ICOT Technical Report: TR-051

TR-051

ソフトウェア開発支援システム

杉 本 正 勝
（富士通）

1984. 3

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191 ~ 5
Telex ICOT J32964

Institute for New Generation Computer Technology

ソフトウェア開発支援システム

杉本正勝　〔富士通（株）〕

1. はじめに

μプロセッサや256Kbit RAMを始めとするハードウェアの進歩による，ハードウェアの低価格化にともなって，パーソナルコンピュータを始めとする安価なハードウェアの大量供給により，ソフトウェアの利用ユーザ層が飛躍的に拡大してきた。また，ハードウェアとソフトウェアの融合型の新商品の可能性に対する期待も大きい。

現状の主要な問題点の一つは，「実現したい機能」と，ハードウェアとの間とをつなぐソフトウェアの開発・保守作業に，論理的な冷徹さと，極端に詳細で正確性を要する人間的でない作業を必要とする点である。優秀な人材を大人数必要とし，かつ保守作業にその人材がかかり切りにならざるを得ないことである。

第5世代コンピュータにおいては，人間の自然な思考レベルに合ったソフトウェアの開発が行えるように，ソフトウェアの設計用ツールや，プログラミング言語等を用意して行くことによって，従来の問題を根本的に解決することを目標としている。

ここでは，今後のコンピュータのソフトウェア開発支援システムに対する考え方やシステムの持つべき機能やシステムの実現方法等を検討する〔Scherlis 83〕，〔Novak 83〕，〔Nikkei 83〕。

2. ソフトウェアの開発・保守作業の検討

ソフトウェアを設計・作成する作業は，疑いもなく人間の問題解決作業の一つである。それ故に，その主役を演じる「人間」の能力をうまく引き出せるような作業環境が必要である。これなくして，ソフトウェア開発・保守作業における創造的な活動は不可能である。

ここで，「人間が作業をするモデル」を検討する。図1に示すように，「人間」の頭脳におけるメモリは，STM（Short Term Memory，短期メモリ）とLTM（Long Term Memory，長期メモリ）とに大別される。人間の思考に直接関するのは，STMである。STMは，その記憶能力が7±2個の情報の固まりと言われているように，記憶容量が小さい。また，記憶の保持時間も分の単位どまりと言われている〔Miller R 82〕。

LTMは，記憶容量は大きいがアクセス時間が，数10mS程度と大きい。そこで，ソフトウェアの設計内容を，「頭の中にいれて」考えると言う形態ではうまくない。

そこで，人間はEM（External Memory，外部メモリ）を使用する。ソフトウェアの開発の場合は，EMにあたる部分は「ドキュメント」に対応する。

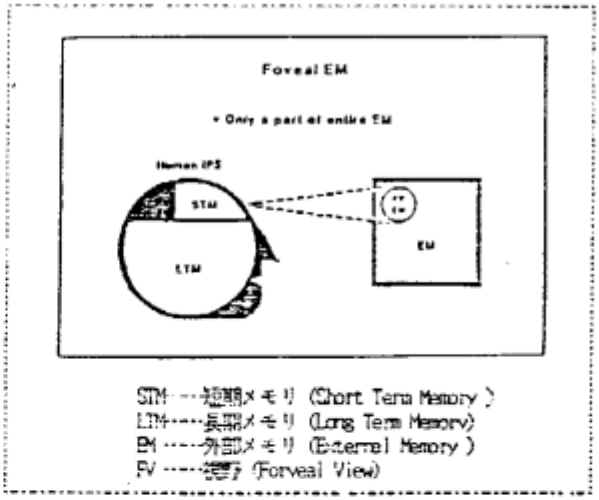


図1 思考活動のモデル

設計者は設計ドキュメントを書き，これを利用してきた。現在のソフトウェア工学のもとでも，各種仕様書をドキュメントとして，きちんと残して行く方法がとられている。

ここで，注目したいのは，「人間がものを

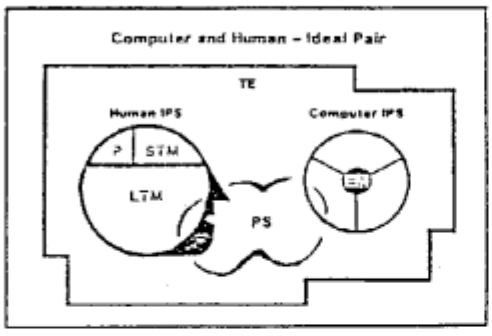
考える」のに関係したEMの量である。実は，人間の視野にある情報だけなのである。この意味で，いかに厖大なドキュメントを書いても，その記述方法が不適当であれば，ドキュメントはうまく使用されないことになってしまう。

それ故，分かり易いドキュメントが必須となる。

ドキュメントは，現在ではワードプロセッサで作成したり，コンピュータの日本語処理システムで作成している。そして，プログラムもドキュメントの一つであると考えられている。今後の動向として，多くのドキュメントをコンピュータの中に格納し，検索して利用する（オンラインドキュメント）ことは確かである。

そこで，今後はコンピュータの中にあるドキュメントが「思考」の対象となる（図2）。この場合も，思考に直接関係するのは，視野にあたる情報だけであるので，ディスプレイの一画面や，タイプされた用紙の一枚程度である。

また，従来から言われているドキュメントの他に，プログラムの実行の結果の表示や，プロンプト・メッセージやエラー・メッセージなどの，コンピュータの出力も広義のドキュメントとみなせる。



TE……思考環境 (Task Environment)
PS……問題空間 (Problem Space)
P ……プロセス
IPS……情報処理(Information Processing System)

図2 オンライン・ドキュメントを用いる思考

ソフトウェア開発支援システムを利用したり，コンピュータの持っている知識を利用して判断するような，「人間とコンピュータとの協調システム」では，人間に分かり易いドキュメントや表現形式が必須となる。

従来から，高級プログラミング言語の発展の過程のなかで，「ソフトウェアの生産性の一つの評価基準であった，プログラムの開発行数（ステップ数）／人月は，アセンブラ，高水準プログラミング言語の進歩のなかで，ほぼ一定」で，記述レベルが向上した分だけ生産性が向上してきたということは，やはり，人間の情報処理能力と直接関係を持った現象であると推測できる。

コンピュータに格納されている情報の分かり易い表現方法を開発し，利用して行かない限り，コンピュータは「巨大なブラック・ボックス」と化してしまうだろう〔Kanda Y. 80〕。

3. ソフトウェアの開発用のドキュメントと仕様書

ソフトウェア開発用のドキュメントの主要な部分は，仕様書と呼ばれる。仕様書にはその記述レベルに応じて，各種のものがあり，同じレベルの記述に対しても，いくつかの名がある。ここでは，単純化して，要求仕様書，機能仕様書，詳細仕様書とプログラム（リスト）を考える。

要求仕様書は，これから設計し，作成し，保守して行くソフトウェアの概要規模，性能要求等を記述し，要求項目のなかでの矛盾の検出を行う。この仕様書は，ソフトウェアの発注者と受注者の間のインターフェースとなる。現状では，この仕様書は人と人とのコミュニケーションに利用されている。記述内容の矛盾の検出や図形表示のための変換処理を行うシステム（例えば，ISDOS，SREM）もある。

機能仕様書は要求仕様書にもとずき，機能を整理したものである。これは，一段下位の設計を行うためのものであり，設計を詳細化する上で必要となる。機能設計証書を記述する過程で，機能の大きなまとまりが分ってくる。論理的な処理のまとまりや，論理的な処

理のまとまり，データのまとまりも，この設計過程で明らかになってくる。この仕様書を作成する過程で，機能の確認，矛盾の検出を行う。

機能仕様書も，現状では，主に人と人とのコミュニケーションに利用されており，人間にとって分かり易い記述形式が重要となってくる。

詳細設計仕様書は従来は自然言語としての日本語で，処理概要を書いたり，ボックスのなかに処理の内容を日本語で書いて図形表示していた（HIPO，FESDD，YAC，HCP，PAD，フローチャート等）。

この仕様書は，次の段階であるプログラミング作業のための仕様書となるとともに，設計内容を他の人に伝えるためのものである。

ここで，一つの見解として，「第５世代コンピュータの被言語によるプログラム作りでは，被言語の記述レベルが高いので，詳細設計仕様は不要である」がある。プログラミング言語の高級化の観点から，このようなことができれば，大きな進展であると言える。しかし，現状ではやはり被言語のプログラムに仕様補助情報（設計者の設計意図についてのコメント等）を追加しないと，詳細設計仕様書と同等とはなりにくい。

4. 詳細設計仕様書とプログラム

詳細設計仕様書は，その記述レベルの高低はあったが，ともかく数10年に渡って使用されてきた。そして，大体において，自然言語表現が使用されてきており，この記述を機械的に処理しようという試みもほとんどなかったので変化もなかった。

一方，プログラミング言語は，長い間，期待されていた処理レベルの高レベル化が最近盛んに研究されるようになってきており，変化の大きい時期であり，詳細設計仕様書とプログラムとの相互関係を再検討する時期である。

「プログラムの理解」と仕様書

被言語で書かれたプログラムを「理解する」処理がある。例えば，被言語で書かれたプログラムをデバッグしたり，他の人が作成したプログラムを分かろうとする場合である。

また，既に作成されたプログラムをシステム側で解析して，プログラムの内容について新しい事実を推論して求めるのも「プログラムの理解」の一形態である。また，従来からソフトウェア工学の理論面で注目されていた「プログラムの検証」もこの分類に入れることができる。

前者が，「人間」が理解する作業であるのに対し，後者は「システム」が理解するのである。

人間が理解し易い表現

人間の立場に立ったプログラムや仕様書の表現を考察するには，まず「人間」が分かるということはどういうことであるか，また「確かであると思う」とはどういうことであるかという，人間の能力についての知見が必須となる。

「別事してある中に入っていることが分かる」，「特に，指定されていないことが分かる」，「同じであることが分かる」，「差があることが分かる」等の基本操作の積み重ね推論処理等で，人間の論理判断処理がされると考えられる。

プログラムの内容を「人間」が理解するのを支援するためのツール

既に書かれたプログラムがあって，この内容を人間が理解しようとするときに，必要な補助情報を与えてくれたり，理解し易い表現に等価変換してくれるツールも必要になってくる。このようなツールの機能の検討を通じて，「人間」とか「設計」や「理解」ということが明らかになってくると予想される（図３）。

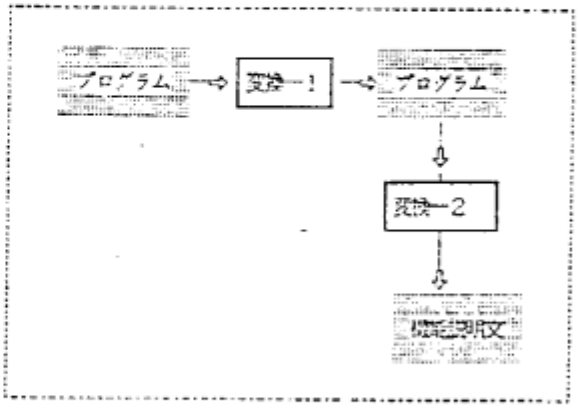


図３ 「プログラムの理解」

図3において，変換-1の処理は使用者の記述した核言語プログラムを，核言語プログラムの仕様の範囲内で等価変換し，より「分かり易い」プログラム形式に変換する。

変換-2は，必要な補助情報を抽出したり，プログラムの理解を容易にするような自然言語の文章に変換したりする処理である。またこのような表現から更に，抽象化された機能仕様を引き出すことも，非常に長期的であるが将来解決される課題と考えられている。

「プログラムの理解」は，あくまで「プログラム」が先にあるという立場であるが，実際には設計の立場からは，人間側に立った仕様書がまず必要で，これが自動的にプログラムとして動作し，速度等を含めて要求仕様を満たしていれば都合がよい。

言い換えると，「分かり易さ」や「プログラミングのし易さ」強調のプログラミングをすればよく，「速度」強調のプログラムは自動変換される方向をめざすことである。この方向は，従来のプログラミング言語の発展の方向と同一線上になることは間違いない。

また，要求仕様レベルや機能仕様レベルから，直接的にプログラムとして実行可能なようにすることも，長期的に解決すべき重要な課題の一つとして認識されている〔Ohno 84〕。

ここに於いて，課題は核言語に対して，より一歩高いレベルの「ソフトウェア記述言語」を見出すことである。この言語を，仕様（記述）言語と名づける。この言語を定めるには，次のような方法が考えられる。

(1) 自然言語の仕様書記述とプログラムとの関係を明確にする
(2) 述語論理による仕様書記述の適用範囲を明らかにする
(3) 核言語のプログラミング経験からプログラム手法を明らかにする

自然言語による仕様書と，述語論理との関係は一階述語論理とその拡張の観点から研究されている。

特に，知識ベースの問合わせについては，ユーザが自然言語で書いた要求仕様または，機能仕様書レベルから，プログラムへの自動変換への道が拓けており，この技術のより一般的な適用が有力であると考えられる〔Kobayashi 84〕。

以上が，ソフトウェア開発，保守作業に必要なドキュメントについての基本的な考え方である。そして，このようなドキュメントを使用しながら，設計作業をうまく進める行ける支援システムが，ソフトウェア開発支援システムである。

設計者が自分の思う通りに，「創造的」な環境でソフトウェアを開発するには，人間とシステムとの協調作業が必要であり，この協調作業をサポートするシステムが必須となってくる。

このシステムを実現するためには次の課題を解決する必要がある。

(1) 人間とシステムとが対話するために，システム側が持つべき知識の構造
（単に，データの格納場所としてだけでなく，人間との対話の観点から）
(2) 人間の質問に対するシステム側の応答が，適当な詳細レベルで，整理された形式となる
(3) 人間とシステムの対話による「設計のプロセス」の追求

以上のように，ソフトウェア開発支援システムの課題は多く，その必要とする研究のレベルも深いので，すべての課題に対して同時に着手するのは，人的リソースの面からも考えにくい。そこで，長期的な研究開発の道筋の中で，ステップを分けて研究を進めるのがよいと考えられる。

具体的には，自然言語によるソフトウェアの機能記述文章と，対応するプログラムとの相対関係を研究して行き，将来の半自動プログラム合成や自動プログラム合成の研究用のデータを収集する。この目的で，核言語プログラムから機能説明文を自動生成するツールを作成し，このツールを利用しつつ多数のデータを収集し，整理するアプローチをとる。

この機能説明文の自動生成ツールは，単にデータの生成用としてだけではなく，ソフトウェア開発のツールとしても利用できる。その後，機能設計・詳細化支援や要求仕様作成支援システムの構築をめざす。

5. ソフトウェア開発支援システム

ソフトウェア開発支援システムは，上記の検討をもとに，次のような目標を達成するシステムである。

(1) 設計ドキュメントをもとに一貫して，設計・作成・保守ができるようにする。

(2) 設計ドキュメントをもとにプログラムを自動的または半自動的に合成し利用できるようにする。

ソフトウェアの設計作業において，設計者は実現していると考えているソフトウェアの仕様を次々に記述して行く過程で，次のような作業を行っている。

(1) オブジェクトの設計
取り扱うオブジェクトの名称づけ，オブジェクトの構成等を記述する〔Novak 83〕

(2) オブジェクト間で成立する規則の記述
オブジェクト間の関係や制約条件，省略条件等を記述する

(3) 説明のためのコメント
設計者自身または他の設計者が設計内容を理解し易いように，コメントを記述する

ソフトウェア設計支援システムでは，対話型で上記の作業を支援することが必要であるとともに，次の3点を実現することも重要となる。

(1) モジュール構造の記述
オブジェクトの単位性と局所性を制御する目的でモジュール構造を取り扱えること〔Furukawa 83〕

(2) 段階的詳細化
人間の思考過程の性質上，部分的に詳細化する作業が行われるが，これらをサポートする機能を持つこと〔Sakai 83〕

(3) 設計コンストラクト
設計者が実現したい機能を記述して行く上での，枠組みであるコンストラクト（Construct）を用意する。従来から，構造化設計のコンストラクトやデータフロー型設計のコンストラクトがあるが，これらに追加して，論理プログラミング型コンストラクトや集合処理用コンストラクトに重点を置く必要があろう。

ソフトウェア開発支援システムでは，次の様な考え方でソフトウェアの設計・作成の支援システムを実現する方向である。

要求仕様，機能設計仕様，詳細設計仕様には基本的に自然言語の文章を利用する。また「機能概念」を利用して，設計仕様書に書かれた機能の対応をとることで，概要記述と詳細記述の対応の追跡性を確保する。

更に，被言語プログラムに対する機能説明モジュールを検討する。このモジュールは被言語プログラムの論理を自然言語で説明する機能を持つ。このモジュールを利用することで，自然言語とプログラミング言語との対応関係のデータを収集できるとともに，被言語プログラムを使用する利用者がドキュメント・ツールとして利用することができる。

機能概要

ソフトウェア開発支援システムの機能の概要を次に示す。

(1) ソフトウェア半自動合成モジュール
ソフトウェア半自動モジュールは，日本語や英語で記述されたソフトウェアの仕様から被言語のプログラムを半自動的に生成する。仕様は自然言語で記述されるものであり，自然言語の持つ意味のあいまいさや，システムに格納されているプログラムの不足等の理由で，プログラムが合成できない場合があるので，ソフトウェアの設計者側で検証を行い，結果によっては，プログラムの変更を行う。合成されるプログラムは被言語とする〔Yasukawa 83〕，〔Abbott 83〕，〔Enomoto 84〕。

(2) 機能検索モジュール
この機能検索モジュールは自然言語で書かれた機能記述のモジュール仕様を検索する。全く等しい機能を持つモジュールを検索するのがこのモジュールの機能ではなく，機能がほぼ等しいモジュールの集合を検索するのが目的である。この目的で「機能概念」を用いる機能検索を実現する。この機能はソフトウェア半自動合成モジュールから呼び出される場合と，ソフトウェア設計者が直接に，この機能を利用することを想定している。また，オブジェクト（データ構造）の検索機能も必要となろう。

(3) 機能説明モジュール

機能説明モジュールは被言語プログラムから，そのプログラムの論理を説明する機能を持つ。プログラムと機能説明文との対応関係のとり方には各種のものが考えられるが，マンマシンインターフェスの良さや，オブジェクトの説明や，機能検索とも関連づけできるものを採用する。

モジュール構成

ソフトウェア開発利用コンサルテーションシステムのモジュール構成を図4に示す。本システムの利用者は，設計中のモジュールの仕様書をモジュール仕様ライブラリに格納しつつ，ソフトウェアの開発を進めて行く。また，このモジュール仕様ライブラリを介して，段階的詳細化（Stepwise refinement）を行う。

半自動合成モジュールは，このモジュール仕様にもとづいて被言語のプログラムを半自動合成して行く。

半自動合成されたモジュールは，プログラムライブラリに格納される。利用者は，このプログラムを検査した後，必要に応じて手直しする。

(1) 機能記述支援モジュール

機能記述支援モジュールは，自然言語による仕様記述を支援するエディタ機能を持つ。また，段階的な詳細化の支援を行う。

(2) 機能記述文の解析モジュール

機能記述文の解析モジュールは，自然言語による仕様記述の文章を解析してコンピュータの内部表現に変換する。

(3) プログラム合成処理モジュール

プログラム合成処理モジュールは，機能記述文の解析モジュールが解析した仕様記述文から，モジュール仕様ライブラリを検索してプログラムを合成する。

機能検索モジュールは，モジュール仕様ライブラリに格納されているモジュールから，利用者の実現したい機能を持ったモジュールまたは実現したい機能に近い機能を持ったモジュールを検索する。

(1) 機能概念による検索モジュール

機能概念による検索モジュールは，実現したい機能をもとにした「機能概念」を介して，モジュール仕様ライブラリを検索する。

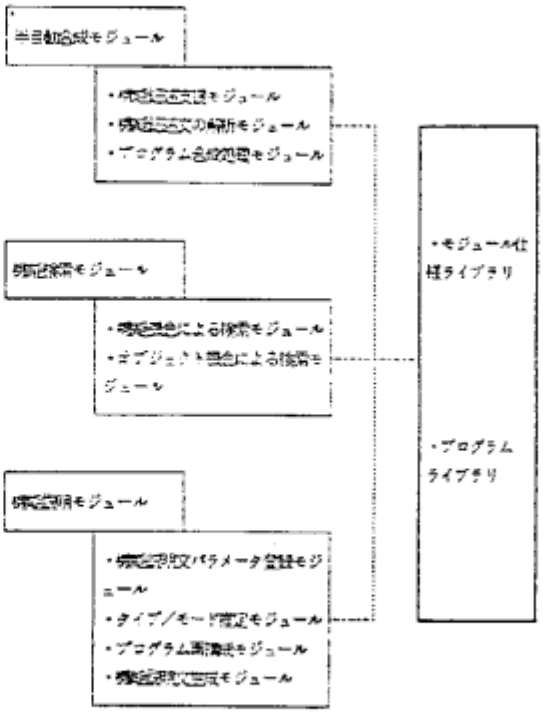


図4 ソフトウェア開発支援システムのモジュール構成図

(2) オブジェクト概念による検索モジュール

オブジェクト概念による検索モジュールは，データ構造について既成のモジュールライブラリを検索する。この処理には，「オブジェクト概念」を利用する。

機能説明モジュールは，プログラムライブラリに格納されている被言語プログラムに対して，プログラムの論理を自然言語で説明する機能を持つ。

(1) 機能説明文パラメータ登録モジュール

機能説明文パラメータ登録モジュールは，機能説明文の核となる文章の形式や変数パラメータ，変数パラメータのデータタイプ等を登録する。

(2) タイプ／モード推定モジュール

タイプ／モード推定モジュールは，被言語のプログラムを解析して述語のパラメータのタイプや入出力モードを決定する。

(3) プログラム再構成モジュール

プログラム再構成モジュールは，被言語のプログラムを解析して，述語の再配置を行う。

(4) 機能説明文生成モジュール

機能説明文生成モジュールは，被言語のプログラムを解析して，各述語に対してそのプログラムの説明文を生成する。

5. おわりに

従来のソフトウェア開発・保守作業の問題点の検討を通じて，今後のソフトウェア開発支援システムの実現の方向について検討を行った。そして，ソフトウェア半自動合成モジュールと機能検索モジュール，機能説明モジュールを主要なモジュールとするソフトウェア開発支援システムを論じた。

今後は，ソフトウェア開発支援システムの実現をめざし検討を深める予定である。

本研究は第５世代コンピュータ・プロジェクトの一環として行われた。本研究の機会を与えて下さった，新世代コンピュータ技術開発機構の方々に深謝いたします。

[文献リスト]

[Scherlis 83] Scherlis W.L. and Scott D.S., "First Steps Towards Inferential Programming", Proc. Information Processing 83, IFIP, 1983

[Novak 83] Novak G.S. Ohsuga S., et.al., "Panel Session PM2 : Knowledge-Based Systems", Proc. 6th International Conference on Software Engineering", 1982, The Information Processing of Japan

[Nikkei 83] 「ソフトウェア要求仕様の段階に使われだすプロトタイピング」日経エレクトロニクス, 1983.8.8

[Kanza Y. 81] 神田, 杉本, 沢井「エンドユーザのための自然語によるプログラミング」情報処理学会誌, Vol.21, No.3, 1980

[Ohno 84] 大野, 阿草「ソフトウェア資産の自動化技術」, 電子通信学会誌, Vol. 67,No.1, Jan., 1984

[Kobayashi 84] 小林, 溝口等「データベース論理設計支援エキスパートシステム」情報処理学会, 記号工学と人工知能23-2, 1984.1.25

[Novak 83] Gordon S. Novak Jr. "GLISP : A Lisp-based Programming System with Data Abstraction" The AI Magazine, Fall, 1983

[Furukawa 83] Furukawa K., Nakajima R., Yonezawa A. "Modularization in Logic Programming, ICOT TR-022,August,1983

[Sakai 82] 酒井, 淵水「プログラム記憶構造の生成, 処理, 文書能力を有するテキスト・エディタ」情報処理学会論文誌, Vol., No. 5, Sept., 1982

[Sai 82] 斎, 藤田, 西田「ライブラリモジュールを用いた仕様の詳細化によるプログラムの生成」, 情報処理学会, 第25回全国大会, 1982

[Yasukawa 83] H.Yasukawa "LFG in Prolog -Toward a formal system for represen- ting grammetical relations", ICOT TR-019,August,1983

[Abbott 83] Abbott R.J. "Program Design by Informal English Descriptions", Communications of the ACM, Nov.,1983,Vol.26,No.11

[Enomoto H 84] 榎本, 米崎, 佐伯「ソフトウェア開発システム（TELL）における自然言語による使用記述言語（NSL）とその応用例」, 情報処理学会, ソフトウェア工学研究会資料, 34-14, 1984

[Miller R 82] Miller R. "Human Problem Solving and its Relationship to Computer Aided Engineering Systems", Boeing Commercial Airplane Company, 1982