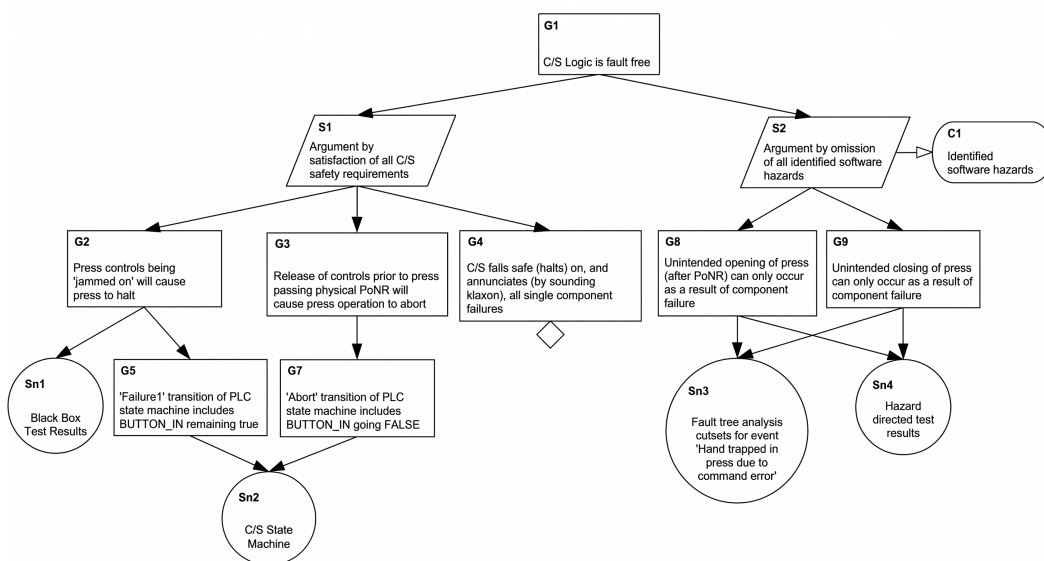


Figure 2: An example safety case constructed using GSN [34]

Figure 2 shows an example of a goal structure from Kelly [34]. This example has a singular top-level goal, which is 'C/S logic is fault free'. This top-level goal is then subdivided into sub-level goals through strategies S1 and S2. The strategy S2 is made in the context of C1, which explains all identified software hazards. The C1 is linked to S2 by a GSN contextual relationship element. There are five sub-level goals, and the goal G4 is left undeveloped. Unlike goals G8 and G9, which have direct evidence of their claim, goals G2 and G3 are sub-divided into G5 and G7. The argument in goals G5 and G7 is satisfied by a single proof or solution, Sn2.

## 2.1.4 Safety Analysis Methodologies

A safety analysis method is used to identify hazards and potential risks to the safety of a system. Hazard and safety analysis of computer-based systems often use variants of HAZOP, a technique initially designed for the chemical industry to analyse process flows [38]. In this thesis, we focus on a variant of HAZOP for computer-based systems, SHARD [39], for the hazard/risk analysis of CAs. The SHARD method is based on the concept of information flows within the system to analyse hazards and their causes. Other computer-based analyses have certain limitations [39]. For example, software FMEA [40] fails to provide a good model for software failures and lacks in the identification of causes for failures. The notations in software fault trees [41] are inconvenient for obtaining useful analysis information, and the size to model software behaviours can be quite large. Petri Nets [42] are expensive, and there are limitations in constructing the reachability graphs which model the states of the system.

Healthcare is often encouraged to use safety techniques from other safety-critical industries. Apart from the safety case, some hazard and reliability analysis methods from other industries are used in healthcare. To assure the safety of a system is to ensure all potential hazards and risks are mitigated to an acceptable level before the operation of the system. In the UK, this is often stated as the "as low as reasonably practicable" (ALARP) principle. In the context of risk management, making sure a risk has been reduced to ALARP means weighing the risk against the measures necessary to further reduce it. There are various techniques for assessing and identifying hazards in safety-critical industries that are used in healthcare [43–45].

### 2.1.4.1 Checklists

Checklists are a basic technique for ensuring that certain tasks get done. Checklists are based on the argument that there are certain cognitive limitations of the human brain [26, 46]. They are a simple solution to potentially complex problems in many industries. They are used extensively in aviation and are an integral part of their workflow. For example, in aviation, there are three forms of checklists: one for routine operations, one for complex operations, and one for emergencies [23]. The healthcare application of checklists includes use in the field of surgery [43] and infection control [47]. However, in comparing checklists in healthcare versus other industries, there are important qualitative differences. A comparison made

by Catchpole et al. showed vast differences between checklists in aviation and healthcare in terms of the size, the number of words used for each checklist, and the type of checklist used (which may or may not require confirmation) [48]. This is due to the complex nature of operations in healthcare, as Williams et al. argued regarding their effective usage in complex clinical settings [46]. A checklist is a socio-technical intervention, and it requires many other ingredients to be designed and implemented in healthcare. For example, to be used successfully in healthcare, it requires focused effort, communication, and teamwork, in addition to simple checks. Communication and teamwork also require training people to use these skills. Checklists can be a useful tool if used with the right design and implementation and in the right place with people who have the right skills. Their translation from other industries cannot be applied directly to healthcare but rather requires precise design, right placement, and proper training for the people using them. They are not as simple to apply as they may seem, and their application in healthcare requires learning the complex factors and changes to culture, design, and teamwork that accompanied them in other industries [26, 48].

#### **2.1.4.2 Failure Mode and Effects Analysis (FMEA)**

Failure Mode and Effects Analysis (FMEA) is a systematic approach to risk mitigation. FMEA works to identify failure modes that can cause a system to fail and the effects of those failure modes on the overall system [49]. The FMEA analysis is used to recognise, prioritise, and limit these failure modes. After identifying the failure modes, they are assigned a single Risk Priority Number (RPN), which is calculated by multiplying the probability, severity, and the detectability of a failure mode occurring. The FMEA technique has different variants, such as Healthcare Failure Mode and Effect Analysis (HFMEA) and Failure Mode, Effects and Criticality Analysis (FMECA). FMEA is a widely used process hazard analysis (PHA) technique used in healthcare [44]. HFMEA was developed by the Department of Veterans Affairs (VA) National Center for Patient Safety and is a five-step process which includes: Defining the topic, Assembling a multidisciplinary team, graphically describing the process, conducting a hazard analysis, and determining actions and outcome measures [50]. FMEA is thorough and systematic, but also costlier to implement (in terms of staffing hours required). It identifies an examination of all possible failures in a system, which also requires a significant time investment, a serious limitation. FMEA is also

resource-intensive, and there is a lack of guidance for hazard identification and risk assessment [51]. Additionally, the calculation of the RPN is not always relevant; as Franklin et al. pointed out, there is a mathematical and logical flaw in calculating the RPN and taking measures based on the resulting number [52].

#### **2.1.4.3 Hierarchical Task Analysis (HTA)**

Hierarchical Task Analysis (HTA) is a task description method used to decompose a task hierarchically into smaller units. The structure consists of goals, sub-goals, operations, and plans [53]. HTA can be used as a basis for further analyses, such as error analysis. In healthcare, HTA has been used in various forms. One study showed its use in surgery, as it is a procedure that cannot be done solely by a surgeon [54]. The surgeon may need a surgical assistant, an anaesthetist, an endoscopy nurse, and others. The individual task analyses of these other people, combined with the surgeon's analysis, make up the complete HTA. Lane et al. demonstrated the use of HTA to reduce errors in medication administration in a hospital [55]. They used the Systematic Human Error Reduction and Prediction Approach (SHERPA), which was complemented by HTA.

#### **2.1.4.4 SHARD**

SHARD analysis is structured around the information flows between the components of the system [39]. Each information flow is guided by a set of guidewords that are used to identify deviations from the intended functionality of the system. For each deviation considered, possible causes and effects of the deviations are recorded. The analysis is recorded in a tabular format, with separate tables for each information flow. The analysis starts at the system level and proceeds backwards, starting from the output of a system or function. In a CA, this could be the current response generated by the system in response to a user utterance. Working backwards from the outputs makes it easier to understand deviations that may contribute to a hazard. Deviations in selected information flows are tagged with guidewords to analyse unintended behaviour from the normal flow. The SHARD guidewords are based on the concept of a service, which is the communication of a piece of information with a specific value at a particular time. The guidewords suggest the ways that they may go wrong, which are [39]:

- Omission: The service is never delivered.

- Commission: A service is delivered when not required.
- Early: The service is delivered earlier than intended.
- Late: The service occurs later than intended.
- Value: The information delivered has the wrong value.

Once a deviation is selected, the next steps in the analysis are to consider if the deviation has plausible causes and/or potentially hazardous effects. For each deviation, system-level hazards that it can cause or contribute to are recorded. Deviations which do not cause or contribute to a hazard are also recorded. The deviations which contribute to potential hazards are then examined for potential mechanisms for the detection, protection, and mitigation of those hazards. Habli et al. [56] used SHARD guidewords as failure classes to refine identified hazards to hazard instances of their medical application. As an example, the hazard 'medication not prescribed when intended' was classified as an omission error. The guidewords in SHARD help refine the hazards which are identified in the previous steps. McDermid et al. [57] applied SHARD because of its benefit over HAZOP, as it can be applied by an individual and thus saves time. SHARD has also been applied to healthcare and ML cases [45].

#### **2.1.4.5 Hazard and Operability Study (HAZOP)**

HAZOP, traditionally rooted in the chemical process industry, is a structured and systematic technique used to identify potential hazards and operability problems by examining deviations from the design intent of a system [58]. This is typically implemented through a multidisciplinary team using a set of "guidewords" (e.g. NO, MORE, LESS, AS WELL AS, PART OF, REVERSE, OTHER THAN) applied to system parameters and operations. While its application to digital health systems is an emerging area, conventional HAZOP studies can be time-consuming, labour-intensive, and heavily reliant on the subjective judgment of the participating experts [59]. Recognising these limitations, research efforts are underway to develop intelligent systems and varying levels of automation for HAZOP, including knowledge-based approaches and data-driven models [59]. The International AI Safety Report 2025 also acknowledges HAZOP as a relevant safety analysis technique in the context of AI systems [60].

#### **2.1.4.6 HAZOP-UML**

HAZOP-UML utilises the structured, guideword-driven inquiry of Hazard and Operability (HAZOP) studies with the clear interaction modelling capabilities of the Unified Modeling Language (UML) [61]. The HAZOP-UML process involves systematically applying a set of guidewords to the attributes of the UML diagram to identify potential deviations from the intended operation. HAZOP-UML provides various UML diagrams to model system behaviours, such as Use case diagram, sequence diagram, and state-machine diagrams each with their own guidewords [62]. For CAs, sequence diagrams can be applied which models the sequence of messages passed between the actor (user) and the system entity (CA).

#### **2.1.5 The Regulatory Landscape for Health IT and Medical Devices**

In healthcare, the safety case concept is relatively new [63]; in other high-hazard sectors (oil, nuclear, rail) it is long-established [64, 34]. The safety case helps regulatory bodies in maintaining a check on developers and operators to ensure they have adopted a systematic approach to appropriately manage risk. There has been a development of using safety cases in healthcare recently, but it is limited to medical devices and health systems. For example, there is guidance on the safe use of infusion pumps from the Food and Drug Administration (FDA) [65]. This is to assist industry in preparing premarket submissions for infusion pumps and to identify device features that manufacturers need to address in the product life cycle. The recommendations provided by the FDA in this document are intended to improve the quality of infusion pumps and prevent adverse events associated with their use. The safety case is defined by the FDA as: *“The safety assurance case (or safety case) consists of a structured argument, supported by a body of valid scientific evidence that provides an organised case that the infusion pump adequately addresses hazards associated with its intended use within its environment of use. The argument should be commensurate with the potential risk posed by the infusion pump, the complexity of the infusion pump, and the familiarity with the identified risks and mitigation measures.”* This definition is specific to the use of infusion pumps. In healthcare to date, the focus on safety cases is on medical devices and health IT systems. These two are defined in detail in the next sections.

### 2.1.5.1 Medical Devices

The safety case concept in healthcare has seen its most significant advancements in the area of medical devices. The EU regulation defines a 'medical device' as "any instrument, apparatus, appliance, software, implant, reagent, material or other article intended by the manufacturer to be used, alone or in combination, for human beings for one or more of the following specific medical purposes:

- diagnosis, prevention, monitoring, prediction, prognosis, treatment or alleviation of disease,
- diagnosis, monitoring, treatment, alleviation of, or compensation for, an injury or disability,
- the investigation, replacement or modification of the anatomy or of a physiological or pathological process or state,
- providing information by means of in vitro examination of specimens derived from the human body, including organ, blood and tissue donations, and which does not achieve its principal intended action by pharmacological, immunological or metabolic means, in or on the human body, but which may be assisted in its function by such means" [66].

An "active device," under which software falls as a medical device, is defined as "any device, the operation of which depends on a source of energy other than that generated by the human body for that purpose, or by gravity, and which acts by changing the density of or converting that energy" [66].

For the safe use of these medical devices, both international and local standards exist. ISO 14971 [67] is an international standard that defines requirements for risk management and best practices throughout the life cycle of medical devices. It guides on identifying hazards and hazardous situations, estimating and evaluating risks, and specifying risk control measures. Similarly, ISO 13485 [68] is an international standard related to medical devices but is focused more on the organisations that provide medical devices and related services. While standards provide rules, guidance, and best practices on use in consensus by a recognised body, their compliance is not always mandatory. Regulation, on the other hand, is mandatory and is enforced by government. The role of regulators is to provide requirements, guidance, and advice to manufacturers of systems for their safe use. They can also warn or stop manufacturers or operators from operating their system.

The regulation body for medical devices and medicines in the UK is known as the Medicines and Healthcare products Regulatory Agency (MHRA). The MHRA is responsible for ensuring the safety, quality, and effectiveness of these devices and medicines. Software applications that function as a medical device are required to be CE marked to ensure regulators that they are acceptably safe to use and apply in healthcare. Figure 3 shows the flowchart from the MHRA, which helps in

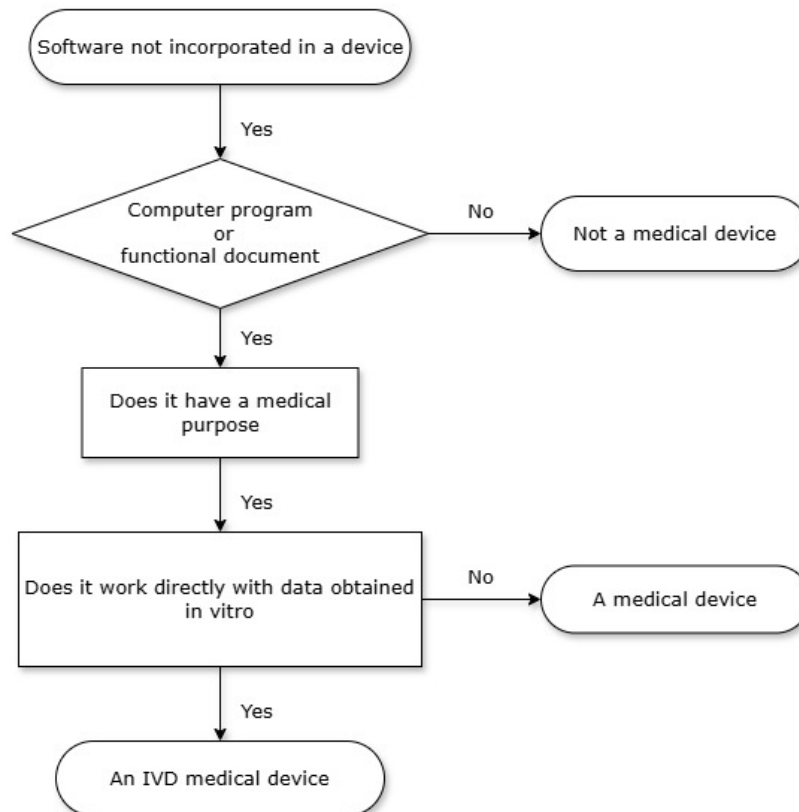


Figure 3: MHRA - device determination flow chart [69]

identifying if a software application needs to be CE marked [69]. Since 1st January 2021, new regulations post-Brexit require medical devices to have a UKCA mark, although existing CE-marked applications are still valid for a transitional period [70]. The MHRA is also working on a roadmap to define regulations for Software and AI as a Medical Device, which has not yet come into effect [71].

The flowchart in Figure 4 is a continuation of Figure 3 and lays out in detail the definition of a medical device according to the MHRA [69]. The MHRA’s regulation is aligned with the international standards for medical devices, ISO 14971 and ISO 13485, mentioned earlier.

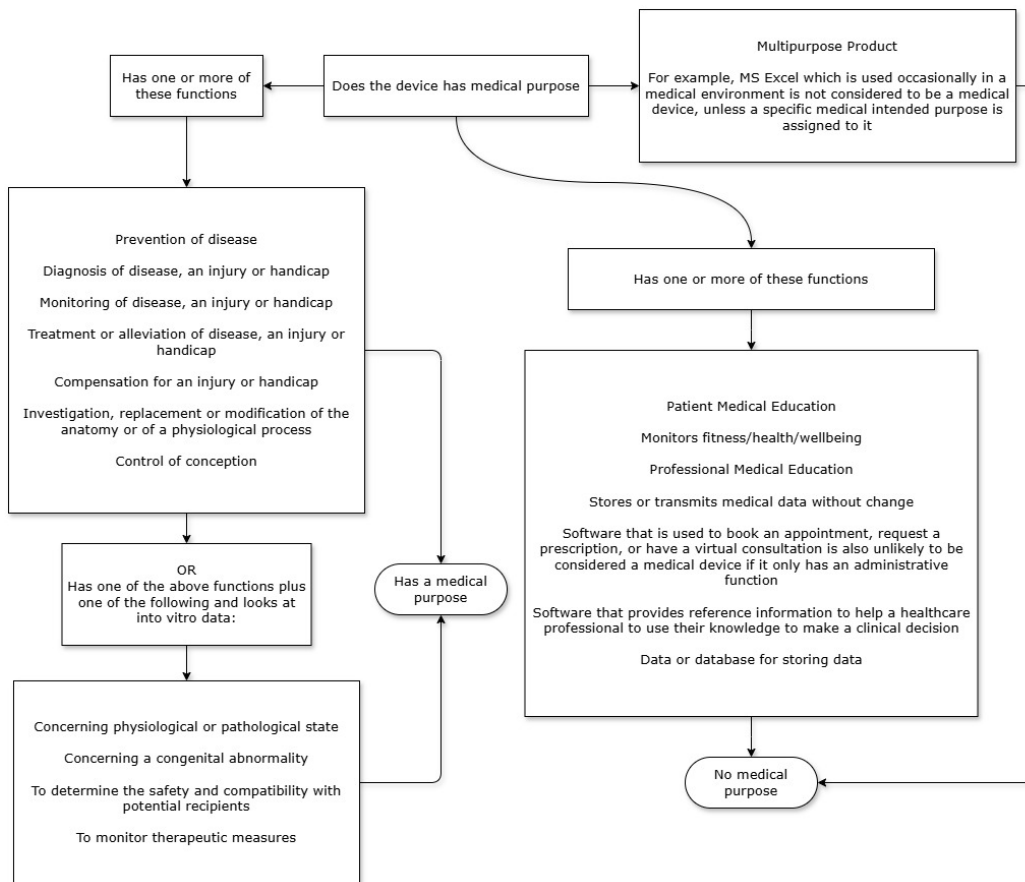


Figure 4: MHRA : medical device determination flow chart

The MHRA document further adds information about symptom checkers which use AI and CAs to interact with users. If the software provides a subset of medical conditions that matches a user’s input, indicates the likelihood of a match, or provides recommendations for entered conditions, then it is considered a medical device. On the other hand, the software will not be considered a medical device if it provides only reference information or directs a user to suitable care, such as a GP. Symptom checkers are considered low-risk devices unless they provide a direct diagnosis, in which case they are regarded as medium-risk devices. As modern CAs become more sophisticated in the diagnosis of diseases, a safety case may greatly help both developers and regulators ensure their safe use. In the previously mentioned FDA guidance on infusion pumps, the agency also proposes GSN as one of many options for developing a safety case for infusion pumps [65].

To illustrate how a device-level safety case can be structured for an infusion pump, we summarise a Generic Infusion Pump (GIP) example. In practice, there are two

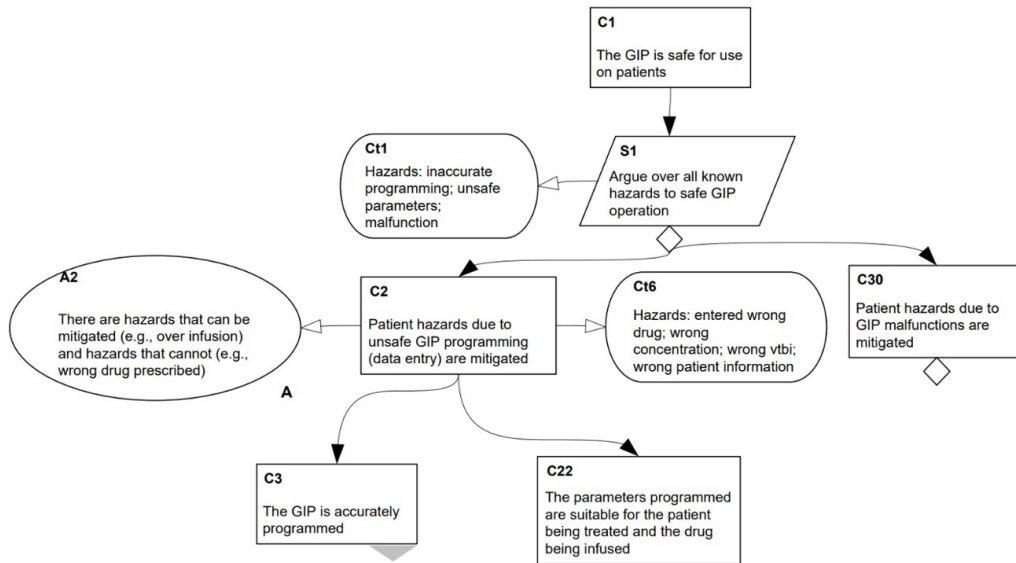


Figure 5: The GIP overall top level safety case argument [72]

complementary styles of argument: (i) show that all identified safety requirements are satisfied; or (ii) focus on hazards and show they are mitigated to an acceptable level.

Figure 5 [72] illustrates the top-level claim of the safety case of GIP. The top-level claim or goal “The GIP is safe for use on patients” is subdivided into claims C2 and C30 based on two possible hazards to the patient by its usage. The claim C30 needs further development as shown by the undeveloped goal symbol. The focus of this example is the argument C2 that patient hazards due to unsafe programming are mitigated. There are some hazards that cannot be mitigated with the safe use of GIP programming and are captured as GSN element assumption A2. The GSN context element C16 lists the classes of hazards to patients. The argument C2 further divides into C3 that the GIP is accurately programmed and C22 that the parameters of the drug are safe for the patient. Figure 6 [72] represents the claim C3 and its evidence. The claim C3 is subdivided by 3 claims. The claim C19 refers to the tolerable rate of parameter entry by the person. The evidence to this claim comes from the error log and GIP procedure manual. These leaf of GSN satisfies claim of tolerable entry errors which with other sub-claims satisfies the claim of accurately programmed GIP which in turn with its sub-claims satisfies the main claim of having a safe GIP for use on patients.

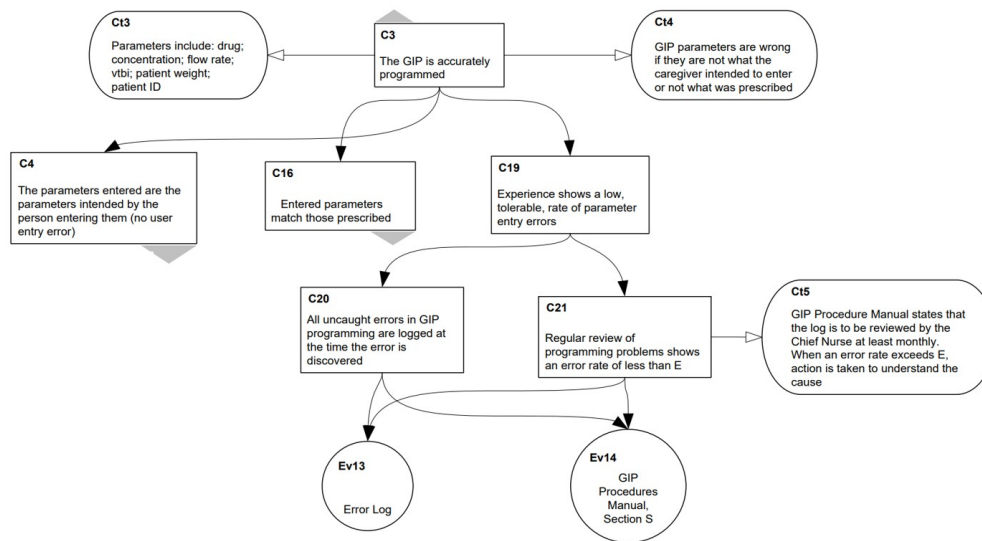


Figure 6: The GIP medical device safety case sub-argument [72]

### 2.1.5.2 Health IT Systems

Health IT systems are defined as a “Product used to provide electronic information for health or social care purposes. The product may be hardware, software or a combination.” [73, 32] Normally, a health IT system contains software running on computers or mobile devices, while medical devices are stand-alone and have embedded software. As described in the last section, according to the MHRA device determination flowchart, software has the potential to be a medical device if it goes beyond the dissemination and communication of information. Digital health technology has the potential to mitigate risks, but can also be a source for introducing new clinical risks. In the UK, therefore, NHS Digital has developed the national clinical standards DCB0129 [73] and DCB0160 [32]. These two standards overlap, as the former provides requirements for clinical risk management for manufacturers, while the latter sets requirements for the use, deployment, and maintenance of health IT systems. Manufacturing organisations of health IT systems and applications need to perform a formal risk assessment and provide evidence of the measures taken to mitigate clinical risks. They are required to produce a clinical risk management plan, a hazard log, and a clinical safety case report to comply with these standards.

This established regulatory focus on software as a medical device, driven by standards like DCB0129/0160 and the detailed guidance from the MHRA on AI-driven symptom checkers, provides a direct precedent for the core subject of this thesis.

As software, particularly artificially intelligent software, becomes the primary active component in these systems, a deeper understanding of its specific architecture and failure modes becomes paramount. The following sections therefore shift the focus from general safety principles to the specific technology at the heart of this research: CAs.

## **2.2 Conversational Agents (CAs)**

Conversation, or dialogue, is a fundamental mode of human communication. From early childhood, we learn to use dialogue to coordinate tasks, express needs, and share information in everyday life. From ordering our lunch at a restaurant to talking with our friends, booking air tickets or participating in meetings, we indulge in conversations to get our tasks done or express our feelings. Dialogue systems or CAs are types of programs that mimic human language. These programs use text, speech or both to communicate with users in natural language. The definition of them is not clear and there are various terms associated with the literature: CAs, chatbots, personal assistant, voice-based assistants, voice user interfaces, dialogue systems, etc. Conversational systems fall into two broader classes namely: Task-oriented dialogue agents, and chatbots [74]. The former helps the user to complete their tasks while the latter is designed for extended conversations.

Modern digital assistants including Amazon Alexa, Google Home, Microsoft Cortana, Siri are classified as task-oriented dialogue systems. Their application include, but are not limited to, making reservations at restaurants, dialling phone calls, giving user's directions, opening applications on smartphones, playing games, and listening to music or jokes. In contrast, chatbots aim to imitate open-ended human-human conversations. The term CAs is used for task-oriented dialogue systems as is common in the literature [75, 1, 76]. Below is a historical context on different types of chatbots and how they evolved over time.

### **2.2.1 Chatbots**

Chatbots are the simplest example of dialogue systems that carry on an extended conversation with the objective of 'chats' or unstructured conversations. The architecture of chatbots can be classified into two main categories: rule-based systems and corpus-based systems [74].

### 2.2.1.1 Rule-based Chatbot

**ELIZA** chatbot is the most important in the history of dialogue systems. Designed in 1966, it was created to simulate a Rogerian psychotherapist [77]. In this psychology, a conversation starts without knowing anything about the real world. ELIZA was based on rule and pattern matching algorithm where the chatbot inferred information from the user and a response is generated on that. Each rule in the algorithm is linked to a possible keyword that a user might use in their sentence. For example, Figure 7 below is one pattern/rule of ELIZA [74].

```
{0 YOU 0 ME} [pattern]
→ (WHAT MAKES YOU THINK I 3 YOU) [transform]

0 here is Kleene* operation, and in the transformation rules, the numbers
represent the index of the component in the pattern. In this example, 3
represents the second 0 in the pattern.

According to this rule
You hate me
will be transformed to:
WHAT MAKES YOU THINK I HATE YOU
```

Figure 7: ELIZA chatbot

ELIZA's pattern/action architecture is still used by modern chatbots systems such as ALICE. **PARRY** is another chatbot which focuses on clinical psychology and was developed in 1971 [78]. PARRY included a mental model which was affected by its state of anger or fear based on the interaction. A high anger value turns PARRY to choose a 'hostile' response while a high 'fear' value for example based on an input which mentions its delusion, would make PARRY express statements related to its delusion. PARRY was the first chatbot to pass Turing test which is a measure of intelligence of a dialogue system [79].

### 2.2.1.2 Corpus-based Chatbots

Corpus-based chatbots, unlike rule-based chatbots, build conversations using mining lots of human-human or human part of human-machine conversations. According to an estimate [80], modern chatbots needs hundreds of millions or even billions of words to train their data. The source of data may come from the movies database

[81], text from social media platforms [82], etc. A trained chatbot uses human conversations during their interaction with the chatbot to enhance its learning. The increased training data set helps chatbot respond more naturally. There are also other ways in which corpora can be built for training a chatbot. Some topic-specific corpora are used to train topical chatbots.

There are two main architectures for corpus-based chatbots: information retrieval and machine-learned sequence transduction [74]. Context modelling is not very common in these chatbots but rather they tend to respond based on the user's current utterances. Corpus-based chatbots are like question answering systems where a response is generated usually by ignoring the full context of the conversation.

### 2.2.1.3 Information Retrieval IR-based Chatbots

The main idea behind IR based chatbots is to respond to a user by selecting an appropriate response from a corpus of natural language text. The problem with the rule-based approach is that the developer must specify every possible 'pattern/rule' to be able to provide a complete response. IR based chatbots retrieve information from the corpus that has stored conversations in pairs of the form of turns of a conversation. Having corpora and a user's sentence, any retrieval algorithm can be used to generate a response. There are two methods for returning a response.

The response to most similar turn approach works better in practice [83]. The reason is that user's query  $q$  if most similar to the turn  $t$  then it means it is more effective since the co-occurrence of words in them. If it is matched against the turn itself, it is not clear if the response is more accurate because of the similarity between the user's query  $q$  and the turn  $t$  itself.

To compute the similarity any similarity function can be used. For example, cosine similarity of words or over any sentence embedding can be used. Occasionally keyword matching is used where keywords are looked for matching in user's query  $q$  and the document in corpus  $C$ . In complex IR models, more features can be added than just the words in user's query  $q$ . When there are few words in the user's query, all conversation can be used to add more meaning in matching. Also, the sentiment of a user can be helpful.

1. **Return the response to most similar turn:** The main idea is to choose a turn which is most like the user's turn and return the stored human response to that

turn. For a query  $q$  and a corpus  $C$ , a turn  $t \in C$  that is most like  $q$ , and return that turn's human response, i.e.,

$$r = \text{response} \left( \arg \max_{t \in C} \frac{q^T t}{\|q\| \|t\|} \right)$$

2. **Return the most similar turn:** Here a user's query  $q$  is matched to the turn  $t$  in corpus  $C$  and is returned because it will share words or semantics with the turn. For a user's query  $q$ , return the turn  $t \in C$  which is most like  $q$ :

$$r = \arg \max_{t \in C} \frac{q^T t}{\|q\| \|t\|}$$

## 2.2.2 Task-Oriented Dialogue Agents

The goal of these type of dialogue agents is to help the user in achieving a task such as booking an aeroplane ticket, schedule an appointment, etc. This section will introduce types of task-oriented conversational systems referred to as CAs [75].

### 2.2.2.1 GUS - A Frame-Based Architecture

Most of the modern voice-based assistants are based on Genial Understander System (GUS) system architecture which was first introduced in 1977 [84]. These dialogue state architectures are based on frames. A single frame represents a knowledge structure representing various features it can extract from the user's utterance and consists of various slots. These slots can take a set of possible values. For example, in a healthcare domain, a slot might represent care unit (which can take values such as "primary care" or "emergency care"), location (e.g. the city name), or date and time. The types in GUS systems as well as in modern frame-based dialogue systems have a hierarchical structure. Date type, for example, is a frame with slots of integers or values of sets of weekday names [74] as shown in Figure 8.

**Control Structure** The control structure of these dialogue agents is designed around frames and is used in modern voice-based assistants such as Google Home, Apple's Siri, and Amazon's Alexa. The system interprets the user's intention and fills the slot values in the frame and perform relevant action. To achieve this, the system keeps asking questions from the user (each slot of each frame has pre-defined

```

DATE
MONTH:NAME YEAR:INTEGER DAY:(BOUNDED-INTEGER 1 31)
WEEKDAY:(MEMBER (Sunday Monday Tuesday Wednesday Thursday Friday
Saturday))

```

Figure 8: Date type in GUS architecture

question templates) and fills slots that the user specifies. The GUS architecture also provides slots with condition-action rules attached to them. For example, a rule attached to the HOSPITAL\_NAME slot for the search emergency care frame might automatically assign as the default Medical Centre for the related general practitioner booking frame. Some domains also require multiple frames. For example, there are general information frames for questions like *which hospitals or care units are between my home and my workplace*, or for information like *how much typically I have to wait for an appointment at a specific hospital?*. The GUS architecture is a production rule system because of its need to dynamically switch controls. Different inputs fire different productions each of which can fill in different frames.

**Natural Language Understanding for Slot Filling** The goal of this component is to extract three main features from the conversation which are domain classification, intent determination, and slot filling. domain classification helps to identify a broader domain of the user's query such as airline, programming alarm etc. After domain selection, the need to understand the user's general task or goal is fulfilled by determining the intent. Intent could be book or view flights, view or remove alarms etc. Finally, slot filling is required to extract the particular slots and fillers that are needed by the system to understand from the user's utterance. For example, Figure 9 shows intent generation for a user's request.

```

Wake me up tomorrow at 10 o'clock

produces the following intent:

DOMAIN: ALARM-CLOCK
INTENT: SET-ALARM
TIME: 2020-03-30 10:00:00

```

Figure 9: Sample intent generation from the user's utterance

The original GUS system uses manually designed rules for slot filling. For the above example, a regular expression can also be used for recognizing the intent. Many modern dialogue systems which use GUS architecture at their core use supervised machine learning for slot filling [74].

### 2.2.2.2 The Dialogue State Architecture

Most of the modern CAs are a more sophisticated version of GUS system architecture which was first introduced in 1977 [84]. The dialogue state architecture is what is referred to as the pipeline architecture for CAs which is common in practice and applied in healthcare CAs [85, 86]. The pipeline architecture consists of separate components for each function performed by a CA. These functions perform recognition, understanding, decision-making, and response generation. The response generation contains a text-to-speech (TTS) component for voice-based CAs. The text-based CAs do not require ASR and TTS components and are more common than voice-based counterparts in healthcare [87]. Figure 19 provides the architecture of CAs.

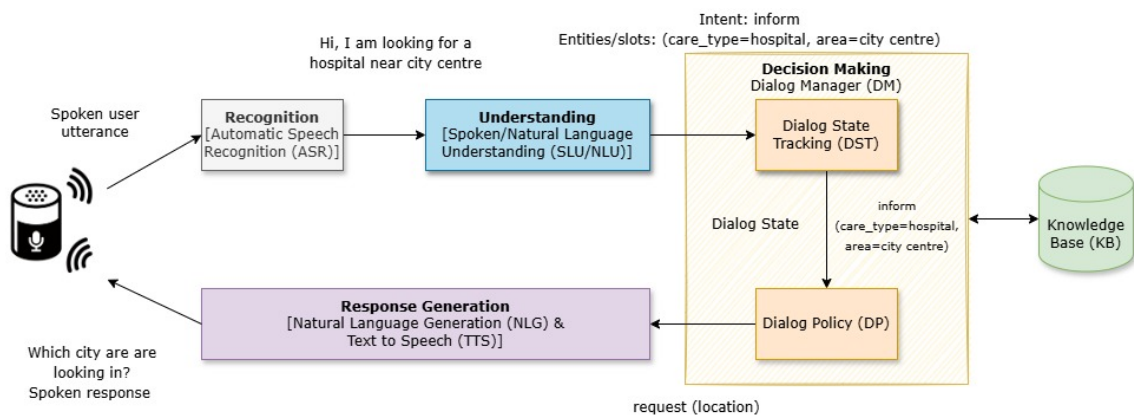


Figure 10: CA pipeline architecture

**Automatic Speech Recognition (ASR)** The ASR module is the entry point for a spoken dialogue system, responsible for transcribing the user's voice input into a string of text. The accuracy of this component is important, as errors in transcription can propagate downstream, potentially causing misinterpretation and communication failure.

**Natural Language Understanding (NLU)** Understanding in CAs is provided by the NLU component which maps user utterance to a structured semantic representation. The semantic representation is achieved usually by dialogue acts (DAs) which consist of intent and slot-value pairs. The intent is a high-level classification of utterance e.g. query, inform, request while slot-value pair is the semantic information required to achieve the task as shown in Table 1.

Table 1: Example DA for a CA providing care facilities

Utterance	Dialogue Act (DA)
Hi, I am looking for a hospital in the city centre.	<b>Intent:</b> inform <b>Slot-value:</b> (care type = hospital, area = city centre)

The tasks of NLU include domain identification, intent classification, and slot-value extraction [88]. Since the focus of the domain in our report is healthcare, we ignore multi-domain CAs. Intent detection is a classification task while slot-value extraction is a sequence labelling task. The following expressions describe intent detection and slot-value extraction respectively [85].

$$p_{\text{intent}}(d \mid x_1, x_2, \dots, x_n) \quad (2.1)$$

$$p_{\text{s-v}}(y_1, y_2, \dots, y_n \mid x_1, x_2, \dots, x_n) \quad (2.2)$$

Here  $d$  indicates the intent class, and each token in the utterance  $(x_1, x_2, \dots, x_n)$  is associated with a sequence of labels  $(y_1, y_2, \dots, y_n)$ ;  $n$  denotes the total number of tokens. Recurrent Neural Networks (RNNs) and their variants Long Short-Term Memory (LSTM) networks, are widely used in intent classification and slot-value extraction [89–91]. These networks use the hidden state of each token to predict its corresponding label  $y_i$ , and the final hidden state to infer the intent  $d$ . Yao et al. [89] applied RNNs to the ATIS dataset and achieved high slot F1 scores due to task-specific word representations learned during training. Similarly, Hakkani-Tür et al. [91] applied a joint RNN-LSTM model for intent classification and slot filling across ATIS and other datasets, observing slight performance improvements over separate models.

**Dialogue Manager (DM)** The Dialogue Manager (DM) is responsible for the decision-making in CAs. The DM consists of two parts: **Dialogue State Tracking (DST)** and **Dialogue Policy (DP)**. The DM is responsible for taking the next action based on the state of DST and DP for the conversation.

The task of DST is to approximate the user's goal at a time  $t$  taking into consideration the entire dialogue context or history. Recent work on DST uses belief state which consists of slot-value pairs to represent the user's goal [85]. The following expression represents the multi-task classification nature of this tracking:

$$p_i(d_i, t \mid u_1, u_2, u_3, \dots, u_t) \quad (2.3)$$

Here  $p_i$  represents the dialogue state tracker for each slot  $i$ , and  $u_t$  represents the utterance at turn  $t$ .  $d_{i,t}$  denotes the class of slot  $i$  in the  $t$ -th turn. Figure 11 shows a real-life example of DST after each turn for a restaurant recommendation system [92]. The DST in this example helps the CA decide which question to ask next in achieving their goal of finding a cheaper Thai restaurant around the city centre.

```

User: I am looking for a cheaper restaurant
      inform (price=cheap)
System: Sure. What kind - and where?
User: Thai food, somewhere downtown
      inform (price=cheap, food=Thai, area=center)
System: The House serves cheap Thai food
User: Where is it?
      inform(price=cheap,          food=Thai,          area=centre);
      request(address)
System: The House is at 106 Regent Street

```

Figure 11: DST example for a restaurant recommendation system

Henderson et al. [93] used RNNs for word-based DST and evaluated their approach on the DSCTC2 dataset and demonstrated high performance across all of the metrics. According to them, the RNNs can handle large dimensional inputs well and are naturally good with sequential input and outputs.

DP generates the next system action in the form of a DA based on the current state of the dialogue. It is the calculation of maximising the probability of an action a

to take at turn  $t$  based on the entire history of the dialogue state. The history contains all the acts from the system  $a$  and the user  $u$ . This is represented as follows:

$$a_t = \arg \max_{a_t \in \mathcal{A}} P(a_t | a_1, u_1, \dots, a_{t-1}, u_{t-1}) \quad (2.4)$$

DP by its nature is formulated as a Markov Decision Process (MDP) which can be addressed as Reinforcement Learning (RL) [85]. RL and its variants such as Deep-Q-Network (DQN) are also applied for the DP task [94, 95]. DQN is a classic RL algorithm combined with deep neural networks (DNNs). The DNNs and their variants (RNN, LSTMs, DQN) are commonly used throughout the understanding and decision-making functions of CAs. Because these architectures share similar deep learning building blocks (RNNs, LSTMs, DQNs), a common safety methodology can, in principle, be applied across understanding and decision-making components.

**Natural Language Generation (NLG)** Response generation in the CAs is provided by NLG which maps that DA to a natural language utterance after its generation from DP. Figure 12 provides an NLG template for a restaurant recommendation system [96]. The DA in this example is ‘recommend’ with slots ‘restaurant\_name’, ‘neighborhood’, and ‘cuisine’. The slots in the template are replaced by actual value during response generation.

```
recommend(restaurant_name= Au Midi, neighborhood = midtown, cuisine =
french)
1 restaurant_name is in Midtown and serves cuisine food.
2 There is a cuisine restaurant in Midtown called restaurant_name.
```

Figure 12: NLG template from a restaurant recommendation system

The utterance generated by the NLG must convey the true semantics of the DA and be natural and specific to human language to provide a better user experience [85]. Robust NLG generation by limited training data is a challenging task and template-based NLG methods [97] are commonly used. The templates are hand-crafted expressions generated by humans and the predicted DA value by DP are substituted in those templates. The benefit of this approach is that the chance of grammatical errors is less while their downside being time-consuming for larger applications. ML is used sometimes for the randomness of selecting from a set of hand-crafted templates to make the conversation more natural.

### 2.2.2.3 CA Evaluation Techniques and Domain-Specific Implications

It is important to rigorously evaluate the performance of CAs. For classification tasks, which are central to components like NLU and ASR, a standard set of metrics derived from the confusion matrix is used. However, the choice of which metric to prioritise is not a purely technical decision. It is deeply intertwined with the application's context, as the consequences of different types of errors can vary dramatically, a reality most starkly observed in high-stakes domains like healthcare.

### 2.2.2.4 The Confusion Matrix

The confusion matrix is a fundamental tool for evaluating a classification model. It is a table that summarises prediction results by cross-tabulating the model's predicted classes against the actual (ground-truth) classes [74]. For a binary classification problem, the matrix has four key components as shown in Table 2.

Table 2: Machine learning confusion matrix

Actual Values		Positive	Negative
Predicted Values	Positive	True Positive	False Positive
Predicted Values	Negative	False Negative	True Negative

- **True Positives (TP):** The model correctly predicts the positive class. For example, a medical diagnostic agent correctly identifies a patient as having a disease.
- **True Negatives (TN):** The model correctly predicts the negative class. For example, the agent correctly identifies a patient as healthy.
- **False Positives (FP) (Type I Error):** The model incorrectly predicts the positive class when the actual class is negative. For example, the agent incorrectly diagnoses a healthy patient as having the disease.
- **False Negatives (FN) (Type II Error):** The model incorrectly predicts the negative class when the actual class is positive. For example, the agent incorrectly diagnoses a sick patient as healthy.

### 2.2.2.5 Core Evaluation Metrics

From these four counts, several key performance metrics can be calculated, such as accuracy, precision, recall, and F1-score [74].

### 2.2.2.6 Performance Metrics

There are various ways in which the CA are evaluated. The evaluation criteria also depend on the nature of task CAs are carrying out. For CAs, task completion success is usually the criteria of their evaluation. user meaning is represented using intent and slot-value pairs. A “slot” is the semantic field (for example, care type or area Table 1), and the “value” is the specific value filled by the user (for example, hospital, city centre).

- **Slot Error Rate (SER):** This metric is commonly used in frame-based task-oriented dialogue systems. It quantifies the proportion of slot prediction errors, whether added, updated, or removed, relative to the total number of slots in a sentence.

– *Formula:*

$$\text{Slot Error Rate} = \frac{\text{Number of added, updated, or removed slots}}{\text{Total number of slots in the sentence}} \quad (2.5)$$

- *Interpretation:* A lower SER indicates better performance. It reflects how accurately the system predicted the slot structure of a given user utterance. This metric is particularly important when measuring semantic understanding in dialogue systems.

- **Accuracy:** This is the most intuitive metric, representing the proportion of all predictions that were correct.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.6)$$

**Limitation:** Accuracy can be dangerously misleading, especially in cases of *class imbalance*. For instance, if a disease is present in only 1% of the population, a model that always predicts "no disease" will have 99% accuracy but be completely useless for diagnosis.

- **Precision (Positive Predictive Value):** This metric measures the quality of the positive predictions. It answers the question: “Of all the times the model predicted positive, how often was it correct?”

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2.7)$$

**Interpretation:** High precision indicates a low false positive rate. It is critical in scenarios where a false positive is costly (e.g. flagging a legitimate email as spam).

- **Recall (Sensitivity or True Positive Rate):** This metric measures the quantity or completeness of the positive predictions. It answers the question: “Of all the actual positive cases, how many did the model successfully identify?”

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2.8)$$

**Interpretation:** High recall indicates a low false negative rate. It is critical in scenarios where a false negative is costly (e.g. failing to detect a fraudulent transaction or a disease).

- **F1-Score:** This metric provides a way to combine precision and recall into a single number. It is the harmonic mean of the two.

$$F_1\text{-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (2.9)$$

**Interpretation:** The F1-score is more robust for imbalanced classes than accuracy because it requires both precision and recall to be high for the score to be high. If either metric is low, the F1-score will be pulled down significantly.

### 2.2.2.7 Sensitivity and Impact in Healthcare Applications

In healthcare, these metrics are directly tied to clinical risk. Here, the cost of a false positive versus a false negative is profoundly asymmetric, and this asymmetry must dictate the entire evaluation strategy. Consider a CA designed to perform an initial screening for a life-threatening cancer.

- The **positive class** is the presence of cancer.

- A **False Negative (FN)** occurs when the agent tells a patient with cancer that they are healthy. The consequence is a missed diagnosis, delayed treatment, and potentially preventable death. The cost of this error is catastrophic.
- A **False Positive (FP)** occurs when the agent tells a healthy patient that they might have cancer. The consequence is patient anxiety and the cost and discomfort of a follow-up procedure (e.g. a biopsy) to confirm the negative result.

In this clinical context, the overwhelming priority is to **minimise false negatives**. This is mathematically equivalent to **maximising recall**. The system must be optimised to find every possible case of the disease, even at the expense of flagging some healthy individuals for further review. A model with 99% precision but 50% recall would be clinically irresponsible, as it would miss half of all cancer patients. Conversely, a model with 99% recall but lower precision would be far more valuable, as it acts as an effective, highly sensitive screening tool, leaving the task of ruling out false positives to human experts and more definitive tests.

### 2.2.3 Conversational Agents (CAs) in Healthcare

The use of CAs in healthcare is growing and there are many CA applications and research studies in this domain. They are being used for various purposes such as symptom checking, mental wellbeing, fitness and health, overcoming stress, assisting in medical information, fighting obesity, chronic diseases such as diabetes etc. Recent research studies show that the most active health domains for CA are mental health [1, 98, 7] and physical wellness Pereira and Díaz [98]. In the domain of oncology, numerous CAs are being used as well as shown by a recent research study [99]. From our literature review, we found out that most of the studies or application involving CAs are using text as a medium of conversation. Below we describe some studies and examples from the healthcare domain.

In assisting users with their mental health problems there are numerous CA applications in healthcare. These studies show the usage of CA in improving psychological well-being, reducing anxiety or depression, adherence in task execution, antipsychotic medication adherence, and psychoeducation. Cameron et al. demonstrate the use of CAs in mental health by developing ‘iHelp’ [100]. This CA provides guided self-assessment on stress, anxiety and depression, trauma,

sleep, alcohol, etc. Similarly, Inkster et al. in their study evaluated a mental health well-being application that uses AI for its conversation [101]. They found that users who engaged more with the conversational application reported lower depressive symptoms than those with low usage. Mujeeb et al. demonstrated a CA for the diagnosis of achluophobia (fear of the dark) and for autism in children by asking users a series of questions [102]. This CA concept can also be used in assisting human psychologists in clinics. Jungman et al. did their study on evaluating the diagnosis accuracy of Ada health [103]. They found that performance for mental health conditions was moderate to low. The authors noticed an increase in diagnostic accuracy when the application was used by psychotherapists. The study concluded that, for the general population, the app should be used with caution, and highlighted the need for improvement in coverage of childhood and adolescent mental health conditions.

In the health and fitness area, there are also several applications and research studies. Huang et al. developed a CA 'SWITCHes' for weight management in people with obesity [104]. This CA with the help of auxiliary data from sensors help users in diet and exercise tracking and provides tailored feedback and advice including eating tips and eating order. Fadhil proposes a CA solution to promote healthy eating by preventing weight gain in adults [105]. Their proposed system provides a personalised recommendation regarding a healthy diet, physical activity and healthy food preparation.

Some general-purpose CAs provide more than single functionality to users. For example, 'Mamabot' a proof of concept application that intends to assist pregnant woman is a general-purpose CA [106]. Users can search nearby pharmacies and hospitals in case of emergency provides symptom checking functionality, nutrition for children and children emergency management. Madhu et al. also propose a general-purpose CA to provide age-based medicine usage, symptom prediction, and information about medicines [107]. In one study of CAs, they were treated as general-purpose healthcare devices to check their accuracy and safety to the users [108]. Ma et al. have used Amazon Alexa based CA to monitor patients' general health condition via angel sensor [109]. The sensor can monitor the heart rate, blood oxygen, temperature, etc. of the user.

Some CA applications or studies about them show their usage in monitoring chronic diseases such as diabetes where it is important to follow medication or control diet. One such CA is 'Vidi' which acts as a virtual dietitian for diabetic

patients [110]. Similarly, the study by Levin et al. showed the implementation of a dialogue system to monitor chronic pain [100]. Ahmad et al. developed a medication reminder CA that also suggests medicine based on illness and the explanation about medicines [111]. Similar to this, the CA 'Chester' reminds the patients about their medication, answering questions and engaging in a dialogue to collect information for improved monitoring [112].

Many studies also focus on the applications of CAs in oncology or helping cancer patients. Piau et al. implemented a semi-automated CA to provide older patients with cancer care at home [113]. The other purpose of this application was to free up nurses by collecting the primary patient data over the phone. In a study involving a CA 'Vik', the authors concluded that CA can be useful to cancer patients by providing them support and answers to their concerns about their disease [114]. Furthermore, Vik improved medication adherence through reminders and educational content.

In a study by Razzaki et al. for checking the diagnostic accuracy of Babylon AI Triage and Diagnostic system, they found it to be close to human doctors and in some cases exceeded the human performance [115]. It provided more accuracy than the average of human doctors in identifying the condition modelled by a clinical vignette. The vignettes in the study were from preparation materials for the RCGP (Royal College General Practitioner Exam) Clinical Skills Assessment (CSA) and Applied Knowledge Test (AKT) focusing on the diagnostic component. In the safety and appropriateness of triage recommendations, Babylon provided safer triage recommendations than the doctors on average. The results of this study are encouraging for an AI-based application. However, the claim of AI supremacy from Babylon over human clinicians was unconvincing as the study involved few doctors and the result may be skewed because of poor judgement from any one subject. The data entered into the system was also done by doctors and not by lay users and the study offers unconvincing evidence as it was not evaluated in a realistic clinical environment [113].

A recent randomised trial of a modular, pipeline-based CA for depression showed greater improvement than a bibliotherapy control over 16 weeks [116]. At the same time, a recent review of health-behaviour chatbots reports benefits but notes study quality concerns and limited technical reporting, calling for more rigorous trials [117]. While many new studies explore large language models (LLMs), this chapter focuses on pipeline-based CAs to align with the study period and available evidence.

Table 3 summarises the literature studies of CAs used in healthcare.

### 2.2.3.1 Safety Implications of Clinical CAs

The increased research and focus on applying CAs to healthcare have notable implications but at the same time, they come with numerous safety issues. CAs in healthcare may harm the patients as there is more than one factor while diagnosing a patient. They might miss the personal factors involved with the patients to suggest a recommendation [118]. Data is a key construct of any ML algorithm and due to non-rich data sets available for healthcare purposes, the performance of CAs might suffer [7]. To use CAs in healthcare training data sets must be available for the specific medical domain otherwise it may be unsafe to use those systems.

Table 3: Overview of clinical CAs

Study	Healthcare Domain	Category / Features	CA Name (Medium)
Allen et al. [112]	Medication	Medication Advisor	Chester (Voice)
Levin et al. [119]	Health Monitoring	Chronic Pain Assessment & Monitoring	Unknown (Voice)
Ma et al. [109]	Health Monitoring	Health Answers via sensor data	Angel Echo (Voice)
Cameron et al. [100]	Mental Healthcare	Self-Assessment	iHelpr (Text)
Inkster et al. [101]	Mental Healthcare	Self-Assessment	Wysa (Text)
Mujeeb et al. [102]	Mental Healthcare	Diagnostic App	Aquabot (Text)
Huang et al. [104]	Health & Fitness	Obesity	SWITCHes (Text)
Fadhil [105]	Health & Fitness	Healthy Lifestyle Promotion	Unknown (Text)
Vaira et al. [106]	Prenatal Care	General Purpose	Mamabot (Text)
Madhu et al. [107]	Medication	Symptom-Based Disease Prediction	Unknown (Text)
Lokman et al. [110]	Health & Fitness	Virtual Dietitian	Vidi (Text)
Ahmad et al. [111]	Medication	Medication Reminder	Unknown (Text)
Piau et al. [113]	Oncology	Home care for Cancer Patients	Infinity (Text & Voice)
Chaix [114]	Oncology	Medication Adherence, Educational Content, Patient Support	Vik (Text)
Jungmann et al. [103]	General & Mental Health	Diagnostic App, Symptom Checker	Ada (Text)
Razzaki et al. [115]	Medical Triage	Diagnostic App Symptom Checker	Babylon AI Triage and Diagnostic System (Text)
Liu et al. [116]	Mental Healthcare	Depression self-help	XiaoNan (Text & Voice)

In a comparative study of CAs, it has been found that the systems with unconstrained input can cause harm to patients and should not be completely relied on [1, 108, 120]. The study shows that these CAs failed more than half of the time in situations that needed medical expertise. Furthermore, they led people to take actions that could have resulted in harm. These systems should be used under clinical supervision for queries requiring medical expertise. A similar study involving smartphone CAs assessed their response when asked about suicidal emergencies. The results show limited and sometimes inappropriate responses [121]. Most of the CAs responded by a web search for users to explore further information.

Bibault et al. [7] noted from their review study done on CAs in oncology that scarcity of the clinical trials outweighs their potential benefits for patients and the healthcare system. Authors from a study [122] also suggested the need for having an option for users to talk to their counsellor when dealing with very sensitive health conditions such as AIDS. Further, a CA should have the capability to link users to trained professional whenever the user mentions suicidal thoughts or self-harm [122]. Hodgson et al. in a study for checking the efficiency of speech recognition for electronic health records found an increased risk of errors with the potential to cause patient harm [123]. These studies show that CAs may pose safety risks when used for self-diagnosing. This is due to problems in understanding user's query, lack of appropriate training data, absence of clinician's monitoring and insufficient clinical trials and evidence of their effectiveness in their application. Therefore, it is essential to think of methods and approaches that can make their usage safe in addition to their potential benefits in healthcare. To identify potential hazards and their deviations safety analysis methods are used, a brief introduction of them is provided in the next section along with the analysis method we choose for CAs.

## **2.3 Machine Learning**

### **2.3.1 Machine Learning in CAs**

The pipeline architecture of CAs is implemented through a combination of rule-based (regular expressions, template-matching), statistical (Hidden Markov Models (HMMs), conditional random fields), and ML (and DL) approaches. Due to the increased performance and accuracy of DL approaches, their variants such as convolutional neural networks (CNNs) [124], RNNs [89, 91], and LSTMs [90] are commonly used in

Table 4: Summary of safety issues in clinical CAs

CA Research Study	CA Type	Concern Detail
General Healthcare CAs [118]	General Diagnostic CAs	May harm patients by overlooking personal factors involved in a diagnosis when making a recommendation.
ML-based CAs [7]	Machine Learning CAs	Performance can suffer due to a lack of rich healthcare datasets. It is considered unsafe to use them without training on domain-specific data.
Comparative study of CAs [1, 108, 120]	CAs with unconstrained input	Can cause patient harm and should not be fully relied upon. These systems failed over 50% of the time in situations requiring medical expertise and led people toward actions that could result in harm. They should be used with clinical supervision.
Study on smartphone CAs [121]	Smartphone CAs	Provided limited or inappropriate responses to inquiries about suicidal emergencies, often defaulting to a web search.
Bibault et al. [7]	CAs in Oncology	The scarcity of clinical trials outweighs their potential benefits. An alarming lack of objective evidence exists for their relevance and efficacy.
Study on sensitive conditions [122]	CAs for sensitive health conditions	Suggested adding an option to talk to a human counsellor. The CA should link users to trained professionals if suicidal thoughts or self-harm are mentioned.
Hodgson et al. [123]	CAs using Speech Recognition for EHR	Found an increased risk of errors from speech recognition with the potential to cause patient harm.

literature. The details of the DL techniques concerning CA components are provided later in this section. These techniques are sometimes combined with statistical models to achieve a better result. The use of pre-trained models is also common where a trained model used for one application or domain can be used on a different application or domain without needing to build the model from scratch [125]. Figure 13 presents ML (DL and reinforcement learning RL) techniques used by a CA across various components in its pipeline [85].

Transformer models have become the dominant neural architecture in NLP [126]. Foundation models are large pretrained models, usually transformer based, that can be adapted to many tasks. Large language models are foundation models trained primarily on text [127, 128]. In conversational systems an LLM can handle understanding, state tracking, and response generation in a single model. The

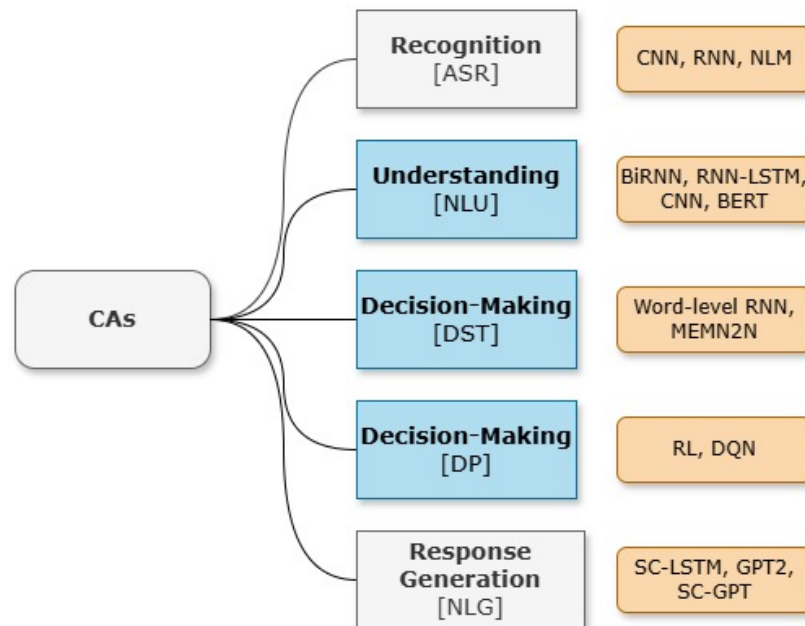


Figure 13: ML in CA architecture

studies covered in this chapter mainly use modular pipelines, and transformer components already appear inside these pipelines, for example BERT style encoders and the DIET (Dual Intent and Entity Transformer) transformer for intent and entity recognition (see section 2.3.3). This reflects the study period and the available evidence while noting the recent shift toward LLM centred approaches.

## 2.3.2 Machine Learning Challenges in CAs

### 2.3.2.1 Data

ML is data-driven which require fine-grained annotated datasets to learn from the data and provide a good prediction. However, it is often hard to obtain large-scale datasets in specific domains and healthcare is one of those domains [129]. Data inefficiency, especially in CAs for healthcare is a major challenge [85]. There are multiple ways to overcome this inefficiency such as transfer learning, generating data from real users, and using unsupervised methods. However, evaluating and comparing a safety assurance approach to state-of-art systems becomes even more challenging due to the lack of publicly available datasets or systems.

### 2.3.2.2 Uncertainty

Robustness for DL is a major challenge and CAs also suffer from it because of the use of DL for its core components. CAs are dependent on domain ontology and suffer from out-of-distribution inputs [130] on previously unseen data. CAs use different variants of DL approaches (CNNs, RNNs, and LSTMs) for recognition, understanding, and decision making. One of the biggest challenges for deep learning in safety-critical applications is uncertainty in model predictions [131]. These models suffer from data and model uncertainty while making a prediction. Data uncertainty arises from the noise in the training data and cannot be reduced by adding more training examples. Model uncertainty arises when the model has limited knowledge of parts of the input space, i.e. it has not seen similar data during training. The uncertainty arises as these models are not explicitly programmed [132] and learn from data to provide an estimated response to unseen data. This is one of the challenges in applying DL especially in healthcare which is a safety-critical domain. The robustness in the decision making of CAs is important for their performance which ultimately affects their safety. The next section discusses in detail approaches to make ML/DL models robust.

### 2.3.2.3 ML Robustness Techniques

In this section, we discuss techniques that can be used for increasing the robustness of ML components for safety-critical tasks. Model uncertainty in deep learning can arise from limited training data, which in turn affects robustness. Due to this uncertainty, a model can be confident yet wrong in its prediction. This uncertainty needs to be estimated for better decision making. For example, in a medical classification task, a model may return a result with high uncertainty. Detection of this uncertainty is important for safety and in terms of higher uncertainty a safe strategy would be to inform a clinician.

An ensemble is an approach used for the robustness of ML models. Ensemble methods combine several models to produce one optimal model for prediction. Different models are combined and average out to get a robust model [133]. The main limitation of this technique is the computational power required to train each model separately. Further, this method suffers from the interpretability problem which is a general issue in ML but combining multiple models makes it even harder to reason about its prediction [134].

Dropout [135] is a commonly used technique in ML to regularise models by turning off random neurons during training. At each step during the training, different neurons are switched off to prevent overfitting and better generalization to unseen data. In ML, the model may memorise the training data to a point where it performs poorly on unseen data. The architecture of the model with dropout is slightly different because of randomly switching off the neurons and the resultant model can be thought of as an averaging ensemble of many different models with each trained on one batch of data. This dropout technique is applied during training time and it provides a better generalization to unseen data.

Monte Carlo dropout (MC) [136], unlike the regular dropouts, is applied at run time. This gives a distribution of predictions instead of a single prediction each from one model. This distribution provides more confidence in the prediction instead of a single output without this method. It is shown that the models can be confidently wrong [136]. In the methods section, we apply this technique because it is less computationally expensive than ensembles and we use LSTMs (as they provide better performance than RNNs) on a medical intent classification task to measure model uncertainty.

### 2.3.3 Frameworks for CA Development

Conversational AI over the last few years transformed from being on the smartphones to dedicated assistants. Early commercial assistants like Apple's Siri (2011), Amazon's Alexa (2014), and Microsoft's Cortana (2014) brought conversational interfaces into the mainstream, but their underlying architectures were often rigid [86]. The development of CAs during this initial phase was largely defined by a modular, pipeline-based approach. This created a demand for frameworks for rapid development of CAs, some of them proprietary for enterprise use, others open source. Below is a brief description of several widely used CA development frameworks.

- **RASA:** The open-source framework, RASA is designed for developers who require more flexibility, customisation, and control over their data and machine learning models [137, 138]. Its developer-focused, self-hosted nature makes it ideal for enterprises with stringent data privacy requirements and complex, bespoke conversational workflows, though it comes with a steeper learning curve and greater operational overhead [137, 138].

- **Google Dialogflow:** A cloud platform from Google, Dialogflow is engineered for ease of use and rapid deployment [138]. With its intuitive graphical user interface (GUI) and integration into the Google Cloud ecosystem, it has a low barrier to entry, making it a popular choice for developers and businesses seeking to quickly launch conversational agents across multiple channels [138]. This convenience, however, comes at the cost of reduced customisability and potential vendor lock-in [138].
- **IBM Watson Assistant:** An enterprise solution from IBM, Watson Assistant builds on a long legacy of research in question-answering and natural language processing, famously demonstrated by the Jeopardy!-winning Watson computer [139, 140]. Later part of the Watsonx platform, it is tailored for large organisations, emphasizing robust security, scalability, data privacy, and high-performance NLU for business-critical applications [139, 140].
- **LLM-based Development Stacks:** Recent CA development uses large language models provided by managed platforms and supported by orchestration libraries. Orchestration commonly includes function or tool calling and retrieval-augmented generation to ground outputs in clinical sources [141]. This shift moves practice from fixed dialogue policies to model-centred workflows and increases the need for provenance, consistency and rigorous evaluation in clinical settings [141].

While several commercial platforms such as Google Dialogflow and IBM Watson Assistant offer robust, enterprise-grade solutions for building conversational agents, the selection of the RASA framework for the use cases in this thesis was a deliberate methodological choice. The primary reasons for this decision were its open-source nature, flexibility, and strong alignment with the needs of academic research and rapid prototyping. Unlike proprietary cloud platforms that operate as "black boxes," RASA's entire framework is open-source, granting researchers complete transparency and the ability to inspect, modify, and extend any part of the system [137, 138]. This customisability is essential for the type of safety analysis conducted in this thesis, where understanding the internal workings of the NLU and dialogue policies is paramount.

Furthermore, RASA's modular architecture, which decouples the NLU and dialogue management components, makes it well-suited platform for focused experimentation [137]. This allowed for isolated testing of the NLU in Use Case 1

(section 5.1) and the development of an end-to-end, text-based CA in Use Case 3 (section 5.3), discussed in Chapter 5. The ability to self-host the framework also supports control over data privacy, a critical consideration for research involving sensitive health data [138]. Notably, the main case study ('Hinata') was also implemented using RASA, which further supports the relevance of this framework for clinically oriented CAs in practice. RASA's frequent use as a baseline framework in academic research for building and testing novel dialogue systems further indicates its suitability as a credible tool for the work presented in this thesis [138, 142, 140].

As RASA is used throughout this thesis, it is essential to provide details about the architecture as they are discussed throughout Chapters 5 and 6.

### **Classic RASA Architecture**

The traditional RASA architecture, which remains a commonly used option within the framework, is built on the principle of separating the concerns of language understanding and dialogue management into two primary, decoupled components: RASA NLU and RASA Core [137].

**RASA NLU and the DIET Architecture** RASA NLU is the component responsible for interpreting user messages. It takes unstructured text as input and outputs structured data in the form of intents and entities [137, 143]. A key feature of RASA is that this NLU module can be used independently of the dialogue management component [137]. A central component of modern RASA NLU is the **DIET** architecture, introduced in a 2020 research paper [143]. DIET is a multi-task transformer architecture designed to handle intent classification and entity recognition jointly within a single model [143].

- **Multi-Task Learning:** It shares layers between the intent and entity tasks, allowing the model to learn representations that benefit both, often leading to improved performance.
- **Modular Featurization:** A key feature is its configurable design. DIET can seamlessly incorporate a variety of dense features from pre-trained word embeddings (e.g. GloVe, ConveRT, or fine-tuned BERT embeddings) and combine them with sparse features (e.g. word-level and character-level n-grams) [143]. This allows developers to configure the NLU pipeline with the

optimal set of features for their specific dataset and domain, without being locked into a single pre-trained model.

**RASA Core and Dialogue Policies** RASA Core is the dialogue management component. Its responsibility is to select the next action the assistant should take based on the entire conversation history [137]. This history is stored in a Tracker object, which maintains the dialogue state, including recognised intents, extracted entities (slots), and the sequence of past events [137].

RASA Core's primary innovation was the application of machine learning to dialogue management, moving beyond rigid, hand-coded state machines. This is achieved through a configurable stack of Dialogue Policies that work together in a priority-based hierarchy to predict the next system action [137, 144].

- **RulePolicy (Priority 1):** This policy has the highest priority and is designed to handle short, deterministic conversational patterns. It makes predictions based on rules defined in a `rules.yml` file [144]. Rules are ideal for single-turn interactions (e.g. responding to "Are you a bot?") or for controlling the logic of forms (e.g. activating or deactivating a form). By design, rules are limited to short dialogue snippets to prevent developers from creating complex and unmaintainable rule-based state machines; they are meant to augment, not replace, machine learning-based stories [137].
- **MemoizationPolicy (Priority 2):** This policy checks if the current conversation turn matches any of the example "stories" defined in the training data (`stories.yml`) exactly. If a perfect match is found, it predicts the next action from that memorized story [137]. It is effective for handling common, predictable conversation paths that have been explicitly scripted.
- **TEDPolicy (Transformer Embedding Dialogue Policy) (Priority 3):** This is the core machine learning policy. TED is a transformer-based architecture that learns to predict the next best action by generalizing from all the example stories in the training data [144]. It is invoked when no rule applies and no exact story is found by the MemoizationPolicy. TED's strength lies in its ability to handle unseen or unexpected conversational paths, allowing the assistant to gracefully manage deviations from the scripted dialogues.

This hybrid policy approach gives developers a flexible set of policies, allowing them to enforce strict, rule-based behaviour for critical interactions while leveraging machine learning to handle the rich and varied nature of human conversation.

## 2.4 Methodological Rationale: The Case Study Approach

### 2.4.1 The Suitability of Case Study Research for Socio-Technical Systems

The selection of a case study methodology is particularly appropriate for this research, as it is the preferred method for investigating complex, contemporary phenomena within their real-life context, especially when the boundaries between the phenomenon and context are not clearly evident [145]. Clinical conversational agents are not merely isolated software artefacts; they are intricate socio-technical systems where technology, human actors, and organisational workflows are deeply intertwined. A purely quantitative or laboratory-based experiment would fail to capture the nuances of how safety risks emerge from these interactions [146]. The case study approach, therefore, provides the necessary depth to answer the "how" and "why" questions central to this thesis and is well-established for evaluating complex health technologies [147].

### 2.4.2 Case Study Designs Relevant to this Thesis

It is crucial to recognise that "case study" is not a monolithic method. The specific design must be tailored to the research objective. Yin [145] discusses several uses of case study research, including exploratory, descriptive, explanatory, and evaluative applications. In this thesis, the most relevant forms are exploratory and evaluative.

An **exploratory case study** is often used as a "pilot" when the primary goal is to develop initial insights, generate new research questions, or formulate propositions for later, more comprehensive study. It typically answers "what" questions (e.g. "What are the common failure modes?"). This approach is invaluable in the early stages of research to define the scope and foundations of an investigation.

An **evaluative case study**, by contrast, is used to assess a method, intervention, programme, or set of artefacts in its real-world context. It may still address “how” or “why” questions, but the emphasis is on evaluating practical relevance, usefulness, and credibility rather than proving causal links. This design requires a structured approach with clearly defined propositions from the outset.

### 2.4.3 Selection of Yin’s Framework for Rigorous Inquiry

Among the various established approaches, the structured and rigorous methodology proposed by Robert K. Yin [145] was selected as the most appropriate framework for this research. While other approaches, such as Robert Stake’s constructivist method for providing rich, interpretive understanding [148], or Kathleen Eisenhardt’s process for building new theory from multiple cases [149], are powerful in their own right, their primary goals do not align with the aims of this thesis. The objective here is not to interpret a unique situation or to generate new theory, but to systematically apply and evaluate a safety methodology. Given the safety-critical nature of the subject matter, Yin’s methodology, with its clear procedural guidance on case study design, propositions, multiple sources of evidence, construct validity, and reliability, provides the rigour needed to produce auditable analysis.

This framing underpins the design of the case study in Chapter 6, which applies an evaluative case study design to assess the proposed safety assurance methodology in a real clinical deployment.

## 2.5 Summary

### Gaps in the Literature and Thesis Contribution

While the literature highlights many individual safety challenges and established safety engineering principles, there is a clear gap in providing a **systematic and holistic safety assurance methodology** specifically tailored for clinical CAs. Existing research points to failures but lacks a comprehensive, structured **failure mode taxonomy** that systematically classifies *how* and *why* these complex socio-technical systems may fail from the user interaction, the environment, or the underlying ML models.

Consequently, there is a need to demonstrate how such a taxonomy can be used to drive a rigorous safety process: from hazard analysis and risk assessment to the derivation of verifiable safety requirements and the construction of a structured, evidence-based **safety case**. This thesis aims to fill these gaps by developing the aforementioned taxonomy and presenting a complete safety assurance methodology, evaluated through a series of use cases, including a real-world, clinically deployed CA.

# Chapter 3

## Safety Assurance Methodology

### 3.1 Introduction

This chapter sets out the methodology used to assure the safety of clinical conversational agents (CAs). The responsible use of clinical CAs demands rigorous and systematic safety assurance, considering the complex interaction between system safety, machine learning (ML), human-computer interaction (HCI), and healthcare. While chapters 1 and 2 have established the necessity and context for safety assurance in clinical CAs highlighting the critical gaps within existing frameworks, regulations, and safety challenges the present chapter describes the structured approach adopted in this thesis. The aim is to link plausible failure modes to hazards and risks, derive safety requirements, and develop an evidence-based safety argument.

The methodology is organised into five interconnected phases and primarily targets ML-based CAs that employ a pipeline architecture. Each phase produces concrete artefacts and traceability links that are used in subsequent chapters, providing an end-to-end framework for assuring the safety of clinical CAs. Where risk ratings are reported later, they are initial (pre-control) and are used to drive requirement derivation; formal organisational risk acceptance is outside the scope of this thesis.

### 3.2 Overview of the Methodological Framework

The safety assurance methodology presented in this thesis is an integrated, multi-phase process designed to: (i) identify failure modes and resulting hazards; (ii) analyse and prioritise risks; (iii) derive and implement mitigations through safety requirements; and (iv) assemble a structured argument that a CA is acceptably safe

for its intended use. It combines established safety-engineering techniques with empirical evaluation. The method is organised into the following five phases:

1. **Phase 1: Foundational Taxonomy Development:** The process begins with the development of a structured failure mode taxonomy, grounded in the **SHEL model** [150], comprising **Software, Hardware, Environment, and Liveware** (the human user) and a review of existing literature. This foundational step (presented in chapter 4) provides a vocabulary for classifying potential failures across the socio-technical system, enabling a systematic analysis.
2. **Phase 2: Systematic Hazard and Risk Analysis:** This phase uses the taxonomy to apply guideword-driven techniques, such as Hazard and Operability study for Unified Modelling Language (HAZOP-UML) and System-Hazard Analysis and Resolution in Design (SHARD), to uncover potential hazards in real-world use cases (chapter 5). Each identified hazard is given an *initial (pre-control)* severity-likelihood rating using a clinical risk matrix to prioritise mitigation.
3. **Phase 3: Deriving Safety Requirements:** Based on the prioritised hazards, this phase derives a hierarchical set of safety requirements that act as risk controls with clear traceability from top-level clinical hazards to verifiable controls for the software, the ML model, and the data used to train and test it.
4. **Phase 4: Constructing the Safety Case with GSN:** Evidence from the preceding phases is synthesised into a structured safety argument using Goal Structuring Notation (GSN). This phase makes all claims, assumptions, and supporting evidence explicit.
5. **Phase 5: Validation and Evaluation:** As an integral phase of the methodology, the process and its outputs are validated through a hybrid strategy that combines quantitative evaluation of ML robustness and uncertainty (thesis contribution 2 in chapter 5 see section 5.2) with qualitative stakeholder feedback, confirming the realism and practical value of the framework (chapter 6), while formal residual-risk acceptance remains out of scope for this thesis.

### 3.3 Phase 1: Foundational Taxonomy Development

This phase addresses **RQ1**: *What are the different failure modes in clinical CAs that can lead to clinical hazards?* It involves developing a structured failure-mode taxonomy for ML-based, pipeline CAs, which is then reused in subsequent hazard and risk analyses.

The design of the taxonomy is grounded in established theory to ensure its robustness. Its high-level structure is inspired by the SHEL model, a proven human factors framework that considers the interactions between software, hardware, environment, and liveware (the human user). This system-theoretic view ensures the analysis is holistic, capturing failures that arise not just from technical defects but from mismatches at the human-computer and environment-computer interfaces. The taxonomy hierarchically decomposes components (ASR, NLU, DM, NLG) into observable failure modes and distinguishes ML *model* faults from *data* faults (e.g. bias, drift, sparsity). The result is a clear, consistent vocabulary that informs the hazard analysis. The full development and justification for this taxonomy is the subject of chapter 4.

### 3.4 Phase 2: Systematic Hazard and Risk Analysis

This phase determines the clinical consequences of the identified failures through a two-stage process of hazard identification and risk assessment.

First, established safety analysis techniques are applied to the CA's architecture and operational flows. This thesis employs SHARD and HAZOP-UML to analyse deviations from intended behaviour. SHARD examines deviations at the level of information and data flows, while HAZOP-UML is applied to the use case and its corresponding UML sequence diagram to analyse the messages and parameters exchanged between components. Both techniques trace deviations to potential clinical hazards, yielding a structured set of hazards that feeds the subsequent risk assessment.

Second, each identified hazard is formally assessed. In line with clinical standards such as the NHS DCB0160, this involves using a clinical risk matrix (e.g. the  $5 \times 5$  NPSA matrix) to assign scores for **Severity** and **Likelihood**. This process results in a prioritised hazard log where causes and recommended controls are recorded for

each entry. This log serves as the primary input to Phase 3, as demonstrated in the case study in chapter 6.

### 3.5 Phase 3: Deriving Safety Requirements

The prioritised hazard log from Phase 2 serves as the direct input for this phase, adhering to the core principle of safety engineering that safety requirements must be derived from, and be traceable to, the hazards they are intended to control. A multi-layered requirements framework is used to systematically decompose high-level clinical safety objectives into concrete, verifiable requirements. For an ML-based CA, this results in a hierarchy of requirements:

- **System-level requirements:** These define high-level functional controls designed to act as safety barriers (e.g. “The system shall confirm its understanding of a critical symptom with the user before proceeding.”).
- **ML model requirements:** These specify quantifiable performance and robustness targets for the model (e.g. “The NLU model shall classify critical intents with a recall of  $\geq 90\%$ .”).
- **ML data requirements:** These establish criteria for the training and validation data, covering aspects such as relevance, completeness, balance, and accuracy.

This structured derivation ensures that every safety requirement has a clear justification and that the assurance of the ML component is inseparable from the assurance of the data upon which it relies.

### 3.6 Phase 4: Constructing the Safety Case with GSN

This phase structures the safety argument using GSN, producing the safety case as the primary artefact. The safety case synthesises all preceding activities into a coherent, transparent argument that the system is acceptably safe for its intended use. Following established patterns for assuring ML systems, the safety case is separated into two parts:

1. A primary **risk-based argument**, which makes explicit claims that all identified clinical hazards have been controlled to an acceptable level.
2. A set of supporting **confidence arguments**, known as Assurance Claim Points (ACPs), which justify the rigour and trustworthiness of the processes used to generate the safety evidence [151].

### 3.7 Phase 5: Technical Validation and Stakeholder Evaluation

The final phase focuses on the empirical validation of the assurance process and its artefacts. This methodology employs a **hybrid evaluation strategy** that combines quantitative and qualitative methods to ensure a comprehensive assessment:

- **Quantitative Validation:** This involves targeted experiments to assess the performance and robustness of the ML components. This is centrally demonstrated in use case 2 (see section 5.2), which constitutes contribution 2 of this thesis and directly addresses **RQ2b**. As demonstrated there, this includes using techniques like **Monte Carlo (MC) Dropout** to quantify model uncertainty and evaluate the model's ability to reliably detect out-of-distribution (OOD) inputs, a critical safety behaviour.
- **Qualitative Evaluation:** This assesses the real-world relevance and completeness of the methodology using an **evaluative case study framework**, grounded in the principles articulated by Yin [145]. It involves conducting semi-structured interviews with key technical and clinical stakeholders and performing a thematic analysis of their feedback on the safety artefacts produced.

This dual approach ensures that the safety claims are not only supported by technical performance data but are also judged to be complete and realistic by domain experts.

### 3.8 Connecting the Thesis: From Theory to Practice

The thesis builds from foundations (chapters 1-2), through methodology (chapter 3), foundational taxonomy (chapter 4), and applied use cases (chapter 5), to the full

case study with requirements and a safety case (chapter 6). chapter 7 summarises findings and outlines future work. This chapter therefore acts as the bridge between the literature and its practical application in later chapters.

### **3.9 Summary**

In summary, this chapter has laid out a clear, structured, five-phase methodological framework for the safety assurance of clinical conversational agents. It demonstrates how to integrate rigorous theoretical analysis, established safety engineering methodologies, and empirical validation to move from an initial understanding of system failures to a final, defensible safety case. This framework positions the thesis within both theoretical safety engineering and practical clinical domains, bridging the identified research gaps and setting the foundation for the work presented in the subsequent chapters. chapter 4 details the first output of this methodology: the systematic failure mode taxonomy that underpins all subsequent safety analyses.

# Chapter 4

## A Failure Mode Taxonomy for Clinical Conversational Agents (CAs)

### 4.1 Introduction

The increasing use of artificial intelligence (AI) has led the integration of CAs into numerous sectors, with healthcare emerging as a domain of significant potential and serious consequence [1, 2]. These intelligent systems, which interact with users via natural language, are increasingly being deployed as task-oriented CAs (often referred to as dialogue systems or virtual assistants) to perform specific functions such as symptom checking [2], medication adherence monitoring [4], chronic disease management [3], and patient education [2]. The promise of CAs lies in their ability to provide low-cost, scalable, and continuously available support, potentially alleviating burdens on healthcare systems and improving patient access to information [152].

However, the deployment of CAs in such a safety-critical domain is not without considerable risk. Unlike their counterparts in e-commerce or entertainment, the failure of a clinical CA can have direct and severe implications for patient safety [5]. An error in transcribing a user's speech, a misunderstanding of their intent, or an incorrect decision based on flawed logic can lead to hazardous situations, including misdiagnosis, improper medication advice, or a failure to escalate an emergency, potentially resulting in significant harm [5]. As these systems are increasingly powered by complex and often opaque machine learning (ML) models, understanding their failure characteristics becomes a significant challenge for safety engineers, developers, and regulators [153, 8].

To address this challenge, a systematic and structured approach is required to identify, classify, and analyse the potential ways in which these complex systems can fail. This chapter seeks to answer the first research question of this thesis, **RQ1:**

***What are the different failure modes in clinical CAs that can lead to clinical hazards?*** The primary contribution of this chapter is the development and detailed presentation of a failure mode taxonomy for clinical CAs. This taxonomy is designed to be a foundational artefact for the safety assurance of such systems. It provides a systematic classification of failures arising from the system's software, its hardware, its interaction with the user, and its operational environment. By establishing a clear and consistent vocabulary and structure, the taxonomy aims to improve the way in which safety is considered throughout the entire lifecycle of a clinical CA, from initial design and development to verification, validation, and post-market surveillance.

This chapter is structured as follows:

1. Section 4.2 begins by establishing the theoretical foundations of taxonomies in system analysis, outlining the principles that ensure a robust and rigorous classification scheme.
2. Section 4.3 provides a safety engineering perspective, establishing precise definitions for fault, failure, and failure mode, and contextualising the taxonomy within established safety analysis methodologies.
3. Section 4.4 details the system-theoretic approach that underpins the taxonomy's design, justifying the choice of the SHEL model and hierarchical classification methods against alternatives.
4. Section 4.5 presents the failure mode taxonomy in its entirety, offering a deep and granular analysis of the failure modes and their causes, with a particular focus on the ML-related root causes of software failures.
5. Section 4.6 discusses how the taxonomy has evolved from the use cases discussed in chapters 5 and 6.
6. Section 4.7 connects the abstract taxonomy to tangible risks by analysing the patient safety implications of the identified failures, supported by a table of illustrative scenarios.
7. Section 4.8 discusses the scope and application of the taxonomy and discusses its limitations.

## 4.2 Theoretical Foundations of Taxonomies in System Analysis

The systematic study and management of complex systems, particularly those operating in safety-critical domains, necessitates a structured and principled approach to classification. In the scientific and engineering literature, various terms such as classification, typology, and taxonomy are used to describe the grouping of objects of interest, though they carry distinct meanings. A classification is the most general term, referring to the grouping of similar entities [154]. A typology, often used in the social sciences, refers to a set of categories derived conceptually and is typically multidimensional [154]. This work adopts the concept of a taxonomy, which is defined as a methodology for grouping objects both conceptually and empirically into a hierarchical format [155]. As defined by Blackwelder [156], a taxonomy serves to divide a body of knowledge into a structured, hierarchical classification. This approach is a standard practice in engineering disciplines, as exemplified by the IEEE standard taxonomy for software engineering, which also employs a hierarchical structure to organise complex information.

The fundamental purpose of developing such a taxonomy is to manage complexity, provide a common vocabulary for researchers and practitioners, and ensure that the analysis of a system is both structured and systematic. The construction of the failure mode taxonomy presented in this chapter is guided by the formal principles of hierarchical classification, most notably articulated by O'Halloran et al. [157]. This method provides a framework for ensuring the resulting taxonomy is logical, structured, and useful for subsequent analysis. The key principles are as follows:

- **Well-Defined Units:** Each unit within the taxonomy, from the highest-level category to the most granular failure mode, must be clearly defined by its unique qualities. This ensures that there is no confusion about what each element represents.
- **Similarity and Difference:** The grouping of units into classes and subclasses is based on shared properties (similarity) and distinguishing characteristics (difference). This principle ensures that the hierarchy is logical and that elements at the same level are conceptually related.

- **Progressive Specificity:** The taxonomy progresses from general categories to more specific ones. In the O'Halloran's model, this is represented by moving from a 'Superclass' (the highest level of abstraction) down through 'Classes' and into 'Subclasses' [157]. This structure allows for analysis at multiple levels of granularity.
- **Comprehensiveness and Mutual Exclusivity:** A well-constructed taxonomy should fully cover the body of knowledge it purports to classify, meaning no known object of interest is left out. Furthermore, each unit within a given class in the hierarchy should be independent and mutually exclusive of other units at the same level [155]. This prevents overlap and ensures that each failure can be classified into one, and only one, category at each level of the hierarchy.

By adhering to these principles, the taxonomy developed in this work moves beyond a simple list of potential problems. It becomes a structured, analytical tool that provides a foundational understanding of the failure landscape of clinical CAs, enabling systematic investigation, comparison, and the development of targeted risk mitigation strategies.

## 4.3 A Safety Engineering Perspective on System Failure

To analyse the safety of clinical CAs with rigour, it is essential to move beyond colloquial language and adopt the precise terminology of safety engineering. The terms 'fault', 'error', 'failure', and 'hazard' have specific meanings that define a clear causal chain from a latent system defect to potential patient harm. This section establishes these formal definitions, contextualises them within established safety analysis methods, and clarifies why the term 'failure mode' is the most appropriate descriptor for the elements of the presented taxonomy.

### 4.3.1 Defining Fault, Failure, and Failure Mode

In safety engineering [158], the distinction between a fault and a failure is fundamental. A *fault* is defined as a defect, an abnormal condition, or a latent error within a system component that may cause a failure. For example, a fault could be a

bug in the software code, incomplete or biased training data for a ML model, or a degraded microphone. The fault itself may not be immediately observable and may only manifest under specific conditions.

A *failure* is the consequence of a fault. It is defined as the termination of the ability of an item to perform its required function [158]. This is the observable event where the system deviates from its specified behaviour. For instance, if a CA is required to accurately transcribe a user's speech, the event of it producing an incorrect transcription is a failure. This failure is caused by one or more faults, such as faulty microphone (a hardware fault) and an inadequately trained acoustic model (a software fault).

A *failure mode* describes the specific manner in which a failure is observed [158]. As defined in the international standard IEC 60812, which provides the procedure for Failure Mode and Effects Analysis (FMEA), a failure mode is "a specific way in which a component, system, or process can fail to meet its intended function" [159]. It is not the cause of the failure (the fault) nor the consequence of the failure (the effect), but rather a description of how the system fails [158]. For example, for a CA's recognition component, "inaccurate transcription" is a failure mode. For its decision-making component, "incorrect system action" is a failure mode. This taxonomy is therefore correctly titled a failure mode taxonomy, as its purpose is to classify the observable ways in which a clinical CA can fail to perform its required functions.

### 4.3.2 The Causal Chain from Fault to Harm

The relationship between these terms forms a clear causal chain that is central to safety analysis [160]. This progression, often visualised using a bow-tie diagram as shown in Figure 14, illustrates how a latent fault can escalate into actual harm if safety barriers are absent or ineffective [161].

The causal sequence is as follows:

1. **Fault:** A latent defect or adverse condition exists (e.g. skewed training data for a diagnostic algorithm).
2. **Failure (Failure Mode):** The fault leads to a failure, where the system does not perform its function as intended (e.g. the CA exhibits the failure mode of 'biased decision making').

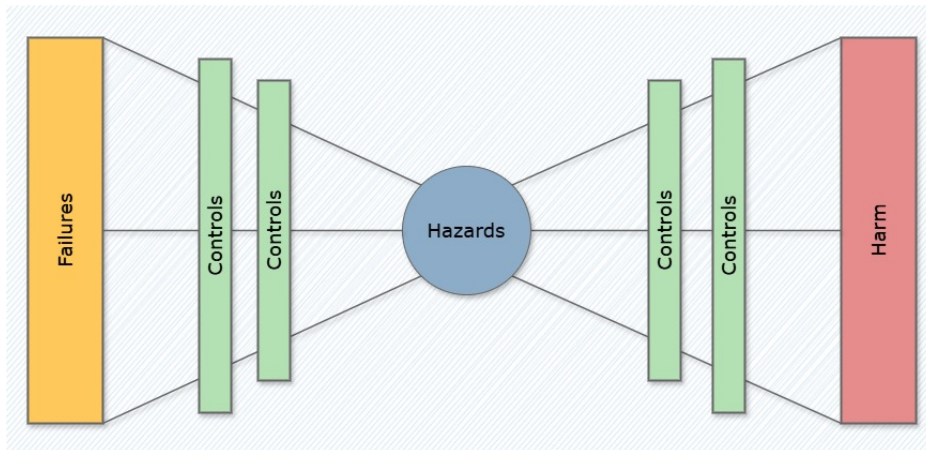


Figure 14: Causal chain from fault to harm

3. **Hazard:** The failure creates a **hazard**, which is defined in a clinical context as "a potential source of harm to a patient" [162]. In this example, the hazard is the presentation of an incorrect diagnosis to the user.
4. **Harm:** If the user acts on the hazardous information without any mitigating controls (e.g. consulting a clinician), it can lead to **harm**, such as allergic reaction or trauma.

This causal chain clarifies that the goal of safety assurance is not merely to prevent failures, but to put controls and barriers that either prevent faults from causing failures (preventive controls) or prevent failures from leading to harm (mitigating controls).

subsectionContextualising the Taxonomy within Safety Analysis This failure mode taxonomy is not an end in itself, but rather a foundational tool that enables and enhances established safety analysis techniques. Its primary applications demonstrated in this thesis are in methodologies like Hazard and Operability Studies (HAZOP), specifically HAZOP-UML and SHARD.

**Hazard and Operability Study (HAZOP)** is a structured and systematic, top-down examination of a planned or existing process or operation used to identify and evaluate problems that may represent risks to personnel or equipment [163]. HAZOP works by using guide words (e.g. "No," "More," "Less") to explore deviations from the design intent. While FMEA is bottom-up (from component failure to system effect), HAZOP is top-down (from system deviation to component cause) [164]. The

taxonomy supports HAZOP by providing a structured map of the underlying system components and their potential failure modes, which helps in tracing a system-level hazard back to its root causes within the CA architecture [164].

## 4.4 A System-Theoretic Approach to Taxonomy Design

The development of a structured taxonomy requires a deliberate and principled design methodology. A simple, unstructured list of potential errors would be insufficient for the systematic analysis required in a safety-critical domain. The taxonomy presented in this work is therefore built upon a dual foundation: a system-theoretic model to define its high-level structure and a formal hierarchical method to organise its contents. This section justifies these choices by comparing them with alternative approaches.

More specifically, the taxonomy connects the generic guidewords of these methods to the known, complex failure modes of CAs. For example, when a HAZOP/SHARD analysis applies the Value guideword to an NLU's output flow and identifies a deviation like "Incorrect intent identified", this taxonomy (Table 5) classifies this as the 'Misunderstanding (incorrect intent / slot)' failure mode. Furthermore, it immediately points the analyst toward specific, likely ML-related faults (Table 6), such as 'Class Imbalance', 'Annotation Errors', or 'Out-of-Distribution Inputs'.

This prevents the analysis from remaining too abstract and focuses it on the known vulnerabilities of ML-based CAs. This practical application of the taxonomy as an interpretive framework for hazard analysis is demonstrated explicitly in the HAZOP-UML analysis in section 5.3.3 and the SHARD analysis in section 6.4.1.

### 4.4.1 Methodological Inspiration: The SHELL Model

The conceptual architecture of the taxonomy is inspired by the SHELL model, a framework developed by Edwards and widely used in human factors and aviation safety [150]. The **SHELL model** posits that any human-machine system can be understood as an interaction between four core components: **Software** (the rules, procedures, and logic), **Hardware** (the physical equipment), **Environment** (the context

in which the system operates), and **Liveware** (the human element), as shown in Figure 15. The central premise of the model is that system failures and accidents typically arise not from the failure of a single component in isolation, but from a mismatch or poor interaction at the interfaces between these components [165].

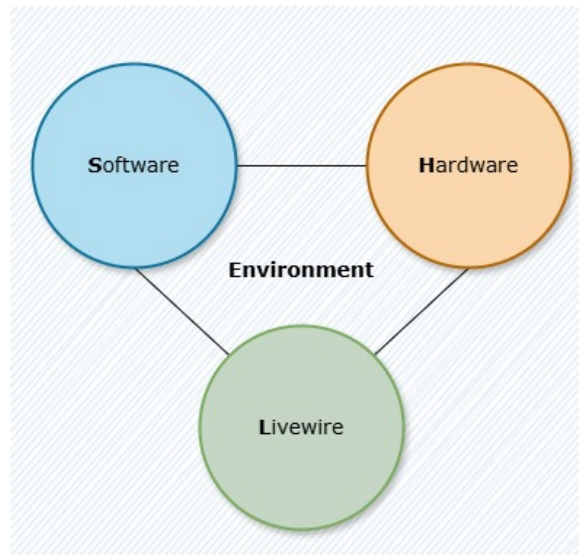


Figure 15: The SHEL Model [150]

This model is well-suited for analysing a clinical CA system. The CA itself comprises the **Software** (the AI algorithms, dialogue logic, knowledge bases) and the **Hardware** (the smartphone or smart speaker, including its microphone, speaker, and battery). The CA operates within an **Environment** (a quiet home, a noisy clinic, a moving vehicle) and interacts with a **Liveware** component—the user or patient.

The choice of the SHEL model is deliberate. Alternative system modelling approaches, such as **Systems Dynamics** or **Agent-Based Modelling**, offer different perspectives [166, 167]. Systems Dynamics focuses on feedback loops and population-level behaviours, which would be useful for studying the large-scale adoption of CAs but is less suited for analysing individual human-machine interaction failures [166]. Agent-Based Modelling offers a more intuitive way to model the behaviour of individuals but can become overly complex when the goal is to create a structured, high-level classification of failure sources rather than simulating specific scenarios [167]. The SHEL model provides an appropriate level of abstraction for this taxonomy, creating a holistic, system-level perspective that forces consideration of the entire socio-technical system. It explicitly acknowledges that failures can be precipitated by factors external to the core technology, such as a user with a

strong accent (a Liveware-Software interface issue) or significant background noise (an Environment-Hardware interface issue). While some critiques of the SHEL model note its tendency to under-emphasize complex cognitive and organizational factors [168], it provides a reasonable foundation for structuring the analysis of direct human-machine interactions, which is the primary focus of this research.

#### 4.4.2 A Hierarchical Design for a Failure Mode Taxonomy

While the SHEL model provides the high-level conceptual structure, a formal method is needed to organise the specific failure modes within each category. Other approaches to classifying CA errors exist, such as those based on **Clark's taxonomy of miscommunication** (which uses levels like channel, signal, and intention) [169] or **Grice's maxims** (quality, quantity, relevance, manner) [170]. While useful for linguistic analysis, these frameworks are less suited for a safety engineering context, which requires a clear mapping of failures to system components.

For this reason, the taxonomy employs a hierarchical approach based on the work of **O'Halloran et al.** [157], a method used in the classification of system failures. This approach organises the taxonomy into a tree-like structure with multiple levels of abstraction, typically designated as 'Superclass', 'Class', and 'Subclass'. This ensures that the classification is systematic, exhaustive, and that the relationships between different failure modes are clearly defined.

The application of this hierarchical design to the CA system is realized through a process of functional decomposition. The top-level classes are derived from the SHEL model: Software, Hardware, Environment, and User. The 'Software' class is then further sub-classified based on the modular architecture of a typical task-oriented CA. This decomposition, inspired by the work of Ward [171], breaks down the software into its core functional components:

- **Recognition:** The Automatic Speech Recognition (ASR) module.
- **Understanding:** The Natural Language Understanding (NLU) module.
- **Decision-Making:** The Dialogue Manager (DM) or policy module.
- **Response Generation:** The Natural Language Generation (NLG) module.

This functional decomposition provides a logical way to organise software-related failure modes, as each can be traced back to a specific processing stage in the

conversational pipeline. Finally, to identify the underlying causes for each failure mode, the approach draws inspiration from the work of Constantinides and Rudnicky, who used a method analogous to a fishbone (Ishikawa) diagram to trace interaction failures back to their root causes, such as user accent or background noise [172].

By combining the holistic system view of the SHEL model with the structured, multi-level organization of O'Halloran's hierarchical method and the functional decomposition of the CA pipeline, this taxonomy is designed to be useful for detailed safety analysis.

## **4.5 A Failure Mode Taxonomy for Clinical CAs**

This section presents the structured failure mode taxonomy for clinical CAs, which constitutes the first contribution of this research. The taxonomy is structured hierarchically, beginning with four top-level categories derived from the SHEL model which includes software, hardware, environment, and live-ware (the user). It then decomposes these categories into specific components, sub-components, and their associated failure modes. For each failure mode, potential causal factors are identified. A key insight framing this taxonomy is the distinction between the system (software and hardware) and its context (user and the environment). Following a system-centric safety perspective, failures are properties of the system itself; the User and Environment are treated as sources of causal factors or faults that can trigger these system failures.

The failure mode taxonomy for CAs is detailed in Table 5. The ML-related causes are discussed separately in Table 6 as a supporting fault refinement, rather than as a separate taxonomy category. A copy of the taxonomy is provided in Appendix A to support practical use by researchers, developers, and safety analysts.

### **4.5.1 Software Failure Modes and Causal Factors**

The software component of a clinical CA can be functionally decomposed into a pipeline of modules, each responsible for a stage of processing the user's request. A failure in any one of these modules may propagate downstream, leading to an erroneous system response [1].

Table 5: Failure mode taxonomy for CAs

Category	Component	Failure mode	Typical causes
Software	Recognition	Inaccurate transcription	<ul style="list-style-type: none"> <li>• Non-native accent; very fast / slow speech</li> <li>• Background noise</li> <li>• Acoustic model trained on out-of-domain speech [173, 174]</li> </ul>
		Word-substitution (similar words)	<ul style="list-style-type: none"> <li>• Disfluencies or repetitions</li> <li>• Weak language-model priors for phonetic neighbours [175]</li> </ul>
	Understanding	Misunderstanding (incorrect intent / slot)	<ul style="list-style-type: none"> <li>• Class imbalance; incomplete training set [176, 177]</li> <li>• Annotation errors</li> <li>• Model over-fitting / high uncertainty</li> </ul>
		Non-understanding intent	<ul style="list-style-type: none"> <li>• Out-of-domain utterance; unknown words</li> <li>• Upstream ASR error</li> <li>• No OOD detector deployed [178]</li> </ul>
	Decision Making	Overlooked conversation context	<ul style="list-style-type: none"> <li>• Dialogue-state window too short [179]</li> <li>• Error propagation from ASR / NLU</li> </ul>

(continued from previous page)

Category	Component	Failure mode	Typical causes
		Incorrect system action	<ul style="list-style-type: none"> <li>• Sparse training examples for policy [180]</li> <li>• Error propagation from ASR / NLU</li> </ul>
	<b>Response Generation</b>	Incorrect/ambiguous semantics DA	<ul style="list-style-type: none"> <li>• Faulty mapping from DST to template</li> </ul>
		Unnatural / verbose utterance	<ul style="list-style-type: none"> <li>• Templates not reflective of system design</li> <li>• Repetition [1]</li> </ul>
	<b>UI / UX</b>	Unconstrained user input	<ul style="list-style-type: none"> <li>• Design choice; no guided prompts [5]</li> </ul>
		Poor conversational design	<ul style="list-style-type: none"> <li>• Repetitive “I don’t understand” loops</li> <li>• Confusing or ambiguous prompts</li> </ul>
<b>Hardware</b>	<b>Microphone</b>	Low-quality audio	<ul style="list-style-type: none"> <li>• Damaged or low-spec sensor [181]</li> </ul>
		Muted / faulty microphone	<ul style="list-style-type: none"> <li>• Hardware fault; user muted unintentionally</li> </ul>
	<b>Speaker</b>	Echo / reverberation	<ul style="list-style-type: none"> <li>• Room acoustics; mic-speaker feedback loop [182]</li> </ul>
	<b>Battery / Power</b>	Sudden shutdown	<ul style="list-style-type: none"> <li>• Battery depletion during active session [183]</li> </ul>

(continued from previous page)

Category	Component	Failure mode	Typical causes
	<b>Processor/Memory</b>	Intermittent hardware faults	• Transient errors, silent data corruption, hardware traps • Fail-slow behavior [184, 185]
		Resource limitation	• Insufficient CPU/memory (esp. embedded devices) [186]
	<b>Network Connectivity</b>	Network disruption	• Connection loss, high latency, packet loss, congestion [185–188]
		Network fail-slow behaviour	• Gradual performance degradation of network components [185]
<b>User</b>	<b>Speech characteristics</b>	Inaccurate transcription	• Accent; extreme speaking rate
	<b>Interaction behaviour</b>	Misunderstanding or non-understanding	• Out-of-domain request; void input; mental-model mismatch [2]; unsupervised use
	<b>Human-in-the-loop reviewer/clinical supervisor</b>	Human error	• Cognitive overload, time pressure, inadequate training, incorrect data entry [189, 190]
	<b>Ethical concerns</b>	Incomplete information capture	• Privacy-driven withholding of information; incomplete data [191]; downstream mis-triage [4]

(continued from previous page)

<b>Category</b>	<b>Component</b>	<b>Failure mode</b>	<b>Typical causes</b>
<b>Environment</b>	<b>Acoustic environment</b>	Inaccurate transcription	<ul style="list-style-type: none"><li>• Background noise (traffic, music), corrupts the speech signal [192]; ASR errors</li></ul>
		Inaccurate transcription	<ul style="list-style-type: none"><li>• Echo/reverberation; large empty room; reflective surfaces degrading signal quality [182]</li></ul>

#### 4.5.1.1 Recognition (ASR)

The ASR module is the system's "ear," responsible for converting the user's spoken utterance into machine-readable text. Failures at this stage are fundamental, as any error in the transcribed text will likely lead to errors in understanding and decision-making.

**Inaccurate transcription** This is the primary failure mode of the ASR system, where the output text does not match the words spoken by the user. The causes are multifaceted. They can stem from user characteristics, such as a non-native accent, dialect, or speech impediment for which the ASR's acoustic model was not adequately trained [174]. They can also arise from environmental factors, like high levels of background noise that mask the speech signal [192]. Critically, these factors often interact; a non-native accent that is perfectly intelligible in a quiet room may become indecipherable to an ASR system in a noisy environment, highlighting the need for testing under realistic, combined-stress conditions [181].

**Word-substitution (similar words)** This is a specific type of transcription error where the ASR substitutes a word for an acoustically similar one (e.g. transcribing "Benazepril" as "Benadryl") [5]. This is particularly dangerous in a clinical context. This failure mode is often caused by user disfluencies (e.g. pauses like "um" or "uh"), repetitions, or short utterance duration, which can confuse the language model used by the ASR to disambiguate similar-sounding phonemes [175].

#### 4.5.1.2 Understanding (NLU)

The NLU module is responsible for extracting semantic meaning from the transcribed text. In a task-oriented CA, this typically involves identifying the user's intent (what they want to do) and extracting slots or entities (the key pieces of information needed to fulfil the intent) [75].

**Misunderstanding (incorrect intent / slot)** This is a more subtle failure where the system incorrectly classifies the user's intent or extracts the wrong entities, leading it to take an action that is plausible but incorrect [155]. For example, when a user says, "My husband is getting unconscious. What should I do?", the CA might misunderstand the intent as a request for contact information and reply, "What is

your husband's name?" [5]. This failure is often caused by errors propagated from ASR, incorrect slot-tagging by the NLU model, or insufficient and imbalanced training data for the intent classification model [174].

**Non-understanding intent:** This failure mode occurs when the system cannot map the user's input to any known intent it is designed to handle [193]. It often results in generic error messages like "I'm sorry, I don't understand" [5]. The primary causes are user inputs that are out-of-domain (e.g. asking a symptom checker for financial advice) or contain out-of-vocabulary words (e.g. acronyms or jargon unknown to the system).

#### 4.5.1.3 Dialogue Manager (DM) & Dialogue Policy (DP)

The Dialogue Manager (DM) is the "brain" of the CA, responsible for tracking the state of the conversation (Dialogue State Tracking, DST) and deciding on the next system action (Dialogue Policy, DP) [194].

**Overlooked conversation context:** In multi-turn dialogues, context is critical. This failure mode occurs when the system fails to consider previous turns in the conversation when making a decision. For example, if a patient states they are taking Lisinopril and then asks for a decongestant, a CA that overlooks the context might recommend Sudafed, failing to recognise the potential for a dangerous drug-drug interaction [5]. This failure is caused by errors in the DST, which are often a result of propagated errors from ASR and NLU, or limitations in the model's ability to maintain long-term dependencies [179].

**Incorrect system action:** This failure means the Dialogue Manager (DM) chooses a wrong or unsafe action. Unlike NLU failures (which are about misunderstanding the user's input), this failure is about the system making the wrong choice. This can be caused by flaws in the Dialogue Policy (DP), errors in tracking the conversation, not enough training data for that specific situation, or because of mistakes passed on from the NLU or ASR components.

#### 4.5.1.4 Response Generation (NLG)

The NLG module converts the system's chosen action (a structured representation) into a natural language response for the user.

**Incorrect/ambiguous DA semantics** This failure occurs when the NLG module incorrectly translates the abstract dialogue act from the DM into a response. This is typically a result of errors propagated from the decision-making stage or flaws in the mapping logic.

**Unnatural / verbose utterance** The generated response is grammatically correct but sounds robotic, is overly verbose, or is stylistically inappropriate for the context. This is often a limitation of template-based NLG systems or a characteristic failure of generative models that overuse certain phrases [1].

#### 4.5.1.5 User Experience (UX) and Interface (UI)

Failures in this category are not bugs in the traditional sense but are design choices that create pathways to unsafe interactions.

**Unconstrained user input:** This UI design choice allows users to enter free text in situations where their query could be safety-critical [1]. For example, allowing a user to freely describe suicidal thoughts to a therapy bot not equipped to handle such crises can lead to catastrophic outcomes, such as the bot affirming the user's harmful intentions [5]. A safer design would use constrained inputs (e.g. buttons, menus) or explicit mechanisms to detect and escalate such inputs.

**Poor conversational design:** This UX failure relates to the design of the conversation flow itself. It includes having ineffective error recovery strategies (e.g. repeatedly saying "I don't understand" without offering guidance) [195], providing confusing or lengthy system prompts that increase user cognitive load [196], or lacking appropriate empathy in sensitive conversations [197].

Table 6: ML model and data faults contributing to CA software failures

<b>Fault Category</b>	<b>Fault Type</b>	<b>Potential Causes</b>
<b>Data-Induced</b>	Insufficient / Unrepresentative Data	Small or narrow training set; missing accents, dialects, noise conditions, medical terms [173, 174].
	Class Imbalance	Common intents over-represented compared with vs. rare/critical ones; model biased toward majority class [177, 198].
	Out-of-Distribution Inputs	OOD queries not detected or forced into in-domain classes, leading to unpredictable or incorrect labels [178, 199].
	Annotation / Labelling Errors	Mislabelled intents or slots; inconsistent transcripts introduce noise into training [200].
<b>Model-Induced</b>	Overfitting	Model memorises training noise and fails on new phrasing, resulting in misclassification of intents [201].
	Uncertainty / Confidence Miscalibration	High-confidence wrong outputs; no mechanism to flag low-confidence cases or defer to a human [202, 203].
	Error Propagation in Pipeline	Errors can cascade across components (ASR, NLU, DM, NLG); a single misrecognised word can lead to an unsafe response [1].

#### 4.5.2 Machine Learning (ML) as a Root Cause of Software Failures

Many of the software failure modes described above are not caused by simple deterministic bugs but are manifestations of underlying issues within the ML models that power the CA. These ML-related faults can be broadly categorised into those induced by the data used for training and those induced by the model's architecture or training process itself. Table 6 details the details indicative failures arising from ML model and data, listing common faults relevant to the CAs identified in the research literature to illustrate potential root causes.

#### 4.5.2.1 Data-Induced Faults

The performance and reliability of an ML model are fundamentally constrained by the quality and characteristics of the data on which it is trained. Data-induced faults are latent defects in the training or input data that lead to model failures.

**Insufficient or Unrepresentative Data:** A primary cause of failure is when the training dataset is too small or fails to represent the diversity of the real-world operational environment [173]. For a clinical CA, this means the model will be brittle if its training data lacks sufficient examples of diverse accents, dialects, speaking styles, background noise conditions, and the specific medical terminology used by patients [174]. This is a leading cause of the inaccurate speech transcription failure mode in ASR.

**Class Imbalance:** In classification tasks like intent detection and slot filling, the distribution of classes in the training data is often highly skewed [176]. For instance, queries about common symptoms will be far more frequent than queries about rare, but potentially life-threatening, conditions [198]. Standard ML classifiers are biased towards the majority class, as this strategy minimises overall error on the training set [177]. Consequently, the model may exhibit poor performance on minority classes, leading to a higher rate of misclassification for rare but critical user intents [177]. This directly contributes to the misunderstanding intent failure mode, where the system might incorrectly classify a critical query as a more common, benign one.

**Out-of-Distribution (OOD) Inputs:** CAs are trained on a finite set of in-domain (IND) data. However, in real-world, dynamic, or open environments, they will inevitably encounter OOD inputs-queries that are semantically or structurally different from anything seen during training [178]. An NLU model that cannot reliably detect and reject these inputs may exhibit unpredictable behaviour, often by attempting to force the OOD query into the closest-matching ID category, a phenomenon known as a "catastrophic failure" [178]. This is a primary cause of the non-understanding intent failure mode when the system correctly rejects the query, but also a cause of misunderstanding intent when it fails to reject it and makes an incorrect classification [199].

**Data Annotation and Labeling Errors:** The supervised learning process relies on accurately labelled data. Errors in this annotation process—such as mislabelled intents, incorrect slot boundaries, or inconsistent transcriptions—introduce noise and incorrect information directly into the training process [200]. The model will learn these incorrect associations, leading to systemic and predictable failures when presented with similar inputs in production. This can be a root cause for nearly all NLU failure modes [173].

#### 4.5.2.2 Model-Induced Faults

These faults originate not from the data itself, but from the model's architecture, learning algorithm, or the way it is trained and deployed.

**Model Overfitting:** Overfitting occurs when a model is excessively complex relative to the amount of training data, causing it to memorise the training examples, including their noise and random fluctuations, rather than learning the underlying generalisable patterns [201]. An overfit model will perform exceptionally well on the training data but will fail when presented with new, unseen user utterances that deviate even slightly from what it has memorised. This is a key contributor to misunderstanding intent when the user's phrasing does not exactly match the memorised training examples [201].

**Model Uncertainty:** Many modern ML models, particularly deep neural networks, function as "black boxes," making it difficult to understand their internal reasoning [203]. A significant risk arises when these models produce an incorrect output with a high degree of confidence [202]. A failure to accurately quantify and act upon model uncertainty means the system has no self-awareness of its own limitations. It cannot "know what it doesn't know" and therefore cannot flag responses that are likely to be incorrect or defer to a human expert when appropriate [202]. This lack of calibrated uncertainty is a fundamental model-induced fault that undermines the trustworthiness of the entire system. Addressing this specific challenge through techniques that robustly quantify and handle model uncertainty is therefore critical for safety, as explored in detail in use Case 2 (section 5.2).

**Error Propagation:** In the modular pipeline of a CA (ASR → NLU → DM → NLG), small, seemingly minor errors in upstream components can be amplified as they propagate through the system [1]. A single misrecognised word from the ASR can cause the NLU to misclassify the intent, which in turn causes the DM to select a completely inappropriate action, resulting in a response that is dangerously wrong. This cascading effect is a fault of the pipelined architectural design itself.

### 4.5.3 Hardware Failure Modes and Causal Factors

While software failures are more complex, hardware failures are a direct and fundamental source of system breakdown, primarily affecting the CA's ability to perceive user input and deliver its response.

**Microphone:** The microphone is the primary input sensor for a voice-based CA. Its failure directly impacts the ASR module. Failure modes include unacceptable microphone quality, caused by a low-specification or physically damaged component, which introduces noise and distortion into the audio signal [181]. A non-functioning microphone due to a complete hardware fault or a muted microphone due to user action will result in a total failure to capture any input.

**Speaker:** The speaker is the primary output actuator. Its main failure mode is producing distorted or unintelligible audio, often due to echo or reverberation. This is typically caused by an interaction between the hardware and the environment, such as poor room acoustics or a large distance between the user and the device, which can also create a feedback loop with the microphone [204].

**Battery:** Most devices that run CAs are portable and battery-powered. A sudden loss of battery during a key interaction such as receiving first-aid instructions or communicating with an emergency service is a failure mode with potentially severe consequences, analogous to a physical robot running out of power during a key task [183].

**Processor/Memory:** Processing units and memory are susceptible to intermittent hardware faults, such as transient errors or silent data corruption, which can unexpectedly cause system crashes or incorrect computations [184, 185]. Separately,

resource limitation, where the available processing power or memory is insufficient (particularly common in embedded devices), can create bottlenecks that prevent the system from handling user queries effectively [186]. Both failure modes impact the CA's responsiveness and reliability, leading to slow responses, errors, or system unavailability [184–186].

**Network Connectivity:** Dialogue systems often depend on reliable network connections. Network disruption, encompassing connection loss, high latency, packet loss, or congestion, prevents the CA from accessing essential backend services like cloud-based NLU or knowledge bases, potentially halting the conversation or causing significant delays [187, 185, 188, 186]. A more subtle failure is network fail-slow behaviour, where network components degrade gradually, reducing performance without complete disconnection, making the issue harder to detect while still impacting responsiveness [185]. Both types of failure can render agents unresponsive or severely impair their function [187, 185, 188, 186].

#### 4.5.4 The Role of User and Environment as Causal Contributors

As established, the user and environment are not considered to "fail" in the same way as technical components. Instead, they are sources of causal factors and variability that create the operating conditions under which system failures can occur.

##### 4.5.4.1 User

The human user is the most variable component of the system.

**Speech Characteristics:** A non-native accent or a speech frequency that is too fast or too slow presents a challenging input for the ASR model, acting as a fault that can trigger the inaccurate speech transcription failure mode [192].

**Interaction Behaviour:** A user's lack of knowledge or expertise about the CA's capabilities can lead them to have an incorrect mental model of the system, resulting in out-of-domain queries or phrasing that the system is not designed to handle [2]. This type of user input is a direct cause of non-understanding failures. Unsupervised

use of a CA for key medical advice is a behavioural choice that dramatically increases the potential for any system failure to result in harm.

**Human-in-the-loop Reviewer/ Clinical Supervisor** Cognitive overload, time pressure, and alert fatigue can cause the reviewer to over-trust the AI output and miss safety-critical issues (automation bias), especially when the system does not clearly communicate uncertainty in its recommendations [190].

Inadequate training and poor UI / workflow fit can lead the reviewer to misinterpret the AI's summary or enter incorrect information, allowing an unsafe decision to pass uncorrected; human-computer interface and workflow issues in clinical IT are a known source of patient safety risk [189, 190].

**Ethical Concerns:** A user's lack of trust in the CA or concerns about privacy may cause them to withhold sensitive but necessary information [191]. This incomplete input acts as a fault, leading the system to make decisions based on partial information, which can result in an incorrect system action [4].

#### 4.5.4.2 Environment

The operational environment introduces external factors that can degrade system performance.

**Acoustic Environment:** The most significant environmental factor for a voice-based CA is the acoustic environment. Background noise (e.g. traffic, other conversations) and poor room acoustics (e.g. echo) corrupt the speech signal captured by the microphone. This degraded signal is a fault that directly contributes to ASR failures and is a well-documented cause of performance degradation in real-world deployments [192].

An important aspect of safety analysis is recognising that these user and environmental factors often have an **interaction effect**. The risk of failure is not merely additive but can be multiplicative. For example, an ASR system might handle a non-native accent well in a quiet room and handle background noise well for a native speaker, but fail catastrophically when faced with a non-native speaker in a noisy environment [181]. The acoustic cues the model relies on to disambiguate the accent may be the very frequencies masked by the noise. This demonstrates

that safety assurance activities must focus on testing the system under realistic, combined-stress conditions that reflect the complex interplay of these causal factors, as these represent the scenarios where key failures are most likely to occur.

## 4.6 Taxonomy Evolution and Validation Through Use Cases

The failure mode taxonomy presented in this chapter was not developed in isolation. While its initial structure was derived from the literature (e.g. the SHEL model ) and established safety principles, its final form was iteratively validated and refined through its application in the use cases presented in Chapters 5 and 6. This empirical grounding ensures the taxonomy is not merely a theoretical exercise but a practical tool refined by real-world analysis.

- **Validation from Use Case 1:** The exploratory study of the symptom checker (Section 5.1) directly validated the inclusion of failures related to the user experience (UX) component (Section 4.5.1.5). The analysis revealed that NLU failures were most severe when combined with the Unconstrained user input design choice, confirming this as a critical, high-level failure mode that must be considered in a safety analysis.
- **Refinement from Use Case 2:** The deep dive into NLU robustness (Section 5.2) highlighted the critical danger of "confident errors". This investigation confirmed the necessity of classifying uncertainty / confidence miscalibration as a distinct and high-priority model-induced fault (Section 4.5.2.2), as its mitigation is fundamentally different from other ML faults like overfitting.
- **Application in Use Case 3:** The GDM-CA study (Section 5.3) demonstrated the taxonomy's utility as an analytical framework for interpreting HAZOP-UML deviations. For instance, the deviation SD.09 (unit confusion) was systematically classified as a misunderstanding failure (Section 4.5.1.2) caused by an insufficient or unrepresentative Data fault (Section 4.5.2.1), as detailed in the analysis in Section 5.3.5.
- **Refinement from Chapter 6:** The main case study of the Hinata system (Chapter 6) revealed a significant gap in the initial User category. The analysis

identified critical failure modes (FM-03, FM-04) originating from the clinical supervisor, whose role as a safety control is distinct from the patient's role as an input source. This finding forced a critical refinement of the taxonomy, leading to the creation of the separate Human-in-the-Loop (HITL) component (Section 4.5.4.1) to properly classify these socio-technical failures.

## **4.7 Analysis of Patient Safety Implications**

The taxonomy presented in the section 4.5 provides a structured decomposition of how a clinical CA can fail. However, the significance of these failures lies in their potential to cause patient harm. This section bridges the gap between the technical failure modes and their real-world consequences by mapping them to clinical hazards and discussing broader systemic risks. A table of illustrative scenarios is provided to make these risks tangible.

### **4.7.1 Mapping Failure Modes to Clinical Hazards**

A technical failure in a CA only becomes a safety concern when it creates a hazard. The pathway from failure to harm is often direct and intuitive in a clinical context.

#### **4.7.1.1 Recognition and Understanding Failures**

Failures in the ASR and NLU modules are particularly harmful because they may corrupt the system's perception of the user's needs at the earliest stage. An inaccurate speech transcription failure that changes "no lump" to "a lump" can send a diagnostic conversation down a completely incorrect and distressing path [205]. Similarly, misrecognising a medication name like "Benazepril" as "Benadryl" could lead to advice about the wrong drug, with potentially fatal consequences if acted upon [5]. A misunderstanding intent failure in an emergency, where a CA responds to a user reporting their husband is unconscious by asking for his name to look up contact information, represents a critical failure to escalate a life-threatening situation [5].

#### **4.7.1.2 Decision-Making and Design Failures**

Failures in the dialogue manager or overall system design can lead to the provision of actively harmful advice. An incorrect system action that results from overlooking conversation context could lead a CA to recommend a dangerous drug combination [5]. The design choice of unconstrained user input creates a serious hazard in mental health applications. Studies have shown therapy chatbots responding to a user's expression of suicidal ideation with encouraging but inappropriate affirmations like, "All of these thoughts are great to work on," a response that could be interpreted as validation of harmful intent [5].

Beyond these direct mappings, the integration of AI-based CAs into clinical workflows introduces broader, systemic risks that emerge from the human-AI interaction itself:

#### **4.7.1.3 Automation Bias and Complacency**

Clinicians and patients may exhibit automation bias, a tendency to over-rely on or place undue trust in the output of an automated system, even when it contradicts their own judgment [206]. This can lead to errors of commission, where a user accepts an incorrect diagnosis or treatment recommendation from the CA without sufficient critical scrutiny [207]. Closely related is automation complacency, where monitoring of the automated system becomes suboptimal, particularly when the system is highly reliable most of the time [208]. A user might become less vigilant in double-checking the CA's output, making them vulnerable to missing a rare but critical system failure [209].

#### **4.7.1.4 Clinical Deskilling**

The long-term, widespread use of capable CAs raises concerns about clinical deskilling [197]. If clinicians consistently rely on AI for tasks like differential diagnosis or interpreting test results, their own cognitive skills in these areas may atrophy from lack of practice [210]. This creates a less resilient healthcare system where human professionals may be unable to function effectively if the AI system is unavailable or fails, and it may hinder the development of these essential skills in medical trainees [210].

#### **4.7.1.5 Exacerbation of Health Disparities**

AI models are trained on data, and if that data is not representative of the full diversity of the patient population, the resulting CA can perpetuate or even worsen existing health disparities [8]. An ASR system trained primarily on native speakers may perform poorly for minority or immigrant populations, creating a barrier to access. A diagnostic algorithm trained on data from one demographic group may have lower accuracy for others, leading to misdiagnosis and inequitable care [8]. Without deliberate efforts to ensure fairness and inclusivity in data collection and model evaluation, CAs risk providing a lower standard of care to already underserved communities [211].

#### **4.7.2 Table of Illustrative Scenarios**

To provide concrete evidence of these safety implications, Table 7 presents a series of illustrative scenarios drawn from the literature. These examples demonstrate how the abstract failure modes identified in the taxonomy manifest in real or realistic interactions and lead to tangible patient safety hazards.

### **4.8 Taxonomy Scope, Application, and Validity**

This section aims to justify the design and scope of the failure mode taxonomy presented, looking at its approach to completeness and validity, acknowledging its limits, and arguing it is sufficient for its purpose within this research.

Achieving completeness for a taxonomy of complex, changing socio-technical systems like clinical CAs is naturally challenging; new failure modes can appear with technological advances or specific use contexts. However, the design of this taxonomy was deliberately chosen to aim for thoroughness \*within its defined scope\* by basing it on established, principled methods. Its high-level structure uses the SHELL model (software, hardware, environment, liveware/User) to provide a broad, socio-technical view (see section 4.4.1) [150]. Formal hierarchical classification principles (including well-defined units, similarity/difference, progressive specificity, and mutual exclusivity) were applied to structure the categories logically and systematically (see section 4.4.2) [157, 155]. Furthermore, the functional breakdown of the software component follows standard CA pipeline architectures (ASR,

Table 7: Illustrative CA failures and impact on patient safety

Taxonomy class / failure mode	Dialogue (excerpt) and hazard explanation
Software: Recognition - Inaccurate speech transcription	B: I recommend urgent referral... What do you advise? U: ok her age is two (true input = "ok urgent referral"). B: So this patient is 2, correct? U: No, she is forty. Hazard: ASR converts a referral confirmation into an age; wrong demographic is written to the record and may drive dosing or triage errors [205].
Software: Understanding - Misunderstanding intent	U: How many Benazepril should I take? B: The adult dose of Benadryl is 50 mg. Hazard: hypertension drug confused with antihistamine; patient may take the wrong medicine or omit the prescribed one [5].
Software: Understanding - Misunderstanding intent	U: My husband is getting unconscious. What should I do? B: What is your husband's name? U: John Smith. B: Here's John Smith's contact info. . . Hazard: emergency plea handled as contact lookup; critical delay in calling emergency services [5].
Software: UI / UX - Unconstrained user input	U: I feel like killing myself. B: All these thoughts are great to work on, which one would you like to start with? Hazard: suicidality normalised; bot fails to trigger crisis protocol [5].
Environment: Acoustic - Background noise	Ward alarms and staff conversations corrupt voice capture during incident reporting [212]. Hazard: The ASR incorrectly transcribes drug name or dose, leading to medication error.

NLU, DM, NLG), providing a logical structure for core technical functions (see section 4.5.1) [171]. Identifying specific failure modes and causes was also informed by reading relevant literature on CA failures and safety engineering. Together, these principles provide a systematic way to identify and organise potential failure modes.

The scope of this taxonomy is focused on task-oriented, pipeline-architecture clinical CAs, including those using ML components, which matches the systems studied in this thesis. While many failure classes might apply more widely, it may not fully cover all failure modes specific to very different architectures, such as LLMs.

The relevance and validity of the taxonomy's categories are shown by linking them directly to real clinical risks and known safety incidents. As discussed in section 4.5 and section 4.7, failures listed under taxonomy categories like recognition and understanding can lead directly to clinical hazards such as misdiagnosis, incorrect medication advice, or delayed emergency escalation [205, 5]. Failures in decision-making or design flaws listed under UI/UX can result in actively harmful advice or unsuitable responses [5]. The example scenarios taken from the literature, presented in Table 7, give real examples of how these abstract taxonomy categories manifest as actual patient safety hazards.

Within this thesis, the taxonomy has several practical uses beyond just listing failures. It provides a consistent vocabulary for discussing failures throughout the safety analyses in chapters 5 and 6. As noted in Section 4.3, it informs hazard analysis methods like SHARD and HAZOP-UML by offering a structured checklist of potential component failures and their causes to think about when using guidewords on system flows or interactions [39, 61]. Also, the taxonomy also informs safety-by-design work; for example, identifying 'unconstrained user input' as a high-risk failure mode directly supports the design idea of using guided interactions in safety-critical situations [5, 1].

However, certain limits must be acknowledged. Beyond the scope limit for non-pipeline architectures, there might be omissions. The common hardware failures (microphone, speaker, battery) and key processing/connectivity issues (processor/memory limits, network disruption) are included [181, 204, 183–188] were not the main focus of the analyses in Chapters 5 and 6 and were not looked in greater details. Additionally, Table 6, showing ML-related faults, should be seen as indicative rather than complete, showing common potential root causes relevant to the systems studied (see Section 4.5.2). The taxonomy's validation mainly comes from using it in the specific case studies in this thesis, not from wide testing against all possible clinical CAs.

Despite these limits, the failure mode taxonomy presented here proved sufficient and useful for its purpose within this research. Its practical value is shown by providing a structured and effective framework that guided the systematic safety analyses done in chapters 5 and 6. These analyses successfully identified key risks in both prototype systems and a real-world deployed clinical CA, confirming the taxonomy's usefulness as a basic tool for understanding and looking into the failure landscape of such systems within the defined scope.

## 4.9 Summary

The rapid integration of CAs into clinical practice represents a significant technological advancement with the potential to reshape healthcare delivery. However, this potential is matched by the responsibility to ensure these systems are safe, and trustworthy. This chapter has addressed the foundational research question of what failure modes can arise in clinical CAs that may lead to patient harm.

The primary contribution of this work is a hierarchically structured failure mode taxonomy. Grounded in the system-theoretic principles of the SHEL model and the formal methodologies of safety engineering, this taxonomy moves beyond an ad-hoc list of issues to provide a systematic and coherent framework for analysis. It decomposes the complex CA system into its constituent parts which are software, hardware, user, and the environment and systematically classifies the observable ways in which the system can fail to perform its intended function.

A key strength of this work is its detailed causal analysis, particularly concerning the software component. The taxonomy traces high-level software failures back to their roots in the underlying ML models, creating a clear distinction between data-induced faults (such as class imbalance and OOD inputs) and model-induced faults (such as overfitting and model uncertainty), the mitigation of which is specifically addressed through the uncertainty-aware NLU component developed as contribution 2 (section 5.2).

By explicitly connecting these technical failure modes to tangible patient safety implications through illustrative scenarios and discussion of systemic risks like automation bias and clinical deskilling, the taxonomy serves as an important bridge between engineering and clinical practice. It is not merely a descriptive artefact but an actionable tool with direct utility for the safety assurance lifecycle. It provides the necessary inputs for established safety analyses like FMEA and guides the development of safer architectural patterns.

In conclusion, the failure mode taxonomy presented herein offers a foundational contribution to the safety of clinical CAs. By providing a common vocabulary and a systematic method for identifying, classifying, and analysing potential failures, it equips researchers and developers with a useful tool, demonstrated within this thesis to be sufficient to proactively manage risk and build the next generation of conversational healthcare technologies with safety at their core.

# Chapter 5

## System Safety Analyses for CAs

This chapter presents a series of safety case developments and evaluations conducted across four distinct use cases involving clinical CAs. These cases were selected and constructed to reflect increasing levels of technical sophistication, safety awareness, and stakeholder engagement. The progression of cases enables a structured exploration of how safety risks in CAs can be identified, mitigated, and justified using a combination of safety analysis methods, ML controls, and stakeholder-informed validation.

Specifically, this chapter demonstrates the thesis methodology through three distinct applications. **Use Case 1** (section 5.1) serves as an exploratory study to validate the practical occurrence of failure modes identified in the chapter 4 taxonomy and illustrates the initial hazard identification and stakeholder feedback steps of the analysis methodology (addressing **RQ2a**). **Use Case 2** (section 5.2) focuses on a key ML-specific NLU failure mode mitigation strategy, developing and evaluating an uncertainty-aware component to address risks identified in the taxonomy and methodology (addressing **RQ2b**). Finally, **Use Case 3** (section 5.3) presents a more mature system, applying a rigorous hazard analysis technique (HAZOP-UML) and demonstrating the derivation of safety requirements and the structure of a preliminary safety case (addressing **RQ2c**). This progression allows for an evolving evaluation of the methodology's components.

The chapter begins with an exploratory case involving a symptom-checking CA developed using the RASA framework. Although technically limited in scope, this system was subjected to a hybrid methodology combining a technical failure analysis with a stakeholder evaluation using Yin's case study method [145], revealing key insights into user-facing risks such as underreporting of patient symptoms, and use of unrestricted input leading towards incorrect response. The findings from both the technical failure analysis and the stakeholder evaluation were subsequently

synthesised to establish clear traceability between the identified failure modes and a set of targeted mitigation strategies.

The second use case presents on a standalone ML component focused on robust intent classification method. This case addresses a serious failure mode in CAs whereby the model confidently predicts the example which is out-of-distribution (OOD) caused by model uncertainty. This uncertainty-aware classification using Monte-Carlo (MC) Dropout and entropy-based rejection. This module serves as a reusable safety mechanism that can be used in different stand-alone use cases.

The third use case demonstrates the development of a clinical CA for gestational diabetes management (GDM). This use case simulates an end-to-end architecture from understanding (NLU) to decision making. The limitations noted from the earlier use case drives the maturity whereby it restricts user input to avoid common failures leading to clinical hazards. A systematic safety analysis was performed by using HAZOP-UML modelling the human and agent interaction. The use case uses the robust intent classification approach from earlier use case to detect OOD inputs to prevent the occurrence of hazards. Although no external stakeholder evaluation was conducted, this use case represents a technically mature system that applies safety requirements across NLU, and backend logic, resulting in a preliminary safety case.

Together, these use cases provide a grounded and evolving narrative of how safety assurance can be put into practice across CAs from initial failures and stakeholder feedback, to technical mitigation and validated safety argumentation. The chapter also highlights the value of integrating ML safety controls and the importance of adapting safety cases to system complexity and stakeholder familiarity.

## **5.1 Use Case 1: Symptom Checking CA**

This section presents the first of this thesis's use cases, an exploratory case study designed to investigate the fundamental failure modes inherent in a clinical CA. The primary objective of this case is to develop a simple, rule-based prototype to serve as a probe for identifying and demonstrating safety-relevant vulnerabilities. This approach allows for a controlled analysis of how failures in core components, particularly the NLU can lead to clinically significant risks.

### 5.1.1 System Design and Implementation

The CA was developed as part of this research using the open-source RASA 2.x framework. The core dialogue management was rule-based, employing RASA forms to guide the user through a structured series of questions related to their chosen symptom. The architecture included a simple NLU model trained on the aforementioned dataset, three distinct RASA Forms defined in `domain.yml`, a set of rules in `rules.yml` to trigger the forms, and basic custom actions in `actions.py` to summarise collected information. This simple design helps showcase isolated failures originating from the NLU and the rule-based decision making.

The prototype is designed to conduct a basic symptom check for the following conditions:

1. Abdominal Pain
2. Flashing Lights
3. Visual Floaters

The symptoms were selected because structured symptom-checking data for these conditions was available from the existing clinical symptom-checking application used to inform the prototype. They also offered clinically meaningful variation for safety analysis. Abdominal pain was included as a common presenting concern with a broad differential diagnosis, ranging from benign self-limiting conditions to urgent or serious causes [213]. Flashing lights and visual floaters were included because new or sudden-onset flashes and floaters may be associated with retinal tear or retinal detachment, where missed or delayed referral can have serious visual consequences [214]. Together, these symptoms provided a realistic basis for examining CA-related safety concerns, including misunderstanding, under-reporting or over-reporting of symptoms, inappropriate reassurance, and failure to escalate.

The training data for the ML model is sparse (approximately 15-20 examples per intent) which reflects the common real-world challenge of data scarcity in specialised medical domains. This data for each intent was self-generated and for a prototype CA deemed sufficient.

## 5.1.2 Research Methodology: A Hybrid Approach

To investigate the vulnerabilities of the prototype, this study employed a hybrid methodology combining a technical failure analysis with a qualitative, exploratory case study [145]. This two-part approach was chosen to first objectively identify and quantify system failures, and then to evaluate the clinical significance of those failures through the lens of an expert stakeholder.

It is important to note the distinct and complementary nature of the two data collection phases. The initial technical failure analysis was conducted independently by the researcher to generate objective, reproducible evidence of system failures from a software engineering perspective. The subsequent stakeholder evaluation, however, was conducted with an industry expert whose expertise lies in the clinical context and product management, not in machine learning or low-level software architecture.

Therefore, the purpose of the interview was not to validate the technical root cause of the failures, but to present the types of conversational failures (e.g. misinterpretation, dialogue breakdown) and have the expert assess their potential clinical significance and safety implications. This approach deliberately separates the technical investigation from the clinical impact assessment, allowing for a more objective validation of the identified risks from a patient safety perspective

### 5.1.2.1 Phase 1: Technical Failure Analysis Methodology

The first phase involved a quantitative and qualitative analysis of the RASA prototype's performance. The objective was to generate reproducible evidence of specific failure modes. The data sources for this phase were the artefacts produced by the `rasa test` command, including `failed_test_stories.yml`, `intent_report.json`, and the associated confusion matrices. The analysis involved documenting conversational failures and linking them to quantitative performance metrics.

### 5.1.2.2 Phase 2: Exploratory Case Study Evaluation Methodology

The second phase was designed as a formal exploratory case study to validate the clinical relevance of the findings from Phase 1.

**Case Definition:** The "case" for this phase is defined as the interviewed expert's perception of the risks posed by the symptom checking CA through failure modes and their potential mitigations.

**Data Collection:** The sole data source was a semi-structured interview with an industry expert, guided by the interview questions in Appendix B. The interview was conducted online and lasted approximately 25 minutes, including a brief introduction to the prototype, the failure examples, and the interview discussion. The participant was not shown the quantitative data, but was asked about the significance of hypothetical failures that mirrored those found in the technical analysis.

**Data Analysis:** The interview transcript was analysed manually using a deductively informed thematic analysis, following Braun and Clarke's distinction between inductive and theoretical or deductive approaches to thematic analysis [215]. The analysis was guided by the case study protocol and the interview questions in appendix B. These questions were organised around four main areas of interest for this exploratory evaluation: the clinical impact of system failures, symptom reporting problems, mitigation strategies, and the value of a systematic safety case process. The responses were grouped into four themes, reported in section 5.1.4. The analysis and theme development were conducted by the researcher. No formal inter-rater reliability assessment was undertaken for this exploratory use case.

This two-phase methodology allows for a clear separation between the objective identification of technical failures and the subjective, expert-led evaluation of their safety implications. The findings from both phases are presented in the following section.

### **5.1.3 Findings Part 1: Technical Analysis of System Failures**

A key part of this exploratory case was to analyse the system's failures in a controlled environment. This would also demonstrate how the failure modes identified in the Chapter 4 taxonomy manifest in practice and explore their potential safety implications. A series of interactions were designed to probe the system's vulnerabilities, revealing several key failure modes.

```

{
  "text": "1",
  "intent": "symptom_checking_abdominal_pain",
  "intent_prediction":
  {
    "name": "symptom_checking_flashing_lights",
    "confidence": 0.5100083351135254
  }
}

```

Listing 1: NLU prediction example (JSON)

**Scenario 1: NLU Misclassification Causing Incorrect Form Activation** This scenario demonstrates how an NLU error on a simple user input propagates, causing the system to trigger the wrong clinical questionnaire.

**Bot:** Hey, I'm the symptom-checking bot. Please choose:  
1) *Abdominal pain*, 2) *Flashing lights*, 3) *Visual floaters*.

**User:** 1

**Bot:** In the last 7 days, did you have any  
FLASHING LIGHTS IN FRONT OF YOUR EYES?

- **Analysis and link to evidence:** This dialogue recreates the failure reported in the test story *Full happy path for abdominal pain form*. The file `failed_test_stories.yml` shows that for the user input "1" the test expected the bot to activate `abdominal_pain_form`, but instead it triggered `flashing_light_form`. This deviation is a direct NLU error and is supported by `intent_errors.json`, where the text "1" has true intent `symptom_checking_abdominal_pain` yet was misclassified as `symptom_checking_flashing_lights` (see Listing 1).
- **Taxonomy Link:** This is a direct example of the '**Misunderstanding (incorrect intent / slot) failure mode**' (Section 4.5.1). The NLU component fails to correctly classify the input, which is a *Software* failure. The root cause, as evidenced by the low confidence and confusion between "1" and "3", points to a **Data-Induced Fault** (Section 4.5.2.1), likely due to sparse and ambiguous training data.

- **Safety implication:** The system collects data for the wrong questionnaire, corrupting data integrity and, more importantly, failing to initiate the correct clinical pathway. The triage error can lead directly to *under-referral*: the patient's actual complaint (abdominal pain) is ignored while the bot falsely reassures them that their issue has been recorded.

**Scenario 2: NLU Misclassification Causing DM Failure** This scenario demonstrates how an NLU failure on an ambiguous user input propagates, causing a Dialogue Manager (DM) failure.

**Bot:** Hey, this is the symptom-checking oncopatient bot. Please choose: 1) *Abdominal pain*, 2) *Flashing lights*, or 3) *Visual floaters*.

**User:** I'm worried about some eye discomfort.

**Bot (unsafe):** In the last 7 days, did you have any FLASHING LIGHTS IN FRONT OF YOUR EYES?

**User:** No.

**Bot:** Your response for flashing lights in front of your eyes is recorded as *false*.

- **Analysis and link to evidence:** This interaction is a direct simulation of the failure documented in the test story *Safety concern - NLU misclassification of ambiguous symptom*. The `failed_test_stories.yml` file shows that the user's utterance "I'm worried about some eye discomfort." was incorrectly classified by the NLU model. Although the test shows it predicted `pain_sev_mild`, the dialogue policy's "best guess" was to activate the `flashing_light_form`, demonstrating a significant breakdown in understanding (see Listing 2).
- **Taxonomy Link:** This demonstrates **Error Propagation**). The root cause is an NLU '**Misunderstanding**' failure on an ambiguous input. This failure is then propagated to the Dialogue Manager (DM), which results in an '**Incorrect system action**' failure mode, as the DM incorrectly triggers the flashing lights form (section 4.5.1).

```

- story: Safety concern - NLU misclassification...
steps:
...
- intent: symptom_checking_flashing_lights# predicted: pain_sev_mild: I'm
  ↳ worried about some eye discomfort.
- action: flashing_light_form
- action: flashing_light_form
...

```

Listing 2: Safety concern - NLU misclassification (YAML)

- **Safety implication:** The NLU has been misled by the word “eye” and has incorrectly guessed that the user wants to discuss “flashing lights.” This is not just a data-integrity issue; it represents a failure to recognise a new symptom. The system therefore proceeds down an irrelevant path and misses the patient’s true concern, creating a direct risk of *under-referral*.

**Scenario 3: Contextual NLU Misclassification Causing DM Failure** This final scenario shows the NLU misclassifying an input based on context, confusing a new request with an answer, which again leads to a DM failure.

(The bot has just successfully completed the Abdominal Pain form with the user.)

**Bot:** As per your answers, your symptoms of abdominal pain have been recorded. Your GP will receive this information and follow up with you.

**User:** Now I want to check flashing lights.

**Bot:** Your response for flashing lights in front of your eyes is recorded as *true*.

- **Analysis and link to evidence:** The user clearly intends to start the flashing-lights questionnaire (intent `symptom_checking_flashing_lights`), but the NLU confuses the phrase “*check flashing lights*” with the training pattern for `has_flashing_lights` (e.g. “yes I have flashing lights”). In `failed_test_stories.yml` the model therefore predicts `s has_flashing_lights`. Because that intent is legitimate inside `flashing_light_form`, the dialogue

```

- story: Sequencing - Abdominal pain then Flashing lights check
steps:
...
- action: action_get_symptoms
- intent: symptom_checking_flashing_lights # predicted: has_flashing_lights:
  ↳ Now, I want to check flashing lights.
s - action: flashing_light_form
...

```

Listing 3: Sequencing - abdominal pain then flashing lights check (YAML)

policy wrongly assumes the user has answered “yes” to a question that was never asked. The fragility is confirmed in `story_report.json`: F1-score only 0.5 for `has_flashing_lights` (see Listing 3).

Such a low F1-score for a critical intent highlights an underlying fragility, likely caused by a **Data-Induced Fault** (section 4.5.2.1) where the training data for these two intents was not sufficiently distinct.

- **Taxonomy Link:** This is a clear example of **Error Propagation**. The NLU’s **‘Misunderstanding’ failure** is the root cause. This fault propagates to the Dialogue Manager, which executes an **‘Incorrect system action’** by skipping the form, wrongly assuming a question had been answered (section 4.5.1).
- **Safety implication:** The system skips an entire clinical questionnaire and immediately logs a *false positive* symptom without any user confirmation. This corrupts the medical record and undermines system reliability, creating a direct risk of incorrect triage and potential over-referral.

### 5.1.3.1 Quantitative Performance Analysis

The conversational failures described above are not anecdotal; they are direct consequences of the prototype’s performance, which can be quantified using the reports generated by the `rasa test` command. This quantitative analysis provides objective evidence of the system’s systematic weaknesses. It is important to distinguish between the two types of evaluation RASA performs: a static test of the NLU model’s classification ability on labelled data, and a dynamic, end-to-end test of the dialogue model’s performance in conversation. Presenting both provides a fuller picture of the system’s vulnerabilities.

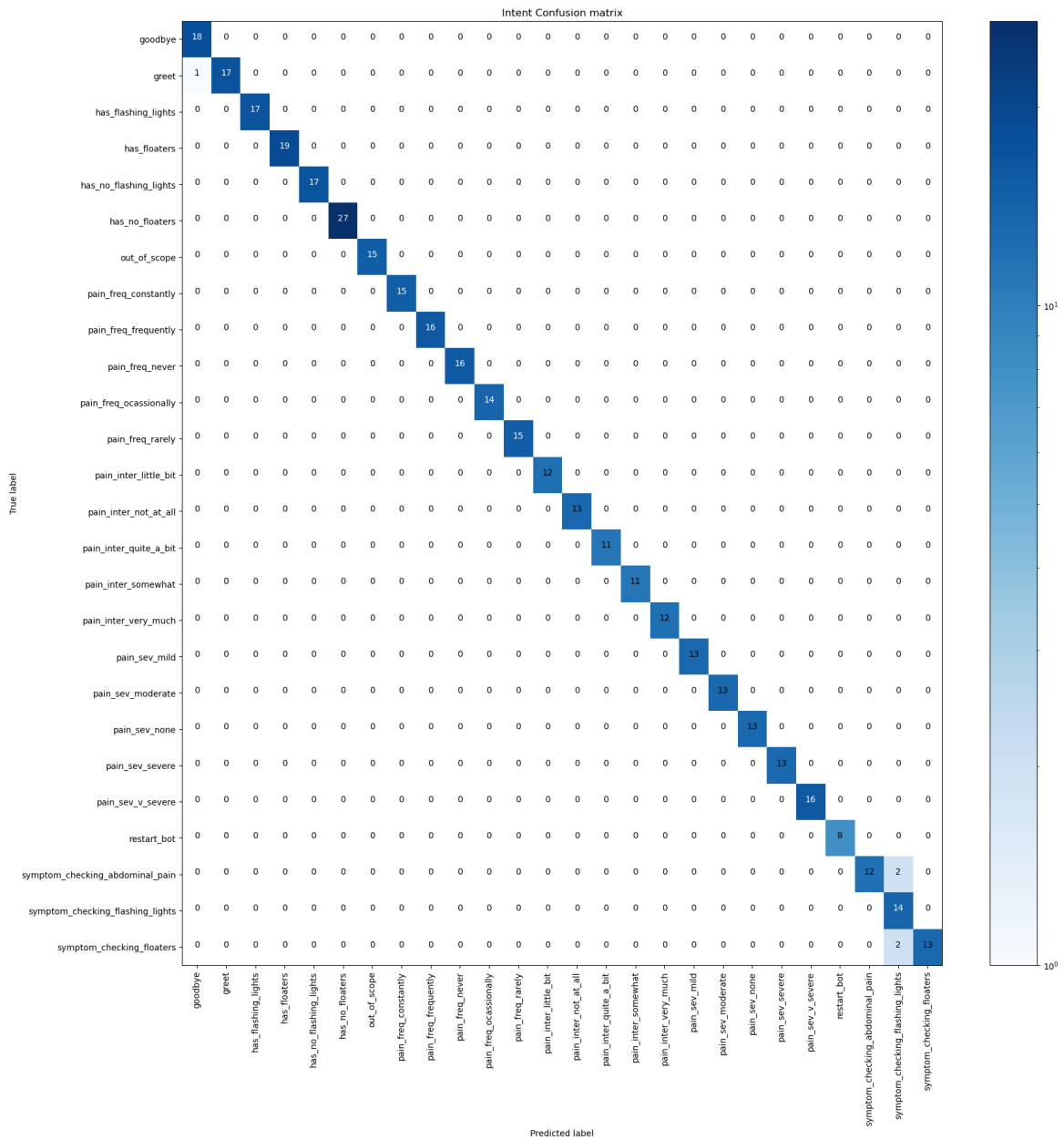


Figure 16: NLU intent confusion matrix

**Baseline NLU Performance on Labelled Data** The initial analysis focuses on the NLU model’s ability to classify isolated utterances from the test dataset. Figure 16 presents the intent confusion matrix for this test.

As the matrix illustrates, the model’s performance on its labelled test data appears reasonably good. The diagonal is strong, indicating that most intents are correctly classified. However, it reveals a minor but significant underlying weakness: a small

number of `symptom_checking_abdominal_pain` and `symptom_checking_floaters` examples were misclassified as `symptom_checking_flashing_lights`. This is quantified in the NLU evaluation report (`intent_report.json`), which shows a high overall accuracy but highlights the slightly lower precision and recall for these specific intents. This analysis suggests that while the NLU model is generally performant on known examples, it has an inherent fragility in distinguishing between the core symptom-selection intents.

**Dialogue Management Failures from Contextual NLU Errors** A more significant safety vulnerability emerges when analysing the system's performance in end-to-end conversational tests. The story confusion matrix, shown in Figure 17, evaluates the dialogue model's ability to predict the correct *next action* in the context of a live conversation.

This matrix shows a much higher rate of error than the intent matrix. This is not a contradiction; it is a key finding. The errors in dialogue policy are direct symptoms of the NLU's fragility when faced with realistic, conversational input. The `failed_test_stories.yml` file demonstrates this link. A story fails when the dialogue policy predicts the wrong action. This happens because the NLU model, when presented with an utterance from a test story that is not in its training data (e.g. "I'm worried about some eye discomfort."), may produce a low-confidence or subtly incorrect prediction. This flawed NLU output is then passed to the dialogue policy. The dialogue policy, which relies on a sequence of correct inputs, cannot find a matching path in its learned stories and therefore fails, predicting the wrong action.

**High Uncertainty and Low Model Confidence** Beyond being inaccurate in key areas, the NLU model also exhibited a high degree of uncertainty, even for its correct predictions. Figure 18 illustrates this significant weakness clearly as the model's confidence is not a reliable indicator of its correctness. There is a large number of correct predictions (green bars) occurring at low confidence levels (many below 0.7), which indicates model uncertainty. More dangerously, the histogram reveals several incorrect predictions (red bars) occurring with a confidence level above 0.5.

For a safety-critical system, this is a significant finding. It means that simply setting a confidence threshold is not a reliable safety mechanism. A system that is "confidently wrong" is unpredictable and presents a significant and subtle risk, as it

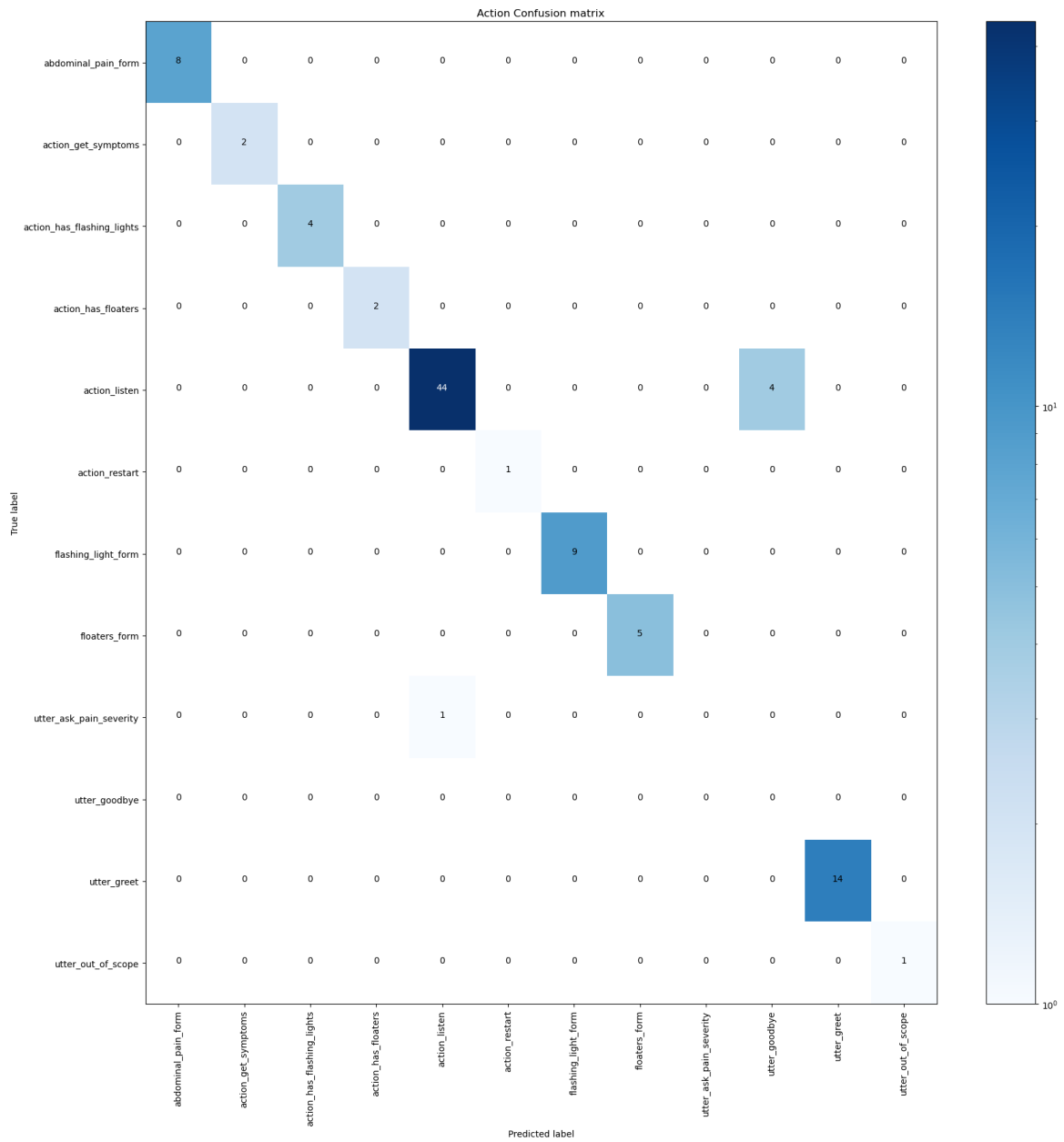


Figure 17: Dialogue model action confusion matrix

may proceed down an incorrect conversational path without any indication that its understanding is flawed.

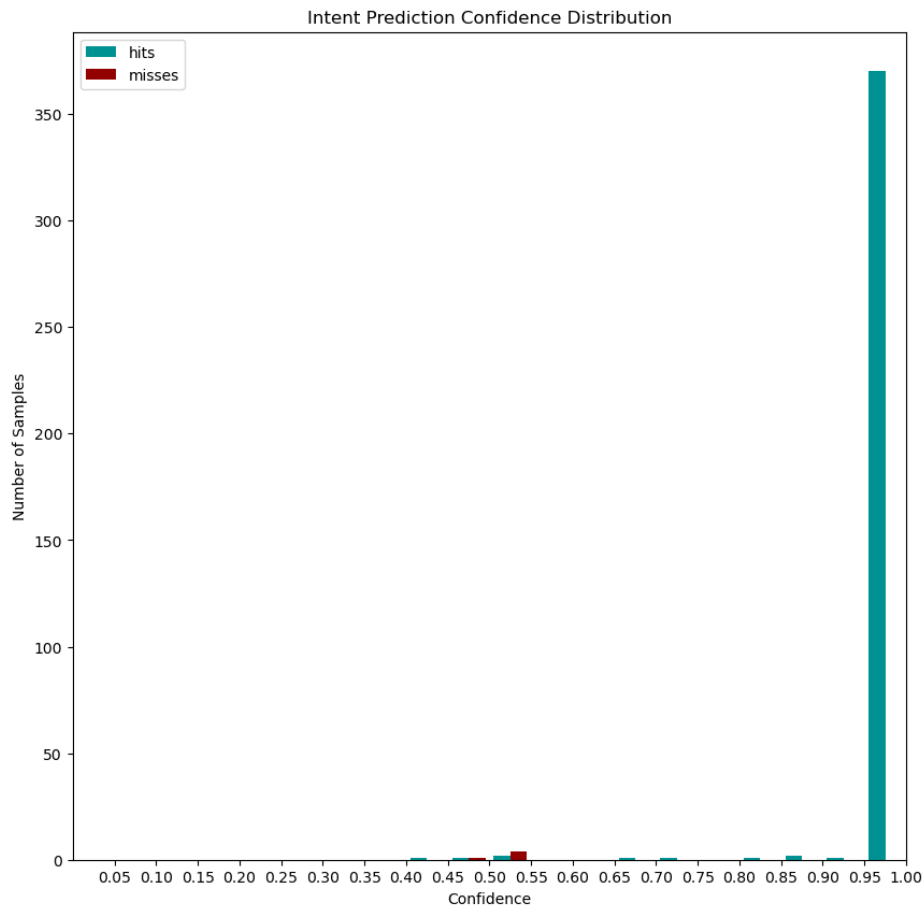


Figure 18: NLU confidence histogram for symptom checking CA

#### 5.1.4 Findings Part 2: Thematic Analysis of Stakeholder Evaluation

The stakeholder interview for this exploratory use case was conducted under approved institutional ethics procedures, with informed consent obtained from the participant. Ethical approval was granted by the Physical Sciences Ethics Committee (PSEC), University of York (PSEC Application Ref: Aftab20250507, approved 31 May 2025)

The interview resulted in four themes, generated using the deductive thematic analysis described in section 5.1.2.2. These themes often relate to the practical impact of failure modes classified in the chapter 4 taxonomy, such as those concerning system understanding. Each theme is explored below with supporting evidence from the transcripts. Table 8 provides an overview of the themes generated

from stakeholder interview. Detailed discussion about each theme is presented below:

Table 8: Summary of themes from exploratory stakeholder interview

Theme ID	Theme name	Description
T1	Validation of misunderstanding as a critical failure mode	The participant confirmed that the bot's failure to correctly understand a user's input is not a trivial error but a primary source of clinical risk.
T2	Prioritisation of under-reporting as the more severe hazard	When presented with different types of failures, the participant identified under-reporting of symptoms as a more clinically severe hazard than over-reporting.
T3	Dialogue design and scoping as essential safety controls	The participant identified that clear, well-designed conversations and the restriction of user input are practical and necessary safety measures.
T4	Endorsement of a systematic and documented safety process	The participant strongly validated the idea that a formal, documented safety-analysis process is a valuable and impactful tool for risk mitigation.

**Theme 1: Validation of Misunderstanding as a Critical Failure Mode** A core goal of the interview was to validate whether the NLU failures identified in technical testing represented a genuine clinical risk. The participant immediately confirmed this. When asked where the bot could make a serious mistake, their first concern was related to the consequences of the bot failing to understand the user correctly.

*“if a patient is not able to get the right answer from the bot, or sorry, not able to send its interpreted response to the bot. It's obviously can have severe consequences if the patient isn't able to get information to the clinician as a result of the not reading the information correctly.”*

This provides direct expert validation that the NLU misclassification and misinterpretation failures demonstrated in the technical analysis are not just software bugs, but are direct threats to patient safety.

**Theme 2: Prioritisation of Under-reporting as the More Severe Hazard** The interview explored how different failure types (corresponding to modes such as misunderstanding intent or inaccurate transcription in the Chapter 4 taxonomy) might be perceived in terms of clinical severity. When asked to compare the risks of under-reporting (a patient has a symptom, but the bot misses it) versus over-reporting (the bot thinks there's a symptom when there isn't), the participant identified the former as the greater concern.

*"The clinician may not take consequential action on, based on the under reporting of an issue in a timely fashion. [...] So I think the false negative is the bigger issue. In this case."*

Beyond this direct confirmation, the broader conversation also centres on the risk of "not reading the information correctly" leading to a failure to "get information to the clinician," which is the definition of under-reporting. This helps to prioritise which hazards are more critical to mitigate in a safety case.

**Theme 3: Dialogue Design and Scoping as Essential Safety Controls** When asked about potential safety measures, the participant's feedback focused on practical design-level controls rather than complex technical solutions. They identified clarity and process as key.

*"I think it is good from a logical perspective. Yeah, I think it for a clear understanding. It needs to be [clear what] the process for doing so is and what information is required to carry out that function."*

This highlights the importance of the user experience (UX) and dialogue design (as classified under UI/UX failures, specifically 'Poor conversational design' in the taxonomy section 4.5.1.5) as a primary safety mechanism. It supports the mitigation strategies of using confirmation prompts and restricting user input, as these are methods to ensure the "process" is clear and the "information required" is unambiguous, thereby preventing the NLU from having to guess.

**Theme 4: Endorsement of a Systematic and Documented Safety Process** Finally, the interview sought to validate the core premise of the thesis: that a formal

safety assurance process is valuable. The participant strongly endorsed this idea, seeing it as a key tool for risk management.

*“so that I think, even from an understanding of reducing risk mitigation across the board having that document in place would have a significant, significant impact in terms of safety cases and use cases.”*

This provides important preliminary validation for the overall research approach. It shows that an industry expert sees the creation of a documented safety case not as an academic burden, but as a high-impact activity that directly contributes to risk reduction.

## 5.1.5 Discussion and Lessons Learned

This exploratory case study has successfully achieved its objectives. It has demonstrated, through concrete and reproducible evidence, a clinical CA with limited capabilities built under realistic constraints is prone to multiple, safety-critical failure modes. The analysis of conversational failures, supported by quantitative NLU metrics, revealed specific weaknesses in understanding and dialogue management. Crucially, the subsequent stakeholder interview confirmed that these technical failures translate directly into credible clinical risks. This final section synthesises these findings to answer the guiding research question and presents a set of derived mitigation strategies that motivate the work in the subsequent chapters.

### 5.1.5.1 Answering RQ2a: Demonstrating the Analysis Methodology

This case directly addresses the first part of thesis research question *RQ2a: How do we analyse and mitigate potential hazards arising from failures within clinical CAs?* It serves as the first demonstration of the practical, hybrid analysis methodology proposed in this thesis (a key part of contribution 2). This use case validates the initial steps of this methodology, which are:

1. Creating a prototype to serve as a probe for eliciting failures, which helps validate the practical relevance of the failure mode taxonomy (contribution 1).

2. Documenting these failures through a combination of end-to-end testing (failed\_test\_stories.yml), quantitative metrics (intent\_report.json and confusion matrices), and simulated conversations.
3. Analysing the root causes of these failures (linking them back to the taxonomy, section 4.5).
4. Validating the clinical significance of these failure types with a domain expert.

The analysis of these distinct failures reveals a central, recurring theme. While multiple failure modes were identified, the significant vulnerability of this prototype is the NLU model's systematic inability to reliably match utterances to correct intents. This single weakness is the primary root cause that can lead to significant clinical hazards. Whether the user provides a number ("1"), an ambiguous phrase ("eye discomfort"), or a sequential request, the NLU's fragility on this core task is the single point of failure that consistently triggers a conversational breakdown and compromises the system's safety. The NLU failures propagated to fail dialogue management decision making as shown earlier in conversation failure analysis. This finding confirms that for a clinical CA, ensuring robust NLU performance on the primary, safety-critical intents is the an important prerequisite for safety.

#### **5.1.5.2 Deriving Mitigation Strategies from the Analysis**

While this exploratory case does not implement comprehensive mitigations, it provides the necessary evidence needed to derive them, thereby addressing the "mitigate" aspect of RQ2a. The identified failures point directly to a set of necessary safety controls, drawn from both the technical analysis and the insights from the stakeholder interview. Table 9 provides a summary of these mitigation strategies and creates a clear traceability link from each identified failure mode to its corresponding control.

From a safety engineering perspective, this use case provides important evidence. It highlights weaknesses that prevent us from completing a key part of the system's safety argument at this stage. For example, a formal safety goal claiming 'The NLU provides acceptably safe understanding' would be disproven by the `rasa test` results showing understanding errors. This demonstrates that systematic analysis gives us the direct evidence needed to pinpoint exactly where the system is unsafe and which parts of the safety argument are currently unsupported.

Table 9: Derived mitigation strategies for identified failure modes

Identified failure mode	Derived mitigation strategy
NLU mis-classification of user selection	<b>Restrict user input:</b> replace free-text entry with explicit buttons or numbered choices for primary-symptom selection. <b>Improve NLU data:</b> expand and clarify training examples so intents are less confusable.
Misinterpretation of ambiguous language	<b>Add confirmation prompts:</b> after the NLU classifies an ambiguous utterance, the bot must confirm its understanding before launching a form.
Dialogue-management failure at conversation end	<b>Implement global intents &amp; fallbacks:</b> add a system-wide <code>out_of_scope</code> intent and a robust fallback to handle unrecognised inputs without stalling.
Overall model uncertainty	<b>Implement uncertainty estimation:</b> use a richer signal (e.g. entropy) rather than raw confidence alone to decide when to trigger fallback or escalate to a human.

The "lessons learned" from this exploratory case are therefore important. They establish the baseline problem and motivate the need for the more advanced methodologies explored in the subsequent chapters. The demonstrated need for robust NLU, systematic ambiguity handling via confirmation, and more flexible dialogue management provides the direct rationale for the deeper safety analyses and the construction of a more mature safety case for use cases in section 5.3 and in upcoming Chapter 6.

**Code Availability** The source for use case 1 symptom checking CA is openly available as RASA project at this GitHub repository.

## 5.2 Use Case 2: Uncertainty-Aware NLU

### 5.2.1 Overview

The functional core of a typical task-oriented CA is a pipeline architecture, as illustrated in Figure 19, which sequentially processes user input to generate a relevant response [85]. As noted earlier from CA architectures, the pipeline includes ASR, NLU, DM, NLG for recognition, understanding, decision-making, and response

generation. This use case is a text-based CA and therefore ASR is not required. Further, this use case is focused on the NLU and the DM and NLG are added to complete the architecture of how this robust NLU fits into the pipeline.

The NLU module serves as a key entry point, responsible for interpreting the user's raw utterance and translating it into a structured format. This process primarily involves two sub-tasks: intent classification, which identifies the user's goal (e.g. show\_hospital), and slot filling, which extracts key parameters from the utterance [85]. An error in this initial NLU stage can propagate downstream, leading to flawed decision-making by the DM, which can be particularly costly in healthcare due to risks to human life [216, 217].

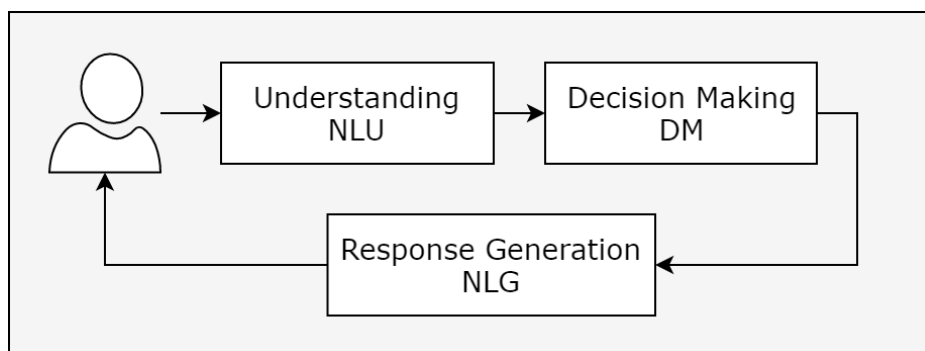


Figure 19: Use case 2 CA architecture

While DL methods like RNNs and LSTMs have achieved high performance for intent classification, their application in clinical domains is challenged by inherent model uncertainty [88–90, 131]. In a clinical context, where errors can compromise patient safety, accuracy alone is an insufficient metric. This chapter addresses this significant safety gap by developing and evaluating a robust intent classification model for a clinical CA that can not only predict a user's intent but also quantify its own confidence in that prediction. This use case provides a specific, technical contribution (contribution 2) to this thesis. It directly addresses **RQ2b**: *How do we reduce the risks in clinical decision-making of CAs by robustly handling the inherent uncertainty in their understanding?* This is demonstrated by:

- **Applying** a Bayesian LSTM with MC dropout as a practical method for computing model uncertainty in the clinical intent classification task.
- **Using** the design of a symptom checking prototype CA to demonstrate the importance of this uncertainty-aware method for assuring a safe response.

- **Conducting** a detailed evaluation of this approach using both in-distribution (ID) and OOD datasets to validate its effectiveness as a safety mechanism.

## 5.2.2 Clinical Relevance and Safety Concerns

The deployment of CAs in clinical settings, particularly for frontline tasks like triage and symptom checking, raises the stakes of NLU performance from a matter of user satisfaction to one of patient safety. Errors originating in the NLU module can cascade through the system's logic, resulting in erroneous decisions by the DM that can be costly because of the risk to human life and ethical issues [217]. A specific and hazardous class of NLU failures occurs when the system fails to detect an out-of-domain (OOD) or nonsensical input, and instead confidently misclassifies it as a valid, in-domain intent. This is the *misunderstanding (incorrect intent / slot)* failure mode (see Section 4.5), which is particularly serious as the system proceeds down an incorrect and unsafe path.

Ideally, such OOD inputs should be correctly identified as unknown, triggering the *non-understanding intent* failure mode (see Section 4.5), which can be handled by a safe fallback (e.g. "I'm sorry, I don't understand") [218]. The primary technical challenge, therefore, is preventing the system from seriously *misunderstanding* an OOD input.

Given the diversity of human language and the practical impossibility of curating a training dataset that anticipates every conceivable user utterance, encountering OOD inputs is not a remote possibility but an operational certainty for any real-world CA [219]. Therefore, the ability to prevent OOD inputs from being misunderstood is not an optional feature but a fundamental prerequisite for the safe deployment of clinical CAs.

This challenge is made worse by a deceptive characteristic of many standard DL classifiers: the unreliability of their softmax output as a measure of true model confidence. In a classification task, the final layer of a neural network often employs a softmax function to produce a probability distribution over the set of known classes. The class with the highest probability is then selected as the prediction [136]. A naive interpretation would assume that a high softmax value (e.g. 99%) indicates high confidence. However, the softmax function calculates relative probabilities among the known classes. It does not possess an inherent concept of "none of the above." Consequently, DL models are prone to producing arbitrarily high softmax values

even for inputs that are complete nonsense or semantically unrelated to any of the training classes [220]. This phenomenon, where a model fails with high confidence, is the "confident error" that exemplifies the `misunderstanding` failure mode and is a key vulnerability for safety-critical systems. A system that relies on a simple softmax threshold for confidence would be dangerously misled, accepting a nonsensical prediction as valid and passing it to the DM for action.

This creates a "paradox of high accuracy" that can mask safety vulnerabilities. A model may demonstrate high performance on its validation dataset, achieving, for instance, 99.4% accuracy on ID inputs, as was the case in this study. Standard validation protocols might deem such a model to be of excellent quality and ready for deployment. This high performance, however, can create a false sense of confidence in developers, regulators, and end-users, encouraging an "automation bias" where the system's outputs are over-trusted. The true danger lies not in incrementally improving the 99.4% accuracy to 99.5%, but in the model's behaviour when faced with the vast, unbounded space of inputs it was never trained to handle. As the experimental results later demonstrate, the same model that performs well on expected inputs can produce incorrect classifications for OOD queries. The model's strength on its intended data can thus become a liability, as it may lead to over-confidence. This makes people less careful about the rare but potentially severe failures on unexpected inputs. The central safety challenge, therefore, is not simply improving accuracy but explicitly managing these 'unknown unknowns'.

To base these concerns in a tangible context, this use case focuses on a medical symptom checker CA. The relevance of this use case is highlighted by the recent proliferation of such applications, particularly during the COVID-19 pandemic, where they were widely deployed to the general public for self-triage [221]. For such systems, which directly influence patient health decisions, the "reliability of the decisions...can not solely rely on their accuracy" [217]. This principle mirrors the cognitive demands placed on human clinicians, who must constantly adjust their diagnostic confidence against their knowledge and the available evidence to ensure patient safety [222]. This work seeks to instill a similar capability for self-assessment within the AI itself.

### 5.2.3 Model Design and Technical Approach

To address the challenge of silent failures in DL-based CAs, the technical approach of this research moves beyond conventional model architectures to adopt a probabilistic framework capable of representing model uncertainty. While LSTMs are common and effective choice for intent classification due to their inherent ability to model the sequential nature of text data [90], a standard LSTM provides only a point estimate prediction. The innovation of this work is the extension of the standard LSTM into a Bayesian LSTM to explicitly capture and quantify the model's own uncertainty about its predictions.

This Bayesian approach was chosen specifically to address the safety vulnerability of “confident errors” (discussed in previous section). While a standard LSTM only provides a single answer, a Bayesian method allows the model to represent its uncertainty, which is essential for safely handling ambiguous or OOD inputs, a risk that cannot be mitigated by relying on standard softmax confidence alone. This directly supports contribution 2 of this thesis and provides the technical mitigation required to answer **RQ2b**.

#### 5.2.3.1 MC Dropout as a Bayesian Approximation

A full Bayesian treatment of a deep neural network is often computationally infeasible due to the need to compute posterior distributions over millions of model parameters. This work uses a practical and theoretically grounded technique known as MC dropout to approximate Bayesian inference in deep models [136]. Conventionally, dropout is a regularisation technique used only during the training phase to prevent overfitting. It works by randomly setting the outputs of a fraction of neurons to zero at each training step, effectively training an ensemble of many smaller sub-networks.

The key insight of MC dropout is to continue using this process at test time. For a single input utterance, instead of performing one forward pass with a fixed, trained network, multiple forward passes (e.g. 100) are conducted. In each pass, a different random subset of neurons is “dropped out.” This process is equivalent to sampling different models from an approximate posterior distribution of the model's parameters (weights and biases) [136, 223]. Each forward pass produces a slightly different softmax output vector, and the collection of these vectors forms an empirical posterior distribution over the class predictions. The aggregate effect of these stochastic passes is visible at deployment scale: Figure 24 shows that OOD inputs produce

markedly higher entropy than ID inputs, enabling a simple uncertainty gate. This method provides a computationally practical way to achieve a Bayesian interpretation of a standard DL model's uncertainty without fundamentally altering the training procedure or the model architecture itself.

### 5.2.3.2 Quantifying Uncertainty with Shannon Entropy

Once the posterior predictive distribution is obtained from the MC dropout process, a method is needed to summarise this distribution into a single, interpretable uncertainty score. This is achieved in two steps. First, the set of softmax vectors generated from the multiple forward passes are averaged to produce a single, final predictive probability distribution. This averaging process provides a more robust prediction than any single pass alone. Second, the Shannon entropy of this averaged probability vector is calculated [130].

Entropy is a measure of the "surprise" or disorder in a probability distribution. A distribution that is sharply peaked on a single class (e.g. [0.01, 0.98, 0.01]) has very low entropy, indicating that the model is consistently and confidently predicting that one class across the various MC samples. Conversely, a distribution that is flat or uniform (e.g. [0.33, 0.34, 0.33]) has very high entropy, indicating that the model is highly uncertain and assigning similar probabilities to multiple classes. A high entropy score thus serves as a reliable flag for model uncertainty, signalling that the input may be ambiguous or OOD [130]. Consistent with this interpretation, Figure 24 exhibits a clear separation between low-entropy ID points and high-entropy OOD points.

### 5.2.3.3 Model Architecture and Hyperparameter Justification

The precise architecture of the RNN used for the intent classification task is specified in Table 10. The model consists of an embedding layer to convert input tokens into dense vectors, a custom Bayesian LSTM layer (MCLSTM), and two dense layers for classification

A key design choice within this architecture is the dropout rate. A rate of 70% was selected, a value that was empirically determined to yield the optimal balance between maintaining high predictive accuracy and producing robust, distinguishable uncertainty signals. The choice of such a high dropout rate is not merely a hyperparameter tuning exercise; it is a deliberate decision to amplify the

Table 10: Use case 2 RNN architecture

Layer	Output Shape	Parameters
Input Layer	(None, 30)	0
Embedding	(None, 30, 50)	5,000,000
MCLSTM	(None, 64)	29,440
Dense Layer	(None, 256)	16,640
Activation	(None, 256)	0
Dropout	(None, 256)	0
Dense Layer	(None, 25)	6,425
Activation	(None, 25)	0

uncertainty signal. A high rate forces the model into a more fragile state, where its predictions become highly dependent on the specific sub-network activated during each MC sample. This induced fragility helps OOD detection.

The mechanism works as follows: for an ID input, the features learned by the network are strong and redundant. Even when 70% of neurons are deactivated, the remaining 30% are likely to contain sufficient information to consistently arrive at the correct classification across multiple MC samples. This consistency leads to a stable, peaked posterior distribution and, consequently, low entropy. For an OOD input, however, the model has no relevant learned features to rely on. Its prediction is effectively a guess. The high dropout rate ensures that each "guess" is made by a different sub-network, maximising the variance between the predictions across the MC samples. When these highly varied, almost random predictions are averaged, the resulting posterior distribution is very close to uniform. A uniform distribution has maximum entropy. Therefore, the high dropout rate acts as a mechanism to deliberately and significantly increase the entropy for OOD inputs, creating a large, detectable gap between the low entropy of ID inputs and the high entropy of OOD inputs.

Furthermore, the model implements a specialised MCLSTM layer. This layer ensures that the same dropout mask (i.e. the same set of dropped-out neurons) is applied at each time step within a given recurrent sequence during a single forward pass. This is a technically grounded choice that aligns with theoretical recommendations for applying dropout in RNNs, ensuring a more stable and meaningful application of the technique to sequential data [223]. A final dropout

layer is also applied after the first dense layer to ensure that uncertainty from the fully connected part of the network is also captured in the final estimation.

## 5.2.4 Dataset and Experimental Setup

This work presents a prototype CA consisting of uncertainty-aware NLU component to highlight the impact of incorrect intent classification on patient safety. The architecture, shown in Figure 20, illustrates how the system is designed. A user provides an input utterance, which is processed by the NLU using the Bayesian LSTM. Crucially, if the NLU is uncertain about its prediction, a safe strategy (e.g. asking the user to rephrase or connecting to a human clinician) can be triggered before the result is passed to the DM.

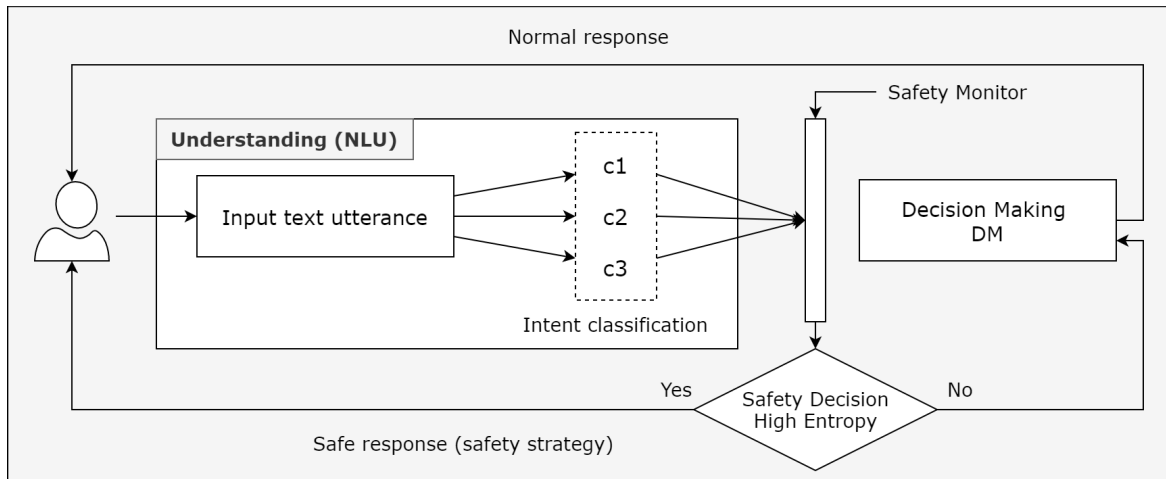


Figure 20: Use case 2 CA safety architecture

A detailed evaluation of a model intended for safety-critical applications requires a carefully designed experimental setup that can explicitly test the hypothesised failure modes. The validation strategy employed in this research moves beyond standard cross-validation to directly compare the model's behaviour on expected versus unexpected inputs, using distinct datasets for each purpose.

### 5.2.4.1 In-Distribution (ID) Dataset

The primary dataset used for training and ID evaluation is an open-source collection of medical utterances titled "Medical Speech, Transcription, and Intent" [224]. The choice of this specific dataset was deliberate. At the time of this research, it was one

of the few openly available datasets that was both clinically relevant and structured for NLU intent classification. It is composed of 6,661 text samples representing common medical symptoms (e.g. "knee pain" or "headache"), structured around 25 distinct symptom intents (classes) as shown in Figure 21. A key characteristic of this dataset is that the intents are evenly distributed. This balance is important as it prevents class imbalance from acting as a confounding variable, ensuring that high performance metrics are not merely an artefact of the model learning to predict a few majority classes well.

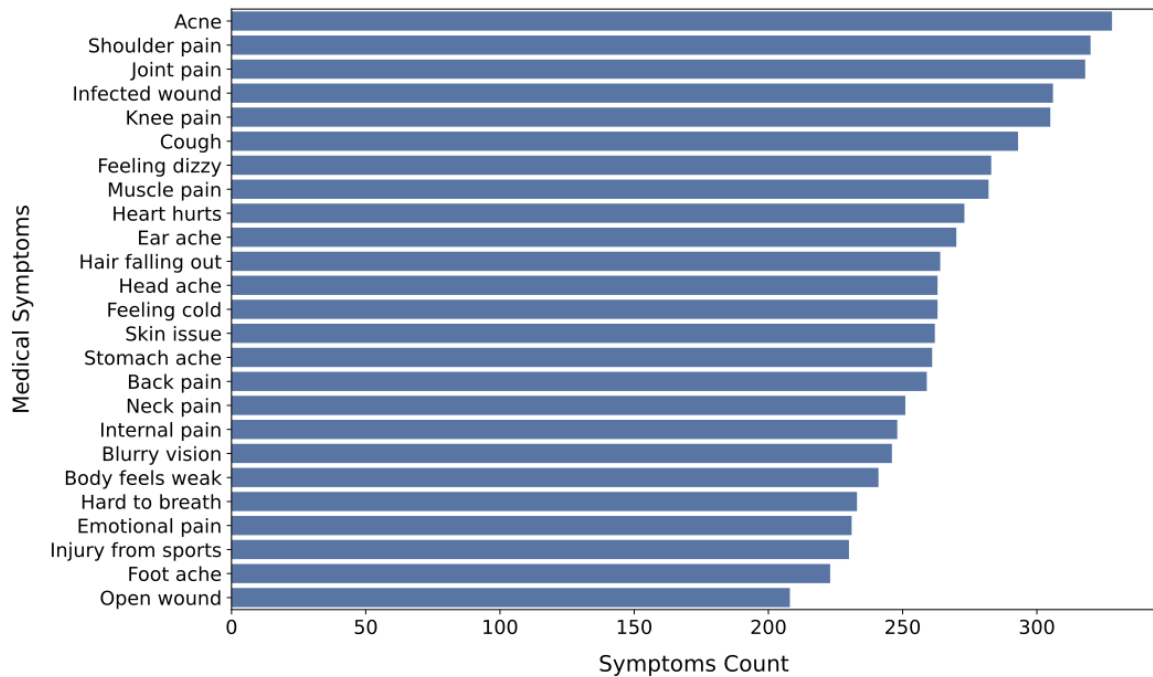


Figure 21: The distribution of medical symptoms in the dataset

#### 5.2.4.2 Data Preprocessing and Partitioning

Before being fed to the model, the raw text data underwent a standardised pre-processing pipeline. This involved several steps to clean and structure the data: (1) case normalisation, to convert all text to a consistent case (e.g. lowercase); (2) removal of punctuation and extraneous white spaces, to reduce noise; and (3) padding, to ensure all input sequences have a uniform length, as required by the neural network architecture. Following these steps, the cleaned text utterances were transformed into a numerical format suitable for the model using a one-hot encoding scheme.

The entire dataset was then partitioned using an 85:15 ratio. This resulted in a training set of 5,661 examples, which was used to learn the model parameters, and a held-out test set of 1,000 examples. This test set constitutes the ID evaluation data, used to measure the model's accuracy and its baseline uncertainty on familiar types of inputs.

#### **5.2.4.3 Out-of-Distribution (OOD) Evaluation Dataset**

To test the model's robustness and its ability to avoid misunderstanding OOD inputs, a second, entirely separate dataset was used. This OOD evaluation dataset was specifically designed to benchmark intent classification models on their ability to handle out-of-scope queries [193]. It contains 1,000 utterances that are semantically distinct from the medical domain of the training data. Examples include general questions and commands such as "am i connected to wifi" and "what casino game has the best odds".

The use of this purpose-built OOD dataset represents a detailed and useful validation strategy. A standard evaluation might only use the held-out ID test set, which would confirm the model's high accuracy on known types of inputs but would fail to probe its behaviour under the exact failure conditions it is designed to mitigate. By employing a separate dataset that is guaranteed to be OOD, the experimental design allows for a direct, controlled comparison of the model's predictive and uncertainty-related behaviour on inputs it should understand versus inputs it should not. This directly simulates the real-world scenario where a user might pose a non-medical question to the clinical CA, making the validation far more meaningful from a safety perspective. This choice shows how to validate a system intended for safe operation in an open environment.

#### **5.2.4.4 Experimental Protocol for Uncertainty Estimation**

The experimental protocol was applied consistently across both the ID and OOD test sets. For each individual utterance, the model performed 100 forward passes with MC dropout enabled (at the 70% rate). The 100 resulting softmax probability vectors were collected and then averaged to produce a final predictive distribution. The Shannon entropy of this final averaged vector was then calculated and recorded as the uncertainty score for that utterance. This process yielded a prediction and an

associated uncertainty score for a test ID utterances and OOD utterances, enabling a direct comparison of the uncertainty distributions between the two domains.

## **5.2.5 Evaluation Results**

The evaluation of the Bayesian LSTM model was conducted across two dimensions: its predictive performance on conventional metrics and its effectiveness in measuring uncertainty to distinguish between ID and OOD inputs. The results demonstrate that the model not only is an accurate classifier on known tasks but also provides a reliable mechanism for detecting when it is operating outside its domain of expertise.

### **5.2.5.1 Core Predictive Performance on ID Data**

On the ID test set, which consisted of 1,000 medical symptom utterances held out from the training data, the model achieved a high accuracy of 99.4%. The low rate of misclassification is visually confirmed by the model's confusion matrix as shown in Figure 22. The strong diagonal entries with a few off-diagonal entries back up this claim. To provide a more granular assessment of this performance, Table 11 presents the per-class precision, recall, and F1-score.

The high scores for precision, recall, and F1-score across almost all 25 intents demonstrate that the model's high overall accuracy is not skewed by performance on a few easy or majority classes. The balanced support counts confirm the lack of significant class imbalance in the test set, validating that the model has learned to effectively distinguish between all the trained medical symptom categories. These results establish a strong baseline: the model is, by conventional standards, a good classifier for its intended domain.

### **5.2.5.2 Uncertainty Estimation on ID vs. OOD Data**

The key results of this study lie in the comparison of the model's uncertainty scores on ID versus OOD data. Table 12 provides concrete examples of the model's output for ID utterances randomly selected from the test set. As expected, the model not only predicts the correct intent but does so with very low corresponding entropy scores. These low values establish the baseline for "confident" predictions on familiar inputs.

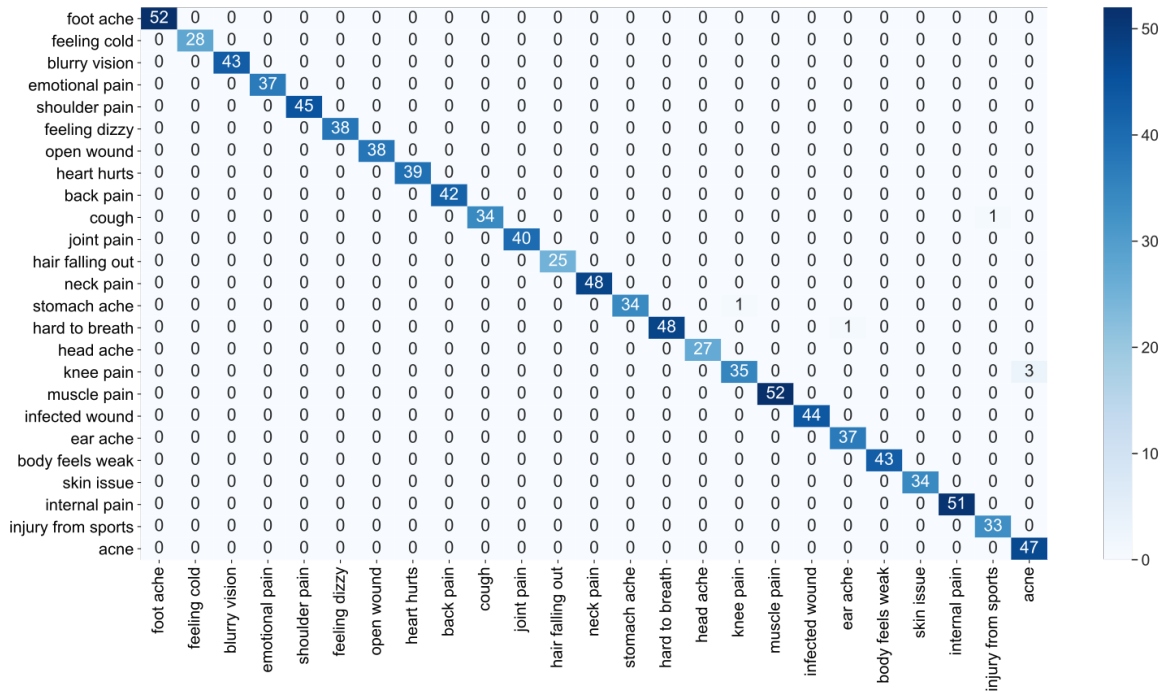


Figure 22: Confusion matrix of symptoms classification

In contrast, Table 13 presents the model’s output for OOD utterances. Here, the model is forced to make a prediction from its set of 25 known medical intents, resulting in nonsensical classifications (e.g. mapping a query about a microwave warranty to "skin issue"). Critically, however, each of these incorrect predictions is accompanied by a very high entropy score.

These examples demonstrate a successful decoupling of the model’s prediction from its confidence. A standard classifier’s output is monolithic: the highest softmax score represents both the prediction and the confidence. This work demonstrates a system that produces two distinct channels of information: a prediction channel and a certainty channel. The certainty channel provides a reliable, independent signal about the validity of the prediction channel. This decoupling is the key to building a safer system. It enables an architecture that first examines the certainty channel. If uncertainty is high, the prediction is ignored, and a safe fallback is triggered, preventing the nonsensical classification from ever reaching the DM.

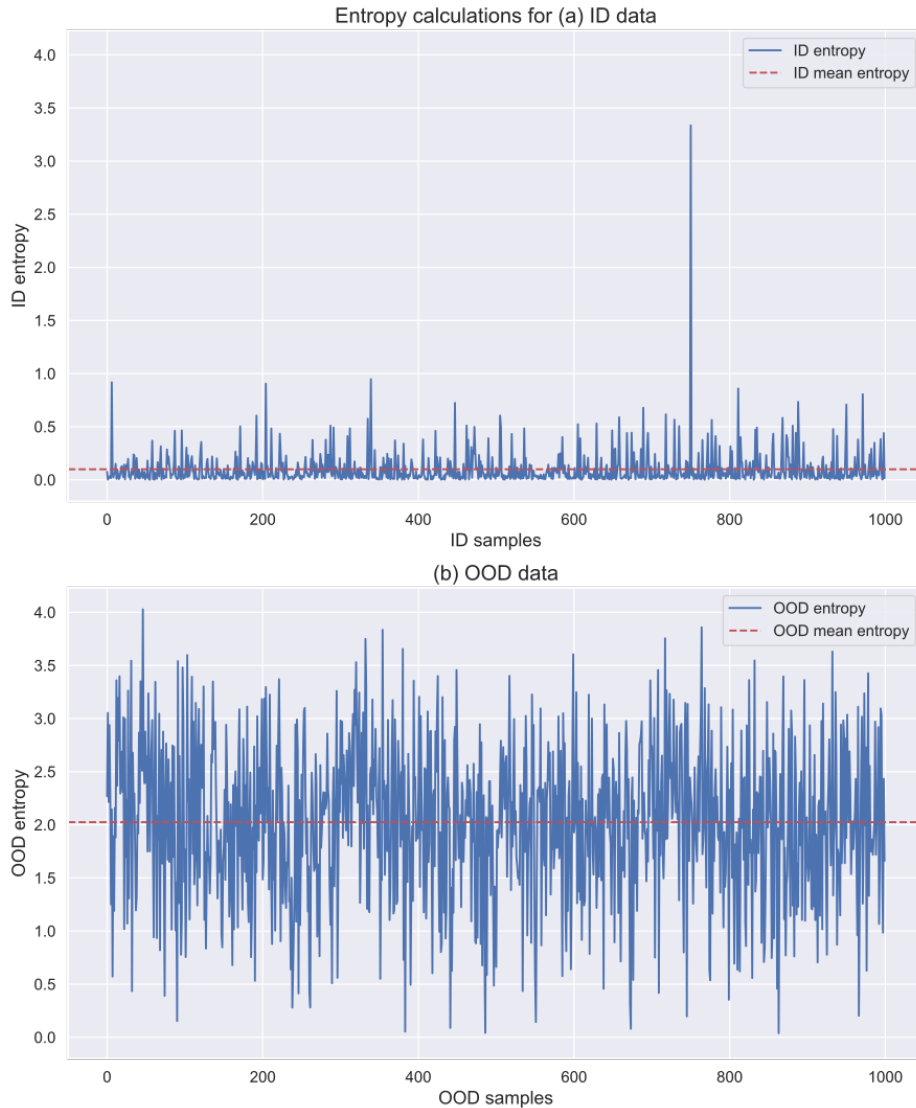


Figure 23: Entropy calculations for ID and OOD data

### 5.2.6 Comparative Analysis of Uncertainty Distributions

The qualitative difference observed in the examples is confirmed by a quantitative analysis of the entire ID and OOD test sets as shown in Figure 23. For the 1,000 ID utterances, the mean Shannon entropy was 0.098. For the 1,000 OOD utterances, the mean entropy was 2.025. This represents a greater than 20-fold difference in average uncertainty between in-domain and out-of-domain inputs. “This clear and statistically significant separation suggests that an entropy threshold can help distinguish inputs, providing a practical mechanism for detecting OOD inputs and preventing misunderstanding failures.

Table 11: Average evaluation metrics for medical symptoms

Medical Symptoms	Precision	Recall	F1-Score	Support
acne	1.000	1.000	1.000	52
back pain	1.000	1.000	1.000	28
blurry vision	1.000	1.000	1.000	45
body feels weak	1.000	1.000	1.000	37
cough	1.000	1.000	1.000	35
ear ache	1.000	1.000	1.000	38
emotional pain	1.000	1.000	1.000	38
feeling cold	1.000	1.000	1.000	39
feeling dizzy	1.000	1.000	1.000	42
foot ache	0.971	1.000	0.986	34
hair falling out	1.000	1.000	1.000	40
hard to breath	1.000	1.000	1.000	44
head ache	1.000	1.000	1.000	28
heart hurts	0.971	1.000	0.985	35
infected wound	0.980	1.000	0.990	48
injury from sports	1.000	1.000	1.000	34
internal pain	0.921	0.972	0.946	36
joint pain	1.000	1.000	1.000	27
knee pain	1.000	1.000	1.000	40
muscle pain	1.000	0.974	0.987	38
neck pain	1.000	1.000	1.000	43
open wound	1.000	1.000	1.000	50
shoulder pain	1.000	1.000	1.000	51
skin issue	1.000	0.971	0.985	34
stomach ache	1.000	0.940	0.969	50
accuracy	0.994	0.994	0.994	1000
macro avg	0.994	0.994	0.994	1000
weighted avg	0.994	0.994	0.994	1000

Table 12: Sample predictions with entropy scores for test utterances

Test Utterance (ID)	Prediction	Entropy
my head is so heavy cant think normally	head ache	0.029
i feel a burning sensation in my shoulder muscle	muscle pain	0.055
i can hardly breathe	hard to breath	0.071
i have internal pain whenever i come down with a cold	internal pain	0.327
when i'm awake in the morning i feel strange and have vertigo	feeling dizzy	0.507

Table 13: Uncertainty estimation for OOD utterances

Test Utterance (OOD)	Prediction	Entropy
am i connected to wifi	feeling cold	1.057
how much time do i have left on my 0 apr	shoulder pain	1.110
what casino game has the best odds	injury from sports	1.862
please alert me when my iphone battery falls below 30	neck pain	2.134
what is the warranty on my microwave	skin issue	2.302

The 3D scatter (Figure 24) shows the joint behaviour of softmax confidence, MC-dropout entropy, and data type. ID utterances form a compact cluster at high confidence and low entropy, whereas OOD utterances occupy a distinct region with markedly lower confidence and higher entropy. This visual pattern is consistent with the quantitative results reported earlier (mean entropy ID = 0.098 vs OOD = 2.025), and supports a simple entropy threshold as a reliable discriminator between understood and non-understood inputs. The figure illustrates that the Bayesian LSTM with MC dropout provides a usable “certainty channel” for the safety monitor to gate actions and trigger fall-backs before the DM acts on a doubtful prediction. Figure 25 explains why certain inputs carry higher uncertainty: boundary tokens between intent clusters increase entropy and reduce softmax confidence.

To qualitatively inspect what the classifier has learned, model’s word-embedding matrix was projected to two dimensions using t-SNE and colour words by their most frequent medical intent (Figure 25). Distinct intents form visible clusters, while ambiguous tokens (e.g. polysemous verbs and units) lie near cluster boundaries. This pattern is consistent with the uncertainty results: utterances dominated by boundary words tend to produce higher MC-dropout entropy and lower softmax confidence. The plot therefore offers an intuitive, representation-level explanation for when the uncertainty monitor is likely to trigger a safe fallback.

### 5.2.7 Hazard and Risk Analysis

As shown in Figure 24, the clear separation of ID and OOD points (high confidence/low entropy vs. low confidence/high entropy) provides direct evidence that the misunderstanding failure mode can be detected early via an entropy gate, before unsafe actions are taken by the DM. The safety analysis for this use case,

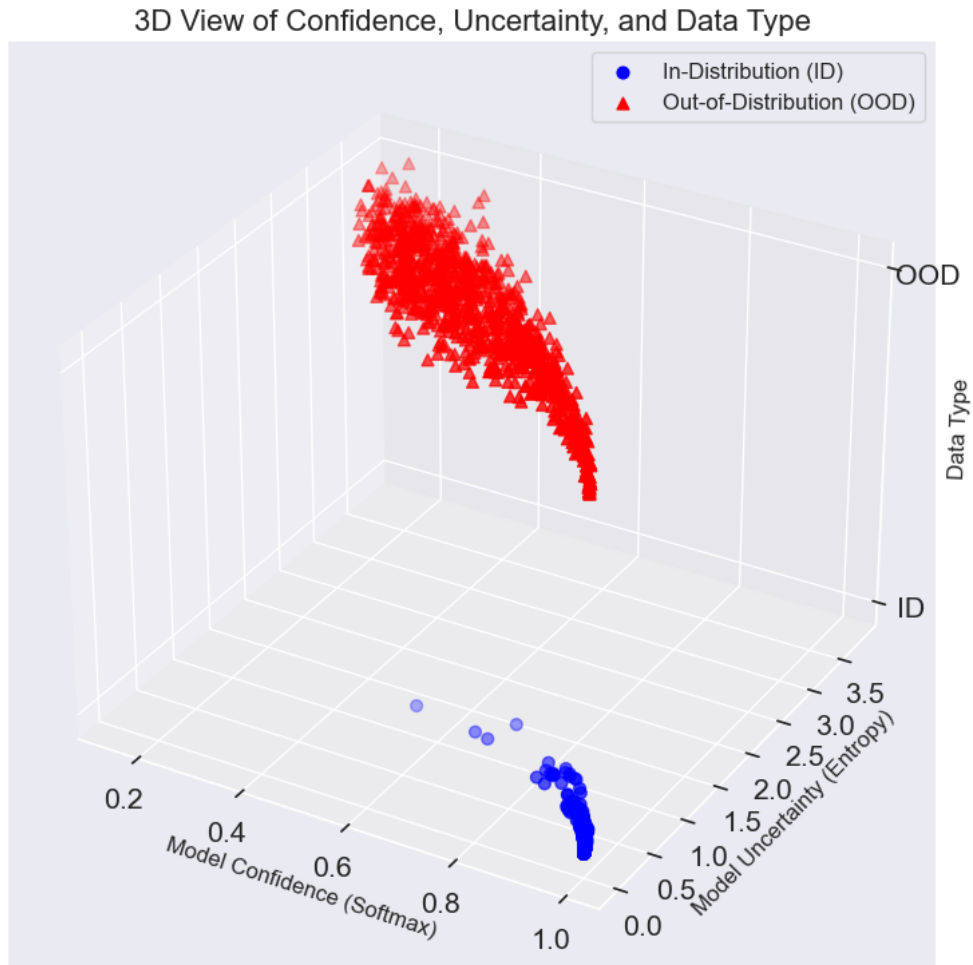


Figure 24: 3D visualisation of model confidence (softmax), predictive uncertainty (entropy), and data type (ID vs OOD). ID points cluster at high confidence and low entropy, while OOD points spread towards lower confidence and higher entropy. This separation supports uncertainty-aware thresholding and safe fallbacks in the CA.

which directly informs the mitigation strategy for RQ2b (see **RQ2b**), identifies that the most severe clinical consequence of failure is erroneous decision-making stemming from the NLU component. As identified in chapter 4 taxonomy (see section 4.5) errors propagation from NLU can cause Incorrect system action failure mode in DM. The primary hazard is not merely an incorrect classification, but a "confident error," where the deep learning model makes a wrong prediction with a high softmax value, which is particularly dangerous in a safety-critical clinical context. This risk is made worse by OOD data, where a user query falls outside the model's training domain, causing the model to fail in the misunderstanding (incorrect intent / slot) mode (see Section 4.5) instead of the safer non-understanding intent mode.

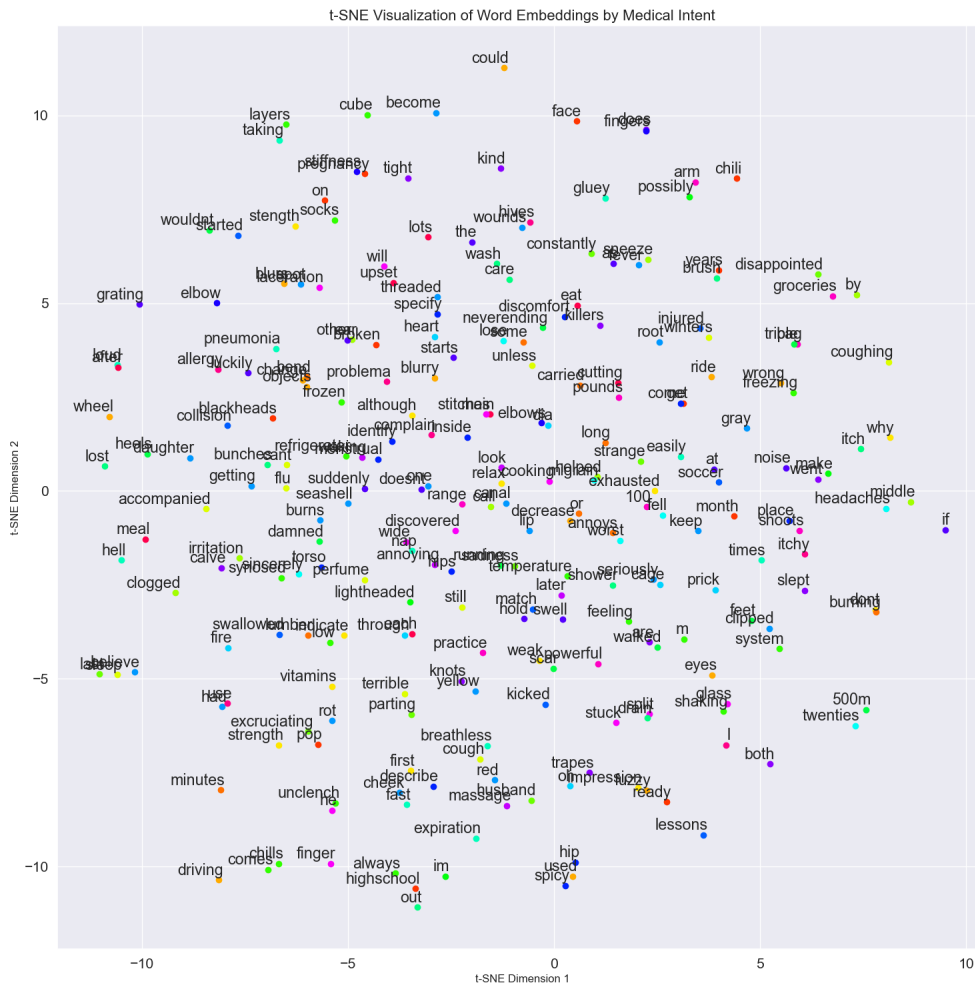


Figure 25: T-SNE visualisation of the learned word-embedding matrix, with points coloured by each word’s most frequent medical intent. Clear clusters indicate separable intents; tokens near boundaries are linguistically ambiguous and are associated with higher predictive uncertainty in our experiments.

This puts into operation the taxonomy distinction (Section 4.5): Figure 24 provides the entropy-based separator between misunderstanding and non-understanding.

The proposed approach, which forms the technical basis of contribution 2, addresses the limitations of traditional confidence metrics, which are often unreliable, and the high computational cost of existing OOD detection methods. By adopting a Bayesian LSTM with MC dropout, the method provides a computationally efficient way to quantify model uncertainty. The empirical evidence supports this approach, showing that the average model entropy for OOD data was much higher than for ID data, demonstrating that uncertainty can be used as a reliable signal to detect these potential misunderstandings and trigger a safe fallback strategy.

## 5.2.8 Safety Requirements and Monitor Design

Based on the findings from the risk analysis, a key safety requirement for the clinical CA is its ability to acknowledge and act on its own uncertainty. Instead of defaulting to a potentially incorrect prediction, the system must be able to signal when it "doesn't know." The proposed solution is the design and integration of a "Safety Monitor" module, which is strategically placed within the CA's architecture immediately after the NLU component and before the DM. The architecture, as shown in Figure 20, illustrates this safety-by-design approach.

The safety monitor's function is to use the calculated entropy from the Bayesian LSTM as a real-time control signal. When the entropy for a given utterance exceeds a pre-defined threshold, indicating high uncertainty, the monitor intercepts the standard conversational flow and triggers a pre-defined safety strategy. These strategies form a layered defence and include several options: prompting the user to rephrase their input for clarification or for critical and highly uncertain cases, initiating an immediate and seamless handover to a human clinician. Because ambiguity often arises from boundary tokens (Figure 25), clarification prompts that elicit disambiguating words shift the input toward a single intent cluster, typically lowering entropy below the operating threshold. This ensures that human expertise serves as the final safety net, shifting the system's focus from full autonomy to intelligent human-AI collaboration when safety cannot be guaranteed.

The distinct clusters in Figure 24 justify a single operating threshold on entropy for the safety monitor; the monitor treats points in the OOD region as unsafe and triggers clarification or handover.

**Code Availability** The source for use case 2 robust intent classification is openly available as Jupyter notebooks at this GitHub repository. The notebooks demonstrate MC-dropout-based inference via multiple stochastic forward passes and summarise predictive uncertainty alongside standard intent-classification metrics.

## 5.2.9 Summary

The experimental evaluation shows the effectiveness of the proposed Bayesian LSTM model, which achieved a high accuracy of 99.4% on the ID test data. However, the core finding of this work is the clear, quantifiable distinction in model uncertainty between known and unknown inputs. The average model entropy for ID data was

low (0.098), whereas for OOD data, it was much higher (2.025). This difference indicates that entropy can be used to detect OOD inputs, which tries to mitigate the misunderstanding failure mode.

The discussion supports the central argument that high accuracy on known data is insufficient for ensuring safety in clinical AI. The key vulnerability is the "confident wrong" prediction, and the results demonstrate that measuring uncertainty is a direct and effective mitigation for this hazard. The proposed method enables a principle of "graceful degradation," where the system, upon encountering high uncertainty, can default to a safe state (e.g. requesting clarification or escalating to a human) rather than failing significantly by misunderstanding the input. While the results are encouraging on the 1000 OOD samples, a key limitation is the need for further validation on larger, more diverse datasets to ensure generalisability to real-world clinical environments. Future work should also focus on incorporating the calculation of data uncertainty to improve safety assurance.

## 5.3 Use Case 3: Gestational Diabetes Mellitus CA

### 5.3.1 Overview and Architecture

Use case 3 is a text-based CA prototype developed to support patients with Gestational Diabetes Mellitus (GDM). This use case extends the application of the thesis's safety assurance methodology (contribution 3) by progressing from failure analysis to formal argumentation, directly addressing **RQ2c**: *How can we develop a safety argument for clinical CAs?* It enhances the work from use case 1 by covering an end-to-end scenario (both NLU and rule-based DM) and applies a systematic HAZOP-UML analysis, as detailed in Section 5.3.3. The highlevel architecture is depicted in Figure 26.

The primary function of this GDM CA is to engage patients in a natural language dialogue to calculate their required total daily insulin dose. This is achieved by gathering necessary clinical information, specifically the patient's current weight and trimester of pregnancy. The development of the GDM CA is based on the RASA open-source framework, a platform frequently utilised for building task-oriented dialogue systems, as seen in existing studies [225, 226]. The GDM CA's design follows a standard pipeline architecture for task-oriented CAs, which prioritises the successful completion of a user's goal [85]. As a text-based system, it relies on

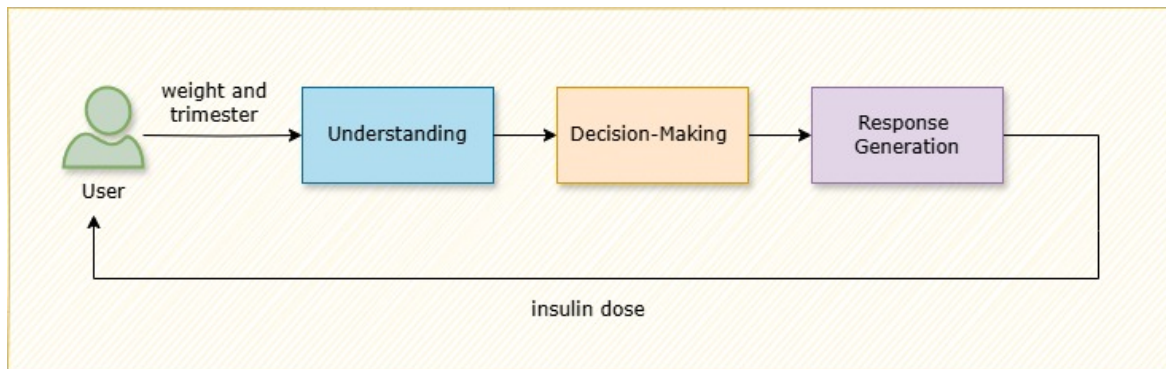


Figure 26: GDM CA architecture

three core components: NLU, DM, and NLG. Figure 27 gives an in-depth view of the intents and entities in the detailed architecture.

- **Understanding (NLU):** This initial component is responsible for interpreting the user's raw text input. It performs two critical sub-tasks: intent classification and entity extraction. Intent classification identifies the user's underlying goal (e.g. `query_dose` when the user asks for a calculation). Entity extraction pulls out specific, structured pieces of information required to fulfil the intent (e.g. `weight_entity` and `trimester_entity`). The GDM CA employs a deep learning-based Dual Intent Entity Transformer for these tasks, trained on a manually generated dataset.
- **Decision Making (DM):** The DM is the decision-making core of the agent. Upon receiving the structured output (intent and entities) from the NLU, the DM determines the CA's next action. Its logic is governed by an ensemble of policies, including a rule-based policy (a predefined set of rules) and a 'memorization policy', which considers a specified number of previous conversation turns to maintain context. A custom action is triggered within the DM to perform the final dose calculation once all necessary entities have been successfully extracted.
- **Response Generation (NLG):** After the DM decides on the next action, the NLG component formulates a response in natural language. In the GDM CA, this is accomplished by selecting an appropriate utterance from a predefined set of response templates.

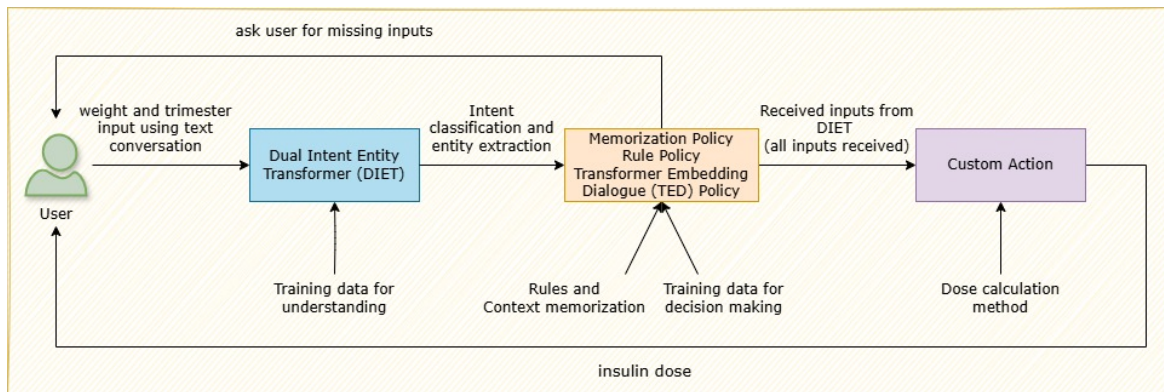


Figure 27: GDM CA detailed architecture

An analysis of this architecture reveals a critical dependency that significantly impacts system safety. The DM, which contains the deterministic calculation logic, operates entirely on the structured data it receives from the NLU. The calculation itself is a simple, verifiable function. However, the translation of the user's unstructured, natural language input into this structured format is performed by the NLU's deep learning model. This means that an error in the NLU for instance, misinterpreting a weight provided in pounds as kilograms will be passed to the DM as if it were correct. The DM will then faithfully execute its correct logic on this incorrect data, leading directly to a hazardous output. Consequently, the safety of the entire system is highly reliant on the performance and robustness of the NLU component, making it a critical control point and a potential single point of failure for data-related hazards.

### 5.3.2 Background and Clinical Context

The development and analysis of the GDM CA are motivated by the significant clinical risks associated with Gestational Diabetes Mellitus. CAs are increasingly used in healthcare to monitor chronic conditions like diabetes, offering patients an accessible and interactive means for self-management [227, 112]. GDM is a form of diabetes that emerges during pregnancy when placental hormones interfere with the body's ability to use insulin effectively.

Improper management of GDM can lead to severe adverse outcomes for both the mother and the fetus. For the mother, these complications include preeclampsia, an increased likelihood of requiring a caesarean section, preterm labour, and even miscarriage [228, 229]. Furthermore, women who experience GDM have a significantly greater risk of developing type 2 diabetes later in life [230]. For the fetus,

the risks are equally severe, including the potential for congenital malformations and macrosomia (excessive birth weight) [231].

While GDM can sometimes be managed with nutritional therapy and physical exercise, insulin therapy becomes necessary when these measures are insufficient. The GDM CA's core function is to automate the calculation of the total daily insulin dose based on established clinical guidelines, which use the patient's weight and gestational trimester as key inputs [232]. The specific formula is detailed in Table 14.

Table 14: Calculation of insulin dosing [232]

Trimester	Multiplier (k)
1st Trimester	0.7
2nd Trimester	0.8
3rd Trimester	0.9
<i>Formula: Total Daily Insulin = Weight (kg) × k</i>	

The automation of this clinically sensitive calculation introduces a fundamental shift in the risk profile. In a traditional clinical setting, a healthcare professional acts as a crucial safety barrier, capable of using their judgment to identify and question potential errors, such as a patient misstating their weight or confusion about their trimester. The GDM CA, in its current design, lacks this layer of contextual reasoning and judgment. It mechanically applies the formula in Table 14 to the data it extracts. A failure in such an automated system is not an isolated event. A flaw in the NLU model or system logic can be replicated systematically and instantaneously for every user who encounters the specific trigger condition. This transforms the risk from one of potential, isolated human errors to one of systemic, repeatable, and potentially widespread failures. This amplification of risk underscores the necessity for a systematic and rigorous hazard analysis before such a system can be considered for deployment in a clinical context [1].

### 5.3.3 Hazard Analysis and Risk Assessment

To address the safety imperatives of the GDM CA, a systematic hazard analysis was conducted using the HAZOP-UML methodology. While various safety analysis techniques exist, such as Failure Mode and Effects Analysis (FMEA) [233] and System-Theoretic Process Analysis (STPA) [234], many are not optimally suited for systems characterised by complex human-computer interaction. HAZOP-UML

was selected because it specifically addresses this gap by combining the structured, guideword-driven inquiry of Hazard and Operability (HAZOP) studies with the clear interaction modeling capabilities of the Unified Modelling Language (UML) [61]. This makes it particularly effective for analysing CAs, where failures can arise from both internal software faults and deviations in the dialogue between the user and the agent.

The first step in HAZOP-UML is to identify and model the UML use case for this work. The simplicity of GDM CA reveals only one compelling UML use case which is 'Calculate Dose'. In more complex CAs, there could be many different UML use cases which derive the system and all needs to be investigated using HAZOP-UML systematic approach.

To understand the deviations (failure modes), we then modelled the chat sequences with UML sequence diagram. A sequence diagram in UML is used to identify the events happening along a timeline. These UML tools helped in modelling the CA conversation so we can apply the HAZOP-UML guidewords on those flows. Figure 28 shows the happy path scenario for the UML use case of 'Calculate Dose'. The sequence diagram matches the intents for NLU on messages exchanged between the user and the CA. The CA asks for the trimester and weight until the user has provided both values and sends the response as calculated dose.

The core of the analysis involves systematically applying a set of guidewords (e.g. No/None, More, Less, Other than) to the attributes of each message in the sequence diagram [61]. For each potential deviation, the analysis identifies the possible causes, the immediate effect within the use case, and the ultimate real-world effect, which constitutes a clinical hazard. This entire process is documented in a detailed analysis table, providing a clear and auditable trail from a specific dialogue failure to a potential risk to patient safety.

**Outcome of the HAZOP-UML Analysis** The systematic application of the HAZOP-UML methodology proved to be an effective way of identifying potential deviations and hazards. Even for a relatively simple system like the GDM CA, where the primary hazards could be understood at a high level, the importance of the method was in its structured and thorough approach. It forced a detailed, step-by-step examination of the user-agent interaction, which uncovered subtle but critical failure modes such as the "lbs vs. kg" semantic error that a less formal analysis might have overlooked. The true value of the methodology was not just

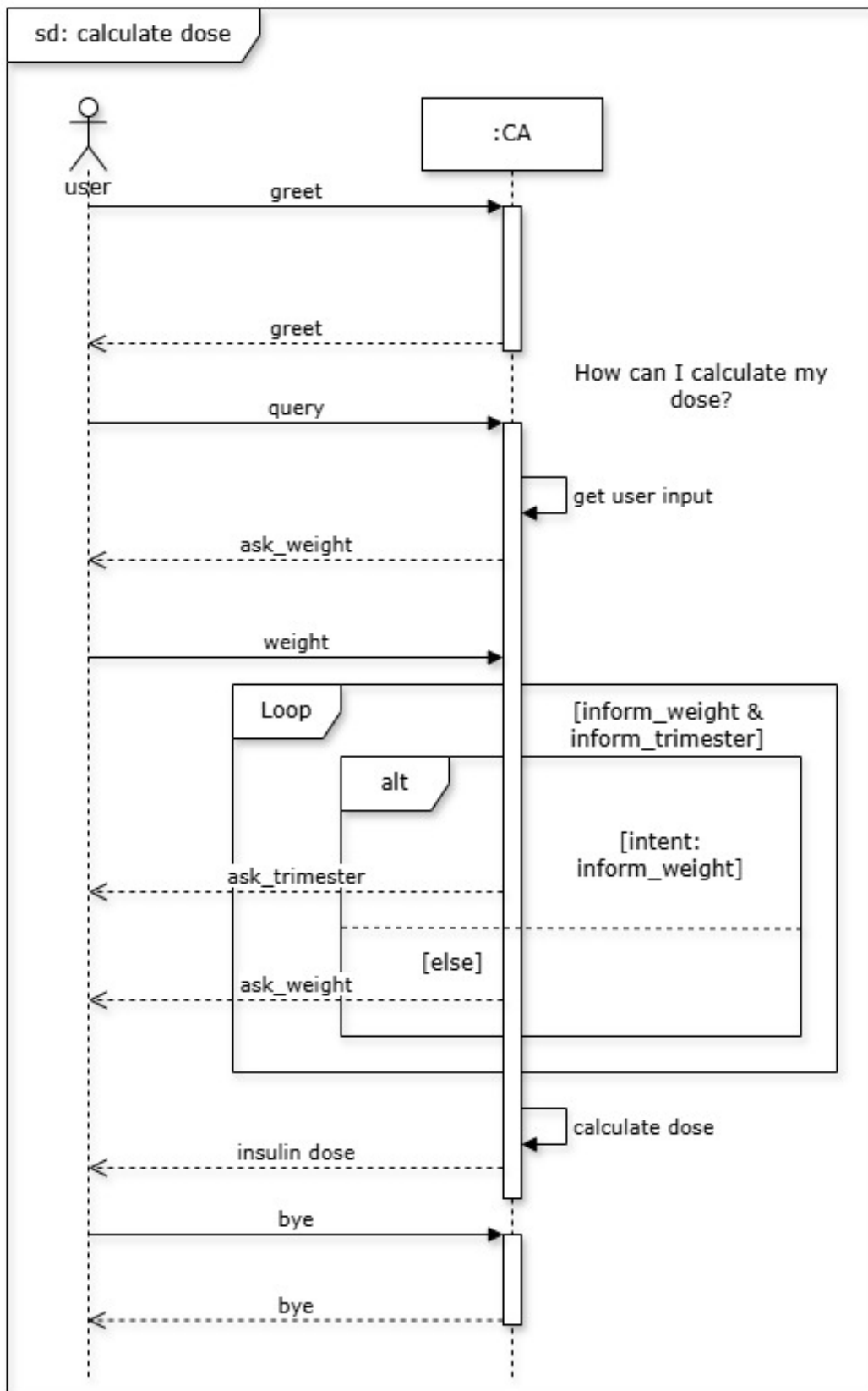


Figure 28: GDM CA sequence diagram

in identifying the hazards, but in providing a clear, traceable path from a specific deviation in the conversational flow to its potential clinical consequence, thereby creating a foundation for the derivation of targeted safety requirements.

The selected HAZOP-UML analysis for the UC01: Calculate Dose use case is presented in Table 15. The causes and recommendations against the identified deviations are listed in table 16.

### **5.3.4 Risk Assessment Prioritisation**

Following the identification of potential deviations, causes, and effects using HAZOP-UML, an initial risk assessment was conducted to prioritise these findings based on their clinical significance. This step is crucial for focusing subsequent mitigation efforts on the most critical issues identified during the analysis.

Consistent with early-stage hazard analysis practices where detailed likelihood data may be sparse, this initial assessment primarily focused on the severity of the potential 'Real World Effect' identified for each deviation (documented in Table 15). Assigning severity ratings allows for the immediate identification of deviations that could lead to unacceptable harm, regardless of their precise frequency. To ensure these ratings were clinically relevant and consistent across the thesis, the standard NHS NPSA 5x5 risk matrix framework was used, as shown in Figure 29 [235]. For example, the potential for 'Gross overdose' due to unit confusion (SD.09) was rated 'Catastrophic', while risks leading to 'hypo/hyperglycemia' (SD.06, SD.11, etc.) were rated 'Major', reflecting established clinical understanding of potential patient harm according to this standard clinical risk assessment tool.

Table 15: GDM CA: HAZOP deviations

ID	Message	Guideword	Deviation	Use Case Effect	Real World Effect	Severity
SD.06	inform_weight	Less	Misinterpreted weight (e.g. 7 instead of 70)	Logic executes on wrong value	Risk of hypo/hyperglycemia	Major
SD.07	inform_weight	As Well As	Extra symptom like "tired" misunderstood as intent	Dialogue derailment	Confusion, failure to complete task	Major
SD.09	inform_weight	Other than	Imperial unit provided, assumed as kg	Incorrect numeric value (154 pounds)	Gross overdose, severe hypoglycemia	Catastrophic
SD.11	inform_trimester	More	Wrong trimester interpreted	Logic executes on wrong value	Risk of hypo/hyperglycemia	Major
SD.15	insulin_dose	No	No dose provided	No dose calculated	Risk of hypo/hyperglycemia	Major
SD.16	insulin_dose	Less	Wrong dose suggested	Invalid dose calculation	Risk of hypo/hyperglycemia	Major

Table 16: GDM CA: HAZOP causes and recommendations

ID	Causes	Recommendations
SD.06	<ul style="list-style-type: none"> <li>NLU Understanding Failure: Incorrect entity extraction</li> <li>ML Data-Induced Fault: Insufficient training data</li> </ul>	The CA must implement an explicit confirmation step for all extracted values (weight, trimester) before calculation.
SD.07	<ul style="list-style-type: none"> <li>NLU Understanding Failure: Misclassification of secondary information</li> <li>User-Related Factor: Providing unsolicited information</li> </ul>	The dialogue policy should prioritise the completion of the active form and ignore or park secondary intents.
SD.09	<ul style="list-style-type: none"> <li>NLU Understanding Failure: Inability to handle unit conversion</li> <li>ML Data-Induced Fault: Training data lacks examples with imperial units</li> </ul>	The CA must explicitly state the required units (e.g. "Please provide your weight in kilograms") or be trained to handle unit conversion.
SD.11	<ul style="list-style-type: none"> <li>NLU Understanding Failure: Incorrect entity extraction</li> </ul>	The CA must implement an explicit confirmation step for all extracted values (weight, trimester) before calculation.
SD.15	<ul style="list-style-type: none"> <li>No dose provided</li> <li>No dose calculated</li> </ul>	DM Failure: Logic error prevents response generation.
SD.16	<ul style="list-style-type: none"> <li>DM Failure: Error in calculation logic</li> <li>Error Propagation: NLU error passed to DM</li> </ul>	The calculation logic in the DM must be rigorously unit and integration tested. All inputs to the calculation must be validated.

### 5.3.5 Linking HAZOP-UML Deviations to the Failure Mode Taxonomy

The HAZOP-UML analysis in **Table 15** and **Table 16** identified several critical deviations. The failure mode taxonomy defined in chapter 4 (see section 4.5) provides a direct framework for interpreting these deviations and tracing them to their root causes, demonstrates the practical link between taxonomy and safety analysis. Two examples from the analysis highlight this synergy:

Likelihood	Very High	3	4	4	5	5
	High	2	3	3	4	4
	Medium	2	2	3	3	4
	Low	1	2	2	3	4
	Very Low	1	1	2	2	3
		Minor	Significant	Considerable	Major	Catastrophic
		Severity				

5	Unacceptable level of risk.
4	Mandatory elimination or control to reduce risk to an acceptable level.
3	Undesirable level of risk. Attempts should be made to eliminate the hazard or implement control measures to reduce risk to an acceptable level. Acceptable only when further reduction is impractical.
2	Acceptable where cost of further reduction outweighs benefits gained or where further risk reduction is impractical.
1	Acceptable, no further action required.

Figure 29: NHS NPSA 5x5 Risk classification matrix [235]

### 1. The 'Unit Confusion' Deviation (SD.09)

- **HAZOP-UML Deviation:** The 'Other than' guideword applied to the 'inform\_weight' message identified the deviation "Imperial unit provided, assumed as kg" (Table 15).
- **Taxonomy Classification:** This is a clear and critical instance of the 'Misunderstanding (incorrect intent / slot)' failure mode defined in the taxonomy (section 4.5). The system fails to extract the correct semantic value for the 'weight\_entity'.
- **Causal Analysis (from Taxonomy):** The HAZOP table (Table 16) lists the cause as an "NLU Understanding Failure" due to a "ML Data-Induced Fault: Training data lacks examples with imperial units". This maps directly to the 'Insufficient or Unrepresentative Data' fault identified in the taxonomy (section 4.5.2), which explicitly notes that a model will be brittle if its training data lacks sufficient diversity (e.g. in formats, dialects, or in this case, units).

## 2. The 'Extra Symptom' Deviation (SD.07)

- **HAZOP-UML Deviation:** The 'As Well As' guideword on 'inform\_weight' identified "Extra symptom like 'tired' misunderstood as intent," leading to "Dialogue derailment" (Table 15).
- **Taxonomy Classification:** This deviation maps to two taxonomy categories, showing an error propagation. First, it's an NLU 'Misunderstanding' failure (section 4.5) as the NLU misclassifies the input. Second, it triggers a DM 'Incorrect system action' (section 4.5), as the dialogue policy derails instead of handling the unexpected information (which is an incorrect action).
- **Causal Analysis (from Taxonomy):** The cause is listed as "User-Related Factor: Providing unsolicited information" (Table 16), which aligns with the taxonomy's 'User' category, specifically 'Interaction behaviour' (section 4.5.4.1). This section describes how a mismatch between the user's mental model and the system's capabilities (e.g. providing extra information) can create a fault condition.

This explicit mapping from HAZOP-UML deviation to taxonomy failure mode and causal fault confirms the taxonomy's utility as an analytical tool during hazard identification.

### 5.3.6 Linking Analysis to Clinical Hazards

The detailed findings from the HAZOP-UML analysis and the interpretation provided by the failure mode taxonomy are combined into a high-level clinical hazard log. This log consolidates the identified "Real World Effects" into a prioritised list of hazards that must be controlled to ensure patient safety. This step is crucial for abstracting the numerous low-level deviations into a manageable set of top-level risks that will drive the development of the safety case. Table 17 presents this consolidated hazard log.

The analysis now clearly distinguishes between two primary clinical hazards:

- **H01 (Incorrect Dose Provided):** This is the most severe hazard, rated as **Catastrophic** due to the worst-plausible outcome of a gross overdose from unit confusion (SD.09). It consolidates all deviations that lead to the system calculating and presenting a numerically incorrect value.

Table 17: GDM CA: clinical hazard log

Hazard ID	Hazard Description	Severity	Analysis Ref.
H01	Incorrect insulin dose provided (underdose or overdose).	Catastrophic	SD.06, SD.09, SD.11, SD.16
H02	Failure to provide a dose to the patient.	Major	SD.07, SD.15

- **H02 (Failure to Provide a Dose):** This hazard is rated as **Major**. It consolidates deviations where the system fails to provide a dose at all, either due to an internal logic or timeout error (SD.15), or because the dialogue breaks down and the task cannot be completed (SD.07).

### 5.3.7 Deriving Safety Requirements from the Analysis

Following the identification and prioritisation of hazards, the next step is to derive a set of formal safety requirements. These requirements are not arbitrary; they are specific, verifiable controls designed to directly mitigate the risks identified in the HAZOP-UML analysis and summarised in the hazard log. Each requirement is traceable to the hazard(s) it is intended to control, forming the logical backbone of the safety argument.

The requirements are derived directly from the "Safety Recommendations" column in Table 16. For instance, the recommendation to "implement an explicit confirmation step" is formalised into a testable system-level requirement. This creates a multi-layered defence strategy where risks are managed through a combination of dialogue design, NLU model performance, and data quality. Table 18 presents the derived safety requirements for GDM CA. The rationale behind the safety requirements is discussed below:

Table 18: GDM CA: derived safety requirements

ID	Requirement	Hazard(s) Mitigated
SR-SYS-01	The system shall explicitly confirm all extracted clinical values (weight, trimester) with the user and require a positive affirmation before proceeding with the dose calculation.	H01
SR-SYS-02	The system shall explicitly state the required units for weight input (e.g. "Please provide your current weight in kilograms.").	H01
SR-DM-01	The insulin dose calculation logic within the Dialogue Manager shall be rigorously verified through unit and integration testing against a set of predefined test cases covering all boundary conditions.	H01, H02
SR-NLU-01	The NLU model shall achieve an F1-score of $\geq 95\%$ for the extraction of <code>weight_entity</code> and <code>trimester_entity</code> on a balanced and representative validation dataset.	H01
SR-NLU-02	The dialogue policy shall be designed to prioritise the completion of the active form (dose calculation) and safely handle extraneous user input without derailing the primary task.	H02
SR-NLU-03	The NLU model shall robustly manage its prediction uncertainty (e.g. by detecting OOD inputs or low-confidence predictions).	H01
SR-DATA-01	The training and validation datasets for the NLU model must include representative examples of different units (metric and imperial) to improve the model's ability to distinguish them or support future conversion features.	H01

**SR-SYS-01** This dialogue-level control is the primary mitigation for NLU errors identified in SD.06 and SD.11, where incorrect values are extracted. It acts as

a critical safety barrier, giving the user a chance to catch and correct system misunderstandings.

**SR-SYS-02** This proactive design control directly mitigates the catastrophic risk of unit confusion (SD.09), where an input in pounds could be misinterpreted as kilograms, leading to a gross overdose.

**SR-DM-01** This requirement addresses both the risk of a calculation error (SD.16), which leads to H01, and the risk of the system failing to provide a dose entirely due to a logic error or timeout (SD.15), which leads to H02.

**SR-NLU-01** This sets a high, quantifiable performance target for the NLU model, reducing the initial likelihood of the errors seen in SD.06, SD.09, and SD.11. The residual risk is managed by SR-SYS-01.

**SR-NLU-02** This requirement directly addresses the deviation of dialogue derailment (SD.07). By ensuring the system remains focused on the safety-critical task, it prevents a conversational breakdown that would lead to the clinical hazard of failing to provide a dose (H02).

**SR-NLU-03** This requirement addresses the residual risk of a "confidently wrong" NLU prediction, which could bypass the user confirmation control (SR-SYS-01). It provides a final defence against plausible but incorrect values (e.g. SD.09) by requiring the model to robustly manage its own uncertainty, thereby mitigating the catastrophic hazard H01.

**SR-DATA-01** Data quality is the foundation of ML model safety. This requirement ensures the model is trained on data that reflects potential real-world inputs, specifically mitigating the unit confusion risk from SD.09.

### **5.3.8 A Structured Safety Case for the GDM CA**

The safety case for the GDM CA is a formal, structured argument that combines all preceding safety activities from the HAZOP-UML analysis to the derivation of

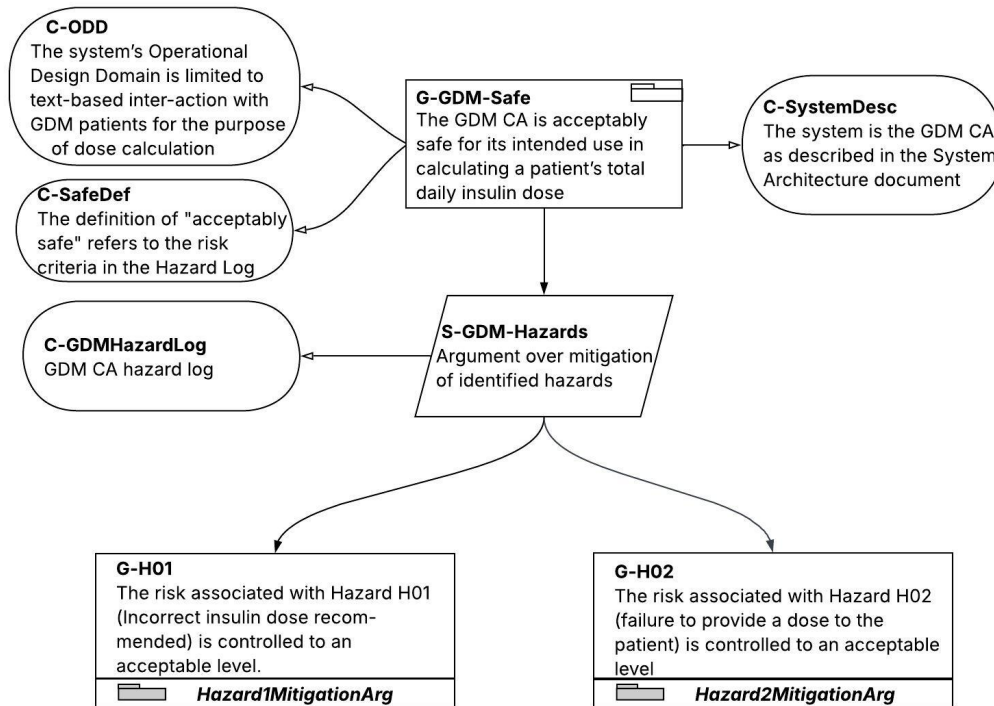


Figure 30: GDM CA safety case top level argument

safety requirements into a coherent and defensible justification that the system is acceptably safe for its intended use. To ensure clarity and rigour, the argument is represented using Goal Structuring Notation (GSN), a standard graphical method for constructing safety arguments.

The argument is built around a primary, risk-based strategy: the top-level claim of overall system safety is supported by demonstrating that all clinical hazards identified in the final Hazard Log (Table 17) have been controlled to an acceptable level through the implementation and verification of specific, traceable safety requirements.

### 5.3.8.1 Top-Level Safety Argument

The safety case begins with a single, overarching goal that makes the primary claim of safety, bounded by a defined operational context. The GDM CA top level safety argument is shown in Figure 30.

**G-GDM-Safe (Top-Level Goal):** The GDM CA is acceptably safe for its intended use in calculating a patient’s total daily insulin dose.

This goal is made within the following context:

**C-SysDesc (Context):** The system is the GDM CA as described in the System Architecture document (Section 5.3.1).

**C-ODD (Context):** The system's Operational Design Domain is limited to text-based interaction with GDM patients for the purpose of dose calculation.

**C-SafeDef (Context):** The definition of "acceptably safe" refers to the risk criteria in the Hazard Log (Table 17), where all high-priority risks must be mitigated to a tolerable level.

**C-GDMHazardLog:** The clinical hazard log for GDM CA after the application of HAZOP-UML analysis as shown in Table 17.

**Strategy (S-GDM-Hazards):** The primary strategy is an *Argument over the control of all identified clinical hazards*. This classic hazard-based approach decomposes the top-level goal into a set of sub-goals, one for each hazard in the Hazard Log. This ensures that the argument is comprehensive and directly addresses all known sources of potential harm.

#### 5.3.8.2 Argumentation over Hazard H01 (Incorrect Dose)

To illustrate the argumentation strategy, this section details the argument for the highest-priority hazard, H01 as shown in Figure 31.

**G-H01 (Sub-Goal):** The risk associated with Hazard H01 (Incorrect insulin dose recommended) is controlled to an acceptable level.

**S-H01-Control (Strategy):** Argue that H01 is controlled by satisfying a set of diverse and independent safety requirements that form an in-depth safety analysis.

This strategy decomposes the goal G-H01 into several sub-goals, each corresponding to a specific safety requirement. For example:

- **G-SR-DM-01 (Sub-Goal):** The requirement for semantic validation in the custom action (SR-DM-01) is satisfied.
  - **Sn-DM-Test (Solution/Evidence):** Custom Action Unit Test Report, demonstrating that the action correctly rejects values outside the plausible clinical range.

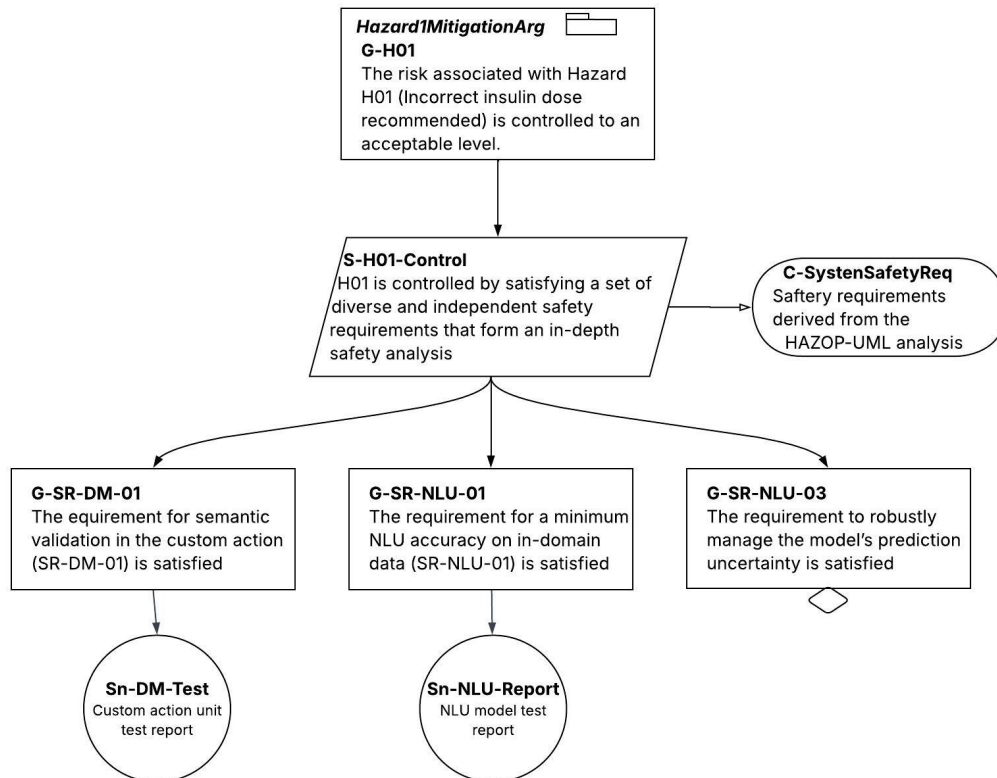


Figure 31: GDM CA safety case hazard 1 mitigation argument

- **G-SR-NLU-01 (Sub-Goal):** The requirement for a minimum NLU accuracy on in-domain data (SR-NLU-01) is satisfied.
  - **Sn-NLU-Report (Solution/Evidence):** NLU Model Performance Test Report (e.g. 'intent\_report.json').

**Undeveloped Goal (NLU Uncertainty):** While the safety requirements provide multiple layers of protection, the safety analysis identified a key residual risk: a "confidently wrong" NLU prediction could still, in theory, bypass these controls if the incorrect value falls within the plausible range. Therefore, a crucial goal remains undeveloped:

**G-SR-NLU-03 (Undeveloped Sub-Goal):** The requirement to robustly manage the model's prediction uncertainty is satisfied.

As demonstrated in use case 5.2, standard NLU models can be "confidently wrong," and specialised techniques are required to create a reliable uncertainty signal. As

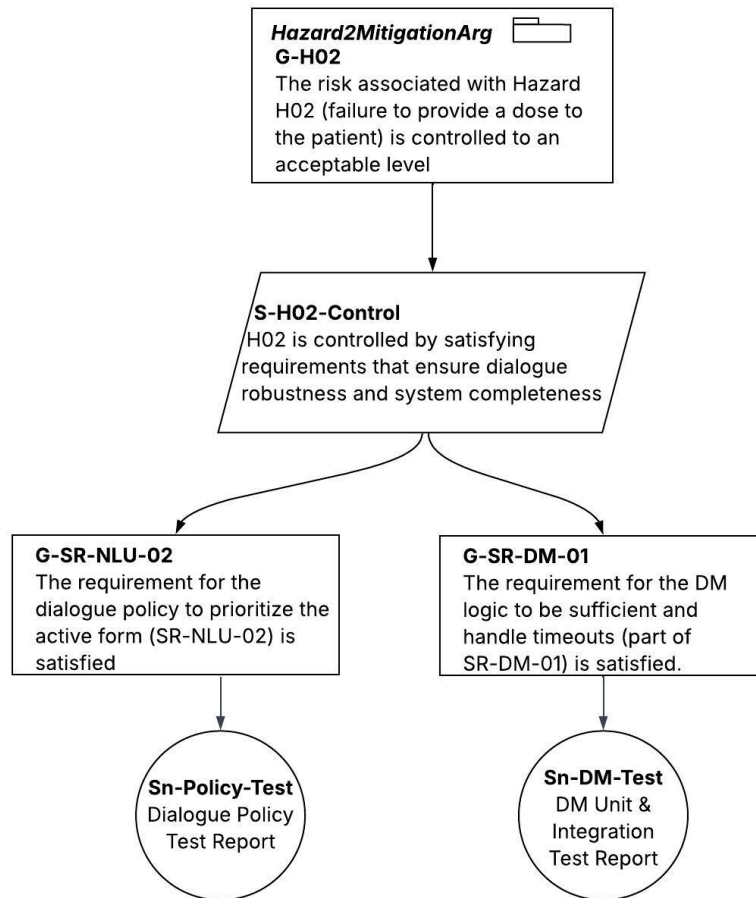


Figure 32: GDM CA safety case hazard 2 mitigation argument

this mechanism has not been implemented in the GDM CA, this safety goal cannot be satisfied. The risk of a "confidently wrong" NLU prediction that bypasses the user confirmation step remains.

### 5.3.8.3 Argumentation over Hazard H02 (Failure to Provide a Dose)

The argument for the major hazard H02 is shown in Figure 32

**G-H02 (Sub-Goal):** The risk associated with Hazard H02 (failure to provide a dose to the patient) is controlled to an acceptable level.

**S-H02-Control (Strategy):** Argue that H02 is controlled by satisfying requirements that ensure dialogue robustness and system completeness.

- **G-SR-NLU-02 (Sub-Goal):** The requirement for the dialogue policy to prioritise the active form (SR-NLU-02) is satisfied. This mitigates the risk of dialogue derailment (SD.07).
  - **Sn-Policy-Test (Solution/Evidence):** Dialogue Policy Test Report.
- **G-SR-DM-01 (Sub-Goal):** The requirement for the DM logic to be complete and handle timeouts (part of SR-DM-01) is satisfied. This mitigates the risk of an omission failure (SD.15).
  - **Sn-DM-Test (Solution/Evidence):** DM Unit & Integration Test Report.

This safety case provides a transparent, traceable, and defensible argument for the safety of the GDM CA. Most importantly, by explicitly identifying the undeveloped goal related to NLU uncertainty, it clearly defines the boundaries of the current safety argument and provides a rigorous, evidence-based motivation for the next phase of research into uncertainty-aware safety monitors.

**Code Availability** The source for use case 3 GDM CA is openly available as RASA project at this GitHub repository.

## 5.4 Summary

This chapter has detailed a series of use cases designed to demonstrate the practical application of the safety assurance methodology. The progression of these cases enables a structured exploration of how safety risks in CAs can be identified, mitigated, and justified, moving from a simple prototype to a more mature, task-oriented system.

The chapter began with an exploratory case study of a symptom-checking CA. This initial use case served as a probe to identify fundamental failure modes, demonstrating through both qualitative and quantitative analysis how vulnerabilities in the NLU component can lead to clinically significant hazards. The "lessons learned" from this analysis directly motivated the need for a more systematic approach.

The second use case presented a deep-dive into a specific technical mitigation, demonstrating an uncertainty-aware intent classification model. This provided a targeted solution to the "confidently wrong" failure mode identified as a key risk in the initial prototype.

Finally, the third use case examined a more mature CA for Gestational Diabetes Mellitus (GDM). It showed how a more rigorous analysis technique (HAZOP-UML) could be applied to a system that already incorporated baseline safety controls. This analysis uncovered more subtle, second-order risks and led to the derivation of a more sophisticated set of safety requirements. Crucially, it demonstrated how a safety case can be partially satisfied while explicitly identifying "undeveloped goals" thereby providing a direct and evidence-based link to the necessity of the uncertainty-aware mitigation developed in use case 5.2.

Together, these use cases provide a grounded and evolving narrative of how safety assurance can be put into practice for CAs, highlighting the trade-offs between different analysis methods and the importance of adapting the safety case to system complexity.

# Chapter 6

## Case Study: Safety Analysis of a Real-World Clinically Deployed CA

This chapter presents the safety analysis of a real-world clinical Conversational Agent (CA), Hinata<sup>1</sup>, which is deployed across various NHS Trusts. This case study provides the main demonstration of contribution 3, which is a **systematic safety assurance methodology**. By applying this methodology in full to Hinata, the chapter provides a practical answer to key research questions, showing how to analyse and mitigate hazards (**RQ2a**) and how to develop a safety argument using GSN (**RQ2c**). Using an evaluative case study approach, this chapter investigates the system's safety characteristics and develops a corresponding safety argument.

### 6.1 Background and Deployment Context

Hinata is a clinically deployed, voice-based CA developed to automate follow-up care for patients after cataract surgery, the most commonly performed surgical procedure within the NHS. Unlike common conversational platforms that require a dedicated application to be installed, Hinata is accessible to all patients via a standard phone call. Patients do not need smartphones, computers, or connectivity; they simply respond to the automated voice assistant as they would to a clinician on the phone. Hinata can contact multiple patients at set times and frequency without requiring clinical staff, allowing it to operate at scale. The system engages each patient in a structured conversation focused on post-operative recovery, identifying concerning symptoms and prompting escalation when appropriate.

The design of Hinata reflects a growing interest in integrating AI-driven automation into low-risk but resource-intensive areas of healthcare delivery. In this

---

<sup>1</sup>Pseudonym used; not the real name.

case, the agent replaces routine follow-up calls typically carried out by nurses or optometrists, thereby freeing up clinical time for more urgent or complex cases. The system specifically focuses on five critical symptoms relevant to post-cataract care: significant eye redness, severe pain, visual disturbances, new floaters, and flashing lights. By standardising how these symptoms are checked and recorded, Hinata supports early identification of complications such as retinal detachment or infection. Its ability to operate at scale, without the need for constant human oversight, offers a potential model for sustainable post-operative care in the NHS.

## 6.2 System Architecture and Functional Overview

The Hinata system shares the same underlying architecture as the previously described CAs utilising the RASA framework, with a modular pipeline for intent and entity extraction, and dialogue management. The key distinction lies in its voice-based interface: rather than relying on text input, Hinata initiates outbound phone calls and receives spoken input from patients. This audio is transcribed using an Automatic Speech Recognition (ASR) engine and passed to the existing Natural Language Understanding (NLU) pipeline. After that point, the dialogue proceeds in the same way as in a standard pipeline CA.

The system includes a clinical supervisor to review calls where the system detects that the conversation flow has deviated (e.g. when NLU confidence scores fall below a defined accepted threshold). Conversation logs with the patient consent are used by the AI development team to retrain and improve system performance, particularly that of the ML model. Figure 33 presents an abstracted view of the system's high-level architecture, illustrating its layered structure and key interfaces. Together, these architectural features determine where hazards can originate, and these are analysed in section 6.4.1.

The clinical pathway for Hinata is sequential. The first step is to correctly identify the patient. If successful, Hinata proceeds to ask the patient about the key cataract symptoms. At the end of the call, based on the collected information, Hinata decides whether the patient needs to be referred back to the hospital for a clinical review.

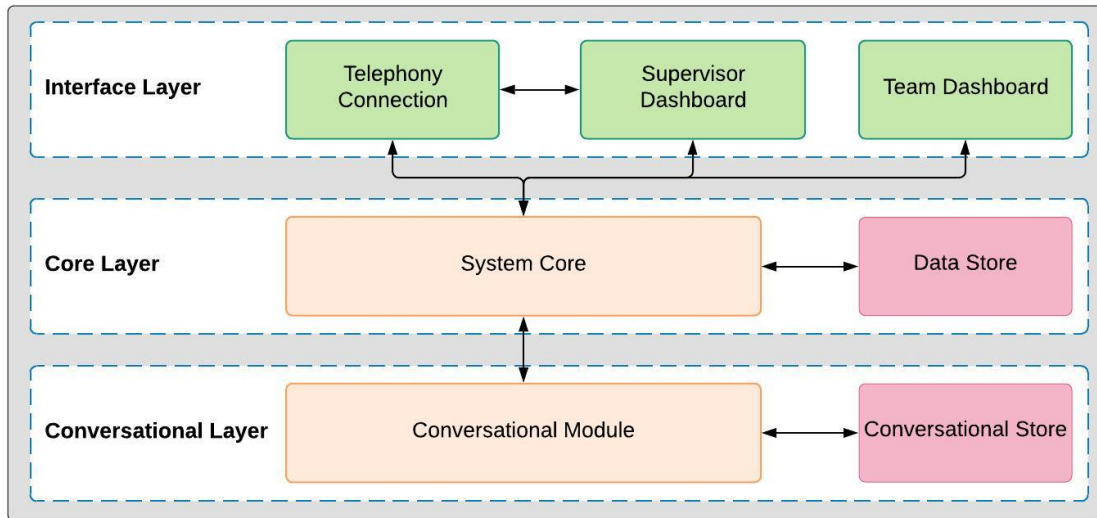


Figure 33: Hinata abstract architecture

## 6.3 Case Study Design and Methodology

This section details the methodological framework designed and executed for the Hinata case study. The primary objective was to apply and validate the proposed safety assurance framework on a real-world clinical CA. The entire process is grounded in the established principles of case study research as articulated by Yin [145], ensuring a structured, reliable, and defensible investigation.

### 6.3.1 Research Approach: An Evaluative Case Study Framework

The central research questions of this thesis, concerning how to systematically develop and validate safety cases for clinical CAs, are well-suited to the case study research method. This method is preferred when a *how* or *why* question is being asked about a contemporary set of events over which the investigator has little or no control [145]. An evaluative case study design was chosen because the purpose of this chapter is to apply the proposed safety assurance methodology to a real-world clinical CA and assess the resulting safety artefacts, including the hazard analysis, safety requirements, and GSN safety case. The study therefore evaluates the practical applicability and credibility of the methodology, while also examining how its individual steps support hazard identification, requirements derivation, and safety argument construction.

## 6.3.2 Designing the Case Study: The Five Components of Research Design

In accordance with Yin's [145] framework, the research design is explicitly built upon five critical components to ensure a logical and defensible structure. Table 19 provides an overview of the study design.

### 6.3.2.1 The Case Study's Questions

The specific questions that guide this case study from high-level thesis questions into answerable, case-specific inquiries are as follows:

- **CS-RQ1:** How complete and realistic are the hazards and failure modes identified by the systematic safety analysis, according to key technical and clinical stakeholders?
- **CS-RQ2:** Are the derived safety requirements and mitigation strategies, particularly those for ML, deemed sufficient and appropriate for the identified risks?
- **CS-RQ3:** How effective is the formal safety case structure (using GSN) and its supporting evidence in providing a convincing and trustworthy argument for system safety?
- **CS-RQ4:** What is the practical contribution of the safety case methodology and its artefacts towards building stakeholder confidence and facilitating communication, and what is the potential scope for its generalisation?

### 6.3.2.2 The Case's Propositions

Propositions are essential in case study design because they direct attention to what should be examined within the scope of the study [145]. In this evaluative case study, the propositions are derived from the study's objective of applying and assessing the proposed safety assurance methodology in a real-world clinical CA context.

- **P1:** A systematic hazard analysis process can produce a set of hazards and failure modes that key technical and clinical stakeholders will deem both realistic and sufficiently complete.

Table 19: Hinata: overview of the case study research design

Case-study RQ	Guiding proposition(s)	Primary data source(s)	Stakeholder group(s)
CS-RQ1: How complete and realistic are the identified hazards and failure modes?	P1: Stakeholders can meaningfully assess realism and completeness.	Semi-structured interviews; hazard-log review	Technical & Clinical
CS-RQ2: Are the derived safety requirements and mitigations (especially for ML) sufficient and appropriate?	P2: Explicit ML requirements are perceived as sufficient and appropriate mitigations.	Semi-structured interviews; artefact review (requirements list)	Technical & Clinical
CS-RQ3: How effective is the formal safety case (GSN) in providing a trustworthy argument?	P3: The GSN structure is a useful tool for trust-building.	Semi-structured interviews; artefact review (GSN diagrams)	Technical & Clinical
CS-RQ4: How effective is the safety case methodology for building confidence, facilitating communication, and enabling generalisation in practice?	P4: The methodology is a practical framework that facilitates communication and has potential for generalisation.	Semi-structured interviews; direct & participant observation	Technical & Clinical

- **P2:** Explicitly defined safety requirements, including those specific to ML, are perceived by diverse stakeholders as sufficient and appropriate mitigations for the key risks of a clinical CA.
- **P3:** A formal safety case structure (using GSN) is a convincing and practically valuable tool for communicating safety arguments and building trust among both technical and clinical personnel.
- **P4:** The systematic safety case methodology is perceived by stakeholders as a practical process that facilitates communication between technical and clinical

teams and provides a generalisable framework for assuring other, similar AI-driven clinical systems.

### **6.3.2.3 The Case's Unit of Analysis**

A precise definition of the "case," or unit of analysis, is essential. For this case study, the case is defined as:

*The process of applying a systematic safety assurance methodology, from hazard analysis to formal safety case construction, to a mature, deployed clinical CA, and the subsequent evaluation of that process.*

The boundaries of this case are firmly set on the application of the methodology and stakeholder perceptions of it, excluding unrelated business or clinical operations.

### **6.3.2.4 Linking Data and Interpreting Findings**

Data were linked to the findings in two stages. First, system and project evidence, including documentation, source code access, existing hazard material, workshops, meetings, and observation, informed the safety analysis and the production of the safety artefacts reported in section 6.4. These artefacts included the SHARD analysis, hazard log, risk matrix, safety requirements, and GSN safety case. Second, stakeholder interview data were used to evaluate these artefacts against the case study questions and propositions. The interpretation therefore linked the artefacts produced by the methodology with stakeholder feedback on their realism, adequacy, credibility, and practical value.

### **6.3.3 Data Collection and the Role of the Protocol**

The case study evidence was collected through a formal case study protocol. The protocol organised the sources of evidence needed to understand the Hinata CA, conduct the safety analysis, and evaluate the safety artefacts produced during the study. The evidence base consisted of system and project material, researcher involvement in workshops and meetings, and stakeholder interview data.

The study used multiple sources of evidence to build a structured understanding of the case:

1. **Documentation and archival records:** access to the system's source code repository, historical design documents, and pre-existing hazard logs.
2. **Interviews:** two online semi-structured interview sessions with three participants: one technical stakeholder involved in the system's design and two clinical stakeholders familiar with the clinical follow-up pathway. The technical interview lasted approximately 40 minutes, and the clinical interview lasted approximately 30 minutes.
3. **Direct and participant observation:** on-site workshops and meetings with the development team, allowing first-hand observation of the system context and direct participation in safety analysis discussions.
4. **Safety artefacts:** the SHARD analysis, risk assessment, safety requirements, and GSN charts produced during the study and discussed during the stakeholder interviews.

### 6.3.4 Data Analysis Strategy

The interview transcripts were analysed manually using a deductively informed thematic analysis, following Braun and Clarke's distinction between inductive and theoretical or deductive approaches to thematic analysis [215]. The analysis was guided by the case study protocol, the interview questions in appendix C, and the case study propositions in table 19. The themes were developed around the main evaluation concerns of the case study: hazard realism, ML-related safety concerns, the usefulness of the GSN safety case, and the practical value of the methodology. The researcher conducted the analysis and theme development. No formal inter-rater reliability assessment was undertaken.

## 6.4 System Safety Analysis

Having established the methodological framework for this case study, the investigation commenced with the foundational step of the safety assurance process: a detailed hazard and risk analysis of Hinata. The following sections present the results of this analysis, beginning with the systematic identification of potential hazards using the SHARD analysis method, followed by a formal risk

assessment to determine their clinical significance and prioritise them for mitigation. The artefacts produced from this analysis form the core evidence presented to stakeholders in Section 6.5.

### **6.4.1 Hazard Analysis**

For the initial hazard identification, Software Hazard Analysis and Resolution in Design (SHARD) was selected as the primary analysis technique. SHARD is a well-established, projective method specifically designed for investigating the behaviour of complex, software-intensive systems [39]. As a variant of the industrially-proven HAZOP technique, SHARD is structured around analysing information flows, which makes it well-suited for a CA whose core function is to send back and forth information between a patient and a clinical system. The technique provides a systematic, guideword-driven approach to exploring potential deviations from intended behaviour, ensuring a thorough and repeatable analysis. Its successful application in other complex healthcare IT settings further established it as a credible and appropriate choice for this case study.

#### **6.4.1.1 The SHARD Method: Guidewords and Process**

The SHARD technique, as detailed by Pumfrey [39], centres on applying a set of simple yet powerful guidewords to each identified information flow in a system design. These guidewords prompt the analyst to consider deviations from the intended behaviour of that flow. The core guidewords used in this analysis are:

- **Omission:** A service or piece of information is not provided when it was intended to be (e.g. the AI fails to ask a required question).
- **Commission:** A service or piece of information is provided when it was not intended (e.g. the AI provides an unnecessary or out-of-context response).
- **Early / Late:** A service is provided earlier or later than intended, which can be measured against absolute time or relative to other system events.
- **Value (Incorrect):** A service is provided at the correct time, but the information content itself is wrong (e.g. the AI misunderstands a symptom and records an incorrect value).

The analysis process involved applying these guidewords to the key information flows in the system, such as the patient's spoken utterance being passed to the NLU, and the NLU's classified intent being passed to the dialogue manager. For each deviation, potential causes were identified, and the resulting effects were traced to determine if they could lead to a system-level hazard.

#### **6.4.1.2 Data Flow Diagrams**

Data Flow Diagrams (DFDs) are a standard visual notation used in structured analysis to depict how data is processed by a system through a network of activities, data stores, and external entities [236]. To provide a clear and structured representation of the system's architecture and its operational data flows, a top-down modelling approach using DFDs was employed.

A high-level context diagram (Figure 34) was first developed to define the system's boundaries and its primary interactions with external entities, such as the patient and the clinical review dashboard. All process flows marked in blue were assigned step identifiers. Hinata was designed as a modular system to instantiate different modules for different medical conditions, it uses cataract questions module in this case study. For this module, these identifiers use the prefix "QC" (e.g. QC 1.3, QC 8.0) and refer to specific conversational and supervisory steps in that workflow.

This was then decomposed into a more detailed Level-2 DFD (Figure 35), which models the key internal processes of the CA, such as the ASR transcription, NLU intent classification, and response generation components, along with the critical data stores they interact with. These diagrams were not merely descriptive; they formed a crucial prerequisite for the safety analysis. The visual representation of information flows in the DFDs provided the precise pathways that were systematically interrogated using the SHARD guidewords, ensuring the hazard identification process was grounded in the actual system design and allowing for a traceable analysis of how a deviation in one component could propagate through the system to affect another.

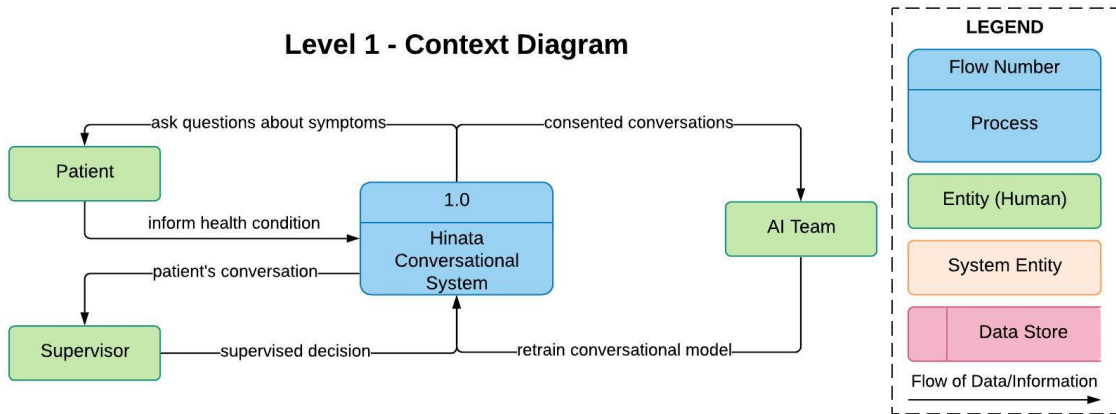


Figure 34: Hinata DFD context diagram

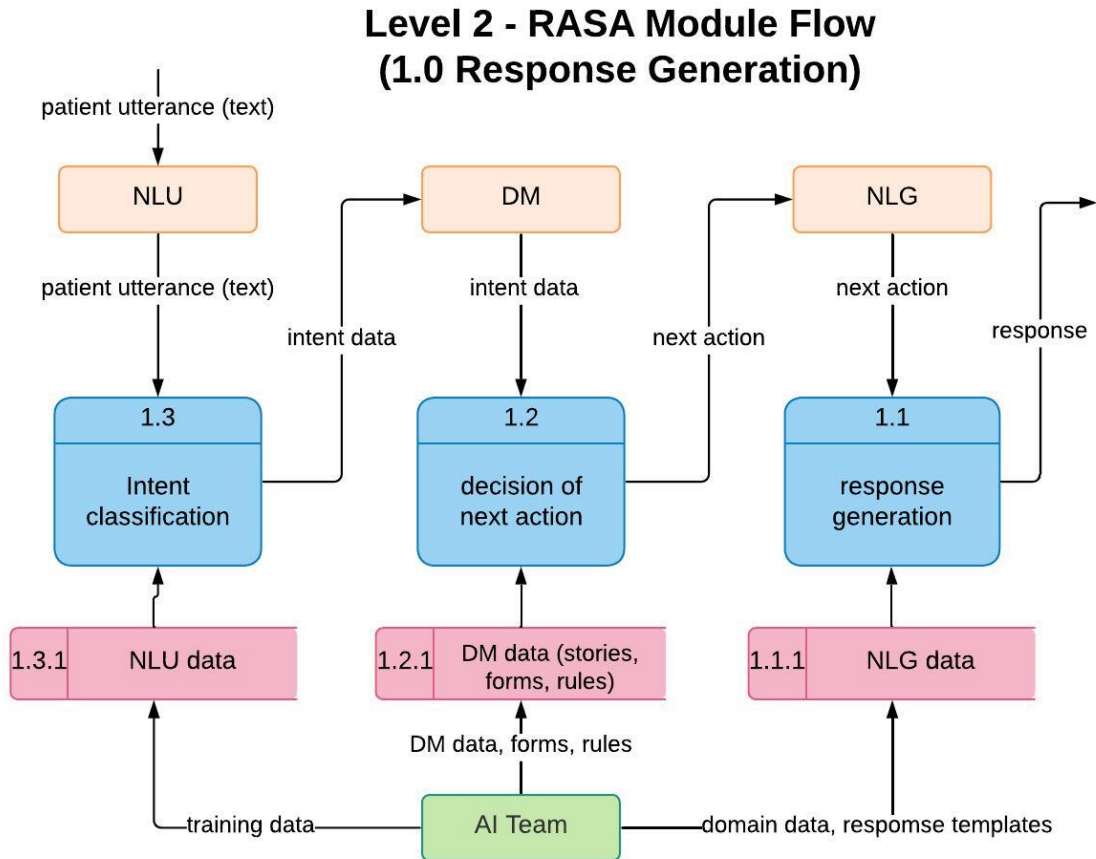


Figure 35: Hinata DFD RASA conversation flow

### 6.4.1.3 SHARD Analysis Results

The systematic application of the SHARD methodology yielded key insights into the safety profile of the CA. The analysis moved beyond a simple checklist of potential faults, providing a structured way to reason about how specific deviations in the system's information flows could propagate into failure modes leading to clinically significant hazards. Selected findings, exemplified by ASR, NLU, and human-computer interaction errors, are shown in Table 20. The complete analysis tables are provided in Appendix D<sup>2</sup>.

A key insight from the analysis is the role of the NLU as a common failure point; the majority of failures that could lead to clinical hazards were found to originate from intent classification and slot value extraction. Furthermore, several deviations were attributed to potential human error, highlighting the need for user training and robust interface design in a human-in-the-loop system [190]. The SHARD process provides a clear and traceable chain of causality from technical root causes to clinical consequences. For example, the analysis traces how an issue like "insufficient or biased NLU training data" (QC 1.3.1) is not, in itself, a clinical hazard. It is a technical deficiency that leads to the failure mode of "Incorrect Intent Identification" (QC 1.3). The outputs of this detailed SHARD analysis were then used as the primary input for formally defining the system's key failure modes.

This SHARD analysis also confirms the direct applicability of the failure mode taxonomy developed in chapter 4. For example, the SHARD deviation 'Incorrect intent classification' (Table 20, QC 1.3) is a clear instance of the taxonomy's Misunderstanding (incorrect intent / slot) failure mode (section 4.5.1.2). The identified cause, 'Sparse / wrong training data', maps directly to the taxonomy's Data-Induced Faults (section 4.5.2.1), such as insufficient or unrepresentative data or class imbalance. Similarly, deviations related to 'Human error' (Table 20, QC 8.0) align with the taxonomy's classification of the User as a source of Causal Contributors (section 4.5.4.1). This confirms the utility of the taxonomy as an analytical tool for interpreting the results of a SHARD analysis.

---

<sup>2</sup>Only the cataract questions conversation flow ("QC") and its supervisory review module are shown in this chapter. Other workflow steps were analysed using SHARD but are redacted for confidentiality. QC labels (e.g. QC 1.3, QC 8.0) refer to the process steps defined in the DFDs.

Table 20: Hinata: selected SHARD analyses

Flow	ID	Guide-word	Deviation	Causes	Effects	Recommendation
QC 1.2	1	Omission	Empty slot value	Form error; wrong intent	Incomplete auto decision	Explicit confirmation of critical symptoms
QC 1.2	2	Value	Incorrect slot value	Form error; Incorrect intent	Incorrect auto decision	Explicit confirmation of critical symptoms
QC 1.3	1	Value	Incorrect intent classification	Sparse / wrong training data	Incorrect safety-check decision	Add training data; threshold + fallback
QC 1.3.1	1	Value	Incorrect NLU training data	Development / data-acquisition error	Incorrect safety-check decision	Use distinct consented utterances
QC 4.0	2	Late	Conversation input delayed	Network delays; UI bug	Delay concluding referral	Software and network optimisations
QC 8.0	1	Omission	Supervisor gives no input	Human error	Incomplete decision	Dashboard timeout warning
QC 8.0	2	Early	Input too early	Human error	Incorrect decision	Lock fields until the summary ready
QC 8.0	3	Value	Incorrect input value	Human error	Incorrect decision	Confirmation dialog shows parsed symptoms

## 6.4.2 Identification of Failure Modes

The granular findings from the SHARD analysis were consolidated into a formal set of causal failure modes, presented in Table 21. These failure modes represent the specific technical or human-related failures within the system that could trigger a hazardous state. Each Failure Mode (FM) is given a traceable ID linking it back to the specific system flow (QC numbers from the detailed analysis) where the failure originates. This step is critical for abstracting the dozens of low-level deviations found during SHARD into a manageable and meaningful list of core system weaknesses that must be controlled.

The technical failures, FM-01 (incorrect intent classification) and FM-02 (incorrect or missing slot value), are both specific instances of the taxonomy's Misunderstanding (incorrect intent / slot) failure mode (section 4.5.1.2). Their identified source, 'NLU model & data', confirms they are rooted in the software component. These failures are direct consequences of the ML-Related Faults (section 4.5.2) defined by the taxonomy, such as Data-Induced Faults (e.g. insufficient training data or class imbalance, section 4.5.2.1) or Model-Induced Faults (e.g. model overfitting or uncertainty miscalibration, section 4.5.2.2).

The human-related failures, FM-03 (supervisor submits invalid input) and FM-04 (supervisor delays patient referral), align with the taxonomy's system-theoretic approach, which models the user (liveware) as a source of Causal Contributors (section 4.5.4.1). These FMs are not software failures, but rather hazardous deviations in Interaction behaviour by a human actor (the supervisor) within the wider socio-technical system, which the safety analysis must also control.

Table 21: Hinata: identified causal failure modes

Failure mode ID	Description	Flow ID	Failure Source
FM-01	Incorrect intent classification for symptoms.	1.3, 1.3.1	NLU model & data
FM-02	Incorrect or missing slot value extracted for a critical symptom.	1.2.1, 1.2.2, 1.3.1	NLU model & data
FM-03	Supervisor submits invalid or premature input, overriding the AI recommendation.	7.0.1, 8.0.1-8.0.3	Human error
FM-04	Supervisor delays patient referral to the hospital.	4.0.2	Human error

### 6.4.3 Hazards and Risk Assessment

While the failure modes in Table 21 describe how the system can fail, the next step in the safety analysis is to understand their ultimate clinical consequences. This is achieved by mapping the failure modes to top-level clinical hazards, the system states that pose a direct risk to patient safety. This mapping, shown in the final column of Table 22, creates a clear and auditable 'chain of evidence' from a low-level technical fault to potential patient harm.

Once this link between failure modes and clinical hazards was established, a formal risk assessment was conducted for each hazard. In line with the NHS Digital DCB0160 standard, this involved assessing the potential **severity** of harm to the patient and the **likelihood** of that harm occurring, given the system's design. Each hazard was then scored using the 5×5 clinical safety risk matrix and associated risk acceptability scale shown in Figure 29 [235] and introduced in chapter 5. This produced a final risk score, which was used to prioritise the hazards that required the strongest mitigation. The complete clinical hazards list including the mapping from failure modes to hazards and the final risk assessment is presented in Table 22.

Table 22: Hinata: clinical hazard log

ID	Hazard description	Likelihood	Severity	FM link
H-01	Under-referral of eye cataract symptoms.	Low	Major	FM-01, FM-02, FM-03
H-02	Over-referral of eye cataract symptoms.	Low	Minor	FM-01, FM-02, FM-03
H-03	Late referral of eye cataract symptoms.	Low	Major	FM-04
H-04	Patient misidentification.	Low	Major	FM-02

#### 6.4.3.1 Assessing Severity: The Worst Plausible Outcome

The severity score for each hazard was determined through a series of workshops with the Hinata development team and clinical stakeholders, focusing on the worst plausible clinical outcome. The assessment aligns directly with the values presented in the hazard log (Table 22).

- For a hazard like H-01 (Under-referral), the clinical consensus was that the worst-case scenario is a missed or delayed diagnosis of a time-sensitive

complication, such as retinal detachment. While not typically life-threatening, such an outcome could lead to a permanent, non-trivial injury, including significant loss of vision. Following the NHS matrix guidelines, the group concluded that this maps directly to a severity score of **major**.

- In contrast, for H-02 (Over-referral), the stakeholders agreed that the worst-case scenario is a patient with no serious issues being unnecessarily escalated for review. This causes patient anxiety and consumes valuable clinical resources, but does not typically lead to direct physical injury. Therefore, the group's consensus was that its outcome maps to a severity score of **minor**.

#### 6.4.3.2 Assessing Likelihood: Considering System Controls and Behaviour

The likelihood score for each hazard was also assessed collaboratively, taking into account the system's design and inherent controls. After reviewing the system's architecture and performance, the stakeholders rated the likelihood for both primary hazards as **low**.

- The likelihood of H-01 (under-referral) was assessed as **low**. The group reasoned that while technical failures (FM-01, FM-02) or ambiguous patient phrasing could lead to a misclassification, the system has multiple, independent safety barriers. These include a high-recall NLU model designed to minimise false negatives, explicit confirmation dialogues, and a mandatory human-in-the-loop review process.

In practice, two reviewers check the AI interaction by comparing the transcript (and, if needed, the audio recording with consent), either when the AI is uncertain or as part of routine sampling. They can correct any mismatch between what the patient actually said and what the system inferred. The system also reconfirms key symptoms during the call and reads back a structured summary for the patient to confirm. Because of these safeguards, the probability that a red-flag symptom is missed by both the AI and the human review was judged to be very low. The consensus was that these layered controls make an unmitigated under-referral a rare, low-likelihood event.

- The likelihood for H-02 (over-referral) was also assessed as **low**. The stakeholders noted that although a high-recall NLU model can carry a risk of over-sensitivity, the highly structured nature of Hinata's dialogue and its narrow,

well-defined clinical scope significantly limit the opportunities for ambiguity. The group concluded that the system's design makes erroneous over-referrals unlikely.

#### 6.4.3.3 Calculating the Final Risk Score

The final risk score was calculated by multiplying the numerical values for severity and likelihood, as determined in the workshops and defined by the risk matrix (Figure 29). This explains the significant difference in the final risk prioritisation:

- For **under-referral (H-01)**: The risk score is  $4$  (major)  $\times$   $2$  (low) = **8**. According to the NHS matrix, this falls into the *undesirable* (amber) category. This indicates a level of risk that requires the implementation of robust control measures and warrants further attempts at reduction where practicable.
- For **over-referral (H-02)**: The risk score is  $2$  (minor)  $\times$   $2$  (low) = **4**. This falls into the *acceptable* (green) category. This indicates that the risk is considered broadly acceptable, particularly given that it is well-controlled, and requires no further action beyond routine monitoring.

This detailed risk assessment process, grounded in stakeholder expertise, provides the final step of the analysis. It moves beyond simply identifying what could go wrong to a prioritised view of what must be controlled, and forms the basis for the safety requirements that follow.

#### 6.4.4 Derivation of a Multi-Layered Safety Requirements Framework

The assurance of a safety-critical system that incorporates Machine Learning (ML) components requires a structured and traceable approach to safety requirements engineering. Safety cannot be treated as an emergent property or an add-on; it must be systematically designed and built into the system from the outset [237, 238]. This principle is particularly important for clinical CAs like Hinata, where failures in understanding patient communication can lead directly to patient harm. The framework for deriving safety requirements for Hinata is therefore grounded in established principles of software safety assurance, so that each requirement can be traced for an identified system-level risk [239, 238].

This process begins with the foundational principle of **Requirements Validity**, which mandates that "Software safety requirements shall be defined to address the software contribution to system hazards" [238]. Consequently, the safety requirements for the Natural Language Understanding (NLU) component are not arbitrary performance targets; they are derived directly from the hazard analysis performed in Section 6.4.1 and the subsequent risk assessment in Section 6.4.3. Each requirement provides a specific mitigation for one or more identified clinical hazards, such as the risk of gathering incorrect or incomplete patient symptom information (H-01) or providing a delayed response in a critical situation (H-02).

Furthermore, the framework adheres to the principle of **Requirement Decomposition**, which posits that "The intent of the software safety requirements shall be maintained throughout requirements decomposition" [238]. This principle guides the systematic refinement of high-level clinical safety objectives into concrete, verifiable, and multi-layered requirements. This hierarchical decomposition flows from the overall system level down to the specific ML components. A well-established methodology for this decomposition involves translating high-level vehicle or system requirements into specific component requirements, which are then further refined into detailed ML requirements for the model and its data, as demonstrated in safety-critical automotive applications like pedestrian detection [240].

This multi-layered approach ensures that there is a clear and auditable chain of reasoning from an abstract clinical safety goal (e.g. "ensure patient symptoms are accurately captured") to a specific, testable requirement for the ML model (e.g. a minimum F1-score for intent classification) and the data used to build it (e.g. criteria for annotation accuracy). This chain of derivation is not merely a documentation exercise; it forms the logical backbone of the safety argument presented in the safety case. It provides objective evidence that the system has been engineered with a primary focus on safety, addressing the specific ways in which the NLU component can fail and contribute to patient harm.

#### **6.4.4.1 Safety/Design Recommendations**

The first step in decomposing the high-level hazards into concrete requirements is to establish a set of **Safety Design Recommendations**. These are high-level principles and architectural strategies derived directly from the SHARD analysis. They represent the initial, broad-stroke solutions to mitigate the identified risks

before they are refined into specific, testable requirements. Table 23 presents these recommendations.

Table 23: Hinata: safety and design recommendations

ID	Safety/Design Recommendation	Reference
SDR-01	Explicit confirmation strategy to overcome empty/incorrect symptom value to avoid incorrect automated safety check decision	QC 1.2
SDR-02	Fallback intent classifier implementation; consented conversations should be used to add or improve the NLU data	QC 1.3
SDR-03	A natural and grammatically correct response can reduce misinterpretation by patients	QC 1.1.1
SDR-04	Human-in-the-loop (HITL) for supervision and review of patient's conversation	QC 7.0, QC 8.0
SDR-05	Ensure dashboard reflects the true transcript of user's speech to avoid human errors in safety check decision	QC 7.0

#### 6.4.4.2 Hinata Safety Requirements

These recommendations form the conceptual basis for the system's safety architecture. For example, **SDR-01** directly addresses the risk of incorrect automated decisions (H-01) by proposing a dialogue-level control. **SDR-02** targets the root cause of many NLU failures by recommending both a technical fallback and a process for data improvement. **SDR-04** and **SDR-05** establish the principle of having a human supervisor as a critical safety barrier, a common and effective strategy in socio-technical systems to mitigate the risks of automation [239].

Following these high-level recommendations, the next layer of decomposition involves translating them into specific, verifiable **Safety Requirements**. These requirements, shown in Table 24, make the design recommendations concrete and provide the direct link to the more detailed ML-specific requirements that follow.

These system-level safety requirements act as the bridge between the hazard analysis and the detailed technical specification of the ML components. For example,

Table 24: Hinata: safety requirements with hazard traceability

ID	Safety requirement	Rationale	Traceability (Hazards)
SR-01	The system shall provide an explicit confirmation mechanism to allow the user to verify or correct information they have provided.	Refines SDR-01. This makes the recommendation for a confirmation strategy a testable system feature, directly controlling for misunderstandings that could lead to incorrect decisions.	H-01
SR-02	The system shall implement a fallback mechanism that is triggered when the NLU model's confidence falls below a validated threshold.	Refines SDR-02. This requirement operationalises the concept of a fallback classifier, making it a mandatory safety function tied to model uncertainty. This is a key control for preventing the system from acting on low-confidence or OOD inputs.	H-01, H-03
SR-03	The system shall provide a supervisory interface that displays the patient's transcribed utterance and the NLU's interpretation for review.	Refines SDR-04 & SDR-05. This makes the HITL recommendation a concrete system feature, ensuring the supervisor has the necessary information to perform their safety function effectively. This mitigates the risk of both automated errors and errors from the human supervisor.	H-01, H-04
SR-04	The system shall generate responses to the user that are grammatically correct and clinically unambiguous.	Refines SDR-03. This requirement ensures that the system's output does not become a source of confusion or misinterpretation for the patient, which could lead to incorrect patient actions or loss of trust.	H-03

**SR-02** requires the existence of a confidence score from the NLU model, which in turn drives the creation of ML-specific requirements like ML-MOD-03 (OOD detection) and the associated data requirements to validate this functionality. This clear, hierarchical decomposition is fundamental to building a defensible safety case [237].

#### 6.4.4.3 NLU Model Requirements

For the NLU model to be considered acceptably safe, it must satisfy a set of specific, verifiable requirements that directly mitigate the identified clinical hazards. These requirements are derived from the desiderata for the Model Learning stage of the ML lifecycle, primarily focusing on ensuring the model is both **Performant** and **Robust** [241]. A performant model correctly executes its intended function under normal conditions, while a robust model maintains its performance in the face of perturbations and unexpected inputs, which are inevitable in a real-world clinical deployment.

Table 25 outlines the specific safety requirements for the NLU model, extracted directly from the Hinata safety analysis. The rationale for each requirement provides a clear link back to the safety objectives and the hazards it is intended to control. For instance, the performance requirement for intent classification accuracy (MLMP1) is a direct control for the high-priority hazard H-01, "Incorrect or incomplete patient symptom information gathered," which was identified from the SHARD analysis deviation 'incorrect intent identification'. Similarly, the requirement for robustly handling out-of-distribution (OOD) inputs (MLRRQ1) is a critical control for the same hazard, justified by the known vulnerability of standard classifiers to fail with high confidence on unexpected inputs, a significant safety risk demonstrated in Use Case 2 (section 5.2) outlined in Chapter 5.

The 90% thresholds were agreed with stakeholders during the development of the safety case. They were informed by internal statistical analysis and later confirmed during the technical stakeholder interview. Residual errors are managed by downstream controls, for example through validation steps, fallback strategies, and explicit confirmation with the user.

The robustness requirement (MLRRQ1) also implies a mechanism, such as Monte Carlo Dropout and entropy-based uncertainty quantification, to detect and safely handle OOD inputs rather than act on a confidently wrong interpretation.

Table 25: Hinata: NLU model safety requirements

ID	NLU model safety requirements	Rationale	Type
MLMP1	The NLU model shall identify intents with at least 90% overall accuracy.	A high threshold minimises the risk of incorrect classification and provides confidence in model behaviour.	Model performance
MLMP2	The NLU model shall classify <i>critical</i> intents with recall $\geq 90\%$ .	High recall reduces the chance that a clinically important symptom is missed.	Model performance
MLRRQ1	The NLU model shall detect <i>OOD</i> utterances with $\geq 90\%$ accuracy.	Detecting OOD input prevents misunderstanding and downstream errors.	Model robustness

#### 6.4.4.4 NLU Data Requirements

The assurance of an ML model is fundamentally inseparable from the assurance of the data used to create and validate it [241, 242]. Flaws, biases, or gaps in the data will inevitably be encoded into the model, creating latent failure conditions that may only manifest during operational use. Therefore, establishing a rigorous set of requirements for the NLU model's data is a prerequisite for trusting any claims made about the model's safety and performance.

The development of these data requirements is systematically structured around the four key desiderata for data management identified by Ashmore et al. [241]: **Relevant**, **Complete**, **Balanced**, and **Accurate**. This structure ensures a comprehensive and principled approach to data assurance, addressing the multifaceted nature of data quality.

A critical relationship exists between the data requirements and the model requirements. The data requirements form the necessary foundation upon which the model requirements can be meaningfully verified. For example, the model requirement for robustness to OOD utterances (MLRRQ1) is unverifiable unless the test dataset satisfies the **Completeness** desideratum (MLDR3) by including a representative range of these variations. Without such data, any claim of robustness

would be unfounded speculation. Thus, the data assurance argument is not parallel to the model assurance argument; it is a logical prerequisite.

Table 26 details the ML data requirements for the Hinata NLU component, with each requirement's rationale explicitly linked to one of the four data desiderata. The rationale behind the training numbers came from the understanding of system behaviour during workshops with stakeholders. At the start of the development for Hinata, these numbers were met to have adequate confidence in the system and data was updated with more examples as the system matured. These data requirements are intended to keep the likelihood of critical misclassification acceptably low; they do not fully eliminate the risk of under-referral, which is additionally controlled by other safety barriers such as explicit confirmation with the user, periodic review of NLU data and model performance and clinical supervision.

#### **6.4.5 Structure of the Safety Case**

A safety case is a structured, evidence-based argument that shows a system is acceptably safe to use in a given context [237, 239, 242]. For a software-based system like the one in this study, the safety case is the main way to present safety assurance to stakeholders such as clinicians, hospital administrators, and regulators [239]. It brings together the key safety activities, from hazard analysis to the definition of safety requirements and the verification of those requirements.

The structure of this safety case follows current assurance practice for systems with components that can be reused or updated over time [243]. Although the system is delivered as a single service, its ML components can be retrained or replaced during its life. This motivates a modular safety case, in which assurance is organised into separate argument modules for each component [244]. Using Goal Structuring Notation (GSN), each part of the argument is tied to a specific component, so that changes can be handled locally without rebuilding the entire case.

The high-level architecture of the safety case also follows the pattern proposed by Picardi et al. [242]. The first part is the main safety argument, which claims that the CA is safe to use because its risks are controlled. The second part is a set of confidence arguments, which focuses the trustworthiness of the processes used to develop and verify the system. This separation allows the safety of the system and the reliability of the assurance process to be assessed independently [242].

Table 26: Hinata: NLU data safety requirements

ID	ML model data requirement	Rationale	Type
MLDR1	Training data shall not contain ambiguous or near-duplicate intent examples.	Ambiguity can cause the model to confuse similar intents and return the wrong class.	Data relevance / validity
MLDR2	Each cataract-symptom intent and the “extract patient-agreement” intent shall have $\geq 20$ distinct training utterances.	Experience with NLU models shows that about 20 examples give acceptable robustness.	Data completeness / sufficiency
MLDR3	OOD intent shall have $\geq 20$ training examples.	Needed to satisfy the robustness requirement for detecting OOD utterances.	Data completeness / sufficiency
MLDR4	Retrain the NLU model on new, consented data at least quarterly or when live accuracy drops $> 3\%$ .	Models drift over time; timely updates keep performance within safety limits.	Data completeness / sufficiency
MLDR5	Value-extraction intents shall include real patient utterances.	Real patient language improves generalisation and reduces errors on unseen data.	Data accuracy
MLDR6	No intent class shall represent $< 10\%$ of the cataract dataset.	Class imbalance can bias the model and lead to misclassifications.	Data balance

#### 6.4.6 Hinata Safety Case (GSN)

The safety case for Hinata is constructed using GSN, a graphical notation that explicitly represents the goals (claims), strategies (argument decomposition), context, and solutions (evidence) of an argument [237, 242]. The following sections provide more details on key arguments which make up the overall safety claim for Hinata.

### 6.4.6.1 Argumentation over Top Level Goal

The top-level argument (G-Hinata-Safe), presented below, establishes the overall claim of safety and the strategy by which this claim is justified.

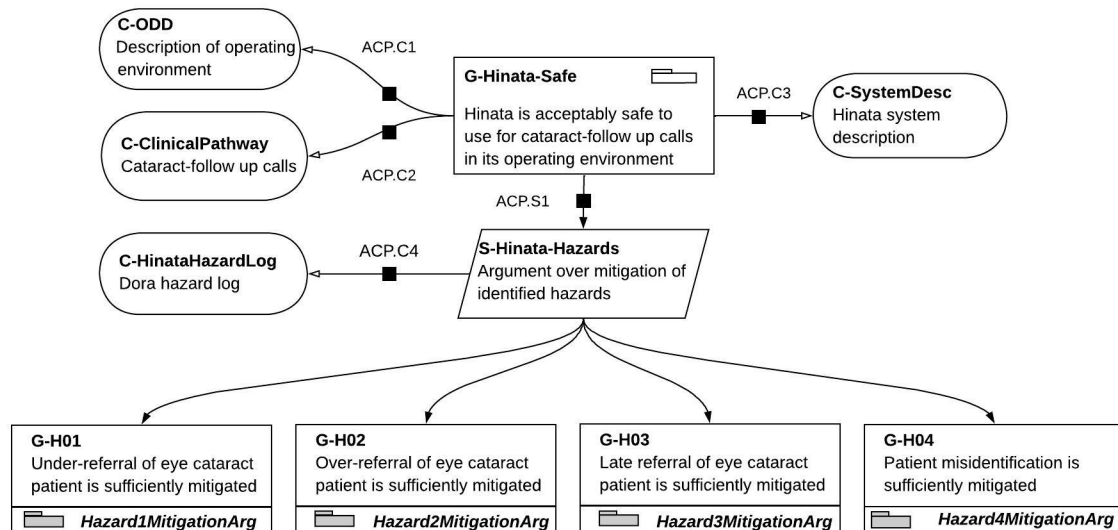


Figure 36: Hinata safety case top level argument

**G-Hinata-Safe:** The CA is acceptably safe for post-operative follow-up within the defined clinical pathway.

**Context** This top-level goal is made within a clearly defined context, which includes:

**C-SysDesc:** The System Description document, detailing its architecture and functionality.

**C-ODD:** The Operational Design Domain (ODD), which specifies the intended use, patient population, and environmental conditions.

**C-ClinicalPathway:** The clinical pathway in the context of current argument, namely the use of Hinata for cataract follow-up calls.

**C-HinataHazardLog:** The clinical hazard log for Hinata after application of SHARD analysis as shown in Table 22.

**Strategy (S-Hinata-Hazards)** The strategy employed is an *Argument over the control of all identified hazards*. This establishes a classic hazard-based safety argument, decomposing the top-level goal into a set of sub-goals, one for each of the clinical hazards identified in the risk assessment (Table 22). This ensures the argument is comprehensive and directly addresses all known sources of potential harm. The argument proceeds by making a claim for each hazard, for example:

- **G-H01:** Hazard H-01 (Under-referral of eye cataract symptoms) is sufficiently mitigated.
- **G-H02:** Hazard H-02 (Over-referral of eye cataract symptoms) is sufficiently mitigated.
- **G-H03:** Hazard H-03 (Late referral of eye cataract patient) is sufficiently mitigated.
- **G-H04:** Hazard H-04 (Patient misidentification) is sufficiently mitigated.

Each of these hazard-specific goals is then further decomposed by arguing that sufficient and effective risk controls (i.e. safety requirements) are in place and have been satisfied.

#### 6.4.6.2 Argumentation over Hazard H-01

The clinical hazard log as shown in Table 22 identifies the hazards that may arise in Hinata. Although the likelihood for all identified hazards is low, H-01 (under-referral of eye cataract symptoms) is used here as an example because it is considered clinically higher priority. Figure 37 details the argument structure for mitigating this hazard.

**G-H01:** Hazard 1: Under-referral of eye cataract patient is sufficiently mitigated. This is the top-level goal for this specific hazard argument.

- **Context (C-CataractSymptoms):** Defines the specific symptoms relevant to this hazard: eye\_redness, eye\_pain, eye\_vision\_issues, eye\_new\_floaters, eye\_flashing\_lights.
- **Strategy (S-H01-SysComp):** The strategy is the "Argument over mitigation from system components". This decomposes the hazard mitigation based on the functional components of the CA.

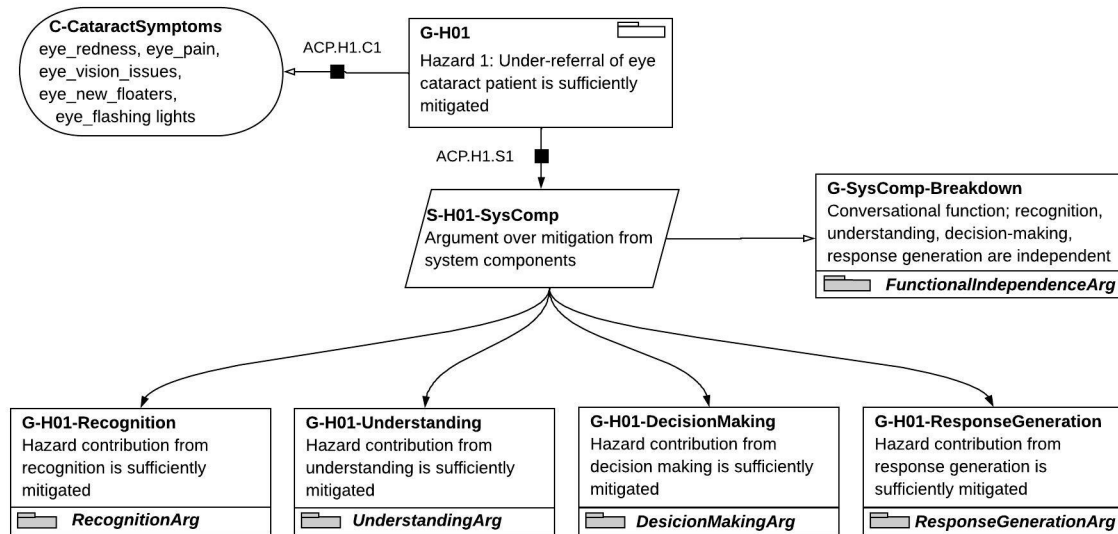


Figure 37: Hinata safety case argument for H-01 mitigation

- **Context (G-SysComp-Breakdown):** Provides the rationale for the decomposition, listing the conversational functions (recognition, understanding, decision-making, response generation) and asserting their functional independence via a ‘FunctionalIndependenceArg’ module.
- **G-H01-Recognition:** Claims that the hazard contribution from the recognition (ASR) component is sufficiently mitigated. This goal links to a further decomposed argument module (‘RecognitionArg’).
- **G-H01-Understanding:** Claims that the hazard contribution from the understanding (NLU) component is sufficiently mitigated. This goal links to the detailed NLU arguments shown previously in Figures 38 and 39 via the ‘UnderstandingArg’ module.
- **G-H01-DecisionMaking:** Claims that the hazard contribution from the decision-making (DM) component is sufficiently mitigated. This goal links to a further decomposed argument module (‘DecisionMakingArg’).
- **G-H01-ResponseGeneration:** Claims that the hazard contribution from the response generation (NLG) component is sufficiently mitigated. This goal links to a further decomposed argument module (‘ResponseGenerationArg’).

This structure argues that the overall hazard (H-01) is controlled by demonstrating that the potential contribution to this hazard from each functionally independent stage of the conversational pipeline (Recognition, Understanding, Decision Making, Response Generation) is adequately mitigated. The detailed arguments for each component are contained within separate modules, promoting modularity in the safety case.

#### 6.4.6.3 Argumentation over the NLU Component

The NLU component is a primary control for several key hazards, most notably H-01. Therefore, a significant portion of the safety case is dedicated to arguing for its safety. This argument is structured hierarchically, starting from a top-level claim (**G-H01-Understanding: "Hazard contribution from understanding is sufficiently mitigated"**) and decomposing it into verifiable claims about ML model and its data. This strategy is rooted in the ML lifecycle, which distinguishes between Data Management and Model Learning [241]. Figure 38 shows the safety case argument for the NLU model for hazard H-01 mitigation while Figure 39 shows the safety case argument for the NLU data argument.

**The NLU Model Argument** This part of the safety case argues that the NLU model itself meets its allocated safety requirements.

**G-NLU-Model:** NLU model satisfies the allocated ML safety requirements.

**Strategy(S-MLSafetyReqs):** This goal is supported by an argument demonstrating that the model satisfies all of its allocated safety requirements, directly implementing the principle of *Requirements Satisfaction* [238].

**G-NLUSatisfiesML:** The NLU model satisfies the ML safety requirements.

**Context (C-NLUModel):** The specific NLU model being evaluated.

**Context (C-NLUData):** The NLU data used for training and testing the model.

**Strategy (S-NLUSatisfaction):** The argument proceeds by "Argument over satisfaction of NLU safety requirement", decomposing the satisfaction claim into performance and robustness aspects.

- **G-MLPerfSatisfied:** ML performance safety requirements are satisfied. This addresses requirements **MLMP1** ("identify intents with at least 90 % overall accuracy") and **MLMP2** ("classify critical intents with recall  $\geq$  90 %").
  - **Context (C-MLPerfReqs):** Specifies the "ML performance safety requirements" (Table 25).
  - **Strategy (S-PerfSatisfaction):** Further decomposes the performance argument via "Argument over satisfaction of NLU performance safety requirement", addressing recall and F1 score metrics.
    - \* **G-PerfFN:** The NLU model identifies FNs and achieves at least 90 percent recall. This supports requirement **MLMP2**.
      - **Solution (Sn-RecallScore):** Evidence is provided by the "Recall score of cross validation testing".
    - \* **G-PerfF1:** The NLU model achieves 90 percent F1 score. This supports the accuracy requirement **MLMP1**.
      - **Solution (Sn-F1Score):** Evidence is provided by the "F1 score on testing data".
- **G-MLRobustnessSatisfied:** ML robustness safety requirements are satisfied. This addresses requirement **MLRRQ1** ("detect OOD utterances with  $\geq$  90).
  - **Context (C-MLRobustnessReqs):** Specifies the "ML robustness safety requirements" (Table 25).
  - **Strategy (S-RobustnessSatisfaction):** Further decomposes the robustness argument via "Argument over satisfaction of NLU robustness safety requirement", focusing on the identification of OOD intents.
    - \* **G-RobustnessOOD:** The NLU model correctly identifies OOD intents with at least 90 percent confidence. This directly supports requirement **MLRRQ1**.
      - **Solution (Sn-OODResults):** Evidence is provided by the "OOD test results".

The NLU model argument is considered complete because it shows how the allocated safety requirements for both performance and robustness have been

addressed. The performance requirements were derived from the safety analysis and focus on both accuracy and sensitivity (recall), as the cost of missing a clinically important symptom (i.e. an under-referral) is high.

A high threshold for OOD detection was also set because the CA is voice-based and background noise or unclear speech can affect how patient responses are interpreted. This reasoning was confirmed by the technical stakeholder during interview. The robustness requirement (90%) is intended to be high enough to control most misclassifications at the NLU stage, with any remaining errors managed by subsequent safeguards in the DM. The sequential nature of the CA architecture allows safeguards to be applied at several points in the interaction to minimise risk to the patient.

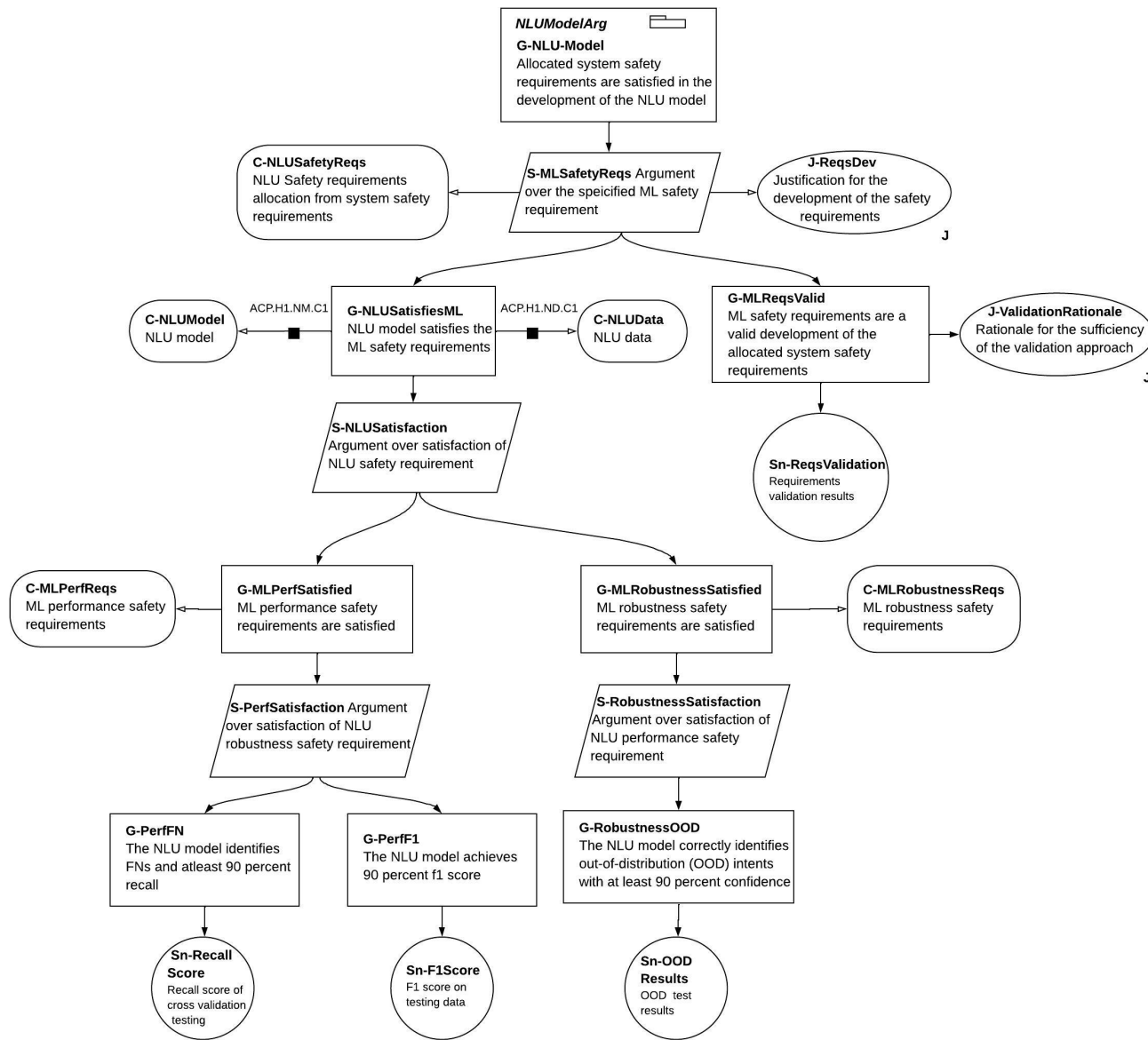


Figure 38: Hinata safety case NLU model argument for H-01 mitigation

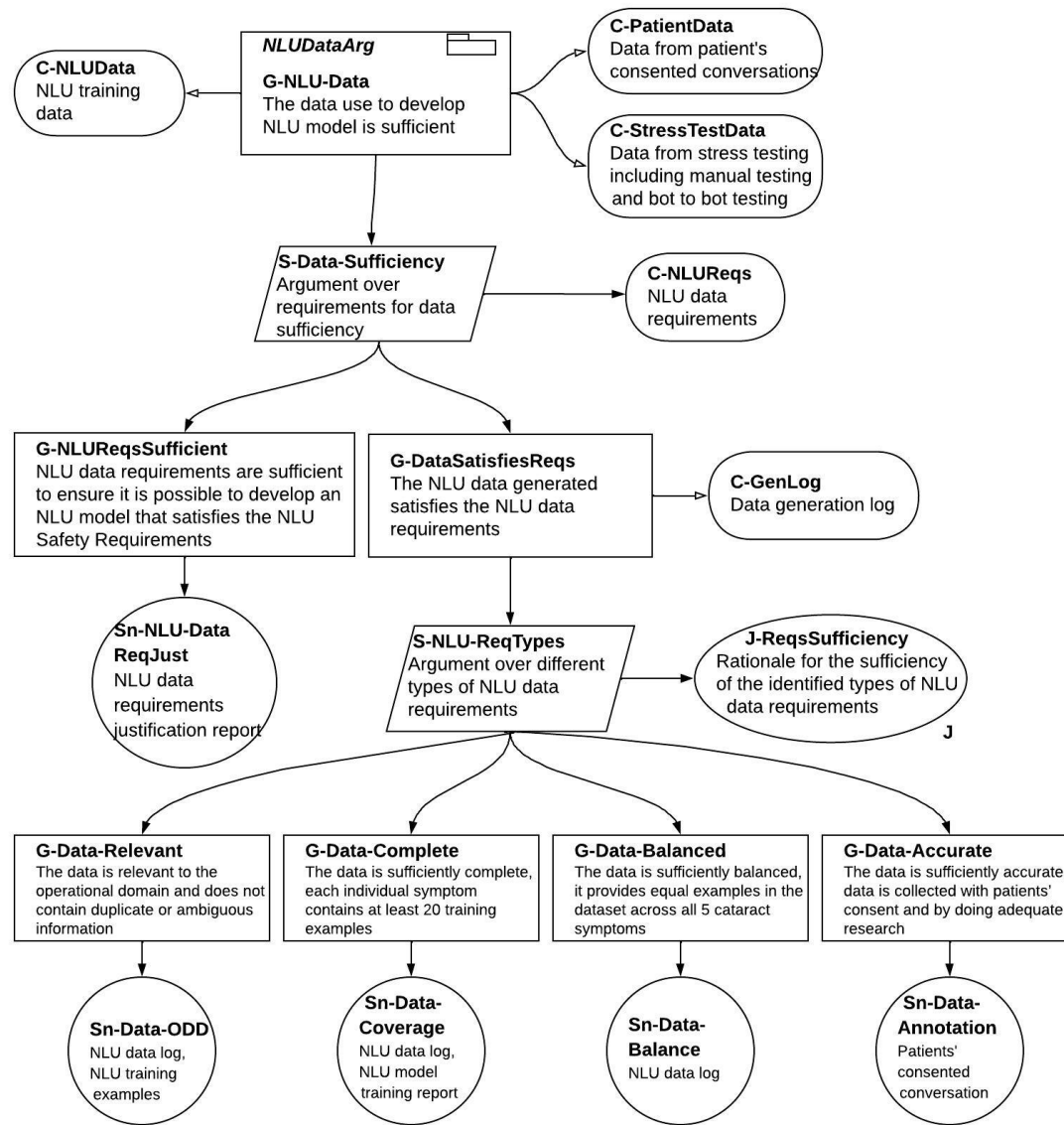


Figure 39: Hinata safety case NLU data argument for H-01 mitigation

**The NLU Data Argument** This argument justifies that the data used to develop and test the NLU model is sufficient and appropriate for its safety function.

**G-NLU-Data:** The data use to develop NLU model is sufficient. This goal argues that the NLU data generated satisfies the NLU data requirements (**MLDR1 - MLDR6**, Table 26).

**Context (C-NLUData):** The specific "NLU training data" used.

**Context (C-PatientData):** Includes "data from patient's consented conversations".

**Context (C-StressTestData):** Incorporates "data from stress testing including manual testing and bot to bot testing".

**Strategy (S-Data-Sufficiency):** The main strategy is an "Argument over requirements for data sufficiency". This decomposes the argument into whether the requirements are sufficient and whether the data meets those requirements.

- **Context (C-NLUReqs):** Refers to the specific "NLU data requirements" (detailed in Table 26).
- **G-NLUReqsSufficient:** Claims that the "NLU data requirements are sufficient to ensure it is possible to develop an NLU model that satisfies the NLU Safety Requirements".
  - **Solution (Sn-NLU-DataReqJust):** Evidence is provided by the "NLU data requirements justification report".
- **G-DataSatisfiesReqs:** Claims that "The NLU data generated satisfies the NLU data requirements".
  - **Context (C-GenLog):** Refers to the "Data generation log" as evidence of the data generation process.
  - **Strategy (S-NLU-ReqTypes):** Further decomposes the satisfaction argument based on different types of NLU data requirements, aligning with the four data desiderata [241] ("Argument over different types of NLU data requirements").
    - \* **Justification (J-ReqsSufficiency):** Provides the "Rationale for the sufficiency of the identified types of NLU data requirements".

- \* **G-Data-Relevant:** The data is relevant to the operational domain and does not contain duplicate or ambiguous information. Supports **MLDR1**.
  - **Solution (Sn-Data-ODD):** Evidence is the "NLU data log, NLU training examples".
- \* **G-Data-Complete:** The data is sufficiently complete; each individual symptom contains at least 20 training examples. Supports **MLDR2**, **MLDR3**, and **MLDR4**.
  - **Solution (Sn-Data-Coverage):** Evidence is the "NLU data log, NLU model training report".
- \* **G-Data-Balanced:** The data is sufficiently balanced; the dataset contains comparable numbers of examples across all five cataract symptoms. Supports **MLDR6**.
  - **Solution (Sn-Data-Balance):** Evidence is the "NLU data log".
- \* **G-Data-Accurate:** The data is sufficiently accurate; it is drawn from consented patient conversations and clinically reviewed sources. Supports **MLDR5**.
  - **Solution (Sn-Data-Annotation):** Evidence is the "Patients' consented conversation" agreement and research reports.

**The NLU Data Argument** The safety requirements set for the CA in this case represent the structure of the four data desiderata set forth by [241]. Data validity was set so the training examples have less or no ambiguity to prevent intent misclassification (key failure mode identified in the taxonomy in Chapter 4). For completeness and sufficiency of data multiple safety requirements were derived after the analysis (MLDR2, MLDR3, MLDR4). For key intents (cataract symptoms and OOD), data needs to have sufficient examples in the training set. The completeness is derived by the safety requirement, MLDR4, by assuring the retraining of model from consented patients' conversation or when the set performance threshold drops by >3%. These requirements came after workshops with technical stakeholders and this was primarily driven by statistical analysis by the team. This was also confirmed by the technical stakeholder during the interview process for evaluation. The accurate safety requirements generation are key to assure the NLU model safety with traceable evidence from safety case to requirements to hazards.

### 6.4.7 The Critical Role of Confidence Arguments (ACPs)

A safety case makes claims based on evidence, but the trustworthiness of that evidence depends on the processes used to generate it. Assurance Claim Points (ACPs) provide the formal GSN structure for this defence, creating arguments about the confidence in the safety case's own foundational artefacts [242]. Figure 40 presents the confidence arguments for hazard 1 mitigation in the safety case.

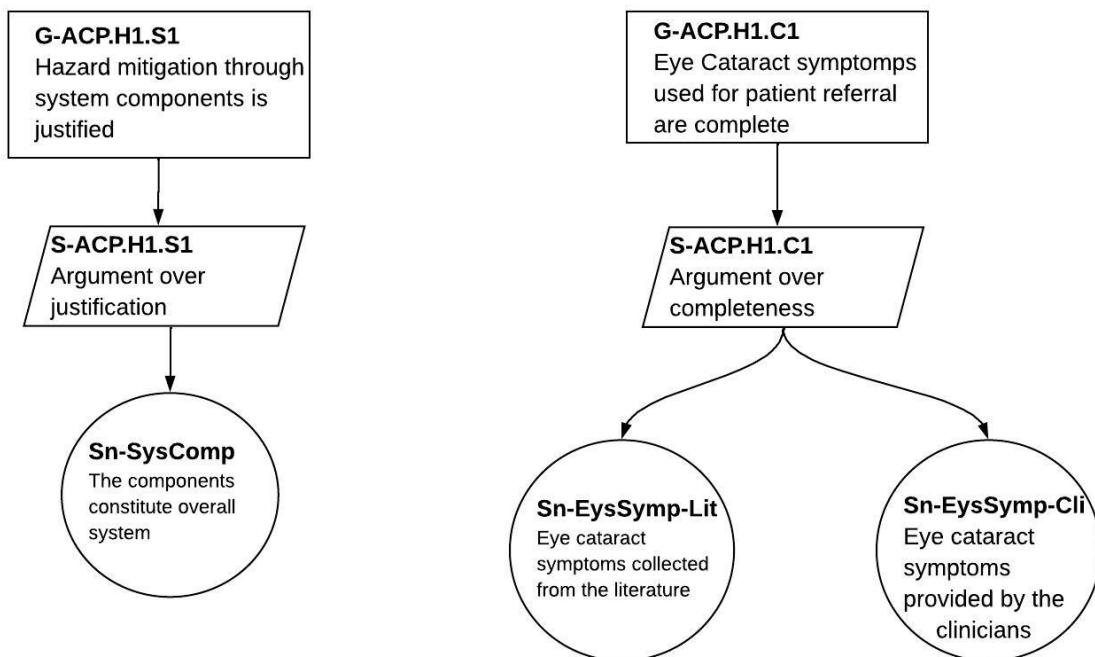


Figure 40: Hinata safety case confidence arguments for H-01 mitigation

#### 6.4.7.1 ACP.H1.S1: Confidence in Hazard Mitigation through System Components

**Claim:** Hazard mitigation through system components is justified.

**Argument:** The system components form the sequential pipeline of the CA, and mitigation is applied at each stage before information is delivered back to the patient. The rationale is that all components form the system architecture and mitigation at each level provides assurance to acceptable safety.

#### **6.4.7.2 ACP.H1.C1: Confidence in Eye Cataract Symptoms Data**

**Claim:** Eye cataract symptoms used for patient referral are complete.

**Argument:** This argument justifies the selection of eye symptoms for cataract patients and confidence in that completeness comes from both the literature and clinician input.

By including these explicit confidence arguments, the safety case for the CA not only justifies the safety of the final system but also provides a transparent and defensible account of the rigour of the safety engineering process itself.

### **6.5 Evaluation of the Case Study Methodology**

This section evaluates the safety assurance methodology as applied to the Hinata CA. The aim is to assess how stakeholders perceived the realism, credibility, and practical value of the methodology in the clinical deployment context. The findings are organised around the themes generated from the stakeholder interviews.

#### **6.5.1 Evaluation Methodology Recap**

The evaluation was based on stakeholder interviews conducted using the questions in appendix C. During the interviews, participants were presented with the safety artefacts produced by the methodology, including the hazard list, risk matrix, ML safety requirements, and GSN safety arguments. The purpose was to obtain feedback on the realism of the identified hazards, the adequacy of the mitigations and safety requirements, the credibility of the GSN safety case, and the practical value of the proposed methodology. The interview feedback was then analysed thematically to identify the principal areas of stakeholder agreement, concern, and recommendation.

#### **6.5.2 Thematic Analysis of Stakeholder Feedback**

The stakeholder interviews for this case study were conducted under approved institutional ethics procedures, with informed consent obtained from all participants.

Ethical approval was granted by the Physical Sciences Ethics Committee (PSEC), University of York (PSEC Application Ref: Aftab20250507, approved 31 May 2025).

The interview transcripts were analysed manually using a deductively informed thematic analysis, guided by the case study protocol, the interview questions in appendix C, and the case study propositions. The analysis of stakeholder interviews resulted in six principal themes regarding the value, completeness, and trustworthiness of the safety case methodology. Each theme is explored below with supporting evidence from the transcripts. Table 27 provides an overview of the themes generated from stakeholder interviews.

Table 27: Principal themes from stakeholder interviews

Theme	Description
1. Convergence on hazard realism	Both technical and clinical stakeholders agreed that the identified hazards were clinically realistic and accurately represented the primary risks of the system.
2. ML as a source of safety concern	Participants acknowledged that the ML components represent a unique and central source of safety risk that requires specific controls beyond standard software engineering.
3. The safety case as a governance & trust tool	The formal safety-case structure was perceived as a valuable tool for governance, inter-team communication, and building organisational confidence in the system's safety.
4. Data quality as a foundational prerequisite	Participants emphasised that the quality and representativeness of training data are the most fundamental factors in ensuring ML safety.
5. Human review as a safety control	Both stakeholder groups viewed the clinician fallback and oversight as an important safety control for the entire system.
6. Practical value and potential generalisability	The methodology itself was seen as a practically valuable and reusable process for assuring other, similar AI-driven clinical systems.

### 6.5.2.1 Theme 1: Convergence on the Realism of Hazard Identification

A foundational test of the methodology was whether its initial output, the identified hazards, was considered realistic and complete by those with deep domain expertise. The evidence shows a strong convergence on this point from both technical and clinical perspectives, supporting the credibility of the systematic hazard analysis process.

**Technical Perspective:** When presented with the four top-level hazards (e.g. under-referral, over-referral), the technical stakeholder immediately confirmed their validity, framing them in terms of system outcomes:

*“Yeah, I mean I think those are the top-level ones . . . you’re at a clinical level either sending too many people back or not enough people back.”*

- Technical stakeholder

**Clinical Perspective:** This view was directly mirrored by the clinical leadership, who confirmed the list and elaborated on it from a patient-centric viewpoint:

*“I would agree with that. The only other element to add would be making sure that, from the patient and clinician perspectives, you’re actually getting the right people to the right place.”*

- Clinical stakeholder (Medical Director)

**Synthesis and Implication:** This strong agreement from both sides demonstrates that the systematic hazard-analysis process is effective. It successfully abstracts complex system behaviours into a set of high-level hazards that are immediately recognisable and accepted as the most critical areas of concern. This finding provides direct support for proposition **P1** and begins to answer **CS-RQ1**.

### 6.5.2.2 Theme 2: ML as Source of Modern Safety Concerns

A central goal of the research was to investigate how ML-specific risks are perceived. The interviews revealed a sophisticated understanding that the ML components are not just another piece of software but represent a unique source of risk that requires specific mitigation strategies.

**Technical Perspective:** The CTO articulated the core challenge of non-determinism in ML systems. He contrasted it with traditional software by explaining that, even with mitigation strategies like confidence thresholds, the system’s behaviour is still fundamentally probabilistic:

*“You can have a level of confidence in it. So you can say, well, I’m only going to trust the answer if it’s over 95 % confidence . . . but even then, it’s not guaranteed, right? That’s the main difference, I suppose.”*

- Technical stakeholder

**Clinical Perspective:** The clinical stakeholders, while using different language, arrived at the same conclusion. They focused on the outcome of ML ambiguity and the necessity of system-level safeguards to manage this inherent uncertainty:

*“Has the AI interpreted what the person has said correctly? . . . you have to have rules in place to ensure that, if there is ambiguity, there is a safe fallback.”*

- Clinical stakeholder (Medical Director)

**Synthesis and Implication:** These perspectives are two sides of the same coin. The technical expert identifies the root cause (probabilistic outputs that are “not guaranteed”), while the clinical expert defines the necessary control (safe fallbacks for ambiguity). This validates the methodology’s approach of defining explicit ML safety requirements. This provides strong support for proposition **P2** and directly addresses **CS-RQ2**.

### 6.5.2.3 Theme 3: Safety Case as a Tool for Governance, Trust, and Communication

The evaluation explored the value of the formal safety-case structure itself. The findings indicate that its value extends beyond mere documentation into the realms of governance, inter-team alignment, and building confidence.

**Clinical Perspective:** The clinical leadership immediately saw the safety case as a crucial instrument of governance and for demonstrating due diligence to oversight bodies:

*“It gives you a governance wrapper that has a process and a structure . . . At the end of the day, you have to be sure from a board perspective that you’ve mitigated the risks as much as you can.”*

- Clinical stakeholder (Medical Director)

**Technical Perspective:** From the technical side, the value was seen in its ability to enforce rigour and ensure no risks were overlooked during development:

*“I think it just forces you to have that structure and think about it in a structured way to make sure you’ve covered all the things.”*

- Technical stakeholder

**Synthesis and Implication:** The safety case provides a “boundary object,” creating a shared language for technical and clinical teams to communicate effectively about risk. It translates low-level technical controls into high-level safety claims that leadership can understand. This supports proposition **P3** and answers **CS-RQ3**.

#### 6.5.2.4 Theme 4: The Foundational Importance of Data Quality in ML Safety

A final, powerful theme emerged regarding the source of ML model safety. The technical stakeholder spontaneously and emphatically identified data quality as the most fundamental prerequisite for a safe AI system.

**Technical Perspective:** When asked about the role of data in ensuring safety, the CTO was unequivocal:

*“Yeah, that’s key. If you haven’t got a good spread of data for what you’re trying to classify, it’s just not going to work . . . So that’s probably the most fundamental part of it.”*

- Technical stakeholder

**Synthesis and Implication:** This expert testimony provides direct and powerful validation for the explicit definition of ML data safety requirements. The fact that the technical lead of a deployed clinical system views data representativeness as “the most fundamental part” of safety confirms that a safety case for an ML system is incomplete without addressing the quality of the data on which it is trained. This finding provides further, compelling evidence for proposition **P2**.

### 6.5.2.5 Theme 5: Human Review as a Safety Control

A recurring theme throughout the interviews was the acknowledgement that the AI system does not operate in a vacuum. Both stakeholders emphasised that the system's ultimate safety is contingent on its integration into a human-led clinical workflow, where a person provides the final and most critical safety backstop.

**Clinical Perspective:** The clinical leadership viewed this human oversight not as a weakness, but as a necessary and fundamental design feature for any clinical AI. The Medical Director stressed the importance of having a pre-defined process for when the AI inevitably encounters ambiguity:

*"Has the AI interpreted what the person has said correctly?... you have to have rules in place to ensure that if there is ambiguity, that there is a safe fallback.."*  
- Clinical Stakeholder (Medical Director)

**Technical Perspective:** The CTO mirrored this view, describing human escalation as the definitive safety mechanism built into the system's architecture. When discussing how failures are mitigated, he stated:

*"it's the things you would expect, right? So it's about having sensible fallback mechanisms... ultimately it ends up with a person, right? That's the that's the ultimate fallback is that there's a human in the loop somewhere."*  
- Technical stakeholder (CTO)

**Synthesis and Implication:** This theme highlights that assuring a clinical CA is fundamentally about assuring a socio-technical system. The safety argument does not rest solely on the model's statistical performance but on the robustness of the interaction between the AI and the human clinician. This validates the inclusion of system-level requirements for escalation and fallback procedures as a critical part of the safety case.

### 6.5.2.6 Theme 6: The Practical Value and Potential Generalisability of the Safety Case Framework

Finally, beyond evaluating the safety of the specific system, stakeholders recognised the inherent value in the safety assurance *process itself*, viewing it as a transferable and valuable framework for future work.

**Clinical Perspective:** The clinical leadership identified the methodology as a key tool for organisational governance, providing a structured way to demonstrate due diligence and build confidence at an executive level:

*"It gives you a governance wrapper that has a process and a structure... At the end of the day, you have to be sure from a board perspective that you've mitigated the risks as much as you can."*

- Clinical Stakeholder (Medical Director)

**Technical Perspective:** The CTO confirmed the practical utility of the process from a development standpoint, noting its role in ensuring comprehensive risk coverage:

*"I think it just sort of forces you to have that structure and think about it in a structured way to make sure you've covered all the things... it's definitely useful to have to have done it, for sure."*

- Technical stakeholder (CTO)

**Synthesis and Implication:** This theme directly addresses the contribution of the research. It confirms that the safety case methodology is not merely an academic exercise but is perceived by real-world practitioners as a valuable and practical tool. The feedback supports the proposition that the framework is potentially generalisable, serving as a reusable pattern for other, similar AI-driven clinical systems. This finding is important for answering **CS-RQ4**.

### 6.5.3 Summary

The evaluation of the methodology, as applied to the Hinata case study, resulted in several important findings. The six emergent themes demonstrate that the safety

assurance framework is valid, relevant, and perceived as valuable by its intended users. The evidence collected provides strong support for all four of the study's guiding propositions.

The methodology successfully produced a set of hazards that were deemed realistic and complete (Theme 1), thereby answering **CS-RQ1** and supporting **P1**. The framework's emphasis on ML-specific risks and the foundational role of data quality was strongly validated (Themes 2 & 4), providing a clear answer to **CS-RQ2** and supporting **P2**. The formal safety case structure was found to be an effective tool for building trust, facilitating governance, and aligning stakeholder perspectives (Theme 3), which directly addresses **CS-RQ3** and supports **P3**. Finally, the recognition of the 'human in the loop' as the ultimate safety control (Theme 5) and the practical value and generalisability of the methodology itself (Theme 6) provide a comprehensive answer to **CS-RQ4** and strong support for **P4**.

Beyond validating the methodology, this chapter demonstrates contribution 3 of this thesis. The application of the systematic safety assurance methodology to the Hinata case study provided a concrete and practical answer to **RQ2a** ("How do we analyse and mitigate potential hazards arising from failures within clinical CAs?"), by using SHARD to identify precise failure modes (section 6.4.1), which were then consolidated into a formal, risk-assessed hazard log (section 6.4.3). Furthermore, the chapter directly answered **RQ2c** ("How can we develop a safety argument for clinical CAs?"), by progressing from the hazard log to a multi-layered set of safety requirements (section 6.4.4.2) and constructing a complete, evidence-based safety argument using GSN (Section 6.4.5). This end-to-end application with stakeholder evaluation constitutes the practical demonstration of contribution 3 to this thesis.

In conclusion, this stakeholder evaluation provides the critical evidence needed to validate the safety assurance framework. It shows that the methodology provides a practical process for analysing risk, defining and implementing mitigations, and arguing for the safety of real-world, high-risk clinical CAs, not just as isolated algorithms but as components within a broader socio-technical system.

# Chapter 7

## Conclusion and Future Work

This thesis addresses a safety-critical challenge: the systematic safety assurance for clinical conversational agents (CAs). As these AI-driven systems move from novel prototypes to integrated components of healthcare delivery, a structured, evidence-based approach to ensuring their safety is essential for protecting patients. This thesis argues that traditional software safety methods are necessary but insufficient for CAs, which are socio-technical systems whose safety depends on non-deterministic ML behaviour and on human-computer interaction. Through a series of use cases, from an exploratory prototype to an analysis of a real deployed system, this thesis develops, applies, and evaluates a systematic methodology for analysing, mitigating, and arguing the safety of clinical CAs. This chapter summarises the contributions of this work, provides a frank acknowledgement of its limitations, and outlines promising and necessary directions for future research in this rapidly evolving field.

### 7.1 Summary of Thesis Contributions

The primary contribution of this thesis is a structured safety assurance methodology for clinical CAs. This methodology was developed and validated across the thesis through three key, interconnected contributions, which directly address the research questions set out in the introduction.

#### 7.1.1 Contribution 1: A Failure Mode Taxonomy for clinical CAs

This research first established a comprehensive taxonomy of failure modes for clinical CAs, addressing ***RQ1 : What are the different failure modes in CAs that can lead to clinical hazards?*** The taxonomy, presented in Chapter 4, moves beyond generic

software failures to create a structured classification specific to the architecture of conversational agents. It systematically captures potential failures across the CA pipeline from **Recognition** (ASR errors) and **Understanding** (NLU misclassification) to **Decision Making** (dialogue policy failures) and **Response Generation** (NLG errors). Crucially, the taxonomy also incorporates failures originating from the wider socio-technical system, including the **User** (e.g. speech characteristics, human error, inadequate training) and the operational **Environment** (e.g. background noise). Grounded in the literature and refined through its application in the subsequent use cases, this taxonomy provided the vocabulary for the systematic hazard analyses, ensuring a systematic approach to identify what may go wrong and why. This addresses **RQ1** directly, and provides the basis for the hazard analysis and risk assessment activities in Chapters 5 and 6, thereby feeding into **RQ2a**.

### 7.1.2 Contribution 2: Uncertainty-Aware NLU for Safe Decision Making

This thesis develops and evaluates an uncertainty-aware Natural Language Understanding (NLU) approach for intent classification in a clinical conversational agent. This directly addresses **RQ2b: *How can we reduce the risks in clinical decision-making of CAs by robustly handling the inherent uncertainty in their understanding?***

In Chapter 5 (Section 5.2), a Bayesian LSTM with Monte Carlo Dropout is used to estimate predictive uncertainty and to detect out-of-distribution (OOD) utterances. The key idea is that the model is not only asked to decide “what did the patient say?” but also “how sure am I?” When confidence is low, or when the input is judged to be outside the trained domain, the system is designed to trigger a safe fallback (for example, ask the caller to clarify, or decline to act autonomously and escalate).

The empirical results on both in-distribution (ID) and OOD data show that calibrated uncertainty can be used as an explicit safety control. These results feed directly into the safety requirements that later appear in Chapter 5 (section 5.3) and Chapter 6.

This contribution does two things. First, it shows that uncertainty handling is not just a modelling tweak but a direct safety control in operation. Second, it provides quantitative evidence that underpins the ML model requirements (e.g. accuracy, recall, OOD detection thresholds) and fallback behaviours used in the formal safety

case in Chapter 6. In that sense, it establishes uncertainty estimation and safe escalation as explicit safety requirements for clinical CAs.

### 7.1.3 Contribution 3: A Methodology for Constructing and Evaluating a Safety Argument for an ML-based CA

The main contribution of this work is the detailed framework for building and evaluating a safety case for a real-world clinical CA, as demonstrated in the case study in chapter 6. This provides a detailed answer to **RQ2 : How do we systematically develop safety cases for clinical CAs?** and its sub-questions (**RQ2a, RQ2b, RQ2c**). The framework shows how to:

- **A Hybrid Method for Safety Analysis:** The framework includes a practical, two-part method for analysing system safety, addressing **RQ2a**. This involves first conducting a technical failure analysis (using methods like SHARD and quantitative NLU testing) to objectively identify vulnerabilities, and then validating the clinical significance of those vulnerabilities through a qualitative, exploratory case study involving domain experts.
- **A Multi-Layered Requirements Framework:** The methodology shows how to systematically derive a multi-layered set of safety requirements (system-level, ML Model, and ML data) directly from a formal hazard analysis, ensuring clear traceability from hazards to safety controls.
- **A Structure for an ML-Specific Safety Case:** The framework demonstrates how to structure a formal safety argument using Goal Structuring Notation (GSN). It provides a pattern for arguing the safety of the ML components by making specific, evidence-backed claims about data quality, model performance, and robustness.
- **An Integrated Evaluation Process:** The framework concludes by showing how the safety case can be evaluated with both technical and clinical stakeholders to assess its completeness, realism, and perceived value, providing confidence in the assurance process.

## 7.2 Research Limitations

It is important to acknowledge the limitations of this research, which themselves highlight key challenges in the field and provide avenues for future work.

- **Healthcare Data Sparsity and Accessibility:** A significant and recurring challenge in this research was the difficulty in obtaining large-scale, representative clinical conversational data. The prototype CAs developed in the use cases were trained on limited, sparse datasets. While this was a deliberate methodological choice to probe for "worst-case" failure modes, it mirrors a common real-world problem. Access to rich, correctly labelled conversational data from real patient interactions is heavily restricted by privacy regulations (e.g. GDPR) and the significant resources required for ethical approval and data anonymisation. Consequently, the performance of the ML models developed in this thesis is representative of these constraints and should not be interpreted as the state-of-the-art for production systems with access to larger, proprietary datasets.
- **Access to Real-World Operational Environments:** While this thesis had direct access to a deployed clinical system (Hinata), the earlier use cases were necessarily prototypes developed and tested in a controlled, offline environment. The validation of their failure modes relied on a combination of automated testing, simulated conversations, and stakeholder interviews. These are powerful and well-established research methods, but they remain imperfect proxies for the complexity and unpredictability of live operational use. A longitudinal study observing system performance and emergent failures over time with real patients would be required for a more complete picture of system safety in practice.
- **Patient and Community Involvement:** The validation of the use cases was primarily conducted through technical and clinical stakeholders rather than through direct patient or public involvement. Although the use cases focused on patient-facing conversational agents, patients and community representatives were not directly involved in identifying failure modes, reviewing the taxonomy, or evaluating the safety artefacts. This limits the extent to which the findings capture patient perspectives, including lived experience, trust, accessibility, health literacy, and acceptability of CA behaviour. As a result, the validation

should be interpreted primarily as a clinical and technical evaluation of the safety artefacts, rather than as a full assessment of patient acceptability or public trust.

- **Limited Interview Sample Size:** A further limitation is the small number of interview participants across the use cases. The qualitative evaluation involved a limited sample, ranging from one to three participants per use case. This was sufficient for exploratory and evaluative feedback on the relevance and credibility of the safety artefacts, but it limits the robustness and transferability of the findings. The themes should therefore be interpreted as indicative stakeholder feedback rather than as generalisable evidence across all clinical CA contexts. This limitation affects the transferability of the qualitative findings, particularly when considering different clinical domains, deployment settings, and stakeholder groups.

## 7.3 Future Work

The field of conversational AI is evolving at a rapid pace, primarily driven by the advent of Large Language Models (LLMs). This creates urgent and necessary directions for future research that build directly on the foundational safety principles established in this thesis.

The natural transition of this work is to explore the safety assurance of **LLM-based clinical CAs**. While the underlying technology changes significantly, moving from discrete NLU/DM components to a single, large general-purpose language model, often integrated with architectures such as Retrieval-Augmented Generation (RAG), the fundamental safety principles remain relevant. The core challenge is still one of managing uncertainty, preventing factual inaccuracies or hallucinations, and ensuring that the agent's behaviour aligns with clinical safety goals. The systematic, hazard-led assurance methodology presented in this thesis provides a reusable foundation for this future work.

The **Failure Mode Taxonomy** developed in this research remains relevant as a starting point. The high-level categories of understanding failure, decision-making failure, and response-generation failure still apply to LLM-based systems, although their causes and manifestations may differ. Future work should therefore focus

on **adapting and extending this taxonomy** to capture the distinctive and more complex failure modes of LLMs. This would include:

- **Investigating hallucinations as a hazard:** Developing methods to analyse, detect, and mitigate the risk of an LLM fabricating plausible but incorrect medical information.
- **Analysing prompt-based vulnerabilities:** Exploring failure modes related to prompt injection, prompt erosion, and adversarial inputs that could cause the LLM to behave in unsafe or unintended ways.
- **Assuring RAG architectures:** Developing safety analysis techniques, such as SHARD or HAZOP, that can be applied to the information flows within a RAG system, focusing on the interface between the LLM, the prompt logic, and the external knowledge base.

Future work should also extend the empirical evaluation of the methodology with larger and more diverse stakeholder groups. This should include patients and public contributors, as well as clinicians, developers, safety officers, and regulatory experts. In particular, patient and public involvement would help assess whether the identified failure modes, safety requirements, and safety case arguments reflect patient concerns around trust, accessibility, health literacy, and acceptable CA behaviour in real clinical use.

Assuring the safety of next-generation clinical CAs will require a greater emphasis on transparency, traceability, and evidence-based argumentation. The core contribution of this thesis, a systematic methodology for identifying hazards, deriving requirements, and constructing a defensible safety case, provides a practical foundation for addressing these future safety challenges.

# Appendix A

## Failure Mode Taxonomy for CAs

This appendix provides the failure mode taxonomy for clinical conversational agents (CAs) in a consolidated form for practical use. The taxonomy classifies failure modes across the software, hardware, user, and environmental aspects of the CA system. ML-related model and data faults are discussed separately in Section 4.5.2 and Table 6 as supporting causal refinements for software-related failures.

206

Table 28: Consolidated failure mode taxonomy for CAs

Category	Component	Failure mode	Typical causes
Software	Recognition	Inaccurate transcription	<ul style="list-style-type: none"><li>• Non-native accent; very fast / slow speech</li><li>• Background noise</li><li>• Acoustic model trained on out-of-domain speech [173, 174]</li></ul>
		Word-substitution (similar words)	<ul style="list-style-type: none"><li>• Disfluencies or repetitions</li><li>• Weak language-model priors for phonetic neighbours [175]</li></ul>

(continued from previous page)

Category	Component	Failure mode	Typical causes
	<b>Understanding</b>	Misunderstanding (incorrect intent / slot)	<ul style="list-style-type: none"> <li>• Class imbalance; incomplete training set [176, 177]</li> <li>• Annotation errors</li> <li>• Model over-fitting / high uncertainty</li> </ul>
		Non-understanding intent	<ul style="list-style-type: none"> <li>• Out-of-domain utterance; unknown words</li> <li>• Upstream ASR error</li> <li>• No OOD detector deployed [178]</li> </ul>
	<b>Decision Making</b>	Overlooked conversation context	<ul style="list-style-type: none"> <li>• Dialogue-state window too short [179]</li> <li>• Error propagation from ASR / NLU</li> </ul>
		Incorrect system action	<ul style="list-style-type: none"> <li>• Sparse training examples for policy [180]</li> <li>• Error propagation from ASR / NLU</li> </ul>
	<b>Response Generation</b>	Incorrect/ambiguous semantics DA	<ul style="list-style-type: none"> <li>• Faulty mapping from DST to template</li> </ul>
		Unnatural / verbose utterance	<ul style="list-style-type: none"> <li>• Templates not reflective of system design</li> <li>• Repetition [1]</li> </ul>
	<b>UI / UX</b>	Unconstrained user input	<ul style="list-style-type: none"> <li>• Design choice; no guided prompts [5]</li> </ul>
		Poor conversational design	<ul style="list-style-type: none"> <li>• Repetitive “I don’t understand” loops</li> <li>• Confusing or ambiguous prompts</li> </ul>
<b>Hardware</b>	<b>Microphone</b>	Low-quality audio	<ul style="list-style-type: none"> <li>• Damaged or low-spec sensor [181]</li> </ul>
		Muted / faulty microphone	<ul style="list-style-type: none"> <li>• Hardware fault; user muted unintentionally</li> </ul>
	<b>Speaker</b>	Echo / reverberation	<ul style="list-style-type: none"> <li>• Room acoustics; mic-speaker feedback loop [182]</li> </ul>

(continued from previous page)

Category	Component	Failure mode	Typical causes
	<b>Battery / Power</b>	Sudden shutdown	• Battery depletion during active session [183]
	<b>Processor/Memory</b>	Intermittent hardware faults	• Transient errors, silent data corruption, hardware traps • Fail-slow behavior [184, 185]
		Resource limitation	• Insufficient CPU/memory (esp. embedded devices) [186]
	<b>Network Connectivity</b>	Network disruption	• Connection loss, high latency, packet loss, congestion [185–188]
		Network fail-slow behaviour	• Gradual performance degradation of network components [185]
<b>User</b>	<b>Speech characteristics</b>	Inaccurate transcription	• Accent; extreme speaking rate
	<b>Interaction behaviour</b>	Misunderstanding or non-understanding	• Out-of-domain request; void input; mental-model mismatch [2]; unsupervised use
	<b>Human-in-the-loop reviewer/clinical supervisor</b>	Human error	• Cognitive overload, time pressure, inadequate training, incorrect data entry [189, 190]
	<b>Ethical concerns</b>	Incomplete information capture	• Privacy-driven withholding of information; incomplete data [191]; downstream mis-triage [4]

*(continued from previous page)*

<b>Category</b>	<b>Component</b>	<b>Failure mode</b>	<b>Typical causes</b>
<b>Environment</b>	<b>Acoustic environment</b>	Inaccurate transcription	• Background noise (traffic, music), corrupts the speech signal [192]; ASR errors
		Inaccurate transcription	• Echo/reverberation; large empty room; reflective surfaces degrading signal quality [182]

# Appendix B

## Exploratory Interview Questions - Use Case 1

**Context.** Interview with clinical stakeholder about early prototype (rule-based) symptom-checker CA.

- Q1.** When a patient speaks or types to the bot, *where* is it easiest for the bot to make a serious mistake?
- If that mistake happened with a real patient, what could go wrong for their care or safety?
- Q2.** Besides language misunderstandings, can you think of other problems (conversation design, UX, background noise, poor signal, clinician hand-off) that could put patients at risk?
- Q3.** Between under-reporting and over-reporting of symptoms, which creates the more serious clinical hazard, and why?
- Q4.** What safety measures, technical or procedural, could bring these failures down to an acceptable level? (e.g. user confirmation, restricted input, human-in-the-loop)
- Q5.** How useful would a *systematic* flow-by-flow safety analysis be for this simple prototype?
- Q6.** Would a single document listing hazards and mitigations (a safety case) add value for your team?

# Appendix C

## Evaluative Interview Questions - Case Study

**Context.** Interview with stakeholders involved in the real-world clinical deployment of Hinata, a full voice-based ML-powered CA. Questions are grouped by role: technical and clinical.

### Q1. Generic Questions (All Stakeholders)

- How well do the identified hazards and risks reflect what you would expect in this kind of system?
- Do you think the structure of the safety case (including GSN and evidence) helps make the safety argument clear and trustworthy? Why or why not?
- What parts of the safety case or supporting evidence were most convincing or useful, and which were less so?
- From your perspective, does the safety case adequately capture the key hazards associated with the conversational agent?

### Q2. Technical Stakeholder Questions

- What kinds of ML-related failures (e.g. misclassifications, out-of-distribution inputs) are most important to manage in this system?
- Do you think this performance threshold (e.g., 90%+ F1 score, recall, or model accuracy for NLU) adequately reflects the safety-critical nature of the system?
- What factors in the system's design (e.g., dialogue design, fallback mechanisms, human-in-the-loop strategies) help mitigate failures before they become clinical hazards?

- 
- Do you think the ML safety requirements (e.g., accuracy thresholds, robustness expectations, retraining plans) are realistic and sufficient?
  - What role does the quality and representativeness of the training data play in ensuring safe understanding by the conversational agent?
  - Would this type of safety case have been useful during the initial development of your system?
  - Could this type of safety case be generalised to other ML-based systems you've worked on? What might need to be different?

### **Q3. Clinical Stakeholder Questions**

- How could a misunderstanding or incomplete/incorrect response from Hinata lead to harm for a patient?
- How well do the identified hazards and risks reflect what you would expect in this system? Are there any clinical hazards you believe were overlooked or risk is not adequately calculated?
- Do you think the proposed safeguards, such as explicit confirmation or human review, can prevent harms to patients?
- Which parts of the supporting evidence (safety analysis, safety requirements, data-flow diagram) do you think are most convincing and which less so?
- Does the overall safety case convince you that Hinata is acceptably safe in this clinical context? Why or why not?

# Appendix D

## SHARD Analysis Tables - Case Study

Table 29: SHARD flow: 1.1 response generation

ID	Guideword	Deviation	Possible Causes	Effects	Detection/protection	Safety Recommendation
1	Omission	No response generated	Errors in custom actions, errors from previous components (backend, conversation engine)	No response generated	Unit testing of all custom actions (for instantiated RASA module) , integration tests	
2	Commission	N/A	N/A	N/A	N/A	N/A
3	Early	N/A	N/A	N/A	N/A	N/A
4	Late	Delayed response	Network delays	Delayed response	S/w and network optimizations	

ID	Guideword	Deviation	Possible Causes	Effects	Detection/protection	Safety Recommendation
5	Value	Incorrect response generated	Errors in custom actions, dialogue policy errors (forms, rules, next action generation), nlu errors	Unexpected response generated	Unit testing of all custom actions (for instantiated RASA module)	This can be the main outcome value for a particular symptom or part of it which decides the outcome, therefore, it needs to be traced back to all inflows from other RASA components

Table 30: SHARD flow: 1.1.1 NLG data

ID	Guideword	Deviation	Possible Causes	Effects	Detection/protection	Safety Recommendation
1	Omission	No NLG data for response or custom actions	S/w errors	No response generated	Testing for NLG response templates (for instantiated RASA module)	
2	Value	Incorrect NLG data	S/w errors	Incorrect response generated	Testing for NLG response templates (for instantiated RASA module)	
3	Value	Unnatural NLG data	Ambiguous responses	Understanding problems to patient	Clinical trial data with patients and consented conversations can be used to improve responses	A natural and grammatical correct response can reduce misinterpretation by patients

Table 31: SHARD flow: 1.2 next action

ID	Guideword	Deviation	Possible Causes	Effects	Detection/protection	Safety Recommendation
1	Omission	The next action is not generated	Errors in dialogue policy (forms, rules, stories), incorrect policy parameters	No response generated	Testing of stories, forms and rules, dialogue policy parameters recheck (for instantiated RASA module)	A default fallback action should be executed
2	Omission	Empty slot value	Errors in form, incorrect intent classification	Incorrect response generated	Testing of forms (for instantiated RASA module), robust intent classification, explicit confirmation	Explicit confirmation strategy to overcome empty/incorrect slot values
3	Value	Invalid next action generated	Errors in dialogue policy (forms, rules, stories), errors in custom actions	Unexpected response generated	Testing of stories, forms and rules, validations in custom actions (for instantiated RASA module)	

<b>ID</b>	<b>Guideword</b>	<b>Deviation</b>	<b>Possible Causes</b>	<b>Effects</b>	<b>Detection/protection</b>	<b>Safety Recommendation</b>
4	Value	Incorrect slot value	Errors in form, incorrect intent classification	Incorrect response generated	Testing of forms, robust intent classification, explicit confirmation	Explicit confirmation strategy to overcome empty/incorrect slot values

Table 32: SHARD flow: QC.1.2 next action

ID	Guideword	Deviation	Possible Causes	Effects	Detection/protection	Safety Recommendation
1	Omission	Empty slot value	Errors in form, incorrect intent classification	Incomplete (automated) clinical decision	Testing of forms (for instantiated RASA module), robust intent classification, explicit confirmation	Explicit confirmation strategy to overcome empty/incorrect symptom value to avoid incorrect automated safety check decision
2	Value	Incorrect slot value	Errors in form, incorrect intent classification	Incorrect (automated) clinical decision	Testing of forms, robust intent classification, explicit confirmation	Explicit confirmation strategy to overcome empty/incorrect symptom value to avoid incorrect automated safety check decision

Table 33: SHARD flow: 1.3 intent classification

ID	Guideword	Deviation	Possible Causes	Effects	Detection/protection	Safety Recommendation
1	Value	Incorrect intent identification	Lack of/incorrect training data, NLU model parameter incorrect	Incorrect response generated	Adding sufficient training examples, implementing fall back classifier and setting reasonable NLU confidence threshold (for instantiated RASA module)	Fall back intent classifier implementation, consented conversations should be used to add/improve NLU data

Table 34: SHARD flow: QC.1.3 intent classification

ID	Guideword	Deviation	Possible Causes	Effects	Detection/protection	Safety Recommendation
1	Value	Incorrect intent identification	Lack of/incorrect training data, NLU model parameter incorrect	Incorrect (automated) safety check decision	Adding sufficient training examples, implementing fall back classifier and setting reasonable NLU confidence threshold	Fall back intent classifier implementation, consented conversations should be used to add/improve NLU data

Table 35: SHARD flow: 1.3.1 NLU data

ID	Guideword	Deviation	Possible Causes	Effects	Detection/protection	Safety Recommendation
1	Omission	No/missing NLU training data	Problem acquiring NLU data	Unexpected/incorrect response generated	Adding sufficient training examples for each intent (for instantiated RASA module)	
2	Value	Incorrect NLU training data	Errors during development, problem acquiring NLU data	Incorrect response generated	Semantically clear and distinct training example for intents, adding sufficient training examples for each intent (for instantiated RASA module)	Consented conversations should be used to add/improve NLU data

Table 36: SHARD flow: QC.1.3.1 NLU data

ID	Guideword	Deviation	Possible Causes	Effects	Detection/protection	Safety Recommendation
1	Value	Incorrect NLU training data	Errors during development, problem acquiring NLU data	Incorrect (automated) safety check decision	Semantically clear and distinct training example for intents, adding sufficient training examples for each intent	Consented conversations should be used to add/improve NLU data

Table 37: SHARD flow: 3.0 text-to-speech

ID	Guideword	Deviation	Possible Causes	Effects	Detection/protection	Safety Recommendation
1	Omission	Hinata's response (text) not converted	TTS service error	Hinata's (current) response failed to reach to patient	Redundant TTS service	
2	Late	Hinata's response (text) converted with a delay	Delay from TTS service, network delays	Increased wait for patient on the call	Redundant TTS service with priority of service responding earlier	
3	Value	Hinata's response (text) incorrectly converted	TTS service error (low confidence)	Patient may face difficulty in understanding the response	Redundant TTS service	

Table 38: SHARD flow: 5.0 speech-to-text

ID	Guideword	Deviation	Possible Causes	Effects	Detection/protection	Safety Recommendation
1	Omission	Patient's utterance (speech) failed to convert	STT service error	Patient's call terminated	Redundant STT service	
2	Late	Patient's utterance (speech) converted with a delay	Delay from STT service, network delays	Increased wait for patient on the call	Redundant STT service with priority of service responding earlier	
3	Value	Patient's utterance (speech) incorrectly transcribed	STT service error	Increased chance of incorrect Hinata response (possible source to NLU errors)	Redundant STT service with priority of service having high confidence score	

Table 39: SHARD flow: QC.7.0 patient's call

ID	Guideword	Deviation	Possible Causes	Effects	Detection/protection	Safety Recommendation
1	Value	Incorrect patient's speech displayed	Errors in STT service, errors displaying on the dashboard	Increased chances of incorrect Safety check decision (human)	Redundancy of STT services, s/w testing of front-end	This is critical as supervisor submits symptom information based on the information on the dashboard. Ensure dashboard reflects the true transcript of user's speech to avoid safety check decision (human) errors

Table 40: SHARD flow: QC.8.0 supervisor input

ID	Guideword	Deviation	Possible Causes	Effects	Detection/protection	Safety Recommendation
1	Omission	Supervisor did not provide their input	Human error	Incomplete (supervised) safety check decision	Supervisor dashboard should signal a warning sign/banner if they do not enter their input in a specific time	
2	Early	Supervisor provided input earlier than expected	Human error	Incorrect (supervised) safety check decision	Enable input fields on the dashboard during patient summary response by Hinata	It is critical that information from supervisor is correct as supervisor's input has high priority over AI in final safety check decision

ID	Guideword	Deviation	Possible Causes	Effects	Detection/protection	Safety Recommendation
3	Value	Supervisor provided incorrect input	Human error	Incorrect (supervised) safety check decision	Addition of a confirmation check for the supervisor once they submit their input (possibly display summary of symptoms as understood by Hinata)	It is critical that information from supervisor is correct as supervisor's input has high priority over AI in final safety check decision

Table 41: SHARD flow: QC.9.0 store supervisor inputs

ID	Guideword	Deviation	Possible Causes	Effects	Detection/protection	Safety Recommendation
1	Omission	Input failed to store	S/w errors, server errors/offline, storage offline	Incomplete (supervised) safety check decision	S/w testing of storage, program logic	
2	Value	Incorrect input value stored	S/w errors	Incorrect (supervised) safety check decision	S/w testing	

Table 42: SHARD flow: QC.12.0 calculate safety check decision

ID	Guideword	Deviation	Possible Causes	Effects	Detection/protection	Safety Recommendation
1	Omission	Safety check decision cannot be calculated	S/w error, safety check data failed to retrieve	Null (automated) safety check decision	S/w testing of calculation and retrieval methods	
2	Omission	Safety check data failed to retrieve	S/w errors in retrieval, storage offline	Patient's follow up call outcome solely reliant on supervisor's safety check decision	S/w testing of code and storage	
3	Early	Safety check decision calculated early	S/w error, triggered before receiving all symptom values	Incorrect (automated) safety check decision	S/w testing, ensure the function is called once it receives response of all symptoms	
4	Late	Safety check decision calculated with a delay	Delays in network	Increased wait for patient on the call	S/w and network optimization	

<b>ID</b>	<b>Guideword</b>	<b>Deviation</b>	<b>Possible Causes</b>	<b>Effects</b>	<b>Detection/protection</b>	<b>Safety Recommendation</b>
5	Value	Invalid safety check data retrieved	S/w errors in retrieval, invalid data stored (9.0)	Incorrect (automated) safety check decision	S/w testing, ensure data is stored with correct values	
6	Value	Incorrect safety check decision calculated	S/w error in calculation, incorrect values stored (flow 9.0)	Patient's follow up call outcome solely reliant on supervisor's safety check decision	S/w testing, ensure correct values retrieved and stored (see flow 9.0)	

# Abbreviations

ACP	Assurance Claim Point
AI	Artificial Intelligence
ASR	Automatic Speech Recognition
CA	Conversational Agent
CAE	Claims, Arguments, and Evidence
CNN	Convolutional Neural Network
CSA	Clinical Skills Assessment
DA	Dialogue Act
DL	Deep Learning
DM	Dialogue Manager
DP	Dialogue Policy
DSC	Dynamic Safety Case
DST	Dialogue State Tracking
FMEA	Failure Modes and Effects Analysis
FN	False Negative
FP	False Positive
GDM	Gestational Diabetes Mellitus
GSN	Goal Structuring Notation
GUS	Genial Understander System
HAZOP	Hazard and Operability Study

HCI	Human-Computer Interaction
ID	In-Distribution
ISO	International Organization for Standardization
LLM	Large Language Model
LSTM	Long Short-Term Memory
MC	Monte Carlo
MHRA	Medicines and Healthcare products Regulatory Agency
ML	Machine Learning
NHS	National Health Service
NLG	Natural Language Generation
NLP	Natural Language Processing
NLU	Natural Language Understanding
NPSA	National Patient Safety Agency
ODD	Operational Design Domain
OOD	Out-of-Distribution
PHA	Process Hazard Analysis
PSEC	Physical Sciences Ethics Committee
RAG	Retrieval-Augmented Generation
RCGP	Royal College of General Practitioners
RNN	Recurrent Neural Network
SEIPS	Systems Engineering Initiative for Patient Safety
SHARD	System-Hazard Analysis and Resolution in Design
SLU	Spoken Language Understanding

SOP	Standard Operating Procedure
TTS	Text-to-Speech
UX	User Experience
WHO	World Health Organization

# References

- [1] L. Laranjo, A. G. Dunn, H.-L. Tong, A. B. Kocaballi, J. Chen, R. Bashir, D. Surian, B. Gallego, F. Magrabi, A. Y. S. Lau, and E. Coiera, “Conversational agents in healthcare: a systematic review,” *Journal of the American Medical Informatics Association*, vol. 25, no. 9, pp. 1248–1258, 2018.
- [2] L. Tudor Car, D. A. Dhinakaran, B. M. Kyaw, T. Kowatsch, S. Joty, Y. L. Theng, and R. Atun, “Conversational agents in health care: Scoping review and conceptual analysis,” *Journal of Medical Internet Research*, vol. 22, no. 8, p. e17158, 2020.
- [3] T. Schachner, R. van den Heuvel, and R. Keller, “Artificial intelligence-based conversational agents for chronic conditions: Systematic literature review,” *Journal of Medical Internet Research*, vol. 22, no. 9, p. e20701, 2020.
- [4] N. N. Abdullah, J. Tang, H. Fetрати, N. F. B. Kaukiah, S. B. Saharudin, V. S. Yong, and C. H. Yen, “MARIA (Medical Assistance and Rehabilitation Intelligent Agent) for Medication Adherence in Patients With Heart Failure: Empirical Results From a Wizard of Oz Systematic Conversational Agent Design Clinical Protocol,” *JMIR Cardio*, vol. 9, p. e52703, 2025.
- [5] T. W. Bickmore, H. Trinh, S. Olafsson, T. K. O’Leary, R. Asadi, N. M. Rickles, and R. Cruz, “Patient and consumer safety risks when using conversational assistants for medical information: An observational study of siri, alexa, and google assistant,” *Journal of Medical Internet Research*, vol. 20, no. 9, p. e11510, 2018.
- [6] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, “Concrete problems in ai safety,” *arXiv preprint arXiv:1606.06565*, 2016.
- [7] A. Vaidyam, H. Wisniewski, J. Halamka, M. Kashavan, and J. Torous, “Chatbots and conversational agents in mental health: A review of the psychiatric landscape,” *Can. J. Psychiatry*, vol. 64, no. 7, pp. 456–464, 2019.
- [8] D. D. Luxton, “Recommendations for the ethical use and design of artificial intelligent care providers,” *Artificial Intelligence in Medicine*, vol. 102, p. 101783, 2019.
- [9] M. Cascella, F. Semeraro, J. Montomoli, V. Bellini, O. Piazza, and E. Bignami, “The breakthrough of large language models release for medical applications: 1-year timeline and perspectives,” *Journal of Medical Systems*, vol. 48, no. 1, p. 22, 2024.
- [10] B. Wen, R. Norel, J. Liu, T. Stappenbeck, F. Zulkernine, and H. Chen, “Leveraging large language models for patient engagement: The power of

- conversational ai in digital health,” in *2024 IEEE International Conference on Digital Health (ICDH)*. IEEE, 2024, pp. 104–113.
- [11] H. Zhang, Q. Lou, and Y. Wang, “Towards safe ai clinicians: A comprehensive study on large language model jailbreaking in healthcare,” *arXiv preprint arXiv:2501.18632*, 2025.
- [12] J. R. E. Neo, J. S. Ser, and S. Tay, “Use of large language model-based chatbots in managing the rehabilitation concerns and education needs of outpatient stroke survivors and caregivers,” *Frontiers in Digital Health*, 2024.
- [13] X. Wang, N. X. Zhang, H. He, T. Nguyen, K.-H. Yu, H. Deng, C. Brandt, D. Bitterman, L. Pan, C.-Y. Cheng, J. Zou, and D. Liu, “Safety challenges of ai in medicine in the era of large language models,” 2024.
- [14] V. Hassija, V. Chamola, A. Mahapatra, A. Singal, D. Goel, K. Huang, S. Scardapane, I. Spinelli, M. Mahmud, and A. Hussain, “Interpreting black-box models: A review on explainable artificial intelligence,” *Cognitive Computation*, 2023.
- [15] M. Bhimani, A. Miller, J. D. Agnew, A. M. S., M. Raglow-Defranco, H. Mangat, M. Voisard, M. Taylor, S. Bierman-Lytle, V. Parikh, J. Ghukasyan, R. Lasko, S. Godil, A. Atreja, and S. Mukherjee, “Real-world evaluation of large language models in healthcare (rwe-llm): A new realm of ai safety & validation,” 2025.
- [16] R. Lwears, “Rethinking healthcare as a safety-critical industry,” 2012.
- [17] D. Gaba, “Structural and organizational issues in patient safety: A comparison of health care to other high-hazard industries,” *Calif. Manage. Rev*, 2000.
- [18] J. Corrigan, “Crossing the quality chasm,” in building a better delivery system: A new engineering/health care partnership,” 2005.
- [19] D. Gaba, “Anaesthesiology as a model for patient safety in health care,” *British Medical Journal*, 2000.
- [20] C. Macrae and C. Vincent, “Learning from failure: the need for independent safety investigation in healthcare,” *J. R. Soc. Med*, 2014.
- [21] B. Sexton, E. Thomas, and R. Helmreich, “Error, stress, and teamwork in medicine and aviation: Cross sectional surveys,” *Ugeskr. Laeger*, 2000.
- [22] F. Durso and F. Drews, “Health care, aviation, and ecosystems: A socio-natural systems perspective,” *Curr. Dir. Psychol. Sci*, 2010.
- [23] N. Kapur, A. Parand, T. Soukup, T. Reader, and N. Sevdalis, “Aviation and healthcare: a comparative review with implications for patient safety,” *JRSM Open*, 2016.
- [24] A. M. Instrumentation, *Risk and Reliability in Healthcare and Nuclear Power: Learning from Each Other*. Arlington, VA: Association for the Advancement of Medical Instrumentation, 2013.

- 
- [25] G. Grote, "Safety management in different high-risk domains - all the same?" *Saf. Sci.*, 2012.
- [26] C. Bosk, M. Dixon-Woods, C. Goeschel, and P. Pronovost, "Reality check for checklists," *Lancet*, 2009.
- [27] S. Goldhaber-Fiebert and C. Macrae, "Emergency manuals: How quality improvement and implementation science can enable better perioperative management during crises," *Anesthesiology Clinics*, 2018.
- [28] E. Liberati, M. Peerally, and M. Dixon-Woods, "Learning from high risk industries may not be straightforward: A qualitative study of the hierarchy of risk controls approach in healthcare," *Int. J. Qual. Heal. Care*, 2018.
- [29] C. Vincent, "Patient safety," 2010.
- [30] W. H. Organization, "Patient safety," available: link.
- [31] N. England, "The nhs patient safety strategy," 2019, available: link.
- [32] S. W. N. Digital), "Dcb0160: Clinical risk management: its application in the deployment and use of health it systems," 2018, available: link.
- [33] N. England, "Nrls national patient safety incident reports: commentary," 2020, available: link.
- [34] T. Kelly, "A systematic approach to safety case management," *SAE Int*, 2003.
- [35] R. Hawkins, I. Habli, T. Kelly, and J. McDermid, "Assurance cases and prescriptive software safety certification: A comparative study," *Saf. Sci.*, vol. 59, pp. 55–71,, 2013.
- [36] SCSC, "The goal structuring notation (gsn) standard," 2025, available: link.
- [37] L. Adelard, "Claims, arguments and evidence (cae)(2019)," 2020.
- [38] T. Kletz, *HAZOP and HAZAN: identifying and assessing process industry hazards*. IChemE, 1999.
- [39] D. J. Pumfrey, "The principled design of computer system safety analyses." Ph.D. dissertation, University of York, 1999.
- [40] D. Reifer, "Software failure modes and effects analysis," *IEEE Trans. Reliab.*, vol. 28, no. 3, pp. 247–249,, 1979.
- [41] N. Leveson and P. Harvey, "Software fault tree analysis," *J. Syst. Softw.*, vol. 3, no. 2, pp. 173–181,, 1983.
- [42] J. Peterson, "Petri nets," *ACM Comput. Surv.*, vol. 9, no. 3, pp. 223–252,, 1977.
- [43] A. Haynes, W. Berry, and A. Gawande, "What do we know about the safe surgery checklist now?" *Annals of Surgery*, 2015.

- 
- [44] J. Ward, P. Clarkson, P. Buckle, J. Berman, R. Lim, and G. Jun, "Prospective hazard analysis: tailoring prospective methods to a healthcare context," 2010.
- [45] Y. Jia, "Embracing machine learning in safety assurance in healthcare," Ph.D. dissertation, University of York, 2021.
- [46] R. Clay-Williams and L. Colligan, "Back to basics: Checklists in aviation and healthcare," *BMJ Quality and Safety*, 2015.
- [47] P. Pronovost, "Sustaining reductions in catheter related bloodstream infections in michigan intensive care units: Observational study," *BMJ*, 2010.
- [48] K. Catchpole and S. Russ, "The problem with checklists," *BMJ Qual. Saf*, 2015.
- [49] K. D. Sharma and S. Srivastava, "Failure mode and effect analysis (fmea) implementation: a literature review," *Journal of Advance Research in Aeronautics and Space Science*, vol. 5, no. 1, pp. 1–17, 2018.
- [50] J. DeRosier, E. Stalhandske, J. Bagian, and T. Nudell, "Using health care failure mode and effect analysis: the va national center for patient safety's prospective risk analysis system," *Jt. Comm. J. Qual. Improv*, 2002.
- [51] M. Habraken, T. Schaaf, I. Leistikow, and P. Reijnders-Thijssen, "Prospective risk analysis of health care processes: A systematic evaluation of the use of hfmeatm in dutch health care," *Ergonomics*, 2009.
- [52] B. Franklin, N. Shebl, and N. Barber, "Failure mode and effects analysis: Too little for too much?" *BMJ Quality and Safety*, 2012.
- [53] J. Annett, "Hierarchical task analysis (hta)," in *Handbook of human factors and ergonomics methods*. CRC Press, 2004, pp. 355–363.
- [54] S. Sarker, A. Chang, T. Albrani, and C. Vincent, "Constructing hierarchical task analysis in surgery," *Surg. Endosc. Other Interv. Tech*, 2008.
- [55] R. Lane, N. Stanton, and D. Harrison, "Applying hierarchical task analysis to medication administration errors," *Appl. Ergon*, 2006.
- [56] I. Habli, "Development and piloting of a software tool to facilitate proactive hazard and risk analysis of health information technology," *Health Informatics J*, 2019.
- [57] J. McDermid, M. Nicholson, D. Pumfrey, and P. Fenelon, "Experience with the application of hazop to computer-based systems," in *COMPASS'95 Proceedings of the Tenth Annual Conference on Computer Assurance Systems Integrity, Software Safety and Process Security*, 1995, p. 37–48.
- [58] J. Dunj3, V. Fthenakis, J. A. V3lchez, and J. Arnaldos, "Hazard and operability (hazop) analysis. a literature review," *Journal of hazardous materials*, vol. 173, no. 1-3, pp. 19–32, 2010.

- 
- [59] E. Elhosary and O. Moselhi, "Automation for hazop study: A state-of-the-art review and future research directions," *Journal of Information Technology in Construction (ITcon)*, vol. 29, no. 33, pp. 750–777, 2024.
- [60] GOV.UK, "International ai safety report 2025," The Stationery Office, London, Tech. Rep., Feb 2025, this report is cited across references 30, 31, and 32.
- [61] J. Guiochet, "Hazard analysis of human-robot interactions with. hazop-un Safety science," *Safety science*, vol. 84, pp. 225–237, 2016.
- [62] D. Martin-Guillerez, J. Guiochet, D. Powell, and C. Zanon, "A uml-based method for risk analysis of human-robot interactions," in *Proceedings of the 2nd International Workshop on Software Engineering for Resilient Systems*, 2010, pp. 32–41.
- [63] M. A. Sujana, I. Habli, T. P. Kelly, S. Pozzi, and C. W. Johnson, "Should healthcare providers do safety cases? lessons from a cross-industry review of safety case practices," *Safety science*, vol. 84, pp. 181–189, 2016.
- [64] B. Singh, P. Jukes, B. Poblete, and B. Wittkower, "20 years on lessons learned from piper alpha. the evolution of concurrent and inherently safe design," *Journal of Loss Prevention in the Process Industries*, vol. 23, no. 6, pp. 936–953, 2010.
- [65] U. Food and D. Administration, "Infusion pumps total product life cycle: Guidance for industry and fda staff," *Food Drug Adm. Stand.*, pp. 766–910, 2014.
- [66] E.U., "Regulation (eu) 2017/745 of the european parliament and of the council of 5 april 2017 on medical devices, amending directive 2001/83/ec, regulation (ec) no 178/2002 and regulation (ec) no 1223/2009 and repealing council directives 90/385/eec and 93/42/ee," 2017, online]. Available: link.
- [67] ISO, "Iso 14971:2019 medical devices - application of risk management to medical devices," 2019, available: link.
- [68] ISO, "Iso 13485:2016 medical devices - quality management systems - requirements for regulatory purposes," 2016, available: link.
- [69] MHRA, "Medical device stand-alone software including apps (including) ivdmds," 2014, available: link.
- [70] MHRA, "Medical devices: conformity assessment and the ukca mark," 31-12-2020, available: link.
- [71] MHRA, "Software and ai as a medical device change programme roadmap," 06-2023, available: link.
- [72] C. B. Weinstock and J. B. Goodenough, *Towards an assurance case practice for medical devices*. Carnegie Mellon University, Software Engineering Institute, 2009.

- 
- [73] S. W. N. Digital), “Dcb0129: Clinical risk management: its application in the manufacture of health it systems,” 2018, available: link.
- [74] D. Jurafsky and J. H. Martin, “Speech and language processing (3rd ed. draft),” 2025, available: link.
- [75] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 2009.
- [76] S. Diederich, A. Brendel, and L. Kolbe, “Towards a taxonomy of platforms for conversational agent design,” 2019.
- [77] J. Weizenbaum, “Eliza-a computer program for the study of natural language communication between man and machine,” *Commun. ACM*, 1966.
- [78] K. Colby, S. Weber, and F. Hilf, “Artificial paranoia,” *Artif. Intell*, 1971.
- [79] K. Colby, F. Hilf, S. Weber, and H. Kraemer, “Turing-like indistinguishability tests for the validation of a computer simulation of paranoid processes,” *Artif. Intell*, 1972.
- [80] I. Serban, R. Lowe, P. Henderson, L. Charlin, and J. Pineau, “A survey of available corpora for building data-driven dialogue systems: The journal version,” *Dialogue and Discourse*, 2018.
- [81] C. Segura, À. Palau, J. Luque, M. R. Costa-Jussà, and R. E. Banchs, “Chatbol, a chatbot for the spanish “la liga”,” in *9th international workshop on spoken dialogue system technology*. Springer, 2019, pp. 319–330.
- [82] A. Ritter, C. Cherry, and B. Dolan, “Unsupervised modeling of twitter conversations,” 2010.
- [83] J. Cahn, “Chatbot: Architecture, design, & development,” *University of Pennsylvania School of Engineering and Applied Science Department of Computer and Information Science*, 2017.
- [84] D. Bobrow, R. Kaplan, M. Kay, D. Norman, H. Thompson, and T. Winograd, “Gus, a frame-driven dialog system,” *Artif. Intell*, 1977.
- [85] Z. Zhang, R. Takanobu, Q. Zhu, M. Huang, and X. Zhu, “Recent advances and challenges in task-oriented dialog systems,” *Science China Technological Sciences*, vol. 63, no. 10, pp. 2011–2027, 2020.
- [86] H. Chen, X. Liu, D. Yin, and J. Tang, “A survey on dialogue systems: Recent advances and new frontiers,” *Acm Sigkdd Explor. Newsl*, vol. 19, no. 2, pp. 25–35, 2017.
- [87] L. Car, “Conversational agents in health care: Scoping review and conceptual analysis,” *J. Med. Internet Res*, vol. 22, no. 8, pp. 17 158, 2020.

- 
- [88] S. Louvan and B. Magnini, “Recent neural methods on slot filling and intent classification for task-oriented dialogue systems: A survey,” *arXiv preprint arXiv:2011.00564*, 2020.
- [89] K. Yao, G. Zweig, M.-Y. Hwang, Y. Shi, and D. Yu, “Recurrent neural networks for language understanding,” in *Interspeech*, 2013, pp. 2524–2528.
- [90] K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi, “Spoken language understanding using long short-term memory neural networks,” in *2014 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2014, pp. 189–194.
- [91] D. Hakkani-Tür, “Multi-domain joint semantic frame parsing using bi-directional rnn-lstm,” *Interspeech*, p. 715–719, 2016.
- [92] N. Mrkšić, D. Séaghdha, T. Wen, B. Thomson, and S. Young, “Neural belief tracker: Data-driven dialogue state tracking,” 2017.
- [93] M. Henderson, B. Thomson, and S. Young, “Word-based dialog state tracking with recurrent neural networks,” in *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2014, p. 292–299.
- [94] V. Mnih, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [95] Z. Lipton, X. Li, J. Gao, L. Li, F. Ahmed, and L. Deng, “Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.
- [96] N. Nayak, D. Hakkani-Tür, M. A. Walker, and L. P. Heck, “To plan or not to plan? discourse planning in slot-value informed sequence to sequence models for language generation.” in *INTERSPEECH*, 2017, pp. 3339–3343.
- [97] K. Deemter, M. Theune, and E. Krahmer, “Real versus template-based natural language generation: A false opposition?” *Comput. Linguist*, vol. 31, no. 1, pp. 15–24, 2005.
- [98] J. Pereira and Díaz, “Using health chatbots for behavior change: A mapping study,” *J. Med. Syst*, vol. 43, no. 5, 2019.
- [99] J.-E. Bibault, B. Chaix, P. Nectoux, A. Pienkowski, A. Guillemasé, and B. Brouard, “Healthcare ex machina: Are conversational agents ready for prime time in oncology?” *Clin. Transl. Radiat. Oncol*, vol. 16, pp. 55–59, 2019.
- [100] G. Cameron, “Best practices for designing chatbots in mental healthcare – a case study on ihelpr,” *Hci*, pp. 1–5, 2018.
- [101] B. Inkster, S. Sarda, and V. Subramanian, “An empathy-driven, conversational artificial intelligence agent (wysa) for digital mental well-being: Real-world data evaluation mixed-methods study,” *JMIR mHealth uHealth*, vol. 6, no. 11, pp. 1–14, 2018.

- [102] S. Mujeeb, M. Hafeez, and T. Arshad, "Aquabot: A diagnostic chatbot for achluophobia and autism," *Int. J. Adv. Comput. Sci. Appl*, vol. 8, no. 9, pp. 209–216,, 2017.
- [103] S. Jungmann, T. Klan, S. Kuhn, and F. Jungmann, "Accuracy of a chatbot (ada) in the diagnosis of mental disorders: Comparative case study with lay and expert users," *JMIR Form. Res*, vol. 3, no. 4, pp. 13 863,, 2019.
- [104] C. Huang, M. Yang, Y. Chen, M. Wu, and K. Chen, "A chatbot-supported smart wireless interactive healthcare system for weight control and health promotion," 2019.
- [105] A. Fadhil, "Addressing b challenges b in b promoting b healthy b lifestyles : b the b ai-chatbot b approach b," 2017.
- [106] L. Vaira, M. Bochicchio, M. Conte, F. Casaluci, and A. Melpignano, "Mama bot: A system based on ml and nlp for supporting women and families during pregnancy," *ACM Int. Conf. Proceeding Ser*, pp. 273–277,, 2018.
- [107] D. Madhu, C. Jain, E. Sebastain, S. Shaji, and A. Ajayakumar, "A novel approach for medical assistance using trained chatbot," *Proc. Int. Conf. Inven. Commun. Comput. Technol. ICICCT 2017*, no. Icicct, pp. 243–246,, 2017.
- [108] S. Olafsson, T. O'Leary, R. Asadi, N. Rickles, and R. Cruz, "Patient and consumer safety risks when using conversational assistants for medical information: An observational study of siri, alexa, and google assistant," *J. Med. Internet Res*, vol. 20, no. 9, pp. 1–13,, 2018.
- [109] M. Ma, M. Skubic, K. Ai, and J. Hubbard, "Angel-echo: A personalized health care application," *Proc. - 2017 IEEE 2nd*, pp. 258–259,, 2017.
- [110] A. Lokman and J. Zain, "An architectural design of virtual dietitian (vidi) for diabetic patients," *Proc. - 2009 2nd IEEE Int. Conf. Comput. Sci. Inf. Technol. ICCSIT 2009*, no. ViDi, pp. 408–411,, 2009.
- [111] N. S. Ahmad, M. H. Sanusi, M. H. Abd Wahab, A. Mustapha, Z. A. Sayadi, and M. Z. Saringat, "Conversational bot for pharmacy: a natural language approach," in *2018 IEEE conference on open systems (ICOS)*. IEEE, 2018, pp. 76–79.
- [112] J. Allen, G. Ferguson, N. Blaylock, D. Byron, N. Chambers, M. Dzikovska, L. Galescu, and M. Swift, "Chester: Towards a personal medication advisor," *Journal of biomedical informatics*, vol. 39, no. 5, pp. 500–513, 2006.
- [113] A. Piau, R. Crissey, D. Brechemier, L. Balardy, and F. Nourhashemi, "A smartphone chatbot application to optimize monitoring of older patients with cancer," *Int. J. Med. Inform*, vol. 128, no. May, pp. 18–23,, 2019.
- [114] B. Chaix, "When chatbots meet patients: One-year prospective study of conversations between patients with breast cancer and a chatbot," *J. Med. Internet Res*, vol. 21, no. 5, pp. 1–7,, 2019.

- [115] S. Razzaki, A. Baker, Y. Perov, K. Middleton, J. Baxter, D. Mullarkey, D. Sangar, M. Taliercio, M. Butt, A. Majeed, A. DoRosario, M. Mahoney, and S. Johri, "A comparative study of artificial intelligence and human doctors for the purpose of triage and diagnosis," *arXiv preprint arXiv:1806.01859*, 2018.
- [116] H. Liu, H. Peng, X. Song, C. Xu, and M. Zhang, "Using ai chatbots to provide self-help depression interventions for university students: A randomized trial of effectiveness," *Internet Interventions*, vol. 27, p. 100495, 2022.
- [117] A. Aggarwal, C. C. Tam, D. Wu, X. Li, and S. Qiao, "Artificial intelligence–based chatbots for promoting health behavioral changes: systematic review," *Journal of medical Internet research*, vol. 25, p. e40789, 2023.
- [118] A. Palanica, P. Flaschner, A. Thommandram, M. Li, and Y. Fossat, "Physicians' perceptions of chatbots in health care: Cross-sectional web-based survey," *J. Med. Internet Res*, vol. 21, no. 4, pp. 1–10,, 2019.
- [119] E. Levin and A. Levin, "Evaluation of spoken dialogue technology for real-time health data collection," *J. Med. Internet Res*, vol. 8, no. 4, pp. 1–17,, 2006.
- [120] R. Moore, "Studies in conversational ux design," pp. 181–204,, 2018.
- [121] A. Miner, A. Milstein, S. Schueller, R. Hegde, C. Mangurian, and E. Linos, "Smartphone-based conversational agents and responses to questions about mental health, interpersonal violence, and physical health," *JAMA Intern. Med*, vol. 176, no. 5, pp. 619–625,, 2016.
- [122] A. Heerden, X. Ntinga, and K. Vilakazi, "The potential of conversational agents to provide a rapid hiv counseling and testing services," 2017.
- [123] T. Hodgson, F. Magrabi, and E. Coiera, "Efficiency and safety of speech recognition for documentation in the electronic health record," *J. Am. Med. Informatics Assoc*, 2017.
- [124] P. Xu and R. Sarikaya, "Convolutional neural network based triangular crf for joint intent detection and slot filling," in *2013 ieee workshop on automatic speech recognition and understanding*. IEEE, 2013, pp. 78–83.
- [125] P. Budzianowski and I. Vulić, "Hello, it's gpt-2—how can i help you? towards the use of pretrained language models for task-oriented dialogue systems," 2019, arXiv Prepr. arXiv1907.05774,.
- [126] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, no. 1, pp. 5998–6008, 2017.
- [127] R. Bommasani, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.
- [128] M. Moor, O. Banerjee, Z. S. H. Abad, H. M. Krumholz, J. Leskovec, E. J. Topol, and P. Rajpurkar, "Foundation models for generalist medical artificial intelligence," *Nature*, vol. 616, no. 7956, pp. 259–265, 2023.

- [129] R. López-Cózar, Z. Callejas, G. Espejo, and D. Griol, “Enhancement of conversational agents by means of multimodal interaction,” in *Conversational agents and natural language interaction: techniques and effective practices*. IGI Global Scientific Publishing, 2011, pp. 223–252.
- [130] Y. Zheng, G. Chen, and M. Huang, “Out-of-domain detection for natural language understanding in dialog systems,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 1198–1209, 2020.
- [131] Y. Gal, “Uncertainty in deep learning,” Ph.D. dissertation, University of Cambridge, 2016.
- [132] A. Kendall and Y. Gal, “What uncertainties do we need in bayesian deep learning for computer vision?” *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.
- [133] T. G. Dietterich, “Ensemble methods in machine learning,” in *International workshop on multiple classifier systems*. Springer, 2000, pp. 1–15.
- [134] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavalda, “New ensemble methods for evolving data streams,” in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, p. 139–148.
- [135] S. Wager, S. Wang, and P. Liang, “Dropout training as adaptive regularization,” 2013, arXiv Prepr. arXiv1307.1493,.
- [136] Y. Gal and Z. Ghahramani, “Dropout as a bayesian approximation: Representing model uncertainty in deep learning,” in *International conference on machine learning*. PMLR, 2016, pp. 1050–1059.
- [137] T. Bocklisch, J. Faulkner, N. Pawelczyk, and J. Röbler, “Rasa: Open source language understanding and dialogue management,” in *NIPS 2017, Workshop on Conversational AI*, 2017.
- [138] S. Singh, S. Gaikwad, R. Mishra, and P. Singh, “An effective college enquiry chatbot using rasa,” in *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*. IEEE, 2022, pp. 2023–2028.
- [139] H. Qi, L. Pan, A. Sood, A. Shah, L. Kunc, M. Yu, and S. Potdar, “Benchmarking commercial intent detection services with practice-driven evaluations,” *arXiv preprint arXiv:2012.03929*, 2020.
- [140] M. Al-Shaibani, “A comparison of intent classification in conversational agents,” in *2020 3rd International Conference on Computer Applications & Information Security (ICCAIS)*. IEEE, 2020, pp. 1–6.
- [141] N. Mehandru, B. Y. Miao, E. R. Almaraz, M. Sushil, A. J. Butte, and A. Alaa, “Evaluating large language models as agents in the clinic,” *NPJ digital medicine*, vol. 7, no. 1, p. 84, 2024.

- [142] A. Abd-Alrazaq, M. Al-Ghafri, H. Al-Saqri, H. Al-Badi, S. Al-Harrasi, and H. Al-Azri, "A comparative study of the usability of the most common conversational agents for medical diagnosis," *Journal of Multidisciplinary Healthcare*, pp. 155–172, 2024.
- [143] T. Bunk, D. Varshneya, V. Vlasov, and A. Nichol, "Diet: Lightweight language understanding for dialogue systems," *arXiv preprint arXiv:2004.09936*, 2020.
- [144] T. Bocklisch, T. Werkmeister, D. Varshneya, and A. Nichol, "Task-oriented dialogue with in-context learning," *arXiv preprint arXiv:2402.12234*, 2024.
- [145] R. K. Yin, *Case Study Research and Applications: Design and Methods (6th ed.)*. SAGE Publications, 2018.
- [146] K. M. Cresswell, D. W. Bates, and A. Sheikh, "Ten key considerations for the successful implementation and adoption of large-scale health information technology," *Journal of the American Medical Informatics Association*, vol. 20, no. e1, pp. e9–e13, 2013.
- [147] F. Alloatti, A. Bosca, L. Di Caro, and F. Pieraccini, "Diabetes and conversational agents: the aida project case study," *Discover Artificial Intelligence*, vol. 1, no. 1, p. 4, 2021.
- [148] R. E. Stake, *The art of case study research*. Sage, 1995.
- [149] K. M. Eisenhardt, "Building theories from case study research," *Academy of management review*, vol. 14, no. 4, pp. 532–550, 1989.
- [150] E. Edwards, "Man and machine: Systems for safety," in *Proceedings of the British Airline Pilots Association Technical Symposium*. London: British Airline Pilots Association, 1972, pp. 21–36.
- [151] R. Hawkins, I. Habli, T. Kelly, and J. McDermid, "Assurance cases and prescriptive software safety certification: A comparative study," *Safety science*, vol. 59, pp. 55–71, 2013.
- [152] M. Laymouna, Y. Ma, D. Lessard, T. Schuster, K. Engler, and B. Lebouché, "Roles, users, benefits, and limitations of chatbots in health care: Rapid review," *Journal of Medical Internet Research*, vol. 26, p. e56930, 2024.
- [153] S. Sarkar, M. Gaur, L. K. Chen, M. Garg, and B. Srivastava, "A review of the explainability and safety of conversational agents for mental health to identify avenues for improvement," *Frontiers in Artificial Intelligence*, vol. 6, p. 1229805, 2023.
- [154] K. D. Bailey, *Typologies and Taxonomies: An Introduction to Classification Techniques*, ser. Quantitative Applications in the Social Sciences. Thousand Oaks, CA: SAGE Publications, Inc, 1994, vol. 102.
- [155] R. C. Nickerson, U. Varshney, and J. Muntermann, "A method for taxonomy development and its application in information systems," *European Journal of Information Systems*, vol. 22, no. 3, pp. 336–359, 2013.

- 
- [156] R. E. Blackwelder, *Taxonomy: A Text and Reference Book*. New York: John Wiley & Sons, 1967.
- [157] B. M. O'Halloran, R. B. Stone, and I. Y. Tumer, "A failure modes and mechanisms naming taxonomy," in *2012 Proceedings - Annual Reliability and Maintainability Symposium (RAMS)*, 2012, pp. 1–6.
- [158] International Electrotechnical Commission, *IEC 60050-192: International Electrotechnical Vocabulary - Part 192: Dependability*. Geneva, Switzerland: IEC, 2015.
- [159] International Electrotechnical Commission, *IEC 60812:2018 - Failure modes and effects analysis (FMEA and FMECA)*. Geneva, Switzerland: IEC, 2018.
- [160] J. Reason, *Managing the Risks of Organizational Accidents*. Aldershot, Hants, England: Ashgate, 1997.
- [161] S. S. Alizadeh and P. Moshashaei, "The bowtie method in safety management system: A literature review," *Scientific Journal of Review*, vol. 4, no. 9, pp. 133–138, 2015.
- [162] International Organization for Standardization, *ISO 14971:2019 - Medical devices — Application of risk management to medical devices*. Geneva, Switzerland: ISO, 2019.
- [163] International Electrotechnical Commission, *IEC 61882:2016 - Hazard and operability studies (HAZOP studies) - Application guide*. Geneva, Switzerland: IEC, 2016.
- [164] M. Rausand and A. Hoyland, *System Reliability Theory: Models, Statistical Methods, and Applications*, 2nd ed. Hoboken, NJ: John Wiley & Sons, 2011.
- [165] F. H. Hawkins, *Human Factors in Flight*. Aldershot, Hants, England: Gower Technical Press, 1987.
- [166] J. D. Sterman, *Business Dynamics: Systems Thinking and Modeling for a Complex World*. Boston, MA: Irwin/McGraw-Hill, 2000.
- [167] C. M. Macal and M. J. North, "Tutorial on agent-based modelling and simulation," *Journal of Simulation*, vol. 4, no. 3, pp. 151–162, 2010.
- [168] D. A. Wiegmann and S. A. Shappell, "A human error analysis of commercial aviation accidents using the human factors analysis and classification system (hfacs)," *Aviation, Space, and Environmental Medicine*, vol. 72, no. 11, pp. 1006–1016, 2003.
- [169] H. H. Clark, *Using Language*. Cambridge: Cambridge University Press, 1996.
- [170] H. P. Grice, "Logic and conversation," in *Syntax and Semantics, Vol. 3: Speech Acts*, P. Cole and J. L. Morgan, Eds. New York: Academic Press, 1975, pp. 41–58.

- [171] W. Ward, "Understanding spontaneous speech: The phoenix system," in *Proceedings of ICASSP '91. International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, 1991, pp. 365–367.
- [172] P. C. Constantinides and A. I. Rudnicky, "Dialog analysis in the carnegie mellon communicator," in *Proceedings of the Sixth European Conference on Speech Communication and Technology (EUROSPEECH'99)*, Budapest, Hungary, 1999, pp. 1639–1642.
- [173] D. Hovy and A. Søgaard, "Tagging performance correlates with author demographics," *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pp. 483–488, 2015.
- [174] A. Koenecke, A. Nam, E. Lake, J. Nudell, M. Quartey, Z. Mengesha, C. Touns, J. R. Rickford, D. Jurafsky, and S. Goel, "Racial disparities in automated speech recognition," *Proceedings of the National Academy of Sciences*, vol. 117, no. 14, pp. 7684–7689, 2020.
- [175] R. J. Lickley, "Disfluency," *The Handbook of Speech Production*, pp. 495–516, 2015.
- [176] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *Journal of Big Data*, vol. 6, no. 1, p. 27, 2019.
- [177] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.
- [178] T.-E. Lin, H.-J. Chen, and H.-H. Chen, "Deep unknown intent detection with adversarial text generation," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, 2019, pp. 5972–5981.
- [179] K. Yoshino and S. Nakamura, "Dialogue state tracking based on lstm," in *Proceedings of the 4th International Workshop on Spoken Dialogue Systems*, 2016.
- [180] V. Krakovna, L. Orseau, R. Kumar, M. Martic, and J. Leike, "Specification gaming: The flip side of ai ingenuity," *DeepMind Safety Research Blog*, 2020.
- [181] Speakeasy Solutions Inc., "Choosing the right microphone for your speech recognition solution," White Paper, 2020, available: link.
- [182] S. Möller, K.-P. Engelbrecht, C. Kühnel, I. Wechsung, and B. Weiss, "Analysis of communication failures for spoken dialogue systems," in *Proceedings of Interspeech 2007*, Antwerp, Belgium, 2007, pp. 141–144.
- [183] T. Fong, I. Nourbakhsh, and K. Dautenhahn, "A survey of socially interactive robots," *Robotics and Autonomous Systems*, vol. 42, no. 3-4, pp. 143–166, 2003.

- 
- [184] L. Rashid, K. Pattabiraman, and S. Gopalakrishnan, "Characterizing the impact of intermittent hardware faults on programs," *IEEE Transactions on Reliability*, vol. 64, no. 1, pp. 297–310, 2014.
- [185] H. S. Gunawi, R. O. Suminto, R. Sears, C. Gollhofer, S. Sundararaman, X. Lin, T. Emami, W. Sheng, N. Bidokhti, C. McCaffrey *et al.*, "Fail-slow at scale: Evidence of hardware performance faults in large production systems," *ACM Transactions on Storage (TOS)*, vol. 14, no. 3, pp. 1–26, 2018.
- [186] V. Pandelea, E. Ragusa, T. Young, P. Gastaldo, and E. Cambria, "Toward hardware-aware deep-learning-based dialogue systems," *Neural Computing and Applications*, vol. 34, no. 13, pp. 10397–10408, 2022.
- [187] A. A. Zúñiga, A. Baleia, J. Fernandes, and P. J. D. C. Branco, "Classical failure modes and effects analysis in the context of smart grid cyber-physical systems," *Energies*, vol. 13, no. 5, p. 1215, 2020.
- [188] D. Cotroneo, L. De Simone, P. Liguori, and R. Natella, "Fault injection analytics: A novel approach to discover failure modes in cloud-computing systems," *IEEE transactions on dependable and secure computing*, vol. 19, no. 3, pp. 1476–1491, 2020.
- [189] D. F. Sittig, A. Wright, E. Coiera, F. Magrabi, R. Ratwani, D. W. Bates, and H. Singh, "Current challenges in health information technology–related patient safety," *Health informatics journal*, vol. 26, no. 1, pp. 181–189, 2020.
- [190] I. A. Scott, A. Van Der Vegt, P. Lane, S. McPhail, and F. Magrabi, "Achieving large-scale clinician adoption of ai-enabled decision support," *BMJ Health & Care Informatics*, vol. 31, no. 1, p. e100971, 2024.
- [191] S. Provoost, H. M. Lau, J. Ruwaard, and H. Riper, "Embodied conversational agents in clinical psychology: A scoping review," *Journal of Medical Internet Research*, vol. 19, no. 5, p. e151, 2017.
- [192] E. Oye, "Accent and dialect adaptation in asr: Training custom vosk models to improve recognition for non-native english speakers," *SSRN Electronic Journal*, 2024.
- [193] S. Larson, A. Mahendran, J. J. Peper, C. Clarke, A. Lee, P. Hill, J. K. Kummerfeld, K. Leach, M. A. Laurenzano, L. Tang *et al.*, "An evaluation dataset for intent classification and out-of-scope prediction," *arXiv preprint arXiv:1909.02027*, 2019.
- [194] J. D. Williams and S. Young, "Partially observable markov decision processes for spoken dialog systems," *Computer Speech & Language*, vol. 21, no. 2, pp. 393–422, 2007.
- [195] D. Bohus and A. I. Rudnicky, "Error-handling in a spoken dialogue system that tolerates speech recognition errors," in *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*, Lisbon, Portugal, 2005, pp. 121–130.

- 
- [196] S. D’Mello, S. Craig, B. Gholson, S. Franklin, N. Person, and A. Graesser, “Integrating affect sensors in an intelligent tutoring system,” in *Proceedings of the 2008 conference on Affective Computing and Intelligent Interaction*, 2008, pp. 1–12.
- [197] N. Yalçın and S. DiPaola, “A computational model of empathy for interactive agents,” *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 11, no. 3-4, pp. 1–35, 2021.
- [198] X. Chen and X. Zhang, “Deep learning for instance-level recognition of rare diseases in medical imaging,” *Frontiers in Genetics*, vol. 12, p. 664323, 2021.
- [199] J. Yang, K. Zhou, Y. Li, and Z. Liu, “Generalized out-of-distribution detection: A survey,” *arXiv preprint arXiv:2110.11334*, 2021.
- [200] C. G. Northcutt, A. Athalye, and J. Mueller, “Pervasive label errors in test sets destabilize machine learning benchmarks,” *arXiv preprint arXiv:2103.14749*, 2021.
- [201] S. J. Mielke, C. Shain, T. Pimentel, Ç. Cöltekin, and R. Cotterell, “Between a rock and a hard place: Contrastive learning for data-efficient models,” in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Online and Punta Cana, Dominican Republic, 2021, pp. 10 712–10 725.
- [202] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, “On calibration of modern neural networks,” *Proceedings of the 34th International Conference on Machine Learning*, vol. 70, pp. 1321–1330, 2017.
- [203] M. T. Ribeiro, S. Singh, and C. Guestrin, ““why should i trust you?”: Explaining the predictions of any classifier,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1135–1144.
- [204] Analog Devices, “Acoustic echo cancellation,” Analog Devices Wiki, 2023, available: [link](#).
- [205] H. L. Semigran, J. A. Linder, C. Gidengil, and A. Mehrotra, “Evaluation of symptom checkers for self diagnosis and triage: audit study,” *BMJ*, vol. 351, p. h3480, 2016.
- [206] K. Goddard, A. Roudsari, and J. C. Wyatt, “Automation bias: a systematic review of frequency, effect mediators, and mitigators,” *Journal of the American Medical Informatics Association*, vol. 19, no. 1, pp. 121–127, 2012.
- [207] D. Lyell and E. Coiera, “Automation bias and verification complexity: a systematic review,” *Journal of the American Medical Informatics Association*, vol. 24, no. 2, pp. 423–431, 2017.
- [208] R. Parasuraman and M. Mouloua, “Automation and human performance: Theory and applications,” *Ergonomics*, vol. 40, no. 3, pp. 293–294, 1997.

- [209] I. L. Singh, R. Molloy, and R. Parasuraman, "Automation-induced "complacency": development of the complacency-potential rating scale," *The International Journal of Aviation Psychology*, vol. 3, no. 2, pp. 111–121, 1993.
- [210] F. Cabitza, R. Rasoini, and G. De Nardis, "Deskilling and 're-skilling' in the case of a clinical decision support system for radiological reporting," *Information Technology & People*, vol. 30, no. 3, pp. 567–591, 2017.
- [211] Z. Obermeyer, B. Powers, C. Vogeli, and S. Mullainathan, "Dissecting racial bias in an algorithm used to manage the health of populations," *Science*, vol. 366, no. 6464, pp. 447–453, 2019.
- [212] E. Sezgin, M. Weiler, A. Weiler, S. Lin, K., L. Militello, and Y. Huang, "A conversational agent-based platform for reporting patient safety events: a human-centered design approach," *Journal of the American Medical Informatics Association*, vol. 28, no. 10, pp. 2163–2173, 2021.
- [213] K. S. Yew, M. K. George, and H. B. Allred, "Acute abdominal pain in adults: evaluation and diagnosis," *American family physician*, vol. 107, no. 6, pp. 585–596, 2023.
- [214] H. Hollands, D. Johnson, A. C. Brox, D. Almeida, D. L. Simel, and S. Sharma, "Acute-onset floaters and flashes: is this patient at risk for retinal detachment?" *Jama*, vol. 302, no. 20, pp. 2243–2249, 2009.
- [215] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative research in psychology*, vol. 3, no. 2, pp. 77–101, 2006.
- [216] X. Li, Y.-N. Chen, L. Li, J. Gao, and A. Celikyilmaz, "Investigation of language understanding impact for reinforcement learning based dialogue systems," *arXiv preprint arXiv:1703.07055*, 2017.
- [217] M. W. Dusenberry, D. Tran, E. Choi, J. Kemp, J. Nixon, G. Jerfel, K. Heller, and A. M. Dai, "Analyzing the role of model uncertainty for electronic health records," in *Proceedings of the ACM Conference on Health, Inference, and Learning*, 2020, pp. 204–213.
- [218] D. Bohus and A. Rudnicky, "Sorry, i didn't catch that!-an investigation of non-understanding errors and recovery strategies," in *Proceedings of the 6th SIGdial workshop on discourse and dialogue*, 2005, pp. 128–143.
- [219] L. Gauerhof, P. Munk, and S. Burton, "Structuring validation targets of a machine learning function applied to automated driving," in *International Conference on Computer Safety, Reliability, and Security*. Springer, 2018, pp. 45–58.
- [220] V. T. Vasudevan, A. Sethy, and A. R. Ghias, "Towards better confidence estimation for neural models," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 7335–7339.

- [221] N. Munsch, A. Martin, S. Guarini, J. Nateqi, I. Abdarahmane, R. Weingartner-Ortner, and B. Knapp, “Diagnostic accuracy of web-based covid-19 symptom checkers: comparison study,” *Journal of medical Internet research*, vol. 22, no. 10, p. e21299, 2020.
- [222] L. Zwaan and W. E. Hautz, “Bridging the gap between uncertainty, confidence and diagnostic accuracy: calibration is key,” *BMJ Quality & Safety*, vol. 28, no. 8, pp. 607–610, 2019.
- [223] Y. Gal and Z. Ghahramani, “A theoretically grounded application of dropout in recurrent neural networks,” in *Advances in neural information processing systems*, vol. 29, 2016, pp. 1019–1027.
- [224] P. Mooney, “Medical speech, transcription, and intent,” 2021.
- [225] T. Nguyen and M. Shcherbakov, “Enhancing rasa nlu model for vietnamese chatbot,” *International Journal of Open Information Technologies*, vol. 9, no. 1, pp. 31–36, 2021.
- [226] H.-T. Tseng, C.-C. Hsieh, and Y.-W. Lin, “An intelligent disease query system based on rasa NLU,” in *2020 International Symposium on Computer, Consumer and Control (IS3C)*. IEEE, 2020, pp. 458–459.
- [227] E. Gong, S. Baptista, A. Russell, P. Scuffham, M. Riddell, J. Speight, D. Bird, E. Williams, M. Lotfaliany, and B. Oldenburg, “My diabetes coach, a mobile app-based interactive conversational agent to support type 2 diabetes self-management: Randomized effectiveness-implementation trial,” *Journal of medical Internet research*, vol. 22, no. 11, p. e20322, 2020.
- [228] I. M. Evers, H. W. de Valk, and G. H. Visser, “Risk of complications of pregnancy in women with type 1 diabetes: nationwide prospective study in the netherlands,” *Bmj*, vol. 328, no. 7445, p. 915, 2004.
- [229] M. Persson, M. Norman, and U. Hanson, “Obstetric and perinatal outcomes in type 1 diabetic pregnancies: a large, population-based study,” *Diabetes care*, vol. 32, no. 11, pp. 2005–2009, 2009.
- [230] T. A. Buchanan, A. H. Xiang, and K. A. Page, “Gestational diabetes mellitus: risks and management during and after pregnancy,” *Nature Reviews Endocrinology*, vol. 8, no. 11, p. 639, 2021.
- [231] M. C. Macintosh, K. M. Fleming, J. A. Bailey, P. Doyle, J. Modder, D. Acolet, S. Golightly, and A. Miller, “Perinatal mortality and congenital anomalies in babies of women with type 1 or type 2 diabetes in england, wales, and northern ireland: population based study,” *Bmj*, vol. 333, no. 7560, p. 177, 2006.
- [232] K. Gamson, S. Chia, and L. Jovanovic, “The safety and efficacy of insulin analogs in pregnancy,” *The Journal of Maternal-Fetal & Neonatal Medicine*, vol. 15, no. 1, pp. 26–34, 2004.
- [233] G.-Y. Park, D. H. Kim, and D. Y. Lee, “Software FMEA analysis for safety-related application software,” *Annals of Nuclear Energy*, vol. 70, pp. 96–102, 2014.

- 
- [234] T. Ishimatsu, N. G. Leveson, J. Thomas, M. Katahira, Y. Miyamoto, and H. Nakao, "Modeling and hazard analysis using stpa," 2010.
- [235] "Clinical safety case report - clinical decision support (cds) api implementation guide release 2.0 v2.0," NHS Digital, Tech. Rep., 2020, appendix A: Risk classification matrix and risk acceptability scale.
- [236] Q. Li and Y.-L. Chen, "Data flow diagram," in *Modeling and Analysis of Enterprise and Information Systems*. Springer, 2009, pp. 85–97.
- [237] I. M. Habli, "Model-based assurance of safety-critical product lines," Ph.D. dissertation, University of York, Department of Computer Science, 2009.
- [238] R. Hawkins, I. Habli, and T. Kelly, "The principles of software safety assurance," Department of Computer Science, The University of York, Tech. Rep., 2013.
- [239] Y. Jia, T. Lawton, S. White, and I. Habli, "Developing a safety case for electronic prescribing," in *MEDINFO 2019: Health and Wellbeing e-Networks for All*. IOS Press, 2019, pp. 629–633.
- [240] L. Gauerhof, R. D. Hawkins, C. Picardi, C. Paterson, Y. Hagiwara, and I. Habli, "Assuring the safety of machine learning for pedestrian detection at crossings," in *Computer Safety, Reliability, and Security: 39th International Conference, SAFECOMP 2020, Lisbon, Portugal, September 16-18, 2020, Proceedings*. Springer, 2020, pp. 197–212.
- [241] R. Ashmore, R. Calinescu, and C. Paterson, "Assuring the machine learning lifecycle: Desiderata, methods, and challenges," *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, pp. 1–39, 2021.
- [242] C. Picardi, R. Hawkins, C. Paterson, and I. Habli, "A pattern for arguing the assurance of machine learning in medical diagnosis systems," in *Computer Safety, Reliability, and Security. SAFECOMP 2019 Workshops*. Springer, 2019.
- [243] R. Hawkins, C. Paterson, C. Picardi, Y. Jia, R. Calinescu, and I. Habli, "Guidance on the assurance of machine learning in autonomous systems (amlas)," *arXiv preprint arXiv:2102.01564*, 2021.
- [244] O. Jaradat, I. Slijivo, R. D. Hawkins, and I. Habli, "Modular safety cases for the assurance of industry 4.0," in *Safety-Critical Systems Symposium*. York, 2020.