三浦真樹@富士通

FUJITSU

- **いま**
  - **生成AIのソフトウェア開発への適用について検討/実践中**
  - 上記に関する社内WG主査
  - Global Fujitsu Distinguished Engineer
    - ソフトウェアオープンイノベーション事業本部所属（2023/4~）
      - 富士通のミドルウェアを開発している本部です

- **これまで**
  - **社会インフラ領域のAI映像認識の技術開発をリード**
    - 車名判別、道路交通事象検知、車両スタック検出、交通量計測、河川監視、人物行動検知、etc.
    - **それ以前**は、画像処理アルゴリズム、組込みファーム、PCアプリ、各種サービス、映像ミドルウェア、お客様業務システム、etc.の開発

https://ses.sigse.jp/2022/

ご参考

- 新技術への挑戦で人々の暮らしを守る - AI映像解析で目指す安心安全な街づくり
- なぜ「素朴な疑問」から、社会を変えるテクノロジーが生まれるのか

LinkedIn

生成AIでのソフトウェア/システム開発

- **不確定性の内在する技術(AI)を使って高信頼なモノを作る**
  - 認識系：　統計的性能で許容可能
  - 生成系：　「不確定性の内在する技術(AI)を使って高信頼なモノを作る」ことへのチャレンジに面白さ&難しさあり
    - 開発者の新しいスキル、生成物チェック、著作権、技術伝承効率化、サポート、etc. … 開発プロセスの進化 (将来?)

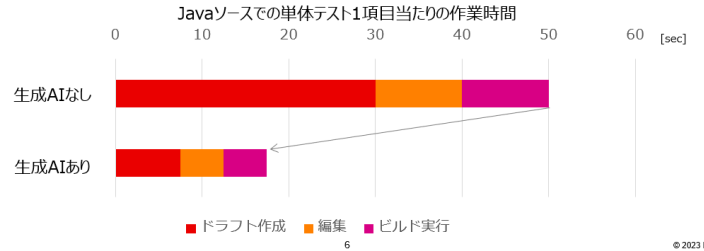
- **現在の開発プロセスにおけるコストパフォーマンス**

生成AIによるコーディング効率化

- 汎用的なコードでの効果大。独自コードでの効果低
  - 実際の開発では、効果がある部分とない部分の想定と定量化(比率)が重要
  - 世の中での評価・声とのギャップはある。ただし開発者の「すごい」という感動は大切
- 自分でコーディングできる&良否判断ができる、中級者以上による活用が必要
  - 教育目的などの場合は評価軸が異なる

生成AIによる単体テスト効率化

- ソースコードからの単体テストコード生成：　50-65%の作業時間を削減可能
  - ソースコードありき、がプログラミングへの適用との違い
- 生成AIが生成したテストコードの品質：　処理網羅率、条件網羅率のカバレッジは十分



SES2023
WS3: チャット型生成AIとソフトウェア工学
サマリー

2023.8.25

西 石川 徳 三浦

WS3概要

- チャット型生成AIの潮流に対してソフトウェア工学がどう変わり，あるいは変わらず，どのような役割を果たすのか，そしてソフトウェア工学の研究者や実践者がどう活動していくべきなのかという二つの問いを議論する.

- 2023年8月23日 9:00-16:30
- 石川先生からの最新動向インプット
- ポジションペーパー発表3編 (教育活用2編,生成AIと開発の今後1編)
- ディスカッション

https://ses.sigse.jp/2023/workshop.html#ws3

ディスカッションテーマ

1. AI for SE

生成AI活用がソフト開発にも拡大(壁打ち,コード生成,etc.)
今後のソフト開発のあり方は?

2. SE for AI

生成AI自体(LLM)/生成物の品質テスト,AIへの信頼,etc.
複数の課題について議論。

3. 教育・スキル・マネジメント

大学のコンピューターサイエンス教育/企業の技術者教育と生成AIの(悩ましい)関係
ChatGPT/生成AI活用のためのノウハウや、著作権リスクなどの問題理解

1. AI for SE

- **生成AIを利用したソフトウェア開発プロセスのあり方**
  - 現在の生成AIは、仕事は速いが質はイマイチな開発メンバー
  - 生成AIは(ソフト開発に人間が不要になる)神ではなくスーパープログラマー
  - **開発プロセスは、現在は既存と変わらず、個々のプロセスでツールとして活用。**
    - しかし次第にV字モデルの底まで行かずに済ませられるようになれば楽にはなる。

- **ソフトウェア開発のどこに使うのがいいのか?**
  - 公開情報、Webの情報になっている部分がいいのではないか?
  - デザイン系のCSSなどは最も分かりやすい例
    (効率100倍UP! 一方領域に特化したソフト開発だと20%UP程度のことも)
  - **開発プロセスについては、生成AIの存在を踏まえると、世の中にあるものにいかにフィットさせていくか、という視点で変えていくのが良い**のではないか?

2. SE for AI

- **生成系AIが生成した成果物の検証・修正の体系化・自動化 (w/生成AI)**
  - どのレベルの抽象度・具体度の入力が、生成系AIへの入力として適切なのか？
  - コード生成について、テストをパスするかでベンチマークだけでいいのか？
  - コード生成でも入力の情報量などで費用対効果のようなこともあるのでは？
  - 保守などのライフサイクルを考えると途中過程がスキップされるのものよくない？
  - コード以外の自然言語などだとどう評価する?
- **いかにハルシネーションを起こすようなテスト設計ができるか**
  - ハルシネーションの定義？よいもの・悪いもの
  - ハルシネーションにつながる特徴の理解など、振る舞いの理解や挙動確認
- **人間は信用できるがAIは信用できない？その境界は？**
  - 高信頼な領域では信用が重要
  - 責任をとれるかどうかの違いか

3. 教育・スキル・マネジメント

- **ソフトウェア技術者教育に対して生成AIをどのように活用できそうか**
  - **生成AIは避けて通れないが、かといって自由な利用では学生のComputer Scienceの力が身につかないジレンマ**
  - 生成AIが生成したコードのレビューやテストの教育。企業としては外注含め生成AI関連のマネジメント人材が必要
  - Computer Scienceを学ぶ初期にはChatGPTを禁止し、どこかのタイミングで解禁などが必要ではないか
  - プログラミング学習をChatGPTを生かして短縮し、データベースや性能などの難しい問題へのチャレンジを強化してはどうか
  - プロジェクト実現の中でソフト開発する教育と、コンピュータサイエンス的な教育を分離し、学生のニーズで選択してもらってはどうか
  - 事例：ChatGPTを使ったプログラミングコンテストを社内でやってみた。プロンプトの工夫のシェアやプロンプトに創意工夫を求めるようにしたらコーディングでの加減が把握できるようになった
- **ChatGPTを使いこなすための教育とは?**
  - プロンプトエンジニアリングがTips集みたいになってるがもっと体系化できないか。（そもそも「エンジニアリング」なのか?）
  - 目的に応じたプロンプトのデザインなどはまだ試行錯誤の状態。
  - 事例：JDLAやOpenAI社の開発者向けのプロンプトエンジニアリング講座を、社内のTipsとして使っている。
- **学生の場合、生成AI以外の技術力の方が、基礎体力として重要ではないか?**
  - コーディングより前にエンジニアリング。要件定義・設計が重要。文章を書く力、情報の構造化や検討ができるようになっていてほしい。
  - 生成AIがコーディングなどの負担を減らすとしても、一方で障害対応などはなくならない。その時対応できる技術力の維持。


WS3にご参加いただいた皆様
有意義なディスカッションありがとうございました
Thank you
SES2023 WS3
チャット型生成AIとソフトウェア工学
討論リーダー 西 石川 徳 三浦

AI for SE

①

生成AIを利用したソフトウェア開発プロセスの在り方

LLMを使うことで変わるプロセスと変わらないプロセスは？

SW開発の世界がどうなるのか？
それに向けて準備すべきこと

生成AIを手放しで信用できる時期がいつ来るか？

これまで研究されてきた技術分野はLLMによってどうなっていくか（例：program repair）

ソフトウェア開発においてLLMと人との協調をどのような形・インタフェースでするべきか

生成系AIのデータに対するマイニングソフトウェアリポジトリ研究は可能か？

生成AIを用いた不具合情報からの製品特性の抽出の可能性

生成AIを利用したソフトウェアシステムのシステムテストで考慮しておかないといけない品質特性

コードレビューでのChatGPT活用。過去のコード欠陥をどう生かしてコードレビューに使えるか

生成AIを使ってコード生成結果についてどのように評価するか（品質，効果）

AIサービスを使用した研究を論文として纏める事を考えると，対象AIのバージョンが提供されなくなり再現できなくなる場合をどう捉えるべきか（再現性への懸念，比較ができなくなる）

オープン情報と企業内のセキュア情報を両方用いる生成系AIの活用

②

ソフトウェア開発の中で生成AIをどこにどう使うのが効果的か？

ソフトウェア開発プロセスの中で生成AIを利用できるフェーズ，その効果について

ソフトウェア開発で生成AIを活用した場合、効果は定量的にどのくらいか?

ディスカッションサマリー
① 生成AIを利用したソフトウェア開発プロセスの在り方
現在は仕事は速いが質はイマイチなメンバー
生成AIは神ではなくスーパープログラマーをめざすはず
開発プロセスは、現在は既存と変わらず、個々のプロセスでツールとして活用。
しかしいずれV字モデルの底まで行きつかずに済ませられるようになれば楽にはなる。

ディスカッションサマリー
② 開発のどこに使うのがいいのか?
公開情報、Webの情報になっている部分がいいのではないか?
デザイン系のCSSなどは最も分かりやすい例
(100倍UP! 一方領域に特化したソフト開発だと20%程度のことも)
そう考えると、全部自前開発ではなく、逆に世の中にあるものにいかにフィットさせていくのかという視点で開発プロセスも変えていくのが良いのではないか?

SE for AI

SE for (AI for SE)

SE for
(Solution = AI + Human Interaction + Traditional Methods)

設計

ファインチューニングの精度向上のための必要な仕組み、ノウハウみたいなのがあるのか。版管理とかどうするものか

社内固有の情報・文書を取り込んで返答させる場合の設計・品質

コード生成でも入力の情報量などで費用対効果のようなこともあるのでは？

保守などのライフサイクルを考えると途中過程がスキップされるのものよくない？

コード生成だとテストをパスするかでベンチマーク

それがいいのか？

コード以外の自然言語などだとどう評価する?

スーパープログラマーだとして，プログラムを評価するのかプログラマー能力を評価するのか

どのレベルの抽象度・具体度の入力が、生成系AIへの入力として適切なのか？

評価・テスト

生成系AIに生成させたソフトウェア成果物の検証・修正の（生成系AIを用いた？）体系化・自動化

生成AIに任せるタスクの要求記述の方法
→
十分性や限界（対応可能範囲）評価の方法
→ 変更時評価の方法

人との協調
他技術との連携
の設計
品質評価
監視

原因分析的な利用をした場合の生成AIのアウトプットを、どのように担保するか

（分析系タスクの出力はにわかに良否を人が判断できない。検算が必要）

生成AIの品質メトリクス？品質メトリクスを報告させる

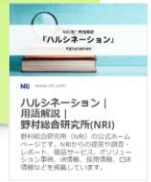
オムツとビールの妥当性と同じ？違う特性もある？（過程がブラックボックス）

運用・監視・適応進化

いかにハルシネーションを起こすようなテスト設計ができるか

ハルシネーションにつながる特徴の理解など振る舞いの理解や挙動，確認

ハルシネーションの定義？よいもの・悪いもの



LLMOpsにおける研究的な要素はどのようなものがあるか？

企業・実践ではいろいろ議論されているが

雨後のタケノコのようにたくさん生まれてる基礎モデルの比較評価や互換性？

様々なLLMが出てきたときに汎用的なプロンプトを作れるのか？

基礎モデルのバージョンアップ？に対する構成管理・変更管理

トラスト

人間は信用できるがAIは信用できない？その境界は？

高信頼な領域では信用が重要

責任をとれるかどうか

教育・スキル・マネジメント

- 要求仕様抽出
- 生成されたコードをレビューするコンテスト
- 生成AIを使わない教育と使う教育のハイブリッドが必要かもしれない
- トレーサビリティをどう確保するか
- バグ修正の練習に
- 読める力の教育
- ChatGPTを使ったプログラミングコンテスト（自分で実装しない），じゃんけんという用語を使わずにあっち向いてほいを実装せよ，等の問題など，使えそうという感覚を参加者に持たせることができた．
- 企業では，プログラムを書くだけでなく，DBやネットワークパフォーマンスを含めた高度な理解が必要．教育内容をそちらに持っていければ．
- GPTを前提とした教育とCSの基礎から積み上げていく教育を分けて実施しては
- とりあえず動くもので良い場合としっかり作る場合がある．とりあえず動くものを対象として生成AIを活用する機会があっても良いし，原理を積み上げていくものも重要．

エンジニア教育

幅広く，ソフトウェア技術者教育に対して生成AIをどのように活用できそうか

答えを開示させないで生成AIを利用する方法

エンジニアのモチベーション
将来のキャリアへの不安への考え方みたいなものは必要か

（ホワイトカラー置き換わる説）

学生と対話するステークホルダーを生成AIに代替させるとして，コンテキストをどのようにして理解させるか既存のベルソナ法との関連は？

パワハラ上司と組ませるよりは、chatGPTを使って若手の心理的安全性を確保できるかもしれない

学生が自主的にやりたくなり、かつ「必要な」技術力もつく方法（当然生成AIも利用した上で）があるといい

Copilotを用いたコーディングの場合，自動生成されたコードを編集する場面が多いと思います．
ある程度コーディングする実力がある人にとっては補助道具として役立ちますが，プログラミングの初学者が利用する場合に，基本的なコーディング能力の習得に向いているのかは疑問です．
使い所を考える必要があるのではないかと思います．(ChatGPTも同様)

chatGPTコードコピペで育った人たちは、性能とかメモリとかの概念は身につくのでしょうか

これから開発者になる人の場合、生成AI以外の技術力の方が、基礎体力として重要ではないか?
障害対応含めてすべてプロンプトで済む世の中になるまでは。

メモリ，オーダー，セキュリティ，マルチスレッドなど，基礎的に身につけてほしい概念を分野毎に明白にして教育？

LLMを利用すると学習効果が出るタイミングはいつなのか？初学者には不要？

ChatGPTを使ったプログラミングコンテストを実施。
早い人にコツを聞いたり、経験年数によって違いがあるのか、どんな人が得意なのか、
参加者からは実務に使えそう、コツがわかった気がする、などの感想があった。

非エンジニア教育

ChatGPTを使いこなすための教育とは。プロンプトエンジニアリングがTips集みたいになってるがもっと体系化できないか

生成AIがどこまでできるのかをどう理解させたり、言い表すか
（IT知識がある場合とない場合で違う説明になる？）

ChatGPTを使って質問の仕方を学生に学習させられるか

どんなコンテンツを作ったら生成AIと親和性が高いか（人が読みにくくても情報がまんべんなく入っている方が良いとか）
→教育じゃなくて仕組みの話かも

プロセス

（他セッションから）
・プロセスで変えないといけないのは何か変わらないのは何か

変わるとして、共通フレームのように、一般で通用する共通するプロセスみたいなものは？何種類かある？

マネジメント

リスクの回避・予防など

技術解出
ファインチューニング済みモデルをWebで触れるようにして、後から怒られたりしない？

逆に海外のファインチューニングモデルを使ってあとで怒られたり？

（他セッションから）
・著作権リスク
・情報漏洩

（他セッション）
ハルシネーションリスク
品質管理