解　説

5. 事　例

5.1 大規模プログラミングのための環境†

細谷僚一††　伊東洋一††

1. はじめに

近年における計算機システムの発展は著しく，その応用分野も急速に拡大され，かつ複雑化，大規模化の傾向にある．いまや家電製品から宇宙衛星まで，計算機の使われていない分野はないといっても過言ではない．これらのシステムは，計算機本体，ネットワーク，端末機などの各種のハードウェアから構成されているが，これらを結合して動作させているのがソフトウェアである．大規模システムにおいてはソフトウェアの価格がシステム価格の半分以上を占め，しかも，この率は急速に増加している．しかし，これを支えるソフトウェアの開発及び管理技術は依然，手工業的性格から脱却し得ず，他の技術に比較して，生産性，信頼性，保守性において問題が多い．近年，この打開策としてソフトウェア生産環境の重要性が叫ばれている．この基本的姿勢はソフトウェア開発をプログラミング段階に限定せず，設計から保守に至る全過程を通じて効率化しようとするものである．

本稿では大規模ソフトウェア開発上の問題点及び，それに対応するためのソフトウェア開発環境の動向について NTT における例を中心に述べる．

2. 大規模ソフトウェアの開発環境の諸問題

NTT ではますます拡大する電話サービスの需要増に加え，ファクシミリやデータ通信などのいわゆる非電話サービスの需要増に応えるため，多量のソフトウェアを現行維持し，さらに新しいニーズに即応すべくソフトウェアの新規開発，改良を行っている．その規模は，数十メガステップのオーダに及んでいる．

また，各種のサービスを実現するために，ターゲットマシンや OS の種類，さらにはプログラム言語も種種にわたっており，ソフトウェア開発環境を構築するうえでも幾多の問題点[1]を抱えている．

（1） ソフトウェア自体の問題

電子交換機のようなシステムはいったん商用に供されると 10 数年間にわたって使い続けられる．その間同一ハードウェア上で種々の新サービスを実現するため，ソフトウェアの改良が継続される．しかも，改良後のソフトウェアは既存のソフトウェアやハードウェアとの上方向互換性の保証が前提とされる．この結果，ソフトウェア構造の複雑化を招き，モジュール化及びモジュールの再利用が犠牲になる．

（2） 複数アーキテクチャ（機種，OS）及び言語の混在問題

一つのサービスシステムは端末装置，通信装置，情報処理装置，交換装置などからなる複合体で構成されるのが一般である．現状ではこれらの構成要素のアーキテクチャ（機種，OS）及びプログラム言語などが異なっており，それぞれのハードウェア，ソフトウェア相互間の整合性確認試験に多大の労力を要している．

また，システムトータルな試験のための実機試験設備を揃えることも多額の費用を要している．

（3） 開発手段（環境）の問題

電子交換機では，リアルタイム性能，信頼性及びコストなどの面で厳しい条件が課せられるため，一般にハードウェアは目的専用に設計され，ソフトウェアもそのハードウェアにチューニングされる．

たとえば，汎用計算機のようにソフトウェア開発に便利なハードウェア，ソフトウェアの機能が揃っていないため，ソフトウェア開発環境として別途なんらかの手段を用意する必要がある．

（4） ドキュメントの問題

長期間にわたるソフトウェアの改良・維持には，多数の開発要員が関与する．そのため対象ソフトウェアに関する知識や経験ノウハウはドキュメント類を用いて継承する．しかし，既存の膨大なドキュメント類の

† Software Production Environment for Large Scale Software by Ryoichi HOSOYA and Yoichi ITO (NTT Software Laboratories).

†† NTT ソフトウェア研究所

かなりの部分が手書きであるため，ドキュメント自体の現行維持及びソースコードとの一致性の管理に多大の労力，費用を要している．

（5） 開発管理の問題

長期間，大規模なソフトウェア開発においては，たとえば，規模，工期，開発手段，要員のスキルなどの開発条件がさまざまであるため，前例や経験則の採用に限界があり，所要リソースの見積り，工程の進捗状況把握，品質・性能の保証などの管理を一層難しくしている．

以下の章では，上記諸問題に対応するための開発スタイル，及びそれに従った開発ツール及びシステム構成の例について述べる．

3. 大規模ソフトウェアの開発スタイル

大規模ソフトの開発は，長期間かつ多人数を要するため，一定の方法論に従って整然と行われる必要がある．このため以下の標準及び手法を用いている．

（1） 作業標準

多数の開発従事者による共同作業の結果得られる生産物を均質化して，製品の品質や生産性の向上を図るため，各工程の作業内容，生産物を定めた作業標準を策定し，これを作業の規範として開発担当者に遵守させている．作業標準の規定内容の例を表-1 に示す．

（2） モジュール化

ソフトウェア開発の生産性向上を図る方策の一つにモジュール再利用がある．品質のよい既存のソフトウェア・モジュールを流用することにより，製造，試験の工数を削減することができると同時に品質の向上にも効果がある．モジュールの再利用を促進するためには，汎用的なモジュールを体系的にライブラリとして管理しておく必要があり，モジュール管理データベースなどの実現が進められている．モジュールの作成（モジュール化あるいは部品化）においては，その汎用性，高品質を確保するため作成基準，記述基準などを規定したモジュール設計ガイドラインを設定している．

表-1 作業標準の規定内容例

分類	項目	内容例
管理規定	工程の定義と生産物	工程区分，生産物，計画・報告書，ドキュメント変更
	組織と運営方法	開発体制，会議の運営，議事録，連絡票
	共通規定	各種名称付与基準
	開発管理	開発管理指標，開発データ収集項目と方法
作業規定	ドキュメント作成要領	種類，形式，目次構成
	メッセージ，標識	メッセージ形式，コンソール入力形式
	帳票と記入要領	障害処理票，設計用紙
	設計手法	設計手法，手順
	コーディング規約	コメント記述規則
	レビュ実施法	レビュ対象ドキュメント，参加者，レビュ実施手順
	試験・検査方法	試験項目設定法，試験手順

（3） 開発管理

ソフトウェアは目に見えず，また一品生産でありプロジェクトごとに環境・要員条件が異なるため標準化が難しく，開発工数の見積り，工程進捗の管理さらには製品品質の保証などの開発管理業務を難しいものにしている．開発管理で考慮すべき要因として工程進捗・品質の他に，資源，要員など多くのものが考えられる．一般にはこれら要因ごとに予定と実績の差に基づく管理が行われている．また，進捗・品質などに関するデータを収集することで定量的に把握し，可視化するためのツール類も開発されている．

（4） 教育

ソフトウェア開発の生産性及び品質向上には技術者のスキルレベルの高揚が重要である．大規模なソフトウェア開発では多くのソフトウェア技術者（システムエンジニア，プログラマ，オペレータなど）が各専門の技術をもって分担開発体制を採ることが一般的である．したがってソフトウェア技術者教育のカリキュラムは，開発各フェーズでの作業内容に対応した，設計，プログラミング，デバッグ，テクニカルライティングなどの開発技術のほかに，工程管理，品質保証などの管理技術も含まれ，非常に幅広いものとなる．教育方法には座学と OJT があり，これらを組み合わせて効果的な訓練体系を構築する必要がある．最近では教育支援システムとして CAI の導入が活発になっている．

4. 大規模ソフトウェアの開発ツール

ソフトウェアの開発工程は，設計，製造，テストデバッグ，保守及び全体に共通な開発管理にわけられる．

以下に各工程をサポートするツール例について述べる．

4.1 設計ツール[2)]

設計作業を CRT のついたミニコンやパーソナルコンピュータで図表現を用いて会話的に行うことのできるツールの使用が盛んになりつつある．

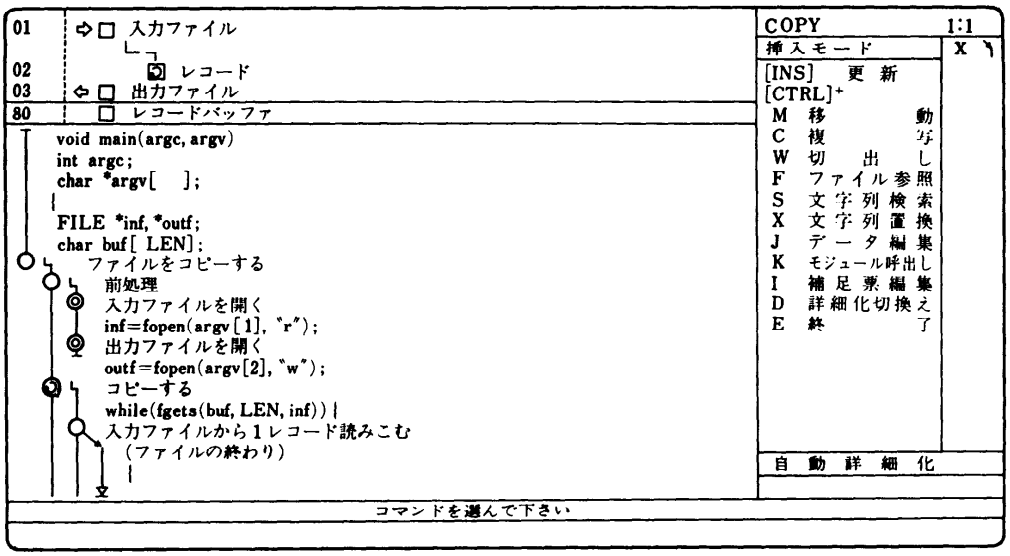

図-1 HCP チャート編集ツールの画面例

この種のツールが実用に供せられてきた背景には,

● 1画面当り約百万画素と, 設計で用いる簡易図形の表示に耐えられる能力をもった CRT の低コスト実現技術,

● 実用に耐える語彙数・変換速度をもった日本語処理技術,

などの技術進歩により, 図要素を含む設計情報の電子化が手軽に行えるようになったことがあげられる.

以下では図を利用したツールの事例としてプログラムの詳細設計作業に用いるプログラム論理図を HCP 記法で編集するツールの機能を紹介する (図-1).

● HCP 記法ガイダンスに従いキー操作で HCP 記号を選び日本語で意味を補っていくことにより, 画面上で HCP チャートを作成する. これにより, 規格化された HCP チャートが作成できる.

● HCP チャートを見ながら穴埋め的にコーディングができる. HCP 記号の並びから, CHILL, Ada, C などのプログラミング言語の制御コードを自動生成する. このためコーディング誤りが減少する.

● HCP チャートとプログラムコードが階層構造を保ち管理されるので保守性が良い.

● HCP チャートレベルで部品流通が可能である. このため流通の促進が期待できる.

4.2 製造ツール

大規模ソフトウェアの開発は, ソフトウェア製造の中核ツールであるプログラム言語とその処理系に対しても種々のインパクトを与えている.

大規模ソフトウェアは長寿命で信頼性, 保守性にすぐれていることが要求されるとともに, その製造は多人数のプロジェクトによる並行開発形態がとられるのが一般的である. これらの要求を満たす言語として, Modula-2, CHILL, Ada などの言語が出現してきた.

以下, 本節では大規模ソフトウェア開発向き言語機能, 処理系及び言語機能, 処理系及び関連ツールを統合したシステム化の例として, NTT でも開発・使用を行っている Ada* を中心に概観する[3),5)] (図-2).

(1) プログラム言語機能

(a) プログラム部品化及び並行開発を支援するモジュール記述機能

● パッケージ

データ抽象概念に基づく機能で, 他のプログラムとのインタフェースを規定した仕様部と外部からは隠蔽された機能の実現を記述した本体部から構成され, 仕様部と本体部の製造は独立的に進められる.

● ジェネリック

処理アルゴリズム, 対象となるデータの構造が同一の場合個々の差異をパラメータすることにより, モジュールの汎用化が図れる.

(b) 信頼性, 保守性の確保のためのコンパイル時及びプログラム実行時の誤り検出機能の強化, 及び並列処理記述機能

* Ada は DoD (AJPO) の登録商標である.

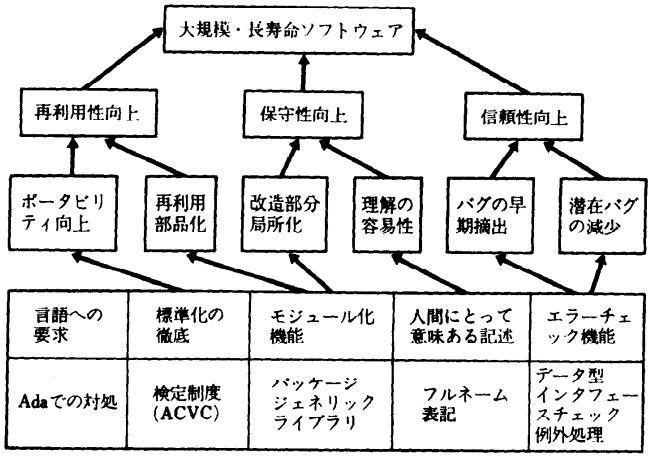

図-2 言語に対する要求と Ada での解決

• データの型定義

データの型を厳格に定義し，同一の型のデータ同士の演算のみを許すことにより，翻訳時の誤り検出を強化している．

• 例外記述

例外が発生したときの処理をメイン処理とは分離して記述する機能により，実行時の誤り処理を明確化している．

• タスク記述

並列処理の記述機能として，タスク記述機能が言語仕様化されている．

（2） 処理系

（a） プログラム単位間の相互関係のチェックにまで拡大された翻訳処理

• 翻訳時のモジュール間チェック

プログラムデータベースを利用してモジュール間インタフェースの効率的なチェックが行える．また，仕様部と本体部及び指定によるサブプログラム単位の分離コンパイルが可能である．

（3） 言語処理系を含むツールの統合化[4),6)] (図-3)

• APSE (Ada Programming Support Environment)

ソフトウェアの生産性，信頼性の向上のためには，言語だけではなく広く関連ツールを統合した環境が必要である．Ada では，仮想インタフェース及びプログラムデータベースを中心に，基本ツール群（エディタ，コンパイラ，デバッガなど），高度ツール群(プロファイラ，パス解析ツールなど）及び構成管理ツールを組み合わせた APSE が構築されている．

4.3 テスト，デバッグツール[7),8)]

（1） 情報処理ソフトウェアのテスト，デバッグツール

計算機複合体を用いた大規模情報処理システムの開発では，実機上でのソフト試験に多大の労力と設備が必要で，実機試験の環境を経済的に疑似する技術，及び実機から離れた場所から開発（リモート開発）する技術が重要である．

このため，NTT では仮想計算機システム（VM システム）及びその遠隔操作システム（リモート VM）を開発し適用している．VM システム及びリモート VM の活用により，1台の計算機上で同時に複数の疑似実機試験を相互干渉なく実施できるため，計算機時間の効率的使用，試験用設備の削減など，生産性向上に大きな効果を得ている(図-4)．

これらの目的を達成するための VM，及びリモート VM の主要機能は次のとおりである．

• 複数 VM に対して，計算機資源 (CPU 時間とメモリ）の配分を制御する機能

• 疑似実機試験環境として必要なハードウェアの疑似とその組み合わせ構成を設定する機能

• プログラム製造用の環境と，疑似実機試験環境との連動制御機能

（2） 交換ソフトウェアのテスト，デバッグツール[9)]

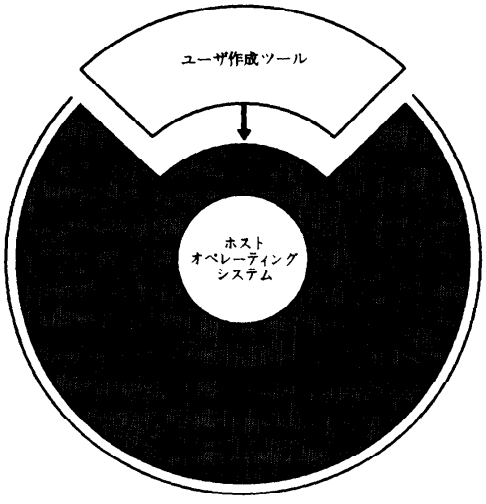

図-3 NTT Ada システムの構成

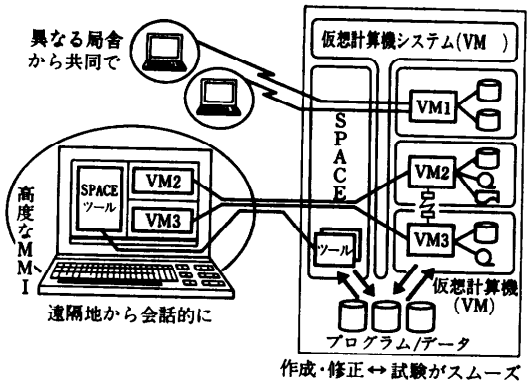

図-4 仮想計算機システムの構成例

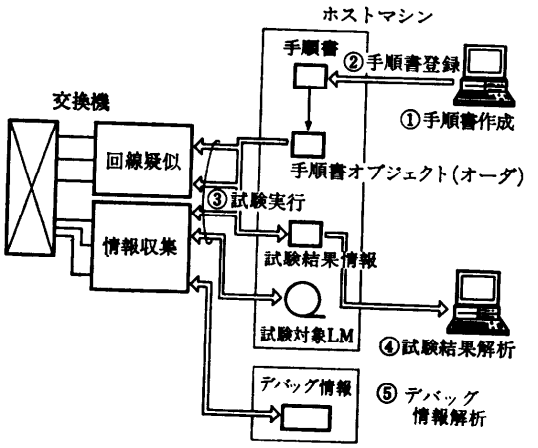

図-5 交換ソフトウェア試験自動化システム(SEINT)の構成例

交換ソフトウェアには高いリアルタイム性と信頼性が要求される．また，通話路，各種信号送受信回路などのハードウェア制御や既存ネットワークとの相互接続など複雑なインタフェースを有する．このような，要求条件から，現在の交換ソフトウェアは，ハードアーキテクチャを仮想化するOS上に実現される汎用コンピュータのAPと違い，プログラム全体がハードウェアに密着した構造になっている．この結果，交換ソフトウェアのテストデバッグ作業は複雑かつ大がかりであり，全開発工数の過半を必要としてきたことから，省力化のための種々の支援ツールが試行されている．NTTの交換ソフトウェア開発でも結合試験作業の自動化，遠隔制御及び遠隔解析支援を狙いに，次の機能を有する試験自動化システム(SEINT*)を開発している(図-5)．

- 交換機の加入者線や中継線などに対する試験用信号送受信手順の記述機能．
- あらかじめ作成，登録された上記手順に基づいて試験を自動的に実施する機能．
- 交換ソフトウェアを変更することなく，交換機の動作をワークステーションからトレースする機能．
- 試験手順書やソースプログラムと関連付けてテストデバッグ情報を解析表示する機能．

4.4 保守ツール

ソフトウェア保守のねらいはシステムの異常動作などからプログラムの誤りが顕在化したとき速やかにその原因を取り除くことにある．

これを達成するために，①製品つくり込みの段階から保守のことを想定する技術，②具体的に異常が検出されて後，修理のために行う情報収集・バグ原因解析・修正・修正の確認・ドキュメントの修正などの作業の省力化，確実化を図るための技術，③試験や試行運用中における各種情報から品質や性能上の不安な部分を事故が起こる前に予測しておく技術，などの開発が必要である．

以下にNTTで利用が進んでいる例として主として②の範疇にはいるツールを紹介する．

(1) 遠隔保守ツール

商用ソフトウェアが搭載されている装置を保守者のいる保守センタから通信回線を経由して保守するためのツールである．

このツールは保守センタの保守者と会話しながら指示の投入を促進したり，結果を表示したりする保守センタ側の機能と，遠隔にある装置上のソフトウェアの一部や当該装置とは独立した装置に，保守者の指示に従った情報の収集・保守センタへの転送機能，保守センタからの応急処理(プログラムパッチなど)の受信・パッチ実施機能などで構成される．

同種のシステムを多数保守し，バグ傾向・履歴を蓄積・検索できるようにしておくことにより原因探索の効率化が期待できる．

(2) 保守ドキュメント生成ツール

大規模なソフトウェアではバグの修理を緊急に行うため，本来ソースプログラムの変更と同時に行われるべきHCPなどで記述されたプログラム論理図などのドキュメント類の修正を後回しにしてしまうことがある．そしてドキュメント類の修正が一度おろそかになってしまうと後は常にソースプログラムリストだけを頼りにバグ修理をしなければならず，作業能率が悪くなる．

☆ SEINT: Support system for ESS software INtegration Test

これを解決するために，保守用としてのプログラム論理図をソースプログラムから自動的に生成するツールが利用されている．これは「繰り返し」，「選択」などのプログラム論理をソースプログラムのキーワードから抽出し視覚的に分かりやすいプログラム論理図を出力するものである．さらにソースプログラム上の注釈の記述規則を言語仕様の範囲で厳密に定めて，プログラムの読解性を向上させる構造情報なども併せて出力できるようにしたツールも使われている．

4.5 開発管理ツール

ソフトウェア開発においては工程進捗，製品品質などの状況を定量的に把握し，これに基づき開発作業を的確にコントロールすることが重要である．

このため，まず開発管理の作業標準として，「ソフトウェア開発管理要領」を定めた．これは，プロジェクトリーダ及び開発担当者のおのおのが，開発プロジェクトのチームの一員として，ソフトウェア開発の各工程において実施すべき作業内容を明らかにし，これらの作業に対応した管理業務及び管理項目を規定したものである．

ソフトウェア開発管理システム (PROM☆) は，本管理要領に沿った開発管理を機械支援するツールであり，開発管理の標準化と効率化を狙ったものである．

(1) 機能と構成

PROM の機能は，定量的な目標設定と，日，週，月，あるいは工程単位の管理データ収集による開発状況の把握などプロジェクトリーダによる個々のプロジェクトの開発管理業務を支援する機能と，組織として複数のプロジェクトを横断的に評価し，以後の開発管理にフィードバックするための開発データの蓄積及び分析を支援する機能から構成される．

- プロジェクトリーダ支援機能

線表，稼動，資源，品質などの計画立案，実績データ収集，品質・進捗・コスト状況に関するグラフ，あるいは週間，工程報告書を自動生成する機能で構成され，パソコン上で動作する．リーダの報告書作成作業を軽減すると同時に，プロジェクトの状況をタイムリにしかも定量的に把握することができ，問題点の早期発見とその迅速な対応が可能になる．

- 開発管理データベース管理支援機能

プロジェクトリーダ支援機能のために収集したデータを，以後の開発管理に有効活用するため，プロジェクトの開発終了までの実績データをセンタのデータベースに蓄積する．蓄積データを基にソフトウェア分野ごとに各種の分析・評価を行う機能，統計解析を基に品質，コストなどのモデルを導出支援する機能をもつ．これにより，プロジェクト横断的な評価，あるいは生産性の年度推移の評価などが可能である．

☆ PROM: software PROject Management system

5. システム構成例

システム構成例として，情報処理分野ソフトウェアの開発環境と交換分野ソフトウェアの開発環境の概要を説明する．

(1) 情報処理分野ソフトウェアの開発環境[8),11)~14)]

情報処理分野ソフトウェアは汎用計算機を用いて実現している場合が大半で，開発環境は汎用機自身のもつ豊富な開発ツール及びパソコンやワークステーション自体のツールを設計，製造，試験，保守の各工程開発作業目的に応じて併用する（セルフ型開発環境）．

実現例 (SPACE システム☆) を図-6 に示す．システム構成は大型汎用機による大規模共同利用の形態から，小型汎用機による少人数向けの利用形態まで，多様な開発環境を同一の方式で実現している．これによりさまざまな開発作業規模や作業形態用の開発環境を柔軟に構成できる．

また，集中，遠隔，分散といった開発サイドの構成に対してもネットワーク機能の具備により柔軟なシステム構成が可能である．

(2) 交換分野ソフトウェアの開発環境[9)]

2. の(3)で述べたように，交換ソフトウェアの開発は交換機本体とは別の汎用計算機やワークステーションを用いて行う（クロス型開発環境）．

実現例 (INSTEP システム☆☆) を図-7 に示す．システム構成は，UNIX☆☆☆ を搭載した複数のミニコンピュータ（またはワークステーション）を LAN で接続した分散型システムであり，システム製造・管理，ドキュメント作成・管理，及び結合試験支援の各サブシステムからなる．

また，一括コンパイルなど計算機資源消費型のジョブは汎用計算機上で実行し，テキストの編集や単体試験のような会話型作業はワークステーションで実行する機能分担となっている．

☆ SPACE: Software Production And Circulation Environment

☆☆ INSTEP: Integrated Support System for ESS Software Production

☆☆☆ UNIX は AT&T の登録商標である

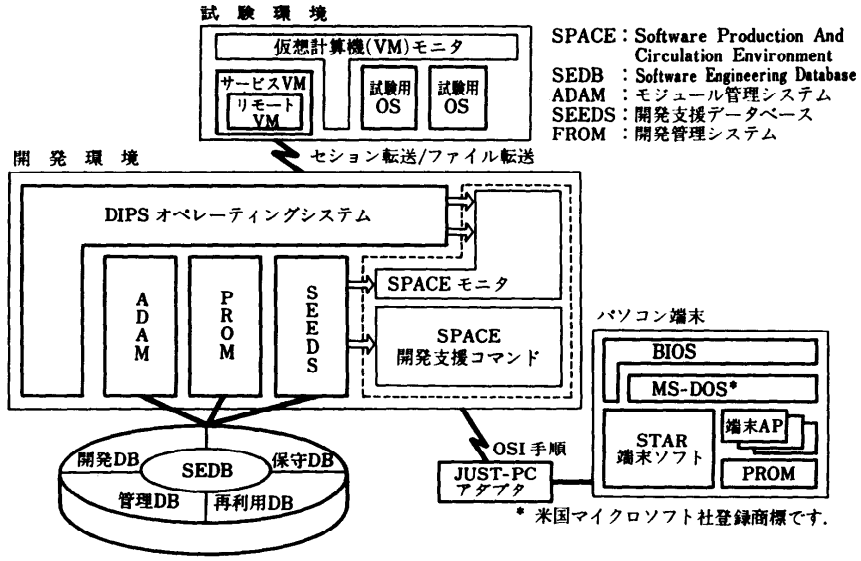

図-6 情報処理分野ソフトウェアの開発環境構成例 (SPACE)

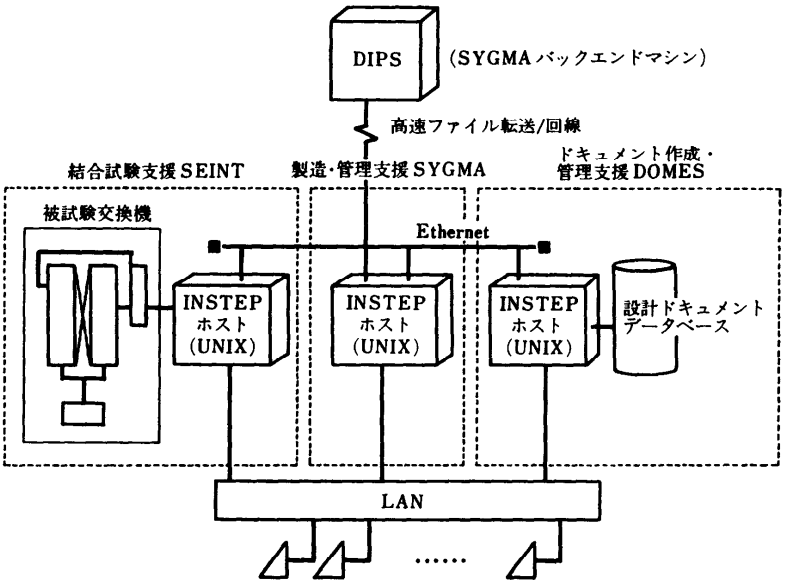

図-7 交換分野ソフトウェアの開発環境構成例 (INSTEP)

6. おわりに

大規模ソフトウェア用の開発環境について概観した．ソフトウェアの開発環境に関する技術は多岐にわたりしかも現在急速な発展途上にある．

これらの技術を組み合わせて，実際の生産現場で役立つ環境を構築するためには，現場からのニーズに対するきめ細かい対応が必須である．

現在ソフトウェア開発支援ツール類の流通及び要員の教育の効率化などのため，各種の標準化が ISO, ANSI, IEEE などで積極的に進められているが，これらをどう開発環境に取り入れていくかが今後の重要な課題である．

参考文献

1) 伊東，佐藤，長野，神谷：総合ソフトウェア生産システムの実用化，研究実用化報告，電気通信研究所，Vol. 33, No. 12, pp. 2879-2893 (1984).
2) 米田，加藤，浅見，忠海：HCP チャートを用いたソフトウェア開発支援システム，研究実用化報告，NTT 電気通信研究所，Vol. 36, No. 11, pp. 1511-1519 (1987).
3) 細谷，田中，藤丸，山口：Ada コンパイラおよび支援システムの実用化，研究実用化報告，電気通信研究所，Vol. 33, No. 12, pp. 2895-2908 (1984).
4) 福山他：DIPS プロジェクトにおける Ada の開発適用状況，情報処理，Vol. 27, No. 3, pp. 237-243 (1986).
5) Reference Manual for Ada Programming Language, ISO Standard IS 8652.
6) U. S. Department of Defence: Requirements for Ada Programming Supports Environments 'STONEMAN' (1980).
7) 長野，戸田，川崎：基本ソフトウェア走行確認作業の遠隔化方式，研究実用化報告，NTT 電気通信研究所，Vol. 34, No. 6, pp. 955-963 (1985).
8) 加藤，川崎，塩川，神谷：高速仮想計算機方式，研究実用化報告，NTT 電気通信研究所，Vol. 37, No. 12, pp. 777-784 (1988).
9) 竹中，堀田，沢井，三宅：交換ソフトウェア用総合生産システム-INSTEP-，研究実用化報告，NTT 電気通信研究所，Vol. 36, No. 6, pp. 799-809 (1987).
10) 釜渕，笠原，小田，高橋：ソフトウェア開発管理方式，研究実用化報告，NTT 電気通信研究所，Vol. 37, No. 12, pp. 785-794 (1988).
11) 長野，神谷，戸田，浅見：シナリオ機構によるソフトウェア作業標準の機械化，研究実用化報告，NTT 電気通信研究所，Vol. 34, No. 6, pp. 927-939 (1985).
12) 村田，中庭，中野，野瀬：大規模ソフトウェア生産環境の統合化，研究実用化報告，NTT 電気通信研究所，Vol. 34, No. 6, pp. 965-977 (1985).
13) 中村，戸田，奥山，小松：統合ソフトウェア生産システム (SPACE)，研究実用化報告，NTT 電気通信研究所，Vol. 37, No. 12, pp. 761-769 (1988).
14) 内藤，小野，金井，桑名：ソフト生産システムの分散処理方式，研究実用化報告，NTT 電気通信研究所，Vol. 37, No. 12, pp. 771-776 (1988).

(平成元年 1 月 11 日受付)