社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2009－SE－163 (17)
2009／3／18

ドメインオントロジを用いた 自然言語文書とソースコード間の追跡可能性の復元

吉川 嵩志[†1] 林 晋平[†1] 佐伯 元司[†1]

ソフトウェアに関わる自然言語文書とソースコードの間の追跡可能性を復元する手法が望まれる．本稿では，ドメインオントロジを用いてソフトウェアの機能を記述した自然言語文書とソースコードとを対応付ける手法を提案する．文書中の単語とコード上の識別子との類似性に基づく関係と，コード上のメソッド呼び出し関係の評価にオントロジによる意味的関係を考慮することで，詳細ではない文書に対しても高精度の対応付けを行う．オープンソースソフトウェア JDraw に対する適用事例では，オントロジを用いない場合と比較して高精度の対応付け結果を得た．

Recovering Traceability Links between a NL Document and a Source Code by Using Domain Ontologies

Takashi Yoshikawa,[†1] Shinpei Hayashi[†1] and Motoshi Saeki[†1]

Recovering traceability links between a source code and their NL documents is significant. In this paper, we propose a technique for recovering the links between functional descriptions and a source code using domain ontologies. By using semantic relationships of the domain ontologies in addition to method-call relationships and the similarity between an identifier on the code and words in the descriptions, we can detect source code fragments corresponding to the descriptions. Through a case study using open-source software JDraw, we obtained results of higher quality than without ontologies.

1. はじめに

ソフトウェアに関わる自然言語文書とソースコードの間の追跡可能性を復元する手法が望まれる．可読性の高い文書と複雑なソースコードとの間の追跡可能性が確立されていることによって，様々なソフトウェア保守・再利用において支援が可能となる．通常はソフトウェア開発中に追跡のための情報を各成果物に追加する必要があるが，実際に追跡を行う必要が生じるまではその効果が直接現れないため，それらの情報が追加されないことが多い．そのため追跡可能性が確保できていないソフトウェアが多数存在する．

既存の復元手法の多く[1)-3)]は，ソースコードと仕様書等の詳細な文書とを対応付けるため，詳細でない文書には適用できない．既存手法では，詳細な文書における文書構造を用いた対応付けを行っている．しかし，ソフトウェア開発において詳細な文書が残らないケースが多くある．特に，近年広がっているアジャイル開発では，仕様書等の詳細な文書は残らない傾向にあるため[4)]，詳細でないマニュアル等の文書との対応付けが重要である．

特に，詳細でない文書と対応付けを行うには，該当ソフトウェアが扱う問題領域（ドメイン）の特徴を考慮する必要がある．詳細でない文書には，コードと文書の両者が含む単語の類似性に基づく一般的な対応付け手法を適用することが可能である．しかし，この手法では，適切な単語が文書中に含まれていない際に未検出が生じ，再現性が低下する．一方，未検出防止のために関数やメソッドの呼び出し関係を用いた場合，該当の機能と関係のない呼び出しにより誤検出が生じ，適合性が低下する．ドメインを考慮することでこれら双方を向上させ，高精度の対応付けを行うことができる．

本稿では，ドメインオントロジを用いて自然言語文書とソースコードとを対応付ける手法を提案する．文書中の単語とコード上の識別子との類似性に基づく関係と，コード上のメソッド呼び出し関係の評価にオントロジによる意味的関係を考慮することで，高精度の

†1 東京工業大学 大学院情報理工学研究科 計算工学専攻
Department of Computer Science, Graduate school of Information Science and Engineering, Tokyo Institute of Technology

対応付けを行う．提案手法では，文書中の単語とコード上の識別子を，オントロジ上の概念に写像する．これを用いて，メソッド呼び出し関係のうち，文書中の単語を含むものとオントロジ上の概念間の関係と対応するものの集合を文書と対応付く候補として複数抽出する．候補はオントロジと対応付く度合いによって順序付けられて開発者に提示される．これにより，開発者は効率よくドメインに関わる機能と対応するコード片を特定することができる．

本稿の以降の構成を示す．2節ではドメインオントロジを使用したソフトウェア追跡可能性の復元手法について述べる．3節では提案手法についてより詳しく述べ，4節でその評価を行う．5節では関連する研究を紹介する．6節で本稿をまとめ，今後の課題を示す．

2. ソフトウェア追跡可能性復元へのドメインオントロジの使用

2.1 自然言語文とコード片集合との対応付けにおける問題

詳細でない自然言語文書とソースコードとを対応付けるためには，単独1文レベルの自然言語文とソースコードとを対応付けられることが必要である．よって，本稿では1文の自然言語文とソースコードとの対応付けを行う手法を考える．自然言語で表された文書とソースコードとを対応付ける先行研究において，対応付けを行うためには詳細な文書が必要とされている．それは，ソースコードが詳細であるために，対応づく文書も同様に詳細でないと対応付けることが難しいからである．

図1に追跡可能性の復元例を示す．この例は，Javaで書かれた描画ソフトウェア中の楕円描画機能の実現例である．例では，マニュアル中の楕円描画機能にかかわる記述 “user can draw a plain oval” に対応するコード片は図右の5メソッドの組み合わせである．該当の機能はメソッド `drawOval` を起点として構成されており，その中では `getColor` が `getColorPallete` によりパレットから描画色を取得し，`getCanvas` で得たキャンバス中の各ピクセルに `setPixel` で着色する．マニュアル中の記述からこの集合を特定することで，例えば描画ソフトウェア開発者が新規の機能を自身のソフトウェアに追加しようと考えたとき，同様の機能のコード例を効率よく確認でき，その構築法の理解が助けられ，再利用が容易になる．

機能を実装している箇所の特定には，機能を実際に実行することによる動的な解析を行うことが有効であるが，これは困難な場合が多い．動的解析にはテストケースの作成が必要であり，そのためには該当ソフトウェアの振る舞いを理解している必要がある．しかし，その正確な理解のためには使用法が記述された詳細な文書等から知識を得る必要があるが，そのような文書は残っていることは少ない．よって，静的な解析により実装箇所を特定する必要がある．

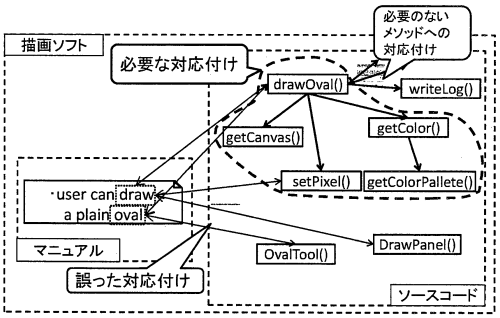

図1 一文とソースコードとの対応付け例

一般的に用いられる識別子と単語の類似性のみでは，高精度の検出は難しい．文書が詳細でなく，文書構造や版管理における同時変更の履歴等に存在するメタデータを仮定できないものであれば，文書から得られる情報は高々，文書を分解することによって得られる単語程度であるため，ソースコードとの対応付けは，その単語とソースコード上の識別子との類似性のみに基づき行われる．図1の例では，マニュアルの一文と対応付くべきメソッドが，図1のように5つ存在する．メソッド `drawOval` 中の識別子は単語 “draw” と “oval” を含み，同様にメソッド `setPixel` はメソッド内に単語 “draw” を含む識別子を持っているため，文書中の単語と対応付けることができる．しかし，残りの3メソッドは文中のいずれの単語とも対応付かない．更に，描画パネルを用意するメソッド `DrawPanel` や楕円ボタンを設定するメソッド `OvalTool` といった，楕円を描く機能の再利用とは直接関係のないメソッドも対応付いてしまう．

未検出のメソッドはメソッド呼び出しの関係を用いて検出できるが，これは一方で適合率の低下を招く．機能は往々にしてメソッド呼び出しによって実現されるため，例においては，メソッド `drawOval` から必要なメソッド二つへの呼び出しが存在し，これをたどることによって正解を網羅できる．しかしこの場合，必要のない対応付けも行ってしまう．例えば，ログに書き込むメソッド `writeLog` は該当の機能とは異なる関心事に基づくメソッド呼び出しであり，該当機能の典型的な構成を特定したい開発者にとって不要な情報で

ある可能性が高い．しかし，メソッド呼び出し関係のみではその差異が判断できない．

以上に述べたことから，一文レベルの文書とソースコードとの対応付けには他の要素が必要である．

2.2 ドメインオントロジを用いた対応付け

ソースコードと自然言語文書との対応付けにドメインオントロジを利用することで，より精度の高い対応付けができる．ドメインオントロジは，ソフトウェアが対象とする専門領域に関する知識を体系化しており，概念とその間の関係から構成される構造を持つ．概念には対象ドメインに関する語彙が単語として関連付いている．

ドメインオントロジ上ではその専門分野における動作を構造的に表現できる．例えば**図 2** に示す描画ソフトウェアのドメインオントロジには，描く (draw) 動作のため，対象として楕円 (oval) と関係があり，描く色 (color) と描く先のキャンバス (canvas) と関係があることが表現されている．この構造とソースコードとを対応付けることで，動作を表現しているソースコードの箇所を得る．

ドメインオントロジの構造とソースコードとの対応付けは，最初にオントロジ上の概念に関連付いた単語と，ソースコード上の識別子とをそれらの類似性により対応付け，次にその対応付けから概念間の関係とメソッド呼び出し関係とを対応付けることにより行う．図 2 の例では，メソッド `drawOval` と概念 draw とが対応付き，メソッド `getCanvas` と概念 canvas とが対応付く．その対応付けをもとに，`drawOval` から `getCanvas` へのメソッド呼び出し関係が，概念 draw と canvas との間の関係と対応付く (対応 a)．同様に，メソッド `drawOval` から `getColor` へのメソッド呼び出し関係も対応付く (対応 b)．対応 b については，`getColor` と `getColorPallete` がいずれも単語 “color” と対応付き，かつこれらの間にメソッド呼び出し関係が存在することから，色に関する役割を共有するメソッドとして，メソッド `getColorPallete` を含めて得る．以上の手順により，文書と対応付くメソッド集合を得ることができる．

本稿では機能について記した自然言語文書と対応付くメソッド集合を得ることを目的とする．Java 等の OOP 言語によって開発されたソフトウェアにおいて，機能は往々にしてメソッドの呼び出しによって実現される．よって以上に述べたように，自然言語文書としてソフトウェアの機能について英語で書かれた文と，Java 言語で実現されたソフトウェアのソースコードとを対応付けることができる．

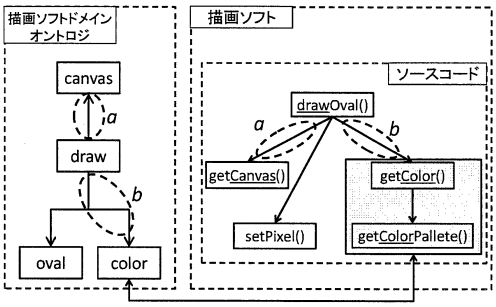

図 2 オントロジとソースコードの対応付け

3. 提 案 手 法

3.1 概　　要

本手法の入力は，ソースコード，自然言語文書，そしてオントロジである．出力は，優先順位付きのコールグラフのサブグラフ集合である．

提案手法の概要を**図 3** に示す．図中の番号は，本稿の節番号を表す．以降にそれぞれの手順毎に説明する．まず，ソースコードからメソッド呼び出し関係を抽出する．次にソースコードと文書から単語を抽出する．抽出された単語とメソッド呼び出し関係に併せ，オントロジを用いて対応付けを行うことにより，文書と対応付く候補となるサブグラフを複数取得する．最後に，そのサブグラフ集合に優先順位付けを行う．優先順位の高い複数の候補を調査することにより，開発者は効率よく文書と対応付くメソッド集合を得ることができる．

3.2 メソッド呼び出し関係の抽出

ソースコードを静的解析することにより，全メソッド集合 M とそれらの呼び出し関係 $\xrightarrow{c} \subseteq M \times M$ を抽出し，コールグラフ $\langle M, \xrightarrow{c} \rangle$ を得る．ここで，Java をはじめとする OOP 言語で開発されたソフトウェアを扱うためには，メソッドのオーバーライドによる隠蔽を考慮する必要がある．よってあるメソッド $m \in M$ とそれをオーバーライドするメソッドの集合 $override(m) \subseteq M$ も併せて抽出しておく．ここで，後の説明を簡単とするため，m が抽象メソッドでない限り $m \in overide(m)$ とする．

3.3 単語の抽出

文書からは，英文をトークンに分割することにより単語を抽出する．区切り文字は空白文字，タブ文字，改行文字とする．トークンに分割された部分文字列集合を全て単語として抽出する．

ソースコード中の識別子からも単語を抽出する．識

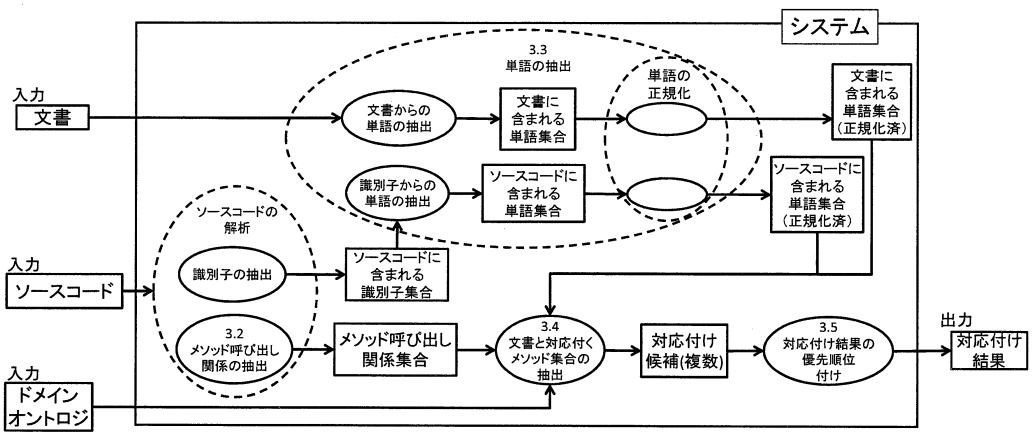

図 3 提案手法概要

別子は，プログラムの中でクラス，フィールド，メソッド，ローカル変数等に付与される名前である．識別子は，おもにアルファベットや数字を用いて構成される．プログラマは識別子を読んだだけでその対象の意味を把握しやすいよう，対象をよく表す単語を並べて識別子を構成することが多いため，これを文書やオントロジとの対応付けに利用できる．　Java 命名規則[5),6)]に基づき，識別子を大文字，もしくは "_" (アンダースコア) を区切り文字として単一語に分割し，単語集合を抽出する．

また，識別子の種類や出現回数も対応付けに用いる．ソースコード内にはさまざまな種類の識別子が存在し，その種類ごとに重要度は異なる．例えば，クラス定義名やメソッド定義名として使われている識別子は対象が関わる機能の意味を強く表しており，重要度が高い．逆に，メソッドの参照名やフィールドに使われている識別子は重要度が低いと考える．また，多く出現している識別子は機能等の重要な情報を伝達している可能性が高い．これらの情報を，ソースコードから抽出された単語に関連付けておく．

最後に，識別子と文書から得た単語集合を正規化及びフィルタリングして保存しておく．正規化として，1) 単語の小文字化，2) 原形への変換，3) 同義となる語の一つの語への統一を行う．3 のために，あらかじめ同義語辞書を用意しておく．また，"a" や "the" 等の不要な単語をストップワードリストとして用意しておき，単語集合から除外する．

このようにして，文書から単語集合 $D \subseteq W$ を，メソッド m のソースコードからその単語集合 $wd(m) \subseteq W$ を得る．ここで，W は全単語の集合である．

3.4 文書と対応付くメソッド集合の抽出

ソースコードから抽出した単語とメソッド呼び出し関係，文書から抽出した単語と併せて，オントロジを使用することで，精度のよい対応付けを目指す．オントロジ上の概念は，単語名としてラベルを持つ．ここでは簡単のため，オントロジを単語上の関係ネットワーク $\xrightarrow{o} \subseteq W \times W$ と定義する．

オントロジを用いてメソッドの役割を考慮しながら探索することにより，機能と対応するメソッド呼び出し関係の集合を得る．アルゴリズムを**図 4** に示す．アルゴリズムでは，入力文書 D に含まれる単語を含むメソッドを開始地点とし，メソッド呼び出し関係をもとにコールグラフを探索する (3 行目)．また，このアルゴリズムでは，そのメソッドの役割として単語をメソッドと対応付け，探索木の根に付加しておく (4 行目)．探索により，以下の 3 条件をみたすメソッド呼び出し関係を抽出し (16 行目)，抽出された呼び出し関係の集合で構成されるコールグラフのサブグラフを機能を実現している候補とする．

(1) D が含む単語を持つメソッドへの呼び出し関係
(2) オントロジの関係と対応付け可能なメソッド呼び出し関係
(3) 同様の役割を引き継ぐメソッドへの呼び出し関係

文書中の単語は重要なためこれを含むメソッドへの呼び出し関係は候補に含める (1)．また，オントロジには機能の動作が表現されており，図 2 に示した例のように，オントロジの関係と対応付いたメソッド呼び出し関係も重要として候補に含める (2)．最後に，同様の役割，すなわち単語を共有しているメソッド間のメソッド呼び出しは，その呼び出しにより単語の示す役割を実現している可能性が高く，重要と考え候補に含

```
1  Function: extract−subgraphs
2    CG ← ∅
3    for all m ∈ ⋃_{w∈D}{m|w ∈ wd(m)}
4      root ← m:D ∩ wd(m)
5      CG ← CG ∪ traverse(root, ⟨{root}, ∅⟩)
6    end
7    return CG
8  end
9  Function: traverse(m:w, ⟨N, E⟩)
10   if ∃m:w' ∈ N. w ⊆ w' then return ∅
11   G ← {⟨N, E⟩}
12   for all m' s.t. m -c-> m'
13     G' ← ∅
14     M_o ← override(m') ∩ {m''|m'':w'' ∈ N}
15     if M_o = ∅ then M_o ← override(m')
16     for all m'' ∈ M_o
17       w' ← wd(m') ∩ (D ∪ {w|w'∈w.w' -o-> w} ∪ w)
18       if w' ≠ ∅ then
19         for all ⟨N, E⟩ ∈ G
20           N' ← N ∪ {m'':w'}
21           E' ← E ∪ {⟨m:w, m'':w'⟩}
22           G' ← G' ∪ traverse(m'':w', ⟨N', E'⟩)
23         end
24       end
25     end
26     if G' ≠ ∅ then G ← G'
27   end
28   return G
29 end
```

図 4 機能実現のためのコールグラフ発見アルゴリズム

める (3). これらの条件に関わった単語群 $\boldsymbol{w} \in 2^W$ を呼び出し先メソッド $m \in M$ の役割として与え，探索を繰り返すことにより役割が付与されたメソッド集合 $N \subseteq M \times 2^W$ を得る．N をノード集合とする有向グラフ $\langle N, E\rangle$ が機能と対応付く候補となり，本稿ではこれを機能コールグラフと呼ぶ．例えば，図 2 の例では，メソッド `drawOval` から探索を開始することにより，5 つのメソッド全てを得ることができる．この時，`drawOval` には {“draw”, “oval”}，`setPixel` には {“draw”}，`getCanvas` には {“canvas”}，`getColor` と `getColorPallete` には {“color”} がそれぞれ役割として対応付く．

また，図 4 のアルゴリズムでコールグラフを得る際，隠蔽性を考慮する．隠蔽している可能性のあるメソッドの数だけ，コールグラフを複製し (14–16 行目)，可能性のあるコールグラフを全て生成する (19–23 行目)．

3.5 対応付け結果の優先順位付け

3.4 節のアルゴリズムより得た機能コールグラフ集合 CG の要素毎に重み付けを行うことで，それらの優先順位付けを行う．優先順位付けの観点は以下の三つである:

観点 1 重要な役割を持つメソッドを高評価

例えば，図 1 及び図 2 の例で，機能コールグラフへの重み付けについて考える．図 4 のアルゴリズムにより得た，メソッド `drawOval` の役割は “draw” と “oval”，メソッド `getCanvas` の役割は “canvas” であり，いずれも役割語はメソッド名として現れている．メソッド名はそのメソッドが関わる機能を表す名前である可能性が高く，重要度が高い．よって，メソッド `drawOval` と `getCanvas` には高い重みが与えられる．同様の理由により，メソッド `drawOval` と `getColor` にも高い重みが付けられる．逆に，メソッド `setPixel` の役割語 “draw” はメソッド内で用いられている程度の重要度であるため，メソッド `setPixel` には低い重みが付けられる．

表 1 識別子の種類と重みの大きさ

識別子の種類 t	$ts(t)$	識別子の種類 t	$ts(t)$
メソッド名（定義）	2.0	メソッド名（参照）	0.1
コンストラクタ名	2.0	パッケージ名	0.1
クラス名（定義）	1.5	クラス名（参照）	0.1
変数名（参照）	0.1	変数名（宣言）	0.05

役割語 w によりメソッド m に与えられる重み ww を以下に示す:

$$ww(m,w) = \sum_{t \in T} ts(t) \cdot \log\{1 + cnt(m,w,t)\}$$

識別子の種類集合 T と識別子の種類 $t \in T$ に応じて与えられる重み $ts(t)$ については**表 1** に示す．$cnt(m,w,t)$ はメソッド m において単語 w が種類 t で用いられた回数を表す．ww において回数の対数をとっているのは，同じ変数名が同じ種類で多量に用いられた場合に過剰な重みを与えることを防ぐためである．

観点 2 オントロジ上の関係にマッチしている役割のメソッド呼び出しを高評価

メソッド呼び出し関係の呼び出し元，呼び出し先それぞれの役割語をオントロジを用いて評価を行い，重みを算出する．メソッド `drawOval` と `getCanvas` それぞれの役割語 “draw” と “canvas” の間にはオントロジ上で関係がある．このとき，これら 2 つの間のメソッド呼び出し関係はオントロジの構造にマッチしているとみなし，これらの間のメソッド呼び出し関係にも高い重みが付けられる．逆に，メソッド `getColor` と `getColorPallete` の役割語は共に “color” であるが，これはオントロジの構造とマッチしておらず，低い重みが付けられる．オントロジにも入力文書にもマッチするものを最高として，2 つの単語 w_1 と w_2 か

の関係への重み wr を以下のように与える:

$$wr(w_1,w_2)=\begin{cases}0.5 & if\ \neg(w_1 \overset{o}{\rightarrow} w_2)\\ 1 & if\ w_1 \notin D \wedge w_2 \notin D\\ 2 & if\ w_1 \notin D \vee w_2 \notin D\\ 8 & otherwise\end{cases}$$

また，特に w_1 と w_2 が同じ単語であるとき等，コールグラフ内における全メソッド呼び出し関係において，同一の役割の呼び出し関係が大量に存在する場合がある．その対処のため，そのような呼び出し関係

$cw(w_1,w_2,E) =$

$\{e \mid e \equiv \langle m_1{:}\boldsymbol{w_1}, m_2{:}\boldsymbol{w_2}\rangle \in E \wedge w_1 \in \boldsymbol{w_1} \wedge w_2 \in \boldsymbol{w_2}\}$

の数 $|cw(w_1,w_2,E)|$ で重みを正規化する．

観点 3 文書中の語をより網羅する結果を高評価

文書中の単語のうちいくつが機能コールグラフ内のメソッドと対応付いてるかを基にした補正を行う．図 1，図 2 の例における自然言語文書からは，“draw”，“plain”，“oval” の 3 語が抽出され，そのうち 2 語が機能コールグラフ内のメソッドと対応している．よって，機能コールグラフ全体の重みにこの割合 p を乗じる:

$$p(N)=\left|D \cap \left\{\bigcup_{m:\boldsymbol{w}\in N} \boldsymbol{w}\right\}\right| / |D|$$

機能コールグラフ $\langle N,E\rangle$ に優先順位を付けるため，以上の観点に基づき機能コールグラフへの重み付けを行う．機能コールグラフの重みは，機能コールグラフに含まれる全メソッド呼び出し関係に重みを付ける．m_1 から m_2 へのメソッド呼び出し関係に対する重み mcw を以下のように求める:

$$mcw(\langle m_1{:}\boldsymbol{w_1}, m_2{:}\boldsymbol{w_2}\rangle, E) = \sum_{\langle w_1,w_2\rangle \in \boldsymbol{w_1}\times\boldsymbol{w_2}} ww(m_1,w_1)\cdot ww(m_2,w_2)\cdot \frac{wr(w_1,w_2)}{|cw(w_1,w_2,E)|}$$

それらの総和を求めた後に，観点 3 に基づく補正を行うことによりコールグラフの重み cgw を求める:

$$cgw(\langle N,E\rangle) = p(N)\cdot \sum_{e\in E} mcw(e,E)$$

4. 事例による評価

4.1 目的と問題設定

提案手法の有用性を確認するため，実装したツールを用いて，オープンソースソフトウェアの JDraw[7] を対象として本手法を評価する．JDraw は Java 言語で開発されたペイントツールである．事例として，JDraw の持つ機能について述べた文を入力文書とし，JDraw 1.1.5 のソースコードとの対応付けを行う．JDraw の総メソッド数は 1,450 である．

JDraw のホームページには，JDraw の持つ機能について述べた文が列挙されている．本事例で対応付ける文を**表 2** に示す．

本事例により，オントロジを用いることは有用であるか，正確な対応付けが行えているかを評価する．このため，対応付けにオントロジを用いる場合と，用いない，すなわち空のオントロジを用いる場合それぞれについて対応付けの精度を測る．オントロジには 38 の概念とそれらの間の 45 の関係を持つ描画ツール用のものを作成して用いる．本手法で対応付いたメソッド集合 M_r を正解のメソッド集合 M_c と比較する．適合率 $|M_r \cap M_c|/|M_r|$ と再現率 $|M_r \cap M_c|/|M_c|$ を精度の指標として求める．本手法では優先順位が付いた複数の対応付け結果を得ることができるので，その上位 5 位までの結果に対し，適合率と再現率を求め，その中で適合率と再現率の調和平均（F 値）が最も高いものを結果として用いる．

4.2 ツールの実装

事例の評価を行うために，提案手法を実現するツールを Java で実装した．本ツールは，自然言語文書，ソースコードとオントロジを入力とし，文書に対応するメソッド集合を GUI 上でグラフ形式で出力する．コードの静的解析には，Eclipse JDT[8] を利用した．

4.3 結　　果

4.1 節で挙げた JDraw の機能を表す文それぞれと対応付けを行った結果を表 2 に示す．なお，文 7 とは対応付けを行うことができなかった．4.4 節で詳しく述べる．

4.4 評価と考察

提案手法による自然言語文書とソースコードとの対応付けは，オントロジを用いることで，全 7 文の対応付けのうち，5 文が 再現率 が 0.9 以上，3 文が 適合率 が 0.7 以上と高い値を示した．これにより，提案手法の有効性を示せている．

オントロジを用いない場合，単語に基づく対応付けのみを行うことになるが，その結果機能実現のために必要なメソッドを得られなくなってしまい，再現率は低くなる．文 1，文 2 との対応付けに注目すると，適合率はオントロジを用いない場合が 1，用いる場合が約 0.83 と，オントロジを用いた場合がやや劣っているが，再現率はオントロジを用いる場合が約 4–5 倍の値を示しており，オントロジを用いる場合の方がはるかに良い結果が出ているといえる．この点から，オントロジを用いる有用性を示せている．

文 3 との対応付けでは，オントロジを用いない場合は，両評価値が共に 0 であり，該当の機能とは関わりのない部分と対応付いているのに対し，用いる場合は対応付けを行えている．この結果からも，オントロジ

表 2 適用事例の評価結果

文書	オントロジ有り 適合率	オントロジ有り 再現率	オントロジ無し 適合率	オントロジ無し 再現率
1. “plain, filled and gradient filled rectangles”	0.8302	0.9361	1	0.1915
2. “plain, filled and gradient filled ovals”	0.8214	0.9574	1	0.2127
3. “image rotation”	0.3529	1	0	0
4. “image scaling”	0.5	0.5263	1	0.5789
5. “save JPEGs of configurable quality”	0.4	1	0.6667	1
6. “colour reduction”	0.7407	0.9524	0.7407	0.9524
7. “grayscaling”	–	–	–	–

を用いる有用性が示せている．しかし，オントロジを用いた結果の再現率は 0.35 程度と高くない．これは，機能 “image rotaion”，すなわち画像を回転させる機能は，回転後の画像の描画の機能を含んでいるが，今回用いたオントロジにはこの関係が含まれていなかったことによる．オントロジの質の向上によって，より高い精度を実現できると考える．

一方，オントロジに表現された動作がソフトウェアで実装されていない場合，オントロジを用いると対応付けの結果が悪化する．文 4 との対応付けでは，オントロジを用いた場合の適合率が 0.4，再現率が 1 であるのに対し，オントロジを用いない場合の適合率が約 0.67，再現率は 1 となり，後者が良い結果を示している．機能 “image scaling” は，JDraw では外部モジュールにより実現されており，この機能の典型的な動作がソースコードに実装されていない．提案手法では，外部モジュールの解析を行わないため，オントロジが悪影響を及ぼす．今後の課題として，外部モジュールへの呼び出しに対する評価を行うべきである．

また，複合語の考慮が求められる例もあった．文 7 では，文書並びにオントロジ内には “grayscaling” と同義の単語がそれぞれ含まれていたが，ソースコード上では，これに類する識別子は “GrayScale” であった．提案手法では，識別子は大文字を区切り文字として分解されるため，この識別子は “gray” と “scale” の 2 単語に分割されてしまい，文書及びオントロジと対応付かなかった．今後の課題として，複合語を考慮，もしくは単語の部分一致を許容した類似度の計算等を検討すべきである．

5. 関連研究

ソースコードと自然言語文書との対応付けを行うことでソフトウェア追跡可能性を復元する研究は複数存在するが，我々の知る限り，既存の手法では文書側の粒度や詳細さにおいて制約があり，本稿での目的には適用できない．

Witte らはオントロジを用いて，ソースコードと自然言語で書かれた仕様書を対応付ける手法を提案した [3]．オントロジはソースコードの構文構造を表現した概念を含むソースコード用のものと，プログラミングに関する諸概念，アルゴリズムやデータ構造，デザインパターンやアーキテクチャ等の情報を表現した文書用のものの 2 種類を用意しておく．ソースコードの構文要素をソースコード用オントロジと，仕様書から抽出した名詞や動詞を文書用オントロジと対応付けることで，推論により仕様書とソースコードの対応関係を発見することができる．この手法では，文書用オントロジを作ることにコストがかかってしまう．また，文書用オントロジはプログラミングに関する記述についての情報を集めたものであり，これに対応付ける文書にも同様に実装レベルの記述が求められる．

Zhao らはプログラムコードのマニュアル等のソースコードに関する記述が含まれた機能要求仕様とソースコード上の関数群を対応付ける手法を提案した [9]．この手法では，TF×IDF と *cosine* 類似度から機能要求に対してソースコードの関数ごとに類似度の高いものを探し，ソースコードから構築したコールグラフを用いて，探し出した関数から呼ばれる関数を集めることで，不足している関数を補完する．この手法で求められる機能要求仕様は，ソースコードに関する記述が含まれているプログラムコードのマニュアルのような粒度の細かい設計レベルの文書である．

田原らは，単語の類似性と文書構造の類似性を用いて仕様書とソースコードの要素を対応付ける手法を提案した [1]．IEEE830 に従って書かれた要求仕様書の章構造に合わせ，コードから構造を抽出し，それらの類似性により対応する構造の一部を特定することで精度を向上させている．この手法では，要求仕様書が IEEE830 の形式に従っていることが前提となっており，マニュアル等には対応できない．

6. おわりに

本稿では，Java 言語を用いて開発されたソフトウェアにおいて，ソフトウェアの機能が記述された自然言

語文書とソースコードとの対応付けを，ドメインオントロジを用いて行う手法を提案した．提案手法では，まずソフトウェアのソースコードからメソッド呼び出し関係を抽出し，更にソースコードと文書から単語を抽出する．文書とソースコードから得られた単語を用い，メソッド呼び出し関係から得られるコールグラフとドメインオントロジとの対応付けを行うことで，精度の高い対応付けを行った．

また本稿では，提案手法とその実装例を示し，JDrawとそのマニュアルを対象とし，適用事例の評価を行った．これによりオントロジを用いて，自然言語文書とソースコードとが精度よく対応付くことを確認した．

今後の課題を以下に示す．

更なる適用事例

本稿では，事例として手法を適用したソフトウェアが一つのみと少ない．同じドメインに関わる複数のソフトウェアへの手法適用，ならびにJava以外の言語で開発されたソフトウェアに対しての手法適用による本手法の汎用性の評価を行うことが望ましい．

手法の改善

事例による本手法の評価に基づき，外部モジュールと複合語に対応できるように手法を改善する必要がある．また，本稿では，提案手法を実現するためにいくつかの値を4節で例として設定し，その値に従って適用事例の評価を行った．更なる事例評価や実験により，それらの値を調整していく必要がある．同様に，自然言語文書とソースコードとの対応付けに対し有用な概念間の関連の更なる種類分け等の検討が望まれる．

機能に関する記述以外との対応付け

提案手法で対応付けることが可能な文書は，機能に関する記述を含むものに限られる．例えば，非機能要求はソースコード上で横断的関心事として実装されることが多く，メソッド呼び出し関係を利用する提案手法ではこれをうまく対応付けられないと考える．よって，その他の種類の文書との対応付けを行う手法が今後の研究として望まれる．

オントロジの自動的な構築

本稿で行った適用事例では，ドメインオントロジを手動で作成した．今後更なる大規模実験を行うためには大量の概念を持つオントロジが必要となる．既存のオントロジ自動構築の研究[10),11)]を導入することの検討が望まれる．

参 考 文 献

1) 田原貴光，林　晋平，佐伯元司：仕様書とJavaソースコードの構造の類似性に基づく対応付け，情報処理学会研究報告,Vol.2008, No.29, pp.139-146 (2008).

2) Antoniol, G., Canfora, G., Casazza, G., DeLucia, A. and Merlo, E.: Recovering traceability links between code and documentation, *IEEE Transactions on Software Engineering*, Vol.28, No.10, pp.970–983 (2002).

3) Witte, R., Li, Q., Zhang, Y. and Rilling, J.: Text mining and software engineering: An integrated source code and document analysis approach, *IET Software*, Vol.2, No.1, pp.3–16 (2008).

4) Turk, D., France, R. and Rumpe, B.: Limitations of agile software processes, In *Proceedings of the 3rd International Conference on Extreme Programming and Flexible Processes in Software Engineering*, pp.43–46 (2002).

5) Code Conventions for the Java Programming Language. `http://java.sun.com/docs/codeconv/`.

6) IBM Java 命名規則. `http://www-06.ibm.com/jp/developerworks/components/001124/j_tip-namingconv.html`.

7) J-Domain [JDraw]. `http://jdraw.sourceforge.net/index.php?page=6`.

8) Eclipse.org. `http://www.eclipse.org/`.

9) Zhao, W., Zhang, L., Liu, Y., Sun, J. and Yang, F.: SNIAFL: Towards a static noninteractive approach to feature location, In *Proceedings of the 26th International Conference on Software Engineering*, pp. 293–303 (2004).

10) Ratiu, D., Feilkas, M. and Jurjens, J.: Extracting domain ontologies from domain specific APIs, In *Proceedings of the 12th European Conference on Software Maintenance and Reengineering*, pp.203–212 (2008).

11) 長谷川亮，北村元博，海谷治彦，佐伯元司：要求分析のためのドメインオントロジ構築支援，電子情報通信学会技術研究報告， Vol.107, No.176, pp.53–58 (2007).