GPT-4o を使った iStar モデル作成支援の研究

M2023SE003 平林義健
指導教員：佐伯元司

1　はじめに

要求の獲得を目的とした要求分析手法の一つであるゴール指向要求分析（Goal-Oriented Requirements Analysis, GORA）は分解や詳細化のためのガイドラインや方法論が不足しているため，初心者がモデルを作成するのは非常に難しく，低品質なゴールモデルを作成してしまう可能性がある．そこで近年では，人工知能技術や生成 AI を活用して，初心者が GORA を行う際の支援を目指す研究が増えている．

先行研究では，KAOS モデリングを支援するために自然言語処理（NLP）を用いたチャットボットを要求工学初心者の支援ツールとして使用する研究 [1]．や ChatGPT が GRL(Goal-oriented Requirement Language) を生成する能力について調査した研究 [2] などが行われている．

我々の知る限り，これまでの研究において，AI が生成した iStar モデルの品質は依然として低いままである．したがって，本研究では，GORA の中でも特に iStar[3] に焦点を当て，GPT-4o[4] を活用して高品質な iStar モデルを自動生成することを目的とした．なお，本論文内では GPT-4o を「GPT」と統一して記述する．本研究の目的を達成するためには，以下の 2 つを満たす必要がある．

- 構文エラーの無い iStar モデルの生成
- 要素記述の品質が高い iStar モデルの生成

2　アプローチ

Chen らの研究 [2] で強調されているように，生成 AI は GORA に関連する一定の知識を持っているものの，AI が生成するモデルには構文エラーが含まれるリスクや，構文的には正しいが要素記述が最適ではない低品質なモデルを生成するリスクがある．

この問題を解決するために iStar モデル生成を大きく 2 つのステップに分けて生成するアプローチを提案する．この 2 つのステップについての内容を以下に示し，また図 1 に 2 つのステップの概要を示す．

1. 生成する iStar モデルが可能な限り構文エラーを含まないようなプロンプト (モデル生成プロンプト) の設計
2. モデル生成プロンプトを用いて生成された iStar モデルの低品質箇所を修正することによるモデルの高品質化

3　モデル生成プロンプト

本研究では GPT に iStar モデルを生成させるための最初のプロンプトを「モデル生成プロンプト」と呼ぶ．このモデル生成プロンプトは (1)Context，(2)Single Sentence，(3)Domain Paragraph，(4)Syntax Description，(5)iStar モデルの生成順序+構文規則の 5 つの要素で構成される．

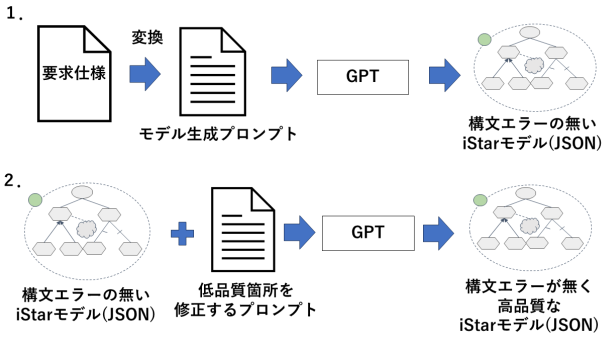

図 1　アプローチの概要

モデル生成プロンプトは，Chen らの研究 [2] に基づき，基本的には (2)Signle Sentence，(3)Domain Paragraph，(4)Syntax Description に分けて定義されており，要求仕様のアクタ，詳しい要求，iStar の記述例などが含まれる．本研究では，これら 3 つのプロンプトに加えて，(1)Context と，(5)iStar モデルの生成順序+構文規則を記述する．(1)Context はプロンプトの初めに記述し，事前条件や GPT の役割等を記述することで GPT の回答性能を向上させる文章である．

(5)iStar モデルの生成順序+構文説明の iStar モデルの生成順序は GPT が iStar 要素生成を各段階ごとに行えるようにプロンプトを定義し，GPT に入力する．また，生成順序ごとの構文規則を与えることで，GPT が生成順序ごとの構文規則を理解しやすいようにしている．これはプロンプトパターンの CoT(Chain-of-Thought) に該当する．以下に生成順序を示す．この生成順序は一般的な iStar モデルの SD(Strategic Dependency) モデルから SR(Strategic Rationale) モデルへの作成手順を参考にしている [5]．

1. アクタの生成
2. 依存関係のゴールを生成
3. アクタ内のゴールを生成
4. アクタ内のタスクを生成
5. アクタ内のリソースを生成
6. アクタ内のクオリティを生成
7. アクタ内の根ゴールの生成
8. 依存関係の依存元・依存先をアクタ内へ変更

4　iStar モデルの高品質化

GPT によって生成された iStar モデルは，構文エラーは無いものの，低品質なモデルを生成する可能性がある．特に，要素記述の品質は入力プロンプトに影響されやすく，漏れ，曖昧さ，一貫性の欠如が生じることがある．そ

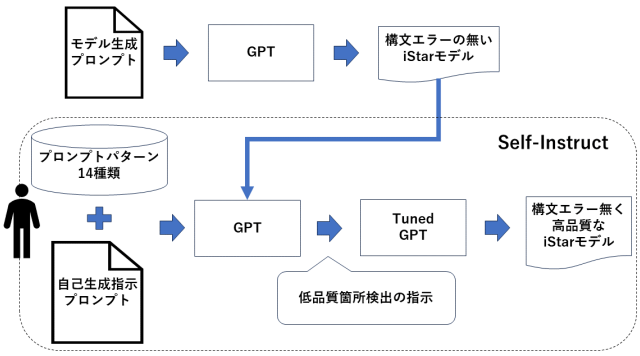

図 2　本研究の流れ

こで，本研究では，iStar モデル内の要素記述の低品質箇所の修正により，モデル生成プロンプトで生成した iStar モデルの高品質化を行う．

要素記述の低品質箇所修正には，アルゴリズムを使って解析することが困難なこと，モデルの意味を考慮する必要があることなどから本研究では引き続き GPT を活用する．しかし，iStar モデルの低品質箇所の定義は曖昧であり，様々な低品質箇所があることが考えられることから iStar モデルの低品質箇所修正に特化した GPT が必要であり，ファインチューニングが必要である．しかし，GPT のファインチューニングには大規模な低品質な iStar モデルと修正プロンプトが必要である．そこで本研究では，Wang らが提案した Self-Instruct[6] を活用し，GPT 自身が低品質箇所を特定するためのプロンプトを自己生成し，検出・修正を行う手法を取る．

Self-Instruct は GPT に人間があらかじめ作成したプロンプトパターン（指示+入力とその出力の組み合わせ）を与え，これを基に GPT が新たな指示と対応する入力・出力例を生成する手法である．これにより，GPT の回答性能が向上する．本研究ではこの手法を応用し，プロンプトパターンを全て iStar モデルの低品質箇所の検出・修正例とすることで実現する．したがって，本研究のプロンプトパターンは以下の 3 つから構成される．

指示 低品質箇所の分類を基に作成した低品質箇所の検出指示

入力 低品質箇所のある iStar モデル (JSON 記述)

出力 低品質箇所の理由と箇所+修正した iStar モデル (JSON 記述)

この Self-Instruct を含む本研究の流れを図 2 に示す．なお，モデル生成プロンプトによる iStar モデルの生成から Self-Instruct を使った iStar モデルの修正，出力まで全て同じ GPT のスレッドで行う．

ここで，低品質箇所は iStar モデルの意味的な低品質箇所 [7] と GRL の意味的な低品質箇所 [8] のうち，要素記述の低品質箇所を修正するために文意に特化した 14 種類の低品質箇所を用いる．14 種類の低品質箇所を表 1 に示す．

5　iStar モデル作成支援ツール

本研究の提案手法を用いて要求仕様から iStar モデルを自動生成するためにツールを実装する．

表 1　低品質箇所

低品質箇所分類	低品質箇所 (略称)
曖昧	複雑な文章
	親要素と子要素が近い
	ゴールが不明確
	多義語を含む文章
	クオリティが非機能要件で無い
	具体的過ぎる要素
	冗長な文章
漏れ	必要な要素が欠けている
	要求の漏れ
	不完全なテキスト
一貫性の欠如	依存元と依存関係要素が無関係
	依存先と依存関係要素が無関係
	親要素と子要素が矛盾
	親要素と子要素の不一致

5.1　実装の目的とツールの概要

iStar モデル作成支援ツールの実装の目的は以下の 2 点である．

1. 初学者に対する iStar モデル作成支援
2. 生成する iStar モデルの安定性の確保

1. を実現するためにツールの入力を「作成したい iStar モデル名」,「アクタ」,「詳細な要求」に限定し，ツールが自動でモデル生成プロンプトに変換する．その後，GPT-4o を OpenAI の API[9] を用いて呼び出し，提案手法を適用することで iStar モデルを生成する．生成した iStar モデルは Web 操作を行い，piStar[10] 上に自動表示する．

2. の安定性の確保とは，GPT が生成する iStar モデルは生成する都度異なるモデルが生成されるため，一度に 4 つのモデルを非同期処理を用いて生成することで iStar モデルの選択肢をツールのユーザに提供することである．

6　評価実験

6.1　実験の目的

実験によって，自然言語で記述された要求仕様に対して提案手法を適用した際の iStar モデルに着目し，評価するために実験を行った．なお，この評価実験は以下に示す Research Question(RQ) に答えるために実施する．

RQ1 モデル生成プロンプトは構文エラーの無い iStar モデルをどの程度生成可能か．

RQ2 Self-Instruct は iStar モデル上の低品質箇所を検出するプロンプトを漏れなく生成できるか．

RQ3 Self-Instruct で生成した検出プロンプトは低品質箇所を全て修正できるか

RQ4 提案手法で生成された iStar モデルの品質はどの程度か

6.2　実験方法と評価方法

6.1 で立てた各 RQ に対する評価実験を以下に示す．

RQ1 3 つの異なる要求仕様に対してそれぞれモデル生成プロンプトを適用し，各要求仕様について 10 回中何回構文エラーの無い iStar モデルが生成されるかを評価する．

RQ2 14 種類の各低品質箇所を入れた iStar モデル (1 モデル：1 低品質) に対して Self-Instruct を 5 回ずつ適用し，5 回中何回低品質箇所を検出できたかを評価する．

RQ3 RQ2 に引き続き，RQ2 で検出した低品質箇所を何回正しく修正できているかを評価する．

RQ4 RQ4 では提案手法の品質を評価するために 2 つの実験を行う．

1. 提案手法で生成される iStar モデルの構造の評価実験
2. 提案手法で生成される iStar モデルと要求仕様のマッピング実験

RQ4 の 1. では iStar モデルの低品質箇所を定めた研究 [7] のうち構造的な低品質箇所の定義を使用し，提案手法を用いて生成された iStar モデルの構造と初学者が作成した iStar モデルの構造を比較することで評価する．以下に本研究で使用した各構造的な低品質箇所を示す．

- 同名のアクタが存在している．
- アクタに直接依存関係が接続されている．
- 依存関係が 1 つのアクタに集中している．
- 2 つのアクタ間で依存関係が集中している．
- アクタ内に他の要素と接続していない孤立した要素が存在する．
- 詳細化がゴールで終了している．

RQ4 の 2. では提案手法で生成した iStar モデルに要求仕様の記述がすべて反映されているかを判定するために，RQ1 で使用した 3 つの要求仕様に対して提案手法を適用し，それぞれ 5 モデル生成する．その後，生成した iStar モデルと各要求仕様とマッピングを行う．

6.3 実験結果

6.3.1 RQ1 の実験結果

RQ1 に答えるための実験として 3 つの要求仕様に対して 10 回 iStar モデルを生成した結果を表 2 に示す．表中の実験結果欄は，10 個のモデルのうち何個のモデルが構文エラーが無かったかの割合を示す．

表 2 RQ1 の実験結果

要求仕様	実験結果
酒屋の在庫管理問題	8/10
病院の診察予約	8/10
会議室の利用予約	7/10
平均	76.66%(23/30)

表 2 より，モデル生成プロンプトは平均約 76.7%の確率で構文エラーの無い iStar モデルを生成した．

6.3.2 RQ2，3 の実験結果

RQ2，RQ3 の実験として各低品質箇所を Self-Instruct を用いて 5 回ずつ検出・修正した結果を各低品質箇所の分類ごとの平均で表 3 に示す．表中の検出 (RQ2) 欄では低品質箇所の分類ごとの検出率の平均，修正 (RQ3) 欄では修正率の平均を示す．

表 3 RQ2，3 の実験結果

低品質箇所分類	検出 (RQ2)	修正 (RQ3)
曖昧	74.28%	100%
漏れ	73.33%	100%
一貫性の欠如	90%	100%
平均	78.57%	100%

表 3 より Self-Instruct は，低品質箇所を約 78.6%の確率で検出し，検出した低品質箇所は 100%修正した．

6.3.3 RQ4 の実験結果

RQ4 のうち，1. 提案手法で生成される iStar モデルの構造の評価実験の結果を表 4 に示す．表中の GPT 欄，初学者欄ではそれぞれ 5 つのモデルの各構造的な低品質箇所があった数の合計値を示す．

2. 提案手法で生成される iStar モデルと生成に使用した要求仕様のマッピング実験の結果を表 5 に示す．なお，マッピング実験は各要求仕様ごとに 5 回ずつ行った結果を平均した値を示す．

表 4 RQ4-1 の実験結果

構造的な低品質箇所	GPT	初学者
同名のアクタ	0	0
依存関係の集中	0	0
依存関係の偏り	0	1
アクタに依存関係	0	25
孤立した要素	1	2
詳細化がゴールで終了	4	3
合計	5	31

表 5 RQ4-2 の実験結果

要求仕様	平均
酒屋の在庫管理問題	84.21%
病院の診察予約	89.09%
会議室の利用予約	88.57%
平均	87.41%

表 4 より，提案手法で生成した 5 モデルの構造的な低品質箇所は合計で 5 箇所指摘され，初学者が作成した 5 モデルは構造的な低品質箇所は 31 箇所指摘された．表 5 より，提案手法で生成した iStar モデルは，要求仕様の内容を平均して約 87.4%反映している．

7　考察

RQ1 について，表 2 より構文エラーの無い iStar モデル生成の確率は平均は約 76.7%であることから，モデル生成プロンプトは構文エラーの無い iStar モデルをある程度高い確率で生成できると言える．一方，実験の結果から GPT がモデル生成プロンプトを無視して iStar モデルを生成する場合があることが明らかとなった．しかし，他の実験で ChatGPT-o1 を活用することでモデル生成プロンプトを確実に実行できることが分かった．

RQ2，3 について，表 3 より低品質箇所の検出結果の平均は約 78.6%，修正結果の平均は 100%であることから Self-Instruct は低品質箇所をある程度高い確率で検出でき，正しく修正できていると言える．一方，実験の結果から Self-Instruct では検出できない低品質箇所があることが明らかになった。検出できない低品質箇所に関しては，その低品質箇所に適したアルゴリズムやプロンプトを作成する必要がある．

RQ4 について，表 4 より構造的な低品質箇所は提案手法の合計で 5，初学者の合計で 31 であることから，GPT が生成した iStar モデルは初学者が生成した iStar モデルより構造的な品質が高いと言える．

さらに，表 5 より提案手法により生成した iStar モデルは，要求仕様を平均約 87.4% 反映していることが分かった．これより、提案手法は要求仕様を比較的高い精度で iStar モデルに反映できると考えられる。一方で，要求仕様の一部が iStar モデルに反映されていない箇所も存在することが明らかとなったが，データ構造などの iStar 自体が記述できないもので本研究の対象外であった．

7.1　妥当性への脅威

7.1.1　内的妥当性

本研究の提案手法を評価する際，GPT が生成した iStar モデルを評価対象としているため，内的妥当性への脅威はある程度軽減されていると考えられる。しかし，RQ4 の iStar モデルの構造評価実験では，モデル構造の評価を我々の先行研究を基に行っている．このため，提案手法は構造的な低品質箇所を回避する設計となっている可能性がある．今後，第三者が定義した iStar モデルの構造評価に関する研究が実施された場合，その定義を用いて提案手法を再評価する必要がある．

また，OpenAI の API を用いて呼び出した GPT は事前のプロンプトと回答情報などを記憶しない。これにより，iStar モデル生成時の GPT に学習効果は無く，この点での内的妥当性への脅威は無い。なお，iStar モデル作成支援ツールでは，モデルを生成する際に初めからプロンプトと回答を履歴に記憶し，次のプロンプトにこれまでの履歴を全て入力している。

7.1.2　外的妥当性

本研究の評価実験では，酒屋の在庫管理問題，病院の診察予約，会議室の利用予約の 3 つの異なる要求仕様を用いて iStar モデルの生成を行った．このことから、外的妥当性への脅威は一定程度軽減されていると考えられるが，よりドメイン知識が必要とされる実社会の要求仕様への適用が求められる。

8　おわりに

本研究では要求仕様から GPT-4o を用いることで iStar モデルを自動的に生成する手法を提案した．さらに，iStar モデル作成支援ツールを開発し，複数の要求仕様に対する提案手法の有用性を評価した．その結果，本研究で提案したモデル生成プロンプトにより構文エラーの無い iStar モデルの生成と Self-Instruct を用いた低品質箇所の検出・修正に一定の効果があることを確認した．

今後の課題は以下である．

- 高性能の GPT-o1 の活用
- 特定できなかった低品質箇所を特定するアルゴリズムの開発
- より詳しい要求仕様への適用
- iStar モデル作成支援ツールの改善

9　参考文献

参考文献

[1] Danilo Arruda, et al. A chatbot for goal-oriented requirements modeling. In ICCSA 2019, pp. 506–519. Springer, 2019.

[2] Boqi Chen, et al. On the use of GPT-4 for creating goal models: an exploratory study. In REW2023, pp. 262–271. IEEE, 2023.

[3] Fabiano Dalpiaz, Xavier Franch, and Jennifer Horkoff. istar 2.0 language guide, 2016.

[4] OpenAI. Hello GPT-4o. `https://openai.com/index/hello-gpt-4o/`, 2024. Accessed: 2025-01-07.

[5] Eric SK Yu. Towards modelling and reasoning support for early-phase requirements engineering. In ISRE1997, pp. 226–235. IEEE, 1997.

[6] Yizhong Wang, et al. Self-instruct: Aligning language models with self-generated instructions. arXiv preprint arXiv:2212.10560, 2022.

[7] Yoshitake Hirabayashi, et al. Defining Bad Smells and Automating Their Detection in Goal-Oriented Requirement Analysis Method iStar. In APSEC2023, pp. 349–358. IEEE, 2023.

[8] Nouf Alturayeif and Jameleddine Hassine. Detection of Linguistic Bad Smells in GRL Models: An NLP Approach. In MODELS-C2023, pp. 318–327. IEEE, 2023.

[9] OpenAI. OpenAI API. `https://openai.com/index/openai-api/`, 2024. Accessed: 2025-01-07.

[10] João Pimentel and Jaelson Castro. pistar tool - A pluggable online tool for goal modeling. In RE2018, pp. 498–499, 2018.