Model Driven Systems Engineeringにおいて，要求文書から分析・設計へとつなぐ形式化のためのデータベース設計

松村　郁生† 　佐藤　洋介†† 　福田　　進† 　松澤　裕史† 　宮下　　尚†

† 日本アイ・ビー・エム株式会社　〒 242–8502 大和市下鶴間 1623–14
†† 株式会社デンソー　〒 448–8661 愛知県刈谷市昭和町 1–1
E-mail: †{ikuo,fukudas,matuzawa,himi}@jp.ibm.com, ††YOUSUKE_SATO@denso.co.jp

あらまし　モデル駆動型システムズエンジニアリングでは製品の品質と生産性を向上させるために，開発の早い段階から要求を適切に形式化し，検証を行う．しかし要求を形式化する際には，分析者の様々な暗黙的思考により要求文書と形式化されたシステムモデルとの関連性は複雑なものとなるため，この関連性を分析結果から適切に突き止めることは容易ではない．本稿ではこの問題に対処するため，特にシステムの構造的側面に関して，要求文章の分析過程とシステムモデルとの対応関係を表すデータモデルと，モデル要素の推移的な依存関係を考慮したトレース手法を提案する．また要求文書の形式化とその上での追跡を例に、データモデルとトレース手法の有効性について述べる．

キーワード　データモデル，Model Driven Systems Engineering, 要求の形式化, トレーサビリティ

Database Design for Bridging Requirement Documents to Analysis and Design in Model Driven Systems Engineering

Ikuo MATSUMURA†, Yosuke SATO††, Susumu FUKUDA†, Hirofumi MATSUZAWA†, and Hisashi MIYASHITA†

† IBM Japan　Shimotsuruma 1623–14, Yamato-shi, Kanagawa, 242–8502 Japan
†† DENSO Corporation　Showa-cho 1–1, Kariya-shi, Aichi, 448–8661 Japan
E-mail: †{ikuo,fukudas,matuzawa,himi}@jp.ibm.com, ††YOUSUKE_SATO@denso.co.jp

Abstract　In Model Driven Systems Engineering, in order to improve quality and productivity of development, formalizing and verifying requirements are to be performed in the early stage of engineering. However, it is difficult to trace relationships among requirement documents and formalized system models that gives rationale of analysis results , because the relationships tend to be complex since various implicit thinking deeply affects analysis of requirements. To address this problem, we propose 1) a datamodel representing analysis process of requirement documents and mappings to system models and 2) tracing method that effectively tracks transitive relation of model elements. And then we exemplify the effectiveness of the proposed datamodel and trace method.

Key words　datamodel，Model Driven Systems Engineering, traceability, formalization of requirements

1. はじめに

組込みシステムの開発においては，システムの高度化に伴いソフトウェアの複雑さが増大し，開発がソフトウェア，エレクトロニクス，メカニクスという異なる領域に跨ること，またグローバル化・分業化されることにより，ステークホルダおよび開発者間の円滑な意思伝達が困難になっている．特に，現在の複雑化したシステムではソフトウェアによって機能が実現される割合が増大しているため，複雑化・大規模化が顕著なものとなっている [1]．ソフトウェアの複雑化・大規模化は欠陥の生じやすさと変更コストの増大を招き，意思伝達の不足は仕様の曖昧性と問題発見の遅れを招くため，製品の品質と生産性を大きく低下させる要因となる．

このような組み込みシステムの品質と生産性を向上させるために，曖昧性がなく，検証が容易である仕様を作成することができる手法が求められている．モデル駆動型システムズエンジニアリング (MDSE: Model Driven Systems Engineering) では，このような目的を達するために，要求文書の段階から UML

や SysML などのモデリング言語を利用して，要求や仕様の適切な形式化を施し，開発の早い段階で検証を可能とする．これによって，領域間・組織間の意思伝達や検証，シミュレーションを行うことで，仕様の曖昧性を取り除き，不具合の早期発見を行い，製品の品質と生産性の向上を達成することを目指している．

我々は，このような要求のモデリングに基づく形式化および分析を Total Trial Sheet (TTS) と呼ばれる表を用いて行う．TTS は，要求の原文を表の項目に適切に展開することによって，UML[2][3] や SysML[4] といった形式検証・実行が可能となるモデルへとつなげる役割を果たす．これによって要求分析者は，モデル言語に対する知識を過度に必要とすることなく，あいまい性および属人性ができるだけ排除された，要求の形式化を行うことを可能としている．さらに，元の要求が検証が可能なモデルへと表形式によって関連付けられているため，分析時にどの要求から問題が起きているかを突き止めるためのトレーサビリティに優れている．

しかし，このような要求文書の分析には，人間の様々な思考による操作が必要となり，要求文書と，検証可能である形式化されたモデルとの関連性は非常に複雑なものとなる．例として，図 1 に，実際に TTS を用いて分析された要求文書と，SysML における BDD(Block Definition Diagram) との関連を示す．ただし図の例に表れる「A#P」「A#[C]#B」の表記はそれぞれ「A に P というプロパティがある」「A が B から構成されている (A Composed-of B)」を意味する．

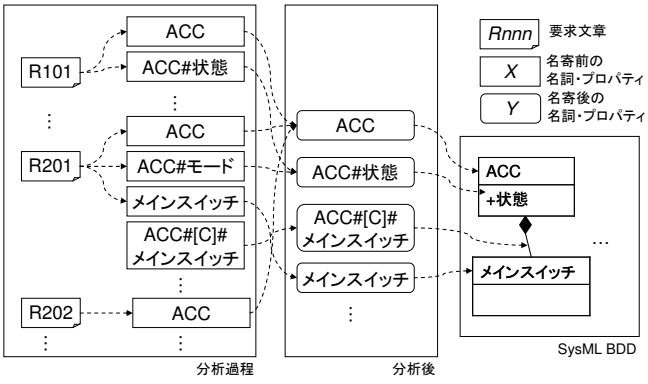

図 1　要求文章とシステムモデルの関係の例

この例においては，ACC が属性「状態」を持ち「メインスイッチ」を集約しているが，実際の Block である ACC は要求文書の R101 および R201 および R202 に由来しており，さらに，ACC の「状態」という属性は「状態」および「モード」から名寄せが行われている．一般に，分析者によって暗黙的に想定されたこのような関連性を，分析結果から適切に突き止めることは容易ではない．

我々はこの問題を解決するために，要求文書の形式化を支援するためのデータモデルである TTS-Schema を導入した．このデータモデルは，要求文書を段階的に形式化してゆく手続きに沿ったトレーサビリティを実現するためのデータモデルである．TTS-Schema は最終的には他の分析や検証のためのモデルへの変換および対応付けが可能であるように設計されており，3 つの情報：a) 原文の展開から分析結果に至る作業経過，b) 分析結果とターゲットモデルとの対応関係，c) 一般化されたシステムの側面を持つ．開発者は TTS-Schema に対応する表を埋めることで，表記の揺れや曖昧な表現を取り除き，暗黙的に示唆されている要素でモデルには記述が必要な情報を知ることができる他，TTS-Schema のデータモデルを用いることで，要求文書とモデルとの関連の，その形式化過程を含めた追跡（トレース）が可能になる．

以上のような目標について関連する研究としては，高い安全性が求められるシステム開発において用いられる SpecTRM[5] が挙げられる．SpecTRM では，システムの目的や動作原理といった意図 (Intent) を明確にすることで要求と分析・検証可能なモデルとの関連性を取り扱うことができるが，要求文章を段階的に分析し，形式化し，関連付けるための手法については研究されていない．

2 章以降の構成は以下の通りである．まず 2 章で TTS を用いた段階的な要求文書の形式化と，本稿で対象とするトレーサビリティについて述べ，3 章では段階的な要求の形式化とトレーサビリティを実現するデータモデルについて述べる．4 章ではデータモデルとその上でのトレーサビリティの実装について述べ，5 章で結論を述べる．

2. 段階的な要求文書の形式化とトレーサビリティ

要求文書の形式化を支援する表形式の表現である TTS は，システムの側面ごとに要求文書を段階的に形式化する作業をサポートするいくつかの表から構成されている．TTS による手法では，システムの側面として構造，振舞い，制約を扱う．本稿ではこのうちシステムの構造的側面について述べる．システムの構造的側面を形式化する作業の概要は次の通りである：

（1） 要求文書を適切な単位（原文）に区切る
（2） 原文単位の要求文章を表の項目に展開する
（3） 名寄せをする
（4） UML や SysML などのシステムのモデル（以下ターゲットモデルという）に変換する

(2) で用いる表を図 2 に示す.「原文」カラムは要求文書を，構造を分析するために適切な単位で区切った文章,「補完後原文」カラムは原文で省略されている主語や指示代名詞を補った文章を記載する．ブロック，プロパティ，関連，対象ブロックの 4 つのカラムは，原文あるいは補完後原文から抽出したオブジェクトとそのオブジェクトに関わる属性と他のオブジェクトとの関連である．(3) で行う名寄せとは，オブジェクトとその属性・関連に関する同一性を確認する作業である．(4) では名寄せ済みの TTS を分析・検証で利用される UML や SysML などのターゲットモデルへと変換する．

このように要求文書が段階的に形式化されてターゲットモデルへと変換されるが，製品の品質と開発の生産性を高めるためには要求文書とターゲットモデルの対応関係（トレーサビリティ）が明確であることが重要である．具体的には次の 2 種類

ID	原文	補完後原文	ID	ブロック	プロパティ	関連	ブロック
1	メインスイッチがON状態で･Aがおこると Bする。デフォルトの車間距離は･･･。	ACCのメインスイッチがON状態で･車車のAがおこると Bする。デフォルトの車間距離は･･･。	O0101	ACC		C	メインスイッチ
			O0102	メインスイッチ	状態		
			･･･	･･･	･･･	･･･	･･･
2	･･･	･･･	･･･		･･･		･･･

図 2　構造 TTS の例

がある：

1) 前向きトレーサビリティ：要求を実現するモデル要素をトレースするもので，仕様変更時の影響度分析の際に重要である．顧客や OEM からの要求に変更があった場合，変更された要求に対応するターゲットモデルの要素をトレースすることで，その変更が後段の開発プロセスにどれくらい影響を与えるかを見積もり，顧客との意思伝達に利用することができる．

2) 後向きトレーサビリティ：モデル要素が実現する要求をトレースするもので，不具合の報告や代替案の検討の際に重要である．製品の構成要素の一部に不具合が見つかった際に，不具合を持つターゲットモデル要素に対応する要求をトレースすることで，顧客にとってどの機能に問題が発生しているかを明らかにすることができる．また部品調達や技術的課題などの新たな問題が発生した場合，実現が困難となったターゲットモデル要素に対応する要求をトレースすることで，その要求を満たすような設計上の代替案や要求そのものの代替案の検討に利用することができる．

以降では，システムの構造的側面を対象に，このような要求の段階的な形式化と前向き・後向きのトレーサビリティを実現するデータモデルについて述べる．

3. 要求文書の形式化とトレーサビリティのためのデータモデル

段階的な要求の形式化とトレーサビリティ（追跡可能性）を実現するためのデータモデルには，次の 3 つの性質が必要である．

a. 分析過程のトレーサビリティ：要求の段階的な分析（名寄せなど）の過程がトレースできること．

b. ターゲットモデルとのトレーサビリティ：形式化された要求と変換されたターゲットモデル（SysML BDD など）が相互にトレースできること．

c. ターゲットモデルに対する一般性：データモデルが対象とするシステム側面（構造・振る舞い・制約など）に対して，異なる種類のターゲットモデル (UML や SysML) への変換を許容する表現となっていること．

システムの構造的側面に関して上の 3 つの性質を満たすために 3 つの情報：a. 原文の展開から分析結果に至る作業経過 (StructuralElement)，b. 分析結果とターゲットモデルとの対応関係 (TargetModelMapping)，c. 一般化されたシステムの構造的側面 (CStructuralElement) を持つデータモデルを設計した．図 3 に TTS-Schema でのデータモデルの概要を示す．

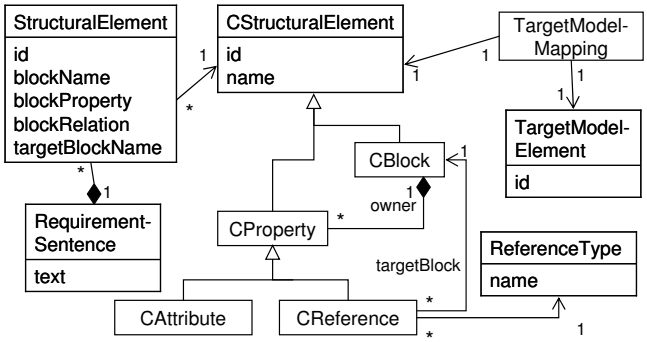

図 3　TTS-Schema のデータモデル（概要）

原文を表す RequirementSentence は複数の StructuralElement を持っている．StructuralElement は原文をブロック名 (blockName)，属性名 (blockProperty)，関連タイプ (blockRelation)，対象ブロック (targetBlockName) の 4 つの列に展開した際の 1 つの行を表している．

CStructuralElement は StructuralElement が名寄せされた後の要素を表す (接頭辞の C は Canonical の意である)．名寄せ後は，名寄せによって識別されたシステムの構造的側面（オブジェクトとその属性およびオブジェクト間の関連）を後述（3.2 節）の CBlock, CProperty, CAttribute, CReference, ReferenceType というクラスによって表現する．

TargetModelMapping は CStructuralElement と TargetModelElement との対応関係を表す．

3.1 データモデル上のトレーサビリティ

複雑なシステムにおいては要求や形式化されたモデルも大規模なものとなる．このため本稿では，大規模データの効率的なクエリに適する関係データベース (RDB) の上でのトレース手法について説明する．

表 1 に図 3 の TTS データモデルを RDB スキーマとして表現したものを示す．関係名と TTS データモデル要素名との対応は表 1 の通りである．CStructuralElement とそのサブクラスは Single Table Inheritance[6] により関係 CSE に対応付けてある．全ての関係は下線表記のキー属性を持ち，関係 SE, CSE, TMM は斜体表記の外部キー属性を持つ．外部キー属性の名前はキーの参照先の関係名を接頭辞に持っている．例えば SE の属性 rs_id は関係 RS への外部キーである．但し CSE の owner_id 属性，tb_id 属性はそれぞれにおける owner，targetBlock 関連を表す，CSE 自身への外部キーである．

RS[rs_id, type]

SE[se_id, *rs_id*, *cse_id*, ...]

CSE[cse_id, type, name, *owner_id*, *tb_id*, *rt_id*]

RT[rt_id, name]

TMM[tmm_id, *cse_id*, *tme_id*]

図 4　TTS データモデルの RDB スキーマ表現

関係名	データモデル要素名
RS	RequirementSentence
SE	StructuralElement
CSE	CStructuralElement, CBlock, CProperty, CAttribute, CReference
RT	ReferenceType
TMM	TargetModelMapping
TME	TargetModelElement

表 1　TTS のデータモデル要素名と関係名の対応

SE (StructuralElement) および CSE (CStructuralElement) を仲介することによって TTS 展開と名寄せに関する形式化過程をトレースすることが可能であり，TMM (TargetModelMapping) をたどることによってターゲットモデルとの対応付けをトレースすることが可能である．表 1 の RDB スキーマを用いると，前向きトレーサビリティおよび後向きトレーサビリティは以降に示す演算により実現できる．但し記号 $\bowtie$ は自然結合で，結合に用いられる属性名は被演算子で共通する属性名である．例えば式 (1) の RS $\bowtie$ SE で用いられる属性名は rs_id である．

$$\mathrm{RS\_CSE} = \mathrm{RS} \bowtie \mathrm{SE} \bowtie \mathrm{CSE} \tag{1}$$

$$\mathrm{CSE\_TME} = \mathrm{CSE} \bowtie \mathrm{TMM} \bowtie \mathrm{TME} \tag{2}$$

$$\mathrm{RS\_CSE_{+}} = \mathrm{RS\_CSE} \cup \underset{(\text{except c\_id})}{\pi} \Big(\underset{\mathrm{c\_id/cse\_id}}{\rho} (\mathrm{RS\_CSE}) \underset{\substack{\mathrm{owner\_id} \\ =\mathrm{cse\_id}}}{\bowtie} \mathrm{CSE} \Big) \tag{3}$$

$$\mathrm{R_D} = \underset{\mathrm{s\_id,t\_id}}{\pi} \Big(\underset{\mathrm{s\_id/cse\_id}}{\rho} (\mathrm{CSE}) \underset{\mathrm{s\_id=owner\_id}}{\bowtie} \sigma_{\mathrm{rt} \in \mathrm{D}} (\mathrm{CSE}) \underset{\mathrm{tb\_id=t\_id}}{\bowtie} \underset{\mathrm{t\_id/cse\_id}}{\rho} (\mathrm{CSE}) \Big) \tag{4}$$

$$\mathrm{RS\_R_D} = \underset{\mathrm{t\_id/cse\_id}}{\rho} (\mathrm{RS\_CSE_{+}}) \Big(\underset{\mathrm{t\_id=s\_id}}{\bowtie} R_D \Big)^{*} \tag{5}$$

$$\mathrm{RS\_TME} = \mathrm{RS\_R_D} \underset{\mathrm{t\_id=cse\_id}}{\bowtie} \mathrm{CSE\_TME} \tag{6}$$

$$\mathrm{FTrace} = \sigma_{\mathrm{rs\_id}=\alpha} (\mathrm{RS\_TME}) \tag{7}$$

$$\mathrm{BTrace} = \sigma_{\mathrm{tme\_id}=\beta} (\mathrm{RS\_TME}) \tag{8}$$

計算の大まかな流れは，まず式 (1) で RS と CSE を SE を仲介して結合することで RS_CSE を得，式 (2) で CSE と TME を TMM を仲介して結合することで RS_TME を得る．RS_CSE と RS_TME を属性 cse_id で結合することによって，要求文章 (RS) とターゲットモデル要素 (TME) の単純な関係を計算することができる．しかしターゲットモデル要素同士は対象システムにおいてしばしば他のモデル要素に依存しているため，このような依存関係を考慮する必要がある．本稿では，依存関係の種類を表す集合 D (D $\subset$ RT) のいずれかを referenceType としてもつ CReference オブジェクトを考え，この CReference オブジェクトの始点の CBlock が終点の CBlock に「依存している」と定義し(注1)，後述する式 (3)(4)(5) の演算により，RS_CSE を RS_CSE からモデル要素の依存関係を推移的にたどることのできるモデル要素で拡張した RS_$\mathrm{R_D}$ を得る．式 (6) で RS_$\mathrm{R_D}$ と CSE_TME を結合することで，全てのトレース情報を持つ RS_TME が得られる．以降では式 (3)(4)(5) の演算を順に説明した後，式 (7)(8) の前向きトレース，後向きトレースの演算について述べる．

式 (3) の $\mathrm{RS\_CSE_{+}}$ は，RS_CSE に対して，図 3 における owner 関連（CProperty→CBlock）によってたどれる親ブロックの CSE を追加したものである．具体的な計算は，RS_CSE の属性 cse_id を c_id に名前変更してから CSE と owner 関係で結合し，その結果を RS_CSE と型適合とするために c_id 以外の属性で射影 $\left(\underset{(\text{except c\_id})}{\pi} \right)$ してから RS_CSE との和をとる．

式 (4) の $\mathrm{R_D}$ は，依存関係の種類を定義した上述の D に基づく CBlock 同士の依存関係を表す 2 項関係である．具体的な計算は，1) 始点 CBlock として，CSE の cse_id を s_id と名前変更したもの，2) 始点の CBlock がもつ CReference として，referenceType が D のいずれかと一致する CSE，3) 終点の CBlock として，CSE の cse_id を t_id と名前変更したものの 3 つの関係を owner 関連と targetBlock 関連に基づいて結合し，その結果を，始点と終点の ID を表す s_id,t_id で射影する．

式 (5) の RS_$\mathrm{R_D}$ は，RS と，owner 関係および依存関係 $\mathrm{R_D}$ の観点で RS から辿れる全ての CSE との関係である．具体的な計算は，$\mathrm{RS\_CSE_{+}}$ の属性 cse_id を t_id に名前変更し，依存関係における推移閉包を計算する．

(7) 式の前向きトレースは RS_TME を特定の rs_id = α に関して選択することで，(8) 式の後向きトレースは RS_TME を特定の tme_id = β に関して選択することでそれぞれ得ることができる．

3.2　データモデルの構造的側面

システムの構造的側面を表現する TTS データモデル要素のうち CBlock はシステムのオブジェクト，CAttribute はオブジェクトの属性，CReference はオブジェクト間の関連を表す．より形式的には，これら 3 つのクラスは，UML Infrastructure Library[2]（UML や MOF[7] 等，他の OMG モデリング言語の基礎として用いられているモデル要素群）の Core パッケージに関して表 2 に示す対応を持つ．

TTS	UML InfrastructureLibrary::Core
CBlock	Basic::Type Constructs::Classifier
CAttribute	Basic::Property Constructs::Property
CReference	Basic::Property Constructs::Property with 　Constructs::Association as association Abstractions::DirectedRelationship

表 2　構造的側面に関するデータモデル要素

(注1)：D の要素の例としては UML における集約 (Composed Of) や依存 (Depends On) が考えられる．

Basic, Constructs, Abstractions は Core のサブパッケージである．Basic パッケージは UML を含むモデリング言語を構築する際の最低限の基礎として設計されたクラスベースの言語であり，表に記載の 1) Basic::Type, 2) Basic::Property はそれぞれ 1) データ型またはクラスと 2) クラスの属性を表す．Constructs パッケージは Basic パッケージにいくつかの拡張を施したもので，UML は Constructs パッケージを利用して定義されている．表に記載の Constructs::Association は型付けされたオブジェクトの組の定義である．Abstractions パッケージは Basic や Constructs を含む他のパッケージに共通して利用されるモデル要素を定義しており，Abstractions::DirectedRelationship は起点と終点を区別する要素間の関係である．

また，ReferenceType は CReference の種類であり，要求分析時にオブジェクト間の関連（UML における依存，集約など）をより詳細に指定する場合に利用する．CProperty は CAttribute と CReference を，CBlock が持つ性質として抽象化したものである．

表 2 の対応関係に基づけば，モデリング言語の定義において，対応する右欄のモデル要素を利用している定義を調べることによって，TTS データモデルにおける a) CBlock, b) CAttribute, c) CReference は UML/SysML において順に a) クラス/ブロックやデータ型，b) 属性，c) 関連または依存として順に対応付けることができる．また一般に UML Infrastructure Library を利用する他のメタモデルに対しても，同様に構造的側面に関して TTS データモデルとの対応付けを考えることができる．

4. データモデルとトレーサビリティの実装

我々が構築した TTS-Schema とそのトレーサビリティのためのシステムの処理概要を図 5 に示す．展開された原文 (1) は構

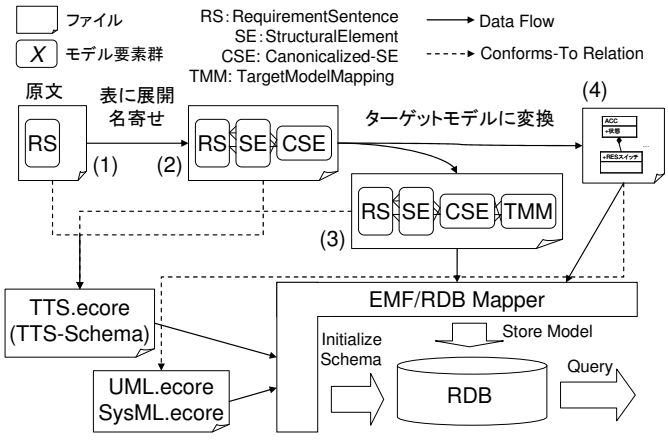

図 5　トレーサビリティシステムの処理概要

造 TTS の表に展開され，名寄せされる (2)．名寄せ済みの TTS モデルはターゲットモデルへと変換され，ターゲットモデルとのマッピング情報を持つ TTS モデル (3) およびターゲットモデル (4) を得る．(1)〜(4) は全て EMF のモデルであり，(1)〜(3) は図 3 で定義したメタモデル TTS.ecore（TTS-Schema）に，(4) は UML.ecore または SysML.ecore にそれぞれ従っている．

EMF モデルを RDB に格納する EMF/RDB Mapper には Teneo を利用した．TTS.ecore, UML.ecore, SysML.ecore に基づいて初期化された RDB スキーマに，上記 (1)〜(4) の EMF モデルが格納され，SQL クエリを用いてこのデータからトレーサビリティの情報を抽出する．

我々はこのシステムを用いて，自動車製品 ESTIMA の ACC(Adaptive Crouse Control) 取扱書 を題材に，構造に関する TTS データモデルを作成しトレースを行った．図 6 は ACC の「-SET スイッチ」というスイッチにより ACC の制御が開始されることについて述べた要求「R101」に対して，前向きのトレースを実施した結果である．

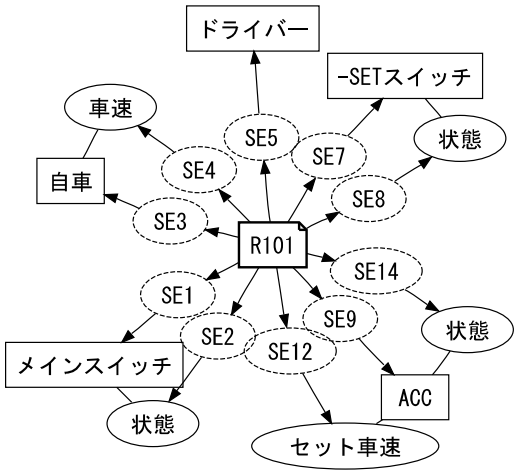

図 6　前向きトレースの例

破線楕円は原文展開後の各行である構造要素 (StructuralElement) を，長方形および楕円はそれぞれ SysML のブロック，属性を表す．構造要素から SysML 要素への過程で経由する名寄せ済み構造要素 (CStructuralElement) は省略している．この結果を用いて，要求 R101 が追加された場合や変更された場合に影響を受けるターゲットモデルの要素を，要求の分析過程とともに知ることができる．

図 7 は「自車」ブロックの「車速」プロパティに関する後ろ向きのトレースを実施した結果である．

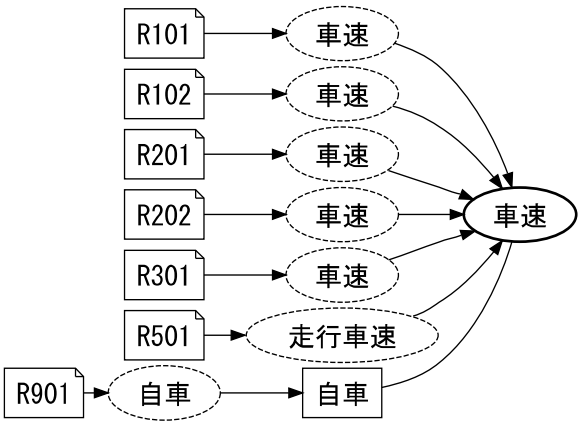

図 7　後向きトレースの例

破線楕円と長方形，楕円の意味は図 7 と同じである．この結

果を用いて，例えばターゲットモデルにおいて，コストの制約などから自車の「車速」属性を省略した際に影響を受ける要求一覧 (R101, R102, …) を，要求の分析過程とともに知ることができる．

5. おわりに

本稿では，要求文書を検証可能なモデルに形式化する際のトレーサビリティの問題に対処するため，特にシステムの構造的側面に関して 3 つの情報：a) 原文の展開から分析結果に至る作業経過，b) ターゲットモデルとの対応関係，c) 一般化されたシステムの構造的側面をもつデータモデルと，モデル要素の推移的な依存関係を考慮したトレース手法を提案した．

要求を段階的に形式化してゆく過程と要求に対応するシステムモデルがトレーサブル（追跡可能）になることで，要求とモデルの間の実現関係における網羅性が向上し，また対応付けの根拠も明確となる．その結果，仕様変更時の影響度の見積もりや代替案の検討が効率的・効果的に行えるようになる．このようにトレーサブルかつ曖昧性のない検証可能な仕様を得ることで意思伝達の困難さとソフトウェアの複雑さに効果的に対処することができ，製品の品質と生産性を向上させることができる．

文　献

[1] 田丸喜一郎，”組込み業界の現状と将来〜経済産業省組込みソフトウェア産業実態調査より〜，”独立行政法人情報処理推進機構ソフトウェア・エンジニアリング・センター，2008

[2] “ OMG Unified Modeling Language (OMG UML) Infrastructure V2.1.2, , ” OMG Available Specification, 2007

[3] “ OMG Unified Modeling Language (OMG UML) Superstructure V2.1.2, , ” OMG Available Specification, 2007

[4] “ OMG Systems Modeling Language (OMG SysML?), V1.0, ” OMG Available Specification, 2007

[5] Nancy G. Leveson., Jon Damon Reese, Mats P.E. Heimdahl, ”SpecTRM: A CAD System for Digital Automation,” 17th Digital Avionics Systems Conference (DASC 1998), 1998

[6] Martin Fowler, David Rice,“ Patterns of Enterprise Application Architecture, ” Addison-Wesley Professional, 2003.

[7] “ Meta Object Facility (MOF) Core Specification Version 2.0, ” OMG Available Specification, 2006