シミュレーションにAIとオブジェクト指向はどう生かされるか

室　善一郎

1. はじめに

近年の製造現場においては多品種少量生産の名の下にＣＩＭを柱として高度なプロダクション・コントロールがなされつつある．このような製造現場において，効率よく生産計画の立案や，設備改善のための投資を行なうためには，シミュレーションによる試行が重要な役割を果たすようになってきている．

しかし，高度なプロダクション・コントロールを行なっている設備をモデル化するには，①対象とする範囲が広い，②モデルの中を動く物の動作が，モデルの中の他の要素の状態によって左右される，③より知的な判断（たとえば，エキスパート・システムで行なわれるような判断）をモデルに取り込みたい，などの理由により多大な開発工数が必要とされる．あるいは，従来のシミュレーション言語では，モデル化が全く不可能なことも起こっている．

一方，コンピュータの技術はソフト・ハードとも競いあって日々進歩してきている．ハードの点から考えると，最近のＥＷＳは，従来の大型機程度の速度をもちながら，低価格であり，しかも，マウス，アイコン，ポップアップ・メニューなどの充実したＭＭＩを備えており，使用する側にとっては画期的な技術の進歩をとげている．

また，シミュレーション技術の発展の歴史を振り返ってみると，第1世代のシミュレーション言語といわれるGPSS, GASP, SIMSCRIPT の時代から，アニメーションをはじめとしたＭＭＩを充実させた言語であるGPSS/H, SLAM&TESS, Automod, SEEWHY&WITNESS, CIMAN&CINEMAなどの第2世代の言語へと移り変わってきている．[1]

ＡＩ技術においても，第1世代の言語といわれるＯＰＳ5，ESHELL の時代から，推論機構を充実させたＫＥＥ，ＡＲＴなどの第2世代の言語へと移り変わってきた．これらの言語は，知識のない推論エンジンのみを備えた言語であったが，第3世代のＡＩ言語では，ある問題領域に特定した知識を備えたドメイン・シェルの時代へと進歩しつつある．

さらに，プログラミングの方法論においては，従来の『ああして，こうして，こうしろ』この手続き型から，『このときは，こうである．そのときは，こうである』の非手続き型へと進化しつつある．その代表がルールベース・プログラミングであり，オブジェクト指向プログラミングである．ルールベース・プログラミングはＡＩの世界から，オブジェクト指向プログラミングはシミュレーションの世界から生まれた概念である．

オブジェクト指向の概念は，物を独立に捉えることからシミュレーションに適しており，オブジェクト指向言語として有名なSIMULA, Smalltalk, KEEは，簡単なシミュレーションを実現する機能を備えている．

そこで，より複雑なシミュレーションを，ごく自然にモデル化するために，最近のプログラミング・パラダイムがどのように生かされるか，というのがこの小論のテーマである．

2. シミュレーション言語の要件

一般にシミュレーションを実現するための言語は次の要件を必要とする．[2]

①シミュレーションの時刻を進めていくための時間管理機能
②システムの構成要素を記述するための機能
③システムの状態変化を記述するための機能
④構成要素の集合を取り扱うための機能
⑤与えられた確率分布にしたがう乱数列を取り扱う機能
⑥統計値を収集し，結果を編集する機能
⑦効率よくデバックするための機能

これらの機能は，最近のプログラミング・パラダイム

むろ　ぜんいちろう　川崎製鉄㈱　千葉製鉄所　企画部
〒260 千葉市川崎町1

表 1　Smalltalk 開発のコンセプト

把握しやすい	把握しにくい
具体的	抽象的
目に見える	目に見えない
模倣する	創造する
選択する	空白を埋める
認知する	生成する
エディットする	プログラミングする
対話型	バッチ処理

によりどのように実現されるのだろうか.

ここでは，LISP をベースとしたオブジェクト指向プログラミングの方法論によって実現されたエキスパート・システム開発環境ＫＥＥ，およびＫＥＥ上で稼働するシミュレーション・ツール SIMKIT を題材に以上の問題について考えてみたい.

KEE&SIMKIT のシステムは，すべて LISP で記述されており，SIMKIT の中で推論を起動したり，LISP 関数をダイレクトに利用したりすることができるシステムである.

3.　より自然なモデル表現

3.1　オブジェクト指向プログラミング

オブジェクト指向プログラミングは，1968年にノルウェー計算センターで SIMULA67 として設計されたのが最初である．SIMULA67は汎用システム言語であるが，その前身の SIMULA Ｉがシミュレーションを目的としたものであったことから，シミュレーション・プログラムがオブジェクト指向プログラミングと密接に結びついていることがわかる.

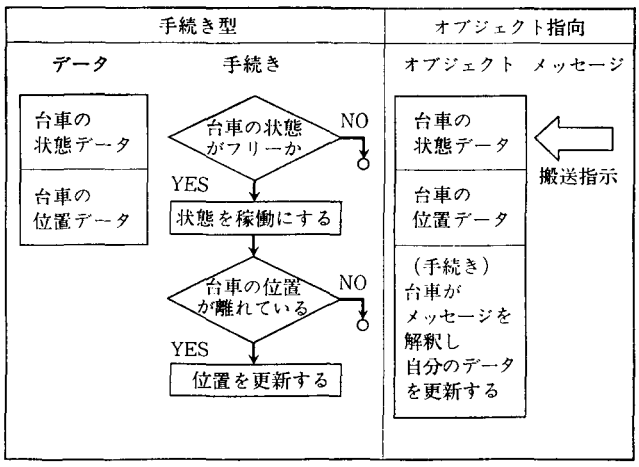

図 1　オブジェクト指向プログラミングの概念

また，オブジェクト指向言語として有名な Smalltalk は，SIMULA の影響を受けて開発されている．Smalltalk はＯＳをはじめとした全体システムを，オブジェクト指向プログラミングとして統一している．そして，その開発コンセプトには，表 1 に示すような思想があり，把握しにくいものをできるだけ排除しようとしたのである．(正確には，Smalltalk の前身である Star 開発のコンセプト) [3]

この思想は，KEE&SIMKITにも受け継がれている.

オブジェクト指向プログラミングは，従来の手続き型プログラミングとは異なり，データと手続きをまとめた単位（これをオブジェクトと呼ぶ）をベースとして問題を取り扱う.

たとえば，シミュレーション・モデルにおいて自動搬送台車をモデル化する場合を考えてみると，図 1 のようになる.

台車に対して搬送指示を出す場合，手続き型言語ではデータと手続きが分離されているため台車のデータ構造をよく知っていないと手続きが記述できない．(たとえば，台車マトリクス i 行 j 列の値が 1 の時は稼働中であり，2 の時が故障中であり，0 の時が待機中であるなど.)

これに対し，オブジェクト指向プログラミングでは，搬送指示をメッセージとして台車に送ることにより，台車がメッセージを解釈し，自分のデータを更新するのである.

シミュレーション・モデルにおいて，台車を使いたい時は，台車のデータ構造を知る必要はなく，搬送指示，行先指示，待機命令などのメッセージを送るだけで台車をシミュレーションの部品として利用できる.

このように作られた個々の部品を，メッセージのやりとりだけにより動かしていくのがオブジェクト指向プログラミングによるシミュレーションである.

また，個々のオブジェクトは，同じ性質を持つもの同士がクラス（上位概念），サブクラス（下位概念），インスタンス（実体）として階層構造を取ることができる．そして上位の性質は下位の概念に継承されてゆくことができる．これがインヘリタンス（継承）といわれる概念である．モデルを開発する時には，この階層構造を意識してオブジェクト，すなわちシミュレーションの中で使われる部品を作っていく.

このような階層構造と，各部品の独立性は，

モデルの管理を行なう上で大きな役割を果たすのと同時に，現実の世界に近いモデルの記述が可能になるのである．

3.2 リストを処理するデータ構造

FORTRAN のプログラムはアレイやマトリクスなどの静的データ構造しか取り扱うことができないのに対し，LISP は，リスト，キュー（先入れ先出し型データ構造），スタック（先入れ後出し型データ構造）などの動的データ構造を取り扱い，ＫＥＥは，ツリー（階層型データ構造）や，ユーザーが自由に定義できる新しいデータの型（たとえば，STATUS という変数は，BUSY か IDLE という値しか代入できないという指定ができる）を取り扱うことができる．

動的なリストを扱うことができれば，将来の事象発生の時間管理や，待ち行列に繋がれている物の操作が非常に簡単になる．（たとえば GPSS では，色が赤いものを待ち行列から取り出しなさいという指定はできるが，色が赤くて丸いものを取り出しなさいという指定はできない．）

これは，シミュレーションを実行する上において，在庫の検索，在庫の品種・納期別ソート等を容易にする．

また，新しいデータの型を定義できることはモデルの状態を表現するのに最適である．たとえば，搬送台車の状態という変数を，空走行，積み走行，降ろし中，載せ中，待機中という値しかとらないと定義すれば，この変数に3.14などという意味の不明な値を代入することもなくなる．さらに，この変数の値が不明な時は，取り得る値のリストの中から選択しなさいという指定もできるのである．

全体がリスト処理型の言語であるため，FORTRAN のように変数名が６文字以内であるという制限はなく，わかりやすい変数名を充分な長さで記述できることも大きな利点である．

3.3 条件付き割り込み（デーモン）

条件付き割り込み（デーモン）とは，システムが，モデルの中で参照・更新される変数をモニターし，その値が指定されたトリガーをキックした時に，隠されていた手続きを呼び出す機能である．これは，ＡＩの世界で生まれた制御構造である．

シミュレーションの実行においては，在庫が適正レベルを下回った時に，原材料を発注するとか，台車が危険区域に入ったら警告をだすなどの表現に適している．

また，待ち行列の長さをモニターしていて，その長さが変化するたびに統計値を収集するという機能の実現にも適している．

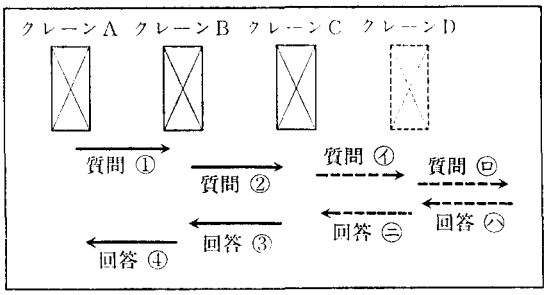

図 2 クレーン干渉時の質問と回答の流れ

一番有効なのは，シミュレーション・モデルの中で更新される座標をモニターし，アニメーションを起動することや，ある変数の値を常にディスプレイに表示する機能を実現できることがデーモンの大きな利点である．

3.4 関数の再帰的呼び出し

関数の再帰的呼び出しとは，関数の中で自分自身を呼び出すことができる機能である．

関数の再帰的呼び出しは，シミュレーションの中で行なわれる判断ロジックの記述を容易にし，人間思考を助けてくれる．

ここでは，天井クレーンの干渉判断を関数の再帰的呼び出しによって記述した例について紹介する．

図 2 は，同じヤードの中でクレーンがN台あるときの質問と回答の流れを示している．

まず，クレーンＡが図の右へ行きたいとする．質問①は，クレーンＢに対して右走行してもよいか尋ねている．

質問①を受けたクレーンＢは，右走行可能かどうかを質問②でクレーンＣに尋ね，回答③を受ける．そこで，クレーンＢは，クレーンＡの状態と，クレーンＣの回答③と，自分自身の状態をもとに，判断を下し，クレーンＡに回答④を返答する．

クレーンＡについて考えてみると，質問①に対する回答は，「待て」・「右走行可能」・「戻れ」のうちの１つである回答④だけである．クレーンＢやクレーンＣの状態を知る必要はない．

クレーンＣについて考えてみると，クレーンＢから受ける質問②は，クレーンＢが右走行可能かどうかという質問であり，前述のクレーンＢが，クレーンＡから受けた質問に対する動作と同じ動作をすることになる．すなわち，クレーンが右側にあるかぎり順次同じ質問を繰り返していくのである．

「右走行可能か？」という質問は，関数として定義さ

れるが，この関数の中でまた「右走行可能か?」という関数を呼び出している．クレーンが右側にあればあるほど，この関数は深くなっていく．このような関数の記述が，再帰的定義である．

3.5 条件表現，推論機構

複雑なシミュレーションを実現するための自由度は，何によって確保されるのだろうか．

一般のシミュレーション言語は，この自由度を確保するために，FORTRANのサブルーチンを呼び出す機能と，システムが管理するモデルの部品の状態を参照する機能を備えている．

しかし，手続き型言語であるFORTRANは，複雑なロジックを記述するのに適していない．

その点 LISPは，ロジックを記述するのに適した言語である．さらには，ＫＥＥの推論機構をシミュレーションの途中で呼び出すことも可能である．

シミュレーションによるケース・スタディを行なっていく上で，ＫＥＥの推論エンジンを起動して，最適化していくことも可能である．

たとえば，窓口の数と待ち行列の長さをエキスパート・システムによって評価・ケース設定を行ない，サービス時間や，窓口の数を最適化していことが可能である．

エキスパート・システムとシミュレーションが，KEE &SIMKIT の上で統合されているのである．

3.6 対話型プログラミング環境

①ポップ・アップ・メニュー，②ビット・マップ・ディスプレイ，③マウス，④アイコンなどの概念は，Smalltalkプロジェクトより生まれた概念である．

そして，KEE&SIMKIT は，このような機能とともに，表1に示したSmalltalkの開発コンセプトを受け継いでいる．

SIMKIT がサポートするモデル・エディターと呼ばれる機能は，マウスのみでシミュレーション・プログラムを作成することを可能にしている．

また，アクティブイメージというデーモンをベースとした機能によって，モデルの変数の値をマウスの操作のみによってディスプレイ上に表示することができる．この機能により，シミュレーションで「何がおこっているか眺めること」ができる．

さらに，KEE&SIMKIT はインタープリターであるため，「モデルを作る前に試すこと」も可能である．この点で，まさしくプロト・タイピング・アプローチが可能なのである．

そして，シミュレーション・モデルの構築にさいして新しい部品を作成する場合，オブジェクト指向プログラミングのクラスの概念にしたがって，新しい部品の上位概念を探し，その上位概念の機能を継承しながら新しい性質を追加していくのである．新しい性質の記述は，その性質に似た性質を模倣し，修正することにより行なっていく．これが，オブジェクト指向プロプログラミングが，従来のプログラミングより少ない記述量ですむ所以である．

3.7 アプローチ方法

シミュレーションのアプローチ方法には，GASPや，SIMSCRIPT のような「イベントにもとづくアプローチ」と，GPSSや，SLAMのような「プロセスフローにもとづくアプローチ」がある．[4][5][6]

オブジェクト指向プログラミングによるシミュレーション・モデル構築のアプローチは，GASP 的なイベントにもとづくアプローチと似ている．状態変化をおこす事象（これをイベントという．オブジェクトの性質として記述される）をすべて記述することによって，シミュレーションを進めていくアプローチである．

GASP と異なる点は，データと手続きをまとめた単位を部品として取扱い，その部品全体を階層構造として開発していこうとするアプローチをとる点で異なっている．人間がものを識別する場合，共通の性質を持っている物どうしをまとめていくつの概念を構築し，異なる概念が現われた時に，いままで分類された概念と似た概念を探し，その相違点だけを分類し，識別しようとする．このような思考が，オブジェクトの階層構造と一致するのである．このような人間の思考に近いという点で，オブジェクト指向プログラムによるアプローチは，より自然なのである．

モデル化という作業は，一般に抽象化という作業と同等であるが，シミュレーション・プログラムの作成という点では，シミュレーション言語が提供する限られた機能やコマンドのうちどの機能を使うかという概念として使われることが多い．

たとえば，GPSSでは，設備をトランザクションにするかファシリティにするかといったことがモデル化と呼ばれている．これは，トランザクションや，ファシリティが抽象的でありすぎるためと考えられる．

オブジェクト指向プログラミングによるアプローチは，設備やモデルの中を流れる物を概念としてより具体的に構造化することをシステムから要求される．また，

限られたコマンドでモデルを作るのではなく，コマンドをLISPの関数でユーザーが記述できる点も従来のモデル化というステップから解放してくれる．

そのため，モデル化というステップは，まさに抽象化というステップとして位置づけられる．この点でも，より自然に近いプログラミングが可能となってくるのである．

3.8 欠　点

KEE&SIMKITによれば，複雑なモデルを，少ない記述量で短期間で構築することが可能である．

しかし，モデルを説明する資料を作るのは，モデルを開発する以上に難しい．

従来の手続き型の思考に馴れた人に，①宣言型で記述されているイベント，②オブジェクトの階層構造，④メッセージのやりとり，④データと手続きが一体となったオブジェクト，⑤再帰的に作られた関数を説明することは相当な時間を要する．

これは，物の流れ，制御の流れ，時間の流れを統一的に記述することが難しいからである．

作るのは簡単だが，説明するのは難しいというのが現在の問題点と思っている．

また，EWS上でLISPをインタープリターによって動かしている関係上，シミュレーションの実行速度は遅い．

しかし，リアル・タイムで使うシステムとは異なり，オフラインで実行するシミュレーションとしては，我慢できる範囲と思っている．

4. ループ台車による搬送システムへの適用事例

KEE&SIMKITを用いたシミュレーションの事例として，熱延工場のコイル冷却ヤードのループ台車によるコイル搬送システムに適用した事例について紹介したい．

図3は，対象となる設備のレイアウトである．

このループ台車は，上工程であるホット・ストリップ・ミルからコイルを受入れて，指定されたコイル冷却ヤードへ搬送する作業と，下工程である熱仕ラインへ数日間冷却された冷却済コイルを搬送する作業を行なう台車である．この台車への載せ降ろしは，天井クレーンを通じてのみ行なわれる．

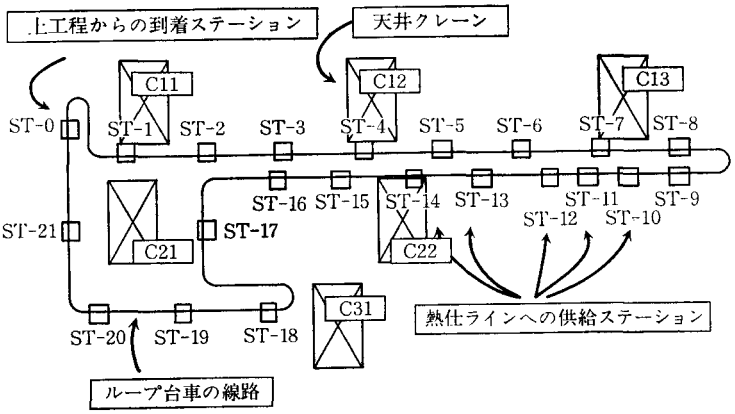

図3　コイル冷却ヤードのレイアウト

シミュレーションの目的は，ループ台車の台数と，クレーンの台数を適正に算定することである．

このシミュレーションの最大の特徴は，クレーンが独自に判断ルールを持っており，台車の動きと連動して作業を行なうことができるようにした点である．

そのルールは，ホット・ストリップ・ミルからのコイルの受入れにあわせて，ちょうどいいタイミングで台車の上で待つようにし，さらに，クレーンが独自に台車を予約して，冷却済コイルを吊って，ループ台車のステーションの上に到着すると，ちょうどいいタイミングで予約した空台車が到着するようなルールである．

この判断ルールは，シミュレーションにより経験を重ね，その経験から得られる知識をルールとして記述していったものである．

このモデルにおいても，それぞれ部品は完全に独立しており，部品の再利用を可能としている．そして，個々の部品は，宣言型のルールによって動いているのである．

5. ま と め

2.で述べたシミュレーション言語の要件は，KEE&SIMKITにより，より自然に近い形でモデル化できることを紹介した．**図4**参照．（乱数発生機能は，手続き型が優れていると思っている）

その根底を流れているのは，表1に示したSmalltalk開発のコンセプトである．

AI（エキスパート・システム）の世界は，ある問題領域に対して専門家の推論の過程をモデル化しようとするものである．そして，AI言語の世界は，シミュレー

ション言語の世界より進んだ機能を作り続けている．

対象となる世界をモデル化する上においては，ＡＩもシミュレーションもベースは同じであり，その両方のいい所を捉えてゆくことが重要であろう．

6. おわりに

KEE&SIMKIT の上でシミュレーション・モデルを作ってゆくことは，おのおの独立したオブジェクトの階層構造を作っていくことである．そして，このオブジェクトの持つ性質（手続き）は宣言型で記述されており，これはまさしく，シミュレーションに関する知識を記述することである．

すなわち，シミュレーションに関するエキスパート・システムを作っているのと全く同じではないだろうか．

従来個人のノウハウであったシミュレーションのモデル化（限られたコマンドをうまく使うという意味でのモデル化）というステップが，KEE&SIMKIT の上でエキスパート・システム的に標準化されるのである．

データと手続きが分離された従来のシミュレーション言語によって作られたモジュールは，再利用が難しく，モデルを開発するたびに似たようなモジュールを作っているのが常であった．（一部では，モジュールの再利用の研究も行なわれている）[7][8]

しかし，オブジェクト指向プログラミングの概念によって，モジュール化が進み，モジュールの再利用が容易になるのである．

そして，そのモジュール（すなわちオブジェクト）の階層構造が，より具体的な概念として広がりを見せた時に，シミュレーションのプログラミングは，誰にでも簡単にできるようになるのである．

現在我々は，これまでのシミュレーションのノウハウを標準化して，鉄鋼の物流シミュレーションに問題領域を絞ったオブジェクトの階層構造を作りつつある．第3世代のシミュレーション言語とは，このようなものではないだろうか．

参考文献

[1] 森戸 晋：離散系シミュレーションの最近の動向，シミュレーション，7/2，91-101，1988

[2] 春木良且：オブジェクト指向への招待，啓学出版，1989

[3] Howard Levine, Howard Rheingold 著，椋田直子訳，コンピュータ言語進化論，アスキー出版局，1988

[4] 山本喜一，浦 昭二：離散系シミュレーション言語の現状と将来の展望(1) GPSSによるモデル化，情報処理，Vol. 22，No. 9，839-845，1981

[5] 山本喜一，浦昭二：離散系シミュレーション言語の現状と将来の展望(2) SIMULA, SIMSCRIPTによるモデル化，情報処理，Vol. 22，No. 11，1012-1023，1981

[6] 森戸 晋，相沢りえ子：SLAMⅡによるシステム・シミュレーション入門，構造計画研究所，1986

[7] 室 善一郎：GPSSを用いたオブジェクト指向プログラミングの一方法論，ＯＲ学会春季研究発表会アブストラクト集，2-F-6，1989

[8] 住田修一，稲守久由：シミュレーション支援システム TEDAS-S，ＯＲ学会春季研究発表会アブストラクト集，2-F-8，1989

[9] 山本喜一：オブジェクト指向とシミュレーション，情報処理，Vol. 29，No. 4，374-381，1988

[10] アスキー書籍編集部編著，Smalltalk 入門，アスキー出版局，1986

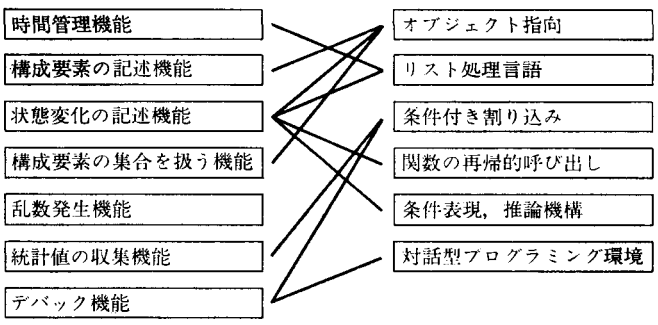

図 4 シミュレーション言語の要件とプログラミング・パラダイム