ソフトウェア工学 64－19
（１９８９．２．３）

ソフトウェア意図伝達支援ツール『ＣＯＭＩＣＳ』

岸本　美江＊　　西田　正吾＊　　後藤　卯一郎＊＊

三菱電機(株)　＊中央研究所　＊＊制御製作所

ソフトウェア開発プロジェクトの生産効率向上は重要な課題である。本研究は、組織的な作業とはメンバー間の相互作用プロセスであるという立場から、開発プロジェクト内のコミュニケーションをサポートするツールを開発した。開発プロジェクトにインタビュー調査を行なったところ、ソフトウェアの意図を表現する適切な方法がないため、コミュニケーションが困難であった。そこで、ソフトウェアの意図やイメージを表現する方法として、劇場モデルを提案する。劇場では、俳優や大道具が舞台の上で劇を演じる。観客は、劇の流れの中から個々の俳優の役割や脚本家の意図を読み取ることができる。ソフトウェアの構造は、劇に類似している。システム設計者の意図は、個々のモジュールやデータではなく、それらの相互作用プロセスの中にある。ＣＯＭＩＣＳは、モジュールやデータがシステムを実現していくプロセスを、ダイナミックに表現する舞台を提供するツールである。

COMICS
COmputer-based Media for Intention Communication in Software

Mie KISHIMOTO　Shogo NISHIDA　Uichirou GOTOU
MITSUBISHI ELECTRIC CORP.
Amagasaki, Hyogo, 661 Japan

Abstract : This paper deals with a supporting tool for communication in software development process , called COMICS. It is difficult to express intention of software and a software project has some communication problems. We propose a method to represent the ambiguous image of software and intention of software designer based on the theater model. Theater shows sequences of the integrated scene composed of actors, setting, etc. on the stage and the audience can understand its scenario and intention. We believe this property of human cognition can be used for intention communication in software development. COMICS is designed on the basis of the theater model. COMICS may be used as a conceptual design tool for system designer , an explanation tool in software development team , and an intention communication tool from development team to maintenance team.

1．はじめに

ソフトウェア需要の急増に伴い、ソフトウェア開発の生産性向上が重要課題になってきた。今やソフトウェアは大規模になり、個人で開発することが困難になってきたため、プロジェクトで開発される場合が多い。組織的な開発を支援するために、標準化やソフトウェアの再利用、ツールの統合化などが研究されている。本研究では、協同作業をメンバー同志の相互作用プロセスと捉える立場から、プロジェクトメンバーの相互作用を支援する。

2．従来の研究

従来、ソフトウェアの生産効率を向上させるために、以下のような研究が行なわれてきた[1]。

1）技術力の向上：ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ技法やﾂｰﾙの開発
2）ソフトウェアの標準化：ソフトウェアの部品化、仕様記述言語の開発
3）プロジェクトの支援：工程管理、レビュー

1）の研究は、プログラマー人当たりにおける生産効率を上げるためのものである。2）は、標準化により個人差・能力差をなくし、ソフトウェアの伝達を支援すると共に、大量生産を可能にするためのものである。3）は、マネージメントの立場から、プロジェクトの運営・管理を支援するものである。

2）・3）は、組織的な開発を支援する技術と言える。標準化は、経験や能力が異なるメンバーが集まって一つのシステムを開発する場合には不可欠である。また、開発後も保守要員にシステムの内容を伝達するために必要である。さらに、標準化により、ソフトウェアが部品化でき、分割生産や再利用が可能になる。しかし、標準化にも問題が残る。一つは、ソフトウェアには完全に標準化しきれない部分があること。もう一つは、標準化された仕様書は大量で読みにくいことである。

一方、プロジェクト管理も重要な技術である[2]。多くのプロジェクトはツリー構造になっている。工程管理や各メンバーの作業内容の調整は、管理者の仕事である。各メンバーの作業が矛盾なく進むように定期的なレビューも行なわれる。このような管理方法は、ソフトウェアに限らず様々なプロジェクトで行なわれているが、ソフトウェアの場合は、特に作業進捗の管理やレビューが困難である。それは、ソフトウェアの適切な表現方法がなく、意図を伝達することが難しいためである。

本研究では、協同作業はメンバー間の相互作用プロセスである、という新しい視点[3]に立っている。相互作用はコミュニケーションによって成り立っている。ソフトウェアの標準化もレビューも、コミュニケーションをスムースにするための手段である。運営・管理も、管理者と担当者とのコミュニケーションである。このように考えたとき、プロジェクトの生産性向上には、コミュニケーションを支援することが効果的なのではないかと予測できる。しかし、どのようなコミュニケーションをどのように支援すればいいのであろうか。

そこで、実際のソフトウェア開発プロジェクトに対してインタビューを実施し、メンバー間のコミュニケーションの実状と問題を調査した[4]。その結果を基に、コミュニケーションを支援するツールを開発した[5]。

3．ﾌﾟﾛｼﾞｪｸﾄにおけるｺﾐｭﾆｹｰｼｮﾝの問題

3．1　コミュニケーションの重要性

調査をする前に、ソフトウェア開発におけるコミュニケーションの重要性を考えてみる。

第一に、ソフトウェアは思考の産物である。複数で開発すれば、複数の思考で作ることになる。全体として統一の取れたシステムにするためには、各人の意図を統一し、矛盾のないものにしなければならない。

第二に、ソフトウェアは、構成要素が密接に関連しているため、単純に分割して開発するのは困難である。しかし、現実には巨大なシステムは、協同で開発せざるを得ない。協同開発する場合、構成要素が相互に関連しているのだから、その担当者も密接に連絡を取りながら作らねばならない。

第三に、一部を担当するだけのプログラマでも、全体を知らなければよいプログラムは書けない。単独のモジュールとしては最良のプログラムでも、全体にとって良いとは限らないからである。無駄に見えるような変数を用意しておいたり、バグが見つけやすいような構造にしておくと、モジュールとしては冗長でもシステム全体としては効率がよくなる。そのため、部分と全体との連絡をスムースにしておく必要がある。

このように、ソフトウェアのプロジェクト開発にとって、コミュニケーションは重要な役割を果たす。

3．2 プロジェクトにおけるコミュニケーション ――インタビュー調査より

では実際に、プロジェクトの中でコミュニケーションはどのように機能しているのであろうか。次の点を知るために、実働プロジェクトについてインタビューを行なった。

○メンバー相互のコミュニケーションの手段と内容
○そのコミュニケーションで十分に意図が伝達されているか
○各メンバーはどのようにシステムを理解しているか

取材したプロジェクトの構成を図1に示す。インタビューは、コーディングの途中から全体試験が終了するまでの期間にわたって6名に実施した。インタビューの内容は録音し、後で書き起こした。インタビューのほかに、週1回のミーティングに5回出席し、情報交換の様子を記録した。

インタビューの結果をまとめる。

1）プロジェクト内で観察されたコミュニケーション

プロジェクトにおけるコミュニケーションは、記録に残る公的なもの（ミーティング、レビュー）と記録に残らない私的なもの（メイル、会話）とがある。公的なコミュニケーションは、システム設計者同志・プログラマ同志・全体会議、という作業単位で行なわれ、報告・連絡・確認などの機能を持つ。私的なコミュニケーションは、所属や座席のまとまり単位で行なわれる。内容は様々で、システムについての質問・ソフトウェア全般に対する考え方・プログラミング技術の指導、作業方法の模倣など、意図の伝達のほかに教育的な要素もある。

2）開発プロセスの中で見られたコミュニケーションの問題

公的なコミュニケーションは、相互理解が得られていない場合が多い。説明や連絡など、一方的な伝達に終わる。相手が自分の意図どおりに理解しているかどうかを確認することはない。疑問点があれば質問するが、自分なりに解釈できればそれでコミュニケーションは終わってしまうので、意図が誤解されたままという危険がある。また、初心者は何となくわかったような気がして、質問点も思いつかないまま聞き流してしまう。しかし、実際には理解していないのかもしれない。

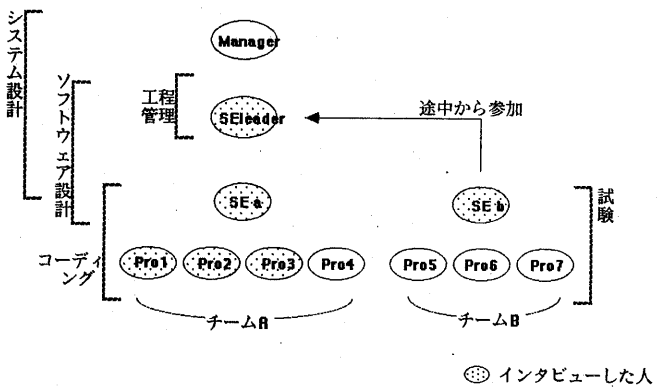

図 1 プロジェクトのメンバー構成

公的なコミュニケーションで伝達されなかった部分を、私的なコミュニケーションが補っている。しかし、前述したように私的なコミュニケーションは作業単位ではなく席や所属で行なわれる。プロジェクトの中でいくつかの私的コミュニケーショングループが生じてしまい、全体としてまとまりにくい。実際、所属によって経験や価値観が異なり、作業方針が異なる場合もある。

3）メンバーが抱いているシステムイメージ

各メンバーが、自分が開発しているシステムに対して持っているイメージはまちまちである。各々は、各自の視点からシステムを理解している。マネージャは、他のアプリケーションとの関係からシステムの機能や工程を考えている。システム設計者は、ソフトウェアの品質、効率という立場からシステムを考えている。プログラマは、自分の担当モジュールを中心にシステムを見ている。自分に関係のない部分は理解していなかったり、システムの具体的なイメージを持っていなかったりして、全体と自分との関連を把握していないことも多い。担当によって視点が異なるのは当然である。が、各自が自分の視点からコミュニケーションしようとすると、相手を誤解したり、理解できなくなる。また、全体の中で自分の位置を正しく捉えられなくなる危険もある。

3．3　コミュニケーションを妨げる要因

プロジェクトのコミュニケーションがうまく行かないのはなぜか。インタビューの記録から、コミュニケーションに関する発言を集め、分類してみた。各要因が絡み合って問題が生じている。

①視点が異なる

そのため、偏見や利害がからんで相手の立場に立てない、相手に対し、まちがったモデルをもっている。

②知識・経験が異なる

そのため、相手の話のレベルについていけない、相手がどの程度理解しているのかわからない。

③ソフトウェア表現が難しい

そのため、自分のイメージを言い表せない、自分の中でもイメージが曖昧だったりする。

プロジェクトには、所属の違うメンバーや経験のない新人プログラマが混ざることも少なくない。①や②のような問題は、ほとんどのプロジェクトに存在すると思われる。視点や能力が異なる者同志がコミュニケーションするときには、共通の表現方法が必要である。ここで、③のようにソフトウェアが抽象的であり、表現が難しいことが大きな問題になってくる。

従来のソフトウェアの表現方法には、

○フローチャート
○仕様書
○プログラムリスト
○ＨＣＰ図

などがある。これらは、個人差や誤解を防ぐために標準化され、詳細な内容を正確に伝達するには効果的である。しかし、コミュニケーションしにくいのは詳細な情報ではなく、曖昧なイメージである。仕様書にソフトウェアの意図を盛り込むことはできないというのは、多くのシステム設計者の意見である。経験の豊富なプログラマであれば、仕様書からシステム全体のイメージをふくらませることもできる。が、経験のないメンバーは、システム全体がどんな動きをするのかわからないとか、自分に直接関係ないところはわからないままプログラミングしている危険もある。その結果、プロジェクトの生産性低下、ソフトウェアの品質低下、などの問題が生じる。

プロジェクトにおけるコミュニケーションを支援するためには、仕様書で伝えきれないシステムイメージや意図の伝達を支援できる表現方法が必要である。ソフトウェアのイメージは、どのように表現できるだろうか。

4．意図の表現方法

4．1　人間の理解のメカニズム

適切な表現方法を考える前に、従来の方法でソフトウェアのイメージが伝わりにくい理由をまとめてみた。従来の表現方法には、

●量が多く、理解するのが困難である。
●詳細で正確な情報だが、全体のイメージがわかない。
●要素間の関連がわかりにくい。

などの問題がある。フローチャート・ＨＣＰ図・プログラムリストは、始めにシステムを理解する際には詳細すぎる。全体の概要を理解するためには、もう少しマクロに述べてあるものが必要である。普通は仕様書でシステムを説明する。

仕様書を見ると、機能別・構造別に記述されている。人間が機械などを理解するとき、構造と機能を見る。それは、機械がどうやって動くかという理屈を知りたいからである。例えば、エンジンが回転し、このギアが力を伝えて車輪が回るという仕組み（理由）と車が走るという機能（結果）が頭の中で一致すれば、車が納得できる。故障しても、どこが壊れているのかが推測できる。しかし、ソフトウェアの仕様書は、仕組みと機能を別々に述べている。プログラマには、自分の作ったモジュールとシステムの動きとの関係がピンと来ない。構造と機能の関係を結び付けるものとして、次の点が欠けているのではないかと考えられる。

●システムの動作が、具体的にわかる。
●部分の動きと全体の動きの関連がわかる。
●部分同志の関連がわかる。

これらの要因がイメージや意図の伝達に必要な理由を、人間の理解という視点から考え直してみた。人間の認識のメカニズムには、

○文脈の中で現象を捉えようとする
○まとまりを見ようとする

という特徴がある[6]。人間は、日常何かの現象を認識するとき、前後の流れや周囲の環境の中で理解する。個々の要素の意味を認識するより先に全体の意味を認識する傾向がある。例えば、文章にサッと目を通せば大体の意味を把握することができる。もし単語が一つ間違っていて、文法的に見れば意味が通らない場合でも、それに気がつかずに大要を理解してしまう。逆に一つ一つの意味にとらわれ過ぎると、全体が見えなくなる。外国語を丁寧に逐語訳していると、結局概要をつかめなかったりすることがある。

また、「走る」行為の説明として、「目的地になるべく早く到着するために急ぐ様子」と言われるより、「遅刻しそうなので走る」と言われたほうが頭の中でイメージがわいてくる。その行為について説明されるより、その周囲の状況・理由・目的を聞くほうがよくわかる。

ソフトウェアも同様である。個々のモジュールにばかり注目していると全体が捉えられなくなる。ソフトウェアの仕様書からイメージが描きにくいのは、個々の要素を詳細に正確に述べることに重点を置き、人間の認知メカニズムに適さないからだと考えられる。

４．２　劇場モデルの提案

ここで、劇場という表現方法を考えてみる。劇場は、抽象的なテーマを観客に伝える手段である。脚本家は、自分の意見や感動などのテーマを表現するために、シナリオを書き、俳優や背景・大道具を設定する。俳優や道具達は、舞台の上でストーリィを演じてドラマを構成する。脚本家の意図は、俳優の一人一人、大道具の一つ一つにあるのではなく、それらの相互作用プロセスの中にある。観客は、舞台の上のプロセスを見て脚本家の感動や意図を読み取っていく。登場人物は、その劇の中で位置づけられる。劇場は、脚本家と劇と観客から構成され、脚本家の意図が劇として表現され観客に伝達される『場』である。

著者は、ソフトウェアの構造と機能は劇と類似していると考える。システム設計者は、ある機能を実現するためにソフトウェアを設計する。システムが効果的に機能するようにフローチャートを考え、基本的なモジュールやデータを用意し、それを動かすハードウェアを設定する。ソフトウェアは、モジュール・データ・ハードウェアなどの相互作用プロセスによりシステムを実現していく。モジュールは、そのプログラムとしての良さよりも、全体の中でどのような役割を果たしているかが重要である。

ソフトウェアの意図も劇と同様に、個々の要素ではなくプロセス全体の中にある。観客が劇から脚本家の意図を読み取ることができるように、プログラマはソフトウェアからシステム設計者の意図を読み取ることができるはずである。しかし、従来はモジュールの相互作用を表現する手段がなかった（図２）。そこで、システム設計者の意図をドラマにしてプログラマに伝えられる舞台を提供する。舞台の上では、

○システム全体のイメージと具体的な動き
○全体の中で個々のモジュールの果たす役割
○モジュール同志の関係性

を表現したい。図３のように、モジュール達が相互作用できる場と、モジュール間の関係を表示する場所が必要である。プログラマはモジュールのドラマを見て、システムのイメージと個々のモジュールの役割を読み取る。そして、プロジェクトにおける自分の位置づけと役割を理解する。

５．劇場モデルの実現ツール　ＣＯＭＩＣＳ

５．１　ＣＯＭＩＣＳの機能

ソフトウェアにおける舞台を提供し、「システム設計者・モジュール達の演じるドラマ・プログラマ」という劇場空間を創りだすツールとしてＣＯＭＩＣＳを開発した。ＣＯＭＩＣＳの目的は、精確で詳細な仕様を伝えることではなく、システムの具体的なイメージを描き、システム設計者の意図を伝え、各メンバーが全体の中での自分の位置づけを理解することをサポートする。

本ツールは、次のような特徴を持つ。

１）ソフトウェアの構成要素が同期を取って動く様子を表現する。ストーリィは紙芝居のように一幕ずつ進行する。従来の説明が個々の要素を順に記述するものであったのに対し、劇場モデルでは要素を常に全体と関係性の中で捉える。

２）ＣＯＭＩＣＳを見るプロジェクトメンバーは、モジュールの相互作用プロセスから設計者の意図やシステムの動きを読み取る。また、視点を変えて、自分が担当するモジュールを中心にシステムを見ることもできる。

３）システム設計者がＣＯＭＩＣＳ上で試行錯誤しながら設計することができる。そのプロセスを記録し、再現できる。

ＣＯＭＩＣＳの画面は（図４）、モジュールなどの構成要素を一覧できるウィンドウと舞台スペースから成る。構成要素ウィンドウは、ソフトウェアモジュールやハードウェア、データ構造などを記述するスペースで、システム設計者が自由に設定できる。構成要素は、ウィンドウ上でＢＯＸとリンクにより表現する。ＢＯＸには詳しい説明をつけることができ、各々の機能を詳しく記述できる。

システムの動きはシナリオで表現する。シナリオは、時間の経過と共にどのＢＯＸがどのように動作するかを示したものである。シナリオを走らせるとＢＯＸが点滅し、ＢＯＸの動作やシステムの状態についての図が表示される。これが、電子紙芝居のように時間を追って変化していく(図５)。

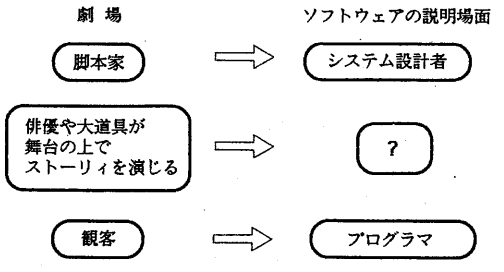

図　２　　劇場とソフトウェアの意図伝達

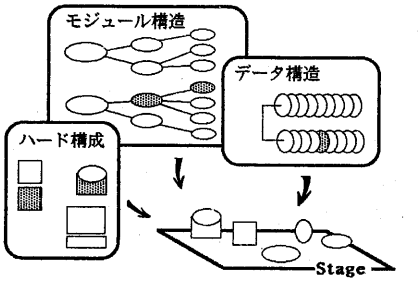

図　３　ソフトウェアが舞台の上で演技する

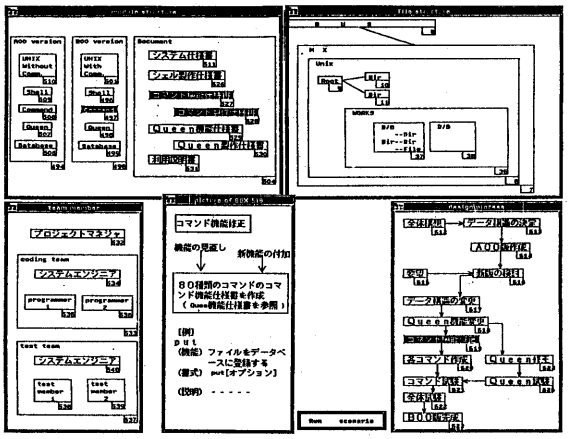

図　4　　ＣＯＭＩＣＳの画面例

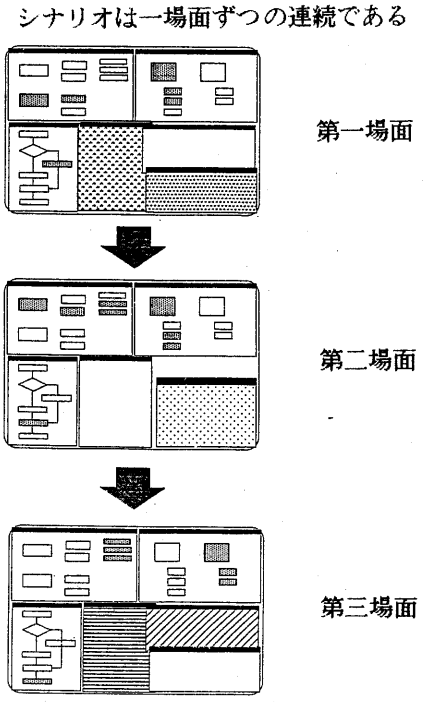

図　5　　シナリオの動き

５．２　ソフトウェア構成

ＣＯＭＩＣＳは、ディスプレイ、キーボード、マウスを持つワークステーション上で実現した。ソフトウェア構成を図６に示す。

１）ｲﾝﾀｰﾌｪｲｽ部：ウィンドウとマウスの管理
２）ＢＯＸ情報部：ＢＯＸの作成と管理
３）LINK 情報部：BOX間のリンク作成と管理
４）picture部：BOXの説明図の作成と表示
５）シナリオ部：シナリオの作成と実行
６）データ部：データファイルの管理

６．ＣＯＭＩＣＳの使用

ＣＯＭＩＣＳは、システムのイメージを表現するためのツールである。しかし、システムの内容は設計者が入力しなければならない。そのため、入力の手間がかからないように入力方法を容易にしている。ＣＯＭＩＣＳは、以下のような場面で使用できると考えられる。

①［ＯＨＰのように］システム設計者がソフトウェアの内容をプロジェクトメンバーに説明する
②［黒板代わりに］メンバー同志でディスカッションする
③［仕様書を読む前に］メンテナンス要因がシステムを理解する
④［開発ノートとして］システム設計者がソフトウェアを設計するとき・後にそのプロセスを知りたいとき

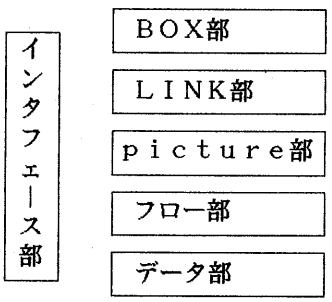

図　6　ＣＯＭＩＣＳのソフトウェア構成

７．おわりに

本研究の目的は、ソフトウェア開発プロジェクトの生産性向上を目指し、プロジェクトにおけるコミュニケーションをサポートすることである。

ソフトウェアは、適切な表現方法がないために意図の伝達が難しく、コミュニケーションが円滑に行かないという問題があった。そこで、ソフトウェアのイメージを表現する方法として劇場モデルを考案し、それを実現するツールＣＯＭＩＣＳを開発した。

劇場モデルは、ソフトウェアをドラマのように表する方法である。システム全体のイメージを表現することもできるし、その中の一つのモジュールもプログラムソースの集合として表現できる。また、そのシステムが他のアプリケーションとどのような位置関係にあるかを描くこともできる。視点を移動することによって様々なレベルでの説明に使える。

ＣＯＭＩＣＳは、劇場モデルの実現可能性を示すプロトタイプである。

ＣＯＭＩＣＳの今後の課題として、第一に評価の問題がある。今は試作の段階であり、劇場モデルの有効性や実用化への問題など、検証すべき点が残されている。第二に、説明機能の強化が問題である。劇場モデルの効果は、シナリオの作り方にも依存する。スッキリとしたソフトウェアであれば、良いストーリィになり、観客に意図が伝わりやすいであろう。複雑で不自然なストーリィであれば、モジュール達のダイナミックなプロセスを見てもイメージがわかないかもしれない。現在のＣＯＭＩＣＳはドラマの見せ方を支援しているが、今後、ドラマの作り方の支援も検討していきたい。

《参考文献》

１）特集：「ソフトウェア工学の現状と動向」
情報処理　VOL.28 NO.7 1987

２）花田・藤野：「ソフトウェア生産技術」
電子通信学会(1985)

３）Bannon.L et al.：" COMPUTER SUPPORT FOR COOPERATIVE WORK : AN APPRAISAL & CRITICIUE " EURINFO 297-303(1988)

４）岸本・他：「ｿﾌﾄｳｪｱ生産ﾌﾟﾛｾｽにおけるｲﾝﾀｰﾗｸｼｮﾝの分析」
第35回情報処理全国大会(1987) 1141-1142

５）岸本・他：「ｿﾌﾄｳｪｱ意図伝達支援ツール
ＣＯＭＩＣＳ（１）・（２）」
第37回情報処理全国大会(1988) 857-861

６）大山・東：「認知心理学講座１認知と心理学」
東京大学出版会(1984)