# 劇場モデルに基づいたソフトウェア意図伝達支援ツール COMICS†

仲谷美江†† 西田正吾††
坂口敏明††† 後藤卯一郎††††

本論文は，ソフトウェア開発プロジェクトにおけるコミュニケーション支援に関するものである．ソフトウェアの生産性向上が急務となっている．本論文では，協同作業を支援し，開発プロジェクトの生産効率を上げることを目指している．プロジェクトにインタビュー調査を行ったところ，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るという問題が見られた．このため，コミュニケーションの支援が生産性向上に有効であると予測される．ここでは，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．劇場は，俳優が舞台の上でドラマを演じ，観客に見せる場所である．観客はドラマの進行を見て脚本家の意図を読み取る．これは，人間が，まとまりの中から意味を把握する認識メカニズムを持っているからである．ここで，ソフトウェアの構造がドラマと類似していることに着目した．ソフトウェアの進行を適切に表現すれば，人間はその中から設計者の意図を読み取ることができると考えられる．そこで，劇場モデルに基づいた表現方法を実現する意図伝達支援ツール COMICS (COMputer-based Intention Communication System) を試作し，その有効性について検証した．

## 1. はじめに

ソフトウェア需要が急増している．従来，ソフトウェア生産性向上のために，以下のような研究が行われてきた[1]．

1) 技術力の向上：プログラミング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) ソフトウェア開発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1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，ソフトウェア開発プロジェクト（以下，単にプロジェクトと呼ぶ）の運営・管理を支援するものである．

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．プロジェクトの生産性を上げることも重要な課題である．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．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．

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の作業内容の調整は，管理者の仕事である．各メンバの作業が矛盾なく進むように定期的なレビューも行われる．

このような管理方法は，ソフトウェアに限らずさまざまなプロジェクトで行われている．現在では問題の複雑化，大規模化が進み，プロジェクトで解決する場合が増えてきた．これに伴って協同作業の支援が新しい課題になり，CSCW (Computer-Supported Cooperative Work) が注目され始めている[3]．協同作業は個人作業と異なり，メンバ間のコミュニケーションが重要な要素になっている．CSCW は，コンピュータを利用してグループワークを支援しようという新しい研究分野である．

本論文では，ソフトウェア開発プロジェクトの生産性向上のために，CSCW という視点からアプローチする．まず，実際のソフトウェア開発のプロジェクトに対してインタビューを実施し，メンバ間のコミュニケーションの実状と問題を調査した[4]．その結果を踏まえて，ソフトウェアの意図伝達方法として劇場モデ

† Computer-based Tool for Intention Communication in Software Development: COMICS by MIE NAKATANI, SHOGO NISHIDA (Central Research Laboratory, Mitsubishi Electric Corp.), TOSHIAKI SAKAGUCHI (Industrial Electronics & Systems Development Laboratory, Mitsubishi Electric Corp.) and UICHIRO GOTO (Power & Industrial Systems Center, Mitsubishi Electric Corp.).

†† 三菱電機(株)中央研究所
††† 三菱電機(株)産業システム研究所
†††† 三菱電機(株)制御製作所

ルに基づいた方法を提案し，それをワークステーション上で実現してコミュニケーション支援ツールを開発した[5]．また，本ツールの試用を通じて，その有効性についても検証した．

## 2. プロジェクトにおけるコミュニケーションの問題

### 2.1 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，現実には大規模システムは，協同で開発せざるをえない．構成要素が相互に関連しているものを協同開発する場合，各要素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良いプログラムを書けない．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良くなる．全体にとって良い部分を作るためには，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ものと考えられる．このことを確認するために，実働プロジェクトに対してインタビューを行った．

### 2.2 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

プロジェクトにおけるコミュニケーションの現状と問題点を探るためにインタビュー調査を行った．対象と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．以下の三つのテーマで質問し，自由に解答してもらう形式で行った．

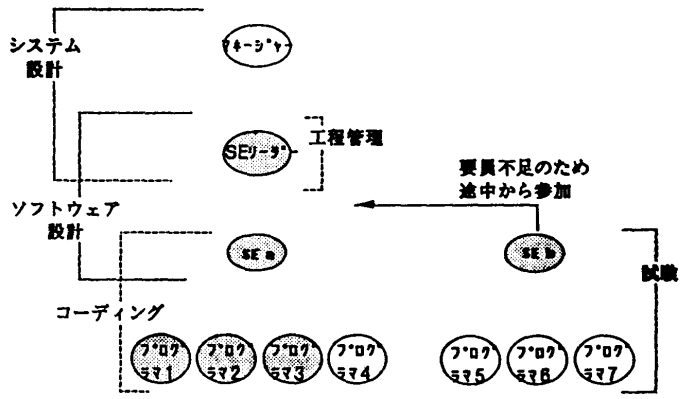



**図1** インタビューしたプロジェクトのメンバ構成
Fig. 1 A structure of the interviewed project.

① メンバ相互のコミュニケーションの手段と内容
② 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
③ 各メンバはどのようにシステムを理解しているか

インタビュー時間は1人約2時間である．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 プロジェクトにおけるコミュニケーション

プロジェクト内での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）に大別される．

- 公的なコミュニケーションは，メンバ全員や，チームごと（コーディングチーム，テストチーム）で行われる．システム内容の説明，開発方針の説明，進捗報告など，プロジェクトの意図と進行を揃えるためのコミュニケーションである．
- 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われる．**図2**に示すように，公的なまとまりとは関係なくコミュニケーションが行われる．内容はさまざまで，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 ①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

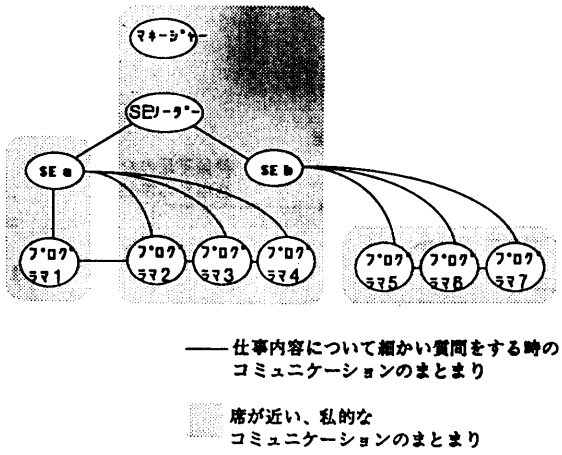



図 2 プロジェクト内のコミュニケーションの形態
Fig. 2 The formation of communication in the project.

- 公的なコミュニケーションは，話し手から聞き手への一方的な伝達になりがちで，聞き手がどう理解しているか確かめることがない．そのため，意図が伝わらなかったり，誤解されたりするケースが見られる（付録・プロトコル1，2参照）．
- 私的なコミュニケーションでは，話し手，聞き手の相互理解が得られ，公的なコミュニケーションで伝達されなかった部分を補っている．しかし，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，プロジェクト全体の意図が統一されにくいという危険がある（付録・プロトコル3～5参照）．

③ メンバが抱いているシステムイメージ

各メンバが，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．マネージャは，システムの最終イメージや工程の進み具合を，他のプロジェクトとの関係で捉えている．システム設計者は，ソフトウェアとの関係で捉えている．システム設計者は，ソフトウェア技術者としての持論を持っていて，ソフトウェアの品質や開発方法にこだわりを持っている（付録・プロトコル6，7参照）．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（付録・プロトコル1，8，9参照）．また，経験の少ないプログラマは，私的なコミュニケーションをする相手の視点に影響される傾向が見られる．例えば，図2におけるプログラマ1はSEaと同じ考え方をし，プログラマ2はSEリーダと同じ考え方をしている．

### 2.3 プロジェクトにおけるコミュニケーションの問題点

以上の結果から，次のような問題点が明らかになった．

1) プロジェクト内のコミュニケーションは，必ずしもうまくいっておらず，意図の統一が取れていない．

2) そのため，プロジェクトメンバが抱いているシステムイメージは，人によりかなり異なっている．

3) また，極端な場合には，システムを理解できていないままコーディングを始めることもある．

また，インタビュー結果を分類したところ，コミュニケーションを妨げる主な要因は，

① メンバの知識・経験・視点の差が大きいこと
経験や能力の個人差ばかりでなく，立場や環境により，考え方の差が大きい．

② ソフトウェアが抽象的で表現が難しいこと
ソフトウェアを表現する適切な方法がない．そのため，自分のイメージを言い表せなかったり，人の意図をうまく理解できなかったりする．また，自分の中でもイメージが曖昧な場合が多い．

の2点にまとめられた．これは，ほとんどのプロジェクトに共通の問題であると思われる．

ソフトウェアの表現方法としては，従来から仕様書やフローチャートが研究されてきた．これらの方法は，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，一方では，

① 量が多く，作成するのも理解するのも困難である．

② 詳細で正確な情報だが，全体のイメージがわかない．

③ 要素間の関連がわかりにくい．

などの問題点があり，具体的なイメージや意図が伝わりにくかった．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（付録・プロトコル10参照）．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．しかし，経験のないメンバは，システム全体がどんな動きをするのかわからない，自分に直接関係ないところはわからないという状態のままプログラミングしてしまう．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，などの危険が生じる．

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，従来の方法では表現できなかったシステムの意図やイメージを表現する方法が必要である．そこで，次章では，ソフトウェアの意図の表現方法を検討する．

## 3. 劇場モデルに基づいた意図の表現と伝達

### 3.1 人間の理解のメカニズム

適切な表現方法を考えるためには，まず人間の理解のメカニズムを知り，それに適した方法を採らなければならない．

仕様書や説明書は，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．プログラマは，自分の作ったモジュールとシステムの動きとの関係をうまく結び付けられない．構造と機能の関係を結び付けるものとして，次の点が必要と考えられる．

- 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．
- 部分同士の関連がわかる．

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る．人間の認識のメカニズムには，

① 文脈の中で現象を捉えようとする
② 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，それに気が付かずに大要を理解できる．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．例えば，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と考えられる．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．

### 3.2 劇場モデルの提案

ここで，意図伝達方法として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ーを演じてドラマを構成する．脚本家の意図は，俳優の1人1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，その劇の中で位置付けられる．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．そして，脚本家が自分の演出した劇を観客に伝達するコミュニケーションと，システム設計者が自分の設計したソフトウェアをプログラマに説明するコミュニケーションの機能と構造も類似している（**表 1**）．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，モジュールやデータ，ハードウェアを設定する．ソフトウェアは，モジュールなどの要素が相互作用してシステムを実現していく．設計者の意図は，個々の要素ではなく，全体の流れの中にある．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．

観客が劇を見て脚本家の意図を読み取ることができるのは，人間の認識メカニズムが，まとまりが全体として持つ包括的な意味を理解できるためである[7]．したがって，プログラマがソフトウェアをまとまりとして見れば，その中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，従来の方法ではソフトウェアをまとまりとして表現できない．

表 1 劇場とソフトウェアの意図伝達
Table 1 Comparison between communication in theater and communication in software.

| 類似点 | 劇場の意図伝達 | ソフトウェアの意図伝達 |
|---|---|---|
| 構 造 | 脚本家，登場人物，観客で構成される． | システム設計者，設計内容，プロジェクトメンバで構成される． |
| 機 能 | 脚本家の意図は，登場人物1人1人が持っているのではなく，動作やセリフの連続の中に表現されている．<br>観客は，その流れの中からテーマを読み取る． | 設計者の意図は，一つ一つのモジュールを見てもわからない．<br>それらの関係性の中に意図がある．<br>意図がわからなければ，モジュールの本当の役割はわからない． |
| | 同じテーマでも，脚本家によって演出の仕方が異なり，観客の印象も異なる． | 同じ機能のシステムでも，設計者によって全然違うソフトウェアになることもある． |
| | 観客は，劇を一つのまとまりとして見ることができる．が，1人の俳優を中心に見ることもできる． | メンバは，システムの全体像を知ることが必要である．が，自分の担当モジュールに関連したところを重点的に知ることも必要である． |

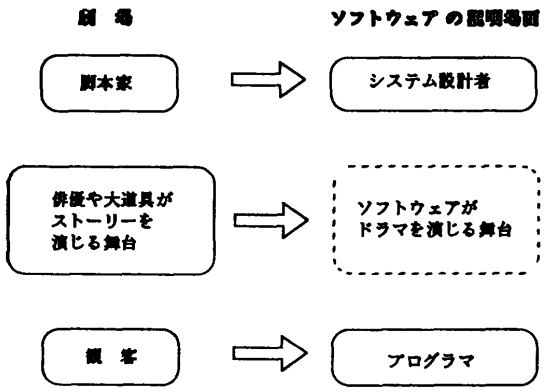



図 3 劇場モデル（劇場からソフトウェアの説明場面へのメタファ）
Fig. 3 Analogy between a theater and an explaining act in software.

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る表現方法『劇場モデル』を提案する(図 3)．劇場には，以下の点が表現できる舞台空間を設定する．

- システム全体のイメージと具体的な動き
- 全体の中での，個々の構成要素の位置付け
- 構成要素同士の関係性

このような舞台を実現することにより，ソフトウェアの具体的な動きを伝達し，プログラマが自分が担当するモジュールの位置付けと役割を理解することを支援する．

なお，劇場というメタファを用いた試みはこれまでにもいくつかなされている．

一つには，Programming-by-Rehearsal システム[8]や対話的グラフィックプログラミング環境 Pict/D[9]などの視覚的なプログラミング環境がある．

また，Performing Art Paradigm[10]は，ソフトウェア開発プロセスそのものをパフォーマンスと見なしたプロセスモデルである．これは，従来の Water-Fall モデルに対するメタプロセスモデルで，ソフトウェア開発の設計からテストまでを Practice : Performance : Postmortem の三つの段階，すなわち練習・実演・反省の繰返しとして考えようというものである．

これに対し，本劇場モデルは，ソフトウェアの表現方法を提案し，コミュニケーションを支援するためのモデルであり，劇場メタファの用い方が異なっている．

## 4. 劇場モデルの実現ツール COMICS

### 4.1 COMICS の概要

ソフトウェアにおける舞台を提供し，ソフトウェアの意図伝達空間を創りだすツールとして COMICS を開発した．COMICS の目的は，正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，ソフトウェアの意図を伝えることである．COMICS は，劇場モデルを基にした表現方法を実現している．

COMICS 上での劇場モデル表現について説明する．劇場モデルは，システムの構造を劇場の構造と対比させ，システムをソフトウェアモジュールやハードウェアらが演じる一連のドラマと見なし，そのドラマをワークステーション上で表現しようというものである．そのほかに，劇場モデルには視点という概念を導入する．システムを説明するとき，説明者の視点や説明内容によって視点の取り方が異なってくる．本論文

では，視点の取り方も説明者の意図を表現する重要な情報であると考え，説明者の視点を明示する方法を取った．以下，劇場モデルを構成する要素についての定義と，COMICS での実現方法について述べる．

（1） 視　　点

脚本家は，自分なりの捉え方でテーマを表現する．同じテーマでも，脚本家によって異なる表現になる．この，ものの見方・捉え方を視点と呼ぶ．

ソフトウェアの場合も同様に，さまざまな視点から眺めることができる．システム設計者の意図や説明の仕方によって，表現方法が異なる．例えば，同じシステムでも，プログラマの視点を取ればモジュールに分割した表現になり，ユーザの視点を取ればインタフェースを中心とした表現になるであろう．

COMICS では，一つの視点を 1 枚のウィンドウで表す．視点は複数設定できる．システムのドラマが進行しているとき，視点のウィンドウは常に画面上に表示されている．

（2） ドラマの構成要素

劇場は，ドラマを作る脚本家，ドラマを構成する俳優・舞台装置・小道具・照明等の要素，ドラマを見る観客，から成る．

ソフトウェアの場合は，システムを説明する者，システムを構成するモジュール・データ・ハードウェア・ネットワーク等の要素，システムの説明を聞く者，から成る．

COMICS は，システムの構成要素を表現するツールである．COMICS では，一つの構成要素を一つのボックスで表す．ボックスは視点ウィンドウ上に記述される．例えば，モジュール構成という視点のウィンドウ上には，ソフトウェアモジュールの構成図がボックスで記述される．各ボックスは，1 枚のピクチャを持っている．ピクチャは，テキストやグラフィックでボックスに関する説明などが書けるメモ用ウィンドウである．任意のときに開いて見ることができる．

（3） リ ン ク

ドラマの構成要素間には，人間関係や人間とモノの関係など，何らかの関係がある．その関係は，ドラマのストーリー展開の上で重要である．

ソフトウェアの場合にも，要素間の関係は非常に重要である．モジュール間，モジュールとデータ，ネットワークなど，要素はすべて何らかの関係でつながっている．

COMICS では，関係をリンクで表現する．任意のボックス同士をリンク付けでき，リンク付けるとボックスの間に線が引かれる．異なるウィンドウ上のボックスをリンク付けしたときは，リンク情報は存在するが，線は引かれない．

（4） シ ナ リ オ

劇場では，脚本家がシナリオを書く．シナリオは，ドラマの構成要素の動きやストーリー展開を書いたものである．

ソフトウェアの場合，システムを説明する者がシナリオを書く．シナリオとは，システムの動き，プログラムのアルゴリズム，開発の進捗などを示したものである．一つのシステムでも，説明の内容によって複数のシナリオができる．

COMICS では，シナリオを場面の連続として表現する．一つの場面は一つのボックスで表し，ボックスを並べたものがシナリオになる．1 本のシナリオは 1 枚のシナリオウィンドウ上に記述する．このボックスにもピクチャが付いており，場面の状態をテキストとグラフィックで書くことができる．

（5） 舞　　台

劇場では，ドラマの構成要素がストーリーを展開していく場所が舞台である．この舞台の時間的展開を追うことにより，観客は脚本家の意図を捉えることができる．

ソフトウェアの場合も同様に，ドラマの構成要素がシナリオに従って時間的に展開していく様子を見せることにより，ソフトウェアの展開，すなわち，システムの動き，プログラムのアルゴリズム，開発の背景等を伝えられるものと思われる．

COMICS では，舞台をステージとして表現する．画面にステージ用の空間を設け，場面の様子を表示する．場面の様子は，その場面に関連するボックスを点滅させると同時に，ピクチャをステージ上に表示することにより表現する．シナリオの展開に従って場面が進み，ボックスの点滅状態とステージに表示されるピクチャが紙芝居のように変化する．点滅するボックスや表示するピクチャは任意に設定できる．

図 4 はソフトウェアの劇場表現イメージである．劇場では，俳優の集合や小道具の集合，効果や背景の集合など，いくつかの要素集合がある．舞台では，それらの要素が交互に登場し，相互作用によってストーリーを表現していく．ソフトウェアも図のようにいくつかの要素集合から成る．COMICS では要素集合間の関係と，それらが舞台の上で相互作用してシステム

というドラマが進行していく様子を表現する．図4のイメージをそのまま画面にしたのが図5である．

図5は実際のシステムのデータを入力したCOMICSの画面例である．これは，システムが第1版から第2版へ更新されたときのプロセスを説明するためのデータである．画面は，プロジェクトメンバの構成図ウィンドウ（左下），ソフトウェアモジュール構成とそれに対応するドキュメント構成のウィンドウ（左上），ネットワークとディレクトリ構造のウィンドウ（右上）から成る．右下のウィンドウはシナリオウィンドウで，システムが第1版から第2版に更新されたときのプロセスをボックスで表現している．ここでは，シナリオの第6場面での状態が表示されている．この場面は，システムの更新に伴いデータ構造がどう変更になったかを説明する場面である．この場面に関連のあるボックスが点滅し，画面中央のステージ部分には場面の状態を説明するピクチャが表示されている．

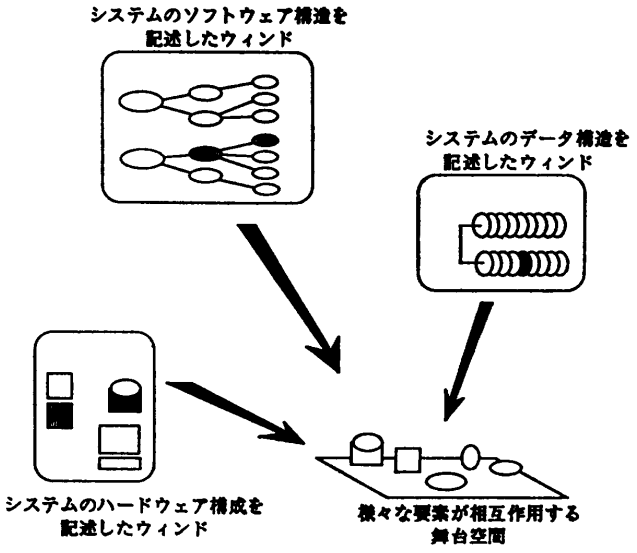



図4 劇場モデルによるソフトウェア表現の例
Fig. 4 An example of software representation by theater model.

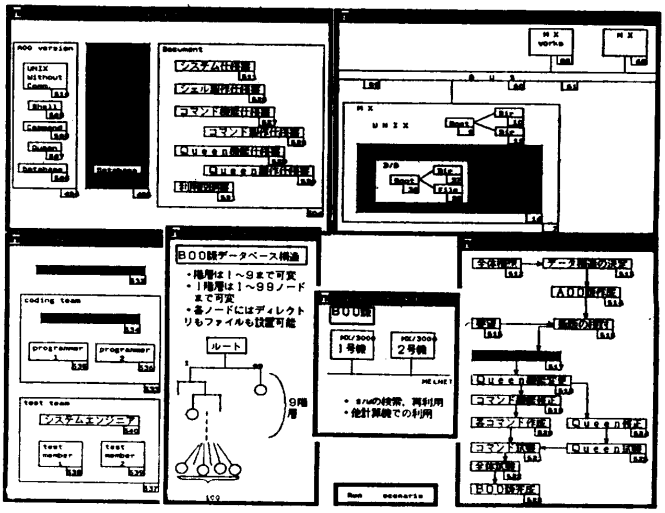


図5 COMICSの画面例
Fig. 5 An example of COMICS.

図6はこのシナリオが第5・6・7場面と進行している様子である．第5場面は第2版の開発方針を説明する場面，第6場面は図5と同じもので，更新に伴うデータ構造の変更を説明，第7場面はそのデータに直接関わるモジュールの機能変更を説明する場面である．場面が変わると点滅するボックスが変化し，ステージ部分のピクチャも変化している．

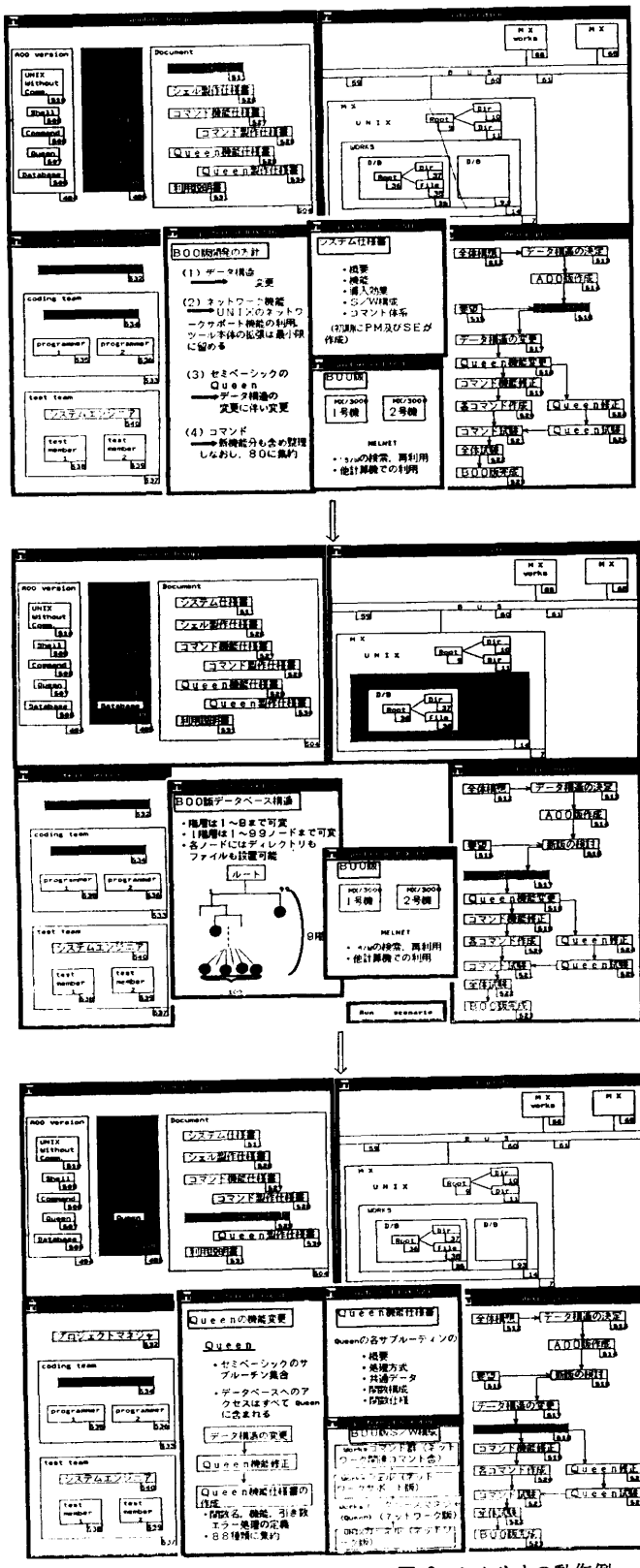
なお，COMICSは，ソフトウェアの説明ツールとして開発したが，ソフトウェアの説明にはさまざまなレベルがある．プロジェクトの初期にシステム設計者がプロジェクトメンバに行うシステムの概要説明，開発途中にプロジェクトのメンバ間で行われるシステムに関するディスカッション，開発終了後，開発担当者から保守担当者に対して行われる説明等，プロジェクトの工程に応じた説明レベルが考えられる．説明の内容・目的によってソフトウェアの記述レベルも異なってくる．COMICSでは，視点や要素の設定の仕方によって概要からプログラムレベルまでの記述が可能であり，さまざまなレベルの説明に利用できるものと考えられる．

### 4.2 COMICSの機能

COMICSには，以下のような機能がある．

① 設計者の自由な視点からシステムの構造を記述できる．

視点は，任意に設定できる．誰にどのようにシステムを説明するか，という意図によって設定の仕方が異なってくる．例えば，ソフトウェアモジュール群，プロジェクトチームのメンバ群，ドキュメント群，を作れば，「各モジュールの担当メンバとドキュメント」という視点でシステムを捉えることができる．また，大モジュール群，サブルーチン群，基本関数群，など同じソフトウェアを記述レベルによって分類することも



第5場面
“システム第2版の開発方針について”

第6場面
“システムの更新に伴うデータ構造の変更について”

第7場面
“データ構造の変更に関わるモジュールの機能変更について”

図 6 シナリオの動作例
Fig. 6 An example of a scenario behavior.

可能である．

② ソフトウェアの動作をシナリオとしてダイナミックに表現できる．

システムの動きはシナリオで表現する．視点とシナリオの設定の仕方によって，システムの動きやアルゴリズムだけでなく，ソフトウェアの設計過程や更新のプロセスをドラマとして表現することができる．

③ プログラマは，自分の理解に合わせたペースでソフトウェアの内容を見ることができる．

シナリオはフローチャートのようにウィンドウに表示されるので，プログラマはシステムの動きを目で読むことができる．また，シナリオを自動で走らせたり，わからない箇所だけを繰り返し見るなど，任意のスピードでシナリオが走る．さらに，個々の要素についての詳細な記述や，要素間のリンクも自由に表示させられる．このような機能により，プログラマは説明を受動的に聞くのではなく，自分でさまざまな角度からシステムを眺め，その動きを確かめられる．自ら主体的に認識しようと働きかけることは，理解を支援する[11]．

④ 入力・編集が容易なので，設計者はCOMICS上で設計できる．

COMICSで説明するソフトウェアの内容は，システム設計者が入力しなければならない．そのため，設計者に対する負荷が増加する．そこで，COMICSはテキスト入力以外の入力・編集をマウス操作のみで行えるようにした．操作が容易なので，設計者はCOMICS上で設計し，その結果をそのまま説明に使うことができる．

また，COMICS上で設計すれば，設計者の設計プロセスが記録・再現でき，意図の伝達にとって効果的であると考えられる．

⑤ ソフトウェアの設計履歴を再現することにより，設計者の思考プロセスをたどることができる．

入力履歴の再現は，まず画面がクリアされ，白紙の状態から入力された順にボックスが登場する．同時にボックスの詳細な説明を表示することもできる．複数の視点で並行してボックスが生成されていく様子から，設計者がそのボックスを設計した経過が表現できる．設計プロセスの再現は『なぜ，どうやって』という視点からソフトウェアを説明でき，設計者の意図の理解を支援するものである．

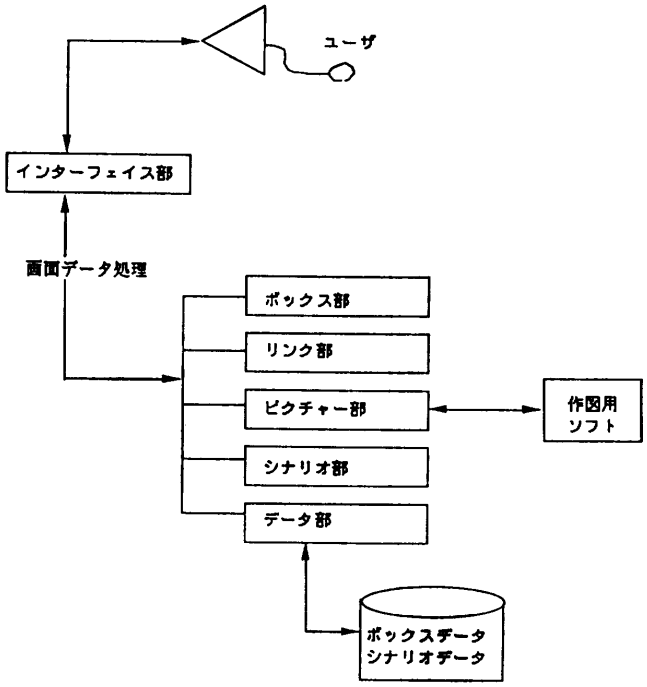



図7 COMICSのソフトウェア構成
Fig. 7 Software architecture.

### 4.3 ソフトウェア構成

COMICSは，LISPにより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7に示す．

1) インタフェース部：COMICS全体に共通なウィンドウとマウス操作を提供する．
2) ボックス部：ボックスの作成と，コピー・修正・移動などの編集を行う．
3) リンク部：ボックス間のリンク作成と，リンクの表示を行う．
4) ピクチャ部：各ボックスの説明や，ステージに表示する画面を作成，表示する．テキストや作図は，外部の作図用ソフトを呼んで行っている．
5) シナリオ部：シナリオの作成と，シナリオの実行機能を提供する．シナリオの作成は，ユーザがストーリーに従って動くボックスをマウスで選択し，そのとき，同時に表示させたい画面もマウスで選択していく．
6) データ部：データファイルの保存，読み込みを行う．

データはフレーム構造で，ボックスデータとシナリオデータの2種類ある（図8）．各ボックスにつき一つのボックス構造体が作成され，属性情報を格納す

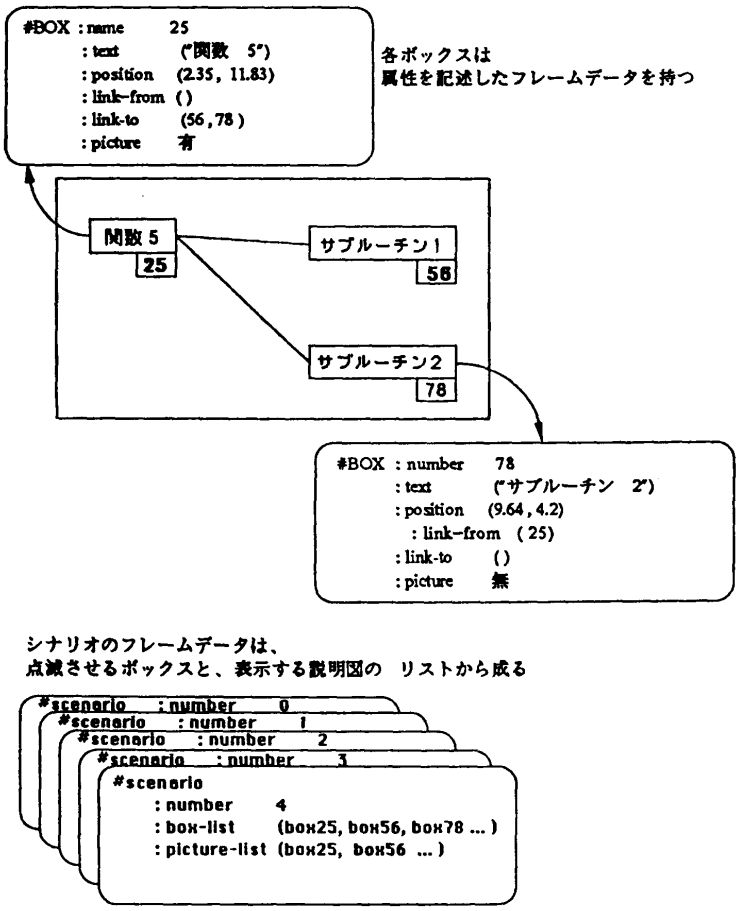



図 8 ボックスとシナリオのデータ構造
Fig. 8 An example of frame data.

る．各シナリオにも一つのシナリオ構造体があり，ボックスのリストを格納する．

## 5. COMICS の使用と効果

COMICS は，コミュニケーションをサポートするためのツールである．ソフトウェア開発のライフサイクルの中で，COMICS は以下のような場面で利用できると考えられる．

1) システム設計者からプロジェクトメンバへのコミュニケーション：

システムの概要や具体的な動作イメージなどをメンバに伝えるとき，OHP のように用いる．また，設計者の意図を理解してもらい，プロジェクトとしての統一を図るためにも使える．

2) プロジェクトメンバ同士のコミュニケーション：

プログラマ間のディスカッションや，設計者とプログラマとのディスカッション時に，COMICS の上で容易にソフトウェアを修正・変更し，シナリオを走らせられるので，黒板のように利用できる．

3) プロジェクトメンバからメンテナンス要員へのコミュニケーション：

保守に先立って，システムの理解が必要である．まず，COMICS によりシステムの概要と具体的な動きを把握し，設計者の意図を理解する．その後，仕様書でソフトウェアの詳細を確認し，保守にあたることが効果的である．

COMICS を開発後，その有効性の検証のために，COMICS を試用した．

対象としたのは，開発に 6 人・年を要した実システムである．このシステムは，まず第 1 版を開発し，その後改善して第 2 版の開発が終了している．このシステムの概要と開発過程のデータを，COMICS を初めて操作する者に入力してもらった．そして，以下のシナリオを作成した．

① 開発プロジェクトが企画される過程を説明するシナリオ
② システム全体の概要を説明するシナリオ
③ 一つのモジュールの役割・フローを説明するシナリオ
④ システムが第 1 版から第 2 版へ更新される過程を説明するシナリオ

すべてのデータの入力には，4 人・日を要した．入力データの規模を**表 2** に示す．入力操作そのものよりも，シナリオを考えるのに時間がかかった．シナリオは，伝えたいことを要領良くまとめ，簡潔でわかりやすいストーリーを作らなければならない．そのため，シナリオを作成し，走らせてみては作り直すという作業を繰り返した．しかし，この試行錯誤によって考えがまとまるという利点もあった．

入力後，このシステムの開発プロジェクトマネージャとシステム設計者に対し，COMICS がシステムの意図伝達に効果的かどうかを評価してもらった．その結果，以下の評価が得られた．

表 2 COMICS に入力したデータ
Table 2 Input data for COMICS.

| シナリオ | 使用した構成要素ウィンドウ数 | ボックス，ピクチャの総数 | シナリオの場面数 |
|---|---|---|---|
| ①開発プロジェクト企画の過程 | 3 | 41 | 9 |
| ②システム全体の概要 | 4 | 93 | 14 |
| ③モジュールのフロー | 3 | 46 | 6 |
| ④システム第1版から第2版への過程 | 4 | 55 | 11 |

○満足な点

- 設計プロセスやバージョンアップの経過などが表現されており，設計者の意図がプログラマに伝達されると期待できる．
- ソフトウェア開発プロセスの一事例として保存すれば，開発チームから保守チームへの意図伝達や，他のプロジェクトを企画・管理するときの参考資料として利用可能である．

○不満足な点

- わかりやすいシナリオの作成をサポートしてほしい．
- 設計者がアイデアを入力し，後から自由に階層化・詳細化できる機能がほしい．

このように，COMICS を用いると，比較的短時間でダイナミックな動きや相互の関係が記述でき，しかも設計プロセスの保存などにも有用なことが確かめられた．

## 6. おわりに

本論文では，ソフトウェア開発プロジェクトの生産性向上のための，コミュニケーション支援方法を提案した．

実際の開発プロジェクトに対するインタビュー調査の結果から，メンバ間のコミュニケーションの実情と問題を整理した．そして，ソフトウェアの意図伝達方法として，劇場モデルに基づいた方法を提案し，それをワークステーション上で実現してコミュニケーション支援ツールを開発した．さらに，その有効性についても検証した．

今後は，COMICS の有効性についてさらに検討を加えると共に，実用化に際して必要な機能を整備していく予定である．

## 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, pp. 844-939 (1987).
2) 花田，藤野：ソフトウェア生産技術，p. 299，電子通信学会，東京 (1985).
3) *Proceedings of CSCW 86*, ACM SIGCHI & SIGOIS (1986).
4) 岸本，西田，後藤：ソフトウェア生産プロセスにおけるインターラクションの分析，第35回情報処理学会全国大会論文集，pp. 1141-1142 (1987).
5) 岸本，西田，後藤：ソフトウェア意図伝達支援ツール COMICS (1)・(2)，第37回情報処理学会全国大会論文集，pp. 857-861 (1988).
6) 大山，東編：認知心理学講座1 認知と心理学，p. 293，東京大学出版会，東京 (1984).
7) ポラニー，M.：暗黙知の次元，p. 146，紀伊國屋書店，東京 (1980).
8) Finzer, W. and Gould, L.: Programming by Rehearsal, *Byte*, Vol. 9, No. 6, pp. 187-210 (1984).
9) Glinert, E. P. and Tanimoto, S. L.: Pict: An Interactive Graphical Programming Environment, *Computer*, Vol. 17, No. 11, pp. 7-25 (1984).
10) Marks, P.: System Development as a Performing Art: A People-Centered View of Leonardo, MCC Technical Report Number STP-365-86 (1986).
11) Gibson, J. J.: Observation on Active Touch, *Psychological Review*, Vol. 69, No. 6, pp. 477-491 (1962).


## 付録

プロトコル

1 「あまり説明はしてもらってないですね．あんなの徐々にわかってくるんですね．だからこの全体がわかったことないです．まだ全体を把握していないんですけども．」

2 「遅れていることしかわからない．外に見えているのは時間的な遅れですね．〜その原因までわからないから〜そんなんじゃハッパかけるしかないんですね．」

3 「やっぱり隣に座っているって利点がありますね．〜いろいろ付随したお話しとかしてくださるから．」

4 「本当は（プログラマが）机を並べてやっていくのが一番いいんですけれども〜横の連絡がすごく難しいです．」

5 「仕事の話は，やはり隣の人に聞くことが多い．

コーディングのことならば（離れているプログラマに）**電話で聞き，わからなければ出かけていきます.**」

6 「もう全部自分で作ってしまいたいという感じで,」

7 「管理者はあまりいらないんじゃないかという持論を持っているんですね.」

8 「始め，（このシステムが）何やってるかわからなかったです．それをわからないまま何か作ってやってた感じ.」

9 「だから，つい最近までどういう風になって動くのか（プログラマは）みんなわからなかったですね.」

10 「設計って言っても，そんな 100% 書ききれないですよ．～これくらいのことは当たり前に解釈して，ここまでやってくれるはずだと，それを前提としないとどうしようもないんですね.」

（平成元年 3 月 9 日受付）
（平成元年 10 月 11 日採録）

**仲谷 美江**（正会員）

昭和 35 年生．昭和 58 年大阪大学人間科学部卒業．同年三菱電機(株)入社．以来，同社中央研究所にて，マンマシンインタフェース，理解支援，協同作業支援の研究に従事．日本認知科学会会員．

**西田 正吾**（正会員）

昭和 27 年 1 月 5 日生．49 年 3 月東京大学工学部電子工学科卒業．51 年 3 月同大学院工学系研究科電気工学専攻修士課程修了．同年 4 月三菱電機(株)入社．現在，中央研究所システム基礎研究部第 4 グループマネージャ．主として大規模システムの運用・制御，教育訓練システム，ヒューマンインタフェースの研究に従事．59 年 3 月工学博士．59 年 9 月より 1 年間マサチューセッツ工科大学メディアラボ客員研究員．61 年電気学会論文賞．IEEE，計測自動制御学会，電気学会各会員．

**坂口 敏明**

昭和 21 年 12 月 20 日生．44 年 3 月京都大学工学部電気工学第 2 学科卒業．46 年 3 月同大学大学院工学研究科電気工学専攻修士課程修了．同年 4 月三菱電機(株)入社．中央研究所にて，主として電力系統の保護・制御・運用技術の研究，ならびにシステム開発に従事．現在，同社産業システム研究所社会システム開発グループマネージャ．56 年 11 月工学博士．59 年電気学会論文賞受賞．IEEE，電気学会各会員．

**後藤卯一郎**（正会員）

昭和 26 年生．昭和 49 年東京大学工学部電気工学科卒業．同年，三菱電機(株)入社．以来，制御製作所にて，電力系統制御システムの開発に従事し，給電自動化システムおよび制御所監視制御システムのシステム設計・製作を担当してきた．現在は，ソフトウェア生産支援のためのツール開発に従事．