中間表現とAIエージェントを用いた
フロントエンド開発における反復修正の負荷軽減

S2310287 山本慎也

研究テーマ

背景・課題

AIエージェントを用いたフロントエンド開発における反復修正の負荷・従来のビジュアルエディタによるハードコーディング

目的

ソースコードとGUI操作が分離したビジュアルエディタの確立により、課題の解決を目指す

背景:フロントエンド開発

LLM主導開発の普及と限界

テキストプロンプトによるコード生成が普及

しかし、UIの「空間的・視覚的な微調整」の段階になると、自然言語での指示は曖昧になりやすく、意図しないレイアウト崩れを誘発する

GUI操作の必要性

視覚的な微調整には、GUIでの直接操作が効果的である

課題:AIを使ったフロントエンド開発

意図しないレイアウト変更

図1：元のWebサイト

図2：AIを使ってUIを微調整したWebサイト

図1

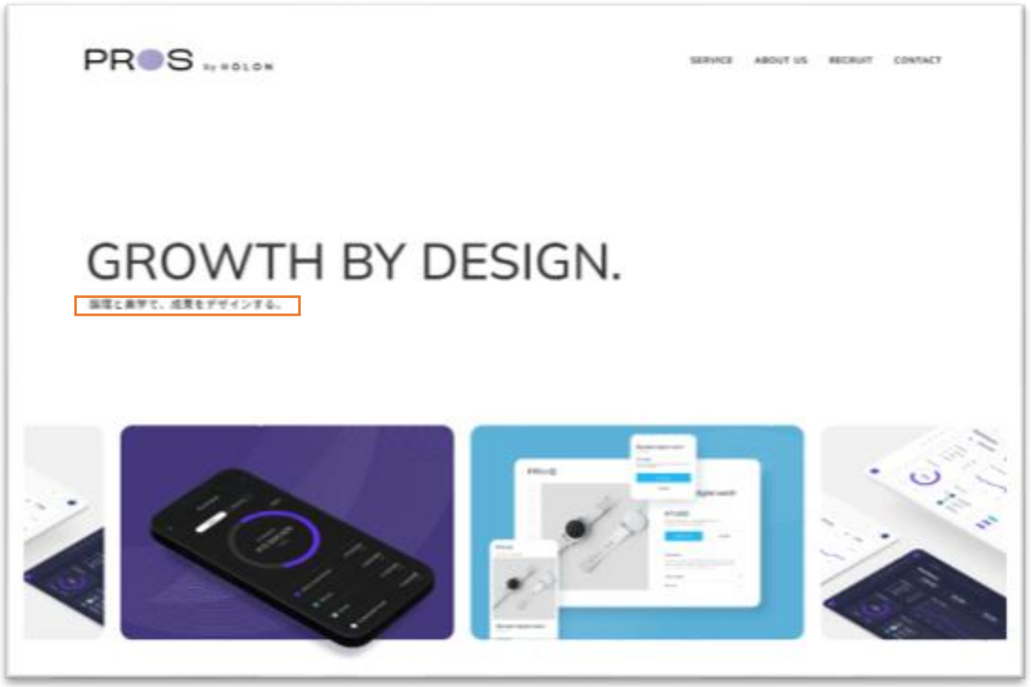

図2

課題:従来のビジュアルエディタ

従来のビジュアルエディタはコードと密結合

- GUI操作がソースコードによる制約を受ける
- 画面上の操作がそのままコードに組み込まれ構造が無視される

課題:密結合による影響

画面上の操作がコードによる制約を受ける

```
<div class="card bg-white w-64 h-64">
  <button class="btn bg-blue-500" style="margin-top: 104px; margin-left: 80px;">
    Click Me
  </button>
</div>
```

コードによる制約

「ボタンが移動した」という事象を、ボタン自身のプロパティ（marginやtransform）に翻訳することしかできない。親要素をいじるという権限を持たない。

プロパティ操作の限界

「要素を動かす＝その要素の margin か position をいじる」というルールに縛られ、自由な開発ができない。

課題:密結合による影響

画面上の操作がそのままコードに組み込まれる

```
<div class="nav-bar" style="display: flex; position: relative; width: 100%;">
  <div class="logo">Logo</div>

  <button class="login-btn" style="position: absolute; left: 632px; top: 10px;">
    Login
  </button>
</div>
```

レイアウトの無視

親要素が display: flexにもかかわらず、そのルールを無視してposition: absoluteという強権的な命令で上書きしている

レシポンシブ非対応

物理的な座標をハードコードしたことで、画面幅の変化に対応できない

提案システム：操作とコードの分離

- Webサイトの要素をキャンバス上に抽出し、GUI操作を可能にする
- GUI操作で発生した視覚的移動要求のみを抽出
- GUI操作はコードの制約を受けない

```
{
  "targetId": "login-btn",
  "visualDelta": {
    "action": "translate",
    "dx": 600,
    "dy": 0
  },
  "context": {
    "parentDisplay": "flex",
    "parentDirection": "row"
  }
}
```

提案システム：AIによる推論

- 発生した視覚的移動要求を実現する上で、「現状のコードと適合する最適な実装をせよ」という旨のプロンプトを生成
- プロンプトとソースコードに基づいて、AIが修正を推論する
- ハードコーディングを避け、構造に適した実装がされる

評価設定パターン

ターゲティングによって評価設定は変わる

（HCI寄り）

A. テキストプロンプト表現の限界

B. 試行錯誤ループによる負荷の軽減

（SE寄り）

C. 従来GUIの密結合による構造破綻の阻止

D. 中間表現設計の最適化

A.テキストプロンプト表現の限界

主張

AIコーディングは便利だが、UIの空間的な微調整になるとテキストで伝えるのが難しい

目的

ビジュアルエディタ化による直接操作を可能にする

比較対象

CursorやChatGPTなどのテキストベースのAIコーディング

評価手法

- **タスク**: モックと完成間近のUIを用意し、モックに合わせた「ボタンを少し右下へ」「ここの余白を少し広げて」といった微調整を数か所行う
- **指標**: タスク完了までの「プロンプト入力回数」、「最終的なデザインモックとの一致度」

B.試行錯誤のループによる負荷

主張

人間が視覚的な結果を得るために、脳内で構造を考え、コードを書き換え、ブラウザをリロードする作業は負荷が高い

目的

ビジュアルエディタ化の直接操作によるダイレクトな反映を可能にする

比較対象

VS Codeとブラウザを用いた従来の手書きコーディング

評価手法

- **タスク**: モックと完成前のUIを用意し、モックに合わせた調整を行って作り上げる
- **指標**: 「認知的負荷」のアンケートスコア、エディタとブラウザの画面切り替え回数、タスク完了時間

C.従来GUIの密結合による構造破綻

主張

従来のビジュアルエディタは操作を直接絶対的なコードに変換するため、構造が破壊されやすい

目的

操作と現状の分離が必要

比較対象

Webflowなどの密結合型ビジュアルエディタ

評価手法

- **タスク**: モックと完成前のUIを用意し、モックに合わせた調整を行って作り上げる
- **指標**: LLMによる完成コードの静的解析、ハードコーディング検出

D.中間表現設計の最適化

主張

キャンバスの状態（JSON）をそのままLLMに投げても、情報が多すぎたり文脈が欠如したりして推論の質が下がる

目的

LLMが正しく意図を解釈できる最適なスキーマの設計を検討する

比較対象

自作システム内の異なるJSONの構造同士の比較

評価手法

- **タスク**: 様々なUI変更タスクを各JSONフォーマットでLLMに100回ずつ推論させる
- **指標**: コンパイル成功率、構造維持率、APIのトークン消費量

対象ペルソナ

本研究の対象者

- 認知的負荷の高い開発者（B,D

視覚的要求を達成するために、修正方法を考え、コードを修正してブラウザをリロードする「試行錯誤のループ」に時間を奪われている

- デザインと実装の境界にいる人材 (B,C,D

自分でUIを微調整したいが、構造を壊すことを恐れて直接コードを触れない

- **プログラム知識のない個人開発者 (A,B,C,D**

AIエージェントにより大まかな機能や画面レイアウトを生成することには成功する。しかし、「ボタンをもっと右に寄せて」「この余白を少し狭くして」といった視覚的・空間的な微調整の段階に入ると、テキストプロンプトで意図を正確に伝えることが困難