グループウェア 6－4
（1994. 4. 28）

協調計算システムの動作仕様群の分散実行系

今城 広志　尹 達雄　岡野 浩三
東野 輝夫　谷口 健一

大阪大学 基礎工学部 情報工学科

グループウェアのような協調計算システムの設計法として，システム全体の動作 (要求仕様) のみを記述し，その記述から要求仕様通りに動作する各ノードの動作仕様を自動生成することが有効である．本研究では，要求仕様と入出力ゲート·レジスタの割当て情報から自動生成された各ノードの動作仕様を分散実行する処理系 (協調計算システム) を作成した．この実行系では，データとして音声やテキストその他のファイルなどが扱え，線描画ファイル編集では同時に共同編集が行なえる．またゲートやレジスタの割当てを変えることにより異なる協調計算システムを容易に実現できる．本稿では医療診察を例に本実行系の概略について述べる．

Simulator for Protocol Specifications and its Application to Groupware

Hiroshi IMAJO, Tatsuo YUN, Kozo OKANO,
Teruo HIGASHINO and Kenichi TANIGUCHI

Dept. of Information and Computer Sciences, Osaka University,
Toyonaka, Osaka 560, Japan

In this paper, a groupware is treated as a distributed system. In the distributed system, each node (protocol entity) must exchange some data values and synchronization messages in order to ensure the temporal ordering of the actions described in a service specification for the distributed system. It is desirable that the protocol entities' specifications can be derived from a given service specification and a resource allocation (allocation of registers and gates). We have developed an algorithm for deriving the protocol entities' specifications and implemented the algorithm. In this paper, we have developed a simulator to execute those protocol entities' specifications in parallel. The simulator deals with many kinds of data types such as sounds and pictures. Some users can edit the same figure simultaneously and each modification is shown to the displays of all users. Our approach is useful for the case that the resource allocation is frequently changed. A medical diagnostic system is used as an example.

1 はじめに

近年，グループウェアに関する設計法や処理系について，多くの研究が行われている[1],[2],[3],[4]．グループウェアでは，人間の共同作業の計算機による支援を行なう．複数の計算機からなるシステムという点で，グループウェアを分散システム (協調計算システム) と捉えることができる．

抽象レベルでは協調計算システム全体としての各作業の実行順序や作業内容の記述をそのシステムの要求仕様とみなすことができる．与えられた要求仕様に対して分散システム上の各ノード (計算機) はお互いに協調しながら全体として要求仕様を満足するように動作する．一般に，1つのノードの動作仕様は，そのノード単独の作業の手順や内容の他，他のノードとのデータや同期信号などのメッセージの送受信に関する動作が記述されたものとなる．

協調計算システムを設計する際，設計者はシステム全体としての作業手順や内容のみを記述した要求仕様を設計し，その要求仕様から各ノードの動作仕様を自動生成することが高信頼性などの点で有効であると考えられる[5]．

我々は文献(6)で，有限個のレジスタを持つ拡張有限状態機械 (EFSM モデル) で記述された要求仕様と，各ノードへのレジスタや入出力を行なうゲート (リソース) の割当てから，全体として要求仕様通りに動作する各ノードの動作仕様を自動導出するアルゴリズムを提案した．さらに文献 (7) では，そのアルゴリズムに従って，実際に各動作仕様を自動生成するシステム (生成系) を開発した．しかしこの生成系は同一ゲートの複数ノードへの多重割当てに対応していなかった．

本稿では，文献 (7)の生成系をゲートの多重割当てに対応するよう改良した．そして，この生成系から生成された動作仕様群を，グループウェア向け協調計算システムとして実現するための分散実行系を作成した．この実行系では，データとして数値などの他に音声や画像情報などが扱える．また，要求仕様ではマイクからの音声の入力やエディタによるファイルの編集などの作業も記述することができる．特に，線描画ファイルの編集では，1つの線描画ファイルを複数人で共同編集できる機構を実現した．

我々のシステムは，要求仕様を変えることにより，異なる協調計算システムを生成できる．また同じ要求仕様に対しても，リソースの割当てを変えることによって異なる協調計算システムを生成できる．

2 動作仕様群の生成の概要

各ノードの動作仕様 (動作仕様群) は生成系に要求仕様やリソースの割当てなどを入力することによって生成される．また生成系ではデータの型として整数，文字，文字列の他に，テキスト，ビットマップ，線描画，音声を扱える．

2.1 要求仕様

協調計算システム全体の要求仕様は有限個のレジスタを持つ拡張有限状態機械 (EFSM) で記述される．EFSM の状態遷移を表す枝には，遷移条件式，入出力動作，レジスタ更新式がラベルとして付いている．EFSM のある状態において，その状態から出るすべての枝の遷移条件式を評価し，真となる遷移のうち1つを非決定的に選択してその入出力動作を実行する．その後レジスタ値が更新され，次の状態に移る．

要求仕様で記述できる動作 (実行系で実現できる動作) として，入出力動作では，すべてのデータの入出力を行なえる．レジスタ更新では，整数・文字・文字列の代入，整数・文字の演算，テキスト・ビットマップ・線描画の複製，テキストへの文字列の追加を行なえる．

図 1は医療診察を例にした要求仕様である．ここでは問診の部分しか EFSM で記述していないが，全体では 8 個のゲートと 12 個のレジスタを使って要求仕様を記述している．以下でその問診部分を説明する．

今，状態 2 にいるとする．ゲート *incon_key*からデータ x(問診結果) を入力し，レジスタ *incon_input* にデータ x を代入する．そして状態 3 に移る．

状態 3 においてレジスタ *incon_input* の値が "*end.*" でなければゲート *incon_dsp* へレジスタ *incon_input* の値 (問診結果) を出力し，レジスタ *pat_log*(問診ログ) にレジスタ *incon_input* の値を追加する．レジスタ *incon_input* の値が "*end.*" なら問診を終了し，ゲート *fcon_dsp* にレジスタ *pat_log* の値 (問診ログ) を出力し，レジスタ *fcon_karte* の値 (カルテ) を再編集する．

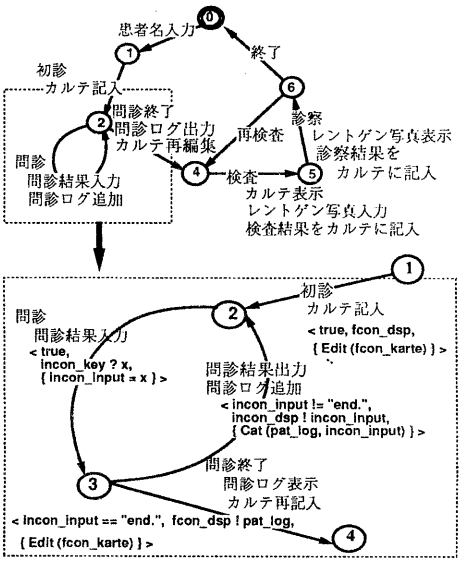

図 1: EFSM で記述された医療診察の要求仕様

2.2 リソースの割当て

要求仕様を複数のノードからなる協調計算システムとして実現する際, いくつのノードで構成するのか, また, レジスタや入出力のためのゲート (リソース) をどのノードに割当てるかを情報として与える. 特にゲートの割当ては, 各ノードがどんな役割をするのかに従って設計者が決定する.

図 1の医療診察では, 全体を初診, 問診, 検査, 診察といった役割に分割することが可能である. 例として, この要求仕様を 3 つのノードで実現する. ノード 1 には初診, ノード 2 には問診, 診察, ノード 3 には問診, 検査を割当てる.

これらの役割をもとに 8 個のゲートを以下のように割当てる.

ノード 1 *fcon_key*, *fcon_dsp*
ノード 2 *incon_key*, *incon_dsp*, *con_key*, *con_dsp*
ノード 3 *incon_key*, *incon_dsp*, *exam_key*, *exam_dsp*

*fcon_key*は初診の入力ゲート, *fcon_dsp* は初診の出力ゲートである. *incon_key*, *incon_dsp*, *exam_key*, *exam_dsp*, *con_key*, *con_dsp* も同様にそれぞれ問診, 検査, 診察の入力, 出力ゲートである.

問診はノード 2 とノード 3 に割当てられているが, ゲートの多重割当てが可能なので, 同じ役割を複数のノードに割当てることも可能である. 入力ゲートが多重に割当てられているときは, いずれか 1 つのゲートで入力を行なう. ただしファイルの入力 (編集) は, システムの外でエディタを立ちあげ共同編集を行ない, 編集終了後いずれか 1 つのゲートで入力を行なうと考える. 出力ゲートが多重に割当てられているときは, すべてのゲートで出力を行なう.

12 個のレジスタも以下のように割当てる.

ノード 1 *master_karte*, *master_log*, *master_roentgen*, *pat_name*, *pat_karte*, *pat_log*, *pat_roentgen*, *fcon_karte*
ノード 2 *con_input*, *con_karte*
ノード 3 *incon_input*, *exam_karte*

master_karte, *master_log*, *master_roentgen* はそれぞれ, 初期設定のカルテ, 問診ログ, レントゲンに相当するレジスタである. *pat_name*, *pat_karte*, *pat_log*, *pat_roentgen* はそれぞれ, 現在の患者の名前, カルテ, 問診ログ, レントゲンに相当するレジスタである. *con_input*, *incon_input* はそれぞれ, 診察, 問診のときに入力された文字列を一時的に保存するレジスタである. *fcon_karte*, *con_karte*, *exam_karte* はそれぞれ, 初診, 診察, 検査のときにレジスタ *pat_karte* の値 (カルテ) を取り込み, その後, 変更 (再記入) されたその値を一時的に保存するレジスタである.

2.3 生成系

文献 (6)のアルゴリズムでは, 要求仕様の 1 つの遷移ごとに各ノードの動作仕様の遷移系列を生成する. 生成系はこのアルゴリズムに従って各ノードの動作仕様を自動生成する.

生成系に対する入力は, 要求仕様 EFSM を記述したファイル (EFSM 型プログラム), リソースの割当てや協調計算システムを構成するノード数を記述したファイルの他, レジスタや入力用変数の型や構造を記述したファイル, 要求仕様で用いた関数を C 言語で記述したファイルの 4 つである[8]. 図 1の要求仕様と 2.2節のリソースの割当てをこの 4 つのファイルで記述すると全体で約 150 行の大きさになる.

生成系は, これらの入力に対して要求仕様 EFSM 型プログラムの構文チェックなどを行なった後, 各ノードの動作仕様 EFSM 型プログラムを出力する.

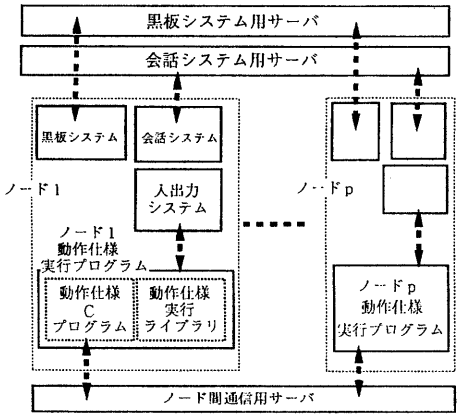

図 2: 実行系の構成

3 協調計算システムの実行系

実行系は生成系から得られた動作仕様群を協調計算システムとして並列に実行する.

3.1 実行系の構成

実行系の構成を図 2 に表す. 1つのノードは, 動作仕様実行プログラム, 支援ツールである黒板システム・会話システムから構成される.

まず動作仕様EFSM型プログラムをCプログラム(動作仕様Cプログラム)に変換する. 我々はこの変換のためのコンパイラをすでに作成している[7],[8]. 動作仕様実行プログラムは, 動作仕様Cプログラムと, 実行に必要な種々の機能を提供する動作仕様実行ライブラリとをリンクすることによって得られる. 動作仕様実行プログラムとユーザとのインタフェースとして入出力システムがある. また全体として協調計算を実現するため, 各ノードの動作仕様実行プログラム間の通信に専用のノード間通信用サーバを用いる.

黒板システムと会話システムは複数ノードの共同作業を支援するツールである. これらを使用することによって共同作業を円滑に進めることが可能となる.

黒板システムは, 複数人が各々キャンバスを持ち, そこに図形や文字を書き込めるシステムである. 1人がキャンバスに対して操作を行なえば, その結果が他のすべての人のキャンバスにも反映される. 全員があたかも共通のキャンバスを使っているように操作が行なえる.

会話システムは, 複数人で同時に会話を行なえるシステムである. 会話権利を得たノードがマイクを使って話し, その音声が他のノードに送信される.

3.2 共同ファイル編集機能

複数人 (複数) で共同作業をする場合, 同一のファイル (テキスト・ビットマップ・線描画) を共同で編集できることが望ましい. ここでは実行系における共同ファイル編集機能について述べる.

例えば, あるノード A がファイルの編集を行なっているとする. 要求仕様で次にノード B がそのファイルを編集することが決まっているとする. ノード A がファイルの編集を終えると, ノード B にはそのファイルのエディタが自動的に起動され, 編集が可能となる. このことを繰り返すことによって同一のファイルを共同で編集する.

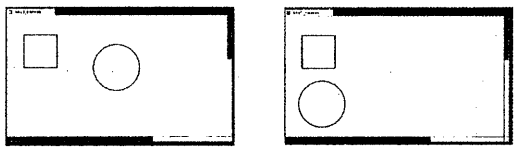

図 3: 線描画編集システムの例

また線描画ファイルの編集に関しては, 同一のファイルを同時に共同で編集することが可能である. この編集は入出力システムで実行される. 今, 複数のノードで線描画ファイルを編集しているとする. あるノードで線描画図形の追加や変更などが行われると, 他のノードへ送られ, この情報を受け取ったノードはその情報に従って図形の変更などを行う. 例えば, ノード X とノード Y の 2 つのノードで共同編集が行われているとする. 今, ノード X が図 3の実行 1 のように円図形を描けば, ノード Y でも同じ図形が描かれる. また, ノード Y が図 3の実行 2 のようにその円図形を動かせば, ノード X でもその図形が動く.

共同でファイルを編集する例として 2 つあげたが, 両者の違いは, 各ノードが順番に編集を行なうのか, それとも同時に行なうかである. また, 前者の共同編集はどんな型のファイルでも可能だが, 後者は線描

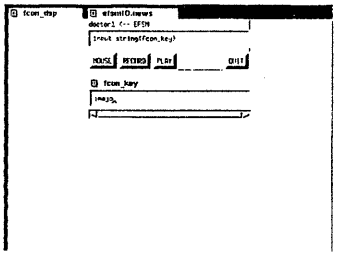

図 4: ノード 1 の名前入力

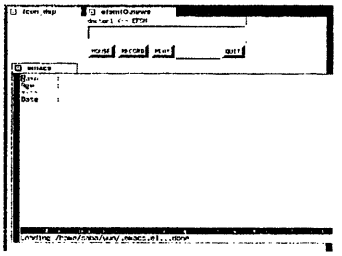

図 5: ノード 1 の初診 (1)

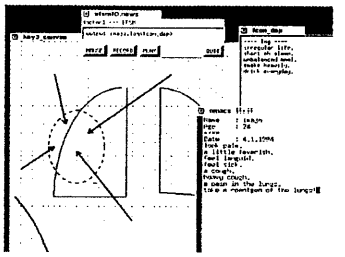

図 6: ノード 1 の黒板システム

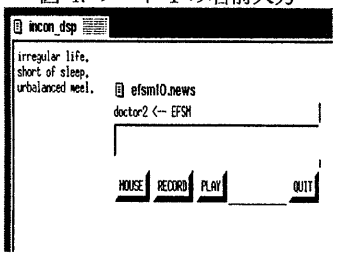

図 7: ノード 2 の問診 (1)

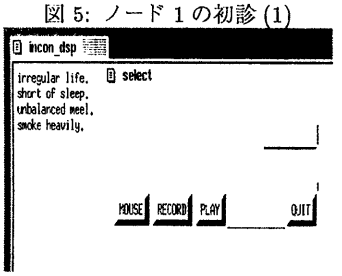

図 8: ノード 2 の問診 (2)

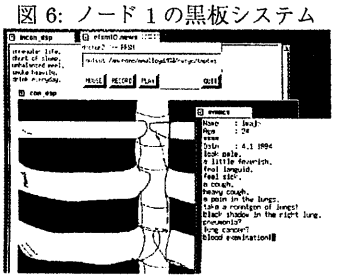

図 9: ノード 2 の診察

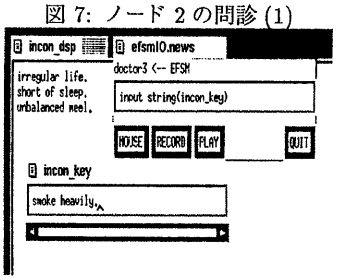

図 10: ノード 3 の問診

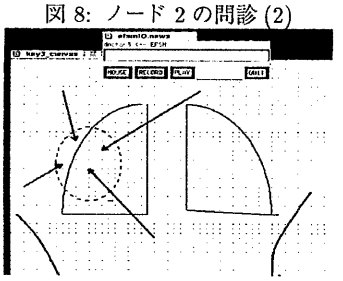

図 11: ノード 3 の黒板システム

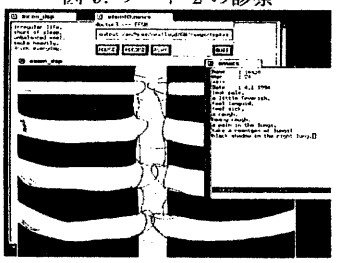

図 12: ノード 3 の検査

画ファイルのみ可能である.

4 協調計算システムの実行例

実行系での実行例を述べる. 図 1の医療診察要求仕様と, 2.2節のリソースの割当てを用いる. また, ここでは状態 i から状態 j への遷移を遷移 $i \rightarrow j$ と略記する.

今, 図 1の状態 0 にいるとする. ノード 1 (doctor1) に割当てられているゲート *fcon_key* で名前を入力するよう要求仕様に記述されているので, ノード 1 が患者名を入力する. ここでは患者名として *imajo* と入力する (図 4). 患者名の入力が終る (遷移 0→1) とノード 1 の画面にカルテ記入のためのエディタ emacs が自動的に起動する (図 5). これはレジスタ *fcon_karte* がノード 1 に割当てられ, 遷移 1→2 でそのレジスタを更新 (ファイル編集) するよう記述されているからである. これにしたがってノード 1 が初診を行ない, カルテを編集する.

遷移 2→3, 遷移 3→2 では問診が行なわれる. 問診用入力ゲート *incon_key* がノード 2 (doctor2) とノード 3 (doctor3) の両方に割り当てられているので, 両者が同時に問診を行なう. そのとき, 問診結果入力の権利取得のためのウィンドウがノード 2 とノード 3 で起動し, その権利を取得したノードだけが実際に問診を行なう. 権利を得られなかったノードは待ち状態に入る.

図 10では, ノード 3 が問診を行なっている. ウィンドウ incon_key が権利を取得したことを表すウィンドウであり, また問診結果入力のためのウィンドウ

でもある．一方ノード 2 は待ち状態に入る (図 7)．問診結果入力の権利を取得できなかったのでウィンドウ incon_key が起動していない．ここでノード 3 が問診結果として "*smoke heavily.*" と入力する (遷移 2→3) と，ノード 2 のウィンドウ incon_dsp に "*smoke heavily.*" と表示され (遷移 3→2)，権利取得のためのウィンドウ select が起動する (図 8)．また問診結果として "*end.*" が入力されると問診は終了する．

遷移 3→4 では制御はまたノード 1 に戻る．問診ログが出力され，カルテ再編集のためのエディタ emacs も起動される．ここでは患者の病気の原因が肺である確率が高いと判断され，レントゲン写真を撮ることにする．よってノード 1 は次に検査を行なうノード 3 と黒板システムを使ってどの部分を写真にとるか確認している (図 6, 11)．

遷移 4→5 ではノード 3 が検査を行なう (図 12)．ウィンドウ exam_dsp にレントゲン写真が出力され，それを見ながら検査結果をエディタ emacs (カルテ) に記入する．

遷移 5→6 ではノード 2 が診察を行なう (図 9)．ウィンドウ con_dsp (レントゲン写真)，ウィンドウ incon_dsp (問診ログ) を参考にしながら，診察結果をエディタ emacs (カルテ) に記入する．カルテの記入を終えると，再検査を行なう (遷移 6→4) か医療診察を終了する (遷移 6→0) か決定する．

5 リソースの割当て変更について

我々のシステムは，要求仕様が同じでも，実現するノード数とリソースの割当てを変えることによって異なる協調計算システムを実現することが可能である．例えばノード数を 2 つとし，ノード 1 には初診，問診，ノード 2 には検査，診察の役割を与える．リソースの割当ては以下のようにする．

ゲート

ノード 1	*fcon_key*, *fcon_dsp*, *incon_key*, *incon_dsp*
ノード 2	*exam_key*, *exam_dsp*, *con_key*, *con_dsp*

レジスタ

ノード 1	*master_karte*, *master_log*, *master_roentgen*, *pat_name*, *pat_karte*, *pat_log*, *pat_roentgen*, *fcon_karte*, *incon_input*
ノード 2	*exam_karte*, *con_input*, *con_karte*

この場合ノード 1 は図 4から図 8, 図 10 の作業を行ない，ノード 2 は図 9, 図 11, 図 12の作業を行なう．

このような変更に対して，生成系の入力の 1 つであるリソースの割当てファイルを変えるだけで対応できる．変更した入力を生成系に与えると CPU 時間約 0.15 秒 (SONY NEWS 5000) で各ノードの動作仕様 EFSM 型プログラムを出力される．これらを実行系によって実行させれば異なる協調計算システムが実現できる．このように我々のシステムは容易に協調計算システムを変更できる．

6 おわりに

本稿では，協調計算システムの要求仕様から動作仕様群を生成し，これらを複数の計算機上で分散実行させるシステムについて述べた．グループウェアに関して在席会議システムなど，特定の共同作業システムを研究・開発している例は多いが，本稿で紹介したシステムは割当て変更を容易に行なえ，それに対応する協調計算システムを簡単に得られる利点がある．また今後の課題の 1 つとしてレジスタ更新における実行系の基本機能の充実等が考えられる．

文献

(1). C. A. Ellis, S. J. Gribbs and G. L. Rein : "Groupware - Some Issues and Experiences", Communication of the ACM, Vol. 34, No. 1, pp.38-58, 1991.

(2). I. Greif and S. Sarin : "Data Sharing in group work", Proc. of the first ACM Conf. on Computer-Supported Cooperative work, pp. 175-183, 1986.

(3). S. R. Ahuja, J. R. Ensor and D. N. Horn : "The Rapport Multimedia Conferencing System", Proc. the ACM Conf. on Office Information Systems, pp. 1-8, 1988.

(4). K. Watabe, et. al. : "A Distributed Multi Party Desktop Conferencing System and its Architecture", Proc. of 9th Int. IEEE Phoenix Conf. on Computersand Communications, pp.386-393, 1990.

(5). R. Probert and K. Saleh : "Synthesis of Communication Protocols: Survey and Assessment", IEEE Trans. Comput., Vol. 40, No. 4, pp. 468-475, 1991.

(6). 岡野 浩三, 今城 広志, 東野 輝夫, 谷口 健一 : "拡張有限状態機械モデルを用いた分散システムの要求仕様から各ノードの動作仕様の自動導出", 情報処理学会論文誌, Vol. 34, No. 6, pp. 1290-1301, 1993.

(7). 今城 広志, 岡野 浩三, 東野 輝夫, 谷口 健一 : "拡張有限状態機械を用いた協調作業向きの計算システム", 情処研報, Vol. 93, No. 58(93-OS-60 , 93-DPS-61), pp. 147-154, 1993.

(8). 今城 広志 : "協調計算システムの各動作プログラムの導出と分散実行系", 平成 5 年度大阪大学基礎工学研究科修士学位論文, 1994.