Origins of RE for AI: A Hot Topic as Viewed by an RE Alter Kaker

Daniel M. Berry
University of Waterloo

Abstract

I explore how not-very-successful attempts to fit requirements for artificial intelligences (AIs) and learned machines (LMs) into the traditional RE mold led to a rethinking about RE for AI. I talk about some implications of the rethinking.

Vocabulary

AI	**artificial intelligence**
ML	**machine learning**
LM	**learned machine**

These slides use “AI” to mean “AI or LM” to save slide space and speaker’s breath.

David Parnas’s Concerns

Back in 2019, I had seen slides from Dave Parnas expressing very negative concerns about AI in his inimitable way.

Parnas's Slides

Slides 1 and 3–6 from talk titled “My Concerns About Artificial Intelligence” by David Lorge Parnas of Middle Road Software, Inc.

You can see see the sarcasm dripping from the first few substantive slides.

MY Concerns About Artificial Intelligence

Abstract

There is growing alarm about the potential dangers of Artificial Intelligence (AI). Giants of the commercial and scientific world have expressed the concern that AI will eventually make people superfluous. A group of activists has argued that AI should not be used in weapon systems, explaining their fear that killer robots might start to fight wars against people and without morality. A Microsoft researcher recently made headlines saying, “As artificial intelligence becomes more powerful, people need to make sure it is not used by authoritarian regimes to centralize power and target certain populations.”

My concerns are very different. I believe that the use of AI techniques will make software intensive systems even less trustworthy than they are now. I see a lot of unscientific thinking and a lack of engineering discipline in AI work.

This talk explains my views. It is designed to stimulate discussion.

Two Instructive Jokes

What's the difference between a horse-breeder and a Computer Scientist?

- When the horse breeder talks about "AI", he can explain what is meant.

What's the difference between a cattle-farmer and an AI fan?

- A farmer knows BS when she sees it.

A commandment for scientists:

Thou shalt not use terms that you cannot define.

- "Intelligence" is one of those terms.
- "Artificial Intelligence" is worse.

Definitions in science must be based on measurement

- Define terms before you use them in a talk or paper.

Three Types of AI Research

- Building programs that imitate human behaviour to understand human thinking better (psychology research)
- Building programs that play games well (challenging and fun)
- Demonstrating that practical computerized products can use the same methods that humans use (risky and often naive).

It is only the third that concerns me.

Responsibilities of Engineers

- *Do no harm!*
- *Make products that are fit for their intended use*
- *Make products that can be trusted*
- *Make sustainable/maintainable products (design for change).*

I doubt that AI meets these responsibilities.

Commandment:

Thou shalt always remember that when building a device for others to use, it is your responsibility

- ***to know when it is trustworthy/not trustworthy,***
- ***to tell the users the circumstances in which it cannot be trusted.***

Surprises are not a "feature".

What Does “AI” Really Mean?

One of the following:

1. **Automation - when arguing that AI is useful.**
2. **A program that appears intelligent — illusionists[1].**
3. **A program that uses methods intended to mimic the thinking methods of people (hopefully intelligent ones)**
4. **A program that uses heuristics[1] (doesn’t always work — also like people).**
5. **A lazy way to solve a problem. Don’t bother to analyze the problem, just let the program learn[1] on the job.**

Commandment: ***Thou shalt say what you mean*.**

[1] To be discussed later.

Unpredictable Behaviors of AIs

He shows several examples of very bad side effects arising from the unpredictable behavior of AIs.

I won’t show all of his slides.

Dave Parnas’s Letter to CACM

Instead I will quote a letter that Dave wrote to the editor of *CACM* that appears in *CACM* August 2019 (62:8) on Page 9:

“[The dangers are] not limited to neural networks or machine learning technology. [These] dangers … exist whenever a program’s precise behavior is not known to its developers. …

Dave Parnas's Letter, Cont'd

… I have heard neural network researchers say, with apparent pride, that devices they have built sometimes surprise them. A good engineer would feel shame not pride. In safety-critical applications, it is the obligation of the developers to know exactly what their product will do in all possible circumstances. Sadly, we build systems so complex and badly structured that this is rarely the case."

— David Lorge Parnas

Dave Parnas's Complaint

Dave's complaining that AI are not behaving logically, in predictable manners, and are stochastic in their behavior …

like the humans that they are replacing.

We are giving AIs powers to do things *faster* than humans can and at a *bigger* scale, without any way to guarantee that they will do so *better*, *more reliably*, and *more predictably* than humans can.

Very Frightening! ☹

This is the stuff of technology-based horror movies!

E.g., Star Trek’s *Doomsday Machine* or V’ger

An AI could have a *more widespread* catastrophic effect *faster* than we humans are able to notice it and stop it.

A Graduate Seminar I Taught

Fast forward to 2019, to my “Advanced Topics in RE” graduate seminar at UW

I asked the AI PhD students in the class to prepare their paper and talk on RE for AI, …

hoping to learn what AI people consider to be a specification for an AI.

A Graduate Seminar, Cont’d

They agreed to do so, and …

I was *really* looking forward to finally learning what a spec for an AI is!

Wotta Disappointment!

They repeated all the usual RE parenthoods:

> **correctness**
> **completeness**
> **consistency**
> **robustness**
> **reliability**

Yeccchhhhh!!!!! ☹

A Conversation

between me (D) and one of the students (S), who shall remain nameless to protect his reputation:

At the end of the talk:

D: OK.. but what *is* correctness?

S: (shrugs)

D: I mean, how do you *know* that the AI you have produced is correct?

A Conversation, Cont'd

S: You don't! It's probably *not*!

D: That's horse s--t! I have seen AIers continuing to revise their AI until something was true. I would hope that that the something is correctness. So what *is* that something?

S: Ah! They keep at it until the recall is high enough and the precision is low enough.

D: Ah! So how do you know that the recall is high enough and the precision is low enough?

A Conversation, Cont’d

S: We guess! We just feel it!

D: Ewwwww!

Want to Use the RE Ref. Model

Dave and *I* desperately want an AI to behave like SW in the RE Reference Model, the (Zave–Jackson Validation Formula) ZJVF …

to be able to validate that an AI is behaving as expected, i.e., as specified.

Slides About ZJVF

Slides about the ZJVF from my talk about the RE Reference Model for my RE course at UW’s BSE degree program.

CS445 / SE463 / ECE 451 / CS645

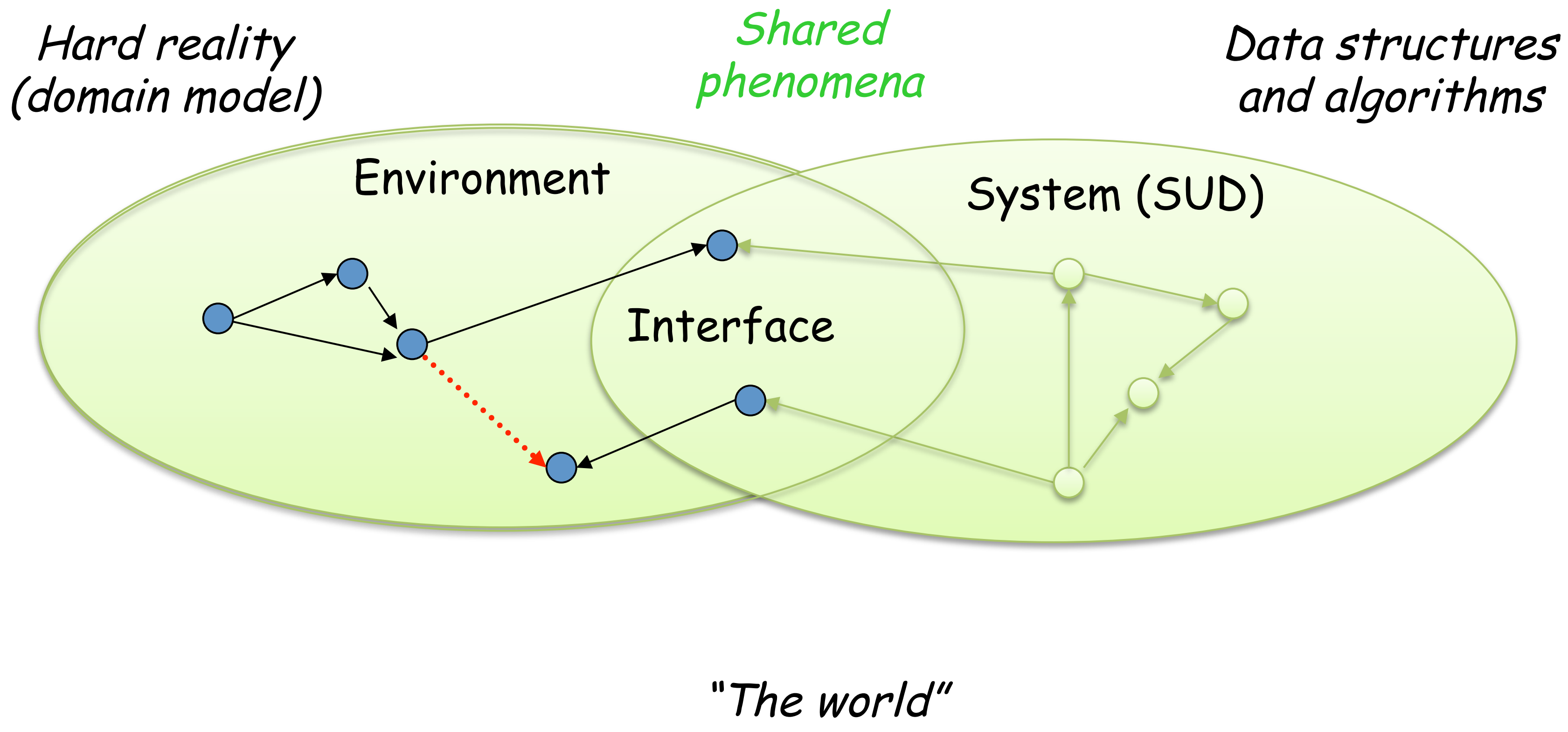
Software requirements specification & analysis

A reference model for requirements engineering

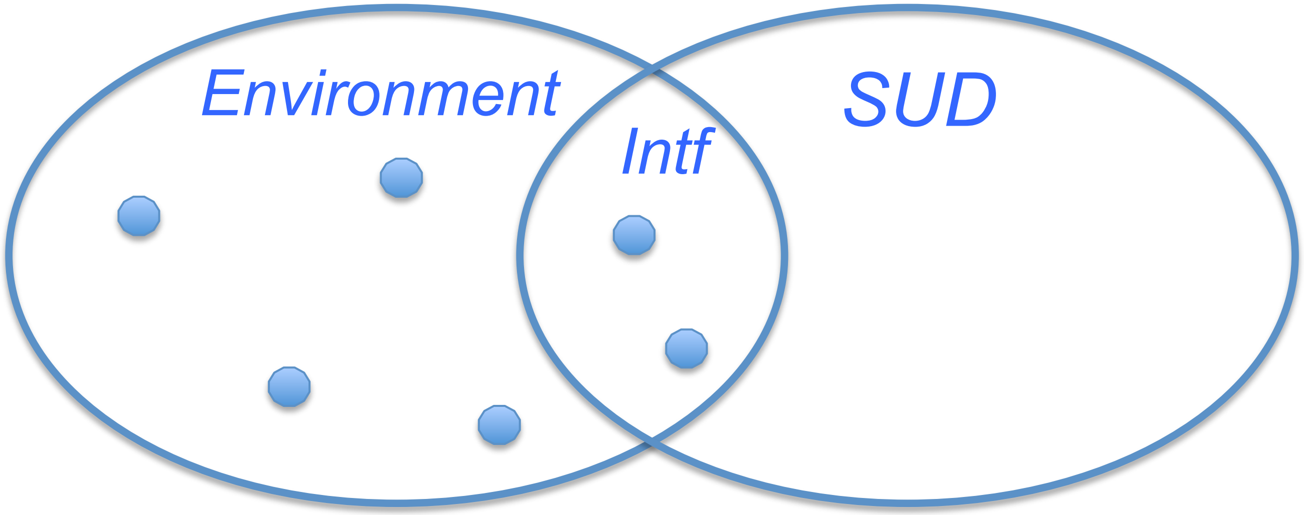
Fall 2015

Mike Godfrey & Daniel Berry & Richard Trefler

Reqs, specs, and programs



Reference model



R – Requirements live in ENV (incl. INTF)
S – Spec lives in INTF, describes behaviour of SUD
D – Domain knowledge lives in ENV (incl. INTF)

Reference model

- Thus, if we enlarge our model to include domain knowledge, then the following relationship must hold:

$$\mathbf{D, S \vdash R}$$

  - D is domain knowledge
  - S is the specification
  - R is the requirements

- The specification describes the behaviour of a system that realizes the requirements.
- The domain assumptions are needed to argue that any system that meets the specification (and that manipulates the interface phenomena) will satisfy the original requirements.

Hidden: Traffic light example

- D = drivers behave legally and cars function correctly
- S = spec of traffic light that guarantees that perpendicular directions do not show green at same time
- R = perpendicular traffic does not collide

Problem: make D unnecessary, steel walls pop up on red, light controls cars by wireless

Reference model

- If you can’t prove this, then at least one of 3 things must have gone wrong:
  - requirements are incorrect / unreasonable
  - system doesn’t do enough
  - we aren’t assuming enough about the environment

Uncertainty in "$D, S \vdash R$"

- The formula $D, S \vdash R$ tries to be formal in the sense of describing what happens completely.
- One would expect computers and software and their combination to be formal in this sense.
- But, the real world intervenes to make this formula only a guideline and not an accurate, precise model.

Hidden: Uncertainty in “D, S $\vdash$ R”

- First, the real world *never* behaves as *any* model.
- Any model D is only an approximation.
- Generally, the simpler the model, the more of an approximation the model is, but the easier it is to prove things about the model.
- Modeling the real world accurately requires complexity to deal with all the weird exceptions.
- A mechanistic description generally has to be replaced by or tempered with a probabilistic model, e.g., 99.99% of drivers stop at a red light.

Hidden: Uncertainty in "D, S $\vdash$ R"

- At the lowest level, a CBS is mechanistic, e.g., a traffic light, the sqrt function, and can be modeled with a consistent S that is mechanistic, that always gives for any input the same answer that the CBS does.
- But floating point arithmetic is not the same as real numbers, and integer arithmetic suffers over & underflow.
- At higher levels, e.g., MS Word, an operating system, process control, etc., the CBS is so large that we cannot understand all of its code and all of its behavior. So, we begin to give probabilistic models of what the CBS does.

Hidden: Uncertainty in "D, S $\vdash$ R"

- All that applies to D, applies to R, because both are models of the real world, one as is, and the other as it is to be.
- R is always an approximation of what we want, because if we overlook something in the real world and it turns out to be relevant to the CBS's behavior, e.g., a gaggle of Canadian geese that fly near a jet engine, then R may not be correct.

Uncertainty in "$D, S \vdash R$"

- The formula $D, S \vdash R$ tries to be formal in the sense of describing what happens completely.
- But, as we have seen, it cannot be completely formal because at least D and R have to describe the real world, which is not formal

What does this do to the hope of formally modeling computer systems?

Molecular Software

- Molecular SW, e.g., DNA, RNA, Proteins, Catalysts
- Molecules designed specifically to achieve a desired effect
- Molecule is shown empirically to behave as specified in S, with 99.95% certainty
- In this case, in D, S $\vdash$ R, also S is informal!

AI’s Behavior is Stochastic

Since the behavior of an AI is stochastic, like molecular SW, the truth of S is empirical, not logical.

So now validation of an AI has three empirical truths instead of just two, and logic plays almost no part.

Brooks's “No Silver Bullet”

No Silver Bullet

Essence and Accidents of Software Engineering

Frederick P. Brooks, Jr.

University of North Carolina at Chapel Hill

Fashioning complex conceptual constructs is the *essence; accidental* tasks arise in representing the constructs in language. Past progress has so reduced the accidental tasks that future progress now depends upon addressing the essence.

Of all the monsters that fill the nightmares of our folklore, none terrify more than werewolves, because they transform unexpectedly from the familiar into horrors. For these, one seeks bullets of silver that can magically lay them to rest.

The familiar software project, at least as seen by the nontechnical manager, has something of this character; it is usually innocent and straightforward, but is capable of becoming a monster of missed schedules, blown budgets, and flawed products. So we hear desperate cries for a silver bullet—something to make software costs drop as rapidly as computer hardware costs do.

But, as we look to the horizon of a decade hence, we see no silver bullet. There is no single development, in either technology or in management technique, that by itself promises even one order-of-magnitude improvement in productivity, in reliability, in simplicity. In this article, I shall try to show why, by examining both the nature of the software problem and the properties of the bullets proposed.

Skepticism is not pessimism, however. Although we see no startling breakthroughs—and indeed, I believe such to be inconsistent with the nature of software—many encouraging innovations are under way. A disciplined, consistent effort to develop, propagate, and exploit these innovations should indeed yield an order-of-magnitude improvement. There is no royal road, but there is a road.

The first step toward the management of disease was replacement of demon theories and humours theories by the germ theory. That very step, the beginning of hope, in itself dashed all hopes of magical solutions. It told workers that progress would be made stepwise, at great effort, and that a persistent, unremitting care would have to be paid to a discipline of cleanliness. So it is with software engineering today.

Does it have to be hard?—Essential difficulties

Not only are there no silver bullets now in view, the very nature of software makes it unlikely that there will be any—no inventions that will do for software productivity, reliability, and simplicity what electronics, transistors, and large-scale integration did for computer hardware.

This article was first published in *Information Processing '86,* ISBN No. 0-444-70077-3, H.-J. Kugler, ed., Elsevier Science Publishers B.V. (North-Holland) © IFIP 1986.

The essence of a software entity is a construct of interlocking concepts: data sets, relationships among data items, algorithms, and invocations of functions. This essence is abstract in that such a conceptual construct is the same under many different representations. It is nonetheless highly precise and richly detailed.

I believe the hard part of building software to be the specification, design, and testing of this conceptual construct, not the labor of representing it and testing the fidelity of the representation. We still make syntax errors, to be sure; but they are fuzz compared with the conceptual errors in most systems.

If this is true, building software will always be hard. There is inherently no silver bullet.

Let us consider the inherent properties of this irreducible essence of modern software systems: complexity, conformity, changeability, and invisibility.

Complexity. Software entities are more complex for their size than perhaps any other human construct because no two parts are alike (at least above the statement level). If they are, we make the two similar parts into a subroutine—open or closed. In this respect, software systems differ profoundly from computers, buildings, or automobiles, where repeated elements abound.

Digital computers are themselves more complex than most things people build: They have very large numbers of states. This makes conceiving, describing, and testing them hard. Software systems have

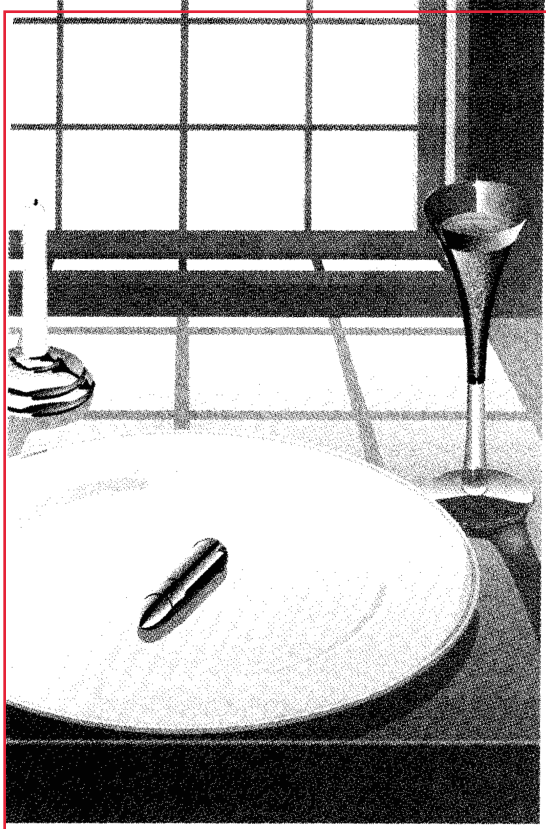
the essential properties of the phenomena. It does not work when the complexities are the essence.

Many of the classic problems of developing software products derive from this essential complexity and its nonlinear increases with size. From the complexity comes the difficulty of communication among team members, which leads to product flaws, cost overruns, schedule delays. From the complexity comes the

alone in facing complexity. Physics deals

The Photo Source International Inc.

Harel’s “Biting the Silver Bullet”



Biting the Silver Bullet

Toward a Brighter Future for System Development

David Harel, Weizmann Institute of Science

A "vanilla" approach to modeling, together with powerful notions of executability and code generation, may have a profound impact on the "essence" of developing complex systems.

In an eloquent and thoughtful 1986 article, Frederick Brooks expresses his feelings about the illusions and hopes software engineering offers.[1] He argues that many proposed ideas are not "silver bullets" that will deliver us from the horrors of developing complex systems.

Brooks' article is reminiscent of Parnas' series of minipapers[2] that accompanied his widely publicized resignation from the Strategic Defense Initiative Organization (SDIO) Panel on Computing in 1985. Parnas claims that current proposals are vastly inadequate to build reliable software as complex as that required for the SDI project.

We thus have two rather discouraging position papers, authored by two of the most influential figures in the software world. Neither is a critique of software engineering per se, although both make an effort to dissolve myths of magical power that people have cultivated concerning certain trends in the field.

This article was triggered by those of Brooks and Parnas. It is not a rebuttal. Indeed, I agree with most of the specific points made in both papers. Instead, the goal of this article is to illuminate the brighter side of the coin, emphasizing developments in the field that were too recent or immature to have influenced Brooks and Parnas when they wrote their manuscripts.

The two main aspects of these developments have to do with a carefully wrought "vanilla" approach to system modeling and the emergence of powerful methods to execute and analyze the resulting models. It can be argued that the combined effect of these and other ideas is already showing positive signs and appears to have the potential to provide a truly major improvement in our present abilities — profoundly affecting the essence of the problem. This might take more than the 10 years Brooks focuses on. It will surely be a long time before reliable software for the likes of the SDI project can be built. Such a system remains an order of magnitude too large and too critical to construct today, mainly because of its first-time-must-work nature. But I also believe that we are on the royal (main) road and that the general impression you get from reading the Brooks and Parnas articles is far too bleak.

On biting bullets

There are two opinions about the origin of the phrase “Biting the bullet.” One is that it came from the need to bite the top off the paper cartridge prior to firing a certain kind of British rifle used in the mid 19th century. This often had to be done under enemy fire and required keeping a cool head.

The other is that it is an old American phrase, rooted in the folklore of the US Civil War. It supposedly emerged from the practice of encouraging a patient who was to undergo field surgery to bite down hard on a lead bullet to “divert the mind from pain and prevent screaming” (R.L. Chapman, *American Slang*, Harper and Row, New York, 1986).

In more recent years, the phrase has come to signify having to do something painful but necessary, or to undertake an activity despite criticism or opposition, while exhibiting a measure of courage and optimism.

Biting the AI Bullet

Instead of Parnas’s despair and unrealistic hope of prohibiting AIs altogether …

or at least making them non-stochastic or logical in their behaviors,

we need to bite the AI bullet.

Biting, Cont'd

Change the nature of a specification of an AI to take into account the stochastic behavior and give empirical measures of acceptable behaviors …

Biting, Cont’d

so that validation becomes akin to empirically proving the hypothesis

“The AI behaves as specified.”

and we give confidence intervals or *p* values to the claims of how well the measures are matched …

***and* we make engineering judgements for close calls.**

Leading to REFSQ 2022 Paper

This realization led to the paper I presented at REFSQ 2022:

“RE for AI: What is an RS for an AI?”

Slides from REFSQ 2022 Talk

RE for AI: What is an RS for an AI?

Daniel M. Berry
University of Waterloo

Main Insight of REFSQ Paper

The main insight of my REFSQ’22 paper is that a specification for an AI for a hairy task consists of

1. **a set of measures used for evaluation,**
2. **criteria that the measures must satisfy, and**
3. **other data about the context of the use of the AI, including the RW data that teaches an LM.**

Set of Measures

The set of measures used for evaluation measures *correctness* in some sense and is usually calculated from a confusion matrix, e.g.,

- **recall and precision,**
- **sensitivity and specificity,**
- **F-measure**
- **accuracy**

Criteria For Measures

The criteria that the measures must satisfy …

help show that the AI …

can be considered as …

mimicking or doing better than a human doing the same task.

Criteria, Cont'd

These criteria will usually include …

the values of the measures that humans *actually* achieve …

when doing the same task.

Other Data

The other data are data about the context of the use of the AI that …

- **allow engineering tradeoffs to help the AI meet the criteria and**

- **decide borderline cases.**