ソフトウェア工学 73−15
（１９９０．７．２０）

リアルタイムシステム分散並行開発環境：ＩＣＡＲＯＳ

川尻 信哉* 中村 正実* 清兼 幸雄** 青山 幹雄**
*富士通北海道通信システム㈱ **富士通㈱

通信サービスが多様に、かつ高度になるに伴い、通信ソフトウェアの分散並行開発が行われている。このような開発パラダイムを支援するリアルタイムシステム分散並行開発環境ＩＣＡＲＯＳ（Integrated Computer-Aided enviRonment for cOoperative Software development）を開発している。ＩＣＡＲＯＳは、通信ネットワークに結合された一人一台ワークステーションにより、図形設計情報の作成、変更、共有を支援し、設計者の能力を増幅する。本稿では、ＩＣＡＲＯＳのコンセプト、アーキテクチャ、ならびに状態図エディタについて述べる。

A DISTRIBUTED CONCURRENT DEVELOPMENT ENVIRONMENT FOR REAL-TIME SYSTEMS: ICAROS

Shinya KAWAJIRI[*], *Masami NAKAMURA*[*], *Yukio KIYOKANE*[**], *and Mikio AOYAMA*[**]

[*]FUJITSU HOKKAIDO COMMUNICATION SYSTEMS LIMITED [**]FUJITSU LIMITED

A distributed concurrent development has been practiced due to the advancement and diversification of communications services. To support such development paradigm, we have developed *ICAROS (Integrated Computer-Aided enviRonment for cOoperative Software development), a distributed concurrent development environment for real-time systems.* ICAROS aims at to augment human ability by supporting the design, change, and exchange of graphical design documents through a workstation connected to networks. This article reports the concept and architecture of ICAROS, and the graphical structured editor which is at the heart of ICAROS.

1．はじめに

近年、通信サービスが多様かつ高度になるに伴い、開発される通信ソフトウェアは、急速に大規模かつ複雑になっている。また開発拠点が分散する傾向にあり、このような変化を考慮した開発効率と品質の向上が重要な課題である。我々はこの問題に対処するため、リアルタイムシステムの分散並行開発を支援するＣＡＳＥ環境、ＩＣＡＲＯＳ（イカロス：Integrated Computer-Aided enviRonment for cOoperative Software development）を開発している。ＩＣＡＲＯＳは普及しつつあるネットワーク技術と、高度な処理能力を持つワークステーション（ＷＳ）をベースに、状態遷移図に基づいて、リアルタイムシステムの分散並行開発を支援する。

本論文では、まず２章で現状におけるリアルタイムシステム設計の問題点を挙げ、３章で問題に対する解決アプローチを述べる。４章ではＩＣＡＲＯＳの基本コンセプトであるアンブ概念について述べる。５章、６章ではＩＣＡＲＯＳシステムアーキテクチャと、そのうち、特に状態遷移図の作成、変更を行なう状態図エディタを紹介する。最後に、今後の課題について述べる。

2．開発の背景

ＩＣＡＲＯＳが狙いとするのは、通信ソフトウェア開発の抱える課題のうち、生産性向上の観点から特に重要だと考えられる以下の３点である。

2.1 分散並行開発

顧客の通信ソフトウェアへの要求の多様化に伴い、地域性を考慮した開発や保守が必要であり、このため開発拠点が各地域へ分散しつつある[1][2]。また、顧客要求に迅速に対応するため、複数の機能を並行して開発する必要がある。しかし、従来の集中逐次型の開発環境では、このような開発プロセスをうまく支援できない。例えば、分散開発拠点間の設計情報の共有（作成や変更、交換、管理）が生産性と品質向上のために必須であるが、従来の紙などの情報形態では、情報の表現力や操作性、再利用性などに制約があり、上記の要件に対応できない。通信ネットワークを介して、各分散開発拠点が共有できるような設計情報の形態と、その管理利用環境が必要である。

2.2 リアルタイムシステム

通信システムのようなリアルタイムシステムは、多くの並行動作するプロセスで実現されており、このようなシステムの挙動を理解するには高度なスキルが必要である。また、プロセスの動的挙動の再現が困難なため、試験やデバッグが難しい。そのため、ＳＤＬ（Specification and Description Language）、Estelleといった状態遷移モデルに基づいた設計検証技法が使われている[3]。生産性向上のために、このような手法を十分支援する開発環境が必要である。例えば、状態遷移図に基づく設計の支援や、試験効率の向上のためにシステムの動的挙動を視覚的に表すなどの、リアルタイムシステム開発支援環境が望まれる。

2.3 機能追加主体の開発

通信ソフトウェアは一般に大規模で、長期間に渡って機能追加が繰り返される。また、リアルタイムシステムの修正や変更に伴う影響範囲は、論理構造のみならず、タイミング条件などにも及ぶので、設計者に高度な知識と経験が要求される。このため、長期に渡って加えられた変更の内容、理由、および影響範囲の把握を可能とする設計情報の管理が求められる。

3．問題解決へのアプローチ

上で挙げた問題を解決するための、アプローチの方法と、現在適用可能な技術を以下に述べる。

3.1 アプローチ

3.1.1 分散並行開発の支援

分散した開発拠点間で並行して開発を行うには、最新の設計情報を相互に共有することが必須である。分散した各開発拠点から同一情報にアクセスでき、また、並行して開発を進めるプロジェクトチーム間での設計情報の交換が可能でなければならない。

分散開発においては、設計情報を一元管理するデータベース（ＤＢ）に、ＬＡＮ／ＷＡＮを介してアクセスすることで、設計情報の共有を実現できる。このとき設計情報が必ずしも物理的に集中している必要はない。論理的に一元管理されていて、ネットワーク上の

ロケーションを意識することなくアクセスできることが肝要である。

並行開発においては、プロジェクトチーム間の設計情報の交換を支援する。一般に各プロジェクトチームは同じ開発工程にあるとは限らないため、上流工程から下流（試験）工程までを一貫して支援する環境と、各工程の生産物を関連付けて管理するＤＢを提供し、プロセスを流れる設計情報の連続性と再利用性を保証する。そして各プロジェクトチームに対して必要な設計情報だけが見えるようなプロジェクトビューを提供する。

コミュニケーションは、分散開発と並行開発の両方に共通する問題である。例えば、設計情報が知らないうちに変更されていたということがあってはならない。そのため、通信ネットワークを介して変更情報を交換し、確実な情報の伝達を行う。また、設計情報の中にレビューを可能にするメモスペースを導入することで、拠点間にまたがった協調開発作業を支援する。

3.1.2 リアルタイムシステムの設計

リアルタイムシステムの設計には、対象システムのアーキテクチャを表現できる設計情報が必要である。すなわち、システム全体のモジュール構成を概観できるレベルから、モジュール内の詳細レベルまでを階層的に表現する。また、設計に用いる状態遷移図の図形設計情報を、図形のままその意味を考慮して取り扱う仕組みを提供する。これにより情報操作における論理的矛盾の混入を減らし、設計段階での品質を向上する。さらに、作成した図形設計情報の、構文、意味、タイミングや並行動作性に関して、それぞれ正当性を検証する機能を提供する。

また、従来のソースコードレベルの試験に加えて、システムの動的挙動を設計書レベルで視覚的に表現することで、試験とデバッグの効率と品質が向上する。

3.1.3 機能追加主体の開発

機能追加が繰り返されるシステムを保守するためには、最新の設計情報と、過去の機能追加の履歴を把握することが重要である。

前者に対しては、ソースコードから最新の設計情報を生成するなどの、リバースエンジニアリング機能を提供する[4]。後者については、設計情報の版数管理を行い、問題管理ＤＢと連携して問題処置や機能追加毎の修正履歴を管理する[5]。

3.2 適用技術

3.2.1 ワークステーション（ＷＳ）

近年、ＷＳはハードウェア性能が飛躍的に向上してきた。また、相対的な価格低下により、一人が一台のＷＳを占有して、そのコンピューティングパワーをソフトウェア開発に活用できる環境が普及しつつある[6]。

ソフトウェア面では各種マルチウインドウシステムが提供され、ヒューマン・マシン・インタフェースが飛躍的に向上している。例えば高度なグラフィカル・ユーザ・インタフェース（ＧＵＩ）を利用して、複雑な構造を持った情報を記述し、操作できるようになっている。また、PostScriptやLaTeXなどのページ記述言語を利用して、プリンタなどのハードウェアに依存せず、複雑な図形印刷を行える。

3.2.2 ネットワーク技術

通信ネットワーク技術が発達してＬＡＮ／ＷＡＮの構築が容易になった。また、標準プロトコルが普及し高速のデータ通信が可能になるなど、分散開発環境の基盤が整いつつある。これらの技術を活用し、ネットワークを介して種々のホスト、サーバ、ＷＳを接続した分散開発環境を実現する。

3.2.3 ＣＡＳＥ技術

中流工程を支援するＣＡＳＥが普及しているが、近年はさらに、上流工程支援の重要性が認識されている。最近のＣＡＳＥの発展に伴い、上流工程支援環境の基盤が整いつつあり、図形を用いた設計手法を支援する環境、あるいは設計情報間の整合性を検証するツールなどが実現できるようになってきている。

このような基盤技術に基づき、リアルタイムシステム固有の設計検証手法を支援する環境を構築できるようになった。

3.2.4 オブジェクト指向

図形設計情報の要素を、オブジェクトと考えることで、現実のシステムをより的確に表すことが可能になる。さらに、図形設計情報を単なるイメージとしてで

はなく、システムに対応したオブジェクトとして操作することができる。

また、オブジェクト指向に基づいたウインドウツールキットを利用することにより、高度なＧＵＩを容易に構築できる。

4．ＩＣＡＲＯＳのコンセプト

4.1 アンプ概念

開発支援の方法は、機械による代行（自動化）と道具提供（開発者の能力拡大）の２つの形態がある。ＩＣＡＲＯＳでは、一人一台のＷＳ環境を生かし、設計者の思考活動を強化する道具提供型のアプローチを採る。人間の頭や手足の延長となり、作業能力を増幅（アンプ）する。その主な対象として、以下の３点を考えている。

4.1.1 設計情報のアクセス

設計者と設計情報の間の距離を短くしアクセスを容易にすることが、作業能力を増幅する第一歩である。これには、物理的な距離と論理的な距離とがある。

物理的な距離とは、地理的な隔たりや時差のために設計情報へのアクセスが困難なことである。このため、分散した各開発拠点の一人一台ＷＳから、最新情報にアクセスできるようにする。どの開発拠点からでも同様に情報にアクセスできることで、設計者の能力アップが開発拠点の分散化に依存せずに可能になる。

論理的な距離とは、膨大な設計情報の中で必要な情報を見つけるための検索作業量のことである。このため、対象システムを概要レベルから詳細レベルまで階層的に表現した設計情報ＤＢを構築し、レベルを変えて検索することで、その作業量を軽減する。また、人間の行う自然な情報検索（連想）を可能にすることで、関係のある情報の検索も容易にする。主要な情報に加えて、関連情報へのアクセスを支援することにより、設計者の対象情報への理解を深め、設計品質を向上させる。

4.1.2 設計情報の理解と操作

設計情報に容易にアクセス出来るだけでなく、その情報が、設計者にとって直観的に理解し易く、操作し易いことが大切である。

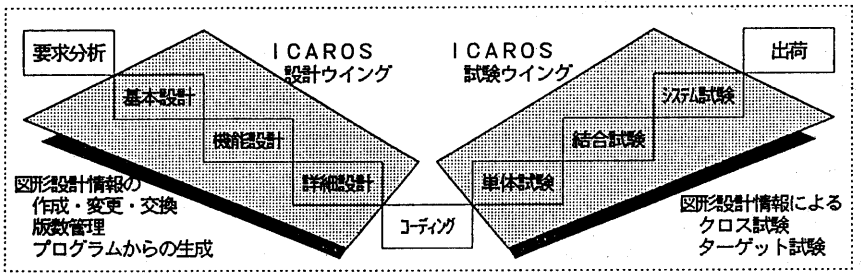
設計者は、設計対象システムについてあるイメージ（概念モデル）を持っている。そして、情報の理解を概念モデルと設計情報ＤＢとを対応付けながら行い、設計や検証を進める。そのため、概念モデルに合った情報表現を工夫することが、理解力の増幅にとって重要である。

一方、操作性を向上させるには、ユーザインタフェースが鍵である。情報に直接働きかけられ、その結果を設計者にすぐにフィードバックできること、および、設計者にとって意味のある単位や表現で情報を操作できることが重要である。これらにより、編集や解析などの情報操作性を増幅させることができる。

4.1.3 設計情報の共有と交換

設計者の能力は、他の設計者との情報の共有や交換、そして、グループによる協調作業とそこで新しく生成される情報によって増幅する。

例えば、レビューでは、複数人が設計内容に対して意見の交換を行うことで、よいアイディアが生まれ、設計者個人では気がつきにくい誤りを発見できる。システム規模の増大や開発組織の分散によって、個人だけで機能追加や変更に伴う影響や設計内容の検証を行



図－１　ＩＣＡＲＯＳの開発プロセスにおける位置づけ

うことが困難となっている。このため、情報の共有と交換を支援することが重要である。

このような情報の共有と交換は、設計情報ＤＢとそれと連携したコミュニケーション基盤の確立によって可能になる。

4.2 開発プロセスにおける位置づけ

ＩＣＡＲＯＳの開発プロセスにおける位置づけを図－１に示す。基本設計から詳細設計までの上流工程においては、工程間での設計情報の整合性を維持しながら詳細な内容を付加していく。例えば、紙の設計書のように、開発のプロセス毎に作り直す必要はない。単体試験からシステム試験までの下流工程では、設計段階で作成された設計情報に基づき、設計書（状態図）レベルでの試験を行う。このようにして、上流工程から下流工程まで設計情報の連続性を保ち、開発プロセスを一貫支援する。

5．システムアーキテクチャ

本章では、ＩＣＡＲＯＳのシステムアーキテクチャと提供機能を紹介する。

5.1 オペレーション
アーキテクチャ

ＩＣＡＲＯＳの運用環境を図－２に示す。設計情報リポジトリとなるホスト、分散開発拠点毎の拠点内ファイルサーバ、一人一台のＷＳを、高速ＬＡＮ／ＷＡＮなどのネットワークで連携した分散開発環境である。UNIX, MS-DOS, X-Windowなどオープンアーキテクチャに基づく要素を用いて実現している。

5.2 ソフトウェア
アーキテクチャ

ＩＣＡＲＯＳのソフトウェアアーキテクチャを図－３に示す。設計情報を中心に各種ツールを統合した。開発から保守までを一貫支援すると共に、既存資産の移行を円滑にするため、リバースエンジニアリングを支援している。

5.3 提供機能

ＩＣＡＲＯＳを構成する各種ツールの狙いと機能を以下に述べる。

5.3.1 状態図エディタ

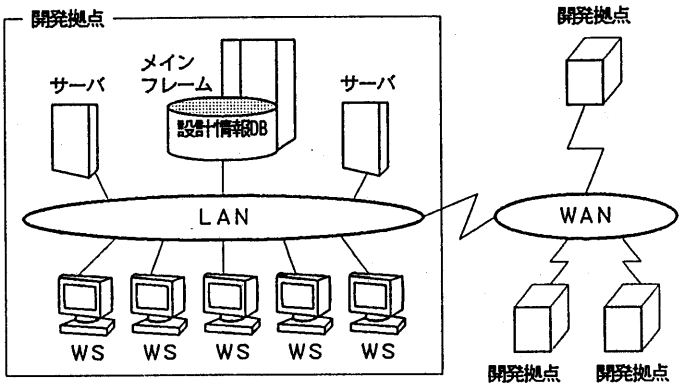
状態図の作成や変更、版数管理、ホスト上の設計情報ＤＢへのアクセスを行う。詳細は６章で述べる。

5.3.2 状態図ジェネレータ

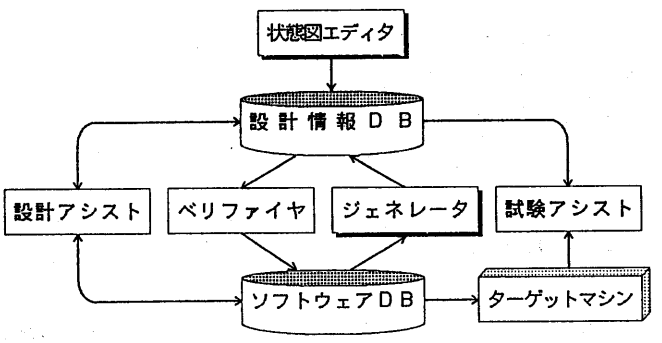
保守工程の支援と既存開発資産の再利用、および新しい環境へ円滑に移行するため、既存のソースプログラムからＩＣＡＲＯＳが支援する状態図への変換を行う。さらに設計書（状態図）レベルでの試験を行うための試験情報を生成する。

5.3.3 状態図ベリファイヤ

設計段階における品質の向上が、開発効率向上の要件となる。設計段階でのシステム仕様検証や性能評価



図－２　ＩＣＡＲＯＳ　オペレーションアーキテクチャ



図－３　ＩＣＡＲＯＳ　ソフトウェアアーキテクチャ

を支援し、問題の早期検出を行う。

5.3.4 設計アシスト

大規模システムの設計には、膨大な設計情報への迅速なアクセスが必要である。ホスト上の設計情報ＤＢ等の設計情報管理機能と連携し、設計情報の検索、および既存資産の再利用を支援する。

5.3.5 試験アシスト

通信ソフトウェアの品質向上の鍵として、従来のソースプログラムレベルでの試験に代わり、設計書（状態図）レベルで試験を行うことが出来れば、システムの挙動が分かりやすくなり試験効率は向上する。ターゲットマシンや試験装置と連携し、状態図エディタあるいは状態図ジェネレータの生成した設計情報に基づき、状態図レベルでの試験実行や、結果の収集・分析を支援する。

6．状態図エディタ

6.1 狙い

状態図エディタの狙いを、以下の４つの観点から述べる。

6.1.1 情報の表現力

概要から詳細に渡るさまざまなレベルの設計情報を紙で表現するには、表現力に制約がある。

状態図エディタではＧＵＩを利用し、設計情報を概要レベルから詳細レベルまで階層的に表現する。さらに、状態図上の各図形要素の重ね合わせや、図形要素の属性などの詳細な内容を埋め込むことができる。

6.1.2 情報の操作性

ワードプロセッサなどを使って設計書を編集する場合、関連性のある設計情報の把握は作業者に任されている。従って、移動や削除などの編集作業において、図形間の意味的な関係は保証されていない。

状態図エディタでは、図形の編集操作を意味的単位で行うために、関連する図形をひとまとまりのオブジェクトとして取り扱う。これにより、例えば、特定の図形を移動や削除した場合、関連する図形群も同時に移動あるいは削除される。さらにエディタ操作に対し、シンタックスに従った制約を加えることにより、編集作業での誤りを軽減し、設計段階での品質向上を図ることができる[7]。

6.1.3 版数管理

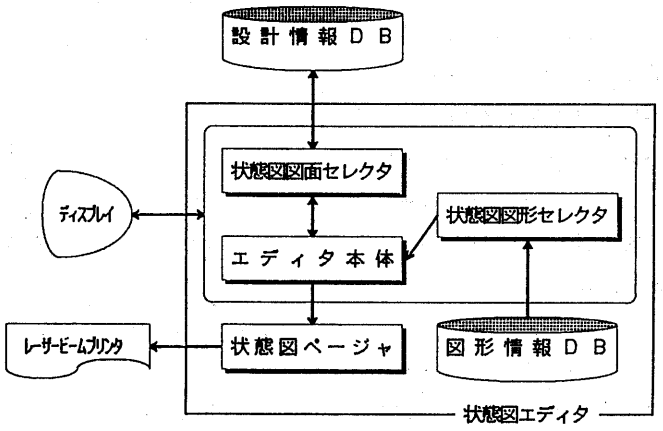
通信ソフトウェアでは機能追加が頻繁に行われるため、設計書の版数管理が重要となる。しかし、空間的な広がりを持つ図形設計情報は、テキストとは異なり版数管理が難しい。

状態図エディタでは、図形設計情報を、状態図記述言語で記述したテキスト形式で扱うことにより、図形設計情報の版数管理を行う。

6.1.4 分散開発支援

開発拠点が分散している場合、紙の設計情報を大量に、頻繁に交換することは困難が伴う。そのため常に最新の設計情報を相互に共有することが難しい。

状態図エディタでは、分散開発拠点からＬＡＮ／ＷＡＮを介して最新の設計情報にアクセスできる。例えば、異なる拠点の設計者が同時に同一図面を参照し、リアルタイムにレビューを行うことが可能である。また、設計内容に対するコメントを蓄積交換することで、時差のある海外拠点間との共同開発を効率よく行える。



図－４　状態図エディタ　システム構成

6.2 アーキテクチャ

6.2.1 システム構成

状態図エディタのシステム構成を図－4に示す。状態図エディタは、状態図の参照や編集を行うエディタ本体のほかに、状態図図面セレクタ、状態図図形セレクタ、状態図ページャといったサブシステムからなる。

状態図エディタと他の機能間のインタフェースは、独自に開発した状態図記述言語によるテキスト形式で行う。従って、状態図ジェネレータをはじめとする他のＩＣＡＲＯＳサブシステムや、版数管理システムなど、既存アプリケーションとの連携が容易である。

6.2.2 エディタ本体

状態図の作成や編集を行うエディタ本体である。状態図エディタによる編集画面を図－5に示す。

図形要素間の意味的関連を内部的に保持しており、状態図として意味のあるまとまりで、複写、移動、削除などが可能である。これにより論理的に矛盾した結果をもたらす編集操作を未然に防ぐことができる。また、対象とする図形要素を選択することで、その属性（端子記号の端子番号や、遷移線の起動要因名など）を示すサブウインドウが開き、属性を記述できる。

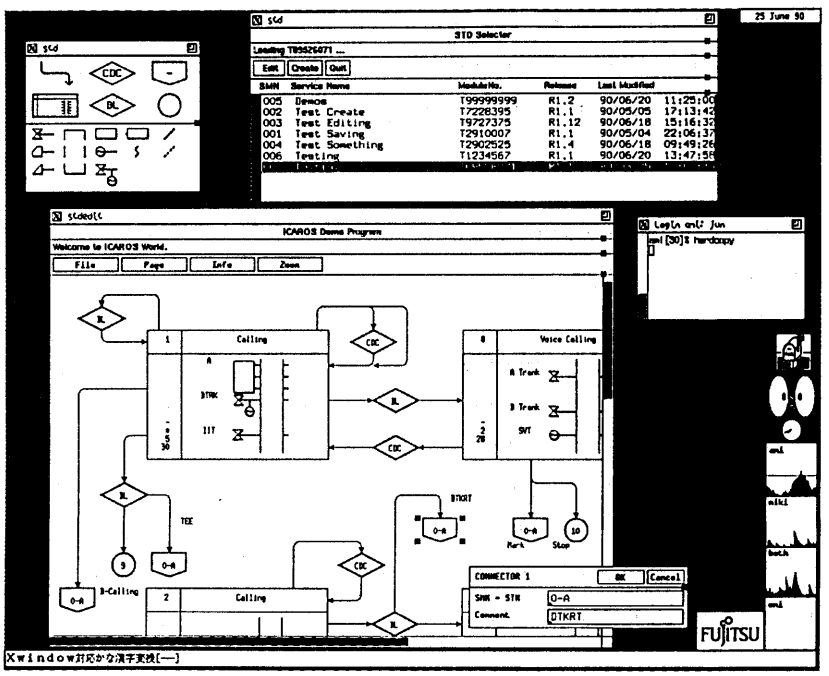
本エディタは、独自の図形クラスライブラリをオブジェクト指向に基づきインプリメントしている。クラスライブラリの継承機能を利用すると、仕様追加や図形定義の変更にも容易に対応できる。

6.2.3 状態図図面セレクタ

状態図の版数管理を行う部分である。設計情報ＤＢに登録されている状態図の一覧表示や検索、編集する図面およびその版数を選択する。選択した図面に対し状態図エディタ本体と状態図図形セレクタの起動を行う。複数の図面を同時に選択し、同一ディスプレイ上で比較しながら編集を進めることができる。

6.2.4 状態図図形セレクタ

電話端末記号や分岐記号といった図形要素に関する情報（画面に表示する形状や属性項目）は、図形情報ＤＢで一元管理する。図形要素のセットを組み替えることにより、異なる記号にも対応する。状態図図形セレクタは登録されている図形要素の一覧を表示する。設計者は、この中から図形要素をマウスで選択し、状態図エディタの編集画面で位置を指定することで、図



図－5　状態図エディタ　表示画面例

形入力操作を行う。

6.2.5 状態図ページャ

状態図ページャは、状態図エディタあるいは状態図ジェネレータで生成された状態図記述言語形式の図面を、レーザービームプリンタの出力形式に変換する。

7．今後の課題

状態図エディタと状態図ジェネレータのプロトタイプを開発し、実プロジェクトへの適用を評価している。今後、プロトタイプを発展させ、大規模交換ソフトウェアの開発保守へ適用する。さらに、関連設計情報を含めた設計情報管理機能を提供する。

これらの機能に基づき、設計アシストと試験アシストの開発と共に、分散開発管理環境との統合、あるいはグループウェアの支援[8]などを進めていく。

8．おわりに

本稿では通信ソフトウェアの分散並行開発を行う上での問題点を挙げ、ＩＣＡＲＯＳシステムが採った解決アプローチと、システムの概要を述べた。今後ますます重要になるであろう分散並行開発を一貫して支援する、このような環境の開発が必要である。

謝辞：ＩＣＡＲＯＳの開発、適用にご支援頂いた当社通信ソフトウェア部門の方々に感謝いたします。

参考文献

[1] 青山ほか "交換ｿﾌﾄｳｪｱの分散並行開発支援環境", 電子情報通信学会・交換ｼｽﾃﾑ研究会, SSE90-25, 1990年6月, pp. 7-12.

[2] 清兼ほか "電子交換システムにおける広域分散開発支援環境について", 情報処理学会第38回全国大会, No. 6L-6, 1989年3月, pp. 1191-1192.

[3] 水野 "標準仕様記述言語の概観", 情報処理, Vol. 31, No. 1, 1990年1月, pp. 47-55.

[4] E. Chikofsky et al., "Reverse Enginieering and Design Recovery", IEEE Software, 1990年1月, pp. 13-17.

[5] M. Aoyama et al., "An Integrated Software Maintenance Environment: Bridging Configuration Management and Quality Management", Proc. Conference on Software Maintenance, 1988年10月, pp. 40-44.

[6] J.H. Morris et al., "Andrew : A Distributed Personal Computing Environment", CACM, Vol. 29, No. 3, 1986 年3月, pp. 184-201.

[7] 原田 "構造エディタ", 共立出版, 1987年4月.

[8] 石井 "グループウェア技術の研究動向", 情報処理, Vol. 30, No. 12, 1989年12月, pp. 1502-1508.