Vol.2014-SE-184 No.9
Vol.2014-EMB-33 No.9
2014/5/20

メタモデル上でのルール定義に基づく 多ドメイン展開可能な依存関係抽出技術

古家直樹† 村上正敏† 中川雄一郎†
三部良太† 小川秀人†

ドメイン特化モデルベース開発環境はソフトウェアの品質や開発効率向上に有効であるが，対象とするモデルのフォーマットが変わるたびに大規模な変更が必要になり，製品開発の中で継続的に適用し続けることが難しくなる．本研究では，ドメイン特化モデルベース開発環境の中で，メタモデル上で依存関係抽出のルール定義を行うことによる，モデルのフォーマットに依存しない依存関係抽出機能を開発した．また，ある組込み製品のユーザインターフェイス仕様に適用して，その有効性を確かめた．次に，その他の事業ドメイン向けのモデルに対しても，開発した依存関係抽出技術は有効であることを確認し，多ドメインで適用できる見込みを得た．

Multi-Domain Usable Dependency Extracting Technology Based on Rule Definition on Metamodel

NAOKI FURUYA† MASATOSHI MURAKAMI†
YUICHIRO NAKAGAWA† RYOTA MIBE† HIDETO OGAWA†

Domain specific Model-Based Development (MBD) environment is effective for improving software quality and development efficiency. However, if it needs large-scale repair work every time model format changes, we cannot meet the products developing cycle. In this study, we focused on dependency extracting among model elements. By defining extracting rules on metamodel, we developed dependency extracting method independent to model format. We applied the dependency extraction tool to embedded system's user interface and we found our method is effective. Furthermore, we found the dependency extracting is applicable to many domains.

1. はじめに

近年，ソフトウェアの品質及び開発効率向上の手段として，モデルベース開発技術（MBD：Model-Based Development）が普及してきている[1]．モデルを開発に用いることにより設計の品質を向上させ，モデルに基づく開発支援環境を用いることで，開発コストを低減することができる．また，モデルベース開発の効果を更に向上させるため，ドメイン特化モデル（DSM：Domain Specific Model）を定義し，DSM に基づく開発環境（ドメイン特化 MBD 環境）を構築する，ドメイン特化 MBD の取組みも行われている[2][3][4].

一方で，製品開発において DSM の記法の変更や作成する DSM の追加が必要な場合，それに基づいて構築されたドメイン特化 MBD 環境にも変更が必要となり，対象製品へ継続的に適用できなくなってしまうという課題がある[4]．そのため，ドメイン特化 MBD 環境に備える機能（ツール）は，DSM の記法の変更があった際にも最小限の変更コストで継続して利用できることが望ましい.

そこで本研究では, DSM の記法に極力依存しないドメイン特化 MBD 環境を構築することを目的とする．本報告では，DSM の中のある項目（モデルの要素）同士の文言の一致に関する依存関係抽出機能を取り上げる．本研究では，DSM ごとにメタモデルを定義し, そのメタモデル上で依存関係抽出のためのルールを定義することで, DSM の記法に依存しない依存関係抽出技術を提案する．また，提案手法に基づくプロトツールを作成し，実際のシステム開発で用いられている DSM を用いて，文言一致関係の依存関係抽出ができることを確認した.

また，DSM の記法がノードエッジ型，表形式，マトリクス形式であれば，提案の依存関係抽出技術は多ドメインで適用可能である見込みを得た. その内容について説明する.

2. 本研究の目的

まず，ドメイン特化 MBD 環境の定義とその構築方法について説明した上で，本研究の目的を述べる.

2.1 ドメイン特化 MBD の定義

ドメイン特化 MBD とは，対象の事業ドメインごとに DSM を記述し，それをもとにコード生成やテスト生成，動作シミュレーション，モデル検証などの各種開発技術を行う開発手法のことを指す．また，DSM に基づく開発環境のことをドメイン特化 MBD 環境と呼ぶ．DSM を記述する理

† (株)日立製作所 横浜研究所
Hitachi Ltd. Yokohama Research Laboratory

Vol.2014-SE-184 No.9
Vol.2014-EMB-33 No.9
2014/5/20

由は，モデル化対象の特徴に合わせた独自の記法を定義することで，UML[5]などの汎用モデルに比べてモデルの表現力，記述力の向上が期待できるからである．

2.2 対象とするドメイン特化 MBD 環境

本研究では，ドメイン特化 MBD 環境が取り扱う機能として，シンタックスレベルの依存関係抽出機能の開発を目指すこととした．その理由は，一般に自然言語で記述された仕様をモデルで記述することで，仕様の抜け漏れや曖昧さが排除される[6]と言われるが，近年のシステムの大規模化に伴い，例え仕様をモデルで記述した場合でも，人手でのモデルのチェックや，モデル間の依存関係の把握が困難になっているからである．

2.3 ドメイン特化 MBD 環境の構築方法

シンタックスレベルの依存関係抽出を行うことを目的としたドメイン特化 MBD 環境の構築は，

（1）対象となるドメインやシステムの仕様に合わせて DSM の記法を定義

（2）DSM を取り扱い可能な開発環境（ツール群）を構築

という手順で行う．

ここで，（1）の DSM の定義には次の 2 つの手順がある．

（a）DSM に記述すべき項目の決定

（b）DSM のフォーマットの決定

（a）においては，対象とする事業ドメインの課題を分析して，DSM に何を記述するかを決定する．

（b）においては，（a）で決定した記述項目を，どのようなフォーマットで記載するかを決定する．フォーマットは DSM のユーザのニーズや適用する開発工程によって異なる．例えば，同じ状態遷移の DSM であっても，開発者が基本設計工程でシステムの動作概要を把握したい場合には，状態遷移図が用いられ，詳細設計工程では抜け漏れなく動作を設計するためにマトリクス型の状態遷移表を用いることがある．

2.4 対象とする DSM のフォーマット

本研究で取り扱う DSM のフォーマットは，ノードエッジ型，マトリクス形式，表形式とすることにした（図 1）．これは，過去のドメイン特化 MBD の取り組みを分析したところ，基本的に DSM に用いられるフォーマットはこの 3 つであることが分かったからである[3][4]．例えばデジタル家電向け DSM は，ノードエッジ型の画面遷移図，マトリクス形式の画面遷移表，表形式の決定表を用いている[4]．

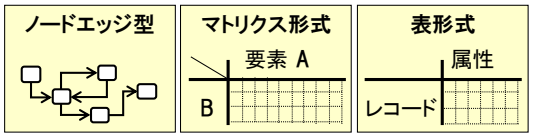

図 1 対象とする DSM のフォーマット

2.5 ドメイン特化 MBD 環境の課題と本研究の目的

従来構築してきたドメイン特化 MBD 環境においては，DSM のフォーマットが変わった場合に変更コストが掛かることが課題であった[3][4]．そこで，本研究では，対象とする DSM のフォーマットが変わった場合にも，最小限のカスタマイズコストで変更可能なドメイン特化環境の構築を目指す．

3. 依存関係抽出技術の方針

3.1 依存関係の把握漏れが引き起こす課題

依存関係の把握漏れが引き起こす課題について説明する．

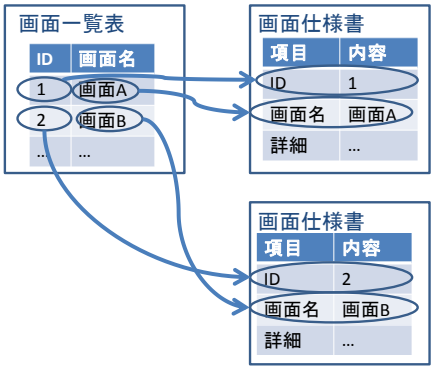

図 2 DSM 間の依存関係の一例

例えば，表形式の DSM として，あるシステムに存在する画面を一覧にした画面一覧表と，個々の画面の仕様の説明を記載した画面仕様書を作成した場合を考える（図 2）．この場合，画面一覧表と画面仕様書に記載される画面 ID と画面名は同一ではなくてはならない．仮にシステムの仕様変更により，ある画面の画面名を変更することになった場合，画面一覧表と画面仕様書の双方で画面名を抜け漏れなく変更しなければならない．

このように，「画面一覧表の画面名の列と，画面設計書に画面名の行では記載されるデータが一致すること」という依存関係が存在する．これは簡単な例であるが，現在のシステム開発では画面数が数千に上るようになっており，DSM 間の依存関係を把握し切れず，設計書の更新漏れが発生する．

3.2 本研究で開発する依存関係抽出技術

本研究で対象とする依存関係抽出技術について説明する．依存関係抽出技術は，DSM に定義されている複数の項目の中で，どの項目とどの項目とが何の依存関係にあるべきかを予め人手で定義することで，その項目に記述されている内容を自動で抽出し，予め定義された依存関係を満たすかを判定するものである．

なお，依存関係には前節で挙げた用語の一致の他に，親子関係（例.ある項目 A は，子要素として A1 と A2 を持つ）や，順序関係（例.あるタスク A は必ずタスク B の後に実施される），CRUD の関係などがあるが，まずは用語の一致関係を対象とし，その他の依存関係の抽出は今後の課題とする．

3.3 依存関係抽出技術の方針

DSM のフォーマットに非依存な依存関係抽出技術を開発するため，次のようなアプローチを取った．

（1）デザイン（フォーマット）とモデル情報の分離
（2）DSM の構造に合わせたメタモデルの構築

（1）について説明する．過去のドメイン特化 MBD の研究[4]においては，DSM をそのままドメイン特化 MBD 環境（ツール群）に入力していたため，ツールの処理が DSM のフォーマットに依存してしまうという課題があった．具体的には，表形式の DSM であれば何列目に何のデータがあるかを決め打ちしたツールの作りにしていたため，DSM のフォーマット変更に対応が困難であった．そこで，今回開発する依存関係抽出機能は，モデル（DSM）はそもそも見た目を表すデザイン（フォーマット情報）と，モデル情報（データ）とに分離できるという点に着目し，デザインを排除した上でモデル情報のみを入力することにした（図 3）．

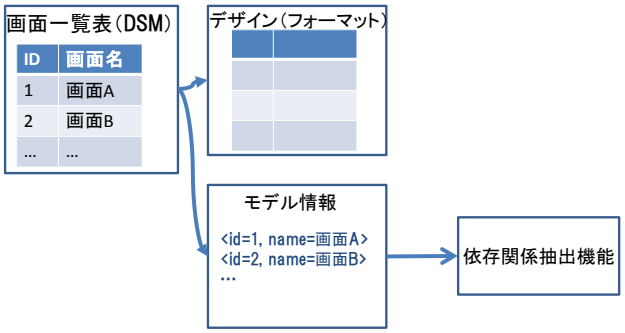

図 3　デザインとモデル情報の分離（画面一覧表の例）

（2）について説明する．依存関係抽出機能のモデル情報の取り扱いを容易にするためには，モデル情報は構造化されていることが望ましい．そこで，メタモデルを作成することにした． 図 4 に画面一覧表と画面仕様書のメタモデルを示す．この例では，表の 1 項目がメタモデルの 1 属性に紐付くことを示している．

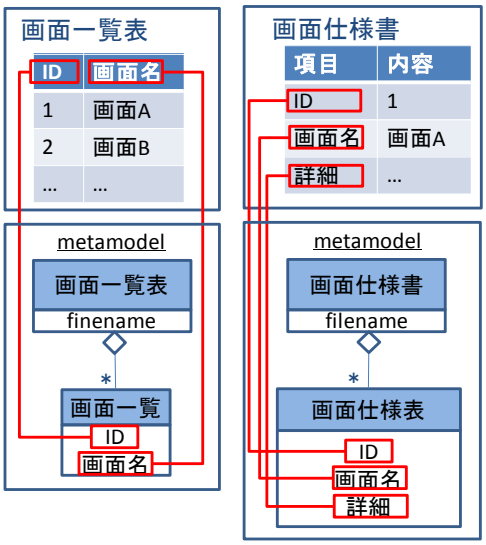

図 4　DSM のフォーマットとメタモデルの関係

（1）（2）を踏まえて，依存関係抽出機能にはモデル情報，メタモデル，抽出ルールを入力する方式を採用する（図 5）．なお，抽出ルールはモデルのどの要素から依存関係を抽出するかを記述する．このような方式にすることで，モデルのフォーマット変更時にはメタモデルを変更することで，依存関係抽出機能は最小限のカスタマイズコストで利用できると考えた．

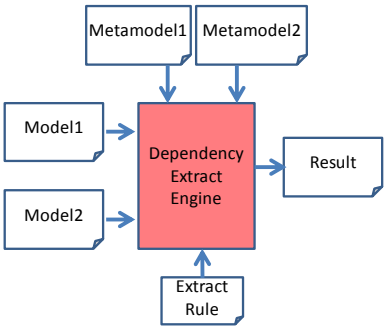

図 5　依存関係抽出機能の設計

なお，モデル情報の定義は XML，メタモデルの定義には XML Schema を用いることとした．その理由は，次の 3 点である．

- XML Schema には拡張の仕組みが備わっており，さまざまなメタモデルの定義が可能
- 既存ツールでモデルの構造チェックが可能
- ともにテキストファイル形式で人手でも機械処理でも扱いやすい

3.4 依存関係抽出ルールの定義方法

次に，依存関係抽出のためのルールの定義方法を検討した．ユーザの依存関係定義の手間を考えると，なるべく定義ルールはシンプルであることが良い．例えば，メタモデルは一つであっても，複数のモデルが作成される場合がある．図 2 に示したような画面一覧表と画面仕様書の例では，画面一覧表は一つであるが，画面仕様書は複数される．この場合，個々のモデルに対して依存関係抽出ルールを定義するのは煩雑である．また，モデルのフォーマットが変わり，モデルに定義する項目が追加された場合などに，依存関係抽出ルールを新たに追加することがあるので，モデルの構造を定義するメタモデルと依存関係抽出ルールは同時に確認できる方式が望ましいと考えた．よって，メタモデル上で依存関係抽出ルールを定義することにした．

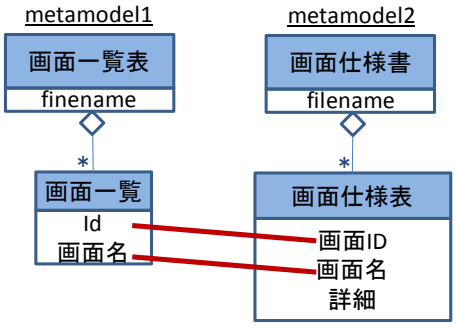

図 6　メタモデル上での依存関係定義

図 6 は，画面仕様書で使用される画面 ID と画面名は必ず画面一覧表で定義されている（＝一致する）というルールを示している．

4. 依存関係抽出ツールプロトの開発

2 章で設計した依存関係抽出のツールプロトを開発した．

4.1 ツールのアーキテクチャ

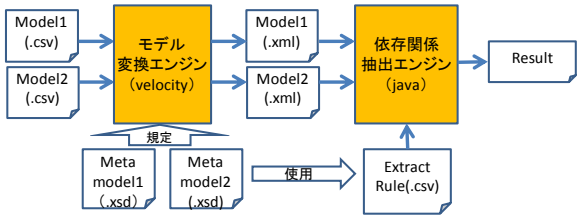

図 7　開発したプロトツールの構成

図 7 に今回開発した依存関係抽出プロトツールの構成を示す．Excel[a]で記述した DSM を csv 形式に変換した後，モデル変換エンジンに入力することで，xml 形式のモデル（モデル情報）に変換する．ここでモデル変換エンジンは，テンプレートエンジンの velocity[b]を用いている．続いて，依存関係抽出エンジンにて，依存関係抽出ルールに基づいて依存関係抽出を行う．このような構成にすることで，メタモデル及び依存関係抽出ルールの入れ替えを行うことで様々な DSM の依存関係抽出が実現できると考えた．

なお，図 6 に示した定義を直接入力するインターフェースの作成は今後の課題とし，それと等価のルールを別途入力することにした．ルールのフォーマットについて次節で説明する．

4.2 依存関係抽出ルールのフォーマット

No	Metamodel1			Metamodel2			Matching Method	
	File Path	Element Path	Regular Expression	File Path	Element Path	Regular Expression	formula1	Formula2
1	Matrix1.xml	//Factor1/Name		Matrix2.xml	//Factor1/Name		M1 eq M2	
2								

図 8　プロトツールに入力する依存関係抽出ルール

図 8 にプロトツールに入力する依存関係抽出ルールを示す．FilePath の欄には各メタモデルのファイルのパスを指定する．パスには「*」「?」のワイルドカードを指定できるため 1 つずつファイル名を指定する必要はない．

ElementPath には，メタモデル内の設計項目を Xpath 形式で指定する．Xpath を採用したため，モデルのフォーマットに関係なくモデル要素の抽出を行うことができる．

Regular Expression はマッチングさせる設計項目の対象文字列に余計な接頭/接尾辞をがある場合に用いる．例えば一方は「担当者（不特定）」と書かれ，もう一方では「担当者（任意）」という場合に，() の中はマッチングの対象外とする場合などに正規表現を用いる．

Matching Method の formula1 の欄には eq(一致すること)の他に neq(一致しないこと)が記述可能である．Formula2 の欄には，AND，OR を記載可能で，No1 と No2 に書いた条件の間で AND 条件，OR 条件を取ることができる．

5. 適用試行

5.1 組込み製品への適用

今回開発した依存関係抽出技術を，ある組込み製品の DSM に対して適用した．対象の組込み製品は，ユーザがシステムの UI 上のボタン（キー）を押すことで，システムの機能状態が切り替わる．同製品開発においては，システムの機能状態ごとに各キーが有効であるか，有効であればどの機能状態に遷移するかを記載したキーマトリクスとい

a Microsoft 社の登録商標である．

b Apache Software Foundation の登録商標である．

う DSM を作成している．また，キーマトリクスには 2 種類存在する．一つはシステム全体の動作を記載した全体キーマトリクス，もう一つはシステムの機能状態ごとに更に詳細化した情報を記載した機能別キーマトリクスである．図 9 にキーマトリクスの概要を示す．全体キーマトリクスには，各キーを押した時の正常系の動作が記載される．しかし，例えば電話機能であっても，サブ状態(例えば電話機能のサブ状態には発信と受信などがある)によっては，同一のキーであっても無効な場合がある．このように機能別キーマトリクスには，各機能のサブ状態によってキーが有効/無効であるかを記載する．なお，キーが有効であれば，基本的に全体キーマトリクスに書かれたものと同じ動作（例: 電話に属するサブ状態で Key1 を押せば機能 A に遷移する）をすることになる．

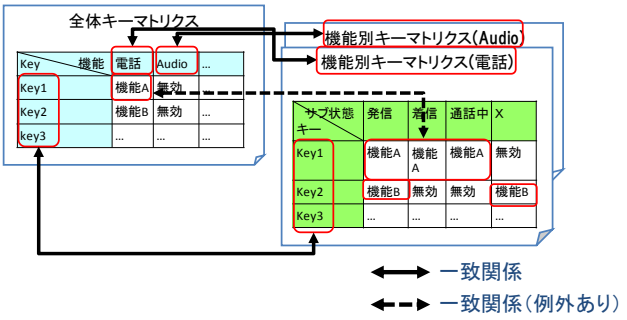

図 9　ある組込み製品のキーマトリクス

5.2 キーマトリクスに存在する依存関係

前述のキーマトリクスに存在する依存関係は次のとおりである．

（1）　全体キーマトリクスと機能別キーマトリクスではキー名称が同一

（2）　全体キーマトリクスのある列（機能）に対応する機能別キーマトリクスでは，基本的に同一のキー名称に対する動作は同一

（1）については，キーマトリクスで使用するキー名称はすべて同じであるという規則がある．ここでこの組込み製品は派生開発を行っており，ある機種のキー名称は前機種からの変更や追加によって決まる．しかし，機能別キーマトリクスは各機能の担当者が作成しているために，担当者によってはキー名称を誤用するという課題がある．

（2）については，全体キーマトリクスには機能状態における正常動作が，機能別キーマトリクスには機能状態のサブ状態における全動作が記載されることから，機能別キーマトリクスでは各機能状態のサブ状態でキーが有効であれば全体キーマトリクスと同じ動作が書かれ，あるサブ状態でキーが無効であれば無効と書かれるはずである．しかし，機能別キーマトリクス作成者が誤って記載したり，機能別キーマトリクスを見てソフトを実装する実装者が，無効な箇所を見落として実装をしたりして，後工程で不具合が発覚するケースが見られていた．

5.3 依存関係の定義

前節で説明した（1）（2）に対する依存関係抽出ルールを作成する．（1）に対するルールを図 10 を用いて説明する．同図右側に示した全体/機能別キーマトリクスのメタモデルにおいて，キー名称が一致することをルールとして定義すれば良い．実際には，図 8 で示したフォーマットを用いてルールの定義を行う．

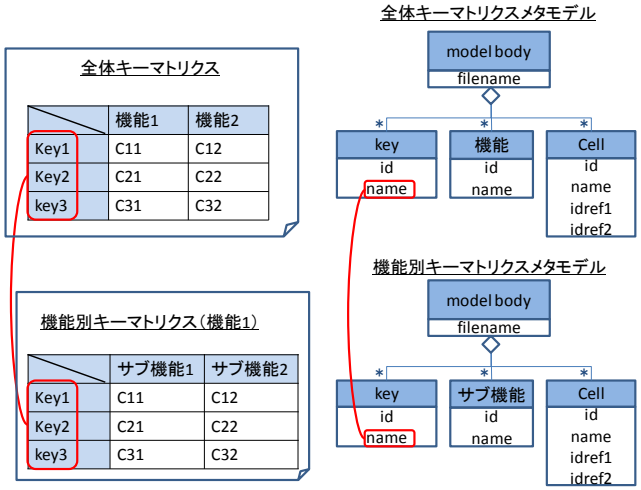

図 10　（1）に対する依存関係抽出ルール定義

次に（2）に対するルールを説明する．図 11 に示すように，次の 3 つの条件を定義する．

（条件 1）全体キーマトリクスの機能名(機能要素の name 属性)と機能別キーマトリクスのファイル名(modelbody 要素の filename 属性．但し正規表現を使用．)

（条件 2）全体キーマトリクスのキー名称と機能別キーマトリクスのキー名称(双方の key 要素の name 属性)

（条件 3）全体キーマトリクスの機能名(機能要素の name 属性)と機能別キーマトリクスのファイル名(modelbody 要素の filename 属性)

また，条件 1 かつ条件 2 が成り立つ場合，条件 3 が成り立つというルールを作成する．式で記述すると，

If(条件 1 and 条件 2)then(条件 3)　・・・(＊)

となる．ここで，図 8 で示したフォーマットでは，if 文は現時点ではカバーしていない．そこで，今回は図 7 に示した依存関係抽出ツールで依存関係抽出エンジンに別途インターフェースを設け，(＊) で示した条件を入力できるようカスタマイズを行い，動作検証を行った．現時点では，(2) の依存関係に特化した処理を行うようになっているが，今

後汎用的に if 文を取り扱えるように，図 8 で示したフォーマットの改良と，依存関係抽出エンジンの拡張を進める予定である．

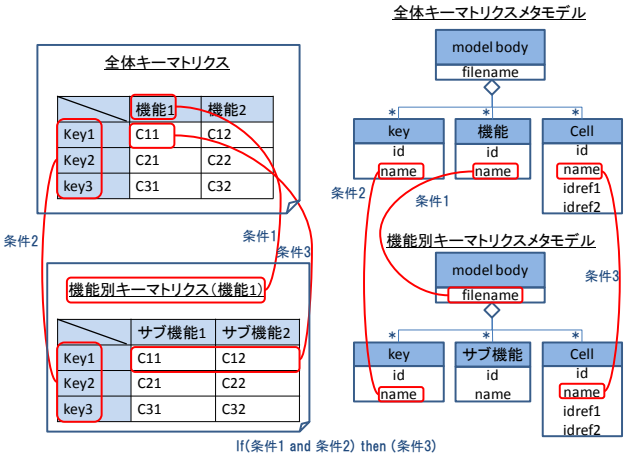

図 11 （2）に対する依存関係抽出ルール定義

5.4 依存関係抽出結果

分類	id	item1	id	item2
一致	1	Key1	1	Key1
	2	key2	2	key2
Model1にのみ存在	3	key3	NULL	NULL
	4	key4	NULL	NULL
Model2にのみ存在	NULL	NULL	4	key4’

図 12 （1）に対する依存関係抽出結果

図 12 に（1）のキー名称の依存関係出力結果を示す．出力結果には，次の 3 種類の情報を出力する．

(a) モデル 1 とモデル 2 の両方に存在するキー名称
(b) モデル 1 にのみ存在するキー名称
(c) モデル 2 にのみ存在するキー名称

(a)は，マッチングが成功し一致する文言を抽出できた例である．(b)(c)は，マッチング失敗し一致する文言を抽出できなかった例である．この場合，どちらか一方のモデルにのみ文言が存在することになる．

EventId/StatusId	S1	S2	S3	S8	S9	S10
E1	完全一致	完全一致	完全一致	完全一致	完全一致	完全一致
E10	一致しない	一致しない	一致しない	一致しない	一致しない	一致しない
E22	完全一致	完全一致	完全一致	完全一致	完全一致	完全一致
E25	完全一致	完全一致	完全一致	完全一致	完全一致	完全一致
E28	完全一致	完全一致	完全一致	完全一致	完全一致	完全一致
E31	完全一致	完全一致	完全一致	完全一致	完全一致	完全一致
E34	完全一致	完全一致	完全一致	完全一致	完全一致	完全一致
E37	完全一致	完全一致	完全一致	完全一致	完全一致	完全一致
E40	完全一致	完全一致	完全一致	完全一致	完全一致	完全一致
E43	完全一致	完全一致	完全一致	完全一致	完全一致	完全一致
E46	完全一致	完全一致	完全一致	完全一致	完全一致	完全一致
E49	完全一致	完全一致	完全一致	完全一致	完全一致	完全一致

図 13 （2）に対する依存関係抽出結果

次に図 11 で示した依存関係の抽出結果を図 13 に示す．依存関係抽出エンジンは，基本的には図 12 で示した形式で全体/機能別キーマトリクスの各セルの対する文言一致の検証結果を出力するが，ユーザが結果を見やすいように結果のフォーマットのカスタマイズを行い，機能別マトリクスに合わせたマトリクス状の形式で，各セルの文言が一致するか否かを出力する．

6. 結果及び考察

6.1 結果

4 章で説明した組込み製品の全体/機能別キーマトリクス間の依存関係抽出を，プロトツールを用いて実施し，4.2 節で説明した（1）（2）双方の条件について，正しく文言一致の関係を抽出，検証できていることを確かめた．

次に，マトリクス型以外の DSM に対して，依存関係抽出技術が適用可能であるかを検証した．これには，エンタープライズ系のシステム開発で実際に用いているノードエッジ型及び表形式の DSM を用いた．具体的には，ノードエッジ型の DSM として，業務の担当者と業務内容を記載した業務フロー図，表形式の DSM として，対象システムのデータ項目一覧など計 4 種類を対象とした．それぞれの DSM のメタモデル構築と，メタモデル上での依存関係定義，及び図 8 のフォーマットでのルール記述を行い，DSM 中の項目間の文言一致に関する依存関係抽出ができることを確かめた．ここで確かめたのは，各 DSM に記載される業務の担当者名や，エンタープライズシステムの機能名などの文言一致の依存関係である。

但し，ノードエッジ型の業務フロー図はノード，エッジなどの各オブジェクトに対応する XML の要素名のテーブルを用意することで，XML へのモデル変換を実施した．この時，モデル変換エンジンと依存関係抽出エンジンに変更は必要なく，メタモデルと依存関係抽出ルール及びモデル変換ルールを書き替えのみを行えば良いことを確かめた．

以上から，マトリクス形式に加えて，表形式，ノードエッジ型の DSM に対して，提案の依存関係抽出技術は，メタモデルとルールを書き替えることでエンジンへの変更を行わずに適用可能である．また，組込み系に限らず，エンタープライズ系のドメインの DSM への適用が可能であることが確かめられた．

6.2 考察

提案手法を用いることで，ひとたびメタモデル上でルール定義を行えば，DSM 中の文言一致の依存関係が抽出，検証

が可能となる．これにより，従来ソフトウェア開発で発生していた DSM 間の文言の不整合に起因する不具合を抑止することが出来る．更に，提案手法をソフトウェア開発の各工程で用いることで, DSM 間のトレーサビリティ確保が可能である．但し，文言が一致する場合という条件に限られる．また，提案手法はメタモデル上で依存関係抽出ルールを定義する方式であるため，例えある同一名称の DSM が複数ある場合（例えば，画面仕様書という DSM が製品の画面の数だけ存在する場合など）であっても，ルール定義はメタモデルに対して 1 回行えば，それら全ての DSM に対する依存関係抽出が可能である． また，デザインとモデル情報の分離と，メタモデル上での依存関係抽出ルールの定義を行ったことで，モデルのフォーマット変更があった場合には，メタモデルの変更と依存関係抽出ルールの再定義を行うことで，依存関係抽出機能は継続的に利用可能である．これは，XML によるモデル情報の記述と， Xpath を用いた依存関係抽出機能の実装を行っているためである。

なお, ここで言うフォーマット変更とは, DSM の形式の変更（状態遷移図を状態遷移表に変更するなど）がある場合と, 形式は同一であるが, 項目が追加/削除される場合（同じ表形式の DSM であるが，列が追加されるなど）の両方を含む.

6.3 関連技術

本研究と同様に，メタモデルを用いた DSM(設計書)の記述内容の検証を行うものとして，SpecPrince があげられる[7]．これは設計書の各項目の記述が予め規定したメタモデルに則っているかのチェック（シンタックスチェック）を行うものである．また，文献[8]では，個々の設計書のメタモデルを作製することで，設計書に記述される項目を設計者間で確認し，設計レビューの観点を明確化する手法が提案されている．しかし，双方とも本研究のように複数の DSM 間での項目の依存関係を抽出する技術ではない．

次に，メタモデルを用いたモデル間の依存関係抽出技術を述べたものとして文献[9]がある．これは, Ecore 上にて, DSM 間の依存関係を表した新たな DSM を別途作成することで，依存関係を抽出するものである．一方本研究では，依存関係抽出の対象となる DSM のメタモデル上で依存関係抽出ルールを定義するアプローチを取っている点が異なる．文献[10]は，UML を対象にメタモデルを用いた仕様間の依存関係追跡技術を提案している．本研究で提案した手法は，現時点では UML は対象外としているが，技術的には UML も対象に含めることが可能と考えている．

また，汎用用途のドメイン特化モデリングのツールとして MetaEdit[11]があり，本研究と同様に，メタモデルの部品間のトレーサビリティを取ることが可能である．

7. まとめと今後の課題

7.1 まとめ

本研究では，ドメイン特化 MBD 環境は対象とする DSM のフォーマットが変わると大規模な変更が必要になるという課題に対して，DSM 間の依存関係抽出を題材に，DSM のフォーマットに非依存なドメイン特化 MBD 環境の構築に取り組んだ．提案手法により，DSM 間のシンタックスレベル（文言一致関係）の依存関係抽出が可能となった．

また，プロトツールを作成し，組込み製品への適用試行を行い，提案手法により依存関係抽出が行えることを確かめた．提案手法においては，

・モデルのフォーマットとモデル情報との分離
・メタモデル上での依存関係抽出ルールの定義

を行う．また，マトリクス形式，表形式及びノードエッジ型（但し，Excel で記述したもの）の DSM からの依存関係抽出を実施可能であることを示した．現時点では，上記 3 種類の記法の DSM に対して，対象とする事業ドメインに限らず依存関係抽出技術が利用可能であることを，組込系及びエンタープライズ系の製品開発で実際に用いられている DSM を用いて確かめた．

7.2 今後の課題

今後の課題は次の 3 点である．

（1）提案手法の詳細な評価
（2）依存関係抽出機能の拡張，応用
（3）ドメイン特化 MBD 環境としての機能の拡張

（1）については，モデルのフォーマットを変えた時の依存関係抽出機能の変更コストが過去の取り組みに比べてどれだけ低減されるかの評価を定量的に行い，提案手法の有効性を検証したい．また，今回試行した以外のシステムの DSM に対しても, 開発した依存関係抽出ツールの適用試行を行い, 適宜依存関係抽出ルールやエンジンの改善を図る．
（2）については，2 つの方向性が考えられる．1 つは，取り扱い可能な DSM のフォーマットの拡大である．現時点では表形式，マトリクス形式，ノードエッジ型を取り扱い可能であるが，例えばシーケンス図やタイミング図などの形式には対応していない．これらにも依存関係抽出技術を適用できるよう, 拡張を進めていく必要がある. 2 つ目は，取り扱える依存関係の拡大である．提案手法では，現在は 2 種類の DSM 間の項目（メタモデル上の要素，属性）同士の文言一致に関する依存関係抽出のみを行っているが，その他に項目間の親子関係や CRUD 関係，なども取り扱えるようにしていきたい．そのためには，シンタックスだけで

Vol.2014-SE-184 No.9
Vol.2014-EMB-33 No.9
2014/5/20

なく，セマンティクスも含めた取り扱いが必要になると考えている．

（３）については，現在開発したのは仕様間の依存関係抽出機能に関するものであるが，今後ドメイン特化 MBD 環境として必要な機能を順次拡張してゆく必要がある．例えばコード生成やテスト生成機能などが上げられる．また，それらの機能の開発に関しては，本報告で述べたとおり，極力モデル（DSM）のフォーマットや DSM の種類に非依存に利用できるような構成を検討する必要がある．

参考文献

1) Frankel, S.，日本アイ・ビー・エム TEC-J MDA 分科会 ： MDA モデル駆動アーキテクチャ，エスアイビーアクセス(2003)
2) Mernik, M., Heering, J. Sloane, A. : When and How to Develop Domain-Specific Languages, ACM Computing Surveys, Vol. 37, No. 4, December 2005, pp. 316-344, (2005).
3) 村田大二郎，団野博文，三部良太：業種特化型設計開発構築基盤の提案，情報処理学会研究報告. EMB， 組込みシステム，2008(55)，pp107-114(2008).
4) 川上真澄，小川秀人：デジタル家電ドメインに特化したモデルベース開発環境，情報処理学会論文誌，Vol.52，No.12，pp.3184-3191(2011).
5) OMG： UML2.0, http://www.omg.org/spec/UML/2.0/ (2005).
6) IPA/SEC 編：組込みソフトウェア開発における品質向上の勧め［設計モデリング編］，アイティメディア（2006）．
7) 位野木万里，松尾尚典，甲田修策：メタモデルに基づき仕様書作成と仕様検証を支援するツール SpecPrince，東芝レビュー 63(12), pp46-49(2008)
8) 元山厚，中谷多哉子：設計仕様のメタモデルに基づくソフトウェアレビュー方法の提案，ソフトウェア品質シンポジウム 2011(SQiP2011).
9) A. Qamar, S. Herzig, C.J.J Paredis : A Domain-Specific Language for Dependency Management in Model-Based Systems Engineering, 7th International Workshop on Multi-Paradigm Modeling(MPM13), 2013
10) 大平直宏，松下誠，岡野浩三，楠本真二，井上克郎，山下裕介，我妻智之:Struts フレームワークにおけるメタモデルを用いた追跡可能性実現手法の提案，情報処理学会研究報告．ソフトウェア工学研究会報告 2005(119), 33-40(2005)
11) MetaCase : MetaEdit+, http://www.metacase.com/http://www.fuji-setsu.co.jp/index.html