The “Oracle Problem” and Its Impacts in Mixed Reality

Sarah M. Lehman, Ph.D.
NASA Langley Research Center (LaRC)

Agenda

- **Setting the Stage**
  - Quick case study
  - Glimpse into the mind (and worries) of a software engineer
  - The case for cooperation

- **The Oracle Problem**
  - What is it
  - Why is it important
  - What does it mean for ML-based systems (generally) and XR systems (specifically)

- **A Vision for The Future**
  - Incorporating oracles into ethics and policy discussions

SETTING THE STAGE

A Quick Case Study

- **PREMISE**: You're developing policy surrounding the OWNERSHIP of virtual objects in augmented reality systems

- **SYSTEM**: Virtual "companion" application which provides emotional support pets for the elderly, homebound, and other isolated communities

- **ISSUE**: One user's pet "attacks" another (or another user)
  - We define "attack" loosely; it could compromise another virtual pet by (un)intentionally exploiting vulnerabilities in the code or discomfort another user by startling them, triggering a phobia, etc.

- **QUESTIONS**:
  - Who "owns" the responsibility for that pet? The owner? The application provider?
  - How do you regulate application and user behaviors so that this doesn't happen?
  - ***How do you translate regulation into TESTABLE software requirements?***

I’m going to let you in on a little secret…

SOFTWARE

ENGINEERS

HATE

AMBIGUITY

How do I know this???

- **First-hand experience**
  - Software engineer for Lockheed Martin before going to grad school
  - Contracted out to NAVAIR to digitize maintenance procedures
  - Easiest gig ever; all protocols were well-established, well-documented, and had very clear metrics for success

- **A very vocal, opinionated partner**
  - Software engineer and architect for more than 12 years
  - Leads multiple teams, struggles DAILY with getting actionable guidance out of the business and design teams
  - Complains MY EAR OFF on dog walks

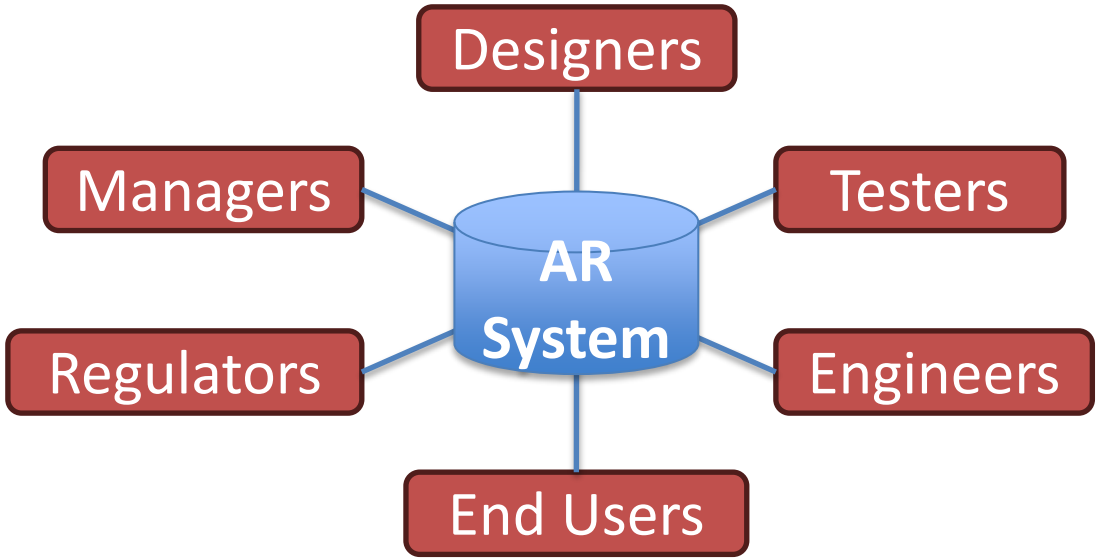
Image Source:
https://commons.wikimedia.org/wiki/File:Fixed_-_Flickr_-_Don_Fulano.jpg

Take Pity on Your Software Engineers!!

This group has spent a lot of time talking about the different philosophical implications of AR systems, but not enough on how we're going to turn those observations into REALITY.

- Creating complex software systems requires many different actors with many different areas of expertise

- These interdependencies create a range of pain points, but particularly around:
  - Communicating requirements
  - Proving requirements compliance



Take Pity on Your Software Engineers!!

Turning philosophical principles (fairness, justice, etc.) into actionable system properties will require much more than just the principles themselves!

- Concrete descriptions of desired system behavior
  - Requirements
  - Specifications
- Detailed guidance on how to measure and test that behavior
  - Testing Frameworks
  - Metrics for Success
  - Oracles
- Collecting system and user feedback to verify that behavior
  - Data Traces
  - In-Lab User Studies
  - In-Situ User Studies

Returning to our case study…

Philosophical Principle

AR systems should respect ownership of virtual objects.

Clarifying Questions

How to handle conflicts between ownership and free speech?

How to protect vulnerable populations from exploitation?

How to differentiate corporate ownership from personal?

Engineering Guidance

Requirements

Tests

Metrics

Making the Case for Cooperation

You don’t have to solve EVERY problem, but being aware of what’s involved will start to bridge the gap!

Image Source: https://commons.wikimedia.org/wiki/File:Dore_canyon.jpg

Not Pictured For Simplicity: legislators, business leaders, and other stakeholders

"Alright, you've convinced me. So, what software engineering aspects CAN I help with?"

THE ORACLE PROBLEM

What is an Oracle?

- **The Oracle of Delphi**
  - Colloquial name for Pythia, high priestess of the Temple of Apollo in Delphi, Greece
  - Delivered prophecies and wisdom using line of direct communication with Apollo
  - Famously difficult to interpret

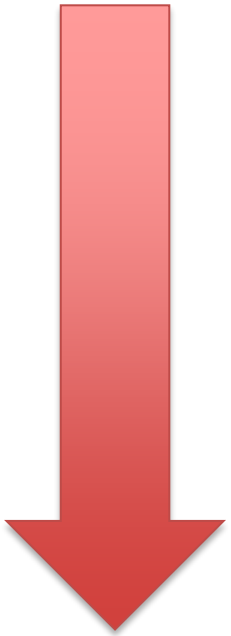
- **Test Oracles**
  - Mechanism for determining whether system output is correct for a given set of inputs
  - Depends on the current system state; the same input / output pair may be correct in one context but wrong in another

Image Source: https://commons.wikimedia.org/wiki/File:Oracle_of_Delphi,_red-figure_kylix,_440-430_BC,_Kodros_Painter,_Berlin_F_2538,_141668.jpg

What is the Oracle Problem?

THE ORACLE PROBLEM:
how to develop, apply, and verify (preferably automated) test oracles for increasingly complex systems with increasingly difficult-to-specify behavior

In order of preference…



- **Specified oracles**
  - Developed from formal specifications (i.e., math)
- **Derived oracles**
  - Developed from system artifacts, such as documentation or prior versions
- **Implicit oracles**
  - Developed to detect "obvious" issues, such as system crashes
- **Human oracles**
  - When all else fails, have a human do it

IT ALL BOILS DOWN TO:

"How do I know if the system is *doing the right thing at the right time*?"

Impacts to ML-based Systems

Machine learning-based systems are particularly sensitive to the Oracle Problem!

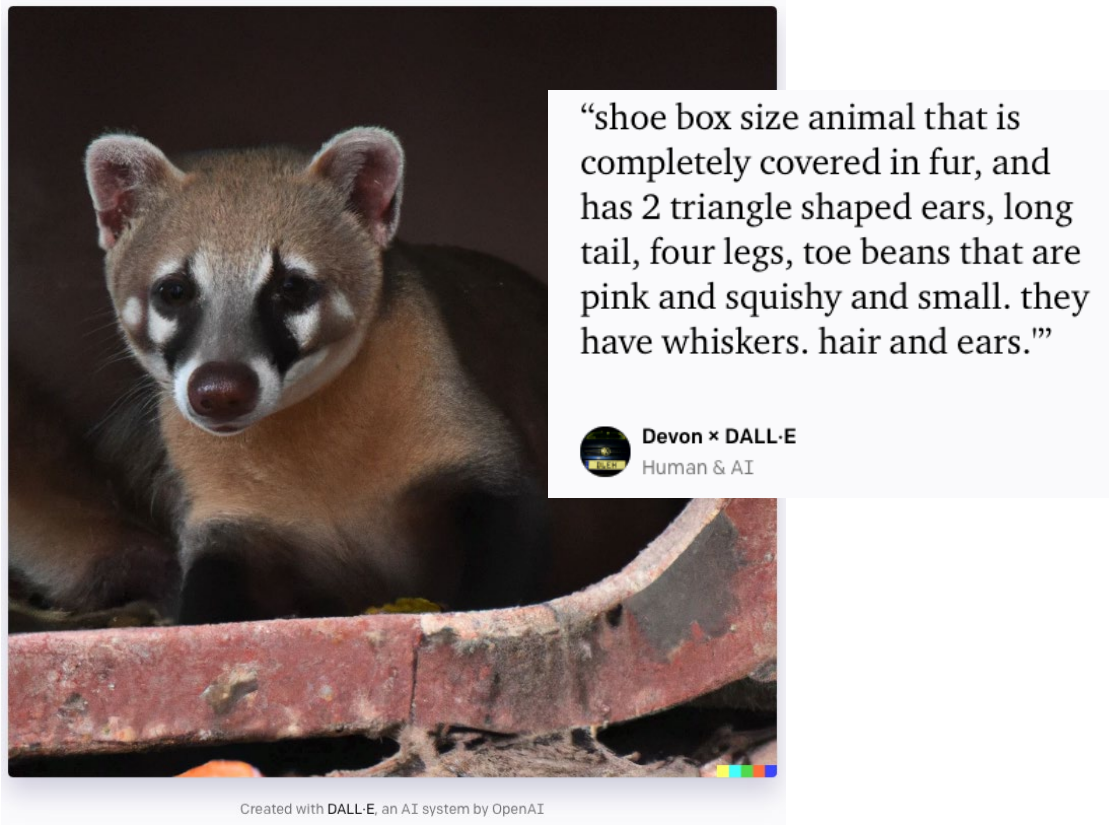
- **ML-based systems are designed to provide answers to problems that humans are incapable of solving manually, where no formal specification exists**
  - "Traditional" software is ***code driven*** (i.e., inherits behavior from human-written code)
  - ML-based software is ***data driven*** (i.e., inherits behavior by iteratively replicating patterns in large bodies of data)

- **In other words, our preferred oracle type ("specified") is no good here!**
  - Remember that specified oracles are ones that can be expressed / proven mathematically
  - If humans could formally specify their preferred solution in the first place, they wouldn't need machine learning

Impacts to ML-based Systems

Machine learning-based systems are particularly sensitive to the Oracle Problem!

- **What about the other automated oracle types?**
- **<u>Derived oracles</u> still require concrete descriptions**
  - Now it's "just words" instead of math, but it's still hard!
  - Ever asked an AI to draw you a cat without saying the words "cat" or "feline"?
- **<u>Implicit oracles</u> are simpler, but retroactive**
  - Easy to wait for a system crash… Not helpful if you're trying to AVOID crashes.



<u>Image Source</u>: DALL-E, my goofball husband, and his friends

Impacts to ML-based Systems

Machine learning-based systems are particularly sensitive to the Oracle Problem!

- **Why not "brute force it" (i.e., create an explicit list of good input-output pairs)?**
- **Might seem appealing for ML-based systems that rely on images or audio**
  - Especially since this is the approach used in the training phase of supervised learning…
- **Such domains represent a HUGE array of possible inputs due to environmental factors (impossible to comprehensively enumerate):**
  - Image: lighting, viewing angle, color and texture, camera occlusions, movement speed, etc.
  - Audio: background noise, speaking speed, accent, word choice, etc.

(a) Classifier error generates wrong warning text

(b) Slow processing, quick user movement prevents classification

(c) Poor label placement covers pertinent real-world content (car in intersection)

(d) Alert correctly placed but difficult to read due to poor color choice

Image Source: Lehman, Sarah M., Haibin Ling, and Chiu C. Tan. "Archie: A user-focused framework for testing augmented reality applications in the wild." *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 2020.

Impacts to Mixed Reality Systems

Mixed reality, as a specific application of machine learning, makes things even harder!

- **Now you must worry, not only about the correctness of the underlying ML model, but also about amorphous, equally hard-to-define human factors concerns**
  - Whether your system is **usable** by humans (cybersickness, UI design, general performance, etc.)
  - Whether your system is **useful** to humans (cognitive burden, task completion, burden of new technology / modalities compared to existing methods, etc.)

- **Whatever policies you decide on must extend across the spectrum of human life**
  - Users will take their XR systems with them (phones, tablets, wearables)
  - Does your policy work outside in a park? Inside in the supermarket? In sensitive spaces such as the bathroom or doctor's office? When there are innocent bystanders? When the system is multi-user?

Why should we care?

- **As a group of philosophical / legal experts and beyond, the concerns of this working group are even larger and even MORE difficult to define:**

Does this system harbor (un)conscious bias that be harmful to users?

Does this system respect and promote users' sense of self and autonomy in self-expression?

How does this system impact users' perceptions of the world and the existence of real or virtual people and objects?

Why should we care?

- **Therefore, it is PARAMOUNT that, as we are having these conversations, we are ALSO discussing how to translate these incredibly important concepts into oracles that are:**

Understandable:
Plainly stated with limited jargon

Justifiable:
Have clear rationale for inclusion

Measurable / Testable:
Produce concrete, verifiable outcomes

Traceable:
Have clear "lineages" back to primary sources

Some Clarifications Before We Continue

Are you asking me to change careers?

No! (No matter how much I love being a software engineer)

Are you asking me to do software engineers' work for them?

Absolutely not! (Though I *AM* asking you to make things a little easier on them…)

Are you asking me to take a system-level view when developing policy?

Yes! Awareness of the implementation process will facilitate better policy creation.

So, what can I actually **do** about this?

A SYSTEM-LEVEL VISION FOR THE FUTURE

Suggestions for Bridging the Gap

<u>Suggestion #1</u>:
Talk to more software and systems engineers.

- **Learn more about how engineers elicit requirements from their stakeholders**
- **Learn more about how engineers design their systems to be testable**
- **Identify implementation challenges that you didn't know you didn't know**, such as:
  - "Latency must be below ___ to avoid making users motion sick."
  - "___ hardware platform is too expensive to be practical."

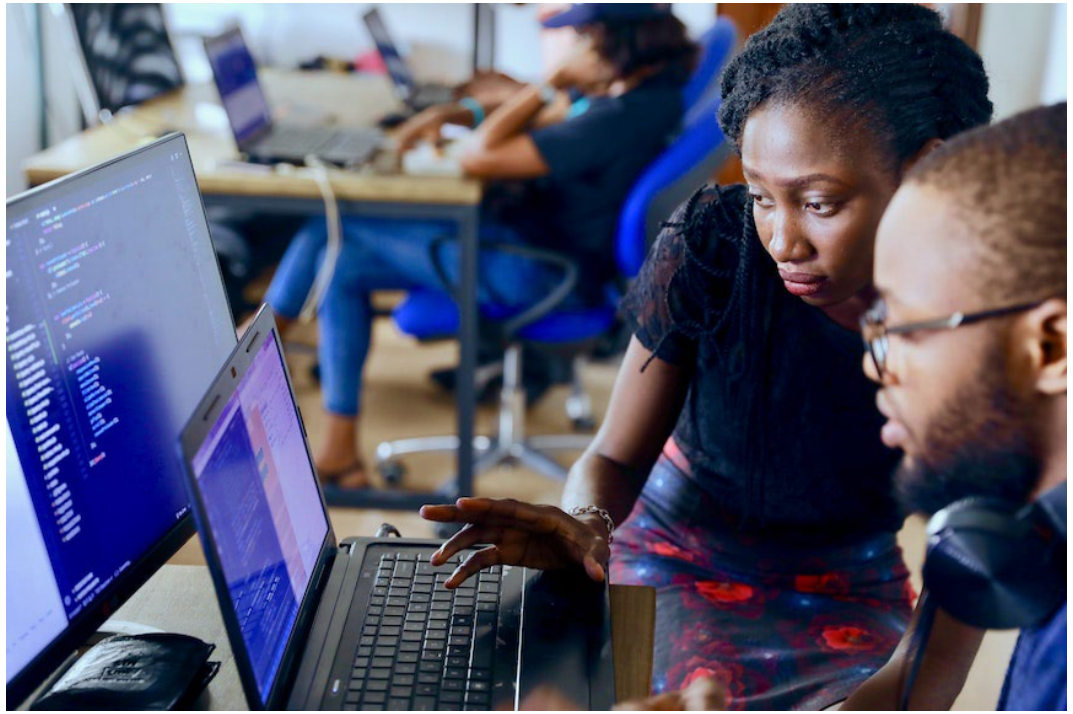


Image Source: https://commons.wikimedia.org/wiki/File:Programmers.jpg

Suggestions for Bridging the Gap

Suggestion #2:
Investigate domain-specific oracle strategies.

- **Software Oracles**
  - Metamorphic relations (relationships between inputs and outputs)
  - Assertions and contracts (hard-coded conditions that can be checked at runtime)

- **Human Oracles**
  - New questionnaires and surveys
  - Biometric monitors

REMEMBER! The goal is NOT to develop the tests yourself, but rather to design policy and guidance that makes the testing process EASIER, not harder.

Image Source:
https://commons.wikimedia.org/wiki/File:Brown_on_aft_flight_deck_with_microphone_during_STS-95.jpg

Suggestions for Bridging the Gap

Suggestion #3:
Talk to other regulatory agencies for inspiration and ideas.

- **The transportation industry is a good start**
  - Aircraft, automobiles need strict certification guidelines to maintain safety
  - The FAA is currently exploring strategies for approval, certification of ML-based components

- **The Overarching Properties**
  - Example of how the FAA has addressed emergent properties that can't necessarily be "tested" in the traditional sense

bit.ly/arg83

INTENT:	The *defined intended behavior* is correct and complete with respect to the *desired behavior*.
CORRECTNESS:	The *implementation* is correct with respect to its *defined intended behavior*, under *foreseeable operating conditions*.
INNOCUITY:	Any part of the *implementation* that is not required by the *defined intended behavior* has no *unacceptable impact*.

Thank you!
Any questions?

Sarah M. Lehman
sarah.lehman@nasa.gov