Vol.2016-SE-191 No.28
2016/3/15

部分グラフの自動検索に基づいた モデリングヒントの推薦

荒木円博[†1]　足立正和[†1]　稲森豊[†1]

概要：システム開発において，システムの要求仕様，構造，振る舞いなどを表すモデルは，品質確保や開発効率化の観点から重要である．しかし一方で，システムをモデリングする過程において，盛り込むべき情報を取捨選択したり，情報の詳細度を適切に揃えることが非常に難しいと指摘されている．効率的なモデリングを可能にするためには，これらを支援する技術が必要である．本研究では，開発者のモデリング負担を軽減する手段の 1 つとして，モデリング作業における情報のレコメンドおよび自動補完に着目する．具体的には，開発中のモデルの不備を補うような情報を既存のモデルから自動的に検索・抽出し，モデリングのヒント（補完候補）としてレコメンドする手法を提案する．提案手法では，既存モデルに対してグラフパターンを用いた照合を行うことにより，開発中のモデルと構造的かつ意味的に類似した，開発者にとって有用な情報を検索し，必要に応じて開発中のモデルへの自動補完を行う．

キーワード：システムモデリング，自動補完，グラフパターン，シソーラス，ジャカード係数

Recommendation of Modeling Clue based on Automatic Subgraph Searching

MITSUHIRO ARAKI[†1]　MASAKAZU ADACHI[†1]　YUTAKA INAMORI[†1]

Abstract: Modeling of system requirements, structure, behavior, etc. is important in terms of quality assurance and efficiency improvement in the system development. However, at the same time, the modeling activity often requires considerable efforts for eliciting essential system information at the appropriate level of abstraction to the developer. In order to reduce such difficulties, we propose a modeling support technique which automatically suggests the missing information as modeling clues and complements the developing model by searching and extracting highly relevant pieces of information to the developing model from the assets of existing models. The proposed method employs graph pattern matching to find a set of structurally and semantically similar information from the assets.

Keywords: system modeling, autocomplete, graph pattern, thesaurus, Jaccard index

1. はじめに

システム開発において，システムの要求仕様，構造，振る舞いなどを表すモデルは，品質確保や開発効率化の観点から重要である．しかし，システムモデリング作業において，盛り込むべき情報の取捨選択や，モデリングの粒度を適切にそろえることの難しさが指摘されており，これらの点がとりわけモデリング経験の浅い開発者にとって大きな障壁となっている[1].

このような課題に対して，モデリング作業を効率的に行うために，モデリング経験の豊富な開発者によって作成されたモデルを参照するのは有効な手段の 1 つであるといえる．洗練されたモデルは，本質を捉えていて，かつ簡潔でわかりやすいため，情報の取捨選択やモデリングの抽象度を決める上での規範として再利用すべきであり，また規範的なモデルに基づいて自身のモデルを適切に修正することは，モデルの質の向上に加え，モデリングのスキル向上[1]につながると期待できる．

しかし，モデリングに従事している開発者が，自身のモデルの手本となるようなモデリング事例を既存のモデルの中から探し出すのは困難である．そこで，本稿では，開発者が着目するモデル要素に対して規範となるような事例を，既存モデルのリポジトリから自動的に検索し，モデリングヒントとして推薦する手法を提案する．

本稿の構成は以下の通りである．まず 2 章で研究の動機を述べ，3 章で提案手法，4 章で提案手法に基づく推薦システムのプロトタイプについて説明する．5 章で手法の適用例，6 章で関連研究について述べ，最後に 7 章でまとめと今後の課題について議論する．

2. 研究の動機

システム開発中に作成する様々なモデルの中でも，特に，開発中の課題解決における解決案，案の評価，案の選択理由などを表す設計根拠モデル[2]は，要求仕様間のトレードオフを考慮したり，仕様変更に的確に対応する上で，非常に重要である．しかし通常，設計根拠モデルを記述するには，システムのステークホルダーやシステムが利用される状況について理解しておく必要があり，経験の浅い開発者にとって容易ではない．本章では，例としてソーティングアルゴリズム[3]の選択に関する設計根拠のモデリングを

†1 株式会社 豊田中央研究所
TOYOTA CENTRAL R&D LABS., INC.

とりあげ，経験の違いによるモデルの有用性の違いを通じて，研究の動機を述べる．以後本稿では，設計根拠モデリングの代表的な手法の 1 つである IBIS (Issue-Based Information System)[4]にならい，《課題》，《案》，《肯定的評価》，および《否定的評価》の要素と，《案》に関連するシステムの《要件》や《制約》の要素を用いて設計根拠をモデリングすると仮定する．

例えば，経験の浅い開発者は，ソーティングアルゴリズムの選択について設計根拠をモデリングする場合，各アルゴリズム（図 1 内の《案》）の計算量的な優劣を重視するあまり，最終的なアルゴリズムの選択に決定的な根拠を見いだせない状況に陥ってしまう．

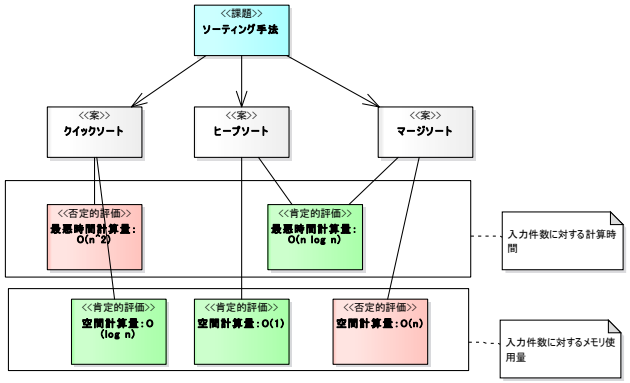
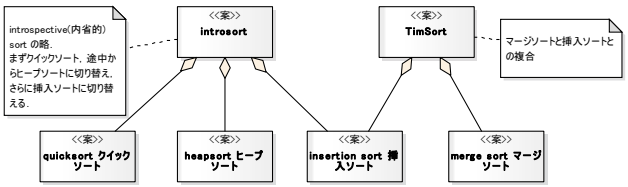


図 1　経験の浅い開発者によるモデリング例

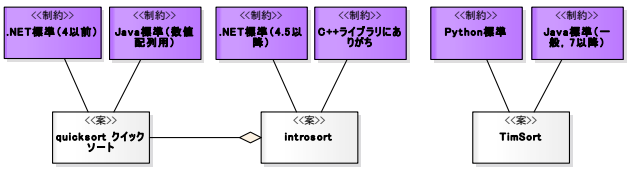
一方，経験豊富な開発者が，代表的なアルゴリズムについて，アルゴリズム間の関連（図 2a），利用可能なプログラミング言語などの実装上の制約との関連（図 2b），および利用目的に対応した性能要件との関連（図 2c）など，採否判断に役立つモデルを過去に作成しているとする．

仮に経験の浅い開発者が，経験豊富な開発者によるモデルを見つけられれば，自身のモデルが修正できるはずである．例えば，Microsoft .NET 4.0 を前提に大量データを扱うシステムを開発しているなら，図 2 のモデル中の《案》「クイックソート」に，《制約》「.NET 標準（4 以前）」と《要件》「データの規模重視」を対応付け，「クイックソート」が妥当な案であることを明示できる．さらに，この修正を通じて，開発者は設計根拠モデリングにおいて要求仕様を考慮することの重要性に気づくであろう．

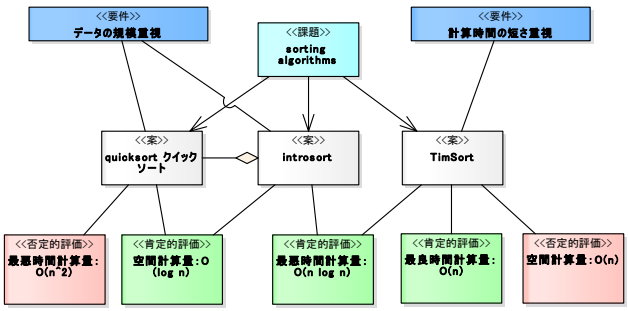
このように，経験豊富な開発者によるモデリング事例は，経験の浅い開発者にとって，自身のモデルを洗練させるための規範として非常に有用であるが，大量に存在する既存のモデルの中からこのような事例をピンポイントで探すのは困難な上，モデリングへの集中力を削ぐ恐れもある．本研究の狙いは，この例にあるように，モデリング中のモデルの不備を補うのに適したモデリング事例を自動で検索できるようにすることである．



a アルゴリズム（《案》）間の関連



b アルゴリズムと制約との関連



c アルゴリズムと要件との関連

図 2　経験豊富な開発者によるモデリング例

3. 提案手法

提案手法は，開発者が着目するモデル内の要素（例えば，編集中の要素）に対して，モデリングの規範となるようなモデリング事例を既存のモデルのリポジトリから自動的に検索し，開発者が着目するモデル要素に対する補完候補（モデリングヒント）に変換して，開発者に提示する．例えば図 1 の《案》「クイックソート」に着目している場合には，図 2 の既存モデルに基づいて《案》「クイックソート」に《制約》「.NET 標準（4 以前）」などを対応付けたモデリングヒントを提示する（図 3）．以後，文献[5]にならって，モデリングヒントの提示のことを，モデリングヒントの推薦と呼ぶ．

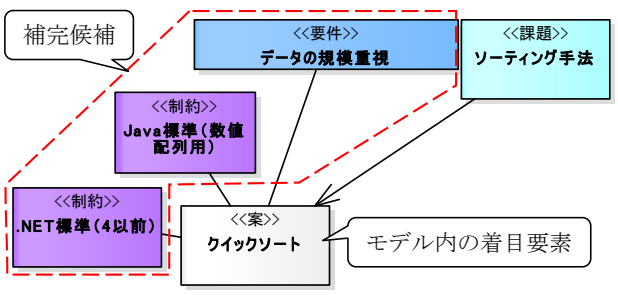


図 3　モデリングヒントの例

提案手法で扱うモデルは，ダイアグラム，すなわち頂点と辺から構成されたグラフとして表現されていると仮定する．提案手法の特徴は，経験の浅い開発者が考慮し忘れがちな種類の要素（例：2 章における《制約》や《要件》）が効率的にモデリング事例として提示されるようにするため，モデリングの規範となる抽象的なグラフ構造（**規範パターン**）をあらかじめ定義しておき，規範パターンと照合しながらモデリング事例を検索することにある．

3.1 提案手法の概要

提案手法は，規範パターンおよび規範パターンをもとにあらかじめ定義しておく**不備パターン**を利用して，モデリングヒントの推薦を行う．

規範パターンはどう補完すべきかを表し，不備パターンはモデリング中のモデルにおける補完の推薦箇所を表す．すなわち，規範パターンは経験豊富な開発者，不備パターンは経験の浅い開発者のモデリングのしかたを抽象的に表現したものであり，それぞれ既存モデルとモデリング中のモデルとの照合に利用する．

規範パターンと不備パターンは，グラフの抽象的な構造（頂点や辺の種類など）を表すグラフパターンとして定義する．グラフパターンは，グラフ書換え系[6]やグラフデータベース[7]で利用されている．

規範パターンと不備パターンの例を，それぞれによる照合結果の例とともに図 4 に示す．規範パターンは「《課題》に対する《案》が検討済みで，さらに《案》採否の決め手となる《制約》や《要件》も 1 つ以上検討済みである」，不備パターンは「《課題》に対する《案》は検討済みだが，《案》採否の決め手が抜けている」ということを表す．

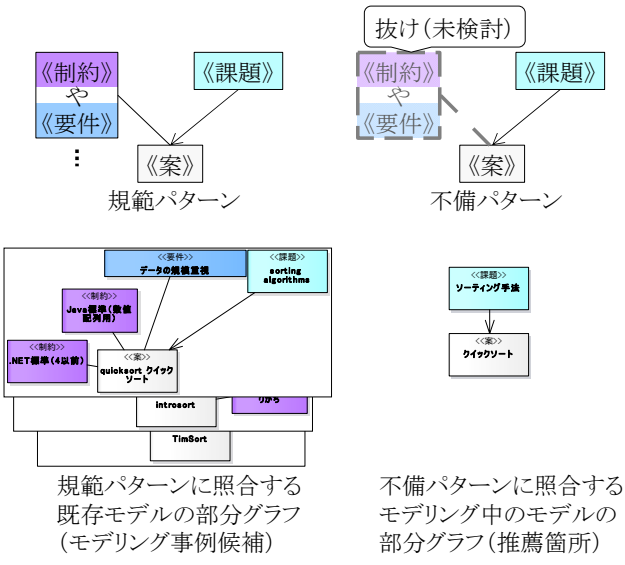


図 4　規範パターンと不備パターンの例

モデリングヒントは，**モデリング事例候補**（規範パターンへの照合結果）を，**推薦箇所**（不備パターンへの照合結果）によって絞り込み，さらに推薦箇所と整合するように変換することによって生成する．

この絞り込みや変換の際に，モデリング事例候補と推薦箇所とが同じ構造になっている箇所（図 4 の場合，《課題》と《案》からなる部分グラフ）を利用するため，照合結果の頂点に対して，グラフ構造上の対応が一意にわかるような番号付けを行う（例：《案》を 1 番目，《課題》を 2 番目）．具体的には，番号付きの変数名を導入し，パターンに照合した頂点にパターン上の変数名を付与する．図 5 に番号付き変数名を含む規範パターンと不備パターンの定義例を示す．この例において，番号付き変数名はv_{c1}と v_{i1}，およびv_{c2}と v_{i2}である．なお《要求仕様項目》は《制約》と《要件》の総称である．

v_{c2}:《課題》--> v_{c1}:《案》-- :《要求仕様項目》[1..*]

a 規範パターン

v_{i2}:《課題》--> v_{i1}:《案》-//- :《要求仕様項目》

b 不備パターン

図 5　規範パターンと不備パターンの定義例

3.2 提案手法の処理の流れ

提案手法は，図 6 に示す 4 つの処理から構成される．

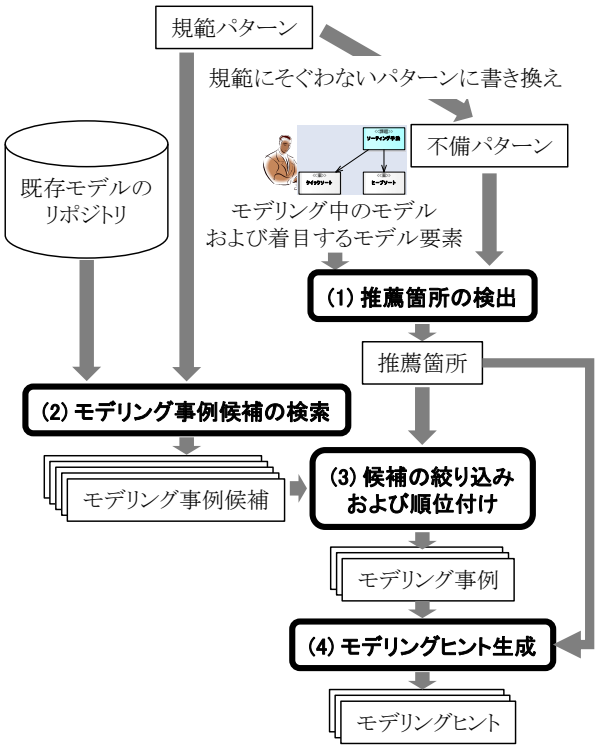


図 6　提案手法の処理の流れ

以降，各処理について述べる．なお 2 章図 1 をモデリング中のモデル，図 2 を既存モデルとみなして説明する．

3.2.1 推薦箇所の検出

推薦箇所の検出では，開発者が着目するモデル内の要素を含み，かつ不備パターンに照合するグラフ（推薦箇所）を取得する．推薦箇所は，モデリング中のモデルの部分グラフであり，開発者が関心のあるコンテキストに相当する．推薦箇所の各要素には，不備パターン上に示した変数名（v_{i1}, v_{i2}）がラベルとして付与される（図 7）．

推薦箇所の検出は，開発者によるモデル要素の指定をきっかけに行う．

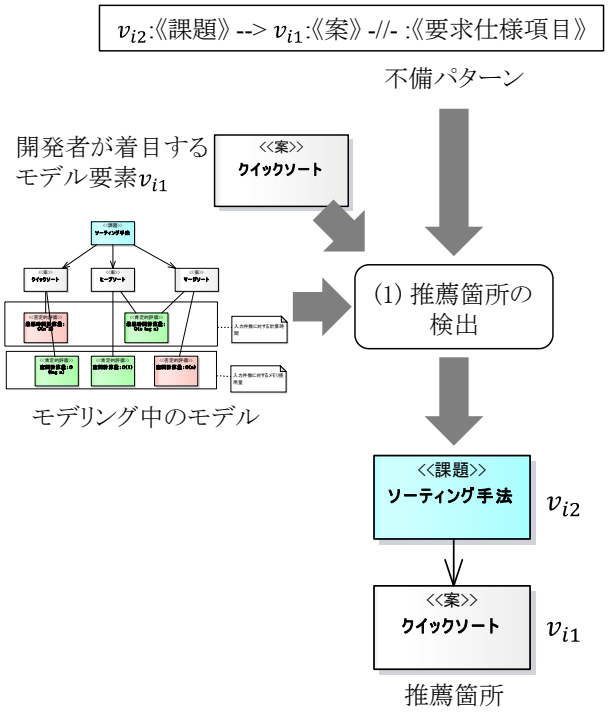


図 7　推薦箇所の検出例

3.2.2 モデリング事例候補の検索

モデリング事例候補の検索では，複数の既存モデルの中から，規範パターンに照合する部分グラフを列挙する．この部分グラフがモデリング事例候補である．モデリング事例候補においても，推薦箇所の場合（3.2.1 節）と同様に，各要素に規範パターン上に示した変数名（v_{c1}, v_{c2}）がラベルとして付与される（図 8）．

モデリング事例候補のうち変数名が付与された部分と，推薦箇所とは，規範パターンと不備パターンの両方に共通するパターンに従っているため，構造的に一致している．

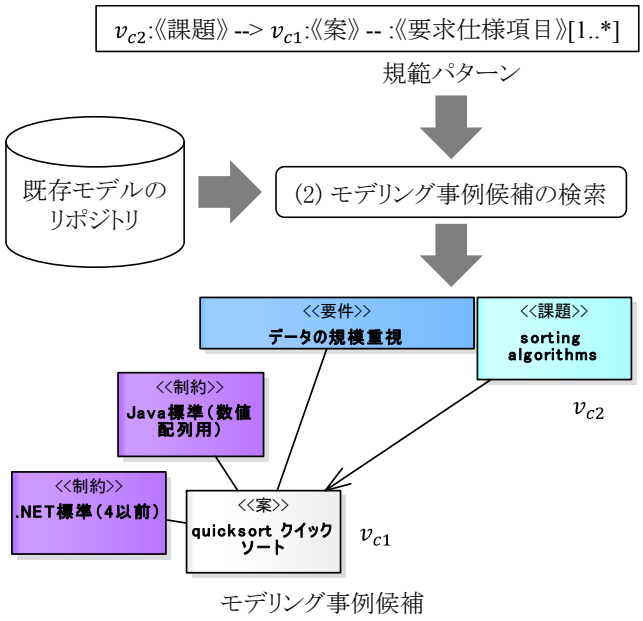


図 8　モデリング事例候補の検索例

3.2.3 モデリング事例候補の絞り込みおよび順位付け

この処理では，複数のモデリング事例候補の中から，変数名が付与された要素が推薦箇所と意味的に対応するものを選び（絞り込み），推薦箇所の不備を補うのに適したモデリング事例として，対応の強い順に順位付けする．

絞り込みにおける意味的な対応とは，モデリング事例候補と推薦箇所，それぞれの要素に付与された変数名の番号が同じ要素対すべてについて，両方の要素名に同一とみなせる単語があることを表す．ここで要素名とは，要素の意味を表す語句や短文を指し，図 8 における「quicksort クイックソート」などが該当する．また同一とみなせる単語とは，字面が一致する単語だけでなくシソーラス（同義語辞書）上で同義な単語も含む．

図 9 にモデリング事例候補と推薦箇所が対応する例を示す．この例では要素対$< v_{c1}, v_{i1} >$の両方に字面が同じ「クイックソート」が含まれており，また$< v_{c2}, v_{i2} >$それぞれの要素名に含まれる「sorting」と「ソーティング」がシソーラス上（例：日本語 WordNet[8]）で同義語である．したがって図 9 のモデリング事例候補は，この推薦箇所に対するモデリング事例となる．

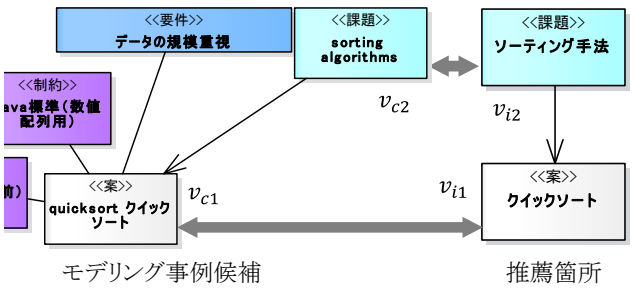


図 9　モデリング事例候補と推薦箇所が対応する例

次に，モデリング事例候補と推薦箇所の対応の強さ（部分グラフ対応度）について述べる．部分グラフ対応度の定義にあたって，補助的な指標として，対の両方の要素名に同一とみなせる単語がない場合には 0，同一とみなせる単語が多いほど値が大きくなる要素名対応度を導入する．この要素名対応度を利用すると，部分グラフ対応度は，要素名対応度のいずれかが 0 の場合（すなわちモデリング事例候補と推薦箇所が対応しない場合）に 0，そうでない場合には各要素名対応度の集約値（例えば算術平均）として定義できる（式 1）．

部分グラフ対応度

$$\equiv \begin{cases} 0 & (\exists \text{要素名対応度} = 0) \\ \frac{1}{n}\sum_{\text{変数名番号}}^{n} \text{要素名対応度} & (\text{otherwise}) \end{cases} \quad (1)$$

ここで要素名対応度の定義にあたって，要素名の特徴について考察する．異なるモデル間で要素名のつけ方が異なり得ることを考慮すると，要素名に出現する単語のうち助詞は無視すべきであり，また語順も無視すべきである．そこで要素名に対して助詞を無視した単語集合を導入し，要素名対応度を共通要素の多さとして定義する．共通要素を求める際，シソーラス上の同義語同士は一致するとみなす．助詞を無視した単語集合は，形態素解析によって得られる．

要素名対応度は，具体的には，集合間類似度の一種であるジャカード係数として定義する．集合XとYの間のジャカード係数$\mathrm{Jaccard}(X,Y)$は，和集合の要素数に対する共通要素数の比率として定義され，和集合が不要な（すなわち計算時間が短い）式で算出できる（式 2）．

$$\mathrm{Jaccard}(X,Y) \equiv \frac{|X \cap Y|}{|X \cup Y|} = \frac{|X \cap Y|}{|X| + |Y| - |X \cap Y|} \quad (2)$$

なお部分グラフ対応度の値が同じモデリング事例については，要素名における字面上の類似度に基づいて，対応度と同様に部分グラフの字面類似度を求め，字面類似度の降順で順位付けする．要素名の字面類似度は，例えば式 2 の共通集合および要素数を，それぞれ最長共通部分列と列長に置き換えた式によって求められる．

3.2.4 モデリングヒントの生成

一連の処理の締めくくりとして，3.2.3 節で得られたモデリング事例を，開発者が着目するモデル要素に関する補完候補（モデリングヒント）に変換して推薦する．この変換では，モデリング事例の中で変数名が付与された要素について，その要素名を同じ番号の変数名が付与された推薦箇所の要素名に置換する（図 10）．

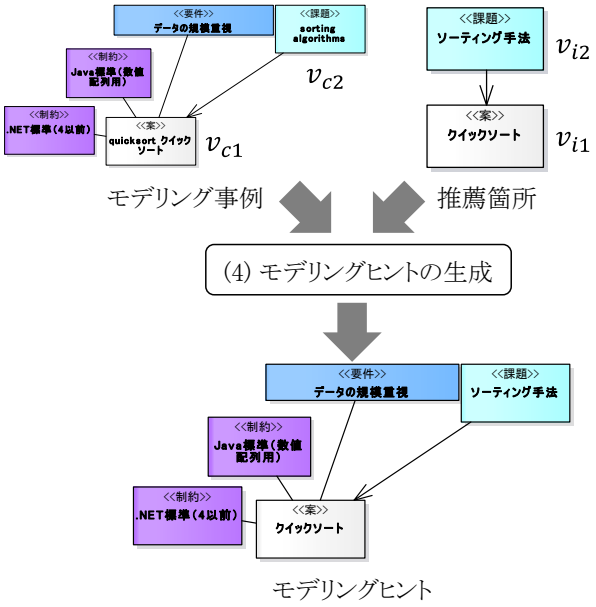


図 10　モデリングヒントの生成例

モデリング事例を推薦箇所と整合するように変換することによって，補完実施可否が判断しやすくなると期待できる[9].

4. モデリングヒントの推薦システム

3 章で述べた提案手法を，モデリングヒントの推薦システムのプロトタイプとして実現した．以下，プロトタイプの実現に利用したソフトウェア，および動作について述べる．

4.1 プロトタイプの実現に利用したソフトウェア

プロトタイプの実現には，フロントエンドにモデリングツール Enterprise Architect[10], グラフパターンに基づく部分グラフ検索にグラフ書換え系の GrGen.NET[6]，モデリング事例候補の絞り込みおよび順位付けにシソーラスの日本語 WordNet[8]と形態素解析器 MeCab[11], そしてモデリングヒントのグラフ可視化のために MSAGL[12]を利用した．プロトタイプは Microsoft .NET Framework 4 上で動作する．

4.1.1 グラフパターンの実装

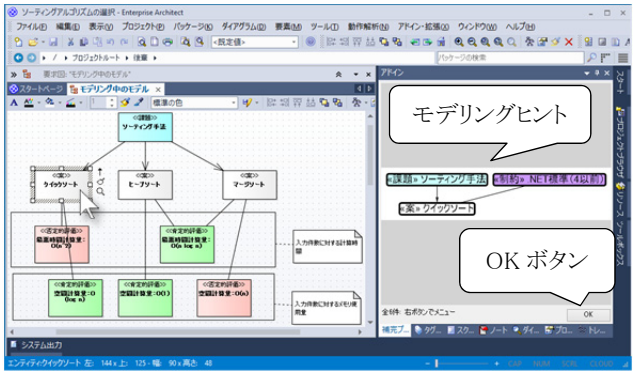
GrGen.NET において，グラフパターンはグラフ検索関数定義の中に埋め込んで表す（図 11）．図 11 の 2〜6 行がモデリング事例候補の検索関数定義で，うち 3,4 行が図 5a の規範パターンを表す．また 9〜14 行が推薦箇所の検出関数定義である．検出関数の 10〜12 行が図 5b の不備パターンを表し，引数（9 行の `vi1`）は着目する要素を表す．どちらの関数も，パターン中の変数に対応する要素を返すように定義する（5,13 行）．

```
1 //  モデリング事例候補の検索関数定義
2 test SearchCandidates : (Node, Node) {
3   vc2:Issue --> vc1:Position;
4   vc1 -- :Req;
5   return(vc1, vc2);
6 }
7
8 //  推薦箇所の検出関数定義
9 test Detect(vi1:Node) : (Node, Node) {
10  if (Position <= typeof(vi1));
11  vi2:Issue --> vi1;
12  if (0 == countAdjacent(vi1, UEdge, Req));
13  return(vi1, vi2);
14 }
```

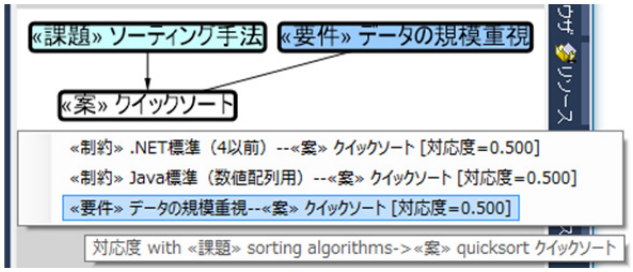
図 11　GrGen.NET によるグラフパターンの表現例

4.2 プロトタイプの動作

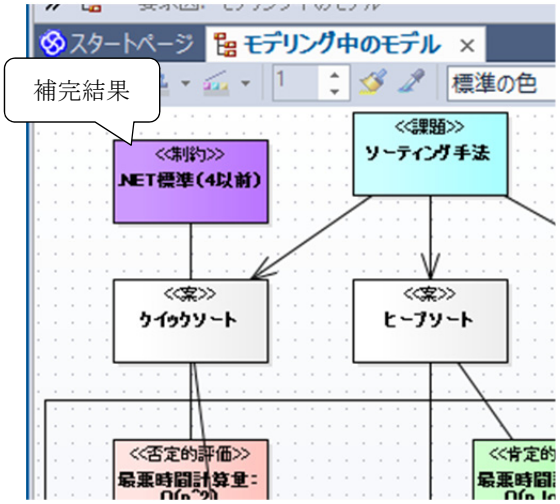
プロトタイプにおいて一連の処理はモデル要素の選択によって開始され，その結果モデリングツールのウィンドウ端の区画に部分グラフ対応度最大のモデリングヒントが 1 つ表示される（図 12a）．モデリングヒントが複数ある場合には，コンテキストメニューによってヒント一覧の表示およびモデリングヒントの切り替えができる（図 12b）．モデリングヒントを使って実際に補完する場合には，モデリングヒントに付随する OK ボタンを押下することで，モデリング中のモデルにモデリングヒントが反映される．図 12a のモデリングヒントによる補完の実施例を図 12c に示す．



a モデリングヒントの推薦例



b モデリングヒントの一覧表示と切り替え例



c 補完の実施例

図 12　プロトタイプの動作

5. 適用例

2 章で例示したソーティングアルゴリズム選択に関する設計根拠モデルを対象に，プロトタイプを適用した結果について述べる．モデリング中のモデルおよび既存モデルとして，それぞれ図 1 と図 2 を再び用いる．

既存モデルの特徴として，個々のアルゴリズムに対する要求仕様項目がない場合でも，そのアルゴリズムを利用している別のアルゴリズムについては要求仕様項目が存在することが挙げられる．例えば，《案》「merge sort マージソート」自身には要求仕様項目が関連付けられていないが，《案》「merge sort マージソート」と《案》「insertion sort 挿入ソート」を利用した《案》「TimSort」には，《制約》「Java 標準」や《要件》「計算時間の短さ重視」が関連付けられている．

このような特徴を考慮すると，開発者がモデリング中のモデルにおいて《案》「マージソート」に着目した時には，これら要求仕様項目が《案》「TimSort」とともに推薦されるのが望ましい（図 13）．

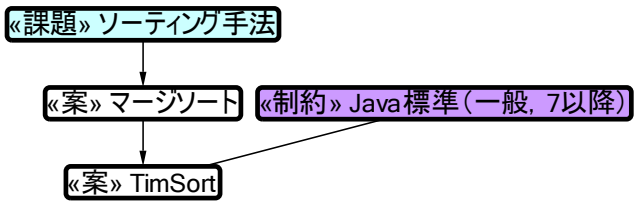


図 13　《案》マージソートの場合に望ましい推薦例

こうした推薦を可能にするため，規範パターンとして，要求仕様項目のみを推薦する図 14a に加え，要求仕様項目が関連付けられた《案》とともに推薦する図 14b を定義した．図 14b は，個々の《案》（v_{c1}）を集約した《案》に，要求仕様項目が関連付けられていることを表している．

v_{c2}:《課題》--> v_{c1}:《案》-- :《要求仕様項目》

a 要求仕様項目の単独推薦用

v_{c2}:《課題》--> v_{c1}:《案》--◇ :《案》-- :《要求仕様項目》

b 要求仕様項目と《案》の組合せ推薦用

図 14 適用例における規範パターン

これら 2 つの規範パターンは，GrGen.NET の条件分岐的な構文（図 15 の 4～11 行）によって，1 つのモデリング事例候補の検索関数として表現できる．

```
 1 //  モデリング事例候補の検索関数定義
 2 test SearchCandidates : (Node, Node) {
 3   vc2:Issue --> vc1:Position;
 4   alternative {
 5     a {
 6       vc1 -- :Req;
 7     }
 8     b {
 9       vc1 -:Aggregation-> :Position -- :Req;
10     }
11   }
12   return(vc1, vc2);
13 }
```

図 15 GrGen.NET による図 14 の規範パターンの表現例

図 14 のグラフパターンによって，《案》「クイックソート」に対する推薦に加え（図 12），《案》「マージソート」についての推薦も可能となる（図 16）．

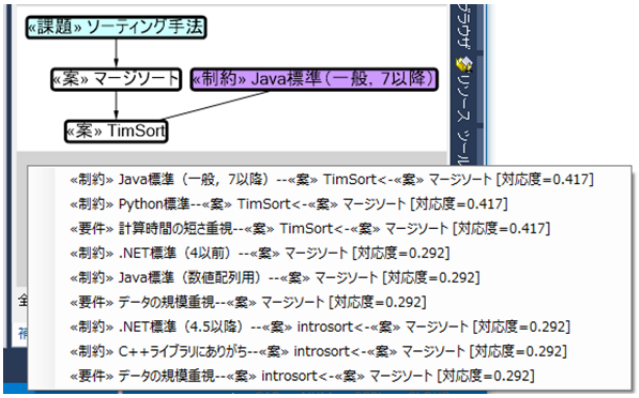


図 16 《案》マージソートについてのモデリングヒント例

このように，提案手法では，規範パターンを蓄積していくことにより，推薦箇所の適用範囲を拡張することが可能となっている．また規範パターンは選択的に適用できるため，開発者の熟練度や，規範パターンの重要性に応じて，モデリングヒントとして推薦する情報を柔軟に制御できる点も有用である．

6. 関連研究

関連研究として，モデリング支援のための推薦システムおよび自動検索に基づく推薦システムについて述べる．

6.1 モデリング支援のための推薦システム

モデリング支援のための推薦システムとして，Dyck らが HERMES Reuse を開発している[13][14][15]．HERMES Reuse は既存モデルを文字列検索する機能，および検索結果によるモデリング中のモデルの補完機能を提供し，Eclipse Modeling Framework 上で動作する．図 17 に HERMES Reuse のデモ版による文字列検索例を示す．この例では検索文字列入力欄（検索窓）への「per」の入力に対し，「per」を含む 5 つのモデルが検索されている．また検索結果一覧上でのモデル名（「Student」）選択に対してモデルの補完プレビューが表示されている．

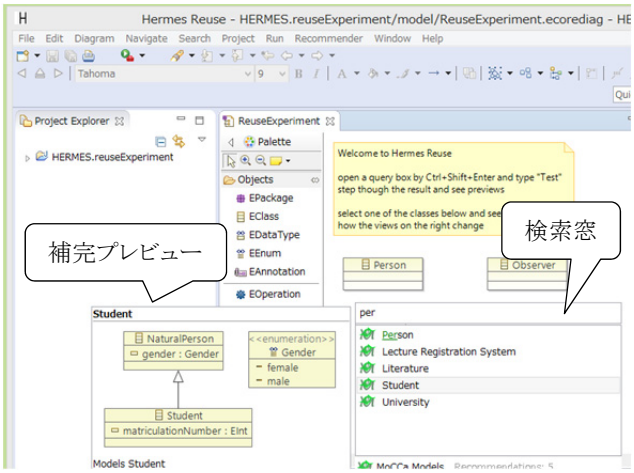


図 17 HERMES Reuse デモ版[15]による文字列検索例

HERMES Reuse のデモ版には，選択したモデル中の要素に基づく自動検索機能も実装されている．図 18 の例では，選択した要素の名前「Person」による検索と，他のモデリング中の要素の名前「Observer」による検索が，別々に行われて提示される．

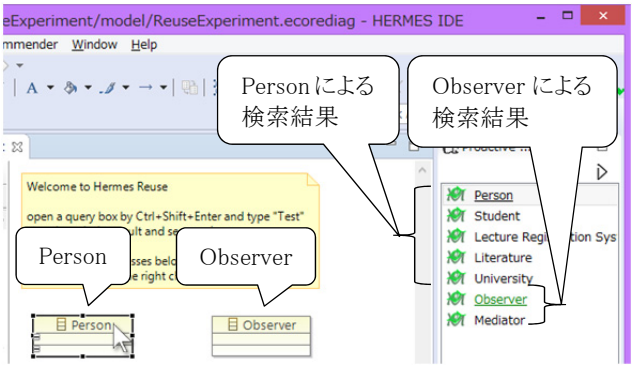


図 18 HERMES Reuse デモ版[15]による自動検索例

一方，提案手法は，着目した要素を含む部分グラフ（推薦箇所）に関する検索を行うため，いわば HERMES Reuse は「or 検索」，提案手法は「and 検索」を行っているとみなせる．すなわち，提案手法は，一般に HERMES Reuse よりも検索結果が絞り込まれる傾向にある．この絞り込みは，開発者の負担軽減の点では有利と思われる．

6.2 自動検索に基づく推薦システム

自動検索に基づく推薦システムとして，島田らが A-SCORE を開発している[16]．A-SCORE はコーディング中の Java クラスに類似した既存 Java クラスを自動検索・推薦するシステムであり，Eclipse 上で動作する．

A-SCORE の特徴は Latent Semantic Indexing(LSI)法[17]によってシソーラス無しでの同義語や類義語の照合を行って類似クラスを検索することにある．LSI 法は多変量解析における主成分分析や因子分析に類似した手法なので，既存の事例が多いほど有効性が高まると思われる．

一方，提案手法はシソーラスの利用によって，既存の事例が少ない場合にも，ある程度の類似検索が可能である．

7. まとめ

システム開発におけるモデリング作業においては，盛り込むべき情報の取捨選択や，モデリング粒度の均一化など，モデリング経験の浅い開発者にとって困難な側面が数多く存在する．このような課題を解決するため，開発者が着目するモデル要素に対して，モデルの不備を補う上での規範となり得る情報を，既存のモデルから自動的に検索・抽出し，モデリングのヒント（補完候補）として推薦する手法を提案した．

提案手法の特徴は，経験の浅い開発者が考慮し忘れがちな種類の要素が効率的にモデリング事例として提示されるようにするため，モデリングの規範となる抽象的なグラフ構造（規範パターン）をあらかじめ定義しておき，規範パターンと照合しながら既存のモデリング事例を検索することにある．

今後，手法評価を通じて，プロトタイプの完成度を高めるとともに，推薦箇所とモデリング事例候補との意味的な類似性を判断する際に，シソーラス不要な手法（例：6.2 節）などを融合し，推薦箇所に関連の強い情報をより適切に抽出できるようにする必要があると考える．さらに，グラフパターンの定義を容易化するため，パターンの機械学習についても検討したい．例えば，提示されたモデリング事例が規範となるかどうかを開発者がラベル付けし，その結果に基づいて各開発者の意図により近い規範パターンを自動的に学習できれば，さらに実用的な推薦システムとなることが期待できる．

参考文献

1) 妻木俊彦 et al.: モデリングは教育できるか？，情報処理学会研究報告, 2007-SE-158 (3), pp.15-22 (2007).
2) Dutoit, A. H. et al. (eds.): Rationale Management in Software Engineering, Springer-Verlag, Berlin Heidelberg (2006).
3) Wikipedia: Sorting algorithm
https://en.wikipedia.org/wiki/Sorting_algorithm
4) Kunz, W., Rittel, H.: Issues as elements of information systems, Working Paper 131, Center for Urban and Regional Development, University of California, Berkeley (1970).
5) Robillard, M.P. et al. (eds.): Recommendation Systems in Software Engineering, Springer-Verlag, Berlin Heidelberg (2014).
6) Geiß, R. et al.: GrGen: A Fast SPO-Based Graph Rewriting Tool, Graph Transformations, pp.383-397, LNCS Vol.4178, Springer-Verlag, Berlin Heidelberg (2006).
7) Neo Technology: Graphs, Patterns, and Cypher, The Neo4j Manual
http://neo4j.com/docs/stable/cypher-intro-patterns.html
8) Bond, F. et al.: Enhancing the Japanese WordNet, Proc. 7th Workshop on Asian Language Resources, pp.1-8, Association for Computational Linguistics (2009).
9) Murphy-Hill, E., Murphy, G.C.: Recommendation Delivery - Getting the User Interface Just Right, in [5], pp.223-242
10) Sparx Systems Japan Co., Ltd.: Enterprise Architect
http://www.sparxsystems.jp/
11) 工藤拓: MeCab
http://taku910.github.io/mecab/
12) Microsoft Research: Microsoft Automatic Graph Layout
http://research.microsoft.com/en-us/projects/msagl/default.aspx
13) Dyck, A. et al.: A Framework for Model Recommenders - Requirements, Architecture and Tool Support, Proc. 2nd Intl. Conf. on Model-Driven Engineering and Software Development, pp.282-290, IEEE (2014).
14) Dyck, A. et al.: On Designing Recommenders for Graphical Domain Modeling Environment, Proc. 2nd Intl. Conf. on Model-Driven Engineering and Software Development, pp.291-299, IEEE (2014).
15) RWTH Aachen University: HERMES
http://modelrecommenders.org/
16) 島田隆次 et al.：開発中のソースコードに基づくソフトウェア部品の自動推薦システム A-SCORE，情報処理学会論文誌, Vol.50, No.12, pp.3095-3107 (2009).
17) Deerwester, S. et al.: Indexing by Latent Semantic Analysis, Journal of the American Society for Information Science, Vol.41, No.6, pp.391-407 (1990).