ビジュアルな AP 作成ツールを考える

昭和 63 年 11 月 4 日
Hirata Software Technology Co.,Ltd
古閑 幸一

1. はじめに

筆者のところ(HST) では、ビジュアル MMI を持つ AP の開発を行なう機会が最近多くなってきている.
例えば、FA ラインの生産管理ソフトである. これはラインの物流を管理するもので、コントローラから読み込んだ機械データをリアルタイムに画面(「Fig 1. FA ライン用モニタ」参照)に表示し、ラインの稼働状態を確認するというものであった. (この FA ラインモニタは X ウィンドウを利用し実現されている.)

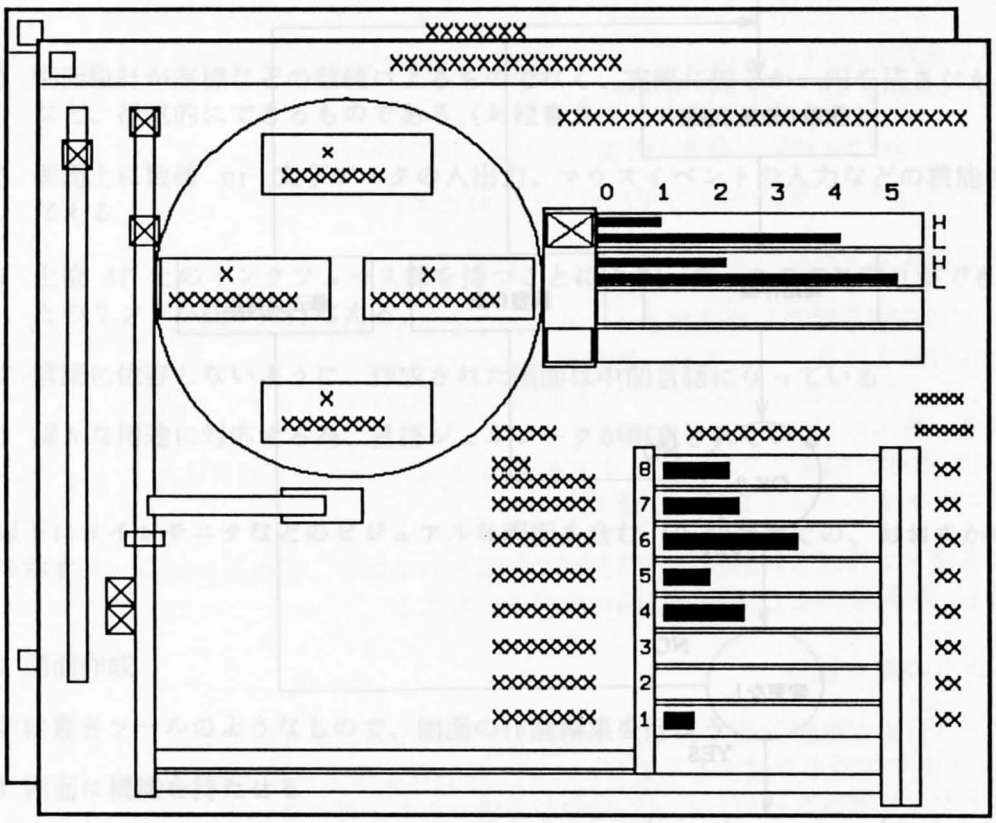


Fig 1. FA ライン用モニタ

FA ライン管理ソフトを作成して感じたことから、ビジュアル性を必要とされる AP 開発において、どのようなツールを用意しておけばよいか私なりに考えてみた.

2. FA ライン管理ソフトを作成してみて

FA ラインのモニタ画面においては、あらかじめ X ライブラリ関数を使ってモニタ画面作成の為のアプリケーション関数を用意し、作成を行なった.
これらのアプリケーション関数を利用することにより、X ライブラリ関数を意識することなく、ラインモニタの為のウィンドウの作成、制御が行なえた. また、グラフィック・ライブラリを変更した場合でも AP 側にはなにも影響しないというメリットがあった.
しかし、この場合でもモニタ画面を設計する時、数値座標によって画面設計を行なう為、全体のレイアウトを調整するのに時間がかかったり、変更などが発生するとさらに時間がかかったりで、あまり効率のよいものではなかった.† (「Fig 2. モニタ画面作成作業フロー」参照)

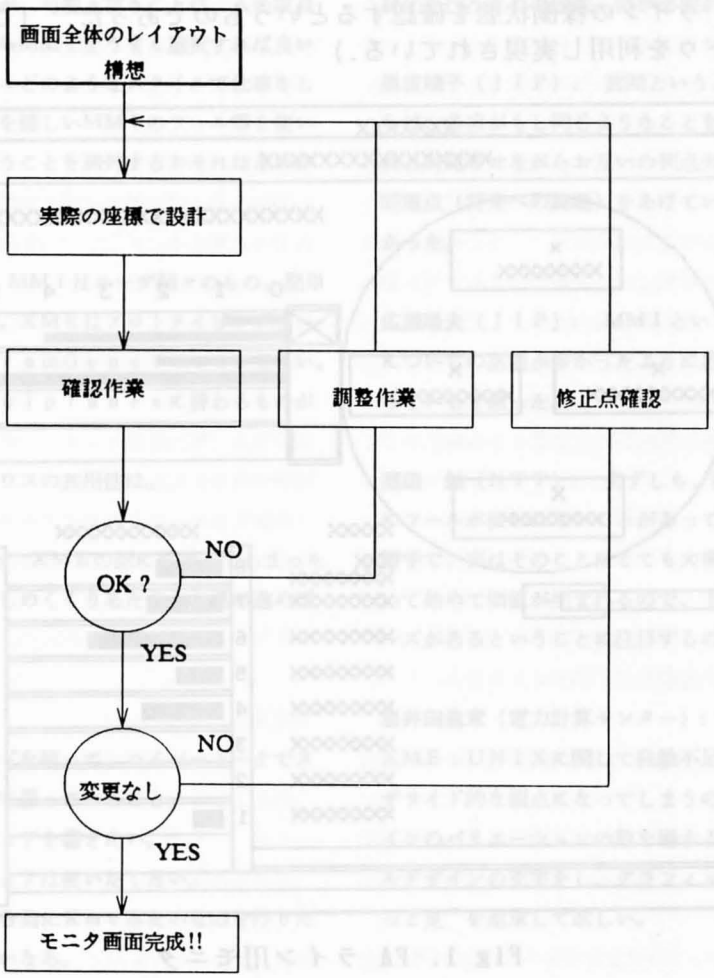


Fig 2. モニタ画面作成作業フロー

今回作成したモニタ画面などのように、ウィンドウ関数を利用したビジュアルな画面作成をもっと簡単に、効率よく開発し、それを AP 化できるようなツールはできないだろうか ?

† 事実、何度も変更依頼があり、その度に座標とのにらめっこであった.

一方、作業の効率化をはかりたいことにドキュメント作成がある． 今回 FA ライン管理ソフトの作成と並行して、マニュアルを作成した．
マニュアルには実際の処理イメージを具体的にわかってもらう為、モニタ画面を多く取り込んだ． マニュアル作成では、モニタ画面を実際にディスプレイに表示し、それをビットマップデータにおとし、さらにそれを roff のデータに変換するという手段で画面データを取り込んだ．
これでは、実際のモニタ画面が変更される度に、同じ作業を繰りかえさなければならない．また実際のモニタ画面とマニュアルに示されているモニタ画面が違っている、などの問題が発生する．†

モニタ画面の AP 作成が終了すると、マニュアルに盛り込むモニタ画面のデータも出来上がっている、というようなツールはできないか ？

3．どのようなツールが欲しいのか

私の考えているツールでは、おおまかには以下のようなものができればいいと考えている．

(1) 画面設計が座標などの数値によるものでなく、実際に線とか、円を描きながら行なうなど、視覚的にできるものである．(お絵書きツールみたいなもの)

(2) 画面上に数値 or 文字データの入出力、マウスイベントの入力などの機能の設定が行なえる．

(3) 上位 AP とのインタフェース部を持つことにより、データのやり取りができ、上位 AP とのリンクが簡単に行なえる．

(4) 言語に依存しないように、作成された画面は中間言語になっている．

(5) 様々な用途に対応する為、言語ジェネレータが用意されている．

以下にラインモニタなどのビジュアルな画面を含む AP 開発までの、おおまかな作業の内容を示す．

(1) 画面作成

お絵書きツールのようなもので、画面の作画編集を行なう．

(2) 画面に機能を持たせる

この作業は画面作成の為のお絵書きツール上で行なう．

a．数値、文字データの表示
数値データ、文字データなどが上位 AP から渡される場合、それらを表示する位置、属性(文字 or 数値)、上位 AP とインタフェースのための名前を設定する．

---

† 筆者は既に経験済みである．

b. キーボード入力

数値データ、文字データなどをキーボード入力し、入力されたものの表示位置、属性(文字 or 数値)、上位 AP とのインタフェースのための名前を設定する.

c. マウスイベントの設定

画面にマウスイベントを取り込む機能が必要とされる場合、表示するマウスボタンの種類を選択し、表示位置を設定する. 次にマウスのボタンがクリックされた時返すイベントを設定し、上位 AP とのインタフェースのための名前を設定する.

上位 AP から画面に持たせた機能を使用する場合は、各々の機能につけた名前を関数形式で呼ぶことで行なわれる.
例えば、作成された画面に上位 AP から渡される数値データを表示する機能を持たせるとする.

その機能名を valDataDisp という名前にしたとする. その場合、上位 AP から実際に画面の AP へ渡すには上位 AP で

```
valDataDisp(数値データ);
```

というように、関数形式で呼び出してやればよい.

(3) 使用目的にあったソースにジェネレートする

(2) までの作業が完了しお絵書きツールを終了すると作成された画面が中間言語の状態でファイルに作成される. これを、任意のソースにジェネレートするジェネレータを用意する. 例えば、中間言語で作成されている状態のものから X ウィンドウ関数にジェネレートしたい場合、専用のジェネレータに通す.
また、roff の図形編集プリプロセッサである pic のソースが必要な場合、 pic のソースをジェネレートするジェネレータに通す.

ツールを使って作成した画面は中間言語で作成されており、使用目的にあったジェネレータを用意し、利用することで使用目的は広がると考える.

4. これからの取り組み

このツールはまだ構想段階であり、これから実際の作成に取り組んでいこうと考えている. ビジュアルな画面を作成する為のツールができれば現在よりは手軽に X ウィンドウ関数などのウィンドウ関数を利用した AP 開発が可能であると考えている.
また、用途も簡単なツールを作成した場合の MMI の作成、数値データをビジュアル化する為のテストツールなどの用途があると考える.

今後このツールを開発していく場合、画面上に持たせる機能の道具化をはかっていきたいと思う.(数種のマウスボタンの用意だとかグラフ、表作成の為の道具類の用意等)

これからの課題としては以下のものがある.

(1) ラインレイアウトのように、レイアウト図等を作成する場合、いかに簡単に便利に作画編集することができるか.(いかに便利なピクチャーエディタを実現できるか)

(2) 画面上に持たせる機能と、上位 AP でどれだけ多くの情報のインタフェースができるか.

(1) については既存のお絵書きツールなどを参考とし、作成していくつもりである. (2) についてはどういう情報が必要であるか検討し、ツールの操作自体が複雑なものとならず、使い勝手のあるものとなるようにしていきたい.

5. 最後に

MMI 作成支援の為のツールを開発することは、作業効率の向上などを考える場合必要となってくる. しかし、それを利用する側が理想の MMI というものを持ち、操作性などを考えて MMI を設計しなければ、本当の意味での MMI の向上はできないと考える.
また、そういう利用者に使ってもらえるツールは幸せであり、ツール自体もいいものになっていくと考える.

4th SEA EWS ではここらあたりの話もまじえて論議したいと考えている.

XMEによる作業環境インターフェイスの改善

柳瀬　健一

(株)ケーシーエス
ソフトウェア事業部　ソフトウェア技術部
yanase@kcs.junet

1　はじめに

開発環境はTSS+キャラクタディスプレイからワークステーション+ビットマップディスプレイの時代へと移って来た。昔は高価であったこれらワークステーションもこの頃比較的安価に手に入る。

これらワークステーションの多くはUNIX+マルチウインドウで基本システムが構成されている。が、UNIXが提供しているユーザインターフェイスは基本的にコマンド形式であり、スクリプトを用いたとしてもメニュー形式のインターフェイスが限度である。決して万人にとって使いよいインターフェイスとは言えない。

マルチウインドウも作業環境を格段に向上させるが、そのままではキャラクタディスプレイをn台専有しているのとかわりはない。

マルチウィンドウ、マルチフォント、マウス等の環境を揃えていても何ら手を加えなければ、満足な機能を発揮するには至らない。

もっと視覚的サポートを加味した初心者にも使いよいインターフェイスはないものだろうか?
このような問題意識を持っている所に、一つの改善提案とも言えるユーザインタフェイス構築ツール『XME』を(株)SRAの酒匂寛氏が紹介してくださった。

以下は、この『XME』を使用した作業環境の改善の事例であり、そのような観点から『XME』の使用感等を整理したものである。

2　「XME」とは

XMEは(株)SRAの酒匂寛氏(sakoh@sra.junet)が開発されたX-Window環境下で作動するユーザインタフェイス構築ツールである。termcap型のファイル(定義ファイル)を記述することにより、パネル(ウインドウ)をデザインすることができ、手続き制御、プロセス制御、通信制御を加味することができる。

XMEの詳細については第三回SEA環境ワークショップ@長野等において酒匂氏が発表されておられるのでそちらの資料をご参照願いたい。

3　「XME」の適用例

以下の3.1~3.4は『XME』を用いた私の適用事例である。

3.1 xmenetwork(図3.1)

ネットワークを介したリモートホストへのアクセスを管理するツールで、rlogin,tip等のコマンドを呼び出し実行する。
例えば「大阪大学」のボタンをクリックすると自動ダイヤリングされ公衆回線を介して阪大サイトと接続することができる。

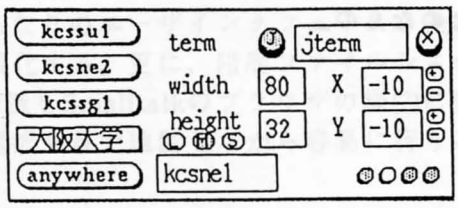


図3.1

3.2 xmepg(図3.2)

プログラミング作業に必要なエディタ、コンパイラ、デバッガ、マニュアル参照等のUNIXコマンドを統合。

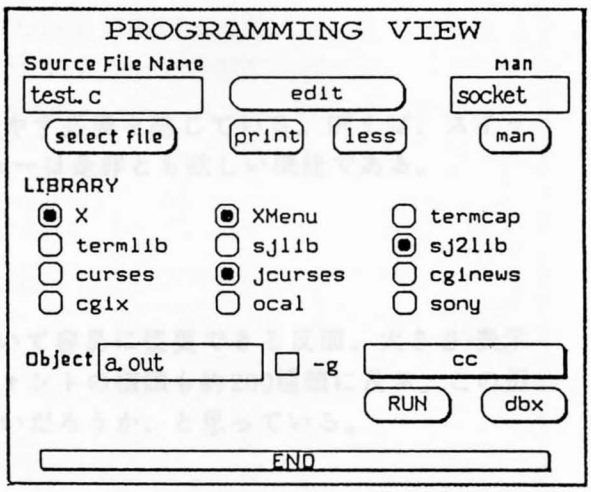


図3.2

3.3 xmejunet(図3.3、図3.4)

JUNETで提供されている電子メール及びニュースの機能をとりまとめたもの。
図3.3のメニューパネルで目的の機能を選択すると、作業目的に応じたパネルをオープンする、等の機能がある。図3.4は「jnews」を選択した例である。

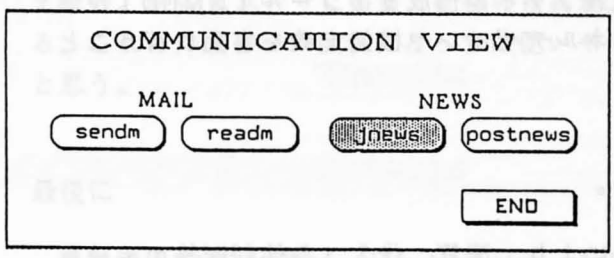


図3.3

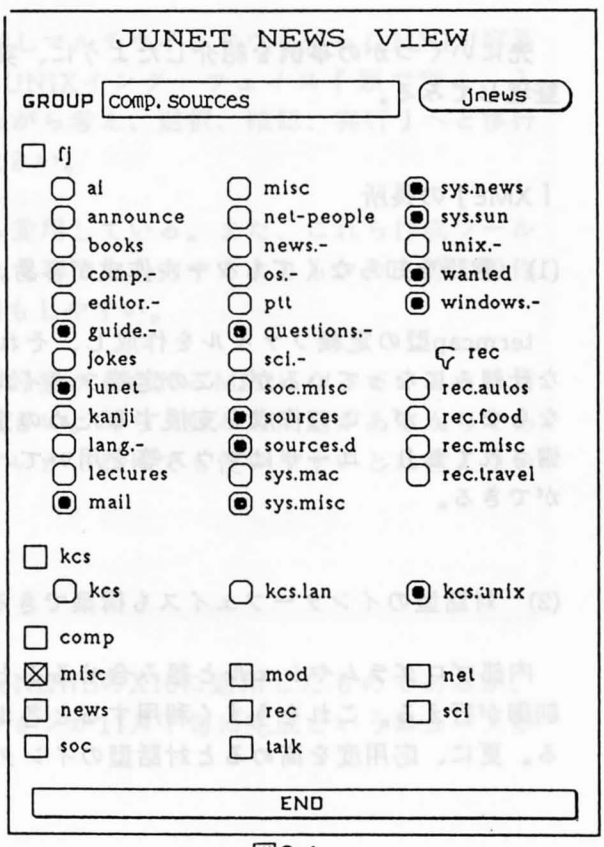


図3.4

3.4 xmemail(図3.5)

社内向け電子メールの送受信機能をとりまとめたもの。

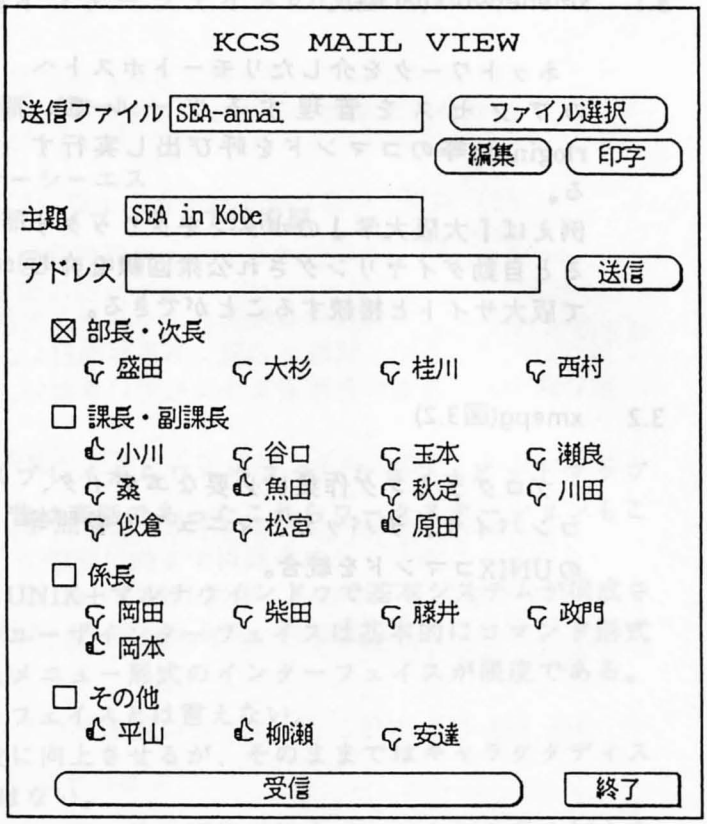


図3.5

4 「XME」の使用感

先にいくつかの事例を紹介したように、実際に『XME』を使用してみた上での感想を整理してみる。

4.1 「XME」の長所

(1) 言語を知らなくてもツール作成が容易。

termcap型の定義ファイルを作成し、それを『XME』がインタプリタで実行するような仕組みになっているが、この定義ファイルは非常に難解な記述言語で作成しなければならない。が、この作成を支援するための定義ファイル作成支援ツール『XMED』が準備されており、ユーザはマウス等を用いてパネルデザインや機能定義を容易に行うことができる。

(2) 対話型のインターフェイスも構築できる。

内部プログラムやシェルと組み合せることにより、演算、代入、条件判断等の手続き制御が行える。これをうまく利用すると各ボタン等にチェック機能を与えることが出来る。更に、応用度を高めると対話型のインターフェイスの構築も可能である。

(3) コマンド等の呼び出しとオプションの指定に関する効果(可視化)。

既存のUNIXインターフェイスではユーザが頭の中でモヤモヤと考え、整理し、コマンドを入力し、目的の作業を実行していた。このあたりのユーザインタフェイスを先に述べた事例のように「XME」はユーザに対して可視化する。更に、階層ファイルシステムも可視化することができる。ファイルシステム構造をSmalltalkのブラウザの様(?)に表示し、ユーザはディレクトリの移動、ファイルの選択を目で確認しながら容易に行うことができる。

4.2 「XME」の短所

(1) パネルデザインのバリエーションの不足。

パネルデザインのバリエーションをもっと増やす必要を感じている。例えば、スイッチは4種類しかない。また、ポップアップメニューは是非とも欲しい機能である。

(2) パネルデザインはたいへん難しい。

パネルデザイン(画面設計)は「XMED」を用いて容易に変更できる反面、大きさ・表示位置等はドット単位で指定することができ、フォントの種類も約200種類に及ぶ。この辺りの支援レベルは更に工夫を加えるべきではないだろうか、と思っている。

5 まとめ

「XME」の最大の魅力は、既存のツールに対しマルチウインドウを使ったMMIが容易に作成できるところである。また、今までのUNIXインターフェイス「頭で考え、入力、実行」を事例に見られるように「目で見ながら考え、選択、確認、実行」へと移行することができ、操作環境の改善には効果が大きい。

私は今回紹介した作成ツールなどを日頃から愛用している。また、これら作成ツールは社内デモ等において多いに役立っている。コマンドで入力するよりもマウス等で操作した方が参加者によりアピールできるし、説明もしやすい。

しかしながらここで紹介した事例は誰にでも好意がもたれるMMIになっているとは言えない。MMIはユーザによって仕様の差異が大きい。即ち、好みの問題があるからである。これら作成ツールは私にとっては使いやすいが、誰にでも使いやすいとは言えないと思う。

6 最後に

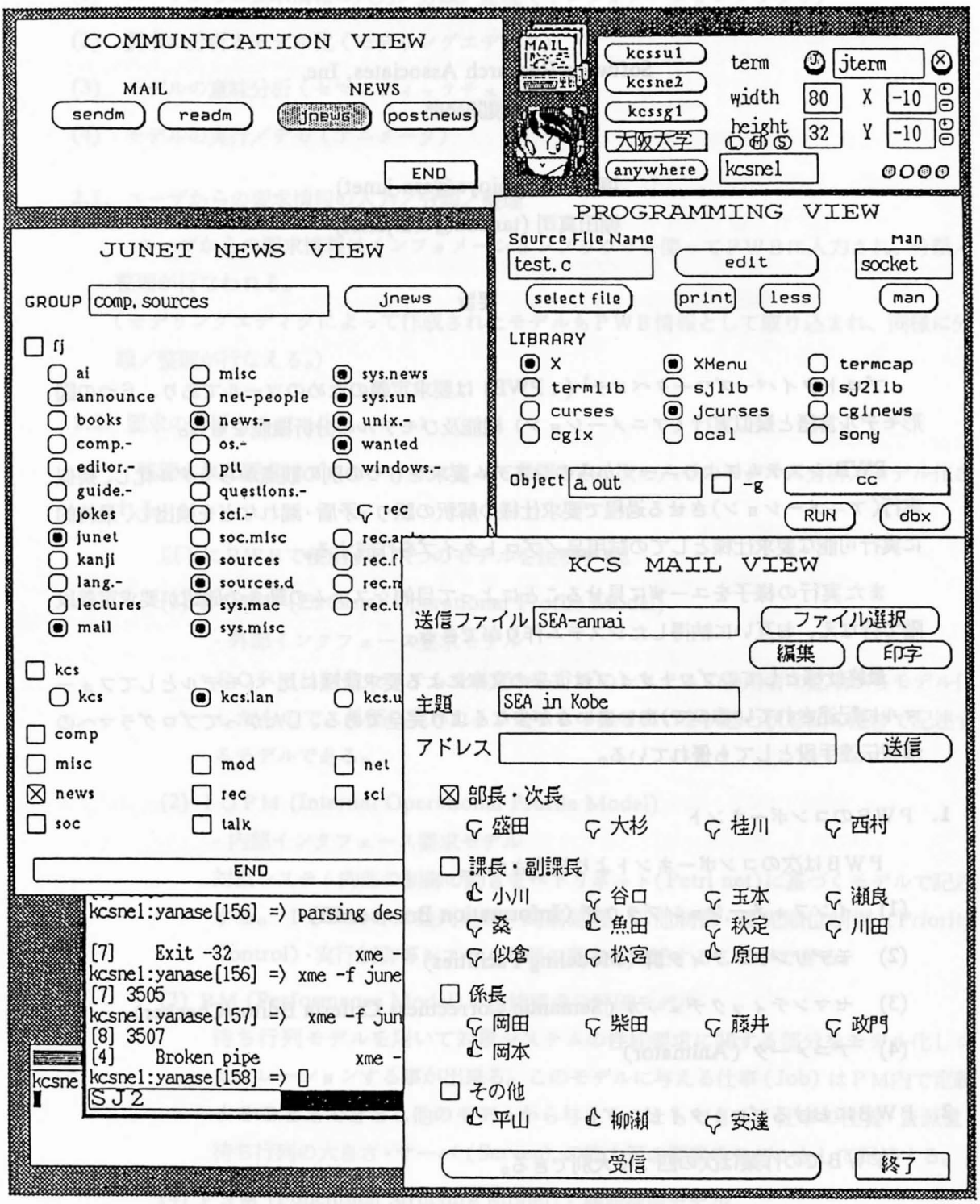
現在の「XME」(現バージョン)はソニー社製NEWSのX10に適用したものであるが、X11全てで起動可能な「XME」のベータバージョンが11月下旬に完成というニュースがあり、楽しみにしています。

7 謝辞

今回事例発表の機会を与えてくださったSEA及び、私に『XME』を紹介して下さり、更に、『XME』のみならず多方面にわたりご指導を頂いております(株)SRAの酒匂氏に心よりお礼を申し上げます。また、今後『XME』の改良、拡張に何らかのお役に立てれば幸いです。

参考文献

酒匂寛:「ツール統合と環境改善について」
第三回SEA環境ワークショップ@長野報告書
酒匂寛:「Xme User's Guide (Ver1.0)」


COMMUNICATION VIEW
MAIL
NEWS
sendm
readm
jnews
postnews
END
MAIL
kcssu1
kcsne2
kcssg1
大阪大学
anywhere
kcsne1
term
jterm
width
80
X
-10
height
32
Y
-10
PROGRAMMING VIEW
Source File Name
test.c
edit
man
socket
select file
print
less
man
LIBRARY
X
XMenu
termcap
termlib
sjlib
sj2lib
curses
jcurses
cglnews
cglx
ocal
sony
Object
a.out
-g
cc
RUN
dbx
JUNET NEWS VIEW
GROUP
comp.sources
jnews
fj
ai
misc
sys.news
announce
net-people
sys.sun
books
news.-
unix.-
comp.-
os.-
wanted
editor.-
ptt
windows.-
guide.-
questions.-
jokes
sci.-
rec
junet
soc.misc
kanji
sources
lang.-
sources.d
lectures
sys.mac
mail
sys.misc
kcs
kcs
kcs.lan
comp
misc
mod
net
news
rec
sci
soc
talk
END
KCS MAIL VIEW
送信ファイル
SEA-annai
ファイル選択
編集
印字
主題
SEA in Kobe
アドレス
送信
部長・次長
盛田
大杉
桂川
西村
課長・副課長
小川
谷口
玉本
瀬良
桑
魚田
秋定
川田
似倉
松宮
原田
係長
岡田
柴田
藤井
政門
岡本
その他
平山
柳瀬
安達
受信
終了
kcsne1:yanase[156] => parsing des
[7] Exit -32 xme -
kcsne1:yanase[156] => xme -f june
[7] 3505
kcsne1:yanase[157] => xme -f jun
[8] 3507
[4] Broken pipe xme -
kcsne1:yanase[158] =>
SJ2

Prototyper's WorkBench (PWB)

Software Research Associates, Inc.
環境開発部

塩谷和範 (shioya@sra.junet)
棚田真司 (tanada@sra.junet)

要旨

プロトタイパーズワークベンチ[1] (：PWB) は要求定義のためのツールであり、６つの図形モデル言語と疑似実行（アニメーション）機能及びモデルの分析機能をもつ。

PWB システムによりユーザからのシステム要求を６つの別の観点からモデル化し、疑似実行(アニメーション)させる過程で要求仕様の解釈の誤り・矛盾・漏れなどを検出し、最終的に実行可能な要求仕様としての試用品／プロトタイプを作成する。

また実行の様子をユーザに見せることによって目的システムの動きの確認が要求定義段階で行なえ、お互いに納得したシステム作りができる。

最終仕様としてのプロトタイプは従来の文章による要求仕様に比べ(モデルとしてフォーマルに記述されているので)あいまいさが少なくより完全である、したがってプログラマへの情報伝達手段としても優れている。

1. PWBのコンポーネント

PWBは次のコンポーネントより構成される。

(1) インフォメーションブラウザ (Information Browser:IB)

(2) モデリングエディタ群 (Modeling Facilities)

(3) セマンティックチェッカ (Semantic Correctness Criteria Editor/Checker)

(4) アニメータ (Animator)

2. PWBにおけるプロトタイピング

PWBでの作業は次の四つに大別できる。

[1]ここでプロトタイパとはプロトタイピング手法を用いる要求分析/定義者を指す。

(1) ユーザからの要求情報の入力／分類／整理（インフォメーションブラウザ）

(2) 要求の分析とモデル化（モデリングエディタ）

(3) モデルの意味分析（セマンティックチェッカ）

(4) モデルの実行／デモ（アニメータ）

2.1. ユーザからの要求情報の入力／分類／整理

ユーザからの要求情報はインフォメーションブラウザを使ってＰＷＢに入力され、分類／整理が行なわれる。
（モデリングエディタによって作成されたモデルもＰＷＢ情報として取り込まれ、同様に分類／整理が行なえる。）

2.2. 要求の分析とモデル化

整理された要求はプロトタイパによって参照され、次の六つのモデルに分析／モデル化される。

以下にＰＷＢで使用する六つのモデルを説明する。

(1) ＥＯＰＭ (External Operational Profile Model)
- 外部インタフェース要求モデル
ＥＯＰＭは対象システムを外部から、そのシステムの使用者の立場からモデル化したもので、外部に現われるシステムの振る舞いを状態と状態間の遷移で記述するモデルである。

(2) ＩＯＰＭ (Internal Operational Profile Model)
- 内部インタフェース要求モデル
対象システム内部の制御の動きをペトリネット(Petri net)に基づくモデルで記述する。ＩＯＰＭでは並列処理・同期処理・排他制御・優先順位制御（Priority Control)・実行制御等システム内部の動きを記述することができる。

(3) ＰＭ (Performance Model) - 性能要求の評価モデル
待ち行列モデルを用いて対象システムの性能要求に関する部分をモデル化しシミュレーションする事が出来る。このモデルに与える仕事（Job）はＰＭ内で定義することもできるし他のモデルから与えることもできる。仕事の性質・資源量・待ち行列の大きさ・サーバ（Server）の能力等の要素をモデルとして記述する。

(4) ＦＳＭ (Functional Structure Model) - 機能構造モデル
対象システムの要求機能間の動的な関係を記述するモデルで三つの基本要素で記述する。この基本要素は“動作（Action)”・動作の主体である“動作主（Actor)”・動作の対象である“対象(Object)”の三要素からなり、これにより動作を中心とし

た各機能の関係をモデル化する。

(5) ＦＭ (Function Model) - 機能要求モデル

対象システムに要求される個々の機能を“ある入力が与えられたときにある出力を得るための変換”として定義する為のモデル。この定義は、入力・出力・変換機能・変換に伴う属性・支援機構（Support Mechanism）の五つの要素を記述することにより行なう。

(6) ＵＩＭ (User Interface Model) - ユーザインタフェースモデル

ユーザインタフェースモデルは対象システムとその使用者の間のインタフェースを記述する事を目的とする。ＵＩＭでは外部に現われるシステムの振る舞いを最終システムのユーザインタフェースになるべく近い形でモデル化する。外部事象（オペレータ操作など）も取り込め実行条件の決定に使うことができる。

これらのモデルは図形構文エディタ（モデリングエディタ）によって構文誤りのないモデルとして作成される。各モデル毎の構文はメタエディタにより定義する。

2.3. モデルの意味分析

モデルの意味上の誤りについては、モデリングエディタに内蔵されている意味分析／チェック機能（セマンティックチェッカ）で検出できる。分析の基準となる意味定義は一階述語論理に基づき記述され、このための意味定義機能（セマンティックエディタ）はメタエディタからを呼び出せる。

2.4. モデルの実行／デモ（アニメータ）

意味的な誤りが除去されたモデルは、アニメータ環境で関係付けられたのち実行させる事ができる。実行時には次のような要求定義の確認と各定義間の矛盾が目視確認できる。

・ＥＯＰＭ　－　外部インタフェース要求の確認
・ＩＯＰＭ　－　内部インタフェース要求の確認
・ＰＭ　－　性能要求の評価／確認
・ＦＳＭ－　機能構造の確認
・ＦＭ　－　機能要求の確認
・ＵＩＭ－　ユーザインタフェース要求の確認

またユーザインタフェース部分を中心に実行させることで目的システムの動きをシミュレートすることができ、完成時の目的システムの操作を疑似体験できる。

4. PWB 適用例（患者容態監視システム）

以下に PWB-Tokyo-WG で作成した「患者容態監視システム」への適用例を示す。

次の原始要求に基づき PWB を使ってプロトタイピングを行なった。

(1) ある病院で患者監視システム(Patient Monitoring System)が必要とされている。

(2) 各患者はアナログ・デバイスで次のような要素をモニターされる。（脈搏・体温・血圧・皮膚抵抗）

(3) プログラムは（患者毎に指定された）一定周期でこれらの要素を読み、データベースに格納する。

(4) 患者毎に各要素に対しての安全範囲が指定される。（例えば患者 X の正当な体温の範囲は華氏 98 度から 99.5 度までである。）

(5) もしある要素が患者の安全範囲を外れたり、アナログ・デバイスが壊れたりしたら看護婦詰所に知らせる。

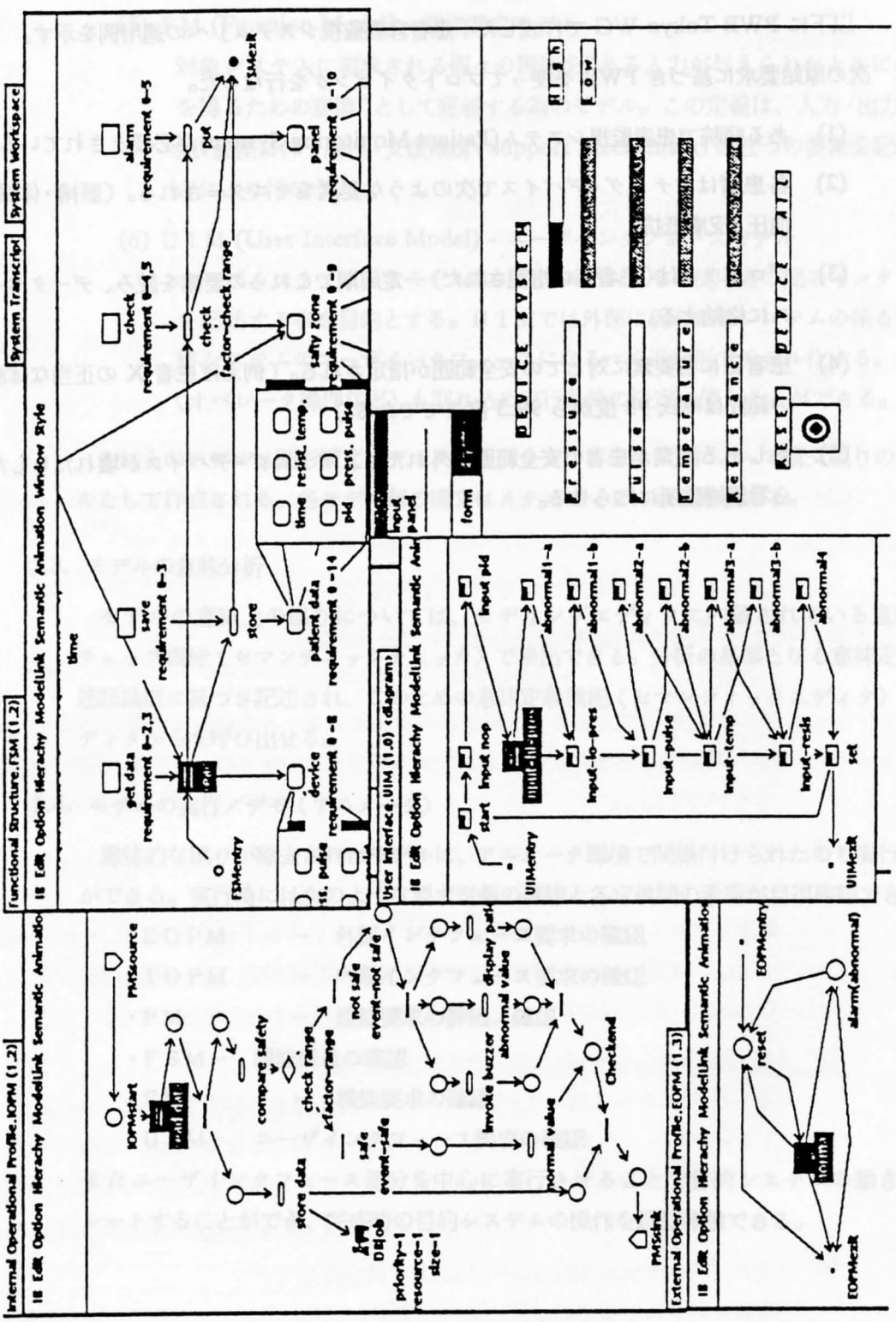


図 1 PWB 適用例（患者監視システム）

第4回　ＳＥＡ環境ワークショップ

セッション３　現状におけるＭＭＬの問題点とその改善の方向を探る

チェアマン　：久保宏志　（富士通）
プレゼンテータ：井川裕基　（ＪＩＰ）
：平山伸一　（ＫＣＳ）
：坂下　秀（Ａｓｔｅｃ）
レポータ　：岡本隆一　（ＫＣＳ）

0. 前ふり

中野（大阪大学）： それではＭＭＬ「マイクロメインフレームリンク」のセッションを始めます。始める前に今日の夕食を心配されている方々がいらっしゃるので，このセッションは一応５時半終了にします。ＢＯＦは６時からですが，５時半がどの辺に成るか分からないので，その後に４５分位の休憩をとりたいと思います。おなかが減った方はそこで軽く食べて下さい。ＢＯＦでは缶ビールが１５０本とウイスキー５本を用意してます。１人あたりビールが３本とウイスキーが１杯ぐらいあります。では，お願いします。

1. はじめに

久保（富士通）： では，始めます。富士通の久保でございます。「マイクロメインフレームリンク」のセッションです。私がこういう事になった言い訳を始めます。中野さんから電話があって，富士通にもＭＭＬの話題提供者が居ないかという事で来たのですが，大体，僕は知らないし関心も無いので，と言って...., 蓋を開けてみたら私をセッションチェアマンにしてあるんです。それで言う事は奮ってるんです。委員会に出て来ないからいけないんだ....！て言う。

酷い話ですが，返す言葉も無いので引き受けました。とにかく皆さんに３時間の時間を差し上げますのでご自由にお使い下さい。コーヒーブレイクと終わりは私の権限でやります。その他は本当に勝手にやって下さい。セッションが楽しくなるかどうは，あまりセッションチェアマンは関係無くて，話題提供者の話に皆さんが敏感に反応するというのがキーの様なので，それを期待したいと思います。但し，何をやるためのセッションなのかという事は知っておいた方がいいだろうと思うので，発案者の中野さんにちょっと解説してもらいましょう。

＆会場から笑い＆

中野： 大阪大学の中野です。どういう事情でこういう風になったかと言うと，ＳＥＡの幹事会で勝手にＭＭＬがテーマのひとつになりました。今回のセッションがどういう風になっているかと言うと，全部英語で...., Ｈｙｐｅｒ，ＭＭＩ，ＭＭＬ，Ｃａｓｅ。それだけの理由なんです。それで，そういう風に言うとプログラム委員長として非常に権威が無いので，ちゃんと言います。

前回，前々回はワークステーションと分散環境というテーマでやりました。それで，前回の分散環境とワークステーションは私がチェアマンをやりました。その時に無理やりＭＭＬの話に振ったんです...., と言うのは，ワークステーションと分散環境という風にやると，大体，１／３位の方は良く分かっておられる。残り２／３の方は，特にホスト周りの方は全然付いて行かない。そういう世界があったんです。それをまた思って，ＳＥＡの幹事会でＭＭＬに振って，ホストの人達も参加して下さい...., そういう話になった様です。

今回のプレゼンテータも一応，一応ですがホストサイドから二人。それで，ホストは知らないよと言う人のために面白い人をひとり入れましたので，私はこのセッションを一番期待しています。そういう事です。

久保： 大体趣旨は理解された事と思いますので宜しくお願い致します。話題の提供者は３人の方。順番は，ちょっと職権の乱用ですが，表題だけ見ると結構良く出来ているので，「一事例」で始めて「諸問題」「理想の....」という展開にしたいと思います。宜しくお願い致します。

2. プレゼンテーション

2.1.1 ＪＩＰ（日本電子計算（株））におけるメインフレームマシンの作業環境とＵＮＩＸネットワーク

井川（ＪＩＰ）： 日本電子計算大阪支店の井川です。そういう順番のリクエストですが，一事例と言うにはチョット色々問題があります。こんなものを事例にして良いのか....？と言う話があると思います。昨日今日の話を聞いてかなりレベルが高いというか，皆さんの発表とか議論されている内容ですね。かなり高い所でやっておられるんですが，私が実際にやったのは非常に手近な事です。発表する内容もそれしか無いので，そういう意味では「当たり前，分かり易い，そんなの出来るよね」という話を致します。ポジションペーパを見ていただければ分かるんですが，平山さんのようなサーベイをやったわけじゃないですし，坂下さんのように夢の話をするわけではないので，つまらないかも知れませんが宜しくお願い致します。

私どもの会社は大型計算機を会社で持っていて，受託計算をするのが基本的な業務の会社です。私自身は直接そういう業務ではないですが，会社の本体はＣＯＢＯＬとかＦＯＲＴＲＡＮとか，あるいは昨日ちょっと話に出ていましたが，ＡＬＧＯＬで業務をしています。それで，プログラムを作る事が仕事ではなくて，その上で処理をする。あるいは，最近はユーザさんが端末を使える事も多く，そういう運用もしています。ですから基本的にマシンを使う事が業務ですね。それで，そのためのソフトですからソフトを作るというのは業務の付随の内容であるわけです。ですからソフトウェアという事には無頓着。ホストマシンを運用するにはタップリお金をかけているんですが，社内のユーザが仕事をするという環境ではかなり遅れています。

これは，メーカの方がご覧になるか先進的なホストユーザの方が見るとショッキングな内容なんでしょうけれども，うちの会社の内容というのは，メインフレームのユーザというのは，こういう状況で仕事をしているんですね。ホストマシンはいわゆる大型機。端末は，最近は色々なものが出ていますが基本的にうちのユーザが使っているのは８０桁×２４行。８０桁というのは明らかにカードの大きさのせいですね。そういう端末を使って，ＶＯＳ３とかＣＡＮＤＥと言うメーカが提供しているユーザインタフェースで仕事をしています。

メーカが提供している物についてはマニュアルを読んで使います。簡単な物は自分で作ります。会社で作って会社独自で使っている物もあります。

ここで問題になるのは，大きな計算機を抱えているために，開発から実行まで全部そこでやらなければいけないという事です。かなり大きなデータベースの業務が走っているマシンでエディティングをする。私自身も中でやっているのにお話するのですが，ＵＮＩＸのネットワークを使ってるユーザには，とても信じられない感じです。しかし，これが現実の姿でもあります。例えば，自分の投入したジョブがどういう状態にあるかを見るために画面端末の前に座ってコマンドを入れて見るという事をしなくちゃいけない。あまりにもこれは悲惨な状況だと私は考えているわけです。先に言った様な，あまりリッチでないユーザインタフェース。それから，単純に言えば端末が足りないだけかも知れませんが，どうもスムースでない仕事の流れ。これらの事が「何とかならないか....？！」という事は誰しも考える事です。

メーカは「開発用のマシンを入れられてはどうでしょうか」という話をされるわけですが，実際には，事は簡単には進まないわけです。お金の問題がまずあります。回収出来るかどうかとか。大体，業務で使っている人達は，どれだけジョブを回せば回収出来るか....，みたいな事をすぐに計算しますので，そうすると開発のためにという事になると全く回収できない事になってしまうわけです。で，疑問点が残る。あるいは，小さい機械で開発してもそれが大型機に載らない。という様な現実の問題があります。これはメーカによって違うと思います。あるいは，メーカでもシリーズによって状況は違うと思います。けれども，現時点で私どもの会社では，そうなっているという事です。で，それは困るという話をしますと，メーカ側は「メーカが作ったＭＭＬをお使い下さい」。ＭＭＬという言葉はメーカも最近は使います。以前は使って無かったんですが「うちで作った開発環境がありますから」というような事を言って来るわけですね。ところが，そういうやつっていうのは，後に控えている者が紹介しますからご覧になっていただけば良いかと思いますが，確かに美しいというか，ある基本の規則に則った開発は出来るんですが，それに沿ってなければ「ダメだ！」と言うんですね。ですから，既存の動く業務をそれに切り換えて効率を上げるという事が出来ないし，やっぱり２０何年も会社をやっておりますと基本のやり方というのがあるわけで，それに微妙に合わないという事が出てきます。そのチューニングは，メーカは「それはお宅の事情ですから関係ありませんね」という事でそ

の通りにしてくれない。何かあるんだけども使えないシステムとしてマシンのリソースは眠っているわけなんです。

で，そういう風に悪口を言った後なんですけれども。その辺の事情。日立の方の実例。ＳＥＷＢというのがあります。これについて，後ほど関口の方からどういう物なのかという事を説明，紹介してもらいます。

そういう状況の一方で，私自身がそうなんですが，ＵＮＩＸのネットワークがここ（ＯＨＰ）に示すように普及しました。８６年位から始まって，実は，私どもはＮＥＷＳというマシンのディーラをしておりまして...，決してＮＥＷＳの質問をしないようにしていただきたいんですけども....！　私もそのサポートやっております。その関係で，社内にマシンが増えてネットワークが広がりました。ついでにＪＳｔａｒとか，その辺にあるものをドンドン組み込んで環境を広げているんですけれども。そうすると，そういう物を見ているホスト，メインフレームのユーザが実際に使う事もあります。そういう物を見ていて，あれをホストの...，先から申し上げております様な，かなり苦しい状況に合わせて，それを「改善できないか....？」というリクエストが出ます。

これは当然の事ですけれども。で，彼らが言うには「非常にＵＮＩＸのマシンはレスポンスが良い」。これは当たり前の話ですね。それから「パワフルである」と言います。大型機のユーザには大型機信仰というのがございまして，「ワークステーションなんかで何にも出来ないじゃないか...，ＰＣなんて所詮は端末さ...，」。と，そういう乗りで僕は見てるんですけれども。我々がやってるのを後ろから見ていて「それは結構使えるじゃない」と，こう言うわけですね。

機械が変わると仕事が出来ないというイメージはあります。若い連中からだんだん馴染んで来る事は，やっぱり必要性があって触っているうちに馴染むのですが...，例えば，非常に大きな障害としてはですね。ホストのユーザ達はプログラムが大文字で書いてないとどうもうまく認識出来ない。＆笑い＆若い人はどちらでも認識出来るわけですが，初めはそんな事すら「それはダメ」という理由になっていたんです。今はそうでも無くなって来つつあります。で，「そこまで行ったんだったら何とかならないかな」という事で，色々と先のＭＭＬから夢を見ている絵（ＯＨＰ）があったわけです。

ちょっとうちのネットワーク。ＵＮＩＸの話になってしまいますが，ネットワークとして，こんな風（ＯＨＰ）に色々な機械を繋いでいます。ＮＥＷＳとかＳｕｎなんていうのは，これは当たり前ですが，ターミナルサーバ経由でシグマ。但し，シグマワークステーションとしてゼロックスのＸＮＳの環境を４．３ＢＳＤで一部サポートしていますので，それで繋いでいます。それで，今の絵にちょっとありましたけれどもホストも巻き込んで行こうとしているわけです。で，巻き込んで何をさせるかというのが，この辺が今日のテーマの本当の所なんでしょうけれども。ただ書いただけでは，少しも何も起こらないんですけれども。ローカルにエディットしたりチェックして，コンパイルして，後は，その成果をドキュメンテーションで受けさせる。これは誰でもが普通に思いつく事。

今日の私の発表は，全然これについてどうこうというわけでは無いんですが。ただですね。「そういう事をやりたいな」という話をしている関係で何が出たかというと，私の所の会社のマシンはネットワークに繋がらないという事実が，俄然，クローズアップされてきたわけです。但し，ネットワークに繋がらないと言いますのは，イーサネットを持ってなくて，増して，ＴＣＰ／ＩＰのプロトコルを理解出来るわけではなくて，それではトークンリングをちゃんとサポートしているかというと，自分の所の会社のやつは繋がるのですが，という話で，他の物は全然分かりません。「保証できない」という事ですね。

ここで突然，話はレベルがトーンダウンして，俄然，下世話な話になってしまうわけです。「どうやって....？」。「何か繋がって....？」。「ＭＭＬも糞もない....！」。「一番低レベルのＭＭＬも実現できないじゃないの....！」。

...，という事で，必ずコンピュータには非同期のポートがありますので。で，ＵＮＩＸは大体非同期の通信。まあ，柔軟ですけど...，位を考えていたようなシステムですから，非同期であれば色々いくつかは出来て，それを使って繋ぐ事が出来るんじゃないかという事で，そういう接続をします。で，そこから出来るのは，せいぜい仮想端末とファイル転送。ファイル転送は，実は，まだ出来ていないんですが。それから，なんせ商用も商用。毎日お金稼いでいる機械ですから「ちょっとＯＳにパッチ当ててね」というわけにはいかないので，ＯＳには，ＯＳからは何も変わったようには見えないという様な条件を設定して接続する事にしました。

その接続の形態なんですが，非同期の回線を繋ぐのに

ホスト直結ではありません。ＴＳＳサービスを業務としてやっていますので，色々な都合上，直結ではなくてポートセレクタと言うＲＳ２３２Ｃのスイッチと考えて頂いて良いんですが，それを介して，ホストにＴＳＳの受け口がありますが，それをそのまま流用して，ＮＥＷＳにＲＳ２３２Ｃのボードを差し込んで，そこからポートセレクタに....，ポートエキスパンダという処理になりますけれども，これに繋いでいます。このＮＥＷＳをＢｕｒｒｏｕｇｈから呼び出してやれば繋がるじゃないかという事です。

で，こういう物理的な接続をしていきまして，この状態で出来る事は何かという事になりました。先程言いましたように，実現出来る内容には制限があるというか，明らかで，非同期の回線でホストとなら，最近は少なくなりましたけれど，ダイヤルアップの音響カプラで繋ぐ。そういうＴＳＳサービス。これをエミュレートしていると言っては変ですが，そういう風に見せかけてホストに入り込みます。それから，（ＯＨＰの）ちょっと真ん中を飛ばして，その下。そういう回線経由で出来る画面端末は日立には一応あるんですね。まあ，そういう物に見せかけてやれば画面端末出来ます。わざわざ画面端末と言ったのは，上の方は画面ではなくってＵＮＩＸで言えばｅｘ。ｖｉじゃなくって，と言うようなラインの形式のエディタ。それからコマンド。シェルのようにコマンドを入れて反応に答えていくわけですね。これで，先の様にジョブの実行状態を見るために画面端末の行列待ちはしなくて良いという様な事が実現してます。その実現方法としては，ごくありふれたサーバクライアントモデルで，先程言いましたポートエキスパンダとそれに繋がったＮＥＷＳにデーモンを走らせておいて，そのデーモンを呼び出す。そうするとネットワーク上のどこからでも今言ったようなサービスが受けられる。この漫画（ＯＨＰ）は実際に実現しています。

で，もう一個。これは普通に思いつく機能。色々と事情がありまして出来てないのが結論なんですが。ＵＮＩＸ側に同期で何か通信をするという事に我々はあまり経験が無くて，手を打っては見たんですがうまくいかない。で，それは今は実験中です。ここで，もし経験のある方がいらっしゃったら情報を頂ければ，と思います。

現在，今，紹介致しました様な状況になっています。で，あれは繋がるだけですので，そうですね。やりかけている物を，まず，これから実現して行きたい。それから，もう少し高速のエディタです。ＩＢＭが割と安いＥｔｈｅｒｎｅｔのインタフェースを出したそうですから，これからＥｔｈｅｒｎｅｔに繋がる様になるらしいです。私どものホストも右へ習えして繋がる可能性が出て来るんじゃないかな。今の所，お話をしてもそういう予定は無いという事なんですが，いつ宗旨変えをするか分からないですから，それを期待しているのと，ＦＤＤＩは，話題になっているわけなんですから，何年か後には各メーカとも対応するんじゃないかという事で，可能性があるんじゃないか。先程「チップレベルで出来る事はソフトウェア屋さんがやるべきじゃない」という意見がありました。現実的にメーカがこういうのを実現してしまえば，私達がやっている事は徒労に終わるわけですが，むしろ，その徒労を期待したいと思っています。

これだけでは，最初に申しました様に非常に下世話な，ただこんな事をやってますと言うだけで，今後は，もう少し実際のソフトウェア開発の方に寄った事に手を伸ばして行きたい。ただ，実現出来てないのが沢山ありますので，その辺を実現してからの話です。言語レベルともうちょっとリンクして，あまり第５世代言語とか，かっこいい事は言えませんが，そういう事にも手を出して行きたいと思っています。と，いう事で，私がお話する内容はここ迄なんですが，先程申し上げました様に，日立の方の日立が言っておりますMMLの一例としてＳＥＷＢの紹介を関口が用意しておりますので，続けて....，

2.1.2 ソフトウェア開発の分散化（EAGLE2/SEWB）

関口（ＪＩＰ）： ただいま紹介にあずかりました日本電子計算の関口です。私は井川君と違いましてホストしかやった事がありません。それで，彼も言いました様に，うちの会社は元々受託計算が主流で，ソフトウェアを作るといっても，結局は，その納品物を作る様な事ばかりやって来たので，作るソフトウェアといっても使い捨てみたいなのが多くて，そんなに大規模なやつは作らなかったんですが，最近，メインフレーム上での大型のＤＢ／ＤＣシステムが出て来ましたので，作るシステムも大きくなって来たので，生産性を上げなくちゃいけないんじゃないかという事です。

（ＯＨＰ）ＳＥＷＢというのはワークステーション上で動く開発支援ツールで，こちらのＥＡＧＬＥ２と呼ばれている物が日立の言うところのホスト上の生産性を上げるためのツールです。それで，ＥＡＧＬＥ２というのはソフトウェア一貫生産システムと言いまして，日立さんの言うところのＨＩＰＡＣＥという開発メソドロジで，

昨日から言われてますけど，古典的なウォータフォールモデルのメソドロジに従って，ソフトウェアを中流から下流まで一貫して生産しようという支援システムです。マシンとしては日立のMシリーズ。うちではS－８１０でやってますが，OSはVOS３です。

SEWBの方は「Software Engineering Workbench」と称して，マシンとしては日立の２０５０シリーズ。OSはUNIXのSystemV（R２）に従っていて，ウインドウシステムは流行のXウインドウではなく，日立が独自で作ったウインドウシステムを載せてます。

それで，MMLから外れるんですが，ソフトウェアの生産性を上げるためのツールとして，最初にEAGLE２を入れたわけです。EAGLE２には....,

- 対話によるシステム開発
- ソフトウェア資源の一元管理
- 部品有効利用システム
- ドキュメントの自動生成

と言う四つの大きな柱がありまして，生産性を上げるわけですが，対話によるシステム開発っていうのはメイフレーム上の端末での対話ですので，今迄みたいに，ただ単純にエディットしてコンパイルしてバッチジョブをサブミットするような感じだとメモリーは少なくて済むんですが，その上にLIMEを載せて，どんどんメモリーを食って行きますので，生産性を上げるためにEAGLE２を入れたんですが，EAGLE２を動かすために端末のリージョン値が増えて遅くなってしまう。もう，どうでも良い位に遅くなってしまってるんですね。で，それを解決するために何か良いものはないかなという事で探したのがSEWBです。

SEWBというのはワークステーションですから，マウスとかアイコンとか，そういうソフトウェアを目で見ようとするビジュアルな所は良く出来てます。要するに，システム設計以降じゃなくて，システム設計前の分析のあたりから使いたいという事で，ソフトウェアのビジュアル化，そしてプロトタイピング。そして三番目にホストの重くなった機能を分散してこちらに持って来ようという事で，ようやくMMLの話に持って来るんですが....,

ちょっと細かくて見辛い（OHP）ですが，これがワークステーション側のSEWBとメインフレーム上のEAGLEという二つのソフトウェアの繋がりですね。MMLの感じです。

まず，この日立が言うところのSDF，一般的にはDFDと呼ばれているデータフローモデルですが，図（OHP）で書くとこんな形ですが，これをワークステーション上で編集をします。その編集したSDFと呼ばれる物をこちらのホストコンピュータの方にアップロードします。そして，アップロードしたSDFを，ここの機械処理というのがシステムフローと一対一に対応していて，マスタを読んで，トランザクションを追加して，マッチングして，エラーデータをはじいて，一覧表を取るそのSDFからこのシステムフローを生成します。

ホスト側の方では，出来たシステムフローを今度は逆にワークステーション上にダウンロードします。出来たシステムフローはこんな形（OHP）になります。これは先程のSDFのボックスにあたる全体の見出し図で，中身はスタートで，入力ファイルがあって出力ファイルがあるなんていう様な事で，標準的には，これに一杯エラーリストとかがついているんですが，この行が必要無いとかいう場合は，実際に追加したり削除したりというのをマウスを使ってやるという事です。そして，出来たシステムフローから，EAGLE２には仕様書からソースを生成するという機能があるので，そのシステムフローから各種の仕様書を作りまして，そこから半完成ソースを生成します。ホスト上でも編集出来ますし，こちらからダウンロードしてワークステーション上でも編集できます。

で，ここから先の作業。単体テストに関しては，ホスト側の方でも出来ますし，ワークステーション側でも出来ます....，と言うのは，日立の言い分です。で，特にワークステーション上の方のテストでは結果をビジュアルに見れます。（OHP）これはPAD図ですね。PAD図を表示しながら現在何処にいるのか，そしてデータは何があるのかというのをこの様に目で見ながら確認が出来る....，と言うのが日立の売り物です。

こういうテストしていると時間がかかりますので，その分，ホストの負荷が分散されて軽くなるだろうという事で，メーカの方の言うところの適用効果というのは，まず「システム開発の品質向上が図れる」。それはSDFとかシステムフローみたいなシステム開発の上流工程における所をビジュアル化，グラフィックなエディタを使ってサポートする事が出来るというところですね。そして，先に見たPAD図の様に，プログラムそのものを図式化して表現する事によって品質を上げられる。また「デバッガによる信頼性の向上」，「プロトタイピング」という様な文句があるんですけれども，実際はホスト上で動くオンラインシステムの画面をワークステーショ

ン上で定義して，その画面間の流れを一応確認する事が出来るという様な....，それを一応「プロトタイピング」と呼んでいるんですけれど，それをする事によって「正当性，ユーザの要求の把握に効果がある」という事ですね。

もう一方の柱として「システム開発の生産性を向上させる事が出来る」。「対話レスポンスの向上」。それは端末処理をワークステーション上に持って来る事によってホストの処理を低くし，対話のレスポンスを上げるという事。ホストマシンというのは開発だけではなくって実際のプロダクションの方も走ってますので，独立した開発環境を持つ事によって「昼間はデバッグ出来ない」とか「昼間は本番ランが走っているので夜にならなけりゃダメだ」なんて事は無くなってホストから独立して開発する事が出来る。

三番目が，先程言いました様に，「ホストの負荷を分散させる事が出来る」。これがメーカの言うところの適用効果ですが，実際に使って見て問題点というのは沢山あるわけですね。で，それを幾つかあげてみます。

1．ソフトウェア資源の分散
- ・ホストとWSで2重管理が必要
- ・アップロード／ダウンロードが自動的に行われない

2．環境の相違
- ・文字コード体系(JIS/EBCDIC)
- ・ソースプログラムの互換性

まず一番目は，ソフトウェア資源が分散してしまうという事です。ホストとワークステーションで二つ。仕様書にしろソースにしろ分散して持ってしまいます。バージョンの管理の問題とか，二重更新が起こってしまうとか，管理に非常に手間がかかってしまうという事です。

そしてもうひとつ。これは非常に解せないんですが，アップロードとかダウンロードが自動的に行われない事です。この日立のSEWBには画面上に転送というボタンがあって，それを押すと勝手に送ってくれるわけではなく，別な画面を開いて，ホストにログオンして，そこでまた，新たなジョブを起動するなんていう非常に面倒臭い事をしなくちゃいけない。解せませんね。＆会場で大きなどよめき＆

それで，環境の相違というのもあります。2050のワークステーション側というのはUNIXだからJISコード準拠。ホストの方はEBCDICコードを使っています。要するに，コードが違うという事は，例えてSORTひとつをとってみてもホストでSORTした場合とワークステーションでSORTした場合と結果が違ってしまうわけです。これでは何のためにテストを分散したのか良く分からないという事になってしまいます。＆会場で笑いとどよめき＆

そして，もう一個。こっちの方が問題なんですが，ソースプログラムに互換性がないんです。＆会場笑い＆全く無いんじゃなくって微妙に違うんですね。同じCOBOLにしてもSELECT文の書き方が違うとか....，まあ，その辺で....，全く異なっているんだったら諦めもつくんですが，少しだけ異なっているんでそれを見つけるのに余計な負担がかかってしまうんですね。ですから，先程お見せしました適用効果はメーカの言う適用効果ですけれど。私が見たところの適用効果を重ね合わせます（OHPを重ねる）と，こういう結果になります。これは非常に大甘に付けた点数ですので。＆会場笑い＆

適用効果

システム開発の品質向上

○ ・SDFによる設計の機械化

△ ・プログラムの図式化

× ・デバッガによる信頼性の向上

△ ・プロトタイピングによる正当性の向上

システム開発の生産性向上

○ ・対話レンポンスの向上

× ・ホストから独立した開発環境

△ ・ホスト負荷分散

SDFによる設計の機械化が○と付いていますが，それは，単に手書きの文書がレーザプリンタに出て来る....，と言う位の意味で....，＆会場笑い＆　元々はDFDにしても，そこの所，変換というのは機械がやってくれるわけではなくって，やっぱり人間が物理モデルや論理モデルに発展させているわけなんで，そこの所はちっともサポートしてくれません。ただ単にアウトプットのドキュメントが奇麗になっているという事で，まあ，一応○だけ付けておいてあげよう。

で，△のプログラムの図式化。これも問題がありまして，例えばエディタです。ワークステーション側の方のエディタにはPADエディタと言ってソースをPAD図にして編集する機能があるんですが，慣れてしまうとPADで入力するよりもソースで入力した方が当然早いわ

けですよね。ですから，分かり易い事を前提に立ててやるよりも入力して早い方が我々は早く帰れますから，そっちの方が生産性が高いわけです。＆座長，噎せるほど笑う＆　ですから，図式化するという事がどうかなという事で△です。

三番目のデバッガによる信頼性の向上。先程言いましたようにホスト側とワークステーション側とのソースレベルの互換性があまり無いんで，ホスト側でチェックする度に直して，ワークステーション側の方でチェックして直して何とかだったら作業負担が倍になってしまいますので，かえって使い辛いので，それをやらないで，結局，元通りの仕事をするのでデバッガによる信頼性の向上は図れない。

そして，プロトタイピングによる正当性の向上。（ＯＨＰ）ここに画面遷移シミュレータというワークステーション上の機能があるんですが，ここからホストに行くのは，ここ....！他は実線ですが点線さんです。要するに参照なんですよ。ここまでは前のステップの結果がそのまま次のステップの入力になっているのに，ここだけは画面遷移シミュレータで出来た結果を人間が参照して，また機械に載せなくちゃいけません。ですから，これは繋がっていないので，ここで一回入れた事は，もう一回ここで入れなくちゃいけないという事になります。これは，ここ（ＯＨＰ）の段階での正当性は向上しますが，ここから先がここに合っているかどうかの保証は別になりますので，結果，△になってしまいます。

生産性の向上。まあ，対話レスポンスは確かに向上します。それはワークステーションに持って来た方がエディットなんていうのは何十倍も速いですから，それは向上します。が，先程言いました様に，自動ログオンというのが出来ないので，その分を含めると△なんですけれども。まあ大甘にして○を付けました。

で，ホストから独立した開発環境というのは，これも先に言った様にデバッガがホストとワークステーションで違いますので×です。

負荷分散。確かにホストの負荷は分散されたんですが人間の負荷が分散されないんで....，これは，ですからホストから見れば○かも知れませんけれども，我々からすれば△。＆会場笑い＆

私どもユーザの視点から見た結果というのはこうなってしまいます。但し，先程も言いましたが，ホストに端末を１０００台，２０００台ぶら下げた開発環境ひとつよりも，やはりワークステーション上で分散開発して，その結果をホストに持って来た方が効率が良い事は自明の理ですから，我々ユーザ側の立場からしてみれば，こういう欠点，問題点をメーカにどんどん言って，メーカの方に直してもらって，少しでも我々ユーザのレベルを上げてもらおうと思っています。以上です。

久保：　ちょっと時間を差し上げますので，どうぞご自由にやって下さい。この話題の範囲内で，後でまとめますが，どうぞご自由に。

深瀬（ＡＳＣＩＩ）：　ちょっとお聞きしたい事があります。ホストコンピュータを提供する会社ですね。幾つか「Host Mainframe Comeback?」として提供して明記すべき失敗例というのが幾つもあるんですけれども。２０５０なんかもその例なんですが。立場をシステム側の人に是非とも聞いて頂きたいんです。何故，そういう物を選んでしまったのかというのを聞きたいんですけれども。他にもそういう立場の方ってあるわけですよね。ホストやＶＯＳ３が動いているという環境で端末何にしましょうか。色々選べると思うんです。何故，選んでしまったのか。何故，そんな悲惨な状況に自らを追い込んでしまったのかというのをお聞きしたいんです。＆チェアマン笑い＆

関口：　まあ，やはり，それはマネージャの決断でしょうね。＆会場笑い＆我々としては，別に入れてくれと言って頼んだわけじゃありませんので，我々として貰った環境を如何に良く使うかという事で色々と四苦八苦しています。ホスト側のＥＡＧＬＥというのも，ここ２年位でようやく使えるようになって来ました。バージョンが６まで来ました。バージョン４は使えなかったんですが，６に来てようやく使えるようになって来ました。ＳＥＷＢは，まだバージョンが２なんですよね。ですから，あと四つ位進まないと....，＆会場笑い＆まあ，その頃になれば使えるかも知れませんが，要するに，ツールっていうのは道具ですから，日頃から使い込んで来ないとどうしても使えないわけですよね。ですから，早い内に慣れておこうという事で。今，多少の欠点は....，多少というか，欠点は非常に多いと思うんですが，ＥＡＧＬＥだけを使っておいて，後，美味しいところは来年なり再来年なりに期待しようっていうところが正直言って私の立場です。

深瀬： わざわざ重い獅子を追って遠き山道を登る事はないと思うんですが。ある意味では諦めに近いわけですか。

関口： 日立のＶＯＳ３上に動くＤＢ／ＤＣシステムを使ってますので，ＤＢ／ＤＣについては各メーカとも違いますから，そこまできっちりサポートしてくれる他のサードパーティ等が在るとは思えませんので。そこのところは，やはりメーカでしかサポート出来ないと思います。そこのところに期待をかけているというのが実際です。

深瀬： 今，２０５０を一生懸命使いこなす努力をする位だったらですね。自分達で選んだ道具だての上で，それでエミュレーション動かした方が早いと思うんですが。

関口： ただ，悲しい事に，うちでは現場の者でそういう技術力を持った人が居ないっていうのは....,

深瀬： やれば出来るんじゃないかと僕は思うんです。大した事はやってないですよね。メーカの人達の言い分は，これだけの工数をかけて作ってますからちゃんとしたものですと話しますけれど，あまり物を考えて作ってませんから。＆会場笑い＆　オンザフライで作ってるんですよね。大概の事は。ですから実際に使い込んで行かなくっちゃ，その良さがなかなか噛みしめられないなんて事になるわけなんです。何か，どうも最初から困難な道をわざわざ選んでいるような気がしてならないんですけれど。

野中（日本電気）： 日本電気の野中です。開発マシンと実行マシンを分けるという様なお話なんですが，僕はファームウェアの開発をやってますから本当に全部ＭＳ－ＤＯＳマシンの上でやりたいんですけれど。泣く泣くターゲットの上に載っけりゃなきゃいけないなんてあると思うんですが，こういうソフトですから本来は一緒の方がずっとやり易くて，マシンのパワーが無いからしょうがなくて，やっぱり分けると言う事でしょう。それは，やっぱり素晴らしい事じゃなくて悲しい事だと思うんです。

関口： そうです。＆会場笑い＆

吉本（日本経済新聞）： 日本経済新聞の吉本です。ホストと言われると，語弊があります。メインフレームと呼んで頂きたい。ホストと言えばパソコンと言えどもホストですから。

それから，データベースもそうなんですが，要するに，メインフレームメーカは，基本的にはインテリジェントターミナルなんかに乗ってもらいたくないんですよ。政策的に。

連中は....，ここではメインフレームメーカの人はちょっと耳を塞いでおいて下さい。セールスマンにしたって一台何億円の機械を売った方がポイント上がるわけで，インテリジェントターミナルがいくら高いと言ったって１００万円か２００万円でしょ。あまり点数にならないから嫌がるんですよ。これは，我々，ＩＢＭのセールスマンと付き合っているので間違いないだろう。日立さんのＥＷＳ見ても如何にも何か未練たらしいですよね。＆会場笑い＆　本当は全部，インテリジェントターミナルの上でやればいいんだけれども。そうやったら，何かメインフレーム買うてくれん様になるんじゃないか....，開発マシンが売れんようになるんじゃないか....，何か，そういう見え見えだから誰かが言われた様に，敢えて使いにくいものを使いこなさなあかんようになるという悲劇が起こるんです。

僕自身の経験では確かにＴＳＯの....，ＩＢＭはＴＳＯですね。何かと言うと，要するに８０カラムと２５カラムのキャラクタディスプレイです。最近になって悲劇なのは，色々なモニタなんかをＩＢＭさんが作るんですが，入り切らんのですよ。出てくる情報が。あの中じゃ。その辺でも可笑しいですがね。そういう意味では，私はもっと徹底してインテリジェントの方に移してしまおうと思うんです。以前は，パソコン用プログラムなんかに分散しなくたってホストでやる。確かにホストは馬力がありますから速いです。しかし，インテルの８０３８６クラスになってきたらかなり普通じゃ無くなった。出来ればコンパイル位まで全部インテリジェントでやって最後のとどのつまりだけホストでやる。

私も色々考えたわけです。ああいう素敵な機械が使えないかどうか。確かにバージョンどうするんだ。ワークステーション側で不一致になってね。管理上，確かに困る事も事実だし，そうするならいっそのこと全部載せてしまって最後の作品だけをホストでやる。そういう恰好にしたいな....，という風に考えるわけ。それで，最後に本当に問題になるのはシームレスな恰好で，どうやってリンクするかという事ですね。一番簡単な例が５５５０の上で動くＩＢＭのエディタです....，ＴＳＯの。これはパソコンの上で動くんで，それはいいんですが，出来上

がりをメインフレームに持っていく時，一回，パソコンを殺してターミナルに切り換えて....，これではプログラマは嫌がって使わんです。

プログラマが小文字では困るという事を先におっしゃいましたけれど確かにそうです。現に僕が８０２８６のアセンブラ書くときも実は大文字で書くのは，そのＩＢＭのメインフレームで育った悲しい習性なんです。そういう事もあるんですが，何か全然意識させんような方法というのがあって，それで最後にターミナルを作るメーカの方に頑張ってもらわんといかんのじゃないか....，メインフレームメーカは，やっぱり大きな機械売りたいわけですよね。基本的に「何だ！メーカ」という事になるから，むしろサードベンダで....，それこそＳＲＡさんみたいなところの方がやり易いんじゃないか....，そういう風に思います。

久保： 坂下さんの宿題が大分出来ました。理想のＭＭＬで解決して頂きたいと思うんですが，次はＦＡＣＯＭをお使いの平山さんの方からお話を頂きます。

2.2 開発環境は誰のため？ ＭＭＬは誰のため？

平山（ＫＣＳ）： ＫＣＳの平山です。ＦＡＣＯＭをお使いの....，という事でしたが，それは社の事であって，私は２年間以上ＦＡＣＯＭは使った事はありません。それで，これから先の話と同じなんですが，ＭＭＬのセッションをやる経緯がですね。私は基本的にＭＭＬは全く知らない。知らないけども「空席があるから何とか埋めて頂戴よ」という事で，正直言いますと本当はハイパーメディアの方に出たかったんです。

で，そういう事情なんで，私自身はＭＭＬについては何もやっていないというのが結論です。実際，社内でも調べたんですが，ユーザにはＭＭＬを使ってアプリケーションを書いているけれども開発環境としては何もやっていないというのが結論です。それで，どうしようかなと思ったんですが，一応，開発環境に絡めた形で，私なりに考えた事を書いてみました。ですから，たぶん登場するのは夜からのＢＯＦの方が良かったんじゃないかと思っています。で，簡単に自己紹介だけやらせて頂きます。

私は未だかつてＳＥＡの環境ワークショップに出た事はありません。理由は他に参加者が一杯いたという事です。で，今回参加の理由については，社内に希望者がいなかった。それはたぶん，場所が神戸でやるからＫＣＳからは行かないという事だと思います。それから，プログラム委員の末席に加えて頂いてますが，個人としてＳＥＡに入会しておりません。ＳＥＡの入会金の行方は何処かというと，会社で使ってるＭａｃのための「Ｉｎｓｉｄｅ－Ｍａｃ」というマニュアルと，Ｌｉｇｈｔ－ｓｐｅｅｄ－Ｃのコンパイラにですね....，会社が買ってくれないもんですからそいつに使っちゃいました。それで，ＳＥＡの入会予定ですが，中島さんにも聞こえるように言っておきたいですが，今年の１２月の２日。たぶんボーナスだと思うので，その日に入りたいと思っています。

現在，仕事はＬｉｓｐのプログラマをやっています。昔は哀愁のＣＯＢＯＬｅｒと呼ばれていました。現在の関心事とは，ＭＭＬじゃなくて，Ｍａｃ２とＭＰＷを如何に手に入れるかという事で，ＭＭまでは一緒です。それから社内では，実はここから開発環境の話になって行きますが，社内の開発環境。うちは何となくＪＳｔａｒという事でやって来ているんでが，根本的に考え直さなくちゃいけないなと思っています。それを何とかしたいというのが，今，社内に対する関心事で，先程，井川さんが「こんな事しかやってないよ」というお話をしてたんですが，うちにとってはとんでもない事で，これからそれをやらなくちゃいけないって言う風に考えています。

開発環境に対する期待っていうのは，実はですね。社内で色々調べてみたんですが，どうも自分と同期から下が，現在，自分の組織の中には半数以上いる。それで，僕らが使って，何か内輪だけで楽しいっていうやつじゃなくて....，何か全体として効率が上がって管理職が喜ぶようなやつじゃなくて....，社内の仲間に仲間に受け入れてもらえるような物を作りたいと思っています。

ＭＭＬに対する期待というのは一言なんですけれども。ホストのパワーが凄くあって....，と言う話を先程されていましたが，私が使ってると，どうしてもそういう風には感じられなくて，毎朝満員電車に揺られてるわけですが，それとほとんど同じですね。ぎゅうぎゅう詰めにあって息もできない位な環境なんですね。それで，それから少しでも解放されるなら喜んで受け入れるよ....，というのが僕の考えです。それから，発表については，たぶんＭＭＬからはずれると思います。申し訳ありません。

それで，先の社内の開発環境ですが，根本的に考え直さなきゃいけないと思っています。実は，この３ヵ年は何故か富士ゼロックスのＪＳｔａｒを中心にして開発環境が構築されて来ました。３年半前の話ですが，私が社に入ったときに環境と呼べるものは，ＪＳｔａｒが３台

とその辺諸々しかなかったわけです。そのときに私が思ったのは，ソフトウェアをやる人の仕事っていうのはドキュメンテーションの事なんだなと思って，１年半位は，それを信じて生きて来ました。その時にやった事を一応お話します。

現在，私の位置は，この辺（ＯＨＰ）にいまして，大半の人は，この辺なんですが，端っこに追いやられています。でも出口は近いので，そのうちに何とか抜け出せるかなと思っています。それで，ＪＳｔａｒを中心としたソフトウェア開発のドキュメンテーション環境の充実を３年間やって来たわけです。何か，ドキュメンテーションにやたら力を入れていたわけです。この事によって各種ワークシートが充実し，電子化された設計ドキュメントの充実というのが進められてきました。そうやってるうちに，何とかそのまま汎用機に渡せないの....，という風な希望が出てきました。それで，それはＦＡＣＯＭのＭシリーズに向けて，一応ＪＳｔａｒの上でＴＳＳとＲＪＥが使えましたので「じゃあＲＪＥを使って何とかしようよ」っていうのを今からたぶん３年ぐらい前ですが，私，一応やりましたので，その時の事を簡単にお話します。

その時の目標としては，せっかくドキュメントを電子化して来たんだから，そいつから人の手を介さずにプログラムソースコードを作りたい。それから，再利用をより上位工程からやれば効果が大きいだろう。それから，もちろん途中で紙を経由したくない。それから日本語プログラミングを考えたい。これは今になって考えると大きな誤りです。それから，既存の環境でＭＭＬは全く無かったのでとりあえずやってみたい。

....，という事で，ＪＳｔａｒで作った設計ドキュメントから何とかプログラムソースコードを作るような仕組みを作ろうとしました。それで，その前の状態というのは設計がここで（ＯＨＰ）分断されていて，ここから先が汎用機の世界です。そこで僕らが考えたのは，ここを（ＯＨＰ）単純に繋ぐだけですけれど，この（ＯＨＰ）設計書からソースコードを作りたいという事で，何か環境の統合とか作業の統合，物の統合という話を読み上げましたが今になると恥ずかしい話です。大体，８人月位で作ったやつなんで大した事ないんですが，以上（ＯＨＰ）のような機能に分割しています。

作業の流れとしては，論理設計位のフェーズでのデータ設計からデータ領域定義書とか，プログラム構造設計レベルからプログラムの概要とモジュール構造図。プログラムの詳細設計図から各種詳細情報をＪＳｔａｒの上に作成して，これらを全てＲＪＥを使ってメインフレームに向けてファイル転送します。その時に，簡単なシンタックスチェックと日本語項目の話がありましたので日本語の項目辞書を作成します。それから中間言語と書いてますけれども途中，一応，色々組み合わせがやり易い様に中間的な情報に置き換えて，最終的にシンタックスチェック等が通ればソースコードが出来るようになっています。それで，その時のワークシートですが，大体，７枚位あります。

これを私がやっていて，途中から，たぶんこいつは使いものにならないなと感じてました。何故なら，何度か社内に向けて皆が使ってくれるかなと思ってデモをやったんですが，メインフレームの方のレスポンスがひどく良く無くって，デモをやるのに３０分は応答が無いという様な事が度々で，そういうのを見ていると誰も使いませんよね。それで，結局だめだなと思いました。そして，その時から得た教訓としては，ホストマシンとの接点は最小限度にとどめたいという事。今，そう思っています。また，何か相互にうまく協調し合ってと言う風な事は，やりたくない。出来るならば，ファイル転送一発だけの接点を持って二度とそこは通りたくないというのが僕の考えです。

最小限にとどめたいという事は，そこに送る迄の過程で殆ど完成してないといけないわけですが，ＪＳｔａｒの場合はドキュメンテーションしか出来ません。ＸＤＥか何かを使えば出来るのかも知れませんが，ユーザでは使用出来ないので，こいつは捨てなくちゃいけないなと思っています。

次に，もう少し小回りの効くものにしたい。僕が作ったこれは何となくドキュメンテーションから再利用すれば，ＭＭＬと全く関係はありまんが，たぶん効果が上がるだろうと思っていました。今でもうまくやれば効果が上がるんじゃないかと思っていますが，プログラムをやる人が使うわけですけれども，その人達の作業というのは何も支援していないんです。使う人，使ってもらう人を支援しない物というか，狙いが漠然としていたので，その辺をもう少し，何処にターゲットを絞るか等，特に利用者を直接的に支援する事を考えなくちゃいけないという風に思いました。極めて当たり前の事なんですが。

こういう事。これは完全に失敗。私がやった失敗ですが，僕もソフトウェアやり始めて，まだ１年かその位で，ソフトウェアの開発そのものを何も知らない時に作って

しまったので，こういうものが出来てしまったわけです。それで今は，この3年間でＪＳｔａｒとかドキュメンテーションを中心にした展開というのは終わりにしたいと思っていて，じゃあ今後どうするかというのが今考えている事です。

一番最初にも言いましたが，入った時，3年前には何も無くて「とりあえず入れてから考えよう」と言うスタンスだったわけです。もう3年経ってます。そして，これからというのは既に過去の例があるわけですから，もう少し慎重に行かなければならないと思っています。それで，これからの開発環境を考えるにあたって社内で簡単なアンケートをやりました。そのアンケートの結果を紹介したいと思います。

私は１００人以上の組織の中で仕事をしていますが，その中で「職種はどういう人なんだ」という事と，「開発環境はソフトウェア開発の中でどれ位ウェートが置いている問題なのか」という調査と，最後に一応「MMLに期待があるか」という事を調べました。それを紹介します。

無回答も結構ありました。担当職務というのは独断で分けたところもあります。選択は択一で選択してもらいました。そして，やはりシステム設計をやっている人とプログラム設計をやっている人で6割を越えています。これ（ＯＨＰ）を見て，僕が，これからの開発環境を考える時に，組織の中で何とかしなくちゃいけないと思ってるのは，この7割です。この7割をターゲットにしないとどうしようもないなという事と，この7割が何をやっているのかという事をブレークしないと，また失敗するんじゃないかと思っています。

次に，メインでプロジェクトマネージャをやっていると言う人が２０％いました。と，言う事はですね。１００人という事を考えると5人に1人はプロジェクトマネージャという事になるわけです。環境というのは個人で考えるというのもあるし，なにがなんでも組織という風な気持ちのやり方もあると思うのですが，やはりプロジェクトという考え方で環境を作るというのもやらないとなかなか柔軟に対応できないと思います。この5人に1人の方というのは，今後の展開で非常に鍵を握っていると思うのです。

それから，この担当職種別に「ソフトウェアの開発においては何が一番問題か」というのを択一に選択してもらいました。たぶん，システム設計やプログラム設計をやっているメンバのこのデータ（ＯＨＰ）は，あまり信頼性が高くないと思います。あまり特徴の無いデータなので，何か分からなくてつけているというのもあると思います。それにしても先のプロジェクトマネジメントをやっている人達の観点というのは，開発環境に対しては非常に低いと見る事が出来ます。他に比べるとです。プロジェクト管理やテクニカルマネジメント，それから見積り等に比べて開発環境が問題だという意識は非常に低いので私は非常に困ったなと思っています。プロジェクトを中心に展開して行きたいと思っているのですが，他に比べて低いという様な結果が出ています。

それから「マイクロメインフレームリンクによって開発環境が改善されるか」という事を同じく主たる担当職種別でやってみました。それで行くとプロジェクトマネジメントをやっている人達は無茶苦茶期待している訳です。それに対してシステム設計やプログラム設計等，たぶんMMLに一番恩恵を被るであろう人達では以外に期待が低いという様な結果が出ています。これをどういう風に受け取るかというのは簡単には行かないですが，僕は，この結果を見て，今まで開発環境という様な事をやって来たんだけれども「自分達がやってきた事は何も紳士的で無かったね」という風な気がしました。それと，たぶん期待されていないと言う事です。確かに僕らの宣伝の仕方というのが非常にまずかったのかも知れません。が，実際，中心でやっている人達に期待を抱かれてないという事は，ちょっとショックです。

だから，これからどうするかについて，結論というのは正直言って申し上げられないです。まだ，もう少し調査してみないと分からないのですが，僕が今やらなければいけないと思っている事は，これは別にMMLに限った事ではないのですが，開発環境をこれから考えていく時に，もう少し現場での開発とユーザの情報をブレークダウンして調査して行かなければいけないと思っています。それと，先程の5人に1人という事が大体分かったので，プロジェクトの情報という単位でつかむ事は，たぶん出来るだろうと思っています。その上で「僕はこれからどうするか」，「どう言う風にしてまとめるか」というのを考えていきたいと思っています。

非常に申し訳ないですが，MMLとは全く関係ない話をしちゃいました。済みません。一応，これだけです。

久保： 少しやってからブレークしましょう。ＨＩＴＡＣの悪口だとかＩＢＭの悪口があってＦＡＣＯＭの悪口位が続くかなと思っていたんですが，井川さん達の話は

MMLの諸問題。平山さんのはMMLの大問題っていう感じの話だったですね。どうもありがとうございました。では，自由にご発言下さい。

野見山（三井銀ソフト）： 三井銀ソフトの野見山です。ＪＳｔａｒの話が出たので話を聞いてみたいと思います。盛田さんの論文を参考にさせて頂いてＪＳｔａｒを１０台増設してしまったばかりですが，平山さんはＪＳｔａｒを諦めて，今後，手をつけない様な事をおっしゃったんですけれども。実際に，ドキュメント，設計書とか沢山入っていると思うんですが，それをどうなさるおつもりですか。今後の参考にしたいと思うのですが。

平山： そうですね。まず，おことわりしておかなければいけない事があります。本当は，社内での前哨戦をこの場を借りてやりたいと思っていたのですが，実行委員長がいなくなってしまったのでちょっとがっかりしています。

それで，これまでのドキュメントですが，僕は正直言いますと，たぶん８割以上はゴミだと思います。ＪＳｔａｒを入れるときの調査とかを見てみるとシステム設計者とかプログラム設計者の作業の何十パーセントがドキュメンテーションに使われているという風な報告がありました。今，自分がやっている事を考えると，そのドキュメンテーションというのは必ずしも清書するための結果を残すためのドキュメンテーションじゃないわけです。色々な事を調べてメモをとるとか，そういうのを目の前で組み合わせて考えるとか等もドキュメンテーションの作業には入っているわけです。

たぶん，その辺がロジック等を考えるあたりだと思うのですが，ＪＳｔａｒでは，たぶんその辺は支援出来ないです。結果として清書する事は出来るのですが，そういうロジックを考える時に色々な情報がダイナミックに出て来て，そういうものをバシバシ入力していく時のスピードについて行かない事と，それに対して殆ど支援していないと言う事です。

そういう意味でドキュメントをどうするかと言えば，もうゴミだと言ってしまったんですが，本当は，チョコチョコと調べたものを，あの様に奇麗に清書する様な形より，もっと簡単に残して行ける様な仕組みを作らないと，後で使える情報の様なノウハウは残らないと僕は思っています。これは僕のスタンスなので，たぶん全体とは違うと思います。

野見山： 分かりました。でもＸｅｒｏｘさんの悪口みたいになっちゃってますね。

平山： いや，そんな事はありません。良いマシンだと思いますよ。良いマシンだと思いますが，僕は使いたくないです。

久保： どうぞご発言下さい。メインフレーマが沢山いらっしゃるでしょ。はい。代表的なメインフレーマです。どうぞ。

佐藤（長野県協同電算）： 長野県協同電算の佐藤と申します。MMLの話は，明日，僕もちょっと絡みます。

今，ＪＳｔａｒの話が出て，ホストと繋いだからMMLの絡みで話されたと思うのですが，僕はＪＳｔａｒというのは使い方を間違えなければかなり効果があると思うんですね。あれで物を作り出そうとか，システムを作り出そうとか，オブジェクトを作り出そうとか，そういう事を考えると絶対ダメですが，そうじゃなくて，いわゆるソフトウェアエンジニアのひとつのＯＡツールで，ワープロの代わりで，もうちょっと高級なやつとして，あれの前に座って物を作り出せないかというと，そうでもない。

例えば，ランダムに色々考えて行って，後でパスを張るというのは，昨日，ハイパーメディアとか，そういうのがありましたが，Ｖｉｅｗ－ｃａｒｄｓみたいなものを使うと出来るわけです。大体そこそこ書いた紙をイメージリーダで読んで，それを合成して，まさに色々なアイコンがあって，それをピックすればパッと飛び出すという様な環境というのは作ろうと思えば出来るわけです。使い方を間違えなければ，そんなにまずい物ではないというのがひとつ。今のは意見です。

それから先のアンケートのMMLに対する期待のところで，かなり現場の人達が否定的で，実際に作業している人達はマネージャよりよく分かっているから期待してないというのは，まさにその通りだと思うんです。ただ問題は，今，平山さんの所で構築されているMMLが前提にあって，それに対して「どうなの」というアンケートなのか，それとも「本来のMMLというものは，この様な姿であって，現実的には問題があるけれど，それがうまく行けばこんな形になるんですよ」という夢。夢そのものを否定しているのかという所がごちゃ混ぜになっているんで，パッとあれだけ見てダメだと思ってシュンとしちゃうのは問題だと思うんですよ。それは，先のＳ

EWBの所でもありましたが，メーカはメーカで確かに売りたくて色々なうたい文句を言うわけで，それを実際に使ったら色々の問題が出て来た。しかし，問題が出たけれどもだから全部否定するんじゃなくて，その問題をひとつひとつ克服して行かないと先に進まないはずで，それだけの事だと思うんです。

立場上「MMLとかメインフレーマは面白くないよ」，「そういう所に無駄な労力を費やするよりも，もうちょっと別の道で，もっと恰好いい方法を考えた方がいいよ」と言うのもあるでしょうが，でも，おそらく今の日本のソフトウェア開発環境の大多数を考えると，それに携わっている人口を考えるとメインフレームの所での仕事ってかなりあるわけで，そういう人達の環境をいかに向上するかという事を考えなればいけないわけですね。

国家プロジェクトでやったシグマの人が大勢来ていらっしゃるので，色々と発言に期待しているんですが，あれも早い話がＵＮＩＸで，あそこでソフトウェアを作ると言うマイクロプロジェクトもあるでしょう。しかし，おそらく事務処理もあるし技術計算もあるわけで，ターゲットとして，実行環境として，メインフレームの上で動くソフトウェアをいかに作るかという事を真剣に考えて来られたと思うんです。そういう色々な思考錯誤があると思うんです。

途中の過程として色々な問題が出たなら，だから，ダメだよって言うのではなくて，それをどういう方向に持って行くのか意味で，立場上，MML推進派です。以上です。

岡本（ＫＣＳ）：　ＫＣＳの岡本です。今回は設備下請けをやっています。佐藤さんのお話については僕もその通りだと思います。それから，先程の方で，今回入れられた１０台のワークステーションは少しも無駄ではないと思います。

それで，平山さんの発表の中で１点だけ訂正しておきたい所があります。それは，平山さんが３年前に遡ると「当社には環境らしい環境は何も無かった」という説明をしましたが，私どもの部門の中には何も無かったという事で，即ち，社内にはメインフレームと言われる大型汎用機の仕組みと数十台のＴＳＳの端末設備がありました。

それから，ＪＳｔａｒが私どもの職場に沢山入った経緯について補足します。私どもの部門では，現在１２６名で仕事をしているのですが，その中の半数は，この３年間に入ってきた若い人達です。ＪＳｔａｒを入れるきっかけは，その前の年にあります。それで，残りの半数の半分はフィールドに出ていて，残ったメンバーは３０人少々になると思います。その人達が事務アプリケーションを開発する中でどの様な工程を主としてやっていたかと言えば，プログラミングの現場は大体離れて，色々なソフトウェアハウスの方々とお付き合いしながらシステムのスペックを書いたり，デザインと言われるような仕事をやっていたわけです。その人達の問題意識がドキュメンテーションに集まったと言う事です。プログラミングという事より，とにかく物を書くという仕事のドキュメンテーションに非常に困っていて，そのときにＪＳｔａｒがあって，その時に選んだツールがドキュメンテーションツールとして...., が，あります。

ツールとしてＪＳｔａｒを選んだ評価は周囲から色々頂いていますが，その評価云々は別にして，ドキュメンテーションというコンセプトが出て来た経緯のひとつをコメントします。

久保：　はい。また職権の乱用をしたくなったのでちょっとまって下さい。シグマの３人。お話願えませんでしょうか。私のポジショポジションペーパをお読みになれば分かりますが，私はシグマの人が来るというので出席した様なもので，来たくないのにわざわざ来たんですよね。どうぞ。

？？（シグマ本部）：　確かに３人来ているのですが，二人はどこかに出掛けて私しか残っておりません。私の方からお話します。私はシグマの中でも特に事務処理分野のツール関係の開発という事で仕事をやっています。特にその中の仕事のひとつなんですが，ターゲット接続というのをやっています。

私自身はシグマに来てから，まだ半年経たないものですからあまり良く分からないのですが，最初にＪＩＰさんがお話されていたような内容というのは，シグマのやっている事に非常に近いです。例えば，ＥＡＧＬＥとかＳＥＷＢのお話がありました。あれもシグマでは，一応，ワークステーション上ですけれども実現しつつあります。MMLという事で言えば，ターゲット接続という事で，シグマワークステーションとメインフレームを接続する機能として一応三つ。シグマとしては考えています。

ひとつは仮想端末と言いますか，ＴＳＳのエミュレーション機能です。それからファイル転送とＲＪＥ。お話

になった事と全く同じですが，同一メーカ間では出来て当たり前だと考えていて，異メーカ間であっても出来るという事を言ってます。実際はシグマのゲートウェイ機能と言いますか，ターゲット接続の機能としては，今言った三つの機能が，一応，富士通さんと日立さんと日本電気さんとＩＢＭの４社で実現されています。それから，ＵＮＩＳＹＳさんがゲートウェイ自体は作っておられて，指定モニタという形で使われている最中ですね。まだ製品ではないという事でちょっと他とは違うんですけれども実際に出来ているという事です。

どういう環境が良いのかという事は私もまだ分からない。必ずしもシグマがお進めですよと自信をもって言えるような所までは行っていないんですが，ひとつ検討の中に入れておいて頂いてもいいかなと思います。

平山：　社内にシグマワークステーションが入っていまして，モニタをやっています。僕は，直接担当をやるわけじゃないですが，特にその辺の所は実際に自分で触ってみたいと思っています。期待出来る出来ないの話ではなくて，それを使ってみて判断したいと思います。

それと佐藤さんの方から，先程，大半の人はメインフレームをやっているというお話でしたよね。僕も僕の殆どの仲間はそこなんです。ですから，何とかその人達にＭＭＬと言ったらこんなイメージが書けるんじゃないかといった様な事をＰＲ出来る材料があればと思ってこの会に参加しています。今はなかなか夢を浮かばせるような話が社内で出来ないので，そういうのがあったら....，と期待しています。あとは坂下さんの方から「理想の....，」といった話をして下さるそうなんで結構期待しています。

久保：　じゃあ，もう休みにしたいんですが，先に，アップロード，ダウンロードを全く人間が操作しなきゃいけないという話。事実だから事実なんでしょうけれども。それを聞いた皆さんが蔑みの笑みを浮かべていたんで，ちょっと僕は日立を弁護したいと思うんですね。

僕にもちょっと思い出があるんですが，基本的に３２７０って言う上でずっと日立も富士通も仕事をしていたわけです。そしてハードが安くなって負荷分散とかのアイデアがすぐに出て来ます。何とかワークステーションだとかインテリジェント端末だとか，そういったところのパワーを利用して使い易くしたいアイデアがすぐ出て来るんですね。私もそういう事を考えた事があります。それで，その時に何とかならないかという話をした時，かなり富士通の中で優れた技術者と言われている人達が否定的だったんです。それは何年位前の話でしょうか。７～８年前でしょうか。ですから，かなり古いです。

今，それから時間が経過して思う事は，その頃は，まだＬＵ６．２というプロトコロルは無かったですよね。それから，サーバクライアントと言うモデルも無かったですね。ですから，やっぱりプロトコルですかね。そういう事を可能にする様なプロトコルの開発が優先課題だったんじゃないでしょうか。またプロトコルそのものもそう簡単に変えられないっていう制約がありますよね。３２７０が非常に世界標準的なプロトコルになっていますから。そういった問題もあって日立さんも苦労されているんじゃなかろうか....，という弁護です。

それでは一応おしまいにして，１５分頃までブレークしましょう。その後，まだありますので恐れいりますが１５分頃までブレークしましょう。

＆コーヒーブレーク＆

久保：　じゃあ始めましょう。深瀬さんの発言を打ち切るような横暴をやりましたが，聞くところによりますと私の弁護は弁護になってないという様な事を言いたかったようです。丁度打ち切って良かったなと思ってます。＆笑い＆

前半はメインフレーマを責める側，守る側，攻防合い半ばしたというところで，ここからは坂下さんがどういう調和の世界を描いてくれるかに期待しましょう。じゃあ，お願いします。

2.3 ＵＮＩＸネットワーク環境から見て　理想のＭＭＬ

坂下（アステック）：　殆ど陰謀だというのがだんだん分かってきました。私は今さっき着いたところなので，前半，昨日の雰囲気，昨日の夜の皆さんの乱れ方とか知らないわけです。ちょっと雰囲気が分からないので，このセッションがあまりに酷いのか全体が酷いのか分かりませんので何とかまとめたいとは思いません。私は普段使っているネットワーク環境の紹介をして，マイクロメインフレームリンクという言葉がありますが，メインフレームもこんな事が出来れば一緒に皆さん仲良く出来るのにね....，と言う関心をお話したいと思います。ですから，たぶん，将来，２～３年のうちには出来るんじゃないかという明るい気持ちがあるんです。それは当たって

いないのかも知れませんが，まず自己紹介から。

私，アステックと言う会社の今はプログラマではありませんので....，毎回言っておかないと会社の方で何をさせられるか分からないのでプログラマではないと言って歩いているわけですが，マニュアル製作と会社のプロダクトの移植をやっています。何故，プロダクトの移植をしているかと言えば，計算機はＵＮＩＸワークステーションしかありませんが，今，ＵＮＩＸワークステーションの管理をしているので，色々なマシンを知っているという事で移植の方もしています。ですから計算機のではないでしょうか。

アステックと言う会社ですが，ここはＵＮＩＸワークステーションで動くグラフィックスソフトウェアの開発と販売をやっています。知っている方はいると思います。ＧＫＳというグラフィックスライブラリの標準があるんですが，その販売です。それとウインドウシステムで動くようなＣＡＤシステム。

ＧＭＷとＷｎｎというのがありまして，パブリックドメインに近いようなソフトウェアの配付もしているので「普段は何で儲けているんだ」と皆に聞かれますが，商売としてはソフトウェア開発です。なにかお仕事がありましたら....，最近は貧乏なので....，

それで私自身何をして来たかと言うと，学生の時代はまだメインフレームしかありませんでした。学生の頃，５年位前はワークステーションは無くてメインフレームだけでした。ですからメインフレームのＶＯＳ３とかＯＳ４等を使って来ました。私は物理の出身ですのでメインフレームでＦＯＲＴＲＡＮ等をバリバリとやっていた様な気がします。

非常に面倒臭かった思い出があります。例えば重い様な遅い様などうしようもない事があります。年末の繁忙期や卒論等の繁忙期の朝にジョブを突っ込むと次の日の朝にＪＣＬエラーで返って来るとか....，です。＆笑い＆

あれはとても悲しくて....，何回も同じ事をやって....，それがメインフレームの時代だったんです。＆笑い＆

その時の大学もソフトウェア資産というものを持っていました。私は高エネルギー物理の実験をしていたので，何処かの何かの粒子をぶつけて，その生成した粒子が何で，エネルギーがどうなる....，といった解析プログラム等がありまして，それは１０年位前から受け継がれているわけです。ＦＯＲＴＲＡＮ６６等の段階で書かれている様な何万行もあるプログラムの保守等をやっていました。

年々手を加えて行くのでドンドン大きくなってとてもやっていられなくなります。これは困ったという事で周りを見回したところ遊んでいた計算機がありました。ＭＶ１００００....，当時，ＶＡＸ－１１／７８０の対抗機として出されたやつですね。それは何とメインフレームばりのメモリが１６ＭＢ，ディスクが２ＧＢ。５年前にしては凄い水準のＭＶが２ユーザ位で使われていました。その上に Kyoto Common Lisp を作っている方が二人だけで使っていらっしゃいました。そのマシンはバリバリとCommon Lispをやっていたんですが「済みません。ちょっとＦＯＲＴＲＡＮを入れたいんですが....，」と言って，ＭＴを持って行って移植した覚えがあります。

非常に大変だったです。ＭＴをまず掛けようとするとやはりＪＣＬのカードを突っ込まないといけないわけです。当時はカードの入力装置が１階でＭＴの入出力センタが地下にあるわけです。それで，まずＭＴのジョブを発行して走って行ってＭＴの装置が開くのを見てＭＴを掛けていたわけです。そんな時に階段で人に遭ってお喋りをしているとＭＴ装置が閉まってしまいまして，また階段を上がるわけです。とっても悲しい思い出でした。＆笑い＆

これでは仕方がないのでフロッピーで読ませる等やってみましたが，結局はＭＴで持って行きました。当時のＭＶは，勿論ＵＮＩＸではなくてＡＯＳと言う特殊なＯＳだったんですが，ＦＩＸＥＤ－ＰＡＲＭのものを読むのにえらい苦労しました。やっぱりメインフレームとそれ以外を繋ぐのは非常に大変だという思い出です。

思い出話はそれ位にして，今，どういった機械を使っているかと言うと（ＯＨＰ）。流石にＵＮＩＸワークステーションで商売をしているだけあってマシンは山のようにあります。別に全てを買ったわけではなくて勝手に来る機械もあります。勿論，ＳＵＮは流行なのでＳＵＮを買いました。それからＧＭＷの関係で立石の機械があります。それからやっとＬＵＮＡが来ました。４．３ＢＳＤの機械を初めて触ったわけですが，まだ製品（ＬＵＮＡ）が落ち着かないのでちょっと大変です。あまり悪口を言うと悪いのでやめときます。それからＮＥＷＳがあります。そして日本では少ないでしょうけれど，Ｍａｃ２の上でＭａｃ－ＯＳを使わないでＵＮＩＸを使っています。Ｍａｃ２を使っている人にはおこられてしまいそうな事ですが，確かにＡ／ＵＸは使えないです。Ｍａｃ２はＭａｃ－ＯＳで使う良いハードが出揃いました。それからＡｐｏｌｌｏのＤＮ３５００がＤＯＭＡＩＮ－

ＯＳで動きます。それからＩＲＩＳ－４Ｄ。Ｓｉｌｉｃｏｎ ＧｒａｐｈｉｃｓのグラフィックスワークステーションでＲ２０００が載っていて凄く速いです。パフォーマンス的にはＳｕｎ－４以上出ている気がします。それからターミナルが山のようにあります。昔はＰＣ－９８がかなりあったのでそれを収容するためにターミナルサーバを入れました。その他，レーザプリンタ等。

ですから最近はメインフレームと縁の無い生活です。この絵（ＯＨＰ）はＭａｃで書いたんですが，ＵＮＩＸでこれだけの絵を書こうとすると大変です。こんな時はＭａｃの方が良いですね。こんな風（ＯＨＰ）に（機械が）グジャグジャとあります。先進的な電総研の様に複数タイプのＥｔｈｅｒｎｅｔがあるわけではなくて，これ位の小規模の会社ですが，これ位の会社でもこの程度は簡単に作れるという事です。

日々こういった環境の中で仕事をしているわけですが，こういった環境で何が出来るかという話をしようと思います。今までの御二人....，私自身もそうだったのですが....，技術者のプレゼンテーションには高い理想があって「今の僕達にはこういう事が出来ない....，こういう事が出来ない」とやってしまいます。「じゃああなた達は一体何をしているの....？　一体何が出来るの....？」と散々に言われるので，これからはプレゼンテーションにも力を入れて，何が出来るかという事を最初に話さなきゃいけないという気がしています。

それで「何が出来るか」ですが，何が出来るかという話をすると話をしている人が「嘘つき」と言われるわけです。＆笑い＆　でも「出来ないよりはましだろう」という事で，理想はあるかも知れませんが「かなりの所までは出来てるんじゃないか」，「そんなに捨てたもんじゃない」と軽くまとめていくところが最後だからいいんじゃないかと思います。

それで「何が出来るか」です。まず「資源の共有」です。つまりネットワーク環境で何が出来るかという事ですが資源の共有です。資源とはディスクとかＣＰＵです。それとレーザプリンタの様なデバイスです。

それから「機能分散」です。ひとつのマシンで全部打ち込んでやっていたものが得意なものや，これ位は出来るでしょうという感じのマシンに機能分散して置いておけるという事です。

それから「マルチプロセス」です。複数のプロセスが協調して動く様なプログラミングが可能になるという事です。「プログラムが巨大化して行って非常にデバッギングが困難になるといった事も機能ごとに分けて複数のプロセスが協調して動く様にすれば，そのプロセスごとにデバッギングすれば良いじゃない」と夢の様な話があるわけです。そうなるとお互いの通信の所のデバッグが大変になりますが，この様なパラダイムもだんだん実現出来るんじゃないかという事です。

それから後は「パフォーマンスの改善」です。マルチプロセシングして，機能の分散が出来て，資源の共有をすれば，全体的にメインフレーム一個に集中しているよりは何かで分散してやれば少しは改善されるという事です。勿論これは理想です。これには人が制御してやらなければいけない事もありますしマシンが勝手にやってくれる事もあります。今の段階ではまだまだ人が触ってやらないとうまくロードがバランスしないといった問題もありますが，それも追々解決されていくでしょう。たぶん....，そのために大学の人達は研究しているわけで，そういう人達には税金払っているわけで，そういうのをちゃんと研究してもらわないといけないわけですね。＆笑い＆

「何が出来るか」という事をもう少し話します。「資源の共有」という事についてです。ここはＵＮＩＸの話になってしまいます。会社にはＵＮＩＸマシンしかない事とメインフレームやＶＭＳ等とうまくファイルを共用しようとなるとちょっと大変ですがＵＮＩＸワークステーションならばかなりの事が出来るからです。ＵＮＩＸに詳しい方なら分かると思います。マシン固有のファイルだけは共有出来ませんがオブジェクトが共通のものは共有出来るという事です。それと各プロジェクトが全てのマシンを共有出来るという事です。この様なファイルの共有等はＮＦＳやＲＦＳがあれば比較的簡単に出来ます。

ＮＦＳは比較的ＯＳに依存しない構造になっているので，移植は非常に困難ですがＶＭＳやＩＢＭのＡＩＸの上で動く物もあるそうです。だから一応メインフレームをファイルサーバにするという事も可能ですね。＆富士通の久保さんに質問＆　ＵＴＳにもありますか？

久保：　たぶん....，

坂下：　ですから，ファイルの共有位はＵＮＩＸを導入すれば出来るという事です。但し，メインフレームの上だとバーチャルマシンの上でＵＮＩＸのＯＳが動くわけで，ＭＳＰとＵＴＳがファイルを共有出来るかどうかに

ついては私は知りませんが，そこら辺がうまく出来れば共有も不可能ではないという事です。

ファイルの共有というのは誰でも思いつく事ですが，デバイスの共有等もネットワークに繋がっていれば簡単です。レーザプリンタのＩＭＡＧＥＮシリーズはＥｔｈｅｒｎｅｔのインタフェースを持っているのでネットワークに噛んでおけば好きなマシンからＩＭＡＧＥＮへファイルを投げられます。そうすれば特定のマシンだけが重くなる事はありません。ネットワーク負荷は上がりますがＥｔｈｅｒｎｅｔで１０Ｍｂｐｓ，ＦＤＤＩを使えば１００Ｍｂｐｓ越えるわけですからネットワークの負荷というのはそんなにうちの会社では問題になっていません。

それからテープドライブです。テープドライブなんていうのは何処かのマシンに一個付いていれば良いわけです。うちにはＳｕｎが５台ありますがテープがついているのは１台です。値段を安くするためにテープドライブは買わないようにしています。ＭＴ装置等は各マシンに繋がっていても仕方がありませんからネットワークで共有しているという事です。

それと，案外貴重な資源としてＣＲＴ。特に，うちの会社はグラフィックスの開発をしているのでカラーのグラフィックス等が必要になりますが，開発にはモノクロのモニタを使ってフィニッシュのところで何処か向こうの方のマシンにネットワーク経由でウインドウを開いておいて「ＯＫ」。カラーグラフィックスのカラーのＣＲＴなんかを共有するという事をやっています。

次にＣＰＵです。勿論，オブジェクトが違う等の問題もありますが，ある程度小規模なアルゴリズムの検証だけ自分のローカルなマシンで検証して「後はＣＲＡＹで.....，」といった事が出来るわけです。例えば電総研ですが，あそこにはＦＸ８，Ａｌｌｉａｎｔ等のパラレルコンピュータがあって，こちらでちょっとやっておいてＡｌｌｉａｎｔ等で巨大な計算をするという事を彼らは実際にやっています。ですから数が少なくてもネットワークに繋がっていれば，それぞれの機能を目標ごとに使えるという事です。

それとネットワーク環境になって来るとプログラミングの概念が大分変わって来ました。やはり複数のプロセスで協調して動く様なシステムが大分増えて来ました。少し前に発明されたサーバクライアントモデルですが，あれが今の所は非常に幅を効かせています。つまり，あるサービスをするサーバプロセスがいて，そのサービスを受けたいプロセスはサーバにネットワークファイルでコネクションを張って使うわけです。

良い例はウインドウシステムでしょう。ウインドウシステムはサーバクライアントのイメージから言うと逆になりますが，それと最近のかな漢字変換です。かな漢字変換で辞書を各個人ごとに持つのは単にディスクの無駄になります。ですから，かな漢字変換サーバがいて，それに対して変換を要求すると言ったシステムの方がやっぱり使い易いわけです。

それからデータベースですね。データベースも最近ではネットワークにフロントエンドが１個あって，データベースを管理する様なプロセスが１個あって，それに対して要求するといったシステムが大分出て来ています。

それと後はグラフィックスですね。これはウインドウシステムが出来たからなんですが，先の様に向こうの方のカラーターミナルに対して出すなんていう事が簡単に出来るようになって来たわけです。これはどちらかと言うと，まだ一体化のシステムなんですが今後は複数のプロセスがごちゃごちゃと協調して動くような対等な分散プロセスというのがきっと概念的に出て来ると思います。現実に会社の中でプロダクトレベルでこういった物にはまだ手をつけられません。サーバクライアントモデルのプロダクトというのはやっと作れるようになって来たかなという感じです。

それで，ああ言ったネットワークがあると何が便利になって来たかと言うと機能分散が図れるという事です。今は色々なマシンがあります。また，色々なサーバとして単純にネットワークインタフェースを持って単機能しか持たない様なシステムがかなり出てきています。その良い例がターミナルサーバです。

ネットワークに噛むターミナルの口が１６個ある。それで，ホストには一切ターミナルを付けないわけです。うちの会社でもＳＵＮとか他のＵＮＩＸマシンに付いているターミナルはありません。レーザプリンタもターミナルサーバの先に繋がっていますから非常に負荷が減るわけです。

ホストに対するターミナルサーバで，うちで成功しているのはＥｎｃｏｒｅのＭｕｌｔｉｍａｘのシリーズと，それに繋がるＡｎｎｅｘです。ｂｉｔの最新号等にウルトラマックスというマルチプロセッサマシンの紹介が出ていますが，あれはターミナルコードが一切用意されていなくてターミナルは全部ターミナルサーバから使うようになっています。こういうネットワークで機能分散が

図れるわけです。

それからファイルサーバですね。最近，ネットワークのニュースで見たんですが，ＮＦＳのプロトコルとＥｔｈｅｒｎｅｔのインタフェースを持ったＮＦＳサーバと言う機械が発売されている様です。それに巨大なディスクが付けられて光磁気ディスクで自動的にマイグレーションしてくれると言ったシステムもある様です。

ですから，色々な機械がネットワークのインタフェースを持っていれば....，ネットワークのインタフェースと言うのは，ＴＣＰ／ＩＰが殆どですが，そういったインタフェースを持っていればネットワークを通じて色々な機能が増設出来るという事です。ターミナルをもっと繋ぎたければターミナルサーバを買えばいいわけですし，ファイル容量が足らなければファイルサーバを買えばいいわけです。今，Ｅｔｈｅｒｎｅｔインタフェースを持ったＳＣＳＩのディスクを繋げてくれるものだとかＩＳＤＮのインタフェースを持っているもの等もあるみたいですね。ですから，ネットワークを通じて色々なデバイスを使えるというわけです。ホストにがんがんと物を付けていくんではなくて，ネットワークにそういった物を付けていくといった事が，今，だんだんと行われています。

それから，今はなかなかヒット作じゃないかと思うんですが，Ｘウインドウ端末と言うのが発売されています。日本だとクボタコンピュータで，アメリカだとベルシステムズとかＩＢＭ／ＰＣのＡＴのカードの形で供給されています。それと台湾の会社も出してる様です。つまり，Ｘウインドウのサーバとネットワークインタフェースだけ持っているわけです。これをネットワークにバヒョっと挿すと，いきなりマルチウインドウターミナルが出来ます。クライアントは勿論他のホストマシンで，即ちＵＮＩＸマシンで動かすわけですが，これだとＸウインドウ端末の管理も要らないし，何時落としてもいいわけですから非常に簡単です。

この様にネットワークを使えば色々な機能分散が自然と行われて行きます。勿論，これは理想でありまして，会社の中でも特定ホストにプロセスが集中する様な事が起こっています。しかし，昔に比べて，こういったネットワークに繋がる機器を増設すれば負荷が簡単に分散出来るというのは非常に大きなメリットです。

この様に話して来たわけですが，こういった事が実際にだんだんと仕事の中に入って来た時には，どういった問題と影響があるかという事をちょっとお話します。つまり，今までの様に単純にポーっとソフトウェアを作っていただけでは明らかにまずくなります。うちの中でもやっぱり色々な人がいまして，確かに今までみたいに簡単に作っていたら明らかに問題になるという事があるわけです。

それはですね。色々なものがトランスペアレントに，透過的に見えるという事とソフトウェアのポータビリティが重要になってくる事です。

何処のホストでソフトウェアがコンパイルされて実行されるのか分からないわけですし，どういったターミナルの上で実行されるかわからないわけです。Ｓｕｎで作っていてＯＭＲＯＮの方で虫等が出て....，そういった事がすぐ起こるわけです。

それからネットワークで何でも繋がって行くわけですから，何処かの会社から全部買うといった事が出来なくなります。必然的にマルチベンダに陥らざるを得ないわけです。そうなると大変です。例えばメモリは何処から買って，Ｅｔｈｅｒｎｅｔケーブルは何処から買って，保守はどうなるんだ....，そういうのがドンドン出て来るわけです。

この中で皆が困っている事。商売の話をしますと「システムインテグレーション」です。ネットワークに機器を接続しようとした時に，どの様な構成で，どれだけメモリを買って，どれだけディスクを買って，どんなワークステーションをネットワークで繋げれば良いかという事をサジェスッションしてくれる会社は今のところ殆どありません。ですから，うちの会社でも次に何のマシンを買うか凄く悩むわけです。ですから，お金を払ってもいいからネットワークの管理とそういった保守をやって欲しいところなんですね。そうなると一部では禁句ですけれども標準化という言葉が出て来ます。

標準化の悪い意味は色々あると思うんですが，否が応でもある部分は標準化して，それを共通に皆で使って行かなければいけないという事です。それがだんだん重要になります。

それで最初の話ですが，今，会社ではファイルが１０何台のマシンにディスクとしてぶら下がっていて，それぞれ全体的に透過的になっています。こちらのマシンでコンパイルしていてもファイルは向こうの方のマシンでがんがんまわっているわけです。だから自分が何かロードが重いと思っても何処のロードが重いか分からないわけです。ＣＰＵが重いのか，ディスクが重いのか，Ｅｔｈｅｒｎｅｔが重いのか全然分からないわけです。つま

りネットワークファイルにファイルがあるとネットワークなんか意識しなくなるわけです。

それから，今は会社の中にはＥｔｈｅｒｎｅｔしかありませんが，Ａｐｐｏｌｏがあるので必然的にトークンリングとかも張らなければいけないわけです。が，そうなるとネットワークが物理的な媒体である事を全く意識しないわけですね。プロトコルは，今は通信に強いバークレイ系ＵＮＩＸの方が流行ですからＴＣＰ／ＩＰ等を使っていますけれどもＶＭＳの世界ではＤＥＣｎｅｔですね。Ｍａｃの世界ではＡｐｐｌｅＴａｌｋがありますしそのうちＯＳＩ等が出て来る。既に今でもＭａｃ２ではＡｐｐｌｅＴａｌｋとＥｔｈｅｒｎｅｔのゲートウェイが存在するわけですから，それこそ，もうプロトコルも全く関係がありません。Ｍａｃの中からＵＮＩＸの資源を利用するといった事が簡単に出来るわけです。ですから，プロトコルがどうだとかはだんだん関係が無くなって来るんじゃないかという感じがします。

それと後，ハードウェアですね。ハードウェアもネットワークに繋がっていてＣＰＵが何であれ何でも良くなって来るわけです。メインフレーマのワークステーションでも....，メインフレーマはちょっとしんどそうですが本当はここら辺が楽になってくれるととても嬉しいわけですね。例えば，メインフレーマでＵＮＩＸを....，あえてＵＮＩＸと言いますが，私自身は「ＵＮＩＸ万歳」とか，そこまでは言いません。たまたま便利だから使っていると言うスタンスですが，ＵＮＩＸを使っている人達から見ればメインフレームというのは凄く面白そうなんですね。勿論，あの上でＵＮＩＸが動いてくれればとてもいいんです。

例えばＵＴＳを使っているセンターの人達に聞くと物凄く速いと言います。ＳＵＮ４なんか目じゃないと言います。ディスクなんか物凄く速いわけです。ですからとても羨ましい。あの資源をもっと私達からもっと有効に使いたいわけです。「計算は明らかにＳｕｎでやるのは馬鹿らしくてメインフレームを放り込みたい....，」そういった事が出来るようになって欲しいのです。

それからオペレーティングシステムです。今ここでＭａｃの話が出ましたが，ＭａｃであろうがＵＮＩＸであろうが何でもいいわけですね。例えばＤＥＣのＶＭＳがあります。ＤＥＣウインドウ等を使えば一応出るわけです。ＤＥＣウインドウはＸウインドウのＶＭＳ版でＶＭＳで動くＸウインドウですが，ゲートウェイがあればＵＮＩＸのクライアントをＶＭＳの上に表示する事が出来るし，逆にする事も一応出来るらしいです。ですから，いずれにしてもだんだん関係が無くなって来る。

だんだん理想になりますが，今，騒がれているＩＳＤＮとインタフェースすればＴｒａｉｌＢｌａｚｅｒ等を高いお金を出して買わなくても６４Ｋｂｐｓの通信が保証されますね。６４ＫｂｐｓでＮＦＳをマウントすると大変そうで，ちょっとそれは無理ですが，簡単なプロセス間通信位なら出来ます。そういった意味ではＩＳＤＮはちょっと遅過ぎるわけで１Ｍｂｐｓ位は欲しいところですね。プロセス間通信でＸのクライアントを動かすとなるとかなりの通信が出ますから本当はもっと欲しいところですが。

それから後は時間です。こうなってネットワークに繋がっていると２４時間運転にならざるを得ません。何時でも何処でも使えるっていうのが理想でしょう。

こういった事が簡単に行われるためには何が必要かという事について敢えて言いますが「役に立つ標準化をして欲しい」という事です。

まず，ネットワークファイルではプロトコルです。ＯＳＩでもいいです。何でもいいですが，ちゃんと使えるプロトコルを決めて欲しい。

それとデータですね。例えばＪＩＳの漢字コードなんてあまり頻繁に変えないで欲しいですね。古いファイルを持っていてプリントアウトすると名前の形が変わっているなんていうのはハッキリ言って困るわけです。ああいった事もちゃんと本当は考えて欲しいわけです。

それから後はインタフェースです。ハードウェア的なネットワークのインタフェースが特に欲しいわけです。こうなって来ると色々なマシンに繋がって来るわけですから....，

Internationalisation

と言う言葉があって....，

Localisation、

私が日本ＵＮＩＸユーザ会の方に所属しているのでこういった言葉が今の流行なんです。つまり，日本国内だけで商売していてはまずいわけです。何か知らないマシンが出て来てネットワークで使えるからいいじゃないかとか言う話ではまずいわけです。アメリカに行った時に日本の中でちまちま作った日本の中の事情に合わせたマシンを使っているわけには行きません。アメリカのをローカルにしなければいけないでしょう。例えば漢字コードですが，特にＥＵＣなんていうのは英語と漢字。つまりＡＳＣＩＩと何かそれは自由に決められますが同時に二

つのコードセットしか使えません。でもそれじゃあ困るわけですね。

ネットワークの上に壮大な話をします。世界中のマシンと通信出来る。それから色々な人とメールをやりとりする。そういった時にアラビア語も読みたいだろうし，現実的には留学生の受け入れなんかで会社の中に中国の方とか中近東ですとか東南アジアの方が増えてきているわけで，そうなると明らかに英語と日本語では足らないわけです。少なくとも中国語位は喋らないとまずいですし，アラビア語とかが喋れないといけない。つまり，マシンの上でそれを同時に編集出来ないといけなくて，今のソフトウェアの作り方ではきっとまずいだろうというわけです。

好き勝手に話して来ましたが，ネットワークの影響というのは物凄く大きくて色々な事が着実に現実に足下に迫っているというわけですね。ですから，確かにファイル転送が出来ないとか，そういった事をしたいとかいうのも第一歩でしょうが，ボヤボヤしていると本当に悩んで仕事をしていく人に取り残されてしまう恐れがあるんじゃないでしょうか。それに向かって，ちゃんと色々な努力をこれからして行かなければいけないんじゃないでしょうか。そういう話でした。どうも有り難うございました。

久保： すばらしい話でした。理想的かつ現実的なレベルで，これでシャンシャンにしたいんですが，ちょっと時間があるので続きをやりましょう。沢山あるんじゃないでしょうか。坂下さんだけじゃなくて全部まとめてやりましょう。あと１０分位しかありませんので。

海尻（信州大学）： 信州大学の海尻です。大学に対する批判もちょっとあった様ですね。うちの大学でも情報工学科という立場ではメインフレームをやっています。良い環境になっているんですが大学全体ではやっぱり遠メインフレームです。かなり酷い前提条件ですが，それで思いますのは簡単な話です。開発関係の方は「自分で選ぶわけではない」と先程の話にありましたのでちょっと違うかも知れませんが「どうしてメインフレームでないといけないか」という事をお聞きしたいのです。

吉本： 確かに「メインフレームでないといけない」と言うのがあります。それは何故かと言えば「大量のデータ」です。先程も言われた様に確かにディスクは速いですが，そのボリュームですね。何千万件というオーダのデータベースになるとメインフレームでないとどうにもならない部分があります。

それから計算についてです。計算については確かにスーパコンピュータの方が速いですが，それにはそれをサポートするソフトの問題があります。最近，フライトスケジュールを立てるのにスーパーコンピュータが要ると考えたんですが，探してみたらソフトが全然無くて泣く泣く３０９０の上で諦めた...., そういう経緯もありました。そういう意味で，やっぱりメインフレームにご厄介にならざるを得ない様な部分があります。

但し，以前みたいにメインフレームで色々まとめてやろうと言う時代では無くなったと思います。と言うのは...., 何千万件のデータベースなんていうのをアクセスすると，一瞬，他のジョブが一切止まります。検索のために色々な式を組みますね。それを実行した途端にＣＰＵ負荷が完全に１００％...., 他のジョブはピクとも動かなくなります。メインフレームはマルチタスキングでＣＰＵを使いっぱなしにするやつはいない...., 誰かがＣＰＵを使っていたら誰かはＩ／Ｏしてる...., と言う様な考えから出来てるんですが，そういうのは非常にやりにくくなりました。

そういう意味で，昔みたいな「メインフレームの上で仲良くやりましょう」と言うのは不可能だし，そういう意味では，メインフレームはデータベースと計算マシンとして残って行くんではないでしょうか。そういう意味では，メインフレームはデータベースサーバであり，ＣＲＡＹではちょっと出来んような計算用サーバという様な形に絞られて来るでしょう。

マイクロメインフレームリンクを私としてこれから考える場合，やはり「ネットワークの中のメインフレーム」。それだったらメインフレームじゃなくて単なるサーバですが，そういう考え方でMMLという考えを進めて行かなければ，この言葉は不適切じゃないかと思います。

久保： そうですね。MMLというのが適切でないと言うのは大体のコンセンサスじゃないでしょうか。

臼井（ＪＩＰ）： それは大賛成ですね。昔，私の生まれる前に戦争がありまして，日本の海軍は戦艦大和を守るためにいっぱい飛行機とか駆逐艦をそこら辺じゅうに引き連れて海を走り回ったおかげでガソリンが無くなって負けたと言うしょーもない話があるんですが，今のメ

インフレームにはMMLと言う延命策を使って悲壮な覚悟で延命を図っているという最後の足掻きが見える様な気がするわけですね。

現代戦というのはやっぱりパーソナルな環境ですし，それは戦闘でもその通りで，飛行機が現れてからは戦艦の役目っていうのも来泊なんで，大体そういう受け口が出来た時には航空母艦と言う風に後ろでちゃんと援護射撃をするだけにとどめないといけない環境になっているのに，まだ爆弾を積んで走ろうなんて言う様な事を考えているからMMLがどうのこうのと言う話になって来るのであって，先程言われたようにデータの提供だけとか補助だけに徹すべきだと思うんです。

そういう役割を完全に切り離して戦争しようとか処理をしようという様な大それた事を考えないで，単なるバックアップ機能という事で存命を図らないと，そのうち皆こけるんじゃないかという風に思います。

久保： 今の話なかなかいいですね。SEAMAILに書いて下さい。今のは説得力があります。

井川： 何故メインフレームか...., 海尻先生の質問に対するお答えなんですが，たぶん実際に自分でプログラムを作って，自分で流して，自分で処理の結果を見守って，結果を自分で使うという様な使い方をされてる方は明らかに不思議だなと思われてると思うんですね。UNIXだとプロセスの状態も全部自分で管理出来ますから。だけど，業務ってなんでも業務なんですが，大きなコンピュータで処理をして結果を出す事で商売をするという観点に立った時に何から何まで全部自分でやらなくっちゃいけない環境っていうのは，これは非常に苦しいですね。

大型機だと機械の管理は誰がする...., 掛ける紙の管理は誰がする...., プロセスの状態を見守るのは誰...., という風に分業が出来ます。とてもじゃないけど人間のスケールで計れない様な巨大なジョブでも流す事が出来ます。「いいじゃない」。「UNIXのワークステーションで大型機の倍位時間がかかるかも知れないけれど結果だけ出れば」という風に，実は私もたぶんそういう言い方をしてしまうんですが，やっぱり，その辺に微妙な違いがあります。大型機でやる時っていうのは分業化が出来る。まさかNEWSやSUNでマシンの管理を分けてしまうわけには行きません。明らかに分けて，そこでバランスをとりながら運用する妙と言うものがメインフレームにはあって，それがたぶん，これからもメインフレームが生き続けていくひとつのポイントになるんじゃないかと思います。

久保： 今のには坂下さんからちょっとコメントがあっていいと思います。

坂下： はい。SUN1台を業務で管理するのは非常に馬鹿らしい事ですが，私達の会社の中ではネットワークに繋がったシステム全体が計算機であってSun1台が計算機じゃないわけです。そうなると管理者は勿論要るし，プリンタの対応を考える人が要るわけです。ですから，そういう類とはちょっと違うような気がしますけれども。

久保： ちょっと突っ込みたいですが....,

田中： 坂下さんのペーパの一番最後に何か理想的な使い方みたいなのが書いてあるんですけれども。

坂下： 冗談です...., 捨てて下さい。

田中： いや。重要だと思うんです。ホストとの繋ぎっていうのを太くする事が出来たとして，その後で，どういう使い方をして行けば良いかというのをパネラーの方達にちょっとお聞きしたいと思います。

井川： 繋いだら何が出来るかと言う話だとして私がやってる立場から言えば「基本的にメインフレームでやらなくてもいい事はメインフレームでやらないようにしましょう」と言うのがそれです。それが目的ですし，負荷が軽くなるとか環境が良くなると言うのはその結果。今はそれだけです。

平山： コンピュータの使い方をお話するわけですか？

田中： 今までは繋ぐための難しさみたいなのが論点だったと思うんです。それでは実際に繋がった時にどういった具体的な使い方をするか。例えば，コンパイルはホストに任せて，dbxみたいなのがワークステーション上で走っているのか。あるいは画面にリストが出て，そこを反転してひょいと直してコンパイルしろと言えばそのままホストに言ってコンパイルして来てくれるとか。そういった様な使い方のイメージがあればお聞きしたいと思ったんです。

平山： あれだけ言われちゃうと「何かそうですね」となっちゃって，ちょっとうまくお答え出来ません。

久保： 割り込むんですか。

上妻（熊本テクノポリス財団）： いえ。今のお話。
久保： はい。どうぞ。

上妻： 熊本テクノポリス財団の上妻と申します。去年まではＤＥＣでＤＥＣｎｅｔ-ＤＯＳというマイクロメインフレームリンクの製品を扱っていました。何か宣伝みたいになっちゃいますが使い方について少し説明させて頂きたいと思います。まず，何故マイクロメインフレームリンクなんていう物が出てきたのかと言う話がありまして...., その辺から始めていいですか。
久保： はい。

上妻： 確かにユーザとしての要求があったと思うんですがメインフレームの延命策というのは確かです。プロセスサーバ，プリントサーバ，ファイルサーバ，ターミナルサーバ等，色々な使い方が出来ます。そういった意味で存在意義というものをそういう所に求めて行くというのは皆が考えた事だと思います。

ちょっと話がアメリカの方に行きますが，ＩＢＭ－ＰＣが数百万台売れて...., 今は一千万台越えてますが，売れている段階でＬｏｔｕｓ1-2-3なんかが数百万本売れるわけです。そうすると現場レベルでアプリケーションとしてＬｏｔｕｓ1-2-3をどんどん使って行きます。ＩＢＭ－ＰＣの事を1-2-3と覚えていたという人がいる位の環境になったところで，皆はそれをフロッピーとかハードディスクに持っているわけです。ところが管理者にしてみればそれが手に取って見れないし，それを社内メールみたいに...., 電子メールじゃなくて社内メールという形でフロッピーを集めて統計解析をかけたりする所が出てきます。酷い所になると私蔵してしまう。

現場だけが分かっているという事になっては非常に困るという事でデシジョンサポートシステムという流れが出て来たました。そこで，それを支援するためにマイクロメインフレームリンクが出来てきたという風に私は理解していますし，そう言った意味では管理者側から需要があったわけです。

次にＤＥＣｎｅｔ－ＤＯＳですが，ＭＳ－ＤＯＳの上で動きます。普通だとＡドライブとＢドライブがフロッピーになって，ハードディスクが入っていればＡドライブがハードディスクでＢドライブ，Ｃドライブがフロッピードライブになります。その後，ＤドライブとかＥドライブなんていうのを仮想ディスクとして定義しておくとＶＡＸのおなかの中に仮想ディスクを作る事が出来るんです。

例えば，一太郎とか1-2-3，Ｍｕｌｔｉｐｌａｎ等のアプリケーションから普通のＭＳ－ＤＯＳのハードディスク的なイメージでそれを使う事が出来るわけです。大体１ボリュームあたり３２ＭＢ迄を４ドライブ同時に開く事が出来るという事で，パスワードさえ知っていればそれを皆が勝手にアクセスする事が出来るわけです。今となっては，ＭＳ－ＤＯＳで１６ビット迄サポートされていますから必要ないですが，非常に狭い空間でキチキチにやってどうしようもなかった頃「データベースなんか作れないよ」と言う話だったのが，そういう物を使う事によって「小間切れながらも使う事が出来る様になりました」と言うのがひとつです。

それから細かい話ですが，一太郎で文書を作る時でも社内文書等は大抵フォーマットが決まっていますね。例えば，そういったフォームが新しくなった時にフロッピーで配るのはなかなか大変で，結局プリントしたものをコピーして手書きで使うみたいな事をやっちゃうわけですね。その様な時に，それをやめて先の仮想ディスクが使えるわけです。仮想ディスクというのはハードディスクに比べるとやはり遅いです。従って，そんなに頻繁にアクセスするものには使えないわけですが，雛形になるフォームをそこに入れておいて，それを皆が勝手に読み込む事が出来る様にしたり，ソフトウェア開発プロジェクトの中でライブラリをそこに置いて，どんどんアップデートされるものを登録しておいて，それを皆で使う様にする等，そういった共有の仕方や使い方があると思うんです。

それと坂下さんもおっしゃっていました資源の共有の話に関係して...., ＰＣを買う度にプリンタが１台１台が増えて行くわけです。そうすると，まだ私達みたいな地方人はいいんですが，東京あたりになるとプリンタの占めるスペースだけでも莫大な家賃になって来るわけです。特に，昨今のラップトップ等でスペースを効率化しようとしている時に，やっぱりプリンタを買ってるわけです。「これは勿体ない」，「どうしようか」と言う話になって来た時に，Ｅｔｈｅｒｎｅｔ上にＲＳ２３２Ｃポートをつける等してそれにプリンタを繋ぐ。「ＤＥＣｎｅｔ－ＤＯＳでＶＡＸの下に繋がったプリンタがそのまま使えますよ」と言うイメージで売っていたわけです。レーザプリンタを１台繋いでおけば，１台あたりの性能は格

段にいいわけですから使い勝手は良くなってトータルのコストは下がるし，しかもスペースメリットがありますからその点でもコストが下がるという事で，一応いい事ずくめの話です。

ところが，ＰＣ１台あたりのＤＥＣｎｅｔ－ＤＯＳではソフトが２０万円，ハードが２０万円という事で４０万円掛かるので，余程付加価値が認められる所でないと「ソフトウェア開発環境にそのような物を入れる位ならＥｔｈｅｒｎｅｔに直接に繋がるワークステーション買っちゃうよ」と言う話になりがちです。

かなり難しい問題ですが，ちょうどＤＥＣｎｅｔ－ＤＯＳが出始めた頃の日本の場合は，むしろアプリケーションレベルで利用者側からの要求が結構高くて，あちこちのお客の所を廻ったんですが「ＤＥＣｎｅｔ－ＤＯＳがあったからＶＡＸを買ったよ」と言うお客が多くてびっくりしました。「何故？」と思ったんですが，要は従来のＰＣがあまりに早く陳腐化した事から来てるわけです。「償却しきれないうちに使い物にならないでは困る」。それでどうするか。

工場等であれば，色々な部品を流す中でＰＯＳ端末として使う手があります。部品票みたいな物をバーコードで読み込んで，その９８で読み込まれたデータをそのままタスク間通信でＶＡＸに送って，そのＶＡＸがプロセッサとしてそれを受け取ってどんどん全体の工程管理をして行く様な事をお客の方でやりたがるわけです。

そういったイメージで考えるとアプリケーションレベルの方が日本の場合は先行しているんじゃないかと思います。

今日の，このソフトウェア開発環境の中でマイクロメインフレームリンクが出て来るといった所には，ちょっと違和感がありました。確かに現在は分散環境という話になって来て，そちらの方については坂下さんからお話があった通りなんですが「一応そういった使い方がありますよ」と言う事でちょっと御紹介致しました。

久保：　どうもありがとうございました。

坂下：　いいですか。ちょっと。

久保：　はい。

坂下：　今の話に関連するんですが，ＤＥＣで新しく「３８６何とか」が出ましたよね。あれに私ひどく感動したんです。御存知ない方があるかと思いますので御紹介しておきます。

これもＰＣサーバと呼ばれるもので，８０３８６機能を使って８０８６のプロセスを幾つか動かせるわけです。それで，それにはＣＲＴも何も付いていません。３８６とＭＳ－ＤＯＳ。中身に「３８６何とか」です。それにＥｔｈｅｒｎｅｔのインタフェースが１個付いているお決まりのそれです。どうやって使うかと言うと，ネットワークでＤＥＣの中から動いているＰＣエンジンに入ってｌｏｇｉｎします。そこでＬｏｔｕｓ１－２－３等を動かしてＤＥＣのＶＭＳのＶＴ－１００等に表示します。それで，同時に四つまで使えるはずです。そうするとＰＣを捨てられるわけです。これはなかなかに非常に良くて，何処かＡＴシェアのある所と付き合いがあって....，これのＰＣ－９８を作れば売れるんじゃないかなと思ったんですけれども。何処か作りませんか！？

久保：　それはＤＥＣの製品ですか。

坂下：　ＤＥＣの製品です。「３８６ＷＡＲＥ」と言う製品で，ＰＣは四つ迄しか使えない制限はあるんですが，ＰＣのパワーが足りなくなればネットワークにどんどんつけて行くわけです。それだから，例えばそれの汎用機的９８版をトラップしてＸウインドウのプロトコルに変換して出してやれば，これこそネットワークファイルでＰＣの一太郎が使えるというわけです。だから....，何処か作って下されば買いますけど....！

久保：　なかなかスマートですね。それ....，

池谷（富士通静岡エンジニアリング）：　マイクロメインフレームコネクションのメインフレームについて，ちょっと弁護したいと思います。

私ども富士通でもメモリが１６ＴＢだとか，或いは膨大なプログラムだとか，或いは４ＣＰＵを組み合わせて１００ＭＩＰＳを実現するという様な事をやっています。

いわゆる金融関係....，その様な所では高トラフィックを処理する様なメインフレームが必要で，やはり日本の場合だとそういうものでもってるんだと....，

確かに今はワークステーションやＰＣ等，色々とＭＭＩで良い物が出て来ているんですが，やはり，日本の経済を育てたのは先の様なものだと言う話だし，そういったものを，やはりより有効に運用して行くための開発環境として，ＰＣやワークステーションとメインフレームを上手くリンクして両方共存する様な展開がマイクロメインフレームリンクじゃないかなと私は思います。

最後のパネラの坂下さんの話に，ＵＮＩＸベースで基本的にはメインフレームであってもワークステーションであっても同一のアーキテクチャ，同一のコード系，同一のＯＳ，それから同一の言語と極めて標準化された世界でも尚かつ問題があります...., とありました。

そこで，現実に私ども富士通が抱えている問題です。はっきり言ってＰＣやワークステーションはＵＮＩＸでやります。だけどＯＳはＭＳＰです。コードはＥＢＣＤＩＣです。それから言語はＰＣやワークステーションでも違う様にコンストラクションを実現するために良い様になっています。

こういった世界になって来た時に，分散環境としてどういう様な開発環境を提供するべきなのか。そういった問題を考えると明日のセッションで検討されるＣＡＳＥツール等が先の様な違いを吸収出来るひとつのコンセプトになるんじゃないかと思います。こういったものをうまくＰＣやワークステーション，或いはサーバ，そういった所で実現されて矛盾を隠し，それからホストでも運用出来るプログラムを生成する。そういった事がひとつの理想的な解じゃないかなと僕は思っています。現実にかなりドロドロした問題を抱えている状況でそういった解を備えていなくちゃいけない。そう思います。

久保： ちょっと実行委員長から圧力が掛かってます。「ＵＮＩＳＹＳの人にも発言の機会を与えなさい」という圧力を受けています。どうぞ列に並んでおいて下さい。何人でも結構です。

山口（電力計算センター）： 電力計算センターの山口です。私は先に話題になっていました日立の大型機のユーザです。昨日のパーティーの自己紹介の時で言いました通りＦＯＲＴＲＡＮで大型でゴリゴリやっています。日立の端末には２０２０を使っていてＡＳＰＥＮでエディットしています。現在の私の仕事を少しお話してから意見と質問に行きたいと思います。

ＡＳＰＥＮでやっているのはいいんですが，非常にエディットが遅いので，ひとつパソコンでも使ってみるか...., という事にしました。

それで，ＰＣ－９８の一太郎やＭＩＦＥＳでＭＳ－ＤＯＳのテキストファイルを作って，パソコンにＦＯＲＴＲＡＮでプログラムを書いて，フロッピーで２０２０の端末に持って行きました。それからＭＳ－ＤＯＳのファイル転送を使ってホストにアップロードしてからコンパイルして実行しました。後の細かい修正は持って帰るのが面倒臭いのでＡＳＰＥＮの中でやってしまいます。こういった事を色々考えてやっているうちにＭＳ－ＤＯＳのファイルだったら殆ど自動的にアップロードあるいはダウンロード出来るという事に気がつきました。最低限，ＬＯＧＯＮはしてやらないとだめですが....,

２０２０という端末について簡単に説明しておきます。表画面にエミュレータの画面があります。これは日立のメインフレームに繋ぐ端末画面です。それと裏画面にＭＳ－ＤＯＳの画面があります。このＭＳ－ＤＯＳの画面でファイル転送のプログラムを動かして表画面の端末画面で同様のＩＤＩＴと言うプログラムを動かすとＰＲＯＣが動いて自動的にファイル転送します。その辺を上手く作ると非常に簡単にフロッピーの中のＭＳ－ＤＯＳのファイルをアップロード或いはダウンロードする事が出来ます。

これからようやく意見と質問になるんですが，ＭＳ－ＤＯＳのファイルを転送出来るという事は，結局，どんなファイルでも転送出来るんです。バイナリでもテキストファイルでも扱えるという事が分かりました。という事から，今はフロッピーベースで入れ換えていますが，これがＭＭＬみたいに線で繋がった場合は非常に快適になりますよね。今，私が是非とも欲しいのはそこら辺なんです。これは私の意見です。

もうひとつは質問です。今，私の所には２０２０端末が横須賀事業部という事で４台しかありません。ところが，我孫子に行くと何十台とあります。そして，エミュレータもソフトのひとつですから何十台のパソコンの中に入っているわけですね。そこで，２０２０の中に入っているエミュレータが，もしバージョンアップされた場合は全部の端末でフロッピーを差してコピーしてやらなきゃいけない羽目になりますね。ところが，ダウンロードが出来るという事が分かっているわけですからリモートメンテナンスという事は非常に簡単だと思うんです。つまり，ＭＳ－ＤＯＳのあるソフトがバージョンアップされたとします。それをメインフレームのディスク上に置いておきます。端末を使う人がそれを意識するかしないかは難しいですが，使う時にダウンロードして来る事が可能になると思います。ところが，何と言って良いのか分からないですが，この様なリモートメンテナンスの話は，私の不勉強かも知れませんが，出て来ない様な感じがありますね。

その点でパネラの方に質問です。こういったメンテナ

ンスはどうなんでしょう。あまり考えるべき事じゃないのでしょうか。それとも，やはりやらなきゃいけないけれども，そこ迄は....，と言う様な事なんでしょうか。その辺をお聞きしたいと思います。

久保： 坂下さんがいいね。それには。

坂下： メインフレームは使っていませんが「その様な物があればいいな」と，いつも思っています。ＡＳＣＩＩの深瀬さんは....？何処かに行っちゃいました....？

ＡＳＣＩＩがＰＣ－ＮＦＳと言うのを出してますよね。あれは９８にＥｔｈｅｒｎｅｔのカードを差してＭＳ－ＤＯＳのファイルの代わりにＵＮＩＸのファイルを自分のＭＳ－ＤＯＳのＡドライブとかＢドライブに入れるというシステムですね。それを使うとアップロードやダウンロード等も要らなくて，ＭＩＦＥＳをこちらで動かしてファイルはネットワーク経由で向こうに書かれます。ＭＩＦＥＳのアップデート等も手元のフロッピー等には置かずにＵＮＩＸ側のファイルシステムの方に置いて，それを更新して実行すれば全体が新しくなるわけです。

ですから，大型機でもそういう物があれば便利だろうと思うんですが，何か認識が甘いんでしょうか。ＮＦＳはＵＮＩＸに限られません。例えばＶＭＳでも動きます。メインフレームにＮＦＳの製品があるかどうかは知りませんが原理的には動くはずで，そういう物を使えば現実的に技術的には出来るし，やっている所もあるわけです。そういった物がメインフレームにもあれば，かなり....，ちょっとは使える....，という事ですけれども。

久保： 今のはリモートメンテナンスの答えにもなっています？

坂下： そうです。

久保： やってる所を具体的に教えて

坂下： ＡＳＣＩＩがそうじゃないですか....？ 深瀬さんは何処かに行っちゃいましたけれども。

久保： 東工大もそういう風にやっていると言ってましたね....？ 特に少ない手数でやっていますから。

坂下： 後，東大もそうですね。

久保： 当たり前だと思っている人達なので，たまたま話題にしなかったと理解していいんじゃないでしょうか。やらないのはおかしいと思っている人達だと思います。今のは聞いていました....？ 深瀬さん....？

深瀬： 聞いていないです....，向こうに行っていましたから。

久保： 大事な証言になるので宜しくお願いします。

深瀬： 何も難しい事なくて普通にやっていますけれども。

坂下： ＰＣ－ＮＦＳはＶＭＳもマウント出来るんですか....？

深瀬： 勿論それは問題ありませんが，それが何か....？

久保： ＡＳＣＩＩにシステムインテグレータをやってもらうと，かなりいいシステムが出来るかも知れませんね。

深瀬： 「マイクロメインフレーム」と先程のどなたかがおっしゃいましたが....，ちょっとおかしいと思いますね。

久保： だから「ＭＭＬ」と言うのはやめましょうという方向で大体の皆さんのコンセンサスが出ています。

深瀬： ネットワーク上のひとつの資源としてメインフレームがあるわけです。

久保： そうですね。そういう意識に皆さんなっています。

深瀬： 開発環境としてはこう言う援助策があります。運用環境としては旧来の環境でしか援助されませんという話で，今は開発環境の話をしているわけですね....？

山口（電力計算センター）： ちょっと確かめたかったわけです。要するに深瀬さんの話はメインフレーム上にパソコンのソフトウェアが載っていて....，と言う様な事ですね....？

深瀬： そうです。

山口： 要するに一個しかなくて，それを皆が使っている。

深瀬： いいえ。それを皆がＦＴＰで持っていくわけです。

山口： ＦＴＰで持っていくんですか....？

深瀬： 勿論，使ってもいいです。運用形態はいくらで

もバラエティがあります。

山口： 要するに裏でファイル転送の様な事をしている....？

深瀬： 欲しい人はＦＴＰで持って来ても良いし，自分のファイルが残り少なければ直接メモリに持ってきて実行してもいいです。

山口： 分かりました。どうもありがとうございました。

深瀬： ネットワークを運用するという事は当たり前の事ですので。

上妻： ちょっと今の深瀬さんの発言で気になった事があります。実行系の共有という事で，ホストに実行系を入れるという事についてはライセンスをお取りになっているんでしょうか....？

深瀬： うちがライセンスホルダーで，全てのライセンスを取ってあります。

上妻 取ってる物に限ってと言う事ですね。

深瀬： そうです。全てのライセンスを取ってあります。

上妻： 私がそれをやりたかった時に一太郎の作り方等をユーザからよく聞かれたんです。

深瀬： 一太郎はパソコン上だけで使っています。ああいうのは作ったファイルを共有しています。

久保： いい問題提起ですね。今のは解決しなきゃいけないですね。

深瀬： うちは訴える事はあっても訴えられる事はありません。

上妻： 分かりました。＆会場笑い＆

野中： 日本電気の野中です。なるべく短く行きます。うちではファームウェアを作っています。大体，殆どの人はＰＣ－９８とＭＳ－ＤＯＳで仕事をしています。

うちにもＭａｉｃｒｏ－ＶＡＸを入れて，４．３ＢＳＤを入れて，Ｅｔｈｅｒｎｅｔを張って，ＰＣ－９８をターミナルサーバとして....，と言う風な環境を作りましたが，見事に誰もＶＡＸを使っていません。皆，ＰＣ－９８の上で仕事をしています。

結局，何を欠いていたかと言うと「ＭＳ－ＤＯＳ主体」に考えて無かったんですよ。何かネットワークを繋いでＶＡＸを繋げば「ＵＮＩＸだからいいじゃない」と言う事だったんですが誰も乗って来ませんでした。それは，ファイルのアップロードやダウンロードが非常に不便だったからという事なんです。ＰＣ－ＮＦＳを買うよりもターミナルサーバを買う方が安かったので，お金を節約したわけですが，結局，それは誰も使わないから今から考えると「勿体ない事をしたな」と言う話です。

久保： いい話ですね。悪い例ですけれどいい話ですね。

片山（ＵＮＩＳＹＳ）： ＵＮＩＳＹＳの片山です。ＭＭＬについて色々お話されていますが聞き落としている部分もあると思います。

先程のパソコン上で作ったソフトウェアやデータを共有する話で「ホストに吸い上げて共有化を図れば価値があるだろう」という所がありました。私どもはホストが，かなり主体になる様なメーカですので，ある意味では「ホスト上で持っているデータを部門に下ろして」と，話としては逆になると思います。部門で活用出来るデータという物をパソコン上に落としてパソコン上の一太郎とかＬｏｔｕｓ等で部門で有効に活用していく様な使い方と先程言われた様なパソコンからのデータをホストに吸い上げて使うやり方の両方向のＭＭＬを考えています。

それから，プログラム自体の共有という事でホストにパソコンで動くソフトを入れて使うという話がありましたが，実際，ホスト上で動いているプログラムやデータにアクセスするプログラムと言うのはパソコン上で動くプログラムだけではありません。

例えば，ホスト上で動いているプログラムで，私どものシステムで行きますとＭａｐｐｅｒは，ホストで作られたデータ....，これをＭａｐｐｅｒデータベースと言いますが，そのデータをパソコン側に落とす事によってＭａｐｐｅｒデータをパソコン上で役立てます。つまり，Ｌｏｔｕｓ等で切り張り的な感じに部分的に使えます。部門でも有効なホストのデータを活用して行くといった様な使い方を思考しています。或いはセールスしています。

ＭＭＬに関しては，そういう風に考えているんですが，先程，パソコンサイドから「パソコン上に埋もれているデータをうまくホスト上に入れて共有化すれば....，」と言う様なＭＭＬのお話をされていた気がしたので，ちょっと聞き落としがあるかも知れませんがコメントさせて頂きます。

久保： ＭａｐｐｅｒやＬｉｎｃという人気商品ですが，それをMMLと絡めてネットワーク上にＭａｐｐｅｒやＬｉｎｃを実現するという要素はありませんか....？ 完全にホストの世界の話ですか....？

片山： 現時点ではホストでの話です。商品計画等の話でマイクロ系に落とすか否かという部分を検討している部門もあります。

久保： 篠田さん，お待たせしました。委員長が時間を伸ばして良いと言ってますので，どうぞ気兼ねなくやって下さい。

篠田： 何か事例の報告大会みたいになってしまった感じがあるんですが，先程，戦艦大和と空母の移り変わりみたいな事が面白いアナロジーとして出て来ましたね。「巨大戦艦は空母に転換するべきだ....，」ですね。発言者は何処かに行ちゃいましたが，そこで僕が言いたいのは「空母には甲板が要る」という事なんです。

甲板のアナロジーと言うのは，要するに他の飛行機と交信出来る能力です。そういうのを「全然考えてくれていない」と言うのが，たぶん，ここにいらっしゃる皆さんのコンセンサスじゃないかと思います。

メインフレーマの方々にお願いします。「甲板」を作って下さい。甲板もあまり中途半端な物を作られると困ります。充分な長さを持った物を作って下さい。これは，要するにバンド幅が高いという事を意味します。

それから，甲板に降りても何もしてもらえないで「すぐに飛び立て」と言われても困ります。甲板に降りたら整備して次の戦闘に備えるという事で「サービス」です。要するに，物だけ作って置いておくのはやめて欲しいという事です。そこに降りて，サービスを受けて，また飛び立てるような状態にして欲しい。

適材適所というのは皆さんからコンセンサスが得られている事です。例えば，うちにはＥＴＡがありますが，どう考えてもあの様な機械はデスクサイドには持って来れません。液体ヘリウムや液体窒素等，１００坪位の土地の中に仕組みが入っていて，あの様な物は持って来られるわけがない。だけど，ちゃんと使ってやれば何十ＧＦＬＯＰＳという威力で走ります。あんなのは持って来なくて良いから遠くに置いておいて，しかもデータ転送は幅広いバンド幅を活かして，ＳＹＳＯＵＴ編集なんて馬鹿馬鹿しい事はやめて，ワークステーション側のあちらこちらで出る様になっていれば良いと思います。だから，適材適所と言うのは皆さんから出てるひとつのコンセンサスですし，飛行甲板が要るというのもひとつのコンセンサスですね。

それから，この現状をどうやって完結に....，それは今からの話です。もう既にハードを買ってしまってリース期間がまだ１０年あるという所はどうにかしないといけないですけれども。

これ迄，皆さんの発表や事例をパネラの方も含めて聞いているとメインフレームってちょっと野良猫みたいな所がありますね。何とかして近づくんだけれどもヒョッと逃げてしまうという感じです。

また事例になりますが，僕は色々な計算機がある環境の中で自分がいつも使っている機械は，ＵＮＩＸのＳｕｎワークステーションがあって，ＥＴＡがあって，日立さんがあります。その中でも特に良く使うのはＳｕｎですが，大型機等も絶対に学生実習という所で使わなければいけません。これは苦痛です。エディットが苦痛です。それでＲＶＩと言う仕組みがあります。

ＲＶＩが何をやっているかと言うと，大型機上で動いているラインエディタの出力を掴まえて....，例えば，最初ＲＶＩを立ち上げた時ですが，手前はいつも使っているｖｉの形です。奥で何をやっているかと言うとラインを掴まえていて，そこへ「１行目から２４行目迄をプリントしなさい」というコマンドを密かに放り込んでおくのです。当然，レスポンスいいですよ。画面の中で動くローカルコマンドだけです。だから，これだと多少通信が遅くても１２００ｂｐｓ位だと完全に使えます。これはＸウインドウのサーバとクライアントの関係に少し似ていると思います。また，この様な考え方でアプローチすれば，僕はちゃんとうまい事行くんじゃないかと思います。

もうひとつ。パッケージングをＦＯＲＴＲＡＮか何かで両方向書いた話です。ＦＡＰの計算か何かを卒論にした人がそれを作りました。ちょっと凝っています。要するに「スピードは稼ぎたいけれどもあのエディタは使いたくない」と言う事ですが，Ｓｕｎの上でひとつのパッケージングをして，そこでデバッグからテストラン迄やって，いざ放り込んで....，であれば，言語の仕様の相違とかがありますから今度は向こうの実機でデバッグしなきゃいけないですが，その時にも手前でエディットして，そのデバッグセッションで変わった差分だけを転送するという仕組みを作ったわけです。

だから，ファイル単位とかそう言うのから外れて，も

うちょっと細かい所を振って来るみたいなアプローチが今の段階では，結構必要じゃないかと思います。

久保： ありがとうございました。岸田さんは後のＢＯＦの方で。その他の方はいらっしゃいませんか。

井川： 今の篠田さんのお話を聞いてて，「ああいう所にも同じ様に感じてらっしゃる方がいるんだな」と思いました。

「甲板を作って欲しい」と言うのは正にその通りですね。それ以前からの話だと実際にメインフレームを使われていない方々は「ファイル転送の機能があるよ」という様な話を聞かれて「ああそうか」と思われていると思います。

それから，シグマの方でも物があると言う耳寄りなお話があったんですが，それが実用的な物であれば素晴らしいですが，私が知っている範囲では先の電力計算センターの方がおっしゃられた様な物で，おそらく実現されているのかも知れません。

例えば，有名な物ではｋｅｒｍｉｔがあるわけですが，それもある程度触ってみると「こんな物が使える？」と言う実際の感想なんです。ファイル転送は出来ますが，「それで？」と言う話です。

ＡＳＣＩＩの深瀬さんの様に，当然，ＮＦＳで繋がっていてと言う様な環境があれば，それはそれでいいんですが，そんなものは無いので....，望んでも無い物はしょうがないんで....，そうすると「手元にある物でやりましょう」となります。そうすると通信回線経由でファイル転送する機能があって「じゃあこれを使いましょう」と言う話になります。

その時，私は使えるわけですよね。色々苦労して，うまく行かなくて，調整して「これだったらうまくいくよ」という線を出して，自分自身の日々の業務では使えるわけです。私はメインフレームを使っていませんが，自分自身の日々の業務では使えるわけです。じゃあ隣の奴にやらせる事が出来るか....，後ろに座っている事務の女の子に「これをやっておいてね」と任せられるかどうか....，全然自分とコミュニケーションが無い人間に簡単なマニュアルを書いて使わせられるかどうか....，と言うと全然別の話になります。つまり，そこの所が非常にクリティカルで使いにくいわけです。

だから「確かにそういう道はあるじゃないか」という事なんですが，先のＵＮＩＳＹＳの方の発言でも「準備しています」という事なんですが，もっとまともな物をちゃんとやって欲しくて，もっと安全に確実に出来るものをどうしてやってもらえないのという気がしています。

たぶん，お話聞かれている中には「それで出来るんだったらそれでやればいいじゃないの」と思われている方がいると思いますが「実際にはそうじゃ無いんだよ」というのが，私が今までの話を聞いて思っている事です。

久保： 先に坂下さんから聞いた事と似た様な話ですが，井川さんがいい話をされていて，分かって頂きたいと思います。何となく気分としてはやめたい気分ですね。

今のいい話でやめておいた方が，ひとつの余韻を残していいんじゃないでしょうか。済みませんが，私の権限で１０分ほど遅延しましたが，これで終わりたいと思います。

では，最後に４人の方に拍手を送って終わらせて頂きたいと思います。どうもご協力ありがとうございました。

3.セッション・サマリー

久保宏志
富士通システム本部
junet: kubo@kubowb.se.fujitsu.junet
NIFTY-Serve: PDG03202

＜なぜMML セションか＞

過去の開発環境ワークショップで「ワークステーションと分散環境」をテーマにしたセションを持ったことがある。ところが，このテーマだと2/3 くらいの人が議論にはいれない。特にメインフレームを仕事場にしている人には話が分からない。一歩引き下がって議論のテーマをMML(Micro Mainframe Link) にすると，メインフレームの世界に生息している人も議論に参加できるであろう。こんな気配りを働かせて，このセションをつくってみた。

これは中野先生が暴露した内幕である。セション・チェアマンである筆者には予め知らされていなかった内幕である。

＜MML セションの議事要約＞

MML セションの議論は，チェアマンが理解するところ，つぎのように要約できる。

メインフレームを神聖にして不可侵なるものとし，マイクロの方をそれに寄せる形で MML 型の開発環境を実現

する試みが紹介された。それは苦労の多い営みであった。その苦労も報われる苦労であればよいのだが，実はそうではなかった。

その一方に，別にそう大した苦労もなく，UNIXマシンを沢山つないだネットワーク環境を拵えて，そこで快適に生活している現場がある。その例が紹介された。そんなことが可能なのは，すべてのマシンをUNIXに揃えているからであろうと思うかもしれない。ところが実はそうではない。NFS をすべてのマシンに乗せればできることである。たとえば，バーチャル・マシンのゲストOSとして，メインフレームのOSとUNIXを乗せる。そのUNIXの下でNFS を動かす。二つのバーチャルマシンの間でファイルを共用させる。このようなアプローチを採用すると，UNIXマシンだけからなるホモジーニァスなネットワーク環境ではなく，様々のマシンを含んだヘテロジーニァスなネットワーク環境が苦もなくできる。

MML という言葉にこだわると，間違ったアプローチに誘い込まれかねない。そんな危険な言葉使いはやめて，あっさり分散環境とかネットワーク環境とかと呼ぶことにするのがよさそうである。

MML なる言葉からの解放が，産業界のコンセンサスになれば苦労はない。ところが，今のところコンセンサスはセション会場の中のものでしかない。このコンセンサスを，セションの会場から野外に持ち出したい。それを可能にするには，ネットワーク環境礼讃がひとりよがりであってはならない。

やればやれるという言い方に嘘はないが，常にそれが可能な訳ではない。ある前提が成立していればできる。何もかもUNIXにしろとは言わない。しかし，UNIXコミュニティが育てたネットワーク環境構築の仕組みを，他のOSも素直に優れたものと認め，その仕組みを取り入れてもらう必要はある。

今すぐすべてのことが可能な訳ではない。MML が支配的であるのには理由がある。そこのところをわきまえないと折角のコンセンサスが稔りを生み出す力にならない。

やればやれるのだからやればいい，といって済ます傾向は、我々の欠陥である。これでは無責任すぎる。つくった分散環境なりネットワーク環境なりは，普通の人が使うのに壁が高かったのでは困る。十分に壁を低くして，使い心地のよいものにする配慮を欠いてはならない。

ネットワーク環境構築はスマートにやれるとはいっても，それを実践するのは高度に知的で専門的な仕事である。すべての環境生活者に，このような構築技術を要求するのではなく，生活者に代わって環境構築を引き受けるシステム・インテグレータが出てきてくれる必要がある。

これらの課題は困難かもしれない。しかし，解決するに値する課題である。解決した暁にはソフトウェア開発者にも次のような世界が約束されている。
アステックの坂下さん曰く:

1:00AMに大阪にあるデータを使って，東京のCrayで計算し，その結果を名古屋と札幌の事務所のワークステーションに表示して，リアルタイムでデータを修正する。なんてしたくありませんか?

このような世界は，他の業界たとえば物流業界の業務用システムでは既に実現され実用されている。ただしそれは，多大な時間と労力を投入して実現したものである。社会的弱者であるソフトウェア開発者が大きな投資を要求しても許されることは期待できない。ソフトウェア開発者が必要とする環境は，一般の業務が必要とする環境に比べるととはるかに複雑であるという事情もある。ソフトウェア開発者は，金によってではなく技術によって，血と汗によってではなく知恵によって，自分の望む世界を手にいれることを考えなければならない。

＜セション・サマリー・レポートの組立＞

上に要約した議事内容を，以下の話の運びで若干詳しく述べる。最後の行動提案は，議事の外である。セション・チェアマン兼セション・レポータとしての筆者のまとめである。

- MML実現の試み 3件
- 別の接近
- 現実と可能性のギャップ
- ネットワーク環境の実現に向けて
- 議論の到達点
- つぎの行動の提案

＜MML実現の試み：事例1＞

JIP(日本電子計算)の例である。井川さんが紹介した。

JIP は，バロースと日立のメインフレームでずっと計算サービスの商売をしてきた。それは商売用だから開発者は遠慮しながら使わなければならない。何とかならないかとメーカに聞くと開発用マシンを入れなさいと言う。そんなことができるなら苦労はない。商売用のマシンは売上に寄与するから買いやすいが，すぐには売上に寄与

しない開発用マシンを買うとなると簡単にはいかない。

メーカは，メーカ製のMML を使えともいう。しかしそれを使おうとすると，仕事のやりかたを変えなければならない。長い年月の間に定着している社内規則を変えろと言われても急にはできない。

メインフレームを使った商売とは別にJIP は，比較的最近のことであるがNEWSのディーラを始めた。そのことも手伝って，　計算サービスの商売用とは別に社内にNEWSのネットワークができた。計算サービスの商売をやっている人は，NEWSネットワークのユーザからすると考えられないような環境で仕事をしている。商売をやっている人もそのことに気がついたのであろう。NEWSを使ってメインフレームの負荷を分散したいと言い出した。

といった経緯があって，NEWSのネットワークをバローズと日立のメインフレームにつなぐことにした。

ところがである。NEWSにあるTCP/IPがメインフレームにはないし，メインフレームにある同期通信がNEWSにはないとくる。

幸い，無手順接続はメインフレームでもできるようになっている。それで行くことにした。色々限界はある。回線の多重使用はできない。回線数のNEWSしかメインフレームにアクセスできない。回線速度は高々1200bps でしかない。こんな限界はあってもNEWS をタイプライタ端末の代わりに使うことは，これでできる。

このレベルを超えてNEWSを画面端末としても使えるようにしようとするとこれがなかなか上手くいかない。非同期通信を通して使えるメーカ製の画面端末があるから，それに合わせたいと思うのだが，合わせるのに必要な情報が手にはいらない。

NEWSとメインフレームの間でファイル転送を実現すべく，3780RJE の実現を考えたが容易ではない。NEWSに同期通信がないからゼロからやらないといけない。

NEWSをメインフレームにつなぐことはできたが，到底MMＬと言える水準に達してはいない。MML と呼ぶことができる快適な開発環境は尋常なことでは実現できない。メインフレームの側でTCP/IPをやってくれると解決する。メインフレームの主流プロトコルと NEWSの間にゲートウェイを置いても解決する。NEWSの方から攻めて，MML を実現するのは容易ではない。

＜MMＬ実現の試み：事例２＞

これもJIP　の例である。関口さんが紹介した。日立のマシンでの話である。EAGLE を入れた。続いてSEWBを入れた。メーカ製のMML を導入した例である。

EAGLE　がメモリをくってどうにもならないくらい遅くなる。この問題を解決するためにSEWBを入れた。MML の始まりである。SEWSでSDF をつくる。それをメインフレームに送ってシステムフローにする。システムフローをワークステーションに持ってきて，そこで編集する。ここから先は，システムフローから仕様書とソースを生成し，単体テストをすることであるが，ワークステーションでやってもいいしメインフレームでやってもいい。これがMML の概要である。

メーカはいろいろと歌い文句を唱える。

しかし実際に使ってみると問題が多い。プログラム資源をワークステーションとメインフレームで二重に管理しなければならない。アップロードとダウンロードが手動である。データのコードがJIS とEBCDICで違う。ソースの互換性がない。PAD で入力するよりソースを手でいれた方が速い。画面遷移シミュレータの結果を別のツールに入力するが，これが人手入力である。

EAGLE は二年くらいで使えるようになった。バージョン4 は使えなかった。バージョン6 になって使えるようになった。SEWBはまだバージョン2 である。あと4 つバージョン番号が進んでバージョン6 になれば，使えるようになるであろう。

いろいろ問題はあるが，メーカ製のDB/DC があるからメーカに依存せざるを得ない。

＜MMＬ実現の試み：事例３＞

KCS(前の神戸コンピュータサービス)　の例である。平山さんが紹介した。　JStar　でつくったドキュメントから3780RJE フレーム用のソースコードを生成する仕組みを作ろうと試みた事例である。データ構造設計書，　プログラム構造設計書，モジュール設計書などをJStar の上でつくる。RJE を使ってそれをメインフレームに送り，そこから先はメインフレームでやる。メインフレームはＦＡＣＯＭである。こういうMMＬである。

やってみたものの，まともなレスポンスタイムを実現できず，デモをやる度に評判を下げていった。

この体験があってメインフレームとのリンクは一回だけにしたいと思った。リンクは会話モードではなく，ファイル転送を一回行うだけにしたいと思った。

それに，ドキュメント作成にしか使えないJStar を採用したことも間違いであった。　JStar はドキュメントの清

書を支援する機能を備えるが，設計支援の機能は備えない。アルゴリズムの設計を支援してはくれない。それにJStar は遅い。

最上流工程からの再利用ということでドキュメントを重視することにしてＪｓｔａｒを入れたが，残ったドキュメントの80% はゴミである。

さてどこにいくか。改めて，MML は開発環境の改善に役立つかを，職場の同僚にアンケートで聞いてみた。期待できないとする人の数が半数を超えた。

＜別の接近＞

アステックの坂下さんが，アステックでの体験を紹介した。

MML を離れて，ネットワーク環境で何ができるか。何ができているか。これを整理した。ワークショップ参加者だけが一人占めするにはもったいない，分かりやすく有益なチュートリアルであった。

(1) 資源の共有ができる。

例: ファイルを共有できる。

例: プリンタを共有できる。

例: テープドライブを共有できる。

例: 高級カラー・グラフィクス・ディスプレイを共有できる。

ソフト開発は手元のモノクロ・ディスプレイでやって，最終確認はリモートのカラー・ディスプレイに表示させてこれを行う。

例: CPU を共有できる。

ローカルマシンでアルゴリズムを検証したあと，crayで実行する。

(2) 機能の分散ができる。

例: ターミナル・サーバは一つの例である。

ターミナル・サーバという単機能の専用機を用意し，ターミナルやレーザプリンタはこれに接続する。ホストマシンには接続しない。こうやってターミナル用のコードをホストから追い出してしまう。

例: ファイル・サーバももう一つの例である。

例: X ウインドー端末も可能である。

X サーバとネットワーク・インタフェースだけを持っている端末である。

上の例は，つぎのように一般化できる。分散させたい機能にネットワーク・インタフェースを持たせる。ネットワークに機能単位の箱を増設することで機能追加あるいは機能のキャパシティ追加を行う。ホストマシンにモノをつける必要はない。

(3) マルチプロセスができる。

機能ごとにプロセスを分けて，プロセスが協調して仕事する。プロセス単位にデバッグすればよいからデバックも楽になる。ネットワークが新しいパラダイムを可能にしたと言える。サーバ・クライアント・モデルといういいパラダイムが生まれて，今幅を効かせている。

例: ウインドー・システム(X)

例: かな漢字変換(wnn)

例: データベースのフロントエンドがネットワークにあって，データベースの管理をやる。データベースにアクセスするプロセスは，それにリクエストを出す。

複数のプロセスがゴチャゴチャと協調するパラダイムが生まれると，一段とプロセス協調の可能性が拡がる。そういうものが求められている。

(4) パフォーマンスの改善ができる。

まだまだ人が介入してやらないとロードバランスしないが，追々解決していくものと期待できる。

以上のようなネットワーク環境でできることは，メインフレームがそこにあるとできないことではない。NFS やRCS があればできることであって，別にUNIXでかためる必要はない。MML という呼称は発想を固定する作用を持つ。メインフレームもメットワークを構成する一つの要素であると位置づけて構想することが肝要である。

JIP の事例 1 のように，メインフレームとワークステーションをつなぐところで苦労していたり，あるいはJIP の事例 2 やKCS の事例のように，つないでも使いにくいといったところでモタモタしていると，メインフレームの世界に生活している人は，進歩から取り残される。

MML にこだわってはならない。

＜現実と可能性のギャップ＞

何もかもメインフレームの上でやる必要はないとは言っても，現実はその形態が支配的である。なぜメインフレームなのか。この問題をはっきり捕らえておかないと，可能性を現実化することができない。現実化するための戦略を誤る。

メインフレームは，大量のデータのためにある。高速処理のためにある。これはその通りである。ソフトウェアが揃っているから幅を効かせている。これもその通りであろう。だからと言って何もかもメインフレームの上でやる必要はない。メインフレームにはソフトウェアが揃っていると言っても分野を開発ツールに限定すると，

必ずしもそうではない。UNIXやMS-DOSの方がより豊富に揃えているかもしれない。だから，メインフレームを別格扱いしてはいけない。すべてを適材適所で扱わないといけない。メインフレームはその特徴を活かして後方に退かせ，そのパワーとキャパシティによって，より快適なネットワーク環境づくりに貢献させるのが順当であろう。ネットワークの中の一つの要素としてメインフレームを活用するのが正しい。

にも係わらずメインフレーマ側からMML ということが言われる。メインフレーマへの反発も手伝ってMMLは，メインフレームの延命策として考えだされたものではないかとする見方がある。この見方にたつと，その作為を見破り，延命策から自由になる必要がある，となる。

しかしそう単純に割り切っていいものだろうか。MMLよりもネットワーク環境が優れているとしても，現実の話，ネットワーク環境が支配的になる情勢にはない。そこのところをきちんと理解しておく必要がある。

理論的に優れているものが，現実には劣位に甘んじている理由は，つぎのように説明できるかもしれない。

計算機システムの管理面を考えると，メインフレームの世界には分業体制ができている。だから使う方は楽である。管理が集中されていることは魅力である。ネットワーク環境になると，ワークステーションの管理を自分でやらないといけない。これが現状である。この負担に耐えられる人はそうはいない。

この面の優位性が拠り所になってMML が考えだされたとする見方ができそうである。実際DEC は，使えないパソコンを救うためにMML を考案した歴史をもっているようである。IBM のPCが売れに売れた。LOTUS の1-2-3 をみんなが使って，一人一人がフロッピーを持つようになった。そのことによって計算機の全体の系が分からなくなった。そこにMML が登場して管理の集中に道を開いてくれた。おかげで全体が分かるようになり，コントロールが効くようになった。そういう見方である。

DEC は，MS-DOSのファイルを仮想的にVAX にとれるようにした。会社の書式改定をフロッピーで配布するのは大変である。仮想ファイルを使えば， この問題は一挙に解決する。プリンタを別々に持つのはもったいない。VAX のものを使えばよい。こういうことをやってくれるという理由でVAX が売れた例も事実ある。使えないパソコンを救うためにMML が登場したとする説明は，そう見当違いではなさそうである。

いまでは，使えないパソコンを，VAX とのMML で救うのよりも優れた方式が生まれている。DEC 自身が「386WARE 」という製品を出している。これはPCを捨ててしまうが， PCのソフトウェア資産はいかす。ソフトウェア資産はそのまま，ネットワーク環境の中で生かし続けることを可能にする。一太郎を386WARE に乗せ，X サーバを乗せたターミナル・サーバにVT100 をつなぎ，そこから一太郎を使うようなことができる。

ネットワーク環境は，こういった優れたアーキテクチャの発明によって徐々に改善しつつある。しかしだからと言って，メインフレームの持つ管理の集中という特徴の魅力にはまだ勝てまい。いろいろ欠陥はあってもメインフレームは，すべての問題を解決してきた。だから安心できる。若干の不便を忍べばよい。対してネットワーク環境は，その気になれば簡単につくれるし，そのメリットを活かして使えば快適だから，大して苦労することなく， 安易に使うところに止まっている。使える範囲で使っている。その範囲を越えて使おうとすると不安である。そう抑えて間違いないのではないか。ここにネットワーク環境の一つの課題がある。

こういった技術論よりもっと大事なのは，姿勢の問題であろう。MMLにネットワーク環境を対置し，ネットワーク環境をかつぎ，MML を否定的に評価してみても，問題は解決しない。日本のソフトウェア技術者はメインフレームのために働いている。この現実がある。その現実から出発して， 彼らの環境をいかによくしていくか。この問題から逃げられないし，逃げてはならない。

<ネットワーク環境の実現に向けて>

ネットワーク環境を実現する上での課題は，議論の中でいくつか提起された。それを書き留めておこう。上述の通り，ネットワーク環境は一人一人の利用者にかかる管理負担が大きい。この問題を解決しなければならない。これが一つである。

この管理負担の問題とも絡むが，システム・インテグレーション・ビジネスの成長発展を促す必要がある。これが二つ目の課題である。ネットワーク環境は，ネットワーク全体を一つの計算機とみる。それは高度なシステムであり，また必然的にマルチベンダー・システムの形態をとる。これを構築し運用するのは決して易しいことではない。この構築と運用支援をその内容とする事業が生まれ，それが発展することなしには，ネットワーク環境の普及はありえないであろう。残念ながら現在まだ，そのような良質なサービスが市場で得られる情勢にはな

い。

一つ例がある。MS-DOSを主体とする職場にMicroVAXを入れた。PC98をターミナル・サーバにしてEthernetを経由してPC98をMicroVAXにつないだ。ところが誰もＭｉｃｒｏＶＡＸを使わなかった。ファイルのアップロード，ダウンロードが不便だから使わなかった。みんなMS-DOSのまま使いたがった。ターミナル・サーバを入れるのではなく，PC-NFSを入れていればMS-DOSのまま使えて皆MicroVAXが活用されたであろう。

こういう失敗を繰り返さないために，それを未然に防止して一般の計算機利用者をよりよい環境に導く事業体の勢力が社会の中に育つことが望まれるのである。

ネットワーク環境は，様々の標準化の一層の推進を必要とする。これが三つ目の課題である。

理想のネットワーク環境では，どのホストでコンパイルされるか分からない。どのホストで実行されるかも分からない。それは時と場合で変わる。利用者にマシンを意識させない。ＯＳを意識させない。マシンとマシンをつないでいるプロトコルも意識させない。利用者が意識するのはアプリケーションだけである。アプリケーション以外のものをトランスペアレントにしえて始めて，理想のネットワーク環境が実現する。

また理想のネットワーク環境では，どのマシンでプログラムを実行するか，どのマシンにデータを置くかもネットワーク自体が決めることになる。これは，プログラムとデータのポータビリティが実現されて初めて可能になる。

トランスペアレンシーとポータビリティは，様々の標準化を必要とする。その代表的なものにはつぎがある。ネットワーク・ファイルのプロトコルの標準化，データの標準化，そしてインタナショナリゼーション／ローカリゼーションの標準化である。

最後の標準化は，次のような意味合いのものである。アメリカに行った日本人は，アメリカにあるマシンを日本語で使えるようにローカライズして使う。一つのサイトには中国人もいるし，アラビア人もいる。日本の大学の研究室は既にそうなっている。そこでは，そこに住む人々はそれぞれの言葉ですべてのマシンが使う。そういうことを可能にする必要がある。

現状はトランスペアレンシーからは程遠い。パソコンで作ったソースをホストに渡すのに，一旦パソコンを殺して，端末として立ち上げるような操作を必要とするものさえある。リンクはシームレスでなければならない。

ポータビリティの現状も一筋縄ではいかない。開発環境が必要とする標準化が，ＵＮＩＸの標準化によって一歩前進するとしても，開発環境と実行環境を分けなければならない状態は続く。

解決すべき課題は上の三つにつきるものではない。メンテナンスの問題，ライセンスの問題も提起された。

ネットワーク環境になって，あっちこっちで同じソフトウェアを使うようになるとメンテナンスの問題が発生するかもしれない。PC-NFSを使えば，VAX のファイルをMS-DOS のA ドライブやB ドライブにできるし，一太郎をVAX に置いておき，VAX の上で使うことも，あるいはftp でVAX からPCに移してから使うこともできる。だからメンテナンスの問題は部分的には解決しているかもしれない。

しかしソフトウェアのライセンスの問題はまだ未解決である。特定のＣＰＵでしか使えないようなライセンスのままではネットワーク環境は困る。一台のVAX に一太郎を一個置いて，それを全部のPCが使ったりすると，ジャストシステムは黙っていまい。

<議論の到達点>

向かうべき目標は共有できた。MMLではなく，

mainframe-in-network

が目標である。

それが現状の技術でどこまで可能かも概略は理解できた。ネットワークにつながる全ての機器がNFS を乗せ，クライアント・サーバ・モデルをパラダイムにしてプロセスを協調させれば，かなりのことはできる。そうすることは易しくはないかもしれないが，できないことではない。現実に，このメリットを享受しているサイトは存在する。

その一方で，まだまだ未解決の問題が山積していることも確認できた。

<つぎの行動の提案>

以上の議論を踏まえて，現状と目標についてコンセンサスを形成するための行動が望まれる。この行動は，意見の共有に止まるのではなく，意見の現物化まで進めることが望まれる。現在既に技術的には実現可能な水準に，できるだけ多くの現場が，できるだけ早期に到達できるよう支援して，コンセンサスを現物化することが，この行動の目標である。

更に一歩進めるための行動も望まれる。目標に向かっ

て，効率的にかつ効果的に，共同して前進するためには，今一度解決すべき問題を整理し，解決戦略を練り，解決行動に優先順位を付与し，問題解決の実践状況のマップをつくり，その実践に選択的に参画していく行動も望まれる。

これらの行動を加速化するためには，メインフレーム擁護派からの痛烈なネットワーク環境批判が有効である。ネットワーク環境へのシフトは歴史的必然であり，メインフレームの時代にあともどりすることはありえない。だからネットワーク環境批判は，ためにする批判の性格をもつ。それを承知の批判が貴重である。その批判を乗り越えたネットワーク環境論にしないといけない。青二才の書生談義では社会は動かせない。

<筆者から読者へ>

当セションの討論が生産的であったことに，参加者の皆さんは同意するであろう。本サマリ並びに，これとは別に編集される詳細記録が，討論内容の共有を，セションに参加できなかった方々にまで拡大するのに役立つことを望む。

セション・サマリは討論内容の忠実な要約ではない。そのことはお断りしておく。必ずしも議論の全体を理解して書いた訳ではない。誤解も多数含まれているであろう。それらの誤解は，またの機会の討論を通して改めることとしたい。

正直言って筆者は，MML セションのチェアマンを引き受けることに乗り気ではなかった。気乗りしないままセションにのぞんだ。しかし議論が進むにつれておもしろくなっていった。今では，セション・チェアマンの役割を与えてくれた中野さんに感謝している。

4.1 MMLの発表に関する感想

海尻賢二（信州大）
MML＝負荷の軽減＋UIの改善
大学もセンターはMainFrame、末端は1200bpsで使っている。2年前まで情報工学科もそうであった。

林　香（SRA）
非常に重要なこと（多くの現場がエンジニアの仕事が楽になるという意味で）なのに、「たいしたことではない」というスタンスの話になっていたのが残念が、地に足のついた話なので重みがあり面白かった。

高橋哲也（神戸製鋼）
私の仕事内容としては、あまり実感のない話なので申し訳ありませんが無感動に聞いていました。

古閑幸一（HST）
現場レベルの苦悩がうかがえた。メインフレームは使ったことがない。

佐藤千明（長野県協同電算）
MMLの課題は多いが、これをものにしないとS／W開発環境の底上げにならない。

野見山和則（三井銀ソフトウェアサービス）
メインフレームメーカが少しでも早く、まともなEWSを提供してくれることを望む。

北野義明（KCS）
メインフレームの世界を理解してもらう場として意味があった。日経の吉本さんのお話がメインフレームの世界の根底にあり、しかも閉じた世界であることの息苦しさを知って欲しい。（オープンアーキテクチュアにすると国産メインフレームはIBMに利益のほとんどを吸い取られ、日本経済がこける）そこで“異機種をつなぐ”ことから、手続き型の第3世代言語での開発の支援までを地道に考えている。

上妻健一郎（電応研）
メインフレームの実体をかいまみた思いがした。

坂本　治（シグマ本部）
いろいろ参考になりました。

三浦あさ子（SRA）
MMLというのはMicroMainframeLinkという意味らしいけれども、実はMan（人）がいっしょうけんめいMainに働きかけようとしている（orなんとか逃げようとしている）作業のように感じられた。

藤岡　卓（三菱電機）
異なる世界の話が聞けておもしろかったです。

杉田義明（SRA）

単にハード的にリンクをつなげただけといった発表と議論が中心のように感じた。開発環境として論理リンクの構築をどのように考えるのがよいか、といったことの重要性が分かった。坂下さんの話はとてもわかりやすくて、面白かった。

熊谷　章（ＰＦＵ）
関口、井川：日立はあんまりだ。開いた口がふさがらない。せめて、ＥｔｈｅｒＮｅｔにつなぎ、ファイル転送ぐらいは最低サポートすべきだ。
平山：ソフトウェア開発の本質をついた発表だった。設計とかプログラミングとは何か、を本質的に考えている。ドキュメンテーションとＭａｉｎＦｒａｍｅは枠外だということがよく分かる。

渡邊　雄一（電力計算センター）
関口さん〔ＪＩＰ〕の発表はメインフレームを使っている一人として，同感の一言。かなり慎重な言い回しで，パソコンを融合しようとして失敗しているＭＭＬの現状を的確に述べていたと思う。
平山さん〔ＫＣＳ〕の発表は半分くらいうなずけるけど，ＲＪＥで３０分も転送がかかるというのは解せない。ＪＣＬや簡単なソースコードくらいなら４８００ｂｐｓでもせいぜい５分でしょう。ひょとしてＫＣＳのセンターの運用に問題あり！なのかも？。（ぜひ１度私にチューニングさせて！）それより，メインフレームでもＮＦＳとは行かないがＳＮＡでメインフレーム同士は接続されていることを考えると，いつまでもＲＪＥを「つながった」と表現することが適当なのかどうか，もう１度考え直すときが来ていると思う。そのような意味でも坂下さん〔アステック〕の発表は脱帽！

山口　郁生(電力計算センター)
発表が、ＭＭＬの悪い面ばかりあげていたように思う。「メインフレーム叩き」のネタ？それとも、メインフレーマーの参加の場？

4.2 ＭＭＬの討論に関する感想

林　香（ＳＲＡ）
なぜＭＭＬが必要なのか？何かあるようなのに、ＴＳＳが重くなければ、その方が良いと言うのではかなしい。現場のエンジニアの労働条件の改善になるはずなのでより工夫して欲しい。

古閑幸一（ＨＳＴ）
資源共有のためのメインフレーム

野見山和則（三井銀ソフトウェアサービス）
臼井さんの意見は面白いと思います。メインフレームメーカは優れたＭＭＩをもったＥＷＳに多くのＣＡＳＥツールをのせて売っていかないと、臼井さんの話が真実になるのではないですか。

上妻健一郎（電応研）
坂下さんの話はマルチベンダー環境のユーザとして貴重な意見でした。

北野義明（ＫＣＳ）
事例を自分自身が知らなさすぎることを感じた。・・・が、おおよそ予想通りの現状でした。

臼井義美（ＪＩＰ）
メインフレームで処理しなければならない業務というのは今後限られたものになろう。そのためには大したＭＭＬ機能は不要で、メインフレームをターゲットマシンにした開発環境をワークステーションで実現するのはナンセンス。メインフレームは戦艦ではなく、航空母艦の役割に徹すべきであると思う。

三浦あさ子（ＳＲＡ）
この問題に関して、いろいろな人がいろいろな方面から（対応）たちむかおうとしている感想です。逃げていてはいけないんですね。

鐘　友良（ＦＸＩＳ）
本質的に資源（ソフト、ハード）の共有と言うことから考えれば（ＭＭＬは）すべきじゃないかなと思います。

藤岡　卓（三菱電機）
おもしろいセッションでした。

海尻賢二（信州大）
理想：メインフレームもＷＳもない。意識しなければならないことが間違い。ＮＣＳ、ＮＦＳをｍｉｘしてユーザの意識は全体としてのシステムを使うということでよ

い。

杉田義明（ＳＲＡ）
実行環境と開発環境の中で、メインフレームをネットワークの単なるサーバとして位置づけることに合意できたように思う。次にはその環境の具体的なイメージ作りが大事だ。

黒坂靖子（ＪＩＰ）
ＭａｉｎＦｒａｍｅ側への具体的な（技術的な）アプローチが全くなく役に立たない。

熊谷　章（ＰＦＵ）
坂下：ＭＭＬセッションをネットワークコンピューティングに持っていき内容を充実させた功績は大きい。面白かった。討論：内容がいま一つ。テーマを少し絞った方が良かった。

野中　哲（ＮＥＣ）
“ＭＭＬの原点はＭＳ－ＤＯＳであった”という話は興味深かった。

渡邊　雄一(電力計算センター)
議論の土台が発言者により大きくズレていたため，良い意味での議論にならなかったと思う。ＭＭＬで何をやるのか，やれるのかという話ならば，メーカの人間に聴けば（ある意味で）必要十分なはずである。しかし，開発環境としてＭＭＬを「どうしていかねばならなのか？」，「そのためには何をしたい！」，「しかし現実には何々がネックとなっている」といった項目がきちんとプレゼンテーションされる必要があったし，それに基づいて意見交換がなされるべきであった。

山口　郁生(電力計算センター)
メインフレームをファイルサーバにしたＷＳネットワークが便利ならば、それを紹介してほしい。だいたい、ＦＡＣＯＭにしても、ＨＩＴＡＣにしても、完全なファイルサーバには、なっていないと思う。そりゃ、ＥＰＯＣシリーズやＯＦＩＳシリーズを使え、というのなら別だが．．．

5. おまけ

渡邊さんから中野さんへのＥｍａｉｌ

中野せんせいへ

電力計算センターの渡邊です。環境ワークショップの原稿を見ながら酒を飲んでいて「もうＭＭＬの話をしても面白くないんじゃないか」「いや来年もう一度」という話が非常に気になりましたので，酔っぱらったついでに戯言（たわごと）を書き連ねたいと思います。酔っぱらいの書いた文章なので適当にあしらって下さい。

私が思うに，@神戸での論点だけから言えば，そして開発環境ということに限って言えば，特に使い勝手の面やより上流工程をサポートするためにはＵｎｉｘなど（上流工程の支援にはＵｎｉｘの上のツールでも悪くはないかもしれないが，私としてはリスプマシン上のアプリケーションに期待している）が良いと思うので，やはりメインフレームの勝ち目はなく，とても議論にならないような気がするのです。

ただメインフレームをお守りしている者からみて，ただ１つ〔これはちょっと言い過ぎかなぁ〕議論が噛み合いそうなところは「開発環境とて運用していかなければならない」ということがあるんじゃないかと思うんですが・・・。例えば，Ｕｎｉｘはスケジューリング出来ない（しにくい）とか機密保護〔無くても良いヨという意見は多々あるが，開発していること自体がマル秘扱いの時はどうするの→別のマシンで開発すれば良いというのは乱暴ではないかなぁ〕といった話があると思うのです。「メインフレームでの解決方法は決して良くはないが，じゃぁ，Ｕｎｉｘだってそこは・・・」といった議論が出来るのではないでしょうか？

それと（ＢＯＦのセッションではなかったですが）ＪＩＰの井川さんほかが話してくれた，ＭＭＬの発表は，普段から多くのメインフレーマーが感じていることを実に簡潔に述べて貰えたと思います。特に，メインフレームをターゲットとして開発に，ソフトウェア環境として等価でないＰＣ，ＥＷＳなどでコードを生成することは，現段階では非常に疑問があります。〔中途半端にお金をかけるてシステムを作ったところで所詮あのΣのようなになってしまう！〕

また最近特に感じていることですが，ターゲットマシンでのソフトウェアの癖みたいなものを知らないと，本当に良いシステムの設計やプログラミングは出来ないように思います。そのような面からも『ＰＣやＷＳでの分散開発環境については，ターゲットと同等なソフトウェ

ア環境をシミュレート出来る必要がある』と言えると思うのです。（だから私はＬＵＮＡ位の大きさのマシンにＩＢＭコンパチのＯＳを乗せて，システムのお守りをもっと田舎ののんびりしたところで静かにやりたい：ー）） それが出来なければ，かなりのノウハウを知識ベースか何かの形で実現しそれを参照しながら動くツールでなければ，決して良いコードは得られないでしょう。そんなことは当分望めないでしょうから・・・。〔今多くの大学でＵｎｉｘやＭＳ／ＤＯＳばかり教育しているのは何か怖い気がします〕

「何の為にメインフレームを使うのか？」 このことをはっきりさせないで，ＭＭＬの議論をするのは非常につまらないと思います。事務系のアプリケーションを組んでいる（例えば多分ＫＣＳやＪＩＰの現場の人）にとっては，大型機上のオンラインデータベースの周りの各種プログラミングが想定されているのでしょうが，Ｕｎｉｘを使って仕事をしている人たちにはそのような具体的な利用状況すら思い浮かばないのではないでしょうか？

そういう意味からも，ここらへんの前提となる問題を事前に整理しておかないと（もし九州は熊本当たりでやるのなら）あまりにも虚しいと思うのですが・・・。せめてワークショップなのですから『現状のＭＭＬの問題点と実践的問題解決提案書』なんてものが成果として出来れば，（日本国内でさほど不便な所でなければ）幾らでも会社は出張させてくれるのですが・・・

渡邊 雄一
㈱電力計算センター 大手町事業部
watanabe@denken.junet

メインフレームマシンの作業環境とUNIX ネットワーク

日本電子計算(株) 大阪支店

井川裕基

メインフレームマシンの周辺では、どんどん高性能化する端末 (!?) を生かして分散開発環境を形成しようという話題がよく取り上げられる。一方、第4世代言語やわかりやすいユーザインタフェース等を使ってソフトウェア開発をより論理的に、構造的にしようという動きも盛んである。この両者を連結すれば、メインフレームのソフトウェア開発に有効な環境となるであろう。しかし、実際にはメインフレーム側の (とくにネットワーク周りの) 制限が多いため簡単にはいかないようである。当社でも UNIXワークステーション/ネットワークの普及にともない、「メインフレームの業務にこれらの使いやすいマシンを利用できないか」という意見が、特に現場でよく聞かれるようになった。そこで、まずは UNIXの環境からメインフレーム(彼らにとってはホストマシン)にアクセスする方法を考え、それを足掛りとしてより効率的な作業環境を構築[redacted]する作業を始めることにした。
ここではその作業の経緯と、まだ初期の段階ではあるが、現状について説明する。

1. メインフレームユーザの作業環境

現在、JIP では B7900(ユニシス)、S-810(日立) を中心としたいくつかのメインフレームマシンが業務用に運用され、COBOL / FORTRAN / ALGOL (ユニシスのみ) によるプログラムのコーディング/デバッグ/実行の全フェーズが同一のマシン上で混在している。

開発環境としては、メーカ提供のツール (LINK、SEWB)が一部使われている程度で、開発環境全体をフォローするようなツールは使われていない。

1.1. ホストと端末

環境は大きく分けてハードウェアとソフトウェアである。メインフレームユーザにとって直接関係のあるハードウェアといえば端末、ソフトウェアといえばユーティリティであろう。

1.1.1. ハードウェア

使用されている端末機は、ホストマシンと同一メーカ製の画面端末が主体であり、それ以外はダイアルアップの低速無手順端末である。

画面端末は、ホストマシンに接続されたマルチドロップの回線によるものがほとんどで、4800〜19200BPS 程度の通信回線に数台〜十数台の端末機が接続されている。
これらは大型機の使い方としては標準的なものであると思われるが、エディタからデータベースまでが同時に動いている状態のマシンでは、とてもレスポンスが良いとはいい難い。(私から見て)

1.1.2. ソフトウェア

標準のコマンドによるユーザインタフェースとエディタが環境のほとんどである。一部、メーカ提供または社内作成のユーティリティが使用されている。

開発者は、シートにコーディングし、パンチアップを待ってコンパイルし、結果をリストに打ち出したうえで、端末に余裕があれはオンラインで、そうでなければ机上でデバッグを行なうという、古典的な利用環境で作業している。

1.1.3. MML へのアプローチ

最近のはやりとして、「エディット」「デバッグ」「実行」などの各フェーズ毎に適当なマシン環境を用意して負荷を分散す

るのがよいといわれている。
一方、ワークステーションと呼ばれるクラスの機器が普及し、いわゆる「端末機」にもワークステーション (とメーカは呼ぶ) ベースのものが増えたため、ホストマシンにつながっていない状態ででもかなりの機能 (といっても、実用的なのはエディタぐらいのものであるが) を持てるようになった。

そこで、前述の負荷分散を端末機 (自称ワークステーション) とホストの間に実現しようとするのは自然な成行きであると言える。

メインフレームメーカは、第4世代言語をベースとした環境や、上流工程からの要求仕様定義ツールを提供しているが、これらは設計の当初からそれらの環境を使ったものでないとうまくサポートできないこと、現実の開発環境とは微妙にずれていることなどから、有効に使うことができずにいるようである。

日立では SEWB という環境を商品化している。これについては、経験者が説明を行なう。

1.2. ネットワーク

JIP で採用しているメーカは、いずれも現時点では TCP/IP などの汎用のプロトコルを使った高速のネットワークをサポートしておらず、また現実的なサポート予定もない。

トークンリングをサポートしているはずであるが、自社の端末機器以外の接続については発表や保証のできるような実績も無いようである。

2. JIP における UNIX 環境とホストの関係

社内に NEWS と Ethernet を主な構成要素とした UNIX ベースのネットワーク環境が広がるにつれ、そのローカルなあくまで、メインフレームをホストとした視点からの言いかたであるが) 機能の豊富さに着目し、これをメインフレームのための開発業務に使えないか、と思うユーザが増えてきた。

そこで、手元の UNIX ワークステーションからネットワークを通じてメインフレームにアクセスする方法を確立しホストに集中している作業を分散することができるかどうかの調査と基礎実験を行なっている。

そういった目的を満たすためには、すくなくともワークステーションとメインフレームの間で仮想端末/ファイル転送程度の機能が必要である。
MT を使ってメインフレームと UNIX マシンの間でデータを交換することも可能であるが、やはり日々の環境として使うには手元のファイルを任意の時にメインフレームに送れなければ現実的ではない。
そのため、Ethernet の世界からメインフレームホストをアクセスする方法について調査をはじめたが、現段階では前述のようにメインフレーム側に汎用のネットワークの受け口がないためストレートには接続できない。
そこでハード/ソフトを工夫して、見かけ上 UNIX のネットワークから直接ホスト機をアクセスする方法を模索した。

なお、今回のアプローチはすべて UNIX 側からのものであり、メインフレーム側には特別な設定などは行なっていない。

2.1. ゴール

大型機と手元のワークステーションの間での NFS などというのは、夢はであるがユーザレベルで出来るようなものではないので、まずは一般の MML の様に手元の UNIXマシンでエディットと簡単なテストを行ない、ある程度以上の作業はホストマシン側で行なうような環境を考える。

FORTRAN、COBOL については既に NEWS などのワークステーション上でコンパイラが動くので、簡単なシンタクスチェックはここで行いたい。

また、emacs の様な構造を意識できるエディタを使えば入力についてはある程度の効率化/標準化が可能と思われるが、emacs の普及状況を考えて、今回は考察しない。

2.2. ハードウェア

現時点で JIP の大型機にアクセスするためには通信回線を経由するしか方法がないので、まずネットワーク上の機器(実際にはNEWS) とホスト機(実際にはポートセレクタ) を RS-232C で直結した。(図 1)

ホスト機で高速な通信を行なおうとした場合は同期通信方式が一般的であるが、NEWS の RS-232C ポートは現時点では同期通信をサポートしていないので無手順での接続とした。
但し、後に述べる RJE 接続では原則として同期通信が必須となる。このため市販の低価格の「同期-非同期コンバータ」を接続してみたが、これはうまく制御できなかった。

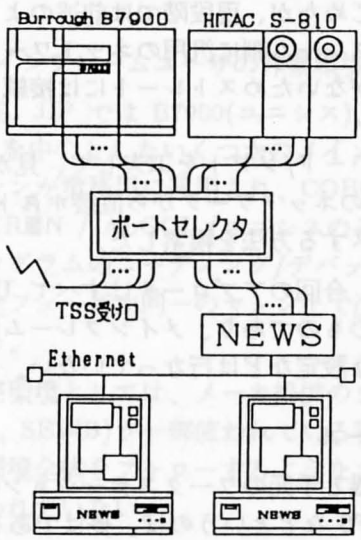


図 1. 接続の様子

2.3. ソフトウェア

ハードの方が上記のような状況であるので、いきおい各種の機能はソフトウェアの方にシワ寄せられてくるのであるが、何分、メインフレームの環境に UNIX の方から合わせてやらなければいけないわけだから、既存のツールもほとんで使えない。

とりあえず許された環境を前提に、ネットワークからメインフレームに通信する方法を考えた。

2.3.1. 仮想端末

メインフレームと非同期の回線を通してできる一番簡単な通信は「無手順端末」である。そこで最初に、ネットワーク上から「仮想端末」を使ってメインフレームをアクセスする方法を考えた。

UNIX 上の簡単なターミナルエミュレータソフトをベースにして RS-232C 経由でホストに接続する機構を作り、ネットワークからこれを使えるようにデーモンとこれを呼び出すプログラムを作った。

これにより、Ethernet 上のどこからでも「無手順端末」としてホストマシンを呼び出して使うことができる。
ただし、回線を多重に使うことができないので実際に引かれている通信ケーブルの本数 (現在 7本) までしか同時に使えない、ホストの受け口の速度が1200BPSまでである、などという制限がある。

2.3.2. 画面端末

非同期回線を通して使えるような画面端末が存在する (日立の ASPEN 等)ので、その画面制御などのシーケンスがわかればネットワーク経由の画面端末を作ることができる。現在プロトタイプが動いているが、制御シーケンスに関する充分な情報がないために機能が少ない以上に、前述の1200BPS という速度制限がネックになって評価できずにいる。

2.3.3. ファイル転送

ファイル転送のように大量のデータを一括してやりとりするためには RJE が使えるのではないかと考えた。3780RJE がもっとも一般的な手順であるようなので、それをNEWS 上に実現することを考えている。

NEWS では、前述のように同期通信をサポートしていない上、BSD 系の UNIX にはベーシック手順をサポートする機能もないのでこれらはゼロから作らなくてはならない。これは現状ではまだ設計段階である。

2.4. 実現状況

まだファイル転送を実現していないので、単に低機能の端末を増やしたに過ぎない。

しかし、手前味噌を恥じずに言えば、どんどん大げさになっていくホスト機端末を、待ちキューをつくってまで使うよりは遥かに簡便であろうと思う。
実際、いろいろな機能がゴテゴテとくっついている画面端末が、作業のどんな場面にでも快適な環境を提供するとは限らないのではないだろうか?

言語レベルに関わる部分については、まだほとんどなにも手をつけていない。現在、F77、COBOL は NEWS 上で動くコンパイラがあるので、実際に開発しているメンバーにこれらをさわってもらい、ホスト機との差などをつめていき、ガイドラインのようなものを出すことができれば、思っている。

2.5. 今後の目標

もちろん当面は予定している機能の実現が目標である。

今回は、TSS/RJE といった、ホストの持つ最低限の通信の機能を利用しているが、これでは実際の通信の効率はよくない。これを改善するには、メインフレーム側に汎用のネットワークインタフェースが用意されるか、既にメインフレームが持っている比較的高速な通信手段に対してネットワークからのゲートウェイ機能を実現することである。
すでにある種のメインフレームマシンではチャネルに接続できる Ethernet インタフェースが用意されている。我々の使っているホストマシンも、いずれはこれに習うのではないか、との見方もある。

また、ネットワークを運営する側としては、これから実現される機能をどのような形でユーザに解放するか、も考えていかなくてはならない事である。できるだけユーザがネットワークを意識せず自然な形で使える様に考えていきたい。

現在予定しているものは、MML　といってもほとんどハードウェアに近いところである。実際に開発作業をより効率のよいものにするためには、やはり「設計」「プログラミング」「テスト」「デバッグ」といった作業を有機的に結び付けてトータルにサポートする必要がある。この部分はむしろ「CASE」の話題に近いと思われるので、そちらでの議論を参考にできれば、と思っている。

ＥＡＧＬＥ２

ソフトウェア一貫生産システム

Ｈｉｔａｃｈｉ　Ｍシリーズ

ＯＳ　ＶＯＳ３

ＳＥＷＢ

ソフトウェアエンジニアリングワークベンチ

Ｈｉｔａｃｈｉ　２０５０シリーズ

ＯＳ　ＨＩ－ＵＸ(Unix SYSTEM V R2)

(独自ウィンドウシステム)

機　能

ＥＡＧＬＥ２

対話によるシステム開発

ソフトウェア資源の一元管理

部品有効利用システム

ドキュメントの自動生成

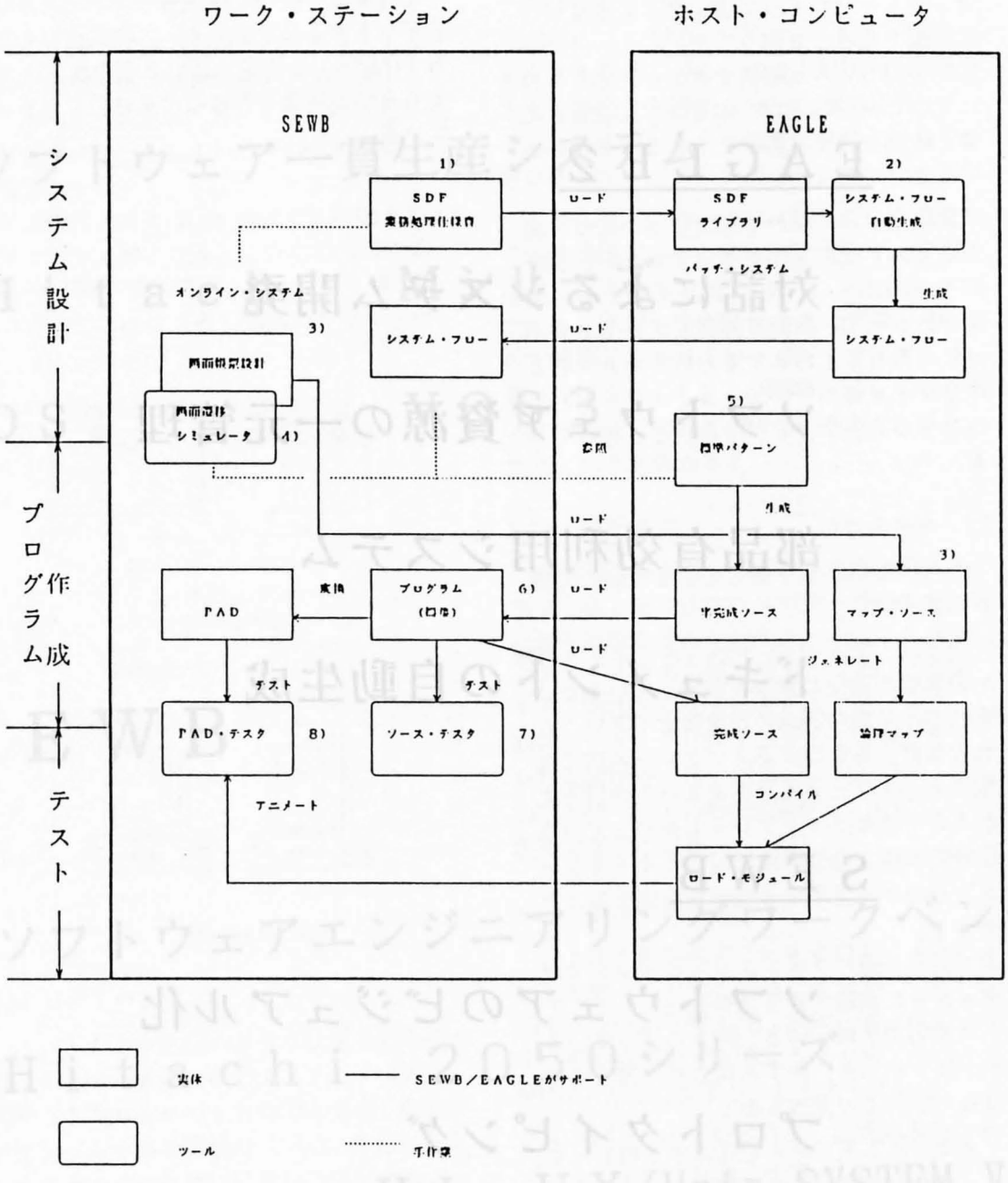
ＳＥＷＢ

ソフトウェアのビジュアル化

プロトタイピング

機能分散

図－２　ＳＥＷＢ／ＥＡＧＬＥ環境でのソフトウェア開発手順



第4回 SEA環境ワークショップ

「開発環境は誰のため？ MMLは誰のため？」

平山 伸一

hirayam@kcs.junet

(株)ケーシーエス

はじめに

近年のパーソナル･コンピュータやワークステーションの発展に伴い、従来からの伝統的なホスト集中型のアプリケーション･システムから、ホスト･マシンとパーソナル･コンピュータやワーク･ワークステーションの互いの長所を利用したものへとシステム化要求は変化しつつある。しかし、ことソフトウェア開発環境ということになると、4GLやRDBMS等が一部、気をはいているに過ぎず、寂しい限りである。さらに、4GL等についても、エンドユーザ指向が売り物となっており、実質的なソフトウェア開発者を支援するには至っていないのかもしれない。また、もしかするとソフトウェア開発者自身は、MMLには何も期待していないのかも知れない。自社で行った「MMLを利用した開発支援ツール構築の事例」と、「開発環境」に関する簡単なアンケートの結果をもとに、現状の問題と我々が進むべき道について考える。「MMLは誰のため?」そして「ソフトウェア開発環境は誰のため?」

1.これまでの取り組み

今から約3年半前、私の所属する部門には幾つかのパーソナル･コンピュータを除いて、開発環境と呼べるものはほとんどなかった。それからの3年間はJ Star IIを中心としたソフトウェア開発ドキュメンテーション環境の充実を図ってきた。ソフトウェア開発ドキュメンテーションが「紙とエンピツ」から「ワークステーション」に移行するにつれて、当然のことながら、そのワークステーションで書かれたドキュメントをインプリメンテーションに直接結びつけたいという要求が発生した。

この要求をうけ2年半前、約4か月をかけて、簡単な開発支援ツールの試作を試みた。このツールでは、J STAR II上で書かれた、(1)「レコード･レイアウト定義書」、(2)「モジュール概要設計書」、(3)「モジュール構造図」および、(4)「モジュール詳細設計書」の4つのドキュメントから、RJEを使ってM-380にファイル転送を行い、簡単なシンタックス･チェックを行いながら、コピーライブラリ、データ項目名辞書、および、COBOLプログラム･ソースを作成する。ソフトウェア開発工程のより高い所から標準化や再利用を行えば、効果も上がるであろうというのが私の考えであった。しかし、何度かのデモンストレーションの後にこのツールは消えていった。

2.そこから得た教訓

このツールが消えていく運命にあることは開発途中にもうすうすは感じていた。理由としては、(1)J STAR IIのレスポンスの悪さ、(2)RJEジョブのプライオリティの低さとホスト･マシンの

レスポンスの悪さ、(3)支援する対象が不明確、などがあげられる。このツールの失敗をとおして得られた教訓は以下のとおりである。

(1)開発支援用ワーク･ベンチの選定

ドキュメンテーション支援は開発支援とイコールではない。また、目的とするソフトウェアのないコンピュータはただの箱である。このことから私は、ワークステーションやパーソナル･コンピュータによる開発支援を考える現在、JSTAR IIを捨てるべき時期にきていると考えている。

(2)ホスト･マシンのレスポンスを考慮して...

単体テストぐらいまでのデバッグはすべてワークステーションやパーソナル･コンピュータ上で行えるようなツールが必要。(多くのジョブが走る)ホスト･マシンのレスポンスを考えると、開発におけるワークステーションやパーソナル･コンピュータからホスト･マシンへの橋渡しは一度きりで行いたい。従って、MML結合のレベルはファイル転送で十分。

(3)支援する対象は組織構成人員のなかでのモード

以下に、私の所属する部門内で行った調査の結果から、担当職務別人員構成表を示す。(サンプル数95)

主たる担当職務	%
営業	3%
プロジェクトマネージメント	19%
要求分析、システム分析	3%
システム設計	27%
プログラム設計	39%
その他	9%
合計	100%

表2-1 担当職務別人員構成表

この表から分かるように、システム設計とプログラム設計で66%を占めている。対象としてはここを支援するべきであり、さらに狭めるならば、39%を占めるプログラム設計支援が最重点課題と考えられる。前出のツールでは使用者はプログラム設計者であるものの、プログラム設計は何一つ支援していない。従って、使用者を無視したお仕着せのツール、環境になってしまった。

3.今後の展開を考える前に

JSTARIIによるソフトウェア開発ドキュメンテーション環境の展開についてはほぼ終結した現在、今後のソフトウェア開発環境の改善へ向けての新たなるスタートを迫られているが、そもそも、部門内においてソフトウェア開発環境はどの程度求められているのか、MMLに開発環境としての期待はあるのか?について把握したく、簡単な調査を行った。

3-1.開発環境はソフトウェア開発における問題としてどの程度意識されているか?

「主たる担当職務別」に、「ソフトウェア開発において一番問題を感じているもの」(単一選択)を表3-1「ソフトウェア開発における問題点」にあげる。

横軸:問題 縦軸:主な職務	プロジェクトマネージメント	テクニカルマネージメント	開発環境	見積り	その他
営業	33.4%	0%	0%	33.3%	33.3%
プロジェクトマネージメント	44.4%	22.2%	5.6%	16.7%	11.1%
要求分析、システム分析	0%	33.3%	33.3%	33.3%	0%
システム設計	23.1%	11.5%	34.6%	30.8%	0%
プログラム設計	33.4%	16.7%	33.3%	16.6%	0%
その他	22.3%	22.2%	11.1%	22.2%	22.2%
合計	30.5%	16.8%	25.3%	22.1%	5.3%

表3-1「ソフトウェア開発における問題点」

この表を見る限り、「開発環境」に対する問題意識はかなり高いように思える。しかし、ソフトウェア開発をもっともよく理解していると思われる「プロジェクトマネージメントを主な職種としている人たち」の「開発環境」に対する問題意識は以外に低い。「実際的なソフト

ウェア開発環境」を展開、構築していく上で彼らはキー･マンであるはずである。さらに詳細な調査が必要である。またこの点について他社での状況はどうなのであろうか?

3-2 開発環境改善にMMLは期待されているか?

「MMLによって開発環境は改善されると思うか?」について、主たる担当職務別に表3-2「MMLによって開発環境は改善されるか?」に示す。

主たる担当職務	される	されない
営業	66.7%	33.3%
プロジェクト･マネージメント	83.3%	16.7%
要求分析、システム分析	66.7%	33.3%
システム設計	50.0%	50.0%
プログラム設計	44.4%	55.6%
その他	88.9%	11.1%
合計	58.9%	41.1%

表3-2「MMLによって開発環境は改善されるか?」

この結果は、先の「開発環境に対する問題意識」と対象的なものになっている。すなわち、プロジェクト･マネージメントを主にやっている人たちの期待は大きく、システム設計者、プログラム設計者とよりマシンに近い工程になるほど期待はうすい。MMLがプロジェクト･マネージメントを主にやっている人たちの作業を直接的に支援するとは思えず、逆に、期待のうすいプログラム設計者のほうが直接的に恩恵を被るであろうはずなのに...。どうやらソフトウェアの開発者にとって「MML」という言葉は胸を踊らせるにはほど遠いものらしい。

4.今後の展開に向けて(MMLとはあまり関係ないが...)

0から1にするのは難しいことはあったかもしれないが、失うものは少なかったのではないだろうか? これからは、すでにあるものとして慎重に開発環境を考えていかなければならない。3-2の「MMLによって開発環境は改善されるか?」の調査結果について考えさせられるものがある。「今でも問題は山積みなのにこれを積み残して、今度はどこにいくの?」という開発現場の声が聞こえてくるような気がしてならない。ハードウェアや物理的なしくみだけでは開発環境を向上させることはできない。ましてや、精神論では何をいわんや? 今後、フィールドの情報をいかに反映していくかが我々に求められているように思われる。このことに関しては、私は自分の所属する部門についてはやや楽観的な見方をしている。担当職務別構成から見ると、5人に1人はプロジェクト･マネージャということになり、5人分のフィールドの情報をプロジェクトという観点から掴むのはそれ程困難とは思えないからである。

マイクロ・メインフレーム・リンク
– UNIX ネットワーク環境からみて –

坂下秀
ss@astec.junet
ASTEC,Inc.

11 月21 日, 1988

- 自己紹介

  私はアステック (ASTEC,Inc.) という会社で、

  1. マニュアル制作
  2. プロダクトの移植
  3. 計算機の管理

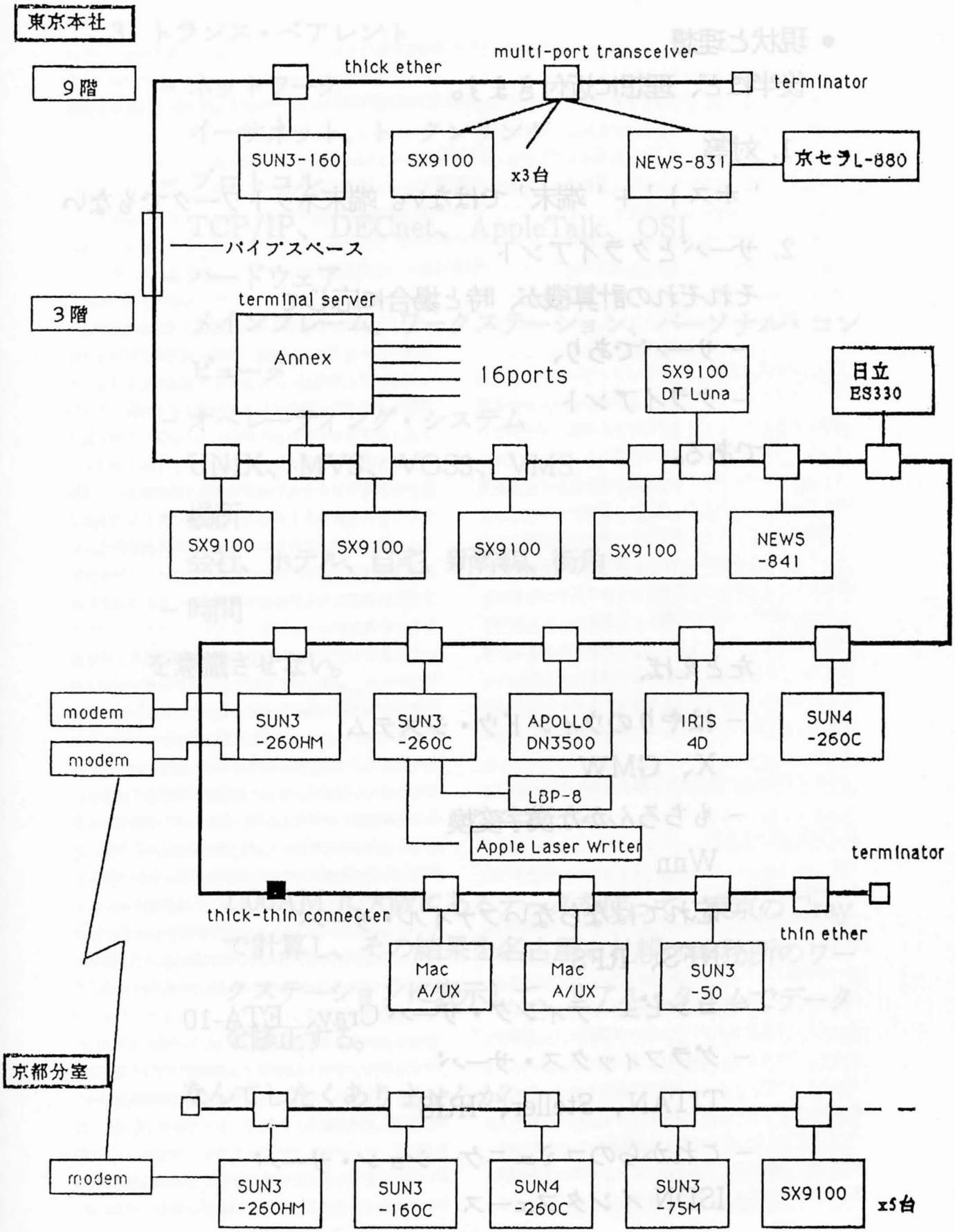
  をしています。

  アステックは、UNIX[1]ワークステーションで動くグラフィック・ソフトウェアの開発、販売をしています。でも皆さんなら、GMW/Wnn の開発と配布の会社といったほうが、よくわかるかも知れませんね。

- ネットワーク環境

  今の環境は、次のようになっています。

  - Sun(Sun MicroSystems) × 5
  - SX-9100(OMRON) × 7
  - NEWS(SONY) × 2
  - MacII A/UX(Apple) × 2
  - Apollo DN3500(Apollo) × 1
  - IRIS-4D(Silicon Graphics) × 1


東京本社
9階
thick ether
multi-port transceiver
terminator
SUN3-160
SX9100
x3台
NEWS-831
京セラL-880
パイプスペース
3階
terminal server
Annex
16ports
SX9100
DT Luna
日立
ES330
SX9100
SX9100
SX9100
SX9100
NEWS
-841
modem
modem
SUN3
-260HM
SUN3
-260C
APOLLO
DN3500
IRIS
4D
SUN4
-260C
LBP-8
Apple Laser Writer
terminator
thick-thin connecter
thin ether
Mac
A/UX
Mac
A/UX
SUN3
-50
京都分室
modem
SUN3
-260HM
SUN3
-160C
SUN4
-260C
SUN3
-75M
SX9100
x5台

- 現状と理想

後半ほど、理想に近付きます。

1. 対等

‘ホスト’＋‘端末’ではない。端末ネットワークでもない

2. サーバとクライアント

それぞれの計算機が、時と場合に応じて

- サーバであり、
- クライアント

である。

たとえば、

- はやりのウィンドウ・システム
  X、GMW
- もちろんかな漢字変換
  Wnn
- 忘れてはならないファイル・サーバ
  NFS、RFS
- コンピューティング・サーバ Cray、ETA-10
- グラフィックス・サーバ
  TITAN、Steller、IRIS
- これからのコミュニケーション・サーバ
  ISDN インタフェース

があります。

3. トランス・ペアレント

- ネットワーク
  イーサネット、トークンリング
- プロトコル
  TCP/IP、DECnet、AppleTalk、OSI
- ハードウェア
  メインフレーム、ワークステーション、パーソナル・コンピュータ
- オペレーティング・システム
  UNIX、MVS、VOS3、VMS
- 場所
  会社、ホテル、自宅、新幹線、街角
- 時間

を意識させない。

1:00AM に大阪にあるデータを使って、東京の Cray で計算し、その結果を名古屋と札幌の事務所のワークステーションに表示して、リアル・タイムでデータを修正する。

なんてしたくありませんか?

第4回　ＳＥＡ環境ワークショップ

セッション４　いまのＣＡＳＥツールはどの程度有用か？どの使えば良いのか？

チェアマン　：野村敏次　（ＪＩＰ）
プレゼンテータ：小野康生　（三菱電気）
：佐藤千明（長野県協同電算）
：桜井麻里　（ＳＲＡ）
レポータ　：澤田宜己　（大阪大学）

1. はじめに

野村（日本電子計算）：　おはようございます。最後のセッションになりましたけど、これからセッションを始めたいと思います。ＣＡＳＥツールの状況と言う事で、３人の方にお願いすることになりました。突如、セッション・チィアマンをすることになりました私は、日本電子計算の野村と申します。宜しく、お願いします。

順番はプログラムにあるとおり、最初に制御系でのＣＡＳＥツールにの利用について三菱電機の小野さんに。２番目にメインフレーム環境でのＣＡＳＥツールということで、長野協同電算の佐藤さん。最後に、ＳｔＰ利用経験と評価ということで、ＳＲＡの桜井さんに話して戴きます。できるだけ早く終った方が皆さんもうれしいかと思いますが、部屋は５時までとってあるそうです。出来れば、１２時半には終りたいと個人的には思っています。

それでは最初に三菱電機の小野さんからお願いします。

2. プレゼンテーション

2.1 制御系でのＣＡＳＥツールの利用について

小野（三菱電機）：　おはようございます。三菱電機制御製作所工業部の小野康生といいます。やっている仕事は一応鉄鋼プラントや紙パルプのプラントでのプログラマブルコントローラを自社生産しておりまして、これのハードウェア設計およびソフトウェア設計を７，８年ずっとやっております。制御系でのＣＡＳＥということで、コントローラについて、ソフトの生産について、ＣＡＳＥについて話したいと思います。

コントローラというのはちょっと耳慣れないかもしれませんが、三菱電機では昔からプラントを制御する制御盤、それから操作盤などを作ってまして、ソフトウェアというものについてはちょっと異質な感じがするかもしれません。マイクロプロセッサの発達なんかで今までハードウェア、例えば、リレーとか半導体なんかを使って制御していたものをマイクロプロセッサを用いて、プログラムによってプラントを制御するということで一応プラントに合った装置であると考えています。コントローラの中身はＤＤＣとしての算術演算とか関数とかパルスをカウントするようなものと、あとはシーケンス制御としての論理演算、それから最近では情報を制御するためのデータハンドリング等のものが多数混在しておりますのでこれらのところをうまい具合にやっていこうというものであります。

ハードウェアですけれども一応ここにＣＰＵとして演算部がありましてこのプログラムメモリー部に制御する内容、処理の方式などを格納します。それから後ここにプロセスのバスがついてまして外部の操作盤のスイッチとかアナログの操作量とかセンサーの信号とかカウンターのパルスなんかがプロセス入力として入ってきます。あと演算処理した結果とか操作量とかいうのがプロセス出力として出てきます。ラントでは特にモーターとか電磁弁とかで直接機械を制御しますのでそれ専用の出力インターフェイスカードがあります。さらにはデータウェイと称して他のＣＰＵとか上位計算機等とやりとりをするインターフェイス部分、最近では操作盤の代わりにＣＲＴを用いておりますので周辺装置とのインターフェイスが大体の構成です。

コントローラーの特徴ですけども、いま言いましたようにソフトウェアというのはプログラムメモリーにかかれるものでありまして、ＣＰＵはその内容を読み取って実行処理してますから、あとはハードウェアで行なうべきことで、ハードウェアとソフトウェアの両方のいい所悪い所あると思うんですけど、そういうところをうまくミックスしておりまして、なんでもかんでもソフトでや

ればいいというんじゃなくて専用のプロセスＩ／Ｏ、機能Ｉ／Ｏとかインタフェイス装置等ありまして高速かつ高機能を実現してます。

そういうことでコントローラーシステムにおいてはソフトよりもハードの比重が高くて、計算機のようにハードの上にプログラムがあるというわけではなくてそのプラントの構成されている操作器具、それから駆動系にあったハードウェアおよび、それを最善に動かすソフトウェアになっております。プログラムの実行周期ですが特に高速の応答が要求されておりまして、だいたい数ｍｓｅｃ、速いもので１～２ｍｓｅｃ、遅くても数１００ｍｓｅｃのサンプリング時間で制御されます。そしてこのようにハードとソフトが一緒になっているものなんですけれどもなにしろ直接プロセスに影響するものですから故障しますとそこのプラントがダウンということになりますので故障しないようにするにはもちろんのこと、故障してもその故障が他に波及しないようにＣＰＵは分散化が大前提になっています。このＩとかＯとかいうプロセス入出力はソフトウェアではメモリーのロードとかストアとかそういうものと全く同格のもので計算機のように入力出力のポートアクセスに時間がかかるとかいうことはございません。そして、これにはＯＳが入っておりません。言語は制御専用のＰＯＬと称している専用の言語でありまして、割り込み信号などのオーバーヘッドをなくしています。また、モニターとかユーティリティーなどのベーシックソフト等も不用にしています。

ソフトウェアエンジニアリングの特異性としては、ソフトウェアの独立性を確保するということで、制御が必要とする機能とか、イベントとかいうものを要求される応答周期とか、精度から分割してソフトウェア構成をしておりましてＳＡとかＳＤとかいう手法を前提としたソフトウェア構築をしております。またプログラムのほうの作り方なんですけれども、コントローラが直接プロセスを制御するということでインターロックを間違えたり実行順序を間違えると、バルブをＯＮしなければいけないのをＯＦＦしてしまってロールが逆に動いて板がぶつかってくるとか、そういう設備を壊すあるいは人身事故を招くとか非常に危険な、一歩間違えますと大きなトラブルにつながりますのでこのソフトウェアの作り方や処理につきましてはとにかくおかしくても止めるというようなロジックフローをしておりましてソフトウェア品質には非常に高い信頼性が必要になります。

だいたい、いま１年間にＣＰＵ台数にして１２０～１６０位でプログラムのステップ数としては１年に８００Ｋくらいです。ステップというのはさきほどお話した専用のＰＯＬ言語で書いたものでありますから汎用のＦＯＲＴＲＡＮとかＣＯＢＯＬとかで書きますとその６倍から８倍位のステップ数がかかると思います。作りだして１０年くらいになるんですけども最近はＣＰＵの能力もアップしてますし、Ｉ／Ｏカードを構成するＩＣとかそういうものも発達してきてまして能力はずいぶん向上してますし、最近いわれる情報化ということでコントローラーの設置台数がずいぶん増えてるんですけれども、そのためにソフトウェアというものが非常に急増してましてなかなかこちらの生産が追いつかないということで、そうするとうまくソフトウェアの品質の面と工程の確保が非常に重要な問題になっております。

現在うちのセクションで行なわれている生産方式を図に示したものです。あるプラントを受注しましたら計画の担当者が客先と打ち合わせをしましてどんな制御をするのかシステム計画を行ないまして、運転方案それから電機品リスト等を作ります。これらはいま全部手作業でやってまして一応図面の形で作るんですけども担当者がそのプラントにアサインされるとそのままそれで何ヵ月も打ち合わせして作るということで、そのまとめかたによってその人の個性がもろに出ますのでそれで完成度がどんなもんであるか非常にいま疑われております。外部仕様が決まりますと今度設計の所にきましてどういうハードウェア構成にするかということと、プログラムで処理するところの機能をチェックします。機能チェックとしてはどんな精度がいるのか、板厚の誤差はどのくらいにするのか、板の尾端を停止するときを±何ｍｍ以内にするのか、１ＣＰＵに入るプログラムがどれだけになるのかを検討していきます。機能について詳細を検討しまして、これの説明書、フローなどを書いていきます。実際プログラムはその詳細検討されたものからさっきのＰＯＬ言語というものでラフスケッチというものを書いてプログラムを作ります。

ＰＯＬ言語というのはプログラムの１ステップが「ＣＰＵが何をどうする」というふうなどっかのメモリーにストアするとか、どっかの入力から読んでくるとかいうことで、「何を」というところのものがソースと呼ばれるんですけれどもこれを１つ１つアサインしていかないとあかんのですね。実際のメモリー番地を考えるんでなくて一応ナンバーというのは自由に決められるんですけれども、それにラベルを付けておく必要があって、ここ

の設計製作の段階でもかなり個人の手作業というところが多くあってなかなかツールを使ってやろうというところまではできてません。そこで担当者が代われば担当者によってその作り方も全く同じにできるというわけでもないし、標準ロジックもあってもなかなかソース番号が違うとそこでミスが発生するとかそういうこともあってなかなか時間も手間暇かかっています。

製作段階ではＣＡＤを使ってシンボルフローとかコンコーダンスを出しています。ここは一応端末との対話処理でソースファイルを作ったり、そのＰＯＬ言語をコントローラの言語に置き換える正アセンブルや図面を自動出力しています。完成したソフトはこの実機の方にローディングして社内でデバッグし、今度現地へ行って調整をして終わるということで、試験の段階におきましても試験者の技量によって試験方法とかその試験結果の評価をやっておりましてなかなか自動試験も進んでおりません。そうして一応計画、設計、製作、試験と各作業部門あるんですけどもそれぞれマイナーにツールなんかもあるんですけれどもなかなかその手作業になる部分がかなりまだでありますから、活用ができてないというのが実情です。

制御分野での一応ツールの問題点としてちょっと考えたんですけども、一応電機メーカ各社でコントローラというものを作ってましていろいろそのスペックについては競争が激しくてハードウェアも新製品を出してもまた次の新製品の開発を始めるということで、なかなか製品ソフトウェアもすぐに変化しますし、作業環境もＣＡＤルームなどありますけれどもなかなかそれが追いつかないということで、ツールが製品ソフトウェアの変化、作業環境の変化こういうものに付いて行かないということが今一番重要な問題になっております。具体的に述べればその下の方に一応書いたんですけれども、ツールのメンテナンスができていない。ツールを実際開発した人がまた次のオーダーをやっていたりして、専任者がいなくて管理がもうひとつできていません。

評価検証システムが確立していない。そのツールがどのくらい有効なのか、どのくらい役に立つのかということについての基準がまだはっきりしていないということや、実際どのくらい使われているのかということのバックデータなんかがありませんのでこういうことが問題になっています。ごく一部の知っている人のもの、汎用性がないというか使い方が良く分からないとしてまだ広く知られていないところがあるみたいです。使ったことに対しての改良・改善の案をなかなか反映できません。メンテナンスのことと関係あると思いますけれども。出力情報の再利用ができていないということで、我々のソフトウェアはプラントごと、それから同じプラントでも客先によって全然違いますからオーダー対応でソフトウェアを組んでおりましてなかなか再利用ができておりません。

制御分野でのツールの活用について話をしたんですけども一応先ほども言いましたようにソフトウェアの品質向上、工程の確保ということが今一番最大の問題でそのためにやはり手作業が多いということからも機械化、人に頼る部分が多いということでそのパーソナリティに依存しているところをすこしでも小さくしようということでソフトウェアの自動作成ツールなんかを検討しています。特に計画・設計・試験というところでそれぞれ下流の部分が上流をチェックするというような生産システムをとっていますので、ツールが各その作業フェーズの所でマイナーに持っている部分をくっつけて一貫した生産ツールとして見直しを計っています。ソフトウェアのパッケージというものについても昔から言われているのですが、なかなかその利用の仕方としては亜流のものが出て来たりしますので担当者ごとに異なるソフトウェアの品質バラツキをなくすため担当者によって変わるというところを、それからケアレスバグ、つまりパッケージを利用したためにまたバグがでたということ、こういうことを阻止するために仕様書を含めたパッケージの一元化が必要でないかと考えております。これで私の発表を終わります。

野村：今の発表を皆さん聞かれていてたぶんここらへんの説明のコントローラの生産の実情までは分かったと思うんですが、突如問題点にいってじゃあここでどんなツールを使われているというお話がなかったんでちょっと質問しにくいんじゃないかなと思うんですね。実際に計画しましてそれから設計・製作という流れがありまして、そのところでどういうツールを使っていてそこでこういう問題点が起こったというそこがちょっと抜けていたんじゃないかと思うんですが、ちょっとそこら辺ご説明頂いたらありがたいんですが。

小野：　いまツールとしてあるのは社内にある大型の汎用計算機がありまして、計画の所でここにＩ／Ｏリストというのがあるんですけどもこれについてそのコメント

・ソースをインプットしまして、その情報から図面を自動的に出力させています。プログラムドキュメントとしてのここで定義しましたＩ／Ｏのソースがプログラムに実際そのまま出てきますので、このアウトプットを引っ張ってきて製作段階でそれをファイルとして使っています。設計・製作でのツールとしては先程言いましたドキュメントの作成の所でラフスケッチで書いたプログラムをＣＰＵがわかる言語に直す正アセンブル、プログラム図面のところ、このドキュメントを作る場合でもここにアセンブルチェックとか未定義ソースチェックとかいうふうにチェック機能が入ってましてツールといわれるとそういうものもありますが、製作の機械化の為には計画でいう仕様決定部門、設計でいう機能検討部門が今全然できていないところで、これはどうしても客先との打ち合わせというかその要求するものが違うということ、ラインの構成が全然違うということでなかなかできてません。ただ試験の所ではいつもハードウェアがあるわけではないので計算機をハードに見立てたエミュレーター装置があってそこにプログラムのデバッグをカードの出荷後からでもできるというツールがあるんですけれども、まだツールというのも自社生産したものばかりで、今例に出ているＥＷＳなどの活用を考えているところなんですけれども、今あるツールとしては社内の大型計算機と計算機端末が主です。

岡本（ＫＣＳ）：　先程のプレゼンテーションの中で年間に１６０ＣＰＵ位、８００Ｋステップ位の開発というお話がありましたけれども、基本的に商品としてのソフトウェアの開発はチーム作業、何人かの人間が協力しあいながら一緒になって作ってるわけですか。例えばプログラムでは複数のプログラマーがいて個別にそのプログラムの開発を分担して最後にガチャッと組み合わせて作るようなスタイルなんですか。あるいは、私が以前こういうふうなコントローラのプログラムを開発している方に伺った話によると、そういうケースももちろん無いとは言えないけれども基本的には頭から最後まで一人で作るというケースも多いと聞いてますけども、そのあたりコメントして下さい。

小野：システムによってマルチ構成になったりスタンドアロンになったりするんですけども一応１ＣＰＵを担当者一人というふうなやり方をしています。だから機能チェックの段階でどのＣＰＵにどういう機能が入っていれば１つのＣＰＵの中の機能がまとまるか、つまり機能をあちこちにばらまきますとその間のインターフェイスが増えますのでなるべくインターフェイスは疎にして中で機能を閉じるというかたちをとっていますので、１つのプラントの中ではＣＰＵ部分がスタンドアロンもあるけれどもホットストリップミルとかいう部分、この部分では２０ＣＰＵ位あります。それについて一応セクションごとに方案とかいうものがあって、あとそれが２ＣＰＵ、３ＣＰＵ位で複数ＣＰＵで成り立っていればどの機能をどのＣＰＵに入れるかということをシステム計画段階で指示しますから、プログラムの製作になると指名された担当者はその決まったＣＰＵについて最初から最後までロジックを組んでいくというやり方をとっています。

久保（富士通）：　１６０ＣＰＵで８００Ｋという話ですけれども、プラントの数でいったら何プラントぐらいなのか感じを教えて欲しいのと、１プラントに複数ＣＰＵの場合だとその納めるものがどの程度に違うのか、複数のＣＰＵで全く違うものが入るのか、その辺の常識がないのでという感じのが１つと、８００Ｋというやつですね、それはおそらくお客さんに納めるステップ数の量だと思うんですけども、そのうち再利用の部分はどの程度あるいは逆に引き算した新しく作るのはどんな感じですか。

小野：　プラントの数というと民間の企業の景気によって設備投資をする場合と全然無い場合とでいろいろありまして、プラントの数は年ごとに浮き沈みあります。ＣＰＵもいま１６０と言ったんですけどもこれも小容量のものから大容量のものまで３・４種類くらいうちではコントローラの種類があるんですけども一概に何個と言うわけにいきませんで、鉄鋼の場合だと１つのプラントに２０ＣＰＵもあれば、紙のプラントであればスタンドアロンが多いんでプラント１つにＣＰＵ１つとかそういうことで数はなかなか言えません。ＯＳとかそういうベージックソフトがうちの場合ありませんので、１ＣＰＵの中に入っているプログラムはイニシャルの部分とかサブルーチンの部分こういうものはうちの設計で標準のものがあって、ファイルで登録してあってそれを引っ張ってきていらないところを消すくらいのことで、それが０．５Ｋ位ですかね。１つのＣＰＵで０．５Ｋ位が使えますけども、あとはプラントによって機能によって全く前と同じ様に使えるものであれば８０％位流用率ありますし、

全く新しい新規機能であれば１から全く作らなければならないということで、今まで１０年間くらいずっとこういうのがありますのでそういうノウハウありますのでなんでもかんでも１から作るというところまでいっていないんですけども、なにせ手作業が多いというのが問題です。

2.2 Ｍａｉｎｆｒａｍｅ環境のＣＡＳＥ

佐藤（長野県協同電算）： どうもおはようございます。私、実は昨日から３つくらい慢性的な病気になってまして、１つは林さんと同じで慢性ＧＰＴ病、２つめはやっぱり林さんと同じで慢性プリン体病、３つめが昨日岸田さんに指摘をされました慢性メインフレーム病ということであります。そういう意味では病床からの報告になりますので病人をあんまりいじめないようによろしくお願いしたいと思います。

今日お話したい内容は半分は建前の部分で、これは全体の話をしてみたいと思います。もう半分は私個人の環境におけるいろんな経験をふまえた形での本音の部分をお話し、最後に若干ＣＡＳＥといわれているものに対しての私なりの主張をお話したいと思います。時間が３０分ちょっとですので駆け足になると思いますけどもよろしくお願いします。

まず、これは釈迦に説法ですか説教ですかどっちにしてもここにいらっしゃるみなさんはＣＡＳＥについてのある程度のお考えをお持ちの方なので、セミナーじゃありませんからそんなに詳しくはご紹介できないんですけど、私のＣＡＳＥについての考え方みたいなものを最初にお話します。これはある雑誌にあった記事を引っ張ってきんですが、いわゆる第１世代といわれているものが今から４・５年前に使われていたわけですがこれは上流支援ということでストラクチャードアナリシスとかストラクチャードデザインとかそういったもので特に絵を中心としたダイアグラムをいかに効率良く作るかというようなところのツールではなかったかと思います。

それに対してもうちょっとそれを上から下まで一貫して支援するようなツールはないかというような形で、（特にこれは第１世代の方はあくまでもデザインとか紙でやっても何とかなったものを機械化したという環境だと思うんですけども）、それがある程度オートメーション化されてきたと。いろんな紙の間のリレイションというものがもうちょっとちゃんとチェックされるようになってきてオブジェクトまで作れるようになってきたというのが第２世代であろうか。尚、今日私がお話するのはメインフレームの環境ですけどその中でもさらに事務処理のオンラインアプリケーションみたいなところの仕事をしてましたのでその辺に絡むお話になると思いますのでそういう前提で聞き願いたいと思います。あとその次にも上から下だけじゃなくて逆に下から上までっていう流れも出てきて、リバースエンジニアリングとかいうような非構造的なソフトウェアをいかに構造化して再利用するかというような環境を持つ必要がある。

ですから昔ＣＡＳＥと言われてなかった時期はいろんなツールがあって、いわゆるソフトウェア開発支援ツールということで特にこの辺はプログラム設計とかプログラミングとかテストとかメンテナンスとかこういったところを簡単にサポートしていた。それがずっと今までの歴史だったわけですけども、それに対してこういう上の方を何とかしろよという世代が第１世代で、第２世代はそのままこの辺、保守までいってずっとつながってきます。第３世代はこっちからこう上へあがってきますよとそんなような位置づけではないかなと思います。

ＣＡＳＥと今までのプログラミング環境とどう違うのか、詳しくは分からないんですが、アナリシスとかデザインというフェーズをサポートしていることと、もう１つはＣＡＳＥの裏にはメソドロジーがあること。即ち、昨日もお話があったんですけどもツールとかこういう環境は裏にある程度の方法論とか手法があってそれをいかに機械化するかというのが発想だと思います。そんな方法論みたいなものはどんなものがＣＡＳＥの場合あったかというと、Ｄｅｍａｒｃｏ、Ｇａｎｅ・Ｓａｒｓｏｎの構造化手法みたいなものがあったり、もうちょっと下の方にいくとＷａｒｎｉｅｒとかＪａｃｋｓｏｎとかいうのがあったり、もっと情報システム全体をどうとらえるかということになるとＪ・Ｍａｒｔｉｎさんがいろいろ言っているようなことがあったりとか、そういった方法論をいかに支援するかということでＣＡＳＥというものがあるんじゃないかなというふうな捉え方をしています。

そういううんと広く捉えた場合のＣＡＳＥとしてどんな技術があるか、今言ったことを別の切り口で見たものがこの絵です。単純にはＵｐｐｅｒＣＡＳＥとかＬｏｗｅｒＣＡＳＥとか言われてますけども、構造化手法は昔から考え方としてはあったけども、これがワークステーションが発達してきたことによって現実的にこういったお絵書きツールがちゃんと使えるようになってきたとい

うのが1つＣＡＳＥの流れです。もう1つプログラミングでは従来のＣＯＢＯＬとかＦＯＲＴＲＡＮの環境の機械化があり、ＣＡＤ、ＣＡＭのアプローチだとか部品とか再利用とかデータデイクショナリーとかいろいろ言われたわけです。その辺のツールは、ここにあげてるような一貫支援をサポートするツールです。特にＣＯＢＯＬのような３ＧＬ環境で何とか機械化したいというと、チャートエディターみたいなものを用いた機械化にみられるＣＡＤ、ＣＡＭ指向の環境です。

次に、もうちょっと言語レベルを上げようということで出てきているのが４までいかないけれども３．５（例えば真ん中取って５としますと）のいわゆるアプリケーションジェネレーター、ＣＯＢＯＬジェネレーターみたいなものがある。スペシフィケーションツール、仕様化支援技術、さらにもうちょっと言語そのものを高級に記述するにはどうしたらいいか。それは構造化設計、構造化プログラミングとか言われてきてるのとはちょっと違うが、いわゆるＤｅｍａｒｃｏ流の考え方でいうデータフローみたいなものがものすごく現実的な事務処理のアプリケーションを解くうえで考えやすい。そういうことは分かっているんですが、それを実際に言語としてインプリするような環境があったんだろうかということで、たまたま私の所で使っているものがありますのでその話を後でちょっとします。

次に、今までの言語の世代とはもう1つレベルを上げて第４世代環境みたいなものが出てきていて、これはもうどちらかというと全部をひっくるめてサポートしようという環境です。それは単に言語レベルだけではなくてビジネスシステムを生成したいという環境だと思います。その中には当然分析手法みたいなものも出てくるし、いかにモデル化するか、更にデータディクショナリーが前提となってスクリーンペインティングしてプログラムまで全部作ってしまって全部ひっくるめてソフトウェアデータベースとして管理しましょうというような環境だと思います。３ＧＬのこの辺のＣＡＤ、ＣＡＭ指向と似てはいるんですが、実は４ＧＬはこのオブジェクトを定義すれば全てビジネスシステムが生成されるのに対して、３ＧＬはそういう意味では個々のフェーズだけをとらえてその機械化を支援している。

次に、データベース技術みたいなものもやはりリレーショナル指向というのが今当たり前になっていますが、その辺の所でＣＡＳＥというかいわゆる構造化手法とか構造化分析とか言われているのはちゃんとやりましょうよという考え方もあります。

もう1つはプロトタイピングというのがあって、これはどちらかというとちゃんとできないからそれをどうやってカバーしようかということを考えているわけですけれども、その辺の所の技術というか環境みたいなものが整備されてきて、実際にビジネス分野においてプロトタイピングをサポートするような環境が出てきている。これによってＳＡとかＳＤでやろうとしているビジュアライゼーションというような環境が実はプロトタイピングをやるとできちゃうわけです。そういったところがＣＡＳＥを取り巻く技術としては有るんじゃないか。

次は構造化手法の環境みたいなところで、ＳＰがあってＳＤがあってＳＡが出てきたわけです。ＳＡそのものの狙いはこの通りですが、膨大な手仕事のためペーパーレスでなくてはどうしようもないというところが問題になり、それに対して今はハードウェアの技術の進展によってちょっと環境が変わってきて、「構造化手法よ、もう一度」というような状況ではないかと思います。但し、私自身は構造化手法とか構造化分析ということに諸手を上げて賛成しているわけではなくて、実はその一番下にあるようなこういった４ＧＬとかプロトタイピングとかリレーショナルデータベースとかビジネスアプリケーションを作る場合にはおそらくこういった技術とうまく組み合わせていく必要が有ると思っています。ちゃんとしたプロジェクト手法で上からちゃんとやってきて下までいくならばウォーターフォールそのものは否定されなかったんでしょうけどもそれがうまくいってない。それは紙だからうまくいってなくて機械化したらうまくいくかというとそうでもなくて、もっと別の要素を取り入れなければいけない。その1つのきっかけとして４ＧＬとかプロトタイピングとかいうところをビジネスアプリケーションの構築の所で使っていく必要がある。

さて、構造化手法で有名なのはＤｅｍａｒｃｏ流の考え方です。システム設計とかもうちょっと情報システムそのものをどう捉えるかということは今日のこの後のＳｔＰでもありますけども、私の所で今実用化しているのは、その後の所のプログラム生成です。これはよくあるデータフローです。これを１個１個プロセスと考えてこの周りをターゲットエリアとすると、いわゆるコンテキストダイアグラムになるわけですけども、これを１つのプログラムとして見た場合どうなるか。やはり、プログラムに対してはそれと絡むデータがあってなんらかのアウトプットがされる。その中は実は最初にマッチングが

あったりして中間的に何か出てきてそれとまた何かをぶつけたり。あるプロセスからなんかアウトプットが出てきて何とかリストになったり。こんな風に考えると与えられたプログラムをどう設計するかというときに、難しい構造化チャートみたいなものを作らなくとも、あるいはモジュール仕様をモジュールの結合度とかなんとか言わなくとも、（この場合バッチの業務のウェイトが高いんですけども）、こういう風に問題を素直に表現してあげればそれがそのまま実際に業務として動くプログラムに生成されるツールがあるわけです。このときのポイントはこういう中間的なものが出てきますから、これを物理的なもので使ったら効率が上がらないわけで、いかにこれをメモリー上でやるかということ。その鍵をにぎっているのはコルーチンの考え方です。Ｊａｃｋｓｏｎシステムプログラミング、ＪＳＰでいう構造不一致が起きたときに、その間を中間ファイルでやるんですが、それをいわゆるコルーチンでライトしたつもりでこのサブルーチンを呼んであげるというような環境です。これ自身は実はもうガンガン動いてまして、私の所ではこのプログラムをどうやってジェネレーションするかというと、プロセスとプロセスの入出力（プロセスフロー）と、そのプロセス自身が何をやるかという構造・機能（例えばこれだったらマッチングをするとか、これは何とかチェックをするとか、いろいろな機能があるわけですが）、この２つで定義します。そして、各プロセスはこういうＪａｃｋｓｏｎ流のプログラムツリーで表現します。例えば、更新というプロセスを考えると、マッチした場合どうなのという、アンマッチの場合どうなのという、マッチした場合はマッチした中のなおかつグループのイニシャルでどうなのというその１歩１歩の場において何をするかということを記述してあげれば、実はそれでプログラムができる。そこまでスケルトンがはっきりしてますと、そういうプログラムをさらにジェネレーションするプログラムというのは簡単にできるわけです。

プロセスで枠が決まって、１個１個のプロセスがどういう構造を持っているかということが決まってきますから、あとはこのはっぱの部分で何をするかだけを決めてあげれば当然プログラムはできるわけで、そういった環境を貧弱なＴＳＳ環境ですからキャラクターオリエンティッドで当然やるわけですが、一応会話型でいろいろ聞いてきて何か指定するとスケルトンが一度に６００ステップ位パッとできる。従来のプログラムジェネレーターというのは枠がある程度決まっていてパラダイムが決まっていて、それをダイナミックに動かすというのではなくて、決められた型をいかに組み合わせるかということでやってきているんですが、これはそういう意味ではもう少しフレキシブルで、プロセスの組み合わせを自由にできるというような環境になっています。いわゆるデータフローみたいな考え方を実際プログラム作成に適用したツールは、今ご紹介した他にあと私が知っているのは２つか３つ位あります。これは日本電気さんのＩＤＬというプロダクトですけども、まさにこれも同じことをやっています。こういうファイル間の関係というものをどういう処理をしてどういうふうに落とすかと考えた場合、これ１個１個のプロセスを１つのジョブステップと考えると従来と同じなんですが、これをまとめて１つのジョブステップにするにはどうしたら良いかというあたりでやはりデータフロー的な考え方です。これはデータオリエンティッドチャート、ＤＯチャートとか言っているんですが、このときのプロセスの機能とは、む連結、編集、結合、更新、マッチングとかいろいろあって、なおかつこれは対話型なところまでサポートしているので画面からもらってくるプロセスなんかも定義できます。これを組み合わせることによって１つのジョブステップ、１つの物理的なプログラムが出来上がる。富士通さんのＣＡＳＥＴも基本的にはこういう考え方でやっているようです。この辺の所までは、メインフレームでもきてるんです。ここまでは、いわゆる言語レベルでの話なんですが、ＣＡＳＥというのはそういう意味ではもっと上の所までサポートする。あるいはもっと全体を通してという話になるわけで、その次のフェーズにいきまして、４ＧＬ環境上での構造化手法支援システムという話をしてみたいと思います。

昨日、一昨日とデモがあったＭＡＰＰＥＲというのはいわゆる第４世代と言われているんですが、私は個人的には第４世代とは思っていません。ちょっと異質なソフトウェアだと思ってましてあんまり好きじゃないんですが、その中で、なぜそんな話をするかというと、それ自身ではなくてそういうＭＡＰＰＥＲアプリケーションをいかに作るかというようなところで構造化手法をちゃんと取り入れているソフトウェアであるからです。今、この上の方を非常に問題としてますから、この辺の所をどうするかと考えますと、トータルなビジネスシステムをどうするか、情報システムをどうするかというようなところから攻めようとします。情報システムそのものをどう作るかという場合の分析ということでは、まずとにか

く何をやりたいかとかいって、（これはコンテキストダイアグラムを作成することと同じことなんですが）、こんなことをやるわけです。その後そこにどんな問題点があるかということをちゃんとチェックをしていく、分析をしていく。それから現状システムが本当はどうなっているかちゃんとモデル化しようということになると、これはレベル０のＤＦＤからどんどん落とすというようなアプローチです。それに対して本来の情報システムは、これからどうありたいかという企業レベルでの目的に対して、どんなものがいいかというような代替案をいくつか作ろうとかいってやはりＤＦＤを作る。それから最終的なものを決める。こういうアプローチというのは今ほとんどのところのＣＡＳＥツールと言われているもの、例えばＪａｍｅｓ　Ｍａｒｔｉｎさんのナリッジウェアという会社のＩＥＷというＣＡＳＥツールもやっぱりまるっきりこれと同じアプローチをとってます。これは抽象的ですから正論なんで、問題はこの下をどうするかということなんですけども、ご紹介したかったのはこういう４ＧＬの環境でもこのようなアプローチをちゃんととっているものがあることです。ちゃんとしたダイアグラムいわゆるＤＦＤができた後、それをあとどうしていくか。これはＭＡＰＰＥＲというリレーショナルな思考で、アプリケーションをどう作るかという手順を示しているわけですけども、単に４ＧＬだから入れればすぐ動くというだけではなくて、この辺の所もちゃんとモデル化して方法論を持っていますということです。時間がありませんから詳しくはご説明できないんですけども、これはいわゆる建前の話です。じゃああなたのところでこれをどこまでガンガン使っているのと言われると、どんなものかっていうことでちゃんと機能を調べて、さてじゃあやろうかやるまいかということになるとまだ足を踏み入れてない。じゃあその他に何かあるのということでもう１つ別の方法をご紹介したいと思います。

そういう意味では５番目の所ですけども、上からちゃんとやろうよというのが理屈としては分かったけどもなかなかそれをすぐ使うというのは難しいということで、別のアプローチとして出てきているのがプロトタイピングとか４ＧＬです。同じ４ＧＬでもちょっと環境が違う話なんですが、これも昨日あったんですが、ＬＩＮＣという世界です。ＬＩＮＣという世界はこの絵が全てを物語っているんですが、基本的には全て端末機の上でスペックを決めていきます。画面のスペックだとかレポートのスペックだとかあるいはロジックなんかをこの中で全部やる。そしてその与えられた情報から、一旦、仕様データベースというものを作り上げる。その時にデータディクショナリーというものがまず前提にあって、それを最初に作って、それを使いながらシステムの仕様を決めていく。ここまでやりますとあとはコンピューターの世界でこう実際に実行環境まで全部作り出すわけですけども、こういったところは全てＬＩＮＣに任せてしまい、ユーザーがやりたいことはここだけやればいい。ここまで作って仕様データベースに入れてしまえばあとはせーのでジェネレーションというと、お昼を食べて帰ってくるとシステムができていてすぐ動くというような夢みたいな話です。

これは八ヶ岳に持ち込んで実演やりましたから嘘ではないということは、何人かの方がご承知だと思います。僕自身は大分これを気に入ってまして、とにかく楽ができる。いままでですとデータベースを使うのにもグチャグチャいろんなパラメータ、スキーマを定義したりとかいろんなことをやらなきゃいけなかった。それからいわゆるオンラインシステムのモニターみたいなもの、システムソフトウェアですけども、当然それのチューニングもしなければいけないし、その環境も作らなければいけない。通信関係に関してもコミュニケーション関係のいろんなソフトウェアを準備しなければいけない。ところがＬＩＮＣの場合にはそんなことは、当然最低限のことは１回やるわけですが、それはシステムがどう考えたって分からないところは外から与えますけども、基本的にどう動くかとかいうソフトウェアの中身に関してはシステムが全部気を使ってくれて作り出してくる。ですから極端なユーザがあって、実は自分達がこうやってプログラミングしたあとそれがＣＯＢＯＬに落ちてどう動いているか知らない、ＣＯＢＯＬに落ちていることすら知らないユーザーがいるわけです。それでガンガン動かすということがあるわけです。そのユーザーにとってみるとこの環境しか見えないわけなんです。この環境しか見えなくても自分達が必要とするアプリケーションができる。そういったところまで今のメインフレームの環境でも、ターゲットはビジネスアプリケーションですけども、実用化レベルにきているという気がします。なんでＬＩＮＣがうまくいくかというと、実は作り出すターゲットシステムが１つのスケルトンを持っているので汎用ではないというのがキーポイントです。ですからどんなことでもなんでもできるというような形で欲張ってしまうと、汎用性を持たせてしまうから逆にユーザーが定義しなけ

ればいけないところがいっぱいでてくるわけです。ところが限られたドメインでやりたいことがある程度決まっていてそれをどう作るかというあたりで制限をしてしまえば、それをターゲットとしたシステムは作り易くなる。そういう意味でモデルというものがありまして、ＬＩＮＣの対象システムでは基本的には何かデータベースがあって、そのデータベースに対してデータを追加するとか内容を変更するとか表示するとか消すとかそういう作業があるわけです。それはデータベースの項目さえ決まってしまえば基本的にはあとは全部システムがやってくれていいわけで、ＬＩＮＣはそこのところをやってくれる。もう１つビジネスとして考えるとこういうデータベースの中のいわゆるマスター的なものに対して日々発生するイベントというものがる。そういうトランザクションみたいなものは画面から入れてくるわけで、その時に入ってきたトランザクションに対して通常の場合ですとマスターを読んで更新するとかいうロジックが入る。

それからマスターも別にレポートを出すということがあるわけで、その場合にＬＩＮＣが考えているのは、データベースそのものは１つのスキーマで切られて固まってしまうが、ユーザーによっては私はこういうレポートを見たい、私はこういう風に分析したい、こういう条件のものだけ引っ張りだしたいというのは当然出てくる。その場合にまるっきりアドホック的な使い方をするといわゆるリレーショナルデータベースみたいな環境になるんですが、ＬＩＮＣの場合にはそこまで完全にフレキシブルではなくて、そのかわりにシステム生成の時にデータベースに対して、サブスキーマ規模ではないんですが、プロファイルいわゆる外部ビューを設定する。私はこういう風に見たい、私はというのはアドホック的ではないんですが、業務としてあらかじめ洗いだせればそれをビューとして提供している。ですから例えばイベントがいくつか出てきたら、その関連するイベントをまとめてある条件のものだけ引っ張ってきて何か処理をしたいといえば、その１つのまとまりに対してこういう見方をしたいというビューをきってあげればいい。それはデータベースに対するビューとイベントに対するビューがある。そういった形でシステムを作ろうというのがＬＩＮＣの基本的な考え方です。

この表題にはプロトタイピングということがあるんですが、じゃあなんでこれがプロトタイピングかというお話を次にしてみたい。今絵でもって仕様定義すればシステムができあがるという話だったが、ユーザーが自分達のアプリケーションをどう作りたいか、自分達のニーズがどこにあるかということを考えた場合に、実はそれは最初から完全には定義できない。そうするとまずおおまかな処理としてこんなデータ入力があるでしょう、こんな画面が考えられますとかいうのを最初に作る。ＬＩＮＣ自身が作り出すのはこの基本的なベーシックな機能しか作りませんからこのへんの細かなロジック、例えば入力された情報がどういうチェックをしなければいけないとかどう絡んでいるかとかそんなことは抜きにしといて、ユーザーに対してはこういう画面でこういう風に答えられればいいんでしょうというのを最初に見せてしまう。その画面を作ればそれでお昼を食べて帰ってくればシステムができるわけですから、それでいいよということになればそれに対してロジックを肉付けしていく。その時作るロジックというのは実はここで大枠のロジックは決まってしまっていますからそれを根本的に作り替えるようなロジックにはならないで、後から追加していくロジックなので比較的作りやすいわけです。ただ一般のビジネスアプリケーションを作るときよく出てくるのは、実はこういう場合こういう風にしたいとか後からいろいろ出てくるわけです。そういったところというのはＬＩＮＣの場合にはやってみてダメだからもう１回ということがどんどんできる。それから画面なんかに関してもこういう項目がちょっと欲しいなといったような場合、昔のやり方ですとちゃんとデータベースから何から変えて、それから画面の内容定義も変えて、電文の内容の定義も変えて、そういった手順を踏んでそしてまたシステムを作り直してテストするといった環境だったんですが、ＬＩＮＣの場合には画面の項目を追加するだけで、それで作り直せば全部動く。その仕掛けのポイントはＬＤＬという仕様データベースです。画面からの情報は全て中間的な仕様記述言語ＬＤＬに変換されデータベース化される。仕様変更はこの仕様データベースに反映されるだけであり、その下のオブジェクトには直接反映しない。生成指示をしたときに始めて、全てのプログラムをいっきに生成しなおす。各オブジェクトを個々に変更しなくて済むため、プロトタイピングの回転が速まるわけです。

次に、ＬＩＮＣをＣＡＳＥとして見た場合どういう評価ができるか、設計フェーズでどうなのという話と実際の開発フェーズでどういう機能があるか、それからユーザーインターフェイスに絡んでどうなのか、管理の面ではどうなのか、これを１つの実行環境としてみた場合どうなのかという分け方をしてみました。設計という意

味で見ますとスクリーンペインティング機能というのがありまして画面入力をすればいいわけで、そういう意味ではプログラムのビジュアル化みたいなものが１つは達成できている。それからレポート作成プログラムのビジュアル化ということで、いろんな帳票とか資料をモニターに出したりプリンターに出したりというようなところが画面でデザインをすれば終わりなんです。昔のツールは画面でデザインをしてそれをさらにまた別のオブジェクトに落とす作業が必要だった。ＬＩＮＣの場合には、決められたパターンの中であれば画面で定義すれば終わりで、確かに考えてみれば当たり前のことで、どのフィールドに何を打つかということだけ定義してあげればそれで基本的にレポートは出るわけなんで、そんなところはユーザーがプロットしてあげればいいじゃないかということです。それからもう少し設計を大きく捉えるとエンドユーザーコンピューティングというものをサポートしている。いわゆる構造化手法、構造化分析、ユーザー仕様をいかにＥＤＰの人間がちゃんとコミュニケーションギャップを埋めながら理解できるか、我々が考えているものをいかにユーザーにうまく説明をして、それは実はＥＤＰの人間とユーザーの人間が両端にいてその溝をどうやって埋めようかということを考えているわけで相手がちょっと遠くにいるわけです。ところがＬＩＮＣの場合はユーザーも一緒にやりなさいよ、どちらかというとむしろユーザーが主導でやりなさい、自分達のニーズは自分達で決めればいいじゃないかというところです。ところがその自分達のニーズを出すときに、あるいはそれをいかにスペシファイするかというときにコンピューターの世界がいきなり来ちゃったんでそれに入れなかったんです。その辺の所をこういうようなビジュアライゼーションというような形の環境を作ってあげて、モデルを定義してあげて、先程のコンポーネント、イベント、プロファイルというようなああいうモデルでもって定義してあげて、このモデルにそった形でユーザーのリクワイアメントをちゃんと定義して下さいというような環境なんで、これだとユーザーが入り易いわけです。そういったアプローチでギャップを埋めるというような考え方です。

それから開発というフェーズで捉えますとプロトタイピング手法なんですが、いわゆるプロタイピングでもこれはＥｖｏｌｕｔｉｏｎａｌプロトタイピングということですから進化していく。つまり最初に作ったもので中の枠が決まった。その次はそれをベースにその上にまたちょっと付け加えていく。それがどんどん繰り返えされていっていつの間にか本番という環境です。そういうアプローチがとれているわけです。それから仕様記述そのものは先程中間的な言語というお話をしたんですが、従来の言語よりは言語水準が上で抽象度の上がったプログラミングです。従来のコーディングというような作業は、ここでの記述がコーディングになってそこから下の実際のオブジェクトまで自動生成されるため、必要ない。又、ＬＤＬは当然、言語としてフォーマライズされており、システム全仕様で管理していますからＬＤＬデータベースからドキュメントが出て来たり、いろんな関係を引っ張りだすことができる。それからデータディクショナリー機能というのが大前提にあってそれをベースにしています。これを使うことによって標準化を図れる。省力化を図れる。正確化を図れる。それから自動的な保守、つまり実際問題どこまであるか分かりませんが、桁が８桁から１０桁になったといった場合にはデータディクショナリーだけ変更してジェネレーションしなおせば関連したところが全て直る。じゃあデータベースの中身はどうなるかというとデータベースの中身も全部構築し直してくれるというようなところまで持っているようです。他にもデータベース自動的に作るとか、データベースとのインターフェイスルーチンも自動的に作る。又、オンライン環境の自動生成、いわゆるミドルソフトウェアといわれている部分、即ち、モニターみたいなところもＬＩＮＣ自身が持っていますから、あるいは作ってしまいますからユーザーは知らない。今オンラインシステムを作ろうとするとコミュニケーション環境のソフトウェアがどうのこうの、データベースがどうのこうのと大変です。それがＥＤＰの人間の飯の種なんですが、そのへんのところが（限られた環境なんですが）自動的にできてしまう。その他のロジック部分というのは当然あるわけで、それはいくら画面を作ってもだめなんでその部分というのはあるんですが、従来よりはエンドユーザーがある程度理解できるというような環境まで非手続的になってきている。あとユーザーインターフェイスのところでいうと一応こういうロジックを入れていくときには全部構文エディターを持ってますからそれが基本的に全部入力時にチェックしてくれる。メニューでやるかコマンドでやるかという問題があるんですがこれは両方持っています。ですからよく分からない時は階層型のメニューでどんどん引っ張ってくればいいし、いきなり飛びたいときはコマンドでポッと飛べばいいという環境まで持っている。

それから管理面ではどうかというと、全ての作業をこのＬＤＬというところで行なっていますから、その進行状況というのを全部持っているわけなんで、そこからどのオブジェクトがいつできたのか、どこまでいってるのかそういったものは全部わかります。その他に実行機能があるんですが、これは開発環境という意味ではあまり関係ないのでちょっと略します。こういうような点がＬＩＮＣのＣＡＳＥ機能として捉えられると思います。

良いことずくめではなくて注文がいくつかあります。やはりユニシスという１つの世界で、しかもその中で今動かせる端末機の機種も限定されるのでもっとポータビリティーが欲しい。実はそれができますと２つの意味の分散ネットができる。１つはＬＩＮＣ自身がかなり下のレベル、マイクロレベルのＬＩＮＣが動けば、いわゆる垂直分散です。昨日いわれたMMLのうち実行環境みたいなものはＬＩＮＣの閉じた世界で構築できる。もう１つは機種を変えて他のメインフレームの上にＬＩＮＣがのりますと水平分散になる。今は決められた機種にしかのらないんですがもうちょっとそれがポータブルになってくれば水平分散みたいなものができる。ただし完全なヘテロじゃないんです。ですから異機種間接属を完全にやろうというのは無理です。それはやはりメーカーの戦略みたいなものがあるみたいですからここまでのものになるとはちょっと考えられないと思います。端末機なんかも今ちょっと標準じゃないですが、大体スケジュールにのっているみたいです。やはり貧弱なユーザーインターフェイスですから今のかな漢変換もホストへ送ってホストでやってるようなところもあります。最新の端末機でやってますけどもそのへんのところとか、使い勝手を良くするんだったらカラー表示とか高輝度表示とか、あるいはいろんな伝票入力するときにマルチウィンドウになってどうのこうのというようなそういった実際にアプリケーションを動かすことを考えるとそういう機能をもうちょっとうまく使って欲しいと思います。もう１つはMAPPER・ＫＩＴのところで言ったんですが、MAPPER・ＫＩＴというのはどちらかというと上の方をちゃんとやろうよということで構造化手法もちゃんとあってそれに沿った形で手順が決まっています。それに対してプロトタイピングですからある意味ではいきあたりばったりで作っていってもシステムが動きますということなんですが、小さい場合にはそれでもいいんですけども、システムが大きくなってくるとやっぱりモデル化ということをちゃんと考えなきゃいけないんですが、その辺の体系がまだ未整理です。たとえばビジネスモデルを作るときに情報システム全体の流れ、データフローみたいなものを分析したらそこからパッとＬＩＮＣのモデルにいける。ＬＩＮＣのモデルはビジネス向きで、はっきりしてますから比較的簡単にいくんじゃないか。ＳＡからＳＤへのスムースな変換というのはなかなか難しいが、ＬＩＮＣのモデルは比較的スマートに変換できるんじゃないかと思うんですが、その辺の方法論ていうのが今ないというのが実態だと思います。それからあとは言語レベルが低いんでもうちょっと上げて欲しいとか。

エンジンに使っているＤＭＳ　はＣＯＤＡＳＹＬ型であるが、ＬＩＮＣはそのうちのＩＳＡＭ機能しか使っていない。外部モデルではプロファイルとして半分外から与えているが、もう少しアドホック的なビューの提供も欲しい。今できるビューの設定の方法はいくつか制限がありまして、本来のデータベースとデータベースの間をある関係で切ってくるというようなリレーショナル的なアプローチがない。（ＬＩＮＣと別のソフトウェアではある）

あと開発環境の実態ですけども、これはたまたま今、例としてプログラム作成ということでＤｅｍａｒｃｏ流のデータフロー的なプログラムジェネレーターの話と、それから４ＧＬということでMAPPERとＬＩＮＣということでお話したんですが、もっとよそのメーカーさんのお話もしないと不公平になるかもしれないんで一応紙だけ用意しました。他意はないんですけどもいろんな切り口があって結論から言いますと分析支援とかシステム設計支援みたいなところは弱い。メインフレームの環境の上では弱い。どちらかというと下から上がってきているのが実態だろうと思います。その中でもあえて言えば，ＣＯＢＯＬ生成系を中心としたＣＡＤ・ＣＡＭ指向です。方法論そのものは変わらない。考え方は変わらないでただそれを機械化して早く綺麗にするというようなアプローチで、いろんなツールがあってそのところの整備を終わったというのが今の状況じゃないか。それはここですが、もうちょっと言語レベルを上げようというと今申し上げたようなデータフローみたいなものとか、もうちょっとインクリメンタルなプログラミングができるようなものとかいくつか出てきている。それからあと４ＧＬという環境ではＥＤＰ部門用ということでＬＩＮＣのお話をしましたが、もっとエンドユーザー指向の、かなりイージーな手法ですけども、そういったものではどんなものがあるかというと、ここに挙げたようなデータ

ベースに対してアクセスしていろんなレポートを出すというような環境がやっと今整いつつあると思います。ですからこの辺（分析支援工程）をちゃんとサポートしないとＣＡＳＥじゃないといわれると、そういう意味では今のメインフレームの上ではそういうＣＡＳＥはない。

最後にまとめます。主張に書きましたけれども、やはりＣＡＳＥツールの活用においては単に自動化ツールみたいなものが先にあるんではなくて、何を自動化するかという方法論みたいなものがちゃんとなければいけない。これはよくある絵です。プロセスモデルがあって、手法があって、自動化があって、これが逆三角形になっているのがＵＮＩＸの例だといわれてますけども、こういったモデルというのがあってその方法論とそれをどうするというこちらの方をちゃんとやっていく必要があると思います。そういう意味でこちらの上の方（自動化）というのはある程度表面的な問題のような気がしています。もっと内面の所をどう捉えるかというところをちゃんと考えなきゃいけない。それから言語レベルの話も、ＣＡＳＥと言えるかどうか分かりませんけども、あえて言わせてもらえばやっぱりいまのＣＯＢＯＬ環境よりはもうちょっと高級化された言語の方が絶対有効だ。ただし条件があって、これはやはり問題範囲を限定しなければいけないだろう。つまりあるアプリケーションを作るときはこれでいいよ。あるアプリケーションを作るときはこっちの方がいいよ。それを使い分ければいいわけで、先程説明したＰＡＳＥというのが完璧かというとそうじゃないが、ただ我々の環境で今実際にＣＯＢＯＬで作るよりはよっぽど早い。そういったものはおそらく他の業務でもいっぱいあると思うんです。そういう意味でいつまでもＣＯＢＯＬの時代ではないと思うんです。補足をすれば、データフロー的なああいう言語をオブジェクト指向とこじつければ言えないこともない。あるプロセスにデータを渡してあげると考えればいいわけで、プロセス間がどう動くかということを従来のプログラマーはプログラミング構造にどうマッピングするかということでいろんなことを考えてたんですが、その辺の所は先程のＰＡＳＥみたいなものを使えばプログラム構造を意識しなくて実現できるということです。

それからいわゆるビジネス分析をして、設計をして、プログラム設計にいくというギャップを埋めるような模範解答みたいなものはあるかというとない。今はどういう状態かというと、こういう横に割った各工程の支援ツールがあって、それを寄せ集めてトータルとしてただ名前としてなんとか一貫支援システムといってるだけで、こうスッとつながってない。そのためのキーは、いかにモデル化するかというのと、そのモデルを記述するために抽象度をいかに上げた仕様記述言語を実用化するか。そういう意味で今お話したＬＩＮＣには、ＬＩＮＣ自身のモデルがあって、そのための設計言語としてＬＤＬがある。プログラミング言語か仕様記述言語かというのはＷｈａｔとＨｏｗと言われますが実はその境界は無いわけで、どれだけ非手続的になるかということでＷｈａｔに近づくんですけども、ＬＩＮＣより仕様記述言語に近くなってきていると思います。ＬＩＮＣはそこまでのアプローチをとっているんじゃないかなと思います。

最後にこれはＢａｌｚｅｒさんが２・３年前のシンポジウムで１９９０年代のＰａｒａｄｉｇｍということでいわれたものです。この辺でグルグル回って、この辺でオートメーション化されてあとは自動的にソースまでできてこう動く。こういう環境でグルグル回りなさいよ。ここのところのフォーマルスペシフィケーション、このプロトタイピングというのを４ＧＬでやればいいんじゃないかなというのが僕の考え方です。ここでメンテナンスをしてけばいい。できたものはお昼を食べていればこっちまでくるという環境です。実はそれとまるっきり同じことをもう一人Ｊａｍｅｓ　Ｍａｒｔｉｎさんは言ってるんでそれをご紹介して終わりにしたいと思います。これは**Automation of Automation**ということでいってるんですが、アナリストはシステムをスケッチングしていく。これは絵を使ったり、図を使ったり、言語表現したりするわけですが、これは利用者による検証をこのフェーズでどんどんどんどんやっていく。この結果が１つの機能モデルとしてアウトプットされる。この機能モデルに対して（これは、ある程度フォーマライズされていますから、１つのソフトウェアが中間的に１つの言語に吐き出しをして、ここからいきなりジェネレーターにいけば最高なんですがちょっと難しいんで）、ここからシステム設計仕様みたいなものに落として、あとジェネレーターが稼働システムを作ってくれる。ここで実際にテストとして動かしてみる。だめならまたここにフィードバックをするというアプローチです。これをいかに早くするかってことがシステムをいかに早く作れるか。これはとりもなおさずインクリメンタルなエボリューショナルなプロトタイピングアプローチになるわけで、そのポイントはここのところをどうモデル化するかということと、この辺のところにあると思うんです。それはあまり欲張

らないである程度限定をしてやれば実用化されていくんじゃないか。これによってコミュニケーションギャップと騒がれているところはエンドユーザーがここに入ることで軽減されると思います。それからシステムアナリストとかデザイナーは、ある意味でフラストレーションがあるんです。自分達はこうだということでやるんですけども、実際に引き渡したあとはプログラマーとかコーダーの世界になっちゃって、意図したところと違う結果になる。今度はその人達が実際に機械の前でデザインをしたり定義をしたりするとそのまま動いてくるということで、むしろこの辺の所が活躍の場になって、ある意味では言われた通りにやればいいというプログラマーは失業するという恐ろしい時代がくるのではないかというのが結論であります。

久保： 最後に主張のちょっと前にメーカーというのを出したでしょう。分析・設計が弱いですねという言い方がなんとなく良いものを期待しているという言い方ですね。

佐藤： プログラム生成に比べてちょっと影が薄いということです。

久保： 僕が言いたいのは当たり前だということです。要するにメーカーから良いものが出てくることを期待しちゃいけないということです。僕が言いたいのは佐藤さん自らが問題だと言いたいんです。ＳＥＡを作って3年もやってなんにもないというのはなんだというこういう反省の方が僕は正しいだろうと思う。メーカーに待ってたってなんにも出てこない。僕メーカーの中にいるから分かるんですが、本当に期待しないほうがいいと思う。

2.3 ＳｔＰの利用経験と評価

桜井（ＳＲＡ）： 今日のセッションはＣＡＳＥツールということで、ＣＡＳＥツールはどの程度有用か、今のＣＡＳＥツールはどう使えばいいかという論点で話をして下さいという風に一応言われたんですけども、その前にSoftware through Pictures(略称ＳｔＰ)の利用経験と評価というテーマで私の発表をさせて頂きます。

発表の概要ですけども、ＳｔＰとは何か、何ができるかというのを簡単にお話して、それからＳｔＰの利用経験と評価、それからＳｔＰを通して感じた今のＣＡＳＥツールの問題点というのをお話したいと思います。ＳｔＰとは何かという話は初日にプレゼンテーションで5分くらい頂きましたし、初日からあそこでずっとデモさせていただいてますので大体ご覧になった方は分かると思いますけども、ご覧になってない方もいらっしゃるのでまた説明します。構造化分析・構造化設計・プロトタイピングを支援するＣＡＳＥツールです。構造化分析・設計・プロトタイピングというものが何かという話は先程長野県協同電算の佐藤さんが説明して下さいましたのでここでは割愛します。ＳＲＡで開発したものではなくてアメリカのInteractive Development Environmentsという会社で作ったものです。それをＳＲＡが日本語化して売っています。

これ†は初日にもお見せしたんですけどもＳｔＰを使う人というのはソフトウェアエンジニアとあとはエンドユーザーと一応書いてありますけども、こっち(ソフトウェアエンジニア)に力を入れている。結局システム仕様書を出すということです。それからこちらソースコードと書いてありますけども全部出るわけではありませんで、部分的にソースコードが出る。それをコンパイルすることもできる。あとはプロトタイピングを支援する環境もありますのでプロトタイプコマンドを入力してこういくということです。ＳｔＰを使って下へとやって丸をクリックしますとこういう図面‡ が出まして、これはＳｔＰを説明したものなんですけども、システムを開発する部分とコンフィギュレーション管理の部分とプロトタイピング部分と仕様書を出版する部分というのが分れています。要するにコマンドを与えてダイヤグラムとかを出すような形になっています。結局できることというのは私のペーパーに一通り書いてありますので読んで頂きたいんですけども、データフロー・ダイヤグラム(略してＤＦＤ)，状態遷移図，構造化チャートといったものが書けまして、ＤＦＤに関していいますと、ＤＦＤを編集して、プロセススペックを編集して、編集したダイヤグラムからデータ辞書を生成して下さいというと辞書が生成されて、ダイヤグラムの詳細間で矛盾がないかどうかチェックする機能と、構文のチェックというのができます。データ定義の方なんですけども、データフロー・ダイヤグラムエディターを用いましてＤＦＤを編集します。それでデータ定義は出ます。データ要素、これは構造を編集してそれから要素を編集する。そういったダイヤグラムからＪａｃｋｓｏｎデータ定義に翻訳する。あとデータ辞書を生成できるわけです。ダイヤグラムを書いてそこからデ

† StPのコンテキスト・ダイヤグラム

ータ辞書を生成するというような形になります。これも似たようなものですけどもＥ－Ｒダイヤグラムも同じようにできます。こちらはそのデータ宣言の部分ですけども、ｐａｓｃａｌとかａｄａとかｃとかそういったもののデータ宣言を生成することができます。こちらはコンフィギュレーション管理なんですけども、ダイヤグラムのアクセス権ですとかバージョン管理、それから新しくシステムを開発するときにはその初期化ですとか、ツールを利用する環境ですね、プリンターを何を使うとか、テキストエディターは何だとか、ドキュメント作成には何を使うかとかいろんなパラメータがあるんですけども、そういったものを編集して変えてやって利用者の環境に合わせて使うことができるといったようなことができます。こちらのプロトタイピングなんですけども、基本的には遷移ダイヤグラムを書いて、ＲＡＰＩＤ　ＵＳＥコードというものを出します。それをコンパイルして、必要ならば、データベースを使ってプロトタイプが実行できる。そういう環境があります。これはシステムの仕様書を出版するというところなんですけども、いろんなエディターを使いまして、データフローエディターとかストラクチャーダイヤグラムエディターとかＥ－Ｒダイヤグラムエディターとか状態遷移図エディターとかいろんなのがありますけども、そういったもので書いたダイヤグラムを全部印刷することができます。データ辞書も当然印刷することができます。この図自体ＳｔＰを使って書いたものなんですけども、例えばこう書いているときに、この見方をちょっと説明しますとこれがデータフローでこれがデータストアなんですけども、このデータストアをクリックしてこれをもっと詳細に定義したいということになりますと、こういう風にまいります。プロジェクト・データベースというのは例えばいくつかのリレーションに分れていてとかいうようなことを定義する。その１つのリレーションを見てみるとそれはさらにこういう風になっているとかそういう感じできています。ＳｔＰでは図の名前とかロックとか履歴とかそういったものを管理できるようになっています。というツールなんですけども実際にそこで動いていますので今日のセッション終わった頃までずっと動かしていますから興味のある方は見て下さい。

さて、利用経験と評価に移ります。利用期間は今年１９８８年の４月から現在まで約８ヵ月で、利用目的は、日本語版のＳｔＰを作るということでそのサポートおよび販売促進の為です。評価対象なんですけども、Ｘ－ｗｉｎｄｏｗ対応のＳｔＰ　ｖｅｒｓｉｏｎ３．０というものがあって、それをＳＲＡで日本語化したものを評価対象としました。もともとＳＵＮの上で動いていましたんでＳＵＮｖｉｅｗだったんですけども、それをＩＤＥ社がＸ－ｗｉｎｄｏｗ対応にしたものをさらにＳＲＡで日本語化したもの。動作環境は、ＮＥＷＳで８４１、メモリ１６ＭＢでスワップ３２ＭＢであとは周辺機器がいろいろ付いているといった環境の中で利用しました。

評価なんですけども、一般に評価といいますとどうしても定量的な評価を期待されるんですが、定量的な評価というのはできません。それは難しいんです。結局何のために使うのかっていうと、生産性を上げるとか信頼性を上げるとかそういった話になるわけなんですが、それの評価基準自体が定まっていませんから、結局定量的な評価はできない。結局お話できるのは自分で使った感想と、今までにずっとやってきましたデモンストレーションおよびプレゼンテーションの感想、それとセミナーの講師の補助というのをやっていまして、ＳＲＡでは毎月１回ＳｔＰのセミナー、構造化分析とは何かという話からＳｔＰを使った実習まで２日間コースでやってるんですけども、そういうときの実習のいろんな反応から感じたことを述べたいと思います。

ＳｔＰでは先程説明しましたように、構造化分析・構造化設計・プロトタイピングを支援しています。けれどもこれらの手法というのはまだ日本では浸透していません。アメリカではもうソフトウェア開発の上流工程の手法としてはもう構造化分析・構造化設計というのがかなり主流になってきまして、当たり前のようになっているらしいんですけども、それでＣＡＳＥツールというのもいっぱい出てきています。

ところがまだ日本ではこういった方法自体が浸透していないので、特に構造化分析を支援するＣＡＳＥツールといった場合にもなかなか難しい。説明するのはとても大変で、手法を身に付けるのはもっと大変なんです。結局システムのモデルを作るために使うような形になるわけなんですけども、それを図式化するのは非常に難しいわけです。ですからいきなりツールを渡されて使って下さいと言われても、結局何を書いて良いか分からないという状況になってしまう。これはもちろんＳｔＰが悪いわけではありません。デモしていると、どうしても要求分析・設計といういわゆる上流工程から下流工程のプロ

‡ StPのDFDで、レベル0のもの。

グラミングへのつながりを期待する人が非常に多いんです。こういうことを期待されても、これはちょっと無理というか、もちろん全部できればいいんですけども、全てを支援するようにといわれてもちょっと無理だということです。私の感想としては、こういう上流から下流へという流れよりも、構造化分析のアイデアとしては論理モデルを作ってから物理モデルを作るというような感じでいわれているんですけども、そのモデル間の移行がというのが非常に問題だと思っています。要するにモデルを作るのが難しいという話なんですけども、こっちの方が問題で、モデル化できちゃえばこっちはそんなに難しくないなという気がしてますので、この辺の所が難しいというのを理解して欲しいと思います。それがもちろん支援できるようなものがあればいいんですけども。ＳｔＰではダイヤグラムのシンタックスチェックというのをやってくれます。基本的な機能はいいんですけども、操作性にちょっと難点があるかなということです。これは強調しておきたいんですけども、ＳｔＰでは他のツールと組み合わせて使うということを前提としていますので、使いこなすにはそれなりの環境と知識が必要だ。それが導入を難しくさせているなという気がします。

ＳｔＰを通して感じた今のＣＡＳＥツールの問題点というのを発表します。ＣＡＳＥツールというのは、今日の午前中のセッションをみんな聞いてもうお分かりだと思いますけども、ソフトウェア工学の成果をコンピュータで支援しようというものなので、結局なんでもありなんです。そこでどうしても話が発散してしまって、みんなそれぞれ描いているイメージというか、ＣＡＳＥツールに期待しているもの、要するに迷信というかそういったものが随分あって、現状でできる範囲でけっこう良いＣＡＳＥツールであっても、期待するものがあまりに大きいとか、そういった問題でなかなか受け入れてもらえないような感じがあるんじゃないか、あるいはあまりうまく利用しきれてないものがあるんじゃないかという気がします。それから先程もいいましたけれども、手法が浸透していない。ＣＡＳＥツールが出てきた背景というのはワークステーションが安く手に入るようになってきたというのがあるんですけども、そのワークステーションも実際にはそんなに浸透していない。それでなかなか難しい部分があるんですけども、これはこれからに期待できるなと思っています。あと、これが一番問題だと思うんですけども、ＣＡＳＥツールは結局方法論を知らないとどうしようもない。一般のＳＥが利用できるようにするためには、ベテランのノウハウを利用する仕掛けとかやり方をガイダンスする仕掛けが必要かもしれないと一時期思っていたんですけども、今描かれたダイヤグラムというのは再利用できます。けれども図を書く過程の考え方とかいったものはトレースできない。こういうものをなんとかできないかとずっと思ってたんですけども、これはちょっと難しいから結局永遠の課題かなという感じになっています。東工大の篠田さんとかずっとこの辺で悩んでるんじゃないかなと思いますけども、これはＣＡＳＥツールがサポートするものではないのかもしれません。ただしこういうところがうまく解決されていかないと使えないという気がしています。

現状では結局どうすればいいかというと、興味のある人を中心にセミナーなどを開いて、利用者の和を広げるのが最善の策だろうという風に考えています。

結局ＳｔＰは、私はＳｔＰを作った人じゃないので分かりませんが、おそらく目指しているのは汎用のＣＡＳＥツールだろうと思っています。つまり、先程佐藤さんが言われていましたけれども、メインフレームの事務アプリケーションとかそういったターゲットを絞り込めば、それなりの使い勝手の良いものとか便利なものとかできると思うんですけども、結局今までの状況を見てまして、要するにソフトウェアがハードウェアの進歩に追い付いていないという話を皆さん再三やられているわけで、そうするといちいちツールを作ったんじゃ間に合わないから、なるべく汎用のツール、もうこれ１個あればいいみたいなものはないだろうかと、結局こうやって摸索していったんだと思いますけども、それをやるためには結局は、どのようなアプリケーションを作るにしても、それからどんな言語で、どんなＯＳで、どんなマシンの上で作るにしても、なんか核になる部分があるんじゃないか。要するに開発するための手法とか、どんなアプリケーションにしても何か核になるものがあるにちがいないと信じてそれを提供する。アプリケーション分野に特異な部分とか、ターゲットの言語の特異な部分というのは既存のツールがあるわけで、そういったものとうまく組み合わせてやれるようなそういった仕掛けを提供するしかできないんじゃなかろうか。そういったことで、ＳｔＰもそのための仕掛けで現状の最善の、というかできる範囲のことは一通りやっているなという感じのＣＡＳＥです。これで一応終わらせて頂きます。

3. 討論

久保： 桜井さんの意見に同調して、佐藤さんに盾突くようにしてみようと思います。桜井さんの中で、どんなＣＡＳＥツールがあっても手法を身に付けなければ役に立たないという言い方をされたと思うんですけども、それは正しいと思うんですけども、僕なんかメーカーの中にいてお客さんを見て思うのは、えらくハウツウを欲しがります。そして即効薬を欲しがります。それはたぶん僕は間違っていると思うんですけども、そういうマーケットの要求に対してこびを売っているのがＬＩＮＣなんかじゃないかという話なんです。ＬＩＮＣというのは非常に成功を納めた第４世代言語だという理解はしてまして、富士通なんかにもＬＩＮＣみたいなものが無いんじゃないか、作れという声も聞こえてはくるんですが、その割には実際には興味を与えるほどの力にはなっていない。勝ち負けというのはいろんな要素で決まってきますけど、今のソフトウェアの生産性にニーズの強さからいうともうちょっとマーケットの競争力の武器になっていいはずなんですけど、実際にはＬＩＮＣを持っているにもかかわらずそれほどの驚異にはなっていない。せいぜい善戦をしているというような感じなんです。それはやはりＬＩＮＣそのものに限界があるんではなかろうかというのが僕の１つの見方です。ＬＩＮＣなんかの、もうちょっと一般化して第４世代言語一般に広げてもいいんですけども、それを導くときの指導原理になっているのが、アプリケーションのドメインかなんかをある考え方でもって狭くする。狭くすることによって非常にその領域ではすごく良いツールにするという原理が働いていると思いますけど、それが間違っている、ドメインを切るというのは正しいが、ドメインの切り方がたぶん違うんじゃないかなと思います。ですから例えば川鉄というお客さんがいらっしゃいますよね、富士通のお客でもあるし、それからユニシスのお客でもあるんですけど、ＬＩＮＣを非常に憧れているお客さんです。ところが全部ＬＩＮＣでいくかというと、とらないというんです。それは要するに不安なんですよ。狭くしていることの不安じゃないだろうか。それは狭くすることによって情報システムの発展を阻害している。ドメインの切り方がお客のニーズに合ってないんじゃないかというのが僕の見方です。要するに第４世代言語的なアプローチからソフトウェア開発の現場の主流になるためには、やはりオールマイティーになっていかなきゃだめじゃなかろうか。さっきのイボリューションを支援するというのはセールストークでは使えるんですが、たぶんお客はそこまで納得してないじゃなかろうかという感じが僕にはあります。それと佐藤さんの主張の４つについての私の意見を申し上げますが、１番目の方法論の習得が必要であるというのは賛成です。こらが無いとやっぱりダメだと思うんです。ツールを非常に効果があるよと売るよりも、その前にソフトウェアの開発の現場で方法論を身に付けるような風潮を作っていく。あるいはそういう意欲を失った人はこの業界から足を洗ってもらうとか、そういうことが必要じゃなかろうかというのが僕の意見です。

２番目は問題領域を限定し言語の高級化が有効というのは賛成ですけども、今の４ＧＬなんかはちょっと高級化の方向を間違っているんじゃなかろうかという見方です。高級化しない限りは生産性が上がらない。佐原さんが紹介されたのもそうですね。高級化によって生産性を上げている。第４世代言語というのが本当に正しいだろうかという疑問を持っているというのが２番目に対する意見です。

３番目のシステム分析・システム設計・プログラム設計間のギャップを埋める模範解答がまだない。キーはモデル化手法と抽象度を上げた仕様記述言語の実用化にあるのではないかというここまでは特に異論はないんですけども、やっぱり１つだけつけ加えたいなと思うのは、要するにアプリケーションドメインのプロフェッショナルを仲間に入れるなり、我々の中にそういう人を作っていくか、やっぱりアプリケーションドメインの知識なしでは絶対うまくいかないだろうという風な意見です。ですからこのソフトウェア技術者協会も残念ながらそういう人っていうのはあまり会員にもないですね。ですからもちょっとアプリケーションの業務の世界を知っている人達との交流して、うまく会話を成立させるような共通の言葉を作っていくというような努力がけっこう大事じゃなかろうかという気がします。

４番目はですからツールが欲しいというのはありますけども、さっきも言いましたけどもツールは自分で作っていくしかないんじゃないかと思います。要するにどれか良いと思えばそれを一生懸命育てようじゃないか。たまたま昨日セミナーを受けましたけども、ああいうのは良ければどんどん育てていくというそういう努力が我々に必要じゃないかというコメントです。

佐藤： 最初の方でマーケットが広がってないからたぶんそんなにたいしたことはないだろうとおっしゃられた

んで、答えはたぶん歴史が証明するんじゃないかと思います。というのはまだ出たところなんで、これが叩かれてこれがかなり叩かれた結果沈んでしまうか、あるいは鍛えられてもっとユーザーに広がるか、あと5年くらいすれば分かるだろう。最初からそういうアプローチをとってなくて、それが成功したときにじゃあいこうって言っても差が出るんで、そういった方法自体をたぶん否定されていないとは思いますけども、僕はそこに賭けたいなという気がしています。それからドメインの切り方が違うというのは、言われてみると確かにそうなんですね。ただどのアプリケーションでＬＩＮＣがどこまで限界があるかっていうのはまだそれなりの経験がないんでまったくお答えできないんですけども、誰もが考えることですが、狭くすれば当然それだけ抽象度が上がって有効になるわけです。ただ問題は狭くして問題だというビジネスエリアがどれだけあって、狭くても良いからそれで適用しようという人はそれでも良いような気がします。

確かに理想を言えばオールマイティーの方なんですが、ＣＯＢＯＬは１０年・２０年動いているわけで、あれはオールマイティーだから確かにそうなんでしょうけども、本当はＬＩＮＣみたいなああいう切り方でなくても良いと思うんですが、もっと大きなものを欲しているんですけれど、無いのが現状でしょう。それから最後は僕１つ反対なんですが、アプリケーション知識を持っている人達を育てていく。エンジニアがアプリケーションの知識を持つべきだ。それでユーザーとうまく言語かなんかで話そうということなんですが、たぶんそれでは解決しない。というのは、ソフトウェア技術者が何十万人足りないとか言ってますが、アプリケーションのことを、自分たちの業務の事を、一番良く知っているのはビジネスマンです。ですからその人達の頭の中にある要求をいかに引っ張りだすかというようなアプローチをエンジニアがとっていくべきではないか、そういう意味で対じしてやるんじゃなくて、いっしょにやるということが大切だと思うんで、そのへんのツールが今までなかったのが現状ではないかと思います。

事務処理の１つ１つはそれほど高級なことはやっていないのだから、ビジネスアプリケーションはビジネスマンが主体となって構築できる環境を、インテグレータであるＥＤＰ人間が構築すれば良いという気がします。

久保：　川鉄のシステム部の人から聞いたことをご紹介しますと、要は大事なアプチケーションには使わないという言い方をするんです。例えばそれは大きなものには使わない。基幹業務には使わない。だけど欲しい。それでもちょっと大きなもの、基幹的なものにつかえるようにＬＩＮＣを良くしてもらいたいという圧力を当然川鉄さんかけてると思いますけども、少なくとも僕の聞いた段階ではそういう言い方でした。というご紹介です。

それから今の２番目の論点、ユーザーと技術者との関係です。ちゃんと精密な議論をするためにはもうちょっと準備がいりますかと思いますけども、やっぱりソフトウェアの開発というのは技術屋だけではできないわけです。どうやってインボルブさせていくか、ちょうど協同作業にしていかなければいけないだろうということです、ここは。

佐原（ＳＲＡ）：　今の久保さんの方のおっしゃった、基幹業務とかに第４世代言語とかを使わないというのは多分ハードウェアを第４世代言語に合うように拡張して本当のハードウェアの環境の中で使わないから、怖くて使えないんです。要はアセンブラよりもＣＯＢＯＬの方がいろいろリソース使うわけです。ＣＯＢＯＬよりも第４世代言語の方がリソース使うわけで、要は例えばアセンブラがかろうじて動くような環境でＣＯＢＯＬを動かそうとすればやっぱり効率の問題なんかが出てきて、あぶなくてつかえないというのが、昔たぶんアセンブラとＣＯＢＯＬの間でそういう風な話があって、それも同じことでハードウェアを増強すれば、別にＬＩＮＣが良いかどうか分かりませんが、まともな第４世代言語であればちゃんと使えると思うんです。

それからもう１つエンドユーザーが一番業務を知っているというのは、日経コンピューターなんかでよく書いてありますが、おかしいんです。エンドユーザーと昔やっていたんですが、エンドユーザーが自分の業務をよく知らないというのが現実なんです。なぜかって言うと、自分が今までやってきたことは知ってますけども、自分の業務で何をやるべきかというのを決して知っているとは限らない。それは我々も知っているとは限らないわけで、結構いい勝負であって、エンドユーザーに聞けというのは、ある意味では間違いなわけです。例えば今証券会社の方で数学出身者をどんどん採っているというのは、実は証券会社の人間はそんなに数学的なアプローチをしてないから、全然気がつかない分野でそういう大きな市場とかアプリケーションの応用の形態があったわけです。だからエンドユーザーに聞けというのはどうも私は嘘だ

と思っています。エンドユーザーを全く無視していいというわけじゃないです。

久保： 僕は佐原さんに賛成です。でも使う人はエンドユーザーですから、巻き込まなきゃいけませんけども、知りません。やっぱり知らないと思ったほうがいい。ですからソフトウェアエンジニアリングのアプローチはある意味では間違っている。

新田（ＳＲＡ）： さっき桜井さんの発表の中にＣＡＳＥの定義みたいなものがあって、ＣＡＳＥツールとはソフトウェアエンジニアリングの結果出てきた何かを使うためのツールだという風におっしゃったんですけども、どうも私はそれは間違っているという気がするので、間違ってると思う私が間違っているかどうかということを確認したいのですが。というのは私の認識ではＣＡＳＥツールとうのはコンピューターを使ってソフトウェアエンジニアリングを行なうためのツールだと思うんです。今までの発表を聞いていますと、それはＣＡＳＥツールというよりもＣＡＳＤというか、つまりCompute Aided Software Designとか Computer Aided Software Developmentとかいうレベルのツールだと思うんです。そういうツールの言語レベルを高めるとか、あるいは統合化していってもＣＡＳＥツールとは全く違うものだと思うんです。それは支援ツールのレベルが低いといってるんではなくて、それはそれで役に立つんだけども、ＣＡＳＥは全く違うものだということを考えているんですけどもどうなんでしょうか。

桜井： 新田さんがおっしゃったのは私も確かにそうだと思うんですけども、一般的にＣＡＳＥツールというと、さっき私がいったようなものになるというか、そう言われているということです。新田さんがおっしゃったのは確かに当りだと思います。ただ、今は、コンピューターを使ってソフトウェア工学をやるといっても、そんなにうれしくない人もいっぱいいるわけです。まず今までやってきたような、どっちかというと今日私がお話したようなＣＡＳＥは、ＳＥのためのというか、コンピューターを使ってＳＥを楽にさせるとか、（ＳＥが何かという話は置いといて)、そういう感じのような捉え方をされてるのが一般的だと考えたので、それに対してちょっと言ってみたということです。あらためて、「一般的にＣＡＳＥツールといわれるとどうか」という話だとどうなんでしょう。とにかく、なんと呼ぶかは別として、新田さんがおっしゃったようなことも確かに必要だと思います。

新田： 私の主張がどういうことかというと、今まである支援ツールにただＣＡＳＥとう新しい名前を付けただけでみんなでチヤホヤしてるんじゃないかということを言いたかったんです。

桜井： それはその通りだと思っています。

久保： 今の新田さんのセリフ気に入ってます。ソフトウェアエンジニアリングを実践するためのものとしてあるという。さっきの繰り返しになりますけども、そういう意識の無い人は業界から出ていってもらったほうがいい。そういう感じです。

野中（ＮＥＣ）： 新田さんにちょっと質問したい。僕、ＣＡＳＥというとそういう支援ツールだと思ってたんですけども、新田さんのイメージされているＣＡＳＥツールというのはどんなものかお聞かせ願えますか。

新田： はっきりしたイメージを言うことはできないんですけども、我々が何か問題を与えられて、プログラムとかデザインをしようというときに、頭の中でミニソフトウェアエンジニアリングみたいなことを行なっていると思うんです。それはつまり、この問題を解決するにはどういう方法論が良いかと選んだり、自分で新しい方法論を作ったりとか頭の中でやって、その結果支援ツールを使ってものを作っていると思うんです。ですから、ＣＡＳＥというのはものを作るというよりも、そういう風にどの方法論が良いかとか研究したり、方法論自身を開発したりとかそういうことを支援するツールがＣＡＳＥツールであって欲しいなという気がします。

野中： ということは要するに一般大衆プログラマーのものではなくて、そういう方法を研究するためのサポートツールだという定義なんですけども。

佐原： ＣＡＳＥツールといった定義は新田さんのは今始めて聞いたんだけど、それが１つと。要はなんでもかんでもＣＯＢＯＬのジェネレーターから思うというのは、コンピューターを使ってソフトウェアの生産性向上に寄与するようなものはなんでもＣＡＳＥツールという立場と、それからさっき桜井さんが言ったちゃんとデータディクショナリーみたいな配下で動いていて、構造化分析

・設計のあたりを特に支援するのをＣＡＳＥツールという３つの定義がある。それぞれの話の中で今自分が話しているのはどの定義についてというのを言えば良いだけじゃないですか。

野中： 私は自分でツールを作っているわけじゃないですけど、ツールを作って売っている人がいますね。そういう人に大分昔に聞いた話なんですけども、自分達が作っているプロダクトに関してそういうツールを使われたんですかというと、このツールはビジネスアプリケーションをターゲットとしてあげてますので、我々がやっているような特殊なソフトウェアにはちょっと使ってませんというような言い方をされたことがあるんですけども、非常にそれはちょっと何かうさんくさいような話だと思うですけども。何か競馬の予想屋みたいな感じで、この券は絶対当たりますよっていうので、じゃあおまえずっと使えと言いたくなるようなところがあるんですけども。何かそういう感じがするんですね。だから一番聞きたいのはΣの方に聞きたいんですけども、そういうようなことをやっておられる方がいるのかどうか分からないんですけども、ツールを作るのに自分達で作ったツールを使っているかどうかというのがちょっと聞きたいです。今までいろんな話が出てパブリックドメインの中に良いソフトがあるって言うのは、自分で使いたいから自分で作るというので良いのができるという話があったと思うんですけど。例えばＳｔＰを作るのにＳｔＰを使っていたかどうか知っていたら教えて頂きたいです。

中田（Σ本部）： ９月１日にΣ本部に来た途端ですので、ちょっと正確に答えられるかどうか、魚津さんとかそういう方のほうが本当はよくご存知だと思うんですけど。Σの中でΣを使っているかということなんですけども、まずΣの本部というのと参加企業というのがございます。それで全体がΣコミュニティーというのを作っていると思うんですけども、参加企業というのは非常に多く参加して頂いていて、ここにお集まりの中でも数社さん以上参加して頂いていると思います。それでΣというのはまだモニター期間中ということで、技術分野と事務処理分野とそれぞれちょっとモニターのレベル違いますけども、本当にΣというのが有効なのかどうかを実践的に使って評価してみようということです。本当にΣを使っているかどうかという局面だけ捉えていいますと、Σツール開発メーカーさんで非常に多くモニターに参加して頂いている。どういう意味合いで言いますと、実際に使ってみて本当に自分達が作っているツールが有効かどうかというのを、少なくとも評価しようとして下さっている参加企業がたくさんあるという状態です。６５年３月末に一応Σプロジェクト終了しますから、それから以降本当に本番利用になったときに、そういうメーカーさんあるいはΣ開発に参加されなかった人達が本当に使い続けるかどうかということを、歴史が証明するというお話が先程ありましたけども、そうなっていくのを我々が期待しているという状態だということをご理解頂ければどうかと思います。

桜井： ＳｔＰを使ってＳｔＰを作ったかという話ですが、それは作っていません。たぶんね。デモンストレーションの方でＳｔＰの構造というか、ＳｔＰを説明したダイヤグラムをお見せしましたが、あれは終わりの方までは書いてないです。いつ書いたものかというのはちょっと分かりませんけども、ＩＤＥ社がＳｔＰのパッケージの中で一応例として付けてきたものなので、あっちが書いたものには違いないんですけども、いつ作ったものなのかというのはちょっと分かりません。だからＳｔＰを作る際に本当にああいうのを書いたかどうかというのは正確には分かりません。ただし、製品の一部である遷移図エディタを使って作ったアプリケーションをサンプル的に製品に含めるということはしています。私は以前にもやはり開発支援ツールというのを作ってサポートしていたんですけども、結局つくって使わないのはなぜかというのは、単に自分達が欲しいものを作ってるわけじゃないからそうなっちゃうわけです。頼まれて作ってるから、要するにツールメーカーですからそうなっちゃう。それはハードウェアにしてもそうでしょうけども。もちろん自分達で作ったもので使っているものも当然あります。いっぱいありますけども、それは依頼されて作ったわけじゃないです。それでお金もらっているわけじゃなくて、お金をもらっているのは、そういう依頼されたものですから、それは使いたくて作ったものとは限らないし、うさんくさいと言われましたけども、そういう話になってしまうのはいたしかたないと思っています。

佐藤： ＬＩＮＣがＬＩＮＣで作ってるかというと、ＹＥＳです。ですからそれは説得力がある。ＬＩＮＣを作った人はＬＩＮＣを使いたくて作ったんで、みんながたまたま賛同して使ったということです。

桑名（ＮＴＴソフト研）：　今なぜツールが使われないかという話で、実は通研内でいろんなツールとかを実験的に作っていて、いろんなところでデモンストレーションなんかをやってきた場合もあるんですけども、実際それを現場に持っていくと、現場からの意見は通研の作ったものは使い物にならないというようなこともたまに聞くんです。どうしてかなといろいろ考えてみて、いろいろ意見を聞いてみると、これはどこでも当てはまると思うんですけども、実はソフトを、交換機のソフトでもなんでもいいんですけども、莫大なソフトウェアを持っている。実はそれを新しく作るのかというとそうではなくて、その元の大きなソフトウェアのある一部の機能追加をしてどんどんどんどん作っていく。そういうところに新しいツールを持ってきて使ってくれなんていっても誰も使ってくれないわけです。だから今既存のその大きなソフトウェアの機能追加をして使っていくしかないんです。現場は今までやってきた手法とか作り方にすごく慣れているわけです。それのほうがずっと速いから、信頼率も高いだろうということで作るんですが、今新しく手法とかツール出てきてますから、これはもう地道に新しいプロジェクトとかそういうところに展開していくしかないんじゃないかという風な気がしてます。

黒坂（ＪＩＰ）：　私はどちらかというとユーザーです。アプリケーションを作っている側です。それも非常に特殊で土木橋梁部門、早く言えば瀬戸大橋を作ったりとか、高速道路を作ったりというアプリケーションを作ってる業務に属します。最初はそういう意味ではなくて、データベースとか作ったりいろいろしてたんですけども、所属がそこになったもので、そこに入ってやっているわけですけども、全然専門部門は違ったんですけども、土木の部分もすごく進んでるわけなんです。ハードの方もすごく進んでるんですけども、ハードのことをもし一生懸命やったとしたら、ソフトウェア開発とか環境とかそういう方に一生懸命やったら、土木の知識についていかれないわけなんです。土木の知識のある方というのは、すごくそういう方面はあるんだけども、コンピューターの上に載せるという手法がまるで無いわけです。というかすごく少ないわけなんです。そういう人達にいろんなニーズがたくさんあって、それをどのような形で引っ張ってきてコンピュータに載せるかというのが私の仕事なんですけども、如何せん土木の知識がまるで無いと、絶対やっていかれないわけなんです。

彼らと同じように微分方程式が解けないと、そのプログラムが組めないわけなんです。そうするとＡＬＧＯＬとかいうみなさんの知らない言語を使ってるわけなんですけど、別にＡＬＧＯＬを使いたいから使っているわけじゃなくて、ＦＯＲＴＲＡＮでもＣでも何でもいいわけで、私にとってはどうでもいいわけなんです。事務計算の処理とはまるで違って、１００％のプログラミングのうちの１０％か２０％が入出力部分なんです。それは誰にでも以外と分かりやすくて、８０％もしくは９０％の全てが解析部分なんです。それはほとんどの方が分からないわけなんです。そこに投入すると、ほとんど入出力のインターフェイスの部分がおろそかになっちゃって、いまだにカードリーダーで入力するようなイメージがあるわけなんです。

私はすごくそこが嫌で、お客さんがそれで満足するわけなんです。なぜかと言うと、データの入力数はすごく少ないわけですし、彼らにとって１つずつ座標点を書くというのは別にどうでもいいわけなんです。結果もなんというかラインプリンターに１３２で打っていって、帳票みたいにいい形で出てこなくても満足なんです。なぜかと言うと、その計算ルーチンがすごく重要なわけでしてて入出力の結果の見栄えなんてどうでもいいという考え方なんですけども、だんだんだんだんＳＯＳが出てきているというのが、だんだん欲望が高まってきて、ＣＡＤで見たいとか、入力は画面でやりたいとか、そういうのが出てるんです。そこにはまたすごくハンディキャップがありまして、彼らは簡単にできると思うわなんです。入力をトントントンと入れて、マウスでピッとやればできるんでしょと言うけれども、如何せん私はＣは全然知りませんから、できますとしか言えないんですけど、現実的にはやっぱりその１ステップを踏まなくちゃいけないし、かなり難しいわけなんです。

だからここに来たらポッポッポッと入力を入れると、Ｃの言語がバーと発生させられて、何か良いものができるんじゃないかな、そうすると中身はＦＯＲＴＲＡＮであろうとＡＬＧＯＬであろうと、メインマシンであろうとワークステーションであろうと、何でも私はすみっこでやっておけばひょっとして助かるんじゃないかと思ってすごく期待したわけなんです。そういうのはきっとみんなが教えてくれるんじゃないかなと思ってずーっと待ってるんですけで、出ないんです。次回また出たら、こうやったらいいんですよ黒坂さんという方がいらっしゃるかもしれないんです。ＬＩＮＣというのはあそこのユ

ニシスのを見せて頂いたんですけども、かなり近寄ったとして、歩み寄ったとして私が随分がんばったとしても、やはり技術計算部門では多分難しいと思うんです。

それをオールマイティーにやるっていうのはすごく難しいと思うんです。そのあたりで何か良い解決法というか、部署に帰るとみんなに怒るんですよ。きっと今度は私が言うわけです。ＳＥＡという良いショップに出たから、君達のやり方は非常にまずい、そんな力づくでコーディングしてるなんて今時遅いと言うわけなんですけども、こちらに来たらみなさんどうしたら良いと思いますって聞くわけです。そういうのすごくあって、もちろんある方向性とか方針とか思想とかいうのがないで、むやみやたらにこんなのできましたから使ってみて下さいってどんどん使っていくことは非常に危険なことなわけです。ずっと動いているもんですから、それやっぱりやめました、そのやり方ちょっと思想が間違ってましたからやめましょうといったら、瀬戸大橋は落っこっちゃうわけなんです。その部分で確かなものはあんまり必要じゃないんですけども、なんとか前に進まなくちゃいけないというのがくあるんです。

それで見せて頂いたいろんなツールとかは私はどんどん使っていきたいと思うんです。使ってだめだったらば、改良できるものだったら改良してもらいたいし、メーカーにあんまり期待できないんですけども、メーカーのほうもちょっと落ち着いていない。こちらのほうの方にもちょっと歩み寄って頂く。それが一番いい方法かもしれないけども、アプリケーションを作る側には非常に不便でやりにくい。そういう思想とか方向性というのを作る人、私みたいなのは身に付けなきゃいけないだろうけど、本当の業務をやっているユーザーの人には絶対にそういうことは言えないんです。そういうところがすごくネックだと思いますけども、そういうとこにあんまり現場の人がいないので、きっとそんなことで悩んでいる人がいないんさだうなと思って、現場の人はこんなことを言ってて、ユーザーはコンピューター使えって言ったら、かなりそういうことですごく悩むと思うんです。

山口（電力計算センター）：　実は私全く同じ状況にいまして、ただ私の場合はプログラムを作ればなんとかなるかなというのことで、半分帰ったらやろうと思ってたとこなんですけど、同じ状況だというのはやはり、プログラムを作ってくれと言うのは、電力研究所の研究者の方なんです。それこういう計算式使ってこういうプログラム作ってくれと言われます。結局解析部分が多くなって、入出力部分は小さくなる。それで同じ様に入力はただ単に８０カラムの数値の列で、出力のほうはただ単なるベタ打ちであったんですが、最近本当に同じ様にやはりグラフィックで描いたりとかそういったような状況になってます。私も多少ＭＳ－ＤＯＳとかパソコン使ってましたから、まず無理だなとは思ってたんですが、多少は期待してた点もあります。そういった座標をとったりとか、あるいは図形を描かせるのがすごく大変なんです。例えばＦＯＲＴＲＡＮで図形を描かせようとすると、パッケージを持ってきて線を１本１本引いてやる。ＦＯＲＴＲＡＮでやらせなきゃいけない。それが非常に面倒くさい。それで何か良い方法がないかなというのはあります。

岸田（ＳＲＡ）：　別に全然問題でも何でもないと思います。要するに面倒くさい科学計算の解析部分はそれはブラックボックスなわけでしょ。それに対して入出力の部分をかませればいいだけの話で、真ん中の解析の部分がＦＯＲＴＲＡＮやＡＬＧＯＬで書いてあるからといって、入出力をＦＯＲＴＲＡＮやＡＬＧＯＬでやる必要な無いわけで、ユーザーインターフェイスマネジメントシステムというコンセプトがあるわけですから、それをはっきり切り離せばそれはすむ話です。アルゴリズムの解析の部分がいろいろと面倒くさいというのは分かります。私も現役の頃はそういうことばっかたわけですが、なにも自分で微分方程式を解く必要はなくて、たぶんそういう専門家の頭の中である程度整理されているわけですが、それをうまくライブラリー化して、それを管理することだけ我々は手伝ってあげればそれですむ話で、解析の難しさとソフトウェアの構造とごっちゃにするといけない。それをはっきり切り離せばすむ話だと思います。

深瀬（ＡＳＣＩＩ）：　解析をするシステムですと、例えばマキシマムというシステムがあるんです。それからレディースというパッケージもあります。それはライブラリーの集合ですから、たぶんそういうものを使うのが一番速いと思います。これは新しいワークステーションではライセンスされてて大抵動きます。それから入出力の話について言えば、例えば構造解析するナストランという巨大パッケージがありますけども、あれもインプット・アウトプットはサポートするパッケージソフトウェアがあるんです。そういうものをうまく利用する。ある

いは例えばＵＮＩＸでいえば、ＰＬＯＴというコマンドがありますし、ＧＲＡＰＨというコマンドもあるんです。それは旧来のテクトロの端末とか想定してますけども、データ列を与えればグラフになってしまうというようなツールがあります。

それから最近出てきてるアプリケーションパッケージですと、たとえばPostscriptのコマンド列を吐いてしまうとか、どんなプリンターでもどんなワークステーションでも出せるようなアウトプットのフォーマットを持ってるわけです。例えば私の会社のＣＡＮＤＹというパッケージソフトウェアはPostscriptを吐くわけです。最初から設計として入力と出力をうまい具合な一般的なインターフェイスで書いてしまうというようなことをやられてれば、それですんでしまうわけです。デバイス・インディペンデントな入出力の約束事にしてしまうということがいいんじゃないかと思います。

野村: 時間になりましたので、まだご意見があるかと思いますが、今、ＣＡＳＥツールにせよ何にせよ、使われてないというか、使ってないか知りませんが、その１つの原因は知らないと言うことと、勉強不足ということもあると、私自身も含めまして思います。

このような所に参加して、大いにいろいろなツールを知って使っていただいたら、それなりの成果があるのではないかと思います。

4. セッション・サマリ

Chairman　野村敏次

ＣＡＳＥセッションでは、最初に「制御系でのＣＡＳＥツールの利用について」三菱電機の小野さん、次に「メインフレーム環境でのＣＡＳＥ」を長野県協同電算の佐藤さんに、そして最後に「ＳｔＰの利用経験と評価」をＳＲＡの桜井さんに３０－４０分程度プレゼンテーションしていただき、質疑・応答、討論を行いました.

三菱電機の小野さんは、プラントにおけるコントローラのソフトウェア及びハードウェア設計／製作過程での、実情と問題点についてお話いただきました. 使用しているツールは、自社開発したものが、汎用大型機上にあるが、条件がいろいろあるので、実際にそれらを有効に利用するのは難しいということでした.

佐藤さんの話は、ＣＡＳＥに対する考え方、歴史的な背景を述べられた後、大型汎用機上の第４世代言語と呼ばれている、ＭＡＰＰＥＲやＬＩＮＣを使用経験から概観してもらいました. 特にＬＩＮＣについては、メインフレーム上のＣＡＳＥツールとして捉え、その評価を話してもらいました.

桜井さんには、ＣＡＳＥツールと言われている、ＳｔＰの利用経験と、その評価を話してもらいました. ＳｔＰは構造化分析／構造化設計、およびプロトタイピングを支援するが、日本ではこれらの方法論が浸透していないため、ＣＡＳＥツールの普及も難しいということでした.

３人の方の発表の後でも、各々簡単な質疑・応答はあったのですが、やはり、熱の入った討論は最後に行われました. おおまかに纏めると、

（１）第４世代言語の位置付け

（２）ＣＡＳＥツールの定義

（３）現実に使えるツールが欲しい

ということでした. これらに対する、結論的な意見は以下の通りです.

（１）については、第４世代言語がキチンと作動する、ハードウェア環境が必要であること、また、４ＧＬを使うといっても、余りエンドユーザに頼らない方が良いということです.

（２）については、３つの定義があって良い. 要は、話す人が、今どの定義で話しているかを明確にすれば良いということでした. 因みに、３つの定義は以下の通りです.

- 一つは、コンピュータを使ってソフトウェア・エンジニアリングを行うためのツール.
- 一つは、生産性向上に寄与するツール.
- 一つは、ある方法論を支援するためのツール.

（３）については、使えるツールを知らないだけではないの？

ということに落ち着いたようでした.

ＣＡＳＥツールに関する討論は、これから本当に必要になってきます. この討論が何等かのきっかけを提供できたら幸いです.

5. ＣＡＳＥの発表に関する感想

藤野晃延（ＦＸＩＳ）:

制御プラントのＣＡＳＥと言う話は、実践的で且つ規模の大きいことに驚いた。

佐藤さんの話は、内容がきちんと整理されていて分かりやすかった。特にビジネス、応用を特定ターゲットと

して捉えているのが、ＣＡＳＥの１つの方向性としての可能性を明らかにすることになる、という主張は納得できると思う。後は規模の問題と管理の問題の面から考察が加われば良いと思います。

麻里さんの、「ＰｒｏｂｌｅｍからＳｏｌｕｔｉｏｎへのＭａｐｐｉｎｇが難しい」というのは本当だと思う。又、「そこを支援するＣＡＳＥは今はない」のも本当と思う。ＳｔＰもそうだと思うが、ＣＡＳＥに対して現状の問題と、何を機能として持つべきかを明確にしていかなくてはならないと思う。

杉田義明（ＳＲＡ）：

ミニコンピュータ・メインフレームの上でのＣＡＳＥ、ＣＡＳＥの使用経験とバランスのとれていた発表だった。

ツールとＣＡＳＥの違いについて明確にする必要があるのではないだろうか。方法論及びそのモデルとしてのＣＡＳＥについての説明が欠けていたように思う。

小野康生（三菱電機）：

準備不足でした。すいません。

中野秀男（阪大）：

佐藤：ＣＡＳＥの総括的な話は始めて聞きました。分かりやすかった。

野村行憲（ＩＣＳ）：

発表内容の割には時間が長い。現状ではＣＡＳＥツールはあまり整備されていない。ＣＡＳＥツール以前に開発法放論を身につけるべきである。ファームウェア・メインフレーム・（ＵＮＩＸ）ワークステーションそれぞれの現状が理解できた。

北野義明（ＫＣＳ）：

長野協同電算の佐藤氏の発表は関心の強いところです。非常に丁寧な発表とＯＨＰ・資料で大変勉強になりました。しかし、ＤＯＡの場合、データの整理、ＤＦＤの作成（大きなシステムほど多段階層になる）：等、労力（人手の時）がかかるはずである。それを要求仕様の入力を対話形式で簡単に行えるでしょうか？

ＳＲＡの桜井さんのＳｔＰにつけ加えればいいなあと思うのは名称（データ・プロセス等）：をつける場合の支援環境です。プロセス名に「ｘｘｘ処理その１」、「主要ｘｘｘ処理」とか、データ名に「△△△情報」、「△△△データ」なんてつけないようにチェックするようなものです。大きなシステムほど結構いい加減で後々困る事が多いからです。

藤岡　卓（三菱電機）：

ＣＡＳＥは範囲が広いから曖昧なのか。

上妻健一郎（電応研）：

小野さんの話は数少ない工場での事例で非常に参考になった。

古閑幸一（ＨＳＴ）：

ＣＡＳＥツール、特にコーディング自動化が行えたとして、それはアーティスト的コーダに受け入れられるか。

辻本恵一（シグマ本部）：

小野：問題がツールにあるのか？運用などに関するところにあるのかはっきりしていない。

佐藤：ＬＩＮＣの良さを強調しているが、ボトムアップ的アプローチであり、情報システムの開発には安易ではないか？トップダウン的アプローチであるべきである。ツールの説明が長すぎたのではないか。

坂本　治（シグマ本部）：

佐藤氏のお話は大変参考になりました。

中田修二（シグマ本部）：

小野氏のプレゼンテーションは制御系マイコンソフトウェア開発現場の事例として参考になりました。独自性の高い分野なので当面ｓｅｌｆ－ｈｅｌｐで対処される点が多いと思います。

佐藤氏のプレゼンテーションは４ＧＬ、特にＭＡＰＰＥＲ／ＬＩＮＣの説明として理解できるものでした。他のメインフレーム系の支援システムなどもウェイトをおけば、実現のアプローチの多少の差はありますが、目的は同じ点を共有していると思います。（ＭＡＰＰＥＲ／ＬＩＮＣの説明はもう少し簡単にして、佐藤氏の見方、意見が正面に出ることを期待したい・・・との印象です）

桜井氏のＳｔＰの説明、意見は明快で分かりやすいと思います。

黒坂靖子（ＪＩＰ）：

参考にしてゆきたい事象はありましたが、Ｉｄｅａだけかな？とも思います。

野中　哲（ＮＥＣ）：
いまいちでした。

平尾一浩（ＨＳＴ）：
ＬＩＮＣ：良い（使える）：部分もかなり多いと思ったが、ユーザインターフェイスが弱いと言うのは、基本的な部分が欠けているとしか言えないと思う。又、（統一された）：上流工程のサポートが欠けているのも悲しい。ＳｔＰ：ＳｔＰ自体の問題よりも、それ以前の問題が大きすぎるように思う。結局、使う側の手法、技術が大きく影響する。しかし、使っていたいと思う。

野見山和則（三井銀ソフトウエアサービス）：
メインフレーム環境のＣＡＳＥツールについて、現在の情報が発表されましたが、予想されていた通りで、やはり自分で作らなければ駄目なようです。
桜井さんの本音の話に納得できました。

林　　香（ＳＲＡ）：
紙でやって効果のないことをコンピュータ化したからといってうまくいくだろうか。これがいちばん大きな問題テイキ！何をＣＡＳＥと言うかは別にして、この点においてはメインフレームも結構頑張っているのかな。
ＣＡＳＥってなに？ＳｏｆｔｗａｒｅＥｎｇｉｎｅｅｒｉｎｇって何？となってしまうなあ。

海尻賢二（信州大）：
個人の意見を長々と聞く場ではないはず、座長の割り込みも必要。

渡邊　雄一（電力計算センター）：
佐藤さん（長野協同電算）：の話は，極めて良質なチュートリアルであった。（ウチの若手に聞かせたかった）桜井さん（ＳＲＡ）の良いツールに巡り合ったなあと思う。羨ましい！

山口郁生（電力計算センター）：
こういったツールは、それぞれのアプリケーションや分野個別に作成されるもので、汎用的には難しいものだと感じた。

5.2 ＣＡＳＥの討論に関する感想

藤岡　卓（三菱電機）：
1．使えるツールを作るのは難しい。
2．作ったツールを普及させるのは難しい。

小野康生（三菱電機）：
ＣＡＳＥツールの活用具体案にまでブレークダウンできなくて残念。

林　　香（ｓｒａ）：
第４世代言語≠ＣＡＳＥと思うが！
ＳＥがまともでないのにＣＡＳＥに頼るのは、ＡＩがまともでないのにＡＡＩに走るのに似ていて、本質的でないものでないような気がするが、これが現実か。
討論がｗｈａｔｉｓｔｈｅＣＡＳＥにたどり着けたのでうれしい。ＣＡＳＥ＝「方法論の開発を支援できたりするもの」という考え方は面白い。売れてない（役に立たない）：ツールにはＣＡＳＥと名づけ、役に立つようになると単なるツールとなる（篠田@東工大・談）

野村行憲（ＩＣＳ）：
メーカとユーザのバトルが良かった。（現状を物語っている）

田中正則（ＳＲＡ）：
ＣＡＳＥツールのユーザは誰か？
エンドユーザかエンジニアか？
使い方次第なのか？
分からなかった。

藤野晃延（ＦＸＩＳ）：
ＣＡＳＥとは
1．誰がＳｏｆｔｗａｒｅＥｎｇｉｎｅｅｒｉｎｇをするのを、Ｃｏｍｐｕｔｅｒが
支援するのか？
2．誰かが、ＳＥを知らなくても、Ｓ／Ｗをｄｅｖｅｌｏｐ出来るようにするものなのか？
3．Ｓ／Ｗ開発に拘る人、全てにＳＥ（工学）の大切さを認識させるための道具なのか？
ウーム、難しいなあ。ソフト工学（ＳＥ）自体は「統計」と同じように学際的なものだと考えられるから、それ自身ではあまり意味はないと思う。つまり、「何か目的を達成した人に」道具として「ＳＥを提供する」のが

ＣＡＳＥだとおもう。

野見山和則（三井銀ソフトウェアサービス）：
ソフトウェア技術者がアプリケーションを学ばなければダメだという意見には大賛成！（ユーザの中に、自分の業務を知らない人はたくさんいます。）

関口純一（ＪＩＰ）：
ＣＡＳＥの定義に関する討論は興味深かった。

中田修二（シグマ本部）：
議論が活発にあって良かったと思います。

野中　哲（ＮＥＣ）：
いまいちでした。

古閑幸一（ＨＳＴ）：
エンドユーザは何をやりたいのか知らない→正しい！
どこかで企画を担当する部門（集団）が必要か？
エンドユーザのコンサルタントが必要？

辻本恵一（シグマ本部）：
ＣＡＳＥとはというテーマより、要求定義、設計・・というテーマ設定で行うべきではなかったか。ＣＡＳＥとは何を話しても意味はない。

棚田眞司（ＳＲＡ）：
単なるオペレータはいやですが、ＣＡＳＥツールでサポートしている部分だけやってくれる専門の人がいるとうれしいです。（私はやりません）でも、プログラムのジェネレーションまでＣＡＳＥがやってしまうようになると、やっぱり使うしかないのかな？

渡邊　雄一（電力計算センター）：
ＣＡＳＥの前提となる，モデリングの話をチャントする必要があると思う。

山口郁生（電力計算センター）：
ＣＡＳＥについては、あまり知らなかったが、結局「なんだ、こういうことか」で終わってしまった。