生成 AI を活用した SynViz S2 の計画作成・編集機能の提案

(Proposal for Generative AI–Driven Planning and Editing Functions for SynViz S2)

門司 太郎 Taro Monji
鈴木 秀明 Hideaki Suzuki
星 魁人 Kaito Hoshi
内海 宏律 Hironori Utsumi
佐藤 悠樹 Yuki Sato

株式会社日立ソリューションズ東日本（以下，HSE）では自社製品で生成 AI 活用する取り組みを進めている。その一つとして，工程管理・プロジェクト管理システム SynViz S2 に AI チャットボットを追加し，計画の新規作成や編集を対話的に実行できるプロトタイプを試作した。チャットでの自然な会話を通じて計画を作成・編集できるようになれば，エンドユーザーの業務効率とユーザビリティを大きく向上できる。一方，チャットボットの実現にあたっては，ユーザーの指示を適切に解釈し，SynViz S2 の API コールに正確に変換する必要がある。生成 AI（Claude Sonnet）から外部ツールを呼び出す仕組みとして Tool use が既に存在するが，SynViz S2 のような大規模な API の場合，これをそのまま適用しても十分な精度が得られない。そこでまず実行に必要な API の種類を絞り込んでから API に変換するという 2 段階の API 実行手法を考案した。さらにプロンプト最適化の手法を組み合わせることにより精度を高めた。

1. はじめに

生成 AI はあたかも人を相手にしているような自然な対話を実現する能力が驚きと衝撃をもって受けとめられ，急速に普及した。ChatGPT のようなチャットボットに加え，チャットを通じてソフトウェアを操作する仕組みや，複雑なタスクを自律的に実行する AI エージェントも登場し，業務利用も広がってきている。サービスやアプリケーションを提供するベンダーにとっても，生成 AI との連携や活用を通じてエンドユーザーの業務の効率化や自動化を実現することは，競争力維持のために不可欠な課題となっている。

こうした状況から，HSE でも自社製品への生成 AI 活用を進めている。本稿では， SynViz S2 に AI チャットボットを導入し，チャット画面から計画を編集できるようにする取り組みについて紹介する。

SynViz S2 は直感的な操作性を備える一方，多機能・高機能であるため使いこなすには一定の習熟が必要である。使い方を熟知していないユーザーでもチャットでの自然な対話を通じて高度な操作ができる仕組みを導入できれば，利便性とユーザー体験の向上を実現できる。

2. SynViz S2 の AI チャットボット

AIチャットボットの画面例を図1に示す。チャット画面はガントチャートの右側に表示される。ユーザーが下部の入力欄に指示を書くと，指示に応じてガントチャート画面が更新される。指示は何度も書くことができるので，複数回に分けて段階的に計画を仕上げていくことができる。会話履歴はチャット画面の上部に表示される。

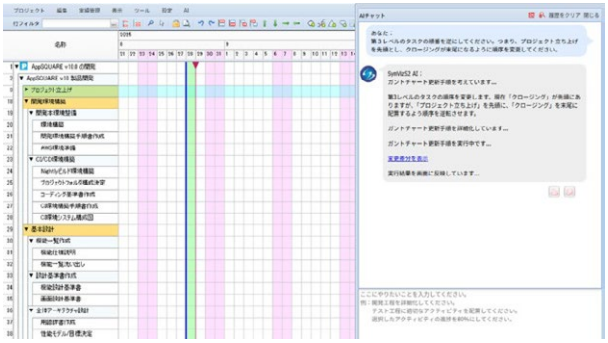


図1 AIチャットボット画面

3. チャットボットによる計画編集の課題

3.1 高精度な API への変換

チャット画面で指示を入力し，計画の新規作成や編集を実行すると，いずれの場合も最終的には一連の Web API コールに変換され，実行される。API が一つでもエラーになると操作全体が失敗する。API 変換は本システムの基盤であり，ほぼ 100%の変換精度を実現する必要がある。

本システムで使用した Claude Sonnet 4.5[1)]は，外部ツール（今回の場合は SynViz S2 の API）を呼び出す仕組みとして Tool use[2)]をサポートしている。少数の API を実行するのであればこれをそのまま使えばよいが，

SynViz S2 の API は大規模なため単純に Tool use を適用しても精度が上がらない。

SynViz の Web API のリソース数と API 数を表 1 に示す。API の総数は 295 個，そのうち本システムで使用するガントチャート関連の API だけでも 143 個にもなる。各リソースの属性も多く，全体として巨大な仕様となっている。

表 1 SynViz S2 API のリソース数と API 数

APIセット	リソース数	API数
SynViz S2 全体	143	295
ガントチャート関連のみ	65	143

ユーザーの指示を API に変換すると，多くの場合さまざまな API を呼び出すことになる。多数の API から必要な API を選択し，適切な順番で呼び出すようにすること，それをほぼ 100%の高い精度で変換することが本システムの最重要課題である。

3.2 ユーザーの指示の的確な理解

高精度な API 変換に次いで重要なのは，ユーザーが入力したテキストから意図を的確に解釈することである。解釈能力が低いとユーザーの期待に沿わない編集が実行され，機能として成り立たない。

ユーザーは，たとえばタスクを 1 個追加するような簡単な操作であればチャットを使うまでもなくガントチャートを直接操作する。チャットを使うのは，「担当者が A さんに設定されているアクティビティをすべて B さんに変更してください」「タスク A を詳細化し 9 月末にすべての作業が完了するように計画を立ててください」など，編集が煩雑な場合や高度な編集が必要なケースである。チャットボットはこうした複雑あるいは曖昧性のある入力からユーザーの意図を的確に理解し，必要に応じて補完，場合によっては聞き返しをする必要がある。

4. 課題の解決

4.1 2 フェーズの API 変換

一般的に生成 AI に与える情報量が多くなると応答の精度も低下する [3]。API の仕様書は PDF で 700 ページ超にもなるため，これをそのまま生成 AI に与えるとプロンプトが長大になり API 変換の精度が低下する。また仮に変換できたとしても，生成 AI への大量の入力により応答性能の低下やコストがかさむといった問題もある。

この課題に対処するため，API 変換を API フィルタリングと API コールの詳細情報の生成（以下，コールスペック生成）の 2 段階で実行する方式を考案した（図 2）。

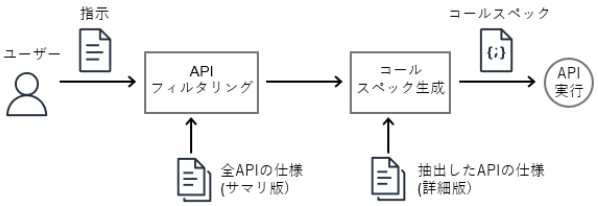


図 2 API 変換のワークフロー

最初の API フィルタリングフェーズでは，全 API の中からユーザーの指示を遂行するのに必要な API を抽出する。次のコールスペック生成フェーズでは，フィルタリング後の少数の API の実行情報を正確に決定する。

フィルタリングには全 API の仕様を生成 AI に与える必要があるが，API コールの詳細までは決めないため API 仕様の記述内容を大幅に簡略化できる。これにより大量の API があっても高速に API を絞り込める。

一方，コールスペック生成時には詳細な API 仕様が必要になるが，少数の API の仕様のみ与えれば良いためサイズを大幅に抑えることができる。この仕組みにより多数の API があっても高い精度での実行が可能になる。

4.2 プロンプトの構造化

生成 AI モデルはプロジェクト管理に関する一般的知識は持っているものの，SynViz S2 の知識は持っていない。この状態で SynViz S2 の API 仕様を与えてもユーザーの指示を適切に API に変換することはできない。まず SynViz S2 に関する全般的な知識をコンテキストに含め，その上で API 仕様を与える必要がある。

また編集対象の計画のデータも与える必要がある。現状の計画を知らないと，ユーザーからの編集指示に対しどのように変更すべきか特定できない。

このように生成 AI にはユーザーの指示に加えて複数の異なる情報を整理して提供する必要がある。Anthropic のプロンプトエンジニアリングガイドに従い，表 2 のとおり XML タグを使って構造化する [4] ことで生成 AI のプロンプト理解を助け，出力の質を向上できるようにした。

表 2 プロンプト中の XML タグ

XMLタグ	概要
<chart_data>	現在の計画データをJSON形式で記述したもの
<chart_data_schema>	<chart_data>のスキーマ定義（JSON Schema形式）
<gantt_chart_specification>	SynViz S2のガントチャートに関する全般的な説明
<tool_usage_rules>	ツール全般に関する規則（例：日付はISO8601形式）
<instructions>	計画更新手順の説明

4.3 プロンプトの最適化

プロンプトの構造化で構造を明確にしたうえで，ユーザーの意図に対する理解度をさらに上げるために以下で説明する 3 つのプロンプト最適化を行った。

(1) CoT プロンプティング

CoT（Chain of Thought）プロンプティング[5]は，複雑なタスクを実行するときに段階的な思考を促すことで回答精度を高める手法である。「ステップバイステップで考えてください」といった単純な指示でも効果があるが，本システムでは具体的な思考ステップを明示するガイド付きプロンプトという手法を用いている。今回設定した思考ステップの構成を表 3 に示す。

表 3 CoT のガイド付きプロンプトの思考ステップ

ステップ	見出し
1	**要求の理解と分類**
1.1	明快度評価
1.2	処理方針の決定
1.3	操作分類
2	**データ取得と検証**
2.1	必要データの特定
2.2	リスク評価と検証レベル
2.3	エラーハンドリング
3	**実行と応答**
3.1	応答メッセージの作成
3.2	ツール実行判断

各ステップで細かく指示を記述しているが，一例として「2.1 必要データの特定」の記述を図 3 に示す。

```
### ステップ2: データ取得と検証

#### 2.1 必要データの特定
<thinking>内で以下を実行：

1. **対象の特定**
   - 操作対象オブジェクト（Task/Activity/Milestone等）
   - 対象ID（既存の場合）
   - 親子関係・依存関係のあるオブジェクト

2. **データ抽出**
   - <chart_data>から関連データを全抽出
   - 参照関係を追跡（ParentItemID, TaskID, ObjectID等）

3. **スキーマ確認**
   - 必須パラメータをリストアップ
   - 条件付き必須パラメータを確認
   - データ型・制約条件を確認
```

図 3 CoT プロンプトの例

なお, Claude では思考過程を出力しないと思考がおこなわれない[5]ため<thinking>タグに出力している。

(2) 算術計算のツール化

生成 AI は厳密な数値演算は得意ではなく，もっともらしいが正確ではない値を出力することがある。本システムでは，アクティビティの期間を営業日ベースで計算する処理を SynViz S2 の API と同様にツール化し，これによって正確な日付計算を実現した。

(3) メタプロンプティング

メタプロンプティングとは生成 AI に自分のプロンプトを改善させる手法である。たとえば前述した<thinking>タグ出力は精度向上には役立つものの，出力が増える分，応答時間がかかるという問題がある。その一方で出力量を抑えると思考が不十分になる。精度を落とさずに出力量を減らすため，Claude 自体に推論能力を落とさずに<thinking>タグの出力を減らすように指示し，プロンプトを改善した。

5. 評価

5.1 評価方法

プロンプト最適化を実装後，ユーザー入力を意図どおりに解釈し，エラーなく API を実行できたかどうかを実機で評価した。

評価にあたり，ユーザー入力を内容の複雑度の観点で 4 段階に分けた（表 4）。次にレベルごとに評価用プロンプトを作成した。各レベルのプロンプト例を表 5 に示す。

表 4 プロンプトの複雑度

複雑度	説明
1	単体オブジェクトの操作（単一APIの実行）
2	単一リソース，複数オブジェクトの操作（単一APIの複数実行）
3	複数リソース，複数オブジェクトの操作（複数APIの複数実行）
4	複雑な操作，状況に応じた創造的な編集（欠落情報の補完，聞き返し，要望・要件を理解して提案）

表 5 評価用プロンプトの例

複雑度	プロンプト例
1	「テスト設計レビュー」タスクに「レビュー完了」というマイルストーンを 2025 年 10 月 2 日に追加してください
2	レベル 2～4 の「テストケース設計」タスクを削除してください。
3	メンバーF の担当作業を全てメンバーG に変更し，テストスクリプトの修正を 11 月 12 日に変更してください
4	山田さんが急遽 2 週間の休暇を取ることになりました。対応してください。

5.2 評価結果

複雑度に応じて各レベル 5 種類（レベル 4 のみ 9 種類）のプロンプトを 5 回ずつ実行した。レベル別の目標値と成功率を表 6 に示す。目標値はレベル 1 と 2 が 90%，3 が 80%とした。レベル 4 は目標値を設定せず，今後の精度向上のため測定のみ実施した。結果，レベル 3 以下については 100%に近い成功率で当初の目標を達成した。レベル 4 は 40%と低い値だが, 原因については後述する。

表 6 複雑度別の実行成功率

複雑度	プロンプト数	目標値	成功率
1	5	90%	100%
2	5	90%	100%
3	5	80%	96%
4	9	-	40%

5.3 結果の考察

(1) 複雑度レベル 1～3

チャットボット機能の基盤となるレベル 3 以下の API 変換についてはほぼ 100%の精度となり目標値を達成した。本稿で説明した API 変換の手法が有効に機能したと判断できる。

さらに 100%に近づけるには，自己反省（self-reflection）メカニズムを組み込み，失敗時に失敗理由も入力として再試行する手法 [6]もある。再試行に時間がかかるためチャットでは導入しにくいが，再試行回数を最大 1,2 回程度に抑えるなどすれば導入できる可能性はある。

(2) 複雑度レベル 4

ユーザーの指示が複雑・曖昧なケースを想定したレベル 4 では成功率が 40%にとどまった。評価用プロンプトに解釈が難しく成功率 0 のものが 2 つあったこと，欠落情報の推測や聞き返しを伴うプロンプトがあり本稿時点では対応が十分ではないことが大きな要因となっている。

編集量が多くなると，解釈の間違いや指示の一部しか実行しないなどの傾向もみられた。これらについては今後の精度向上が必要である。SynViz S2 は Undo 機能があり，実行結果が意図どおりではない場合，ボタン一つで元に戻せる。また指示の一部しか実行されなかった場合，ユーザーが残りの編集を追加で指示することで回避できる。プロンプト解釈能力だけでなくアプリケーション全体での対応を進めることで実質的な成功率を上げていくことができると考える。

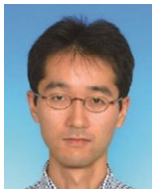
6. おわりに

本稿では，SynViz S2 に AI チャットボットを組み込む取り組みを紹介した。実用的なチャットボットを実現するには簡潔なユーザー入力に対し，的確にユーザーの意図を解釈し，正確に API に変換する必要がある。独自の API 実行手法と Anthropic のプロンプトエンジニアリングガイドに沿ったプロンプトの構造化と最適化により精度を高め実用化できることを確認した。

今後はさらなる精度向上に加えて，編集機能以外に工程状況の分析とレポート生成，リスク検知など，より幅広くプロジェクト管理業務をサポートする予定である。

参考文献

1) Claude Sonnet 4.5 (2025/12/4 閲覧), https://www.anthropic.com/claude/sonnet
2) Claude でのツール使用 (2025/12/4 閲覧), https://docs.anthropic.com/ja/docs/agents-and-tools/tool-use/overview
3) Context Rot: How Increasing Input Tokens Impacts LLM Performance (2025/12/4 閲覧), https://research.trychroma.com/context-rot
4) XML タグを使用してプロンプトを構造化する (2025/12/4 閲覧), https://platform.claude.com/docs/ja/build-with-claude/prompt-engineering/use-xml-tags
5) Claude に考えさせる（思考連鎖プロンプト）でパフォーマンスを向上させる (2025/12/4 閲覧), https://platform.claude.com/docs/ja/build-with-claude/prompt-engineering/chain-of-thought
6) Yu Du, et al.: AnyTool: Self-Reflective, Hierarchical Agents for Large-Scale API Calls, Proceedings of the 41st International Conference on Machine Learning (ICML 2024). PMLR, Vol. 235, pp. 11812–11829 (2024), AnyTool: Self-Reflective, Hierarchical Agents for Large-Scale API Calls



門司 太郎　　1994 年入社
研究開発部
自社サービスの研究開発

鈴木 秀明　　2019 年入社
研究開発部
自社サービスの研究開発

星 魁人　　2020 年入社
研究開発部
自社サービスと生成 AI 連携したシステム開発

内海 宏律　　2006 年入社
サービス基盤開発部
自社パッケージソフトウェアの開発，生成 AI の活用・適用研究

佐藤 悠樹　　2025 年入社
研究開発部
生成 AI の応用研究