解　説

基本ソフトウェア開発ツールの実状†

河田　汎††　中田光宏††

1. はじめに

情報化社会と言われる今日，益々高度で多様な機能と利用技術を持った計算機システムが必要とされてきた．これに伴いソフトウェアの規模も増大の一途をたどっている．このような状況の中で，いかにして高い品質を持った大規模ソフトウェアを，経済的に開発していくかが重要な課題となってきている．近年ソフトウェア工学の立場から，より科学的な技法を用いて，ソフトウェアの開発技術を改善していくことが叫ばれており，その一環として，種々のソフトウェア・ツールが利用され，多くの成果を収めている．本稿では，他のソフトウェアと比べ，比較的その開発環境が特殊であるため，様々なソフトウェア・ツールが利用されている基本ソフトウェアに的を絞り，その開発，特に製造から保守の段階で用いられるソフトウェア・ツール（以降ツールと略す.）について当社の開発現場の実状を中心に報告する．

以下，2章では，

基本ソフトウェアの開発環境の特徴についてふれ，更にツールを必要とする背景およびツールの果すべき役割について述べる．

3章では，

基本ソフトウェアの開発現場で実際に用いられているツールを，その狙いから分類し，それらの特徴および具体的な機能について表にまとめてみる．更にこれらのツールについて，その規模や開発工程との関係，および進化の過程などについて分析してみる．最後にツールの共同利用や，ツール体系の充実の問題など，最近の動向および今後の課題について簡単にふれる．

2. 基本ソフトウェア開発の特徴とツールの役割

基本ソフトウェアの開発で用いられるツールを論ずるに当って，基本ソフトウェアとはどのような特徴を持ったソフトウェアであるのかについて明らかにしておく必要がある．一般的に基本ソフトウェアについての明確な定義はないが，ここでは図-1 に示すように，制御プログラム，サービスエイド・プログラム，言語処理プログラムなどのように，それによってハードウェアの持つ能力を十分引き出して，アプリケーション・プログラムやユーザ・プログラムを効率良く実行させるような基本的なプログラム群を指すものとする．

このような基本ソフトウェアの特徴を挙げると次のようなものが考えられる．

（1） ハードウェア技術の進展と共に，基本ソフトウェアには多種多様なハードウェアのサポートと，それらの性能を十分に発揮させることが求められる．そのためチャネルやデバイスの動作，あるいはそれらの状態など，ハードウェアの物理的な振舞いを直接捉えることが必要とされ，これに伴って入出力制御プログラムや機器構成を管理するプログラムなどのようにハードウェアのアーキテクチャに密接な関係を持つプログラムが基本ソフトウェアには不可欠とされている．

（2） ユーザの適用業務拡大に伴う高度で多様なニーズを充たすため，大規模ソフトウェアの典型となっている．

（3） 計算機システム全体の運用効率を最大限に発揮させるため，通常の事務計算や科学計算の場合とは

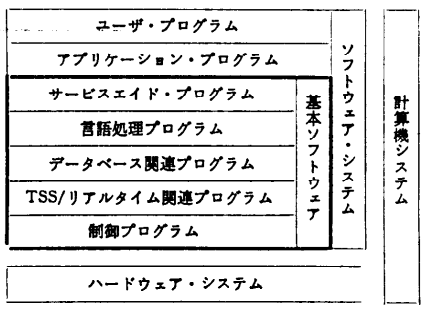

図-1　基本ソフトウェアの構成とその位置づけ

† The Present Status of Software Tools for Basic Software Development by Hiroshi KAWADA and Mitsuhiro NAKATA (Software Division, Fujitsu Ltd.).

†† 富士通（株）ソフトウェア事業部

異なったタイプの複雑な制御構造やデータ構造を持ったソフトウェアの実現が要求される.

2.1 基本ソフトウェア開発過程の特徴

基本ソフトウェアの開発は，通常計算機メーカによって行われる．開発されたソフトウェアはハードウェアと共に1組の製品として不特定多数のユーザに提供され，様々な業務を遂行していくうえで計算機システム全体の制御および操作面で重要な役割を果たす．このため，その品質の確保，保守性の向上は，計算機メーカがその企業活動を行っていくため，極めて重要な課題とされている．このような背景のもとで，基本ソフトウェア開発の特徴について次のように考えてみた.

(1) 通常，計算機メーカも営利をひとつの目的としてその企業活動を行っている以上，ユーザの望む製品をいかに早く，確実に，そして合理的な費用の範囲で開発していくかが，常に求められている.

特に，基本ソフトウェアの場合には，その開発に多額の費用と多くの労力を必要とするため，適切な開発計画の立案と適確な管理を行っていくことが不可欠となっている．このためソフトウェア開発に必要とされる様々な資源についての正確な見積り技法の確立が要求されている.

(2) 基本ソフトウェアはユーザへ提供後もハードウェアの進歩，ニーズの多様化，高度化あるいは性能改善の要求などに応じて短い期間をおいて，繰返し改造が重ねられる．このため，基本ソフトウェアには拡張性に富んだ柔軟な構造を持つことはもちろん，保守性に優れたプログラミング言語の活用，およびドキュメントの整備が必要とされている.

2.2 ツールの役割

基本ソフトウェアの開発に用いられるツールを，それらが必要とされる背景から分類すると図-2 のようになる．この分類に従い，各々のツールの役割を示すと次のようになる.

(1) 開発技術支援ツール

基本ソフトウェア自体のプログラムとしての特徴，例えば，多種多様なハードウェアとのインタフェース，規模の大きさ，および複雑な制御構造と言った開発上の様々な技術的制約を克服していくためのツールである．それらには次のようなものがある.

a. システム記述言語

ハードウェアの持つ機能や能力を十分にひき出すため，アセンブラ言語との結合を可能とし，かつ読解性と記述性を損わない基本ソフトウェア記述用のプログラミング言語のことをシステム記述言語と言う．特に特権命令を用いるための機構とか，ポインタ・データ，ビット列データを扱う機能などが必要とされる.

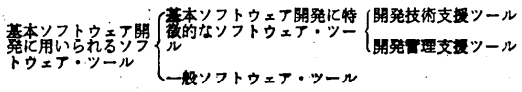

図-2 基本ソフトウェア開発ツールの分類

b. 環境シミュレータ

制御プログラムの中で，スーパバイザやユーティリティの一部は，通常その動作環境を与えるソフトウェアやハードウェアの完成を待たないうちに開発が進められる．そのためテストやデバッグの際，このような動作環境を作り出すシミュレータが必要とされる．また，オンライン関係やハードウェア障害によるエラー・リカバリ処理を行うプログラムなどを開発する場合，様々なハードウェア構成や複雑な動作条件を想定したテストが不可欠となる．しかしながらこのような状態を実際の入出力機器などを使って実現することが難しいため，これをソフトウェア的に作り出すのに環境シミュレータが利用される.

c. デバッグ支援ツール

一般に，基本ソフトウェアの欠陥は，システム全体の機能停止につながることが多い．そのような場合，原因の究明と復旧には，仮想マシン手法を用いて基本ソフトウェア自身の動作環境を作り出すようなデバッグ支援ツールが有効に利用される.

d. 性能評価支援ツール

設計値どうりに作成されたか否か，また設計値が妥当なものであったかどうかをテストしたり，システム・チューニングに必要なデータを得たりするために性能評価支援ツールが利用される.

例えば，入出力状況を測定し，システムのボトル・ネックがどこかを検出するようなツールが入出力動作の特徴をとらえて，種々作成されている.

(2) 開発管理支援ツール

基本ソフトウェアを開発する企業の立場を擁護し，効率的な開発を進めていくためのツールであり，次のようなものがある.

a. 開発費用管理支援ツール

円滑に企業活動を運営していくためには，あらゆる活動，資材の利用が原価という形で正確に捉えられ，製品の価格に適切に反映されていく機構を持つ必要がある．又製品の開発を計画する段階で，その開発およ

表-1　基本ソフトウェア開発ツールの機能一覧

大分類	中分類	ねらいおよび背景	機能例
開発技術支援	コーディング	特権命令などのハードウェアに依存した命令の記述が必要である.	・ハードウェア依存の命令の記述を可能にした高級言語（システム記述言語）
	デバッグ支援	・プログラム実行に必要な情報の設定や，実行結果の収集を任意の時点で行えること. ・異常発生時に，その原因を究明するのに必要な情報が容易に得られること.	・ブレーク・ポイントの設定 ・割込みのトレース ・メモリダンプ ・ダウン時の情報収集
	テスト支援	・過負荷状態，ハードエラーといった環境条件の設定が困難である. ・テストデータの種類，量が多く，かつテスト手順が複雑である. ・テスト結果（出力）が多く，結果の判定に手間どる.	・テスト環境生成（ハードエラーの模擬，事象の模擬など） ・テストデータの生成 ・テストの自動実行 ・テスト結果の判定（回帰テストなど）
	性能評価	・システムのチューニングに必要な情報の収集が必要である（システム生成時のパラメータに反映）. ・次システムへのフィードバック情報を得る必要がある.	・入出力状況の測定（ボトルネックの検出） ・ダイナミックステップの測定 ・メモリ使用状況の測定
	保守・運用	・障害の発生に対し，その解析に必要な情報が速かに得られること，また，過去の障害履歴が容易に参照できることが必要である. ・遠隔地での障害発生に対し，迅速な処置が必要となる. ・長時間の運転に対する省力化が必要である.	・障害発生の履歴管理 ・障害情報の遠隔地への転送 ・システム運転の自動化
	ファームウェア化支援	・システムの処理スピードの向上を図る必要がある. ・システムの部品化を促進する必要がある.	・汎用マイクロプログラム記述言語 ・マイクロプログラムの動作シミュレーション
開発管理支援	開発情報管理	高品質ソフトウェアの作成，納期の厳守，開発工数の削減といった生産管理面で，合理的かつ計画的な手法により，プロジェクトの運営・管理を円滑に行う.	・費用管理 ・進捗（工程）管理
	製品管理および障害管理	製品のデリバリ情報や，各々の障害発生／処置状況の把握が，ユーザに対する迅速な対処，応答のために必要である.	・製品のデリバリ管理 ・障害の履歴管理
	オペレーティング・システムの生成／統合	多種，多様化するソフトウェア構成に柔軟に対応しなければならない（ユーザの要望に応じたソフトウェア構成の編集）.	・プログラムを部品化し，任意の部品の組込み，取りはずしを可能にする.
一般ソフトウェアツール	プログラム管理	プログラム実体の維持管理およびそれらの関連情報の正確な把握が必要である（特に，基本ソフトウェアのように大規模になる程，これらの事は繁雑な作業となる）.	・プログラム実体の維持管理 ・プログラムの生産管理（プログラムの規模，修正量，障害件数などの管理） ・プログラムの世代管理
	文書化	ソフトウェアの開発から保守に至るまでの全般にわたって作られるドキュメントの量は膨大なものであり，その作成，維持には極めて多くの工数を要する.	・ドキュメントの作成，編集，検索，再生，複写（文章処理，漢字処理，グラフ／図形処理，作表処理，会話形式での編集）
	エディタ	・ソースファイル，データファイルなどを会話形式により，容易に変更できることが必要である. ・ファイルの内容を見易い形で表示できるようにする必要がある.	・ディスプレイを利用した汎用的な文字列データの編集 ・流れ図の作成

び保守に要する費用をかなり正確に見積ることが必要とされる．このような理由から基本ソフトウェアの開発および保守に要する費用の見積り，実績の把握を支援するツールの整備，拡充は重要な課題となっている．

b．品質管理支援ツール

品質の測定，および品質に関する情報の管理を支援するためのツールであり，製品に対する信頼性の確保の面から極めて重要な役割を持っている．

c．工程管理支援ツール

開発工程の遅延を防止するため，開発計画の管理と，進捗状況の実績管理を行うツールである．工程の明確な定義と開発用資源の客観的かつ定量的な基準が前提となるものである．

（3）　一般ソフトウェア・ツール

基本ソフトウェアと言っても通常のソフトウェアの場合と同様，製造から保守の各段階で，ソース・プログラムの管理やデータ・ファイルの管理を支援するツール，あるいは文書化支援ツールやエディタなどが，効率良く作業を進めていくため多数利用されている．

3. 基本ソフトウェア開発ツールの実状

2章では，基本ソフトウェアの開発で用いられるツールを，それが必要とされる背景から分類することを試みた．ここでは我々の開発現場で実際に用いられているツールのうち比較的良く利用されているものについてその機能を整理し分類してみる．

その結果を表-1 に示す．又それらの一部を付録 1～3 に示す．更に基本ソフトウェアの開発に用いられるツールの実状を明らかにするため，幾つかの観点からもう一段堀り下げた分析を以下に試みる．

3.1　ツールの規模

図-3 に規模別にみたツールの分布状況を示す．こ

の図からも判るように，基本ソフトウェア開発の現場では，比較的小規模のツールが多く利用されていることが判る．近年，Kernighan 達成，PWBの成果を基に小さなツールの有効性を提唱している[1]ことも手伝って，小さなツールが脚光を浴びつつあるが，基本ソフトウェアの開発現場では，従来からも，例えば，テストラン結果比較ツールや，ジョブストリームの自動生成ツール，ファイル形式変換ツールなどのように極めて有効に活用されている小さなツールが多い．特にテストやデバッグ用のツールには，まとまった単位の作業を全面的に機械化することが技術的に難しいばかりでなく，開発効率の面でもそれ程有利でないため，人間の判断，操作を適切に支援するような構造をもつものが多い．

他方，大規模なツールの代表的なものとしては，開発情報管理支援ツール，障害情報管理支援ツール，システム統合支援ツールなどのように，基本ソフトウェアの開発を総合的に管理運営していくためのツールが多い．これらは，当初から汎用使用を目的として開発されており，今後更に改良され，総合的な開発支援システムに統合化されていくものと思われる．

3.2 開発工程とツールの関係

図-4 は機能別にみたツールの分布状況の変遷を示したものである．基本ソフトウェアの開発に用いられるツールの中で，当初は，製造・デバッグ支援ツールの整備が最優先課題とされ，その後順次，テスト支援，保守・運用支援に目が向けられていることが示されている．

特にシステム記述言語の開発の必要性は早くから認識されており，10 年近い歴史を経て，生産性，保守性の両面で着実な進歩をもたらしている．又，近年運用面や広い意味での品質管理に重点をおいたツールが着実に増えつつあり，これらが今後の基本ソフトウェア開発技術の進歩に大きな貢献をすることが期待されている．

3.3 ツールの評価

ツールを評価する尺度としては，そのツールが生産性や品質の向上にどの程度の効果をおよぼしているのかを量的に捉えることが理想であろう．

しかし，現実には，生産性の定義すらも十分には解明されてはおらず[2]，このような尺度を定量的に得ることは難しい．ここでは，現場で広く活用され，評判が良いものを，良いツールと呼び，その一般的な性格をまとめてみる．

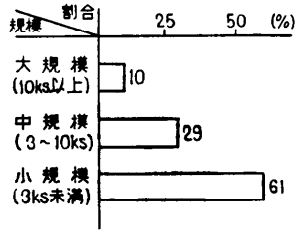

図-3 規模別ツール分布状況

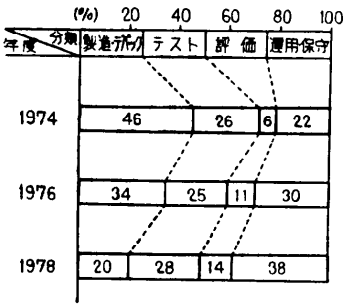

図-4 機能別ツール分布状況の変遷

（1） 使用目的が明確であること．

ツールの適用範囲，利用する工程，およびその効果などが明らかにされていること．必ずしも汎用である必要はなく，むしろ基本ソフトウェア開発に特徴的な，多種多様なハードウェアに対して有効なツールには，性能評価のための動的分析ツールや，ダンプ情報を編集して見せるツールなどのように，専用化された小道具の類が少なくない．

（2） 操作性が良いこと

使用方法が繁雑なもの，極端に出力量が多いものや実行時間がかかるものなどは利用されない．使い勝手の良さは，一般の道具にも共通した必須条件である．

（3） 信頼性が高いこと

信頼性が低いツールは有害ですらある．

（4） 構造が柔軟であること

広く利用されているツールは，例えば環境シミュレータのように，一般に開発済みのツールを母体に，次次に機能の追加，変更が加えられ，拡張されていく場が多いため，改造が容易で，柔軟な構造を持っている必要がある．

3.4 ツールの進化とその動向

ツールも通常の道具と同様，その形態と機能の発展の歴史を持っており，種々の技術的インパクト，あるいはツールへの要求などを背景に進化を重ねてきている． 図-5 は，当社の代表的なツールの進化の跡を示

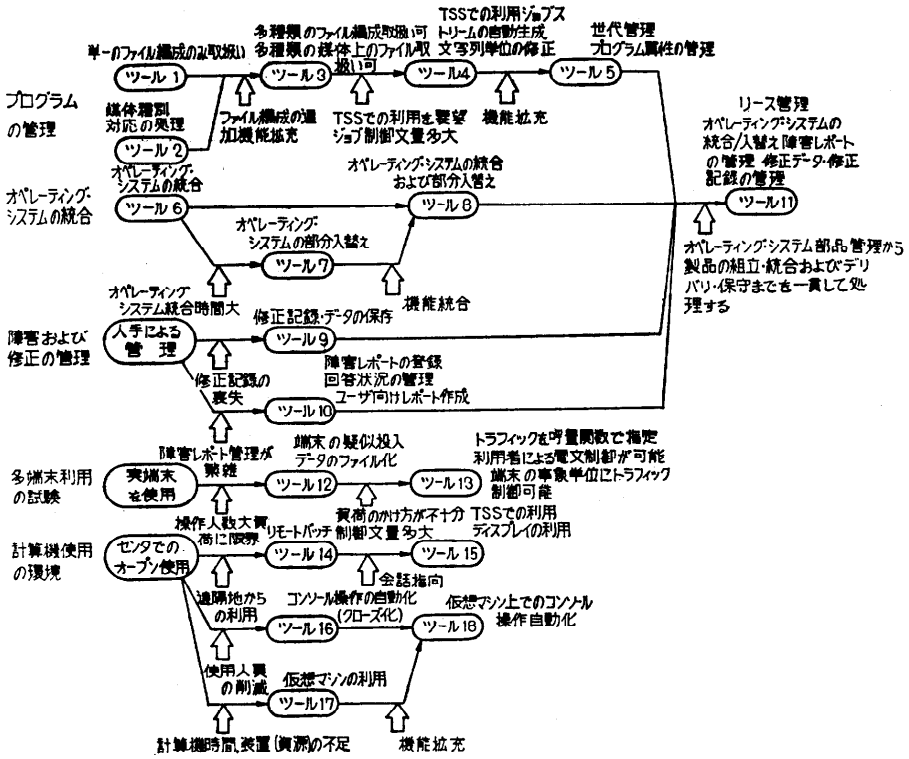

図-5 ツールの進化の例

したものである.

この図から，今後のツールの発展の一般的傾向を探ると以下のようになる.

(1) 何でもツールに任せるのではなく，人間の判断を求めながら作業を進めていくスタイルが増えてきている.

(2) 計算機利用形態の進歩に合わせて，センタでの利用から，遠隔地からでも利用可能となるよう改善される方向にある.

(3) 個々に作成されていたツールを，共通的な機能を共有するやり方で，統合化され，システム化される方向にある.

(4) 例えば，ダンプ情報からディスプレイを介して必要な部分のみを選択出力させるとか，テスト結果をまとめて表にして見せるとかのように，ツールからの出力を効率良く見せる工夫が増えてきている.

3.5 基本ソフトウェア開発に必要なツールの量

基本ソフトウェアの開発に必要とされるツールの総規模は，典型的な開発実績をもとにその使用状況を分析した結果によれば，基本ソフトウェアそれ自身の規模に比して，6から10数パーセントに達することが判った. 更にこの比率は，基本ソフトウェアの規模が大きくなる程，小さくなる傾向があることも明らかにされた. これは，開発に必要とされるツールのうちシステム記述言語処理プログラムや各種管理支援ツールなどのような基本ソフトウェアの規模とは無関係に必要とされるツールの占める割合がかなり大きいことを示している.

Brooks は，彼のエッセイの中で開発プロジェクトを運営していく際の共用ツール作成のための資源の確保の必要性を強調している[3]. 基本ソフトウェア開発

の場合も，通常新たな技術開発を必ず含むものであり，これに伴って多くのツールが必要とされる．このため，これらツールを作成するための人的，物的資源の確保は開発計画を立案する際，見過すことのできない重要な要因となる．

4. 最近の動向および今後の課題

3章でみたように，基本ソフトウェアの開発現場では開発過程のあらゆる段階で様々なツールが活用され，ソフトウェア開発技術の進歩に大きな役割を果している．

これは開発管理技術の向上，開発担当者の技術力向上などと相まって，基本ソフトウェア開発の危機的な状況を回避させることに少なからず貢献したと言えよう[4]．すなわち，年と共に高度化するユーザのニーズに対し，それに答える基本ソフトウェアを開発し続ける技術力を確立したと言える．しかしながら，これが直ちに優れたソフトウェアを確実に開発していく技術力を完全に手に入れたことを示す訳ではなく，今後解決を図っていかねばならない問題もまた多い．

ツールに関する問題は，その一角を占めているに過ぎないが，ここではその中の2，3の問題について簡単に述べてみる．

4.1 ツールの共同利用

プログラムを作成するとき，通常我々は，システムに登録されたマクロや標準関数を当然のように利用する．基本ソフトウェアの開発に用いられるツールの場合にも，技術の蓄積，発展あるいは類似ツールの重複開発の防止により，品質や生産性の向上を図っていく必要がある．それには，利用可能なツールの一元管理による有効利用の促進，あるいは汎用ツールの開発を進めていくことが非常に重要である．

通常，機械工場には，治工具の類を専門に取扱う部門があり，加工材料あるいは加工方法などに最適な道具を，いつでも手軽に利用できる制度が整っており，生産技術の向上に大きく寄与している．

IBM のユーザ団体である SHARE では SPLA (SHARE Program Library Agency)を通じてツールの登録，管理，配布のサービスを行い，ツールの共同利用の促進に大きな役割を果している．

又，UNIXのユーザ・グループの間でも最近，共通ツールの普及を目的に会合がもたれており，ツールの共同利用が重要な課題とされている．

我々の開発現場でも，近年このような制度を手本に

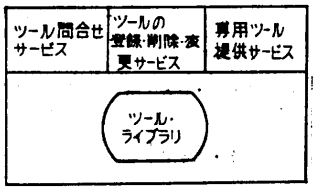

図-6 ツール共同利用管理制度

図-6 に示すようなツール共同利用管理制度の整備が進められている．今後，ツールが真に合理的，近代的な開発手段として位置づけられるためにはこの充実・発展に対し次のような観点から一層の努力が必要となろう．

（1） ツール・ライブラリ

共同利用に値する既存のツールを標準化の観点から機能・用途によって分類し，必要ならば改良を加えた後，データベースに蓄積したものである．これにより，必要なツールに対する検索，利用および管理が容易となる．今後は，このツール・ライブラリが，あらゆる開発条件，開発方法のもとでも活用できるよう，順次ツール体系の整備・充実を図る必要があろう．

（2） 専用ツール提供サービス

基本ソフトウェアのように多種多様なハードウェアとのインタフェースを持ったプログラムを効率的に開発していくためには，汎用ツールのみでは不充分である．このため，基本的な機能のみを持った標準ツールに対して，特定の開発環境に合うよう，機能の一部を変更したり，利用者持ち込みのツールとつないだりして専用ツールを提供するサービスを行う．

（3） ツール問合せサービス

基本ソフトウェアの開発に有効なツールや技法などに関する情報を円滑に流通させるために，問合せサービスを行う．

4.2 ツール体系の充実

基本ソフトウェアの開発に使用されるツールは，従来生産性の向上の一助として，それを作成するための工数と省力化工数とのトレード・オフ，あるいは，品質の向上に対する要求から言わば自然発生的に作成されてきたものが少なくない．今後は，ハードウェア技術，ソフトウェア工学などの進歩をとり込みながら，次の点に着目してツール体系を系統的に整備拡充していく努力が必要であろう．

（1） 人間—機械系を重視した開発手段の提供

言わば，人間中心のツールを指向するものでありTSS環境のもとで人間に代って機械が代行できる部分

表-2 開発中，あるいはその開発が期待されているツールの例

分類	ツール名	機能の説明
テスト支援	シミュレータ・ジェネレータ	基本ソフトウェアの一部，例えば，タスク単体などのテストを行う場合，その外部インタフェースを記述することによりシミュレータを機械的に作り出す．
	VMテスト・エイド	ハードウェア機器の操作も含め，テスト実施の大部分を VM (Virtual Machine) を利用して機械化してしまうものである．テスト作業の大部分を予めスケジュールすることにより人間の思考効率と計算機の使用効率を向上させることができる．
	プログラム・プロファイル生成ツール	プログラムの静特性や動特性を測定し，プログラム間結合の度合からモジュラリティをテストしたりプログラムの走行具合から性能テストのためのデータを得たりする．
デバッグ支援	汎用ダンパ	ダンプの内容および編集の仕方を，簡略に表記する言語を用いて，予め収集されたダンプ情報を必要とするイメージに編集・表示するものである．
	インタラクティブ・ルーチン・デバッガ	ルーチン単体でデバッグしたい場合に用いる．プログラムを記号実行したり，あるいは，実際に走行させて，未定義ルーチンの呼出しや，分岐があると，端末の人間と会話をしながら次の動作を決定していくような機構を持つ．
	プログラム・ビジュアライザー	プログラムの構造や呼出し関係などをディスプレイを使って視覚化して見せる．
保守・運用	走行履歴スタッカ	例えば，特定領域の変化の様子やタスク動作などのシステムの振舞いを一定時間，過去に遡って追跡するための履歴を保持する．トラブルが発生した近傍の情報を得ることにより，その状況を再現することが出来る．
	コンバージョン・ツール	基本ソフトウェアの移行手段として VM を利用することが考えられる．この場合 VM 機能のハードウェア化を含め性能向上が重要な課題となる．

を会話的に支援する形態である．

（2） ソフトウェア開発手段の標準化の推進

誰もが，一定の品質を持った製品を作成できるよう開発・保守に用いられる文書や，手順およびツールの標準化を推進することは，生産性の向上や品質の確保に重要な役割を果たす．

（3） 総合的なソフトウェア開発手段の提供

ツール体系をより高度で合理的な開発手段として発展させていくためには，それぞれのツールが個々独立に支援するのみではなく，例えばツール間の入出力インタフェースを合わせることによりそれらが有機的に組織化され総合システムとして支援できることが望ましい．

（4） 信頼性の高い開発手段の提供

一般にツールの故障は，生産物の信頼性低下や開発費用の浪費のみならず，開発要員の志気を低下させる原因にもなるため，信頼性の高いツールを用いることは重要である．

表-2 に，開発中，ないしはその開発が期待されているツールのうち代表的なものについてまとめてみる．

4.3 その他

基本ソフトウェアに限らず，ソフトウェアの開発技術を向上させるためには，良いツールを求めることも必要であるが，プログラマの技術力向上を図る実地教育や，標準化などの規約とそれらを推進する組織の充実などが必要である．

特にプログラマの技術力向上には，地道な学習と経験の蓄積が必要であり，ちょっとしたツールを使えば誰でもうまくソフトウェアが作れると思うのは現在の段階では幻想であろう．

5. あとがき

以上，述べてきたように，基本ソフトウェアの開発現場では，品質や生産性の向上を目的として多種多様なツールが作成され，利用されているが，それらの中で広く使われ，実際に役立っているものは，各種管理支援ツールと並んで，ソフトウェア開発の小道具とも言うべき，小規模なツールが多い．

これは，基本ソフトウェアの開発環境が非常に多様性に富んでいるため，高度な機能をもった汎用ツールを開発し利用するよりも，却って小規模なツール群を効果的に使いこなしていく方が合理的であったためと考えられる．ソフトウェアの開発が本質的には人手に頼る創造活動であるため，今後は更に使い勝手の良い小さなツール群を，道具としての限界を知りつつ，人間が中心となって自由に使いこなしていけるようなツール体系を整備充実していくことになろう．そして，今後益々拡大するソフトウェアの開発を効率良く行っていくためには，開発活動のあらゆる面で，ツールの利用を通じて計算機の有効利用を図っていくことが必要であろう．

最後に，与えられた課題が，全般的な展望を述べるには余りにも多くの問題を含んでいたため，本解説が著者らの属する一企業の基本ソフトウェア開発現場の実状の一面を報告したものであったことをお断りしておきたい．

謝辞 本解説を執筆するに際し林浩児，渋谷進の両氏には実状の調査，資料の整理などに多大な御援助を

お願いした．また，佐々木高夫，辻ケ堂信，秋山文男氏を始め多くの方々には熱心な討論を通じ多数の貴重な御意見をいただいた．ここに深く謝意を表わす次第である．

参考文献

1) Kernighan, Brian W. and Plauger, P. J.: Software Tools, Addison-Wesley, Reading, Mass. (1976).
2) Jores, T.C.: Measuring programming quality and productivity, IBM System J., Vol. 17, No. 1 (1978).
3) Brooks, F.: The Mythical Man-month, Addison-Wesley, Reading, Mass. (1975).
4) 辻ケ堂 信，君島 浩：大規模オペレーティング・システムの開発管理，電気学会，Vol. 98, No. 1, pp. 20-23 (1978).
5) 石井幸雄，市川一見，鈴野茂樹，杉浦和彦：ソフトウェアテスト用ツール，FUJITSU, Vol. 29, No. 6, pp. 225-234 (1978).
6) 三次 衛，武田 穣：ソフトウェアの保守，ビジネス・コミュニケーション，Vol. 14, No. 4, pp. 66-70 (1977).

付　録

付録1　性能評価用トレースプログラム (POATS)

POATS (Probrem-program One-step Address Trace and Simulation program) は，一般の処理プログラム（コンパイラや各種ユティリティ）の性能評価用データを妥当な時間内で手軽に収集することを目的として開発されたトレーサである．

POATS は被解析プログラムの命令を一つづつ取出して実行する．SVC 命令でも，実行後必ずその直後の命令に制御が戻ってくるものは通常の命令と同様に実行される（SVC ルーチン内はトレースの対象外である）．被解析プログラムの動作環境を変えたり制御の移行を伴う命令（別プログラムへのリンク，子タスクの生成など）に対しては，被解析プログラムの動作環境を出来るだけ変えないようにするとか，制御の移行先を見失わないようにするといった工夫が払われている．

POATS の用途としては

- VS 向きプログラム作成のための基礎データの収集（ワーキングセット計算のための基礎データ）
- 処理プログラムの CPU 時間の削減（ボトルネックの検出）
- CPU 設計資料の収集（命令頻度，オペランド長，オペランドの境界条件の分布など）
- オンライン・アプリケーション・プログラムのステップ数の測定

など多岐にわたっている．出口機能（トレースデータをファイルに出力する直前に，使用者の指定した出口ルーチンに制御を渡す機能）によって，利用者が自分の望むようにデータを加工することができ，POATS を使い易いものにしている．

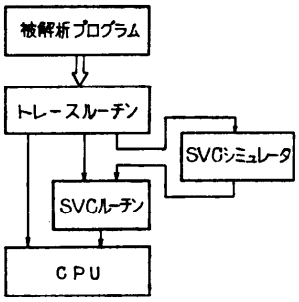

図-7　POATS の実行論理

付録2　ハード・トラブル・シミュレータ (HTS)

HTS (Hard Trouble Simulator) は，入出力装置の各種障害をソフトウェア的にシミュレートするもので次のような機能を持った環境シミュレータである．

（1）入出力装置の動作結果をシミュレートする．動作結果には正常終了と異常終了（入出力装置の障害）があり，いずれもシミュレーションの対象とすることができる．

（2）制御プログラムからの入出力要求の中で，特定の要求を指定でき，その要求に対する動作結果のみをシミュレーションの対象とすることができる．

（3）シミュレーションを行ったとき，その内容を記録し蓄積することができる．

（4）シミュレートする内容，および手順は，プログラムによって指示できる．これにより障害を発生させる条件と障害内容とを任意に組合わせて，複雑なシミュレーションを実現することができる．

HTS は，入出力スーパバイザ・プログラムにシミュレーション結果を通知する方式（すなわち，入出力装置に最も近いところでシミュレートする方式）を採っているので，入出力処理を行うすべてのプログラムがその結果を利用することができる．また HTS を動作させる環境を設定するために，制御プログラムはプログラム変更する必要もないし，特別にシステム・ジ

ェネレーションを行う必要もない.

HTS の用途としては,

- 過負荷テスト（過負荷状態で動作中のシステムに対し，障害を発生させたときのシステムの安定度を向上させるテスト）
- 複雑な障害に対するテスト（マイクロ・プログラム化されたハードウェアなどに対して特殊な障害を発生させたときのテスト）
- 障害発生のタイミングや場所などを正確に指定した，きめの細いテスト

など，他の手段ではかなり困難な分野のテストも含め広い範囲に渡っており，信頼性と生産性の向上に大きな役割を果している.

付録3　ダンプ・プロジェクタ (DUPRO)

DUPRO (Dump Projector) は，制御プログラムに障害が発生した場合，その調査，解析を能率良く行うためのツールである．障害が発生した時点での記憶装置のダンプ情報は，別のツールでいったん磁気テープに格納される．DUPROはそれをディスプレイ装置に部分的に表示しながら，会話形式で障害原因を調べることに利用される.

（1）　障害発生時点の状況を，システム・サマリ，領域リスト，および論理空間，実空間の領域などにより表示する.

（2）　制御プログラムが持っている制御表を各フィールドごとに区切って，フィールド名とその内容を表示する.

これにより，利用者は記号化したイメージで障害調査を進めることができる.

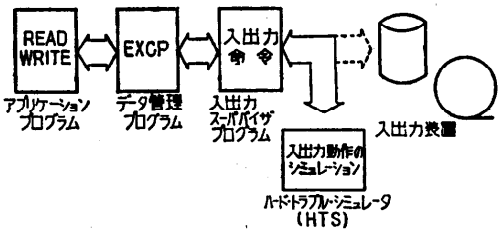

図-8　HTS の概要

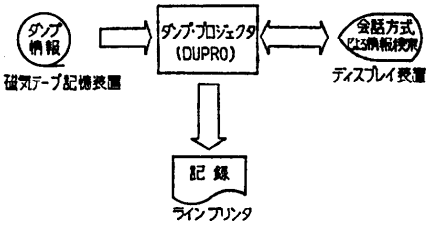

図-9　DUPRO の概要

（3）　表示された内容から望みの情報へ能率良く接近するための機能として，画面操作機能，アドレス計算機能およびポイント機能（基本となる制御テーブルを指定して，その下位にある制御テーブルを指す.）がある.

（4）　画面情報を記録として残す必要がある場合，指定により画面情報をラインプリンタに印刷することができる.

（昭和54年5月7日受付）