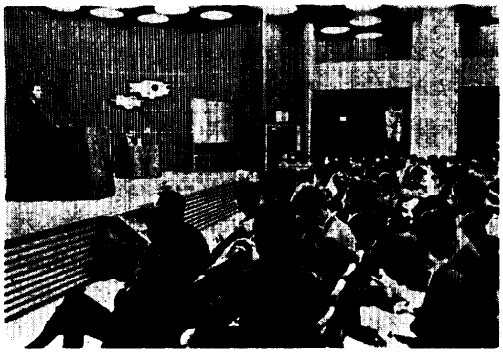
シミュレーション用言語の開発*

G. ゴードン**

1. 序

近年，問題解決の手法にシミュレーションがかなり多く使われるようになりました．シミュレーションそのものは決して新しいものではなく，何十年も前から，いろいろな形で，実用されてきております．変ったことといえば．電子計算機が応用できるようになったことです．計算機にいち早く眼をつけたのは自然科学や工学の分野の人達でした．中でも計算機科学や原子力やミサイル工学の研究者は，計算機を使ったシミュレーションにかなり影響されてきております．



いまあげた分野で，システムの問題を研究する時にはシミュレーションが有効な手段です．ミサイルを例にとってみまと，火器体系としてのミサイルの有効性は単にミサイルの威力だけでなく．発射装置や照準装置や後方からの補給力といったものを含めて考えて初めて評価できるのです．システムのエレメント相互の間にインターアクションがある場合は．上の例のようにシミュレーションは非常に有効な研究方法で，これが多く使われるようになったのは，システムの研究が盛んになったためと考えられます．

近年，シミュレーションの応用分野がさらに広がって参りました．使用頻度が多くなったため，シミュレーション用の言語が開発されるようになり，計算機でシミュレーションを行なうのが大変易しくなっております．この事実は，計算機の種類が増したことや計算コストが下ったことと相まって，シミュレーション人口の増大に大きく寄与しております．また，シミュレーションが自然科学や工学だけでなく，産業界の問題に応用されたり，経済学や社会学，さらには政治の分野の問題を把握するために使われるようになってきているのは見逃してはならない事実です．

シミュレーションを使って研究しているシステムの種類は多種多様ですし，問題として取扱われているものも広範囲に及んでいます．したがって，シミュレーションの応用手法は複雑多岐にわたります．しかし，シミュレーション用言語については，開発が進むに従い，かなりはっきりした方向が見出されております．私がこれからお話することは，現在使用されている言語を簡単に紹介するとともに，将来の方向について議論しようとするものであります．

2. シミュレーションの方法

私はシミュレーションを，次のように定義しています．“The technique of solving problem by following the changes over time of a dynamic model of a system” この定義に明らかなようにシミュレーションは本来実験方法の一種です．実際にシミュレーションを行なう場合には，系の中の一部の変数や関係に焦点をしぼることはなく，むしろ全部の変数を平等に観察してそこから何らかの結論を導く方法をとります．

シミュレーションの応用分野は大きく三つにわかれます．

システム・アナリシス：これは，現在あるシステムや，システムのタイプから一つを取上げて，シミュレーションによってその性質を調べようとする行き方です．使用するモデルに対応するシステムがわかっているので，モデルで取上げる変数はシステムの重要と思われる状態変数をよく反映しているのが特色です．

システム・デザイン：これは特定の条件を満すシス

---

* The Development of Simulation Languages.

** Geoffrey Gordon (New York Scientific Center, IBM Corporation)

昭和42年10月12日（木）機械振興会館で，日本 IBM（株）後援にて開催

テムを作り出すために，シミュレーションを応用する場合です．システムデザインのモデルは，あらかじめ性質がわかっている要素を組合わせて，その相互関係を表現しようとするものです．工学関係では，システム・アナリシス型やシステム・デザイン型のシミュレーションが非常によく使われております．

システム・ポスチュレーション：（訳注：システムの構造を仮定して作ったモデルを使うシミュレーション）システム・ポスチュレーションは，工学以外で発展しつつあるシミュレーションの一方法であります．

システム・ポスチュレーションの型のシミュレーションは社会学，経済学，政治学，医学などの分野でおもに使われるという特色があります．この種のシミュレーションのモデルでは，実際の現象の因果関係がわかっていないにもかかわらず，現象については表面的な知識が既知であるのが普通です．シミュレーションによってモデルの動きを現象と合致させ，そのモデルから現実の世界を理解しようとするのが，ポスチュレーションの目的です．すなわち，シミュレーションの結果が現象と一致していれば，そのモデルと現実が合致しているという仮設が立てられるのです．さらに一歩話を進めると，ポスチュレーション・モデルを使うことによって，システムの中味がのぞけるばかりでなく，最初たてた仮設に裏付けを与え，ひいては，さらに細かな仮設をたててシステムの動きを詳細に調べることができようというものです．このような研究は人体生理などに広く応用されています[1]．その他，経済社会の研究にもシステム・ポスチュレーションがだんだんとり入れられてきております[2]．

3. 数式モデル

シミュレーションには，アナログ・モデルもいろんな型のものがたくさん使われています．アナログ・モデルでは実在するシステムを使って，他の実在するシステムのモデルを作るのが特色です．このよい例は機械系を電気系でシミュレートするもので，摩擦，ダンピング，慣性といった要素からなるシステムを抵抗，コンデンサ，インダクタンスをもって置き換えるわけです．このアナロジーはまた非常によく合致するシステムとして有名です．

シミュレーションを使う研究には，数式モデルに依存する部分が多いので，ここでは数式モデルに範囲をしぼって話を進めたいと思います．もっともアナログ・コンピュータは非常に数式モデルと縁が深いので，ここに含めて議論します．これは直流増幅器やハイドロリック・シリンダが加算や積分に使われていることを思いおこしていただければ理解できると思います．というわけで，アナログ・コンピュータにのるモデルは本質的に数式モデルであります．換言すれば，アナログ・コンピュータは数式モデルを解く方法の一つと考えられます．

歴史的にシミュレーションは，変数が時間に対して連続的に変化するものと，不連続的に動くもの（離散系）の2種に分けられています．アナログ・コンピュータは，同次線型微分方程式をとくのに適しているので，連続系のシミュレーションに，広く用いられており，一方，離散系はディジタル・コンピュータのよい問題として扱われているわけです．この二つのシステムの境界は，最近とみにぼやけてきました．というのは，元来，連続系のモデルと不連続系のモデルの間にはっきりした区別があったわけではありませんし，一方においてアナログ・コンピュータの機能をディジタル・コンピュータにプログラムするための言語が発達してきたからです．

4. 連 続 系

連続系の数式モデルの中には，一連の不連続なステップを計算することによって，シミュレーションを行なうようなものもあります．Leontiff の Input-output モデルはその一つの例です[3]．これは経済学では有名なモデルであり，本来スタティックなモデルの範疇に入ります．一連の線型代数方程式を使って，種々の産業での原料と製品のバランスを解明しようとするものです．この種のモデルは，一定の時間間隔に従って計算をくり返えすことにより，ダイナミック・モデルとしても使えます．そういう形に変型したものを recurrence model と呼びますが，その数学的表現は本質的に Leontiff のものと同じです．ある状態変数を時間 t について求めるのに，他の時間 t における変数の値1個と，時間 $t-\Delta t$ における種々の変数の値を用います．このモデルについて研究した人の中では，G.H. Orcutt[4] が有名です．

元来この種のモデルは手計算のために作られたのですが，ディジタル・コンピュータにのせられるようになってからは，大きなモデルが扱われるようになり，かつ，確率論的変動がもちこめるようになりました．また，モデルの用途も経済学ばかりでなく，Urban Development すなわち地域社会の研究にも広く使わ

れております[5].

この種のモデルは計算の過程が単純であるため，これから議論するような本格的なシミュレーション言語には，とうとう結びつきませんでした．しかし，大量のデータの取扱いを必要とする上，統計的手法が種々と要求されるので，そのためのプログラミングシステムが開発されております．そのよい例は SPAN[6]で，urban study のために System Development Corp. が開発したものです．言語とはいえませんが，大量のデータを含むモデルの取扱いと分析法を一般化したオン・ライン・ライブラリーとして注目に値いします．

複雑な連続系のシミュレーションでは recurrence model は不十分となります．このような場合には定数係数の線型微分方程式がモデルを作るのに使われることが多いようです．微分方程式を使ったモデルはアナログ・コンピュータでとくのが従来の方法ですが，前にのべたように，いわゆるディジタル-アナログ言語を使って，ディジタル・コンピュータでとくこともできます．ディジタル-アナログ言語については，雑誌 Simulation に良い参考文献がのっております[7].

ディジタル-アナログ言語に使われている手法は，基本的にはアナログ・コンピュータに使われているものと同一です．まず，システムの状態変数に全部名前をつけ，それを使って関係式の中の最高次の導函数を始めとして順次他の変数で書きあらわします．いま，低次の導函数のその時の値がわかっているものと仮定すると，数値を代入することにより求めようとする変数の値が計算できます．順次，次数の低い導函数に対し同じことをくりかえせば，連立微分方程式がとけることになるわけです．もちろん必要な初期値を最初に系に入れてやる必要があります．

ディジタル-アナログ言語には，幸いなことにアナログ・コンピュータの欠点を埋合わせるような特徴があります．まず，計算の精度が上げられますし，ドリフトや変数のスケーリングなどの問題もなくなります．さらに掛算や割算のような非線型要素が使えるので，非線形の微分方程式をとけますから，事実上あらゆる型の微分方程式型モデルに使えます．最近，開発された IMB の Continuous System Modelling Program[8] はディジタル-アナログ言語の良い例で，フレキシビリティに富んだ使いやすいプログラムです．

しかし，ディジタル-アナログ言語が，アナログ・コンピュータにすっかりとって替わると考えるのは早計です．アナログ・コンピュータの精度で十分な問題がいくらでもありますし，計算速度の面でもくり返えしを必要とするディジタル・システムより有利な場合があります．それに加えて，最近技術的な開発が進んでいるハイブリッド・システム[9]は両者の長所を兼ね備えており，広範囲の問題に対処できるようになってきております．

この他にもアナログ・コンピュータは沢山の長所をもっています．シミュレーションが進行するのに応じてパラメータの値を変えたり，ディスプレイを通して中間結果をモニタできるなどがその例です．もっとも，上にあげた利点はディジタル・コンピュータの進歩につれて，ハードウェア[11]やソフトウェア[10]が完備すると差がつめられてしまいます．

5. インダストリアル・ダイナミックス

前にも指摘したように連続系と離散系の区別はシミュレーションを行なう上では本質的な問題ではありません．Recurrence システムは離散系のモデルで連続系のシミュレーションをしております．ここに述べようとするインダストリアル・ダイナミックスは連続なモデルで，離散系のシミュレーションをしている例です．

M.I.T. の Forrester らによって作られたインダストリアル・ダイナミックスはフィード・バック・ループの考え方を実業の世界に持ち込んだ独特な理論です．会社の中では，ある部門で下される決定は他の部門で行なわれる決定のもととなる情報の一部であることも考えると，部門の間に決定-情報-決定といったサイクルが起きることになります．すなわち，会社という一つのシステムはこうした情報のループが複雑にからみ合ったサーボ・メカニズムと考えられます．サーボ理論の手法を適用して，こうしたシステムの過渡応答や安定性を調べ，それから会社の業績と経営方法の関係を知るのがインダストリアル・ダイナミックスの目的であるわけです．

会社の内部での活動状況は細かくみると離散的な現実の集まりです．注文を出したり，仕事を一つ片附けたり，といったものがその例です．インダストリアル・ダイナミックスでは連続変数でそのような変化を表示します．たとえば，注文というアクションは発注率として，すなわち **RATE** として表わされるのです．RATE は一つ一つの注文によっておこるふらつきを時間的に平均して考えたものなので，積分を行なえば，発注量といった連続量に転化できます．積分された変

数をインダストリアル・ダイナミックスでは **LEVEL** と呼んでいます．これらの変数を使うと，モデルは単純な一次線型微分方程式の集まりとなります．このモデルを解くのには手計算でも，アナログ演算でも，あるいはディジタル－アナログ言語のモデルによっても良いのですが，実用上は M.I.T. で開発された DYNAMO[13] という言語が使われます．

インダストリアル・ダイナミックスは，産業界の問題を取り上げて，かなりの成功をおさめています．個々の例について文献を調べるのには参考文献の 12) が適当でしょう．

6. 離 散 系

離散系のシミュレーションの根拠となるような理論はありませんが，この分野で言語の比較をしたりするとき使われる術語には，いくつか定まったものがあるので，最初にそういった言葉について議論したいと思います．

離散系で通常見受けられる術語にはエンティティ，アトリビュート，アクティビティといったものがあり，既存のシミュレーション言語は直接または間接にその考え方を取入れております．いろいろな系の中で，そうした言葉が何を意味するかをまとめたものが**第 1 図**です．

TYPE OF SYSTEM	ENTITY	ATTRIBUTE	ACTIVITY
FACTORY	MACHINE	AVAILABILITY	MAKE PART
BANK	CUSTOMER	CREDIT STATUS	MAKE DEPOSIT
COMMUNICATION	MESSAGE	LENGTH	TRANSMIT

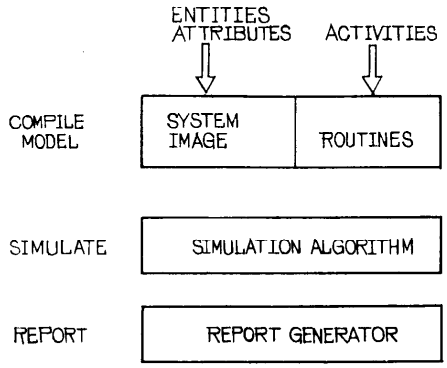
第 1 図 System concepts

エンティティは工場における機械，銀行では窓口にくるお客さん，通信システムではメッセージというように系の中で個々に区別できる要素です．アトリビュートはエンティティの性質を定める因子で，一つのエンティティには通常いくつかのアトリビュートがあります．機械が空いているか使用中かといった状態，お客さんの借金の状態，メッセージの長さなどはそれぞれアトリビュートの例です．アクティビティは通常エンティティと関連しており，そのエンティティがシステムの中で果している役割を指示します．機械は部品の加工をするためのものであり，銀行のお客さんは借金の支払いにきているのであり，また，メッセージは送信中であるなどのシステムの変化を説明するのがアクティビティであります．とりわけ，システムの中でおこる状態の変化は重要で，イベントと呼ばれます．機械に部品材をのせたり，新しいお客さんが窓口に現われたり，メッセージの送信が終るといった事象はその例で，システムの状態がその時点で変るので重要な意味を持ちます．

工学系の問題では，エンティティ，アトリビュート，アクティビティなどを定めるのはさして困難ではありません．しかし，社会学や経済学の領域の問題を扱うときは要素を分析するのに非常な注意が必要です．一つの例をあげると，ある経済社会モデルでは社会階層を一つのエンティティとして考え，個人個人をエンティティとして（実際はそうであるが）区別することを避けております．このモデルはアクティビティは数式的表現を用いています．たとえば，経済政策は各社会階層に対する平均所得の増減といった形でモデルに取り入れているのです．

7. 汎用シミュレーション言語

いままで言葉でシミュレーションのシステムを表現してまいりましたが，これをシミュレーション・プログラムの立場からみると**第 2 図**に示したような三つの



第 2 図 Requirements of a simulation language

タスクが考えられます．第 1 のタスクは言葉で表現されたものを数学的モデルにコンパイルすることです．エンティティとアトリビュートを変数の組合わせとしてとらえ，変数の値を記憶するのが，モデル・コンパイレーションの第一歩です．この変数の組合わせをシステム・イメージと呼びます．この変数がその時々のシステムの状態を示すのです．アクティビティはプログラムのルーティンとして翻訳されます．

シミュレーションを進めてゆくのには，時間の経過とイベントの内容によってシステム・イメージを変えてゆくことが必要です．そのためには，イベントがおきる時刻とそのイベントに関係している変数をレコードとして記憶してゆかねばなりません．

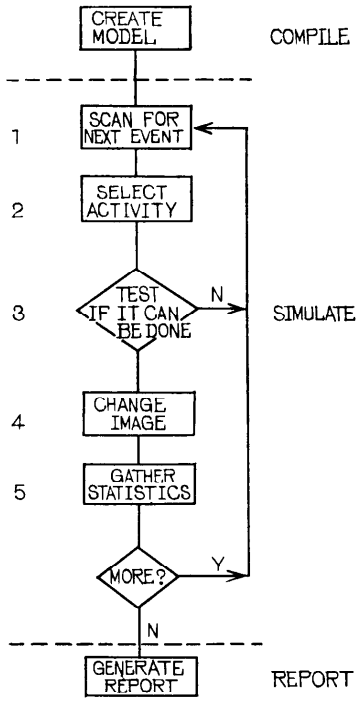
「システムを変化させてゆくのに，一般的にいって二つの大きな物の見方があります[14,15]．Particle oriented（または material based）と呼ばれる方法では，モデルの中のエンティティを中心にしてシミュレーションを進めてゆきます．エンティティがあるアクティビティから次のアクティビティに移ってゆくと考えるのがこの方法の特色です．そして状態の変化はアトリビュートの変化としてとらえられます．さらに一方の見方はまたは event oriented（または machine based）と呼ばれ，アクティビティを中心とするものです．シミュレーションはいろいろなアクティビティが順次エンティティの上に作用してゆく歴史として定義されます．この方法をとるときは，状態の変化はアクティビティの中で起きるものとされています．

どちらの考え方を採るにせよ，シミュレーションの第2段階はシミュレーション・アルゴリズムをプログラムすることです．このアルゴリズムは，まず状態変化に関するレコードを時間順にスキャンし，次におきる変化の時刻とアクティビティを定め，適当なアクティビティルーティンにブランチするといった動作を含む必要があります．

第3段階は，最後のステップでシミュレーションの結果をプリントするレポート・ジェネレータが働きます．これに使われる統計データを集める作業は，普通シミュレーション・アルゴリズムの中に含まれます．

次にシミュレーションのエクゼキューション(実行)段階を**第3図**に示しました．一番上のコンパイル部分と下のレポート・ジェネレーションの部分はそれぞれ1回しか実行されません．しかし，シミュレーションアルゴリズムはくり返えし実行されます．その内部は大要五つのステップに分割できます．

まず次のイベントを **SCAN** する必要があります．次にイベントを **SELECT** して所要のアクティビティ・ルーティンにブランチします．そこで，選んだイベントが論理的に実行可能か否かを **TEST** し，もし可能ならば状態の変化，すなわち **CHANGE** をシステム・イメージの中におこさせます．最後に統計データを集める **EXTRACT** のステップを通り，もし，くり返えしの必要があれば，もとのステップに戻るわけです．



第3図　Execution of simulation

以前はほとんど全てのシミュレーションが特別な言語を使わないで行なわれました．いまでも大部分のシミュレーションはそういった言語を使っていません．一般に使われていたのは機械語か FORTRAN, ALGOL のような科学計算用言語です．しかし，シミュレーションの応用が専門化するにつれて，そのための特殊な言葉がジョブ・ショップや在庫の問題のために開発されてきました．こうした言語は応用分野の専門語がそのまま使えるようになっている反面，専門知識のない人には非常に使いにくいものです．

近年になって汎用シミュレーション言語がいくつか作られるようになりました．どれをとってみてもエンティティやアトリビュートやアクティビティを表現する能力を備えており，かつ，さきほど述べた三つの基本的機能，コンパイル，シミュレーションの実行，レポート・ジェネレーションを行なっております．しかし，それぞれのワールド・ビューすなわち，システムの表現法はみな異なっています．したがって，言語の使用者はまずその言語のワールド・ビューを理解しなければなりません．また，それぞれの言語はさきほど

述べた五つのステップのどこをどれだけ自動化しているかという点でも異なっております．

今回は全部の汎用シミュレーション言語を比較することは致しませんが，もっと広くお知りになりたい方は文献 15, 16, 17）を参照してください．ここではたくさんあるうちワールド・ビューの点からも，自動化の程度からもかなり対照的な二つの言語を取り上げ，比較検討してみたいと思います．

8. GPSS

初めにとり上げるのは GPSS です．GPSS は 1961 年に[18] 作られ，最新の GPSS Ⅲ は IBM S/360 用に書かれております[19]．GPSS は particle oriented で，しかもすべてのステップが自動化されている例です．

GPSS のワールド・ビューが**第 4 図**です．おもなエ

ENTITIES	ATTRIBUTES
Transactions	Parameters Priority
Facilities (Time Shared Equipment)	Availability
Storages (Space Shared Equipment)	Capacity Occupancy

ACTIVITIES

Standard Block Types

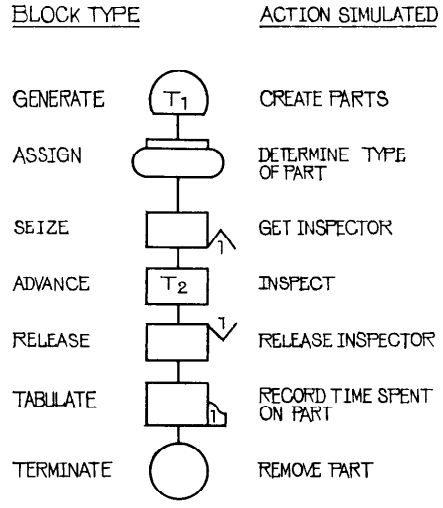
第 4 図 GPSS basic concepts

ンティティにはそれぞれトランズアクション，ファシリティ，ストレッジという名前がつけられています．**トランズアクション**はシステムの中を流れてゆく素子です．工場の中を流れてゆく部品はトランズアクションのよい例です．トランズアクションのおもなアトリビュートは**パラメータ**と呼ばれ，その内容は数字で示されます．部品の例を使うと型番とか注文番号がパラメータの例です．トランズアクションのアトリビュートには，この他**プライオリティ**があり，プライオリティ・スケジューリングが可能です．

他の二つのエンティティはファシリティとストレッジです．**ファシリティ**は一度に一つのものしか扱えない，すなわちタイム-シェアの形で使われる道具を表わします．一度に一つしか受けつけない工作機械は良い例です．**ストレッジ**はたくさんのものを並行処理できるもの，すなわち空間をシェアするものを示します．加熱炉はその良い例です．バッチ・プロセスはしたがってストレッジで示されます．ファシリティのアトリビュートは空いているか否かの状態であり，ストレッジのそれは収容量と現在量です．

GPSS の特徴は，ブロック・ダイアグラムにあります．システムのアクティビティはいくつかのスタンダード・ブロック・タイプをつぎ合わせて表現します．ブロック・タイプはそれぞれきまった型の動作（または機能）を代表しており，それぞれユニークな名前がつけられています．プログラムはこのブロック・タイプをつぎ合わせるだけですみます．トランズアクションをシステムの中で発生させるには，ある定められたブロックがあり，モデルはトランズアクションがブロックからブロックへとつたわってゆく形で表現されるので，非常にわかりやすいのが特色です．

GPSS で書いた簡単なモデルの例を**第 5 図**に示しま



第 5 図 GPSS simulation of inspection process

した．ここでは工場における検査工程が例示されています．一つ一つの部品が製造工程の手を離れて，検査工のところに流れてゆくわけですが，この検査は一度に一つずつしか行なえないようになっております．この例では検査に要する時間が部品の型によって異なっています．

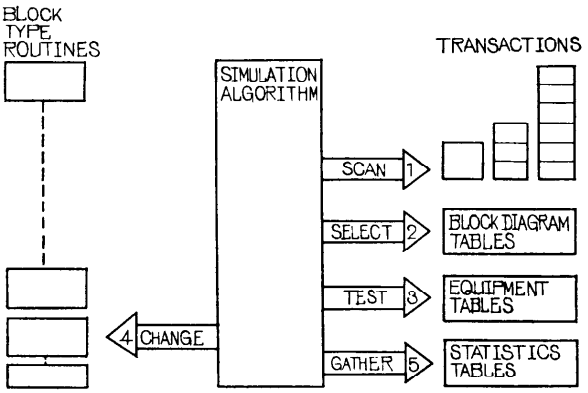
おわかりのように，部品がトランズアクションとして取扱われており GENERATE ブロックがトランズアクションの発生を制御しています．発生率は図に示

すT₁で定まります．ASSIGNブロックはプログラマが与えたデータに従って，そこを通るトランズアクションの型を定める役割をしています．ここでは検査工を一人と考えているので，それを表わすのにファシリティが使われています．最初にSEIZEブロックに到着したトランズアクションが，このファシリティ(No. 1)を専有するわけです．ファシリティは定義により，一度に一つのトランズアクションしか扱いません．したがって次のトランズアクションはその前のトランズアクションがファシリティを出るまで待たされることになります．GPSSのシステムはこのようなとき，自動的にそのファシリティに対する待行列を作り，さらにファシリティが空いた場合には，次のトランズアクションを送り込む働きをします．

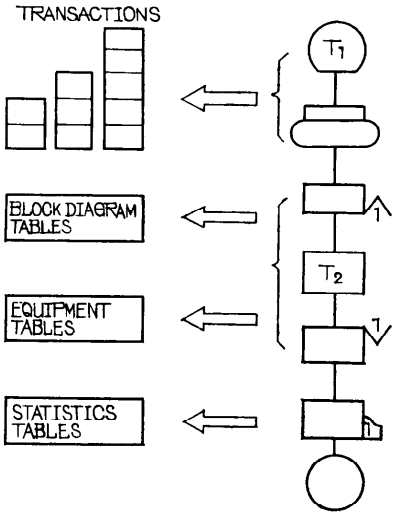
ADVANCEは時間の経過を示します．ここでは検査工が部品を調べる時間を示し，かつ，それが部品の型によって変化します．検査が終るとトランズアクションはファシリティをRELEASEします．検査時間の分布を知るために，終りにTABULATEブロックを加えて統計をとります．最後にTERMINATEによってモデルを流れ終ったトランズアクションをシステムから取り除きます．

以上のブロック・ダイアグラムを，カードにパンチし，コントロール・カードやデータ・カードを多少つけ加えればそのままGPSSのインプットができあがります．**第6図**はそのインプットからシステム・イメージが作られる過程を示したものです．システムの**データ・ストラクチュア**はGENERATE, ASSIGNなどのブロックをコンパイルすることによって得られます．個々のブロックに関するデータや，対応しているエンティティについてのデータは，いろいろなテーブルに収容します．統計をとるために使われるテーブルは，関係のある種々なブロックから集めたデータに従ってまとめあげられます．

以上がコンパイルの段階に行なわれる作業で，他のシミュレーション・システムと異なり，アクティビティ・ルーティンはプログラムしたりコンパイルしたりする必要はありません．GPSSではアクティビティはスタンダード・ブロック・タイプ・ルーティンの組合わせとして表現されているからです．



第7図　Execution of simulation in GPSS-A



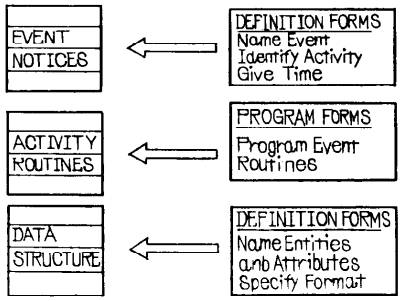
第6図　Compilation of model in GPSS

第7図に示した汎用シミュレーション・アルゴリズムはすべてGPSSの中に含まれているのでプログラムをする必要はありません．システムの状態を変えるイベントはトランズアクションの移動の形で表現されています．あるトランズアクションが，いつ移動するか，すなわちイベント・タイムはトランズアクション・アトリビュートの一つです．GPSSのシミュレーション・アルゴリズムはまずトランズアクションをスキャンし，実行すべきトランズアクションを探します．次にブロック・ダイヤグラムから移動先を探し，移動可能か否かを調べ，すべての条件が満たされればブロック・タイプ・ルーティンにブランチします．それが終って状態変化が起るとシステムは所定の統計量と，プログラマが指定した統計量を記録して次のスキャンを行ないます．

9. SIMSCRIPT

GPSS と対比して論議すべきものに SIMSCRIPT があります．これは 1963 年に発表され[20]，最近 SIMSCRIPT 1.5 へと進展してきています[21]．SIMSCRIPT 1 または SIMSCRIPT 1.5 はほとんどの大型計算機に対して書かれているので，使いやすいのが長所です．SIMSCRIPT は GPSS とは対照的に event oriented system として作られているので，特にここで取上げた次第です．

シミュレーションのワールド・ビューを論ずるとき使われるエンティティ，アトリビュート，アクティビティといった言葉は SIMSCRIPT に始まったものです．しかしながら SIMSCRIPT それ自身はエンティティ，アクティビティを自動的に定義したりするプログラムではありません．むしろプログラマが一つ一つそういった要素を定義する必要があります．その関係を示したのが**第 8 図**です．



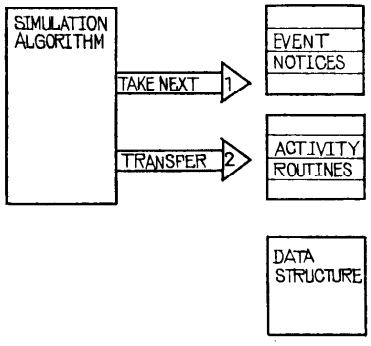
第 8 図 Compilation of model in simscript

SIMSCRIPT では，きまったコーディング・フォームを使ってプログラマがエンティティなどを定義するようになっています．アトリビュートに固定小数点のデータばかりでなく，浮動小数点のものも，使えますし．記憶装置を有効に利用するためにデータをパックする指示も使えます．これを DEFINITION FORM と呼び，SIMSCRIPT はシステム・イメージをこのフォームからコンパイルするのです．

アクティビティは PROGRAM FORM を使ってプログラムします．このプログラムはシミュレーションに必要な指定が行なえる特別な言葉を使って書きますが，それが FORTRAN によく似ているのが特色です．こうして書かれたアクティビティは，それぞれ独立したサブルーティンとしてコンパイルされます．

第 8 図の上の所に書きましたように，プログラマはそれぞれのアクティビティに対してイベント・ノーティスなるものをプログラムする必要があります．イベント・ノーティスに用いられる DEFINITION FORM はエンティティなどに対するものと同一です．イベント・ノーティスにはアクティビティに対応した名前がつけられることになっており，それから容易にアクティビティ・サブルーティンを呼ぶことができます．イベント・ノーティスにはそのイベントの時刻がつけられているほか，アクティビティ・サブルーティンに渡すデータがパラメータとして付属しています．イベントが起こる可能性がある時には，常にそれに対応するイベント・ノーティスが用意されているのが SIMSCRIPT の特色です．

第 9 図は SIMSCRIPT に含まれているシミュレーション・アルゴリズムを示したものです．SIMSCR-

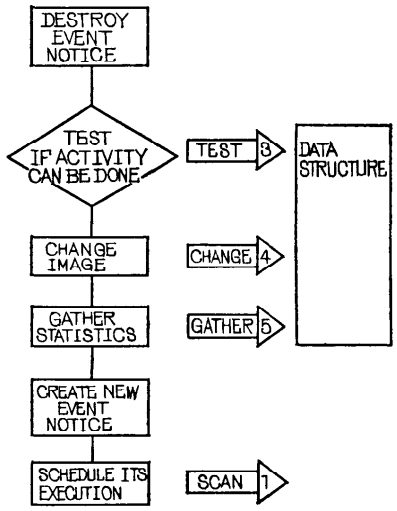


第 9 図 Execution of simulation in simscript

IPT はまずイベントノーティスを **SCAN** して次のイベントを定めます．実際のプログラムではイベント・ノーティスはすでに時間順に並べられており，アルゴリズムは単に最初のノーティスを拾い上げるだけです．ついでイベント・ノーティスの名前からアクティビティを割り出し，それにブランチします．このときパラメータとしてデータが渡されるわけです．

残りのステップについては**第 10 図**に示したような順に従って，プログラマが一つ一つのアクティビティ・ルーティンの中に書く必要があります．アクティビティを呼んだイベントノーティスは通常すぐシステムから取除かれます．プログラマは次いでそのイベントの実現性を **TEST** し，もし可能ならば先へ進むわけです．状態の **CHANGE**，総計データの **EXTRACT** はすべてプログラムの責任です．統計データを集めるテーブルなどに関しては最初データ・ストラクチュア

を定めるとき，一緒にきめておかなければなりません．



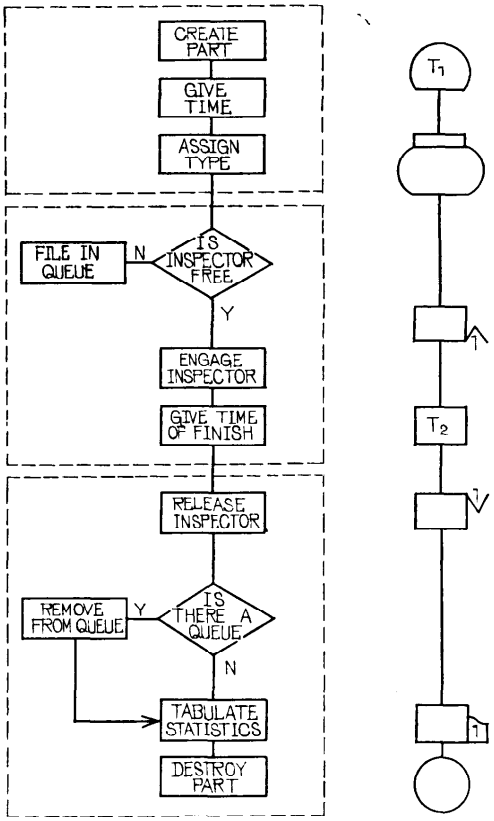
第 10 図　Execution of an event routine in simscript

最後にアクティビティ・ルーティンは次のイベント・ノーティスを作り出してシミュレーションが続行できるようにしなければなりません．これはアクティビティの名前と予定時刻を指定することで行なえます．次いでそれをイベント・ノーティス・リストに入れるための特別なステートメントをつけ加えれば一応アクティビティ・ルーティンは完了致します．実際のシステムではこの時イベントノーティスのソートが行なわれる（**SELECT**）ようになっています．

前にあげた検査工程の例を SIMSCRIPT で示せればよいのですが，GPSS のように簡単にゆきません．そこで**第 11 図**に SIMSCRIPT のプログラムが GPSS のブロック・ダイヤグラムとどのように対応するかについて示しました．

例の検査工程を思いおこしていただければわかるように，図の中には 3 種類のアクティビティが含まれています．

第 1 は部品のデータに関するレコードをシステムの中に作り出し，部品の型についてのデータを定める部分です．GPSS では GENERATE と ASSIGN のブロックがそれに対応します．第 2 は検査工程そのもので，GPSS では SEIZE と ADVANCE が対応します．SIMSCRIPT ではここで部品の待行列を取り扱うプログラムが必要です



第 11 図　Simscript simulation of inspection process

最後のアクティビティは，検査の終了を告げるもので，同時に統計データをとり，処理を終った部品に関するレコードをシステムから取り除く作業をします．ブロックでいえば RELEASE と TABULATE と TERMINATE が対応しています．SIMSCRIPT では再びここで待行列関係の処理が必要になりますが，これは案外やっかいなプログラムです．

10. 終　り　に

いままで議論した中にでてきましたように，シミュレーションの使われる研究分野や研究手段には実にいろいろなものがあります．一つのシステムにはそれに対する分析手法がいくつもあり，しかもその手法に従ったシミュレーション言語にも種類があるので，シミュレーションを単一の分野として考えるのは困難であります．

どんな研究についてもいえることですが，基本とな

ることがらに対する概念構成や定義をゆるがせにできません．進歩したシステムに対しては，さらに進んだ概念の構成を行なうことが必要であり．また当然のなりゆきであります．シミュレーションも論外でなく，そのような研究を行なうことが，シミュレーションの応用手法を明確なもの，簡単なものにしてゆくと思われます．

進んだ基本概念を作り出し，統一してゆくことも重要ですが，それと同時に現在あるシミュレーション言語の基本となっている概念を理解することはシミュレーションを実際使う場合に必要なことです．いわゆる汎用シミュレーション言語の場合は特にその基本構想を知るべきであります．いまのところ汎用とはいっても，その実，特殊目的のシミュレーションから出発したものが多いので，その言語の応用上の適応性を見分けることは，製作者にとっても大変むずかしいことなのです．

参考文献

1) Deland, E.C., and Bradham, Gilbert B.: Fluid Balance and Electrolyte Distribution in the Human Body, Proc. IBM Scientific Computing Symposium, Simulation Models and Gaming, IBM, White Plains, N.Y.
2) Orcutt, G., Greenberger, M., Korbel, J., and Rivlin, A.: Microanalysis of Socioeconomic Systems: A Simulation Study, Harper 1961
3) Leontiff, W.: The Structure of the American Economy, Cambridge, Mass. 1951
4) Orcutt, Guy H.: Simulation of Economic Systems: Model Description and Solution, Proc. IBM Scientific Computing Symposium, Simulation Models and Gaming, IBM, White Plains, N.Y.
5) Harris, Britton: The Uses and Theory in the Simulation of Urban Phenomena, Journal of the American Institute of Planners, XXXIII, No. 5, Sept. 1966
6) Almendinger, Vladimir V.: Span Reference Manual. System Development Corp., Santa Monica, Calif. 1965
7) Brennan, R.D., and Linebarger, R.N.: A Survey of Digital Simulation: Digital-Analog Simulation Programs, Simulation, Vol. 3, No. 6, pp. 23～36, Dec. 1964
8) CSMP, Application Description Manual, Form No. H 20-0240 IBM Corp., White Plains, N.Y.
9) Wait, John V.: A Hybrid Analog-Digital Differential Analyzer System, Proc. AFIPS 1963 Fall Joint Comput.-Conf., Las Vegas, Nev., 1963, pp. 277-293, Spartan Books, Baltimore, Md.
10) Brennan, Robert D., and Sano, Harlan: PACTOLUS-A Digital Analog Simulator Program for the IBM 1620, Proc. AFIPS 1964 Fall Joint Comput. Conf., pp. 299～312. Spartan Books, Baltimore, Md.
11) Bartee, T.C., and Lewis, J.B.: A Digital System for On-Line Studies of Dynamical Systems, Proc. AFIPS, Spring Joint Comput. Conf., Boston, Mass. 1966, Spartan Books, Baltimore, Md.
12) Forrester, J.W.: Industrial Dynamics, MIT Press, Cambridge, Mass., 1961
13) Pugh, A.L. III: DYNAMO, User's Manual, MIT Press, Cambridge, Mass., 1961
14) Tocher, K.D.: Some Techniques of Model Building, Proc. IBM Scientific Computing Symposium: Simulation Models and Gaming, IBM Corp., White Plains, N.Y.
15) Tocher, K.D.: Review of Simulation Languages, Operational Research Quarterly, Vol. 16, No. 2, pp. 189～217, June 1965
16) Krasow, H.S., and Merikallio, R.: The Past, Present and Future of General Simulation Languages, Management Science, Nov. 1964
17) Teichroew, Daniel, and Lubin, John Francis: Computer Simulation-Discussion of the Technique and Comparison of Languages, Comm. A.C.M., Vol. 9, No. 10, Oct. 1966
18) Gordon, G.: A General Purpose Systems Simulation Program, Proc. APIPS, Eastern Joint Comput. Conf. 1961, The Macmillan Company, New York.
19) General Purpose Simulation System/360, User's Manual IBM Form No. H 20-0326-0, IBM Corp., White Plains, N.Y.
20) Markowitz, H.M., Hausner, B. and Karr, H.W.: SIMSCRIPT, A Simulation Programming Language, Prentice Hall, Inc.
21) Karr, Herbert W., Kleine, Henry, and Markowitz, Harry M.: SIMSCRIPT 1.5, California Analysis Center Inc., Santa Monica, Calif., Oct. 1966