Why AI Won’t Solve the Software Problem

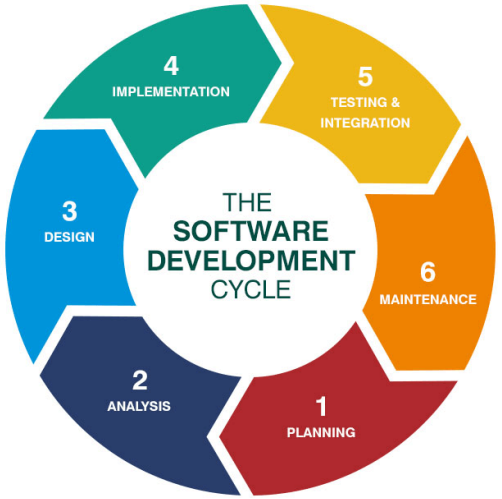
Prof. Jonathan I. Maletic

Department of Computer Science

KENT STATE UNIVERSITY

Software Engineering Research

- Concerned with better understanding how we:
  - Build
  - Evolve
  - Maintain
  - Understand
- Large software systems (>10 million line of code)
- Software is built by teams of developers and is constantly evolving to address new requirements over time



Software Quality

- Most software systems need to work correctly *most* of the time
- Some software systems need to work correctly *all* the time
  - Modern cars have 100 million+ lines of code

- Getting software to work perfectly correct (in general) is a very difficult (impossible) problem

- Why?

Halting Problem

- Will a computer program eventually halt?
- Can we construct a program to determine if another program ever finishes?

- Alan Turning proved [1937] that this problem is undecidable

- That is, no general solution exists (i.e., it is undecidable)

Proof - informal


```
void foo() {
     if (halt(foo))
          while (true);
}
```

- If **`halt(foo)`** returns **true**, then foo will never halt – *contradiction*
- If **`halt(foo)`** returns **false**, then foo will halt - *contradiction*

- Thus, the assumption that **`halt`** is a solvable is false Q.E.D.

Proof - informal

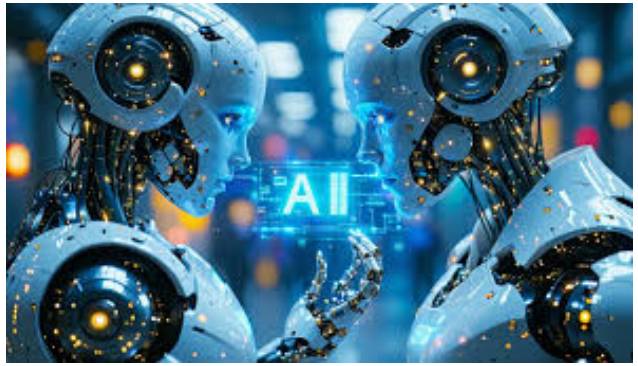

```
void foo() {
     if (halt(foo))     //Assume solvable
          while (true);
}
```

- If **`halt(foo)`** returns **true**, then foo will never halt – *contradiction*
- If **`halt(foo)`** returns **false**, then foo will halt - *contradiction*

- Thus, the assumption that **`halt`** is a solvable is false Q.E.D.

Implication

- We do not have the means (in general) to automatically determine (via a computer program) if a computer program is correct



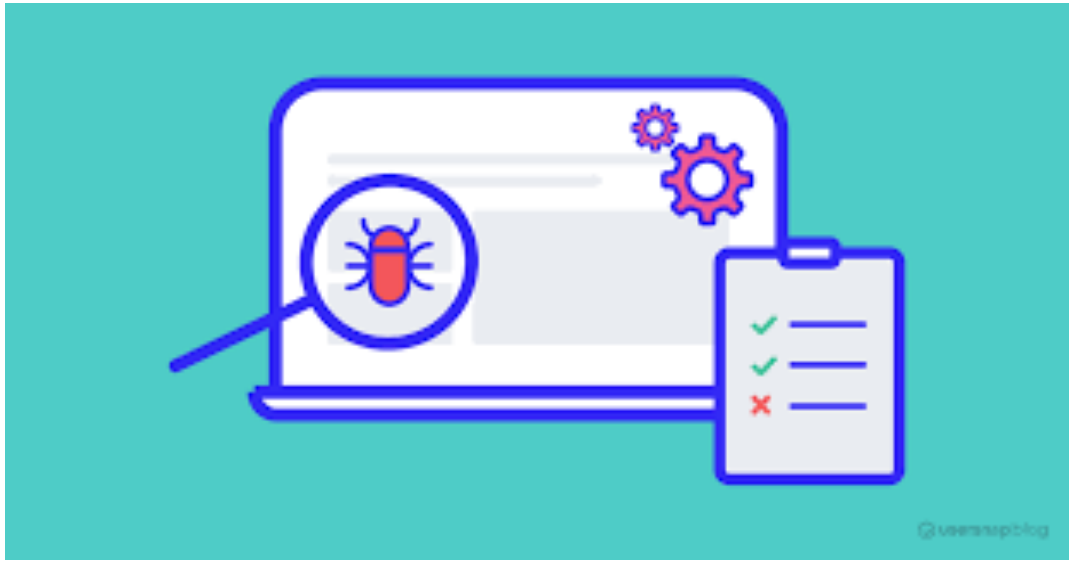
- AI tools (LLMs) are computer programs
- We can’t know if they are correct

- Any software written by an AI tool most likely has errors
- Any software written by a person also most likely has errors

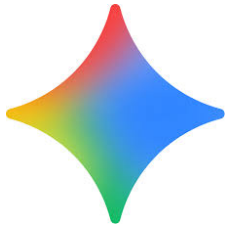
Testing, Testing, and More Testing

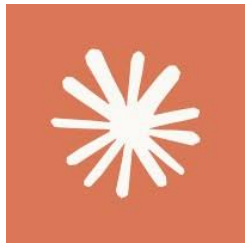
- Software quality is predominately driven by testing
- Develop test cases and run them against software
- Develop more test cases (more clever ones)
- Automated tools exist to help generate test cases
- Manual code reviews pay off

- However we still have bugs, performance problems, security flaws, etc.



Large Language Models

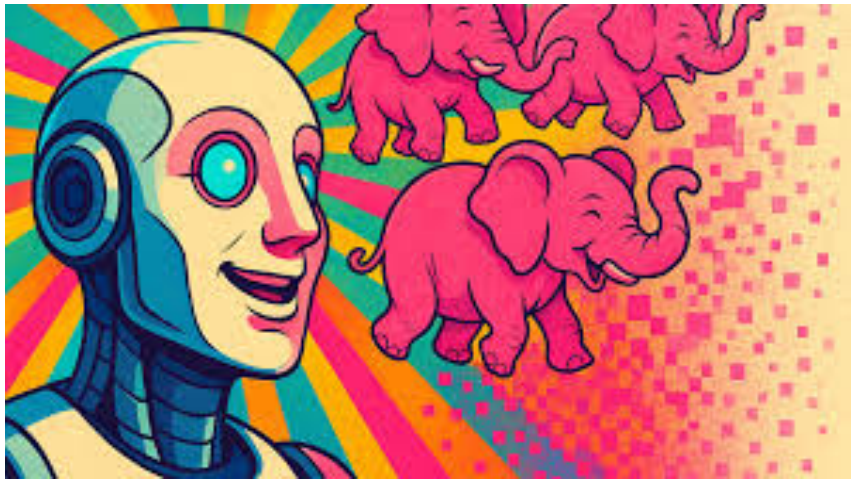




- ChatGPT, Claude, Gemini, etc.
- Trained on massive amounts (trillions) of documents
- Builds a predictive mathematical model (billions of parameters)
- Given a prompt, generates the best sequence of words to respond with

Problems with LLMs

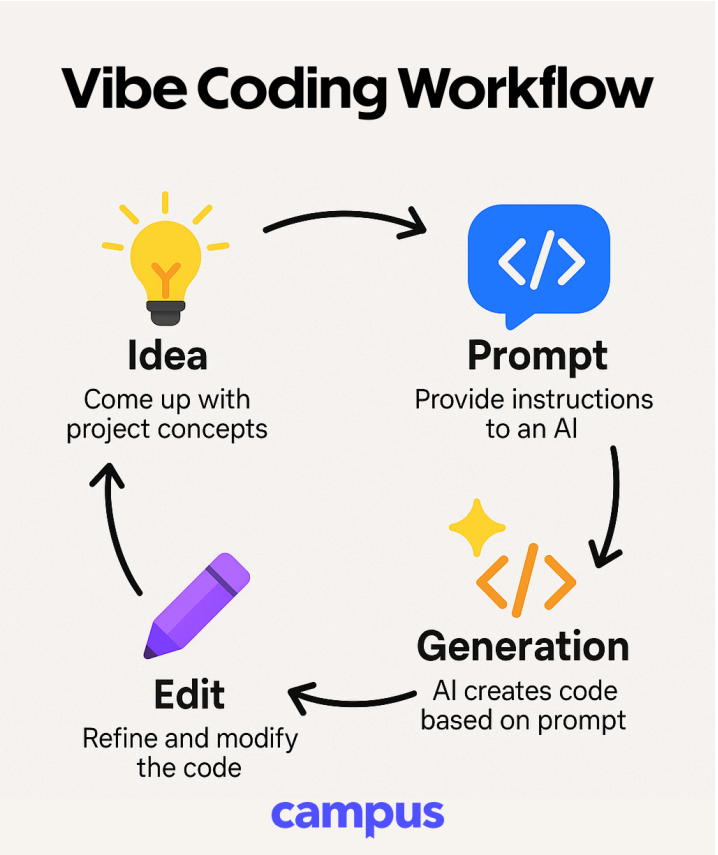
- No thinking! Absolutely does not think harder!
- Only as good as we are (in the very best case)
- No creativity – only trained on existing material
- Junk-in Junk-out
- Nondeterministic
- Hallucinations are not a feature (flaw)



Vibe Coding

- Developer focuses on the “vibe” of what they want solved rather than the technical (coding) details of the solution

- Prompt an LLM, in plain English with what you want to solve
- Generates the code to address the problem
- Iteratively improve the solution



Does it work?

- Works fairly well for low risk tasks:
  - Simple applications
  - Webpages
  - User interface, front end code

- “I’d rather eat glass than write more JavaScript” – grad student quote

- Only can solve something that is common and already solved

What Can AI do for Software Engineering?

- Help with the boring/mundane
- Help building test cases

- Allows access to expert knowledge on little known subjects
  - Software libraries
  - Rarely used tools/features

- Rapid prototyping – trying out a new idea
- Productivity tool

What AI Can’t do for Software Engineering

- Can’t write bug free code
- Can’t build complex systems

- They suggest solutions that are just plain wrong
- A productivity tool but with a quality trade off

- Can not replace experienced engineers/developers

Take Aways

- Building high quality software is extremely difficult, costly, and time consuming

- Each technical/productivity advancement brings a proportional increased appetite for larger and more complex software systems

- AI tools are productivity tools that allow developers to build a bit more software per hour

No Silver Bullet

- Society has an insatiable need for more complex, larger software systems - AI feeds this, does not solve it

- Without human experts continuously in the loop, AI will generate seriously flawed code that endangers users