

Towards AI-Native Software Engineering (SE 3.0): A Vision and a Challenge Roadmap

Presentation by Asim Waheed
CS846 – Mike Godfrey – Week 2

Introduction

- Roadmap paper
 - Identifies **key issues** in current Software Engineering (SE) pipeline
 - Provides **roadmap** for AI-native SE
- Provides **new vision**
 - Questions what Software Developer means
- Identifies **challenges**
 - What are potential roadblocks for vision?

Is the current software development pipeline **broken**?

III The Problem

SE 2.0: AI-Assisted Software Engineering

- Current copilots are advanced code completion systems
- AI supports humans, but process still **inefficient**
- Identifies following key problems:
 - **High cognitive load** on humans
 - **Inefficient** model training
 - **Suboptimal** code quality

III New Idea

How can we **fix** AI-assisted Software Engineering?

SE 3.0: AI-Native

- Intent-First approach
(vibe-coding, anyone?)
- **Conversation-oriented** interactions
- Humans' job: **thinking**
- AI's job: **implementing**

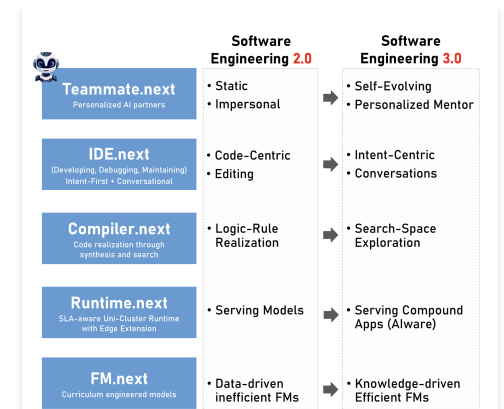


Fig. 3. Software engineering 3.0 technology stack.

Teammate.next

- Personalized **mentor**
- Self-evolving
 - **Learns** from its' **mistakes**
- Has **Theory of Mind**
 - Understands humans have different thinking pattern
 - Ideally, humans understand AI's thinking pattern

IDE.next

- **Conversation-based** UI
- Code **hidden** by default
- Conversations are **key asset**
 - Version control applies to conversations

Compiler.next

- Convert conversations to code
- Balances multiple objectives:
 - Accuracy
 - Latency
 - Cost
 - *Security?*
 - *Readability?*
 - *Scalability?*

Runtime.next

- Specific runtime meant for *FMWare*
 - Apps developed using AIs = FMWare
- Model and software *live on the same cluster*
- Resources *managed by usage*
 - Fine-tune model when usage is low
- Edge-computing
 - Simple requests routed to *smaller models*

FM.next

- Current Foundational Models (FMs) *trained on text*
- Introduce Curriculum Engineering:
 - Curate and organize *high-quality domain-specific knowledge*
 - *Externalize knowledge* from model

Positives

Did we like the paper?

Reimagination of a Software Developer

- Questions idea of **what it means** to be a software developer
 - Software developer != **programmer**
 - Human's job is to **think**
- Imagines entirely different skillset requirement
 - Humans **focus more on ideas** and providing value
 - Less time spent tinkering with technical details
 - *"Just get it done"*

Questions entire SE pipeline

- Changes workflow from **ground-up**
- Each aspect of pipeline talked about in detail
- **Provides direction** towards required advancements

Democratization of software



Before:

Subset of **highly skilled people** able to create software

Time spent programming = time **not spent thinking**



After:

People without tech backgrounds may be able to build software

Focus shifted to user and providing value



Negatives

What ideas were a **miss**?

Obfuscation of code

- Code is now in **background**
 - Do manual code reviews still exist?
- Security vulnerabilities
 - Likelihood of **same vulnerability** introduced to many systems high
- Loss of skill
 - Even trained software engineers may eventually **lose ability to code**

Democratization

- Truth: more people can now build software
- However, **SE3.0 is more resource intensive**
- **Increased gap** between high vs. low resource development

Monopolies

- Best FM will be **monopolized**
- Already true in status quo: paper does not address it

Feedback loop

- Briefly mentions it, does not give ideas to solve it
- Will all code eventually **look** and **run** the **same**?
 - Is that a **good** thing?

Rating: 3

Average.



Future Work

How can we extend this?

1. Incorporating security reviews

- How would prevailing security practices be affected?
- Security vs. utility tradeoff
 - Will AI make the tradeoff clear?
 - Will AI ignore it to enhance user experience?

2. Survey

- Different parts of roadmap require different levels of work
 - Teammate.next: *already quite close*
 - Runtime.next: *completely new idea*
- How much more research is required to build this vision?

Discussion Points

- How does this impact the software engineering curriculum?
- How would we prevent feedback loops?
 - AI learning on AI generated code
 - AI prompting AI with AI generated prompts
 - AI writing code that works better with other AIs
- Democratization vs Monopolization, which is it?
- Is code obfuscation something we want?