ソフトウェア工学　73－19
（１９９０．７．２０）

プラガブルなネットワーク・アプリケーション・ツールの開発

葛岡英明、三井博隆、広瀬通孝、石井威望
東京大学工学部

グループウェアにおけるネットワークアプリケーション開発を容易にするための概念と、この考え方に基づいて開発したネットワークアプリケーション記述ツール、CTK について概要、プログラミング方法、応用について述べる。CSCW システムが有効であるための要因として、「いかに短時間システムをセットアップ可能か」という事柄があげられる。著者らはこれを「BPSS」、「プラガブル」という概念で表している。CTK を利用することで簡潔にアプリケーションを記述し、柔軟にプロセスのネットワークを構成することができた。この結果、当研究室内外においてネットワークアプリケーションの数が増大している。

Development of the Programing Tool for Pluggable Network Applications

Hideaki Kuzuoka, Hirotaka Mitsui, Michitaka Hirose Takemochi Ishii
Dept. of Mechanical Enginering for Production, Faculty of Eng.
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113 ,Japan

kuzuoka@ghidorah.ihl.t.u-tokyo.ac.jp
mitsui@ghidorah.ihl.t.u-tokyo.ac.jp

In this paper, the concept of groupware development support tool and the programming toolkit "CTK" which was developed according to the concept are introduced. For CSCW system to be successful, it is important that the system can reach high performance quickly. To express this characteristics, we propose the concept"BPSS" and "Pluggable". Using CTK, we have been able to write network applications quite easily, and able to construct process networks quite flexibly. Using CTK, the number of network applications has been increasing.

1 はじめに

コンピュータ・ネットワークが整備され、保有するコンピュータの台数が増えるにつれて、プロセス間通信を利用したアプリケーションを必要とするような機会が増えつつある。特にCSCW(Computer Supported Cooperative Work)、すなわちコンピュータ、および情報ネットワークを利用して人間の共同作業を支援するという分野が盛んになりつつあるが、CSCW用のソフトウェア(グループウェア[1][3][9])ではプロセス間通信は必要不可欠の要素である。このような状況で、多くのユーザ、研究者がこのようなソフトウェアを記述することが多くなってきている。ところがプロセス間通信を利用したソフトウェアの記述はそれほど容易ではない。それには以下のような理由があげられる。

- 通信をおこなうコンピュータの種類、言語、通信手段によってプログラミングの方法が異なるため、アプリケーションの利用目的、利用状況などに応じて通信の低レベルからプログラミングを開始する必要がある。
- 2つ以上のプロセス間通信を記述するのは難しい。
- そのため、目的とするアプリケーションの開発に非常に時間がかかる。
- 複数のプロセスが関わるため、デバッグが困難である。

我々はこれらの問題点を解決し、プロセス間通信を利用したアプリケーションを簡単、かつ短時間で開発することを支援するためのライブラリ、CTK(Colon Toolkit)、及びツール群を開発した。

本論文ではまず、CTK開発の際の基本コンセプトとなった概念、「BPSS」、「プラガブル」に関して説明し、次にCTKの概要、プログラミング、機能の解説、そしてアプリケーションを紹介する。

2 CTK開発の基本概念

2.1 BPSS

著者らは人間同士のコミュニケーションを達成するためのシステムは、「短時間に高い通信速度に到達することができる」ことが重要であると考えている。このようなシステムの性能に対して、著者らはBPSSという概念を提唱している[5][6][10]。BPSSとはBit Per Square Secondの略である。BPS(Bit Per Second)は通信速度を表現するために通常良く利用される単位であるが、BPSSは通信の開始を決定してから、いかに速く高いBPSを達成可能かという単位、すなわち通信加速度である。この考え方からいうと、たとえ通信速度は低くてもシステムのセットアップにかかる時間が非常に短ければシステムとしては高く評価されることとなる。

システムのセットアップの意味を広くとらえて、システムの開発段階から考えれば、グループウェアなどシステム構築のためのソフトウェア開発を効率化することも重要であると考えられる。事実、実験的なシステム構築の場合には多少性能が悪くとも素早くシステムを作成し、実験を行なうことが必要となる場合が多い。従ってプロセス間通信を行なうソフトウェアの開発においても、短時間に通信速度の高いプロセス間通信アプリケーションを開発可能なツールが望まれる。

2.2 プラガブル

著者らは高BPSSを実現するための概念としてプラガブルという概念を取り上げた。プラガブルとは、ある実行中のアプリケーションあるいはそのネットワークに、別のアプリケーションを容易に接続切離することができることをいう。

たとえば、家において掃除をしたいと思ったとする。そのとき、掃除器を持ってきて掃除したい部屋のコンセントにプラグを差しこむだろう。そして、掃除が終わったらプラグを抜いて掃除器をしまうだろう。その間、家のその他の機能は阻害されない。

人間の行動は一般にDemand Activatedであり要求、欲求にしたがって行動を起すことが多い。プラガブルシステムとは要求、欲求にしたがってある機能を使いたい間だけ使うことをユーザに許すシステムである。そのため、環境を全て持ちあるく「統合環境」システムに比べて軽量かつポータブルとなる。これはまた、環境を持たない既存言語へ組み込めば既存のソフトウェア開発システムにプロトタイプ機能をつけくわえることができるということでもある。

我々はこのプラガブルシステムによって従来構築の困難であったアプリケーションプログラムネットワーク構築が容易にできるようになると考えている。

そもそもプラガブルという断面で見ると、世の中には多くのプラガブルシステムが存在する。ハードウェアはプラガブルにすることでメインテナンスフリーになる利点があり先の家電などもその例である。ソフトウェアシステムでも例えばUNIXのShell(Shell script)では、cat, sort, awk などのコマンドをパイプやリダイレクションを使って組み合わせて使え、これはcat,sort,awkなどといったアプリケーションをプラグインして使っているといえる。また、X-Window systemでは各クライアントがX serverにプラグインし

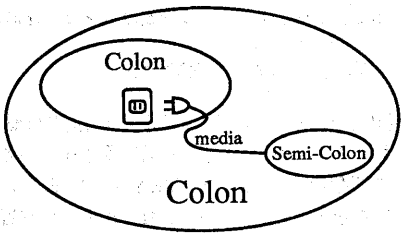

図 1: Colon, Semi-Colon

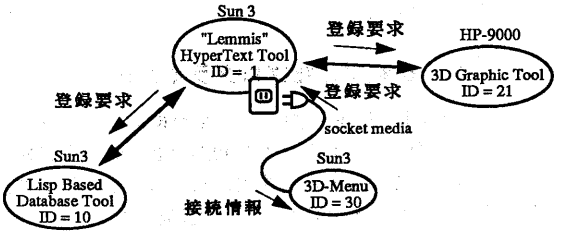

図 2: Media, Colon ID

ている。Macintosh のリソースという概念もプラガブルである。

システムをプラガブルとする利点は、まず BPSS を高くすることができプロトタイピングの武器になること。各アプリケーションをツールとして再利用できること。また、各 CTK アプリケーションはモジュールとして完全に独立しているため明確に仕様を記述でき開発コストを下げることができる、などが挙げられる。

著者らはこの概念に基づき以下に述べる CTK を開発した。

3 CTK の概要

3.1 Colon, Semi-Colon

CTK では複数のプロセスが互いにメッセージを送り合うことによって作業が進行していく。あるプロセスにメッセージが到着するとそのメッセージに対応した関数が自動的に起動される。この、互いに通信を行ないながらある作業を行なうプロセスの一群のことを Colon と呼ぶ。また、そこへプラグインされる一つのプロセスを Semi-Colon と呼ぶ。 各 Semi-Colon は基本的に単独で存在し、動作することが可能であり、自分自身でユーザ・インタフェースを持っていればそれだけでアプリケーションとなり得る。現在のインプリメンテーションでは通信を管理するような特殊なサーバは存在せず、基本的に等価な機能を持った Semi-Colon のみから構成される。

3.2 media

メッセージを交換するための物理的な通信路として、Eathernet のソケット、RS232C、電話回線による login を利用した通信などを利用することができる。CTK ではこれらの通信路を media と呼んでいる。一つのプロセスは複数の media を持つことができ、プログラマはプログラムの中で利用したい media を必要なだけ宣言する。この時以外はメッセージの授受に関して通信路の違いを意識する必要はない。

3.3 Colon ID

Semi-Colon は各々 ID 番号を持っている。また、各 Semi-Colon は自分の持つメディアの先にどの ID 番号を持った Semi-Colon が存在するかを知っている。これによって ID 番号を指定することでメッセージの送信先を同定することができる。

新たな Semi-Colon が接続されるとまず自分が接続されたことをブロードキャストし、各々の Semi-Colon のデータベースに登録させる。次に、接続した相手に対して、他にどの Semi-Colon が存在するかのデータを送ってもらい、自分のデータベースに Colon 内に存在する ID を登録する。接続が切れる場合は、メディアの異常を検知した Semi-Colon が自分のデータベースからその ID 及び利用されていたメディアを削除し、Colon としての動作を継続しようとする。

3.4 複数言語、インタフェースへの対応

CTK は複数の言語、インタフェースに対応している。現在インプリメントされているものの一覧を下表に示す。この特徴による利点を箇条書する。

言語	C, Lisp, Prolog, HyperTalk
インタフェース	X-Window, Starbase(HP), HyperCard
マシン	Sun3, HP9000, PC9801, Macintosh

- Lisp の様なインタプリタは CTK によるアプリケーション開発時のデバッグ作業に有効である。
- 各言語、インタフェースの得意な分野のプログラムを記述し、作業を分担することが可能である。例えば、知識処理は Lisp で記述し、その結果を X-Window 上のハイパーテキスト・ツールに反映させたり、HP9000 上に視覚化して 3 次元表示を行なったりすることが可能である。HyperCard

などはユーザインタフェースのプロトタイピングに有効である。

4 CTKを利用したプログラミング

CTKではBPSSを極力高めるためにプログラマができるだけ簡単にプログラムを記述できることを目標としている。そのために機能的には不自由を感じる部分も多いが、BPSSを高めることを優先した。

4.1 基本的なプログラミング

CTKを利用したCの基本的なプログラミング・パターンと、最も単純なプログラム例を示す。まず、プログラミング・パターンは以下の通りである。

- ctk.hをインクルードする。
- 自分のColon IDを決定する。
- mediaを生成する。
- CTKのイニシャライズを行なう。
- 受け取ることのできるメッセージを登録する。
- メッセージの受信、実行のループに入る関数を記述する。
- メッセージに対応した関数を記述する。
- メッセージを送り出す場合、送る相手のID、メッセージ名、引数を設定してメッセージを送出する。
- ライブラリをリンクしてコンパイルする。

最も簡単なプログラム例は以下のようになる。

```
#include "ctk.h"                /* 1 */

#define MYID 10                 /* 2 */

main()
{
  CtkCreateSock();              /* 3 */
  CtkInitialize(MYID);          /* 4 */
  addMsg("Test", ctkTest);      /* 5 */
  CtkMainLoop();                /* 6 */
}

int ctkTest(org, arg)           /* 7 */
  int  org;
  unsigned char *arg;
{
  /* 8 */
  sendMsg(org, "reply", strlen(arg)+1, arg);
}
```

コンパイルは以下のように行なう。

```
cc -c test.c
cc -o test test.o ctk.o ctkSock.o ctkLib.o
```

以下、一部の項目に関して解説する。

ID番号の決定 同一のSemi-Colonを複数起動することもあり得るため、同一のID番号が一つのColon内に複数存在することが許されている。しかしながら異なるSemi-ColonのIDが互いに重複しないことが好ましい。また、ID番号の0番はブロードキャスト用のIDとしてあらかじめシステムに予約されている。

mediaの生成 Colonに対してどのメディアで接続を行なうかによって、利用する数だけメディア生成用の関数を呼ぶ必要がある。メディア生成関数としては、例えばCtkCreateSock(), CtkCreateTtya()などが既に準備されており、ソケットを利用するか、RS232Cを利用するかなどによって選択する。

CTKのイニシャライズ 関数CtkInitialize(int MYID)によって実行される。この関数で自分のIDの決定、自分のIDのブロードキャスト、他のID情報の獲得、全てのSemi-Colonが共通に理解できるデフォルトのメッセージ群の登録が行なわれる。デフォルト・メッセージとは、個々のSemi-Colonは接続時にColonとして振舞うことが可能となる重要なメッセージである。

メッセージの登録 addMsg(char *msg, int *func)によって理解可能なメッセージと起動される関数を登録する。msgは送られるべきメッセージ名、funcはそのメッセージによって実行されるべき関数である。呼び出される関数は、メッセージを送ってきた相手のIDとメッセージのアーギュメントを引数として渡される。アーギュメントはバイナリ・データをそのまま送るものであるため、どのような型のデータでも授受することができるが、Lisp, Prologなどとの通信を考慮する場合は数値も文字列に変換して送る場合が多い。

メッセージ処理ループ このループではまず自分の持つ全てのメディアを監視し、入力のあったメディアからメッセージを取り出す。自分宛のメッセージであれば対応する関数を実行する。自分宛でなければ適当なメディアへそのメッセージを転送する。

メッセージの送出 メッセージは関数sendMsg(int dest, char *msg, int arglen, unsigned char *arg)によって送出する。destは宛先、msgはメッセー

ジ名、arglen はアーギュメントのデータ長、arg はアーギュメント自身である。

ライブラリのリンク メディア毎にライブラリが用意されており、その時利用するメディアに応じてライブラリを選択する。新たなメディアを利用する必要が生じた場合は、そのメディア用の新たなライブラリを記述るだけでよい。

5 CTK の高度な利用法

5.1 名前によるメッセージ送信

メッセージの送出先は通常 ID 番号によって指定するが、sendMsgBN() という関数によって、宛先として名前を利用することができる。このためにはまず、ctk.hosts ファイルの中に ID 番号とそれに対応する名前を登録する必要がある。この名前は UNIX のディレクトリ名のように"/"で区切ることで階層構造を設定することが可能である。例えば"ctk/sc1", "ctk/sc2" という名前を持った二つの Semi-Colon が存在したとすると、sendMsgBN("ctk", msg, arglen, arg) を実行することで"ctk" という名前を階層に持った両方のプロセスに対してメッセージが送られることになる。このようにして、名前を階層的につけておけば、あるグループに対してメッセージを送ることが可能である。

5.2 リダイレクションとフィルタ

CTK にはメッセージの送り先を実行中に変えられる機能がある。これを利用することで動的なリダイレクションを行なえる。すなわち、センサから得たデータをコントローラ A から、コントローラ B へ切替える様なことができる。また、リダイレクションによってフィルタプログラムも作ることができる。

5.3 モニタリング

デバッグ時など、Colon 内部の接続状況、メッセージの交換状況のモニタリングが必要な場合がある。CTK は Colon 内に同一の ID 番号が存在することを許すが、モニタリングを行なうためには個々を区別できる必要がある。そこでこの個別のモニタリング ID を設定するためのメッセージが全 Semi-Colon 共通のメッセージとして登録してある。また、この個別 ID には media の接続情報が含まれているため、Semi-Colon 同士の接続情報も同時に取得することが可能となっている。

メッセージの送出情報をモニタに報告することを命じられた Semi-Colon はメッセージを送出する毎にモニタに対してその情報を提供する。従って、モニタは

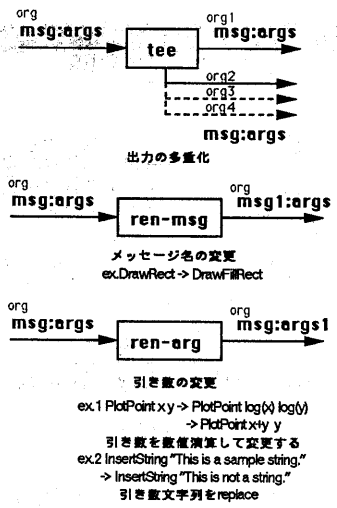

図 3: フィルタ

リアルタイムにメッセージの送出状況をモニタリングすることができる。

5.4 Help

CTK は、addMsg() によって登録されたメッセージのヘルプを表示する機能を持つ。各 CTK アプリケーションはヘルプメッセージを受け取ると、ヘルプを処理する CTK ツール (ctkHelper) に対して自分の ID と自分の理解できるメッセージを送る。ctkHelper はそれを表示用の CTK ツール (現在は Lemmis) に対してそれを文字列として表示させる。これによって、CTK ユーザは自分の使いたい CTK アプリケーションのオンラインヘルプを見ることができる。

6 CTK アプリケーション

CTK を利用した CTK アプリケーション開発支援ツール、また実際のアプリケーション作成例、使用例を紹介する。

6.1 BodyBuilder

CTK を利用するに従って以下に示すような、ユーザが理解しづらい部分が明らかになってきた。

- ソケットのサーバ、クライアントモデルを理解していない場合など、どの様なメディアを利用し、どの様にして接続すれば良いかがわからない。
- 各種ツールは別々の人間によって作成されることが多いため、既に存在する Semi-Colon を利用する場合、どの様なメッセージ名を理解できるのか、引数は何であるかがわからない場合が多い。

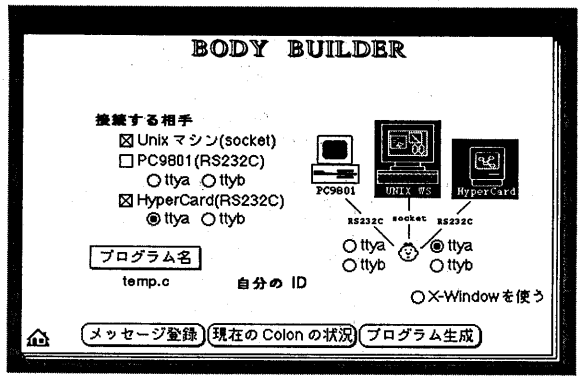

図 4: ソースコード自動生成インタフェース

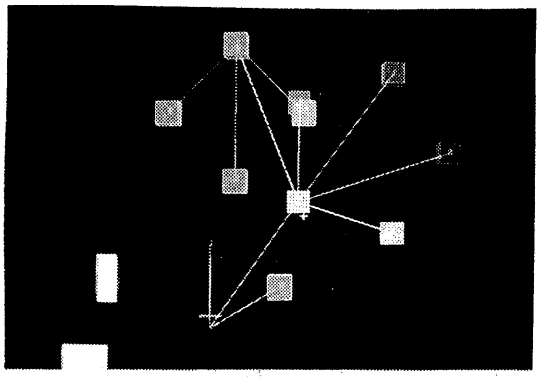

図 5: CTK Monitor

- メッセージのシークエンスの把握が困難になるため、あまり大規模なネットワークは構成しづらい。

BodyBuilder はこのような問題点を軽減し、CTK の BPSS を増大させるための各種支援ツール群である。

6.1.1 ソースコード自動生成

先に述べた通り、CTK のプログラミングは非常に単純であり、定型的な記述がほとんどである。このようなプログラミングにはプログラムのテンプレートを自動生成してユーザに提供することによって、プログラマの負担が軽減されると考えられる [7]。そこでメニュー形式で CTK のソースコードのテンプレートを生成するためのツールを作成中である。このツールでは、何を利用して通信を行なうか、通信の相手は何であるかなどをによって、視覚的にメディアの選択が行なえる。また、メッセージの追加もメニュー形式で行なうことができる。この時、メッセージに対するコメントも付加することが可能である。データベースと接続することによって、他の Semi-Colon のメッセージを見ることも可能である。ID 番号の重なりもこの時点でチェックされる。最終的にはソースコードのテンプレート、及び UNIX であれば Makefile も自動生成される。このツールを利用することで、違う人間が記述したソフトウェアでも統一的なフォーマットでプログラムが記述されることになり、ソースコード・レベルでの理解度が高まることとなる。

6.1.2 CTK Monitor

CTK の動作を確認するために、CTK 内の接続状況、及びメッセージ授受の状況をモニタリングしたい場合がある。そこでグラフィック・ワークステーション上で CTK を視覚的にモニタリングするツールを作成中である。このツールでは Semi-Colon をノード、メディアをアークとして表現してある。Semi-Colon の追加、削除に応じてリアルタイムに表示が変更される上、メッセージの発信状況もノードとアークが点滅することによってリアルタイムに表示される。CTK では各種メディアが混在するため、例えば RS232C と Eathernet では通信路の容量が大きく異なる。そのためメディア毎のメッセージの集中度を知りたい場合がある。そこでネットーワークを発熱器にたとえることとした。すなわちアークを抵抗、そこを流れるメッセージを電流とし、抵抗値はメディアの容量に応じて変えておく。メッセージの流れに応じてアークは発熱するが、時間の経過とともに放熱していく。この温度に色を対応させることによってどこにメッセージが短時間内に集中しているかを視覚的にモニタリングすることができる。今後の発展として、Semi-Colon でのメッセージの処理時間を考慮してノードにも温度を持たせさらにそのプロセスと関連の深いプロセスとの間に伝熱性のアークを張ることによってプロセス群の負荷を表現することも可能となる。

6.2 Lemmis

著者らは CTK のアプリケーションとしてハイパーテキストインタフェースサーバ Lemmis を開発した。Lemmis は X-Window 上で動作するマルチディスプレイ、マルチウインドウをサポートするインタフェースサーバである。Lemmis は、拡張可能なプログラマブルインタフェースを提供し、ユーザに可能な操作は全て C の関数および CTK メッセージとして用意されプログラマブルになっている。CTK の組み込みによって他のアプリケーションとの外部インタフェースを提供しているのが特徴であり、これによって、異言語間、異機種間の接続が容易となった。即ち、Lemmis を起動しておけば、CTK を組み込んだ任意の言語で、同一のマルチウインドウ、マルチディスプレイインタフェースを使うことが可能である。

Lemmis の基本的構成要素はカードである。各カードはテキストエディターを持ち、ユーザは任意のカー

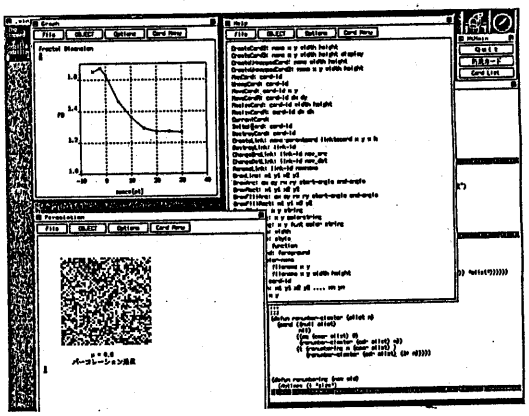

図 6: Lemmis(と CTK の Help 表示)

ドを生成、削除でき各カードにたいしてテキスト編集、図形描画ができる。またカード間に Global to Local なリンクを張ることができる。

テキスト編集機能
- NEmacs のサブセット

図形描画機能
- 基本図形の描画
- ビットマップのロード、セーブ

カード処理機能
- カードの作成
- カードの削除
- カードのマップ、アンマップ
- カードの位置、大きさの変更、カード名の変更

リンク処理機能
- リンクの作成
- リンクの削除
- リンク先、リンク元の名前、位置、大きさの変更

Lemmis は以上の機能を持つが、その全てに対して CTK のメッセージが用意してあり、他のアプリケーションから容易に利用することができる。

具体的には、例えばテキスト編集機能が外部インタフェースとして用意されているので、Emacs の様に拡張が可能である。ユーザの可能な操作はプログラムでも可能なので、一連の操作をプログラム化することが容易である。これは、拡張されたキーボードマクロ機能である。また、描画機能を利用したグラフ作成ツールも作成した。現在作成したグラフツールは高速化のため Lemmis 内部に組み込まれている (C の関数としても外部インタフェースを提供しているため C で記述して CTK を利用しなければ高速化できる)。

6.3 加工の臨場感通信システム

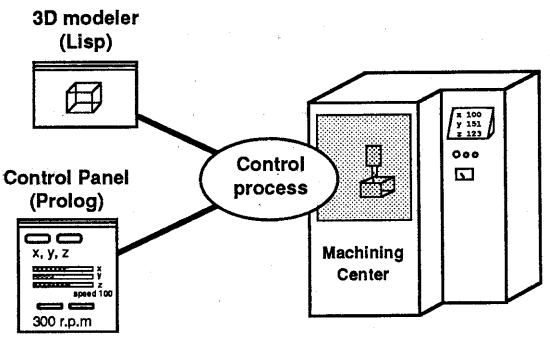

図 7: 加工の臨場感通信システム

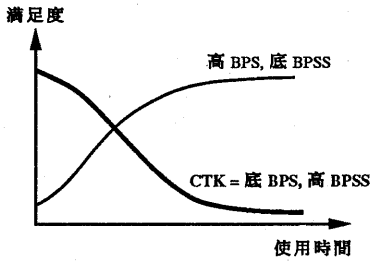

図 8: BPSS と BPS

マシニング・センタ (MC) の遠隔操作の実験のためのシステムにも CTK が利用されている [8]。操作パネル (Prolog)、MC の制御プロセス (C)、3D グラフィック・モデラ (Lisp) が通信を行なっており、さらには音響インタフェース、カメラのコントロールプロセスなども統合される予定である。

7 結論

7.1 BPSS と BPS

現在の CTK を利用している経験から、BPSS の高さは実感している。しかし通信量の増加、ネットワークの拡大に伴って BPS が減少するのも事実である。著者らは BPSS と BPS との関係を以下のように考えている。

すなわち、BPSS は高いが BPS が低い場合は初期の一次的な利用には満足できるが、定常的に利用するには時間が経過するに従って満足度が低下する。逆に BPSS は低いが BPS が高い場合は一次的な利用には適さないが長時間、定常的に利用する場合には適する。CTK は前者の特徴を持っていると思われるため、研究開発用のためのプロトタイピングなどに適していると思われる。今後 CTK の BPS を向上させることによっ

てさらに強力なツールとする予定である。

7.2 CTK システムの特徴

CTK を利用することで以下に挙げる利点がある。
・システム記述とインターフェース記述が独立して行なえるので、再利用が容易である。
・言語、機種を選ばないので、各マシン、言語の得意な分野を有効利用できる。これもまた再利用に有効である。
・インプリメントが容易でかつ既存のアプリケーションに ctk を組み込むことも容易であるため従来の資産を有効活用できる。
・プラガブルであるため部品の取り替えのイメージで機能強化を行える。今後のコンピュータシステムはこのプラガブルという概念を前面に押し出して設計されるべきだと我々は考える。
・既存のソースコードに CTK の組込を簡単に行なえるので、既存のツールを有効利用したり、とりあえず CTK をあまり意識せずにプログラムを記述しておける。

7.3 おわりに

この CTK やプラガブルという概念自身は独創的なアイディアではないが、CTK によって研究室内の各ソフトウエアプロダクツの再利用が急速に促進された。実際 CTK を利用したアプリケーションとしては Sun workstation 上の C で記述された Hypertext システム、LISP で記述された DB、PC9801 上の画像取りこみシステム、HP9000 上の 3D Graphic システム、Macintosh の Hypercard、マシニングセンタの遠隔制御システム、全て CTK によって接続可能であり、現在複数接続して利用されている。この CTK によるアプリケーションは、コンピュータの立場から高度で複雑な機能を持つことよりも、人間の立場から単純で理解しやすいことを選択した好例である。このような例は Basic, HyperTalk などにも見られるが、CTK はネットワーク型アプリケーションへの適用例である。

8 謝辞

本研究にあたり、東京電力制御研究室の名井、甘利両氏、および石井研究室の小池、木島両氏の御協力に感謝します。

参考文献

[1] J. Grudin, "Perils and Pitfalls", *BYTE*, December 1988, pp.251-264.

[2] H.Hirose, T.Myoi, H.Amari, K.Inamura, L.Stark, Development of visual 3D virtual environment for control software., *Human Machine Interfaces for teleoperator and virtual environment*, 1990.

[3] 石井裕, "グループウェアとマルチユーザインタフェース", 電子情報通信学会 OS89-53, 16 March 1989. pp.19-24.

[4] 石井威望, 広瀬通孝, 葛岡英明, "リモート・コラボレーション・システム", 計測自動制御学会第4回ヒューマン・インタフェース・シンポジウム論文集, 1988.

[5] 石井威望, 広瀬通孝, 葛岡英明, "多人数作業における協調", 計測自動制御学会第9回自律分散システム研究会講演論文集, Tokyo, 1989. pp.5-6.

[6] T.Ishii, M.Hirose, H.Kuzuoka, T.Takahara and T.Myoi, Collaboration System for Manufacturing System In the 21st Century., *Proc. MSET21*, Tokyo, 1990. pp.295-300.

[7] 石井威望, 広瀬通孝, 小池英樹, "プログラム読解支援に対する自然言語理解的アプローチ", 情報処理学会ソフトウェア工学研究会, 1988.

[8] M.Mitsuishi, Y.Hatamura, T.Nagao, H.Inoue, Development of a user friendly manufacturing system, *Proc. MSET21*, Tokyo, 1990. pp.167-172.

[9] 中村正弘, "グループウェア: 電子メール普及を受け模索始まるグループ・コミュニケーションの姿", 日経コンピュータ, 1989.7.31, pp.48-65.

[10] 大村裕子, 葛岡英明, 高原勉, 広瀬通孝, 石井威望, "生体情報を利用した画像コミュニケーション支援", 情報処理学会ヒューマン・インタフェース研究会, 1990.

[11] 佐藤洋一, 小池英樹, 広瀬通孝, 石井威望, "Multi-dimensional Visualization Tool の開発", 情報処理学会ヒューマン・インタフェース研究会, 1990.