ソフトウェア工学　46－11
（１９８６．２．６）

Ａｄａによる分散トランザクション

Implementing Distributed Transactions using Ada

藤　田　昭　平

Shohei Fujita

東京工業大学　工学部

Tokyo Institute of Technology

１．まえがき

従来のコンピュータ・ネットワーク用のソフトウエアにおいては，人間の介在が前提になっている場合が多く，下位レベルの通信機能にのみ重点が置かれがちであった([２],[38],[42])。ＣＩＭ (Computer Integrated Manufacturing [32],[26]) 用コンピュータ・システムにおいては、従来のコンピュータ・ネットワークが対象にしていたのとは相異なる新しい問題が生じて来る。
例えば，

- 複数のロボットが協調作業を行う機能
  ---- ロボット群の制御
- ＣＡＤ／ＣＡＭデータベースの有効利用
  ---- 分散データベース管理

等の実現に際しては，従来看過されていた困難な諸問題が解決されなばならない。

１．１　即答性

ＣＩＭ用のコンピュータ・システムは，物理現象のモニターリング及び制御に必要な最大許容時間 (temporal constraint) ― この最大許容時間は物理現象を支配する法則により定められ，既知である場合が多い―に適う性能を有するように設計されねばならない([20]),[24])。

１．２　高信頼性・高稼働率

ＣＩＭ用コンピュータ・システムは，電磁障害等の悪環境の中で作動し続けねばならない。ＣＩＭ用コンピュータ・システムは本来分散システムであり，柔軟な拡張性を有している必要がある。さらに，ＣＩＭにおける事象は自律的かつ同期的である。

ＣＩＭ用コンピュータ・システムは，以上に示した高度な機能を実現しなければならない。リアル・タイムの厳しい要求は，新しいハードウエア技術の出現により解決されるものと思われる。しかし，超高速のＶＬＳＩチップをもってしても解決できない問題は，システム全体の信頼性・稼動率を高めることである。

本研究は，ＣＩＭ用コンピュータ・システムにおける高信頼性・高稼働率を有する分散ソフトウエアの設計法及びその実現法を目標とするものである。ロボット，知能センサ，データベース等を分散オブジェクトとして表現し，各オブジェクトの操作は分散トランザクションを介して行う。

２．分散オペレーティング・システム

共有メモリにより結合されたプロセッサの集合体をサイトと云う。（縮退した場合として，単一プロセッサの場合を含む）。複数のサイトが通信媒体（例えば，ＬＡＮ）により結合されたシステムを分散システムと云う。ただし、サイト間には

メモリ及びクロックの共有はなく，サイト間にはメッセージの交換のみが可能である。

メッセージとは，

- オブジェクトに対する操作の要求，

及び

- オブジェクトに対して履行された操作の結果

を意味する。メッセージの本質的な特徴は，ある動作を実行するのに必要な情報の単位である点にある。各サイトがメッセージの交換を行うことにより“一つの目的”を達成するシステムを協調分散システムと云う。

協調分散システムを特徴づける大きな要因は，

- “一つの目的”を達成するようシステム全体の制御が存在する
- このシステム全体の制御は，集中化された大域情報に基づいてではなく，局所情報にのみ基づいて行われる

ことである。協調分散システム上で実行され得るプログラムを分散プログラムと云う。特に，局所情報に基づいて，システム全体の制御機能を実現したプログラムの集合を分散オペレーティング・システムと云う。分散オペレーティング・システムの重要な役割は，信頼性のある分散オブジェクトの管理である。このためには，トランザクション機能（アトミック・アクション）および（相異なるサイト上にある）オブジェクト間通信機能の実現が必要となる。

分散オペレーティングシステムが具備すべき条件の中でも次の機能：

(i) ネットワーク・トランスペアレンシイ

(ii) 高信頼性・高稼動率

は特に重要である([28], [25])。

ネットワーク・トランスペアレンシイにより，サイト外にあるオブジェクトもサイト内にあるオブジェクトと同様に扱うことが可能になり，分散ソフトウエアの作成が容易になる。また，ソフトウエアの可搬性も高まる。

分散システムの大きな特徴である高信頼性・高稼動率を実現する手段としては，

(ii-a) 重複度 (redundancy)

(ii-b) エラー回復機能 (error recovery)

等がある。

まず，ハードウエアそれ自体の重複度を高め，エラー回復機能を付加する必要がある。しかし，分散オペレーティング・システムなくしては，システム全体としての機能を発揮できない。

2．1　設計思想—基本概念と構造

多数のプロセッサ，記憶媒体，及び通信媒体等の物理素子を用いて，より回復力のあるロバストな分散システムを構築するのが最終目標である。このためには，階層化・モジュール化により安定な分散オブジェクトを構築し，これらのオブジェクトの操作は分散トランザクションを介して行うトランザクション指向モデル([19])を用いる。

記憶媒体とは，システムの稼動中，種々の原因による情報損失の確率は零ではない物理的な記憶素子を示す。システムの稼働中に，プロセッサがその制御情報を失う確率も零ではない。通信媒体においては，電磁波障害によりメッセージの損失が生じたり，データグラム方式に基づいている等の場合には，メッセージの遅延確率が高いことがある。以上のような特性を有する多数の物理素子を用いて構成された大規模・複雑なシステムを制御し，協調動作を行わせるためのＤＯＳは次のような階層構造を有している。

第一段階としては，安定なプロセッサ機能，安定な記憶機能，及び安全な通信機能を実現する仮想素子を構築する。

第二段階では，安定な識別子（ＩＤ）を生成する。このＩＤはシステム全体で一意な名前であり，すべてのオブジェクトにそれぞれ与えられる。

すなわち，オブジェクトに与えられたObject__ＩＤにより，そのオブジェクトを識別することができる。これらのObject__ＩＤによりシステムにおける単一の論理的名前空間（single naming domain)が定義される（[33]),[41]）。

分散オブジェクトに対する操作のアトミック性（及び整合性）を保持するために分散トランザクション機能を実現する。

第三段階では，以上の機能を用いて，より安定かつ安全な分散オブジェクトを構成する。　　さらに，オブジェクトの移動を可能にし，システムの再構成を迅速に行う。

各モジュールの機能を以下に示す。

カーネル：

－－　Ａｄａタスクの効率的な実現，特にマルチ・プロセッサ上のタスキングの実現。

通信・モジュール：

－－　リモート・エントリ呼び出し及びマルチ・キャスト機能の実現。

ネーム・サーバ・モジュール：

－－　論理的名前空間の物理的アドレス空間への写像。

トランザクション・モジュール：

－－　分散トランザクション管理機能の実現。

リカバリ・モジュール：

－－　安定な記憶機能の実現。

２．２　ソフトウエア工学との関係

ソフトウエア工学の“ルーツ”（の一つ）は，オペレーティング・システムの設計・開発にあることは広く認識されている（[１]）。階層化・モジュール化は，その典型的なものである（[27]）。

大規模・複雑なオペレーティング・システムの設計・開発は，ソフトウエア工学の事例研究であった。協調分散システムの構築は、一つの抽象マシンを構築することであり，分散オペレーティング・システムの構築という事例研究を通して，従来の集中システムに重点をおいたソフトウエア工学では看過されていた重要な課題が生じて来る。

２．３　Ａｄａとの関係

従来のソフトウエア工学には，ともすると実体のない“虚業（？）”に陥りがちな傾向が見られた。Ａｄａ誕生のルーツは，埋め込み型コンピュータ・システム用ソフトウエア開発の“工業化”にある（[５]）。Ａｄａとソフトウエア工学の関係，Ａｄａの利点・欠点についてはここではこれ以上論じない。現在最も欠如しているのはＡｄａの使用経験であり，実践を踏まえて，“Ｘ＿Ａｄａ”を開発すべきである。

［注２－１］　ここでは，協調分散システムのソフトウエアに関心があり，通信媒体（のハードウエア）等については論じない。現在最も欠如しているのは，協調分散システム用のソフトウエアである。ソフトウエアの質は，そこで用いられている言語に大きく依存する。

［注２－２］　本稿では，並行プログラムと分散プログラムを明確に区別する。並行プログラムの実行に際してはプロセス間における共有メモリの存在が許されるが，分散プログラムを実行するサイト間には共有メモリは存在しない。ＣＩＭ用コンピュータ・システムに有用なのは分散プログラムである。

［注２－３］　分散オペレーティング・システムに関する研究は現在幾つかのプロジェクトで行われているが（例えば，[22]，[29]，[37]等），各目的およびその手段はそれぞれ相異なっている。これらの比較検討を行うためには，もう少し“時の経過”を待たねばならない。

３．　分散オブジェクト

オブジェクトとは，（データの内部）状態とそれに対する一連の操作から成り立っている。ロボット，知能センサ及びデータベース等を分散オブジェクトとして表現する方法の詳細な記述については紙面の制約によりここでは省略する。オブジェクト指向の利点は

- 現実世界（例えば，ロボット，データベース等）のモデル化に適していること，
- Von Neumann マシン上で効率的に実現しやすいこと，

等にある。

Ａｄａがどの程度オブジェクト指向言語であるかに関しては様々な議論がある（例えば[31]）。パッケージは“第二種”のオブジェクトではあるが，“第一種”のオブジェクトではない（[40]）。１９８０年代後半に予定されているＡｄａの改訂に

際しては，この点が一つの焦点になるであろう。

本稿では，とりあえずＡｄａパッケージによりオブジェクトを表現する。パッケージはライブラリ要素であるが，タスクはライブラリ要素とはなり得ないことに注意。

４．分散トランザクション

“Nothing ever works perfectly 100 percent of the time”([３])。

ＣＩＭ等に要求される大規模なコンピュータ・システムでは，何時，何処で，故障が起きるかを予測するのは非常に困難である。一方，何時，何処で，故障が起きても，たとえシステム全体として機能は少々縮退するにせよ，システムは作動し続けることがリアルタイム・システムでは不可欠である。ＣＩＭ用コンピュータ・システムは、電磁障害等([注4-1])による悪環境の中で作動し続けねばならない。

分散ソフトウエアの設計・作成に際しては、故障によるフェイラを明確に規定することが必要になる。故障により発生するフェイラは　　二種に大別できる。

（Ａ）不作為なフェイラ（Omission Failures）
故障により，アルゴリズムを遂行するのに必要なある動作が行われない。

（Ｂ）越権的なフェイラ（Commission Failures）
故障により，アルゴリズムに不必要なある動作が行われる。

（Ｂ）の場合には誤動作が存在する場合であり，いわゆる“Byzantine Agreement ”の問題になる([注4-2])。
この問題に関しては稿を改めて述べる必要があり，本稿では（Ｂ）の場合は考えない。したがって，あるサイトまたは通信網に故障が発生したときには，例えばメッセージの送受信に必要なある動作が行われなくなるだけであり，犯意的なメッセージを送信するような誤動作は行われない。

４．１　サイト・フェイラ

各サイトにおける回復機能の実現に際しては，フェイラにより情報損失が生ずるか否かが重要な要因となる。

（Ａ－Ｓ－１）　情報損失を生じないフェイラ

入力データ等の異常によりトランザクションの実行に障害が生じた場合

（Ａ－Ｓ－２）　揮発性ストレージの損失を生ずる場合

システム・クラッシュ等により主メモリの情報が失われる場合。ただし，二次メモリ（例えば，ディク）上の情報損失は生じない。

（Ａ－Ｓ－３）　不揮発性ストレージの損失をも生ずる場合。

ヘッド・クラッシュ等により，デイク上の情報損失が起きる場合。

４．２　通信フェイラ

分散ソフトウエアの実現に際してさらに重要なのは通信機能における障害である。通信網を電磁障害等から守る一方法としては，光ケーブルの使用が考えられるが，光ケーブルを含む通信系でもフェイラが発生する確率を零にすることはできない。さらに，移動ロボットを含むＣＩＭにおいては，将来通信は電波により行われるものと思われる。この場合には，通信機能における障害がより顕著になって来る。

（Ａ－Ｃ－１）　メッセージの損失による場合：

通信系の不完全動作によりもとの（あるいはＡＣＫ）メッセージが消失する場合，またはメッセージの大巾な遅延が生ずる場合。

通常は，数回試みてだめなら，通信網又はサイトにフェイラが生じたものとみなす。

（Ａ－Ｃ－２）　通信系の分断による場合：

各サイトが幾つかのグループに分断され，グループ内では通信可能であるが，グループ間の通信が不可能になる場合。ＣＩＭにおいては，種々の機器をゲートウエイを介して接続することが多いが，ゲートウエイ・プロセッサにおけるフェイラは通信系の分断を生じ，重大な障害になる。

4．3　分散トランザクション

- 首尾よく完了した場合には，オブジェクトの状態をその最終状態に遷移させる
- 障害が生じた場合には，オブジェクトの状態をその初期状態に戻す

一連の操作をアトミック動作と云う。トランザクションとは，前述の種々のフェイラ及び並行性の下で，

- アトミック性（Atomicity）

及び

- 整合性（Consistency）

を保持するのに必要な，一体不可分な操作の単位として定義される[18]。（一つの）トランザクションが複数のサイト上で実行されるとき，これを分散トランザクションと云う。

サイト・フェイラは各サイトをマルチ・プロセッサ構成とし，ハード面での耐故障設計を施すことにより，その発生確率を少なくすることが可能と思われ，かつすべきである。

一方，通信系における障害，特に通信系の分断は一度発生すると大混乱を引き起す恐れがあり，非常に重大である。しかし，（Ａ－Ｃ－２）に対する回復機能の実現のためには，そのアルゴリズム，パフォーマンス等困難な問題に直面する。

高信頼性・高稼働率を有する分散ソフトウエアの実現には，分散オブジェクト（アクティビティ）間の通信が特に重要である。すなわち，送信側のアクティビティにとって必要なのは，受信側のアクティビティがそのメッセージを受理したか否か，さらにそのアクティビティによる動作の結果を知ることである。下位レベルの通信系におけるＡＣＫの受理は，送信側のアクティビティにとってはほとんど御利益はない（[21]，[34]）。

分散システムにおけるフェイラに対する回復機能を困難にする大きな要因は，受信側のアクティビティからの応答がない場合，どのようなフェイラが生じたのかを判定するのが困難なことである。送信側のアクティビティが，有限のプロトコルでこれを判定するのは不可能である。

［注４－１］　例えば，電磁波ノイズによる電磁波妨害が大きな問題になる。

［注４－２］　Byzantines Agreementに関する理論面での研究は沢山あるが，そのアルゴリズムを実際に実現するためには，未解決な問題が沢山残されている。

［注４－３］　下位レベルの通信系の役割は，

- 正しい送り先へメッセージを送信すること，
- メッセージの正しい順序を保持すること，
- 重複したメッセージを除去すること，
- 誤ったメッセージを修正すること，

等である。

５．　２－ＰＣのＡｄａによる制御構造

トランザクション・モジュールが提供すべき機能は，上述の種々のフェイラの発生如何にかかわらず，分散トランザクションのアトミック性及び整合性を保持することである。すなわち，（一つの）トランザクションが複数のサイト上で実行されるとき，各サブトランザクションをコミットするかアボートするかに関して，全員一致の決定を行うようにしなければならない。ただし，各サイトの自律性（Autonomy）もできるかぎり保持することが必要である。このことは，分散システムの稼働率（availability）を高めるのに特に重要である。

両者を常に両立させることは至難であるが，いわゆる“２－ＰＣ（two-phase commit）プロトコル”[21]はサイト・フェイラおよびメッセージ損失の発生如何によらず，全員一致の決定を可能にする。

５．１　Ａｄａによる制御構造

２－ＰＣのＡｄａによる記述を図５－１に示す。（一つの）分散トランザクションには大域的な識別子（Trans＿ＩＤ）が付けられており，複数のタスクにより実行される。これらのタスクは各サイトにそれぞれ分散されている。ある（一つの）分散トランザクションに関係しているサイトを参与者（participant）と云う。この中で，ある特定の役割りを果すサイトを指揮者（Coordinator）と呼ぶ。各サイトには，それぞれ識別子（SITE＿ＩＤ）が与えられている。

（サイト＿0）

あるユーザ・プログラム（例えば，ロボット群を制御するプログラム）が存在し，エントリ呼び出しによりある（一つの）分散トランザクションの実行を要求する。このサイトを（便宜上）指揮者とする。

（サイト＿i）（i＝1，…，n）

その他のサイトは参与者となり，指揮者の指揮権にしたがう。ただし，ある時点までは，自律性を発揮することができる。

各サイトでは，分散トランザクションを構成している各サブ・トランザクションが実行される。2－PCの基本的なアイデアは，サイト・フェイラ及びメッセージの損失が生じても，すべてのサイトがそれぞれのサブ・トランザクションをコミットするかアボートするかに関して全員一致の決定ができることである。このプロトコルは二つのフェーズから成っている。

［フェーズ1］

指揮者（図5－1）

6行：ユーザ（例えば，ロボット群の制御プログラム）からの要求を受理する。この要求はエントリ呼び出しより実現される。

7行："prepare" ログ・レコードを安定なストレージに記入する。"prepare" ログ・レコードには，この分散トランザクションを構成しているすべてのサブ・トランザクションの識別子が記録されている。

8行：すべての参加者に，準備ができているか否かを尋ねるためにメッセージを送る。これはリモート・エントリ呼び出しにより実現される。

参加者（図5－2）

6行：指揮者からのリモート・エントリ呼び出しを受理する。

7行：もし準備がすべて出来ているならば，

8行：サブ・トランザクションのログ・レコードを安定なストレージに書き込む。

9行："ready" ログ・レコードを安定なストレージに書き込む。このことは，このサイトにおけるすべての資源がこのサブトランザクションのためにいつでも使用可能なことを保証する。

以上により。このサイトでたとえフェイラが発生しても，このサブ・トランザクションの実行が可能なことが保証されている。

10行：リモート・エントリ呼び出し

SEND＿READY（－－－）

により，このことを指揮者に伝える。

11－14行：もし都合が悪い場合には，"ABORT" レコードを安定なストレージに書き込み，リモート・エントリ呼び出し

SEND＿ABORT（－－－）

により，このことを指揮者に伝える。

［フェーズ2］

指揮者（図5－1）

9行：select文の実行に入り，参加者より応答が返って来るのを待つ。

10行：参加者のすべてが，READYと応答した場合には，この分散トランザクションをコミットする決定を下す。

11行："GLOBAL＿COMMIT" ログ・レコードとして，この決定を安定なストレージに書き込む。

12行：リモート・エントリ呼び出し

SEND＿COMMIT（－－－）

により，すべての参加者にこの決定を知らせる。

15行：ある参加者がABORTと応答した場合には，この分散トランザクションをアボートする決定を下す。

16行："GLOBAL＿ABORT" ログ・レコードとして，この決定を安定なストレージに書き込む。

17行：リモート・エントリ呼び出し

SEND ＿ABORT（－－－）

により，すべての参加者にこの決定を知らせる。

20－22行：ある一定時間が経過しても応答がない場合には，ＡＢＯＲＴと判断し，この決定をすべての参加者に知らせる。

24－27行：すべての参加者よりＡＣＫを受理し，新しいレコード“ＣＯＭＰＬＥＴＥ”レコードを安定なストレージに記入し，この分散トランザクションを終了する。

5－28行：再び新しい分散トランザクションの実行を（無限に）くり返す。

参加者　（図５－２）

16行：select文の実行に入り，各サイトは指揮者からの指令を待っている。

17－18行：ＣＯＭＭＩＴの指令を受理した場合には，“ＣＯＭＭＩＴ”レコードを安定なストレージに書き込む。これにより，たとえサイトにおけるフェイラが発生しても，サブ・トランザクションは実行される。

19行：指揮者へＡＣＫを送る。
たとえ，下位レベルにおける通信系でＡＣＫの送受信が行われていても，陽にＡＣＫを送ることにする。

20行：サブ・トランザクション（例えば，ロボットへの指令）を実行する。

23－26行：ＡＢＯＲＴの指令を受理した場合には，このことを安定なストレージに書き込み，リモート・エントリ呼び出し

ＳＥＮＤ＿ＡＣＫ（－－－）

により，指揮者にＡＣＫを送信する。

5－28行：以上の動作を（無限に）くり返す。

5．2　２－ＰＣの長所と短所

長所

（1）（参加者の）サイト・フェイラに対して，回復力(resiliency)

がある。

（2）　各サイトに自律性（Autonomy）がある。（フェーズ1）。ただし，一度“ＲＥＡＤＹ”と宣誓した後は，自分勝手に変更することは許されない。（フェーズ2）。

（3）　メッセージ交換の数が少なく，比較的単純である。

短所

（1）　指揮者のフェイラ　又は通信系のフェイラにより指揮者と参加者が分断された場合には回復力がない。この場合には，“ＲＥＡＤＹ”状態の参加者はブロッキングされる(16-27行)．このブロッキングのために，システムの稼働率(availability）が減少する。非ブロッキング・２－ＰＣプロトコルについては稿を改めて述べる［17］。

［注５－１］　分散トランザクション管理アルゴリズムの有効性を規定する目安としては，稼働率の外に

- 交換されるメッセージの数
- トランザクション処理における並行度
- 〃　　に要する時間

等が挙げられる。

［注５－２］　２－ＰＣはコミッション・フェイラに対しては回復力はない。例えば，指揮者がある参加者に間違った指令を発した場合には救いようがない。このような事態に対する対応策は，いわゆる“Byzantine Agreement ”の範疇に属する問題であり，ここでは論じない。

6．　システム構成

３台のワークステーション（２台のMicro-Engines 及びＳＵＮ－2/120)からなる非同質なシステム構成になっている。現在，各ワークステーションはＲＳ－２３２－Ｃを介して同軸ケーブルで結合されているため，パケットの送受信はソフトウエアで行っている　［11］。

Micro-Engine上ではＳＴＣ－Ａｄａが，ＳＵＮ－2/120 上ではTeleSoft－Ａｄａが用いられている。両コンパイラ共にＡＣＶＣの検定を通過したものである［35］。

［注６－１］　下位レベルの通信方式の標準化としては，

ＣＳＭＡ／ＣＤ	（ＩＥＥＥ　８０２．３）
ＴＯＫＥＮ　ＢＵＳ	（ＩＥＥＥ　８０２．４）
ＴＯＫＥＮ　ＲＩＮＧ	（ＩＥＥＥ　８０２．５）

等があり，これらの機能はＶＬＳＩチップとして，各メーカに

より提供されつつある。各方式には一長一短があるが，ＣＩＭ用としてはＴＯＫＥＮ　ＢＵＳ方式が有望視されている。ただし，現時点においては，ＴＯＫＥＮ　ＢＵＳはまだ手軽に使える状況ではない。

7．あとがき

ＣＩＭの基盤技術（“infrastructure”）とも云うべき分散オペレーティング・システムのニーズ，特徴，要求される機能を論じ，Ａｄａによる実現法の概略・問題点を示した。すなわち，

（1）　ロボット，知能センサ及びデータベースを分散オブジェクトとして扱い，Ａｄａパッケージによるその表現法。

（2）　リモート・エントリ呼び出し及びマルチ・キャストによるサイト間通信機能とＡｄａによる実現法。

（3）　分散トランザクション機能とＡｄａタスクによるその制御構造

（4）　３台のワークステーションから成る（原始的な）実験システム上での実現。

について述べた。

ＣＩＭ用コンピュータ・システムの死命を制する高信頼性・高稼働率の分散ソフトウエアの実用化を目ざして，今後さらにハード・ソフトの両面に渡る多くの研究課題が解決されねばならない。

謝　辞

本研究の一部は，科研費（５９４６０１２０）による。関係各位に深甚の謝意を表する。

参考文献

[1] Browne, J.C.: "The Interaction of Operating Systems and Software Engineering," **Proc. IEEE,** Vol. 68, No. 9, pp. 1045 - 1049, (1980).

[2] Cypser, R.J.: **Communications Architecture for Distributed Systems.** Addison-Wesley Pub., (1978).

[3] Date, C.J.: **An Introduction to Database Systems.** Addison-Wesley Pub., (1984).

[4] Dertouzos, M.L.: "Control Robotics: The Procedural Control of Physical Processes," **Proc. IFIP 74,** pp. 807 - 813, (1974).

[5] Fisher, D.A. and Williams, J.H. (eds.): Proc. of a DoD Sponsored Workshop - Design and Implementation of Programming Languages, **LNCS** 54, Springer-Verlag, (1977).

[6] Fujita, S.: "On the Observability of Decentralized Dynamic Systems," **Information and Control,** Vol. 26, No. 1, pp. 45 - 60, (1974).

[7] Fujita, S.: "Distributed MIMD Multiprocessor System with MicroAda/SuperMicro(TM) for Asynchronous Concurrent Newton's Algorithms," **Proc. 5th ACM-SIGSMALL Symp.,** pp. 49 - 59, (1982).

[8] Fujita, S.: "Asynchronous Algorithm and Programming for Decentralized Computing Systems: A Pragmatic Example," **Proc. 6th Conf. Soft. Eng.(PS),** pp. 103 - 104, (1982).

[9] Fujita, S.: "Software Components for Real-Time Advanced Robot Control Systems - A Case Study with Ada Workstation," **Proc. Int. Conf. Advanced Robotics,** pp. 409 - 416, (1983).

[10] 藤田： Ａｄａ並行処理機能の実証的評価，情報処理学会・ソフトウエア工学研究会資料，28－21，pp. 121-126，(1983).

[11] 藤田他： 分散オペレーティング・システム―Ａｄａワークステーションを用いた事例研究，電子通信学会・技術研究報告，Vol. 84, No. 174, pp. 39-50, (1984).

[12] 藤田： ＡｄａによるＣＩＭＳ用分散オペレーティング・システム，日本ロボット学会・学術講演予稿集，pp. 293-294, (1984).

[13] 藤田： 分散ワークステーション上のＡｄａ―事例研究，電子通信学会・技術研究報告，Vol. 84, No. 295, pp. 13-24, (1985).

[14] 藤田： 分散オペレーティング・システムにおける例外処理，電子通信学会・技術研究報告，Vol. 85, No. 43, pp. 9-16, (1985).

[15] 藤田： 分散オペレーティング・システムにおける異常検出，電子通信学会・技術研究報告，Vol. 85, No. 184, pp. 61-72, (1985).

[16] 藤田： ＣＩＭにおける分散オペレーティング・システム（II），日本ロボット学会・学術講演予稿集，pp. 43-44, (1985).

[17] 藤田： トランザクション指向分散オペレーティング・システム，電子通信学会・フォールトトレラント・システム研究会，(1986).

[18] Gray, J.N.: "Notes on Data Base Operating Systems," **LNCS** 60, pp. 393 - 481, (1978).

[19] Lampson, B.W. et al.: **Distributed Systems - Architecture and Implementation: An Advanced Course.** Springer-Verlag, (1981)

[20] Le Lann, G.: "On Real-Time Distributed Computing," **Proc. IFIP 83,** pp. 741 - 753, (1983).

[21] Lindsay, B.G.: "Single and multi-site recovery facilities," **Distributed Data Bases** (J.W. Draffan and F. Poole eds.), pp. 247 - 284, (1980).

[22] Liskov, B.: "Primitives for Distributed Computing," **Proc. 7th ACM-SIGOPS Symp.,** pp. 33 - 42, (1979).

[23] Liskov, B. & M. Herlihy: "Issues in Process and Communication Strucutre for Distributed Programs," **MIT Laboratory for Computer Science,** PMG-MEMO 38, (1983).

[24] Mok, A. K_L: "Fundamental Design Problems of Distributed Systems for the Hard Real-Time Environment," **MIT/LCS/TR-297,** (1983).

[25] Neumann, P.G. & L. Lamport: "Highly Dependable Distributed Systems," **SRI Int.** Project 4180, (1983).

[26] Nevins, J.L., D.E. Whitney, and Simunovic: "Report on Advanced Automation," No. R - 764, **The Charles Stark Draper Lab.,** (1973).

[27] Parnas, D.: "On a "Buzzword": Hierarchical Structure," **Proc. IFIP 74,** pp. 336 - 339, (1974).

[28] Popek, G.J.: "Notes on Distributed Systems of Microprocessors," **LNCS** 126, pp. 303 - 348, (1982).

[29] Popek, G. et. al.,: "LOCUS: A Network Transparent, High Reliability Distributed System," **Proc. 8th ACM-SIGOPS Symp.,** pp. 169 - 177, (1981).

[30] Rosenblum, D.S.: "Correct and Efficient Runtime Task Communication in Distributed Implementations of Ada," **Program Analysis and Verification Group, Computer Systems Lab., Stanford University,** private communication, (1985).

[31] Rentsch, T.: "Object Oriented Programming," **SIGPLAN NOTICES,** Vol. 17, No. 9, pp. 51 - 57, (1982).

[32] Rosen, C.A.: "Robots, Productivity, and Quality," **Proc. ACM Natl. Conf.,** (1972).

[33] Salter, J.H.: "Naming and Binding of Objects," **LNCS** 60, pp. 99 - 208, (1978).

[34] Saltzer, J.H., D.P. Reed, and D.D. Clark: "End-to-End Arguments in Systems Design," **Proc. 2nd Int. Conf. on Distributed Computing Systems,** pp. 512 - 519, (1981).

[35] Softech. Inc.: **Ada Compiler Validation Summary Report.** (1983).

[36] Spector, A.Z.: "Multiprocessing Architectures for Local Computer Networks," **Stanford University,** STAN-CS-81-874, (1981).

[37] Spector, A.Z. et al.: "Support for Distributed Transactions in the TABS Prototype," **Proc. 4th Symp. Reliability in Distributed Software and Database Systems,** pp. 186 - 206, (1984).

[38] Tanenbaum, A.S.: **Computer Networks.** Prentice-Hall, Inc., (1981).

[39] U.S. Department of Defense: **Ada Programming Language,** ANSI/MIL-STD-1815A, (1983).

[40] Wegner, P.: "On the Unificattion of Data and Program Abstraction in Ada," **Proc. 10th POPL Conf.,** pp. 256 - 264, (1983).

[41] Wupit, A.: "Comparison of UNIX Network Systems," **ACM Conf. Personal and Small Computers,** pp. 99 - 108, (1983).

[42] Zimmerman, H.: "OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection," **IEEE Trans. Communications,** Vol. COM-28, No. 4, pp. 425 - 432, (1980).

```
<      1 >   with RECOVERY; use RECOVERY;
<      2 >   with COMM; use COMM;

<      3 >   task body COORDINATOR is


<      4 >   begin
<      5 >       loop
<      6 >           accept REQUEST ( ... ) do
<      7 >               -- write "prepare" record in the log
<      8 >               SEND_PREPARE ( ... ); -- to all participants
<      9 >               select
<     10 >                       :
                                  :
```