基専

2 社会シミュレーションと組織・社会の情報処理のアーキテクチャ・デザイン

出口 弘 東京工業大学知能システム科学専攻

情報システムのアーキテクチャ・デザインの課題

IOE 時代のアーキテクチャ・デザイン

ネットワーク上で，膨大な数のセンサ，アクチュエータ，コントローラ等と人間が混在し，それらが結びついた巨大なネットワークが人類の社会経済システムの基盤インフラとなる時代が訪れようとしている．この IOE（Internet of Everything）と呼ばれる人工物としてのシステムを設計・実装・管理・運営する技術を開拓することは喫緊の課題である[1]．すでに我々は，インターネット上のコミュニケーション環境など，生活世界の基盤が人工物となりつつある社会に暮らしている．この人工物としてのシステムについて，人工物の科学（Science of Artificials）を唱導したのは，システム科学者の H. A. Simon であり，今日のオブジェクト指向のシステム設計は，Simon と組織科学者の March による組織の情報処理パラダイムがその淵源となっている[4), 6)]．この組織の情報処理パラダイムは，UML（Unified Modeling Language）というシステムの開発技法へと結実した．だが社会や組織の人工物の設計という視点から見てそれは望ましい方向であったのだろうか．また膨大な数の自律的エージェントが結びついた IOE 時代のシステム設計・導入・管理に耐えられるアーキテクチャなのであろうか．本稿ではこの課題に対して，ネットで結びついた人間やものやソフトエージェトなど実世界のエージェントからなるシステムを分析・設計・実装・管理する技術としてのエージェントベースのシステム・アーキテクチャについて解説する．そこではエージェントベースのシミュレーションで培ったマルチエージェントと分散ストリーム計算の技術が，実世界のマルチエージェントの制御技術と融合することで新たなアーキテクチャが見えてくる．

ビジネスプロセスの複雑性とアーキテクチャ

本稿では，インターネットで接続された自律的エージェント間での協調分散的なシステム（Distributed Corporative System）をいかに構築するかを考える．自律的エージェント間の協調分散的制御のためのコンピューティングをここではエージェント・コンピューティングと呼ぶ．またそのための基幹のアーキテクチャとしてリアルワールド OS（実世界 OS）という概念を導入する☆1．すでに IOE のためのネットワーク OS としては「Cisco IOS」を Linux OS と統合した「Cisco IOx」がフォグ・コンピューティングのためのプラットフォームとして提案されている．これに対しここで言うリアルワールド OS は，インターネット上の実世界エージェントに対する協調分散的な制御を行うプログラムを開発・実装・管理するためのもので，よりサービスシステム寄りの枠組みとなっている． 家庭や工場・職場・地域等のローカルな環境で，人やセンサ・さまざまな機械等のインターネットに接続している自律的エージェントを，協調分散型システムとして機能させるための方法を課題とする．そのためには実世界のエージェント間関係をまずエージェントベー

☆1 リアルワールド OS は東京工業大学の COI-T プログラム「オンデマンド・ライフ＆ワークを全世代が享受できる Smart 社会を支える世界最先端 ICT 創出 COI 拠点」で筆者によって提案されたアーキテクチャであり，本稿の一部は（独）科学技術振興機構（JST）の研究成果展開事業「センター・オブ・イノベーション（COI）プログラム」の支援によって行われた．支援に謝辞を表したい．

ス・シミュレーションで表現する．次にそれを部分的に実世界エージェントと入れ替えながら実装しエミュレートすることでこれをテストする．これを可能とするエージェントベース・シミュレーションのエンジン部分が最終的には実世界エージェントを管理するシステムとなるというのが，実世界 OS というアーキテクチャであり，本稿ではそれについて解説する．そのための原型となるエージェントシミュレーション・アーキテクチャである SOARS について説明しよう．

SOARS というマルチエージェントデータ処理のアーキテクチャ

SOARS のアーキテクチャとマルチエージェントプログラミング

SOARS（Spot Oriented Agent Role Simulator あるいは Social & Organizational ARchitecture Simulator）は大規模なマルチエージェントモデルを構築するための枠組みとして提案，実装されたフレームワークであり，オープンソースで提供されている☆2．マルチエージェントのシミュレーションは，サンタフェの Swarm に淵源を持つシカゴ大学とアルゴンヌ研究所の開発した Repast や，ジョージメイソン大学の開発した MASON，それに NetLogo などいくつかの種類があるが，これらはいずれもセル型のモデルに強く影響されている☆3．エージェントはセル上の 2 次元の格子状空間を移動し，エージェントとセルに属性値が与えられ，位置と属性の変化によって動的なシステムは記述される．このアーキテクチャは実世界モデルを 2 次元でシンプルに写像化したもので，エージェントが 2 次元平面を移動するモデルを表現するのには簡便な記述方法を与えるが，より抽象度の高い社会的な相互作用が必要な場合アドホックなモデルを導きがちである．これに対して SOARS では，人が組織の中で行う複雑な情報処理活動に規範をとったアーキテクチャを持つ．

H. A. Simon が組織の情報処理パラダイムを言う以前から，人間は人と物（経済学で言うところの労働と設備資本）が絡む複雑な組織を運営してきた．そこでは特に複数の人々が並列に遂行可能なタスクを混乱なく実行することが重要である．たとえば会計プロセスでは単位期間の中の多くのタスクが，並列部分と順序づけられた部分に区分され，自律分散協調的に処理されている．取引伝票処理から，部門ごとの伝票の締め，残高試算表・損益計算書・貸借対照表の計算，繰越処理などの一会計期間の中での一連のタスクは，タスクそのものは順番に実行されるが，それぞれのタスクの中ではたくさんの従業員が並列に伝票処理等の仕事を行っている．個々の従業員の役割はタスクと結びつけられ，タスク間でタスク処理の順序が決められていることで，並列実行を含む複雑なプロセスのイベント処理を行っている．SOARS ではこの実世界の主体が行っているタスクの順序関係をステージという概念で整理して，1 つの単位期間を複数のステージに分割する．エージェントがステージと結びつけられた「役割」を持つことで，ステージ内での並列処理と，ステージ間での因果的順序関係を持つ処理を切り分けた上で，複雑な主体の相互作用を後述の「役割概念」で記述する．これは組織の中でのマネジメントを抽象化してプログラミング原理として抽出したものとなっている．

SOARS では，エージェントが並列な処理を行うステージという単位で離散時間を分割する独自の制御構造を導入する．またエージェントは役割を持ち，その役割に階層構造が導入できる．エージェントは状況に応じてその役割を切り替える役割指向という，オブジェクト指向とは似て否なるエージェントプログラミングの方法が導入される．この SOARS のステージや役割指向に基づくモデルの設計思想は，セル概念の背後にある空間的な場の概念に基づくモデル概念ではないが，社会科学にとってはむしろ自然なモデル構築の方法を与えてくれる．SOARS はそれゆえ，社会科学や政策科学の領域でのエージェントベースのモデリングツールとして使われるよう

☆2 http://www.soars.jp
☆3 Repast, http://repast.sourceforge.net
MASON, http://cs.gmu.edu/~eclab/projects/mason/
NetLogo, http://ccl.northwestern.edu/netlogo/

になってきている[3), 8)].

ステージモデルと役割指向のプログラミング

SOARSのステージモデルは会計的なビジネス処理にその淵源を持つ並列処理のアルゴリズムである．会計プロセスでは，1つの会計期間を，伝票を集めるステージ，それを集計して残高試算表やP/L, B/Lを計算するステージ，決算繰越のステージ等，複数のステージという時間スライスに分割する．伝票を集めるステージでは，どのエージェントが先に伝票を出しても結果には影響しない．このように実時間を論理的にステージに分割して複数のエージェントの活動を管理する方法は，現実の組織ではきわめて広範に観察されるが，それ自体アルゴリズムとして研究されてこなかった．ステージモデルでは，離散時間システムでの単位時間を複数のスライス（それぞれのスライスをステージと呼ぶ）に分けて，スライスの内部ではエージェントの動作は順序によらないという条件を満たすようにプログラミングする．ステージ間では前ステージでの結果を前提とする因果関係を認める．エージェントの行為に関して，その因果順序を示すシーケンス図を描く必要はなく，エージェントのタスクの実行順序を厳密に制御する必要もない．大規模なエージェントの相互作用がステージ概念だけで制御できるため，マルチエージェントのモデリングでは非常に強力な手法となる．

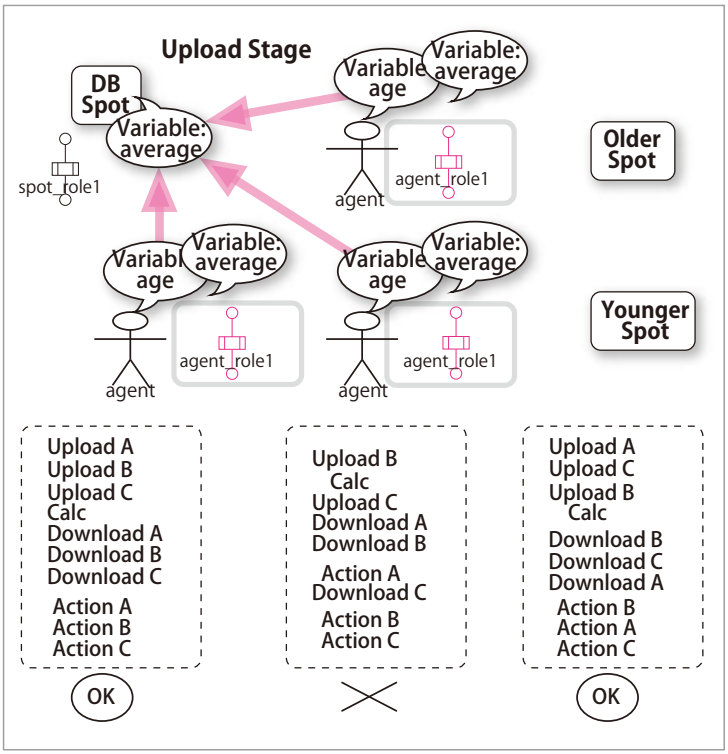

図-1　平均年齢より若いか歳をとっているかで移動先を変えるモデル

これを簡単な事例を用いて説明する．複数のエージェントが自分の年齢を申告し，その平均年齢を計算して，自分が平均より若ければAスポットに，同じか歳をとっていればBスポットへ移動するというプログラムを考える．**図-1**はこのUpload Stageとエージェントの実行順序の影響を図示している．

この事例を表現するために，Upload, Calc, Download, Actionという4つのステージを想定する．それぞれのステージでエージェントや場の役割が定義される．Uploadステージで平均年齢の計算を行う計算スポットに年齢データが送られ，Calcステージで平均年齢の計算が行われ，Downloadステージで個々のエージェントがその平均年齢の取得を行い，Actionステージで結果に基づきエージェントは場所を移動する．図-1ではUploadステージで3つのエージェントが計算スポット（図中のDB）にデータを送っている．またステージの中ではエージェントの実行順序を変えても意図した動作は変わらないが，ステージの順序が崩れると結果が意図とずれることを例示している．

N個のエージェントの間の相互作用は論理的には2のN乗種類になるが，そのうち結果が実行順序に関係ない非本質的な因果関係をステージ概念で整理することで，遥かに少ないステージ数で本質的に因果的な相互作用だけを切り分けることができる．ただし現在のSOARSはエージェントごとにスレッドを立てているわけではなく，その意味では因果的な意味で並列だが，実処理としては並列処理を行っているわけではない．これに対して後述する実世界でのエージェントコンピューティングでは，実世界エ

ージェントの相互作用を本質的に並列に制御できる．

SOARS ではステージ概念のほかに，役割指向のプログラミングと呼ぶ手法が用いられる．エージェントは役割を取得し活動するが，役割そのものを切り替えることでまったく別の振舞いが可能となる．たとえば医者役割のエージェントが，モデルの中で感染すれば入院中は患者役割に変化する． エージェントの振舞いは，役割単位で分割され，さらに役割に継承概念を導入することで，複雑な振舞いを階層的観点からも分割できる．役割指向は，オブジェクト指向のクラス概念やその継承とは異なる枠組みである．オブジェクト指向のプログラミングでは，クラスのメソッドを継承で複雑化したり上書きしたりはできるが，まったく異なったものと切り替えることは普通はしない．エージェントの複雑な相互作用を記すにはステージと役割概念がきわめて有効に機能する．

協調分散的なシステム設計と分散ストリーム計算

エージェントコンピューティングと分散ストリーム計算

IOE 時代には，注文や取引等のイベントやセンサの測定等のデータ上流から目的に応じたさまざまな計算プロセスを経てデータの利活用の下流までの一連のデータ処理が，多くのエージェントが結びつくことで可能となる．このプロセスをいかに自律分散協調システムとして構築するかが大きな課題である．そこでは，エージェントのタスクの並列実行という課題と同時に，データ処理のストリームに，新たなデータ処理のためのノードや新たなデータソースが付け加わっても，システム全体としてロバストに機能するようなシステムの構築法が求められる．これらの課題を満たす，自律分散協調システムについて，本章では分散ストリーム型の「計算」タスクに特化して論じる．ただしここでの「計算」は広義の組織の情報処理を強く意識している．これは組織の情報処理パラダイムが明らかにしたように我々の社会的活動は，そこで行われる情報処理により特徴付けられ，また組織や社会的活動を設計するということは，その情報処理を設計することだからである．エージェントの活動はこの情報処理と深く結びつけられる．

フィルタ型のストリーム計算と代数的仕様記述

分散ストリーム計算では，ストリームを構成する個々の計算ノードで，入力データに対して何らかのデータ変換の処理を行い出力データを得るという意味でのフィルタ型の計算を行う．この計算プロセスを構成する個々のフィルタが，データを扱う現場でのそれぞれの領域専門家の用いる概念水準に適合した形で仕様記述がなされることが重要である．だがこれは容易ではない．会計処理でいえば，本来簿記の概念に従って組み立てられるべき計算が，RDB (Relational Database) や XML (Extensible Markup Language) 上のデータ構造へとコンバートされることにより異なった水準で仕様記述がなされている．それは会計処理を行う現場の担当者にとって水準の異なるデータ構造と計算概念への変換が行われていることを意味する．社会や組織のあらゆる現場には，そこでの領域専門家にとって自然でかつ固有のデータ構造とデータ処理がある．その表層構造は比較的単純なテーブル形式で表現されることが多い．データは何らかのデータ変換プロセスを通じて結びつくことで，複雑なデータ処理が行われる．この現場のデータ表現とデータ処理を取り込むために，データ構造を代数的に表現し代数的な仕様記述を行う．組織のデータ処理のための基本データ構造として，フラットなテーブル（レコード型）と借方貸方の構造を持つテーブル（会計型）という 2 つの表現型が区分できる．この 2 つの表現型の背後にはそれぞれ，データ代数，交換代数という代数的に定式化されたデータ構造がある．それをクラスとして実装することで，これらのデータ構造に関するフィルタ型の計算は数理的に仕様記述した上でプログラムとして実装できる．フィルタの仕様記述は，代数

☆4 特願 2012-108592，「情報処理システムおよびエネルギー情報の記録装置」；特願 2011-190601「データ編集装置およびデータ編集方法」

的なオペレータを含む集合論的な内包記述でなされる[5)☆4]．集合論的な内包表現と代数的なデータ構造を用いることで，計算フィルタの抽象的な仕様記述と，その実装としての言語記述をきわめて近い水準で揃えることが可能となる．

図-2ではデータが源流データか計算の結果得られる派生データかを区別しつつ，派生データを生成するプロセスを計算フィルタとして表現している．分散ストリーム計算は，実ビジネスでのデータの取得や変換，さらに関連諸部門でのそのデータを利活用する流れと自然に対応する．あらかじめライブラリとして用意したフィルタを利用する分散ストリーム処理のための環境さえあれば，さまざまなフィルタを組み合わせることで複雑なデータ処理を直感的に組み立てることもできる．またデータストリーム型の計算モデルは，絶えず変化するビジネスプロセスに対応して情報システムの方もロバストかつ容易に再構築できる点にその特色がある．たとえば，図-2の下段のFilter4のように新しいデータとそれを処理するノードを付け加えたとしても，既存のデータ処理のフィルタノードと干渉しないので，ロバストな形で機能の追加が可能となる．協調分散型のシステムでは，常にその構造が変化しても全体としてのプロセスのロバストネスは担保される必要があり，分散ストリーム型の計算モデルはこの要件を満たす．分散ストリーム型のデータ処理では，数千万から1億程度のレコードであれば，ブラウジングによる可視性を保った形で，ロバストかつ柔軟に変更が可能な情報処理のシステムが構築できる．

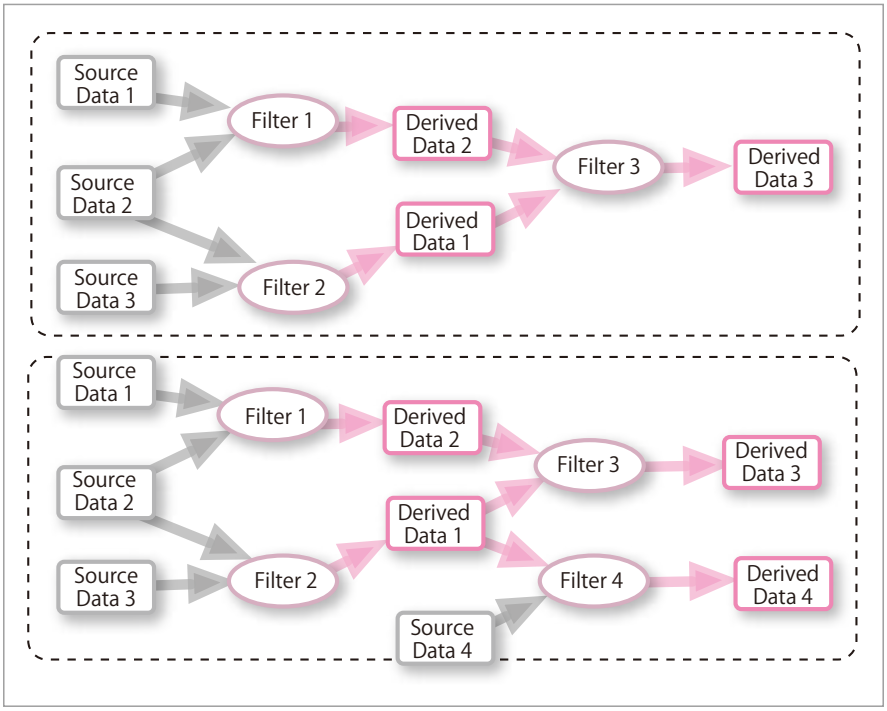

図-2　分散ストリーム計算としてのデータ構造とノードの付け加え

組織の情報処理は，マルチエージェントの相互作用であると考え，自律分散的なエージェント・コンピューティングに基づく計算と見なすことができる．エージェントの実世界相互作用では，自律分散協調的な計算は，個々には入出力フィルタ型の計算を行う計算ノードが分散的に存在し，それらが連結することで計算ノード間のストリームとして行われる．これはエージェントベースのモデリングと相性のよい計算の手法である．現在は分散ストリーム計算というとオンメモリでの高速計算が言及されることが多い．だがここでいう分散ストリーム計算は組織や社会のデータが発生し，そのデータの加工と利用が必要とされる場所や部門で，それぞれ局所的な計算を行い，それに必要な他の部門のデータを入力とし，出力をそれが必要とされる部門へ送る，フィルタ型の計算ノードを跨がるストリームという意味での分散ストリーム計算である．分散ストリーム型のデータ処理モデルは柔軟で頻繁な組み替えをロバストな形で可能とし，かつ今後ますます大規模化するデータフローに耐えられるアーキテクチャを与えてくれる．だがそのためには，3つの課題がある．第1の課題は，すでに述べたフィルタ型計算の仕様記述とそこで用いられるデータ構造をそれぞれの領域でどう標準化するかという課題である．第2はストリーム計算のためのノード間のデータ転送とデータ共有

をいかに行うかという課題である．最後の課題は複雑な計算とその計算結果の利用をさまざまなノード（エージェント）間の計算の流れとして分散並列制御を行うための制御上の仕組みをどう構築するかという課題である．次章ではこの第2と第3の課題を扱う．

プロジェクト・プログラミングと実世界モデリング

ワークフロー遂行のための2つの課題

本章では，複数のタスクを半順序関係で整列したワークフローベースでのシステムデザインを考える．前章で取りあげたストリーム型のデータ処理プロセスも，個々のフィルタ型の計算ノードをタスクノードとするワークフローとなっている．プロジェクトであれ定型業務であれ，順序付けられたタスクの集合という意味でのワークフローは，現場で何らかの仕事を体系立てて整理し，理解し，実行するための基本的な認識の単位である．ワークフローはそれを構成する個々のタスクに対して，それを遂行する（人であったり，プログラムであったり，ソフトエージェントであったりする）資源を割り当てることで遂行可能となる．

従来もビジネスワークフローを記述する理論や言語は，ペトリネットやその拡張，UMLでのアクティビティ図，あるいはBPEL（Business Process Execution Language）や，より抽象化されたYAWL（Yet Another Workflow Language）などがあり，複雑で非同期なプロセスを記述できる．他方でこれらの理論や言語は，現実のワークフローで求められる2つの機能，すなわち（1）タスクを実行するためのリソースとしての実世界エージェントを個々のタスクにマッピングし，（2）タスクに割り付けられた実世界エージェントを管理してワークフローを遂行するためのスケジューリングを行う，という作業を支援してくれるわけではない．

プログラム＆プロジェクトマネジメント（以下P2M）の領域では，このような実世界の複雑なプロジェクトを遂行するために，リソースの割当や，スケジューリング管理の方法を研究してきた．クリティカルパスという最長時間のボトルネックとなる経路の解析法はその1つである．このP2Mの考え方を，エージェント・コンピューティングに取り入れるためにはワークフロー（しばしばタスクパスや医療の領域ではクリニカルパスとも呼ばれる）を構成する個々のタスクに実世界のリソースを割り当てるという課題と，その上で全体のタスク進行をワークフローに従って管理するという2つ課題を区分して論じる必要がある．

第1の課題には，たとえば医療の治療の標準化されたワークフローであるクリニカルパスへの医療資源の割当問題がある．患者ごとに必要となる診療のワークフローとしてのクリニカルパスには，そのタスクの遂行に必要な医者や看護師や技師や，検査機器などの実世界エージェントを割り当てる作業が必要となる．患者の数に対して，医者の数や看護師の数等の資源の量は一般に遥かに少なく，クリニカルパスに対して，順次医師や看護師，検査技師や検査機械等の割当を行い，タスクが終わればリソースを開放して，それをまた別のワークフローのタスクに割り当てるという作業が必要となる．これはまた内装工事のワークフローに対して，配管工や電気工等のリソースを割り当てて，マンション等の複数の部屋の内装工事を遂行する問題でも同型である．ここでも作業者の資源は希少であり，それを順次割り当てるアルゴリズムが必要となる．これはジョブショップスケジューリング等のアルゴリズムを遥かに複雑にした課題だが，実世界ではさまざまな工夫で，総処理時間が最小になるような割り付け問題を管理している．またそのためのアルゴリズムの研究は，P2Mの領域でも行われている[2]．

第2の課題は，膨大な数のセンサやコントローラ，アクチュエータ等の実世界エージェントを対象として，あるワークフローでその自律分散協調制御を行うという課題である☆5．前節の分散ストリーム計算も同様の課題となる．実世界エージェントの自律分散協調制御

☆5 特願2014-083629「ワークフロー管理装置，ワークフロー管理方法およびワークフロー管理プログラム」．

を行うための，この 2 つの課題を解決する必要がある．

プロジェクトのワークフローでの 2 つの並列概念

プロジェクトのワークフローでは，タスク間関係は半順序で表される．この半順序構造はペトリネットや UML のアクティビティ図でも示されている．半順序構造を持ったタスク間関係では，あるタスクの後に可能なタスクが複数ある場合には，それらのタスクは互いに比較不可能である．これはタスクの実行という視点からは，互いに順序関係にないタスクは並列に実行可能であることを意味している．本稿では 2 種類の並列性が扱われている．1 つはあるステージで作動する複数のエージェント間で，実行順序によって結果が影響を受けないという意味での並列性である．これは SOARS で扱われているステージ内でのエージェントの並列動作の実世界エージェント版である．だが実世界エージェントではもう 1 つ重要な並列性がある．それがタスクの半順序関係の中で，タスクそのものが並列に実行可能であるという意味での並列性である．

この 2 つの並列性の区別が，エージェントベース・シミュレーションから実世界エージェントに対するエージェント・コンピューティングへと拡張するためには必須となる．プロジェクトのワークフローを構成する個々のタスクを遂行するエージェントは 1 つとは限らない．たとえば，ネットに接続されたアンケートシステムで年齢を，電子血圧計で血圧を測定し，それを用いて何らかのデータ処理を行う事例を考える．この場合対象となる人間（エージェント）は複数いると仮定するのが自然だろう．複数のエージェントに対して，年齢を申告するタスクと，血圧データを収集するタスクはタスクとして並列に実行可能となる．さらに年齢を申告するタスクで個々のエージェントが年齢を答える順番も，血圧データを収集するタスクで血圧を電子的に測定し送る順番も，それぞれのタスク内で順序に依存しない．つまり 1 つのタスクでエージェントの実行順序にタスクの結果が依存しないという意味での並列性と，ワークフローの中でのタスクの実行順序の並列性の 2 つの並列性が区別されねばならない．

SOARS でのステージ内のエージェントの実行順序の並列性は前者の意味での並列性である．SOARS ではステージの実行順序が線形順序であり，半順序になっていなかったため，後者の並列性は扱われていなかった．

タスクマネージャによるエージェントの協調分散システム管理

ワークフローにおけるタスクの順序を示すために，ワークフローの中でのタスク名をステージを示すものとしてステージ概念を拡張する．ワークフローにおいて，個々のエージェント（ノード）は，今どのタスクのステージにあるかを知ることで，それに対応する自律的プログラムを起動させると仮定する．エージェントは当該のステージでの作業が終了したことを報告し，次のステージの指示を待つ．このようにワークフローに基づき，エージェントのタスクの実行順序を制御する機能が，エージェント・コンピューティングでは必須となる．これをリアルワールド OS のタスクマネージャ機能と呼ぶ．タスクマネージャは，今どのタスクステージを遂行すべきかをエージェントに通知する．さらにタスクの完了の通知をエージェントから受け取る．このタスク管理のデータフローは，実世界エージェントとタスクマネージャの間のデータフローとなる．自律分散協調システムでは，エージェント間で協調的に活動するために，相互にデータを受け渡すためのデータフローも必要となる．前章での分散ストリーム計算では，個々のフィルタ型の計算タスクの実行結果は，次にそれを入力するフィルタ型の計算タスクに受け渡されねばならない．センサの計測でも，それを用いて次のタスクステージで計測の平均値を計算するなどの計算タスクが設定されている場合，その計算を実行するエージェントにデータを受け渡すデータフローが必要となる．この後者の意味でのデータフローは，タスクマネージャを経由する必要はない．これを説明するためにタスクマネージャの実装例として，

MQTT（Message Queuing Telemetry Transport）として知られている，Pub-Subのスキームを用いる．Pub-Subのスキームでは，エージェント間でデータを共有するためにブローカと呼ばれる特殊なエージェントを用いる．ブローカにトピックと呼ばれる特別なタグを付けてデータを送るエージェントは，Publisherと呼ばれる．これに対して，あるタグの付いたデータを送ってほしいというリクエストをブローカに登録したエージェントは，Subscriberと呼ばれる．特定のタグを登録しておくことで，登録したタグを持ったデータがブローカを経由してSubscriberに自動的に送られる．このPub-Subのスキームを用いてエージェント間のタスク遂行に必要なデータ転送と，タスクマネージャとエージェント間のステージ情報等の管理情報の転送を別々に行うために，この実装例ではさらにデータ転送用のブローカとタスク管理用のブローカの2種類のブローカを仮定する．

1）タスクマネージャはタスク管理用のブローカに，現在実行可能なタスクステージ名を通知する．エージェントはあらかじめ自分の遂行するタスクのステージ名を登録しておく．タスクの遂行が終了したエージェントは，ステージの終了を管理ブローカにパブリッシュすると同時に，データブローカに必要なタグを付けてデータを転送する．

2）タスクマネージャは，タスクの遷移のための判断を行う．それにはタスク終了情報などをもとに次のタスクステージに遷移してよいかを判断するタスク遷移述語が必要とされる．分散ストリーム計算では，計算が無事終了したという情報でタスク遷移述語はTrueと判断する．また，何らかの異常終了のデータが送られてきた場合には，通常のワークフローから非常用のワークフローに遷移する等の分岐の判断もタスク遷移述語によって行う．

3）このプロセスを繰り返すことで，タスクマネージャはタスクのステージごとに実世界エージェントとのハンドシェイクを遂行できる．この場合，ステージ間の並列性は，並列実行可能なタスクのステージ名を同時にパブリッシュすることで示される．また同じステージに属する複数のエージェントは，実時間で順序が入れ替わったり遅延があってもその結果には影響がない．これは人間によるプロジェクト管理でも，エージェントの分散協調制御でも同様である．

エージェントの協調分散的管理の事例

ここではSOARSのようなバーチャルな世界でのエージェントベース・シミュレーションと実世界のエージェントの制御を混合して扱うことのできる実世界OSの概念に基づいた設計と実装の事例として，複数の温度センサと光度センサのデータから，平均温度と最小光度を計算し，それをもとにコントローラがエアコンとライティングのコントロールを行う簡単な制御について説明する．そこでは同時に，エアコンとライトのコントローラからの稼働情報によってエネルギー消費を計算し，それをステークホルダに通知するというタスクも行う．これらの一連の協調分散制御のためのプロジェクトのプログラミングとそのためのリソースマッピングは次の**図-3**のように与えられる．ここではセンサに対するデータ収集のタスクに対し，複数のセンサが割り当てられている．またリソースのタスクへの割当は固定的である．タスクマネージャによる管理プロセスとそこでのデータフローの順序を示したのが**図-4**である．ここではタスクマネージャと実世界エージェントのステージ管理を仲介するステージブローカと，実世界エージェントの間のデータフローを仲介するデータブローカの2種類のブローカが，実世界エージェント間の制御フローとデータフローをPub-Subスキームで仲介している．図-4ではエージェント間のデータフローを示すのに，意図的にシーケンス図を記しているが，そこに記された矢印間の遂行順序は，タスクマネージャのステージ情報によって管理されるもので，シーケンス図の特性上，矢印の上下で順番がついているが，本質的に並列なものも含まれている．たとえばセンサから計算ノードへのデータの転送はどの順序で行われてもいい．また計算ノード1，2からそれぞれコントローラ1，2への計算結果

の転送も順序に依存しないし，それぞれのコントローラからエアコンと電灯への制御情報の伝達の順番も，計算ノード3へのエネルギー消費データの伝達の順番も結果に影響しない．

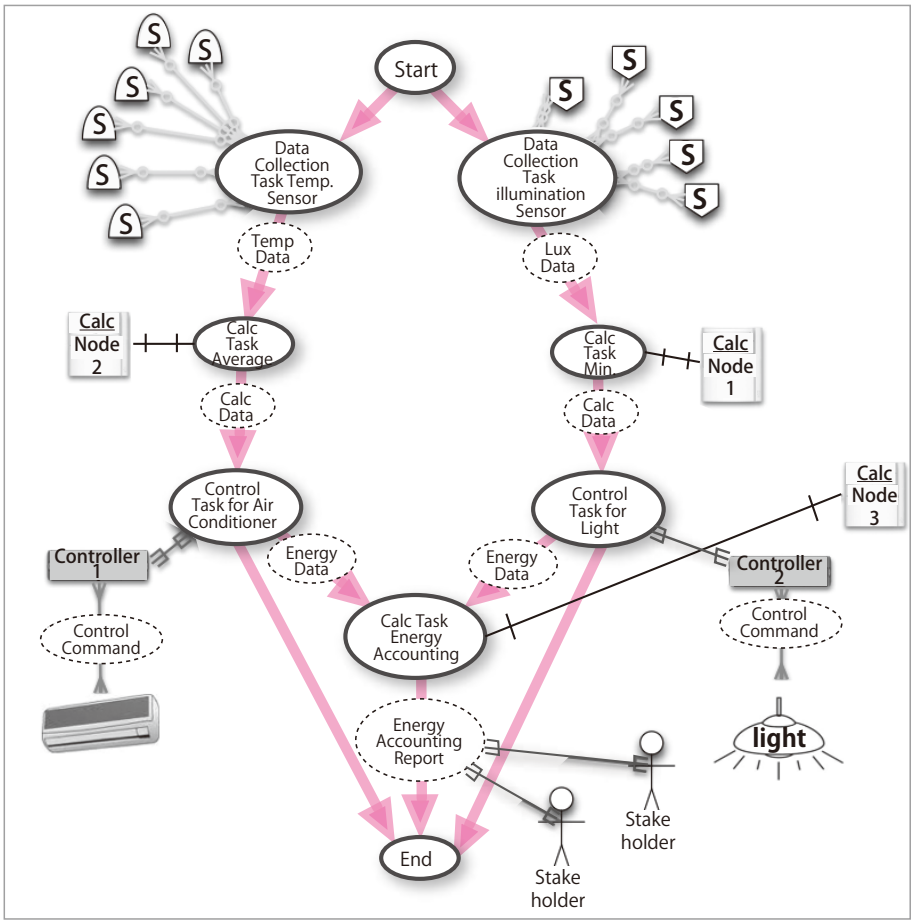

図-3　タスクマネージャとその管理する実世界エージェント

シミュレーションからリアルワールドOSへ

IOE時代には，さまざまなサービス機能を持つ自律的エージェントが分散的に同期し協調し，ワークフローとしてデザインされた全体サービスを支援する実世界OSが求められる．そこで支援可能なサービスには次のようなものが考えられる．まずセンサリング＆コントロール系サービスであれば，センサリングとそれによる機器の制御のサービスや家やオフィス，コミュニティの見守りやセキュリティのサービス等が，データの分散検索系サービスであれば，たとえば複数の病院等の組織を横断しての並列検索サービス等が考えられる．ビジネスデータ処理系のサービスであれば，先に述べたような自律分散的なビジネスデータ処理が，既存のERP（Enterprise Resource Planning）のような大規模な統合業務パッケージに取って代わる可能性がある．安心安全やコミュニティ系のサービスとしては，災害時のデータ収集と共有，あるいは日常的なコミュニティでのアンケート調査とその共有・データ加工等のサービスが考えられるだろう．さらにセンサリングと併せた統計・会計処理のサービスとしては，家計・ビル・コミュニティでのHEMS（Home Energy Management System），BEMS（Building Energy Management System），CEMS（Community Energy Management System）とそれぞれ呼ばれるエネルギーマネジメントシステムでのセンサリングに基づいたエネルギー統計や会計情報の提供サービスが考えられる．ライフログ系のサービスでもさまざまなライフログの収集と利用のシステムが，さらに工場系のサービスでも，センサネットワークを用いた，リアルタイムの生産計画や原価計算，エネルギー会計等が考えられる．このように生活世界や産業社会の諸サービスは多かれ少なかれ，IOEエージェントを利活用した自律分散協調システムとしてデザインできる．それを実世界で設計，実装，制御するプラットフォームが実世界OSとなる．

コンピュータが出現する以前から，複雑な組織の運営を通じ深化してきた複雑なシステムのマネジメントの原理はまだ十分に抽出されたとは言いがたい．組織の現場がそれぞれ認識しているタスク処理のプロセスを全体として繋ぎ合わせさらにそれを逐次改変しながら運用できる，頑健性と可視性を併せ持っ

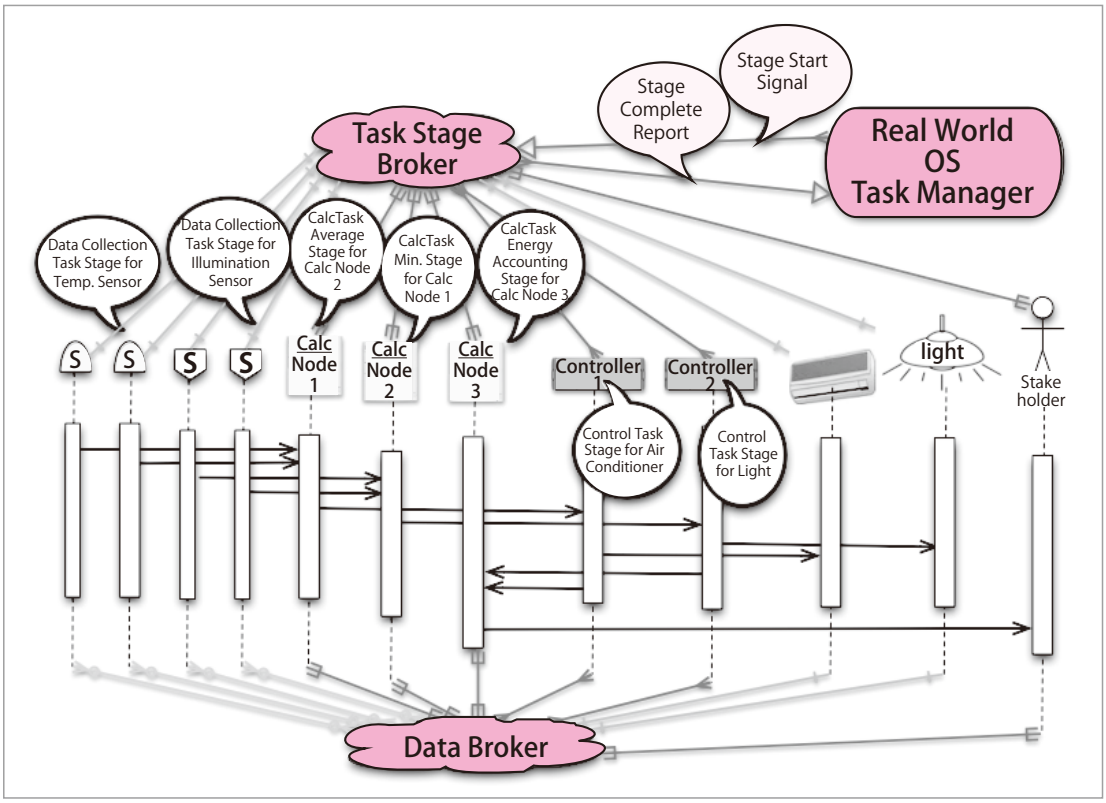

図-4　タスクマネージャによるタスクステージブローカの制御構造

たシステムデザインが求められる．実社会のシステムは，毎年その遂行タスクを修正し，タスクを実行する資源である人が次々に入れ替わるという中でその複雑なワークプロセスを維持している．このような複雑性の処理手法を情報処理システムに取り入れることは，組織や社会のアーキテクチャ・デザインとしてきわめて妥当である．本稿ではバーチャル世界でのマルチエージェント・シミュレーションから，それを部分的に実世界のエージェントと入れ替えたエミュレーションによるデプロイメント，さらに実世界のエージェントに対する自律分散協調制御のためのプロジェクト・プログラミングのコントロールという，バーチャル・リアル連携を一気通貫に行うアーキテクチャのビジョンを解説した．そこでは実世界のIOEエージェントの複雑な相互作用をバーチャルにデザインし，エミュレーションによるテストを行いつつ徐々にデプロイし，最後は実世界のエージェントの自律分散協調制御を行うという新しいシステム開発の手法も論じてきた．IOEによって構築される次の時代の人工物としての社会や組織をボトムアップにデザインするためのシステム方法論として，H. A. Simonに始まる組織の情報処理アプローチが，新たな形で再構築されることが，このリアルワールドOSのビジョンの中で求められている．

参考文献

1) シスコ IoT インキュベーションラボ, Internet of Everything の衝撃 IoT/M2M 基盤上で人・モノ・データ・プロセスがつながる.
2) 出口　弘, 市川　学, 石塚康成, 志手一哉, 染谷俊介, 湯浅洋一: 並列プロジェクト・タスク処理への多能工割付けの動的スケジューリング，国際P2M学会誌，国際P2M学会，Vol.6, No.1, pp.179-189 (Oct. 2011).
3) 出口　弘：研究のための方法論—社会システムの制度デザインの方法論：政策科学の方法としてのエージェントベースモデリング＆シミュレーション，計測と制御，Vol.52, No.7, pp.574-581 (2013).
4) March, J. G. and Simon, H. A. : Organizations, Wiley (1958).
5) 大貫裕二：国民経済計算の推計と利用を支援する手法に関する研究，東京工業大学大学院総合理工学研究科知能システム科学専攻博士（学術）学位請求論文 (2013).
6) Simon, H. A. : Administrative Behavior, The Free Press (1957).
7) 田沼英樹，出口　弘：エージェントベース社会シミュレーション言語SOARSの開発，電子情報通信学会論文誌D, Vol. J90-D, No.9, pp.2415-2422 (2007).
8) 出口　弘，田沼英樹，市川　学：社会シミュレーションSOARS,『ロボット情報学ハンドブック』, pp.616-626 (2010).

(2014年3月7日受付)

■ 出口　弘（正会員）　deguchi@dis.titech.ac.jp

東京工業大学大学院総合理工学研究科教授．理学博士．博士（経済学）．1955年5月生．福島大学助手，国際大学助教授，中央大学商学部助教授，京都大学経済学研究科助教授を経て，2001年より現職．