Vol.2019-SE-201 No.14
2019/3/8

word2vecによるゴール間類似度算出手法の実践と評価

石川 公一[1,a)]　小形 真平[1,b)]　中川 博之[2,c)]　岡野 浩三[1,d)]

概要：派生開発へのソースコード再利用支援を目的に, 形態素解析や集合類似度を用いて, ゴールモデル間で類似したゴールを検出する手法が研究されている. しかし, 書き手により異なる語彙で記述されるゴール間を, 同一な単語の数に頼り類似度を算出することには限界がある. そこで本研究では, ゴール間類似度算出という題材において, 単語の意味までを考慮できるとされる word2vec の効果を検証するため, 形態素解析と集合類似度を用いる手法との比較評価について述べる. ロボコンで開発するソフトウェアについて, KAOS 法を用いて作成したゴールモデルを適用事例とした結果, word2vec により算出したゴール間類似度が従来に比べて精度的に優れていた. 一方では, 集合類似度を用いる手法よりも精度が良いことを示すことはできたが, 依然として精度は十分に高められていない. それを向上するための課題を整理し報告する.

キーワード：ゴールモデル，ソフトウェア再利用，Word2Vec, 自然言語処理, 類似度

Practice and Evaluation of a Method to Calculate Similarity between Goals Using Word2Vec

Koichi ISHIKAWA[1,a)]　Shinpei OGATA[1,b)]　Hiroyuki Nakagawa[2,c)]　Kozo OKANO[1,d)]

Abstract: To support reusing software components for derivative development, methods to detects similar goals between goal models by using morphological analysis and term-based similarity measures have been proposed. However, goals' representation is affected by writer so there is a limit to precisely calculating similarity between goals by using measures. In this study, we performed comparative evaluate between a method using term-based similarity measure and one using word2vec in order to verify the effectiveness of word2vec. The word2vec tool can catch semantic similarity of a word. As a result of an evaluation applying those methods to goal models about robot-contest software, the resulting accuracy by the method with word2vec was better than the method with the term-based similarity measure. Meanwhile, the accuracy was still not enough by the method with word2vec. Hence, we discuss challenges for improving the accuracy of calculating the similarity.

Keywords: Goal Model, Software Reuse, Word2Vec, NLP, Similarity

1. まえがき

近年, ソフトウェア開発現場において納期の短縮化, 開発規模の著しい増大によって開発者の負担が増大している. そのため部品化したソフトウェアを再利用して新規・派生ソフトウェアを開発することが提案されている [1], [2], [3]. そのような手法では, 新規に開発するソフトウェアにどの部品が再利用できるかを調べる必要があり, 従来では様々な手法が提案されてきた [4], [5], [6], [7], [8], [9], [10]. 再利

[1] 信州大学 工学部　〒 380-0928 長野県長野市若里 4-17-1,
Faculty of Engineering, Shinshu University, Wakasato 4-17-1, Nagano-shi, Nagano, 380-0928 Japan
[2] 大阪大学 大学院情報科学研究科　〒 565-0871 大阪府吹田市山田丘 1-5,
Graduate School of Information Science and Technology Osaka University, 1-5, Yamadaoka, Suita, Osaka 565-0871 Japan
[a)] 18w2010f@shinshu-u.ac.jp
[b)] ogata@cs.shinshu-u.ac.jp
[c)] nakagawa@ist.osaka-u.ac.jp
[d)] okano@cs.shinshu-u.ac.jp

用を前提としたソフトウェア開発では, 要求仕様やコード間などの, ソフトウェア部品の情報間でトレーサビリティを確保することは重要である.

そのような背景があるため, 要求仕様とコード間のトレーサビリティを与える研究がされている [11], [12], [13], [14]. ソフトウェア開発におけるトレーサビリティとは, 成果物内の情報間の追跡可能性を意味する [11]. 情報間の具体的な繋がり一つ一つをトレーサビリティリンクと呼ぶ [11].

上記研究により各種技術が発展したならば, ソースコードまで追跡できる要求仕様から, 新規に開発するソフトウェアの要求に類似する要求を検出することで, コードやデザインの部品などの既存ソフトウェア資産を再利用できる [15]. しなしながら, このような再利用手法は未だ十分に確立されておらず, 主要な課題の一つに, 新規開発での要求と類似する要求を既存の要求に関するドキュメントから精度良く検出することが挙げられる.

類似した要求を検出する手法として中村ら [16] は, 複数ソフトウェアの共通した機能を自動判別するために，共通したゴールを探し出す手法を提案している. これはゴール間の類似度をジャッカード係数 [17] を用いて算出する. ここでのジャッカード係数とは, ゴールが含む単語集合の類似度を測るものである. この類似度に対して一定の閾値を設け, 算出した値が閾値以上であれば共通ゴール, それ以下であれば可変ゴールに判別するといったものである.

しかし，単語集合が大きく異なる複数のゴール間では, たとえ同じ機能で実現できるゴールであるとしても, ジャッカード係数による類似度は低くなってしまう問題がある. 例えば, 『私はこれから野球する』『僕は今からベースボールをやる』を比較する場合, これらの意味は同様であるが, 文章間で登場する単語 (名詞や動詞) 集合間で要素は重複しないため, ジャッカード係数では低い類似度が算出される.

そこで, 機械学習により単語の分散表現を得る手法である word2vec に着目する. この分散表現は, 単語の意味を表現するベクトルであり, コサイン類似度と組み合わせると単語間の意味的類似度を得ることが可能になる [18], [19], [20]. 例えば, 『Gold』と『Au(元素記号)』の 2 つの単語は字面は全く異なるものであるが, word2vec では類似したものとして捉えることができる. このように word2vec を用いた手法は, 既存手法 [16] で用いている集合類似度による類似度算出手法の課題を克服し, ゴール間の類似度を従来より精度良く測ることができる可能性がある.

そこで, 本研究では従来よりも高精度なゴール間類似度算出手法を確立することを目的として, 単語の分散表現を用いたゴール間の類似度算出手法を示すと共に, 集合類似度を用いた手法との比較評価を行う. 評価の結果, word2vec を用いた手法の方が精度的に優れていることが示唆された.

2. 準備

2.1 KAOS 法におけるゴールモデル

ゴールモデルとは, ゴール指向要求分析手法に用いられる要求モデルである [21]. ステークホルダーのソフトウェアに対する抽象的な要求をゴールとして捉えて, ゴールの具体的な実現方法を分析することをゴール指向要求分析という [21]. ゴールモデルにおいて, 図 1,2 のように抽象的なゴールの具体的な実現方法を分析することをゴールの分解といい, ゴールは平行四辺形で表される.

本研究では分解元のゴールを親ゴール, 分解先のゴールを子ゴールと呼ぶ. 図 1 のような分解を AND 分解といい, 親ゴールの達成には子ゴールが全て達成される必要がある場合に用いる. 図 2 のような分解を OR 分解といい, 親ゴールの達成には子ゴールのどれかが達成される必要がある場合に用いる. KAOS 法では, ソフトウェアに対する要求を戦略的に分析することが可能であるため [21], [22], [23], 本研究では KAOS 法に則って記述されたゴールモデルを要求モデルとして扱う.

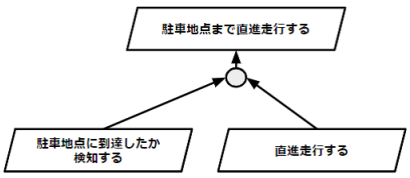

図 1 AND 分解

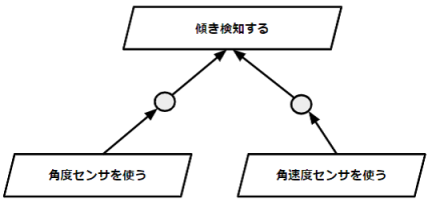

図 2 OR 分解

2.2 文章の形態素解析

後述する類似度算出手法は日本語で書かれたゴールモデルを対象とするため, 文章に対して自動で形態素解析するツールが必須である. 本研究では, 高速かつ軽量に動作することから, 形態素解析エンジンである MeCab[24], [25] を用いた. 文章を形態素解析し, 単語集合を得る際には既存手法 [16] と同様に, 動詞と名詞のみを残し, 単語を全て基本形にする. 例えば, 『私は野球をしに行きたい』という文章は, {私, 野球, する, 行く} のように変換される.

3. 類似度算出手法

本項では, 評価に用いる類似度算出手法について説明する. word2vec を用いた類似度算出手法を手法 A, 集合類似度を用いた手法による類似度算出手法を手法 B とする.

3.1 手法 A: word2vec による類似度算出手法

3.1.1 単語の意味ベクトル化

手法 A では, 単語の意味ベクトルを獲得するために, 鈴木ら [26] が配布している word ベクトル [27] を用いる. 単語ごとのベクトルの次元は 300 で, 2018 年 10 月 1 日時点での日本語 Wikipedia[28] の記事を用いて学習されている.

3.1.2 文章のベクトル化

一つ一つのゴールは自然言語で文章として記述され, 意味ベクトルは単語に対して構成されているので, 文章を学習モデルに入力しても意味を為さない. そこで, 形態素解析で得た単語集合に対し, word2vec により単語ごとに意味ベクトルを得る. 文章は通常複数の単語から成るため, ベクトルの和を算出する. 本研究では, この和ベクトルを文章の意味ベクトルとして扱う [29]. 得られた単語を $w_1, w_2 ... w_n$ とすると, 文章の意味ベクトル $\boldsymbol{d}$ は式 1 によって算出される. ただし, $w2v(w_i)$ は i 番目の単語の意味ベクトルを算出する関数である.

$$\boldsymbol{d} = \sum_{i=1}^{n} w2v(w_i) \tag{1}$$

3.1.3 コサイン類似度

手法 A では, 文章の意味ベクトル間の類似度を測定するために, コサイン類似度を用いる. 比較したいベクトルを $\boldsymbol{x}, \boldsymbol{y}$ とすると, コサイン類似度は式 2 によって算出できる. 2 つのベクトルのなす角が小さいほど近い方向を向いたベクトル, 即ち類似したベクトルとなる. 2 つのベクトルが同じ向きの場合 $cos(0) = 1$ のように類似度 100 %と算出できる.

$$cos(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{|\boldsymbol{x}| \times |\boldsymbol{y}|} \tag{2}$$

3.2 手法 B: 集合類似度による手法

集合類似度による類似度算出手法を説明する. 2.2 節の形態素解析で得た単語集合に対してジャッカード係数 [17] を算出し, それを類似度とする. ジャッカード係数は式 3 によって与えられる. ただし, X, Y は単語集合を表す.

$$Jaccard(X, Y) = \frac{|X \cap Y|}{|X \cup Y|} \tag{3}$$

4. 評価実験

4.1 概要

類似度算出における精度を両手法間で比較評価するために, 実際に作成されたゴールモデルに手法 A と手法 B を適用した. 適用事例は, プログラミングやモデリングの基礎知識を有する学生が開発した ET ロボコン 2017,2018[30] 用に作成した KAOS 法に従う 2 つのゴールモデルである. 各チームのゴールモデルの一部を図 8, 図 9 に示す. 例えば『BT 通信が使えるなら使う』と『BT 通信が可能なら BT 通信経由で受け取る』等, 文章だけを見ても類似している判断できるゴールペアの存在が分かる. 表 1 に示すようにゴールモデルは異なるチームにより作成されている. また, ゴールモデルの規模としては, ゴールの数はそれぞれ 80 個, 110 個である. 本評価では, 作成者の異なるゴールモデル間において, 両手法が算出するゴール間の類似度を算出するものとした.

表 1 評価に用いるゴールモデル

チーム 1 ゴール数	80
チーム 2 ゴール数	110
ペア数	8800

4.2 準備

精度の評価を行うにあたり, 正解を用意した. まず, 異なるモデルから一つずつゴールを取り出し, それをゴールペアとして構成した. これを全ての組み合わせ分行った. 具体的には, 表 1 に示すように 80 * 110 で 8800 個のペアを得た. そして, 各ペアに対して, ET ロボコンのソフトウェア実装経験者 1 名が, ゴールを満たす手続き (コード片) の類似性を考慮して, 類似すべき (正解となる) ペアを手動で識別した. 結果として 98 個の正解を得た.

4.3 実験方法

手法 A と手法 B を用いて, ゴールのペアごとに類似度を算出した. 類似度の算出だけでは精度の評価はできない. そこで, 閾値を設け, 類似度が閾値以上であれば類似していると予測し, 閾値未満であれば類似していないと予測するものとした. 閾値に応じて精度が変化するため, 閾値を 0%, 1%, 2%...100%と 1%ずつ変化させて精度評価を行った.

精度の評価指標として, 適合率と再現率及び 2 つの調和平均である F 値を定義する. 類似すると予測し, 実際には類似していた個数を TP, 類似すると予測し実際には類似していなかった個数を FP, 類似しないと予測し実際には類似していた個数を TN, 類似しないと予測し実施には類似していなかった個数を FN としたとき, 適合率 ($Precision$), 再現率 ($Recall$) 及び F 値 (F) はそれぞれ式 4, 式 5, 式 6 のように定義される.

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

$$F = \frac{2 * Precision * Recall}{Precision + Recall} \tag{6}$$

表 2　類似していない例

文章 1	走行体の進行方向を逆に変える
文章 2	一定距離を直進したら停止する
手法 A 類似度	79%
手法 B 類似度	0 %

表 3　類似している例

文章 1	スタート命令を受け取るまで待機している
文章 2	開始合図を受け取るまで準備姿勢で待機する
手法 A 類似度	90%
手法 B 類似度	30%

4.4　実験結果

手法 A, 手法 B の適合率, 再現率, F 値を閾値ごとに計算した結果を図 3, 図 4, 図 5 に示す. 手法 A の場合, 閾値が 21%の時に F 値が最大値 0.13, 手法 B の場合, 閾値が 88%の時に F 値が最大値 0.21 となった. また, 各手法で F 値が最大をとる時の, 手法 A で類似と予測したゴールペア集合 (図中 AP) と, 手法 B で類似と予測したゴールペア集合 (BP) のベン図を図 6 に示す. なお, U は全てのペア集合である. 加えて, 手法 A, 手法 B で算出したそれぞれの類似度を図 7 に示す.

また, 正解と不正解となるいくつかのゴールペアの具体例と, その類似度算出結果を表 2, 3 に示す.

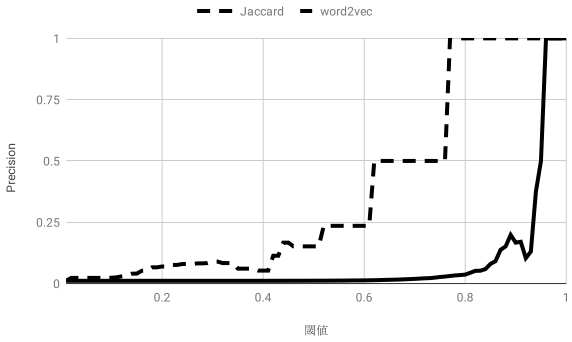

図 3　適合率

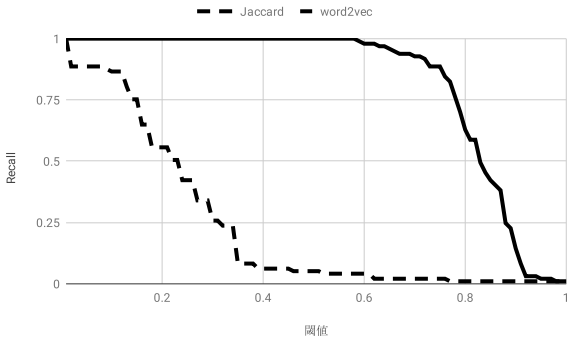

図 4　再現率

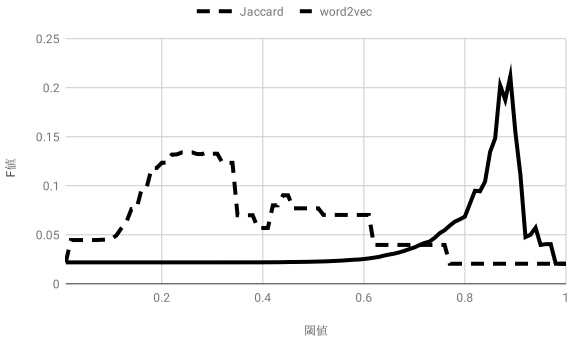

図 5　F 値

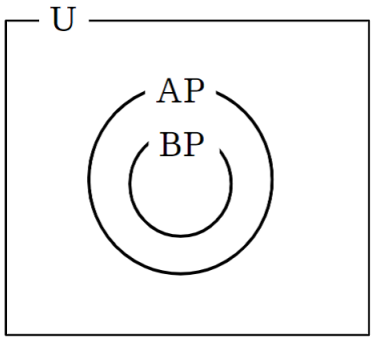

図 6　正解検知ペア集合の関係

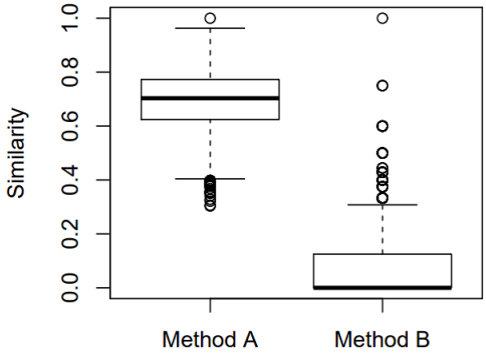

図 7　手法 A, 手法 B で算出した類似度

4.5　考察

4.5.1　両手法の比較

評価実験によって, 手法 B よりも手法 A の方が最大 F 値が大きいことから, word2vec を用いた手法 A の方が精度がより優れていることがわかる. 加えて, 図 6 より, 手法 B で類似として検出できたペアは, 全て手法 A で類似として検出できる. よって, 手法 A の方が精度がより優れていることが分かる.

4.5.2　ゴールペア具体例に対する考察

表 2 に示すように, 紐付いているコードに類似性が見られないゴールのペアが 79%と算出され, 類似ペアとして検出されやすい結果となった. 一方, 手法 B では類似度 0%となり, 類似と検出されにくい結果となった. このように, 手法 A では, 図 7 でも示されている通り, 全体的に類似度

が高く算出され, 類似性が見られないペアで類似度が高く算出される例が多く見られた. これは, 『走行』という単語と, 『直進』という単語が類似しているせいで, 文書のベクトルとして平均化したときに, 類似したベクトルになってしまうことが原因だと考えられる. 類似すべきゴールペアに対して絶対的に高い類似度を与え，類似すべきでないゴールペアに絶対的に低い類似度を与えるための方法としては，たとえば手法 A と手法 B を併用し，各類似度に重みを与えて平均化して最終的な類似度を算出する方法などを検討していきたい.

4.6 妥当性への脅威

4.6.1 内的妥当性への脅威

本研究では, 形態素解析器に MeCab[24], [25] を用いた. これは要求モデルのような ET ロボコン固有の単語には対応していない. 例えば, 『尻尾モータを停止させる』は {尻尾, モーター, 停止, する}といったよう単語集合が得られる. しかし, ET ロボコンでは『尻尾モータ』として単語が定義されている. このため, 学習時や単語ベクトルを得る際に, 適切な単語単位で意味ベクトルの学習が行えなかったり, 適切な意味ベクトルが得られない可能性がある. MeCab の機能であるユーザー辞書によって形態素解析の結果はカスタマイズすることは可能だが, 人手で行うと膨大な作業量になるため現実的ではない. よって, 今後はドメイン固有の未知語を捉え, 形態素解析の結果をドメインに適する方法について模索する必要がある.

また, 単語の意味ベクトルを得るには鈴木らが配布している学習済みモデル [26], [27] を用いた. このモデルは Wikipedia の記事から学習されている. そのため, ET ロボコンのようなドメイン固有の単語の意味を適切に捉えられているか定かではない. 例えば, 前述した『尻尾』という単語も, ET ロボコンではしばしば『モータ』という単語の近くに登場する. しかし, Wikipedia では動物の名前や動物に関する記事内に出現するため, ET ロボコンというドメインで得るべき意味ベクトルとなっていない可能性がある. 加えて, 学習結果はコーパスは勿論, エポック数や, ベクトルの次元数等のパラメータなどに大きく依存する. そのため, 精度は単語の意味ベクトルの学習結果に大きく依存する. 今後は fastText 等 [18], [31] に代表される word2vec 以外の様々な学習法について試すと共に, パラメータによって精度が変化するかも調査していきたい.

4.6.2 外的妥当性への脅威

本稿で評価対象にしたのは, それぞれ異なる開発者が作成したゴールモデル 2 つである. この適用事例では, word2vec の方が精度が良いという結果が得られたが, 他の事例でも同様の評価結果が得られるかは定かではない. したがって, 今後は様々な事例に適用して評価を重ねていきたい.

5. 関連研究

文章内に現れる単語を文脈から推定する重み行列 (学習済みモデル) を注目単語と文脈の情報を用いて機械学習によって得ることができ, 字面が違う単語間でも正確に類似度を測ることが可能になる. この機械学習によって得られた重み行列が各単語の意味を表すベクトルである. このように単語の分散表現を得る方法が研究されている [18], [19], [20]. 学習モデルは様々で, 例えば CBoW モデル [19], [20] は文脈から注目単語を推定するモデルで, Skip-gram モデル [20] は注目単語から文脈を推定するモデルである. 他にも fastText[18] と呼ばれる手法がある. これは注目単語の部分語も捉えて, 単語の活用形なども考慮して学習する手法で, 学習も高速である. 本研究では要求仕様間の類似度を測定するために単語の分散表現を得る手法を利用した.

中村ら [16] は Software Product Line 開発 [1], [2] を行う際に顧客が欲しい機能を選びやすいように結合ゴールモデルを作る目的で共通ゴールモデルを検出する手法を提案している. より詳細には, 文書の類似度をベースにゴールの親子関係を利用した共通ゴールモデルの検出する. これはゴール間の類似度をジャッカード係数 [17] を用いて算出し, 一定の閾値を設け, 算出した値が閾値以上であれば共通ゴール, それ以下であれば可変ゴールに判別する. また, 『子ゴールの中に明らかに共通なゴールが 1 つでもあれば, その親ゴールは共通ゴール』といったゴールモデルの親子構造を利用した判別ルールを与えて, 判別精度を上げるという手法である. 本研究ではゴール間の類似度算出に着目している点は共通であるが, 単語集合の表現上の一致ではなく, 意味の類似を捉えた類似度算出手法の評価を趣旨としている.

横森ら [9] や楠木ら [8] はソースコードに登場する変数名などのキーワードを基にして再利用可能性があるソースコードの検索を行うシステムの提案・開発を行っている. 本研究ではソースコードの情報は使わず, ソースコードに紐づくゴールモデルを介して, 開発初期から再利用可能なソースコードを特定することによって, 開発の効率化を目指すものである.

Gu らは CodeNN というニューラルネットワークの提案及びそれを用いた自然言語によってコード片を検索する手法を提案した [4]. 既存のソフトウェアリポジトリに存在する Javadoc に書かれているコード片と, その説明を用いて学習を行うものである. 事前に特定の言語で記述されたコード片とその説明のコーパスが必要で, 実装言語が限られてしまう. 本研究では実装言語に縛られない手法の確立を目標とする.

6. まとめ

本稿では word2vec を用いたゴールモデル内のゴール間類似度算出手法を示すと共に, 集合類似度を用いた手法との比較評価実験を行った. 評価の結果, word2vec を用いた提案手法の方が精度が優れていることがわかった.

7. 今後の課題

word2vec を用いる手法が集合類似度を用いる手法よりも精度が優れていることを実験的に示したが, 最大 F 値が 0.21 と好ましい結果ではない. 更には, 類似していないペアが類似度が高く算出されるなどが原因で誤検出が発生するなどの課題は多い. 今後は, ドメイン固有の単語への対応方法の調査及び考案と, 意味ベクトル学習の手法について調査を重ね, fastText[18], [31] のような別の手法も試していきたい. また, 手法内で利用した文書のベクトル化は, 語の順序が無視されてしまう等の問題点がある. この点については, SCDV と呼ばれるより精度の良い手法が提案されている [34]. このような手法も取り入れて精度の向上を図りたい.

また, 本稿での評価対象は構造的に記述されたゴールモデルである. 本研究では, テキストのみを用いたが, KAOS 法特有の構造的特徴は活かされていない. よって今後は KAOS 法の構造的特徴 [16], [33] も活かした手法も試行したい. 例えば, KAOS 法で提唱されている洗練パターン [21], [32] が一致しているゴールを抽出して, それらのゴール間で類似度を算出する等が挙げられる.

参考文献

[1] Klaus Pohl, Gunter Bockle, Frank VanDer Linden: *Software Product Line Engineering:Foundations, Principles And Techniques*, (Springer 2005)
[2] 岸知二: "ソフトウェアアーキテクチャ", 共立出版 (2005)
[3] 高橋直久, 丸山勝久: "ソフトウェア工学", 森北出版 (2010)
[4] Xiaodong Gu, Hongyu Zhang, Sunghun Kim: *Deep code search*, ICSE '18 Proceedings of the 40th International Conference on Software Engineering, pp933-944(2018) pp.933-944
[5] Apache Lucene: 入手先 ⟨http://lucene.apache.org/⟩, (参照 2018-11-30)
[6] Linstead, E., Bajracharya, S., Ngo, T. et al.: *Sourcerer: mining and searching internet-scale software repositories*,Data Mining and Knowledge Discovery, pp300-336(2009)
[7] Sonia Haiduc, Gabriele Bavota, Andrian Marcus, Rocco Oliveto, Andrea De Lucia, Tim Menzies:*Automatic query reformulations for text retrieval in software engineering*,ICSE '13 Proceedings of the 2013 International Conference on Software Engineering, pp842-851(2013)
[8] SPARS プロジェクト, 入手先 ⟨http://sel.ist.osaka-u.ac.jp/SPARS⟩, (参照 2019-01-30)
[9] 横森 励士: "Java ソフトウェア部品検索システム SPARS-J", 電子情報通信学会論文誌 D-I, VolJ87-D-I, No.12, pp1060-1068(2004).
[10] Steven P. Reiss: *Semantics-based code search*, ICSE '09 Proceedings of the 31st International Conference on Software Engineering, pp243-253(2009)
[11] 土屋 良介, 鷲崎 弘宜, 深澤 良彰, 加藤 正恭, 川上 真澄, 吉村 健太郎: "派生プロダクト群における要求・実装間のトレーサビリティリンク抽出", 電子情報通信学会ソフトウェアサイエンス研究会 11 月 (IEICE-SIGSS), 2012.
[12] Ryosuke Tsuchiya, Hironori Washizaki, Yoshiaki Fukazawa, Tadahisa Kato, Masumi Kawakami, Kentaro Yoshimura: *Recovering Traceability Links between Requirements and Source Code in the Same Series of Software Products*, Proceedings of 17th International Software Product Line Conference (SPLC 2013), pp.121-130
[13] G.Antoniol, G.Canfora, G.Casazza, A.De Lucia, E.Merlo: *Recovering Traceability Links between Code and Documentation*, IEEE Transactions on Software Engineering, vol.28, no.10, pp.970-983, 2002.
[14] 伊藤 弘毅, 田邉 浩之, 波木 理恵子, 鷲崎 弘宜, 深澤 良彰: "トレーサビリティリンク回復を通じたトレーサビリティ測定と改善支援", コンピュータソフトウェア, 2013 年 30 巻 3 号 pp.123-129
[15] Fatma A. Mihany, Hanan Moussa, Amr Kamel, Ehab Ezat: *A Framework for Measuring Similarity between Requirements Documents*, INFOS '16 Proceedings of the 10th International Conference on Informatics and Systems, pp334-335
[16] 中村祐貴, 本田耕三, 中川博之, 田原康之, 大須賀明彦:"ソフトウェア再利用に向けた共通ゴール判別手法の提案", コンピュータソフトウェア, 31 巻 (2014) 2 号 pp. 267-283
[17] M. Ilyas and J. Kung: *A Similarity Measurement Framework for Requirements Engineering* 2009 Fourth International Multi-Conference on Computing in the Global Information Technology, Cannes, La Bocca, 2009, pp. 31-34.
[18] Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov: *Enriching Word Vectors with Subword Information*, Transactions of the Association for Computational Linguistics, vol. 5, pp.135-146(2017)
[19] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean: *Efficient Estimation of Word Representations in Vector Space*, In Proceedings of Workshop at ICLR, 2013.
[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean: *Distributed Representations of Words and Phrases and their Compositionality*, In Proceedings of NIPS, 2013.
[21] Axel van Lamsweerde:*Requirements Engineering: From System Goals to UML Models to Software Specifications*, Wiley(2011).
[22] 本田 耕三, 中川 博之, 田原 康之, 大須賀 昭彦: "洗練パターンによるゴール指向ユースケースモデリング", ソフトウェアエンジニアリングシンポジウム 2014 論文集, pp45-50(2014).
[23] 堀田 大貴, 本田 耕三, 平山 秀昭, 清 雄一, 中川 博之, 田原 康之, 大須賀 昭彦: "リファインメントパターンを利用した KAOS ゴールモデルから BPMN モデルへの変換", コンピュータ ソフトウェア 32 巻 4 号 pp141-160(2015).
[24] MeCab: 入手先 ⟨http://taku910.github.io/mecab/⟩, (参照 2019-01-30)
[25] Taku Kudo, Kaoru Yamamoto, Yuji Matsumoto: *Applying Conditional Random Fields to Japanese Morphological Analysis*, Proceedings of the 2004 Conference

on Empirical Methods in Natural Language Processing (EMNLP-2004), pp.230-237 (2004.)
[26] Masatoshi Suzuki, Koji Matsuda, Satoshi Sekine, Naoaki Okazaki and Kentaro Inui. A Joint Neural Model for Fine-Grained Named Entity Classification of Wikipedia Articles. IEICE Transactions on Information and Systems, Special Section on Semantic Web and Linked Data, Vol. E101-D, No.1, pp.73-81(2018)
[27] singletongue/WikiEntVec, 入手先 ⟨https://github.com/singletongue/WikiEntVec⟩, (参照 2018-12-20)
[28] 日本語 Wikipedia: 入手先 ⟨https://ja.wikipedia.org⟩, (参照 2019-01-30)
[29] 加藤和平, 大島考範, 二宮崇: "word2vec と深層学習を用いた大規模評判分析", 言語処理学会年次大会発表論文集 21 巻 pp.3-4 (2015)
[30] ET ロボコン 2018 公式サイト 入手先 ⟨http://www.etrobo.jp/2017/⟩, (参照 2019-12-20)
[31] facebookresearch/fasttext, 入手先 ⟨https://github.com/facebookresearch/fastText⟩, (参照 2019-01-30)
[32] Emmanuel Letier: *Reasoning about Agents in Goal-Oriented Requirements Engineering*, University College London(2002)
[33] 石川公一, 小形真平, 岡野浩三, 鷲崎弘宜: "洗練パターンの適用履歴に基づくゴール間の類似度算出手法", 信学技報, vol. 118, no. 69, KBSE2018-3, pp. 7-11(2018 年)
[34] Dheeraj Mekala, Vivek Gupta, Bhargavi Paranjape, Harish Karnick: *SCDV : Sparse Composite Document Vectors using soft clustering over distributional representations*, Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp.659âĂŞ669

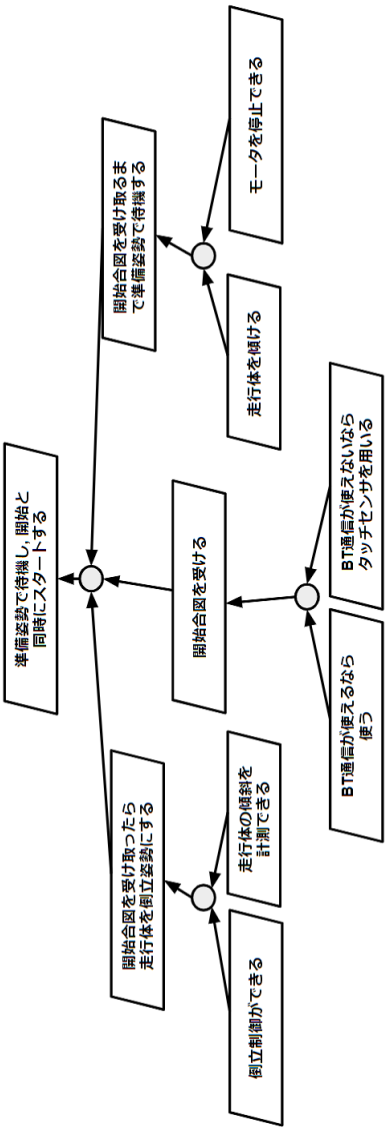

図 8 チーム 1 のゴールモデル (一部)

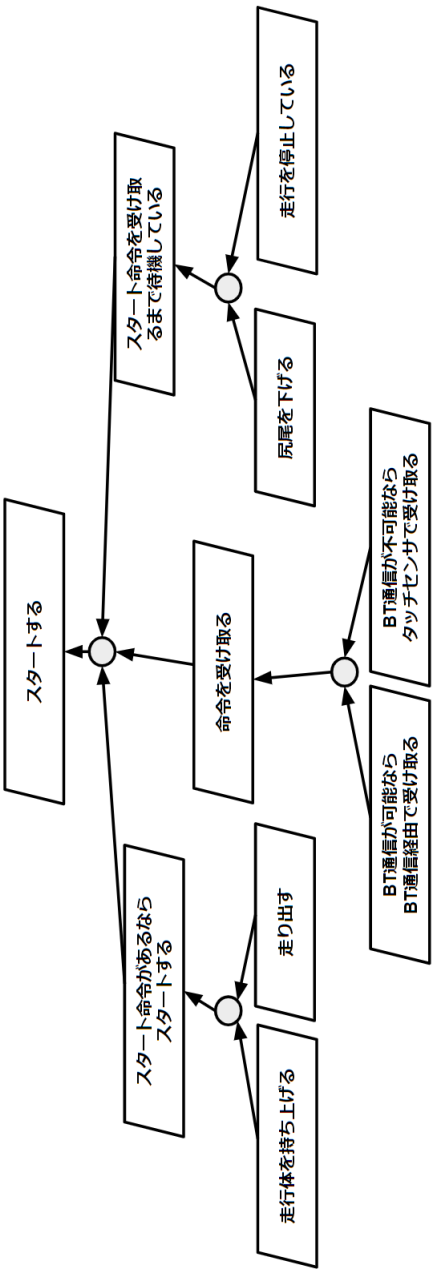

図 9　チーム 2 のゴールモデル (一部)