生成AIを活用したプロセス定義レビューにおける

構造化データ形式と文書表現形式の精度比較

第４１年度（２０２５年度）ソフトウェア品質管理研究会
研究コース１ ソフトウェアプロセス評価・改善
２０２６年３月１３日（金）

研 究 員　：池永 直樹（株式会社デンソークリエイト）
主 査　　：田中 桂三（オムロン株式会社）
副 主 査　：白井 保隆（株式会社東芝）
アドバイザー：中森 博晃（ビースラッシュ株式会社）

アジェンダ

- 背景
- 課題
- 先行研究
- 仮説、研究課題
- 提案手法
- 実験
- 実験結果と考察
- まとめ、今後の展望

背景：標準プロセスの役割

- **プロジェクトのQCD目標の達成と安定化**
  - 担当者によるバラツキの低減
  - 不要な作業の抑制

  など

- **自動車業界の状況**
  - 技術潮流の変化から対応が求められる法規・国際／業界標準が増加
    - 機能安全、サイバーセキュリティ、AI品質保証、Automotive SPICE、など
  - 法規・国際／業界標準の要求事項を標準プロセスへ取り込み

短周期で変化するビジネス環境や開発技術の進化に追従し、
競争力を維持・向上するために、
標準プロセスの継続的改善は必要不可欠

背景：標準プロセスの記述やレビューの現状

プロセスの定義・記述及びそのレビューに要する工数は増加傾向

- **原因**
  - 短いサイクルのプロセス改善
  - 標準プロセスの情報量と複雑度が増大

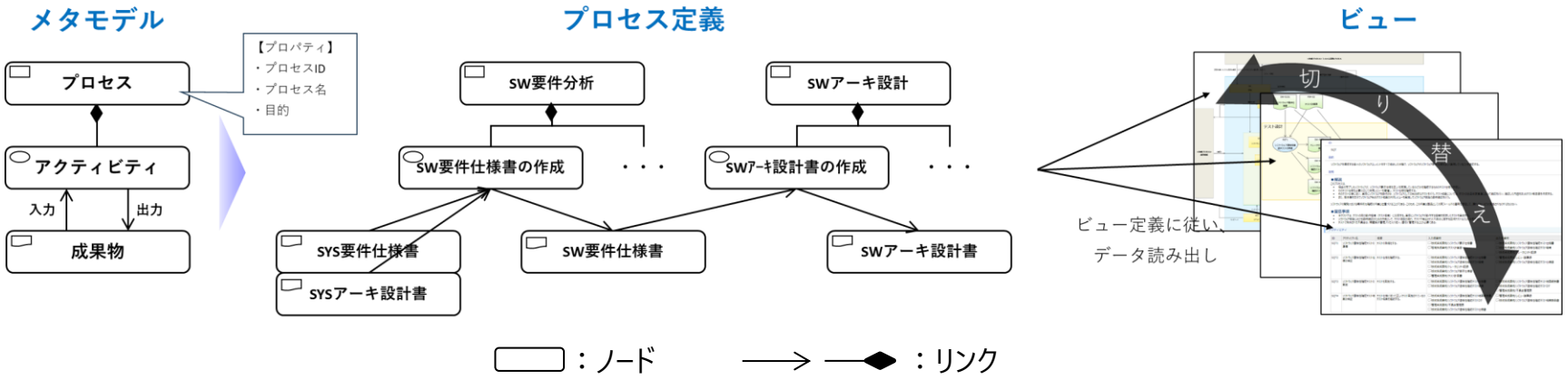
- **工数増加抑制の対策**
  - プロセス記述のデジタル化手法[1] の適用

    狙い
    - プロセス定義の効率化
    - プロセス定義書の品質及び保守性の向上

  - プロセス記述のデジタル化手法の実現に
    システム・ソフトウェア設計ツールである Next Design[2] を使用

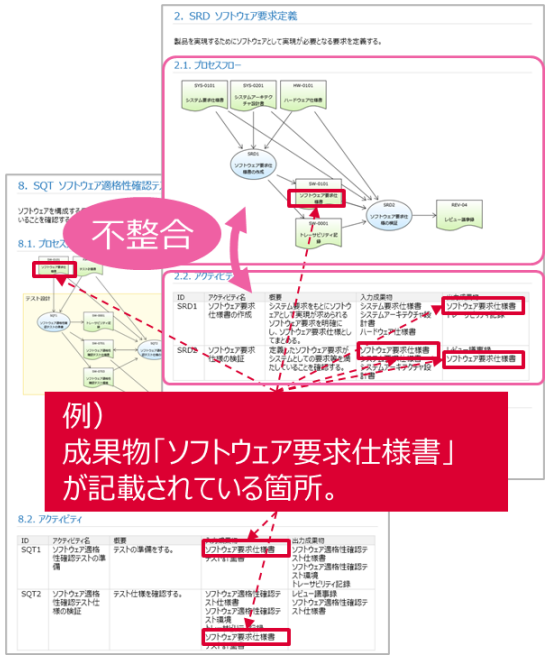
背景：プロセス記述のデジタル化手法（１）

- **概要**
  - プロセス定義のメタモデルに基づいてプロセスを定義
  - 定義されたプロセスは「ノード＋リンク」のグラフ構造として**一元的に保持**
    - ノード　：アクティビティ、成果物、などのプロセス定義要素のインスタンスで、プロパティ（例:ID、名称、目的）を持つ。
    - リンク　：所有（例:プロセスとアクティビティの所有関係）、参照（例:アクティビティと成果物との入出力関係）等のノード間の関係。
  - **データ（プロセス定義）とビュー（画面表示形式）が分離**されているが、ビュー定義（見せ方）に基づいて人が理解しやすい形式に整形して表示



背景：プロセス記述のデジタル化手法（2）

- **効果**
  - プロセス記述の不整合・矛盾の予防
    - ID・名称等のプロパティの不整合
    - 同一プロセス定義に対する異なるビュー間
      (例:ドキュメント形式とPFD)での定義内容の不整合
  - 情報複製に起因する工数の低減
  - 関心ごとに応じたビューの定義



➡ **プロセスの記述やレビュー作業の工数を削減**

- **対象外（効果なし）**
  - プロセス定義内容そのものの読み易さ・表現設計などの表現品質
  - 有効性・適切性・規格類への準拠性・正確性・完全性などの内容品質

➡ **人手での保証が必要**

課題

- **プロセス記述のデジタル化手法では対象外の側面への対策**

有効性・適切性など →	価値判断を要する →	人手を集中させる
表現品質・準拠性・正確性・完全性など →	機械的検証に親和性にあり →	**生成AIを活用する**

- **生成AI活用時の課題**

プロセス記述のデジタル化手法	⇔	**生成AI**
「ノード＋リンク」のグラフ構造を**構造化データ**として保持		**自然言語**処理に強み

課題

自然言語処理を得意とする生成AIに、どのようなデータ形式でプロセス定義内容を入力すればよいか

先行研究

- **データ記述言語（YAML、JSONなど）**
  - 比較的簡単な構造では高精度に解釈できる一方で、深い入れ子や長文では性能が低下する[6]
- **グラフ記述言語（GraphMLなど）**
  - 構造・意味理解タスクにおいて、LLM は一定の性能は示すものの専門モデルに劣り、入力設計やプロンプトの工夫が性能向上に重要である[7]
- **表形式の設計文書**
  - JSONとMarkdownへ変換し見出し・値の関係を明示化することで、GPT系モデルによる欠陥検出の再現率が向上した[8]

仮説

- **プロセス定義書（プロセス記述のデジタル化手法）**
  - プロセス定義要素間に所有・参照などの有向関係を含む**構造化されたデータ**
- **既存知見**
  - **記述量の多さや関係の複雑さ**が増すと精度が低下し得る

    上位概念スキーマ（メタモデル）で構造的・意味的制約を提示することで改善の可能性あり
- **仮説の着眼点**
  - 構造化データ形式：ノードとリンクを一元管理 → 冗長性低
  - 文書表現形式：冗長記述・文脈依存 → 冗長性高・曖昧

仮説

構造化データ形式は、トークン効率が高く、ノード間の関係理解を必要とするレビュー観点において、文書表現形式より高い精度を発揮する

研究課題

構造化データ形式と文書表現形式の性能を比較する

RQ1

ノード間の関係の理解が不要なレビュー観点において、構造化データ形式は文書表現形式と同等以上の精度であるか？

プロセス定義において、生成AIが文書表現形式だけでなく構造化データ形式も適切に扱えるかを確認するため、ノード間の関係理解が不要な単純な検索型のレビュー観点で両形式の精度を比較する。

RQ2

ノード間の関係の理解が必要なレビュー観点において、構造化データ形式の方が文書表現形式より精度が高いか？

プロセス定義の構造を明示的に保持した構造化データ形式の優位性を確認するため、ノード間の関係理解が必要なレビュー観点で両形式の精度を比較する。

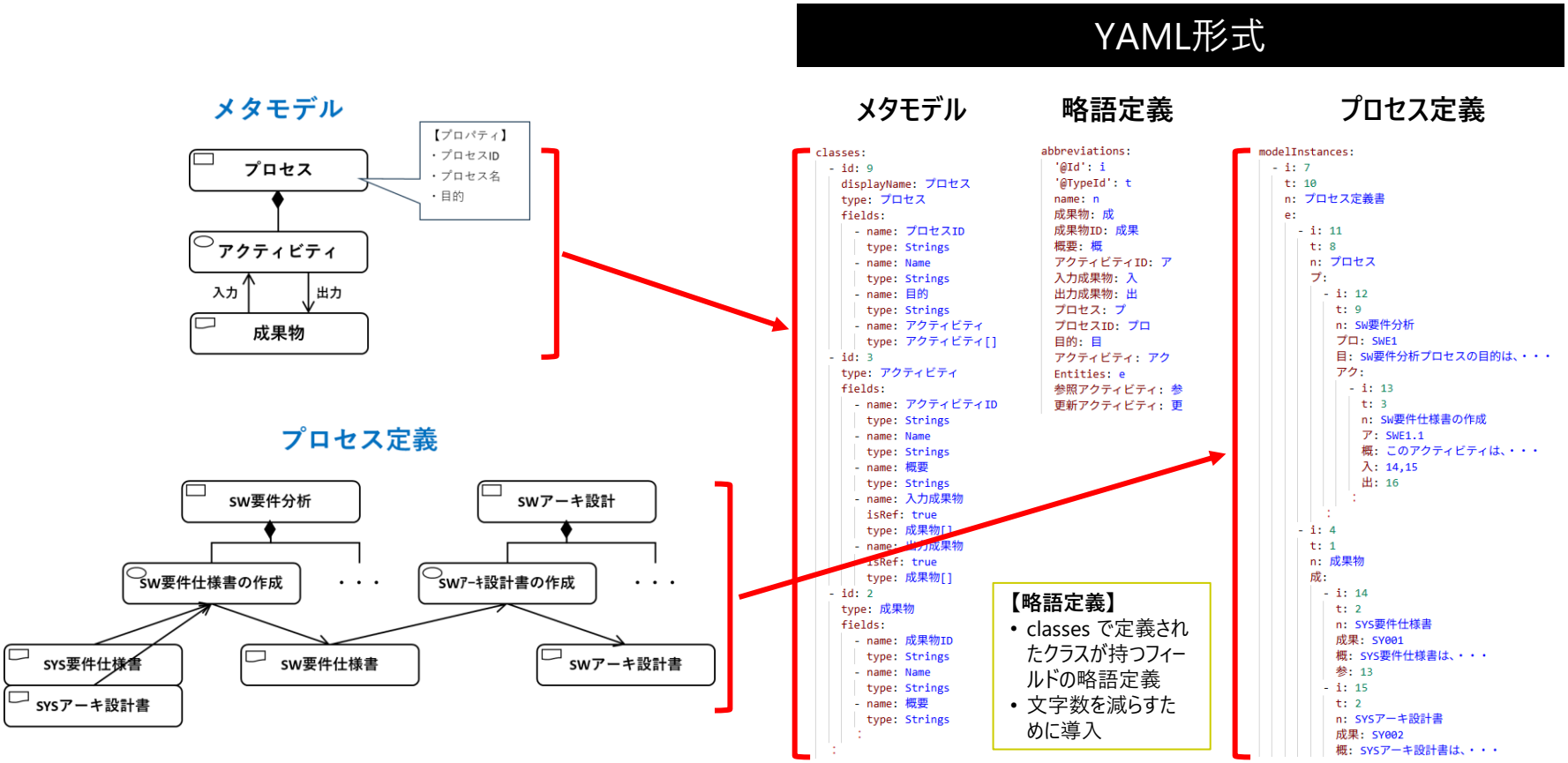
RQ3

構造化データ形式は文書表現形式よりトークン効率が高いか？

トークン数の観点で構造化データ形式の優位性を確認するために、同一のプロセス定義書に対する両形式のトークン数を比較する。

提案手法：構造化データ形式（YAML）

プロセス定義のメタモデル 及び メタモデルに基づいて定義された
プロセスの「ノード＋リンク」のグラフ構造を、YAML形式に変換



提案手法：文書表現形式（Markdown）

ツール画面に表示される人が読める形に整形されたビューの内容をMarkdown形式に変換

ツール画面

プロセス SW要件分析

プロセスID

SWE1

目的

SW要件分析プロセスの目的は、…

アクティビティ

アクティビティID	アクティビティ名
SWE1.1	SW要件仕様書の作成
:	:

アクティビティ SW要件仕様書の作成

アクティビティID

SWE1.1

概要

このアクティビティは、…

入力成果物

成果物ID	成果物名
SY001	SYS要件仕様書
SY002	SYSアーキ設計書

出力成果物

成果物ID	成果物名
SW001	SW要件仕様書

Markdown形式

```
# プロセス定義書
## プロセス
### SW要件分析
#### プロセスID
SWE1
#### 目的
SW要件分析プロセスの目的は、・・・
#### アクティビティ
- SWE1.1: SW要件仕様書の作成
：
#### SW要件仕様書の作成
##### アクティビティID
SWE1.1
##### 概要
このアクティビティは、・・・
##### 入力成果物
- SY001: SYS要件仕様書
- SY002: SYSアーキ設計書
##### 出力成果物
- SW001: SW要件仕様書
：
```

ツール画面

成果物一覧 成果物

成果物

1. 成果物 SYS要件仕様書

成果物ID

SY001

概要

SYS要件仕様書は、…

参照アクティビティ

アクティビティID	アクティビティ名
SWE1.1	SW要件仕様書の作成

更新アクティビティ

(未設定)

2. 成果物 SYSアーキ設計書

成果物ID

SY002

概要

SYSアーキ設計書は、…

参照アクティビティ

アクティビティID	アクティビティ名
SWE1.1	SW要件仕様書の作成

Markdown形式

```
## 成果物
### 成果物
#### SYS要件仕様書
##### 成果物ID
SY001
##### 概要
SYS要件仕様書は、・・・
##### 参照アクティビティ
- SWE1.1: SW要件仕様書の作成
##### 更新アクティビティ
-
#### SYSアーキ設計書
##### 成果物ID
SY002
##### 概要
SYSアーキ設計書は、・・・
##### 参照アクティビティ
- SWE1.1: SW要件仕様書の作成
##### 更新アクティビティ
-
#### SW要件仕様書
##### 成果物ID
SW001
##### 概要
SW要件仕様書は、・・・
##### 参照アクティビティ
- SWE2.1: SWアーキ設計書の作成
##### 更新アクティビティ
1. SW要件仕様書の作成
：
```

実験（1）

- **評価指標**

RQ	評価指標
RQ1	レビュー精度を適合率(Precision)、再現率(Recall)で評価する
RQ2	レビュー精度を適合率(Precision)、再現率(Recall)で評価する
RQ3	トークン効率をプロセス定義のそれぞれの形式でのトークン数で評価する

- **実験対象のプロセス定義書**
  - 筆者の組織内にある３つのプロセス定義書
    - メタモデル、ビュー定義、ならびにプロセス定義内容が異なる

- **利用する生成AI**
  - GPT-5（Azure OpenAI）
    - 本実験ではプロセスの専門知識を生成AIに与えない

実験（2）

- **レビュー観点**
  - RQ1、RQ2 を評価するために、筆者の組織でプロセス定義書をレビューする際に用いられている代表的なレビュー観点をピックアップして使用

RQ	ID	レビュー観点	必要なノード間の関係
1	1-1	活動が定義されているか	なし
	1-2	活動の具体的な実施方法(手順・基準・観点等)が定義されているか	なし
	1-3	成果物が定義されているか	なし
2	2-1	入力成果物が活動間の順序と整合しているか	活動と成果物の入出力関係 活動間の順序関係 活動間の集約関係
	2-2	プロセス・アクティビティなどの各レベルで定義されている入出力成果物が整合しているか	活動間の集約関係
	2-3	活動の開始基準が活動間の順序と整合しているか	活動間の順序関係 活動間の集約関係

実験（3）

- **実験手順**

① プロセス定義書から YAML 形式 及び Markdown 形式の実験データを作成する

② ①で作成した YAML 形式 及び Markdown 形式の実験データのトークン数を、OpenAI 社が提供しているトークナイザー[10]で計測する　} RQ3

③ 筆者がプロセス定義書をレビューして指摘を出す

④ 生成AI に①で作成した YAML 形式 及び Markdown 形式の実験データを入力し、出力を得る

⑤ ③を正答として、④で得た生成AI の出力を評価する　} RQ1,2

実験結果と考察：RQ1の実験結果

RQ1

ノード間の関係の理解が**不要な**レビュー観点において、
構造化データ形式は文書表現形式と同等以上の精度であるか？

ID	試行数	YAML形式(構造化データ)		Markdown形式(文書表現)	
		適合率	再現率	適合率	再現率
1－1	20	0.87	0.57	0.76	0.54
1－2	20	0.67	0.78	0.62	0.78
1－3	20	0.69	0.69	0.49	0.69
合計	**60**	**0.71**	**0.71**	**0.61**	**0.70**

実験結果と考察：RQ1の考察

結果サマリ

	適合率	再現率
YAML	0.71	0.71
Markdown	0.61	0.70

確認できた傾向

形式	傾向
YAML	• **関係のない箇所**を検出することが少ない（誤検出が少ない） • 記載箇所が**分散している場合**に見逃しやすい
Markdown	• **複数の候補がある場合**において**関係のない箇所**を検出する（誤検出しやすい） • **意味的に近い記述**や複数箇所に**分散した記述**を"**広く拾う**"傾向があり、見逃しは減るが正確性に欠ける（誤検出となる）場合がある

適合率はYAMLの方が高く、再現率はほぼ同じだったことから、RQ1は有効、つまり、構造化データ形式は文書表現形式と同等以上の精度である

実験結果と考察：RQ2の実験結果

RQ2

ノード間の関係の理解が**必要な**レビュー観点において、
構造化データ形式の方が文書表現形式より精度が高いか？

ID	試行数	YAML形式(構造化データ)		Markdown形式(文書表現)	
		適合率	再現率	適合率	再現率
２－１	２０	０.８９	０.７７	０.８８	０.６７
２－２	１７	０.９７	１.００	０.９３	０.９３
２－３	１１	０.８９	０.５２	０.７０	０.４５
合計	**４８**	**０.９１**	**０.７６**	**０.８６**	**０.６８**

実験結果と考察：RQ2の考察

- **結果サマリ**

	適合率	再現率
YAML	0.91	0.76
Markdown	0.86	0.68

- **確認できた傾向**

形式	傾向
YAML	• **活動間の順序の理解が必要**な場合の見逃しが少ない • **成果物の入出力・活動の上下階層の理解が必要**な場合の誤検出が少ない
Markdown	• **活動間の順序の理解が必要**な場合の見逃しが多い • **プロセス定義の文書構成（関連語句の出現順）**により誤検出・見逃しが生じる • 一方で、冗長記述がプラスに働き**冗長箇所を拾う**ことで再現率がYAML形式を上回るケースがあった

適合率・再現率ともにYAMLの方が高かったことから、RQ2は有効、つまり、構造化データ形式の方が文書表現形式より精度が高い

実験結果と考察：RQ3の実験結果

RQ3　構造化データ形式は文書表現形式よりトークン効率が高いか？

プロセス定義書	YAML形式 (構造化データ)	Markdown形式 (文書表現)	比率*
＃１	７０,１８９	７６,５７１	９２％
＃２	１０４,０７４	１９３,４５０	５４％
＃３	１４５,６７２	１５０,９９５	９６％
合計	**３１９,９３５**	**４２１,０１６**	**７６％**

*Markdown形式に対するYAML形式の比率

３件ともYAMLの方がMarkdownよりトークン数が少なく
合計で２４％の削減が確認できたため、RQ3は有効、
つまり、構造化データ形式は文書表現形式よりトークン効率が高い

実験結果と考察：RQ3の考察

- **プロセス定義書ごとの YAML vs Markdown の差の違いの考察**
  - プロセス定義書＃１（比率：９２％）
    - 冗長化の対象である**参照関係が少ない**メタモデル

    ➡ 差が小さくなった
  - プロセス定義書＃２（比率：５４％）
    - 活動名に「プロセス名_アクティビティ名_タスク名」で連結する命名規則を採用
    - Markdown形式ではこの長い活動名が**複数箇所に繰り返し登場**

    ➡ 差が大きくなった
  - プロセス定義書＃３（比率：９６％）
    - 使用されていないプロセス・アクティビティなどのプロパティが多数あり
    - これらがYAML形式には含まれていたがMarkdown形式に含まれおらず、Markdown形式の**冗長化を相殺**

    ➡ 差が小さくなった

実験結果と考察：妥当性への脅威

- **実験で分かった事実**

  プロセス定義書の**特性**によって両形式の指摘検出力が変わる。

  【特性】

  - 分散された記述
  - 順序性定義の有無
  - 登場頻度

  など

- **妥当性への脅威**

  これらの特性の偏りによっては構造化データ形式（YAML）の優位性が下がる可能性があるが、それらの特性が与える影響を十分検証できていない

まとめ

背景

プロセス記述のデジタル化手法の実現方法である
“Next Design による**「ノード＋リンク」のグラフ構造**でプロセス定義を保持する仕組み”
を前提にプロセス定義レビューへの生成AIの活用を考えている

課題

自然言語を得意とする生成AIに、どのようなデータ形式でプロセス定義内容を入力すればよいか

実験

構造化データ形式として YAML 、文書表現形式として Markdown
の二つの形式の「レビュー精度」と「トークン効率」を比較

結論

構造化データ形式（YAML）は文書表現形式（Markdown）に比べ、

- ノード間の関係理解が**不要**なレビュー観点では同等以上
- ノード間の関係理解が**必要**なレビュー観点ではより高精度
- トークン数も少ない

ことが確認でき、妥当性への脅威は残るが
プロセス定義書レビューには**構造化データ形式（YAML）を採用することが有効**と判断

今後の展望

今回の実験では妥当性への脅威が残っているため、以下に取り組む。

1. 生成AIでのレビュー精度を考慮した
   メタモデルやプロセス定義の記述ルールの定義

2. プロセス定義書の特性（分散記述や順序性定義有無など）
   によるレビュー精度への影響の測定

謝辞

論文作成にあたり、ご指導・ご協力いただいた
皆様に深くお礼申し上げます。

研究コース１
　指導員の皆様
　研究員の皆様

株式会社デンソークリエイト イオタ推進部 の皆様

参考文献

[1] 池永直樹, DXアプローチによるプロセス記述 -プロセス記述の効率化， 品質および使用性向上に向けた取り組み-, 4th NSPICE Conference
[2] 株式会社デンソークリエイト， システム・ソフトウェア設計ツール Next Design, https://www.nextdesign.app/
[3] 清水吉男, PFDの基本, AFFORDD カンファレンス資料, 2018
[4] 西本 真由, 生成AIを活用したレビュー効率の向上と有効性の検証, SPI Japan 2025
[5] Lucas Koch, Towards AI-ready Software Engineering Bridging Research and Industry, 10th Korean SPICE Network International Conference x 2nd Asian SPICE Conference
[6] Zhouhong Gu et al., StrucText-Eval: Evaluating Large Language Model's Reasoning Ability in Structure-Rich Text, arXiv:2406.10621
[7] Jiayan Guo et al., GPT4Graph: Can Large Language Models Understand Graph Structured Data? An Empirical Evaluation and Benchmarking, arXiv:2305.15066
[8] Takasaburo Fukuda et al., Development of Automated Software Design Document Review Methods Using Large Language Models, SANER 2025
[9] https://jsontoyamlconverter.com/yaml-vs-json/performance/
[10] https://platform.openai.com/tokenizer