社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

要求分析におけるオントロジーの活用法

海谷　治彦†　　佐伯　元司††

†† 東京工業大学 大学院情報理工学研究科　〒 152-8552 東京都目黒区大岡山 2-12-1
† 信州大学 工学部　〒 380-8553 長野市 若里 4-17-1
E-mail: †kaiya@cs.shinshu-u.ac.jp, ††saeki@se.cs.titech.ac.jp

あらまし　特定の業務分野 (ドメイン) のオントロジーと要求仕様を対応付けに基づく，意味まで考慮した要求分析法を提案する．オントロジーはクラス図で表現されたシソーラスと論理式の集合で表現された推論規則で表現する．意味処理に適した概念や関係をオントロジーに記述することにより，比較的単純な対応付けと推論のみで，要求仕様の自然言語処理では困難な要求分析が可能であることを示す．具体的には，(1) 欠落事項や矛盾の検出，(2) 意味まで考慮した要求仕様の品質評価，(3) 変更履歴を用いた将来の変更予測の処理方法を例題を通して説明する．
キーワード　軽量化意味処理, シソーラス, 完全性, 無矛盾性, 妥当性, 非曖昧性，検出，メトリクス

Ontology Based Requirements Analysis

Haruhiko KAIYA† and Motoshi SAEKI††

†† Graduate school of Information Science and Engineering, Tokyo Institute of Technology　2-12-1, Ookayama, Meguro-ku, Tokyo, 152-8552, Japan
† Faculty of Engineering, Shinshu University　4-17-1, Wakasato, Nagano city, 380-8553, Japan
E-mail: †kaiya@cs.shinshu-u.ac.jp, ††saeki@se.cs.titech.ac.jp

Abstract　We propose a requirements analysis method by using a mapping between a requirements specification and an ontology that represents a specific application domain. Our ontology system consists of a thesaurus and inference rules that have concepts and relationships suitable for semantic processing. The ontology system enables requirements engineers to analyze a requirements specification with respect to the semantics of the application domain even though the system does not directly support natural language processing techniques. We demonstrate following three kinds of semantic processing through a case study, (1) detecting incompleteness and inconsistency about a requirements specification, (2) measuring the quality of a specification with respect to its meaning and (3) predicting requirements specification changes by mining change history.
Key words　Lightweight semantic processing, Thesaurus, Completeness, Consistency, Correctness, Unambiguity, Detection, Metrics

1　ま え が き

ゴール指向やシナリオ分析などの要求獲得の適した方法，ユースケースモデリングなど後段に接続するための方法は既に提案されている．IEEE830 などのように要求仕様書の構成の標準もあり実用に附しつつある．しかしながら，記述は非形式的な自然言語で行われることが多いため，現在の自然言語処理技術をもってしても，無矛盾性のチェックなどの意味処理は難しい．コンピュータ処理できるほど構文を限定した自然言語で仕様を記述する試みもあったが，これらの言語で高品質の要求仕様を書かせることは困難である．また，公理系と推論体系を用いて意味処理を行える形式手法は，safety critical system といった一部の分野では活用されているが，他の分野ではまだ実務者スキルの問題等があり，普及にはほど遠い．

オントロジー技術は知識処理分野で盛んに用いられるようになってきた．オントロジーは現実世界に存在する概念の分類とその間の階層構造，関係を記述したもので，ドメイン知識として活用することができる．また，構造や関係に基づいて推論が行えるため，種々の意味処理を行うことができる．

本論文では，要求分析においてオントロジー技術を適用し，これまで複雑な自然言語処理がなければ行えなかったような，要求の軽量化意味処理 (Light Weight Semantic Processing)，例えば要求獲得時の要求の欠落事項や意味的矛盾の検出を行う技術を提案する．ここでのオントロジーは対象領域固有のドメイン・オントロジーで，すでにオントロジーは構築されているものとする．どのようにオントロジーを構築するかは 6 章で述べる．要求分析中に生成される要求項目を，あらかじめ用意したオントロジーの構成要素に対応付けることにより，要求記述の意味

付けを行う．意味処理はオントロジーの推論規則を用いて行う．

本論文の構成は以下のとおりである．次章ではオントロジーを用いて要求の意味付けをどのように行うかを述べる．3, 4, 5 章では，どのような意味処理が可能かを PC 上で動作する音楽ファイル (MP3 (MPEG Audio Layer-3) 等) のプレイヤーの要求分析を例に取り説明する．6 章で事例や関連研究をもとに本手法の特長を議論する．

2 オントロジーを用いた意味づけ

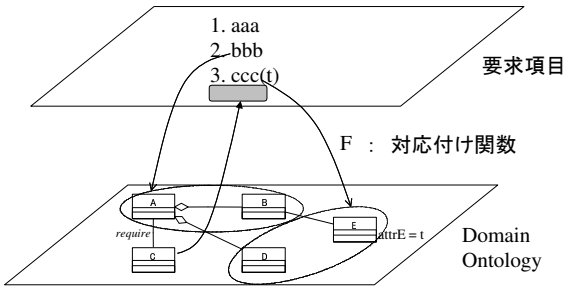
2.1 要求分析のためのオントロジー

文献 [1] では，オントロジーの定義は，“formal explicit specification of shared conceptualization”となっている．また，“classifications of the existing concepts” と定義することもできる．本論文では，記述に対して意味を提供する意味素 (atomic constituents of meaning) としてオントロジーを使用するため，「万人が共通かつ共有した解釈を持つことが可能な原子的概念」をオントロジーは含む必要がある．概念のラベルを誰でも共通に理解できる語句でふると，オントロジーの概念構造はシソーラス (thesaurus) とみなすことができる．語彙分解 (lexical decomposition) の手法 [2] により，ある物事の意味を自然言語で次々と詳細に定義していったとき, 誰でもが共通の意味理解をできるようになるまで分解していったときに，最下位の説明文の中に出現する語句がオントロジーの概念を表現していると考えられる．我々はオントロジーを意味処理に使用するため，単なるシソーラスではなく，概念間の関係に基づいた推論機構を持っていることが必要である．つまり，シソーラス + 推論機構を我々のオントロジーとみなすことにする．

シソーラス部分は節が概念を示し枝が概念間の関係を示すグラフであり，推論機構は論理式の集合とする．概念間の関係としては，欠落事項や矛盾事項を推論するため，ある概念 A が要求に含まれていたとき，概念 B も含まれていなければいけないという関係，A と B が共に含まれていれば矛盾する (A と B は相互に矛盾) という関係が含まれていなければならない．推論はこのような関係の上で行われる．ソフトウェアに深く関連した概念は，機能やオブジェクト (データを含む) などがあり，要求はこれらに対応付けることができる．例えば，音楽プレーヤーで「再生する (play)」や「巻き戻し (rewind)」は機能に対応付けられる概念であり，「音楽 (music)」，「曲名 (title)」はオブジェクトに対応付けるべき概念である．機能概念に対する推論と，オブジェクト概念に対する推論は異なっており，そのため概念の種別を明確にする必要がある．そこで我々のオントロジーでは，概念のタイプ付けを導入する．

意味付けは，概念への対応付け以外にも考慮すべき事項がある．次の要求項目を考えてみよう．「選択できる曲数の上限は 10 である」この要求項目は，オントロジー概念「選択する」，「曲」，「曲数上限」などに対応付けられる．では「10」はどこに対応付づくのであろうか．音楽プレイヤーによっては，別の値，例えば「100」の場合もあるだろう．この値は，概念「曲数上限」の属性値とすべきであろう．意味付けを行う際に，属性値に 10 を割り当てる操作が必要になる．上限値 100 の音楽プレイヤーの要求仕様の場合は，意味付けする際に，属性値が 100 となる．従って属性値に具体的な値を代入する代入関数への対応付けも必要である．

以上をまとめると，我々のオントロジーは以下のように定義される．



$$F : ReqItems \rightarrow (2^{Concepts \cup Associations} \times (Attribute \rightarrow Value))$$

F(ccc) = ({D, E}, As)　　As(attrE)=t

図 1　要求項目の意味付け

$\mathrm{Ontology_System} = (\mathrm{Con}, \mathrm{Assoc}, \mathrm{Attr}, \mathrm{Rules})$

ただし，

Con: 概念の集合．$\mathrm{Con} = \cup_{t \in \mathrm{Type}} \mathrm{Con_t}$

$\mathrm{Con_t}$は Type t の Concept の集合

Attr: Concept が持っている属性の集合

Assoc: 関係の集合．$\mathrm{Assoc} = \cup_{u \in \mathrm{Association}} \mathrm{Assoc_u}$

$\mathrm{Assoc_u} = 2^{\mathrm{Con} \times \mathrm{Con}}$: 関係 u が成り立つ概念のペアの集合

実際にどのような関係を推論のために用意しなければならないかは 3 章で述べる．

代入関数の集合 $Assign$ は以下のように定義される．

$\mathrm{Assign} = \{\mathrm{as} : \mathrm{Attr} \rightarrow \mathrm{Value}\}$. ただし，Value は属性値の集合.

要求との意味付けは，図 1 のように ,[3] の手法と同様に，記述からオントロジーの構成要素の集合への写像で定義される．意味付け関数 F は，

$\mathrm{F} : \mathrm{ReqItem} \rightarrow (2^{\mathrm{Con} \cup \mathrm{Assoc}} \times \mathrm{Assign})$

となる．

この写像の特徴と写像された概念とそれらの間の関係，それらの上の推論によって，記述内容の軽量化意味処理を行う．図中では，要求項目は写像 F によってオントロジーの構成要素に対応付けられる．例えば，$\mathrm{F(bbb)} = (\{\mathrm{A}, \mathrm{B}, \mathrm{aggregate(A, B)}\}, \phi)$, $\mathrm{F(ccc)} = (\{\mathrm{D}, \mathrm{E}\}, \mathrm{as})$ ただし $\mathrm{as(attr)} = \mathrm{t}$ となる．

推論は一階述語論理の推論体系で行い，関係は述語で表現する．例えば，図のオントロジーは，

Con = {A, B, C, D, E}

Rel = {require(A,C), aggregate(A,B), generalize(A,D), associate(B,E)}

となる．

Rules は推論規則の集合で，述語論理式の集合である．例えば, ある概念 x が別の概念 y のスーパークラスになっているとき，x に対して成り立つ性質 P は，y に対しても成り立つという推論は，以下のように述語論理式で記述できる．

$\mathrm{generalize(x, y)} \wedge \mathrm{P(x)} \rightarrow \mathrm{P(y)}$

図の例では，x=A, y=D, P(x)=require(x,C) とすることにより，$\mathrm{generalize(A, D)} \wedge \mathrm{require(A, C)} \rightarrow \mathrm{require(D, C)}$ となり，require(D,C) も成り立つ．これにより，概念 D の意味をもつ要求項目 bbb が要求に含まれているとき，概念 C の意味をもつ項目も要求に中に含まれていなければならない．もし含まれていなければ，これが欠落事項であり，追加するように示唆できる．

2.2 要求分析オントロジーへの要件

オントロジーのシソーラス部分には，概念間の関係としては，generalization (is_a), aggregation (has_a)，同義語 (synonym)，反意語 (antonym) といった通常の関係に加えて，処理内容に固有の関係を考慮する必要がある．本論文での意味処理は，(1) 欠落事項，矛盾の検出，(2) 意味まで考慮したメトリックス，(3) 変更履歴からの変更予測である．これらの処理のためにどのようなオントロジーが必要かの詳細は 3 章で述べる．

2.3 オントロジーのメタモデル

図 2 に本論文でのオントロジーのメタモデルを示す．このメタモデルはクラス図で書かれており，Concept クラス，Relation クラスが各々オントロジーの概念, 概念間の関係を表す．Relation クラスのサブクラスである require や contradict が欠落事項や矛盾の検出のために使用する関係である．関係は全て単方向の有向関係である．それぞれの関係固有の特性は推論規則として記述する．他の関係の使用法や特性については 3 章で述べる．

メタモデルのインスタンスが個々の問題領域ごとのドメインオントロジーとなる．前章の議論で，概念の型，function と object に言及した．これらは機能要求に関するもので，非機能要求などの品質に関する要求を扱うために quality や constraint を導入した．さらに実現環境を扱うために，actor と platform 概念からなる environment を導入している．

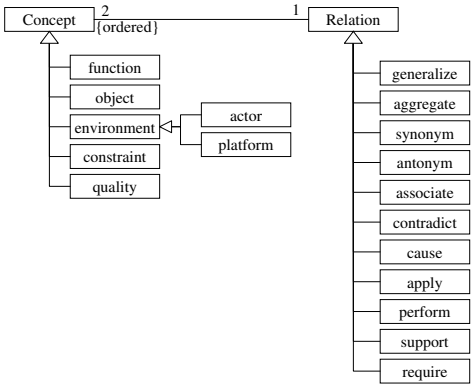


図 2　オントロジーのメタモデル

2.4 オントロジーの例

(1) 曲の再生や一時停止，前後曲の頭だし
(2) 曲の早送り, 巻き戻し
(3) ボリュームの調整と消音
(4) プレイリストをリピート再生
(5) プレイリストをランダム再生

図 3　音楽プレーヤーへの要求仕様 A

図 3 に示すような音楽プレイヤーへの要求仕様 (要求仕様 A) があったとする．この要求仕様に含まれる要求項目の意味をオントロジーを用いて表現したものが図 4 である．

図中の破線の囲みが対応する要求項目を表している．例えば要求項目 (2) は {rewind, forward, music, music op} で意味付けされている．この図は要求項目の意味付けを示すためのものではあるが，同時に要求項目の意味を表せるように構成し作成してある．このことは要求の自然言語記述からオントロジーを構築

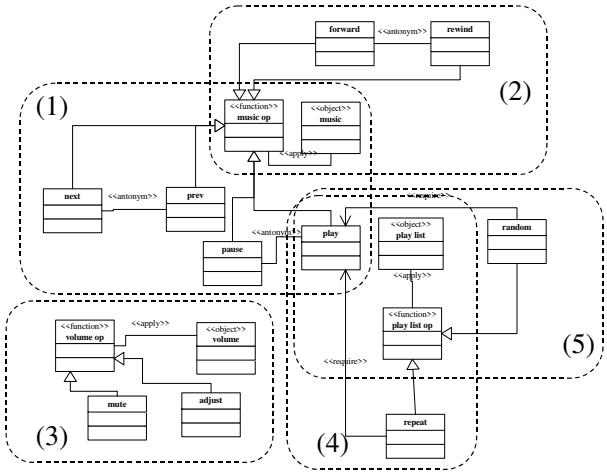


図 4　要求項目から生成した音楽プレーヤーのオントロジー

できることを表している．つまり，同じ問題領域の大量の文書からマイニング技法を用いて当該領域のオントロジーを構築できる可能性があることを示している．この点については 6 章で議論する．

要求項目と図 4 を見ると，いくつかの曖昧な点や不足項目が点がある．例えば，要求項目 (4) と (5) にある「プレイリスト (play list)」は何を指して曖昧である．音楽プレイヤーについての知識を持っている者であるなら，「プレイリスト」が「音楽のリスト」であることは容易に理解できるが，図 4 のオントロジーには，その情報が記述されていない．また，このプレイリストの作成，編集，削除等の操作が必要であることが想像できるであろう．問題領域の知識に基づくより完全なオントロジーがあればこのような曖昧な点も解消できたり，不足している項目も補えたりすると思われる．

図 5 に，この論文で使用する音楽プレイヤーのより完全なオントロジーを示す．続く章では，このオントロジーを用いて意味処理方法を説明する．

以降では推論規則を述語論理を使って表現するため，以下のような表記法を用いる．

- x の type が P である: P(x). 例えば function(rewind), object(music), obejct(codec) 等．
- x と y の間に関係 assoc がある: assoc(x,y). 例えば aggregate(player op, usability), generalize(player op, play) 等．
- 概念や関係 x が要求仕様 S に含まれている: InSpec(x, S). 要求仕様のオントロジーへの対応付け関数を F とすると，$\mathrm{InSpec(x,S)} = \exists \mathrm{r} \in \mathrm{S} \cdot (\mathrm{x} \in \mathrm{F(r)})$ と定義できる．ここで r は要求仕様 S に含まれる 1 つの要求項目もしくは要求文を表す．例えば，図 3 と 4 については，InSpec(rewind, 要求仕様 A), InSpec(volume, 要求仕様 A) が真である．

3 矛盾と欠落事項の検出

この章ではオントロジーを用いた欠落事項と矛盾の検出を例示する．2. 1 節でも述べたように，これらに重要な役割を果たすのは require と contradict の関係であるが，それ以外にも検出のための推論規則がある．

図 3 の要求仕様 A から出発して，要求を追加，精密化することにより，完全な音楽プレイヤーの要求仕様を得るプロセスを考える．まず，要求仕様 A には「プレイリスト」の管理に関す

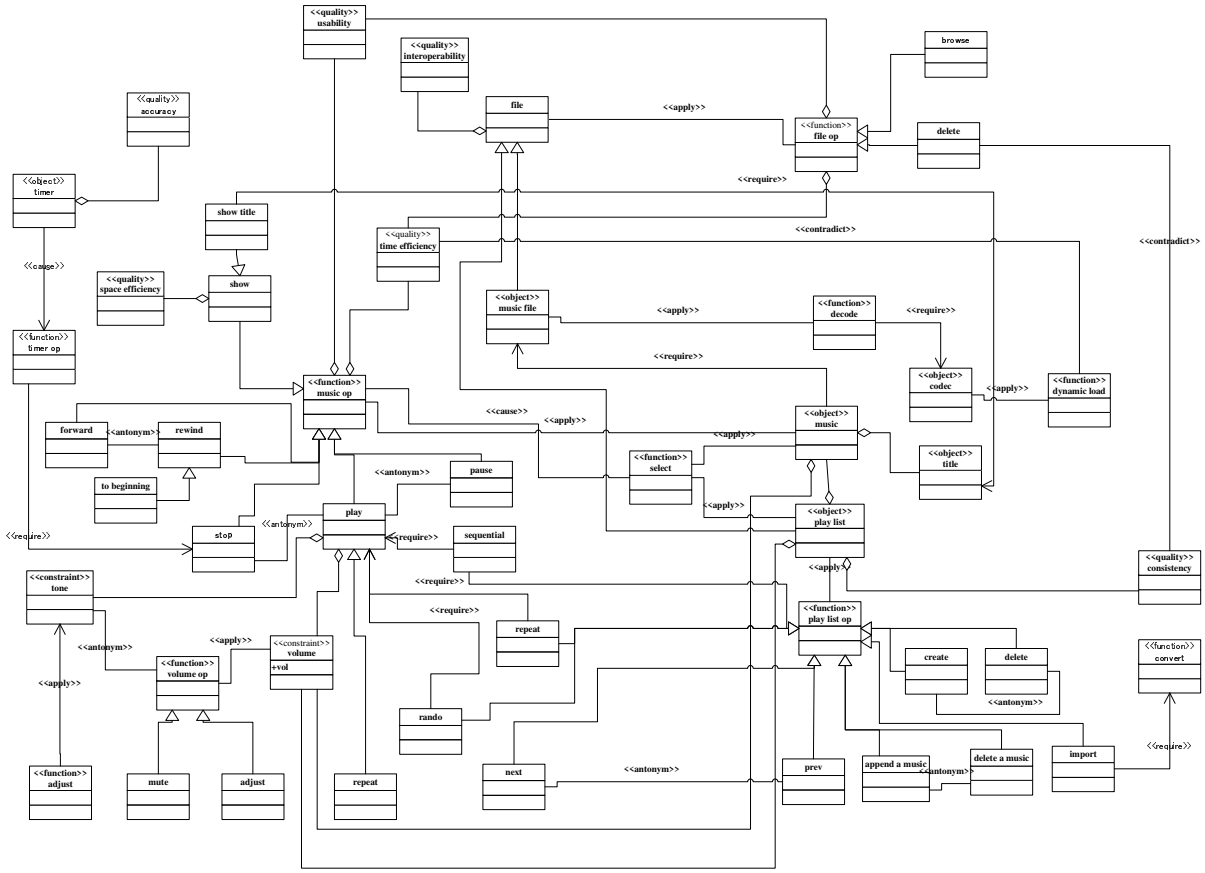


図 5　音楽プレイヤーのオントロジー

る操作が何も書かれていない．要求仕様 A にある語「プレイリスト」は図 5 のオントロジーにおける object(play list) に写像される．以下のような推論規則により，ある object に関する操作 (function) の追加可能性を推論することができる．

$$\forall s \forall x \exists y \cdot ((\mathrm{object}(x) \wedge \mathrm{inSpec}(x, s)) \rightarrow (\mathrm{function}(y) \wedge \mathrm{apply}(x, y) \wedge \mathrm{inSpec}(y, s)))$$

要求仕様 A と object(play list) に関しては，function(append a music) や function(import) 等の追加を検討すべきであることがわかる．実際にどの概念を追加するかは要求分析者や顧客等の判断による．

ここで，object(play list) に適用できる function(import) に着目し，

(6) プレイリストを外部から輸入する．

という要求項目が要求仕様 A に追加されたとする．そして，要求項目 (6) が追加された要求仕様を「要求仕様 B」と呼ぶことにする．function(import) は，function(convert) (プレイリストの形式を変換する機能，他のプレーヤー等のリストを輸入するため．) を必要としていることが図 5 のオントロジーからわかる．このような require 関係に基づいて新しい概念を要求仕様に追加する推論規則は以下のようになる．

$$\forall s \forall x \forall y \cdot (\mathrm{InSpec}(x, s) \wedge \mathrm{require}(x, y) \rightarrow \mathrm{InSpec}(y, s))$$

ここでの例では，

$$\mathrm{InSpec}(\mathrm{import}, \text{要求仕様 B}) \wedge \mathrm{require}(\mathrm{import}, \mathrm{convert}) \rightarrow \mathrm{InSpec}(\mathrm{convert}, \text{要求仕様 B})$$

となり，この推論規則とオントロジーを用いることで，系統的に欠落する可能性のある概念 (ここでは function(convert)) を検出することが可能である．antonym (対語) に対しても同様の推論を行うことができる．例えば，object(play list) に適用可能な function(create) の antonym として function(delete) を系統的に検出できる．

次に矛盾箇所の検出について例示する．要求仕様 B に以下の要求項目群が加わったものを要求仕様 C と呼ぶことにする．

(7) 操作は軽快であること．
(8) 未知形式の音楽ファイルのためのデコーダーは自動的にダウンロードして実行する．

図 5 のオントロジーから，要求仕様 C に対応する概念として，function(music op) をもとに quality(time efficiency) が，object(music) をもとに object(music file), function(decode), object(codec), function(dynamic load) が追加される．これら新規に追加された概念間の関係を吟味すると，

contradict(quality(time efficiency), function(dynamic load))

の関係が検出でき，これによって，追加された要求項目 (7), (8) の間には矛盾がある可能性を検出できる．直感的には「動的なコードのロードと実行を許すと，操作性が損なわれる場合がある」ことを示している．上記の矛盾は以下の推論規則で検出可能である．

$$\forall s \forall x \cdot (\mathrm{InSpec}(x, s) \rightarrow \exists y \cdot (\mathrm{InSpec}(y, s) \wedge \mathrm{contradict}(x, y))$$

検出された矛盾をどのように解消するかは欠落事項の場合と同様に，要求分析者の判断，例えばある種のネゴシエーションを行う等にゆだねられる．

このようにオントロジー上の推論規則を用いることにより，インクリメンタルに要求を増殖させていく要求獲得プロセスにおいては，欠落事項や矛盾点の検出に有効であることがわかる．

4 意味まで考慮したメトリクス

メトリクスは IEEE830 に挙げられている要求仕様書の品質特性のうち要求分析段階のみに関連した以下の 4 特性と扱う．

- 妥当性 = $\frac{\text{オントロジーと対応付いた要求項目数}}{\text{要求項目総数}}$

  オントロジーで意味付けできなかったものは，その問題領域の要求として不適切な可能性があるため．
- 完全性 = $\frac{\text{分母のうちで要求仕様と対応付くものの個数}}{\text{オントロジーの概念と関係の総数}}$

  対応付けられなかったオントロジーに対応する要求が欠落している可能性があるため．
- 無矛盾性 = $\frac{\text{分母の中で contradict 以外の個数}}{\text{要求に対応付いた概念間にある関係の総数}}$
- 非曖昧性 = $\frac{\text{相互に無関係な要素に対応付く要求項目数}}{\text{全要求項目数}}$

  1 つの要求項目が複数の意味を取りうるという点で曖昧である可能性があるため．contradict と antonym 以外の関係で推移に到達可能な概念を相互に関係があるとみなし，それ以外の場合を相互に無関係とする．

(1) 音楽の再生できる．
(2) 一時停止できる．
(3) 前後曲の頭だしができる．
(4) 曲操作は早い．
(5) 音楽の早送り，巻き戻しができる．
(6) 音量調整と消音ができる．
(7) 曲毎に音量設定ができる．
(8) 再生速度を変更できる．
(9) 再生音程を変更できる．
(10) プレイリストのランダム再生ができる．
(11) プレイリストに関して削除ができる．
(12) プレイリスト毎に音量設定ができる．
(13) リピート再生ができる．
(14) 未知の記録形式にもソフトウェアの自動追加で対応可能．
(15) 他のアプリから曲のファイルを操作可能．

図 6 音楽プレーヤーへの要求仕様 X

図 5 のオントロジーをもとに，図 6 の要求仕様 X についてのメトリクスを計算すると以下のようになる．

- 妥当性 = 13/15 = 87%
  要求項目 (8), (9) がオントロジーに写像できない．
- 完全性 = 51/115 = 44%
  115 の内訳は概念が 48, 関係が 67 個．
- 無矛盾性 = 35/36 = 97%
  要求項目 (4) と (14) に含まれる概念で contradict の関係で結ばれるものがある．
- 非曖昧性 = 13/15 = 87%
  要求項目 (11), (13) が無関係な概念に同時に写像される．(11) はリスト自体の削除かリストにある曲の削除かが曖昧．(13) はリストのリピートか単曲のリピートかが曖昧．

それぞれのメトリクスの妥当性について議論する．オントロジーには contradict の関係が含まれているため完全性が 100%である要求仕様は，本質的に矛盾を含むことになるため，完全性の直感的な定義に反する．また，現在の無矛盾性の定義では見落としてしまう矛盾項目がある．例えば，要求項目 (15) を認めれば，(10) から (12) にあるプレイリストの一貫性を損ねる恐れがあり，これは矛盾を引き起こす可能性がある．上記の点を考慮しつつメトリクスの定義を改善したい．

5 変更予測

5.1 オントロジーを用いた変更パターン

要求の変更は頻繁に起こる．開発の最中にも生じ，その際の影響解析が大きな問題となる．ある機能要求を追加したために，既存の要求と矛盾を生じることもある．意味的な側面からの影響解析や矛盾の検出は 3 章で述べた手法，すなわち要求と対応付けられたオントロジーの要素間の関係を解析することにより行うことができる．本章では，これらの応用とは異なる，オントロジーを用いた要求変更に関する支援技術を述べる．頻繁に起こる変更をパターン化し，次の時点で起こる要求変更を予測できれば，

(1) 要求仕様のどの部分が安定しているか (将来の変更がない)，揮発性が高いか (変更が将来も頻繁に起こり得る) を判断できる
(2) 予測結果をガイドとし，より完成度の高い要求仕様書をインクリメンタルに完成していく作業の支援が行える

という利点がある．変更パターンを考える際には，単なる要求仕様の構文的な変更ではなく，変更の意味を考慮しない限り，有用な変更パターンは得られない．本章では，これまでの音楽プレイヤーの例をもとに，オントロジーを使って起こった変更の意味を解析し，どのようなパターンが作れるかを考察する．

音楽プレイヤーの変更履歴にあるバージョンアップに着目し，どのような変更パターンがあり，一般的に要求の変更は，以下のようなものが考えられる．

(1) 機能の新規追加: 不足している機能を発見して追加する．
(2) 既存機能の品質の改善: 既存機能のパーフォーマンス，ユーザビリティ，インタオペラビリティなど，品質に関する非機能的な要求の改善や追加で，実際にプロトタイプやソフトウェアを稼動させ評価を行った後で，変更がなされることが多い．

(1) はオントロジーの require 関係を辿ることにより，既存機能と関係のある機能であれば機械的に発見できる．この機構は 3 章で述べた欠落要求の発見・追加である．(2) は図 2 の function のインスタンスが追加された後，それと aggregate の関係を持つ quality のインスタンスに意味付けされる要求が追加されたり変更されたりすることを意味する．要求仕様の第 i 版を S_i と記述し，i が大きいほど新しい版とすると，上記の 2 の変更パターンは，

$$(\neg \mathrm{InSpec}(x, S_i) \wedge \mathrm{InSpec}(x, S_{i+1})) \rightarrow \exists j \exists y \exists r_{i+1} \exists r_j \cdots ((j > i+1) \wedge ((\mathrm{InSpec}(y, S_j) \wedge \mathrm{quality}(y) \wedge \mathrm{Rel}(x, y) \wedge ((y \in F(r_j) \wedge ((y \notin F(r_{i+1})) \vee ((y \in F(r_{i+1}) \wedge r_{i+1} \neq r_j)))$$

と書ける．ここで，Rel(x,y) は，function(x) と関係を持つ quality(y) を表し，x に aggregate で接続された y か，x が適用される object に aggregate で接続された y を表す．継承関係を考慮すると，以下のように定義される．

$$\mathrm{Rel}(x, y) = \exists a \cdot (\mathrm{generalize}(a, x) \wedge \mathrm{aggregate}(a, y)) \vee$$
$$\exists b \exists c \cdot (\mathrm{generalize}(b, x) \wedge \mathrm{apply}(c, b) \wedge \mathrm{aggregate}(c, y))$$

generalize は別の推論規則で反射かつ推移的であると定義されているとする．変更パターンの上記の式は，要求仕様第 i+1 版で新しい機能 x が追加されたとき，それ以降の j 版では，x と関係 (Rel) のある quality(y) に関する記述が，(1) r_j として新たに追加される ($((y \in F(r_j) \wedge ((y \notin F(r_{i+1}) \ldots)$ か，(2) 記述が r_{i+1} から r_j に変更される $(((y \in F(r_j) \wedge (\ldots \vee ((y \in F(r_{i+1}) \wedge r_{i+1} \neq r_j)))$ かであることを表している．このパターン記述は論理式の形式をとっているが，分析者はバージョン列に，この論理式を適用し満足しない部分を発見し，分析者の判断で変更を行う．

次節で評価した例の中からこのパターンに合うものを示す．ある音楽プレイヤーの開発者は，リリース後，以下のようなバージョンアップを行っていた．

(1) version 1.00 で「Play list を複数作成でき，それらを保存する機能を追加した」
(2) version 1.11 で「Play list を削除できる機能を追加した」
(3)「Play List の操作を高速化した」
(4)「Play List の選択が一覧表からできるようにした」

(1) から (2) の変更は図 5 のオントロジーには，生成・保存 (create) と削除 (delete) の間に antonym 関係があるため，分析者に削除機能の追加を示唆することができる．(3) と (4) への変更は，Play List のスーパークラスである object(file) およびその操作 function(file op) と関係がある quality(time efficiency), function(browse) から変更予測を行うことが可能である．

5.2 評　　価

実際に音楽プレイヤーのバージョンアップ記録 (自然言語で書かれている) をオントロジーに対応付け，前節で述べたパターンが当てはまるかどうかを分析した．このソフトウェアは version 1.00 から version 2.02 まで 35 の version の更新履歴を持っている．変更はバグや不具合の修正を除き 50 個なされた．これらのオントロジーに対応づけた結果を表 1 に示す．合計が 50 を超えているのは 1 つの変更が複数のオントロジー概念に対応づいているからである．

表 1　変更内容の内訳

対応するオントロジー要素のタイプ	出現数
Function (機能の追加)	12
Reliability (Error Handling)	3
Time Efficiency	3
Interoperability	6
Usability	30
Accuracy	1

例えば，Interoperability に対応付けられた変更は再生可能なファイルの種類を増やす，Time Efficiency は再生処理の効率化，Usability はショートカットキーの導入やポップアップメニューの改善などである．12 の機能追加のうち，以降のバージョンでその機能に関連した Quality に関するバージョンアップがあった機能が 2 つあった．残りの 10 の機能のうち，8 は Quality に関する変更を伴っている，あるいは Quality の変更を行ったことによる機能変更で，同時にあったとみなすことができる．残りの 2 つが純粋な機能追加である，このことより，純粋な機能追加があったとき，その半分が以降のバージョンでその機能の Quality に関する変更が行われていることがわかる．つまり，機能追加，機能に対する品質の向上 (time efficiency，usability, interoperability など) が変更のパターンとして続くことがわかり，これをオントロジー上でパターン化した規則を作り，それによって将来の変更を示唆することができる．

6 議　　論

本研究ではドメイン固有の高品質なオントロジーが必須であるため，その構築法について検討の必要がある．我々は種々の自然言語記述 (操作手引き，変更履歴，ユースケース記述，シナリオ) をマイニングすることによりオントロジーを構築する方針である．多くのオントロジー構築法は具体的な手順が曖昧であり，結局，オントロジー構築者の個人技量に頼る部分が多いことが知られている [4] が，マイニングによりこの問題を解決できるのではないかと思われる．

自然言語処理を利用した要求分析技術の研究は多数ある [5], [6], [7]．しかし，これらの研究ではドメイン知識の扱いが不明確であったり，要求文自体の品質の分析が不明確な部分がある．我々の手法では分析対象の要求仕様とドメイン知識の記述であるオントロジーと対応付け処理する点が特長である．

我々の研究では品質項目もオントロジーの概念要素として記述している．しかし，オントロジーとゴール分析モデルを組合わせることで，品質特性 (非機能要求) を扱う試みもある [8]．我々は既に固有のゴール分析法 [9] を提案しているため，それとの連携を今後検討してゆきたい．モデルの拡張という意味では，実現構造を表すオントロジーも必要も必要であるが，これは winwin 手法 [10] の知識が利用可能ではないかと思われる．

文　　献

[1] T. R. Gruber. A translation approach to portable ontologies. *Knowledge Acquisition*, Vol. 5, No. 2, pp. 199–220, 1993.

[2] M. Saeki, H. Horai, and H. Enomoto. Software Development Process from Natural Language Specification. In *Proc. of 11th International Conference on Software Engineering*, pp. 64–73, 1989.

[3] S. Brinkkemper, M. Saeki, and F. Harmsen. Meta-Modelling Based Assembly Techniques for Situational Method Engineering. *Information Systems*, Vol. 24, No. 3, pp. 209 –228, 1999.

[4] Karen Koogan Breitman and Julio Cesar Sampaio do Prado Leite. Ontology as a Requirements Engineering Product. In *11th IEEE International Requirements Engineering Conference (RE'03)*, pp. 309– , Monterey Bay, California, USA, Sep. 2003. Mini-Tutorial.

[5] Didar Zowghi, Vincenzo Gervasi, and Andrew McRae. Using Default Reasoning to Discover Inconsistencies in Natural Language Requirements. In *Eighth Asia-Pacific Software Engineering Conference (APSEC'01)*, pp. 113–120, Macao, China, Dec. 2001.

[6] Scott P. Overmyer, Benoit Lavoie, and Owen Rambow. Conceptual Modeling through Linguistic Analysis Using LIDA. In *23rd International Conference on Software Engineering (ICSE'01)*, pp. 401–410, Toronto, Canada, May 2001.

[7] Vincenzo Gervasi and Bashar Nuseibeh. Lightweight Validation of Natural Language Requirements: A Case Study. In *4th International Conference on Requirements Engineering (ICRE'00)*, pp. 140–148, Schaumburg, Illinois, Jun. 2000.

[8] Luiz Marcio Cysneiros, Julio Cesar Sampaio do Prado Leite, and Jaime de Melo Sabat Neto. A Framework for Integrating Non-Functional Requirements into Conceptual Models. *Requirements Engineering*, Vol. 6, No. 2, pp. 97 – 115, Jun. 2001.

[9] Haruhiko Kaiya, Hisayuki Horai, and Motoshi Saeki. AGORA: Attributed Goal-Oriented Requirements Analysis Method. In *IEEE Joint International Requirements Engineering Conference, RE'02*, pp. 13–22, Sep. 2002.

[10] Barry Boehm and Hoh In. Identifying Quality-Requirement Conflict. *Software*, Vol. 13, No. 2, pp. 25–35, Mar. 1996. IEEE.