ソフトウエア工学 34−11
（１９８４．２．１０）

開発環境と運用環境とを結ぶインタフェースの設計

寺野　隆雄
（電力中央研究所　情報システム部）

1．　はじめに

最近，事務処理ソフトウエアの分野では，従来からの定型的なバッチ処理に加え，次々に発生する新たな情報要求に迅速に対応するために，エンド・ユーザが対話的に直接データを操作できるようなシステムの必要性が高まってきている．このようなシステムではエンド・ユーザの要求をあらかじめ正確に把握することは難しく，従来のライフ・サイクル・モデルに基づく方法論では効率的な開発を行うことはできない [Tam83]．

一方，UNIXの急速な普及 [Ker81]，Interlispの利用経験の蓄積 [Tei81]などによって，ソフトウエアの生産性向上にはたす，開発環境の重要性が大きく認識されるようになった．さらによりよい開発環境の整備をめざして，APSE [Ste81]，GandarfとSpice [Sai83]，SPS [Boe82] などの研究開発もすすんでいる．これらの高度なソフトウエア開発環境は確かにプログラマにとっても，管理者にとっても望ましいものである．しかし，プログラミングのフェーズの支援を主な対象とするものがほとんどであるため [Bal83]，我々がプロジェクトを組む機会の多い電力会社のような大規模ユーザの（いわゆるメインフレーマの）計算機システムに，これらの概念のみに基づいて，エンド・ユーザ指向の事務処理システムの開発に適した環境を構築するのは非常に困難である．

本稿では，エンド・ユーザ指向のシステムに適した事務処理システムの基本構成概念を示し，この開発を支援する環境を整備する際に生ずる問題に注目する．そして，これを解決するためのひとつの方法として，開発環境と運用環境とを結ぶインタフェースのあり方について考察する．

2．データ管理を基礎とする事務処理システムの基本構成

本節では [Ban83a] にしたがって，エンド・ユーザ指向のシステムの開発に適した事務処理システムの概念を示す．Bachman [Bac83] は，このような形態が今後の事務処理システムの中心となると主張し，これを概念スキーマ・システムと呼んでいる．

システムの基本構成は図１に示したとおりであり，この設計，開発，利用にあったっては，まず各業務に対応するビジネス・モデルを明確に定義する必要がある．ＥＤＰ部門ではデータ管理を主要な業務とし，データの量的な管理，整合性の保持，ドキュメントの整備などを担当する．一方，特定の業務処理機能は，エンド・ユーザの情報の収集，利用形態にあわせて必要に応じて組立てればよく，担当者自身が対話的に操作できる余地を充分に残すことが重要である．

データ管理に基づく事務処理システムの利点は次の３点である．

(1) 開発は必要な部分から順次行えばよく，それぞれ短期間で利用に供することができる．
(2) エンド・ユーザが直接情報を操作できるので新しい要求を満たすうえで有効である．なお，定期的な処理は従来のバッチ処理と同様，ほとんど自動化することができる．
(3) 事務処理においてつきものの原始データの入力ミスに対する原因究明，機能拡張が容易である．

また，欠点としては，システムが開かれているために，ユーザ要求を設計時に予想するのは難しくしたがって，システムの運用状況，利用方法を常に的確に把握していなければならないことが上げられる．

3．高度なソフトウエア開発環境の特性

よいソフトウエア開発環境はどのようなものであるかについては，参考文献に示したようにさまざまな意見がありうるが，現在までに成功をおさめた環境を概観すると，次の３点を目的として開発，利用されてきたと考えられる．

・　プログラマの意図を手早く，正確に計算機に伝達し，これを効率的に処理する．
　　――ユーザ・インタフェース；ツール群の整備
・　計算機のわかっている情報（対象システムの構成，機能など）をプログラマに正しく伝達するとともに，必要な情報（設計仕様，ドキュメントなど）を計算機上に蓄積して，自由に処理できるようにする．
　　――資源管理機能の整備
・　一定の方法論にしたがって（大規模な）システ

ム開発の手順を標準化する．

——方法論のサポート

以下では，これらの項目について１つ１つ考察する．

（１）使いやすいユーザ・インタフェース

開発環境を有効なものとするためには，統合化された利用しやすいインタフェースが重要である．その設計の要件としては，コマンド体系，メニュー体系の整備；マルチ・ウインドウ，マウスなどのハードウエアの利用法を含むヒューマン・ファクタ；入力を支援するエディタの問題が上げられる．また，開発環境のユーザ・インタフェースはユーザが，自由に拡張，変更してカストマイズできるものでなくてはならない．

（２）便利なツール群の存在

Interlisp [Tei81] における DWIM,FIX,REDO, UNDO に見られるような一つ一つは単純であるが，高度な機能をもつツール群が必要である．これらのツールは，[Ost81] で指摘されているように個々の工具として使いやすいばかりではなく，任意に組合わせて使用できるように統合化されていなくてはならない．

（３）ソフトウエア資源管理機能

ソフトウエアのライフサイクルを通じた支援を行うためには，開発時の情報を保存する資源管理機能が必要である．これには，ソース・プログラム，仕様書などのバージョン管理，ドキュメンテーション・ツール，大規模システムの構成管理，プロジェクト管理，などの諸機能がある．

（４）方法論のサポート

Lispのプログラミング・スタイルにしたがい実験的なシステムの開発の効率化を目的としたInterlisp [Tei81]，プログラムの逐次的構築と，多人数によるプロジェクトの管理とをねらった Gandarf [Sai83] など特定の方法論に基づくシステムにはすぐれたものが多い．また，事務処理システムの開発には，方法論に基づく手順の標準化が特に重要であり従来からさまざまな手法が提案されている．そして，事務処理システムを対象とする機械化システムもいくつか発表されている [Har84]．

さらに，以下の２点も高度な開発環境の特性として重要である．

（５）環境の自己完結性

最初は個人が使用するごく小規模な環境を作り，それを徐々に拡大することによって大きなプロジェクトにも適用可能な，よりよい環境を構築することができる [Ost81],[Sai83]．これが成功するのは，環境の開発者と利用者が同一であり，試作したツ－

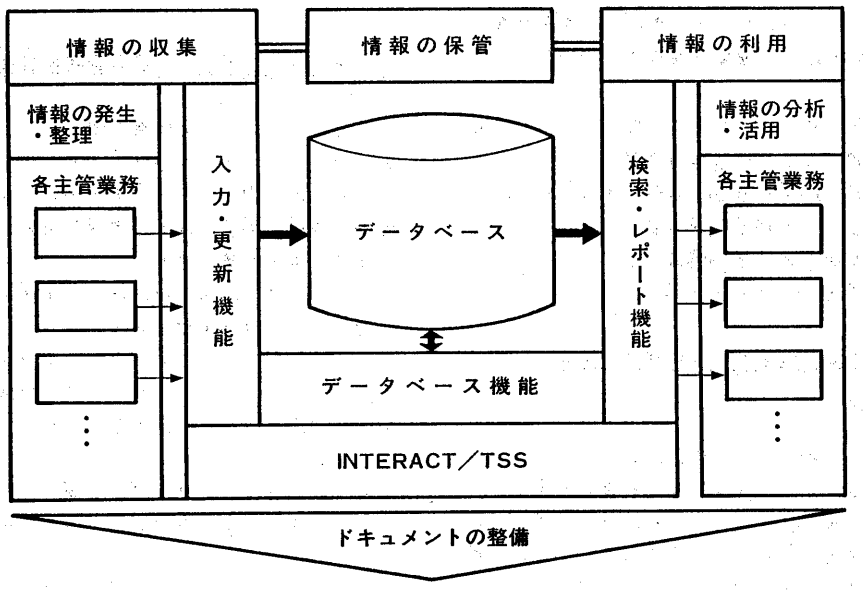

図１．データ管理を基礎とする事務処理システムの基本構成

ルをすぐにテスト，改良することができ，また，環境の持つ機能を自らの開発に利用することが可能なためと考えられる．すなわち，環境が自分自身の体系のなかで閉じており，その開発が容易となるからである．この意味では，よい環境を整備するためには開発環境と運用環境が一致していることが重要なポイントとなる．

（6）開発機と運用機の分離

UNIXでは，ソフトウエア開発と完成したプログラムの実行は全く別の作業であると考えて環境を構築している[Ker81]．また，Adaでも組込み型ソフトウエアの開発を前提として，APSEの実現を計画している[Ste81]．このような環境で開発されたシステムは，当然その環境には適合しにくい性質をもつが，それをターゲット・マシンに移動させることによって，運用後に生ずる問題を切り離している．

4．大規模ユーザで環境を構築する場合の問題点

本節では，§2で述べた各項目に対し，大規模計算機ユーザが開発環境を構築する場合に生ずる問題点について論ずる．これが困難な理由としては，以下の3点が上げられる．

（1）ツール群の整備

最近，大型計算機用でも，個々の機能においては比較的よいツールが手にはいるようになった．また，簡易言語，TSSコマンド・プロシジャなどによってもユーザが単純な機能のツールを容易に作成できるようになった．しかし，個々のツールを組合わせ，統合化して開発環境を構築しようとすると，大型計算機のＯＳが複雑すぎるために，繁雑な作業が必要となる．

（2）既存のアプリケーションの存在

新しい環境に適合しにくい既存のアプリケーション・ソフトウエアの存在を無視できないために，開発環境のみを充実させてもなかなかその効果が現れないことが多い．そのため，原理的には単純な機能を実現しようとしても，システムは大規模となる．

（3）開発環境と運用環境の分離と統合

開発環境の整備と事務処理システムの開発とは本来全く性格の異なる仕事であり，前節（5）で述べた自己完結性は大規模ユーザの場合期待できない．したがって，開発環境を整備していくためには開発と運用とを明確に区別しておくことが望ましい．しかし，事務処理システムの規模が大きいほど，開発環境と運用環境とがレベルの揃わぬまま混然一体と

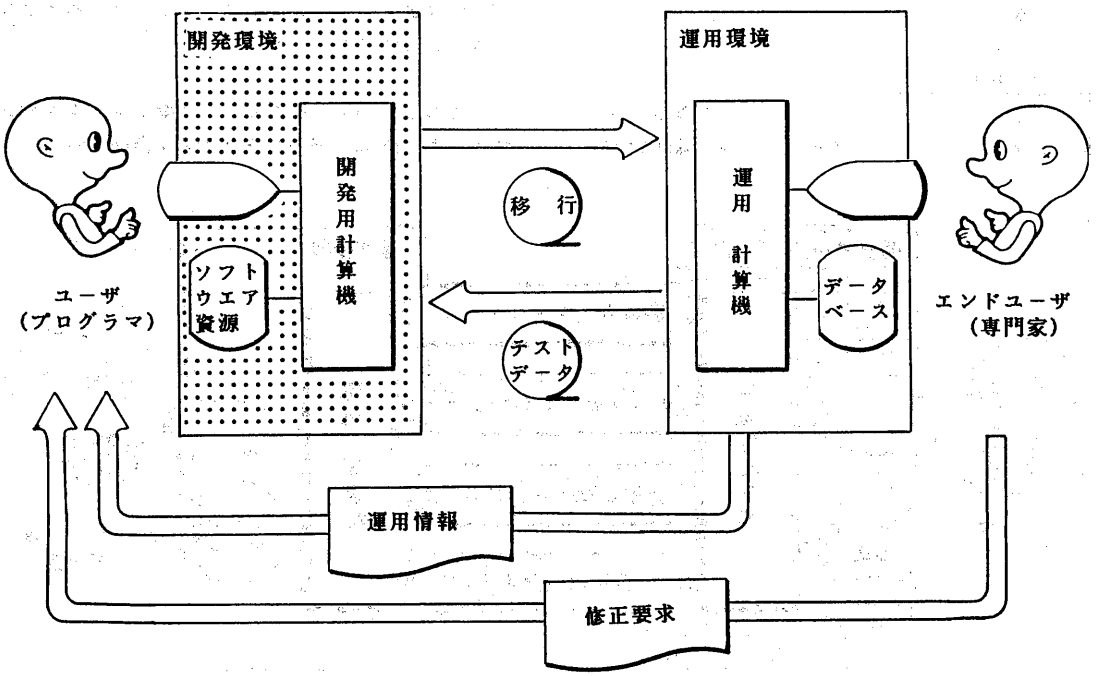

図2．ソフトウエア開発環境と運用環境の関係

なって存在しているのが現状である．

ところが，これを前節（6）で述べた形で単純に分離することは，§2で示したエンド・ユーザ指向のシステムを考える限り不可能である．すなわち，このようなシステムでは，既存のデータベースや，ソフトウエア部品を利用することが多いため，新規開発の場合にも，運用中のシステムの情報が必要となる．したがって，開発用計算機と運用計算機とを分離するためには，単に開発環境のみを整備して専用システムとするだけでは不充分で，そのほかに，エンド・ユーザのフィードバック；運用中のシステムの性能評価；開発時のテスト・データ収集；完成したソフトウエアの移行；が容易にできなくてはならない（図2を参照）．

5．開発，運用環境間のインタフェース

本節では，上のような状況を改善するための1つの解決策を示す．すなわち，（少なくとも概念的に）開発用計算機と運用計算機とを分離して，その間のコミュニケーションの場を定義することを提案する．このためのインタフェースとしては図3に示すような，移行用のインタフェース；テスト用のインタフェース；運用情報収集用のインタフェース；利用者の要求収集用のインタフェース；の4つが考えられる．

(1) 移行用のインタフェース

完成したソフトウエアをスムーズに運用にうつすためのインタフェースである．これと資源管理やバージョン管理の諸機能とを組み合わせることによって，ジョブ制御言語やデータベースの切替えなどの移行作業を効率化できる．

(2) テスト用のインタフェース

既存のデータベースから必要なテスト・データを収集するためのインタフェースである．これを用いることによって，新規システムに対する要求にあわせてテスト用に既存のデータベースのスキーマを変更する，すなわち，新しいビジネス・モデルを定義することができる．

(3) 運用情報収集用のインタフェース

システムの稼動状況などの運用情報を収集するインタフェースである．これは，プログラムより上のレベルで業務処理機能の動的特性を測定する一種のダイナミック・プロファイラと考えることができる．これによって，処理内容や処理効率を改善するためのデータとすることができる．

(4) 利用者の要求収集用のインタフェース

従来の環境では把握しにくかった，ユーザがソフトウエアを利用してはじめて得られる要求の変化を積極的に収集するためのインタフェースである．これにより，エンド・ユーザの意見をドキュメントとして収集するほか，さまざまなユーザ特性を把握することができる．

この4つのインタフェースの概念は，次節で示すように，開発と運用とを同一の計算機で行ってい

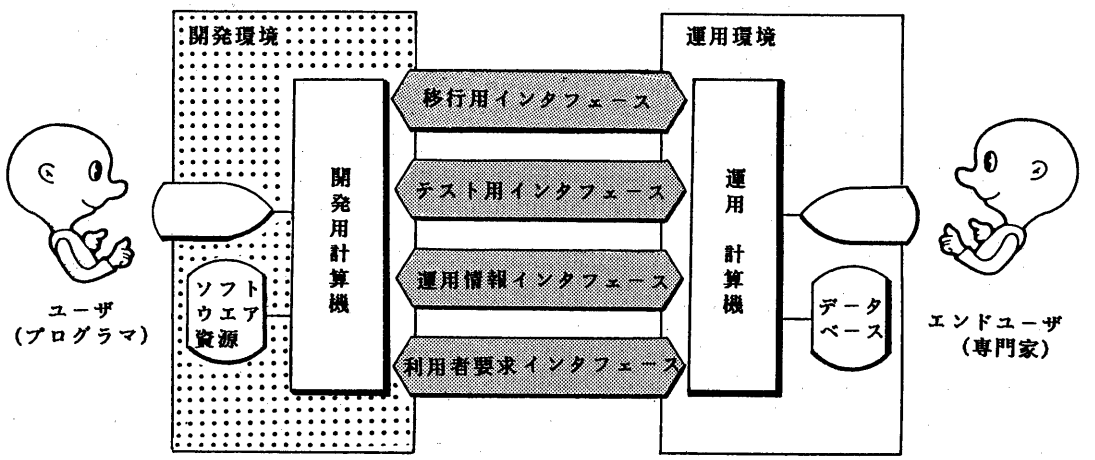

図3．開発環境と運用環境の間のインタフェース

る場合も有効であり，これらを明確化することによって開発環境と運用環境を概念的に分離して扱えるようになる．また，これらのインタフェースは，開発環境の側からみると，ソフトウエアの動的特性を測定したり，テスト・データを収集したりするテスト環境を拡張したものとして認識でき，また，運用環境の側からみると，これまで大きな部分を占めていた不明瞭な開発環境の影響を運用の外部へまとめ上げたものとして認識することができる．

5．事務処理システムにおける実験

電力中央研究所では，事務処理システムをメイン・フレーマのツール群とデータベースを組合わせることによって，ラピッド・プロトタイピングの方法で開発，運用中である[Ban83a]．このシステムは2節で示したような基本構成を持ち，ＥＤＰ部門によるデータ管理とエンド・ユーザによる簡易な業務処理の実行をめざしている．このような開かれたシステムでは，ユーザの使用法，要求の変化をあらかじめ予想するのは難しい．そこで，これを例題に用いて前節に示したインタフェースの有効性を確認する実験を行った．

現在のところ事務処理システムの規模が比較的小さいので，移行用のインタフェースとしては通常のエディタ，テスト用のインタフェースとしてはメーカ提供のツールを用いて評価を行うことが可能である．そのため，運用情報と利用者要求インタフェース，つまり，利用状況の把握の支援を目的としてPROLOG/KR[Nak83] を利用したエンド・ユーザの作業ログ解析システムを試作した．このシステムは前節で説明したインタフェースの意味からは，図４のように位置づけることができる．

システムの機能は以下の通りである．

I) 各事務処理業務に対応するコマンド・プロシジャに，ログ収集用のプローブを挿入する．
II) エンド・ユーザが計算機を利用するたびに，実行シーケンスを表わすログ情報を収集する．
III) ログ情報をPROLOG/KR で処理できるようにデータベース化する．（PROLOGのFACTの形式に変換する．）
IV) 各種の検索を行い，業務処理の作業量，TSS コマンドの実行パス，データベースのアクセス回数などの統計レポートを出力する．

TSS コマンドやコマンド・プロシジャの実行パスは，事務処理の業務内容を表わすものであるから，上記の情報を用いることによりエンド・ユーザが実際に行っている作業内容を把握することができる．

試作したログ解析システム，対象とした事務処理システムは，ともに利用を開始してからまだ日が浅いために，本稿では，システムの評価を定量的に示すことができないが，定性的には，本システムを利用することにより，次のような対象事務処理システムの性質を把握することができた．

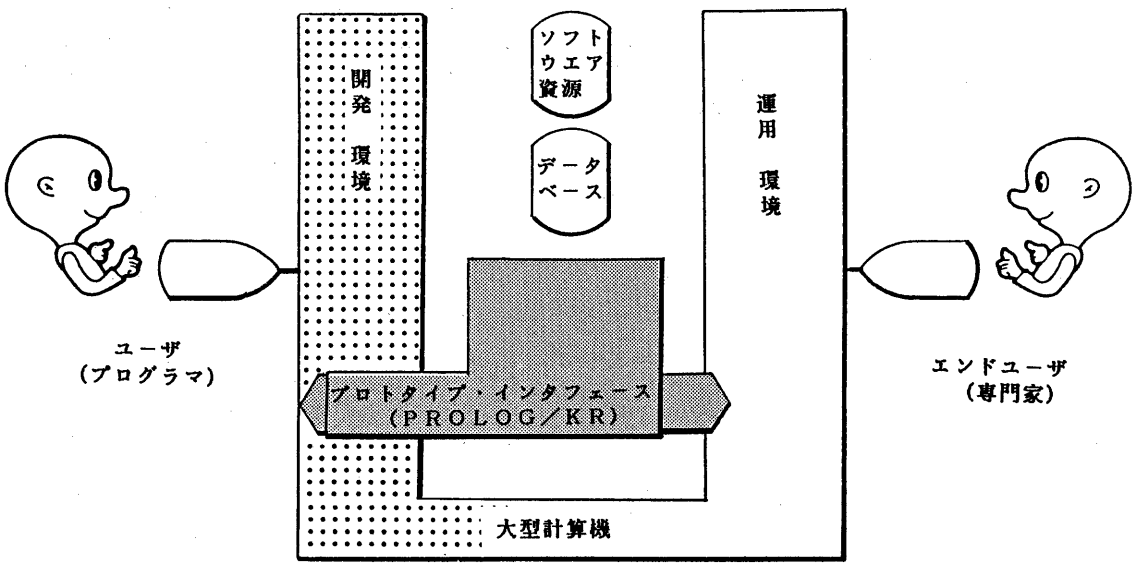

図４．プロトタイプ・システムにおける開発環境と運用環境の間のインタフェース

(1) 現在のところではエンド・ユーザは事務処理システムの機能のごく一部のみを利用しているだけであり，この意味では対象事務処理システムの設計は機能をもりこみすぎたと考えられる．
(2) 対象システムは，非定型的な業務を支援できるように設計したが，利用頻度を分析すると周期的に特定の機能が使われることがわかった．これは，業務処理の一部をバッチ化できることを意味する．
(3) 特定の機能（マスタ・ファイルのチェックと印刷）が頻繁に繰り返されることが多く，この部分をより容易に実行できるような機能が必要である．

電力中央研究所の計算機利用の大部分は技術計算であるため，そのごく一部である対象事務処理システムの特性を把握することは，一般には非常に難しいが，本システムを使用することにより容易に分析を進めることができたと考えている．

6．おわりに

本稿では，エンド・ユーザ指向の事務処置システムの開発はデータ管理を基礎とするべきであること，ならびに，それに適したソフトウエア開発環境を整備する場合には，開発，運用環境の間のインタフェースを明確に定義する必要があることを主張した．そして，小規模な対話型システムにおける実験によって，この概念の有効性を示した．

本稿の内容は中国電力株式会社との共同プロジェクトである メンテナンス・サポート・システム MSF の第Ｉ期，第II期分の開発[Ban83b],[Ban83c]と，電力中央研究所における事務処理ソフトウエアの開発の経験の中から生まれたものである．今後は，本稿で示したインタフェースの概念をさらに精密化し，一般の大規模計算機ユーザの環境開発に役立てていきたい．

参考文献

[Bac83] Bachman,C.: The Impact of Conceptual Schema Systems on Business Information Systems. to appear in Proc. Software Maintenance Workshop, (Dec. 1983).

[Bal83] Balzer, R., Cheatham,Jr.,T.E., and Green, R.: Software Technology in the 1990's: Using a New Paradigm. COMPUTER, Vol.16, No.11, pp.39-45 (Nov. 1983).

[Ban83a] 坂内広蔵，鈴木道夫：データ管理を基礎とした業務処理システムの構築－ある管理システムの構築・活用を例に－．電力中央研究所報告 582018, (1983 年 8 月).

[Ban83b] 坂内広蔵： MSF プロジェクト報告書第II期中間報告 テスト・サポート・システムの要求仕様．電力中央研究所報告 583502, (1983 年 8 月).

[Ban83c] Bannai,K., Suzuki,M., and Terano,T.: A Total Approach for the Maintenance Problems through Configuration Management - Maintenance Support Facility MSF －. Proc. COMPSAC 83, (Nov. 1983).

[Boe82] Boehm,B.W., et al.: The TRW Software Productivity System. Proc. 6th.Int. Conf. on Software Engineering, pp.148-156 (Sep. 1982).

[Har84] 原田 実：移り変わるソフトウエア生産－手作りからオートメーションへ－．日経コンピュータ掲載予定.

[Ker81] Kernighan,B.R., and Mashey,J.R.: The UNIX Programming Environment. COMPUTER, Vol.14, No.4, pp.12-25 (Apr. 1981).

[Nak83] Nakashima,H.: Prolog/KR User's Manual. 東京大学計算機センター．(1983 年 3 月).

[Ost81] Osterweil,L.: Software Environment Research: Direction for the Next Five Years. COMPUTER, Vol.14, No.4, pp.25-34 (Apr. 1981).

[Sai83] 斎藤信男： ソフトウエア開発環境とプログラミング環境のプロジェクト - Gandarf と Spice プロジェクト． bit, Vol.15, No. 1, pp.19-29 (1983 年 1 月).

[Ste81] Stenning,V. et al.: The Ada Environment: A Perspective. COMPUTER, Vol.14, No.6, pp.26-36 (June 1981).

[Tam83] 玉井哲雄：ソフトウエア開発におけるプロトタイピング． bit , Vol.15, No.13, pp. 9-15. (1983 年12月).

[Tei81] Teitelman,W., and Masinter,L.: The Interlisp Programming Environment. COMPUTER, Vol.14, No.4, pp.25-34 (Apr. 1981).