データベースシステム 93－8
システムソフトウェアと
オペレーティング・システム 59－8
（1993．5．28）

オブジェクト指向設計法における分岐制約処理の検討

＊龍　忠光，＊村川雅彦，＊豊田雅信，
＊足立武史，＊戸島哲夫

＊㈱富士通研究所　情報処理研究部門
＊富士通ネットワークエンジニアリング㈱　開発部

オブジェクト指向については、その利点としてシステム構築におけるソフトウェア部品の再利用性やシステムの保守性の良さが言われているが、リアルタイム向けのアプリケーションを考えた場合現状の設計法やイプリメント法においてはこれらの目的を充分に満足しているとは言いがたい。リアルタイム向けのアプリケーションを考える時、複雑な事象を容易にモデル化しこれを高速でリアルタイム的に動作させることが必要でありこの為には、部品としてのオブジェクトの独立性を高めるとともにこれらのオブジェクトを使って自由な組合せが出来、そしてその稼働環境に合わせて高速で動く最適な実行用のモジュール（ＥＸＥモジュール）を作ることが出来なければない。
富士通研究所ではこれらの観点に立ち、部品やシステムにおいて再利用や変更保守が容易に行える「リアルタイムアプリケーション向けの次世代システム構築環境」の研究開発に取り組んで来た。本稿では当該研究開発の主要な部分の一つである「オブジェクトの動作時における分岐制約処理」を中心にこのインプリメントの考え方について紹介する。

A proposal of implementation how to control the Object module using restricted condition

＊Tadamitsu Ryu, ＊Masahiko Murakawa, ＊Masanobu Toyota,
＊Takesi Adachi, ＊Tetsuo Tojima

＊INFORMATION PROCESSING DEVISION, FUJITSU LABORATORIES LTD.
＊FUJITSU NETWORKENGINEERING LTD.

Many reseachers are expecting that Object Oriented Approach can be improve a reusability of software module and a maintenability of software system. But, in the field of " Real time processing application ", there are not enough concepts or implementations to satisfy these merits. Therefore we must promote the independency of object modules and run its system rapidly in verious environment (even though in very small system : for example, poor memory workstation etc.). From this point of view, we have been promoting the concepts and implementation technics about the reusage of software mudules and maintenance of sotware system. In this paper, we mainly describe a implementation for restrictions (BRANCH CONTROL, LOOP CONTROL etc.) among the object modules which simplify the relations of modules.

1．はじめに

システム構築の生産性のキーポイントは、「問題の最適なモデル化」と「プログラムの再利用の徹底」である。前者については、そのモデリング設計のやり方としてコード／ヨードンのＯＯＡ(Object-Oriented Analysis)〔３〕やランボーらのＯＭＴ(Object Modeling Technique)〔４〕などが発表されている。これらの設計法はデータ項目及びデータの振る舞いの分析などの法則的な関係を中心に取り扱われており、対象モデルの動的な関係（同期取り）の取扱いが不明確である。また後者については、プログラムの部品化・再利用という観点からの有効なインプリメント法の提案はあまり行われていない。

前者について、筆者らはＯＭＴを改良し、実世界を「情報隠蔽の世界」と「オブジェクトの世界」（外延的に解釈できる外辞〔これを外延的外辞と呼ぶ〕の世界）の２つに分け、この２つの間を外延的外辞で結びつけ徹底した情報隠蔽を可能にするとともにオブジェクトーＩＤを工夫し、「オブジェクトの世界」における情報隠蔽の不便さを排除させた。更に「オブジェクトの世界」を静的モデルと動的モデルの２つの側面から捉え、これらの関連を整理することにより複雑な問題のモデル化を容易とした。　また後者について、筆者らは各オブジェクトの独立性の向上を図るため各オブジェクト間に必要な演算機能などについては当該オブジェクトに組み込むことを止めこれらのオブジェクトとは独立な因果関係オブジェクトを導入し、ここで各オブジェクト間の因果関係を定義することによりオブジェクトの独立性の向上を図った。

本稿では、これらのベースとなっているアクターオブジェクト指向，オブジェクトコマンド，ハイパー処理機能について概観した後、オブジェクトの独立性と同期取りの要となる因果関係の中の強制分岐を中心にその具体的なインプリメントの仕掛けについて述べる。

2．オブジェクト指向設計法の概要

筆者らは、オブジェクト指向設計法において徹底した人間指向を追求し「人間の直感的な世界」と「複雑な機能を機械的に処理するコンピュータの世界」との間を外延的外辞を介してリンク取りすることによってシステム構築者は「人間の直感的な世界」で設計できるようにし、従来のようなコンピュータの専門家を必要としないシステム構築を目指した。〔５〕〔６〕以下にその概要を述べる。

(1)実世界のモデル化

オブジェクト指向設計法は、人間の思考を模倣することによって、人間が実世界を直感的に把握しているやり方をそのままモデル化し、コンピュータが扱うモデルの世界に持ち込もうとするものである。我々は実世界の事象（オブジェクト）の性質や関係を考える場合、多くはその外延が意味する内包は仲間内では共有されており、外延のみでコミュニケーション出来る。筆者らは、このような観点に立ち、実世界を直感的に扱う外延の世界（人間による作業の場）と情報隠蔽された情報を扱う内包の世界（機能的な世界でありコンピュータが得意とする作業の場ともいえる）の２つに分けて捉え、外延の世界はさらに静的世界と動的世界の２つに分けて捉える。　そして実世界の事象をモデル化する場合は、静的世界，動的世界，内包の世界の３つの世界の捉え方に沿ってモデル化しそれぞれ「静的モデル」，「動的モデル」，「情報隠蔽モデル」の３つを構成する。

「静的モデル」とは、実世界の仕組みを表すもので、クラスやクラスを組み合わせた複合クラス，さらにはそれらのテンプレートとして表現される。これらは時間的な経過に係わりなく各事象間に存在する関係や仕組みを表す。　そしてこの静的モデルにおけるクラスでは、オブジェクト指向におけるクラスの属性を「属性の振る舞い」と「属性の状態値」に拡張したもとし属性の追加や変更を自動的に行ったりオブジェクトの自己成長（機能アップ）や空間的推移によるオブジェクトの状態変化に対応出来るものとしている。

一方「動的モデル」とは、実現しようとしているモデルの時間的な動きについて示したもので

ある。この動的なモデルにおいては、静的なモデルで定義したクラスをもとにして実際に属性値を与えて出来るインスタンスやインスタンスを組合せたセッション，セッションを組み合わせたシステムなどに、後述の因果関係を定義することにより時間的な動きの制御を行うことが出来る。

また『情報隠蔽モデル』とは、外延の示す性質や内容を表現しているモジュール及びこれらを組み合わせた複合モジュールであり、外延の世界からは情報隠蔽された内包の世界を表すものである。　この中には、各種機能オブジェクト，バッファーオブジェクト，制約オブジェクト，タイマーオブジェクト，静的オブジェクト，動的オブジェクトなどがある。これらの実体はオブジェクトや手続き型プログラムから構成される。　この情報隠蔽モデルは外延の世界からは外延的外辞で必要に応じて参照できる。

筆者らが提案するオブジェクト指向設計法においては、まず外延の世界において外延的外辞を使って静的モデル（クラス）を作り、このクラスからさらにインスタンスやセッションを作ってゆく。そしてインスタンスやセッションの時間的な動作関係に関わる性質をこれとは別に定義する。ここで定義される時間的な動作関係を因果関係と呼ぶ。このように各インスタンスやセッションは、時間的な動作関係を外部で持つことで自分のオブジェクト内に組み込む必要がなくなり部品としての独立性が向上する。

これらの３つのモデルの関係を図１に示す。

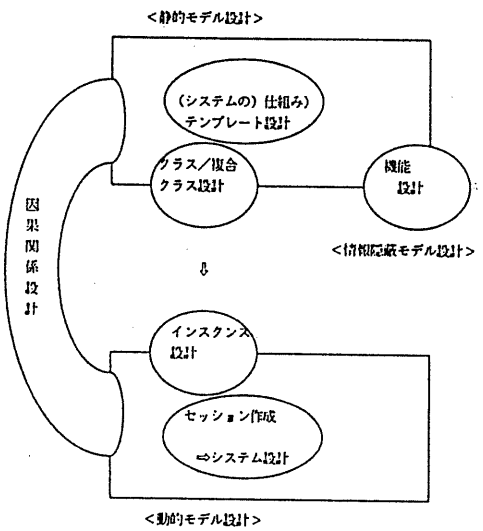


図１．静的モデル，動的モデル，情報隠蔽モデル

３．オブジェクト指向設計法の全体構成

オブジェクト指向設計法を実現させる為の基本機能は『ハイパー処理機能』と『ダイレクトオブジェクト実行機能』である。この関係を図２に示す。

(1)　ハイパー処理機能

ハイパー処理機能によってユーザは自分の要求仕様に最適な部品をネットワーク上に置かれた全ての部品ファイルの中から検索，選択し組み合わせてクラス設計，静的モデル／動的モデルを構築し，さらには動作のシミュレーションまで行える。詳細は〔12〕を参照されたい。

(2)　ダイレクトオブジェクト実行機能

仮動作により目的通り機能することが確認出来たシステムは実際のインストール環境に合わせて高速で動作させるための最適化を行う。

ここでは従来の手続き型のプログラムに準じたＥＸＥモジュールを作成する。

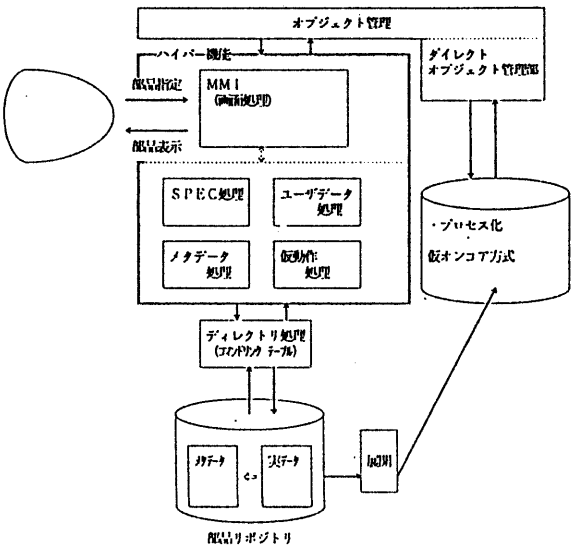


図２　オブジェクト指向設計法の全体構成

4．再利用を容易にする部品化の考え方

(1) 部品の再利用を考える場合、部品を使う側のシステム設計はトップダウンで進められ、一方使われる側の部品を考える場合は、ボトムアップ的に積み上げていくことになる。
部品を使う側と使われる側には基本的にこのようなギャップがある。このギャップを埋める技術を超部品化技術と呼ぶ。
超部品化においては、基本単位である原子部品と，クラスやセッションのような複合部品の２つに分けて考察を進める。
これらの部品に、原則として『独立性』，『直交性』，『正規性』，『抽象性』，『単純性』の５つの性質を持たせることにより、ユーザは要求をそのまま入力するだけで自動的にそのユーザに必要な部品が繋ぎあわされ新しいプログラムを作ってくれる。このような仕掛けを持つ部品を超部品と呼ぶ。　図３に示すように、これらの部品を使って動的世界のインスタンスやセッションを作り、部品間の因果関係をこれとは独立に定義する。
次に因果関係について述べる。

(2) 因果関係とは

構築したシステムが稼働する場合を考える時、インスタンス，セッション，などの動的モデルの動作の間には時間的，空間的，法則的な関係が作用している。　図４において、ａ→ｂ→ｃというセッション（セッションｘとする），ｄ→ｅ→ｆというセッション（セッションｙとする）の２つからなるシステムがある時、『インスタンスｂはインスタンスｅが終了しなければ動作出来ない』という条件がある場合、両者の間にはこのような同期関係を確認する手だてが必要となる。これらのインスタンスやセッションに必要な関係の定義を外部で設定することによりオブジェクトの独立性を向上させることができる。図４の例においてインスタンス（ａ，ｂ，ｃ，ｄ，ｅ，ｆ）は起動時と終了時に因果関係の参照や状態変更のための書き込みを行う。前記のような条件が与えられた時、インスタンスｂは起動前に因果関係を参照しインスタンスｅの終了を確認する。終了していなければインスタンスｂはインスタンスｅが終了するまでｗａｉｔし、終了を確認後動作する。

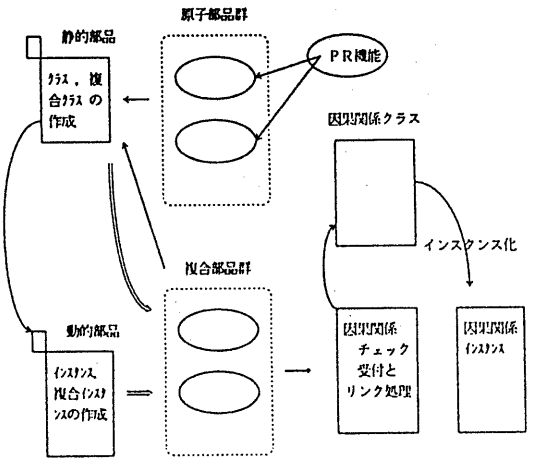


図３　部品作成の工程と因果関係

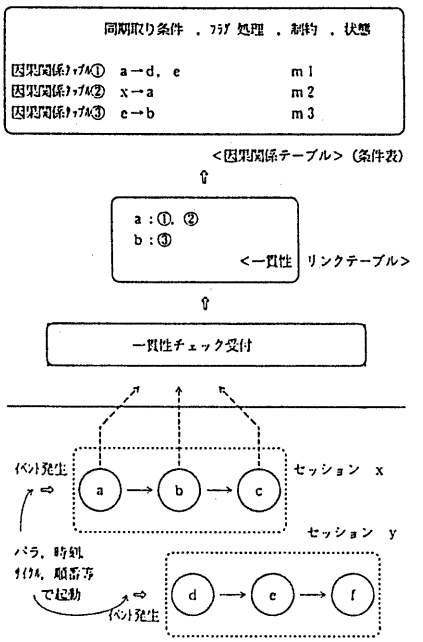


図４．因果関係

アクターオブジェクト指向の大きな特徴の１つはこのような因果関係のトランザクションマネージメントにある。並列に動くセッションに対するこれらの間の時間的な同期をとる制御機能である。

５．因果関係の仕組み

(1)　因果関係を実現させる基本的な仕掛けは、『オブジェクトの同期取り』と『起動判断条件（制約処理）』である。これらは図５に示すスキーマ項目を持っている。

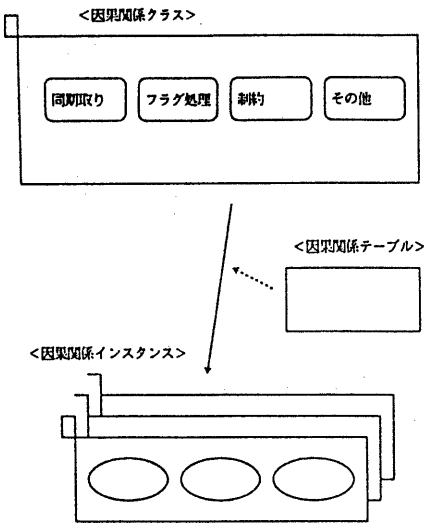


図５．因果関係のスキーマ項目

図において、①同期取りは、どのオブジェクトとどのオブジェクトの状態を対象としているかを示す。

②制約は、同期取りの条件が整ったときに実行されるもので、その時の環境を制約がチェックし制約に設定された条件に従って処理が進められる。

③フラグ処理は同期取り条件に従って排他制御を行うフラグコントロールを司る。この条件が整った時に指定された制約プログラムが起動される。

④その他としては、『従属条件による制御』，『優先順位による制御』，『サイクルの条件がくずれたときの制御』，『ロックした時の制御』などが納められている。基本的には同期取り条件や制約条件をオペレータが外部からセットするときに従属関係や優先順位やサイクル条件をチェックし矛盾が起こらないように設定するのであるが、これらの設定に漏れや矛盾が有った場合の処理の逃げ道としてその他の条件がセットされる。

因果関係をチェックする因果関係インスタンスは、因果関係クラスをもとに因果関係テーブルから必要なタップルを受け取って因果関係のチェック項目毎に発生され、これらは全て並列に動作する。

(2)　因果関係を司るオブジェクト管理機構

因果関係を司っているのが図６に示すオブジェクト管理である。オブジェクト管理部は『シーケンス読込』，『起動処理』，『終了処理』，『因果関係チェック処理』により構成される。

①シーケンス読込によって読み込まれたシーケンスデータに従い起動処理部に指示を出し起動処理部はインスタンスオブジェクトを順次動かす。②動かされたインスタンスオブジェクトは終了した時点で終了処理部に連絡する。　該当オブジェクトが終了したかどうかは終了部で把握出来る。③終了処理部では各オブジェクトの終了毎に因果関係処理を起動させ因果関係チェックを行う。これらの機能を持つオブジェクト管理部はセッション毎に生成される。ということはオブジェクト管理もクラスとして存在しておりセッション毎にオブジェクト管理のインスタンスが生成されるようになっている。

また、セッションを構成しているインスタンスオブジェクトに起動をかけると、これらのインスタンスに対応してオブジェクト管理部ができる。つまりオブジェクト管理部は階層的に生成される。オブジェクト管理部を起動させるものはイニシャルである。この起動条件としては、

サイクル起動，時刻指定起動，やパラ起動がある。

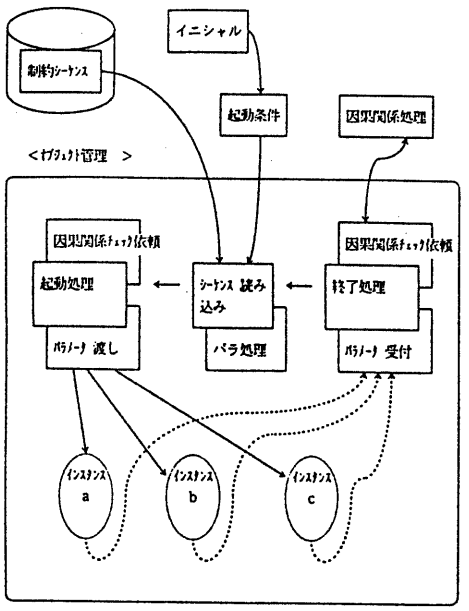


図6．因果関係とオブジェクト管理

(3) 因果関係の動かし方

セッションの中のインスタンスやインスタンスの中のメソッドから因果関係チェックが因果関係チェック部に通知されるとリンククラスに依頼しチェック要求をしてきたオブジェクトが関連している因果関係の状態データ（同期取り，状態図，制約など）を取りに行く。

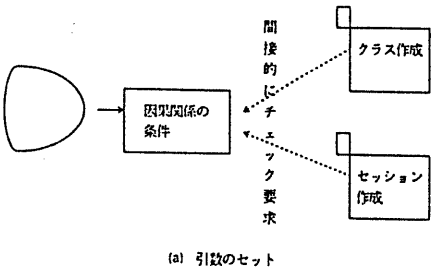
このデータを因果関係クラスに渡すことにより因果関係インスタンスが生成され因果関係の監視体制にはいる。

これらの詳細については、〔13〕を参照願いたい。

(4) 引数処理とそのチェック機構

因果関係クラスを作成する時、関連するクラスの間の引数合わせをいかに簡単に（出来れば自動的に）行うかが一番問題となる。

また因果関係のチェックを依頼するときにおいも因果関係の側で判断するためのデータとして引数を渡してやる必要が発生する場合がある。これらの引数合わせをいかに行うかについて述べる。　図7において、筆者らはこの問題の解決のためオブジェクト間の直接的な引数の受け渡しは止め、クラスを作成する時オペレータが必要な引数を制約として因果関係の条件に組み込むこととした。これにより、オブジェクト指向における独立性と直交性を活かしながら通信の分野で要求されるリアルタイム処理（筆者らは、これをセンサーベースシステムと呼んでいる）をさせることが可能となった。



(a) 引数のセット

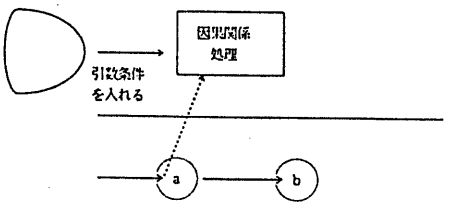


図7．引数渡しのやり方

(5) 引数を使わずデータを渡すための仕掛け

因果関係のチェックを要求するときにセッション側から因果関係オブジェクトに渡すデータは、引数として渡すことを止め、クラスやセッションを作る時にオペレータと会話的に相談しながら必要なデータを共通資源として前もって組み込んでおく方式を取っている。

因果関係の基本はインスタンスやセッションをシーケンスに並べることである。これらがシーケンスに並んでいるかぎり引数の受渡しは不要

であり、データの受渡しの場としては黒板モデルでいう黒板的なものとこれにリンクするためのコマンドリンクの仕組みがあれば良い。
同期がきちんと取られているからオブジェクトは必要な時点で因果関係の状態データを取りににゆける仕掛けとなっている。　　これらを高速に処理するためのディレクトリのコントロール機能の処理フローを図８に示す。
図において、あるプログラムが別のプログラムのデータを取りにゆく時、このデータが一次メモリー上にある時はそのまま一次メモリ上から持ってくる。なければ今までと同じく２次メモリ上から持ってくる。通常因果関係の制約プログラムが動くときは既にこれに関連した他のプログラムが一次メモリ上にあることが多く、ヒット率は高くなる。

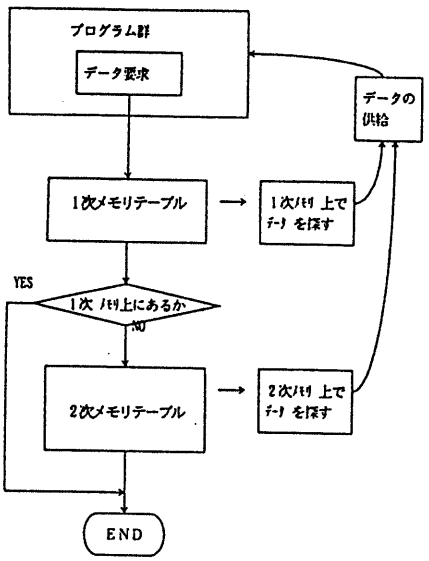


図8．ディレクトリ　コントロールフロー

６．分岐制約

分岐制約には次の２つがある。図９にその概念を示す。１つは、設計時にあらかじめ組み込まれた分岐でありこれを弱性分岐と呼ぶ（図中①）。もう１つは、設計時には組み込まれておらず実行中に突如として強制的に分岐の割り込みが掛かるものである（図中②）。

②の方法を使えばダイナミックに強制分岐を行わせることが出来る。

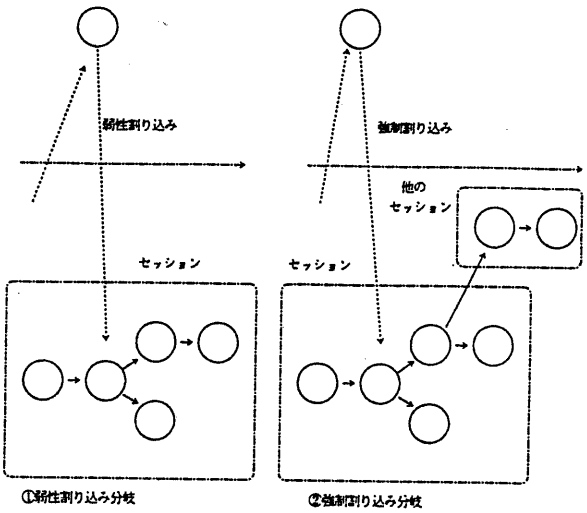


図9．強制分岐と弱性分岐

(1)　正常制約と異常制約

因果関係を使って強制分岐を行わせる場合の概念を図１０に示す。
図１０においてセッションＡのインスタンスαから因果関係チェックが入ってきた場合、因果関係が受け取り、関連する制約プログラムを起動させる。通常の終了通知の場合は、セッションＡのβに起動が掛けられるが、制約の条件にＢへの強制分岐が入っていると正常に終了せず因果関係はセッションＢに起動をかける。
セッションＢが終了した時点でセッションＡのβ（強制分岐の制約がなければ当然この次に起動が掛けられているところ）に起動を掛ける場合が正常制約であり、セッションＡに戻らず、そのまま強制中止してしまうのが異常制約である。強制分岐するかどうかまた強制ストップするかどうかについては、オペレータまたは他のオブジェクトからの指定で制約プログラムにセットできる。

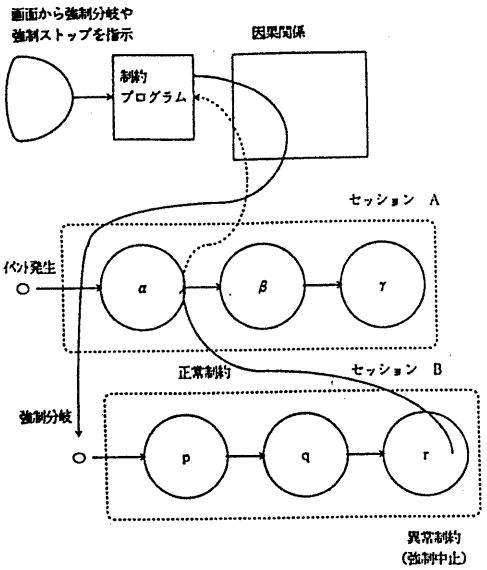


図１０．強制分岐と正常制約／異常制約

(2)　分岐コントロールＪＣＬの自動追加

①　分岐コントロールＪＣＬの作用の概念を、図１１に示す。図において①はシリーズなシーケンスであり、ａ→Next→ｂ→Next→ｃの順に処理される。ｂに強制分岐のコントロールを組み込んだものが②である。②においてｂの処理の終了後、条件によりｃまたはｄに処理が進められる。

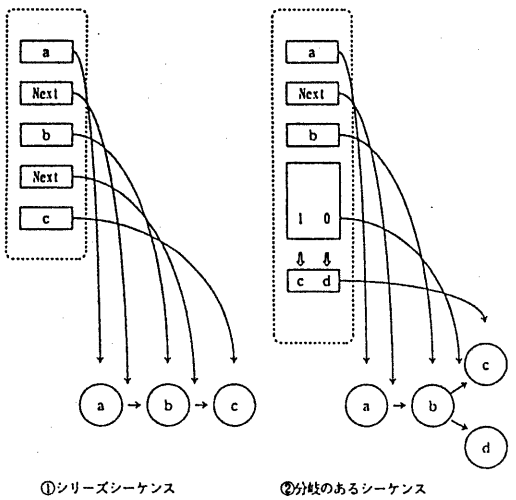


図１１．分岐コントロールＪＣＬの作用

②強制分岐は、因果関係をもとにした分岐コントロールの自動追加により実現することができる。　　図１０を例にした強制分岐の仕組みを図１２に示す。

図においてセッションＡのインスタンスαが終了した時点で因果関係にチェックを依頼にゆきこの時正常なシーケンスであるβに起動をかけるか，セッションＢに起動をかけるかの判断が行われる。そのため①にフローコントロールオブジェクトが自動的に追加されなければならない。次に、セッションＢに起動がかけられセッションＢのなかのインスタンスｒの処理が終了した時点では、制約プログラムＣＰの状態によっては、そのままここで終了する場合とインスタンスαにゆく場合とがある。そのため②にもフローコントロールオブジェクトが自動的に追加されなければならない。

またセッションＡのインスタンスαから見ればダイレクトオブジェクトに展開することを考えた場合、α→βへ進むと思っていたものが、α→ＣＰ→？となる。？はＣＰの状態によってβにゆくかｐにゆくかが決まってくる。このコントロールのために③にコントロールオブジェクトが自動的に追加されなければならない。

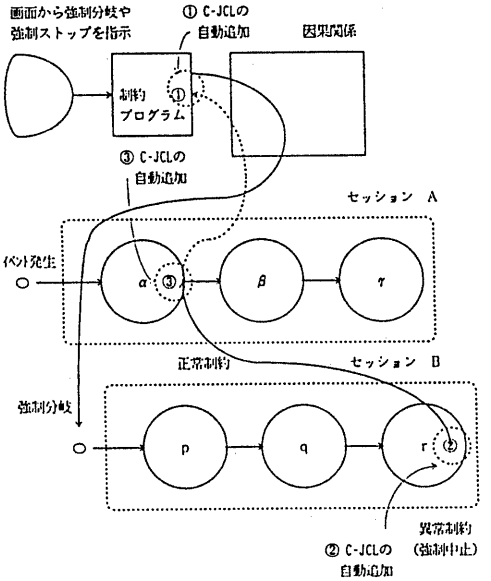


図１２．コントロールオブジェクトの自動追加

7．まとめと今後の課題について

(1) まとめ

本報告においては、従来のオブジェクト指向の考え方を拡張させ、徹底的な人間指向を追求したアクターオブジェクト指向の考え方の概要を述べ、部品の独立性の向上と組合せの柔軟さを狙った因果関係の仕組みや動かし方の概要及び強制分岐について述べた。

筆者らが狙う主たる適用分野であるリアルタイムアプリケーションにおいては、各オブジェクトの同期取りや条件の変更をいかに簡単に行うかが難しい問題となっている。今回、本報告ではこれを実現する仕掛けとして各オブジェクトとは独立な因果関係を外部から設定する方式の提案を行った。

オブジェクトの同期の変更や条件の変更が発生した場合、従来方式では各オブジェクトの修正を必要としていたが、ここに述べた因果関係を導入することにより各オブジェクトには修正を加えることなく因果関係のテーブル変更のみで対応できる仕掛けを得ることが出来た。

(2) 今後の課題

今回紹介した因果関係の引数渡しのやり方は分散黒板モデル的なものと考えている。この考えを更に発展させていくことによりグループウェア，エージェント指向，ロボット指向，バーチュアルリアリティ，電子会議システム，プロセス制御指向などの水平展開を進めてゆく。

なお、今回当該研究の開発の機会を与えて頂いた大槻社長に感謝致します。

参考文献

〔1〕E.Yourdon,and L.Constantine,Structur-ed Design, Yourdon Press, 1979.

〔2〕S.シュレィアー，S.J.メラー著，本位田，山口訳，オブジェクト指向システム分析，啓学出版，1990.

〔3〕P.Coad,and E.Yourdon,Object-Oriented Analysis,Yourdon Press, 1990.

〔4〕J.ランボー他，羽生田監訳，オブジェクト指向方法論ＯＭＴ，トッパン，1992.

〔5〕龍，佐藤，高原「分散処理における新データモデルの提案」 SITA '90

〔6〕龍，佐藤，若林「ネットワークシステム構築から見たオブジェクト指向データベースの提案」情報処理学会ｱﾄﾞﾊﾞﾝｽﾄﾃﾞｰﾀﾍﾞｰｽｼﾝﾎﾟｼﾞｭｰﾑ 1991.7

〔7〕村川，龍「オブジェクトセンサーモデルにおけるメタデータアーキテクチャアの提案」 情報処理学会ｱﾄﾞﾊﾞﾝｽﾄﾃﾞｰﾀﾍﾞｰｽｼﾝﾎﾟｼﾞｭｰﾑ 1991.7

〔8〕龍，若林、村川「オブジェクトの捉え方とオブジェクトセンサーモデル」 情報処理学会ｱﾄﾞﾊﾞﾝｽﾄﾃﾞｰﾀﾍﾞｰｽｼﾝﾎﾟｼﾞｭｰﾑ 1991.7

〔9〕龍，戸島，村川，豊田「自己組織化通信方式の提案」 SITA '92

〔10〕豊田，足立，龍「複合オブジェクトを表すセマンティックID方式」情報処理学会データベース研究会 1992.11

〔11〕市川，龍，戸島 「セマンティックIDによるメッセージパッシング方式の提案」処理学会ｱﾄﾞﾊﾞﾝｽﾄﾃﾞｰﾀﾍﾞｰｽｼﾝﾎﾟｼﾞｭｰﾑ 1992.12

〔12〕村川，龍，戸島，豊田「オブジェクト指向設計を支援するハイパー処理機能の提案」 情報処理シンポジューム 1993.1

〔13〕泉，龍，村川「リアルタイム性を追求したオブジェクト思考設計法」 1993.5.