Vol.2019-SE-201 No.18
2019/3/8

要求・仕様記述文の共通性／可変性分析と派生的導出に関する一手法

中西 恒夫[1,a)]　吉村 賢治[1,b)]　乙武 北斗[1,c)]　田辺 利文[1,d)]　古庄 裕貴[1,e)]

概要：プロダクトライン開発では，要求，仕様，設計，コードなどさまざまな抽象度の既存資産の現状を把握し，製品間の共通性／可変性分析を行うことが必要となる。要求や仕様は自然言語で記述されていることが多く，その共通性／可変性分析には相当量の工数を確保することが強いられる。本稿では，複数製品の要求／仕様記述文を構文解析して抽象構文木を組み立て，それらを共通性／可変性が明示されるかたちで合成し，共通性／可変性を記述したフィーチャモデルの構築を容易にする手法を提示する。あわせて，抽象構文木で表現された要求／仕様記述文に対して，事前に用意された概念辞書と書換え規則に基づいて変形を施し，派生的な要求／仕様の記述文を生成する手法も提示する。

Commonality/Variability Analysis and Derivative Generation of Natural Language Requirements and Specifications

Abstract: It is required in product line development to conduct commonality/variability analysis of existing products and comprehend existing assets of different abstraction levels including requirements, specifications, designs, and codes. A large amount of time is needed for commonality/variability analysis of requirements and specifications described in natural languages. The authors presents a scheme to ease construction of the feature model describing commonality/variability of requirements and specifications in natural languages. The scheme performs syntax and semantic analysis of requirement and specification sentences of multiple products, constructs syntax trees of the products, and unifies them with representing commonality/variability explicitly. Moreover, the author presents a scheme to generate derivative requirements and specifications by applying modification based on the concept dictionary and rewrite rules to original requirements and specifications.

1. はじめに

我が国のソフトウェア開発現場では，多くの共通する機能や品質を備えつつも，仕向地先の要求にあわせて少しずつ異なる機能や品質を備えた製品群のためのソフトウェアを継続的に開発している。開発現場では，最も近い過去製品のソフトウェアを変更する派生開発手法 [1] や，規定されたプロセスに即して共通ソフトウェア資産を再利用し開発対象製品のソフトウェアを開発するプロダクトライン開発手法 [2], [3], [4] を用い，こうした製品群のためのソフトウェアを開発している。プロダクトライン開発手法は，製品ごとの派生開発を繰り返すよりも大きなコスト削減効果が得られる手法であるものの [5], [6], [7], [8]，共通ソフトウェアとその再利用プロセスの構築に要する初期コストが大きく，それゆえに導入を躊躇する開発現場も少なくない。その理由として，現場の開発プロセスや開発組織を変えていく政治的な労力もさることながら，すでにある多くの既有製品群の現状を，膨大な既存ドキュメントやコードを読み込んで理解しなければならない技術的な問題があることが挙げられる。

ソフトウェア開発現場では，UML や形式仕様記述言語といった形式的，あるいは準形式的な記述が用いられるようになりつつあるものの，それらの表現力に問題があったり，あるいはそれらを使って適切に物事を表現するには相当の熟練を要したり，現場の技術者がそうしたことを警戒・

[1] 福岡大学工学部電子情報工学科
Fukuoka University, Fukuoka, Japan 814–0180, Japan
[a)] tun@fukuoka-u.ac.jp
[b)] yosimura@fukuoka-u.ac.jp
[c)] ototake@fukuoka-u.ac.jp
[d)] tanabe@fukuoka-u.ac.jp
[e)] furusho@fukuoka-u.ac.jp

敬遠したりで，要求や仕様の記述には依然，自然言語が用いられる場合がほとんどである。こうした自然言語で書かれた要求や仕様は，各社各様にワードやエクセル上で箇条書きで書かれたり，USDM[9] のように構造化されたかたちで書かれたりである。

プロダクトライン開発手法では，製品間の共通性と可変性を記述，理解し，製品が継続的に進化しても綻びにくい構造のアーキテクチャを構築すべく，フィーチャモデリング [10] が広く実践されている。フィーチャモデリングのためには，複数の製品に関する，こうした自然言語で書かれた既有ドキュメントを読まなければならない。フィーチャモデリングの要員確保も大きな問題であり，フィーチャモデリングを担う人材は，大量の自然言語の文書を読み込む根気と，自然言語で書かれた記述に埋もれている製品間の共通性と可変性を見出し，それらを整理できる言語能力と抽象化力とを備えていなければならない。以上の背景を踏まえたうえで，本稿では，自然言語処理技術，具体的には構文／意味解析を用いて，複数製品の自然言語による要求や仕様の記述に対して，それらの共通部と可変部を抽出し，フィーチャモデリングの一助とする手法について論じる。

プロダクトライン開発では，将来製品の要求や仕様も予想し備えることも，実務面では極めて重要である。将来製品の要求や仕様は，原製品の要求や仕様の一部を抽象化したり具象化したりすることによって，派生的に予想できることも少なくない。本稿では，事物の抽象化／具象化関係を定義する概念辞書と原要求／仕様をパターンマッチングにより改変する書換え規則を用いて，そうした派生的な要求，仕様を生成する手法も示す。

以下，本稿では，第 2 節において提案手法に係る関係する自然言語処理関連の用語を解説したうえで，第 3 節において自然言語で記述された複数製品の要求，仕様に対する共通性／可変性解析を支援する手法を述べ，第 4 節において派生要求，仕様を生成する手法を示す。最後に第 5 節で本稿を総括する。

2. 用語の定義

本節では，本稿で用いる自然言語に係る用語の定義を与える。

2.1 単文と複文

単一の述語を中心に構成された文を**単文**，複数の述語で構成される文を**複文**と呼ぶ。

複文の各々の述語を中心に構成される語のまとまりを**節**と呼ぶ。複文の背骨となる節を**主節**と呼び，その他を**接続節**と呼ぶ。

接続節は**並列節**と**従属節**に分けられる。

並列節は主節と意味的には対等な接続節であり，例文「警報 LED を点灯し，警報ブザーを吹鳴する。」では，「警報ブザーを吹鳴する」が主節，「警報 LED を点灯し」が並列節となる。

従属節は主節に対して意味的な主従関係を有する接続節であり，以下のように分類される。

- **補足節:** 名詞相当表現（「〜すること」,「〜するの」など）と格助詞で構成され，述語を補う役割を有する。例文「初期化エラーが発生したことを通知する。」では，「初期化エラーが発生したこと」が補足節である。
- **副詞節:** 述語や文全体を修飾する役割を有し，代表的な意味的機能として以下のようなものが挙げられる。
  - 時を表す副詞節（「〜するとき」 etc.）
  - 原因・理由を表す副詞節（「〜なので」 etc.）
  - 条件・譲歩を表す副詞節（「〜ならば」「〜しても」 etc.）
  - 付帯状況・様態を表す副詞節（「〜ながら」「〜ように」 etc.）
  - 目的を表す副詞節（「〜ため」 etc.）
  - 程度を表す副詞節（「〜ぐらい」 etc.）
- **名詞修飾節:** 名詞を修飾する役割を有する。
  - 補足語修飾節： 被修飾名詞が当該修飾節の述語の補足語となっている。例文「加速度センサが取得した加速度」の「（補足語＝加速度を）加速度センサが取得した」
  - 相対名詞修飾節： 被修飾名詞が当該修飾節の述語の特定の補足語と相対的な関係になっている。例文「シャットダウンする直前に」の「シャットダウンする（補足語＝時の）直前に」
  - 内容節： 被修飾名詞の具体的内容を表す。例文「過大な電圧がかかっている状態」の「過大な電圧がかかっている」

詳細は文献 [11] 等を参照されたい。

2.2 深層格

動詞とその動詞に係る語の間に見られる意味的関係の規則を格文法と呼び，また動詞と動詞に係る語の意味的関係を深層格と呼ぶ。格文法は Fillmore によって提唱され [12]，多くの言語学者によって拡張がなされ，さまざまな深層格が定義されてきた。以下は Fillmore によって定義された深層格である。

- **主格（agentive）：** 動詞の動作を起こす者。意思を持つ生物である。
- **経験者格（experiencer）：** 心理動詞（「喜ぶ」「愛する」など）の心理状態になる者。人間の心理を扱うことが稀なソフトウェアシステムの要求や仕様の記述にはあまり出現しないものと思われる。
- **道具格（instrumental）：** 動詞の動作の原因となる物。
- **対象格（objective）：** 動詞の動作の対象となる物。

- **源泉格（source）：** 動詞の動作の空間的，時間的，あるいは様態的な変化の起点。
- **目標格（goal）：** 動詞の動作の空間的，時間的，あるいは様態的な変化の終点。
- **場所格（locative）：** 動詞の動作が起こる場所。
- **時間格（time）：** 動詞の動作が起こるとき。
- **経路格（path）：** 動詞の動作の一連の空間的な経由点。

動詞ごとに省略が許されない深層格と許される深層格があり，前者を**必須格**，後者を**任意格**と呼ぶ。たとえば，「制御する」という動詞の場合，対象格は必須格であるが，道具格は任意格である。

3. 要求・仕様記述文に対する共通性／可変性分析の自動化

本節では，自然言語による要求や仕様の記述に対する共通性／可変性分析の自動化について述べる。

自然言語記述に対する共通性／可変性分析は以下の手順で行う。

(1) 自然言語記述中の用語や表現を正規化する。
(2) n 個（$n \geq 1$）の自然言語記述に対して形態素／構文解析を実施し抽象構文木を生成する。
(3) n 個の抽象構文木の共通部分を一体化して抽象構文木を合成する。
(4) 合成された抽象構文木の可変部内の共通部を簡約化する。
(5) 並列構造を含む可変部を簡約化する。

以下の 6 つの要求記述文を題材に上述の手順を解説する。

(a) トナーが切れたとき，システムは印刷を中止することをユーザに知らせる。
(b) トナーが切れたとき，システムは印刷を中止することをシステム管理者とユーザに知らせる。
(c) トナーが切れそうなとき，システムは印刷ができないことをあらかじめユーザに通知する。
(d) マゼンダのトナーが切れたとき，システムは印刷を中止することをユーザに知らせる。
(e) シアンのトナーが切れたとき，システムは印刷を中止することをユーザに知らせる。
(f) イエローのトナーが切れたとき，システムは印刷を中止することをユーザに知らせる。

これら要求記述文は 4 種類の製品 A，B，C，D に関するものとし，どの製品がどの要求記述文の要求を実現しているかは表 1 に示すとおりとする。

用語と表現の正規化： 自然言語記述中で使用されている同義語，同義表現をその同義クラスの代表語，代表表現に置換し，用語や表現の揺らぎをなくす。たとえば，上述の例文 (c) の「ユーザに通知する」は「ユーザに知らせる」に置換される。

抽象構文木の生成： 形態素解析器と構文解析器を用いてひとつ以上の要求記述文のそれぞれの抽象構文木を生成する。上述の 6 つの例文の場合，図 1 に示す 6 つの抽象構文木が生成される。

表 1 製品要求表

	要求					
	(a)	(b)	(c)	(d)	(e)	(f)
A	○					
B		○				
C	○		○			
D				○	○	○

抽象構文木の各節点には，当該要求を実現している製品の集合を付記する。

抽象構文木の合成： 生成された複数の要求記述文の抽象構文木を合成する。抽象構文木の根から語の意味と深層格の一致したところまでを共通化し，そこから葉に至る部分木については，分岐点を設けて相違部分の部分木をぶら下げる。

分岐点には選択性に係る属性も付加する。複数の要求記述文中に出現したりしなかったりする部分木についてはオプショナル属性，択一的に現れる部分木についてはオルタナティブ属性を付加する。

また，合成の際，抽象構文木の各節点に付記されていた製品の集合については和集合をとる。

図 1 の 6 つの抽象構文木を合成した結果は図 2 のようになる。図中の白丸はオプショナル属性となっている部分木の根を，円弧はオルタナティブ属性となっている部分木の根を意味している。

なお，ここで合成された抽象構文木にはフィーチャ図のようにオプショナル属性，オルタナティブ属性がつけられているが，これらは決して製品間の可変性を表現しているのではなく，実行時の振舞いの可変性を表現していることに注意されたい。

抽象構文木の可変部内の共通部分簡約化： 合成された抽象構文木のうち，オルタナティブ属性の分岐点にぶら下がる複数の部分木について，抽象構文木を合成したときと同様に，共通化の処理と相違部の切り分け処理を行って共通部分を簡約化する。新たに設けられる分岐点に選択性に係る属性を設けるのも同様である。このとき相の異なる動詞については，同じ動詞として扱うものの，相の違いを表現するオルタナティブ属性の部分木を動詞の節点の下に配置する。

図 1 の合成された抽象構文木の共通部分を，上述の通り簡約化すると，図 3 の木構造が得られる。「切れた」と「切れそうな」はいずれも同じ動詞「切れる」であり，異なるのはその相である。「切れた」は完了を意味する相であり，「切れそうな」は確度の高い未来を意味する相である。

共通部分簡約化の処理は，すべてのオルタナティブ属性の分岐点（共通部分簡約化の過程で新たに生じるオルタナ

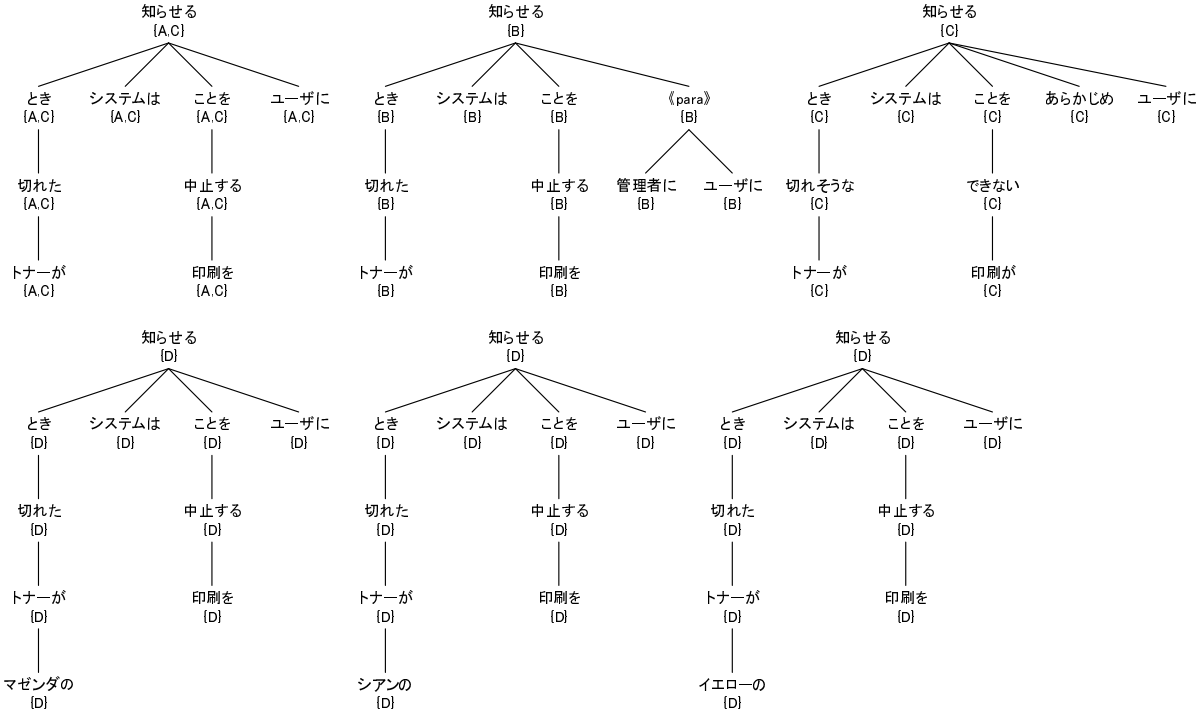

図 1 抽象構文木の生成

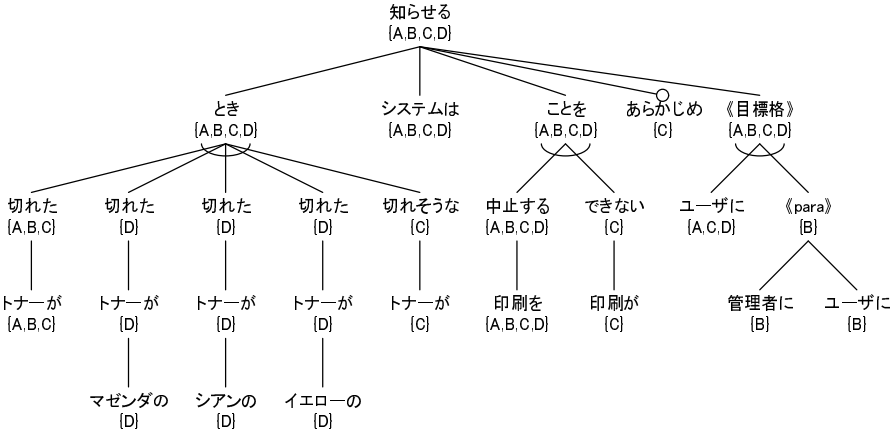

図 2 抽象構文木の合成

ティブ属性の分岐点も含む）を処理するまで繰り返す。

並列構造を含む可変部の簡約化: 最終的に得られた木構造の中で，並列構造を含む可変部，すなわちオルタナティブ属性の分岐点以下を簡約化する。たとえば，図 3 の木構造の場合，文節「ユーザに」と，文節「管理者に」と「ユーザに」からなる並列構造がオルタナティブ属性の分岐点で選択されるようになっている。この構造は，図 4 に示すように，「ユーザに」は常に選択され，「管理者に」はオプショナルに選択される構造に簡約化できる。

すべての選択肢に共通して出てくる要素を必須属性に，そうでないものをオプショナル属性とすることで簡約化を実現する。オルタナティブ属性の分岐点と並列構造が複雑にネストされている場合，このような単純な簡約化では，もともとの要求記述文から導出される選択構造と完全に同一とは，一般的にはならない。しかし，本研究のねらいは，自然言語による要求記述の共通性／可変性の現状を厳密に分析することではなく，考えられる可変性を示して，フィーチャモデリング，さらには将来要求され得る派生要求の導出の助けとすることにある。そのため，共通性／相違性を厳密に再現することは求めない。

フィーチャモデルの作成: 最終的に得られた木構造をもとにフィーチャモデルを作成する。この作業は，要求にふさわしい簡潔な名前をつけるといった抽象化が必要な作業であり，人手で実施しなければならない作業である。

抽象構文木からのフィーチャモデルの作成例を図 5 に示す。抽象構文木の「トナーが」を頂点とする部分木からフィーチャモデルの「トナー」以下の部分木が，「とき」-「切れる」-「《aspect》」を先祖とする部分木からフィーチャ

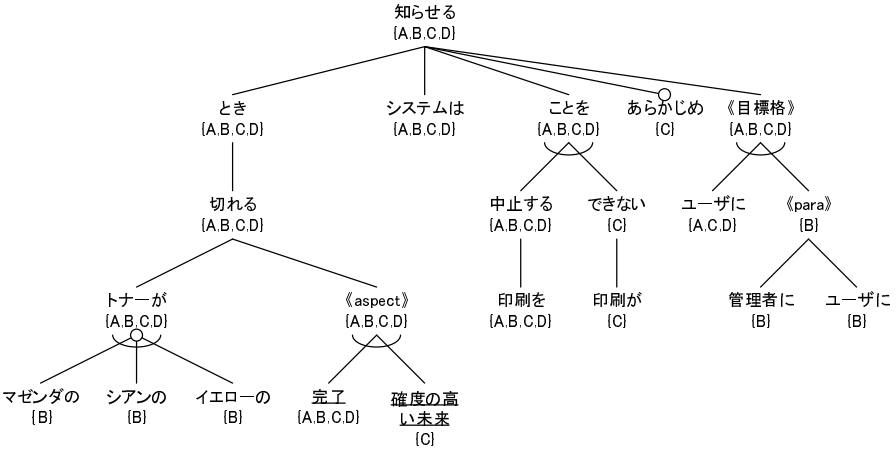

図 3 共通部分簡約化

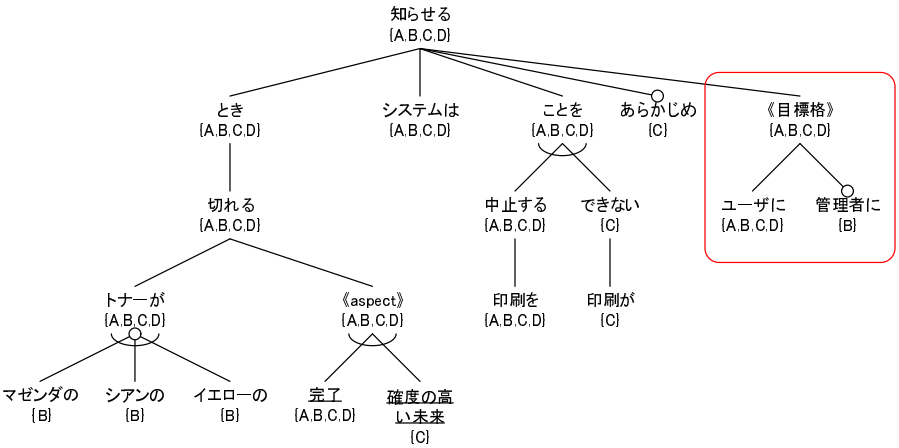

図 4 並列構造を含む分岐点の簡約化

モデルの「印刷エラー条件」以下の部分木が，「《目標格》」を頂点とする部分木からフィーチャモデルの「エラー報告先」以下の部分木が，「ことを」を先祖とする部分木からフィーチャモデルの「通知内容」以下の部分木が導き出されている。「あらかじめ」は，フィーチャ「印刷不可」の概念に含まれるものとし，フィーチャモデル上には「あらかじめ」と直接対応するフィーチャは設けられていない。

フィーチャモデルのオプショナル属性，オルタナティブ属性は，実行時の振舞いの可変性ではなく，製品間の可変性である。これらの属性は，抽象構文木の各節点に付記された製品の集合から決まる。節点に付記された製品の集合が，親節点に付記された製品の集合と全く同じあれば，その節点に対応するフィーチャは必須属性となる。兄弟の節点に付記された製品の集合が互いに素であり，かつその和集合が親節点に付記された製品の集合と同じであれば，それら兄弟節点の親節点に対応するフィーチャはオルタナティブ属性となる。そうでなく，かつ節点に付記された製品の集合が親節点に付記された製品の部分集合となっている場合は，その節点に対応するフィーチャはオプショナル属性となる。

4. 派生要求ならびに仕様導出の支援

本節では，前節で述べた自然言語で記述された要求，あるいは仕様の共通性／可変性分析の結果を踏まえ，原要求，原仕様から派生的に新たな要求や仕様の導出を図る方法論を提案する。提案する派生要求／仕様導出法は概念辞書と書換え規則を用いるものである。

4.1 概念辞書

提案手法は概念辞書を必要とする。

概念辞書は事物の抽象／具象関係を記述したツリー構造の辞書である。図 6 は概念辞書の一部の例であり，ユーザへのイベント通知に関する語の関係を記述したものである。方形の節点は具体的な自然言語による記述を，円形の節点は抽象化／具象化の観点（概念軸）を意味する。

用語の使われ方，抽象／具象関係の捉え方がドメインによって異なることは少なくなく，事物の抽象／具象関係を一般的，画一的に定義するのは非現実的である。そこで概念辞書は，ドメイン非依存のもの，ドメイン依存かつプロダクトライン非依存のもの，ドメイン依存かつプロダクト

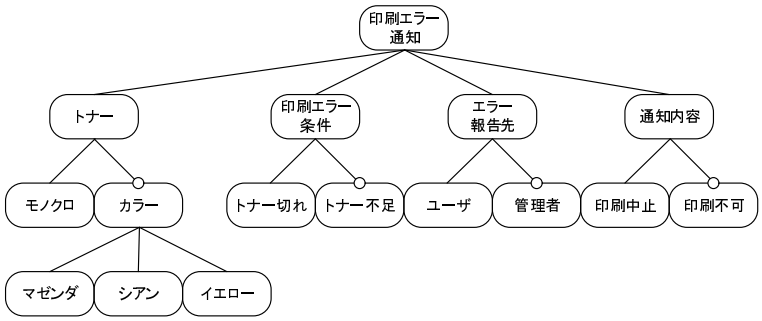

図 5 人手で構築されるフィーチャモデル

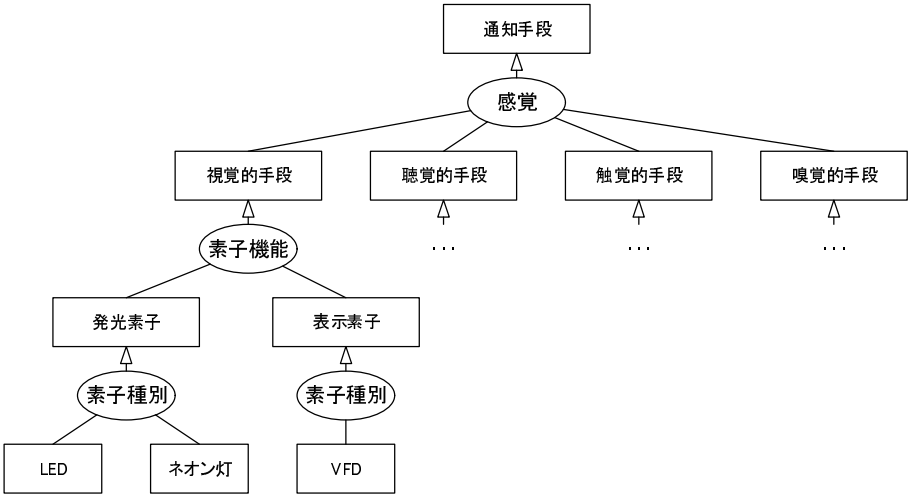

図 6 概念辞書

ライン依存のものの 3 種類に分け，それぞれを管理するものとする。ドメイン依存かつプロダクトライン依存の辞書はプロダクトラインのコア資産として保守していく。

4.2 書換え規則

書換え規則は，原要求，原仕様をどのように書き替えて派生要求，派生仕様を導出するかを定義するものであり，動詞ごとに定義され，パターン定義と導出定義とで成る。表 2 に書換え規則の例を示す。

パターン定義には，原要求，あるいは原仕様の記述中のひとつの動詞とその相，当該動詞が関係する深層格，当該動詞を修飾する副詞のパターンを記述する。原要求，原仕様の記述中の動詞，相，各深層格，ならびに副詞がそれぞれ，パターン定義中の動詞，相，各深層格の語かそれを概念辞書に従って具象化したもの，ならびに副詞に一致した場合に限り，当該原要求，原仕様を導出定義に従って書き替えて，派生要求，派生仕様を導出するものとする。但し，パターン定義中の「*」は Don't Care を意味する。

原要求，原仕様を書き換えるときには，動詞，相，各深層格，ならびに副詞をそれぞれ，導出定義中の動詞，動詞，相，各深層格，ならびに副詞に書き替える。但し，– は原要求，原仕様の語そのもの（つまり書き替えない），x は消去すること，$d(w)$ は語 w を概念辞書により具象化した語を意味する。

たとえば，「システムはユーザに給紙エラーを通知する」という原要求記述は，「給紙エラー」が「エラー」を具象化したものとして概念辞書に定義されているなら，表 2 の書換え規則のパターン定義にマッチする。このとき，導出定義に基づいて，「システムは視覚的手段，発光素子，LED，ネオン灯，表示素子，VFD で給紙エラーをユーザに通知する。」という派生要求が生成されていくこととなる。

以上の書換えを前節で述べた手法により構築された抽象構文木に対して適用する。

5. 考察

本稿執筆時点で提案手法の実装と評価には至っていないため，本節では，提案手法の実問題への適用に係るいくつかの考察を述べる。

要求，仕様の書かれ方の問題: 第 3 節で提案した共通性／可変性分析の自動化手法は，要求，仕様を文単位で比較のうえ，文中の互いの共通部分と可変部分の分離を図る。そのため，関連の強い要求，仕様項目が同じひとつの文に記述されていなければ，効果的な共通性／可変性分析を行うことはできない。たとえば，第 3 節の要求記述文 (a) が「トナーが切れる。このときシステムは印刷を中止することをユーザに知らせる。」といったように文を分けられた

表 2　書換え規則

深層格	主格	道具格	対象格	源泉格	目標格	場所格	時間格	副詞	動詞	相
パターン定義	システム	*	エラー	*	ユーザ	*	*	*	通知する	*
導出定義	–	d(視覚的手段)	–	–	–	–	–	–	–	–

なら，他の要求記述文との共通性／可変性分析を満足に行うことができなくなる。もっとも開発現場では，要求，仕様は箇条書きで，統一のとれた形式で書かれることが多いため，このような問題が生じることはあまり多くはないかと思われる。

文脈に隠された意味関係の問題: 提案手法は，要求，仕様の背骨となる動詞に関係する深層格や副詞がすべて文中に記載されることが前提となっているが，現実の自然言語記述では文脈に意味関係が隠されることが少なくない。箇条書きされた要求，仕様の場合，要求，仕様項目が 2 段，多くても 3 段に階層化されて記述されることが少なくない。このとき，子の要求，仕様は，親の要求，仕様を具象化したり，構造観点で分解したり，時系列で分解したり，補足したりしていることがほとんどであるが，そうした関係が明記されていることはまずない。こうした親と子の要求，仕様の記述は，文としては別の文になっているため提案手法を適用することはできないし，適用できるように何かしらの前処理をするにしても，親と子の意味的関係を人間の介在なく解析することは困難である。

並列節にもまた，こうした暗黙的な意味関係が見られる。「電磁弁を開栓し，内圧を下げる。」と「エラーを表示し，警報音を鳴らす。」とあった場合，どちらも表層的には同じ文であるが，深層的には前者については時間的な前後関係やさらには行為と想定されるその結果という関係が，後者については時間的な前後関係があるか，あるいは単なる並記という関係がある。こうした並列節と主文の関係も機械的な解析は容易ではない。

用語と表現の正規化の問題: 提案手法を適用する際には，第 3 節で述べたように用語や表現の正規化を行う必要がある。基本的にこうした正規化は妥当な辞書やルール記述があれば，相当のことができるものと楽観しているが，辞書やルール記述の整備にはそれなりのコストがかかる。幸いなことに，現実の要求，仕様記述に使われる語彙は限定的であり，文学的な修辞がなされることもほとんどない。著者らが，過去に企業の実製品の開発文書を解析した際には，使われている動詞は限定的であり，副詞や形容詞はほとんど出現していなかった。名詞についてはそれなりの種類のものが出現するが，（実質上は動詞である）サ変名詞を除けば，名詞は事物を表す語であり，活用もしないことから比較的扱いは容易である。動詞が少ないことは，格フレームの辞書を作るうえで極めて有利である。

複合名詞の問題: 要求や仕様の記述では，「モータ回転制御」や「摺動摩擦抵抗」といった複合名詞が好んで使われる傾向がある。複合名詞は名詞ではあるが，それを構成する単独の名詞まで分解しなければ共通性／可変性の分析が正確にできないことが少なくない。さらには複合名詞を構成する名詞間で修飾／被修飾の関係があり，辞書に事前に知識として登録しておくか，何かしらのルール定義をしておかなければ，修飾／被修飾関係を解析することができない。複合名詞については頻繁に，しかも定義を与えなくても人間の読み手は理解できることから場当たり的に使用されることが多く，プロダクトラインやドメインごとに複合名詞の辞書を作成する必要があるものと考えられる。

6. まとめ

本稿では，複数の製品から集めた，自然言語で記述された要求・仕様記述文に対して構文／意味解析を適用し，得られた抽象構文木を共通性／可変性を明示したかたちで合成し，共通性／可変性分析の成果物たるフィーチャモデルの構築を支援する手法を提案した。システムやソフトウェアのあるべき振舞いを記述する要求，仕様の記述では動詞が中心的な役割を果たす。構築する抽象構文木の基本構造は動詞を根とし，その深層格や副詞を葉とするものであり，複文では節ごとにこの基本構造が再帰的に出現する構造となる。共通性／可変性分析ではこの抽象構文木の共通部分と可変部分の分離を行う。最終的には，フィーチャモデルに近い構造の木構造が機械的に得られるので，あとは分析者がそれを参考にフィーチャモデルを構築する。

また，本稿では自然言語で記述された要求，仕様から，概念辞書と書換えルールをもとに，派生要求，派生仕様を導出する手法も述べた。その手法は，動詞，相，深層格，副詞の単位でパターンマッチングを行い，パターンに適合した場合は事前に規定された変更を原要求，原仕様に対して，これもやはり動詞，相，深層格，副詞単位で適用するものである。

あわせて本稿では，提案手法の実問題への提要に係るいくつかの問題を述べ，箇条書きの親子構造や並列節，複合名詞などにおいては語の意味関係が暗黙的に表現され，機械的な解析が困難になることを指摘した。

提案手法の実装と実問題への適用と評価については続報を待たれたい。

謝辞 本研究は科研費（課題番号: 17K00116）の助成を受けている。

参考文献

[1] 清水 吉男, 「『派生開発』を成功させるプロセス改善の技術と極意」, 技術評論社, 2007 年 11 月.
[2] K. C. Kang, S. Kim, J. Lee, K. Kim, E. Shin, and M. Huh, “FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architecture,” *Annals of Software Engineering*, Vol. 5, No. 1, pp. 143–168, Jan. 1998.
[3] P. Clements and L. Northrop, *Software Product Lines: Practice and Patterns*, Addison-Wesley, 2001.
[4] K. Pohl, G. Böckle, and F. v. d. Linden, *Software Product Line Engineering: Foundation, Principles, and Techniques*, Springer, 2005.
[5] T. Iwasaki, M. Uchiba, J. Ohtsuka, K. Hachiya, T. Nakanishiy, K. Hisazumi, and A. Fukuda, “An Experience Report of Introducing Product Line Engineering across the Board,” *Proc. 14th Int. Conf. on Software Productline Conf. (SPLC 2010)*, pp. 255–258, Sep. 2010.
[6] J. Otsuka, K. Kawarabata, T. Iwasaki, M. Uchiba, T. Nakanishi, K. Hisazumi, and A. Fukuda, “Small Inexpensive Core Asset Construction for Large Gainful Product Line Development: Developing a Communication System Firmware Product Line,” *Proc. 15th Int. Software Product Line Conf. (SPLC2011)*, 5 pages, Aug. 2011.
[7] 西浦 洋一, 浅野 雅樹, 中西 恒夫, 「自動車ボディ系製品のプロダクトライン開発への移行に関する事例報告」, 組込みシステムシンポジウム 2018 論文集, pp. 59–66, 2018 年 8 月.
[8] Yoichi Nishiura, Masaki Asano, and Tsuneo Nakanishi, “Migration to Software Product Line Development of Automotive Body Parts by Architectural Refinement with Feature Analysis,” *Proc. 25th Asia-Pacific Software Engineering Conf.*, Dec. 2018.
[9] 清水 吉男, 「要求を仕様化する技術・表現する技術: 仕様が書けていますか？」, 改訂第 2 版, 技術評論社, 2010 年 6 月.
[10] K.-C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, “Feature-Oriented Domain Analysis (FODA): Feasibility Study,” CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, Nov. 1990.
[11] 益岡 隆志, 田窪 行則, 「基礎日本語文法」, くろしお出版, 1989 年 9 月.
[12] J. Fillmore, “The Case for Case,” *Proc. Texas Syposium on Language Universals*, Apr. 1967.