U.D.C. 681.32.06.002:681.326

ソフトウェアエンジニアリングワークベンチシステム ——構成と基本技術——

Software Engineering Workbench System ——Configuration and Basic Technologies——

前澤裕行* *Hiroyuki Maezawa*
小林正和* *Masakazu Kobayashi*
林　利弘** *Toshihiro Hayashi*
片岡雅憲*** *Masanori Kataoka*

計算機システムでのソフトウェアの重要性への認識が深まるにつれて，その生産を効率的に行なうための技術の開発に拍車がかかっている。ソフトウェアの生産を支援するため各種のツールが開発されており，これらが体系的に結合されて総合的な開発環境へと脱皮しつつある。ソフトウェアエンジニアリングワークベンチはその中核となるもので，種々のツールが効率よく使用できるように搭載された作業台である。

本報では，ワークベンチの要件を明らかにし，その種類と構成を概説する。更に，ツールの有機的結合の側面からワークベンチの基本技術を明らかにする。

1　緒　　言

ソフトウェアの大規模化，複雑化，多様化に伴い，生産形態が変化しつつある。手工業・労働集約型の形態から，機械工業・知識集約型の形態，いわゆるソフトウェアファクトリーへの移行である。ソフトウェア ファクトリーの実現には，組織，人員，教育，管理などの改善が必要であるが，中でも，設備面での改良は特に重要である。

設備には，開発用の計算機設備のほかに，設計，製造，検査などを支援するツールが含まれる。ツールの多くは，方法論，技法と表裏一体であるものが多い。この意味で，方法論，技法を含めて「開発環境」が重要であるといえる。

ワークベンチ(ソフトウェアエンジニアリングワークベンチ)は，開発環境の中核となるものである。ワークベンチはその名の示すとおり，ソフトウェア開発用の作業台である[1]。この作業台には，ソフトウェア開発のために用意された種々のツールが手軽に利用できるように装着されている。大工仕事に例えると，のこぎり，ドリル，グラインダーなどの道具と，それを使うための万力，電源などを用意した作業台に相当するものである。

ワークベンチは，単なるツールの寄せ集めではない。ツールに一定の規格を設けることにより，ツールを有機的に結合させて利用することを可能とするものである。本報では，ワークベンチについて，要件，種類，構成，基盤技術などについて概説する。

2　ワークベンチの要件

ワークベンチの目的は，ソフトウェアの開発に従事する人人に，統合的なツール利用環境を提供することである。この実現のためワークベンチに必要とされる主な条件は，以下に述べるとおりである。

(1)　ツールが多次元的に収集されていること。

ソフトウェア開発支援ツールの始まりは個別ツールであった。それらが，二つの方向から統合化されつつある。一つは，ライフサイクルモデルに基づく時間軸(工程横断)的一貫化であり[2]，他は，米国ベル研究所のUNIX/PWB[3]を代表とする空間(工程内ツールの充実)的統合化である。ワークベンチには，時間と空間を二つの軸とした平面的なツールの統合化，更には対象分野を軸に加えた立体的統合化が必要である。また更に，進化するツールを常に更新させていく管理的機能も重要である。

(2)　ツールの有機的結合が実現されていること。

ワークベンチでは，ツールを単に集めるだけでは効果は少ない。ツール間のデータの授受，重複機能の統合化，マンマシンインタフェースの統一など有機的な結合が必要である。

(3)　独占的使用が可能なこと。

ツールの利用で，他人の妨げを受けないことは重要である。CPU(Central Processing Unit)，メモリなどのリソースの共有により，必要なツールがすぐ使えなかったり，ツールの応答が著しく劣化したりすることは，開発環境を阻害する。

(4)　親和性の高い利用者インタフェースであること。

ワークベンチでのマンマシンインタフェースは，操作性，ビジュアル性，応答性の高いものであり，学習容易なものでなくてはならない。更に，初心者にもベテランにも使いやすいものとする必要がある。

3　ワークベンチの種類

ソフトウェア開発でのすべての業務に適用できる万能的なワークベンチを作ることは，技術的には可能であるが，コスト的には高価となる。幾種類かのワークベンチを用意し，それを体系的に構築するのが現実的である。

ワークベンチの分類には種々の尺度を考えることができるが，中では，対象となる情報と処理内容のカテゴリーに対応して種類を設ける(**表1**)のが，コストとの対応で合理的である。この尺度に従って種類分けした結果が同表である。

第一のワークベンチは，プログラムコードに代表されるテキスト情報を対象として，種々のツールを利用できるようにするもので，プログラミングテスト段階に適用されることからプログラマ用と名付けた。第二は，表やテンプレート図面

* 日立製作所システム開発研究所　** 日立製作所大みか工場　*** 日立製作所ソフトウェア工場

表1　ワークベンチの種類と機能　対象とする情報と処理内容のカテゴリーに対応して，ワークベンチを4種類に区分けした。

ワークベンチ種類		適用業務	機能	備考
1	プログラマ用	• コーディング • テスト • デバッグ	• テキスト処理(プログラム，コマンドほか) • コンパイラ • デバッガ • テストツールほか	――
2	設計者用	• ソフトウェア設計 {モジュール分割 ファイル，テーブル構造 処理手順ほか	• 図表処理 (プログラム構造図，PAD，モジュール仕様書ほか) • ソフトウェア仕様情報管理 • 対話形設計支援ほか	プログラマ用機能をも包含
3	計画者用	• ニーズ分析 • システム設計 {機能 入・出力構造	• 画像処理(写真，手書き文書) • システム仕様情報管理 • 対話形設計支援ほか	プログラマ用，設計者用の各機能をも包含
4	管理者用	• 工程管理 • ライブラリ管理	• ビジネスグラフ，テキスト処理 • 工程情報管理 • ライブラリ管理	プログラマ用機能をも包含

注：略語説明　PAD(Problem Analysis Diagram)

など仕様書情報を処理対象とするツール群のためのもので，図形処理機能をもたせる。これは，主に設計段階に用いるので設計用と名付けた。第三は，計画用で，手書き図面や写真など画像情報を処理対象とするツール群を搭載するものである。第四は，表やビジネスグラフなどの処理機能をもたせた管理者用ワークベンチである。

これらの四つのワークベンチは，全く別個の構成とすべきでなく，プログラマ用，管理者用，設計用，計画用へと上位互換性をもたせることが重要である。

4　ワークベンチの構成

ワークベンチは，ハードウェアとソフトウェアの統合体である。ハードウェアとしては，計算機，端末，通信機器を要素とし，ソフトウェアとして，オペレーティングシステム部，統括管理部，応用ツール部を要素とする(**図1**)。

ワークベンチのハードウェア形態は，集中形から分散形へと移行しつつある(**図2**)。複数の端末から，タイムシェアリング環境下で，大形計算機を共同利用する集中形は，リソース割当て上の制約が多いこと，サービスの切換に伴うオーバヘッドなどから大形機の高性能が必ずしも生かされない。ソフトウェア開発業務での計算機との対話の内容は，テキストや図面の編集など比較的単純であり，個人別でみると必ずしも大形機の高性能を必要としない。この点に着目して，個人に計算機を割り当てることにより，集中形の問題を解決するのが水平分散形である。水平分散形の実現に当たっては，安価で高性能のパーソナルコンピュータが不可欠であるが，現

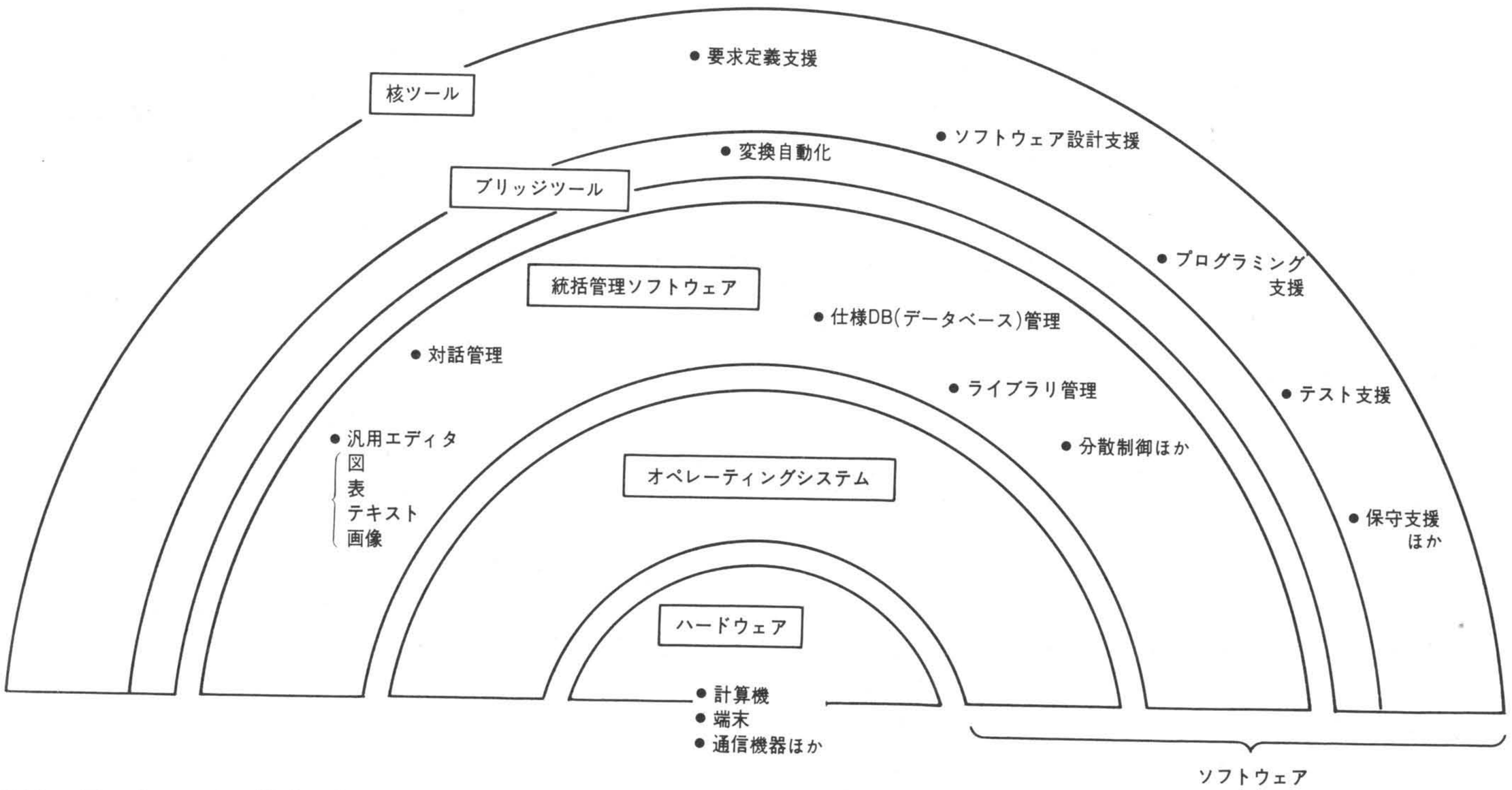

図1　ワークベンチの構成要素　ワークベンチは，ハードウェアとソフトウェアの統合体である。ワークベンチのソフトウェアは，核ツールと，その接続を行なうブリッジツール，ツール間の共通部分を抜き出した統括部などから構成される。

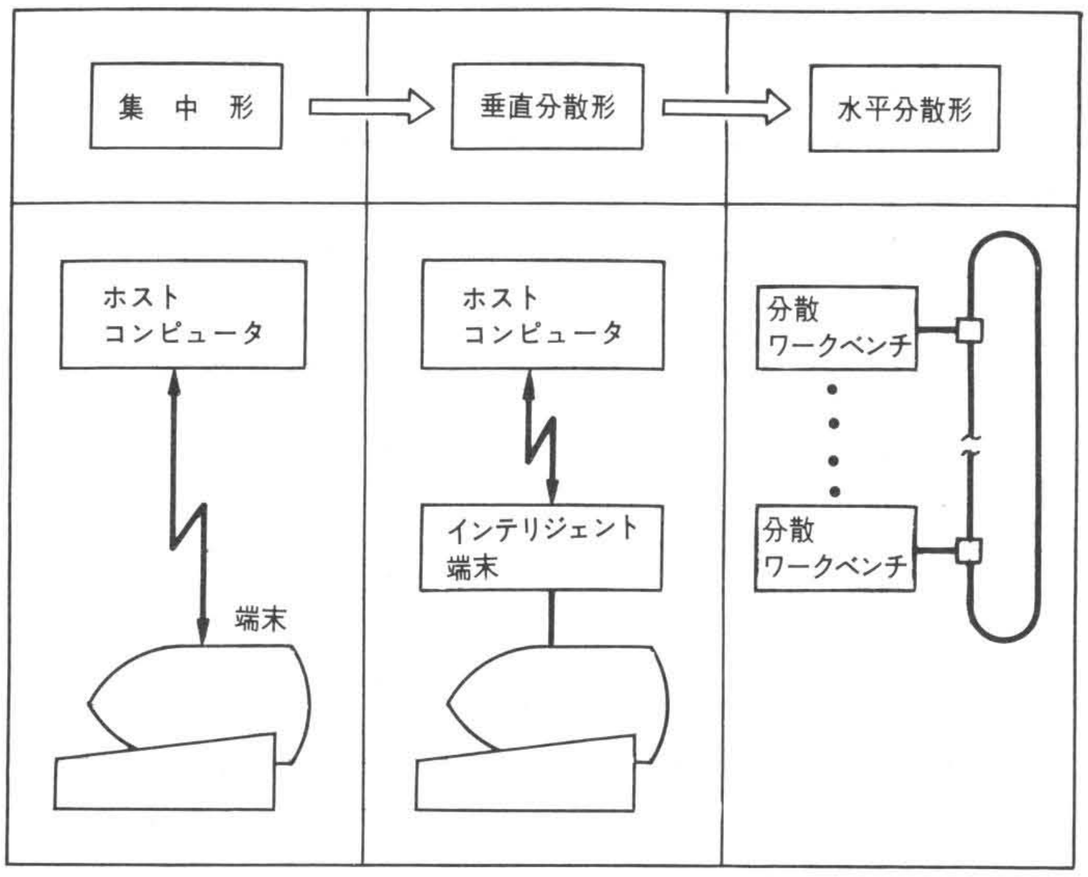

図2　ワークベンチのハードウェア構成の推移　ハードウェアの高性能化，低コスト化に応じて，ワークベンチは分散形化し，個人専用となってゆく。

時点では，設計用，計画用に適用可能なものは高価である。このため，過渡的な現実解として，両者の折中案である垂直分散形が構築されている。

ワークベンチソフトウェアでの応用ツール部の中心となるのは，設計情報やプログラム情報の生成を支援する核ツールである。これらの核ツールに加えて，それらを接続(ある核ツールの出力を他の核ツールへの入力へ自動変換)するブリッジツールをワークベンチはもつ。

統括管理部は，ツール間の共通機能を独立化したものである。対話管理，図，表，テキストなどの汎用エディタ，仕様やプログラムなどのデータベース管理，ドキュメントの出力，プログラムやツールのライブラリ管理，分散制御などの機能がこの部分に含まれる。

5　ワークベンチの基盤技術

ワークベンチは，極めて統合的なシステムである。それを構築するために必要な技術は，計算機，ローカルエリアネットワークなどのハードウェア技術，オペレーティングシステム，ソフトウェア設計，コンパイラなどのソフトウェア技術など多岐にわたる。しかし，これらの中で幾つかは，他の分野，例えばOA(オフィスオートメーション)と基盤を共通とするものも多い。ワークベンチ固有の基盤技術は以下に述べるとおりである。

(1)　ツール間の有機的結合のための技術

ワークベンチの主要な特徴は，搭載されるツールの数が多いことである。設計や製造の技法開発が進めば進むほど，ツールの種類は増加するであろうし，それらのツールが使い込まれるにつれてその周辺ツールが必要となる。これらのツール間の結合を有機的なものとし，ツール群を一つの体系にまとめるには，三つの側面からの結合が必要である。

(a)　マンマシンインタフェースの結合

利用者にとって，多数のツールの対話インタフェースの統一は重要である。このためワークベンチでは，テキスト，表，図形，画像の各情報を編集するための統一的エディタ(汎用化技術)が必要となる。このエディタでは，対象がソフトウェア開発に関連する情報であることを踏まえて，言語や図面の文法を内蔵化したもの(構造エディタ)であることが必要である(**図3**に構造エディタの例を示す)。

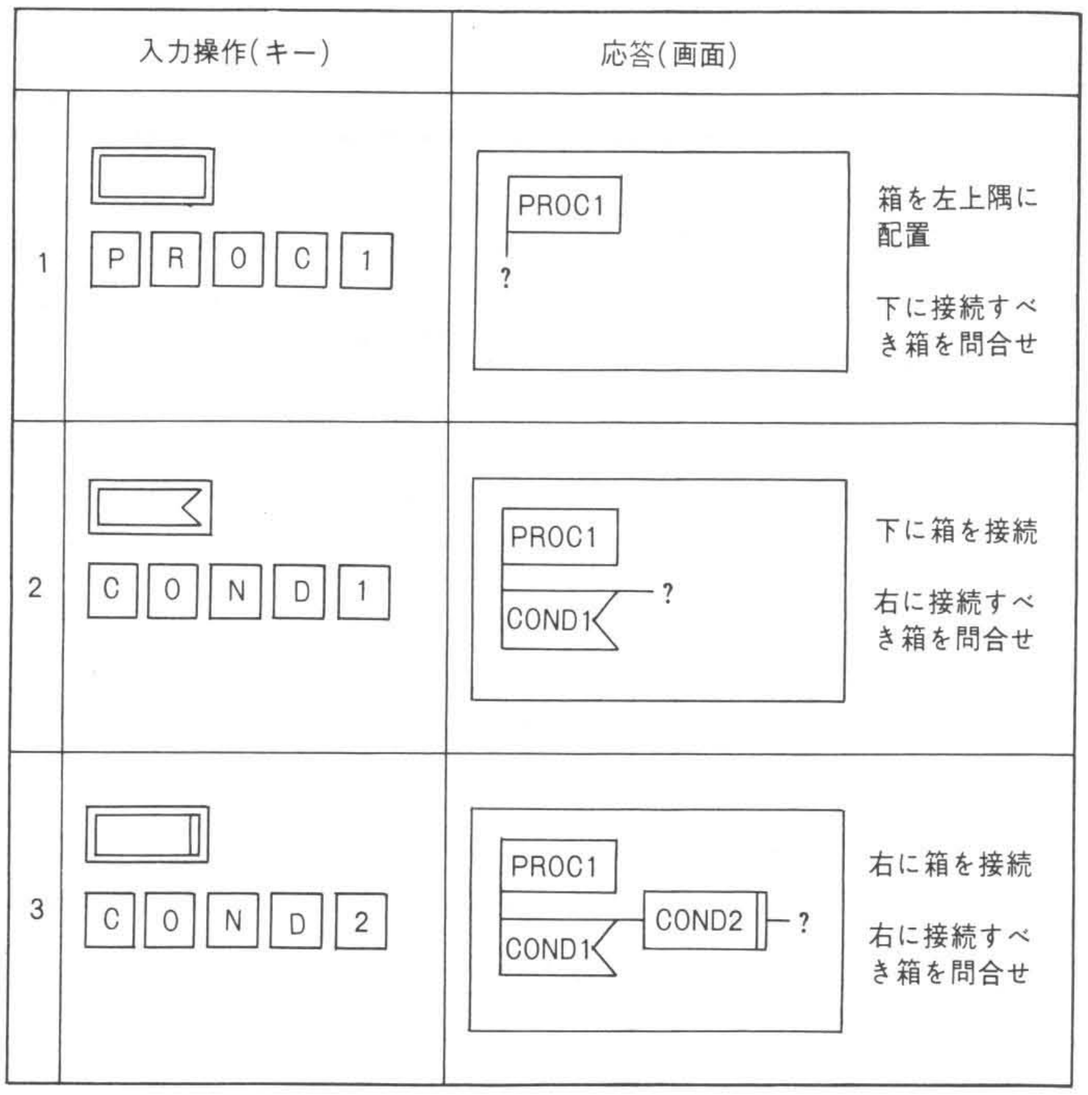

図3　構造エディタでの対話例　図形の種類とその接続規則，文字と図形との位置関係など作図文法が計算機に内蔵され，それに基づき配置などが自動的に定められる。

コマンドを統一する規格化技術も重要である。規格化の一例として，**図4**に動詞と目的語の分類に着目した統一コマンドの設計例を示す。

これらの技術のほかに，ビットマップ，マルチウインドウ，仮想画面，メニュー制御などの諸技術も，親和性の高いインタフェースの構築には必須である。

(b)　入出力結合

一つのツールの出力から別のツールの入力情報を生成す

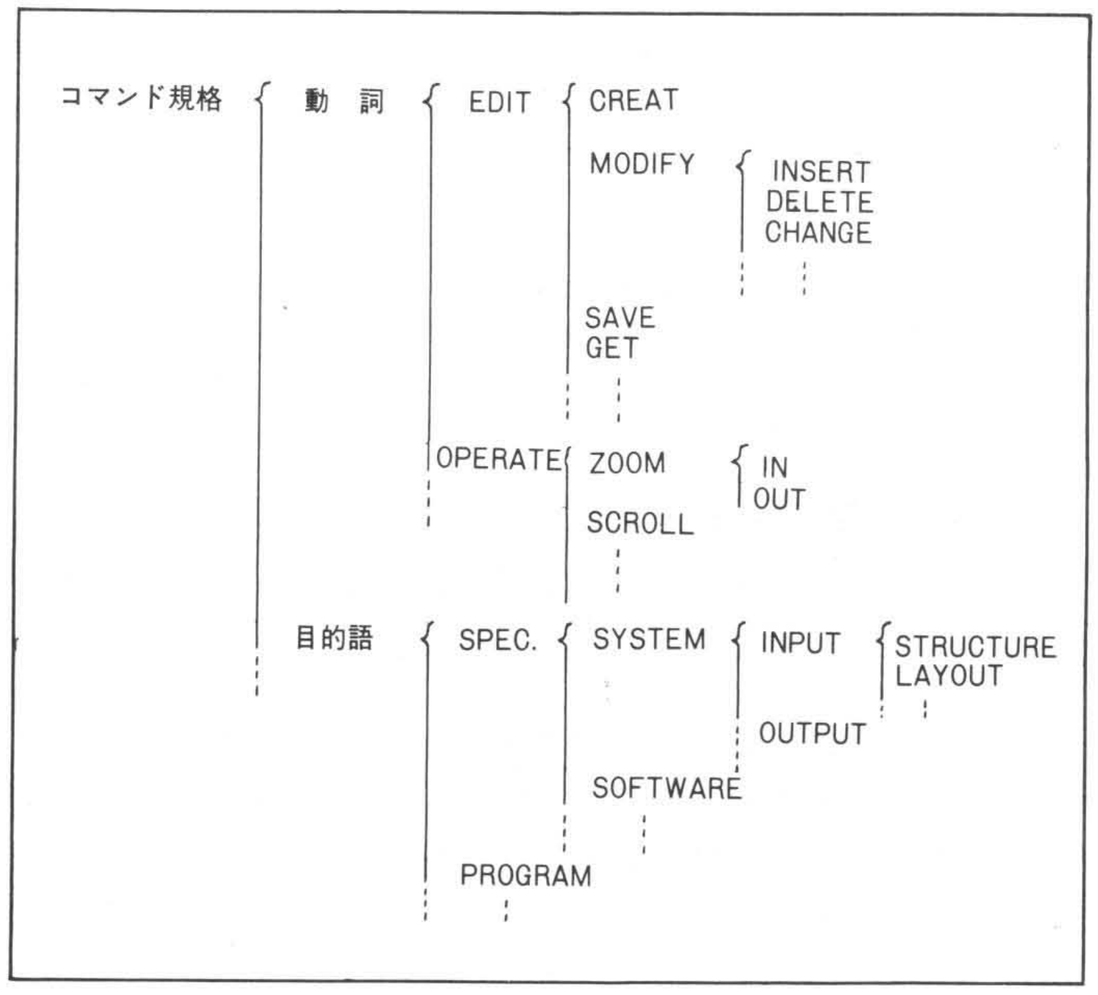

図4　コマンド規格の例　動詞コマンドと名詞コマンドを組み合わせて指定する。各々のコマンドについては，名称と体系があらかじめ定められている。すべてのツールは，このコマンド体系に従った対話機能をもたせられる。

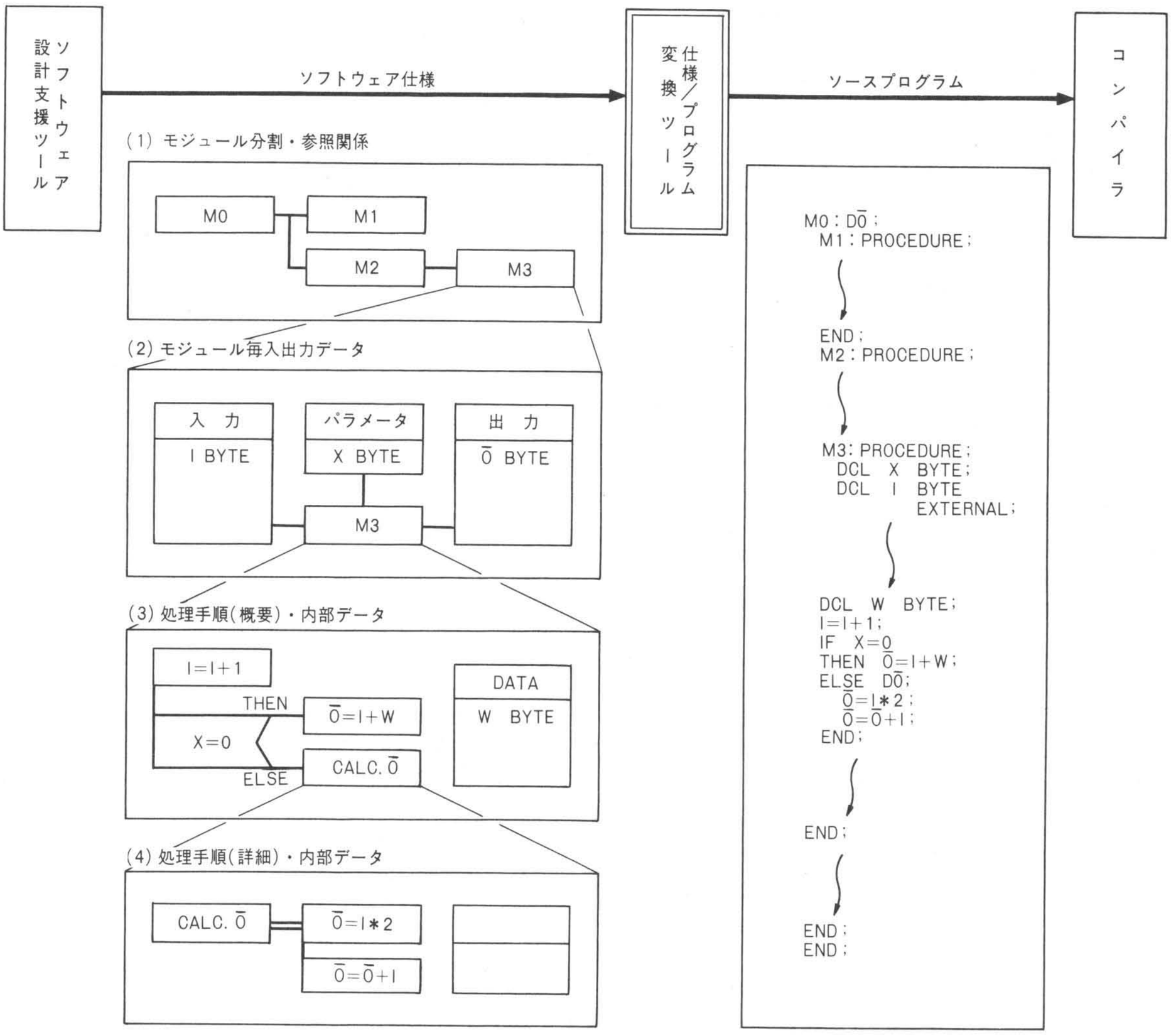

図5 入出力結合の事例 ソフトウェア設計ツールの出力である仕様情報から，プログラミングツール(コンパイラ)への入力であるソースコードを自動生成するブリッジツールの例である。

る自動化技術は重要である。この技術は究極的には自動プログラミング技術である。この自動生産技術の基礎的実施例を**図5**に示す。

(c) データ結合

ソフトウェア開発のライフサイクルでは，諸小工程間で共通な仕様情報をもつことが少なくない。例えば，システム設計での入力設計の結果は，ソフトウェア設計でのモジュール仕様の一部に流用され，プログラミングでのデータ宣言部に反映される。このような工程横断的な情報を，一元管理するためのデータベース構築技術が必要である。

(2) 分散制御のための技術

第一に，分散化されたプロセッサやメモリなどのリソースを有効活用することが必要である。このため，地理的に離れた場所にある空リソースを利用するための仕掛けが必要となる。これは分散形のオペレーティングシステム技術であるといえる。

第二に，分散したプロセッサ間の情報交信技術の開発が必要である。ワークベンチでは，プログラム，仕様情報，ツール，変更情報の交信が行なわれる。これを行なうための通信プロトコルの開発が重要である。

最後に，分散した情報(プログラム，仕様，ツール)の変更履歴管理，変更作業管理，修正波及解析を行なうための技術開発が必要である。この技術は，構成管理(**Configuration Management**)と呼ばれる。

6 結 言

ソフトウェア開発での統合支援環境としてのワークベンチは，ソフトウェアファクトリーの中核であり，その価値は極めて重大である。ワークベンチの基盤技術のうち，有機的結合技術については，ほぼ開発が終わっている。今後は，OA分野で活発化しているパーソナルコンピュータとLAN(ローカルエリアネットワーク)の開発を受けて，分散化技術の開発が焦点となろう。

分散化の実現に当たっては，知識工学の応用，再利用，プロトタイピング，専用マシン化などが併せて課題となる。この意味でワークベンチは，技術のクロスオーバーが不可欠であり，システム技術が特に必要とされるテーマである。

参考文献

1) Wasserman A. I.: Tutorial: Software Development Environments, IEEE Computer Society (1981)
2) Kobayashi M. et al.: ICAS: An Integrated Computer-Aided Software Engineering System, digest of papers COMPCON '83, 238～244 (1983)
3) Ivie E. I.: The Programmer's Workbench-A Method for Software Development, Commun. ACM, **20**, 10, 746～753 (Oct. 1977)