IMPULSE：KAOSを利用したマルチエージェントシステムの分析モデル構築

中川博之[†,††] 吉岡信和[†††] 本位田真一[†††,†]

近年，高度で複雑化するソフトウェアを実現する手段としてマルチエージェントシステムが注目されている．マルチエージェントシステムは複数のエージェントを構成要素として持つことから，一般のソフトウェアと比較してシステム分析・設計も難しく，多くのモデル構築方法論が提案されている．しかしながら，いずれの既存方法論も要求モデルとの乖離が大きいために分析モデルの構築が困難であり，結果，システム構築の大きな障壁となっている．そこで本論文では，要求工学の分野で成果をあげているゴール指向要求分析法 KAOS を拡張利用した分析モデル構築法 IMPULSE を提案する．IMPULSE を用いることで，要求モデルを利用した分析モデル構築プロセスが明確化され，分析モデル構築の難しさが解消される．

IMPULSE: Analysis of Multi-agent Systems Based on KAOS Modeling

Hiroyuki Nakagawa,[†,††] Nobukazu Yoshioka[†††] and Shinichi Honiden[†††,†]

Agent technology offers a solution for producing complex software systems characterized by autonomous behavior and a high degree of distribution, however, development of multi-agent systems (MAS) needs a more feasible methodology for requirements analysis because of the difficulty in generating an analysis model. The purpose of this study is to reduce the gap between the requirement analysis and analysis phases of developing multi-agent systems. This paper describes the IMPULSE methodology, which utilizes the KAOS, goal-oriented analysis method as a requirement analysis method, and provides developers with a tool for model translation into an analysis model of multi-agent systems. This paper also shows the result of evaluating IMPULSE through analysis experiments. IMPULSE enables simple and effective development of multi-agent systems.

1. はじめに

近年，高度で複雑化するソフトウェアの実現手段としてマルチエージェントシステム[1]（Multi-Agent Systems：以下 MAS）が注目されている．MAS は，自律性を有したエージェントを構成要素とし，エージェント間の協調により機能を実現するシステムである．MAS の開発[2)~4)]は通常のソフトウェア開発プロセスと同様，要求・分析・設計・実装・テストフェーズにより構成されているが，現状要求フェーズと分析フェーズとを十分に関連付ける方法論が確立されていないことにより，分析フェーズにおけるモデル構築が困難となっている．その結果，要求モデルと分析モデルとの間に矛盾が生じたり，分析フェーズで要求フェーズと同様の活動が必要となったりする場合が多い．

一方で，ソフトウェア工学の分野において要求フェーズを対象とした要求工学（Requirements Engineering）[5),6)]が近年特に注目されている．これはソフトウェアが複雑化する現在において，ソフトウェア要求の無視あるいは管理不足による開発コストの増加や開発スケジュールの遅延，開発プロジェクトの失敗件数が急増していることによるものである．この要求工学の分野においては，現在 KAOS[7),8)]，i*[9)] に代表されるゴール指向要求分析法が成果をあげている．

そこで本研究では，MAS 開発における分析モデル構築の難しさを軽減するために，ゴール指向要求分析法 KAOS を拡張利用した分析モデル構築法 IMPULSE☆を提案する．本研究においては，特に問題を具象化するために MAS の分析・設計方法論として

† 東京大学
The University of Tokyo
†† 鹿島建設株式会社
Kajima Corporation
††† 国立情報学研究所
National Institute of Infomatics

☆ IMPULSE: Integrated Modeling Process Utilizing Logically Structured Elements の略．

広く知られる Gaia[10),11)] を利用した開発プロセスを想定し，KAOS を拡張利用した MAS 要求分析法と，拡張 KAOS モデルから Gaia 分析モデルへのモデル変換手法について提案する．

本論文は以下のような構成となっている．まず 2 章で MAS 分析の難しさと KAOS の特徴について述べる．続く 3 章では，提案する分析モデル構築法 IMPULSE を利用した開発プロセスとして，拡張 KAOS を利用した MAS の要求分析手法と Gaia モデルへのモデル変換手法を示す．さらに 4 章では MAS に対する分析モデルの構築実験結果を示し，本手法の有効性を考察する．5 章で関連研究について言及した後，6 章でまとめを述べる．

2. MAS 開発における分析モデル構築

本章では，まず MAS における分析モデル構築の難しさを Gaia による分析を例に説明し，難しさに対する既存方法論の現状を示す．続いて本研究で要求分析法として拡張利用する KAOS について述べる．

2.1 MAS 分析の難しさ

現在 MAS 開発プロセスの分析フェーズをサポートする方法論として，Gaia[10),11)]，ROADMAP[12),13)]，Tropos[14)]，Prometheus[15)]，MaSE[16)] など様々な方法論が提案されている．この中でも特に Gaia は，分析・設計フェーズを対象とした MAS 開発方法論であり，ROADMAP など，多くの MAS 方法論が基礎としている広く知られた方法論である．本研究においても MAS の標準的な分析モデルの獲得を目指すことから，Gaia 方法論に従った開発プロセスを想定する．

Gaia は**図 1** に示す概念を分析・設計モデルの記述要素として持つ．ここで，エージェントとは自身の目的を達成するために自律的に動作・意思決定するソフトウェアであり，MAS の中心的な構成要素である．また，ロールとはエージェントが目的の一部を達成するために MAS 内において一時的に担う役割を指す．Gaia の分析フェーズでは，ロールモデル，インタラクションモデル，組織モデル，環境モデルの 4 つのモデルに対して分析を実施する．**ロールモデル**は後継の設計フェーズにおいてエージェントモデルの基礎となるものであり，**インタラクションモデル**はロール間のインタラクション（情報伝達）を定義するモデルである．また，**組織モデル**は複数のロールにより構成される組織を定義するためのモデルであり，分析フェーズではシステム内に存在するサブ組織を抽出し，各組織における制約（組織ルール）を定義するために用いられる．さらに，**環境モデル**は MAS が情報を参照あるいは変更可能なオブジェクトや，モニタリングの対象となるエージェントを環境リソースとして定義するモデルである．以降，本論文ではオブジェクトとは受動的なデータを抽象化したものを，エージェントとは能動的な動作を抽象化した自律的なソフトウェアコンポーネントを指す．

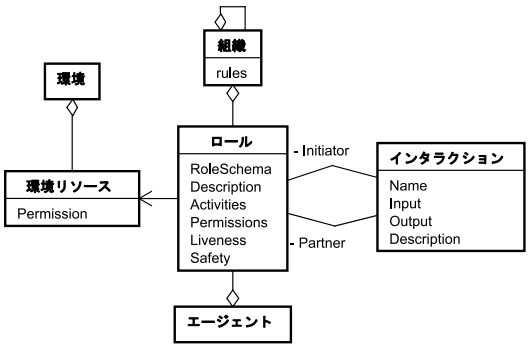

図 1 Gaia のモデル記述要素
Fig. 1 Model description elements in Gaia.

Gaia による MAS 開発では，開発者は分析フェーズにおいてこの 4 種類のモデルを構築する必要があるが，次にあげる 3 つの要因により分析モデルの構築が困難となっている．

2.1.1 ロールの抽出

MAS におけるロールとはシステム構成要素のエージェントが組織内で果たすべき役割であり，MAS 分析モデルを構築するうえでも基盤となる重要なモデル構成要素である．Gaia も他の MAS 開発方法論と同様に，ロールの抽出から分析フェーズの活動が開始するが，前フェーズの要求フェーズにおいて得られるシステム要件は，通常ロールに割り当てられる責務のレベルではない．したがって，これらのシステム要件を直接ロールに割り当てようとすると，複数の要件に重複して含まれている同一責務に対して 2 つ以上のロールが同時に割り当てられたり，単一ロールに複数の責務が割り当てられたりする可能性がある．MAS 開発におけるロール抽出はオブジェクト指向開発におけるオブジェクト抽出とも類似するが，オブジェクト指向設計原則の 1 つである単一責任の原則[17)] でも言及されているとおり，ロールの抽出に関しても MAS 開発者はシステム要件から責務を抽出し，それらを各ロールに重複なく割り当てなければならない．

たとえば，各エリアに配備されたセンサのセンシング情報をもとに対象を追跡するトラッキングシステムを分析対象として考えた場合，「センサ情報から対象を追跡できる」というシステム要件からは，「対象を追尾する」や「各センサの情報を管理する」という責務，

さらにセンサ数が膨大である場合には「センサ情報を集約する」といった責務が抽出される必要がある．また,「センサ情報を利用できる」というシステム要件に関しても,「センシング情報を検知する」といった一時利用的な責務だけでなく，場合によっては「過去のセンサ情報から現在地を推測する」や,「過去のセンサ検知結果を永続化する」といった責務が必要になる．開発者はこれらの責務をシステム要件・制約から判断し，抽出したそれぞれの責務を重複がないよう整理したうえで，システムに必要なロールを抽出しなければならない．

2.1.2 ロールモデルの構築

ロールが抽出されると，Gaia では各ロールに対してロールモデルを構築する．**表 1** は Gaia のロールモデルを表したものであるが，本モデルにはロール単体で実行可能な処理を表す**アクティビティ**やロール間の情報伝達手段である**インタラクション**（分析フェーズではプロトコルとも呼ばれる），環境リソースに対しての権限（Permissions），アクティビティとインタラクションの列により構成される実行系列（Liveness），ロールがつねに満たさなければならない制約（Safety）などが含まれる．しかしながら，特にロールの実行オペレーション系列（Gaia における Liveness 属性）や制約（Gaia における Safety 属性）の定義は，ドメイン専門家には記述が難しい一方で，開発者にはロール単位での責務・振舞いの正確な把握が要求される[13]など，従来の方法論ではロールモデルの構築に高度な分析能力とドメイン知識が要求される．

たとえば，トラッキングシステムにおいて追跡対象を追尾するトラッカロールに関しては，センサ情報により追跡を追尾する通常プロセスだけでなく，センサ情報から対象の場所が判断できない場合などの異常時に対応する実行系列も Liveness 属性に記述する必要がある．また Safety 属性に関しても，要件記述からシステム内でトラッカロールに求められている制約を判断し☆，その制約を評価可能な評価基準を検討したうえで，制約条件を決定しなければならない．

2.1.3 組織制約条件の抽出

MAS は複数のエージェントにより構成されるため，システム全体の動作を保証するには各ロールの制約だけでなく，システム内に組織を定義し，各組織に対しても制約を付与する必要がある．特にオープンな環境では，安全性を保証するために各エージェントの動作を制約する必要があり，厳しい組織の制約（組織ルール）を設定しなければならない．そのために Gaia 分析では必要に応じてシステム内にサブ組織を定義し，各組織に対して制約条件である組織ルールを課す．組織ルールは，組織構造などの静的な制約を表すルールと，実行順序のような動的な制約を表すルールの大きく 2 つに分類することができるが，いずれも抽出にあたっては事前に適切な情報伝達手段を決定したうえで，各局面において各ロールがどのように組織に関与しているかを明確に把握しておく必要がある．

表 1 Gaia ロールモデルの構造
Table 1 Structure of the role model in Gaia.

属性	記述すべき内容
Role Schema	ロールの名称 例）トラッカ
Description	ロールの概要
Protocols and Activities	ロールが実行するアクティビティとインタラクション 例）**Protocols**：追跡を依頼，センサ情報送信を依頼，… **Activities**：追跡を開始，ゾーンへ移動，…
Permissions	環境リソースに対するアクセス権 例）**changes** 移動履歴ログ
Liveness	ロールの実行オペレーション系列 例）追跡を依頼．追跡を開始．センサ情報送信を依頼．ゾーンへ移動，…
Safety	つねに満たすべき状態（制約条件） 例）トラッカーが存在するゾーン ＝ 追跡対象が存在するゾーン

たとえばトラッキングシステムでは，追跡プロセスを実行するためのサブ組織がシステム内に存在すると考えられる．追跡プロセスでは特にセンサ情報が重要な情報と想定されるが，監視業務の性質上，どのセンサ情報をどこで管理するか，どのような形で情報を伝達すべきかを把握しておかなければ，組織構造や組織の動的な振舞いを把握することができず，組織に適した制約を課すことはできない．

これらの難しさは Gaia 分析に限定されるものではなく，MAS の分析プロセス全般にあてはまる難しさであり，MAS の規模が大きくなるにつれて急激に顕在化する．たとえば MAS の適用が期待されている Web Services[18]やユビキタス環境では，サービスが多様化し，多くのサービス提供エージェントと多様なサービス統合エージェントが登場すると考えられる．このようなシステムを MAS で構築する場合，まず登場するエージェントが多様化するため，それらが担うロールも今まで以上に多様化すると考えられる．また，このようなシステムでは多数のロールが多数の局面で関与することとなり，システムの規模が大きくなるだけ

☆ たとえばトラッキングの精度などが考えられる．

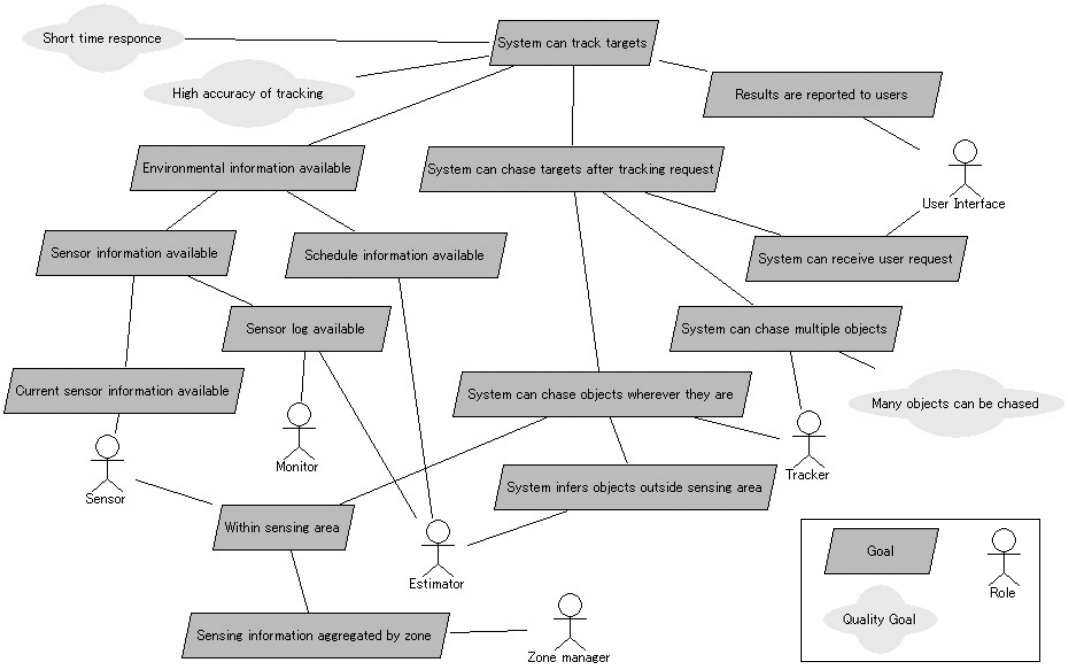

図 2 ROADMAP の要求モデル
Fig. 2 Requirements model in ROADMAP.

でなく，ロール間で構成・定義される組織数やインタラクション数も急激に増加する．さらに，これらの分野はオープンな環境が想定されるため，セキュリティとシステムの安全性を保証する観点からも組織に対して厳しい制約が求められる．したがって，MAS 分析モデル構築の難しさを解消し，開発者を支援する分析モデル構築方法論の早期確立が必要とされている．そこで本研究では，特に分析フェーズの前プロセスにあたる要求フェーズに注目し，フェーズ間のギャップを解決することで分析モデル構築の難しさを解消する．

2.2 既存 MAS 開発方法論による要求分析

要求フェーズの活動も支援するように Gaia を拡張した既存の MAS 開発方法論として，ROADMAP [12),13)] があげられる．**図 2** は ROADMAP における要求モデルの一例であるが，ROADMAP はシステムに対する要件を明示的に記述し，それを分解・洗練化することでシステムが達成すべきゴールを実現方法へと変換する**ゴール指向要求分析法**[6)] をモデル構築に利用している．ゴール指向要求分析法を用いることにより，システム要件をロールに割り当てるべき責務のレベルにまで分解することが容易になり，得られた責務集合からシステムに必要なロールを抽出することが可能となる．したがって，ゴール指向要求分析法は MAS 分析の難しさの 1 つであるロール抽出の難しさを解消する有効な手法であるといえる．

しかしながら，ROADMAP にはゴールの記述粒度に関する制約がなく，また同一ゴールに対して複数ロールを割り当てることも可能であるため，各ロールの責務を厳密に区分することができない．また，ゴールやゴール間の関係はシステムが達成すべき状態やロールが果たすべき責務を記述しているにすぎず，各ロールの振舞いを把握することは難しい．その結果，ROADMAP ではシステムに必要なロールを抽出することは可能であるが，ロールモデルの構築は依然として困難であるといえる．また，モデルからシステムの各局面（シナリオ）における振舞いを想起することも難しく，組織制約条件の抽出に関しても ROADMAP の要求モデルは有効であるとはいえない．以上の観点から，ROADMAP は要求フェーズを包含している方法論ではあるが，分析フェーズとの関連を考慮すると，2.1 節であげた MAS 分析の「ロールモデルの構築」，「組織制約条件の抽出」に関する難しさを本質的に解決しているとはいえない．

2.3 KAOS による要求分析

ここで，MAS 分析に有益な要求モデルを構築するために，要求工学の分野で広く知られている要求分析法 KAOS に着目する．KAOS はゴール指向要求分析法の 1 つであり，まずゴールモデルにおいてシステ

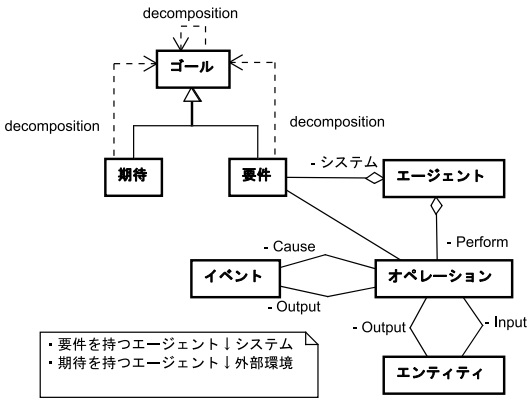

図 3 KAOS のモデル記述要素
Fig. 3 Model description elements in KAOS.

ムが目標とするゴールからシステム要件を抽出する．ゴールには機能的な要求を表現するハードゴール（以下，特に断らない限りゴールとはハードゴールを指す）と性能や信頼性，ユーザビリティなどの非機能要求[19]を表現するソフトゴールの 2 種類があるが，ゴールはゴールツリーにより，システムが実現すべき**要件**と，ユーザ・他システムといった外部環境が実現すべき**期待**に分割される．続いて責任モデルでこれらに対して責任を持つ主体である**エージェント**☆が抽出され，要件を実現するための**オペレーション**が定義される．最後にオペレーションモデルにおいて，オペレーションとオペレーション間を関連付ける**イベント**，さらに環境の構成要素である**エンティティ**から，シナリオを表現するオペレーション系列が定義される．KAOS モデルの要素間関係を**図 3** に示す．

このように KAOS は，複数のモデルと複数の記述要素を利用することで系統立った分析が可能な要求分析法である．また，KAOS は要求フェーズだけでなく，オペレーションの定義やシナリオ記述といった分析フェーズの活動もサポートしているため，オペレーションやシナリオといったシステムの動的な側面を記述することができる分析手法である．

ただし，KAOS は通常のソフトウェアに対する要求分析法として提案されているため，システムに対する要求と外部環境に対する期待とを分離することを分析の第 1 目的としている．したがって，アクタの記述レベルがシステムと外部環境といった粒度であり，MAS の構成要素であるロールとは記述の粒度が異なる．また，同様の理由によりシステム内部のインタラクションや組織構造を記述することができない．その結果，MAS に対して KAOS 分析を実施したとしてもモデル要素の記述粒度が異なるため，得られた要求モデルと MAS 分析モデルとの間には依然としてギャップが残る．

3. MAS 分析モデル構築法 IMPULSE

2 章であげた MAS 分析の難しさを解消するため，本研究では MAS 分析モデル構築法 IMPULSE を提案する．IMPULSE においては，KAOS を MAS 分析モデルと関連付けるよう拡張することで，従来のフェーズ間での独立したモデル構築から，モデルの重ね合わせによるモデル変換へと構築手段をパラダイムシフトさせる．その結果，MAS 分析モデルと要求モデルとの乖離が解消され，モデル構築の難しさが解決される．以降，本章では提案手法 IMPULSE について述べる．

3.1 IMPULSE による MAS 開発プロセス

IMPULSE による MAS 開発プロセスを**図 4** に示す．開発者はまず，本研究で提案する拡張 KAOS モデリングを利用してシステムに対する要求を分析する．KAOS はソフトウェアの分析モデルを記述するためのモデル要素を包含しているが，本研究においては KAOS のモデル記述に制約を設けることで，KAOS 要素と MAS 分析モデル要素とを対応付ける．この拡張により，拡張 KAOS モデルと Gaia モデル間に対応関係を定義することが可能となる．

続いて，要求分析結果である拡張 KAOS モデルは，本研究で提案するモデル変換手法により Gaia 分析モデルスキーマに変換される．このモデル変換手法はツールにより自動化されているため，モデルスキーマは拡張 KAOS モデルから自動的に獲得することができる．生成されたスキーマはロールやインタラクション，サブ組織，環境リソースといった Gaia 分析モデルを構成するモデルのインスタンスと，各インスタンスの多くの属性情報を含んだものであり，開発者は拡張 KAOS モデルを参照しながらスキーマに分析結果を追加することで Gaia 分析モデルを構築することができる．

3.2 拡張 KAOS における要求分析

対象が MAS であっても，要求フェーズにおいては通常システムと同様の手順で KAOS 分析が可能である．これは要求フェーズがシステム開発の初期段階であり，システムに対する要求とそれを実現するために必要な機能の抽出が活動の主目的であることによる．しかし分析フェーズで構築するモデルとの対応を考えた場合，2.3 節で述べたとおり，従来の KAOS モデ

☆ KAOS ではエージェントという用語は，システムやユーザなどオペレーションを実行可能な動作主体を表す意味で用いられる．

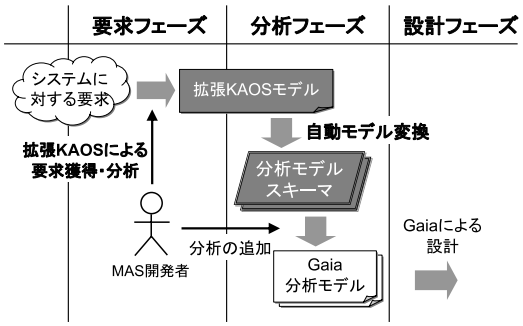

図 4 IMPULSE による MAS 開発プロセス
Fig. 4 Multi-agent systems development based on IMPULSE.

ルと MAS 分析モデルとでは要素が指す概念が異なるため，MAS 分析に必要なロールや MAS 内部の動的な振舞いを正確に記述することができない．そこで IMPULSE では，KAOS モデルの記述に以下の制約と分析の終了条件を定義することで KAOS を拡張し，KAOS・Gaia モデル間に対応関係を構築する．

拡張 KAOS モデリングにおける制約と終了条件

- **制約 1：**要件はロールに対する責務のレベルにまで分解し，(KAOS の）エージェントはシステム単位ではなく，ロール単位で記述する．
- **制約 2：**オペレーションはロールが単独で実行可能なアクティビティレベルで記述する．
- **制約 3：**オペレーションモデルはシナリオ単位で作成する．
- **終了条件：**ゴールモデルの葉の要素となるすべての要件，期待，ソフトゴールがそれぞれ 1 つのエージェント（制約 1 によりロール・外部環境）に割り当てられ，かつ対応するオペレーションが定義されている．

拡張 KAOS のゴール分析は従来 KAOS と同様にシステムに対する漠然とした要求に対しての分析から開始するが，従来 KAOS ではシステム要件を抽出することを分析の終了条件としているのに対し，拡張 KAOS では制約 1 によりロールの責務の粒度にまでゴール分析を継続する．また制約 1 により，システム要件がロールの責務にまで分解され，責務を割り当てられた KAOS 上のエージェントが Gaia 上のロールと対応付けられることとなる．つまり，従来の KAOS でエージェントと対応付けられているオペレーションや要件が拡張 KAOS においてはロールに対応付けられ，MAS 分析時におけるロールモデルの情報抽出に拡張 KAOS のエージェントが利用できることとなる．

制約 2 は拡張 KAOS モデルのオペレーションと Gaia モデルのアクティビティとを対応付けるものであるが，同時に本制約により，拡張 KAOS モデルにおいてオペレーション間を結ぶイベントは Gaia における環境イベントかロール間のインタラクションに対応することとなる．これらはメッセージ送受信者により分類可能であり，制約 1 とあわせると，拡張 KAOS モデルからインタラクションも形式的に抽出できることとなる．したがって，モデル記述に制約 1，2 を与えることにより，拡張 KAOS モデルから Gaia，つまり MAS の主要構成要素であるロールとインタラクションを形式的に抽出することが可能となる．

さらに制約 3 により，拡張 KAOS のオペレーションモデルと MAS における組織活動を対応付ける．シナリオを構成するアクタにより組織が形成されることが多いが，本制約により，各オペレーションモデルごとに登場するロールが何らかの組織を形成することとなり，モデル内のロール集合をサブ組織情報として抽出することができるようになる．また，組織制約条件である組織ルールに関しても，対応する各オペレーションモデルを参照することで検討が可能となる．

一方で終了条件は，拡張 KAOS 分析により定義された要件・オペレーション（Gaia でのアクティビティ）がロールに確実に継承されていることを保証するものである．この終了条件により，拡張 KAOS モデルから各ロールの責務と振舞いの抽出が可能となり，各ロールの責務に関する明確な把握が可能となるだけでなく，ロールモデルにおける Liveness 属性の形式的な抽出も可能となる．また，制約 3 とあわせることで，システム要件を実現するための想定されるすべてのシナリオがオペレーションモデル上に記述されることとなる．開発者はこれらのオペレーションモデルを参照することで，各組織におけるロールの振舞いから制約条件を検討することが可能となり，結果として組織制約条件の抽出が容易になる．

以上の制約と終了条件を定めることで，KAOS・Gaia モデル間に**表 2** の対応関係が定義され，Gaia モデルへの形式的モデル変換が可能となる．

3.3 Gaia 分析モデルへの変換

Gaia の分析フェーズにおいて構築すべきモデルは，ロールモデル，インタラクションモデル，組織モデル，環境モデルである．IMPULSE ではこのうち，ロール，インタラクション，サブ組織，環境リソースの各インスタンスを拡張 KAOS モデルから Gaia 分析モデルのスキーマとして形式的に獲得する．IMPULSE では，

(1) **ロールの判別**
(2) **インタラクションモデルの抽出**

表 2 拡張 KAOS と Gaia のモデル間対応関係
Table 2 Correspondence between expanded KAOS and Gaia models.

拡張 KAOS	Gaia
エージェント	・ロール ・外部環境（ユーザ・他システム）
要件	・ロールが達成すべき状態
期待	・外部環境が達成すべき状態
オペレーション	・アクティビティ（ただし，後続イベントと同一名の場合は結合してインタラクションとする） ・外部環境の処理内容
イベント	・インタラクション ・システム・環境間のイベント
エンティティ	・環境リソース
オペレーションモデル	・各シナリオ（サブ組織の構成単位）

(3) **組織モデルの抽出**
(4) **環境モデルの抽出**
(5) **ロールモデルの抽出**

の順序で Gaia 分析モデルへのモデル変換を実現する．本モデル変換のアルゴリズムを付録 A.1 に記載する．

以下本節では，IMPULSE における Gaia 分析モデルへの変換法を例をあげて説明する．**図 5** は，会議日程調整システムに対して拡張 KAOS 分析を実施し，その際に構築されたオペレーションモデルの 1 つである．本モデルは会議開催日時を決定するシナリオを記述したものであり，コーディネータロールは外部環境である議長の提示する会議条件に従い，各参加者のスケジュールを個別管理するスケジュールマネージャロールに会議参加者のスケジュールを問い合わせたうえで会議日時を決定する．オペレーションモデルは拡張 KAOS 分析における最終モデルであり，IMPULSE では Gaia モデル構築のための多くの情報をこのモデルから抽出する．

図 5 の拡張 KAOS モデルは，以下の点で通常の KAOS モデルと異なる．まず，3.2 節の制約 1 に従い，（KAOS における）エージェントは会議日程調整システムとしてではなく，システムを構成するロールの単位で記述されている．また制約 2 に従い，オペレーションも「会議開催日時を調整」といった複数のロールにより実現可能な処理単位ではなく,「スケジュールをチェック」や「参加者スケジュールを集計」などの各ロールが単体で実行可能な単位で記述されている．さらに制約 3 に定められるとおり,「会議開催日時決定」という 1 つのシナリオが 1 つのオペレーションモデルに記述されている．

3.3.1 ロールの判別

IMPULSE ではまず拡張 KAOS の責任モデルを参照し，割り当てられている要素が要件であるか期待であるかにより，拡張 KAOS 上のエージェントをロールと外部環境とに分離する．ここで，拡張 KAOS における責任モデルは従来 KAOS 同様に，各要件に対して責任を持つエージェントを割り当ててオペレーションを定義するモデルであるが，3.2 節の制約 1，2 により要件，エージェントとオペレーションの記述粒度が従来 KAOS とは異なる．この判別プロセスは，従来 KAOS でシステムと外部環境とを分離するプロセスと同様のものである．本論文では紙面の都合で責任モデルを記載していないが，図 5 の例では議長は期待を割り当てられた外部環境であり，コーディネータとスケジュールマネージャは要件を割り当てられたロールであると判別される．

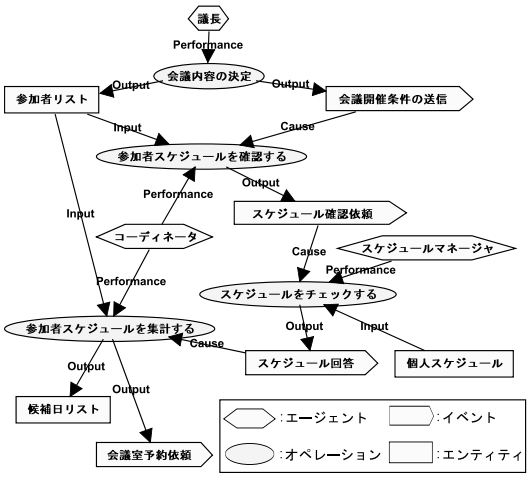

図 5 会議開催日時決定プロセスを表現した拡張 KAOS オペレーションモデル
Fig. 5 Operation model for deciding the date for meeting.

3.3.2 インタラクションモデルの抽出

インタラクションに関しては拡張 KAOS モデルのイベントに注目し，ともにロールが実行するオペレーション間で送受信されるイベントをインタラクションとして抽出する．図 5 のモデルからは,「スケジュール確認依頼」と「スケジュール回答」がインタラクションとして抽出される☆.

3.3.3 組織モデルの抽出

Gaia の分析フェーズでは組織の構造までは決定せず，構築システム（組織）内にサブ組織を構築する必要があるかを判断し，サブ組織を構築する場合は構成

☆ 実際にはイベント「会議室予約依頼」に対しても受信者確認の必要があり，本論文には記載されていない別オペレーションモデルをチェックすることで「会議室予約依頼」もインタラクションとして抽出される．

表 3 ロールモデル属性値の拡張 KAOS モデル要素との対応
Table 3 Attributes of the role model correspond to expanded KAOS model.

属性	対応する拡張 KAOS モデル要素
Role Schema	エージェント名
Description	エージェントが割り当てられた要件（開発者が記述）
Protocols and Activities	ロールが関与するオペレーションとイベントの集合（必要に応じて Monitor$^{\omega}$ を追加）
Permissions	エンティティに対する権限
Liveness	表 4 の生成ルールから得られた実行系列
Safety	割り当てられた要件に関する制約（開発者が記述）

するロール集合をサブ組織情報として抽出する．3.2 節の制約 3 により各オペレーションモデルが 1 つのシナリオに対応付けられているため，IMPULSE では各オペレーションモデルに記述されているロールの集合をサブ組織情報として抽出する．図 5 のモデルにはエージェントとして議長，コーディネータ，スケジュールマネージャが記述されているが，ロール判別プロセスの結果により，サブ組織情報としてロール集合 { コーディネータ，スケジュールマネージャ} を抽出する．

3.3.4 環境モデルの抽出

環境モデルとは，MAS が参照あるいは変更可能な環境リソースを定義するモデルである．IMPULSE ではまず，拡張 KAOS モデル上で定義されたエンティティを環境リソースとして抽出する．リソースに対する権限はオペレーションモデル上の関係線に注目し，ロールが担当するオペレーションとの間に出力関係が 1 つでもあれば変更権限を，そうでなければ参照権限を付与する．図 5 の例では，参照権限を付与するリソースとして参加者リストと個人スケジュールが，変更権限を付与するリソースとして候補日リストが抽出される．

3.3.5 ロールモデルの抽出

ロールモデル構築にあたっては，まず分析対象となるロールを抽出する必要があるが，ロール判別プロセスによりロールはすでに抽出されている．**表 3** は，IMPULSE においてロールモデルの各属性値と対応付けられる拡張 KAOS モデルの要素を示したものである．このうち，Description，Safety 属性に関しては開発者が拡張 KAOS 分析結果をもとに属性値を検討する必要があるが，その他の属性はすべて拡張 KAOS モデルから自動的に取得することができる．ここではロールの主要な属性である Liveness 属性の獲得手順を説明する．その他属性の獲得手順に関しては，付録 A.1 に記載してある．

表 4 ロールモデルの Liveness 属性生成規則
Table 4 The translation rule to liveness properties of Gaia role model.

対応する拡張 KAOS モデルの要素・状態（変換前）	変換後
ロールが実行するオペレーション ただし，直後のイベントと同一名の場合はインタラクションとするため抽出しない	x （アクティビティ）
ロール間で送受信されるイベント ただし，直前のイベントと同一名であり，送受信者が直前イベントと対称である場合は抽出しない	x （インタラクション）
オペレーションあるいはインタラクションに対応するイベントが x，y の順に連続している	x.y
オペレーション系列中に系列 X，Y の分岐がある	X|Y
オペレーション系列 X はループを形成している	X$^{\omega}$
X，Y は独立したオペレーション系列	X||Y
外部イベントによりオペレーション系列 X が開始される	Monitor$^{\omega}$.X

Liveness 属性とはアクティビティ・インタラクションにより構成される実行系列を表現したものであり，Gaia では活性表現（Liveness Expressions）[10] に従って記述される．**表 4** は拡張 KAOS モデルから Liveness 属性を生成するための変換規則を示したものである．IMPULSE ではロールごとに各オペレーションモデルに記述されている自身のオペレーションと関与するイベントを発生順に並べ，得られた時系列リストを変換規則に従って変換することで，1 つのシナリオにおける Liveness 属性を生成する．その後，オペレーションモデルごとに生成された複数の Liveness 表現を集約することでロールの Liveness 属性とする．

ここで，図 5 のモデルを用いてコーディネータに対する Liveness 属性の抽出過程を説明する．最初に，コーディネータが関与するオペレーション，イベントを時系列順に並べたリスト [会議開催条件の送信，参加者スケジュールを確認，スケジュール確認依頼，スケジュール回答，参加者スケジュールを集計，会議室予約依頼] を拡張 KAOS モデルから抽出する．続いて，このリストの各要素を先頭から順に表 4 の生成規則に適合させる．まず「会議開催条件の送信」は，事前のインタラクション抽出プロセスにより外部環境からのイベントであることが分かっているため，外部環境からのイベントを継続的に監視するアクティビティである Monitor$^{\omega}$ に変換される．続いて，オペレーション「参加者スケジュールを確認」がアクティビティと

して追加され，イベント「スケジュール確認依頼」と「スケジュール回答」はともにインタラクション抽出プロセスの結果からインタラクションとしてLiveness属性に追加される．さらに，オペレーション「参加者スケジュールを集計」とイベント「会議室予約依頼」が，それぞれアクティビティ・インタラクションとしてこの順に追加される．以上をまとめると，本モデルにおけるコーディネータのLiveness属性は次のとおりとなる．

Monitor$^{\omega}$. 参加者スケジュールを確認. スケジュール確認依頼. スケジュール回答. 参加者スケジュールを集計. 会議室予約依頼

3.4 モデル変換の自動化と分析モデルスキーマ

IMPULSEは，3.3節で述べたGaiaモデル変換手法を実現する変換ツールを提供することで開発者を支援する．開発者はまず拡張KAOSモデリングを実施するが，モデリングにはKAOSモデル記述ツールであるObjectiver[20]を利用する．分析結果の拡張KAOSモデルは，Objectiver上でXMLデータに変換可能であり，モデル変換ツールはこのXMLデータを入力情報とする．モデル変換ツールはモデル変換規則をJava言語により実装したものであり，入力として得られた拡張KAOSモデル要素をGaiaモデル要素へ変換し，以下の情報により構成されるGaia分析モデルスキーマを生成する．

- ロールのインスタンスと，各ロールにおけるDescription，Safety属性以外の情報
- インタラクションのインスタンスと，各インタラクションにおけるInitiator，Partner属性情報
- サブ組織を構成するロール集合
- 環境リソースのインスタンスとその属性情報

4. 分析モデル構築実験

IMPULSEのモデル構築手法としての有用性を評価するために，本研究では実際に構築されたMASを対象とした，既存方法論とIMPULSEによる分析モデル構築実験を実施した．本章では実験内容と実験結果を示し，MAS分析の難しさの観点からIMPULSEの有用性を考察する．

4.1 実験概要

実験で分析対象としたMASは，エージェントプラットフォームJADE[21]上で構築された追跡対象トラッキングMASである．本システムは各領域ごとに対象の侵入・退去を検知するセンサエージェントと，複数のセンサをグループ管理するゾーン管理エージェント，追跡対象を追尾するトラッキングエージェント，対象を見失った際に次に追跡対象が出現すると想定される領域を推測するロケーション推定エージェントなど6種類のエージェントにより構成される．

実験では実際に本システムを分析・設計・実装した開発者2名に従来のGaia分析とIMPULSEによる分析，さらにROADMAPによる分析を依頼し，構築された分析モデルの内容と分析過程を2.1節のMAS分析の難しさの観点から比較することでIMPULSEの有効性を評価した．今回は被験者が事前にシステム分析を実施しているため分析順序が与える影響は少ないと考え，また被験者の分析能力が構築モデルに与える影響を排除するため，同一被験者による分析モデル構築実験を実施した．なお，各分析は2名が協力して1つのモデルを構築する形態で，通常Gaia，IMPULSE，ROADMAPの順序で実施した．

4.2 実験結果

まず本分析実験の結果として，IMPULSE，ROADMAPの分析過程で得られた要求・分析モデルを示す．図6〜図8はIMPULSE分析により構築された拡張KAOSモデルの一例である．**図6**は対象追跡のために必要な環境情報に関して要求分析したゴールモデルであり，**図7**はセンサ検知結果のログと対象者のスケジュール情報を基に対象者の現在地を推測するロケーション推定ロールの要件をまとめた責任モデルである．また，**図8**はトラッカロールが追跡対象が存在する領域へ移動するまでのプロセスを表現したオペレーションモデルである☆．今回IMPULSEによる分析では，拡張KAOSモデルとして分析過程で6個のゴールモデルと，8個（うち6個がロール，2個が外部環境に対応）の責任モデル，2個のオペレーションモデルが作成された．これらは分析スキーマ獲得のための入力モデルとしてだけでなく，被験者がスキーマに分析を追加する際にも参照された．一方，ROADMAP分析時に構築された要求モデルは図2で示したモデルである．本モデル構築にはROADMAP要求モデル作成ツールREBEL[13]を利用した．

続いて，IMPULSE分析により実際に構築されたロールモデルの例として，ロケーション推定ロールのモデルを**表5**に示す．このうち，Description，Safety以外の属性に関しては，モデル変換により形式的に取得された属性値である．また太字の属性値は，従来Gaia分析，ROADMAP分析時には抽出されなかっ

☆ Objectiverでは，すべてのモデル要素を通じて同一名を付与することができないため，拡張KAOSモデリングにおいてオペレーションやイベントに同一名を付与する場合は，それぞれ先頭に“[I]”，“[R]”という接頭辞を付けることとした．

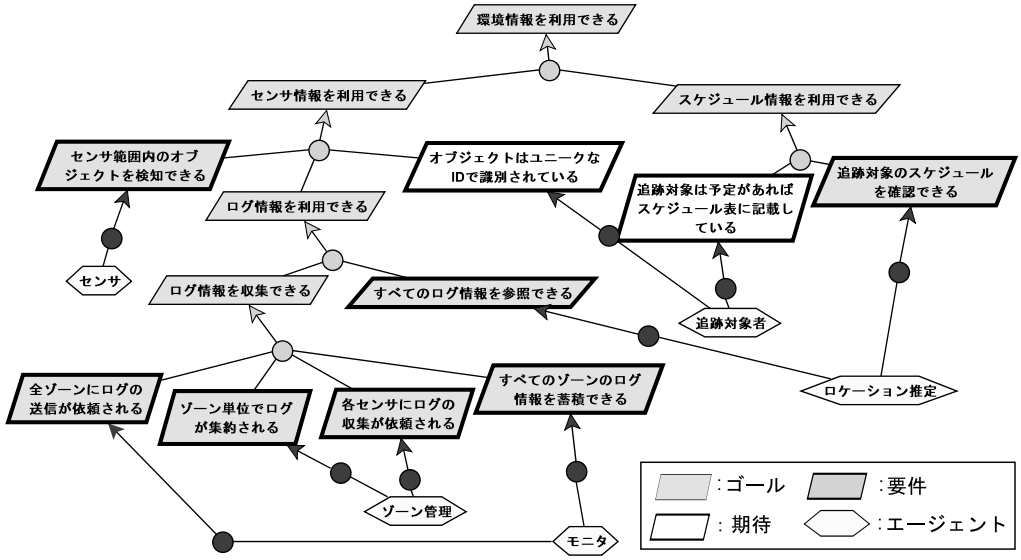

図 6 環境情報に関するゴールモデル
Fig. 6 The goal model for analyzing environmental resources.

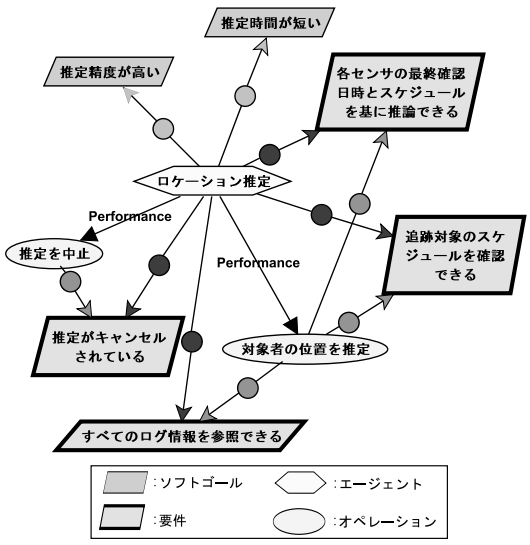

図 7 ロケーション推定ロールの責任モデル
Fig. 7 The responsibility model for role “Estimator.”

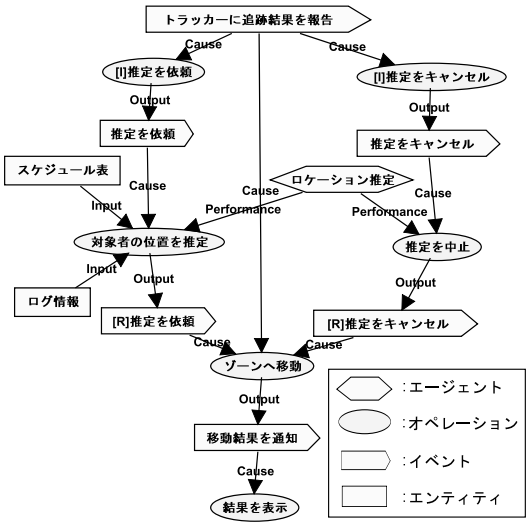

図 8 追跡プロセスを表現したオペレーションモデル
Fig. 8 The operation model for tracking process.

た要素を示している．

表 6 は，本実験により得られた各方法論による構築モデルを MAS 分析モデル構築の難しさの観点から評価したものである．ロールモデル構築の難しさに関しては，ロールを正確に分析しているかを評価するために，ロールの動的な側面を記述する Liveness 属性とロールの制約を記述する Safety 属性を評価項目として利用した．Liveness 属性に関しては，属性値の記述に利用されている要素数の総和を計測した．一方，組織ルールに関しては，全組織を通して定義されたルール数の合計と，組織あたりの平均ルール数を評価基準として用いた．以下，MAS 分析モデル構築の難しさの観点から実験結果を詳細に評価する．まず，ロールモデル構築の難しさとしては，第一に Liveness 属性の要素となるアクティビティの個数において，表 6 に示すとおり従来 Gaia，ROADMAP 分析と IMPULSE 分析間で大きな差（4 個）が見られた．これは，IMPULSE 分析では Gaia 分析あるいは ROADMAP 分析で抽出されなかった「対象者移動履歴ログの蓄積」や「喪失・発見情報を集約・分類」・「センサの検知結果を収集」・「追跡依頼の受付」といった受動的なアクティビティも

表 5 IMPULSE 分析で構築されたロケーション推定ロールのロールモデル
Table 5 Acquired role model for “Estimator” in IMPULSE analysis.

属性	属性値
Role Schema	ロケーション推定
Description	追跡対象者の移動履歴とスケジュール情報から対象者が存在すると思われる領域を推定する
Protocols and Activities	・Protocols: 推定を依頼, **推定をキャンセル** ・Activities: 対象者の位置を推定, **推定を中止**
Permissions	reads **ログ情報**, スケジュール情報
Liveness	推定を依頼. 対象者の位置を推定. 推定を依頼 || **推定をキャンセル.** **推定を中止. 推定をキャンセル**
Safety	**・推論結果抽出時間≦ MaxResponseTime** **・推論結果の信頼度≧ AcceptableRate**

確実に抽出していたことによるものである．これは拡張 KAOS モデルの終了条件により，ゴール分析結果の責務に対してオペレーションが必ず定義されていることを保証し，形式的にロールのアクティビティとして抽出する IMPULSE の有効性を示している．一方，インタラクションに関しては 3 手法とも通常プロセスで必要なものは抽出していたが，IMPULSE 分析では加えて，表 5 の「推定をキャンセル」インタラクションのように例外時に発生するインタラクションも抽出していた．これは，ゴール指向要求分析法における状況に応じた要求記述パターン Case-driven Decomposition [25] が適用された結果であるが，同じくゴール指向分析手法を採用している ROADMAP ではゴール分析の終了条件を設けていないため，分析結果が局面に応じた責務を記述できる単位にまで分解されなかったことが抽出漏れにつながったと考えられる．

以上のように IMPULSE 分析では，多くのアクティビティ・インタラクションと適用シナリオを抽出できたため，結果として Liveness 属性記述で出現する要素数が他分析結果を大きく上回る結果となった．これは，ROADMAP がシステムの動的な側面を記述することができないことから従来 Gaia と同程度の分析しかできなかった一方で，IMPULSE では拡張 KAOS モデルを利用することでシステム実現に必要なオペレーション・インタラクションを漏れなく記述するとともに，オペレーションモデルで作成したシナリオ図から属性値の自動抽出を実現した結果であると考えられる．

表 6 分析実験結果（表中の “IMP” は IMPULSE を，“ROAD” は ROADMAP を指す）
Table 6 Experimental result of each analysis.

評価項目	Gaia	IMP	ROAD
ロールモデル数	6	6	6
アクティビティ数	5	9	5
インタラクション数	8	13	10
Liveness 属性に含まれる要素数	31	43	29
Safety 属性値数	2	8	5
組織ルール数	10	15	8
サブ組織数	5	2	3
1 組織あたりの平均ルール数	1.67	5	2

一方の Safety 属性に関しても，従来の Gaia 分析が 2 個，ROADMAP 分析が 5 個，IMPULSE 分析が 8 個と，IMPULSE 分析が多くの制約条件を抽出していることが分かる．従来の Gaia 分析より ROADMAP，IMPULSE 分析が多くの属性を獲得できたのは，ゴール指向要求分析を利用することで各ロールの責務が整理されたことによるものと考えられる．しかしながら IMPULSE 分析では，たとえば表 5 のロケーション推定ロールのモデルに示される「推論結果返信までの時間」・「推論結果の信頼度」といった非機能要求に関する制約を抽出しているのに対し，ROADMAP 分析ではこれらを獲得できていない．これは，IMPULSE 分析では図 7 に示されるとおり，ロケーション推定ロールに「推定時間が短い」・「推定精度が高い」というソフトゴールが明確に対応付けられているのに対し，ROADMAP 分析では図 2 が示すようにシステムのトップゴールに応答時間や精度に関するソフトゴールが設定されているものの，これらのソフトゴールをどのロールが対処すべきかといった段階まで分析されていないことが原因と考えられる．つまり，ROADMAP では要件・ソフトゴールとロールが 1 対 1 対応していないが，IMPULSE では拡張 KAOS モデルの終了条件により，ロールと要件・ソフトゴール，オペレーション（Gaia でのアクティビティ）が明確に対応付けられ，その結果ロール責務の明確な把握が可能になったものと考えられる．

さらに今回の実験結果では，従来の Gaia 分析結果においてロールモデルの Permissions 属性値と環境モデル間において情報の不整合が確認された．Gaia モデルではモデル間で関連を持つ要素が多く含まれるため，手作業による分析ではこのような情報の不整合が発生する可能性があるが，IMPULSE では，拡張 KAOS モデルから形式的にモデル情報を抽出するため，要求フェーズ・分析フェーズ間だけでなく，分析フェーズのモデル間においても不整合を排除することができる．

続いて，組織制約条件の抽出に関しては，まず組織ルールについても，IMPULSE 分析は他手法と比較して多くのルールを獲得していることが確認できた．得られた組織ルールの内容を確認すると，従来の Gaia 分析では「〜した後に... する」といった Liveness 属性から推測できる単純な局所的時系列ルールの抽出が主であったのに対し，ROADMAP 分析では「ゾーンは 1 つ以上存在する」,「各センサは 1 つのゾーンにのみ所属する」といった組織構造に関するルールが主であった．その一方で，IMPULSE 分析では組織構造に関するルールだけでなく,「センサ単位の捜索結果はゾーン管理により集約され，ゾーン単位の捜索結果として対応するトラッカに報告される」といった組織の大局的なデータフローを表現するルールや,「0 以上最大許容数以下の追跡プロセスと 1 つのログ収集プロセスを持つ」といった組織維持のためのルールも抽出されていることが確認できた．これは，ROADMAP の要求モデルではシステムの動的な側面を記述できないが，IMPULSE ではロール・システムの振舞いが把握可能なオペレーションモデルを参照可能であり，結果として各シナリオにおける組織構造や処理の流れを明確に把握することができたことによるものと考えられる．

また，サブ組織数に関しては IMPULSE 分析はオペレーションモデル図で 1 つのサブ組織を定義するため，抽出数は他手法より少なかったが，他手法では結果として組織ルールが定義されないサブ組織も多く，有益なサブ組織抽出ができたとはいえない．その結果，組織あたりの平均抽出ルール数も IMPULSE は他手法よりも多く，組織ルールの検討にオペレーションモデルが有益であることが確認できた．

4.3 考　　察

以上の実験結果をもとに，2.1 節であげた MAS 分析モデル構築の難しさに対して本手法を評価する．

4.3.1 ロールモデルの構築

IMPULSE はゴール指向要求分析法である拡張 KAOS モデルを利用することで，開発者にゴールモデル構築時にシステム要件を責務に分解する程度のドメイン知識を要求するものの，得られた責務集合からシステムに必要なロールを抽出することを可能にする．また実験結果が示すように，モデル記述に制約・終了条件を設けることで，責務やオペレーションをロールに明確に対応付けることが可能であり，ロールに関する制約やアクティビティの抽出が容易となる．さらに拡張 KAOS の利用により，ゴールモデルでシステム要件に対応するための責務とロールを抽出し，責任モデルにおいてオペレーションを定義し，オペレーションモデルで Liveness，Permissions などの各属性値を検討するといった系統立ったロールモデルの構築が可能となる．特にオペレーションモデルにより分析時のシナリオ記述が可能となることは，ロールの振舞いを分析するうえでも，既存方法論と比較して大きな優位点となる．

IMPULSE では拡張 KAOS モデリング後にモデル変換を実施するが，このモデル変換は要求モデルと分析モデル間の情報損失や，分析モデル要素間での矛盾を排除するという観点からも有効である．以上の特性により，要求モデルと分析モデルのギャップから生じるロールモデル構築の難しさが解消されると考えられる．

4.3.2 組織制約条件の抽出

従来の Gaia，ROADMAP 分析ではロール・インタラクションモデル構築と並行した組織の検討が必要であったが，IMPULSE では拡張 KAOS オペレーションモデルを利用することで組織構成・組織制約条件を検討することが可能である．オペレーションモデルを利用する利点は，大きく 2 つあげられる．まず 3.2 節であげた拡張 KAOS の制約 3 により，不要なサブ組織の抽出が排除され，システム分析において考慮すべき組織のみを抽出することができる．また，1 つのオペレーションモデル図をサブ組織により実現されるシナリオと見立てることで，組織ルールの検討が容易になる．特に，オペレーションモデルにおけるオペレーション・インタラクションの実行順序を確認することで組織の動的な制約条件の抽出が可能となる．これは動的な側面を分析できない ROADMAP に対する大きな優位点であり，開発者に組織構造を想起する程度のドメイン知識があれば，オペレーションモデルにより組織制約条件が十分抽出できるものと考える．Gaia では設計フェーズでさらに具体的な組織構造を検討する必要があり，オペレーションモデルが包含するシステムの動的側面に対する分析結果は組織設計・構築においても開発者を支援するものと考えられる．

以上の考察から，IMPULSE は拡張 KAOS モデルを効率的に活用することで分析モデル獲得の難しさを解消しているといえよう．また，モデル変換手法により要求分析結果を継承した分析モデルの構築を実現するという意味でも，有効な分析モデル構築法である．

4.4 関 連 研 究

MAS 開発の要求フェーズと分析フェーズを関連付ける方法論としては，ROADMAP のほかに Tropos [14] が提案されている．Tropos は KAOS 同様に，要求工

学の分野で成果をあげている i*を要求分析手法として利用しているが，i*による要求分析では局面ごとのモデル図を記述できないことや，システムの動的な側面を記述できないことから，情報伝達手段や組織構造を判断することが難しく，MAS 分析モデル構築の難しさを解消するには至っていない．

要求モデルのモデル変換手法に関しては，ソフトウェア工学の分野でも提案されている．Heaven らは KAOS のメタモデルを利用することで，KAOS モデルを UML モデルで記述する方法を提案している[22]．ゴールの制約条件など各要素内に記述された属性値もマッピングされるため，この手法を利用することで，要求分析結果をセマンティックな側面からも UML 上で利用することができる．ただし，この提案はモデル情報をモデリング言語間で変換することを目的としたものであり，要求フェーズと分析フェーズ間でのモデル変換を実現するものではない．

開発プロセス全般におけるモデル変換に関しては，OMG のモデル駆動アーキテクチャ（MDA）[23),24)] が注目されている．MDA はモデル間のマッピングルールを定義することにより，後継フェーズへのモデル変換を実現するアーキテクチャであるが，現在は設計・実装フェーズの活動を想定したモデル変換を対象としている．一方で，本研究は開発の上流フェーズである要求フェーズと分析フェーズ間でのモデル変換を対象としたものである．

5. ま と め

本論文では，MAS 開発における分析モデル獲得の難しさを軽減するために，ゴール指向分析手法 KAOS を拡張したモデリング手法と，得られた拡張 KAOS モデルから Gaia モデルスキーマを形式的に取得するモデル変換手法を包含する分析モデル構築法 IMPULSE を提案した．IMPULSE により MAS 開発の障壁であった分析モデル獲得の難しさが解消され，要求分析結果を反映した MAS の開発が実現されると考える．

現在我々はロール重要度という評価指標を導入することによる，分析モデルの妥当性検証法を検討している．また同時に，獲得される分析モデルの質をさらに向上させるために，拡張 KAOS モデリングに有効なアナリシスパターン[25),26)] の導入を検討している．今後はさらに MAS 開発を通してのメタモデルを定義することで，設計フェーズ以降の MAS 開発プロセスも包含する方法論へと拡張したい．

参 考 文 献

1) Wooldridge, M.: *An Introduction to Multiagent Systems*, John Wiley & Sons (2002).
2) Luck, M., Ashri, R. and D'Inverno, M.: *Agent-Based Software Development*, Artech House (2004).
3) Padgham, L. and Winikoff, M.: *Developing Intelligent Agentsystems: a Practical Guide*, Wiley Series in Agent Technology (2004).
4) Jennings, N.R. and Wooldridge, M.: Agent-Oriented Software Enginnering, *Proc. 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World: Multi-Agent System Engineering* (*MAAMAW-99*), Valencia (2000).
5) Thayer, R.H. and Dorfman, M.: *Software Requirements Engineering, 2nd Edition*, IEEE Computer Society (1997).
6) van Lamsweerde, A.: Requirements Engineering in the Year 00: A Research Perspective, *Keynote Paper for 22nd International Conference on Software Engineering* (*ICSE 2000*), Limerick, ACM Press (2000).
7) Dardenne, A., van Lamsweerde, A. and Fickas, S.: Goal-Directed Requirements Acquisition, *Science of Computer Programming*, Vol.20, pp.3–50 (1993).
8) Leiter, E.: Reasoning about Agents in Goal-oriented Requirements Engineering, Ph.D. thesis, Universite Catholique de Louvain (2001).
9) Yu, E.: Modeling organizations for information systems requirements engineering, *Proc. 1st IEEE International Symposium on Requirements Engineering*, San Jose, January pp.34–41, lEEE (1993).
10) Zambonelli, F., Jennings, N.R. and Wooldridge, M.: Developing Multiagent Systems: The Gaia Methodology, *ACM Trans. Software Engineering and Methodology*, Vol.12, No.3, pp.317–370 (2003).
11) Cernuzzi, L., Juan, T., Sterling, L. and Zambonelli, F.: The Gaia Methodology: Basic Concepts and Extensions, *Methodologies and Software Engineering for Agent Systems*, Kluwer (2004).
12) Juan, T. and Sterling, L.: The ROADMAP Meta-Model for Intelligent Adaptive Multi-Agent Systems in Open Environments, *Proc. 4th International Workshop on Agent Oriented Software Engineering*, pp.53–68 (2003).
13) Kuan, P.P., Karunasekera, S. and Sterling, L.: Improving Goal and Role Oriented Analysis for Agent Based Systems, *16th Australian Software*

Engineering Conference (*ASWEC 2005*), IEEE Computer Society, pp.40–47 (2005).
14) Bresciani, P., Perini, A., Giorgini, P., et al.: Tropos: An Agent-Oriented Software Development Methodology, *Autonomous Agents and Multi-Agent Systems*, Vol.8, pp.203–236 (2004).
15) Padgham, L. and Winikoff, M.: "Prometheus: A Methodology for Developing Intelligent Agents", *Agent-Oriented Software Engineering III*, Vol.2585 of LNCS, pp.174–185, Springer, New York (2003).
16) Wood, M.F. and DeLoach, S.A.: An Overview of the Multiagent Systems Engineering Methodology, *Proc. 1st International Workshop on Agent-Oriented Software Engineering*, Vol.1957 of LNCS, pp.127–141, Springer, New York (2000).
17) Martin, R.C.（著），瀬谷啓介（訳）：アジャイルソフトウェア開発の奥義，ソフトバンククリエイティブ (2004).
18) Singh, M.P. and Huhns, M.N.: *Service-Oriented Computing: Semantics, Processes, Agents,* Wiley (2005).
19) Chung, L., Nixon, B.A., Yu, E. and Mylopoulos, J.: *Non-functional Requirements in Software Engineering*, Kluwer Academic Publishers (1999).
20) CEDITI: Objectiver. http://www.objectiver.com/
21) Telecom Itaria: JADE. http://jade.tilab.com/
22) Heaven, W. and Finkelstein, A.: A UML profile to support requirements engineering with KAOS, *Proc. IEEE Software*, Vol.151, No.1, pp.10–27 (2004).
23) Object Management Group: MDA. http://www.omg.org/mda/
24) Raistrick, C., Francis, P., Wright, J., Carter, C. and Wilkie, I.: *Model Driven Architecture with Executable UML,* Cambridge University Press (2004).
25) Darimont, R. and van Lamsweerde, A.: Formal refinement patterns for goal-driven requirements elaboration, *Proc. 4th ACM SIGSOFT symposium on Foundations of software engineering*, San Francisco, United States, pp.179–190 (1996).
26) マーチン・ファウラー（著），堀内　一（監訳），児玉公信，友野晶夫（訳）：アナリシスパターン再利用可能なオブジェクトモデル，ピアソン・エデュケーション (2001).
27) Nakagawa, H., Karube, T. and Honiden, S.: Analysis of Multi-Agent Systems based on KAOS Modeling, *28th International Conference of Software Engineering* (*ICSE 2006*), Shanghai (2006).

付　　録

A.1 Gaia 分析モデルへの変換アルゴリズム

```
/* ロールの判別 */
for all KAOS_agent do
  if (KAOS_agent が要件を割り当てられている)
    then KAOS_agent を Gaia_Role とする;
  if (KAOS_agent が期待を割り当てられている)
    then KAOS_agent を Gaia_Environmental Actor とする;

/* アクティビティの抽出 */
for all KAOS_Operation do
  if (KAOS_Operation が Gaia_Role により実行される)
    then{
      KAOS_Operation を Gaia_Activity とする;
      Gaia_Activity を Gaia_Role の Activities リストに追加する;
    }

/* インタラクションモデルの抽出 */
for all KAOS_Event do
  if (sender_op:KAOS_Operation |
     sender_op.Output == KAOS_Event
     && sender_op が Gaia_Role により実行される
     && KAOS_Event.Caurse == receiver_op
     && receiver_op が Gaia_Role により実行される)
    then{
      KAOS_Event を Gaia_Interaction とする;
      Gaia_Interaction を sender_op.Performer の Protocols リストに追加する;
      Gaia_Interaction を receiver_op.Performer の Protocols リストに追加する;
    }
    else KAOS_Event を Gaia_Environmental Event とする;

/* 組織モデルの抽出 */
for all KAOS_OperationDiagram do
  KAOS_OperationDiagram を Gaia_SubOrganization とする;
  for all operation in KAOS_OperationDiagram do
    if (operation が Gaia_Role により実行される)
```

```
35       then Gaia_Role を Gaia_SubOrganiza
tion の構成メンバーとして追加する;
36
37 /* 環境モデルの抽出 */
38 for all KAOS_Entity do
39   KAOS_Entity is Gaia_Object;
40   for all Gaia_Role do
41     if (Gaia_Role.input == Gaia_Object
42         && Gaia_Object.permission != "Cha
nges")
43       then{
44         Gaia_Object.permission:="Reads";
45         add "Reads" Gaia_Object in Gaia_
Role.permissions;
46       };
47     if (Gaia_Role.Output == Gaia_Object)
48       then{
49         Gaia_Object.permission := "Chang
es";
50         add "Changes" Gaia_Object in Gaia
_Role.permissions;
51       };
52
53 /* ロールモデルの抽出 */
54 for all Gaia_Role do
55    変換規則により Liveness 属性を生成する
(Gaia_Role);
```

(平成 18 年 12 月 1 日受付)
(平成 19 年 5 月 9 日採録)

中川 博之（学生会員）

1974 年生．1997 年大阪大学基礎工学部情報工学科卒業．同年鹿島建設株式会社に入社．2007 年東京大学大学院情報理工学系研究科修士課程修了，現在同大学院博士課程に在籍．エージェント技術とその開発方法論の研究に従事．電子情報通信学会会員．

吉岡 信和（正会員）

1971 年生．1993 年富山大学工学部電子情報工学科卒業．1998 年北陸先端科学技術大学院大学情報科学研究科博士後期課程修了．博士（情報科学）．同年（株）東芝入社．2002 年より国立情報学研究所に勤務，2004 年より同研究所特任准教授，エージェント技術の研究，ソフトウェア工学の研究に従事．現在に至る．日本ソフトウェア科学会会員．

本位田真一（フェロー）

1953 年生．1976 年早稲田大学理工学部電気工学科卒業．1978 年早稲田大学大学院理工学研究科修士課程修了．（株）東芝を経て 2000 年より国立情報学研究所教授，2004 年より同研究所アーキテクチャ科学研究系研究主幹を併任，現在に至る．2001 年より東京大学大学院情報理工学系研究科教授を兼任，現在に至る．2006 年より早稲田大学客員教授，現在に至る．2002 年 5 月〜2003 年 1 月英国 UCL ならびに Imperial College 客員研究員．2005 年度パリ第 6 大学招聘教授．工学博士（早稲田大学）．1986 年度情報処理学会論文賞受賞．ソフトウェア工学，エージェント技術，ユビキタスコンピューティングの研究に従事．日本ソフトウェア科学会理事，情報処理学会理事を歴任．IEEE Computer Society Japan Chapter Chair，ACM 日本支部会計幹事，情報処理学会フェロー，日本学術会議連携会員．