ゴール指向洗練パターン駆動によるユースケースモデリング

ゴール指向洗練パターン駆動によるユースケースモデリング*

本田　耕三[†a)]　平山　秀昭[†]　中川　博之[††]　田原　康之[†]
大須賀昭彦[†]

Goal Oriented Refinement Pattern Driven Use Case Modelling*

Kozo HONDA[†a)], Hideaki HIRAYAMA[†], Hiroyuki NAKAGAWA[††],
Yasuyuki TAHARA[†], and Akihiko OHSUGA[†]

あらまし　ゴール指向要求分析手法 KAOS は，要求を系統的，論理的にモデリングできる要求分析手法である．本論文では，ゴール指向要求分析の成果をオブジェクト指向設計プロセスに組込むことを意図して，KAOS モデルをユースケースモデルに変換するアプローチを提案する．この変換は洗練パターンを媒体としている．提案アプローチにより作成したユースケースモデルとあらかじめ用意された基準モデルを比較し，提案アプローチ適用の効果を評価するために，米国 ATM システムを事例とするユースケースモデリングに提案アプローチを適用した．この結果，提案アプローチは効果的に適用され，モデル変換における洗練パターンの意味（シナリオ）の継承と属人性の排除について効果を確認できた．

キーワード　ゴール指向要求分析，KAOS モデル，要求抽出，要求定義，ユースケースモデル

1. まえがき

要求定義工程から設計工程への移行とは，ユーザの視点で定義した要求を内部の構造と振舞いに置換え，その実現方法を具体化することである．要求定義と設計の要素間には，論理的根拠に基づく妥当性と相互追跡性が必要となる [1].

要求定義工程と設計工程にはそれぞれ実績ある手法が存在する．要求定義工程では，要求を網羅的，論理的に抽出・分析して体系的に定義できると定評のあるゴール指向要求分析手法がある．KAOS [2]，i* [3]，及び NFR フレームワーク [4] はその代表的な手法である．ゴール指向要求分析手法を使うと，個人の能力に過度に依存することなく，抽象目標から論理的に要求を抽出できる．このようにして抽出された要求は，必要となる根拠や要求間の関係が体系的にモデル化されていて，その合理性の確保を期待できる．しかし，こうして合理的に定義することができた要求を，漏れなく設計に反映して実装する体系だった手順は，まだ確立されていない．

設計工程では，要求の定義から設計，実装の各工程までを幅広くサポートし，静的構造と振舞いの実現方法を効果的に記述する統一モデリング言語（以降，UML）[5] を用いたオブジェクト指向開発手法が広く利用されている．要求をユースケースモデルで定義し体系的に設計・実装へと具体化する手法としては，ICONIX プロセス [6], [7] や Rational Unified Process（以降，RUP）[8] のように確立されたものがあり，実務で活用されている．しかし，ユースケースモデルのみによる要求の抽出・分析は，要求の体系化や論理的根拠の面で十分とは言えず，ゴール指向やその他の要求分析手法を事前に利用して要求を抽出することが多い．そして，抽出された要求をこれらの要求分析手法からユースケースモデルに反映するときに発生するギャップ，すなわち抽出した要求の情報が漏れてしまうことが，問題となっている．

本論文では，KAOS による要求モデルをユースケースモデルに変換するアプローチを提案する．提案アプローチは大きく次の二つのプロセスで構成され，全体

† 電気通信大学大学院情報システム学研究科，調布市
Graduate School of Information Systems, The University of Electro-Communications, Chofu-shi, 182–8585 Japan
†† 大阪大学大学院情報科学研究科，吹田市
Graduate School of Information Science and Technology, Osaka University, Suita-shi, 565–0871 Japan
a) E-mail: khonda@ohsuga.is.uec.ac.jp
* 本論文は，学生論文特集秀逸論文である．
DOI:10.14923/transinfj.2015PDP0017

を通して，ゴールモデルの洗練パターンによって駆動される．

（1） ユースケース図への変換

（2） イベントフロー図への変換

本提案の目的は，KAOS モデルのセマンティクスを，洗練パターンを介して ICONIX プロセスに反映することである．洗練パターンとは，それを使うとゴールを戦略的に分解できると Lamsweerde [9] が推奨している AND グラフの基本的な分解パターンである．モデル間の変換をそれぞれのセマンティクスを含む洗練パターンを介して実施することにより，洗練パターンで意味づけられた要求定義モデルの情報（シナリオ）を，各モデル間の変換で逐次継承することができる．また，洗練パターンを介したモデル間の変換それぞれは変換規則に従って実施されるため，変換の多くの部分を自動化でき，モデル作成における属人性の排除を期待できる．

また，提案アプローチの有効性を確認するために，米国の ATM システムを事例にとり提案アプローチを適用した．その中で，提案アプローチにより作成したユースケースモデルと要求記述とともに提供されているユースケースモデルを比較・検証した．

2. 以降の内容について説明する．まず **2.** では，既存の技術でゴール指向要求分析の代表的な手法の一つである KAOS を「遮断バーの安全踏切」のモデルを使って説明する．次の **3.** では，オブジェクト指向開発プロセスにおけるユースケースモデルの位置付けを説明する．**4.** では KAOS モデルからユースケースモデルへの変換アプローチを提案し，「遮断バーの安全踏切」を具体例として説明する．**5.** では，そのアプローチの有効性を確認するため，事例（米国 ATM システム）に提案アプローチを適用し検証した．また，次の **6.** では，関連研究について述べ，最後に，**7.** で今後の課題，研究等について述べる．

2. ゴール指向要求分析手法 KAOS

KAOS はゴール指向要求分析の代表的な手法の一つである．抽象的なゴールを最終目標に設定し，それを AND/OR グラフで洗練・詳細化することによってシステムに対する要求を分析・抽出化する [2]．システムに対応するエージェントに達成の責務を割当てることができるリーフゴールが，システムに対する要求である．本論文では以降，説明のため，リーフゴールとして抽出した要求を要求ゴールと呼ぶ．KAOS は，

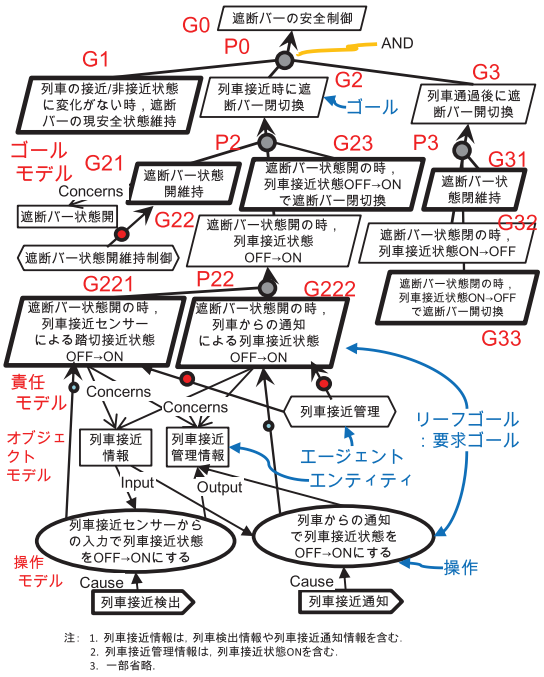


図 1 遮断バー安全制御の KAOS モデル例 (一部省略)
Fig. 1 Example of KAOS model expresses portions of cross bar safety control.

ゴール（Goal），責任（Responsibility），オブジェクト（Object），操作（Operation）の基本 4 モデルで構成され，ゴールモデルを中心に要求を分析する．KAOS はゴールから操作要求を導くのに有効である [10]．また，要求モデリングのみならず設計モデリングにも踏み込んで使用されている [11], [12]．

KAOS モデルの例として，図 1 に鉄道踏切を安全に制御する踏切コントローラの一部である遮断バー安全制御のモデルを示す．列車の接近/非接近に応じた遮断バーの安全な開閉制御に関する要求モデルであるが，本提案の説明において冗長と判断した部分は省略している．

3. ユースケースモデル

ユースケースモデルはユースケース図とユースケース記述で構成される．ユースケース記述には，ユースケース図の説明としてユースケースのシナリオや事前・事後条件などが記述される．シナリオはイベントフローで表現されることが多く，ユースケース記述の中心的な情報である．本論文でも，ユースケース記述に該当するものとして，イベントフロー図を採用している．ユースケースモデルは，オブジェクト指向開発

における要求定義モデルとして頻繁に使用される．例えば，ICONIX はユースケース駆動プロセスであり，獲得された要求がユースケースモデルに適切に反映されていれば，要求を正しく反映した設計が期待できる．しかし，ユースケースモデルは要求を利用者等のアクターとシステムとの間のインタラクションとして定義するものであり，要求を体系的，論理的に抽出する方法を提供するものではない．したがって，要求抽出にはユースケースモデルよりも体系的，論理的に要求を抽出，分析できるゴール指向要求分析手法 KAOS の方がより適していると言える．

4. KAOS モデルからユースケースモデルへの変換アプローチ

4.1 変換アプローチの概要

全体に渡って変換を駆動するゴールモデル洗練パターン 6 種類の一覧を表 1 に示す．KAOS ゴールモデルを，ユースケースと親和性が高い操作モデルを介して，ユースケースモデル（ユースケース図とイベントフロー図）に変換する．これによって，ゴール指向に基づいたユースケースモデルを作成することができる．入力となる KAOS ゴールモデルは，洗練パターンによる詳細化でシステム範囲が決定され，関連するエージェントやエンティティ等オブジェクトを含めてモデル化されたものを対象としている．一方，出力となるユースケース図では，ユースケースとアクターとの関連及びユースケース間の関連が表現される．そのユースケース記述として，ユースケースのシナリオをイベントフロー図で表現した．事前・事後条件はアクター間のメッセージや参照する情報に含めるようにした．提案アプローチは，ユースケース図に変換する STEP1〜5 とイベントフロー図に変換する STEP6〜8 で構成される．変換処理全体は図 2 のフローとなる．図 2 の中の数字は，ステップ番号（ex.STEP1.）を示している．各ステップの自動/半自動/手動の区別，新規/既存の区別，及び概要は次のようになる．ここで，先頭の各数字はステップ番号に相当し，“新規” と書かれたステップは，提案アプローチにおいて新規に導入するステップを表す．

1. 一段の AND グラフ抽出（自動，既存）
2. ゴールモデル → 操作モデル変換（半自動，新規）
3. 操作モデル → ユースケース図変換（自動，新規）
4. ユースケース図/KAOS モデル相互洗練（手動，既存）

表 1 ゴールモデルの洗練パターン一覧
Table 1 View of Goal model refinement pattern.

洗練パターン名	説 明
マイルストーン駆動	幾つかのマイルストーンを経由して目標状態に到達するパターン
ケース分解	複数の独立したサブ状態から成る目標状態を実現するパターン
ガード条件導入	ガード条件成立時に目標状態に到達できるパターン
分割・統治	単純なサブゴールに分解してそれぞれごとに実現するパターン
モニタ不能駆動	関心事のモニタ能力がないときにモニタ機能を外部に割振るパターン
制御不能駆動	関心状態の制御能力がないときに外部に制御を割振るパターン

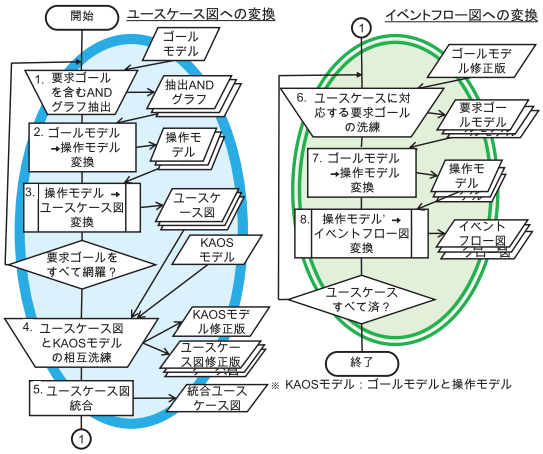


図 2 ゴールモデルからユースケースモデルへの変換フロー
Fig. 2 Transformation flow from a goal model to a use case model.

5. ユースケース図の統合（自動，新規）
6. 洗練パターンによる要求ゴール分解（手動，既存）
7. ゴールモデル → 操作モデル変換（半自動，新規）
8. 操作モデル → イベントフロー図変換（自動，新規）

STEP2，STEP3，STEP7，STEP8 では，それぞれ洗練パターンを介したモデル間の変換を実施する．各モデルにおいて洗練パターンに基づいた変換テンプレートを定義した．変換テンプレートとは，洗練パターンに対応した各モデルのパターンである．それぞれの変換では，QVT [13] による変換規則に従って，変換テンプレートの置換えと情報のマッピングを実施する．この結果，洗練パターンによる要求定義モデルシナリオの継承と，変換における属人性の排除を期待できる．

上述の内容を踏まえ，STEP1〜STEP8 までの変換

アプローチ全体の流れを説明する．以降，“一つの親ゴールとその直下のサブゴールによる AND グラフ”を説明のため “一段の AND グラフ” と呼ぶ．STEP1で，リーフゴール（要求ゴール）を探索し，それをサブゴールとして含む一段の AND グラフを抽出する．次の STEP2 では，抽出した AND グラフそれぞれを，操作モデルの変換テンプレートを利用した変換規則に従って，機械的に操作モデルに変換する．この STEPでは，新たに環境エージェントとイベントの名称を手動で追記する．更に STEP3 で，それぞれの操作モデルを，ユースケース図の変換テンプレートを利用した変換規則に従って，機械的にユースケース図に変換する．ここで得られるユースケース図は，STEP1 で抽出された一段の AND グラフそれぞれに対応している．STEP4 では，それぞれのユースケース図を全体として俯瞰し，必要があれば修正する．例えば，共通ユースケースの抽出，包含・拡張・汎化/特化関係の見直し，ユースケースの分割・結合等の修正である．修正結果を KAOS モデルに反映し，それぞれのユースケース図を確定する．これまでに得られた各階層一段ごとのユースケースについて，STEP5 では，上位階層のユースケースに下位階層のユースケース（サブユースケース）を代入することによって機械的に統合する．ここまでの STEP で，要求ゴール間の関連に相当するユースケース図を得ることができる．以降の STEP では，個々のユースケースのユースケース記述に相当するイベントフロー図を求める．

STEP6 では，要求ゴールにおけるシステムと外部環境とのインタラクションシナリオを明確にするために，一つのイベントだけに関わるゴールをサブゴールとする AND グラフを抽出するまで要求ゴールを階層的に分解する．このゴール分解にも，洗練パターン（表 1）を利用する．このようにして分解した要求ゴールごとに，STEP1 と同様に AND グラフを一段ずつ抽出し，以降のステップでイベントフロー図に変換する．STEP7 では，STEP2 と同じアルゴリズムで，一段ごとの AND グラフをそれぞれ操作モデルに変換する．最後に STEP8 で，それぞれの操作モデルを，イベントフロー図の変換テンプレートを利用した変換規則に従って，機械的イベントフロー図に変換する．こうして，STEP5 でユースケース図が得られ，STEP8 でユースケースそれぞれのイベントフロー図が得られる．

4.2 変換テンプレート

洗練パターンごとの各モデルの変換テンプレート一覧を図 3 に示す．KAOS モデルのメタモデル（図 4）及びユースケースモデルのメタモデル（図 5）に基づいて，これら変換テンプレートを定義した．

KAOS モデルのメタモデル（図 4）は，Heaven and Finkelstein [14] の KAOS モデルのメタモデルに，変換テンプレートで利用している関連情報を明示のために追記したものである．ユースケースモデルのメタモデル（図 5）は，OMG の Superstructure Specification [15] で定義している Use case のメタモデルに，Event Flow のメタモデルを UseCaseDescription で関連付けて作成した．Event Flow のメタモデルとして，Eclipse Modeling Framework（EMF）で使用されている Sequence Diagram のメタモデル [16] を用い，KAOS モデルのメタモデル同様，変換テンプレートで利用している関連情報を明示のために追記した．

変換テンプレート（図 3）の構成について，各モデルごとに説明する．

ゴールモデルの変換テンプレート

親ゴールの実現シナリオをより明確に表現するために，Lamsweerde の洗練パターン（ゴールモデル）にソフトウェアエージェント，環境エージェント，及びエンティティを関連付けたものを，ゴールモデルの変換テンプレートとした．

操作モデルの変換テンプレート

親ゴールを実現する操作の流れを構成している．各操作は，必要なエンティティを入出力することによってそれぞれのサブゴールを実現する．その結果，最終的に親ゴール実現に相当する結果イベントを出力する．また，操作の流れと利害関係者を明示するために，操作を起動する開始イベント，操作間をつなぐ中間イベント，最終結果となる結果イベント，及びそれらイベントと関連付けられる環境エージェントも構成要素として記述している．

操作モデルの変換テンプレートは，洗練パターンのシナリオに従ってそれぞれ構成されており，それによってゴールモデル洗練パターンの意味を継承している．すなわち，図 3 の操作モデルの列を参照しながら説明すると，マイルストーン駆動では，マイルストーンに対応するイベント 2 を仲介として操作 B，C を逐次駆動し，親ゴールに相当するイベント A を出力する；ケース分解では，それぞれのケースに対応して操作 B，C を実行し，それぞれのケースにおける親ゴー

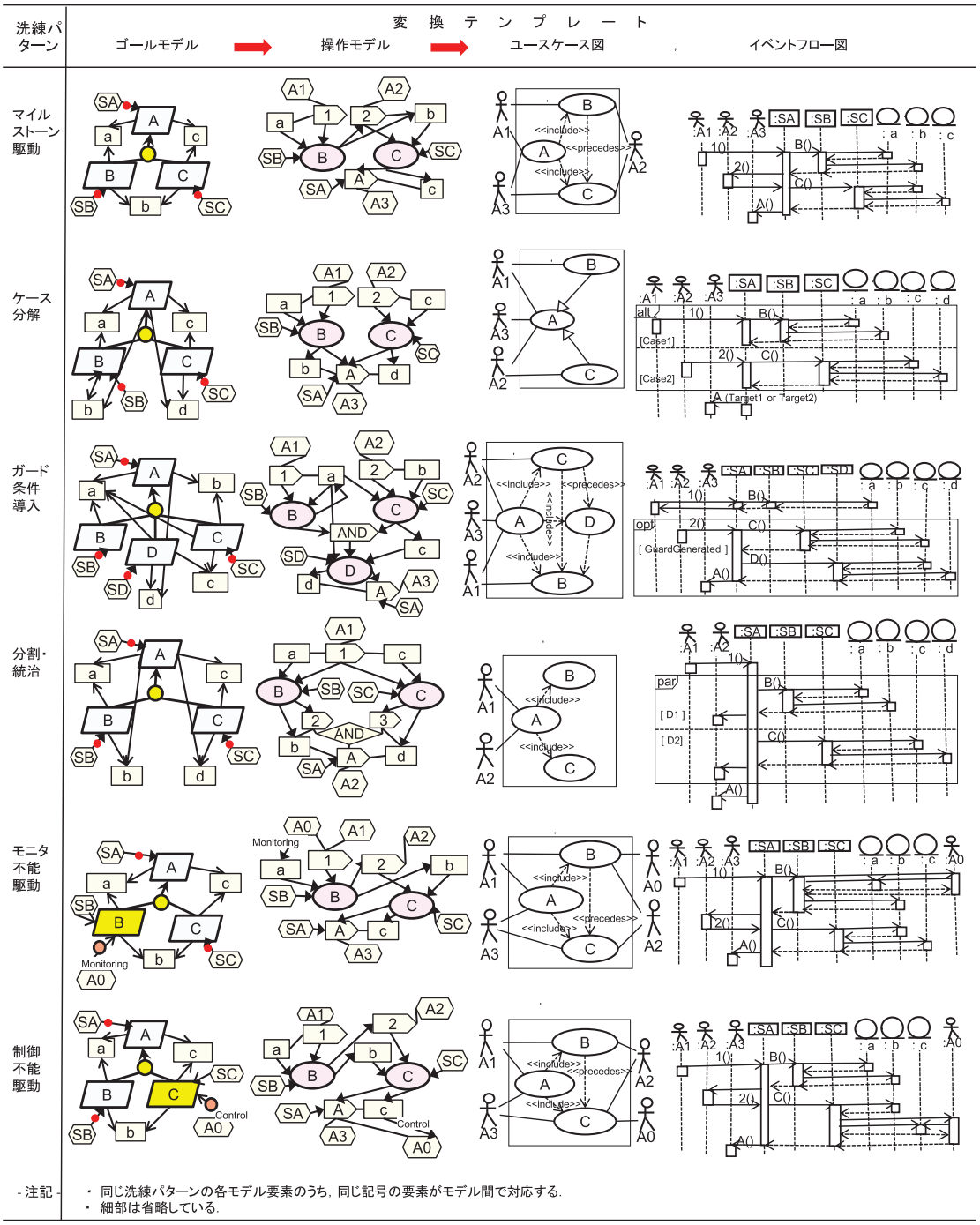


図 3　KAOS →ユースケースモデル変換テンプレート一覧
Fig. 3　A list of templates to use for transforming KAOS models to use case models.

ルに相当するイベント A を出力する；ガード条件導入では，操作 B で現状態を維持しているときに，ガード条件発生を操作 C が認識すると，それによって操作 D が親ゴールに相当するイベント A を出力する；分割・統治では，サブゴールをそれぞれ独立に実現する操作 B と C の出力イベントを統合して，親ゴールに相当す

るイベント A を出力する；モニタ不能駆動では，環境エージェントによるモニタ結果を操作 B で取得し，それを使って操作 C が親ゴールに相当するイベント A を出力する；制御不能駆動では，操作 B による制御内容を操作 C で環境エージェントに委託し，その結果親ゴールに相当するイベント A を出力する．

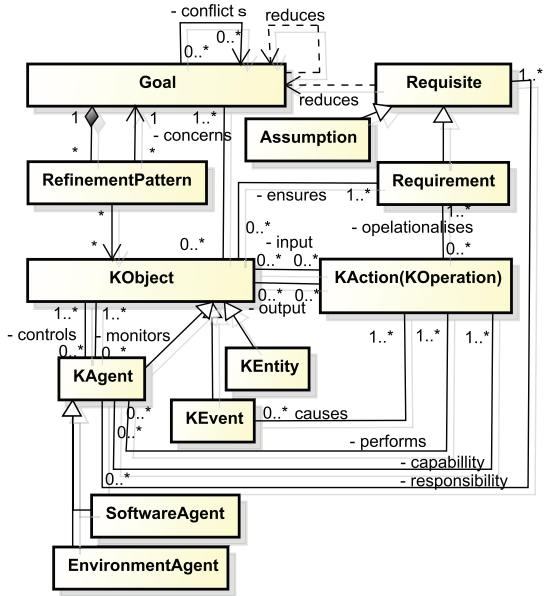


図 4　KAOS モデルのメタモデル（[14] を一部省略/詳細化）
Fig. 4　KAOS Meta Model.

ユースケース図の変換テンプレート

ユースケースにアクターを関連付け，親とサブユースケースの分解関係を明確にするため，ケース分解の場合を除いて，親ユースケースはサブユースケースを <<include>> する．更に，それらユースケースを洗練パターンのシナリオに従って関連付けることによって，ゴールモデル洗練パターンの意味を継承している．すなわち，図 3 のユースケース図の列を参照しながら説明すると，マイルストーン駆動では，サブユースケース B と C を <<precedes>> で関連付けマイルストーンによる駆動順序を定義している；ケース分解では，サブゴールそれぞれの達成が各ケースにおける親ゴールの達成に相当するので，サブユースケース B と C それぞれと親ユースケースは汎化/特化の関係で関連付ける；ガード条件導入では，サブユースケース C でのガード条件発生認識結果を受けて，サブユースケース D で目標を達成する順序とするため，サブユースケース C と D を <<precedes>> で関連付けている．更に，それらの実行は現状態であることが条件となるので，サブユースケース C と D は現状態を維持するサブユースケース B に依存する構成となる；分割・統治では，分割されたサブユースケース B と C の実行結果を親ユースケース A で統合する構成とする；モニタ不能駆動では，外部でのモニタ内容をサブユースケース B で認識し，その結果を受けてサブユー

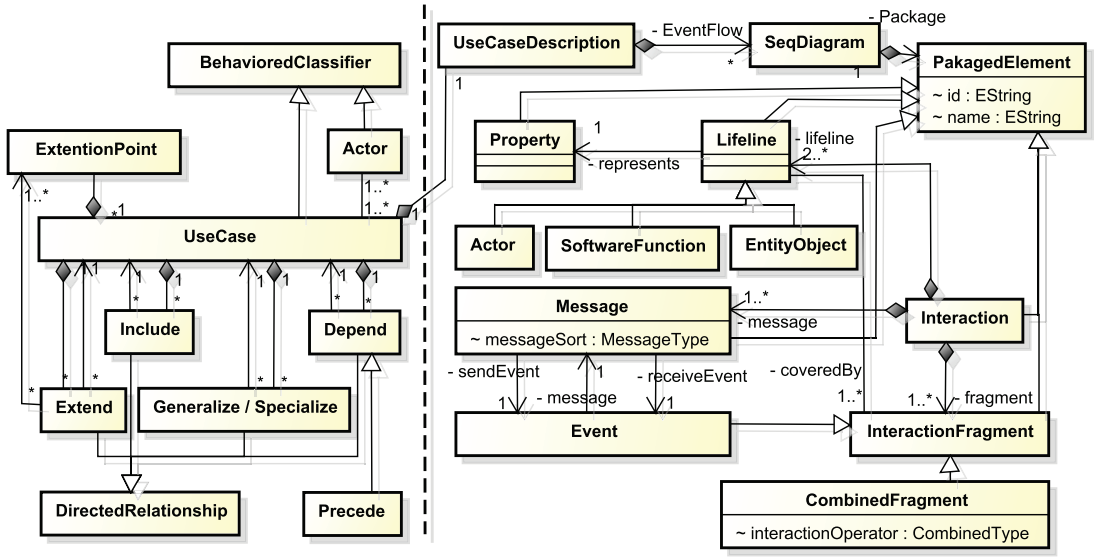


図 5　ユースケースモデルのメタモデル（[15], [16] を一部省略/詳細化）
Fig. 5　Meta Model of Use Case Model.

スケース C で目標を達成する構成となる．そのため，サブユースケース B と C を <<precedes>> で関連付けている；制御不能駆動では，サブユースケース B で仮想的に制御した内容をサブユースケース C で具体的な制御として外部に委託するので，サブユースケース B と C を <<precedes>> で関連付ける．

イベントフロー図の変換テンプレート

イベントフロー図の表記はシーケンス図に準じている．ライフラインとしてアクター，ソフトウェア機能，エンティティを配置している．更に，ソフトウェア機能をメインとサブの機能に分け，洗練パターンの親子関係を継承できるようにした．メインの機能はサブの機能を起動し，洗練パターンのシナリオに従って，イベントフローを駆動する．サブの機能はエンティティを入出力し具体的なイベント処理を実行する．洗練パターンそれぞれのシナリオを，主要なメッセージを添えた相互作用フラグメントの組み合わせで表現することにより，ゴールモデルの洗練パターンの意味を継承している．すなわち，図 3 のイベントフロー図の列を参照しながら説明すると，マイルストーン駆動では，マイルストーンの順番に従ったイベントの流れを表現している．図中の “A()”，“B()”，“C()”，“1()”，及び “2()” はそれぞれメッセージを示す．他のパターンでも同様である；ケース分解では，複合フラグメントの分岐による処理を使って，ケースごとの相互作用を表現している；ガード条件導入では，複合フラグメントの特定条件による処理を使ってガード条件発生時の相互作用を表現している；分割・統治では，分割されたイベントが個々に実行され，最後に統合される必要があるため，複合フラグメントの並行処理を使って相互作用を表現している；モニタ不能駆動では，外部でモニタされた情報を使った相互作用のフローとなっている；制御不能駆動では，仮想的に処理した制御内容を外部に委託するフローとなっている．

4.3 変換アルゴリズム

4.1 で概説した変換アプローチの各 STEP のうち，新規に提案する STEP2 (STEP7)，STEP3，STEP5，STEP8 について，詳細に説明する．

STEP2 (STEP7)： 図 6 に示した QVT による変換規則に従い，ゴールモデルから操作モデルに変換する．上段の左右のモデルはそれぞれゴールモデルと操作モデルにおける変換テンプレートのメタモデルである．ともに，KAOS モデルのメタモデル（図 4）に基づいている．すなわち，ゴールは洗練パターンで関連づけられ，ゴールを分解して得られる要求（要求ゴール）はアクション（操作）によって操作できるようになるという関係から，操作も洗練パターンで関連付けられる．また，エージェント，イベント，エンティティなどは，KAOS モデルのメタモデルに基づき，ゴールまたは操作に関連付けられる．下段の When 節と Where 節は，上段左右に記述されているモデルの要素間のマッピング規則を示している．

前 STEP で抽出した AND グラフを操作モデルの変換テンプレートで置換えた後，下段の When 節と Where 節の規則に従ってモデル要素間の情報をマッピングすると変換が終了する．ただし，イベントとそれに関連付けられる環境エージェントはこのステップで初めて抽出されるため，それらの名称を最後に手動で

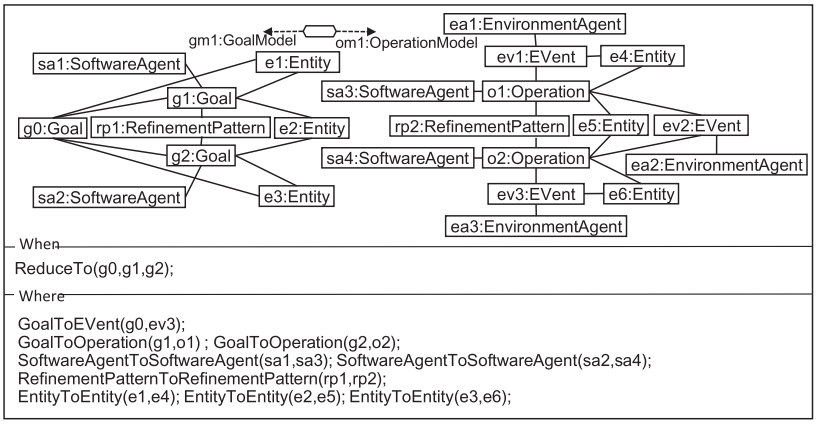


図 6 ゴールモデル→操作モデル変換規則（QVT）
Fig. 6 Rule for Goal Model to Operation Model.

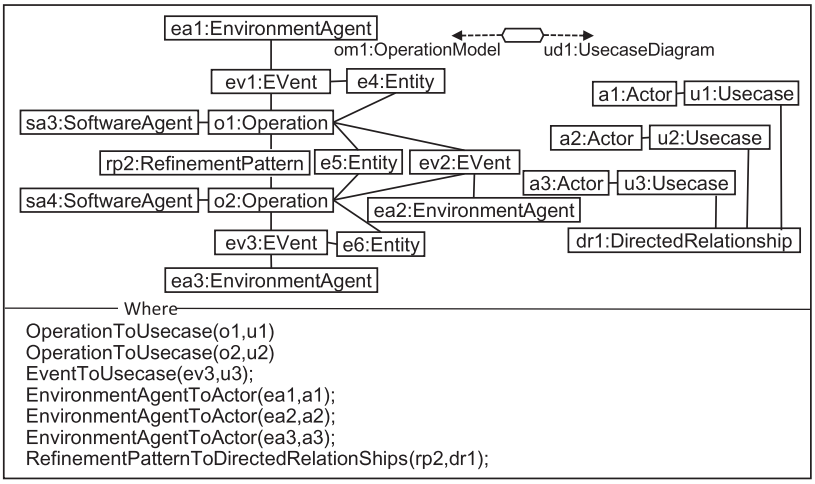


図 7　操作モデル→ユースケース図変換規則（QVT）
Fig. 7　Rule for Operation Model to Use Case Diagram.

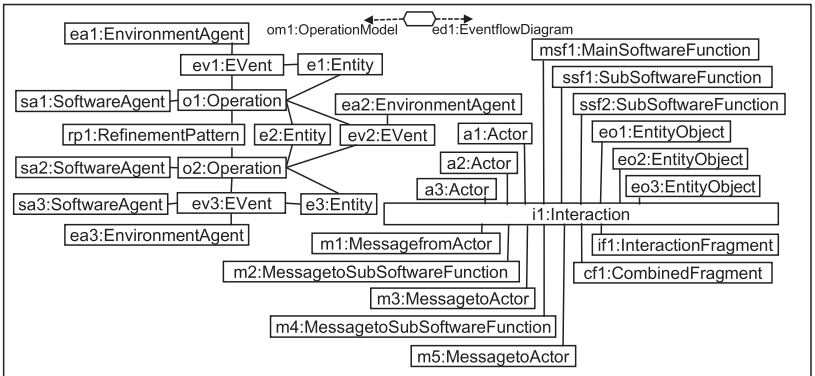
入力する．

STEP3：　前 STEP で変換した操作モデルを，QVT による変換規則（図 7）に従って，ユースケース図に変換する．図 7 における上段左右のモデルは，それぞれ操作モデルとユースケース図の変換テンプレートのメタモデルである．左側のモデルは，図 6 の出力側のモデルと同じものである．右側のモデルは，ユースケースモデルのメタモデル（図 5）における UseCase の部分（図の左側部分）に基づいている．すなわち，洗練パターンの意味（シナリオ）に合うように，DirectedRelationship の関係（Precedes 等）を使ってユースケースを関連付け，そのユースケースにアクターを関連付けている．下段の Where 節の規則に従って，左右モデルのモデル要素間をマッピングする．

STEP5：　STEP3 または STEP4 での変換結果である各階層一段ごとのユースケースについて，上位階層のユースケースにそれを分解した下位階層のユースケース（サブユースケース）を代入し統合する．これを全階層について実施することにより，最下層のユースケース間の関連に統合することができる．ケース分解洗練パターンでは，サブゴールそれぞれの達成は各ケースにおける親ゴールの達成に相当するので，親ゴールとサブゴールは汎化/特化の関係にある．したがって，ケース分解の場合については，この関係をユースケースの統合前後でも明示するために，親ユースケースとサブユースケースを汎化/特化の関係で関連付け，統合後もこの関係を明記する．

STEP8：　前 STEP で変換した操作モデルを，QVT による変換規則（図 8）に従って，イベントフロー図に変換する．図 8 上段左，右のモデルは，それぞれ操作モデルとイベントフロー図の変換テンプレートのメタモデルである．左側のモデルは，図 6 の出力側のモデルと同じものである．右側のモデルは，ユースケースモデルのメタモデル（図 5）における UseCaseDescription の部分（図の右側部分）に基づいている．すなわち，アクター，ソフトウェア機能（メインソフトウェア機能またはサブソフトウェア機能），エンティティオブジェクト，及びメッセージからなるライフラインと，それらライフライン間のメッセージからなるインタラクションを使って，洗練パターンに合ったイベントフローを構成する．また，インタラクションはインタラクションフラグメントで構成され，複合フラグメントもインタラクションフラグメントの一つである．洗練パターンの種類によって，単純なインタラクションフラグメントを使ったり，複合フラグメントを使ったりする．下段の Where 節の規則に従って，左右モデルのモデル要素間をマッピングする．

以上，新規に提案するそれぞれの変換アルゴリズムを詳細に説明した．これまでに説明した変換テンプレート（図 3）や QVT による変換規則（図 6，図 7，図 8）は，各洗練パターンの基本的な型に対する変換の基本形を定義している．マイルストーン駆動のマイルストーンが二つ以上の場合，ケース分解で 3 ケース以上の場合，分割・統治で 3 分割以上の場合など基



Where

```
If ( RefinementPattern == DecompositionByCases ) RefinementPatternToCombinedFragment:alt(rp1,cf1)
elseif ( RefinementPattern == GuardIntroduction ) RefinementPatternToCombinedFragment:opt(rp1,cf1)
elseif ( RefinementPattern == DivideAndConquer ) RefinementPatternToCombinedFragment:par(rp1,cf1)
else RefinementPatternToInteractionFragment(rp1,if1) ;
EnvironmentAgentToActor(ea1,a1);  EnvironmentAgentToActor(ea2,a2);
EnvironmentAgentToActor(ea3,a3);
SoftwareAgentToMainSoftwareFunction(sa3,msf1);  SoftwareAgentToSubSoftwareFunction(sa1,ssf1);
SoftwareAgentToSubSoftwareFunction(sa2,ssf2);
EntityToEntityObject(e1,eo1);  EntityToEntityObject(e2,eo2);  EntityToEntityObject(e3,eo3);
EventToMessagefromActor(ev1,m1);  OperationToMessagetoSubsoftwareFunction(o1,m2);
EventToMessagetoActor(ev2,m3);  OperationToMessagetoSubsoftwareFunction(o2,m4);
EventToMessagetoActor(ev3,m5);
```

図 8　操作モデル→イベントフロー図変換規則（QVT）
Fig. 8　Rule for Operation Model to Event Flow Diagram.

本的な型からはずれる場合は，基本形で変換の枠組みを決定した後，ゴールモデルの情報に基づきマイルストーンを増やす等拡張することで対応可能である．また，各ゴール，各イベントにそれぞれソフトウェアエージェント，環境エージェントを一つずつ割り当てているが，それらは同一エージェントであっても良い．また，親ゴールに複数のエージェントが関与する場合やゴールに関連するオブジェクトが複数ある場合は，できるだけそれらを汎化した名称にしたり，それらを併記した名称とすることで対処できる．逆にない場合には，とりあえずダミーとして付与し，STEP8 終了後に見直すことによって必要であれば修正する．なお，STEP6 の要求ゴールの分解はゴールとイベントが 1 対 1 になると終了するので，ゴールに対応するイベントは必ず存在する．

提案する変換アルゴリズムは，洗練パターンを用いてモデリングしたゴールモデルを対象としている．洗練パターンを使用していない AND/OR 分解の部分は対象外のため変換されない．その詳細は本論文の範囲外となるので省略するが，STEP4 または STEP8 終了後に見直しが必要である．OR 分解の場合は，代替処理になるので修正せずそのまま分離するのが適切であると考える．AND 分解の場合は，必須処理になるので，ユースケース若しくはイベント処理として追加するのが適切であると考える．また，アクターとユースケースとの関連で多重度を明記する場合があるが，現時点では対応していない．

4.4　具体例によるユースケースモデルまでの変換の流れ

図 1 を事例として，提案アプローチによる変換の流れを具体的に説明する．Pxx は洗練パターンによる AND グラフである．

4.4.1　要求ゴールを含む AND グラフ抽出（STEP1）

図 1 において，要求ゴールを含む AND グラフ一段の抽出例として，G2 を分解している P2 と P22 がある．複雑化による図の認識悪化を防ぐためゴールに関連するソフトウェアエージェントやエンティティを一部省略している．P2 はガード条件導入の洗練パターンによる分解である．遮断バー状態開の維持が達成されているときに，列車接近状態の OFF から ON への切換えが達成されると遮断バー状態閉への切換が達成されるシナリオになっている．

4.4.2　ゴールモデル→操作モデル変換（STEP2）

P2 は，QVT 変換規則に従って図 9 の操作モデルに変換される．ガード条件「列車接近状態が OFF →

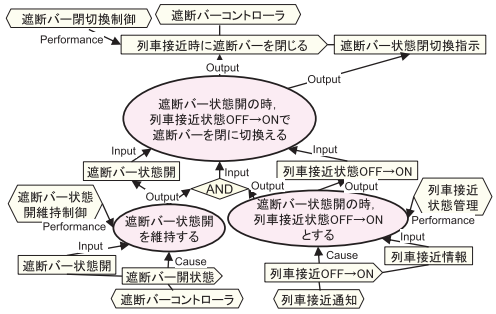


図 9　遮断バー閉切換操作モデル
Fig. 9　Operation model of closing cross bar.

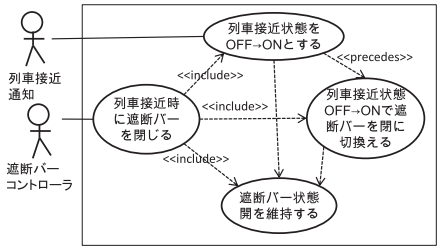


図 10　遮断バー閉切換ユースケース図
Fig. 10　Use case diagram of closing cross bar.

ON」に変わったら遮断バーを閉に切換える操作の流れになっている．

4.4.3　操作モデル→ユースケース図変換（STEP3）

図 10 は，操作モデル図 9 を QVT 変換規則に従って変換したユースケース図である．ガード条件に対応するユースケース「列車接近状態を OFF → ON にする」が先行して実行されれば，ユースケース「列車接近状態 OFF → ON で遮断バーを閉に切換える」が実行されることを示している．

4.4.4　ユースケース図と KAOS モデル相互洗練（STEP4）

ここでは，共通ユースケースの抽出を例にとって説明する．前節で導出した図 10 にユースケース「遮断バー状態開を維持する」がある．それから，「遮断バーの現状態を維持する」を共通ユースケースとして分離でき，その結果をユースケース図に反映させると図 11 が得られる．

4.4.5　ユースケース図の統合（STEP5）

図 11 のユースケース図を図 10 のユースケース図に組込み，更にケース分解 AND グラフ P22 から変換したユースケース図を組込むと，図 1 の G2 を P2 と P22 で詳細化した系統のユースケース図として図 12

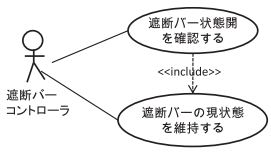


図 11　遮断バー開維持改訂ユースケース図
Fig. 11　Revised use case diagram of cross bar maintained open.

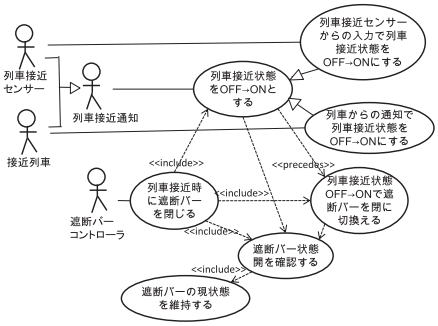


図 12　遮断バー閉切換改訂ユースケース図
Fig. 12　Revised use case diagram of closing cross bar.

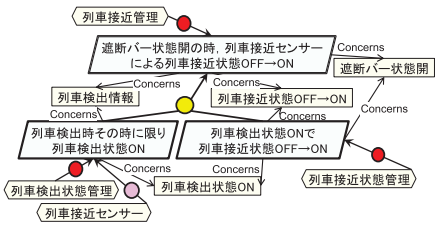


図 13　センサーによる接近検出ゴールモデル
Fig. 13　Goal model of train approaching detected by sensor.

に統合される．

4.4.6　洗練パターンによる要求ゴール分解（STEP6）

前節で作成した図 12 に含まれるユースケース「列車接近センサーからの入力で列車接近状態を OFF → ON にする」を例にとり説明する．このユースケースに対応する要求ゴールは，図 1 の G221 である．モニタ不能駆動洗練パターンを使うと，G221 を図 13 のように分解できる．ゴール「列車検出時そのときに限り列車検出状態 ON」によって「列車接近センサー」からモニタ結果を受取り，その結果ゴール「列車検出状態 ON で列車接近状態 OFF → ON」を実現するというシナリオになっている．

4.4.7　ゴールモデル→操作モデル変換（STEP7）

ゴールモデル図 13 を，モニタ不能駆動の QVT 変

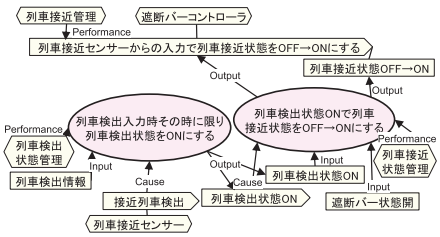


図 14　センサーによる接近検出操作モデル
Fig. 14　Operation model of train approaching detected by sensor.

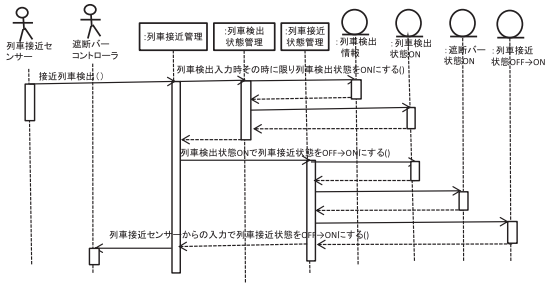


図 15　センサーによる接近検出イベントフロー図
Fig. 15　Event flow diagram of train approaching detected by sensor.

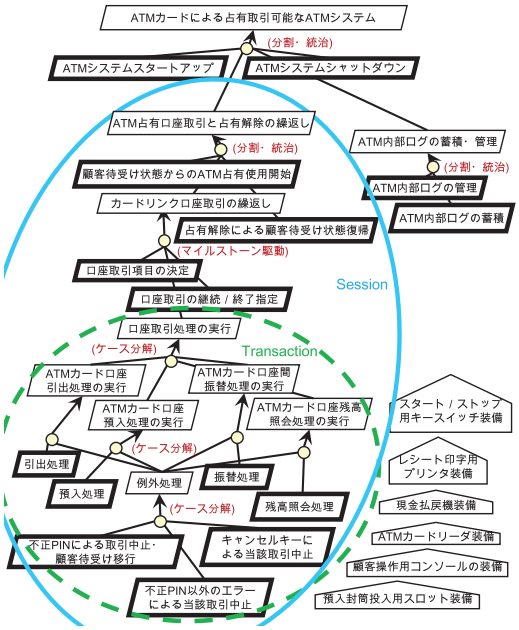


図 16　ATM カードによる占有取引可能な ATM システムゴールモデル（一部省略）
Fig. 16　Goal model of ATM system exclusively used with ATM card.

換規則に従って，操作モデルに変換すると図 14 になる．モニタ不能な部分を列車接近センサーに委ねたことに基づく操作「列車検出入力時そのときに限り列車検出状態を ON にする」を受けて操作「列車検出状態 ON で列車接近状態を OFF → ON にする」を実行する．

4.4.8　操作モデル→イベントフロー図変換（STEP8）

操作モデル図 14 を，モニタ不能駆動の QVT 変換規則に従って，イベントフロー図に変換すると図 15 になる．列車接近センサーからの「列車接近検出」メッセージ入力を処理した後,「列車検出状態 ON で列車接近状態を OFF → ON にする」を処理する流れになっている．

5.　適 用 評 価

提案アプローチを適用することにより，ゴールモデルを半自動的にユースケースモデルに変換できる．ユースケースモデリングの初心者を被験者として提案アプローチの適用実験を実施した．実験により，ゴールモデルの洗練パターンによる要求定義シナリオはモデル変換において逐次継承されているか（洗練パターンの継承），またモデル変換において属人性は排除されているか（属人性の排除）を評価した．

実験では，Bjork [17] が米国 Gordon College のサイトに公開している ATM System に対する教育用題材を事例とした．また，このサイトに提供されている ATM システムのユースケース図とシステムスタートアップ，システムシャットダウン，及びセッション部分のイベントフロー図を基準のユースケースモデル（以降，基準モデルと呼ぶ）とした．

本章では，適用結果のユースケースモデルと基準モデルとの差異は属人性に由来するとして，その差異の評価をもって属人性の排除の評価とする．ただし，複雑化を避けるため，ATM 内部ログ関連の機能は対象範囲外とした．また例外処理の分解は省略し，これをリーフゴールとして扱った．

「ATM カードによる占有取引可能な ATM システム」をトップゴールとするゴールモデル（図 16）を要求記述書に基づき準備し，被験者はそれに対して提案アプローチを適用することで，ユースケースモデルに変換した．すなわち，被験者は，要求記述書を参照しながら，STEP2，7 の一部と STEP4，6 を手動で実施し，その他の自動化可能な部分はアルゴリズムに

従って変換した．

被験者は情報工学系の大学院学生 2 名（A，B）であるが，要求分析及び KAOS ゴールモデルについては未経験であり，ユースケースモデリングについては講義による知識を有している程度である．また，ATM システムについては，日本国内 ATM の一般ユーザとして有する知識のみである．

5.1 評価の方法と判断基準

洗練パターンの継承評価については，ゴールモデル図 16 を構成する洗練パターンによるシナリオが，ユースケース図への各変換においてそれぞれ継承されているか，及び STEP6 で要求ゴールを分解した洗練パターンによるシナリオが，イベントフロー図への各変換においてそれぞれ継承されているかを評価した．この場合の判断基準は，変換規則を遵守した変換であることと変換結果が洗練パターンの意味を表現できているかどうかである．

モデル変換における属人性の排除評価については，基準モデルに対する変換結果モデルの適合率と再現率を算出し，非適合・非再現内容の発生原因と属人性との関係について評価した．適合率と再現率は，ユースケース図またはイベントフロー図を主要なモデル要素に分け，それごとに適合するか否かを判定し適合数と再現数をカウントして計算した．ユースケース図のモデル要素は，ユースケースと，一つのアクターとユースケースの組の二つであり，イベントフロー図のモデル要素はイベントごとの処理とした．これらのモデル要素はモデルそれぞれの特徴を直接的に表現するものである．

被験者モデルのモデル要素が，基準モデルのモデル要素に対して，表記が異なっている場合であっても同等な意味，役割，若しくは振舞いを意図している場合には適合しているとし，被験者のモデル要素の中に適合しているものがある場合は再現しているとした．また要求の定義であるため，ユースケース等の内容で適合しているか否かを判断し，セッション/トランザクションどちらの制御機能に含まれるか等の機能割当ては判断基準とはしなかった．後述する “表 3 ATM システムユースケース図作成結果の比較” と “表 5 イベントフロー図作成結果の比較” では，‘正’ の欄に適合を ‘○’，非適合を ‘×’ で示している．更に本論文では，基準モデルのモデル要素としては適合していないが，ATM システムのモデル要素としては有効であり適合していると見做せるものを ‘△’ で示した．

基準モデルに対する被験者 A モデルの適合率は “被験者 A モデルのモデル要素のうちで適合しているものの総数÷被験者 A モデルのモデル要素の総数” としている．また再現率は “基準モデルのモデル要素のうち被験者 A モデルのモデル要素によって適合されたものの総数÷基準モデルのモデル要素の総数” としている．

5.2 評価結果

○　洗練パターンの継承評価

事例における各変換で扱っている洗練パターンの種類と数を表 2 に示す．これに基づき変換した．

STEP2，3，7，8：　図 16 から抽出された 1 段ずつの AND グラフは，STEP2，STEP3 により，それぞれ変換規則図 6 に従って操作モデルへ，変換規則図 7 に従ってユースケース図へと正常に変換された．それぞれ，マイルストーン駆動，ケース分解，分割・統治の洗練パターンを継承した操作モデルとユースケース図になっている．例えば，マイルストーン駆動洗練パターンの変換結果を例にとると，操作モデルでは “「口座取引処理を選択する」→「口座取引処理を実行する」→「口座取引継続を指定する」” の順番で操作するように，ユースケース図では <<precedes>> を使って操作手順と同じく “「口座取引処理を選択する」→「口座取引処理を実行する」→「口座取引継続を指定する」” の順でユースケースが正しく駆動されるようにそれぞれ変換されている．

同様に，STEP6 で被験者 A，B が，洗練パターンを使って要求ゴールを分解したゴールモデルは，STEP7，STEP8 により，それぞれ変換規則図 6 に従って操作モデルへ，変換規則図 8 に従ってイベントフロー図へと正常に変換された．それぞれ，被験者 A のモデルはマイルストーン駆動，ケース分解，及びモニタ不能駆動の洗練パターンを継承した，被験者 B のモデルはマイルストーン駆動とガード条件導入の洗練パターンを継承した操作モデルとイベントフロー図になっている．例えば，被験者 A のモニタ不能駆動洗練パターンの変

表 2　事例における洗練パターンの種類と数
Table 2　Types and numbers of refinement patterns in the case study.

洗練パターン ＼ 変換STEP	STEP2，7	STEP3	STEP8
マイルストーン駆動	15	1	14
ケース分解	6	5	1
ガード条件導入	2	0	2
分割・統治	2	2	0
モニタ不能駆動	4	0	4
制御不能駆動	0	0	0
パターン延数	29	8	21

STEP2,7　ゴールモデル→操作モデル
STEP3　操作モデル→ユースケース図
STEP8　操作モデル→イベントフロー図

表 3　ATM システムユースケース図作成結果の比較
Table 3　Modeled use case diagrams compared with provided one.

サイト提供教材（基準モデル）			被験者Aモデル					被験者Bモデル				
ユースケース		アクター	ユースケース		正	アクター	正	ユースケース		正	アクター	正
System Startup		Operator	システムスタートアップ		○	顧客，銀行，オペレータ	①	システムスタートアップ		○	顧客，銀行，オペレータ	①
System Shutdown		Operator	システムシャットダウン		○	同上	①	システムシャットダウン		○	同上	①
Session		Customer	占有使用開始		○	同上	①	占有使用開始		○	顧客，銀行	①
			占有使用解除		○	同上	①	占有使用解除		○	同上	①
(transaction type選択) (another transaction指定)		Customer，Bank	セッション	取引処理選択	○	顧客，銀行	②	セッション	取引処理選択	○	同上	②
				取引継続指定	○	顧客，銀行，オペレータ	②		取引継続指定	○	同上	②
Transaction	Withdrawal		トランザクション（口座取引処理実行）	引出処理	○	顧客，銀行	②	トランザクション（口座取引処理実行）	引出処理	○	同上	②
	Deposit			預入れ処理	○	同上	②		預入れ処理	○	同上	②
	Transfer			振替処理	○	同上	②		振替処理	○	同上	②
	Inquiry			残高照会処理	○	同上	②		残高照会処理	○	同上	②
	Invalid PIN			例外処理	○	顧客，銀行，オペレータ	②		例外処理	○	同上	②

注．　○：正解，　×：不正解，　○で囲った数字：正解の組数（例．被験者Aのシステムスタートアップとアクターの組は全部で3組あり，正解が1組，不正解が2組である．）

換結果を例にとると，操作モデルではキー操作スイッチ ON 情報を受取った後，ATM を起動するように，イベントフロー図ではキー操作スイッチからの ON 情報を取り込むイベントに続けて，ATM を起動するイベントを処理するようにそれぞれ変換されている．

これまで述べたように，各変換は正常に実行され，洗練パターンの意味（シナリオ）は正しく継承されていることを確認した．

STEP5：被験者 A，B の統合結果ユースケース図はともに，ケース分解，マイルストーン駆動，及び分割・統治の関係でユースケースが関連付けられており，正常に統合されていることを確認できた．

例題の妥当性：各変換は洗練パターンの種類によらず，それぞれ一つの変換規則に従っている．すなわち，STEP2，7 では図 6，STEP3 では図 7，及び STEP8 では図 8 の QVT による変換規則に従ってそれぞれ変換されている．それぞれの変換規則は各洗練パターンを抽象化したメタモデルの領域で規定されているので，洗練パターンによる違いは変換規則そのものには影響しない．このことから，提案したそれぞれの QVT による変換規則を評価するのに，基本的には全ての洗練パターンを網羅する必要はない．表 2 に示すとおり，STEP2，7，STEP3，STEP8 で確認対象となっている洗練パターンの延数は，それぞれ 29，8，21 となっており量的にも十分である．

以上述べたことを前提にしたとしても，更に補足的な評価として洗練パターンの変換を網羅的に確認したい場合は，類似のパターンで代替的に確認できる．すなわち，STEP2，7 の制御不能駆動は類似のパターンであるモニタ不能駆動で代替確認でき．STEP3 では，同様に，ガード条件導入とモニタ不能駆動及び制御不能駆動はマイルストーン駆動で代替確認できる．また，STEP8 では，分割・統治はマイルストーン駆動で，また制御不能駆動はモニタ不能駆動で代替確認できる．

表 4　ATM システムユースケース図要素の適合，再現数
Table 4　Numbers of precise or recalled elements for modeled use case diagrams.

基準モデルに対して	基準モデル		被験者Aモデル		被験者Bモデル	
	UC	[Actor，UC]	UC	[Actor，UC]	UC	[Actor，UC]
適合数			11	18	11	18
非適合数			0	10	0	6
抽出要素計	8	13	11	28	11	24
適合率(%)			100	64.3	100	75.0
再現数			8	13	8	13
非再現数			0	0	0	0
基準モデル要素計	8	13	8	13	8	13
再現率(%)			100	100	100	100

注1. UC：ユースケース，[Actor，UC]：関連するActorとUCの組
注2. Transaction（口座取引処理実行）のUCは，Withdrawal（引出処理）等の各処理に分割した．

これまで述べたように，提案している QVT 変換規則による変換の評価は，この適用事例で十分である．

○　属人性の排除評価

ユースケース図の評価：　表 3 は，ユースケース図のユースケースとアクターについて，基準モデルと被験者 A，B モデルの変換結果を比較したものである．関連するユースケースとアクターは同じ行に並べて表示している．アクターの右側の ‘正’ の欄は，一つのアクターとユースケースの組が適合するか否かを示しており，‘○’ で囲った数字は適合している組の数である．表 3 に基づき計算した適合率と再現率を表 4 にまとめた．ユースケース（UC）については，被験者 A，B モデルともに，適合率，再現率は 100%である．一方，アクターとユースケースの組（[Actor，UC]）は，再現率はともに 100%だが，適合率については被験者 A モデルの 64.3%に対し被験者 B モデルは 75.0%と差が出ている．

[Actor，UC] の適合率について検証する．被験者 A，

B モデルとも，再現率が 100%であることから，アクターの冗長的な抽出が適合率悪化の原因であると考えられる．STEP2 の最後に環境エージェントの名称を手動で入力し，それがアクターにマッピングされる．このことから，この環境エージェント名称の手動入力が適合率悪化の根本的原因であり，唯一属人性が混入するところである．モデル変換における属人性の混入はないと言える．環境エージェント名称の付与規則を細かく規定することによって属人性の混入を抑制できると考えられる．(今後の研究課題)

イベントフロー図の評価： 表 5 は，イベントフロー図のイベント処理について，基準モデルと被験者 A，B の変換結果を比較したものである．基準モデルのイベント 3 及び 16 に記された×は，内部処理として記述されているため，イベント処理としては除外したことを示している．それに対して被験者 A，B は，銀行またはオペレータ間のイベント処理として新たに抽出している．表 5 に基づき計算した適合率と再現率を表 6 にまとめた．'△' は不適合として計算している．被験者 A，B モデルの適合率はそれぞれ 62.5%，78.6%であり，再現率はそれぞれ 76.9%，84.6%である．

表 5　ATM システムイベントフロー図作成結果の比較
Table 5　Modeled event flows diagrams compared with provided one.

	イベント	イベントフロー						
		サイト提供教材（基準モデル）		被験者Aモデル			被験者Bモデル		
		アクター	機能	アクター	機能	正	アクター	機能	正
1	スタートアップ実行	OperatorPanel	ATM	オペレータ	スタートアップ キースイッチ	○	オペレータ	スタートアップ （キースイッチON検出時）	○
2	初期現金セット	OperatorPanel ATM （額設定要求） （現金要求） CashDispenser		オペレータ	スタートアップ 現金投入機 （現金投入後，額自動設定）	△	オペレータ 銀行	スタートアップ （現金投入なし，額設定あり） （銀行との接続オープン後実行）	×
3	銀行との接続オープン	NetworkToBank	ATM （内部処理のため除外）	オペレータ	スタートアップ	△	銀行 オペレータ	スタートアップ	△
4	カード挿入・セッション 起動	CardReader	ATM Session						
5	カード読込	CardReader	Session	顧客 カード読取機	セッション （4を含むがセッション起動なし）	○	顧客	セッション （4を含むがセッション起動なし，カード読取は内部的に処理）	○
6	PIN読込	CustomerConsole	Session	顧客	セッション	○	顧客	セッション （PIN情報入力は内部的に処理）	○
7	銀行へカード・PIN情報送信			顧客 銀行	セッション	△			
8	占有使用開始	（5,6に暗黙的に含まれる）		顧客 銀行	セッション	○	顧客	セッション	○
9	取引処理選択	（トランザクションの導入部分；メニューからの処理選択）		顧客	セッション	○	顧客	セッション	○
10	トランザクション実行	Transa	Session -ction （7,9,11,12を含む）	セッション	トランザクション （7を再度実行）	×	セッション	トランザクション （7,11を含む）	○
11	レシート印刷				セッション	×			
12	取引継続／終了	（トランザクションの継続／セクション復帰部分；メニューからの選択）		顧客	セッション	○	顧客	セッション	○
13	カード排出	CardReader	Session	顧客	セッション	○	顧客	セッション	○
14	占有使用解除	（13に暗黙的に含まれる）		顧客	セッション	○	顧客	セッション	○
15	シャットダウン実行	OperatorPanel	ATM	オペレータ	シャットダウン キースイッチ （顧客待受け時有効）	○	オペレータ	シャットダウン （顧客待受け状態でキースイッチOFF検出時）	○
16	銀行との接続クローズ	NetworkToBank	ATM （内部処理のため除外）	オペレータ 銀行	シャットダウン （シャットダウンの中で実行）	△	オペレータ	シャットダウン （シャットダウンの中で実行）	△
17	シャットダウン無効	（15に暗黙的に含まれる）		オペレータ	シャットダウン キースイッチ （顧客使用時無効）	○	（15に含まれる）		○

注1. 複雑化を避けるため，細部を省略した概略フローを示している．
注2. 機能：ソフトウエア機能，○：適合，×：不適合，△：不適合だがシステム的には適合

被験者 A モデルの不適合等の内容を検証する．イベント 2（△）では，基準モデルは手動で現金額を設定しているのに対し，被験者 A モデルは現金投入後自動で設定している．イベント 3，16（△）は，基準モデルではイベント処理としては扱われていないものである．被験者 A モデルは，イベント 7 をトランザクションの中と重複して実行している．カード・PIN 情報を入力時点で銀行に送信することによって，いろいろなチェックに使用できるという拡張性を，要求ゴール分解作業の中で考慮したということである．このように，これら '△' は基準モデルと厳密には異なるが，システム的には有効であると言える．イベント 10，11（×）は，レシート印刷をトランザクションの中ではなくセッションで印刷しているので不適合とした．また，イベント 4 のセッション起動は再現されていない．

次に，被験者 B モデルの不適合等の内容を検証する．イベント 2（×）では現金投入が記述されていない．イベント 3，16（△）の不適合，及びイベント 4 の不再現については被験者 A モデルの場合と同様である．

上述した被験者 A，B モデルの基準モデルに対する不適合，不再現内容の発生原因を調べてみると，STEP6 での要求ゴールの分解結果や STEP7 でのイベントや環境エージェントの名称付与結果に原因がみ

表 6　ATM システムイベントフロー要素の適合，再現数
Table 6　Numbers of precise or recalled elements for modeled event flow diagrams.

基準モデルに対して	イベント処理		
	基準モデル	被験者Aモデル	被験者Bモデル
適合数		10	11
不適合数		6	3
イベント処理総数	13	16	14
適合率(%)		62.5	78.6
再現数		10	11
不再現数		3	2
再現対象数	13	13	13
再現率(%)		76.9	84.6

注. △は不適合としてカウントした.

られる．このことから，STEP7，8 のモデル変換における属人性の影響は排除されていることが確認された．STEP6 や STEP7（STEP2 と同じ作業）は，ゴールモデリングに対する熟練度や ATM システムのドメイン知識など属人的な要素に大きく影響される部分である．

しかし，これらの部分は逆にゴール指向モデリングの有効な特徴を発揮できる部分でもあり，‘△’ のイベント抽出にその効果が表れている．ここで，‘△’ を適合と見做すと，被験者 A，B モデルの適合率はそれぞれ 87.5%，92.9%，再現率は 84.6%，84.6%となって，被験者 B モデルの再現率は変化しないが，それ以外は大幅に改善する．このことから，STEP2（STEP7）と同様に，STEP6 のゴール分解についても，洗練パターンのシナリオに沿った分解規則を細かく規定することによって，属人性を更に抑制しつつゴール指向分析の有効性を効果的に追及できると考えられる．（今後の研究課題）

上述したユースケース図，イベントフロー図の評価結果から，STEP2，3，7，8 のモデル変換に起因する属人性は排除されていることが確認された．手動作業部分において混入する属人性については，STEP4，若しくは STEP5，8 終了後の部分的な見直しと修正で対処できると考える．

妥当性の脅威　何を正解モデルとするかが妥当性の脅威となる．実務面でのゴール指向要求分析手法の活用実績は少なく，提案する一連のモデル変換への入力と出力となる正解モデルを決定するのは難しい．本論文では，要求記述書から入力となるゴールモデルを作成し，その変換結果と既存のオブジェクト指向設計によるユースケースモデルを基準として比較した．更に，その差を属人性の影響として評価した．この入力と出力の正解モデルの内容は，各モデル変換の妥当性評価への影響は少ないと思われるが，最終変換結果であるユースケース図やイベントフロー図の評価には影響を与える．本論文では，その影響をできるだけ緩和できるように判定条件を考慮した．

6. 関連研究

KAOS への洗練パターン適用に関する研究として，Darimont ら [18] は，時相論理で定義した KAOS ゴールの一般的な洗練パターンと，それを操作可能とするイベントによる刺激—応答に基づく操作パターンを提案している．この洗練パターンは，本論文の提案アプローチでも採用しているが，ゴールモデル作成者の負担となるため形式手法による記述は使っていない．

ゴール指向からオブジェクト指向開発への展開については多くの研究がなされている．例えば Lamsweerde [9] は，ゴールモデルから要求ゴールを操作可能とする操作モデルを導出し，そのモデル中に定義されたそれぞれの操作と環境エージェント（当該操作の入/出力オブジェクトを制御/モニタする）を関連付けることによってユースケース図を定義している．操作モデルの作成は事前/事後条件の特定等経験則によるところが大きく，その経験則はそのままユースケース間の関連に反映される．

Robinson and Elofson [19] は，UML による既存手法と既存のゴール指向要求分析手法を統合し，ゴールから UML による設計仕様を導くアプローチを定義している．サブゴールまたはサブ要求ゴールをユースケースステップとして捉え，ゴール階層を生成するのに洗練パターンをよく使うと説明しているが，ゴール階層からユースケースを導出する手順の明記はなく，サブゴール（サブ要求ゴール）をユースケースステップに対応させるという説明に止まっている．

これらのプロセスではゴールからユースケースを経験則によって導いているが，本論文の提案アプローチでは洗練パターンを仲介したアルゴリズムによる変換アプローチを定義し，経験則をできるだけ排したより負荷の少ない一般的なアプローチとなっている．

[20] は，KAOS モデルをユースケースモデルを介してロバストネス図に変換するアプローチを説明している．また，[21] は，KAOS モデルからユースケースモデルへの変換について，その考え方を具体例を示しながら説明している．本論文は，それらを発展させ，KAOS モデルからユースケースモデルへの変換を具体的なステップに分け，より明確なアルゴリズムでの半自動による変換アプローチを提案している．

7. むすび

本論文では，KAOS モデルからユースケースモデルへの変換アプローチを提案した．提案アプローチでは，KAOS モデルを洗練して抽出した要求ゴールを操作モデルを介してユースケースにマッピングし，洗練パターンによる KAOS の階層構造をユースケース間の関連に反映させる．同様に，要求ゴールを洗練パターンで分解してイベントを抽出し，ユースケースのイベントフロー図に変換する．全体的には半自動的な変換

であるが，モデル変換の部分は QVT による変換規則に従い機械的に実施できる．提案アプローチによる一連のモデル変換は，洗練パターンの意味（シナリオ）を継承するとともに，属人性を排除する効果がある．

ICONIX ではロバストネス分析を介してシーケンス図を導出し，クラス間のインタラクションを規定する．ロバストネス図はユースケースの振舞いをオブジェクトの絵として表現し設計への橋渡しとなるので，ゴールモデル洗練パターンによる変換ルールにおいて，ロバストネス図も変換対象に加えることは有効と考える．研究の範囲を広げていきたい．

また，本論文では，米国の銀行 ATM システムを事例として適用評価を行い，提案アプローチの効果的な適用を確認した．適用結果のユースケースモデルは，基準モデルに対する適合率，再現率ともにほぼ妥当な値であり，提案アプローチの有効性を示している．適合率や再現率のマイナス要因は，手動によるゴールモデリング結果に由来しており，被験者の知識や熟練度に起因している．これらゴールモデリングの実施要領を詳細に規定すれば改善できると思われるが，今後の研究課題である．

今回の評価は一例に関するものであり，今後，その他の事例についても評価していきたい．提案アプローチにおいて，導出したユースケース図と KAOS モデルとの相互洗練は手動で実施したが，これをある程度定型化することも今後の研究課題である．

謝辞　本研究は JSPS 科研費 24300005，26330081，26870201 の助成を受けたものです．本論文では，KAOS ツールとして国立情報学研究所 GRACE センター所有の「K-tool」を使用させて頂きました．当ツールの利用に関してご協力を賜りました，国立情報学研究所 GRACE センター長/東京大学本位田真一教授を始め関係者の方々に深く感謝致します．

文　献

[1] IEEE，ACM，松本吉弘（訳），ソフトウエアエンジニアリング基礎知識体系：SWEBOK2004，オーム社，2005.

[2] A. Lamsweerde, “Goal-oriented requirements engineering: A guided tour,” RE’01, pp.249–263, 2001.

[3] E. Yu, “i* an agent- and goal-oriented modelling framework.” http://www.cs.toronto.edu/km/istar/

[4] L. Chung and J.C.S. doPrado Leite, “On non-functional requirements in software engineering,” Mylopoulos Festschrift LNCS 5600, pp.363–379, Springer-Verlag, 2009.

[5] OMG, “Unified modeling language.” http://www.uml.org/

[6] ダグ・ローゼンバーグ (著)，マット・ステファン (著)，三河淳一 (監訳)，船木健児 (翻訳)，佐藤竜一 (翻訳)，ユースケース駆動開発実践ガイド，翔泳社，2007.

[7] M. Stephens and D. Rosenberg, “Iconix process.” http://iconixprocess.com/iconix-process/

[8] IBM, “IBM rational unified process (rup).” http://www-01.ibm.com/software/rational/rup/

[9] A. Lamsweerde, Requirements Engineering: From System Goals to UML models to Software Specification, WILEY West Sussex England, 2009.

[10] E. Letier, Reasoning about Agents in Goal-Oriented Requirements Engineering, Phd Thesis 2001, pp.1–68, 2001.

[11] A. Lamsweerde, “From system goals to software architecture,” LNCS2804, pp.25–43, Springer-Verlag, 2003.

[12] E. Letier and van Lamsweerde, “Reasoning about partial goal satisfaction for requirements and design engineering,” SIGSOFT ’04/FSE-12, vol.29, no.6, pp.53–62, ACM SIGSOFT Software Engineering Notes, 2004.

[13] OMG, “Documents associated with Meta Object Facility (MOF) 2.0 Query/View/Transformation, v1.0.” http://www.omg.org/spec/QVT/1.2/

[14] W. Heaven and A. Finkelstein, “UML profile to support requirements engineering with KAOS,” IEE Proc-Softw, vol.151, pp.10–27, 2004.

[15] OMG, “Documents associated with uml v2.4.1.” http://www.omg.org/spec/UML/2.4.1/

[16] C. Li, L. Dou, and Z. Yang, “A metamodeling level transformation from UML sequence diagrams to Coq,” ICTCS 2014, pp.147–157, CEUR, 2014.

[17] R.C. Bjork, “Atm simulation links - by topic.” http://www.cs.gordon.edu/courses/cs211/ATMExample/index.html

[18] R. Darimont and vanLamsweerde, “Formal refinement patterns for goal-driven requirements elaboration,” SIGSOFT’96, pp.179–190, ACM SIGSOFT Software Engineering Notes, 1996.

[19] W.N. Robinson and G. Elofson, “Goal directed analysis with use cases,” J. Object Technology, vol.3, no.5, pp.125–142, 2004.

[20] K. Honda, H. Nakagawa, Y. Tahara, and A. Ohsuga, “Goal-oriented robustness analysis,” Proc. Tenth JCKBSE, pp.171–180, IOS Press, 2012.

[21] 本田耕三，中川博之，田原康之，大須賀昭彦，“洗練パターンによるゴール指向ユースケースモデリング,” SES2014, pp.45–50, IPSJ/SIGSE, 2014.

（平成 27 年 6 月 1 日受付，10 月 6 日再受付，
12 月 3 日早期公開）

本田　耕三

1953 年生．1976 年九州大学工学部電気工学科卒業．同年日本電気（株）に入社．2011 年電気通信大学大学院情報システム学研究科博士前期課程修了，現在，電気通信大学大学院情報システム学研究科博士後期課程に在学．情報処理学会学生会員．

平山　秀昭　（正員）

1958 年生．1981 年慶應義塾大学工学部管理工学科卒．同年（株）東芝入社．2001 年電気通信大学大学院情報システム学研究科博士後期課程了．2003 年より東芝ソリューション（株），2014 年より電気通信大学協力研究員，2015 年より東京電機大学非常勤講師，2015 年より東芝ソリューション販売（株）に所属．博士（工学）（電気通信大学）．並列分散処理，ソフトウェア工学等の研究に従事．情報処理学会，電子情報通信学会会員．

中川　博之　（正員）

1974 年生．1997 年大阪大学基礎工学部情報工学科卒業．同年鹿島建設（株）に入社．2007 年東京大学大学院情報理工学系研究科修士課程修了，2008 年同大学院博士課程中退．同年電気通信大学助教，2014 年大阪大学大学院情報科学研究科准教授，現在に至る．工学博士（早稲田大学）．要求工学，形式手法，エージェント及び自己適応システム開発手法の研究に従事．情報処理学会，電子情報通信学会，IEEE CS 各会員．

田原　康之

1966 年生．1991 年東京大学大学院理学系研究科数学専攻修士課程修了．同年（株）東芝入社．1993～1996 年情報処理振興事業協会に出向．1996～1997 年英国 City 大学客員研究員．1997～1998 年英国 Imperial College 客員研究員．2003 年国立情報学研究所入所．2008 年より電気通信大学准教授．博士（情報科学）（早稲田大学）．エージェント技術，及びソフトウェア工学などの研究に従事．情報処理学会，日本ソフトウェア科学会会員．

大須賀昭彦　（正員）

1958 年生．1981 年上智大学理工学部数学科卒．同年（株）東芝入社．同社研究開発センター，ソフトウェア技術センター等に所属．1985～1989 年（財）新世代コンピュータ技術開発機構（ICOT）出向．2007 年より，電気通信大学大学院情報システム学研究科教授．2012 年より，国立情報学研究所客員教授兼任．工学博士（早稲田大学）．主としてソフトウェアのためのフォーマルメソッド，エージェント技術の研究に従事．1986 年度情報処理学会論文賞受賞．IEEE ComputerSociety Japan Chapter Chair，人工知能学会理事，日本ソフトウェア科学会理事を歴任．情報処理学会，電子情報通信学会，人工知能学会，日本ソフトウェア科学会，IEEE Computer Society 各会員．