ABSTRACT

Title of Thesis: A GOAL, QUESTION, METRIC APPROACH TO COHERENT USE INTEGRATION WITHIN THE DEVOPS LIFECYCLE

Kelsey Anne Rassmann
Master of Science, 2022

Thesis Directed by: Professor William Regli
Department of Computer Science

The development of high-stakes artificial intelligence (AI) technology creates a possibility for disastrous errors of misuse and disuse. Despite these risks, AI still needs to be developed in a timely manner as it has the potential to positively impact users and their surrounding environment. High-stakes AI needs to "move fast" but it must not "break things." This thesis provides developers with a methodology that will allow them to establish human-AI coherency while maintaining the development speed of the DevOps software development lifecycle. First, I will present a model of the human-machine interaction (HMI) which will motivate a new mode of AI use entitled 'Coherent Use.' Then, I will describe a Goal, Question, Metric approach to maximizing Coherent Use which will integrate directly into the DevOps lifecycle. Finally, I will simulate the usage of this template through an existing software product.

A GOAL, QUESTION, METRIC APPROACH TO COHERENT USE INTEGRATION WITHIN THE DEVOPS LIFECYCLE

by

Kelsey Anne Rassmann

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2022

Advisory Committee:
Professor William Regli, Chair/Advisor
Dr. Craig Lawrence
Professor Adam Porter

© Copyright by
Kelsey Anne Rassmann
2022

Acknowledgments

First, I want to thank my advisor, Dr. William Regli, for providing me valuable guidance and giving me the opportunity to work with ARLIS. I am incredibly thankful for all that he has done for me. I would also like to thank the other members of my committee, Dr. Craig Lawrence, and Dr. Adam Porter for lending their time to serve on my committee.

Second, I would like to thank the mentors that I had the pleasure of working with at ARLIS: Dr. Jana Schwartz, Dr. Julie Marble, and Dr. Joshua Poore. Throughout this process they have all provided me with valuable advice. I am grateful for the countless lessons I have learned from them, all of which will follow me throughout my career.

In addition, I would like to acknowledge all the incredible members of the ARLIS community who have been extremely welcoming and were always willing to provide their expertise.

Finally, I would like to thank my parents, sister, brother, and lifelong friends, Susan, Colleen, and Ellie, without whom I would not be where I am today. My successes would not have been possible without their constant support.

Table of Contents

Acknowledgements ii

Table of Contents iii

List of Tables v

List of Figures vi

List of Abbreviations vii

Chapter 1: Introduction 1
- 1.1 High-Stakes AI Needs to "Move Fast..." . . . 1
- 1.2 ...But, It Mustn't "Break Things" . . . 2
- 1.3 Example: The Z-Virus Diagnosis Decision Aid . . . 4
- 1.4 Solution: Building Coherent Use within the DevOps Lifecycle . . . 5
  - 1.4.1 Contributions . . . 6

Chapter 2: Background 7
- 2.1 Trust in Human-AI Systems . . . 7
  - 2.1.1 Capabilities . . . 9
  - 2.1.2 Overall Goals & Considerations . . . 12
  - 2.1.3 Trust Can Be Developer-Controlled . . . 14
- 2.2 DevOps Overview . . . 15
  - 2.2.1 The DevOps Lifecycle . . . 15
- 2.3 The Goal, Question, Metric Approach . . . 17
- 2.4 Summary of Related Works . . . 19

Chapter 3: Understanding Coherent Use 20
- 3.1 Modeling the Human-Machine Interaction . . . 20
  - 3.1.1 Five Spanning Use Cases . . . 22
  - 3.1.2 The HMI Model . . . 27
- 3.2 Defining Coherent Use . . . 36

Chapter 4: A Goal, Question, Metric Template for Coherent Use 39
- 4.1 Defining the GQM Template . . . 39
  - 4.1.1 The Overarching Goal . . . 40
  - 4.1.2 Template Questions . . . 40

4.1.3 Example Metrics . . . 41
4.2 Template Integration into the DevOps Lifecycle . . . 43
4.2.1 Active Development . . . 45
4.2.2 Pre-Deployment Testing . . . 45
4.2.3 Post-Deployment Testing . . . 46
4.3 Example: Evaluating the ZDDA . . . 47
4.3.1 Phase 1.1: Active Development . . . 47
4.3.2 Phase 2.1: Pre-Deployment Testing . . . 48
4.3.3 Phase 3.1: Post-Deployment Testing . . . 49
4.3.4 Phase 1.2: Active Development, Again . . . 50
4.3.5 Discussion . . . 50

Chapter 5: The Surveillance Use Case 53
5.1 The Security Surveillance AI-enabled Technology . . . 54
5.2 Analysis via the GQM Template . . . 55
5.2.1 Active Development . . . 55
5.2.2 Pre-Deployment Testing . . . 58
5.2.3 Post-Deployment Testing . . . 64
5.2.4 Active Development, Again . . . 68
5.3 Discussion . . . 69
5.3.1 The Surveillance AI Plugin . . . 69
5.3.2 The GQM Template . . . 71

Chapter 6: Conclusions 73
6.1 Limitations and Future Work . . . 73
6.2 Major Takeaways . . . 75

Appendix A: Non-Developer-Controlled Trust Factors 77
A.1 External Trust Factors . . . 78
A.1.1 Environmental Characteristics . . . 78
A.1.2 User Characteristics . . . 79
A.1.3 Task Characteristics . . . 82
A.2 Emerging Factors . . . 83
A.2.1 User Perceptions . . . 83
A.2.2 Reliance and Its Relationship with Trust . . . 84

Bibliography 87

List of Tables

3.1 Use case classification methodology. 21
3.2 Summary of use case classifications. 22
3.3 HMI model function descriptions. 29
3.4 HMI model variable descriptions. 29
3.5 Human-Machine Interaction failure modes. 38

4.1 The GQM template for Coherent Use integration into the DevOps lifecycle. 44

List of Figures

2.1 Example GQM breakdown based on the ZDDA. 18

3.1 Workflow model of the Human-Machine Interaction. 28

4.1 Comparison of development methodologies for the ZDDA. 52

A.1 Factors influencing trust and reliance in AI-enabled software. 77

List of Abbreviations

AI	Artificial Intelligence
ARLIS	Applied Research Laboratory for Intelligence and Security
Avg. FDA	Average Frame Detection Accuracy
DoD	Department of Defense
FBI	Federal Bureau of Investigation
FN	False Negative
FP	False Positive
GQM	Goal, Question, Metric
HMI	Human-Machine Interaction
IoU	Intersection Over Union
MPRD	Mission Product Requirement Document
TAMS	Tactical Autonomous Mobility System
TP	True Positive
T&E	Test and Evaluation
ZDDA	Z-Virus Diagnosis Decision Aid

Chapter 1: Introduction

The creation of high-stakes artificial intelligence (AI) has the potential to greatly impact users and their surrounding environment. For the purposes of this thesis, high-stakes AI is considered AI that has the potential to cause significant harm to the life or liberty of a user or those impacted by the AI's use. When creating this type of technology, developers can be self- and team-motivated to achieve immediate project goals while ignoring the operational goals of the human-AI system. What is the point of creating an AI with unmatched performance if no one uses it? How can developers make sure that operators know when to trust the AI and when to second guess its output? Overlooking these considerations can lead to cases of misuse and disuse of the AI, which, when dealing with high-stakes scenarios, may lead to dangerous consequences. Operators must understand when they should and should not trust the AI tools provided to them so that they may benefit from the AI while simultaneously using the tool safely.

1.1 High-Stakes AI Needs to "Move Fast..."

There is an ever-increasing need to rapidly develop high-stakes AI technology as it has a potential to positively impact users and their surrounding environment. This is apparent within the context of both medicine and the Department of Defense (DoD).

The medical domain has already begun to use AI technology as an aid to doctors [1]. This technology not only has the potential to help doctors in performing medical tasks (e.g., the reading of medical scans), but also to promote the patient-doctor relationship and possibly reduce the cost of medical care [2]. As demonstrated during the on-going COVID-19 pandemic, the use of innovative AI technology helped with the quick creation of vaccines that offer protection against the virus [3, 4]. Although medical AI is not without faults [5], the possible societal benefit makes it easy to want to create these systems as quickly as possible.

Providing an even greater sense of urgency is the DoD. To remain competitive with their adversaries, the DoD needs act now to start effectively implementing AI [6]. However, within the DoD, there is an awareness that the traditional developmental test and evaluation approach is incompatible with the dynamic nature of AI technology. The realization that AI software requires a more flexible testing approach led to the suggested use of software engineering practices like DevOps (or DevSecOps) [7], which also aids in the DoD's ability to quickly deploy these technologies [8].

1.2 ...But, It Mustn't "Break Things"

Despite the growing need for AI in a variety of contexts, the "move fast and break things" mentality does not work with the high-stakes nature of some AI technologies. This popular Silicon Valley practice is observed through the sheer number of software updates and patches that are deployed by companies like Apple, Microsoft, and Google. While this general structure can be great when dealing with low-stakes software (e.g.,

targeted advertising, video games, etc.), it cannot be extended to high-stakes applications where life and liberty are at stake.

If a piece of high-stakes AI results in critical errors of misuse or disuse, it can produce devastating consequences which have already been observed within the context of self-driving cars and facial recognition software. An over-reliance on Tesla's Autopilot software has already led to the death of consumers who had a false view of the car's ability to drive itself [9]. However, despite these accidents, Tesla continues to deploy beta software to consumers with a vague warning to "still be careful, but it's getting mature" [10, 11]. Critics, including those who previously worked on Autopilot, have shown concern for the practice of over-the-air updates, explaining that "it can be hazardous because buyers are never quite sure what the system can and cannot do" [12]. Similar concerns for the advancement of high-stakes AI can be observed through the use of facial recognition technology by police. Some police departments are using facial recognition software in an attempt to arrest suspects of various crimes [13]. While this technology may not have life or death consequences, it can certainly result in the loss of liberty. Software defects and over-reliance have led to false arrests and, in some cases, individuals serving time for a crime that they never committed [14, 15]. These issues have led companies, like Amazon, to prohibit the police use of their facial recognition software [16].

High-stakes AI software has the ability to result in major losses of life and liberty. While these two examples merely scratch the surface of the devastation that can be caused by this type of AI software, they emphasize the importance of developing high-stakes AI with the care and attention it deserves.

1.3 Example: The Z-Virus Diagnosis Decision Aid

To illustrate these problems further, consider the following scenario, inspired by movies like [17], with current high-stakes AI development practices:

Imagine the world is experiencing a zombie apocalypse cause by a new virus (Z-Virus). Doctors recently discovered that Z-Virus has an incubation period of 5 days. During this incubation period, doctors can successfully cure a Z-Virus infection with a powerful new drug. However, Z-Virus is incredibly difficult for doctors to diagnose and the drug created to cure Z-Virus is not only expensive, but has dangerous side effects for those that take it without a Z-Virus infection. Thus, doctors need to ensure that they can detect early signs of a Z-Virus infection accurately. This will minimize the amount of individuals who are zombified but also ensure the drug is not administered to an individual who has not been infected. To help doctors with this task, a team of developers have come together to create an AI tool that will aid in the diagnosis of a Z-Virus infection, the Z-Virus Diagnosis Decision Aid (ZDDA). The team utilizes a linear, waterfall software engineering methodology, described in [18], which can be common when creating high-stakes software.

As a first step, the developers identify requirements for the ZDDA. Then, they use a dataset to train a deep learning model that will produce a positive or negative diagnosis decision upon the input of patient data by a doctor. This output is presented to the doctor in a black-box fashion, only displaying the positive or negative diagnosis outcome. The development of the ZDDA is followed by rigorous model accuracy testing until the AI has achieved an F_1 score of 0.95. With immense excitement, the developers deploy the

ZDDA, however, to their surprise, the incidences of zombified humans only decreases slightly. What happened?

The developers discovered that a few things were going wrong post-deployment with the ZDDA. First, some doctors did not trust the ZDDA to produce an accurate diagnosis and chose to diagnose patients manually. This led to undetected cases of Z-Virus infections as well as non-infected individuals receiving unneeded treatment. In other failure cases, doctors over-relied on the ZDDA, following its output blindly and resulting in similar problems. Overall, the developers found that the ratio of failures was 5:1 (for every case of correct use of the AI, there were 5 cases of misuse or disuse). Further, the developers measured that the ZDDA was producing an F_1 score of 0.75 post-deployment as evolutionary changes in the virus caused bias in their training dataset. These problems caused the ZDDA to only have a slight impact on the ongoing apocalypse.

1.4 Solution: Building Coherent Use within the DevOps Lifecycle

While the previous example is certainly hypothetical, it highlights the need for high-stakes AI to communicate coherently with the user. This coherency should not be a burden that is placed on the user; rather, it is the developers responsibility to ensure that they are creating a system that allows for safe and effective interactions. As this thesis will show in the following chapters, there are influences of appropriate reliance and trust that developers begin to decide upon as early as design time. However, we cannot expect developers to make the right decisions if we do not provide them the tools to do so while maintaining their development speed.

1.4.1 Contributions

Given the established need for human-AI coherency, the three main contributions of this thesis are as follows:

- The development of a model of the human-machine interaction (HMI) based off of five spanning use cases;
- The establishment of 'Coherent Use' as a necessary consideration for high-stakes AI-enabled software; and,
- A Goal, Question, Metric (GQM) test and evaluation approach for the maximization of Coherent Use that integrates directly into the DevOps lifecycle.

Chapter 2: Background

To understand the context surrounding the proposed Goal, Question, Metric (GQM) test and evaluation framework, it is necessary to establish the background knowledge that led to its creation. This includes an overview of the developer-controlled factors that influence human-AI trust, the concept of DevOps (or DevSecOps), and the GQM approach.

2.1 Trust in Human-AI Systems

End user trust is of particular importance to AI technologies. These technologies can often times be data-driven and thus non-deterministic. This can potentially lead to unexpected or inaccurate results and therefore an incorrect level of trust placed on the technology. However, despite this non-determinism, AI still have the potential to positively impact users and their surrounding environment. When looking to integrate these AI technologies in high-stakes scenarios, where life and liberty may be at risk, trust can be of particular importance for AI adoption [19, 20]. This motivates a focus on end user trust in these types of technologies.

As a first step in understanding the role of trust in human-AI systems, a brief literature review of the factors that influence trust was performed. After reviewing the

relevant AI and automation literature, it was discovered that the different factors that influence trust can be organized into three different categories. First, there are various influences within the technology itself. These internal, developer-controlled factors are viewed as built-in features and characteristics of the technology. Second, there are factors that exist externally from the technology, including environmental factors, user influences, and task characteristics. Finally, in the last category are factors that emerge from the HMI (i.e., perceptions of the technology and reliance). In the following subsections, I will discuss the developer-controlled trust factors that were found within this search. Specifically, these factors are broken down into sub-categories of Capabilities and Overall Goals & Considerations. For more information regarding the external trust factors and emerging factors, refer to Appendix A.

Note that in the next section sources describing both AI and automation are used. While AI and automation differ slightly, I argue that when it comes to trust and reliance, they interact in similar ways. This can be understood by [21]'s breakdown of trust in technology as a combination of human characteristics, environment characteristics, and technology characteristics. The authors explain that regardless of what the technology is, human characteristics and environment characteristics will be approximately the same. This leaves only technology characteristics that really account for the difference between automation and AI, which the authors describe as the performance, process, and purpose of the technology. Thus, for each developer-controlled factor described, automation papers are only used when the technology factor is relevant to both AI and automation and it is believed that the information can be reasonably generalized for both technologies.

2.1.1 Capabilities

The capabilities sub-category refers to the requirements of the machine. In other words, what the AI is intended to do and what features it includes. When designing an AI, the following capabilities can play a role in the development of trust in that technology:

- **Inclusion of a Training Interface.** When designing any piece of technology, one careful consideration is the inclusion of a user training interface that will provide the user with a general introduction to the technology. If done in the right way, a training interface can be a useful resource for fostering trust in a machine. This is discussed by [22] when describing how to achieve trust with robotic teammates. Specifically, they mention a variety of training guidelines that can facilitate trust within these human-robot teams (e.g., informing users on how the robot should be interacted with and how it should be used). Training is also mentioned within [23] where again the authors provide guidance on how to properly train users with the intention of gaining trust. In this case, the authors provide tips on addressing other trust factors, like gossip and culture (both of which are described in Appendix A).

- **Display of Model Confidence.** The decision to display a model confidence level can influence the trust that a user places in an AI. To understand what is meant by model confidence, consider the following example. Suppose a neural network has the ability to classify an image between a dog and a cat. Along with the output classification (i.e “dog” or “cat”), the neural network may also output a percentage showing how much confidence it has in its answer. This percentage is the model

confidence. [24] addresses how model confidence can influence trust in a user study. They measured the importance of different factors of trust as reported by 362 participants and found that model confidence has a significant impact when an AI is the decider of a system. However, even though they found that this impact was less profound when a human is the decider, participants still reported that they felt having a display of model confidence is necessary. The importance of displaying model confidence is also shown in a study done by [25]. They found that the inclusion of this feature had a positive effect on the user's trust in the technology within a mobile phone domain.

- **Inclusion of Social Cues.** Social cues within an AI can be thought of as pro-social behaviors within the technology that attempt to account for the social nature of humans. [26] references this concept as an "immediacy behavior" and describe it as follows: "socially-oriented gestures intended to increase interpersonal closeness, such as proactivity, active listening, and responsiveness." Within their work, they describe the positive trust outcomes of these behaviors as it relates to robotic AI. [23] also addresses social cues by giving the example of synthetic speech. However, the authors provide a warning, explaining that this method of facilitating trust has the potential to build too much trust in the technology. This trust can then lead to high expectations of performance and potential misuse based on uncalibrated trust.

- **Explicit Bias Detection.** Bias Detection is a tool that can be used with data-driven AI to rid the data of bias. The removal of bias can help to ensure fairness and facilitate a correctly functioning AI. [27] discusses bias detection in terms of their

Chain of Trust concept. They explain how pre-processing data in an attempt to mitigate bias can help to foster trust in an AI. Bias is also discussed by [28]. The authors discuss not only detecting bias in training data, but also finding bias that is introduced to the technology.

- **Explainability.** Explainability is a property of AI software, with DARPA defining explainable AI as "AI systems that can explain their rationale to a human user, characterize their strengths and weaknesses, and convey an understanding of how they will behave in the future" [29]. Much of the literature surrounding trust in AI focuses on the need for explainability and transparency as a method for fostering trust in the technology. [7] discusses how a lack of explainability can undermine trust in an AI and also regards explainability as a standard and metric for trustworthiness. In addition, [30] describes how providing users with explanations and transparency can help to calibrate trust. Beyond these two examples, other papers discuss providing clear, consistent information and explainability including [22–24, 27, 28, 31, 32]. While some of these papers describe automation and others refer to AI, it is clear that providing the user with information regarding how a decision was reached is increasingly important for allowing users to trust a piece of technology appropriately.

- **Transparency of Failure.** The handling and transparency of machine failures refers to how the technology informs the user and deals with failures that occur. Within their Heuristics for Trusted Autonomy, [32] describes the transparency of failures as a heuristic that can inform the development of a trusted system. Their

heuristic covers not only alerting to the user that a failure occurred, but also includes information on how to mitigate the error in the future. Another way to view the handling of failures is with respect to failures that negatively impact trust in a user. Again, [27] describes this within the context of their Chain of Trust. The authors explain that data breaches or the introduction of bias to a machine learning algorithm could negatively impact trust in the technology, however, the handling of that failure could also lead the organization to more trustworthy practices. Thus, it is again shown how the handling of failures, especially those that could negatively impact the trust a user puts in the machine, is an important factor that can influence trust in technology.

2.1.2 Overall Goals & Considerations

The overall goals and considerations sub-category refers to overarching technology considerations (e.g., security, documentation, etc.). When developing an AI, the following goals can influence the trust that a user places in the technology:

- **Safety & Security.** The safety and security of an AI refers to how safe the AI is to use in terms of data privacy and protection. As explained by [21], data security is extremely important with respect to fostering trust, as users will likely feel uncomfortable using risky technology. Especially when using AI technology like machine learning, that depends heavily on data, users want to be sure that their data is safe. Another paper developed by IBM researchers describes transparency in relation to safety and security as a great way to instill trust in the consumer

[20]. They describe this in reference to their idea of a Supplier's Declaration of Conformity that presents all of this information to the user in a document [20].

- **Training & Test Descriptions.** This particular trust factor is specifically related to data-driven AI technologies as this type of AI usually requires some way to train a model based off of pre-existing data. It has been shown by a study done in [24], that providing information on how an AI was trained and tested created a positive impact regarding reported ratings of trustworthiness. Providing this information improved the perceived trustworthiness of the technology.

- **Performance.** The performance of the technology in the context of this literature review refers to the past and current abilities of the AI. This includes how well the technology functions as well as the functionality the machine is capable of (or should be capable of). Performance is mentioned many times throughout the literature both directly and indirectly. Within [32], an indirect reference to this meaning of performance is represented by three of the author's seven Heuristics for Trusted Autonomy, namely, heuristics 2, 4, and 5 which represent "Visibility of probable system behavior", "Visibility of system capabilities & limitations", and "Awareness of latencies & delays" respectively. More directly, [21] discusses performance in relation to the continual development of trust in AI. Specifically, the authors describe characteristics such as usability and reliability, collaboration and communication, sociability and bonding, security and privacy, and interpretability. While I classify some of these items in other categories (e.g., security and privacy is extracted into its own factor as discussed previously), the authors still highlight

the importance of performance with factors such as usability and reliability, which they explain as the "competence of AI in completing tasks and finishing those tasks in a consistent and reliable manner" [21].

2.1.3 Trust Can Be Developer-Controlled

While this literature review was brief and likely does not include every influence of trust, it shows that although trust can often be viewed as a user-centered, humanistic quality, there are factors that influence trust that are directly controlled by developers. In fact, developers can start to consider these trust factors as early as design time to make a piece of AI-enabled technology that can be used appropriately. So, while it is important to acknowledge external and emerging factors that influence trust (see Appendix A), for developer's purposes, they have almost no control over these factors. Due to this lack of control, the following sections of this thesis focus specifically on things that developers can control. In addition, although much of the current literature places great emphasis on explainable AI (as described previously), this is not the only means with which trust can be controlled by the developer.

Note that later sections of this thesis will describe terms such as trustworthiness and trust calibration. These concepts, while related, are distinct from trust itself. However, the realization that trust can be controlled by a developer so early in the development lifecycle sparked the inspiration for the final GQM framework that will be described in Chapter 4.

2.2 DevOps Overview

DevOps is a software engineering process that seeks to merge the development and operations sides of a software product. This leads to intercommunication between developers and operators on cross-functional teams that seek to deliver continuously at quick speeds [33]. A major aspect of DevOps is the concept of continuous integration and continuous delivery. Continuous Integration refers to developers working on the same version of code at the same time, continually pushing changes into a code repository, and checking for errors in the code [34]. Continuous Delivery refers to a continuous deployment of code at a very fast pace [34].

DevSecOps is an extension of DevOps that specifically seeks to integrate security within the process [34]. The goal of DevSecOps is to start embedding security from the start of development and ultimately within the operations processes as well [35].

2.2.1 The DevOps Lifecycle

The DevOps Lifecycle illustrates how developer practices (plan, develop, verify, test) and operator practices (deploy, operate, and monitor) can be intertwined to create a cross-functional process that integrates operations into the development process. The process includes the following steps explained in [36]:

1. **Plan:** Plan system requirements and delivery timelines.

2. **Develop:** Develop the code for the software product, including the creation of unit tests.

3. **Verify:** Verify that the requirements of the software product are fulfilled by the code.

4. **Test:** Continuous testing performed by automated tests as well as possible release of user demo versions.

5. **Deploy:** Continuous deployment of code changes keeping in mind configuration and resources needed on the platform used for deployment.

6. **Operate:** Management of the software post-deployment.

7. **Monitor:** Monitoring of the software product including the collection of data that will help to inform the next planning cycle.

With the addition of security in the DevOps software process, this lifecycle was updated to include considerations at each stage of the cycle that inform important security practices. While DevOps does attempt to draw more focus on the end user through the collection of user data during the monitoring phase, this focus should be more intentional for high-stakes AI software. As shown in Chapter 1, errors of AI use can lead to dangerous errors of misuse and disuse. Given the assertion that trust factors can begin to be decided upon at design time, the integration of this HMI consideration can and should be baked into the DevOps lifecycle. This intentional focus on human-AI coherency will allow developers to have an iterative understanding on how coherency is changing from release to release and thus create an AI that can be used correctly by the user.

2.3 The Goal, Question, Metric Approach

The Goal, Question, Metric (GQM) software evaluation approach, described by [37], utilizes a top-down methodology for the selection of appropriate metrics that aid in the achievement of high-level goals. The determination of overarching goals leads to the creation of quantifiable questions which is followed by the selection of metrics that answer these questions. Given the amount of measures that can be taken against a piece of software, this paradigm seeks to ensure that all measures are characterized by a high-level goal and are taken with purpose.

When utilizing this approach, there is generally a process that is followed. Although processes varied in the literature, a pattern arose from [38–40] referencing 4-phases of GQM: planning, definition, data collection, and interpretation. Generally, planning consists of administrative tasks (e.g., the establishment of a GQM team, identification of areas in need of improvement, etc.), the definition phase defines the formal GQM breakdown, data collection involves the logistics of collecting data needed for the GQM assessment, and finally interpretation draws conclusions based on the collected data. While this short description merely scratches the surface of the 4-phase GQM process, a complete review of these phases is outside the scope of this work. However, an in depth view of each phase can be found within [41].

Shifting focus to the anatomy of a GQM breakdown, defined GQM goals consist of four main components as described in [37]: the *purpose* of the measurement, the *object* being measured, the *issue* attempting to be remedied, and the *viewpoint* from which the measurement is being taken. Note that other works, [38, 42, 43], also discuss

a fifth component of a GQM goal describing the *context/environment* characteristics that surround the goal (e.g., the organization, resources, etc.). However, this component is omitted in the final template as there is an assumption that the context surrounding the AI is relatively constant and involves a development team seeking to maximize human-AI coherency. Once the goals are defined, questions are created based off of a model. Finally, metrics are defined that answer the questions and thus measure progress toward the overarching goals. Figure 2.1 provides an example GQM breakdown based on the ZDDA introduced in Chapter 1 and an example provided in [37].

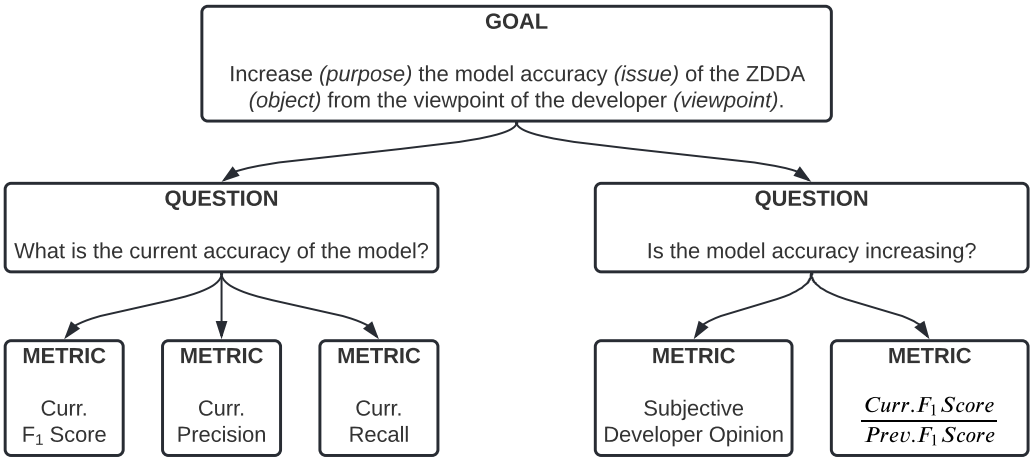


Figure 2.1: Example GQM breakdown based on the ZDDA.

After considering a variety of different test and evaluation (T&E) methodologies, GQM was chosen because of the belief that this approach is the most logical and easy to understand. This reduces the barrier to entry for software developers who may have never used the GQM approach. The paradigm itself also ensures that each measure is taken with a specific purpose which allows the focus of the measurement to remain on the maximization of human-AI coherency.

2.4 Summary of Related Works

When analyzing the current state of how trust is considered and measured, it is clear that there are a variety of different scales that aid in the measurement of trust, for instance [44, 45]. However, while these scales are certainly helpful for measuring trust, they do not necessarily provide guidance on how trust can be considered during development of different technologies. Other references, (e.g., [23, 32]) shift their focus to development and offer heuristics or advice to developers. Again, while this is certainly helpful for developers, it is also necessary for developers to have a methodology for establishing and maintaining human-AI coherency within a pre-existing software engineering process and as their product matures. This motivates the integration of trust-related considerations into the DevOps lifecycle which will not only utilize a well-known software engineering methodology, but will ensure the consideration of human-AI trust from the very start of AI development and throughout the lifetime of the software product. In an effort to make this transition as simple as possible, the use of GQM as a means to realize this integration was chosen. In doing this, developers will be able to easily understand this test and evaluation methodology and this should provide a low barrier to entry for developers who may have never used this approach before.

Chapter 3: Understanding Coherent Use

In this chapter I will present two of the three major contributions of this thesis. First I will describe a workflow model of the HMI as well as the process that was taken to derive the model. Informed by this model, I will then describe the second contribution of this thesis, an additional state of machine use called *Coherent Use*.

3.1 Modeling the Human-Machine Interaction

Building up to the concept of *Coherent Use*, it is first necessary to understand the motivation behind this new mode of AI use. This motivation can be illustrated by a model of the HMI. To ensure that the workflow model works with a variety of AI-enabled technologies, five different technologies were used to inform its development: the Z-Virus Diagnosis Decision Aid, Tesla's Autopilot, the Tactical Autonomous Mobility System, a Roomba, and a Smart Doorbell. These five AIs span four different categories: *stakes*, *machine perception*, *complexity*, and *autonomy*, all of which were chosen through the observation of the main differences between the technologies. For the purposes of this thesis, each of these categories are described as follows:

- **Stakes**: The risk of harm to the user or those surrounding the AI's use in the event of either a malfunction or error in operator use.

- **Machine Perception**: Level of human-machine teaming needed for the interaction specifically focusing on the user's general perception of the AI as a teammate or tool.

- **Complexity**: The technical complexity of the system focusing on the hardware and software components along with the ability for the machine to perform a variety of tasks.

- **Autonomy**: The level of machine autonomy specifically focusing on the degree of human intervention required when using the AI.

In order to ensure that each of the five use cases spanned the space of these four categories, each use case was classified as either high, medium, or low for each category. Table 3.1 describes the classification distinctions for each category.

	Stakes	Machine Perception	Complexity	Autonomy
High	Errors can lead to significant harm to the user and others. In some cases, these errors may be fatal.	The user generally views the machine as a true teammate that understands team goals and priorities.	The machine can perform many tasks and is extremely complex in terms of software and hardware. Machines of this caliber may not exist yet.	The machine can act almost entirely on its own accord without human intervention.
Medium	The machine has a potential to harm the user, however, this harm is likely not fatal.	The machine interacts actively with the user to complete a task; however, the user does not generally view the system as their teammate.	The machine consists of various components put together to achieve a goal. This machine may have the ability to perform multiple tasks.	The machine performs some tasks and makes some decisions on its own but still requires human intervention occasionally.
Low	There is no risk of injury or severe harm. Errors result in a minor inconvenience.	The machine is generally viewed as a tool by the user.	The machine is simple in terms of software and hardware. The machine itself provides only one function to the user.	The user must start the machine and tell it what to do often.

Table 3.1: Use case classification methodology.

3.1.1 Five Spanning Use Cases

Using the previously described classification method, each of the five use cases can be described in terms of these categories. A summary of all the classifications for each of the following use cases is shown in Table 3.2.

	ZDDA	Autopilot	TAMS	Roomba	Smart Doorbell
Stakes	High	High	High	Low	Medium
Machine Perception	Low	Medium	High	Low	Low
Complexity	Low	Medium	High	Medium	Medium
Autonomy	Low	Medium	High	Medium	Medium

Table 3.2: Summary of use case classifications.

3.1.1.1 The ZDDA

The first use case is the Z-Virus Diagnosis Decision Aid (ZDDA). Recall from Chapter 1 that this device is a very simple decision aid that helps doctors to diagnose the fictional Z-Virus infection.

This AI is certainly high-stakes in that it is being used to prevent the deaths of those infected by the virus. Any malfunction or human error has the potential to harm others, and may be fatal. The device is also considered to be low in the machine perception category as it is generally viewed by doctors as a tool that they use to come up with a final diagnosis decision. In terms of complexity, this AI is extremely simple. The only function that the machine provides to the doctor is the final diagnosis outcome. Finally, the ZDDA is considered to be low in the autonomy category. This is because the doctor

must start the machine in order for it to perform the task and would need to continually tell the device what to do if the doctor wanted to get a new diagnosis upon the discovery of new information.

3.1.1.2 Tesla's Autopilot

The second use case considered was Tesla's Autopilot technology [46]. Autopilot is an AI-enabled device within Tesla cars that assists drivers in certain aspects of driving. This includes steering, accelerating, and braking, as well as navigation features that will suggest possible lane changes and summoning technology that will drive the car to the user in a parking lot setting. Tesla does this through the use of cameras and sensors that gather data which is then processed through neural nets [46].

Autopilot is considered to be high-stakes in this classification methodology. In the event that Autopilot malfunctions or a user does not use the system properly (misuse or disuse) there is a risk of severe injury or even death. As mentioned in Chapter 1, this has happened before [9]. In terms of machine perception, Autopilot is considered medium. The user actively interacts with the machine to complete the driving task (e.g., giving the car permission to go through a traffic light through human intervention [47]), however, the car is still generally viewed as a car and not a teammate. Autopilot is also considered of medium complexity. This is because the car has the ability to perform multiple tasks (e.g., navigation, summoning, etc.) and consists of many different hardware components with highly complex software. However, even though Autopilot is certainly advanced, it is not considered high in complexity because this category is reserved for machines that

are so complex that they may not exist yet. Finally, Autopilot is considered medium in terms of autonomy. When using Autopilot the car can make a variety of driving decisions on its own, however, there are cases when human intervention is required. In fact, Tesla states this themselves explaining that "Autopilot features require active driver supervision and do not make the vehicle autonomous" [46].

3.1.1.3 The Tactical Autonomous Mobility System

The third use case is the Tactical Autonomous Mobility System (TAMS) from the science fiction novel *Burn-In* [48]. The book follows Federal Bureau of Investigation (FBI) Special Agent Lara Keegan who is told to "burn-in" the TAMS for potential future use by the FBI. The TAMS itself can be viewed as the quintessential robotic teammate who learns from its surroundings, performs data collection and calculation, and acts autonomously. The AI robot can also take orders from Keegan and often converses with her as if the TAMS was human itself.

When classifying the TAMS, it is evident that the robot is considered high in all four categories. The robot is high-stakes since it is performing dangerous investigations and, although the robot is only given non-lethal weapons, Agent Keegan and those around the TAMS can still be severely harmed (and even killed) in the event of an error. The TAMS is considered high in the machine perception category as the robot is a true robotic partner to Agent Keegan when performing investigations. The AI is also extremely complex and although *Burn-In* bases technologies on the real world, something as advanced as the TAMS certainly does not yet exist. Finally, the TAMS is almost entirely autonomous.

Once turned on, the robot can make its own decisions based on its surroundings, even changing the radio station to a particular song based on the emotions it senses from Agent Keegan.

3.1.1.4 Roomba

The fourth use case is the Roomba robotic vacuum cleaner by iRobot [49]. The Roomba can map out rooms and uses computer vision software to identify pieces of furniture. The robot can also provide cleaning suggestions to users on specific areas to avoid or clean [50].

The Roomba is considered low-stakes. In the event that there is either a malfunction or error of use, it will only result in an inconvenience to the user and has very low probability of causing any sort of physical harm. Although the Roomba does provide the user with suggestions, the robot is considered low in the machine perception category. This is because the Roomba is generally seen as a tool by the user and does not require constant interaction in order to complete the cleaning task. In addition, the Roomba is considered to be of medium complexity. This is because the Roomba consists of fairly complex hardware that allows the vacuum to not only "see" its surroundings but also physically clean the floor. With this hardware design, the Roomba also has the underlying advanced vision software and the ability to perform multiple different tasks (i.e., cleaning, room and furniture mapping, user suggestions, etc.). Finally, the Roomba is considered to be of medium autonomy. The robot itself has the ability to make cleaning decisions on its own and rarely needs human intervention.

3.1.1.5 Smart Doorbell with Computer Vision

The fifth and final use case is a fictional Smart Doorbell (inspired by products like Amazon's *Ring* [51]) fitted with computer vision software that enhances its capabilities over a normal doorbell. This doorbell performs a home monitoring task and will alert users when different events occur. For instance, if a salesperson holding a box approaches the user's home, the device may alert the user with the following message: "salesperson approaching with 15 inch cardboard box." In other cases, the doorbell will alert the user of different threats that may be outside of their home, including possible burglars. Along with this alert would be a description of what the doorbell "sees" outside of the home.

The Smart Doorbell is considered to be medium-stakes. While there is a risk of harm to the user if an error occurs while there is threatening activity outside of the user's home, this harm is likely not fatal. Since the Smart Doorbell performs a monitoring task, there is very little interaction with the user and the device itself is certainly viewed as a security tool, resulting in a low machine perception score. In terms of complexity, the Smart Doorbell is considered of medium complexity because it has various components (i.e., a camera, computer vision software, doorbell functionality, etc.) and it can perform multiple tasks (i.e., doorbell, camera to the front of the home, alerting system, etc.). Finally, the Smart Doorbell is of medium autonomy. The doorbell itself performs the monitoring task at all times on its own but may require human intervention to occasionally dismiss an alert or if the user wishes to access the camera.

3.1.2 The HMI Model

Using the above five use cases as a basis, a model of the HMI can be created. The process of creating this model started with the creation of five separate models for each of the five use cases. Since these five technologies spanned the four different categories described, they each provided a unique perspective to each model. These perspectives helped to create one general workflow model that works for all five use cases and any other HMI that fits in the domain of the classification methodology. This model (shown in Figure 3.1) is based on an idea from [52] that *Value = Benefit/Cost*. In general, I argue that if the user perceives that the benefits outweigh the costs of using a machine, then the user is likely to use the machine. Otherwise, the user will not use the machine. This basic principle lays the foundation of the model.

3.1.2.1 HMI Model Components

The model itself contains five main components: objects, functions, variables, decisions, and error states. The objects in the model are the environment, user, and machine. Note that the model shows the user and machine as objects that exist within the environment. A description of each of these objects is as follows:

- **Environment**: The current state of the world including the user, machine, and all influences surrounding the interaction.
- **User**: The human currently using the AI-enabled technology.
- **Machine**: The AI-enabled technology being used.

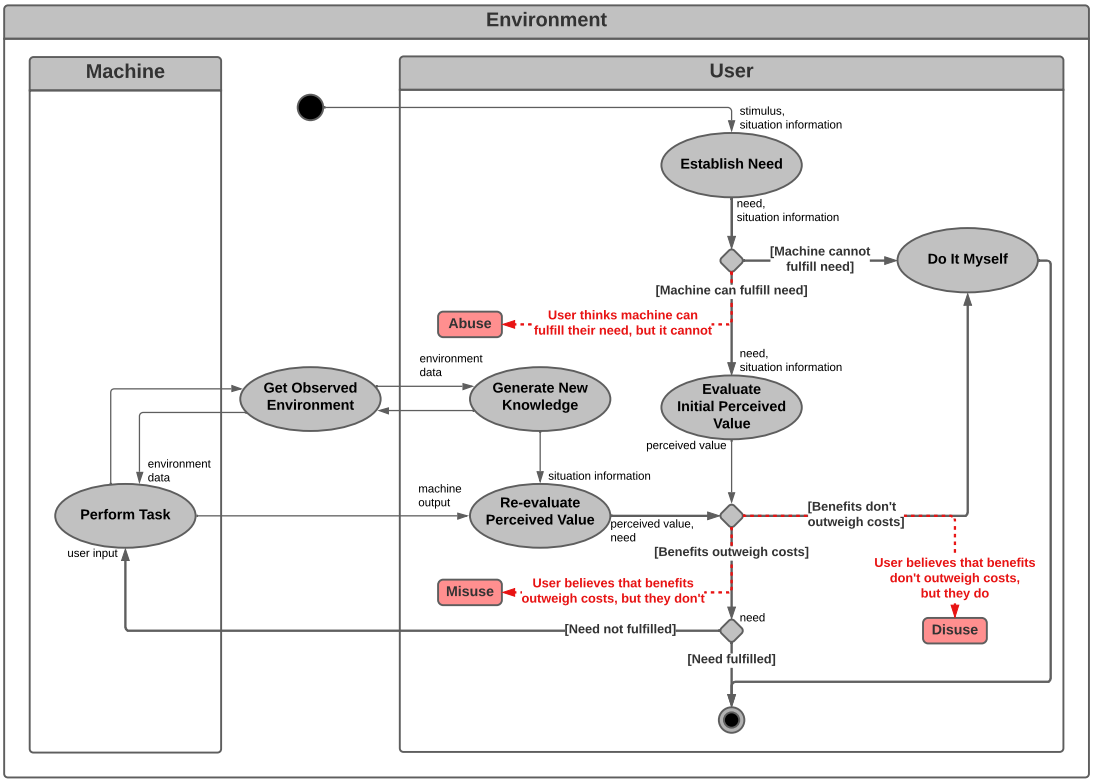


Figure 3.1: Workflow model of the Human-Machine Interaction.

Functions are represented by ovals in the model and are actions that are performed by the user, machine, or environment. A description of each of these functions along with their inputs and outputs can be found in Table 3.3. Along with functions, the model uses variables (shown as plain-text in Figure 3.1) to describe the inputs and outputs of each function. An overview of these variables is shown in Table 3.4. Within the workflow diagram, the user must also make decisions. These decisions are represented by diamonds in Figure 3.1 and show different paths the user may take given their understanding of the machine. Rather than describe these decisions in a table, with little situational context, these decisions will be described in terms of two examples described in the following sub-section. Finally, the HMI model contains three red rectangles that show three error

states a user could produce based on their decisions. These error states represent abuse, misuse, and disuse and will be described further in section 3.1.2.4.

Object	Function	Input	Output	Description
User	Establish Need	Stimulus, Situation Information	Need, Situation Information	Given an environmental stimulus and surrounding situation information, the user establishes a need they must fulfill. Situation information is also output to be used in a later function.
	Evaluate Initial Perceived Value	Need, Situation Information	Perceived Value	Given what the user knows initially about the situation and the need they have established, the user generates their initial perception of the value that the machine could provide them.
	Do It Myself	None	None	This function represents the user doing whatever is necessary to fulfill the need themselves (without the help of the machine).
	Generate New Knowledge	Environment Data	Situation Information	Performed by the user while they wait for the output of the machine. Represents the user updating their awareness of the current situation surrounding them.
	Re-evaluate Perceived Value	Machine Output, Situation Information	Perceived Value, Need	Represents the user's re-evaluation of their perception of machine value based on new information they have acquired while waiting for the machine output and the machine output itself. This function also updates the user's need, as it may have changed or may now be fulfilled.
Machine	Perform Task	User Input, Environment Data	Machine Output	Represents the machine performing its intended function.
Environment	Get Observed Environment	None	Environment Data	Returns information about the environment which is later used by the machine and user to inform their actions.

Table 3.3: HMI model function descriptions.

Variable	Description
Stimulus	An environmental factor that brings about a need in the user.
Situation Information	The user's perception of all the information regarding the current state of the environment that surrounds the user and machine at the time of the interaction.
Need	Provides the user with some sort of problem that needs to be solved or an issue that must be remedied. This need motivates the user to consider using the machine to fulfill the need. A need is internal to the user.
Perceived Value	The output of the user's cost/benefit analysis. The user's perception of the machine's worth to them.
User Input	The input that the user puts into the machine for it to perform the task. This may look different for different machines and may even consist of no specific action by the user.
Machine Output	The result of the machine performing its task. This can take the form of a diagnosis decision, unit of automated driving, etc.
Environment Data	Data from the environment describing the current state surrounding the interaction.

Table 3.4: HMI model variable descriptions.

3.1.2.2 Example: The ZDDA

Continuing to use the ZDDA as an explanatory example, the HMI model in Figure 3.1 works as follows:

1. **Start.** A patient enters the doctor's office with a variety of symptoms. This patient represents the *stimulus*. The doctor's perception of the current zombie apocalypse, existence of the ZDDA, and coworker gossip surrounding the ZDDA represent elements of the *situation information*.

2. **Establish Need.** The doctor uses the *stimulus* and *situation information* to establish their *need* to help the patient feel better. This is done utilizing the *Establish Need* function, which also passes along the *situation information* for use in Decision #1.

3. **Decision #1.** Given the *need* and *situation information* the doctor determines if the ZDDA will help make the patient feel better. Let's say the doctor does, following the *Machine can fulfill need* path. In the event that the doctor did not, they would follow the *Machine cannot fulfill need* path and perform the *Do It Myself* function which then terminates the interaction.

4. **Evaluate Initial Perceived Value.** The doctor now utilizes the *Evaluate Initial Perceived Value* function to perform a cost-benefit analysis based on the *situation information* and *need*. This generates an initial *perceived value*.

5. **Decision #2.** Based on this *perceived value* the doctor decides if the benefits of using the ZDDA outweigh the costs. For this example, the doctor does and therefore follows the *Benefits outweigh costs* path toward Decision #3. If the doctor did not believe this, they would follow the *Benefits don't outweigh costs* path and perform the *Do It Myself* function which will then terminate the interaction.

6. **Decision #3.** The doctor now checks if the current *need* has been fulfilled. In this

case, the need has not yet been fulfilled so the doctor proceeds down the *Need not fulfilled* path and starts to use the ZDDA.

7. **Perform Task.** Utilizing the *user input* (i.e., patient information) from the doctor, the ZDDA now begins to create a diagnosis decision. To do this, the ZDDA must query the environment using the *Get Observed Environment* function. This function returns *environment data* so that the ZDDA can generate a diagnosis. For this example, *environment data* can be viewed as the training data used to create the model. The ZDDA then creates *machine output* in the form of a diagnosis decision.

8. **Generate New Knowledge.** While the ZDDA is creating *machine output*, the doctor continuously generates knowledge about the surrounding environment. This is done through the *Generate New Knowledge* function, which again queries the environment for *environment data* and interprets it as a new perceived *situation information*. In this case, *environment data* represents the situation surrounding the interaction (e.g., the patient miraculously recovers, a co-worker spreads gossip about the ZDDA, etc.).

9. **Re-evaluate Perceived Value.** The doctor then performs the *Re-evaluate Perceived Value* function which takes the *machine output* and new *situation information* to produce an updated *perceived value* and *need*.

10. **Decision #2.** The doctor now returns to Decision #2 to decide if the benefits outweigh the costs of accepting the diagnosis. Below are both scenarios:

    (a) **Benefits don't outweigh costs.** The doctor thinks that the benefits don't

outweigh the costs and rejects the diagnosis. Thus the doctor diagnoses the patient themselves (*Do It Myself*).

(b) **Benefits outweigh costs.** The doctor thinks the benefits do outweigh the costs and accepts the diagnosis. This leads them to Decision #3 where their need to help the patient is now satisfied.

11. **End.** The interaction has now ended.

3.1.2.3 Example 2: The TAMS

To show how the HMI model can work with a widely different machine, the TAMS interaction is explained below:

1. **Start.** The interaction begins when Agent Keegan's boss tells her to use the TAMS, providing a *stimulus*. Along with this *stimulus*, Agent Keegan perceives the current *situation information* (e.g., her views on robots, the current year, etc.).

2. **Establish Need.** Using this *stimulus* and *situation information*, she establishes a *need* to use the TAMS. Note again that the *Establish Need* function also passes along the *situation information*.

3. **Decision #1.** Agent Keegan now decides if using the TAMS will help to fulfill her *need*. Since her *need* is to use the TAMS, she continues on the *Machine can fulfill need* path. If she had a different *need* where the TAMS would not help, she would consider following the *Machine cannot fulfill need* path and perform the *Do It Myself* function, which terminates the interaction.

4. **Evaluate Initial Perceived Value.** Given the *situation information* and *need*, Agent Keegan uses the *Evaluate Initial Perceived Value* function to perform a cost-benefit analysis of using the TAMS. This outputs an initial *perceived value*.

5. **Decision #2.** This *perceived value* is used to determine if the benefits of using the TAMS outweigh the costs. For this example, we assume they do and therefore follow the *Benefits outweigh costs* path. If they did not, Agent Keegan would follow the *Benefits don't outweigh costs* path and perform the *Do It Myself* function, terminating the interaction.

6. **Decision #3.** Agent Keegan now checks if her current *need* has been fulfilled. In this case it has not, so she continues on to use the TAMS, following the *Need not fulfilled* path.

7. **Perform Task.** Agent Keegan turns on the TAMS and may also provide commands to the TAMS (*user input*). The TAMS takes this input along with *environment data*, in the form of sensor data, obtained through the *Get Observed Environment* function, to create a single unit of *machine output* using the *Perform Task* function.

8. **Generate New Knowledge.** While the TAMS is generating *machine output*, Agent Keegan is learning new knowledge about the current situation by using the *Generate New Knowledge* function. This function calls *Get Observed Environment* to obtain information about the environment. This *environment data* can then be perceived as an updated *situation information*.

9. **Re-evaluate Perceived Value.** Using the *machine output* and *situation information*,

Agent Keegan updates the *perceived value* and *need* with the *Re-evaluate Perceived Value* function.

10. **Decision #2.** Agent Keegan now uses the new *perceived value* to determine if the benefits of continuing to use the TAMS outweigh the costs. Below are both scenarios:

    (a) **Benefits don't outweigh costs.** Agent Keegan believes that the benefits of keeping TAMS on do not outweigh the costs. So, she turns TAMS off and uses the *Do It Myself* function to fulfill her *need*. Note that Agent Keegan's original *need* to use the TAMS may have changed given new *situation information*. It is possible she is now looking to complete a mission where the TAMS may get in her way.

    (b) **Benefits outweigh costs.** If Agent Keegan believes the benefits outweigh the costs of using the TAMS, she will continue to use the TAMS and move on to Decision #3. Once at Decision #3, if Agent Keegan's *need* has been fulfilled, then she will terminate the interaction, otherwise, she will continue to use the TAMS.

11. **End.** The interaction ends when either Agent Keegan believes the benefits of the TAMS don't outweigh the costs and she uses the *Do It Myself* function or her *need* is fulfilled.

3.1.2.4 HMI Failure Modes: Abuse, Misuse, and Disuse

The nature of HMI has the potential to produce three types of failure modes (shown by dotted red arrows in Figure 3.1). These errors are described by [53] as abuse, misuse, and disuse. Abuse refers to a machine that is being used outside the scope of its intended purpose; misuse describes an over-reliance on technology; and disuse is characterized by an under-utilization of technology.

An error of abuse occurs when a user decides whether or not a machine will be able to fulfill their need (i.e., the first user decision in the HMI model). In terms of the provided examples, this could happen if a doctor attempts to use the ZDDA to diagnose a condition other than a Z-Virus infection or if Agent Keegan attempts to use the TAMS for a task that requires human intuition. Misuse and disuse errors can both occur during the cost-benefit analysis portion of the HMI model (i.e., the second user decision). These errors are directly related to an inaccurate perception of machine value, with misuse being an overestimation of value and disuse being an underestimation of value. For instance, a misuse error occurs in the ZDDA example when the doctor accepts an incorrect diagnosis from the ZDDA. Similarly, this could happen if Agent Keegan believes that the TAMS performs better than it actually does, causing a mission failure due to over-reliance. Disuse errors occur in these examples when the doctor using the ZDDA rejects a correct diagnosis or if Agent Keegan does not believe that the TAMS is correct about a situation, causing her to ignore the robot. For the purposes of this thesis, I focus specifically on the second area of failure (i.e., errors of the second user decision) and assume that users have a general understanding of the intended purpose of an AI.

In an ideal scenario, the user would be able to use the AI as often as possible while always making a decision that avoids these error modes. This ensures that developers are spending their time and money developing AIs that will actually be used by operators while simultaneously avoiding the over-reliance and under-utilization of the AI.

3.2 Defining Coherent Use

In most cases, developers tend to focus on providing unmatched performance to their users, focusing specifically on the *Perform Task* function within the workflow model. This can be understood through a meta-analysis of papers from both the International Conference on Machine Learning and the Neural Information Processing Systems. The analysis shows that top keywords from these publications include "learning," "networks," "optimization," "efficient," etc., with words like "human" and "interaction" not making the list [54]. This emphasis on performance can be equated with a focus on optimizing a technology's *trustworthiness*. [23] describes trustworthy automation as "automation that performs efficiently and reliably." Thus, when a developer is attempting to provide the best possible value to the user, they are attempting to ensure an AI's trustworthiness in terms of technical performance. However, this focus on performance only represents half of the story, as the model of HMI demonstrates that errors of misuse and disuse result when a user has a false sense of the machines value (resulting at the second user decision of the HMI model). The perception of value can be equated with the concept of trust calibration (or simply calibration) in the human-machine trust/interaction literature [23, 30, 55]. From [23], "*calibration* refers to the correspondence between a person's

trust in automation and the automation's capabilities (Lee & Moray, 1994; Muir, 1987)." In terms of AI, this means that trust calibration relates to the perceived value of an AI, which is about correctly identifying when to trust an AI and when to question its output.

In order to avoid errors of misuse and disuse, developers need to focus not only on providing value in terms of performance, but also ensuring that the perception that a user has of this value is accurate. This dual focus is characterized within the concept of *Coherent Use*, which is demonstrated in Table 3.5. As shown, any deficit in either trustworthiness or trust calibration results in either misuse or disuse. Note that for the purposes of this thesis, disuse is extended to include scenarios where a machine is not being used, even if this is the correct choice by the user (the machine is untrustworthy). Although this is not dangerous from a high-stakes AI perspective, it is still an error in development since developers spent time and money to create an AI that is not being used. Thus, Coherent Use is a state of machine use in which the machine is trustworthy and allows for trust calibration from the user. This means that the machine provides high value to the user (in terms of high performance) and that the user is able to correctly identify this high value, therefore establishing coherency with the machine. I argue that this state is the basis for the best HMI.

One important note regarding trustworthiness is the popular concept of Trustworthy AI, which is described in a variety of recent works [56–59]. These references explicitly mention transparency and explainability when referring to "trustworthiness" and therefore lead to the belief that this phrase encompasses both trustworthiness and trust calibration. While the inclusion of trust calibration components within this concept is admirable, I assert that trust calibration is important enough when attempting to avoid misuse and

Trust Calibration	Trustworthiness: Correct Output	Trustworthiness: Incorrect Output
Calibrated	Coherent Use (Machine was correct and the user trusted it)	Disuse (Machine made a mistake but the user didn't trust it)
Uncalibrated	Misuse/Disuse (Machine was correct but the user didn't trust it)	Misuse (Machine is not trustworthy yet the user trusts it)

Table 3.5: Human-Machine Interaction failure modes.

disuse that it should be properly abstracted from trustworthiness. This ensures that trust calibration is given the attention that it deserves as calibration errors can lead to disastrous outcomes on the user and surrounding environment.

Chapter 4: A Goal, Question, Metric Template for Coherent Use

This chapter introduces a GQM template that seeks to aid developers in maximizing the Coherent Use of a piece of AI-enabled technology. This approach integrates directly into the DevOps lifecycle as a means to not only ensure the quick development and delivery of these technologies but also to assert the consideration of human-AI coherency from the very start of development.

4.1 Defining the GQM Template

As explained in section 2.3, the GQM software evaluation approach involves three main components: overarching goals, questions that aid in achieving those goals, and metrics that answer the defined questions. In the following subsections, I will first define an overarching goal based on the previously established need for human-AI coherency, then I will propose template questions that will help to achieve this goal based on the model of HMI, and finally I will provide example metrics that may be used to answer these questions.

4.1.1 The Overarching Goal

Recall from section 2.3 that GQM goals have four main components: a *purpose*, *object*, *issue*, and *viewpoint*. Based on the need for human-AI system coherency, Coherent Use is the *issue* that is being focused on for the goal. The *purpose* of the goal is the maximization of human-AI coherency as this focuses on the need to minimize the errors of misuse and disuse that were described in Chapter 3. In this case, the *object* of the goal is the AI-enabled technology for which the Coherent Use is attempting to be maximized. Finally, the *viewpoint* of the goal is from the perspective of the developers themselves as they will be doing the measurements and interpreting the results.

Therefore, the template overarching goal is as follows: **Maximize (*purpose*) the Coherent Use (*issue*) of <*the AI-enabled technology*> (*object*) from the viewpoint of the developer (*viewpoint*)**.

4.1.2 Template Questions

Based on the defined goal, template questions can be made that help to characterize the achievement of this goal. Since the GQM process has questions follow a model, the HMI model described in section 3.1.2 will be used as a basis. This model shows the components of Coherent Use to be trustworthiness (*Perform Task*) and trust calibration (*Decision #2*). From this, four initial questions can be created:

1. Is <*the AI-enabled technology*> trustworthy?

2. Does <*the AI-enabled technology*> allow for trust calibration by the user?

3. Is Coherent Use increasing, decreasing, or remaining constant between delivered software builds?

4. Is the current rate of Coherent Use sufficient from the viewpoint of the developer?

Questions 1 and 2 are based directly on the workflow model for HMI, describing the need for trustworthiness (in terms of system performance) and easy trust calibration by the user. Both questions 3 and 4 are more goal oriented, focusing more on the *purpose* and *viewpoint* of the goal. Question 3 seeks to ensure that progress is being made toward the maximization of Coherent Use. However, maximization is difficult to measure and will likely never be achieved since errors of misuse and disuse may never be fully mitigated (there are still influences of trust that are outside of the developer's control). Therefore, this question monitors the levels of Coherent Use and describes the fluctuations that occur in reference to this coherency. Question 4 is included as a means to understand the developers perception of coherency and whether they believe that the current levels of Coherent Use are sufficient. While questions 3 and 4 go beyond the model of HMI, their inclusion is imperative to measure the overall human-AI system performance.

4.1.3 Example Metrics

The GQM approach requires metrics that help to answer the goals defined above. Below are some example metrics that could be used to answer each question.

4.1.3.1 Question 1

The first question described focuses on the trustworthiness of a piece of AI-enabled technology. Following the view that trustworthiness can be equated with AI performance, this question is answered by measuring a technology's capabilities. This encompasses not only raw mathematical performance but also all relevant "ilities", such as reliability, robustness, etc. [60]. Note that these metrics may look different from technology to technology, as different development teams may have different resources and each AI will have unique capabilities. However, some general metrics that could be used to answer this question are an AI's accuracy, robustness, and frequency of failure.

4.1.3.2 Question 2

Question 2 focuses on an AI's ability to allow for easy trust calibration by an operator. Within the reference model for HMI (Chapter 3), this is associated with a user's perception of the value a piece of technology can provide. When a fully deployed system exists, this question can be answered more explicitly through user testing, however, many times development of the AI is still in progress. This motivates a need to understand how to answer this question when a piece of AI-enabled technology is still immature. Within the literature, trust calibration is generally associated with transparency and explainability [30, 55, 61]. Therefore, metrics for this question can also focus on these elements when a software product is not yet mature. Transparency itself is a rather large topic and extends to not only transparency upon AI output but also transparency in the form of user awareness of the software's limits of use. Some example metrics for trust calibration

include the number of misuse and disuse accidents, the performance/correctness of an explainable AI interface, and communicated limits of use.

4.1.3.3 Question 3

The purpose of question 3 is to aid in measurement toward the goal of Coherent Use maximization. As explained, the maximization of this property is hard to measure, as errors of misuse and disuse will likely never be fully mitigated. Thus, this question focuses on Coherent Use fluctuations, which involve comparisons to prior measurements. This can be measured through the compared frequencies of current and prior use and the compared levels of misuse and disuse compared to past levels.

4.1.3.4 Question 4

Question 4 depends on the developer's perception of Coherent Use sufficiency. This question can be answered through the subjective analysis of overall coherency by the developer.

4.2 Template Integration into the DevOps Lifecycle

The above questions can be integrated into the DevOps lifecycle. There are a number of benefits that motivate this integration, including the assertion of agility into the AI development process, the consideration of Coherent Use early on in development, and the ability to integrate into a well-know lifecycle that many developers are likely already aware of.

However, during the DevOps lifecycle, there are times when certain capabilities do not yet exist to be tested and this makes it difficult to answer the questions described as they are currently worded. Thus, it is necessary to break down the lifecycle into phases which will then allow for the creation of questions that can be meaningfully answered in each phase. The three GQM testing phases, along with the corresponding DevOps lifecycle phases are as follows:

1. **Active Development**: Plan, Develop;
2. **Pre-Deployment Testing**: Verify, Test, Deploy; and,
3. **Post-Deployment Testing**: Operate, Monitor.

Accompanying each of these phases, are a set of questions and example metrics that correspond to the state of the technology during each phase. An overview of the entire GQM template is shown in Table 4.1.

<table>
<tr><th colspan="4">Goal</th></tr>
<tr><td colspan="4">Maximize (purpose) the Coherent Use (issue) of <the AI-enabled technology> (object) from the viewpoint of the developer (viewpoint).</td></tr>
<tr><th>GQM Testing Phase</th><th>DevOps Phase</th><th>Questions</th><th>Example Metrics</th></tr>
<tr><td rowspan="2">Active Development</td><td>Plan</td><td rowspan="2">Q1.1: Does <the AI-enabled technology> design promote trustworthiness?

Q1.2: Does <the AI-enabled technology> design create a potential for trust calibration?

Q1.3: Is the current rate of Coherent Use sufficient from the viewpoint of the developer?*</td><td rowspan="2">M1.1: AI accuracy requirements; AI robustness requirements

M1.2: # of requirements asserting AI transparency; design meeting notes

M1.3: Developer subjective understanding of current AI coherency*</td></tr>
<tr><td>Develop</td></tr>
<tr><td rowspan="3">Pre-Deployment Testing</td><td>Verify</td><td rowspan="3">Q2.1: Is <the AI-enabled technology> showing trustworthy behavior in controlled scenarios?

Q2.2: Are <the AI-enabled technology>'s components related to trust calibration working properly?</td><td rowspan="3">M2.1: AI Accuracy; AI Reproducibility; # Passed Requirements

M2.2: Accuracy of XAI/Model Confidence; Documentation of AI Performance; Documentation of Limits of Use</td></tr>
<tr><td>Test</td></tr>
<tr><td>Deploy</td></tr>
<tr><td rowspan="2">Post-Deployment Testing</td><td>Operate</td><td rowspan="2">Q3.1: Is <the AI-enabled technology> showing trustworthy behavior post-deployment?

Q3.2: Are users able to calibrate their trust while using <the AI-enabled technology>?

Q3.3: Are instances of Coherent Use increasing, decreasing, or remaining constant between delivered software builds?</td><td rowspan="2">M3.1: Post-Deployment Accuracy; Incidences of Software Failure

M3.2: # Misuse/Disuse Accidents

M3.3: Compared # of Misuse/Disuse Accidents; Compared Software Usage</td></tr>
<tr><td>Monitor</td></tr>
</table>

Table 4.1: The GQM template for Coherent Use integration into the DevOps lifecycle.

4.2.1 Active Development

During the Active Development phase, the software product does not actually exist yet to be tested explicitly. Therefore, the questions in this phase focus on a *potential* for trustworthiness and trust calibration. These types of questions are generally answered through requirement analysis and documented design considerations. It is also important to remember that DevOps is an iterative process, and thus the Active Development phase will follow the Post-Deployment Testing phase. This provides the GQM template an opportunity to reflect on the observed levels of Coherent Use during the Post-Deployment Testing phase and therefore motivates the inclusion of the previously defined Question 4 during this phase. However, as noted in Table 4.1 by a *, this question cannot be answered during the first iteration of the DevOps lifecycle as there is no understanding of the current level of Coherent Use since the product has not yet been deployed. During the second iteration and for all iterations that follow, this question should be answered through the subjective understanding of Coherent Use levels by the developer.

4.2.2 Pre-Deployment Testing

The Pre-Deployment Testing phase occurs during the Verify, Test, and Deploy DevOps phases and thus can begin to test Coherent Use on the actual software product. In this phase specifically, there is a focus on testing the *performance* of the software. This includes the overarching AI performance (trustworthiness) as well as the performance of transparency and explainability features that aid in trust calibration. This helps to ensure that the AI is not only proficient in performing its defined task (e.g., driving, providing

a diagnosis, etc.) but also to assert that the methods used to promote trust calibration work (e.g., providing the correct model confidence level, giving accurate explanations, providing accurate documentation on limits of use, etc.). Example metrics for these questions include overall AI accuracy, number of passed transparency requirements, etc.

4.2.3 Post-Deployment Testing

The last testing phase in a single iteration of the DevOps lifecycle is the Post-Deployment Testing phase. This phase focuses on the AI's trustworthiness and calibration in the real world post-deployment. These questions can be answered with the analysis of post-deployment accuracy, number of misuse and disuse accidents, etc. In addition to these types of questions, the overall human-AI system coherency can be evaluated with the previously defined Question 3. This question measures the fluctuations of Coherent Use and can be measured through compared software usage metrics, compared numbers of misuse and disuse accidents, etc. However, in order to answer this question, a developer needs an understanding of past levels of Coherent Use which would be unavailable during the first iteration of the DevOps lifecycle. Thus, this question cannot be answered until the second DevOps iteration. Note, however, that metrics that aid in the understanding of Coherent Use (i.e. misuse and disuse accidents, software usage, etc.) still need to be gathered during the first iteration to be used in the next iteration. Thus, Table 4.1 does not mark this question with an * (as it does with Q1.3) because a measurement still needs to take place. The answer to this question will also aid in the subjective developer understanding of Coherent Use needed to answer the last question in the Active

Development phase.

4.3 Example: Evaluating the ZDDA

To show how this template can be used in practice, I return once again to the ZDDA. Recall from Chapter 1 that the ZDDA was created using a linear, waterfall methodology that paid little attention to the coherency of the AI. Now imagine if the above template was used when creating the ZDDA with the following goal:

Maximize (*purpose*) the Coherent Use (*issue*) of the ZDDA (*object*) from the viewpoint of the developer (*viewpoint*).

4.3.1 Phase 1.1: Active Development

As per Table 4.1, the first Active Development phase focuses on two questions:

- **Q1.1**: Does the ZDDA design promote trustworthiness?
- **Q1.2**: Does the ZDDA design create a potential for trust calibration?

During the ZDDA planning and development, the developers address these questions through the use of two metrics: the F_1 score asserted by the requirements (Q1.1) and the number of requirements asserting algorithm transparency (Q1.2). For the sake of this example, let's say that the developers assert that the algorithm must achieve an F_1 score of 0.90 in their established requirements. From this, they answer Q1.1 as they do believe that this ZDDA design promotes trustworthiness. However, when moving to Q1.2, the developers discover that they have 0 requirements that assert the transparency of the ZDDA algorithm, therefore, they do not believe that the ZDDA design will create

a potential for trust calibration. From this realization, the developers decide to add a new feature to the ZDDA that will display a model confidence percentage along with the positive or negative diagnosis. This new model confidence feature adds 5 new algorithm transparency requirements which the developers believe now creates a potential for trust calibration. The developers then move on to the Pre-Deployment Testing phase.

4.3.2 Phase 2.1: Pre-Deployment Testing

The Pre-Deployment Testing phase contains the two following questions:

- **Q2.1**: Is the ZDDA showing trustworthy behavior in controlled scenarios?
- **Q2.2**: Are the ZDDA's components related to trust calibration working properly?

The developers decide to measure the ZDDA's F_1 score for Q2.1 and model confidence accuracy percentage for Q2.2. They calculated the F_1 score as 0.90, which passes the performance requirement described earlier. This answers Q2.1 as the developers believe that this shows the ZDDA is trustworthy in a controlled testing scenario. In this example, the model confidence accuracy percentage is the percentage of time that the displayed model confidence percentage is within 5% of the true model confidence percentage. The developers found that their model confidence feature had an accuracy percentage of only 60%, which they felt was insufficient to confirm that the feature was working properly. Therefore, they iterated on their design until they were able to achieve an accuracy percentage of 95%. Once this was achieved, they deployed the ZDDA.

4.3.3 Phase 3.1: Post-Deployment Testing

Now that the ZDDA has been deployed, the focus shifts to the following three questions:

- **Q3.1**: Is the ZDDA showing trustworthy behavior post-deployment?
- **Q3.2**: Are users able to calibrate their trust while using the ZDDA?
- **Q3.3**: Are instances of Coherent Use increasing, decreasing, or remaining constant between delivered software builds?

To answer these questions, the developers decide to use the post-deployment F_1 score (Q3.1) and the ratio of misuse and disuse incidences to incidences of Coherent Use (Q3.2 and Q3.3). They find that the overall post-deployment F_1 score of the ZDDA was only 0.75 due to bias in their training dataset. This led the developers to believe that the ZDDA was not showing trustworthy behavior post-deployment. The misuse and disuse ratio was found to be 1:10 (for every case of misuse/disuse there are 10 cases of Coherent Use). This shows that while developers were able to calibrate their trust, there is still more that the developers could do to improve this ratio further (Q3.2). Unfortunately, at this stage, Q3.3 cannot yet be answered since there is no previous ratio to compare the current ratio to. However, this question is still included in this iteration of the DevOps lifecycle because it establishes the importance of measuring Coherent Use in this phase. In later iterations of this template, this value will be used to answer Q3.3.

4.3.4 Phase 1.2: Active Development, Again

After completion of one DevOps iteration, the information gathered can be used to inform new requirements and feature development. In this phase, Q1.1 and Q1.2 are asked once again, however, this time a third question is asked as follows:

- **Q1.3**: Is the current rate of Coherent Use sufficient from the viewpoint of the developer?

The purpose of this additional question is to motivate the continuous improvement of the software. While the AI did perform well, the developers were not satisfied with the levels of Coherent Use that were measured during Post-Deployment Testing. This drives the developers to create new requirements that assert the scope of their training dataset (which will aid in performance and thus trustworthiness) as well as establish new requirements for additional transparency features (asserting improved trust calibration). The developers do this with the hope that these additional requirements improve their misuse/disuse ratio even further.

4.3.5 Discussion

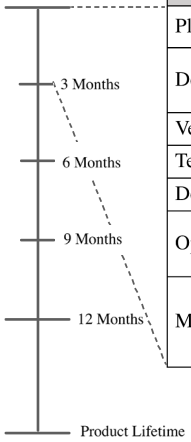
The purpose of this example is to show the benefits of using the proposed GQM framework. Figure 4.1 highlights these advantages with a comparison to the traditional method described within Chapter 1. This graphic was created assuming a realistic year long development process for the traditional method and quarterly releases in the DevOps approach. When using the traditional waterfall method, not only did the ZDDA take

significantly longer to deploy, but once released, it had major issues of misuse and disuse. These issues led to an overall failed product. In contrast, the DevOps method with GQM integration continually caught and fixed bugs related to Coherent Use from the very start of development. The developers were then able to iterate on their product and only release until after it satisfied the given questions. Although the ZDDA did not perform perfectly on the first release with this method, it did perform significantly better with a misuse/disuse ratio of 1:10 rather than 5:1.

4.3.5.1 Cost Savings

A notable advantage to this approach is the discovery of bugs pertaining to human-AI coherency from the very start of the DevOps lifecycle. As explained by the IBM System Science Institute in reference to security bugs, issues that are discovered during testing can be 15x more expensive to fix than bugs that are found during the design phase (within the Waterfall model) [62]. These metrics easily carry over to bugs related to human-AI coherency when dealing with high-stakes software as these issues must be remedied quickly to avoid fatal errors of misuse and disuse. The sooner that bugs can be discovered, the cheaper they are to fix, asserting the importance of coherency bug discovery early on in the software lifecycle.

Traditional Waterfall Method (1 Year of Development)	
Waterfall Phase	Summary
Requirement Analysis	No bugs found
Design	No bugs found
Implementation	Minor bugs found and fixed throughout development
Testing	Performance fixes until $F_1 = 0.95$
Operation and Maintenance	**Bug found –** High levels of misuse and disuse (5:1 ratio) ***Defect costly to fix***



GQM for Coherent Use w/ DevOps Integration (Quarterly Releases – 3 Months of Development)		
DevOps Phase	GQM Testing Phase	Summary
Plan	Active Development	**Bug Found –** Lack of system transparency requirements
Develop		
Verify	Pre-Deployment Testing	**Bug found –** Inaccurate model confidence
Test		
Deploy		
Operate	Post-Deployment Testing	**Bug found –** Inaccuracy of model **Bug found –** Elevated levels of misuse and disuse (1:10 ratio)
Monitor		

Figure 4.1: Comparison of development methodologies for the ZDDA.

Chapter 5: The Surveillance Use Case

In this chapter, I will describe a final use case utilizing a DoD AI-enabled software product of which researchers at the Applied Research Laboratory for Intelligence and Security (ARLIS) performed a usability T&E. The ARLIS team provided access to a variety of documents from which actual metrics could be analyzed, including the original product design, the T&E report on the software performance, and the final usability report created by the ARLIS team. Using these documents along with video of the software in use, one iteration of the DevOps lifecycle utilizing the GQM template could be simulated in a similar way as the ZDDA example provided in Chapter 4.

Although this simulated GQM DevOps iteration is purely hypothetical (developers did not use this template during the creation of the technology) there are still valuable insights to be gained. By performing this GQM iteration actual data and metrics that were used during the creation and T&E of the software product can be analyzed. This process will also aid in the understanding of how these technologies are made in practice and will provide insight into how easy this template can be integrated into an existing development process. In addition, it will allow for an understanding of whether the template questions are worded in such a way that they can be fully answered (of course from the somewhat subjective viewpoint of the developer).

In the following sections I will first describe the AI-enabled surveillance product used for this final evaluation. Then, I perform one iteration of the GQM analysis. This will involve defining the goal and example questions as well as data analysis and interpretation at each testing phase. Finally, I will discuss insights from the hypothetical template utilization.

5.1 The Security Surveillance AI-enabled Technology

The technology used for this final use case analysis is a camera-based security surveillance technology used by the DoD to detect potential threats in Force Protection Missions (e.g., base security). This technology uses a variety of cameras to display video feeds to human operators. The original surveillance technology did not contain any AI plugin tools and relied on human monitoring for the detection of potential threats. This led to the risk of missed detections and required continuous operator monitoring that was limited to the attention and perception of these operators. To assist the human operators, a computer vision plugin software was developed that performs two main functions. The first is the detection of a threat given the surveillance camera input. The second is the classification of this potential threat as either a human or vehicle. Along with this output, the surveillance technology, once fitted with the computer vision plugin, displays bounding boxes over these potential threats which human operators can look at to understand what the software believes to be a threat.

As mentioned previously, this AI tool has already been developed and deployed without the use of this GQM approach. However, in the next section we place ourselves

in the shoes of the developer to imagine how these questions could be answered and, through continuous iteration, could lead to a goal of Coherent Use maximization.

5.2 Analysis via the GQM Template

Given the GQM template described in Chapter 4, a hypothetical analysis of the template can be performed on the camera-based security surveillance AI plugin utilizing actual product design and T&E reports. Before jumping into each DevOps GQM testing phase, the goal must first be established from the template provided in Chapter 4:

Maximize (*purpose*) the Coherent Use (*issue*) of the AI plugin (*object*) from the viewpoint of the developer (*viewpoint*).

Given this goal, each of the template questions shown in Table 4.1 can be described and answered using appropriate metrics from the AI plugin documentation. This analysis will assume that these documents represent the first iteration of the DevOps lifecycle. Note that given the sensitive nature of the data produced during the design and T&E of the AI plugin, some of the actual data has been omitted in this thesis. Rather, at each phase, there will be a general discussion of the key findings from the document analysis. This is done to avoid the public disclosure of actual data.

5.2.1 Active Development

The Active Development phase of the GQM template focuses on the planning and development phases of the DevOps lifecycle. This phase contains three questions of which the following two can be answered on the first iteration of the DevOps lifecycle:

- **Q1.1**: Does the AI plugin design promote trustworthiness?
- **Q1.2**: Does the AI plugin design create a potential for trust calibration?

Given this definition of the GQM questions, metrics obtained from the system documentation can be utilized to answer them. The main document that will be used to answer these questions is the Mission Product Requirement Document (MPRD), which outlines the purpose and intended capabilities of the AI plugin.

5.2.1.1 Q1.1 - M1.1: MPRD Requirement Analysis

Q1.1 focuses on the potential for trustworthiness given the design of the software. Recall from Chapter 3 that, for the purposes of this thesis, trustworthiness is equated with system performance. The MPRD describes a variety of performance requirements for the AI plugin including three key performance parameters. These parameters define the objective recognition rate of recognizing a human through the video feeds, predicting whether that human is a friendly, and predicting whether that human is an adversary. Note that these initial performance requirements seemingly shifted during the development of the software as the final deployed product only recognizes threats and classifies them as human or vehicle. The key performance parameters are followed by the description of key system attributes which assert additional requirements. Of relevance to performance are three additional requirements asserting the objective Area Under Curve value, minimum Average Precision value, and the situations in which the system will be trained and thus expected to work (partial occlusion, crowded areas, etc.). Additionally, the design document outlines a T&E plan. This plan describes a variety of algorithmic metrics that

will later be tested, including a confusion matrix, precision, recall, and average precision, among others.

From these documentation observations, it is evident that the developers had a well thought out plan to create an incredibly performant AI tool. They define explicit requirements for the algorithm performance and describe a variety of metrics that can later be used to test the performance of the technology. Given the size and relative simplicity of the software it is likely that the developers would have felt this level of documentation creates a potential for trustworthiness allowing them to answer this question affirmatively.

5.2.1.2 Q1.2 – M1.2: MPRD Software Transparency Design Analysis

Q1.2 shifts the focus toward a potential for trust calibration by the operator. As discussed previously, calibration is associated with the transparency of the AI plugin and thus the metric used to answer this question is an analysis of the MPRD document in reference to designing for algorithmic transparency. Unfortunately, the MPRD does not discuss transparency nearly as much as it discusses the planned performance of the software. However, the design document does describe the classification of potential threats. This can be viewed as a form of transparency because the operator can understand what type of threat the surveillance technology believes is a problem. The operator could then have a better understanding of why the plugin believes that an object poses a potential risk and gives the operator an opportunity to catch mistakes made by the AI. In addition to this classification, video footage of the working plugin (provided by the ARLIS research team) shows a bounding box that surrounds the potential threat after detection. This

bounding box itself is not only useful to show where the threat is but is yet another form of software transparency. Given this bounding box, the software can show the operator where they believe this potential threat is and points the object out directly on the video feed. From this information the operator can determine if the algorithm is correct or incorrect in their classification and detection of a potential threat. Along with this, the MPRD also describes T&E measures that assert the correctness of the bounding box placement. Specifically, these measures include both the Polygon Similarity Metric and the Intersection over Union measure.

Although none of the above transparency considerations were described as actual requirements of the AI plugin, they do at least show that the developers were considering the user when designing the software. It is possible that had the developers used the GQM template, they may have added additional transparency features and set concrete requirements for algorithm transparency. There is certainly room for improvement within the developer's consideration for trust calibration, however, given that they could not have used this template, the design does appear to promote at least a base level of trust calibration potential.

5.2.2 Pre-Deployment Testing

The Pre-Deployment Testing phase consists of the verify, test, and deploy phases of the DevOps lifecycle. Recall from Chapter 4 that the questions asked in this phase focus on various aspects of the *performance* of the software. Following the GQM template, the questions in this phase are as follows:

- **Q2.1**: Is the AI plugin showing trustworthy behavior in controlled scenarios?
- **Q2.2**: Are the AI plugin's components related to trust calibration working properly?

To answer these questions, an additional document can be used to gather data regarding the software's performance: the T&E Report produced by a DoD contractor. The metrics used to answer these questions come from this report, however, note that this data is not reported in this thesis to avoid public disclosure.

5.2.2.1 Q2.1 - M2.1: Precision, Recall, and F_1 Score

Q2.1 focuses on the trustworthiness of the software within controlled scenarios. Based on previous chapters, this question can be answered by analyzing the performance of the AI plugin during the T&E of the software product. Luckily, the T&E Report provided by the ARLIS research team has recorded a variety of useful metrics that can be used to measure the performance of the software including Precision, Recall, and F_1 Score. Before discussing the analysis of the data collected, it is necessary to first define what is meant by True Positive, False Positive, False Negative, Precision, Recall and F_1 Score.

True Positive. A True Positive (TP) represents the case where a classification algorithm correctly identified a positive result. In the context of this use case, the T&E Report explains that a TP occurs when the AI plugin returns a bounding box in the correct location around a threat and correctly identifies the classification of that threat. The threshold for the correctness of the bounding box is described in the Report as a specific Intersection over Union value (which will be discussed further during the discussion of

Q2.2).

False Positive. A False Positive (FP) represents the case where a classification algorithm incorrectly identified a positive result. The T&E Report describes that this occurs when the AI plugin returns a bounding box around a non-threat or incorrectly classifies a threat. An example of the latter being a correctly identified threat that was classified as a vehicle when it was a person.

False Negative. A False Negative (FN) represents the case where a classification algorithm misses a positive result. Within this use case, this occurs when the AI plugin completely misses a threat, returning no bounding box or classification.

Precision. The Precision metric used for classification algorithms represents the proportion of positive classifications that were correctly classified. This metric can be obtained through the following equation:

$$P = \frac{TP}{TP + FP} \tag{5.1}$$

An AI model with high precision is rarely incorrect about a positive result, however, it could have still potentially missed some positive classifications. In the context of the AI plugin, a precise model returns a bounding box and classification that is usually correct, although, it may still have some FNs (i.e., missing a threat).

Recall. The Recall metric is used for classification algorithms and represents the proportion of TP outcomes that were correctly classified. This metric is obtained using the following equation:

$$R = \frac{TP}{TP + FN} \tag{5.2}$$

An AI model with a high recall rarely misses a positive classification, however, it may result in many FPs. For this use case, a high recall means that the AI plugin is unlikely to miss/incorrectly classify a threat, although, it may still return many FPs (e.g., showing a bounding box where there is no threat).

$\mathbf{F_1}$ Score. An F_1 Score is a metric that seeks to combine both the Precision and Recall of the model by taking the harmonic mean of both metrics. Thus, to have a high F_1 score, both Precision and Recall need to be high. Below is an equation that calculates the F_1 score given the Precision (P) and Recall (R):

$$F_1 = 2 * \frac{PR}{P + R} \tag{5.3}$$

Given the above information about the relevant metrics, data from the T&E Report can be used to evaluate the trustworthiness of the AI Plugin in terms of its performance. One thing to note in reference to performance is that the T&E Report describes a variety of different computer vision models that could be used. For the purposes of this analysis, the highest reported metrics are used. The reported overall Precision was subjectively quite high, and the Recall was subjectively low for this final model. Using the above equation, the F_1 score of this model was calculated and, unsurprisingly, was subjectively low. Thus, while the AI plugin is quite precise (there are few FPs), it does not do nearly as well at recall. This means that the plugin has the tendency to miss a lot of threats which also brought down the F_1 score of the plugin significantly.

Although these results are far from a perfect machine learning model, they do show that the model is somewhat precise. As part of the T&E Report there was also mention

that the developers were aware that the model could be improved with additional training data that they did not have at the time. Due to this, it is possible that the developers thought that the model was as performant as possible given the tools that they had. However, had the developers had access to this GQM breakdown and were aware of the importance of baseline performance of their algorithm when it comes to Coherent Use, it is possible that they may have held off on deployment until after their model improved. If users relied on the AI plugin too heavily, this lack of performance could certainly lead to a variety of serious security incidents.

5.2.2.2 Q2.2 - M2.2: IoU, Avg. FDA, and Limits of Use

Q2.2 places a focus on the performance of the capabilities that promote algorithmic transparency along with communicated limits of use. Note that some of the metrics used to answer Q2.1 are still relevant to this question as a TP requires both a correct classification as well as a correctly placed bounding box. Both the classification and bounding box were described as adding to overall transparency. This is because they point out what the algorithm is understanding as a threat, as well as the type of threat it believes it to be. Compared to a software that simply alerts of a threat at a location, this is certainly an added feature of transparency. Along with these metrics are three other metrics including the Intersection over Union (IoU), Average Frame Detection Accuracy (Avg. FDA), and communicated limits of use. I describe both IoU and Avg. FDA below.

Intersection over Union. The Intersection over Union or IoU is a computer vision metric that can be used to assert the correctness of bounding boxes. In this case, the

T&E Report describes the calculation of the IoU as the overlapping area of a ground truth bounding box with the bounding box produced by the AI plugin (Area of Intersection) divided by the union of the area of the two bounding boxes (Area of Union). The IoU can thus be calculated via the following equation:

$$IoU = \frac{Area\,of\,Intersection}{Area\,of\,Union} \tag{5.4}$$

A high IoU value indicates that the ground truth bounding box is very similar to the bounding box outputted by the AI plugin. Note that this metric is not explicitly reported within the T&E Report, however, it is used to provide a threshold for what is a correct bounding box placement.

Average Frame Detection Accuracy. The Average Frame Detection Accuracy (Avg. FDA) is an average of the FDA values across the frames in the video feed. Each FDA value for a single frame is calculated by taking the average of the IoU values of that frame. This allows the metric to assert the correctness of the bounding boxes as well as the detection of the correct number of threats. A high Avg. FDA value means that the model is producing bounding boxes that are close to the ground truth bounding box and is finding the correct number of threats in each frame.

Although the T&E Report does not specifically reference the IoU values that were measured and calculated, it does report the Avg. FDA value of the overall model. For the final deployed model, this value was subjectively low. Given the previously described performance metrics of the overall AI plugin, this is somewhat unsurprising. This means that the model does an okay job at creating accurate bounding boxes while at the same

time providing the correct number of detections per frame of the video feed. Again, it is possible that the developers saw this low value as a result of the limitations that were placed upon them from their lack of access to additional training data. However, in an ideal situation this value would be much higher, and it is possible that this lack of performance has an impact on the overarching human-AI system. In reference to communicated limits of use, the T&E Report does describe that the operators of the AI should be made aware of the performance limitations of the plugin. As a part of the deployment recommendation, the authors of the report explicitly mention that this tool should not be relied upon to detect every threat. This communication of the plugin's limitations will aid operators in understanding that the AI plugin will not be correct in a variety of circumstances, allowing them to calibrate their trust appropriately.

Given these metrics, it is possible that, had the developers used the GQM template during development, they would have worked to increase the Avg. FDA of the plugin. However, since the developers may have seen this low performance as a limitation due to a lack of training data and with their transparency in terms of communicated limits of use, they may have answered this question affirmatively.

5.2.3 Post-Deployment Testing

The Post-Deployment Testing phase is the last phase of the DevOps iteration and includes the operate and monitor phases of the DevOps lifecycle. Table 4.1 shows the three questions asked during this phase:

- **Q3.1**: Is the AI plugin showing trustworthy behavior post-deployment?

- **Q3.2**: Are users able to calibrate their trust while using the AI plugin?
- **Q3.3**: Are instances of Coherent Use increasing, decreasing, or remaining constant between delivered software builds?

The main document that will be used to gather metrics that will aid in answering these questions is the final report produced by the ARLIS research team. This document was created through the comparison of the baseline surveillance technology (without the AI plugin) and the AI-enabled surveillance technology (with the AI plugin). The researchers performed operator interviews and simulations that aided in their evaluation of the technology.

5.2.3.1 Q3.1 - M3.1: Theoretical Precision, Recall, and $\mathbf{F_1}$ Score

Unfortunately, there is no data that describes the accuracy of the AI plugin post-deployment. Although the report created by the ARLIS team describes the operator's perceptions of accuracy, this would not be a good representation of actual, ground truth model accuracy. Thus, for the purposes of the GQM analysis, we assume that the model shows similar performance as it did during the T&E of the plugin. Recall from the Pre-Deployment Testing phase that this included a subjectively high Precision metric, low Recall, and low F_1 Score.

In practice, it would be useful to know the post-deployment accuracy of the model as it could inform the development team of possible problems in their datasets used to train and test the AI. However, for this analysis, an assumption is made that there was no bias in the data and therefore the accuracy of the model is comparable prior to deployment

and post-deployment. Given the developers belief that the performance of the AI pre-deployment was adequate for deployment, it follows that they would also believe that this performance post-deployment is sufficient.

5.2.3.2 Q3.2 – M3.2: User Interviews and ARLIS Report Analysis

Q3.2 focuses on the ability of operators to calibrate their trust to the AI plugin's trustworthiness. To answer this question an analysis of the user interviews and key findings from the ARLIS Usability T&E Report will be used.

Within the report created by the ARLIS team, it is explained from user interviews that operators were able to appropriately rely on the surveillance AI plugin and could thus calibrate their trust correctly to the trustworthiness of the AI. Note however, that this meant that the operators did not necessarily rely on the AI plugin blindly, rather, they relied on it to help them find more potential threats than they could have by themselves. The users expressed that, while not explicitly told, they had an awareness of the AI's general accuracy. This means that they knew the technology had limitations including misclassifications, misses, and false positives. From this awareness, the operators showed that this led to less reliance on the AI plugin, even explaining that no action was ever taken based solely on a detection made by the algorithm. These detections were only used to inform the operator, who would then take it upon themselves to investigate the video feeds and determine if further action was needed. However, even though operators did not necessarily rely on the AI plugin, they did express that they felt it helped them find more detections than if they were to analyze the feeds themselves. This led the operators

to rarely turn off the alerts given by the AI (this was only done in high traffic areas) showing that, in general, they did like the technology enough to use it.

Overall, the analysis done by the ARLIS team indicates that operators were able to calibrate their trust, allowing a developer to respond to this question affirmatively. Note, however, that the ARLIS team only had the ability to interview three operators. Therefore, the subjective opinion that these operators had on the AI plugin may skew the results and it is entirely possible that, while these operators were able to appropriately rely on the AI plugin, other operators may not be able to do so as effectively.

5.2.3.3 Q3.3 - M3.3: User Interviews and ARLIS Report Analysis

The final question for this phase of the GQM breakdown is meant to help gauge the fluctuations of Coherent Use between DevOps iterations. Recall that this question cannot be answered during the first DevOps iteration as there is no other measure to compare it to, however, it is still important to discuss the current state of Coherent Use for the following iterations. Similarly to Q3.2, user interviews and insights gained from the ARLIS Usability T&E Report will be used in an attempt to measure Coherent Use.

While there are no measurements that can be taken from the ARLIS Usability T&E related to Coherent Use (e.g., number of misuse/disuse accidents), there are still insights to Coherent Use that can be described given the user interviews and key findings of the report. Recall from Q3.2 that operators were successful in calibrating their trust to the trustworthiness of the AI plugin. However, this calibration also meant that operators were not fully trusting the AI to make decisions for them, rather they used it to detect more

potential threats than they would have before. In this case, the machine is not being used coherently because the machine is not as performant as it would need to be for the operator to rely on it fully. Thus, the interaction between the AI and operator has subjectively low levels of Coherent Use. While this did not pose much of a threat to operations (operators report that they never acted on a potential threat due to the AI plugin alone), it is possible that increased model accuracy could allow operators to place more reliance on the AI. This could help to take even more workload off the operators who would not need to pay as much attention to the video feeds and only check on the AI when an alert is made.

5.2.4 Active Development, Again

After going through one iteration of the DevOps lifecycle, developers can circle back once again to the Active Development phase. Although there is no documentation that will allow us to answer Q1.1 and Q1.2, it is possible to reflect on the previous DevOps iteration and assert whether the levels of Coherent Use were sufficient. This would provide an answer to Q1.3:

- **Q1.3**: Is the current rate of Coherent Use sufficient from the viewpoint of the developer?

This question does not require any specific quantitative metrics, rather it is answered from the subjective viewpoint of the developers given the previous DevOps iteration. In doing this, developers are performing a retrospective and thinking about the current state of their AI with respect to Coherent Use. Given the previous iteration, it is likely that developers would feel that the AI is not sufficient in terms of Coherent Use. While

operators were able to successfully calibrate their trust, there are significant improvements to the AI plugin's trustworthiness that need to be made for the operators to use the machine coherently. Although the operators reported rarely turning off the AI plugin, they did not place much reliance on its output. While this may seem like Coherent Use (since there were very little accidents due to misuse or disuse), it is not. Coherent Use requires that the technology is also providing high levels of performance which would allow the operator to place more reliance on the machine, ideally removing some of the workload off themselves. Based on these insights, developers may consider improving their model to increase the AI's trustworthiness.

5.3 Discussion

Through this simulated iteration of the GQM analysis, it was shown how developers could use the template to evaluate their AI-enabled technology throughout the DevOps lifecycle. Since developers did not actually utilize the template during development, some questions were difficult to answer with concrete metrics. However, using the documents provided there are still some insights to be had on both the AI plugin and the template GQM analysis itself.

5.3.1 The Surveillance AI Plugin

First, it was clear that the surveillance AI plugin did not provide Coherent Use due to a lack of trustworthiness. Looking back at the HMI model presented in Chapter 3, user interviews revealed that operators chose to *Do It Myself* when they saw that the AI

plugin produced a false positive, of which it consistently did. Here, *Do It Myself* means to reject the output of the machine because the benefits of using the machine output do not outweigh the costs. However, this rejection of the machine output is not dangerous because this was an accurate perception of value from the user. According to the lack of trustworthiness of the AI plugin, the users should not be trusting it. Ideally, the machine would provide more correct outputs. This would allow the operator to agree with the machine and fulfill their need for security through the AI plugin's use. This has the potential to relieve even more of the workload off the operators.

However, even though the AI plugin did not produce Coherent Use during this first DevOps simulation, the technology itself was very impactful to the operators. The user interviews performed by the ARLIS team revealed that users felt the AI plugin helped them to find more potential threats than they could have done on their own. This is unsurprising because human operators have limitations. As humans they can only analyze one camera feed at a time, performing the equivalent of a serial search, while the AI can look at all feeds at once, "seeing" a full three-hundred and sixty degrees at any moment in time. Human operators may also have other tasks that they need to perform which require them to multi-task. The AI plugin does not have this issue because the only thing it needs to be concerned about is the analysis of the video feeds. This made the AI plugin deployment a great success as an augmentation technology rather than an automation technology. Although the operators would never act on an alert alone, they are certainly able to "see" more of their surroundings with the use of the plugin and user interviews showed that operators felt the plugin increased their efficiency. Just because something does not achieve Coherent Use on its first iteration does not automatically mean that it is

not a useful tool, rather, an awareness of a lack of Coherent Use can help developers to understand how their technology can become even more useful in the future.

5.3.2 The GQM Template

The surveillance AI plugin analysis revealed three main insights regarding the GQM template.

First, it showed the relative ease with which the template could be integrated into the DevOps lifecycle utilizing metrics that are already being recorded by developers. The developers of the AI plugin were not using this template when creating the software product, however, using the documents that were already being created, a hypothetical GQM analysis could take place. Although this analysis was somewhat limited by the metrics that could be taken given the documentation, it was still shown that the AI plugin suffered from a lack of trustworthiness and was successful in establishing calibrated trust. If the developers were creating this product with Coherent Use in mind, they could have used these existing metrics and easily added a few more to complete the analysis (specifically post-deployment performance metrics). Note, however, that it is possible that documentation exists regarding the AI plugin that was not given to the ARLIS research team. This additional documentation could have more metrics that help to round out the analysis.

Second, the GQM analysis highlighted the notion of sufficient transparency. One idea that is not explicitly discussed in the GQM template is how much transparency is necessary for the user to calibrate their trust. Although the developers could have put

much more transparency into the software, they were successful in calibrating the user's trust with the amount that they had put in. Since the users were able to calibrate their trust, there is little reason for the developers to spend the time and money necessary to develop more transparency features. The idea behind Q1.3 is that this type of insight would be discussed among the developers, and they would understand that their current design allows for trust calibration, answering Q1.2. This is not something that is explicit in the wording of Q1.3 but is certainly a conclusion developers could reach during the process of answering Q1.3.

Third, the analysis has shown that a lack of Coherent Use does not mean that a product is not useful. Although this insight is not surprising, it is important to point out and this use case is an excellent example. Even though the developers did not succeed in establishing human-AI coherency, they were able to deliver an impactful AI that improves operations.

Chapter 6: Conclusions

Within this thesis, I have presented a model of the HMI, defined a new mode of AI use entitled *Coherent Use*, and described a GQM template that aids developers in considering Coherent Use from the very start of the DevOps lifecycle. In addition, I have demonstrated how the GQM template may be used through the hypothetical ZDDA example and the real surveillance AI plugin example. This final chapter will discuss limitations, future work, and key takeaways.

6.1 Limitations and Future Work

The work presented in this thesis has a few limitations. First, due to time constraints and a lack of available software product, I was unable to fully apply the T&E approach with a development team over many DevOps iterations. To understand if this GQM breakdown is effective in establishing and motivating the maximization of Coherent Use, it would be necessary to follow a software product from initial design all the way to a mature AI that had undergone many DevOps iterations. Second, there are aspects of machine trustworthiness that are ignored for the simplicity of the GQM breakdown, namely security/privacy which is mentioned in a variety of references [57–59]. This was done because usually the DevOps lifecycle is performed in terms of DevSecOps which

already bakes security into the lifecycle. However, it would be useful to understand if there are questions related to security that may better analyze the trustworthiness of a software product. Third, Chapter 3 discusses five use cases that were used to create the HMI workflow model. To ensure that different types of AI would work within the model, a classification methodology was created. However, this was not a standardized classification or scoring based off pre-existing scales for stakes, machine perception, complexity, and autonomy.

Based on this work, there are a variety of improvements and future work that can be done on the GQM template and general consideration of Coherent Use. First, security was mentioned as a specific limitation when it comes to measuring and understanding trustworthiness. In the future it would be helpful to investigate how questions Q1.1, Q2.1, and Q3.1 could be reworded to include security considerations or if the inclusion of security requires the addition of questions to each of the GQM testing phases. Second, it would be useful to transition to a more standard classification and formal scoring for the five use cases described in Chapter 3, which again was described as a limitation of the work. Third, it would be interesting to understand how much AI transparency is enough to establish calibrated trust. Having some sort of understanding as to how many transparency considerations are necessary from design time would be extraordinarily helpful to developers so that they may ensure they are not wasting valuable time or money developing unnecessary transparency features. Finally, it would be interesting to understand if there are any scenarios or types of AI technologies in which the GQM template doesn't necessarily aid in user interactions. In the event that these technologies do exist, it would be important to analyze what makes the AI different from other AI

technologies and hopefully iterate on the GQM template to allow it to adapt to these scenarios.

6.2 Major Takeaways

Throughout this thesis, I have discussed the importance of establishing human-AI system coherency to allow for the optimal interaction between a user and AI. However, the development of an AI that can be coherently used must not stop developers from creating an AI in a timely manner. This motivated the integration of Coherent Use considerations throughout the DevOps lifecycle using the GQM approach. I have shown how this T&E methodology could be applied via the hypothetical ZDDA and the surveillance AI plugin use case in Chapter 5. The application of this methodology led to the following two important takeaways:

- Based on the surveillance AI plugin use case, developers should be able to easily integrate the GQM template within their pre-existing development practices.
- A lack of Coherent Use does not automatically mean that an AI-enabled technology is not useful. Although the AI plugin described in Chapter 5 did not establish Coherent Use with the operators, the product itself was still successful in improving operations.

Beyond the GQM template, this thesis has highlighted one overarching takeaway. Trust, trustworthiness, and trust calibration are difficult concepts to fully understand, model, measure, and account for in a human-AI system. Every user has characteristics unique to them. These ranging attributes provide complexity that extends beyond the

technical challenges associated with creating AI. As developers we must consider and pursue an understanding of these topics so we may create optimal human-AI systems, which is imperative when creating high-stakes AI-enabled technology. Understanding, modeling, measuring, and accounting for human-machine trust is very difficult, but as developers, we have a responsibility to take on this challenge to avoid devastating errors of misuse, disuse, and abuse.

Appendix A: Non-Developer-Controlled Trust Factors

While external and emerging factors that influence trust are outside of the control of the developer themselves, it is important to be aware of these influences. The following subsections go through each category and each factor that was understood to influence trust. Figure A.1 displays these factors together with the Developer-Controlled Trust Factors in an overarching diagram which also seeks to highlight the relationship between trust and reliance.

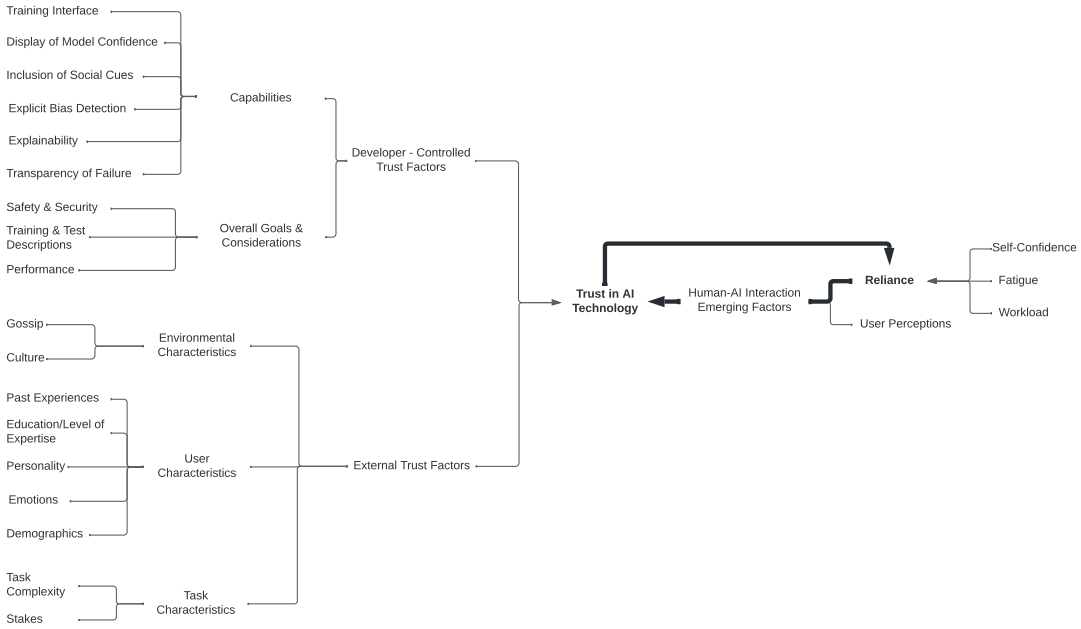


Figure A.1: Factors influencing trust and reliance in AI-enabled software.

A.1 External Trust Factors

External factors that influence trust refer to influences that exist outside of the software itself and rely on the environment that surrounds the technology. This includes environment characteristics, user characteristics, and task-related factors. Below is an overview of the factors that influence trust that exist externally from the technology broken down by the categories of environmental characteristics, user characteristics, and task characteristics.

A.1.1 Environmental Characteristics

In terms of this literature review, environmental characteristics are defined as trust influencing factors that exist in the environment that surrounds the user. Here, rather than speaking of the natural environment, the user environment is discussed as the factors that surround the user during the time in which they are utilizing the technology. This encompasses the time when the user is physically using the product as well as when they are taking a break from the technology and will return to using it in the future.

- **Gossip.** Gossip from the environment plays a large role in the trust that a user puts into a piece of technology. Gossip can be viewed in two distinct ways. First, there is gossip that arises from the media. This includes popular newspaper or magazine articles as well as other means of media communication with the general public. Secondly, gossip can be spread from peer to peer. This includes talk among co-workers, friends, bosses, etc. Within the literature, gossip was described

in both ways, however, more information and emphasis seemed to arise from the media form of gossip. Gossip in the media was described within [63] as they go over trust in AI systematically through the person, close environment, extended environment, and overall environment. Another paper, [64], sought to understand how the media was portraying the ethical issues associated with AI, thus also pointing out the importance that the media has on user trust. The authors found that the media surrounding AI had fairly neutral tones, with a balanced discussion of positives and negatives associated with AI. In addition, they found that the media was generally realistic and practical when it came to the ethics associated with AI [64]. Lastly, [21] brings up both the peer to peer and media aspects of gossip within their description of the formation of initial trust. This came in the form of the image/perception of AI (in terms of sci-fi movies and books) and reviews from other users.

- **Culture.** Another aspect of the environmental factors that influence trust is culture. This is discussed within [32]'s seventh Heuristic for Trusted Autonomy as a fit with users and operations and is also referred to within [23]. [23] explains a study that shows the general differences that exist among trust between Americans and Japanese. This highlights the notion that culture plays an important role in trust.

A.1.2 User Characteristics

User characteristics are factors that influence trust that exist internally to the user. Below each of these characteristics are described. Note that in this section [65] is cited

frequently as these authors provided significant useful information regarding users.

- **Past Experiences.** The past experience that a user has with technology can have a large impact on their trust. This encompasses the idea of learned trust discussed in [65], described as trust "based on past experiences relevant to a specific automated system." In this way, past experiences can be viewed as the way in which humans understand learned trust in a technology. Another way to view past experiences is with respect to general experiences an individual has within their lifetimes. This is briefly mentioned in [27] when describing Humane qualities that can affect an individual's risk-taking. Specifically mentioned were factors like personality, past experiences, and an individual's cultural background.

- **Education / Level of Expertise.** An individual's educational background and level of expertise plays an important role in their willingness to trust the technology. A study performed by [66] took university students (with at least some level of computer science knowledge) along with members of the general public and asked them to fill out a questionnaire regarding their perceptions of certain AI scenarios and technologies. From this survey, they were able to discover that some differences existed between the two groups depending on the AI application they are being asked to trust. More specifically, they found that university students were more likely to perceive a greater risk involved with AI than the general public. At the intersection of education and culture, [63] describes China's plan to integrate AI history and applications into primary and secondary education. The authors then point out that this has the potential to increase general trust in AI due to the early

exposure that individual's have with it. Finally, rather than speaking specifically about education, [65] mentions how subject matter experts of a technology tend to be less likely to trust the technology, again, confirming a strong relationship between education, expertise, and trust.

- **Personality.** The personality of an individual is yet another user characteristic that plays a big role in a person's trust of AI. In [65], the authors are led to the conclusion that "operators may be more likely to trust or rely upon automation when they are extraverted, emotionally stable, and have intuitive rather than sensing personalities." Supporting the previous author's statement regarding extraversion, [67] conducted a study with college students that measured the difference between introverted and extroverted individuals when it came to trusting an agent. They found that individuals with high extraversion had a greater amount of trust initially but later discovered that after the agent performed poorly (i.e., was highly confident in an incorrect response) the trust the individual had in the technology decreased dramatically.

- **Emotions.** Supporting the notion that an individual's emotion influences the trust that they place in a piece of technology, [65] discuss emotion as it relates to the formation of trust. Citing information from [23], the author's explain that emotions play a large role in the trusting behavior of an individual. The author's also describe two studies that looked at mood as it relates to trust in technology. In addition, [68] performed a study that sought to understand the relationship between trust and emotion within the context of drone operation. Among other findings, the author's

found that their results suggest an individual's emotional intensity may indicate that individual's level of trust. In addition, the author's suggest that trust may increase when an individual has a positive emotional experience.

- **Demographics.** The demographics of a user surrounds many user characteristics including age and gender identity. With respect to gender identity, both [69] and [65] discuss its relationship with trust in technology. In addition to their take on gender identity, [65] also discusses age, another element of an individual's demographics. They come to the conclusion that age plays an important role in how to analyze the trustworthiness of a piece of technology but the overall outcome (whether or not the technology is viewed as trustworthy) is context-dependent.

A.1.3 Task Characteristics

Task characteristics are elements that impact trust that are specific to the nature of the task being performed by the technology. This includes the overall difficulty of the task as well as the stakes associated with the task. These factors are described below:

- **Task Complexity.** The overall complexity of a task has been shown to contribute to an individual's overall trust in a technology. [65] describes how operators use the complexity of the task that the technology is performing as a way to evaluate the capabilities of the technology as a whole. [27] also mentions task complexity as an environmental quality, saying that the "sophistication of the task has a potential to attract trustworthiness or cause distraction." This affirms the importance of task complexity as it relates to trust in technology.

- **Stakes.** Inherent within the concept of trust is the notion of risk. Many technology solutions deal with high and low stakes scenarios that can range anywhere from a self-driving car (that deals with a life threatening task) and an advertisement recommendation system (which is seemingly harmless upon failure). Within their study, [24] found that participants were less willing to place trust in technologies that were dealing with high stakes scenarios. As one participant put it, "'I simply don't trust any system that has input into important decisions'" [24]. This attitude toward an AI shows that the stakes of a task can completely change whether or not a user is willing to place trust in that AI.

A.2 Emerging Factors

Emerging factors refer to the influences on trust that emerge from the human-machine interaction. This includes user perceptions of the technology and the overall reliance on the machine from the user. Below is an overview of each of these factors.

A.2.1 User Perceptions

The overarching user perceptions category encompasses all of the perceptions that the user has regarding the technology. This stems from environmental characteristics (gossip and culture) and emerges further from the continued interactions the user has with the device. [21] discusses one example of a user perception. Specifically they describe the idea that users could perceive AI as a technology that will replace them in the workforce. This job replacement has the potential to cause concern within the user

and thus lead to distrust in the technology. Another user perception exists within the idea of system provider benevolence. This concept is briefly discussed by [63] in the context of companies choosing to put out ethical mission statements and creating a positive brand image. While these two brief examples merely scratch the surface of user perceptions, it is clear that these user views have a significant impact on the trust that a user puts in a technology.

A.2.2 Reliance and Its Relationship with Trust

A user's reliance on technology also contributes as a trust factor. Yet, trust and reliance have an interesting relationship that defines how the two interact with each other. As described by [23], reliance is a factor that influences trust because it allows the user to gain more experience and understanding of the capabilities of the technology. However, trust also informs reliance [23]. This creates a closed loop cycle of trust informing reliance and in turn reliance informing trust.

Just as trust was influenced by various factors, the decision of the user to rely on a piece of technology is also informed by different factors. These factors include user fatigue, user workload, self-confidence, and, of course, trust. Below is a brief overview of each factor (omitting trust itself):

- **Self-Confidence.** A user's self-confidence has been found to have a significant impact on their reliance on a machine. [70] explains these effects in relation to automation. They explain that, in general, studies have shown that people tend to be overconfident in their abilities. In addition, the authors performed an experiment

for the purposes of understanding the relationship between trust, self-confidence, and reliance on automation. Ultimately, they found that, in general, operators relied on manual control when their own self-confidence exceeded their trust in the automation. Conversely, they found that when trust in the automation exceeded their own self-confidence, they relied more heavily on the technology.

- **Fatigue.** User fatigue refers to the tiredness of an operator while they are using the technology. User fatigue was specifically examined within a study done by [71] where the effects of sleep loss were investigated in relation to automated decision aids. The authors found that when tired, participants used the decision aid more carefully and gave more effort to verifying the machine. They also found that participants were less likely to fall victim to automation bias when experiencing an increased level of sleepiness. Another automation study performed by [72] found that users had the tendency to disuse the technology as their fatigue increased. While this did not support the author's original hypothesis, it does pair well with [71]'s previous finding regarding automation bias. In this way, it could be suggested that as user fatigue increases, a machine is relied upon less and more cognitive effort is used to combat the risk taking behavior involved with performing tasks while in a state of fatigue.

- **Workload.** Workload refers to the task load associated with a user during the time the user is expected to use the technology. When it comes to workload's effect on reliance, [53] has found mixed results. The authors state that many times the initial reasoning for bringing about automation is to decrease a users high workload,

however, they found that this does not always occur. They cite two different studies, one finding no relationship between workload and automation usage and another showing a trend toward using automation under conditions of high workload, thus the relationship between workload and reliance remains unclear. However, the authors do note that when asked, operators often use workload as an explanation as to why they choose to use automation. In addition to [53], [23] briefly mention workload as a factor that helps to form a users intention to rely on automation, however, the authors also mention an additional factor that is seemingly related to workload, effort to engage. Effort to engage is very similar to the notion of cognitive overhead described by [53] and has the potential to be viewed as a part of workload. This is because workload has a significant influence on whether or not the effort to engage automation is too high to perform even if the intention to use the automation is there.

Bibliography

[1] Paras Malik Amisha, Monika Pathania, and Vyas Kumar Rathaur. Overview of artificial intelligence in medicine. *Journal of family medicine and primary care*, 8(7):2328, 2019.

[2] Anahad O'Connor. How Artificial Intelligence Could Transform Medicine. *The New York Times*, March 2019.

[3] Pfizer. How a Novel 'Incubation Sandbox' Helped Speed Up Data Analysis in Pfizer's COVID-19 Vaccine Trial | Pfizer. https://www.pfizer.com/news/hot-topics/how_a_novel_incubation_sandbox_helped_speed_up_data_analysis_in_pfizer_s_covid_19_vaccine_trial. Accessed: 2022-03-23.

[4] Sam Ransbotham and Shervin Khodabandeh. AI and the COVID-19 Vaccine: Moderna's Dave Johnson. Me, Myself, and AI, July 2021.

[5] Alvin Powell. Risks and benefits of an AI revolution in medicine. *Harvard Gazette*, November 2020. Section: Health & Medicine.

[6] David Vergun. Artificial Intelligence Key to Maintaining Military, Economic Advantages, Leaders Say. https://www.defense.gov/News/News-Stories/Article/Article/2567486/artificial-intelligence-key-to-maintaining-military-economic-advantages-leaders/, April 2021. Accessed: 2021-12-11.

[7] Michèle A. Flournoy, Avril Haines, and Gabrielle Chefitz. Building trust through testing. https://cset.georgetown.edu/event/building-trust-through-testing/, 2020. Accessed: 2021-12-11.

[8] Software Engineering Institute. 2020 sei year in review. Technical report, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2021.

[9] Tom Krisher. 3 crashes, 3 deaths raise questions about Tesla's Autopilot. *AP NEWS*, January 2021.

[10] Elon Musk. If you want the Tesla Full Self-Driving Beta downloaded to your car, let us know. Doubling beta program size now with 8.2 & probably 10X size with 8.3. Still be careful, but it's getting mature., March 2021.

[11] Faiz Siddiqui. Tesla owners can now request 'Full Self-Driving,' prompting criticism from regulators and safety advocates. *Washington Post*, September 2021.

[12] Cade Metz and Neal Boudette. Inside Tesla: How Elon Musk Pushed His Vision for Autopilot. *The New York Times*, December 2021.

[13] Anderson Cooper. Police departments adopting facial recognition tech amid allegations of wrongful arrests. https://www.cbsnews.com/news/facial-recognition-60-minutes-2021-05-16/, May 2021. Accessed: 2021-10-26.

[14] John General and Jon Sarlin. A false facial recognition match sent this innocent black man to jail. *CNN Business*, April 2021.

[15] Kashmir Hill. Wrongfully Accused by an Algorithm. *The New York Times*, June 2020.

[16] Karen Weise. Amazon indefinitely extends a moratorium on the police use of its facial recognition software. *The New York Times*, May 2021.

[17] Matthew Michael Carnahan, Drew Goddard, and Damon Lindelof. World War Z, 2013. Screenplay.

[18] Adetokunbo A.A. Adenowo and Basirat A. Adenowo. Software engineering methodologies: A review of the waterfall model and object-oriented approach. *International Journal of Scientific & Engineering Research*, 4(7):427–434, 2013.

[19] Onur Asan, Alparslan Emrah Bayrak, Avishek Choudhury, et al. Artificial intelligence and human trust in healthcare: focus on clinicians. *Journal of medical Internet research*, 22(6):e15154, 2020.

[20] Matthew Arnold, Rachel KE Bellamy, Michael Hind, Stephanie Houde, Sameep Mehta, Aleksandra Mojsilović, Ravi Nair, K Natesan Ramamurthy, Alexandra Olteanu, David Piorkowski, et al. Factsheets: Increasing trust in ai services through supplier's declarations of conformity. *IBM Journal of Research and Development*, 63(4/5):6–1, 2019.

[21] Keng Siau and Weiyu Wang. Building trust in artificial intelligence, machine learning, and robotics. *Cutter business technology journal*, 31(2):47–53, 2018.

[22] Peter A Hancock, Deborah R Billings, and Kristen E Schaefer. Can you trust your robot? *Ergonomics in Design*, 19(3):24–29, 2011.

[23] John D Lee and KA See. Trust in technology: Designing for appropriate reliance. *Human Factors*, 46(1):50–80, 2004.

[24] Maryam Ashoori and Justin D Weisz. In ai we trust? factors that influence trustworthiness of ai-infused decision-making processes. *arXiv preprint arXiv:1912.02675*, 2019.

[25] Stavros Antifakos, Nicky Kern, Bernt Schiele, and Adrian Schwaninger. Towards improving trust in context-aware systems by displaying system confidence. In *Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, pages 9–14, 2005.

[26] Ella Glikson and Anita Williams Woolley. Human trust in artificial intelligence: Review of empirical research. *Academy of Management Annals*, 14(2):627–660, 2020.

[27] Ehsan Toreini, Mhairi Aitken, Kovila Coopamootoo, Karen Elliott, Carlos Gonzalez Zelaya, and Aad Van Moorsel. The relationship between trust in ai and trustworthy machine learning technologies. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 272–283, 2020.

[28] Francesca Rossi. Building trust in artificial intelligence. *Journal of international affairs*, 72(1):127–134, 2018.

[29] David Gunning and David Aha. Darpa's explainable artificial intelligence (xai) program. *AI Magazine*, 40(2):44–58, 2019.

[30] Kazuo Okamura and Seiji Yamada. Adaptive trust calibration for human-ai collaboration. *Plos one*, 15(2):e0229132, 2020.

[31] Margarita Konaev, Tina Huang, and Husanjot Chahal. Trusted partners: Human-machine teaming and the future of military ai. *Center for Security and Emerging Technology*, pages 18–23, 2021.

[32] Kimberly F Jackson, Zahar Prasov, Emily C Vincent, and Eric M Jones. A heuristic based framework for improving design of unmanned systems by quantifying and assessing operator trust. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 60, pages 1696–1700. SAGE Publications Sage CA: Los Angeles, CA, 2016.

[33] Christof Ebert, Gorka Gallardo, Josune Hernantes, and Nicolas Serrano. Devops. *Ieee Software*, 33(3):94–100, 2016.

[34] Håvard Myrbakken and Ricardo Colomo-Palacios. Devsecops: a multivocal literature review. In *International Conference on Software Process Improvement and Capability Determination*, pages 17–29. Springer, 2017.

[35] Víctor Mayoral-Vilches, Nuria García-Maestro, McKenna Towers, and Endika Gil-Uriarte. Devsecops in robotics. *arXiv preprint arXiv:2003.10402*, 2020.

[36] Ahmad Alnafessah, Alim Ul Gias, Runan Wang, Lulai Zhu, Giuliano Casale, and Antonio Filieri. Quality-aware devops research: Where do we stand? *IEEE Access*, 9:44476–44489, 2021.

[37] Victor R Basili, Gianluigi Caldiera, and H Dieter Rombach. The goal question metric paradigm. *Encyclopedia of Software Engineering*, pages 528–532, 1994.

[38] Heiko Koziolek. Goal, Question, Metric. In Irene Eusgeld, Felix C. Freiling, and Ralf Reussner, editors, *Dependability Metrics: Advanced Lectures*, Lecture Notes in Computer Science, pages 39–42. Springer, Berlin, Heidelberg, 2008.

[39] Patrik Berander and Per Jönsson. A goal question metric based approach for efficient measurement framework definition. In *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*, pages 316–325, 2006.

[40] Armin Beer and Michael Felderer. Measuring and improving testability of system requirements in an industrial context by applying the goal question metric approach. In *Proceedings of the 5th International Workshop on Requirements Engineering and Testing*, RET '18, pages 25–32, New York, NY, USA, June 2018. Association for Computing Machinery.

[41] Rini van Solingen and Egon W Berghout. *The Goal/Question/Metric Method: a practical guide for quality improvement of software development*. McGraw-Hill, 1999.

[42] Alfonso Fuggetta, Luigi Lavazza, Sandro Morasca, Stefano Cinti, Giandomenico Oldano, and Elena Orazi. Applying gqm in an industrial software factory. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 7(4):411–448, 1998.

[43] Jamaiah H Yahaya, Zaiha Nadiah Zainal Abidin, Noorazean Mohd Ali, and Aziz Deraman. Software ageing measurement and classification using goal question metric (gqm) approach. In *2013 Science and Information Conference*, pages 160–165. IEEE, 2013.

[44] Heather M Wojton, Daniel Porter, Stephanie T. Lane, Chad Bieber, and Poornima Madhavan. Initial validation of the trust of automated systems test (toast). *The Journal of Social Psychology*, 160(6):735–750, 2020.

[45] Maria Madsen and Shirley Gregor. Measuring human-computer trust. In *11th australasian conference on information systems*, volume 53, pages 6–8. Citeseer, 2000.

[46] Tesla. Autopilot. https://www.tesla.com/autopilot. Accessed: 2022-03-23.

[47] Tesla. Model 3 Owner's Manual. https://www.tesla.com/ownersmanual/model3/en_us/GUID-A701F7DC-875C-4491-BC84-605A77EA152C.html, 2022. Accessed: 2022-03-23.

[48] P.W. Singer and August Cole. *Burn-In: A Novel of the Real Robotic Revolution*. HarperCollins Publishers, 2020.

[49] iRobot. Roomba® Robot Vacuum Cleaners | iRobot®. https://www.irobot.com/roomba. Accessed: 2022-03-25.

[50] James Vincent. iRobot is giving its vacuum cleaners a new AI-powered brain. *The Verge*, August 2020.

[51] Ring. Video Doorbells | Smart Doorbell Cameras to Monitor Your Door. https://ring.com/doorbell-cameras. Accessed: 2022-03-23.

[52] Leonard Polizzotto and Arthur Molella. The value balance. *IEEE Engineering Management Review*, 47(4):24–31, 2019.

[53] Raja Parasuraman and Victor Riley. Humans and automation: Use, misuse, disuse, abuse. *Human factors*, 39(2):230–253, 1997.

[54] Kakao AI Report. Meta-analysis on 6,163 papers of ICML&NIPS. https://medium.com/@kakaoaireport/meta-analysis-on-6-163-papers-of-icml-nips-cbef530eaaf6, September 2017. Accessed: 2021-12-09.

[55] Richard Tomsett, Alun Preece, Dave Braines, Federico Cerutti, Supriyo Chakraborty, Mani Srivastava, Gavin Pearson, and Lance Kaplan. Rapid trust calibration through interpretable and uncertainty-aware ai. *Patterns*, 1(4):100049, 2020.

[56] Scott Thiebes, Sebastian Lins, and Ali Sunyaev. Trustworthy artificial intelligence. *Electronic Markets*, 31(2):447–464, 2021.

[57] Sonali Jain, Manan Luthra, Shagun Sharma, and Mehtab Fatima. Trustworthiness of Artificial Intelligence. In *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 907–912, March 2020.

[58] Kush Varshney. Foundations of trustworthy AI: How to conduct trustworthy AI assessment and mitigation. https://www.ibm.com/blogs/watson/2021/06/trustworthy-ai-assessment-mitigation/, June 2021. Accessed: 2021-12-11.

[59] Haochen Liu, Yiqi Wang, Wenqi Fan, Xiaorui Liu, Yaxin Li, Shaili Jain, Yunhao Liu, Anil K Jain, and Jiliang Tang. Trustworthy ai: A computational perspective. *arXiv preprint arXiv:2107.06641*, 2021.

[60] Olivier L. de Weck, Adam M. Ross, and Donna H. Rhodes. Investigating relationships and semantic sets amongst system lifecycle properties (ilities). 2012.

[61] Michael W. Boyce, Jessie Y.C. Chen, Anthony R. Selkowitz, and Shan G. Lakhmani. Effects of agent transparency on operator trust. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*

Extended Abstracts, HRI'15 Extended Abstracts, pages 179–180, New York, NY, USA, 2015. Association for Computing Machinery.

[62] Maurice Dawson, Darrell Norman Burrell, Emad Rahim, and Stephen Brewster. Integrating software assurance into the software development life cycle (sdlc). *Journal of Information Systems Technology and Planning*, 3:49–53, 01 2010.

[63] Marisa Tschopp and Marc Ruef. On trust in ai—a systemic approach, 2018.

[64] Leila Ouchchy, Allen Coin, and Veljko Dubljević. Ai in the headlines: the portrayal of the ethical issues of artificial intelligence in the media. *AI & SOCIETY*, 35(4):927–936, 2020.

[65] Kevin Anthony Hoff and Masooda Bashir. Trust in automation: Integrating empirical evidence on factors that influence trust. *Human factors*, 57(3):407–434, 2015.

[66] Keeley Crockett, Matt Garratt, Annabel Latham, Edwin Colyer, and Sean Goltz. Risk and trust perceptions of the public of artifical intelligence applications. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[67] JS Elson, Douglas Derrick, and Gina Ligon. Examining trust and reliance in collaborations between humans and automated agents. In *Proceedings of the 51st Hawaii International Conference on System Sciences*, 2018.

[68] Theodore Jensen, Mohammad Khan, Yusuf Albayram, Md Abdullah Al Fahim, Ross Buck, and Emil Coman. Anticipated emotions in initial trust evaluations of a drone system based on performance and process information. *International Journal of Human-Computer Interaction*, 36:1–10, 07 2019.

[69] Alex Wong, Anqi Xu, and Gregory Dudek. Investigating trust factors in human-robot shared control: Implicit gender bias around robot voice. In *2019 16th Conference on Computer and Robot Vision (CRV)*, pages 195–200. IEEE, 2019.

[70] John D Lee and Neville Moray. Trust, self-confidence, and operators' adaptation to automation. *International journal of human-computer studies*, 40(1):153–184, 1994.

[71] Juliane Reichenbach, Linda Onnasch, and Dietrich Manzey. Human performance consequences of automated decision aids in states of sleep loss. *Human factors*, 53(6):717–728, 2011.

[72] Ryan W Wohleber, Gloria L Calhoun, Gregory J Funke, Heath Ruff, C-Y Peter Chiu, Jinchao Lin, and Gerald Matthews. The impact of automation reliability and operator fatigue on performance and reliance. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 60, pages 211–215. SAGE Publications Sage CA: Los Angeles, CA, 2016.