ゴール指向要求分析手法 iStar における不吉な臭いの検出

2019SE003 藤井涼香　2019SE009 平林義健　2019SE047 大田真史
指導教員：佐伯元司

1　はじめに

要求分析は開発の最初の工程であるため，正しく行わないと手戻りが生じ，開発コストを増大させるばかりでなく，低品質のソフトウェア製品が作られてしまう．手戻りを防ぐためには，要求を正確に特定する要求分析手法が求められる．

要求を正確に特定する要求分析手法の 1 つとして，ゴール指向要求分析手法が提案されている．この手法は顧客の要求をゴールとし，分解・詳細化を繰り返してシステムに対する具体的な要求を導出していく手法である [3]．この手法の成果物であるゴールグラフは、詳細化のやり方などについての指針がないため，文法エラーではないが低品質のゴールグラフを作ってしまう可能性がある．本研究では，このような品質低下の潜在的な問題を引き起こす兆候を不吉な臭いと呼び，ゴール指向要求分析手法のひとつである iStar における不吉な臭いの特定・自動検出を研究課題とする．本研究の貢献は，

1. iStar 用の不吉な臭いリストを作成したこと．
2. 不吉な臭いリストをもとに自動検出を行うツールを開発したこと．

不吉な臭いは，グラフのノード数などの構造的な観点，グラフ要素の記述内容の意味的な類似性などの意味的な観点で検出する．

本論文は，2 章で関連研究，3 章で本論文で使用する技術やツールについて説明する．4 章でアプローチ，5 章で不吉な臭いリスト，6 章で不吉な臭いの自動検出，7 章で評価，8 章でまとめを述べる．

2　関連研究

2.1　AND-OR グラフによる不適切なゴール詳細化の自動的な検出

浅野らはゴール指向要求分析手法の 1 つである AND-OR グラフを用いて，ゴールグラフにおける不適切なゴール詳細化の自動的な検出を行った [1]．この手法では，ゴール記述の意味的な類似度，兄弟枝の数および葉ゴールの総体的な深さを評価することによって検出を行っている．開発したツールの評価実験で不適切な詳細化を行っている箇所全部のうち 60 %以上を検出でき，被験者の見落としていた不適切な詳細化の指摘もできたとしている．ゴール指向要求分析手法には単純な AND-OR グラフを用いた手法以外に，iStar[4]，KAOS[3] などいくつかの種類がある．このうち浅野らが対象としたのは単純な有向グラフである．しかし，iStar は多様な要素と要素間関係を持っており，浅野らの詳細化関係だけに注目する手法では検出が不十分である．

2.2　Word2Vec を用いたゴール記述間の類似度算出方法

飯島らは Word2Vec を用いてゴール記述間の類似度算出手法を 2 種類提案した [2]．1 つは，係受け解析により文節を抽出して類似度を算出し，もう一方ではゴール記述を単語毎に分解し単語の頻度による加重平均によりゴール記述類似度を算出した．この手法では MeCab[8] や Cabocha[9] により単語または文節を抽出し，最後に Word2Vec[7] を用いてゴール間類似度を計算し，閾値を下回った枝を不適切な詳細化としている．

2.3　設計段階の不吉な臭いパターン

Bad smell pattern は，設計の構造や振る舞いを退化させる危険を含むものを集めて，パターン化，カタログ化したものである．Sara．H．らの論文では，Bad smell pattern は 28 種類特定されている [10]．Bad smell pattern は設計やコーディング段階での不吉な臭いをパターン化したものであり，要求分析段階のものではない．しかし，ゴールグラフとオブジェクト指向の概念との間の関係を前提とすると，Bad smell pattern と本研究の不吉な臭いとの間に類似性がある．詳細は 7 章 7.5 節で述べる．

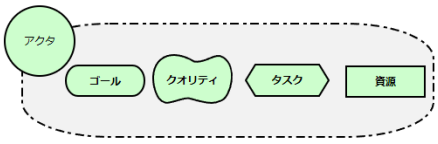

図 1　iStar の構成要素

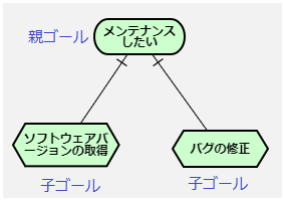

図 2　AND 分解

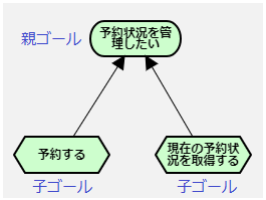

図 3　OR 分解

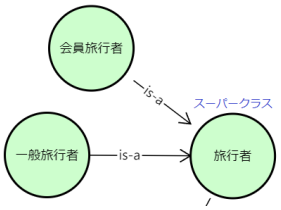

図 4　アクタ間の依存関係

3　準備

3.1　iStar2.0

3.1.1　iStar2.0 とは

iStar とは，ゴールおよびアクタ指向のモデリングおよび推論フレームワークとして 90 年代半ばに発表されたものであり，iStar2.0[5] はその最新版である．アクタ，ゴール，クオリティ，タスク，資源の 5 つの要素を用いてアクタ内部とアクタ間の依存関係を分析する手法である．5 つの構成要素は，それぞれ**図 1** のような図形を使用する．iStar の特徴は，アクタ間の依存関係を記述できることである．以降の説明において，本論文内では iStar2.0 を「iStar」と統一して記述する．

3.1.2　AND 分解と OR 分解

図 2，**図 3** に iStar における AND 分解と OR 分解の例を示す．AND 分解は，子ゴール全てが達成されれば親ゴールが達成されることを表し，十字の線で記述される．OR 分解は，子ゴールのうち 1 つが達成されれば親ゴールが達成されることを表し，通常の矢印で記述される．

3.1.3　is-a 関係

is-a 関係とは，iStar におけるアクタ間の一般化・特殊化を表す．アクタ間の依存関係の例を**図 4** に示す．

3.1.4　dependency

iStar では，依存関係を dependency といい，**図 5** のように依存関係の受益者を depender，依存関係の提供者を dependee, 提供者が受益者から要求されるものを dependum という．また，depender 内の要素を dependerElmt といい，dependee 内の要素を dependeeElmt という．

3.2　piStar

piStar は，iStar においてモデルの入力編集を行うツールである [6]．piStar は iStar モデルの 12 種類の文法エラーをチェックする機能があるが，本研究での不吉な臭い

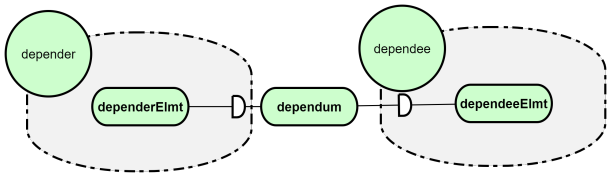

図 5　アクタ間の依存関係

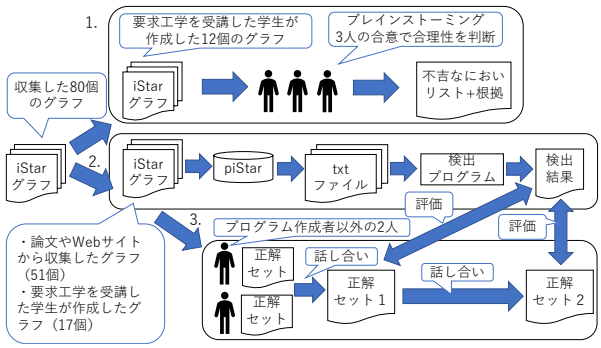

図 6　アプローチの流れ

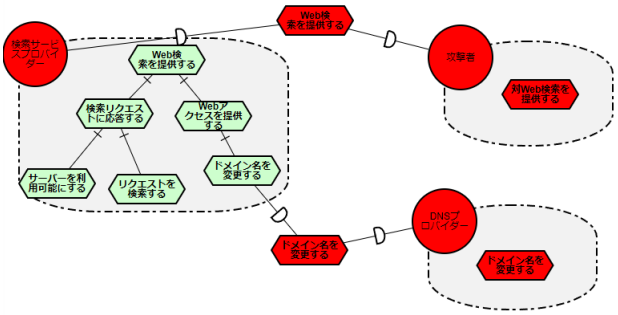

図 7　piStar 検出例

は文法エラーではないため piStar のこの機能をそのまま使うことはできない．

4　アプローチ

本研究では以下の 3 点を行う．

1. iStar ゴールグラフにおける不吉な臭いの特定
2. iStar ゴールグラフにおける不吉な臭いの自動検出
3. 自動検出プログラムの評価

1，2，3 のアプローチの流れを**図 6** に示す．

5　不吉な臭いリスト

浅野らの研究 [1] を基に，検出のための観点として，列挙した不吉な臭いを構造的な観点と意味的な観点の 2 つに分類した．構造的な不吉な臭いとは，グラフの構造に関するもので，ノード数や要素関係の深さに着目している．意味的な不吉な臭いとは，ゴールの記述内容に関するもので，ゴール記述の類似度に着目している．それぞれの観点における不吉な臭いリストを作成した．作成した不吉な臭いリストを**表 1**，**表 2** に示す．

6　不吉な臭いの自動検出

不吉な臭いの自動検出は piStar が出力する json 形式のテキストファイルを不吉な臭いリストを元に python を使用し検出する．出力結果としては，コマンドプロンプト上に不吉な臭いの箇所を表示，実行したファイルの背景色情報を赤色に変更する．このファイルを再度 piStar でロードすることで，**図 7** に示すようにエラー箇所が視覚的に理解できるようにしている．

構造的な不吉な臭いの検出は，**表 1** の不吉な臭いに書かれている項目に従って，閾値を設定したり，条件を満たす要素の存在をチェックしたりすることによって検出を行う．

意味的な不吉な臭いの検出は，飯島らの単語ベース

表 1　構造的な不吉な臭いリスト

ID	不吉な臭い 根拠
k-001	1 つのアクタに依存関係が集中しすぎている 集中しているアクタに変更が必要になった時 変更箇所が広範囲となる
k-002	dependency の要素が多すぎる アクタ間の関係が強くなり，片方のアクタに 変更が生じた際のもう片方の変更箇所を 限定することが難しくなる
k-003	アクタ内に何もつながっていない要素がある どこで必要か，どこで発生した要素か 分からず，後に変更が必要となる
k-004	多重継承 継承するものが競合を起こす時， どの競合先を選択し設計，実装するかが曖昧となる．
k-005	ゴールで and，or 分解が終了している 詳細化が不十分なため，そのゴールを 達成するための追加要素が後に発生する 可能性がある
k-006	同じ名前のアクタが存在する 依存関係が不明確となり曖昧な設計 となる可能性がある
k-007	詳細化されているアクタに 直接依存関係がある 詳細化が不十分となる
k-008	participate-in と is-a のリンクが多い 継承するものに変更が生じた際に， 変更箇所が広範囲となる
k-009	participate-in と is-a のリンクが深い 継承するものに変更が生じた際に， 変更箇所が広範囲となる

表 2　意味的な不吉な臭いリスト

ID	不吉な臭い 根拠
i-001	dependum と dependerElmt の記述内容 が無関係 (アクタから直接 dependum が 伸びているものはカウントしない) それぞれのアクタで実行することが不明確
i-002	dependum と dependeeElmt の記述内容 が無関係 それぞれのアクタで実行することが不明確
i-003	ゴールグラフの親子間の記述内容 が無関係 それぞれのアクタで実行することが不明確
i-004	同じアクタ間で共通の資源が何度も出てくる 曖昧な設計となる
i-005	矢印の向き (is-a と participates-in) 抽象的 なものから具体的なもの 文法ミスが起きる

Word2Vec[2] を使用する．この検出手法では，形態素解析に MeCab[8]，単語ベクトルの導出に Word2Vec[7]，TD-IDF(Term Frequency-Inverse Document Frequency) を用いて各単語ベクトルに重みをつけ，文ベクトルを計算している．しかし，この手法では単語のベクトルは計算できるが，抽象的な単語や具体的な単語を判別することはできない．したがって，i-005 の「is-a と participate-in」の検出は今後の課題とする．コーパスは東北大学日本語 Wikipedia エンティティベクトル [11] を使用し，次元数は 200 次元である．

7　評価

7.1　実験の目的

検出プログラムが不吉な臭いをどの程度検出できるかを調べるために，実験を行った．この実験では，プログラム作成者以外の著者 2 人が iStar モデル中にある不吉な臭い

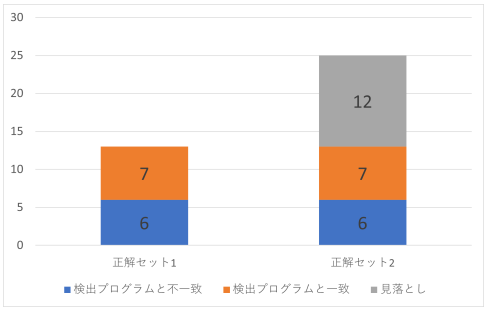

図 8　指摘内訳

の箇所を指摘し，これを検出プログラムによる検出結果と比較することで検出プログラムの有用性を考察する．検出プログラムが不吉な臭いをどの程度発見できるかを測るために，以下の問いを設けた．

- Q1 検出プログラムは人間の見落としをどの程度検出できるか
- Q2 検出プログラムは不吉な臭いをどの程度検出できるか

それぞれの問いに答えるために，プログラム作成者以外の著者 2 人が提示した不吉な臭いの箇所と比較し，適合率および再現率を算出する．

7.2　実験の流れ

プログラム作成者以外の著者 2 人には，熟練度の高い人が作った iStar モデルとして論文や Web サイトから収集した 51 個の iStar モデルと，熟練度の低い人が作った iStar モデルとして要求工学を受講した学生が作成した iStar モデル 17 個の計 68 個の iStar モデルを対象に，不吉な臭いを指摘させた．熟練度の高い低いで iStar モデルを分けた理由としては，熟練度の違いにより不吉な臭いが出現する頻度が違うと予想したためである．

この実験ではプログラム作成者以外の 2 人による指摘と検出プログラムによる検出箇所の比較を 2 回行う．初めに 68 個の iStar モデルについて不吉な臭いリストを参考に 1 人ずつ不吉な臭いの箇所を指摘させ，後にそれぞれの指摘結果を照らし合わせ話し合い正解セット 1 を作成した．次に検出プログラムの意味的な不吉な臭いの検出結果をプログラム作成者以外の 2 人に見せ，その結果を参考にもう一度意味的な不吉な臭いである箇所を指摘させ，それを正解セット 2 とした．意味的な不吉な臭いである箇所のみを再度指摘させた理由として以下の 2 つが挙げられる．

- 意味的な不吉な臭いは自然言語を扱っていること．
- 検出プログラムは，正解セット 1 の段階で構造的な不吉な臭いを全て検出できたため．

7.3　結果と考察

7.3.1　不吉な臭いの人間間の相違

プログラム作成者以外の著者 2 人がそれぞれ作った 2 つの正解セットには，いくつか相違点があった．相違があったのは 62 か所，全体の 1.48 %（小数点第 3 位を四捨五入）であり，相違が生じた原因の 60 %以上は見落としであり，約 30 %は不吉な臭いリストの細かな認識の違いであった．したがって，以下の事が分かった．

1. 自動的な検出の重要性
2. 不吉な臭いリストを作成する際は，あらゆる可能性を考慮して検出条件を定める事が重要である

7.3.2　Q1. 指摘箇所の見落とし

図 8 に正解セット 1 と正解セット 2 において不吉な臭いであると判断された箇所の内訳を示す．

構造的な不吉な臭いは見落としが無かった．正解セット 1 で検出プログラムが指摘できなかった意味的な不吉な臭いは 6 箇所であった．また，プログラム作成者以外の著者 2 人が見落としていた箇所は全体で 12 箇所であり，正解セット 2 に入っている意味的な不吉な臭い全体の 48 %に相当する．したがって，プログラム作成者以外の著者 2 人は意味的な不吉な臭いの検出結果を 48 %見落とし，検出プログラムによってそれを補うことができた事が分かる．

表 3　正解セット 2 と比較結果

熟練度	検出数	正解セット 2	正解数	適合率	再現率
高い	50	20	13	0.26	0.65
低い	20	5	5	0.25	1.00
全体	70	25	18	0.26	0.72

7.3.3　Q2. 不吉な臭いの検出

実験結果より検出プログラムは構造的な不吉な臭いを全て検出することができたため適合率，再現率は共に 100 %となる．この結果になった理由は，構造的な不吉な臭いの検出条件は著者ら 3 人の合意で決め，正解セットを作成したのも著者の内 2 人であったためである．これについては，7.4.1 節の内的妥当性への脅威で考察する．

意味的な不吉な臭いについて，正解セット 2 と検出プログラムの比較を行った．**表 3** は熟練度の高い人が作った iStar モデル群と熟練度の低い人が作った iStar モデル群および全体での適合率と再現率を示す．

検出数の列は検出プログラムにより検出した箇所の数を，正解セットはプログラム作成者以外の 2 人で作成した正解セットの数を，正解数は検出プログラムによる検出箇所と検出プログラム作成者以外の著者による指摘箇所が一致した数を表す．**表 3** から，適合率は低いものの熟練度の高い人が作った iStar モデル群の再現率は 0.65 であり，熟練度の低い人が作った iStar モデル群は 1.00 であった．全体としては 0.72 となっており，検出プログラムによって意味的な不吉な臭いのうち 72 %を検出できていることがわかる．

適合率が低い原因として，以下の 4 つが挙げられる．

1. ドメイン知識の考慮不足
2. 未知語の存在
3. Word2Vec の単語ベクトル算出方法による問題
4. iStar の要素に記述された文は短い

Word2Vec の単語ベクトル算出方法による問題について説明する．例えとして評価に使った iStar モデルの 1 つである図書館システムを見ると，AND 分解されている親の「本を貸す」と子の「貸出期限の通知」の類似度は「貸す」と「貸出」があるためある程度高くなると考えられる．しかし，類似度は約 0.24 と低い．この理由として，Word2Vec は単語ベクトルをその単語の周辺語を使用して計算している．このことにより，「貸す」と「貸出」のような同じ漢字で同じ意味の単語は同じ文章内で使用されることは考えづらいため類似度が低くなったと考えられる．実際に Word2Vec での「貸す」と「貸出」の類似度は約 0.17 と低い．

次に iStar は要素に記述された文が短いという特徴がある．これにより文における 1 単語の比重が増大し，未知語などがあると適切な類似度算出を行うことができないと考える．

7.4　妥当性への脅威

7.4.1　内的妥当性

検出プログラムを作成した著者は正解セット作成には関与していないため，この部分の内的妥当性への脅威は軽減されていると考えられる．しかし，著者であることには変わりがなく，内的妥当性への脅威はぬぐえない．また，7.3.3 節で述べたように構造的な不吉な臭いのリスト作成条件を著者ら 3 人の合意で作成したが，正解セットを作成したのも著者のうち 2 人であったため内的妥当性への脅威に影響を及ぼすと考える．これに関して，不吉な臭いリスト作成にまったく関与しなかった場合を考え，プログラムが最低限構造的な不吉な臭いをどの程度検出できるかを調べる．すると，再現率は約 64 %，適合率は約 96 %となるため，検出プログラムの再現率や約 64 %から 100 %の間，適合率は約 96 %から 100 %の間になると考える．さらに，実験で用いた iStar モデルの一部は英語であったため著者らが日本語に翻訳して行った．翻訳の質が影響を及ぼすことも考えられる．

表 4 オブジェクト指向と iStar 要素の対応

オブジェクト指向	iStar 要素
システム全体	ゴールグラフ全体
システムが達成すべき責務	ゴール
クラス	アクタ
メソッド	タスク
インスタンス変数	資源
関連	dependency
継承	is-a

表 5 Bad smell pattern

bad smell	内容
God class	システム内で多くの タスクを実行し， 多くの責任を負っている クラス
Intenseive Coupling	他のクラスからの メソッド呼び出しが 複数あるメソッド
Shotgun surgery	変更を行う際に 複数のクラスに変更が 必要となるメソッド
Feature envy	他のクラスを過剰に使用する メソッド
Class grime	クラスの役割とは 無関係なクラス関連要素
Schizophrenic Class	鍵となる概念を 複数持つクラス

表 6 不吉な臭いリスト ID と bad smell pattern

ID	bad smell pattern
k-001	god class・Intensive Coupling Shotgun surgery・Feature envy
k-002	god class・Shotgun surgery Feature envy
k-003	Class grime
k-004	schizophrenic class

表 7 bad smell による新たな不吉な臭いリスト

bad smell 内容	考えられる不吉な臭い 根拠
long method いくつかの機能を 実装し複雑で理解 しにくいメソッド	要素内の記述が長すぎる 記述内容が複雑化し， 誤認や手戻りの原因
refused bequest 決して使用しない 機能を継承する クラス	必要のない継承 (is-a , participate-in) クラス関係が複雑化し， 誤認や出戻りの原因

7.4.2 外的妥当性

インターネットや論文，文章から多種類の分野のモデルを集めて行ったが，より一般性をあげるため実際のプロジェクトなどで作成された iStar モデルをもっと収集し評価する必要がある．

7.5 不吉な臭いと Bad smell pattern

オブジェクト指向と iStar との対応を考えると，Bad smell pattern のいくかと対応づくものがある．例えば God class は k-001, k-002 と対応づく．Bad smell pattern は主に設計段階のもので，要求分析段階ではないが，類似の不吉な臭いがあることは興味深い．オブジェクト指向と iStar 要素の対応関係を**表 4** に示す．考察で使用する Bad smell pattern を**表 5** に示す．**表 5** で示した不吉な臭いリストと bad smell pattern の関係を**表 6** に示す．**表 5** に則り，bad smell 行の上段は不吉な臭いの内容，下段は根拠に対してそれぞれ対応する bad smell を表している．

bad smell pattern の考え方を参考に，新たに考えられた不吉な臭いを**表 7** に示す．

8 まとめ

本研究では，ゴール指向要求分析法 iStar の問題点のグラフの作り方，特に詳細化のやり方について指針がないため，文法エラーではないが低品質のゴールグラフを作ってしまう可能性を挙げた．この問題点を解決するため，不吉な臭いリストを作成し，自動検出できる不吉な臭いを python を用いた検出プログラムを作成した．

作成した検出プログラムは構造的な不吉な臭いを最低限 64 %から 100 %で検出することができ，意味的な不吉な臭いを 72 %検出することができた．今後の課題として以下の 5 点が挙げられる．

1. 不吉な臭いリストの質向上
2. 検出プログラムの実用性向上
3. 実装できなかった不吉な臭いの検知プログラムの作成
4. ドメイン知識を考慮した検知プログラムの作成
5. 概念辞書を用いた類似度算出プログラムの作成

参考文献

[1] 浅野圭亮, 林晋平, 佐伯元司,”ゴール指向要求分析法の不適切なゴール詳細化の検出 ”, 電子情報通信学会ソフトウェアサイエンス研究会, Vol.116, SS-512, pp.127-132, 2017 .

[2] 飯島慧, 林晋平, 佐伯元司,”不適切なゴール類似度算出法の比較”, 不適切なゴール詳細化検出のためのゴール記述類似度算出法の比較, 電子情報通信学会ソフトウェアサイエンス研究会,vol.120,SS-407, pp.62-72,2021.

[3] A.van Lamsweerde. Requirements Engineering:From System Goals to UML Models to Software Specifications. Wiley,2009.

[4] E.S.K. Yu. Modeling Strategic Relationships for Process Reengineering.PhD thesis,University of Toronto, 1995.

[5] F.Dalpiaz, Xavier Franch, Jennifer Horkoff,” iStar 2.0 Language Guide ”, 2016. https://arxiv.org/pdf/1605.07767v3.pdf.

[6] Fabiano Dalpiaz, Xavier Franch, and Jennifer Horkoff3 https://arxiv.org/pdf/1605.07767v3.pdf

[7] T. Mikolov, K. Chen, G. Corrado and J. Dean: Efficient Estimation of Word Representations in Vector Space, Proc. International Conference on Learning Representations 2013 (ICLR) Workshop Track, arXiv: 1301.3781, (2013)

[8] “MeCab Yet another part of speech and morphological analyzer,” https://taku910.github.io/mecab/. (Accessed on 02/07/2021)

[9] “CaoboCha Yet another japanese dependency structure analyzer,” http://taku910.github.io/cabocha/. (Accessed on02/07/2021)

[10] H.Sara. S. Almadi, Danial Hooshyar and Rodina Binti Ahmad bad smells of Gang of Four Design Patterns: A Decade Systematic Literature Review 2021

[11] “日本語 Wikipedia エンティティベクトル,” http://www.cl.ecei.tohoku.ac.jp/m̃-suzuki/jawiki_vector/ (Accessed on 01/17/2021)