The True Cost of AI Assistance to Programming of Software

Daniel M. Berry,
University of Waterloo

Cost to Correct Defects

ChatGPT Released

ChatGPT was released to the world in 2022

with much hype.

Used to Help Write Code

Soon after, people began to use ChatGPT and other LLMs to help write code …

and to rave about how great the code is.

But …

But, …

I saw reports:

About 25% or more of ChatGPT-generated code is wrong and must be corrected.

I Began to Wonder

I began to wonder about the cost to correct code generated by ChatGPT,

in response to a human's prompting.

The code will not be perfect.

So, the prompter will have to correct it, …

but at what cost?

Historical Correction Cost

Given the historical 10-fold-correction-cost (10FCC) factor,

if indeed, 25% of ChatGPT-generated code is wrong, …

then a human's correcting the code should cost 2.5 times

the cost of the human's writing the code from scratch.

Factor Will Be Even More

But, the factor to correct ChatGPT-generated code will be …

***even* worse than 10 fold.**

Unspoken Assumption

Unspoken assumption of the old data:

Humans are correcting code written by themselves and others in their development teams.

So, correctors have some familiarity with the code they are correcting.

Even if project uses independent quality assurers, not in project, to *find* defects,

their *correction* is left to project members.

Why Unspoken Assumption

Assumption was unspoken because

no one even thought of the possibility that something other than a person would write code,

and people corrected their own defects,

because anyone else doing it would have to spend time studying the code to understand it well enough to find the origin of any defect.

ChatGPT-Generated Code

Code generated by ChatGPT does not meet this assumption.

Therefore, the human who prompted for the code

will have to study the code thoroughly to

even begin considering how to correct its defects.

Humans Studying ChatGPT's Code

Also, when the human starts to study the code for defects,

E* has *no* idea what and where the defects in the code are.

*** "E", "em", and "er" are gender non-specific third-person-singular pronouns in subjective, objective, and possessive forms, respectively.**

Humans Studying, Cont'd

The defects could be anywhere,

including nowhere,

on slim chance that the code is correct to begin with.

Only careful study of the code can say.

Reports of Success

Nevertheless, people I respected as honest scholars were

reporting success at engineering prompts

that persuade ChatGPT to generate correct code.

Final Conclusion:

ChatGPT could be successfully prompted to produce code for any set of requirements, …

but along the way, …

the generated code would have lots of defects.

These would have to be found, with no idea what they even are.

Final Conclusion, Cont'd

Only then can they be corrected, and

the correction will be of unfamiliar code,

written by someone or something else.

Final Conclusion, Cont'd

The whole process of finding and correcting the ChatGPT-generate code

will be significantly more expensive

than finding and correcting code written

by humans in the same development project.

I therefore hypothesized:

High AI Copiloting Cost

HAICopC Hypothesis:

The cost for a set of humans
to correct the code generated
by an LLM
to implement a set of requirements
is significantly larger than
the cost for a set of humans
to correct the code programmed
by the same set of humans
to implement the same set of requirements.

Hypothesis is My Theory

The HAICopC Hypothesis is my theory and

it must be subjected to empirical tests to confirm it.

Very difficult to conduct experiment to soundly test the hypothesis.

Anecdotal Evidence

Later, I started seeing articles, blogs, and posting reporting observations consistent with the HAICopC Hypothesis: E.g.,

https://www.cio.com/article/3540579/devs-gaining-little-if-anything-from-ai-coding-assistants.html

https://www.linkedin.com/posts/saranyan_llm-vibecoding-ai-activity-7329881611460575232-gH5w

https://www.linkedin.com/posts/victor-schwartz_i-went-all-in-on-ai-coding-tools-to-build-activity-7335676769817001986-DsEL

Vigraham's Post

Let's examine one that provides data:

Saranyan Vigraham, Director of Engineering at Meta, kept data for 8 [sic] different development tasks,

ranging from feature development to migration,

that differed also in the level of Vibe-Coding-AI involvement, from 10% to 80%

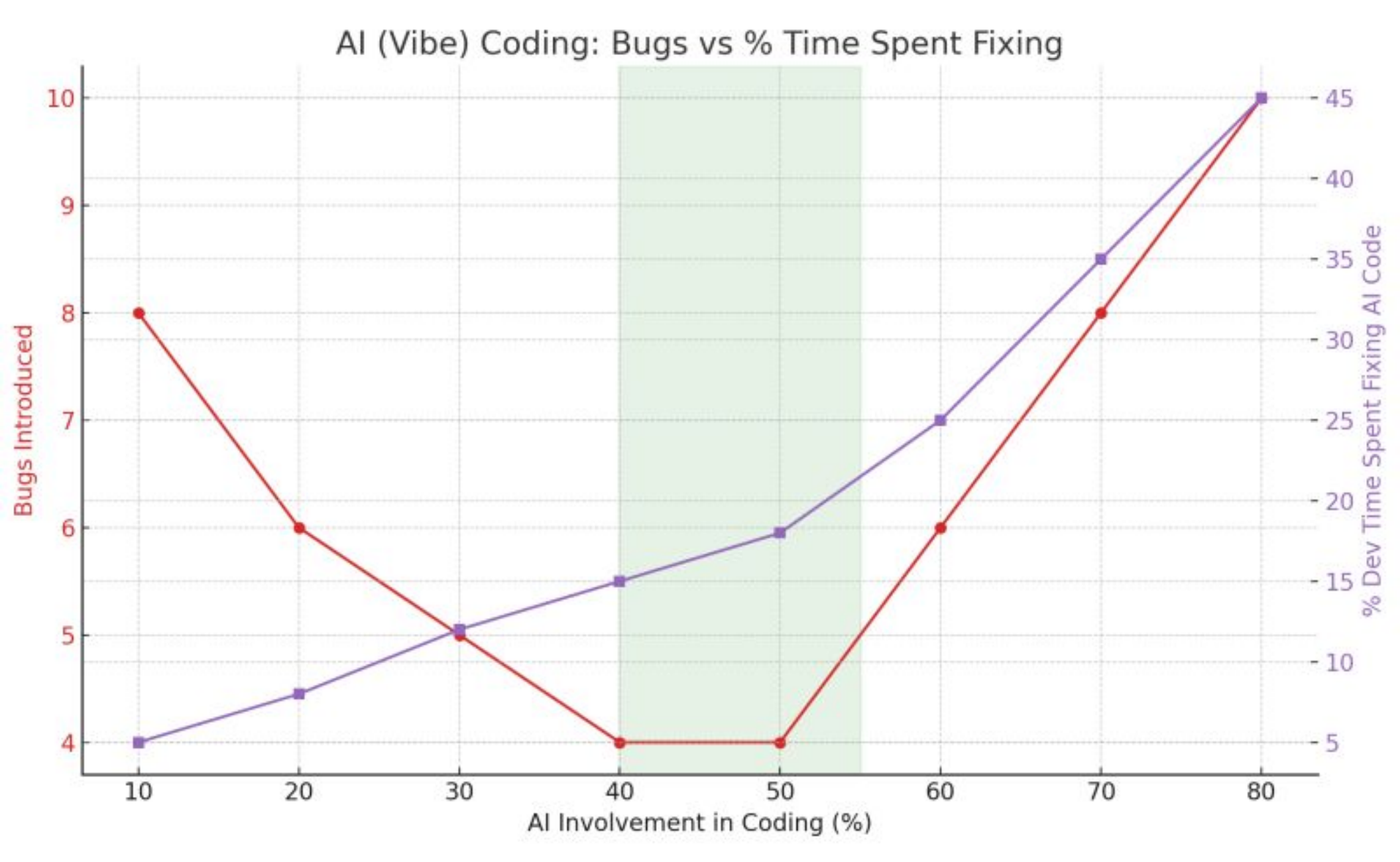
His Data

His graph plots both

1. **bugs introduced and**

2. **percentage of development time spent correcting these bugs**

against the 8 percentages of AI involvement.


AI (Vibe) Coding: Bugs vs % Time Spent Fixing
Bugs Introduced
% Dev Time Spent Fixing AI Code
AI Involvement in Coding (%)
10
9
8
7
6
5
4
45
40
35
30
25
20
15
10
5
10
20
30
40
50
60
70
80

Vigraham's Conclusions

He concluded that the "sweet spot … [was] 40–55% AI involvement. Enough to accelerate repetitive or structural work, but not so much that the codebase starts to hallucinate or drift".

Addressing Hypothesis?

Vigraham's data do not address the HAICopC Hypothesis:

He uses percentage of AI involvement as the independent variable.

View the same data with task as the independent variable:

Then they indirectly address the HAICopC Hypothesis.

Change Independent Variable

Vigraham designed the sequence of 8 development tasks, T1, …, T8

to involve the AI in approximately 10%, …, 80%, respectively, of the task.

Vigraham reveals the tasks for the percentages:

Tasks

“Where AI shines:

1. **Boilerplate and framework code**
2. **Large-scale refactors**
3. **Migration scaffolds**
4. **Test case generation**

Tasks, Cont'd

Where it stumbles:

1. **Complex logic paths**
2. **Context-heavy features**
3. **Anything requiring real systems thinking [and new architectures etc.]**
4. **Anything stateful or edge-case-heavy"**

Change Variable, Cont'd

There are 8 task categories, one for each percentage of AI involvement.

Vigraham's sweet spot is right there

between the last AI-shining task (10% – 40% AI involvement)

and the first AI-stumbling task (50% – 80% AI involvement).

Change Variable, Cont'd

Traditionally,

the first 4 tasks are done at the beginning of a CBS development

in preparation for

the second 4 tasks, which constitute the meat of the development.

In the Past

In the past, none of the first 4 tasks were considered programming.

It was called building the platform with modules from an artifact library (AL).

In the Past, Cont'd

Any serious programming shop had debugged boilerplate and framework code, migration scaffolds, and test cases for its suite of CBSs in its AL.

No one would program these from scratch if E could find what E needed in the AL, …

and in any serious shop, E *could* find what E needed.

First 4 Tasks

Thus, none of these first 4 tasks would be introducing defects that would be subject to the 10FCC factor.

Traditionally, search engines are used to help in these tasks,

and ChatGPT's recall is known to exceed the recall of all traditional search engines.

Last 4 Tasks

The 10FCC factor would show up only in the last 4 tasks, …

which involve

- **complex logic paths,**
- **context-heavy features,**
- **real systems thinking [and new architectures etc.], and**
- **statefulness or edge-case heaviness,**

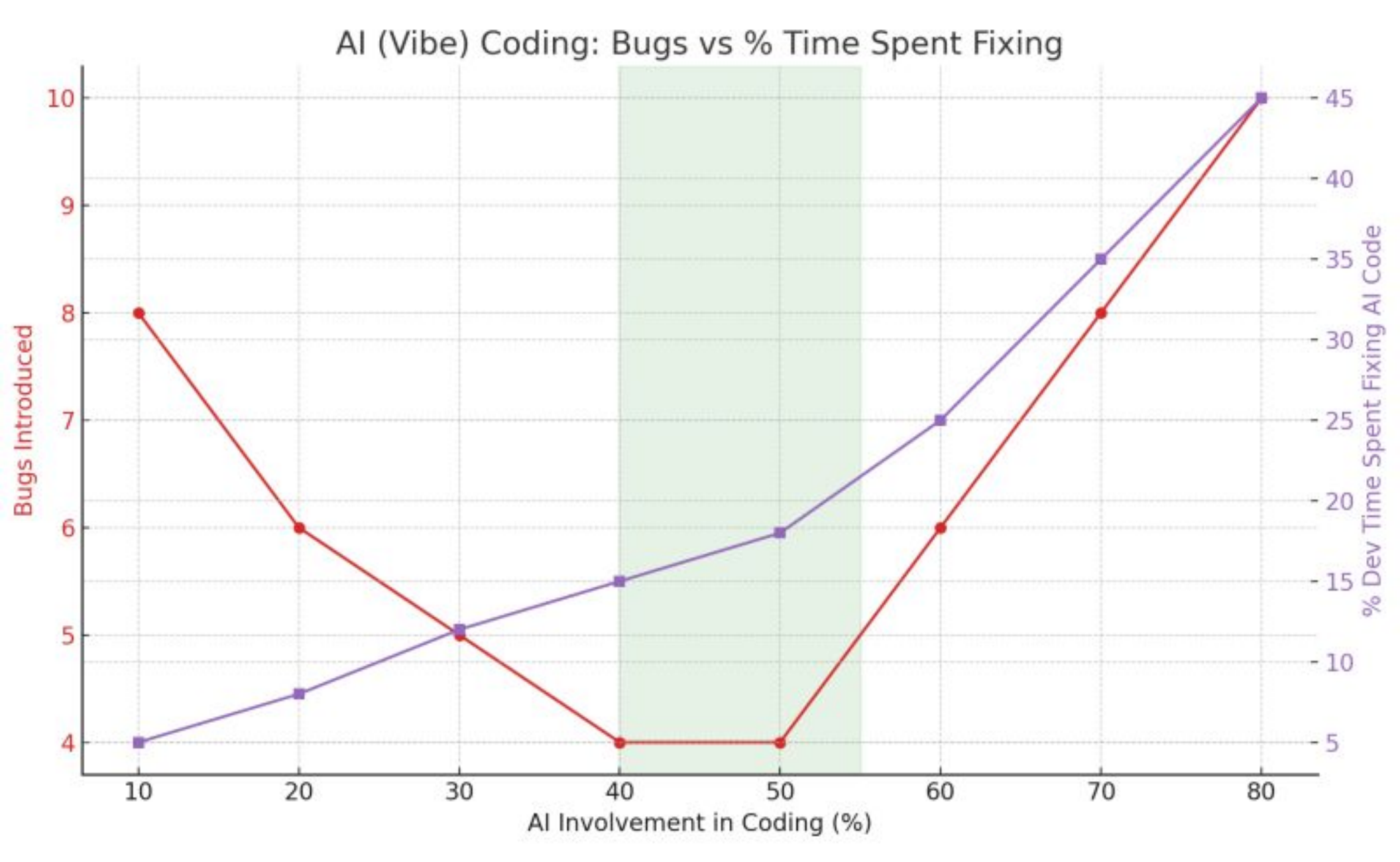
in short, which involve real programming,

Real Programming

Therefore, the part of Vigraham's data

that is from what is considered real programming,

can be seen as indirectly addressing the HAICopC Hypothesis.


AI (Vibe) Coding: Bugs vs % Time Spent Fixing
Bugs Introduced
% Dev Time Spent Fixing AI Code
AI Involvement in Coding (%)
10
9
8
7
6
5
4
45
40
35
30
25
20
15
10
5
10
20
30
40
50
60
70
80

Addressing the Hypothesis

In this part,

as AI involvement grows from 50% through 80% AI,

the number of bugs and

the time to correct the bugs

are growing,

thus, beginning to address the HAICopC Hypothesis.

Schwartz's Post

Victor Schwartz, with a CS degree and years of programming experience from an early age, does not consider himself to be a production engineer that builds large-scale CBSs.

He reports that when he started to build his most recent product, AI tools were magical.

With them, he was able to build working front-ends in minutes and prototype complex features in hours.

Schwartz's Post, Cont'd

He "Went from zero to demo faster than ever."

For building prototypes he found AI tools be be "genuinely transformative".

Once the demo yielded a go-ahead, he would normally turn the working prototype over to production engineers to build the production version.

Schwartz's Post, Cont'd

But this time, he thought that he could push ahead, prompting the AI to build the production version.

Then he hit the brick wall of reality.

Schwartz's Post, Cont'd

Now that he needed to [write *real*, hairy code], the AI tools started hallucinating.

He realized too late, after many months of refactoring, that these activities have a steep learning curve that is way beyond the AI tools' capabilities.

Conclusions

Many people are jumping on to the AI coding bandwagon, …

and are prompting ChatGPT and other LLMs to generate for their requirements.

But, these generated programs are chock full of defects that must be manually found and fixed.

Conclusions, Cont'd

That correcting a program costs 10 times what writing it correctly from the beginning costs, is long and well established.

Unlike with code written by humans in the same project, correcting ChatGPT-generated code requires the correcting humans to study the code first in order to even begin to incur the 10FCC

Conclusions, Cont'd

So I hypothesize that the HAICopC factor will be significantly larger than the 10FCC.

Empirical studies are needed to validate or falsify the hypothesis.

In the meantime, …

Conclusions, Cont'd

Those programmers who are having lots of fun engineering prompts to coax their favorite LLMs to finally generate code that meets their requirements

need to consider the cost of what they are doing and

whether it is cheaper and more reliable to do all but the preparatory platform building manually.