5. 中 間 発 表

中 間 発 表

報告者
中尾 博司・方 学芬
（静岡大学）

中間発表の討論の経過を時間順に報告する．

1. PWB & Hyperbook グループの発表

われわれは，まず課題の中から，ガス給湯システムを選択しました．図書館システムにしようかという案も出たりして，途中いろいろ考えたんですが，多分内容の説明が詳しい方が簡単だろうということで，ガス給湯システムを選択しました．

問題を最初眺めたときは，手がかりがまったく掴めませんでした．そこで，モデルを選んでみようということになりました．PWB では，昨日の説明にあったように，6 つのモデルが用意されています．まず，最初は（難しい）FSM モデルから入るのがふつうだという話が出て，システムの機能構造のモデル化を試みました．

しかし，実際に考えはじめてみると，なかなか理解しがたく，構造記述がうまく行きません．それでは他のモデルで試してみようということになり，EOPM モデルに移りました．そのほうが書きやすいのではないかと思われたからです．これは，システムをユーザの立場からモデル化するものです．具体的には，状態遷移図を書いて行くわけですが，これもなかなかうまくいきませんでした．全体の書き方がよくわからなかったのです．

そこで，給湯システムを 3 つの部分に分けました．給湯する部分（図 1）と，お湯はりする部分（図 2）と追いだき部分（図 3）です．みんなで各部分ごとに状態遷移図を作成して，PWB に入力しました．

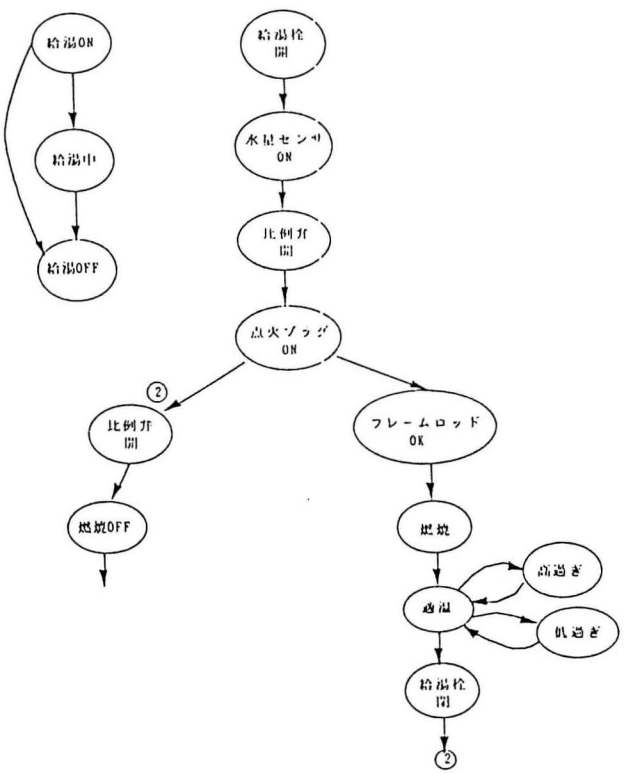


図 1 給湯する部分 (PWB)

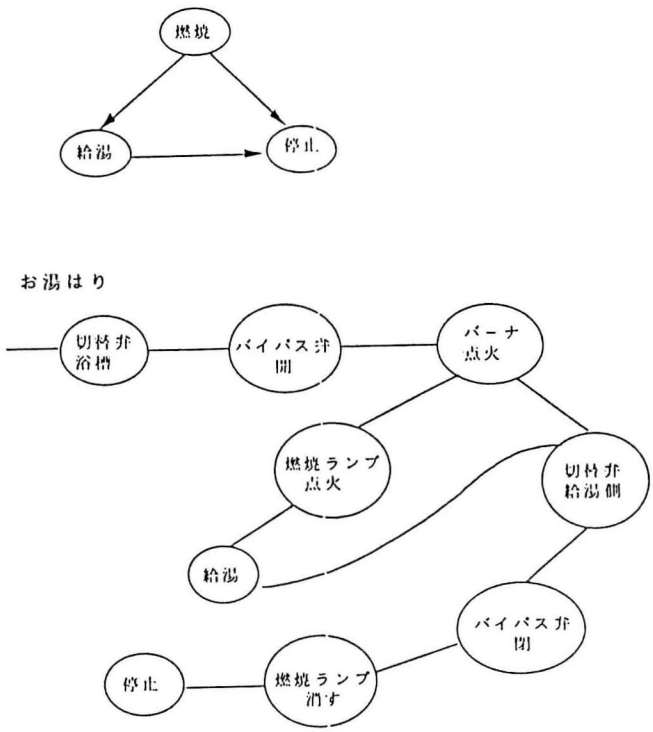


図 2　お湯はりする部分 (PWB)

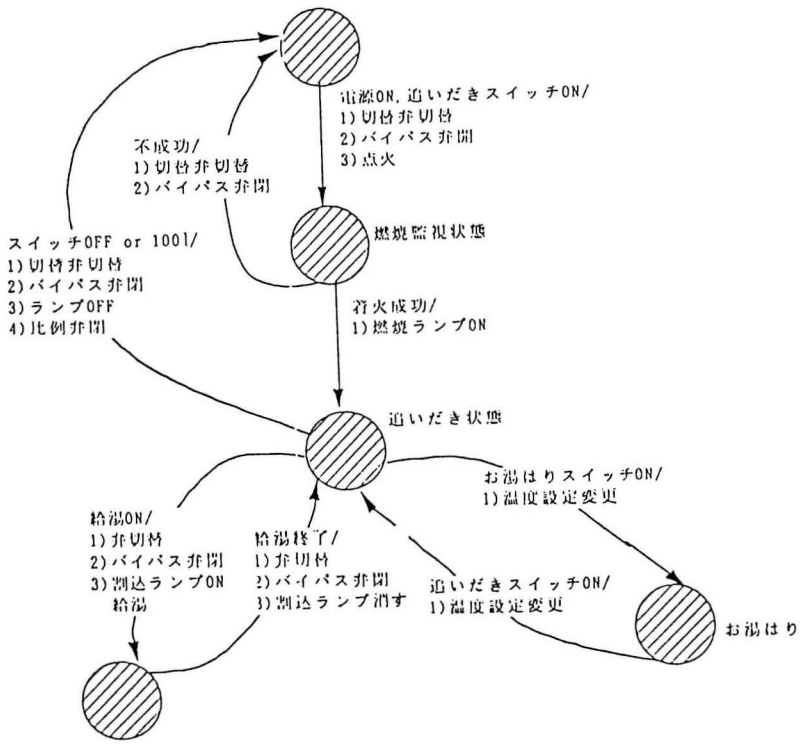


図 3　追いだき部分 (PWB)

ツールの操作に慣れていなかったので，なかなか仕事が進まず，入力し終わった時点で予定の時間がきてしまいました．PWB にはなかなかよいアニメーション機能が用意されているので，それを使って一応の動作を確認しました．

さらに，この状態遷移図が本当に正しいかどうかを検討したかったのですが，そこまでの時間はありませんでした．以上です．

落水浩一郎（静岡大）: 状態遷移図をちょっと説明していただけませんか？

PWB: いや，まだあまり検討していないので，あとで検討がすんでからの方が，議論がより豊かなものになると思います．

2. OOA/OOD グループの発表

われわれのグループもやはり給湯システムを選び，OOA ツールを使っての分析を試みました．われわれが今回目標にしたのは，昨日の青木さん (FXIS) の話にあったオブジェクトの確定・属性の定義・サービスの定義という段階までです．今日はこのうちの確定作業から始めたのですが，ご覧になったようなところまでできました（図 4）．次にオブジェクト間に is-a 関係と has-a 関係を定義しました．三角がついている線が has-a 関係，丸がついている線が is-a 関係を示します．

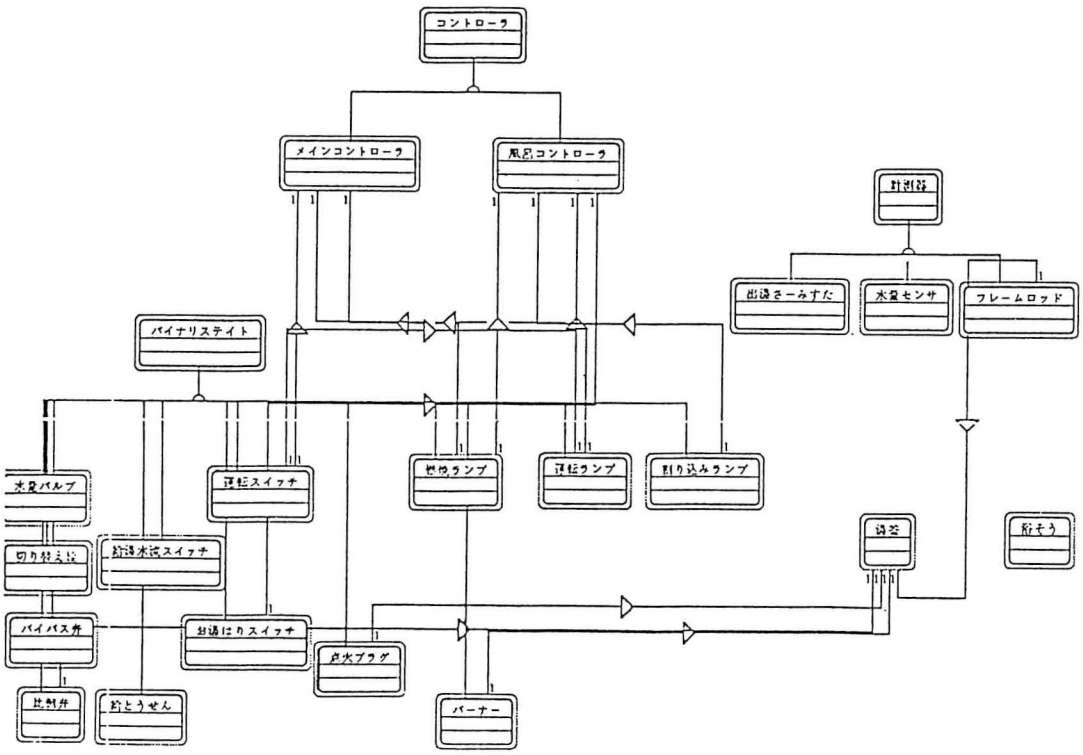


図 4　確定されたオブジェクト (OOA/OOD-1)

苦労をした点ですが，湯わかしシステムには沢山の名前が出てきます．問題に直接あらわれる名前は簡単に決定できたんですが，抽象化することで得られる概念，たとえば，スィッチ群に対して，それらは状態を 2 つ持つので，バイナリ・ステイトという上位のオブジェクトを認識する，これがものすごく難しくて，インストラクタの青木さんの顔色を窺いながら，いろいろキイワードをポンポン並べ，難しい顔をした場合には取り下げ，当ったものを案とするような結果になりました．残りの部分は完成度が多分まだあまりよくないと思います．とりあえず，現在の合意を形にしようというところまで行きまして，中間発表にしました．

浴槽が一人寂しく孤立しているのですが，これが明日の発表までにどんな風に変って行くか，それとも消えているか，何となくわからない部分が残っていますが，そのへんがポイントだと思います．

人数が多いので，グループを 2 つに分けたのですが，もう 1 つのグループもまったく同じ問題と同じツールを使っています．

最初に進め方を議論しました．詳しく中身を読んでからやろうか，それともオブジェクトとしてのキーワードを取り出してからやろうか，ということをまず話し合って，その結果，キーワードからオブジェクトを決めて行くアプローチを採用しました．一応キーワードを 4 人で考え，どういうものがあるかを決めました（図 5）．今のところ，まだ is-a 関係までしかできていません．

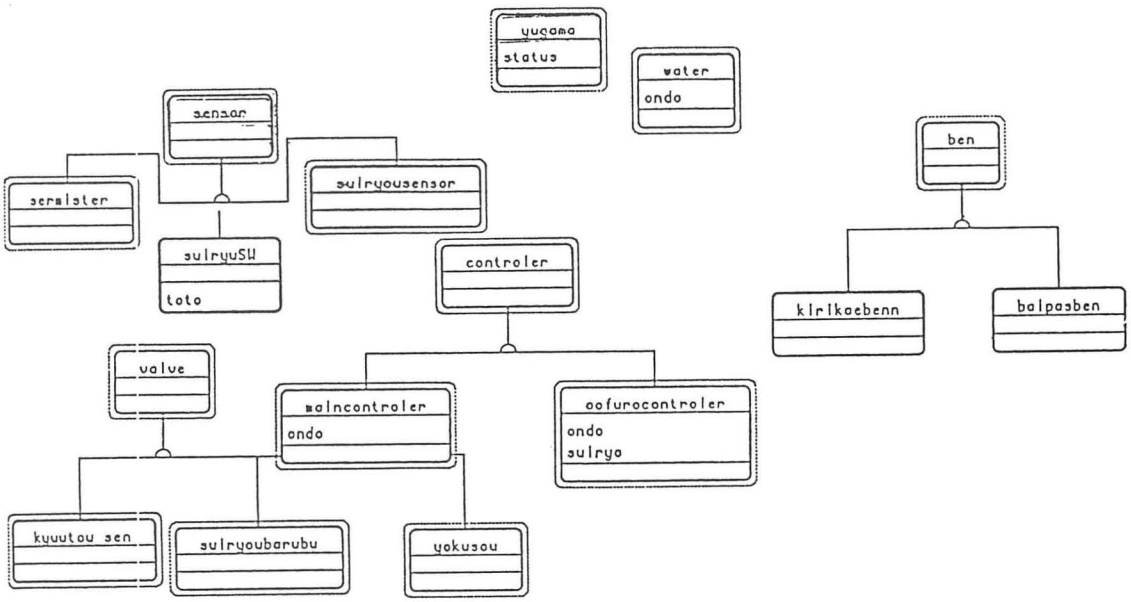


図 5　確定されたオブジェクト (OOA/OOD-2)

3. CASE Bench グループの発表

われわれのグループが選んだ課題は，酒屋倉庫問題です．CASE Bench のデータフロー図から構造図への変換機能が強力であるということを信じて，明日の昼までにゆっくり完成しようというふうに考えて進めています．

まず，今日最初にやったことは要求内容の検討ですが，問題の中の「受付け」というのは実は在庫管理じゃないかとか，倉庫という言葉があるがこれは実は運搬係じゃないかとか，課題に文句をつけることから始めました．その後データの洗い出しをやりました．その結果をちょっとだけ見ていただきます（図 6）．

それから予備的なデータフロー図を書きましたが，この時「ビックバン」という手法だそうですが，まずデータフロー図を書くんじゃなくて，とにかくバブル（処理）をまず書いて，それをあらかじめ洗い出したデータでつないで行くという方法をアドバイスしていただきました．その後で，コンテキスト・ダイアグラムを書きました（図 7）．これが皆の統一見解ということになっています．

・データを操す

積荷票
受取確認書

出庫依頼

出庫指示書

在庫不足リスト

空コンラナ通知

受付者が
在庫量
受引先リスト
商品名

不足連絡

依頼者名簿

・受付係は在庫係でもある

図 6　洗い出されたデータ (CASE Bench)

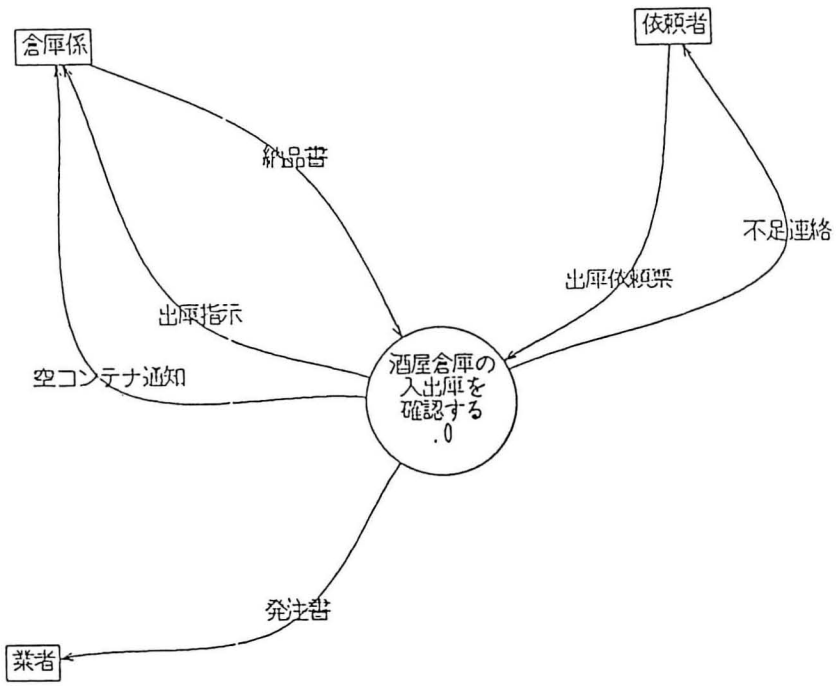


図 7　コンテキスト・ダイアグラム (CASE Bench)

現在，コンテキスト・ダイアグラムはバージョン 1 ですが，第 2 階層までできました(図 8)．その後でデータ辞書を登録しました(図 9)．私がやったのではありませんが，入力が大変だったそうです．内容については，このあと第 3 階層まで詳細化して，みんなでレビューし，正当性を検証しようと考えています．

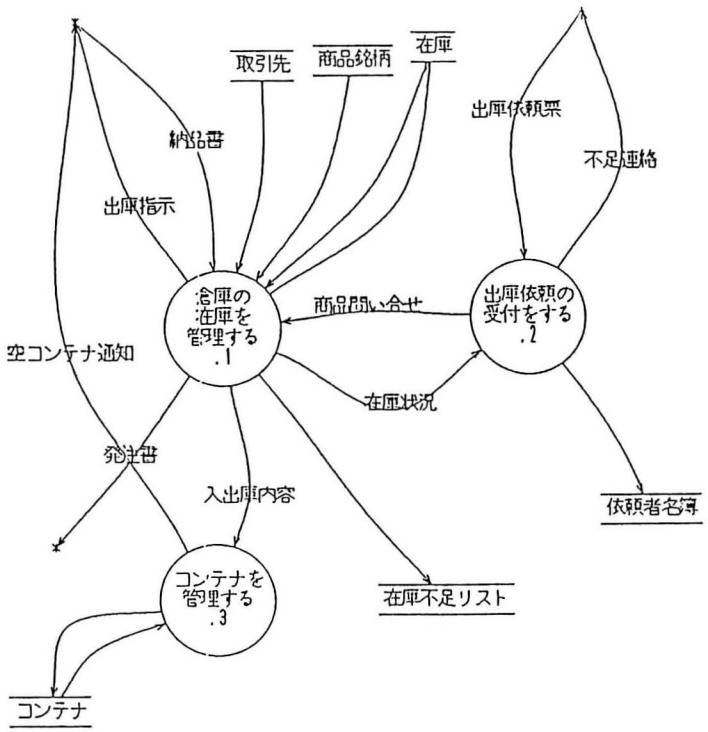


図 8　データフロー図 (CASE Bench)

コンテナ = コンテナ番号 + { 品名 + 数量 }
依頼者名簿 = 依頼者番号 + 依頼者名 + 依頼者住所 + 依頼者電話番号
空コンテナ通知 = コンテナ番号
在庫 = 品名 + (数量 + コンテナ番号)
在庫状況 = 品名 + 数量
在庫不足リスト = 送り先名 + 品名 + 数量
取引先 = 取引先名 + 取引先住所 + 取引先電話番号
出庫依頼票 = 品名 + 数量 + 送り先名
出庫指示 = 注文番号 + 送り先名 + { コンテナ番号 + 品名 + 数量 + 空コンテナ + 搬出マーク }
商品銘柄 = 品名 + 品名コード
商品問い合せ = 品名 + 数量
入出庫内容 = コンテナ番号 + (品名 + 数量)
納品書 = コンテナ番号 + (品名 + 数量)
発注書 = 品名 + 数量
不足連絡 = 品名 + 数量

図 9　データ定義 (CASE Bench)

4. StP グループの発表

われわれのグループは図書館システムに取り組んでいます．いま，コンテキスト・ダイアグラムの入力が終わったところです．StP が乗っているマシンが 2 台があるので，グループを 2 つに分けて，それぞれが独自に分析・設計を行なう予定です．いまのところ，進度は大体同じぐらいです．明日以降は 2 つの分析結果をマージして，それをもとにステップを踏んで設計を進めて行きたいと思っています．

いまできているコンテキスト・ダイアグラムは，図 10 に示す 2 通りのものです．これまでにぶつかった主なトラブルは，記法へのとまどい，たとえば入出力の線の書き方などです．それから，私のグループで特に注意した点は，たとえば図書館では，本というものが動くのではなくて，たとえば貸し出し帳にあたるもの，そのような概念をデータとして取り扱うということを，順次確認して行きました．苦労した点は記法につきます．たとえば，例外処理はどう書けばいいのか，入出力の線をどういうふうに書けばいいのか，そうしたことがらに，何しろ初めてなので，とまどいを感じています．明日の最初の作業は，今日の結果を統合して，レビューすることです．

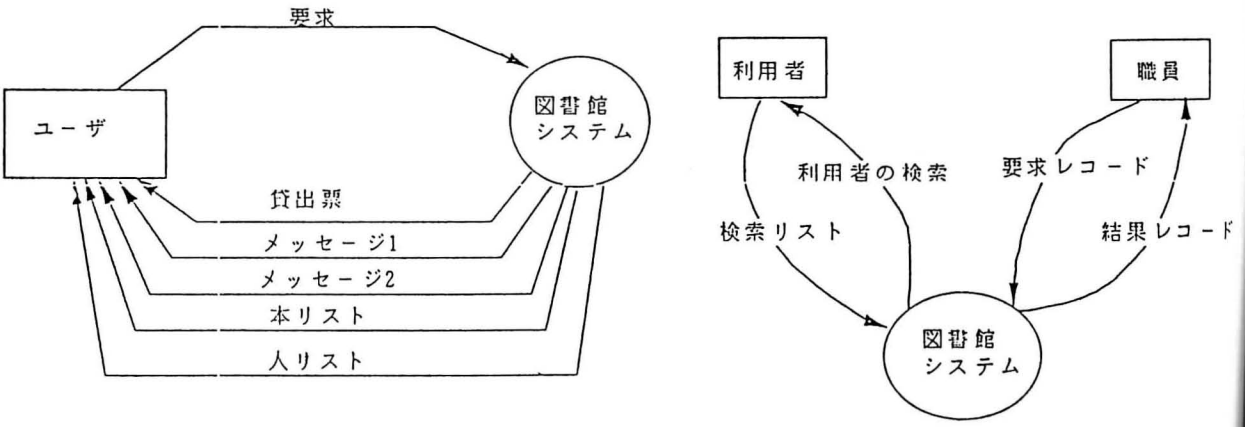


図 10　2 種類のコンテキスト・ダイアグラム (StP)

5. Teamwork グループの発表

われわれのグループは制御系の経験者が多かったので，給湯システムを選択しました．まず，問題を十分認識しようということで，モデルを読みこなすという作業をはじめにやりました．そこで，ちょっとした意見の対立がありました．一方は，ハードウェアの構成をしっかり確認したいという意見であり，他方は，システム分析を DFD を使ってやっていこうという意見です．結局，後者の考えにしたがって，システム分析を進めて行くことになりました．

そこで，Teamwork でデータを入力しつつ，すべて画面の前でみんなで議論しながら，問題を解決して行こうという話になりました．9 人のメンバ全員が Teamwork に触れるようにしたいということで，DFD を少しづつ作りながら，オペレーションを交替で行いました．こうしてできたのが，図 11 に示す給湯システムのコンテキスト・ダイアグラムです．

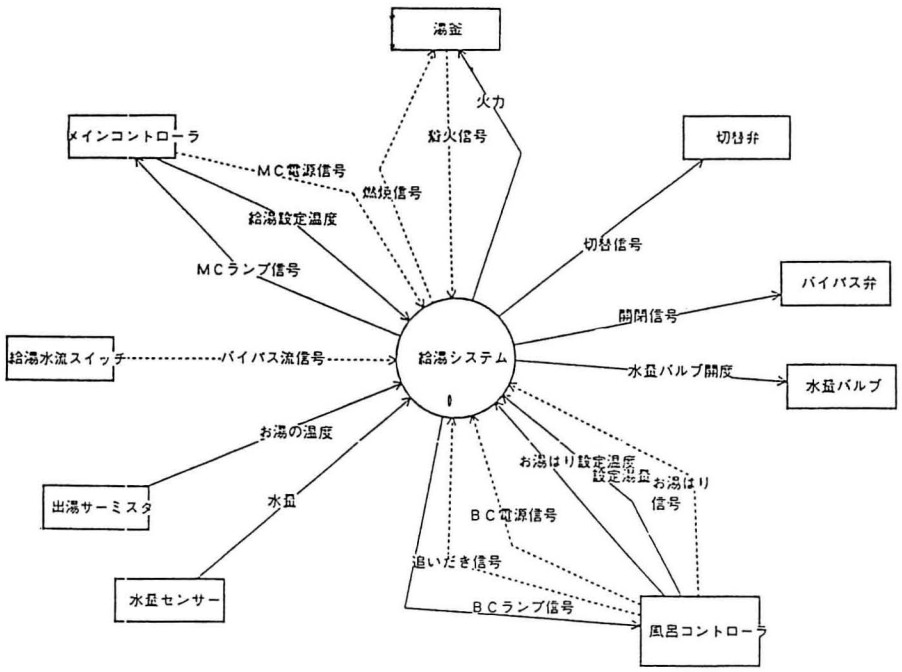


図 11　コンテキスト・ダイアグラム (Teamwork)

この段階までは，まあ順調に，割合スムーズに話し合いが進みました．ところが，この後，第 2 階層目に入ったところで，ちょっと話がもめました．給湯システムを外側から見て，給湯栓に給湯する機能，浴槽にお湯をはる機能，それから追いだきをする機能，この 3 つの主要な機能に分解しようとしたのですが (図 12)，これに対して，データを付けていこうとした時にうまくいかない．

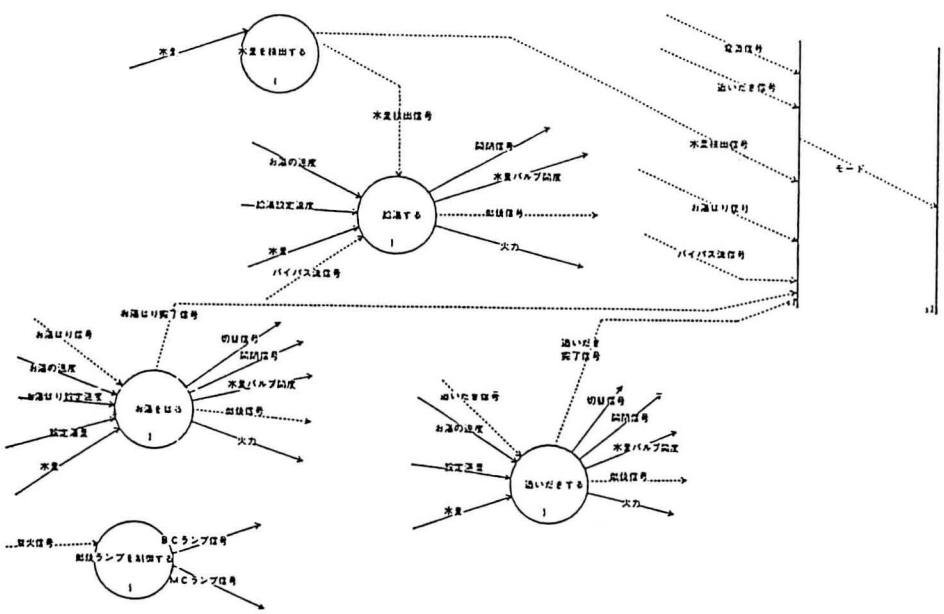


図 12　3 つの機能分解 (Teamwork)

温度を調整する機能がそれぞれの場所にある．これはまずいのではないかという意見が出ました．また，もう少し内部実現に関わる機能を含めて考えたほうがいいのでは，という意見も出て，外側から見た機能を主体にしてプロセスを考える派と，内部実現も考慮して機能とプロセスを決める方がよいという派に分かれたわけです．

そこで，温度を調節するとか水量を監視するといったプロセス，それから，お湯をわかすというプロセスもここで出してしまった方が，下部で重複がなくなるのではないかという話が出て，ごじゃごじゃともめてしまいました．現時点では，まだうまい解決策が見つかっていません．明日の朝，頭が冴えたところでまた考えなおして，どちらかを取るか，あるいはグループを 2 つに分けてそれぞれのやり方で進めて行くか決めたいと思っています．

6. 全体の感想と討論

佐原伸（SRA，OOA/OOD グループ）： 最後の Teamwork グループのような形で発表をしていただくと，他のグループの方にも，だいたいどんなプロセスで事態が推移したのかがトレースしやすいと思います．最終報告のときも，この線に沿って，どんな手順で，どんなところまで行ったのか，結果はどうだったのかを報告していただければ，非常にありがたいと思います．その他のグループの状況は，私には何となくが想像できるのですが，多分他の人にはあまりわからなかったんじゃないかと思います．最終発表では，そのあたりを考慮して発表してくださいい．よろしくお願いします．

野村行憲（岩手電子計算センター，実行委員長）： 思いのほか，早く発表が進んだので時間があります．もう少し時間がかかるだろうという想定で，このあとの「わんこそば」は，7 時スタートでお店にお願いしてありますので，まだ，おそばはゆだっていないと思いますから（笑い）．

落水浩一郎（静岡大，プログラム委員長）： 一般に，CASE ツールに対する不信感が存在します．たとえば，方法論そのものが役に立たない．教科書にはいろいろ書いてあるけれども，具体的なことがわからないし，教科書で指定されたやり方は，自分のこれまでのやり方と違っているので，結局方法論なんか使いたくない，ということになってしまう．また，別の形の不信感は，CASE ツールって，ただ，お絵書き機能のついたワープロではないか，別にそんなもんを使わなくても設計はできるという意見です．

私がこのワークショップを通じて知りたかったのは，ある課題を特定の方法論にしたがって，CASE ツールを用いて解決する時に，それによって得られるメリット，たとえば黒板に書いたり，頭の中で考えたりする場合に比べて，CASE ツールの枠組みにしたがって，自分が考えたことをコンピュータの中にどんどん投入していった時に，その見返りとして，環境つまりツールの方から，自分自身の思考が整理されて返ってくる．そのような現象が，いつどういうところで起こるのか，そのような感想を，最後にお聞きしたいと思っています．

今回の発表は，全体として，順調にスタートしました，とにかく手掛かりを掴みましたということなので，まだ議論できる状態ではないと思います．それにしても，単なる状況報告にとどまらず，もう少し何かを話していただきたかった感じがします．発表の後に質疑を入れるべきだったかもしれません．そのあたりについて，もう少し議論してみたらどうでしょう？ それとも，今日は止めにしますか？ 今のところでは，「まあいいじゃないか」という顔のほうが多いみたいですね．

佐原： ちょっと，いまのレベルでは，まだ CASE ツールの効果とか方法論の限界とかを云々するのは，早すぎるのではないですか？

落水： いや，今の時点でも議論できることはあるし，またすべきであるという意味です．このまま食事に行くのでは，ちょっともったいない．他のグループに質問をしておきたいとか，確認してみたいとか，何かあるでしょう？

たとえば，1 番目のグループでしたら，なぜうまくいかなかったのかという質問が出て然るべきかと思うのです．PWB を利用するにあたって，一番適切なモデルを選択して進めれば，後はまあ順調に行くというお話でしたが，モデルの選択がキーポイントなのかな？ と私は思うわけです．私がもともと理解しているのは，PWB は，いくつかのモデルに並行的に情報を入れながら，システムのコンセプトを作りあげていくという方法を支援するツールであるはずなので，このままではうまくいかないのではないかと，私は思います．

それから，2 番目のグループでは，バイナリ状態のオブジェクトを発見するところで，皆さんすごく苦労され

た，というお話があったかと思います．Smalltalk との関係はどうなっているのでしょう？ Smalltalk 環境に準備されたクラス階層を Top-down にブラウジングしていって，目的のものに近いもの，またはその手がかりをみつけるというようなことはやられたのでしょうか？

3 番目の発表の中では，「いろいろ考えました．大事なことは全部考えました．そのあとで，何かを CASE ツールに打ち込みました．しかし面倒くさいですから私はやりませんでした」という発言が，すごく印象的でした．すでにわかったことを，なぜわざわざあらためて CASE ツールに入力しなければならないのだろう，とうふうに考えていらっしゃるとすれば，それは，やはり CASE ツールを信用していないというようにも，受けとられます．

4 番目の StP グループでは，コンテキスト・ダイアグラムを一応作られたという状態でしたね．「マージする」というのはどういうことなのでしょう．全体の活動を，最後に一つにまとめあげようという意味で，おそらく，設計というよりは，グループの協調という観点から，マージという言葉をお使いになったかと思いますけれども，一見したかぎりでは，簡単にマージできるような図ではないような気がします．

それから，最後の Teamwork グループですが，このグループの発表が遅れてしまったことについては，私がちょっとお詫びしなければならないことがあります．そろそろ中間成果ができあがって，あと 30 分ぐらいでまとめに入らなければ，という時に，私が横を通ってコードを足にひっかけ，電源をポンと引っこ抜いてしまいました．申し訳ございませんでした．私のほうにそういう弱味があるので，あまりいえません（笑い）．

全般的に CASE ツールを使いこなす活動に専念していくなかで，中間発表や，最終発表という節目では，事例にしたがってやってきたことを少しまとめ直してみたり，もし問題点があるなら，それを見つけてみようというように話を進めていただきたいのです．

みなさんがワークショップに参加された目的はいろいろだと思いますが，全体の共通の目標は，４つの課題を解くというだけではなく，その問題解決過程の中で，CASE ツールを使った時にどういうところに有効性があるか，どういうところに問題点があるか，そのへんを少し見通しを付けるということだということを忘れないでいただきたいのです．

佐原： 現状は，何がどう役に立ったかを考える以前の状態でしょう，モヤモヤといろいろ試行錯誤している段階ですから．もう少し先に進むと，エラーチェックその他の活動が出てくるはずです．

Teamwork はそういう状態ですし，こちらの Smalltalk のグループはまだほとんど仕事が進んでいなくて，「オブジェクトは何か？」を議論しているところです．Smalltalk のクラス階層とか，あと，どううまくいくかという話はまだ全然出ないですね．たぶん，その見当や評価は，皆さんの中には，まだほとんどないんじゃないかと思います．

落水 そうですか....

CASE Bench： 私たちのグループは，ちょっと Teamwaork グループに似ていますけれども，いま DFD をやっと作ったところです．実情はまだその前の段階なんですけれども．

われわれのグループは，インスチラクタの渡部さん（ソニーテクトロニクス）のご指導で，まず情報の流れに着目して，情報とは何だろうかというアプローチをしました．そうすると，比較的に順調に，一応 DFD ができあがりました．ところで，Teamwork のグループは，どのようなアプローチしたのか？ その辺をちょっと教えていただきたいのですが....

Teamwork： われわれのグループも，基本的にデータの方から入りました．ハードウェアを押さえてやろうという意見は一応引っこめまして，DFD で全部分析しようということになっています．最初は私もこんがらがっていたんですが，ハードというか，物理的なものと情報の区別をはっきりと明らかにして，情報というものをベースにやっています．また DFD の作り方としては，システムに流れる情報が何かということについて，全員で画面を眺めながら，誰かが案として挙げたものをたたき台にして討論するという進め方をしています．基本的にはデータをベースでやってきました．

CASE Bench： そうすると，あらかじめデータを全部洗い出したわけですか？

Teamwork： そうですね．コンテキスト・ダイアグラムの中で，データを全部出しつくしたということです．

CASE Bench： 先ほどうまく行かないとおっしゃっていましたね．あれはどこがうまく行かなかったんです

か？

Teamwork: お湯の量とか，温度とか，こういったデータは，いずれの機能にも関係してくる．いずれの機能にも入ってくるということになると，１つのデータがいくつかのところに分散されて行くことになります．ということは，さらに機能分解していくと，同じような機能があらわれてくることになる．それはまずいのではないか，という意見が出たのです．

CASE Bench: 線の中に埋めちゃうと，全部に入っちゃうと，そういう意味ですか？

Teamwork: はい，そうです．たとえば，ここにも入るし，ここにも入ります．ですから，そういうふうに書いていって，あとで分割してもいいのではないかという意見と，それはちょっとまずい，後々になって，共通の同じような機能を持ったものがこの下にそれぞれできてくるのだはないか，という意見の２つに分かれたわけです．

CASE Bench: ああ，そういうことですか！ いまはそういうふうに意見が分かれていて，先ほど説明があったように２グループに分けている．どちらをとるかはまだ決心がついていないが，とりあえず並行的にアプローチをしたいということですね？

Teamwork: 明日の朝，もう一度考えてみようということです．

野村: よろしいですか？

CASE Bench: 制御系のプロセスにとって何か必要か，それが大体見えてきた，ということですね？

Teamwork: そうですね，実際３つの機能に分けていた段階で，１つコントロールがいるだろう．このコントロールを一緒に合わせて考えて行くと，温度が全部に入ったりして，えらいややこしくなる．そういうイメージもあったと思いますけれども....

CASE Bench: はい，大体わかりました．

落水: 成果物でまだ十分でないものをあまり説明するのはまずいという配慮があったようですが....，つまりPWB のグループです．このことは，やはり，ちょっと討論した方がおもしろいんのではないかなと思います．

というのは，実は３年程前に，私は学生たちに「状態遷移図の手法を使ってヒーティングシステムを解け」という課題を出したことがあります．そうしたら，「できました」といって持ってきた一番最初の状態遷移図は，たしかに丸と線がたくさんあり，入力（トリガー）と出力（レスポンス）も書いてあるが，およそシステムイメージとは程遠いわけです．問題の世界をそのまま丸と線の世界に書き直した形になっていました．

その時私が学生に説明したことは，「状態とは一体何か？」ということです．つまり，「問題の中に含まれている言葉の中で，将来プログラムの変数になるもの，つまり状態を決めるものをまず列挙しなさい．そのようなステートベクトルが，どのように変化するかというシステムの動作イメージを描きなさい．その時，設計のポイントになるような変数の値の変化に注目して，それを丸にしなさい」というふうに説明しました．

次に提出してきた状態遷移図は，あっという間にプログラムになりました．状態遷移図を書くところまでが難しい，すなわち状態を認識するところが難しいのです．私が学生に教えたアプローチがよいかどうかは別として，結局 CASE ツールは，状態遷移図を書いたり編集したりする作業は支援してくれるわけですが，状態を発見するところを支援させるかどうかというところが議論のわかれ目になります．

PWB グループの発表で，「状態遷移図は一応できましたが，どうもうまくいきません」というのは，まさにそのへんを議論するきっかけになります．そういう意味で，状態遷移をちょっと説明してほしいと先程質問したのですが，どうですか？ お答えいただけませんか？

PWB: まことにもっともな御質問ですが，一応の狙いとしては，とりあえず書いてみようということでした．いま，まだ見せられないという理由は，だれが見ても，ちょっとどうかなという感じのするものがいまできあがっているので（笑い），これからもう少し検討を加えて人に見せられるようにしたいということなのです．

まだ部分的に３～４人ずつで書いて，それらを入力しただけの状態で，だれも他のグループの書いたものをほどんと見ていない状態なんで，一応一通り検討してからだったら，みんなが納得して，この場に出せるようになると思います．

野村: わかりました．あと，先程の落水先生のコメントの中で，StP グループのマージが云々ということがあ

りましたが....

荻原剛（大阪大，StP グループ）: 先程はコンテキスト・ダイアグラムだけしか見せなかったんですが，実は A,B 2つのグループに分かれて，最上層の記述までは，両方とも進んでいます．コンテキスト・ダイアグラムだけではなく，その先の階層とか，あるいはデータの定義のところまで，ある程度はできたいます（図 13, 14）.

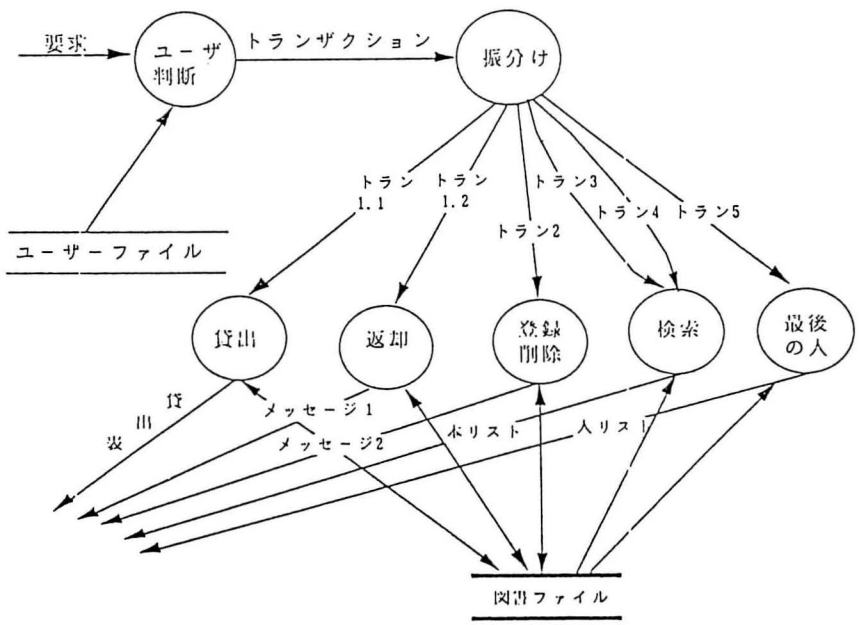


図 13　データフローダイアグラム (1)

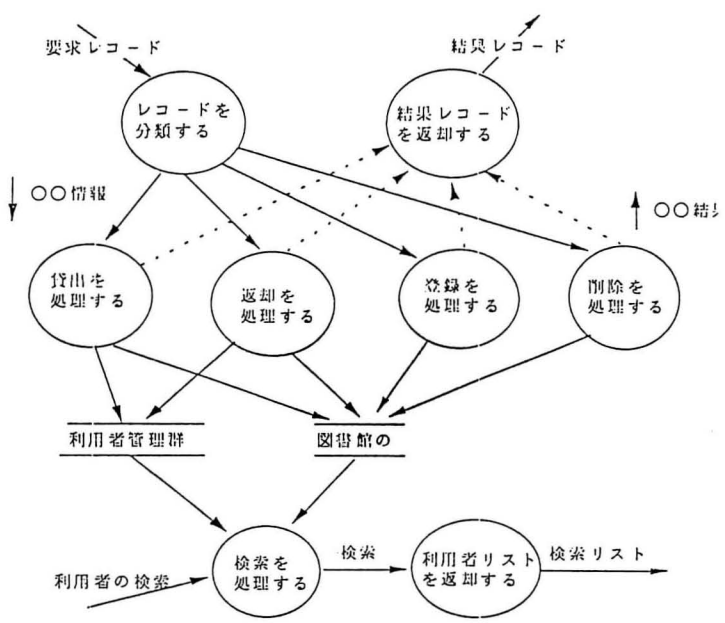


図 14　データフローダイアグラム (2)

先程，マージという表現をしましたが，要するにどこで詰まっていたかといえば，見ている感じでは，自分の頭の中でもやもや考えていることを，コンテキストダイアグラムという明確な形式に置きかえるという作業に慣れていないせいもあって，どうやって表現したらいいかわからない，なぜこのように図形に書かなければならないのか，とか，データの階層表現にとまどっているとか，そのへんでちょっと詰まっているという感じがしました．

実は，最初のうちは，２つのグループがやっていることが全然違っていて，もしかしたら，まったく別個の，全然考え方の違うシステムができあがってくるのではないかなと思っていたのですが，しだいに似たような格好になってきて，両方のコンテキストダイアグラムを比較すると，違いは入力源が１つか２つかということです．ところが中身の具体的な処理の部分でやっていることはだいたい似ていて，要するにトランザクション分析になっているわけです．システムに到着した処理要求をトランザクションと見て，それを振り分けて，貸し出しとか返却とか登録･削除とか検索などという処理をして，結局それぞれが図書ファイルという，情報が入ったデータを読んだり書いたりする，というような仕組みになっています．

先程，一見マージができるようには思えないといわれたのですが，詳細部分を書きすすめているうちに，かなり似通ったシステムになってしまったということです．

今日の段階では，DFD を書いて，設計上の思考を DFD に反映させるというトレーニングをしたということで，その点ではかなりの成果があがったと思います．実際に問題を解くということに関しては，明日以降の進捗がどのぐらいかにかかっています．そういうことなんですけれども....

落水： たいへんよくわかりました．最初からそういうふうに説明してくだされればよかったのに（笑い）．

野村： いまの説明に対して，ご質問は？　いいですか？　ありがとうございました．あと，Teamwork グループは何か補足とかコメントがありますか？

佐藤千明（長野県協同電算，Teamwork グループ）： 途中経過について一言だけコメントしますと，先程，落水先生がいわれたワークショップの目的の中の，方法論や CASE ツールの有効性しついてですが，これは明後日の最後のパネルにならないと，まだいまはみなさん意見が出せない状況だろうと思います．というのは，CASE ツールというのは，佐原さんもいわれましたが，ある程度のところまで設計を進めた上で，その後やり直しが効くとか，チェックできるとか，そういった点で有用性なわけです．ものを考える時に，ツールが自分の思考を助けてくれるかというと，はっきりいって，まだ，いまのツールはそこまでいっていないと思います．

ですから，最初はみんな机上でやってしまったわけですけれども，機械に取りかかる以前の問題として，ある方法論でその問題を解くことがまだ自分の技術として身についていないために，ツールにそのまま飛び付いても，ツールが何の答えもメリットも返してくれないという状況ではないかと思います．今回のワークショップは，そもそもツールを使うということなんで，最初から機械に向かおうということで始めたわけです．結果として，コンテキストダイアグラムまでなんとか行きました．

これはある意味では，ツールがよかったからそこまで行ったわけではなく，ものの考え方として，今までのところ構造化手法の第一歩のところまでやっと理解を得たということです．問題はその次です．先程，いっぱいあるバブルをどうしたらいいかというところでもめた理由はですね，必ず解が得られる方法論というものはなかなかなくて，ほとんどの方法論はいろいろ経験しなければ解が出てこないわけで，恐らくそれが今日から明日にかけてみんなで苦しんで，ああでもない，こうでもないとやってみた結果になると思います．

そこで仕事をやり直す必要があるとみんなが思い，その時途中まで戻ってやり直せるという点でツールのメリットが出てくる，というのが最終的な落ちなのかなという気がします．ただ今の進捗からいうと，まだそういう意味で，ツールの本当のメリットというのは出てきていない．そういう意味で，方法論という壁にぶちあたっているわけです．どっちにしようかな，おそらくどっちでやっても最終的にはできるもんですけれども，そこでこっちがいいよ，というように，チューターみたいな役割をしてくれるツールが出てくれば，もっと使いやすいだろうという感じを持っています．

落水： 皆さんお疲れで，そろそろ終わりだなと思われた時に，私がわざわざ議論をふっかけた意図は，いま佐藤さんにお答えていただいたようなことを，一応議論しておきたかったのです．このワークショップは，よいツールを探すのが目的ではないし，ツールの講習会でもないわけです．ですから，方法論という枠組みにしたがって，ツールを使いながら，問題を解決していく進め方の中で，整理できることは整理しながら進めたいと思いました．

とにかく，今の時点で何か話ができるようなこと，または今後考えなければいけないこと，そのへんのところをし少し整理しておきたいと思ったので，あえて挑戦的ないい方をしました．いまの佐藤さんのコメントで満足しています．

野村： あと何人かの方にコメントをお願いしたいと思いますが，Smalltalk グループからは何か？

佐原： 私自身もオブジェクト指向の方法論をよく知らなかったのですが，青木さんの話を聞きながら考えていたのは，Smalltalk に依存しすぎた OOA/OOD になってないか，たとえば，具体化を Object Pascal とか Objective C とかを用いて行った場合でも，分析や設計のプロセスが変わらなければ，それは完成された技法だといえるが，いまのところ，まだどうもそうでもないかな，という気がしています．そのへんは明日あたりに見きわめたいと思います．

最初に，青木さんから説明があった時，「Smalltalk にはこれくらいの記述力があるんですよ」といいながら，ちょこちょこっと細工して，あるウィンドウが線に沿って動くプログラムを 10 行ぐらいで作られましたね．あれで，「すごいなあ！」と，グループのみんなが意識したと思います．それから，Object Works についても，やはり，青木さんが作ったツールをまとめて，Smalltalk の動きを視覚的に表現しました．それを見てやはり全員が，すごいと感じたはずです．つまり，Smalltalk の方法論的原理は，イメージ的には好感をもって理解されていた．

しかし，実際に練習問題にぶつかって，オブジェクトをどう決めればよいかを考えようとすると，インスタンスレベルのオブジェクトは，(問題にほとんど書いてありますから)すぐにリストアップできるのですが，青木さんに「もっと抽象化しなければいけない」といわれたとたんに，みんな頭を抱えて，(私も含めて)「うう〜ん」とうなっていたわけで，その抽象化レベルが Smalltalk に依存するのか，何か他の処理系に依存するのか，そのへんがまだ，見えないということですね．

それから，ツール自体も，Mac のツールとしてはまだ未完成で，操作のやり方にもいくらか苦しみました．オブジェクトの配置とか，並べかえとかは，人間の側で一生懸命努力しました．以上です．

青木： いま佐原さんがいわれたことは，ほとんどその通りだと思います．オブジェクト指向アプローチが他と決定的に違うのは，「まず，何かの処理を決めて，その入出力を見て,....」とかいうものではないということです．「状態を考える」ということでもない．オブジェクトの確定というところから始まる HOw To だけをみなさんにお示しし，それにしたがって仕事を進めてもらっています．

しかし，たとえば，私が最初に例題として取り上げた大学のモデル化ですが，そこに出てくる「教員」や「学生」というオブジェクトは，この問題領域には必ず存在する自明なものですから，簡単に定義できます．ところが，その構造関係を is-a とか has-a に分けて行くとき，その上に，ぽこっと「人」という抽象的なオブジェクトが出てくる．この抽象化プロセスにみなさん慣れていないように見える．

実際には，最初そうした抽象オブジェクトなしに属性やサービスを定義して行って，その後であちこちに同じものが存在することに気づいて，抽象化の階段を 1 段上がるわけですが，慣れていないとなかなかむずかしい．

そこで，グループのみなさんの心理状態としては，それとはなしに私の顔色を窺うんですね．そのことがこっちにはわかっているから，おもわず「ニヤッ」と笑うと，「うん，そうか」という感じでやっているのが現状です(笑い)．

それから，佐原さんがいわれたように，Smalltalk には豊富なクラスライブラリがあるので，それを有効に利用しないのは損だというのが，私のスタンスです．その知識が分析・設計のアプローチに影響を与えることは，やはり否定できない．ですから，私はあまり口出ししないようにしているのですが，私のそうした知識や経験をどんどん提供してほしいというメンバーも多い．むしろ，何とか聞き出そうとする感じです．

給湯システムでいいますと，ランプとかスイッチとかバルブというものが，みんなすべて開閉するものだ．つまり，開いているか閉じているか，いずれかの状態を必ず持っているということに気づいて，このバイナリステート(二値状態)という抽象オブジェクトを上に配置しました．この時，私がいったアドバイスは，「ランプだのスイッチだのというのは，そのプログラムを作った時の財産にはならない．会社の知的財産として残るのは，このバイナリステートという抽象クラスなのだ」ということです．まあ，現状はそんなふうに進んでいます．

落水： そうした上位のオブジェクトを見つけるというのは，How to じゃダメだと思います．訓練しかないと思いますが....?

青木： そうですね．

落水： 何か有効な訓練手段があるのですか？

青木： やはり，まずは全部作ってみることだと思います．そうすると，同じ情報があちこちに散在していて，いざシステムを動かした時に，何か変えたいと思ったら，それら全部いじらなければならないという悲惨な状況を，実際に何度か経験する．そのプロセスを踏まない限り，こういうノウハウはなかなか身につきません．私自身も，そうした経験を何度も繰り返してきています．

佐原： それから，また，Smalltalk の基礎文法に通じていれば，結果がうまく行くかどうかの見通しもつくのではないですか？

青木： それはそうです．これらがバイナリステートだと気がつけば，その上に Smalltalk のスイッチというクラスがあって，まさにぴったりあてはまる．そうすると，あとはもう簡単にシステムが動かせる状態に持って行ける．まあ，そんなふうにいうと，ずいぶん Smalltalk を意識しているなあ，と見られても仕方がないのですが，他の言語で，そういう抽象化のライブラリが提供されていない現状の方が，CASE ツール云々よりも問題だと思います．

野村： このまま行くと，またチュートリアルに戻ってしまいそうなので，ドクター・ストップをかけます(笑い)．あと他にコメントをいただいていないのは，天池さんのところですか？

天池学（カシオ計算機，CASE Bench グループ）： 先ほど落水先生から御指摘があったように，われわれのグループは，最初のコンテキストダイアグラムを作るさいに，全員の意志を統一するために，紙の上で作業しました．それは，別に私の独断でやったわけではなく，インストラクタの渡部さんと御相談して，「まあ，最初はいいんじゃないか？」という判断でしたが，渡部さん，そのあたりのことをちょっとコメントしてください．

渡部健次（ソニー・テクトロニクス，CASE Bench グループ）)： なぜ紙を使ったかということですが，それは 9 人ものメンバが全員で 1 台のワークステーションを使うのは，とうてい無理だったからです．もしマシンが 3 台あれば，3 人ずつグループに分かれてということもできたかもしれませんが....

とういう状況を考えて，最初のコンテキスト図は紙で作りました．多分 CASE ベンダの方はみなさんそう思っていらっしゃると思うんですが，日本語変換とか，ツールの使い方がよくわかっていれば，最初からマシンの前に座って，1 人で入力しながら考えるのがよいと思います．今日はそういう状況ではなかったということです．

野村： あと他に，何かいい忘れたこととか，いいたいこと，どこかのグループに聞きたいことなど，ありますか？

落水： せっかくですから，田中さんと桜井さんにも，ご感想とかコメントをお聞きたいのですが....

田中耕市（東陽テクニカ，Teamwork グループ）： 渡部さんも少し触れておられましたが，やはり，ツールを提供させていただく側としては，そのツールを教授しながら，お使いいただきたいと思っています．ただ，最初は操作になれるというところがありますので，今日はまだ最初の小競り合いということで，簡単なダイヤグラムを書くところまでしか行っていないということでしょうか....

落水： 実は，私はもっと悲惨な状況を想像していたのです．「方法論を理解する，ツールの操作法を覚える」というところで，一日の時間を消費して，とても設計どころではない．そんな状況を予想していた．ところが，これまでの報告をうかがうと，結構方法論をそれなりにこなしている．多分インストラクタその他の方の指導もよかったのかもしれませんが，方法論にしたがって作るということ，CASE ツールを使うということは，基本的にはそんなに難しくないんだなと感じました．もちろん，エキスパートになるには，ちょっと時間がかかるかもしれませんけども，昨日のチュトリアルからいきなり始まって，CASE ツールも初めてだという方が多かった割には，予想以上のスピードで成果が出ています．

桜井麻里（SRA，StP グループ）： われわれのところは，グループの人数を考えて，X-Window 端末を 2 台用意したのですが，ところが，そのうち片方の日本語入力の設定がうまくできていなくて，最初は，ツールを使わずに，紙の上で問題を考えていただき，その間にマシンの設定していようかなと思っていました．

しかし，メンバーの方々から「せっかくツールがあるんだから，使いたい」という強い希望が出て，また，「せっ

かく２台あるなら，やはり２台とも使ってやろう」ということになりました．グループを２つに分けると，双方それぞれ５人ぐらいですから，全員でワークステーションを囲んでも，画面が見えるんですね．最後にそれぞれのやったことを，お互いに発表しようということで，２つのサブグループで作業をしました．結局，片方の端末は漢字の設定がうまくいかなくて，アルファベットでの入力ということになりましたが....

一応，データフロー図を書きながらデータ構造図を書くというあたりは，みなさん体験できたにではないかと思います．私は，ある程度話が進み始めたら，ほとんど何もしないでボーッとしていたのですが，印象としては，このように１つのツールを何人かで囲み，１つの画面を見ながら討論し，入力するというやり方は一応うまくいったんのではないかと思います．

野村： それでは，さっきから手を上げていた佐原さん，どうぞ．

佐原： 構造化分析に挑戦したグループの方々のやり方を聞いていて，少し定石と違うなと思った点が２つあります．

１つは，構造化分析では，あらかじめ最後の具体化段階を予想して，「こうなりそうだから，こうしたらいいんじゃないか？」という議論をしないように，あるフェイズでは，あまり先のことを考えずに，そこだけで収束できるようにする．少しづつスライドして，そこである基準で決めたら，次におろしていく．そういう段階的詳細化をしている時に，具体化の時にまずいんのではないかなどと考えたりすると，考える範囲がどんどん広がって，大きなシステムになると，結局わからなくなってしまいます．そのあたり，ちょっと考え直した方がいいんじゃないかなと思いました．

もう１つは，プロセスをあらわす丸い記号，あれをポンポンポンと作って，データフローでつないでいるということですが，実は，それをやっていると，モジュール構造図を作っているのとほとんど同じになってしまう．つまり，記法は違うけど，従来通りのモジュール構造を作っていることになってしまうので，これもちょっとまずいのではないかと思いました．

構造化分析は，どちらかといえば，データ中心の方法論であり，そのデータフローを出すためには，何が必要かといえば，データフローの出力 t に対して，入力として何が必要とかを考えながら，それらをつないでいって，間に糊として丸印の記号がある．それらの丸印には名前を付けずに置いておき，一番最後になって，ここを流れるのはどういうものかを考慮して名前を付けるのが，たぶん定石だろうと思います．別に，定石をいつも固執する必要はなく，時にははずしてもいいと思いますけれども，基本的には，定石に沿った進め方をしたほうがいいのではないかなと思います．

落水： それはどうしてですか？

佐原： 理由は，いわゆるモジュール構造を作ってしまうと，結局，人間は，どうしても，うまく具体化ができるかどうか，ハードウェアの特性なんかを一生懸命考えてしまうんですね．そうすると，対象となっている１つのシステムだけを作るだけならいいのですが，そのシステムを他の環境に移植しようとか，他のシステムとの連携を考える場合にうまく行かなくなってしまう．しれないんです．

システムのハードウェア的な特性を早めに考慮するのは，そういう理由で，よくありません．

落水： わかりました．

野村： 何となく，みなさんもう課題の回答を出してこれで終りというような雰囲気になりかけていたところを，落水先生に修正していただいて，このワークショップの本来の目的が何かということが，明確になったと思います．その意味で，貴重な議論だったのではないでしょうか？

あと最後に何かいいたい方があれば，話をうかがいます．なければ，これで議論を打ち切りたいと思います．

落水： １つお願いがあります．いまのプレゼンテーションで使われた OHP シートに，グループの名前と通し番号を記入していただいて，提出してください．記録のためです．よろしくお願いします．

佐原： 通し番号の他に時間を書いておくといいですね．何日の何時ごろとか....

野村： では，このセッションを終りにします．６時半にわんこそば大会に出発します．

6. 最 終 発 表

最 終 発 表

報告者
下津 直武・方 学芬
（静岡大学）

1. PWB & Hyperbook グループの発表

われわれのグループでやってきたことは，図 1 に示す状態遷移図を作ったということにつきます．その製作過程で，われわれの CASE ツールである PWB の持つ強力なアニメーション機能を利用し，内容を序々に洗練させながら，ここまできました．

だいたいの経過を説明すると，初日は，まずツールの説明を受け，午後半日ぐらいでこの状態遷移図を作りました．次の日の午前中に入力を始め，午後にかけてアニメーションができるような段階までもって行きました．

状態遷移図は，御存じのように，システムがある状態のときにあるイベントが起こると，システムはどんな挙動を示しどの状態に移行するかということを，一枚の絵に書いたものです．

われわれが選択した課題は給湯システムで，図 1 が，システム全体の状態遷移図を示します．「給湯」,「お湯はり」,「追いだき」などの状態があって，それらの状態において,「お湯はりスイッチを ON にする」とか,「給湯栓をひねる」とかいって外部イベントが発生する可能性があります．そうしたことすべてを表したのがこの図です．

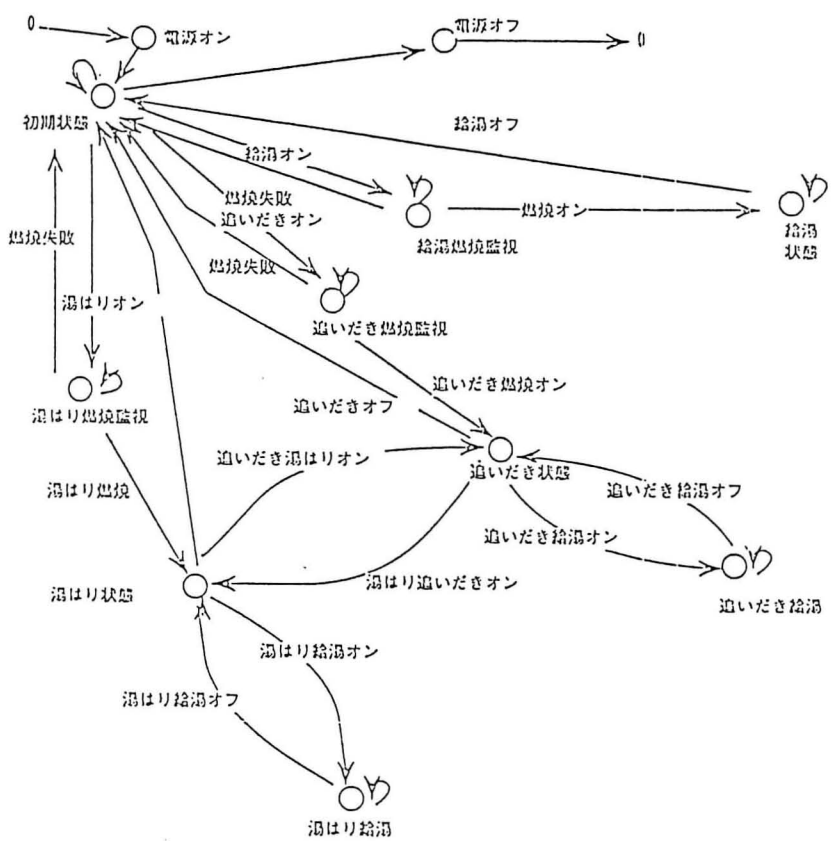


図 1　給湯システムの状態遷移図 (PWB)

少し小さくて見にくいので,「追いだき」状態の部分を中心に拡大したものを，次にお見せします（図 2)．もちろん，状態遷移図を作るにあたっては，課題に述べられた仕様をまず第一基準にして，あまり明確に示されていない部分については，全員で討論して，こういうふうにしようということを決めました．

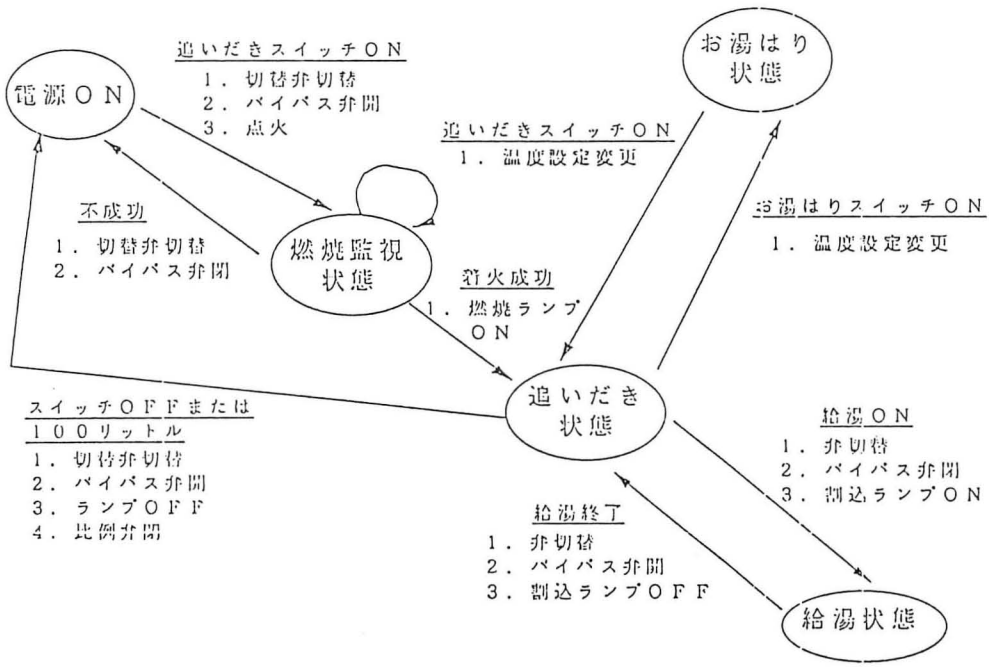


図 2　追いだき状態 (PWB)

さて，中央の２つの状態（燃焼監視と追いだき）ですが，これらの状態に遷移することが可能な状態としては，初期状態があげられます．また,「お湯はり」,「給湯」の各状態からも，ここに遷移する可能性があります．まず，電源 ON だけの状態，つまり初期状態にあるときに追いだきスイッチが入ると，切り換え弁を切り換え，バルブ弁を開き，点火操作を行って,「燃焼監視」の状態に行きます．

この状態では，本当に火がついたかどうかを見ていて，もし火がつかなければ初期状態に戻ります．そのときには当然，切り換え弁を切り換えてバイパス弁を閉じるという操作をします．着火成功であれば，燃焼ランプを ON にして，追いだき状態に行きます．

この状態では，温度を 93C の一定値に保ってお湯を出し続けます．ただ，このときにスイッチが OFF されたり，あるいは 100 リットルを越えたら，切換弁を切り換えてバイパス弁を閉じ，ランプを OFF にし，比例弁を閉じ，火を消して，初期状態に戻ります．

一方，追いだき状態でお湯はりスイッチが押されれば，今度はお湯はり状態に移行します．そこで追いだきスイッチが ON されれば，再び追いだき状態に戻ります．この状態のときに給湯，すなわち蛇口が開かれれば，給湯状態に移行します．給湯が終わる，つまり蛇口の栓が閉じられれば，また当然追いだき状態に戻ります．

設計にあたって，仕様書に定義されていない不明確な部分は，一応安全サイドでものごとを考えるようにしました．この給湯システムでは，最高 93 度というお湯が出ます．給湯栓を開いたとき，いきなり 93 度のお湯が出たのではたいへんですよね．そんなシステムを，もしソ連に輸出したりすると，また火傷の治療のために札幌から飛行機を飛ばさなければならなくなるので（笑い)，そういう事態は極力避けたいということです．

たとえば，燃焼監視という状態が設けてありますが，この状態のときに追いだきスイッチが押されたり，給湯栓がひねられるという可能性も十分考えられます．しかし，この段階では，点火操作の後に火がついたかどうかを確認することが主要な動作であり，また，時間的にいえばそういうイベントが来る可能性は非常に少ないので，もしそういうイベントが来ても，それを無視してまた自分の状態に戻るように考えています．それから，電源がいきなり OFF にされるとか，停電になってしまうとかいう場合もあり，本当にこのシステムを作るのであれば，そのへんも考慮しなければいけないのですが，今回はそこまではやりませんでした．

デモ用のマシンがここにおいてあります．百聞は一見に如かずですので，後で見てください．その前にちょっと簡単にデモの内容，つまりアニメーションがどういう感じになるかを説明しておきます．

全体の状態遷移図の上で，まず入り口から電源 ON の状態に入って，次に初期状態に移ります．この状態において，いろいろなイベントが起こる可能性があります．アニメーションにさいしては，現在の状態を示す黒い丸がピピピッと動いて行ってこの状態に来ると，１つのウィンドウが開き，その中に可能なイベントがリストアップされています．そこで，たとえばマウスを給湯のところに持って行ってクリックすれば，給湯燃焼監視の状態に移るわけです．

そこでもまたウィンドウが開かれます．このウィンドウに示されたイベントは，実際に燃焼しているかあるいは燃焼に失敗したかということで，これは本来ならばシステムが適当なセンサーによって判定するのですが，アニメーションでは人間が選択して入力します．燃焼失敗の場合は初期状態に戻りますが，ここでもし燃焼 ON をクリックすると，次に給湯状態に移ります．するとまた新しいウィンドウが開き，その中から適当なイベントを選んで先に進むといったぐあいです．

最後にまとめですが，回われわれのグループで使った方法論は，状態遷移図によってシステムの構造を把握し，分析しようというものでした．グループ・メンバの中で，そういう方法論に慣れていた人間はほとんどいなくて，初めはずいぶんとまどいました．あらかじめ勉強しておくべきだったと反省しています．つまり，どういう状態をまず切り出すか，そのときの外部イベントとして何を考えたらよいか，このへんでちょっと時間がかかりました．それと，グループでの仕事の進め方ですが，いきなり最初に３つのグループに分けて，給湯，お湯はり，追いだきの３つの部分を解析しようということだったのですが，さきほど述べたような事情から，あまりうまく行かなかったかなあというのが感想です．

次に，PWB ツール自体の評価ですが，利点から列挙していきますと，まず，undo が何段階か使えること，当然それに対して元に戻すための redo が使えることが，まちがいを犯しやすいわれわれのような初心者に対しては，有用だったと思います．

それから，分析段階での検証支援も有効でした．われわれの場合は，一昨日足立さんが説明されたユーザ向けプレゼンテーションというよりは，システム設計者自身によるレビューというか，設計作業の成果の検証に使ったわけですが，そういう意味で非常に有効でした．アニメーションがレビュー時のチェック漏れをなくしてくれます．それもほとんどの仕事を自動的にワークステーションがやってくれるので，ずいぶん楽だなあと感じました．

それから，状態遷移図を書くのに使ったエディタ EOPM と，スイッチを書くエディタ UIM についてですが，制御系のシステムの設計には有効だなと感じました．マウスでコマンドが選べるというか，ウィンドウ・ドリブンになっていて使いやすいし，また，エディタ自身が文法を知っていて，誤った入力をチェックしてくれるので，記法とかをあまりよく知らない初心者にも使えます．また，このツールは Smalltalk の上で動くのですが，Smalltalk を全然知らなくても使うことができるようになっています．

さて，次に欠点ですが，UIM エディタの機能がそれほど強力でなく，つまり簡単な図しか作れないことがあげられます．それから，チュートリアルというか，ツール自身のガイダンス機能が用意されていないので，初めて使う人にはちょっと使いづらいように思いました．他にもいろいろ欠点はあるでしょうが，まだそれが見えるまで使いこなしていないというのが本音です．

最後に全体的な感想を述べさせていただくと，今回の演習では，アニメーションが使えてよかったと思うのですが，実際の場面でどれだけ有効かはまだよくわかりません．今回は制御系のシステムを課題に選んだのですが，事務処理系とかバッチ系などに，この分析・設計アプローチがうまく適用できるかどうかは疑問です．

それから，ちょっと思ったのですが，状態遷移図はまあ何とか書けます．すると，当然そこには，状態とか外部からのイベントが全部記述されているので，それをもとにして状態遷移表を自動的に作ってくれて，その表の中を全部埋めると，それをもとに C プログラムのスケルトンを出してくれるようなツールがあれば，非常にうれしい．

状態遷移図の形に作ったモデルの検証を自動的に支援するシステムも欲しい．今は，とにかくモデルを作り，アニメーションで動かして，おかしければ人間が直すというスタイルですが，たとえば，仕様を最初に打ち込んでおくとそれがルールとしてシステムに記憶され，設計者が書いた状態遷移図を，システムがそのルールにてらし合わせてチェックし，ここは違うんじゃないのと自動的に注意しくれるようであれば，もっと便利ではないか

と思いました．まあ，わがままな意見ですね．

ツールのメリットを十分に知る前にタイムアップになってしまったので，PWB そのものの設計意図をすべて吸収できなかったのが残念です．まあ，今回は時間も２日間しかなかったし，使ったエディタも１つだし，課題も１つしかこなせなかったので，それはやむを得ないかもしれません．

以上で発表は終わりですが，せっかくワークステーションがありますので，是非われわれの成果を目で見ていただきたいと思います．デモ自体は短時間で終わりますので，２～３グループに分かれて見ていただけばよいでしょう．

Q（高橋秀行，SRA）：ふつう，状態遷移図では，ある状態から別の状態へ遷移するとき，どんなトリガでその遷移が起こり，どんなアクションが実行されるかを記述するはずですが，いま見せていただいた図では，イベントの名前のようなものが書いてありますね．PWB で，この種の状態遷移図を作るときのルールはどうなっているのですか？

A（小林透，NTT）：丸印は１つの状態を示し，その下に書いてあるのが状態の名前です．それらをつなぐ矢印のまん中に書いてあるのは，すべてイベント名です．作成の手順は，まず，丸印の状態をどんどん書いて行き，その後でそれらの間のリレーションを張り，そのトリガとなるイベントの名前を記入します．そうすることによって，さっきのアニメーションを実現できるわけなんです．それぞれのイベントがどんな機能を果たすかは，アニメーションに必要がないので，ここでは書きません．

2. OOA/OOD グループの発表

われわれが使ったのは，Smaltalk-80 上の OOA ツールで，分析対象は前のグループと同じ給湯問題です．一番最初にツールの解説をしていただき，このときに OOA ツールとか，Grapher Gear のデモを見せていただきました．Grapher Gear はすばらしいものだと感じましたが，実際の作業では OOA ツールを使っただけです．

どんな方法論かは，すでに初日に説明がありましたが，図 3 に示すように，オブジェクトの確定，構造の確定，属性･サービスの定義という手順で仕事を進めて行きます．中間発表のときに，この OOA のフェーズ 1 から 5 までを実施することを目標にしたいとお話しましたが，そのあと佐原さんから OOD までやらないと，その効果がわからないだろうというアドバイスを受け，希望としてはそこまでは行きたいと考えていましたが，実際にできたのは OOA だけでした．

OOA	OOD
1. オブジェクトの確定	1. ハードウェアアーキテクチャの決定
2. 構造の確定	2. ソフトウェアアーキテクチャの決定
3. サブジェクトの定義	3. 特殊化
4. 属性の定義	4. 組み立て
5. サービスの定義	5. 抽象化
	6. 要素分解
	7. サービスの詳細化

図 3　OOA と OOD

OOA の各フェーズをたどって行く過程で感じたことがらをいくつか挙げてみます．

第１は，オブジェクトの確定についてです．このフェーズでは，問題文中に出てくる名詞をキーワードとしてオブジェクトにして行くのですが，それが本当に必要なものなのかどうかの判断に，ずいぶん悩みました．中間発表のときに浴槽というオブジェクトの候補があり，その取り扱いがよくわからないという話をした覚えがあります．このように，候補を挙げるのは簡単ですが，それが本当に必要かどうか，判断に困りました．

オブジェクト指向の「ぐにゃぐにゃ」した雰囲気（笑い）が好きで集まったグループだったのですが，割に一生懸命突き進む雰囲気があり，適当なとこでけりをつけることができませんでした．そういうことで，ここに時間かけすぎてしまいました．グループ内の同意を求めすぎたことも問題です．佐原さんからも指摘されました

が，「いいよね，いいよね？」といってるとなかなか先へ進めません．

次は構造の確定についての問題です．構造の確定というのは，オブジェクト間にどういう関係があるのかを知ろうとする活動です．このフェーズでの問題点は，機械の見た目にごまかされたということです．今回の課題は給湯システムだったのですが，スイッチとかコントローラのイメージをなんとなく想像していたのですけれども，分析のとき，外から見たイメージにとらわれすぎてしまって，そういう状態で関係を定義して行くのはなかなか大変でした．

抽象化がうまくできないし，やり方がわからないんです．初日に，バイナリステイトという抽象化の案を出したとき，このときだけ青木さんが笑顔を見せてくれたので，うまくできたと思ったんですが，逆にのぼせてしまったというか，ものすごくそれにとらわれてしまって，他の面ではちょっと力不足の結果になったと思います．

また，オブジェクト間の is-a，has-a という関係を定義していく段階で，１の「オブジェクトの確定」の不備に気づいて後戻りするということもありました．それから is-a，has-a の関係を結ぶのに，is-a のほうはそれほど難しくなかったのですが，has-a の関係を定義するのは，ずいぶん難しかったなあと全員が感じました．その原因は，オブジェクトの確定が不十分だったというか，自信が持てなかったということがあると思います．

サブジェクトの定義は必要がなかったので，やっていません．

最後のフェーズとしては，属性定義とサービス定義をやりました．ここまでは「これはこれでいいかねぇ？」という風に，みんなの同意を求めながらやってきたのですが，それではなかなか作業が進まないので，１人１人がまず考えて，それぞれのアイデアを出し合って討論する形に切り替えました．

このフェーズでの問題点は，構造の確定が各自まちまちであったこと，そして，何が属性かわからなかったことです．バイナリステイトの属性とはいったい何だろうと考え出すと，あとどうしょうもなくなってしまう．また，いったんオブジェクトとして挙げたものが，別のオブジェクトの属性になってしまうというようなこともあり，よくわかりませんでした．

サービスの定義は，属性の定義より簡単でした．いったん消したオブジェクトがまた復活することもあり，これを僕なんかは「湯釜の復活」と呼んでいるんですが，１回湯釜を消したんですが，途中でやはりそれがどうしても欲しくなって，復活させたというようなことがありました．また，サービスの属性を定義するとき，単にツールで提供されている枠の中に，言葉を埋めていくだけのレベルで仕事が終わってしまったという反省があります．それから，悪戦苦闘しているときに，青木さんがペトリネットとか，GrapherGear のシミュレーションをしてくれて，頑張れよとはげましてくれました．

ツールと方法論についての感想ですが，線を自動的に引いてくれれるのですけども，それが見にくかった場合，きれいに直すのがなかなか大変でした．今回使ったのは，ツールのごく一部の機能でしかなかったので，全体の感想は述べられません．抽象化については相当苦しんだのですが，そういうところでは，やはり青木さんがいっておられましたけども，第６感みたいなものが必要で，それがなかなか働かなかったという気がします．方法論の有用性については，まだそれをうんぬんするまでの経験を積むには至りませんでした．

記録係の方の感想は，「ツールを使うというよりも方法論に振り回されていたみたいだ」ということと，「フェーズの切り換えがうまくできない，ある程度のところで止めることができずにズルズルやってしまっている」といったところです．「えらいグループに入ってしまった」という方もいました（笑い）．

で，われわれの最終成果は図４に示す通りです．時間がないので，詳細の説明は省かせていただきます．

Q（発言者不明）： さっきオブジェクトが属性に変わったという話がありましたが，具体的に何がどうなったのですか？

A（発表者）： 割り込みランプは１つのオブジェクトなんですが，同じものが別のオブジェクトの属性のところに入っているということです．

Q（佐原伸，SRA）： 属性の中にランプが入るのではなくて，ランプの種類という属性があって，それが何々ランプという値を持っているというほうが普通ですね．

A（発表者）： そういわれるとそんな気がしてきます．

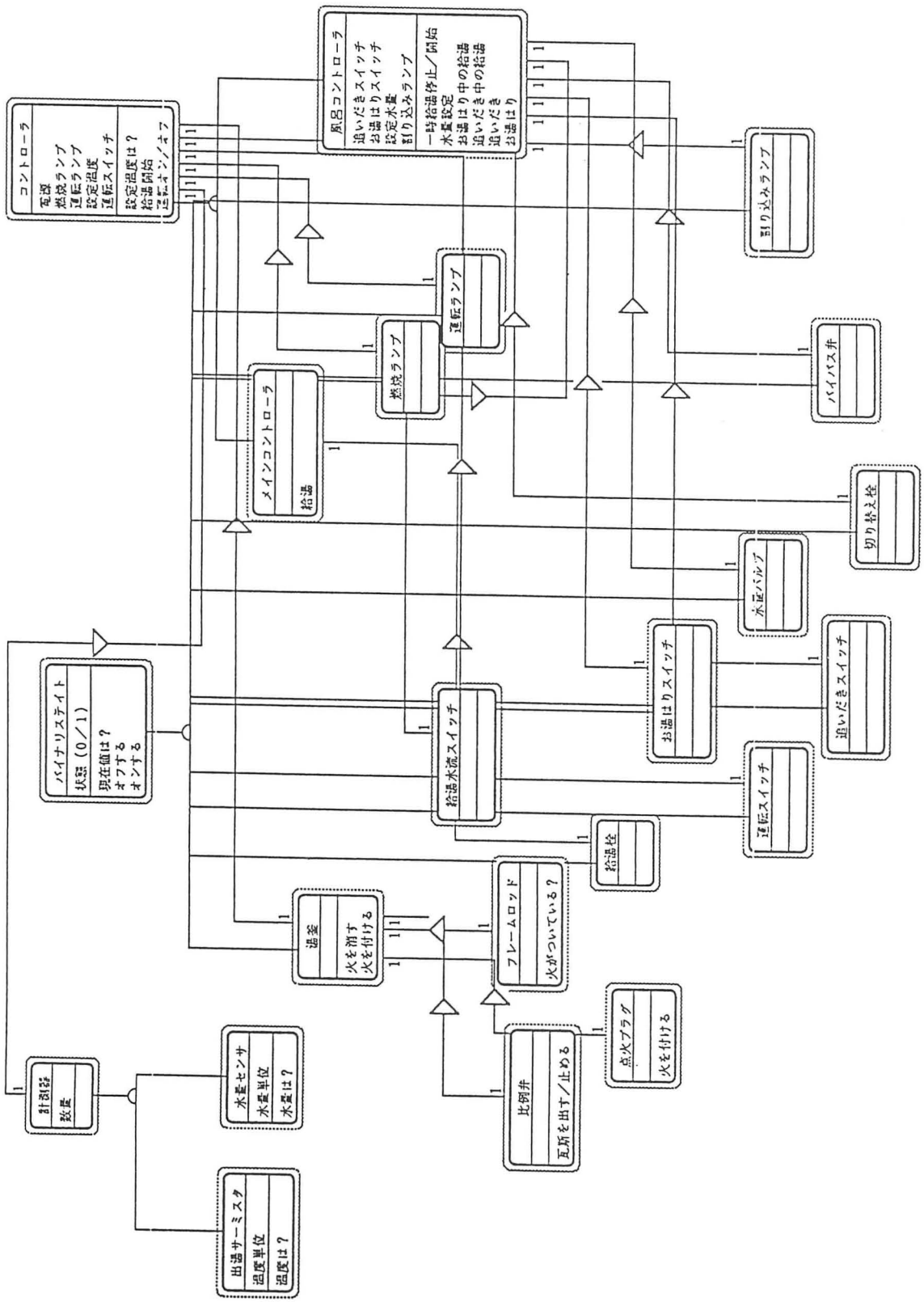


図 4 給湯システムの OOA ダイアグラム (OOA/OOD)

3. CASE Bench グループの発表

われわれが経験した試行錯誤をみなさんにいっしょに追っていただいて，その過程で CASE ツールがどんなふうに役立ったかをお話する予定でいたのですが，時間の関係で結果中心の報告に変えさせていただきます．分析対象は酒屋倉庫問題です．

われわれが使った CASE Bench は構造化分析のツールですので，DFD を作るのが１つの大きな目標になります．最終的なコンテクストダイアグラムは図 5 のようになったのですが，その作成にあたっては，まずデータを抽出しようというアプローチをとりました．これはたいへんうまく行って，コンテクストダイアグラムはかなり簡単に作成できました．名前の付け方でちょっと苦労はありましたが，まずまずスムーズに決まりました．

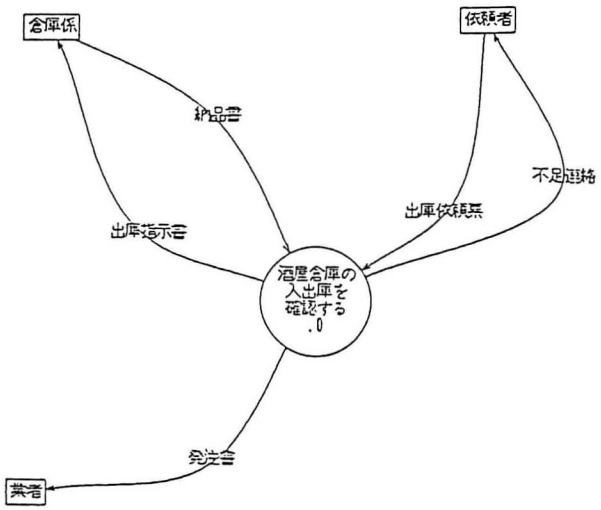


図 5 酒屋倉庫の DFD － 全体 (CASE Bench)

次に詳細化のステップですが，図 5 の「酒屋倉庫の入出庫を確認する」というバブルを２つに分けると，図 6 のようになります．

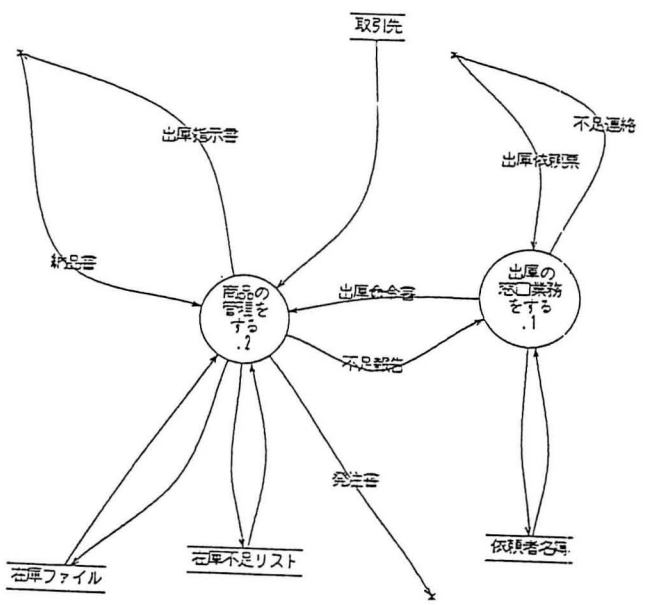


図 6 酒屋の入出庫を管理する DFD (CASE Bench)

さらに，その中の「商品を管理する」というバブルを詳細化したものが図 7 です．ここで，この図で「出庫を管理する」というバブルの入出力がかなり多いので，それを図 8 のように分割しました．ここまで詳細化が進んだとき，ER 図を使ったデータの分析と，データ辞書登録の作業を行いました．この作業はなかなか難しく感じました．

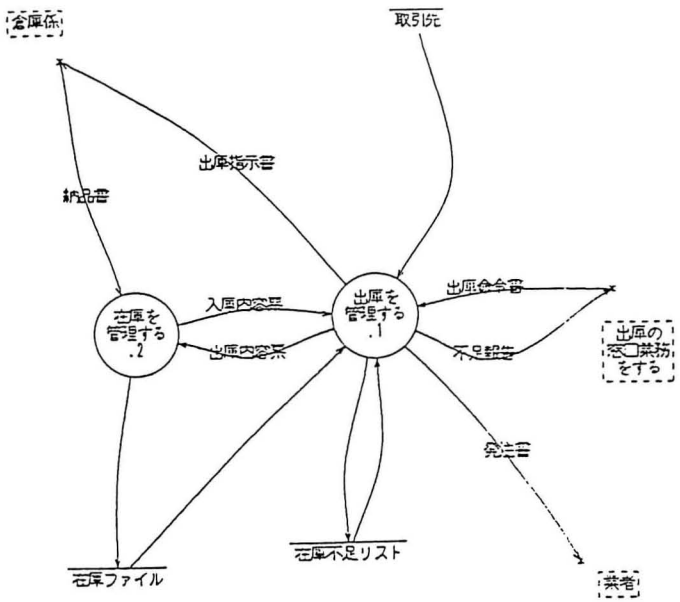


図 7 商品を管理する DFD (CASE Bench)

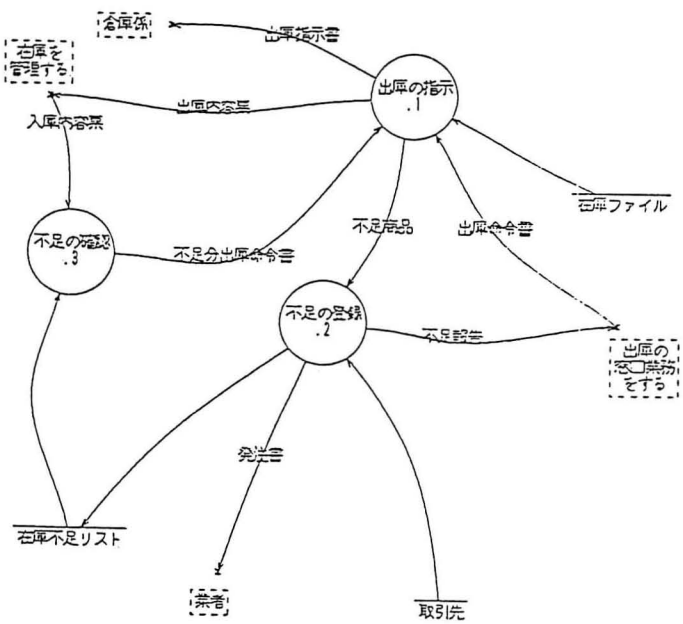


図 8 出庫を管理する DFD (CASE Bench)

結局こういう事務処理の問題では(というより，そもそも構造化分析では)，データが非常に大切なので，まず ERD を使って，エンティティが何かをはっきり把握してから作業を始めるべきだと思います．また，データディクショナリの作成にさいしては，1 つのコンテクストを作ったら，すぐそれに関するデータディクショナリ登録を行なうようにするのがよいと思います．機能分割という面では，階層化のやり方について，いろいろとわからないことがありました．

次の問題は，各階層でのファイルの扱い方です．図 6 のレベルで，すべてのファイル（在庫ファイル，在庫不足リスト，依頼者名簿，行き先）を書いてしまったのですが，「商品の管理をする」というバブルの意味合いがはっきりしていません．取り引き先との具体的な関係がこの図からは読めないのです．もちろん，さらに下のレベルへ落ちていくにしたがって，詳細はわかってくるのですが....

完成したデータディクショナリと ER 図は，図 9 および図 10 のようになりました．

```
依頼者名簿 =  依頼者番号 + 依頼者名 + 依頼者住所 + 依頼者電話番号
在庫不足リスト =  送り先名 + 品名 + 数量
取引先 = 取引先名 + 取引先住所 + 取引先電話番号
出庫依頼票 = 品名 + 数量 + 送り先名
出庫指示書 = 注文番号 + 送り先名 + { コンテナ番号 + 品名 + 数量 + 空コンテナ搬出マーク }
納品書 = コンテナ番号 + { 品名 + 数量 }
発注書 = 品名 + 数量
不足連絡 = 品名 + 数量
在庫ファイル = 品名コード + 品名 + { 数量 + コンテナ番号 }
出庫内容票 =  品名 + { コンテナ番号 + 数量 }
出庫命令書 =  品名 + { コンテナ番号 + 数量 }
入庫内容票 = コンテナ番号 + 搬入年月 + 日時 + { 品名 + 数量 }
不足商品 =  注文番号 + 送り先名 + 品名 + 数量
不足分出庫命令書 =  品名 + { コンテナ番号 + 数量 }
不足報告 = 品名 + 数量
送り先名 = 依頼者名
コンテナ番号 = "数字" *コメント*
依頼者住所 = "住所"
依頼者電話番号 = "電話番号"
依頼者番号 = "数字"
依頼者名 = "氏名"
空コンテナ搬出マーク = "記号"
取引先住所 =  "住所"
取引先電話番号 =  "電話番号"
取引先名 = "会社名"
数量 = "数字"
注文番号 = "数字"
品名 = "銘柄"
品名コード =
```

図 9 データディクショナリ (CASE Bench)

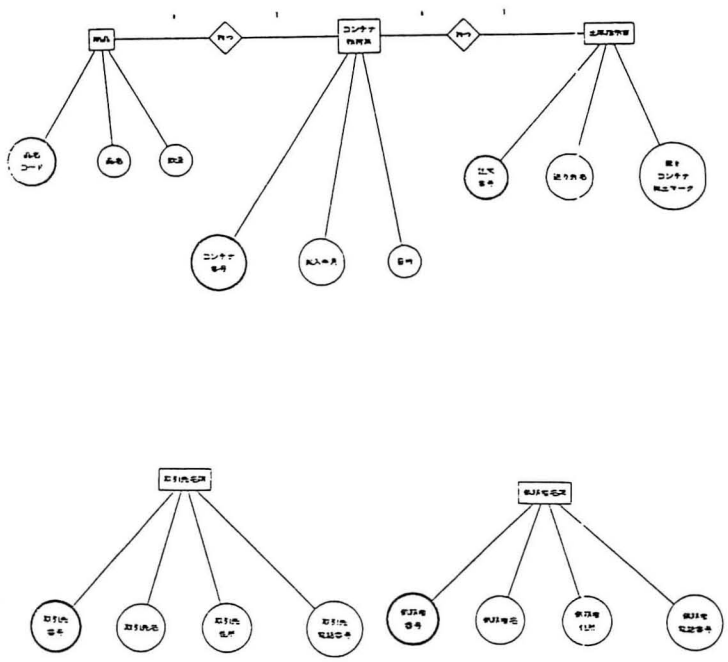


図 10 ER 図 (CASE Bench)

CASE Bench の 1 つの強力な機能に，エラー・チェック，つまり DFD やデータディクショナリが正しいかどうかを評価してくれる機能があります．図 11 はその結果を示します．

```
DD `dd' を評価中です。
重複して定義されている名前:
定義文が`未定義'となっている名前:
DFD中に存在しているにもかかわらず定義されていない名前:
他の定義文やDFD中で使用されていない名前:
定義文中で現れ、再定義の必要のある名前:
        コンテナ番号          依頼者住所          依頼者電話番号
        依頼者番号            依頼者名            空コンテナ
        取引先住所            取引先電話番号      取引先名
        数量                  送り先名            注文番号
        搬出マーク            品名                品名コード


DFD `dfd0' を評価中です。
        `dfd0'の評価中に検出された定義の無いフロー要素:
          コンテナ番号          送り先名            コンテナ番号
        依頼者住所            依頼者電話番号      依頼者番号
        依頼者名              空コンテナ          取引先住所
        取引先電話番号        取引先名            数量
        送り先名              注文番号            搬出マーク
        品名                  品名コード
    データ項目`倉庫の 在庫を 管理する'に対し5本を越えるデータフローが存在しています。
        データ項目`商品銘柄'が読み出し専用になっています。
        データ項目`取引先'が読み出し専用になっています。
        データ項目`依頼者名簿'が書き込み専用になっています。
        データ項目`在庫不足リスト'が書き込み専用になっています。
```

図 11 エラー・チェック結果 (CASE Bench)

上半分は，データディクショナリに対するチェックの結果です．定義文中に現れ再定義が必要である名前のリストが出力されています．下半分は DFD に対する評価です．これはと同じ時点でのものですから，先ほどと同じエラーメッセージがまず出ています．もう一つのエラーメッセージは，「あるバブルに対し５本を越えるデータフローが存在する」というもので，バブルをもう一段分解すべきだという警告です．

次の手順としてはミニスペック作るはずだったのですが，時間が足りなくて，途中で終わりました．図 12 は，DFD から自動的に作成された構造図です．これはコマンドを３つ打ち込むだけでできてしまいます．

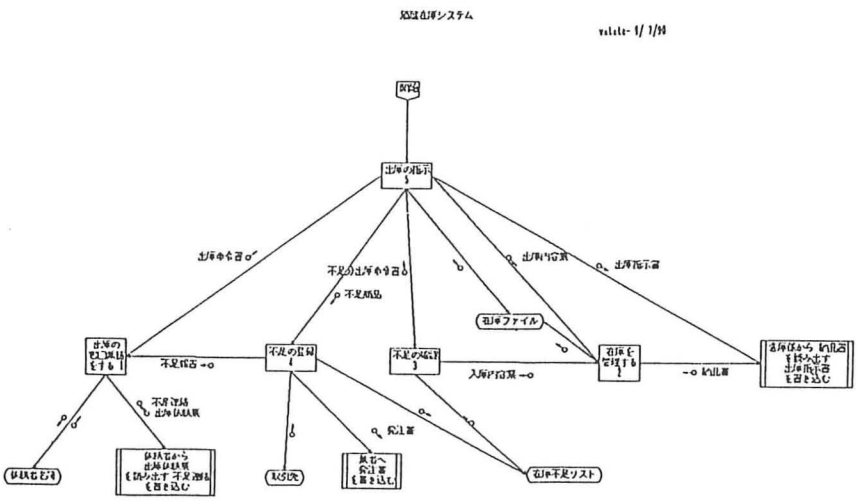


図 12 モジュール構造図 (CASE Bench)

この図には，まだいくつかの問題が残っています．「出庫の指示」モジュールからいくつかの下位モジュールを呼び出していますが，それらの下位モジュール間に相互呼び出しの関係があるとういう点です．にやっていただいたきました．

図 13 はモジュールのミニ仕様書の一部です．これをもとに，C プログラムの枠組みが作れます．中身が空でパラメータだけのスケルトンができてきます．

```
出庫の指示 5 ()
  在庫を 管理する 2 (入力カプル: `出庫内容票')
    在庫ファイル (入力カプル: `')
    倉庫係から 納品書 を読み出す 出庫指示書 を書き込む (出力カプル: `納品書')
  在庫ファイル (出力カプル: `')
  出庫の 窓口業務 をする 1 (出力カプル: `出庫命令書')
    依頼者から 出庫依頼票 を読み出す 不足連絡 を書き込む (出力カプル: `出庫依頼票', 入力カプル: `不足連絡')
    依頼者名簿 (入力カプル: `', 出力カプル: `')
  倉庫係から 納品書 を読み出す 出庫指示書 を書き込む (入力カプル: `出庫指示書')
  不足の確認 3 (出力カプル: `不足分出庫命令書')
    在庫を 管理する 2 (出力カプル: `入庫内容票')
      在庫ファイル (入力カプル: `')
      倉庫係から 納品書 を読み出す 出庫指示書 を書き込む (出力カプル: `納品書')
    在庫不足リスト (出力カプル: `')
  不足の登録 4 (入力カプル: `不足商品')
    業者へ 発注書 を書き込む (入力カプル: `発注書')
    在庫不足リスト (入力カプル: `')
    取引先 (出力カプル: `')
    出庫の 窓口業務 をする 1 (入力カプル: `不足報告')
      依頼者から 出庫依頼票 を読み出す 不足連絡 を書き込む (出力カプル: `出庫依頼票', 入力カプル: `不足連絡')
      依頼者名簿 (入力カプル: `', 出力カプル: `')
```

図 13 ミニスペック (CASE Bench)

時間がなくなったので，あと少し感想を述べて終りにします．

まず第１は，思ったより作業が進まなかったということです．１日目は簡単に進んで，この分では明日の昼にはできあがるなとかいってたんですが，いろいろな試行錯誤（やり直しとか）が発生して，予想通りには行きませんでした．

２番目にツール操作ですが，これはもう大変簡単でした．

それから，３番目ですが，CASE ツール以前の問題が多かったように思います．方法論の習熟度が低かったための誤りやムダが多かったように感じました．もっと事前にみんなが勉強してきていれば，より大きな成果がえられたのではないかと思います．

それから，人数が多すぎました．これは，どこのグループでもそう感じられたと思います．CASE ツールを使ったという満足感が得られなかったのです．システム分析というのは非常に感動的な仕事で，何かごちゃごちゃしたもの，どろどろしたものが，ある瞬間にパッときれいにまとまって，「そうだ，これで行こう！」という感動があるはずだなのですが，今回は，何かいちおうできあがったんですけれども，「ああ，こんなもんか！」という程度の感覚でした．それが方法論のせいなのか，人数が多かったせいなのかは，わかりませんが.... ですが，「もっとツールを使いたかった」，「成果物の検討をもう少しやりたかった」，「プリントアウトの操作が不便だった」等の感想も出ました．これは，人数が多かったせいで，レビューのためにいちいち中間結果をプリントアウトするのですが，それには，いったんエディタから抜けてコマンドを打ち込む必要があり，そのあたりもう少し楽にできないのかなという意見です．あと「いまの自分の仕事には生かせない」という感想もありました．

最後に，私自身の個人的意見をいわせていただくと，構造化分析の方法論自体がほんとうに役立つのかどうかについては，研究者の方々はいろいろおっしゃっていますが，まだ確信を持つにはいたっていません（笑い）．た

だ，もしこの方法論を使うということが決まった場合には，ツールというのは大変ありがたい存在だと感じました．

Q（奥村吉彦，CSK）： CASE Bench そのものに関する質問です．１つのバブルに出入りするフローについて，たとえば５つとかいう制限を設けているというお話でしたが，それはどういった仕組みになってるのですか？

A（渡部健次，ソニーテクトロニクス）： ５個というのは，あくまでデフォルトの値にすぎません．ユーザが自由に１０個とか２個とかに変えることができるようになっています．

Q（奥村吉彦）： 今回与えられた演習問題のシートは，一種のインタビューシートのようなものだと思います．それを DFD のような道具を使って整理して行く過程で，問題のいろいろな側面が見えてくる．整理が一段落したら，それをモジュール構造図に落とし，さらに C プログラムのスケルトンを生成するという手順ですが，その過程で，手作業でやる部分と，コンピュータで自動化する部分との切り分け分けが必要になってきます．そのへんはどうしたらいいのでしょうか？

A（渡部健次）： 何の切り分けですか？

Q（奥村吉彦）： マニュアルでやる仕事とコンピュータ化できる仕事との切り分けです．

A（渡部健次）： コンピュータ化というのは，ツールによる機械的チェックとか，自動生成とかいうことですか？

Q（奥村吉彦）： そうではなく，対象業務の中でどの部分をコンピュータ化するかです．何らかの区分けが必要でしょう？ 今の発表だと，全部が最後にはソースコードになってしまうような印象ですが，そうではないんじゃないかと思うのですが....

A（渡部健次）： 一応，トップダウンで DFD を作り，仕様を書いて構造図を作るという設計プロセスが，上から下まできれいに流れることを前提として，ツールが作られているんですが，実際には，もちろん，そんな風には仕事は進まないわけで，要求仕様が完成したら，自動的に設計ができるなんてことはありえません．

ですから，あそこで出てきた構造図のスケルトンは，大まかな設計書のひな型だと思ってください．ところで，その構造図の源になってるのは要求仕様そのものなので，それは，一応構造図の形はしていますけれども，要求仕様のある表現だと見ていただきたいのです．ですから，最終的には構造図をエディタの上でバラバラに分解していただいて，そこで人為的な，おっしゃるような，つまり大まかな設計書の完成という作業が次に始まります．それからスケルトン的なものを作るということになります．

Q（奥村吉彦）： モジュール図まで行ってから．そこで選択するわけですか？ たとえば，その制御構造を見て，たいていマニュアルでやる部分，機械化する部分に分けますよね．その後で，機械化する部分をモジュール構造まで最終的には落としていくんですけど，そこあたりで，どの部分を機械化するかを選択することによって，全部がソースコードになるというわけじゃなくて，その見切りができるような仕組みは別にないのでしょうか？

A（佐原伸）： 難しいですね．CASE ツールといっても，すべて自動的に判断してくれるようなことは期待できません．どの部分を構造図に落とすべきかどうかは，最終的には人間が判断せざるを得ないでしょう．いまから 100 年後はどうか知りませんが，現在の CASE ツールには，そんな仕掛けはありません．

Q（奥村吉彦）： 私がいいたかったのは，とりあえず全部自動化するということを前提として分析を進めるとして，ある時点で人間がチェックをかけて，それに従っていく部分をどこに入れたらいいのかなということです．

A（発言者不明）： 私も事務系のシステムをやっていますが，分析というフェーズでは，ある程度業務の全体が見えています．そして，分析対象業務の中でここを機械化しようというターゲット，つまりドメインも決まっていて，それを詳細に分析して行くと，当然機械化すべきところもあるし，人間の手作業が必要な部分もある．

つまり，分析の結果を DFD に落として行ったときにできあがるバブルの中には，もしかしたら機械化しなくてもいいプロセスが交じっています．DFD をそのまま構造化チャートに変換すると，それぞれのバブルから１つのボックスができ，入力と出力が決まり，ミニスペックも，さらにはモジュール仕様書もできてきますが，必ずしもそれらをすべてコンピュータ・プログラムにするわけではなく，どこかで切り分けをしなければなりません．

そういう意味でいえば，DFD から構造化チャートへの変換は，ある種の論理的モデルの作成であって，実際

に開発すべきソフトウェアの構造を表す物理的モデルを作る作業は，それとは別に必要だというような気がします．

Q（天池学，カシオ計算機）: CASE Bench にはそういう切り分けを支援する機能があるのかということですね？

A（渡部健次）: われわれのツールはただ，ふつうの構造化分析の手順を機械的に支援するだけのものです．デマルコの教科書に書かれているのは，何をどう変換するか，あるいは構造図に落とすか，それをどうフォーマットしどう分解するか，そういった手続きだけです．構造図が意味しているのは，あるモジュールの機能が全体の中でどのように位置づけられるかということだけであって，それをそのままコードに落としていいかどうかということではありません．構造図の中のマンマシンのバウンダリを自動的に分析し判断してポンとプログラムに変換する，そんな夢のようなツールは，まだまだ当分でてこないと思いますが....

Q（奥村吉彦）: 渡部さん自身は，そうした自動化の方向はそれほど重要だと考えていないんですか？

A（渡部健次）: ツールを使う人間のほうがちゃんとわかっているわけですから，切り分けて使えばそのように使えるということでしょう．論理モデルと物理モデルとの区別は，あくまでも人間の側の認識の問題で，どんなツールを持って来ても不可能だと思います．す．

Q（奥村吉彦）: 実際に，私自身がシステム分析とか要求定義とかをする場合には，まず最初は物理的に見て現状がどうなっているかをモデル化しします．その結果を眺めながら，ここはこういうふうに変えた方がいいのではないかという形で，新しい論理モデルを作る．次に，それぞれの機能をどこの部署がやるのか，誰が担当するのかという話になって，再び物理的なモデルに変えて行く．

そういった論理的なものと物理的なものとの切り換えを，たとえばワープロ的なイメージでもいいから，はたして CASE Bench というツールが，どこまで支援してくれるのかという疑問が頭にあったので，こんな質問をしたのです．そのへんは，あとで個人的にお話を聞きたいと思います．

３つ目の質問ですが，グループの方の感想の中に，今の仕事には生かせないという意見がありましたが，具体的にどういうことなのでしょうか？

A（上田賢一，SRA）: 私自身，CASE ツールについてあまり予備知識がなかったので，短時間でそれを使いこなせるレベルにまで到達できなかったということが１つです．それと，現在の CASE ツール自体の完成度がまだそれほど高くないように感じました．私はいま，証券取引業務を担当していますが，一応それなりの設計支援ツールがあります．それに代えて CASE ツールを導入し，ほんとうに仕事に生かせるようになるには，まだかなり時間がかかるのではないかというのが，私の意見です．

Q（落水浩一郎，静岡大学）: 必ずしも否定的な意見でないことはよくわかりましたが，ただ，新しい技術が出てきたとき，いますぐ使えないからといってつぶしてしまったら，いつまでたっても現状から脱却できないわけで，そのへんについては，どういうふうにお考えですか？

A（上田賢一）: 昨夜，飲みながら渡部さんとお話したのですが，仕事に導入するのではなくて，１人１台ずつそうしたツールを載せたマシンを与えておいて，遊びながらでも一応の使い方を覚えさせ，それから自然発生的に仕事に生かしてみようという考えが生まれるのを待つ．そうしたアプローチが最適ではないかと考えています．

Q（天池学）: 環境が整わないとできないということですか？

A（上田賢一）: いや，ちがいます．個人の能力とか意欲の問題だと思います．

4. StP グループの発表

時間もないので，簡単に報告します．まず作業経過ですが，中間発表でいったように，２グループに分かれて作業を進めました．最初は，コンテクストダイアグラムとダイアグラム０の間をを行ったり来たりしながら，作業を進めていたのですが，いまから考えると，ツールをただエディタとして使っていただけではなかったかと思います．つまり，ツールの使い方をみんなで覚えたという段階でした．

最初は，方法論を何も考えずにとりあえずスタートし，外部エンティティの洗い出しについても，何を基準に洗い出したらよいかわからないまま，思いつきにしたがってやっていました．データフローから考えるという原則を無視して，機能からデータフローを考えるプロセス中心のアプローチです．ファイルについても同様に，プロセスに必要なデータを入れようという考え方でした．あとデータフローの本数ですね，「いくつかの線を１個にまとめていいのではないか？」「いやそうではない！」というあたりでもめていたのが１日目です．

中間発表でも述べたように，次の日には２グループの成果をマージするということになっていて，これは比較的簡単に終わるとと思っていました．その時点で，インストラクタの桜井さんおよび荻原先生のほうから，名前の付けかたについて注意がありました．

たとえば，プロセスに関しては，「何々を処理する」とかいう名前ではだめだ，処理するというのは「する」と一緒なので，何をするのかということを考えなさい．また，データフローについても，「何々情報」では何のことだかわからない．何のデータなのかをはっきり示す名前を考えなさい．また，それぞれのダイアグラム単独で何のことかわかるような名前をつけなければいけない．

このような指摘をうけました．

それまで，なぜみんなが安易な名前をつけてたかといえば，プロセスから考えてデータフローを作り，名前をつけようとしたので，それぞれの名前がはっきりしなかったのです．

そこで，コンテクストダイアグラムを書き直しました．今度は，ダイアグラム０を作る時点で，まずそれぞれのデータ定義をはっきりさせることにしました．データ中心で考えたコンテクストダイアグラムは図 14，それを詳細化してできたダイアグラム０は図 15 のようになりました．あとはもう時間もなかったので，ツールを使っていろいろなチェックをしただけです．

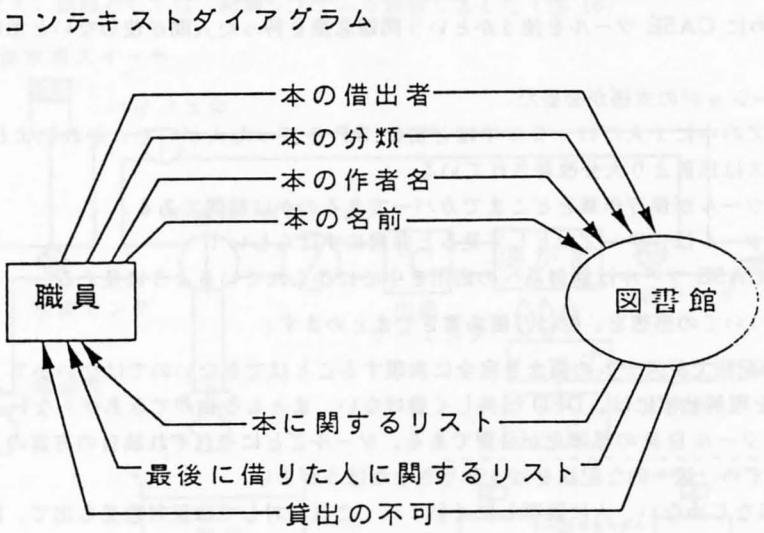


図 14 コンテクストダイアグラム (StP)

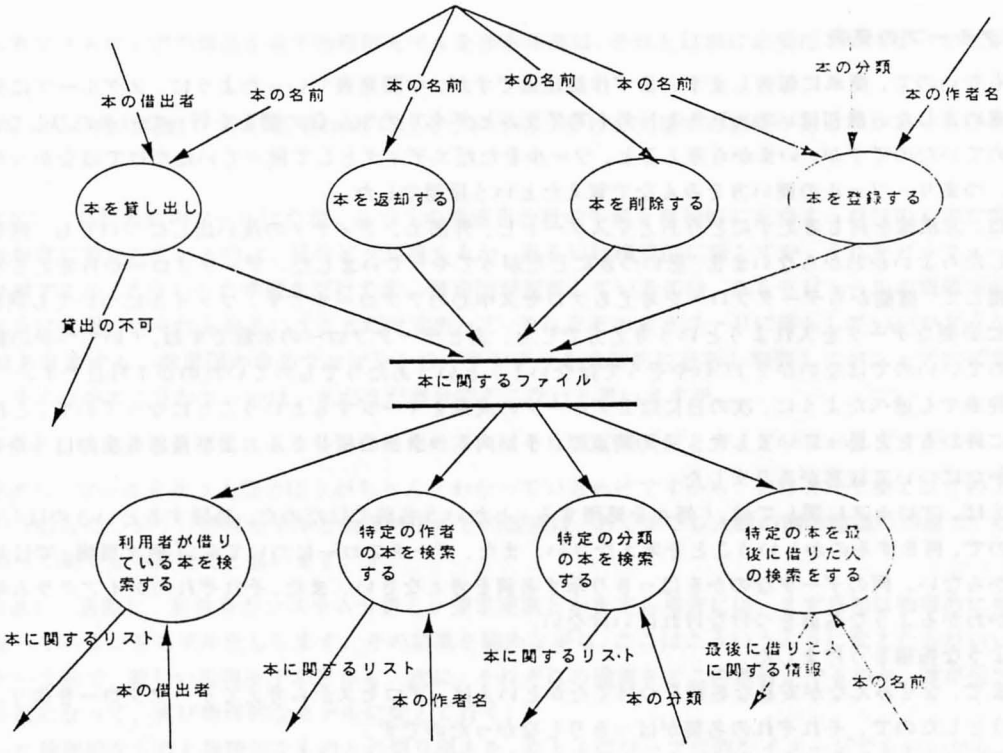


図 15 ダイアグラム 0 (StP)

全体的な感想を以下に述べます．

まずツールについてですが，箇条書きにすると次の通りです:

(1) 操作が少しまどろっこしかった（ツールの操作性に問題があったのか，それとも習得の容易性に問題があったのか，はっきりとした答は２日間では出ませんでした）．

(2) ダイアグラムを同時に２つ開いて，比較しながら設計を進めたかった．

(3) 何のために CASE ツールを使うかという問題意識を持った人間が使わないと意味がないような気がする．

(4) ナビゲーションの支援が必要だ．

(5) グループの中に１人だけ，２年半ほど前に StP を使った人がいた．かれによれば，ユーザインターフェースは以前より大分改善されている．

(6) CASE ツールが保守作業をどこまでカバーできるのかは疑問である．

(7) CASE ツールは，ワープロとして見ると非常にすばらしい！

(8) いまの CASE ツールは制御系への応用を中心に作られているように見える．

次に，記法についての感想を，やはり箇条書きでまとめます:

(1) DFD の記法でシステムの概念を完全に表現することはできないのではないか？

(2) 方法論を理解せずには，DFD は美しく書けない．まともな物ができあがらない．

(3) CASE ツール自身の標準化が必要である．ツールごとにそれぞれ独自の方言のような記法が出てくるのはまずい．統一的な記法をはっきりさせたほうがよい．

(4) DFD はなじめない，人に説明しにくい － これに対しては反対意見も出て，議論を呼びました．記法をユーザが理解できるのか（一般的でない）という意見や，ユーザもやはりプロセス中心の考え方をしているので，データ中心の記法でよいのかという疑問も出ました．

方法論については，初日に説明を受けて，一応はわかったつもりでいたのですが，ほんとうのところは理解できていなかったので，実作業でいろいろと悩みました．分析者とユーザの２役を演じたのですが，どこが方法論

でサポートされている範囲なのかがうまくつかめず，その結果として，自分がいま何をしているのかがわからなくなったことが，途中で何度かありました．同じようなことですが，分析と設計がごちゃごちゃになってしまった（たとえば，DFD を書いてるつもりが，実際は構造図を書いていた）といったことがあります．

グループ編成についてですが，１つの課題に９人は多すぎたと思います．われわれのところは，WS が２台あったので，それぞれ４人と５人で使わせていただきましたが，それでもまだ人数が多かったように感じました．課題の選択にあたって，私たちは図書館システムを選んだのですが，結果的には，問題使用書の日本語の少なさにだまされてしまったような気がしました（笑い）．

最後に，グループの中でアンケートを取りましたので，結果を報告します．まず，課題の数や難易度は，ちょうどよかったのではないかという意見が圧倒的多数を占めました．記法と日程については次の表を見てください．

1. 記法は？		
	わかりやすい	わかりにくい
自分自身にとって	8	1
ユーザにとって	4	5
開発者にとって	5	4

2. 日程に関して			
	短すぎる	ちょうどよい	長すぎる
全体	4	5	0
チュートリアル	2	5	2
課題演習	5	4	0
討論	9	0	0

5. Teamwork グループの発表

最初に経過説明をします．課題としては，給湯システムを選択しました（図 16）．

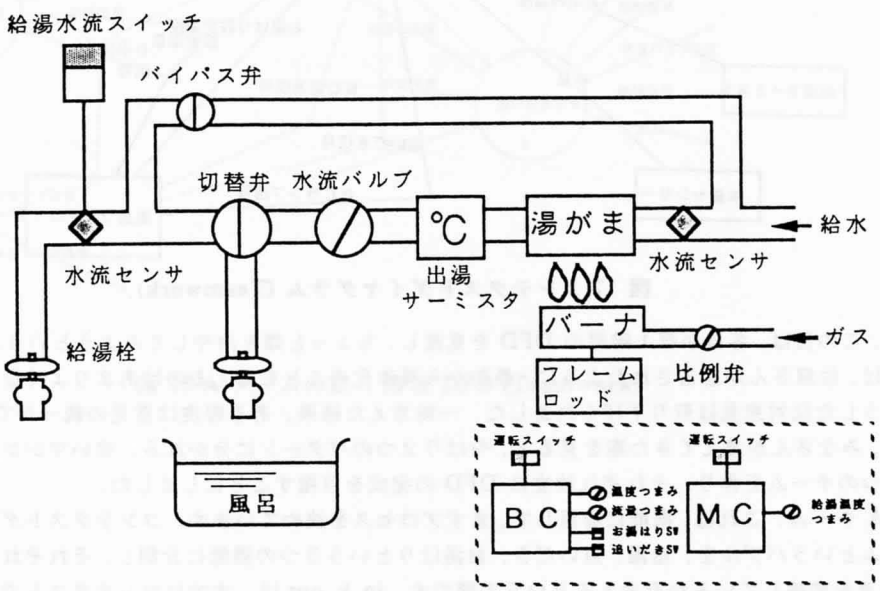


図 16 ガス給湯システム

中間報告で簡単に紹介したように，まず課題を理解してコンテクストダイアグラムを作成しました．そのあと第１版の DFD 作成の過程でいくらか紛糾しましたが，２日目の午前中に DFD およびコントロールスペックの一部ができあがりました．すべてツールを前提として作業を進めようという方針で，コンテクストダイアグラムの作成にさいしては，外と内との切り分けをまず行い，それからデータフロー，次に構造図を考えるという教科書通りの手順にしたがいました．

全体としては順調に仕事が進みましたが，２点ほどちょっとした問題がありました．１つは，ハードウェアモデルとの混同，たとえば，風呂がまとか給水といったような部分についても，コンテクストダイアグラムに書こうとしたことです．実際にはそれらは情報として必要がありません．もう１つはデータです．最初は，ふつうのアナログデータと (0,1) の信号だけで，データとコントロールフローを切り分けながら書いていたのですが，ほんとうにそれでいいんだろうかということが，あとで議論になりました．

こうしてできあがったのが図 17（中間発表の図 11 と同じ）に示すコンテクストダイアグラムですが，このあと，第１階層の DFD を作るときに，大きな意見の相違が生じました．つまり，中間発表の図 12 に示したように，お湯をはる，追いだきする，給湯するという３つの主要機能をベースとして，機能主体で考えて行こうという方針で進もうとしたのですが，これに対して，どうもそれでは具体化にさいしてムダが多いとか，やはりデータ主体で考えるべきだとか，いろいろな意見が出て，議論が発散しかけました．

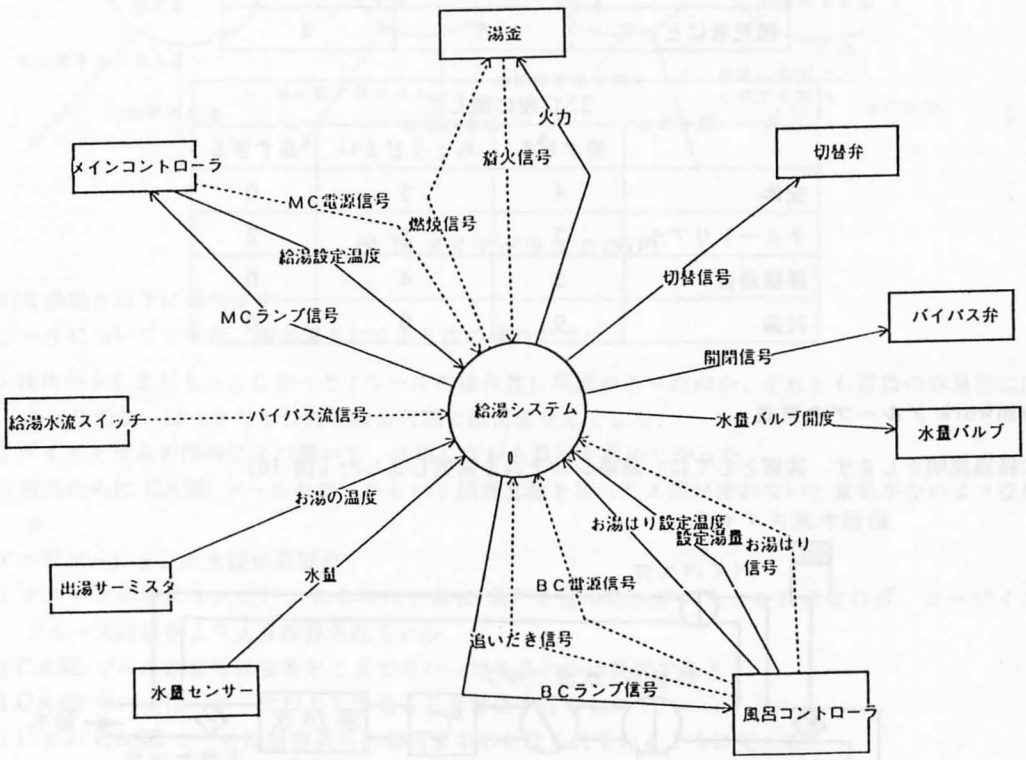


図 17 コンテクストダイヤグラム (Teamwork)

そこで，この日は，各自が第１階層の DFD を見直し，ちょっと頭を冷やしてみようということになりました．結果的には，佐原さんが忠告されたように，最初から具体化のことを考えるのはあまりよくないということがわかり，そうした反対意見は取り下げられました．一晩考えた結果，ある程度は意見の統一ができたわけです．

しかし，みなさんが考えてきた案を見ると，やはり２つのパターンに分かれる．幸いマシンが２台ありましたので，２つのチームを作り，それぞれ独立に DFD の完成を目指すことにしました．

まず，A チーム．これは，機能に着目して，まずプロセスを決めています．コンテクストダイアグラム中の給湯システムというバブルを，給湯，追いだき，お湯はりという３つの機能に分割し，それぞれのプロセスがどういうデータを変換しているかを考えるという手順です．in と out は，すでにコンテクストで全部決まっているので，その３つの機能がどういうふうに振り分けられるかを考えます．まずデータフローに着目してその処理を

考え，コントロールスペックは後で考えるという形で作業を進めました．

一方の B チーム．こちらは，データの流れに着目し，まずデータを out から in の方向でトレースして考えています．こちらに in が全部あり，コンテクストで out が全部決まっているので，それぞれの out データが変換されるプロセスとしては何が必要かを，逆方向から考えて行きました．そして，入出力データができるだけ少なく（できれば入力１で出力１に）なるように，プロセスを構成して行きました．A チームとはちがって，データフローとコントロールスペックを並列的に作って行きました．

それぞれのチームが作成した DFD を図 18 ～図 19 に示します．図 18 ～ 21 が A チーム，図 22 ～ 25 が B チームです．図の形式だけでなく，内容的にもかなりかけ離れたものになっています．

まず，A チームですが，こちらは，給湯・お湯はり・追いだきという３つの機能を中心にしてデータを集中させ，DFD を段階的に階層分解しています．

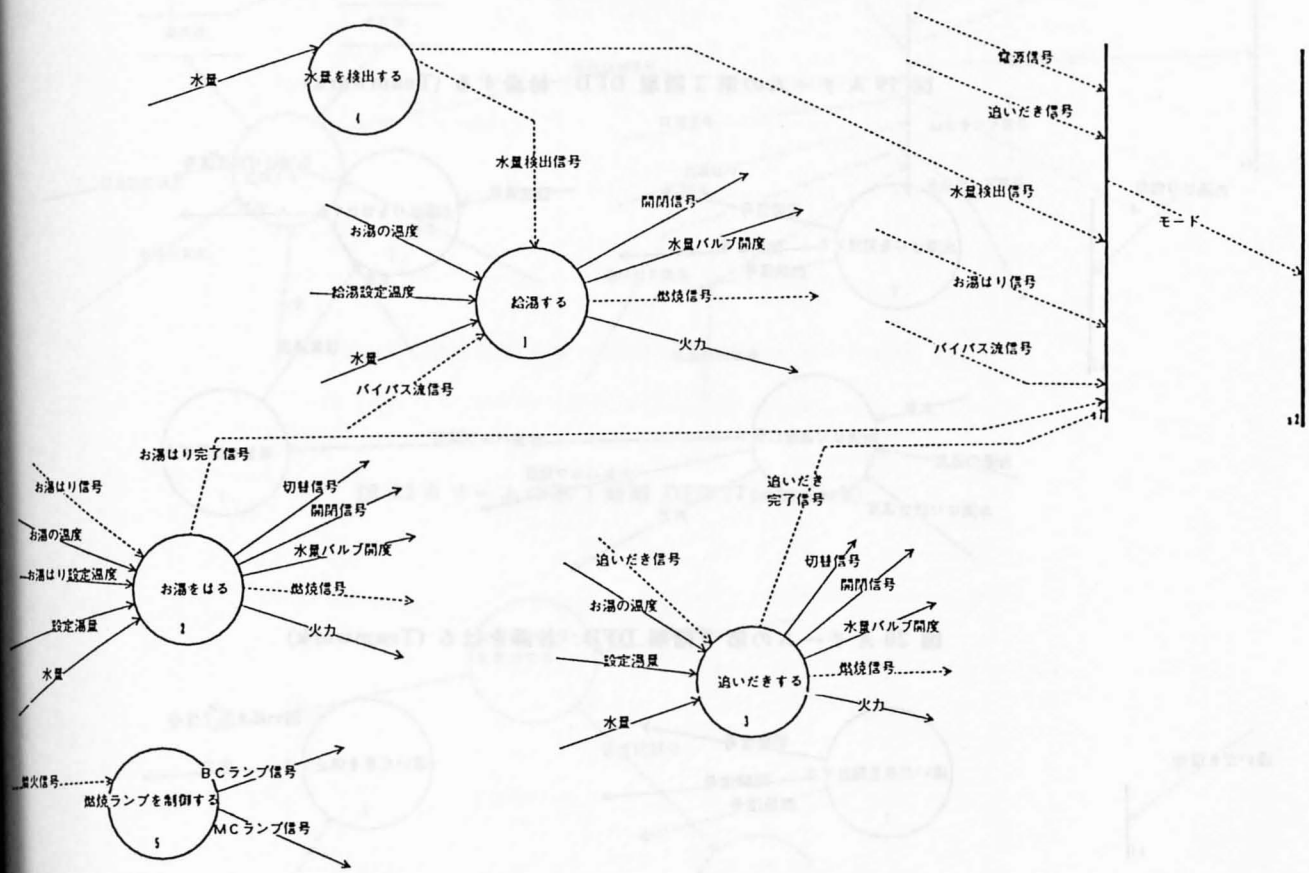


図 18 A チームの第 1 階層 DFD (Teamwork)

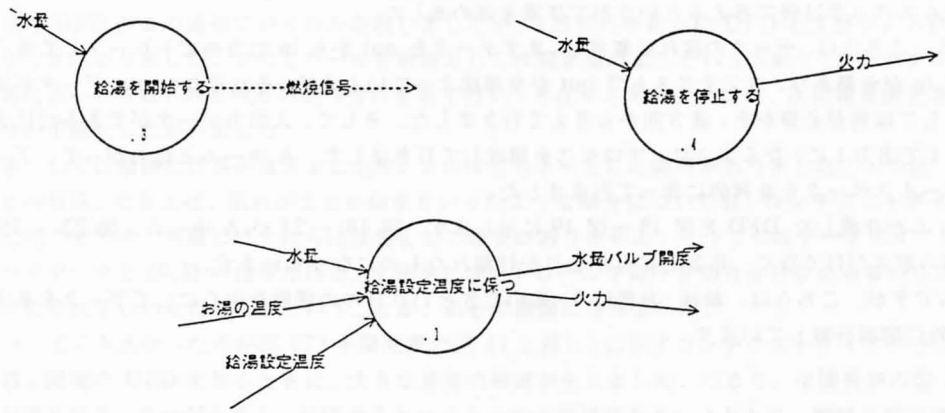


図 19 A チームの第 2 階層 DFD: 給湯する (Teamwork)

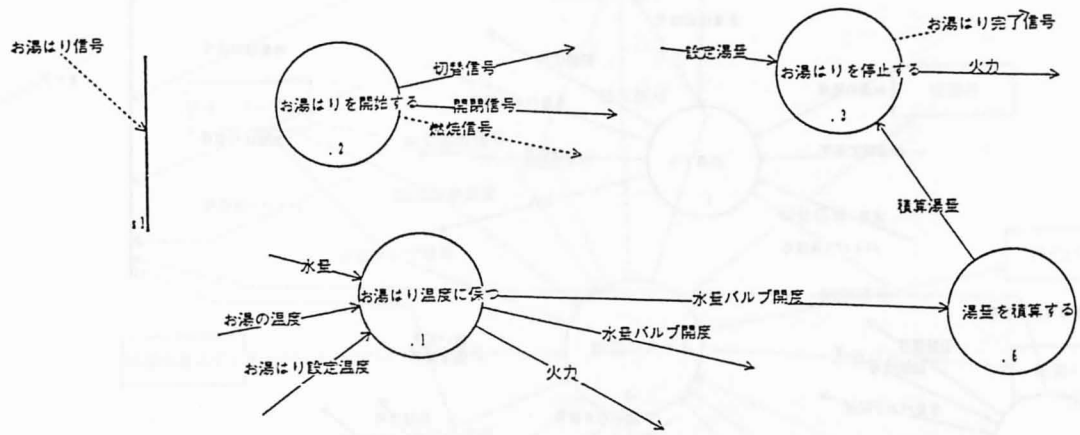


図 20 A チームの第 2 階層 DFD: お湯をはる (Teamwork)

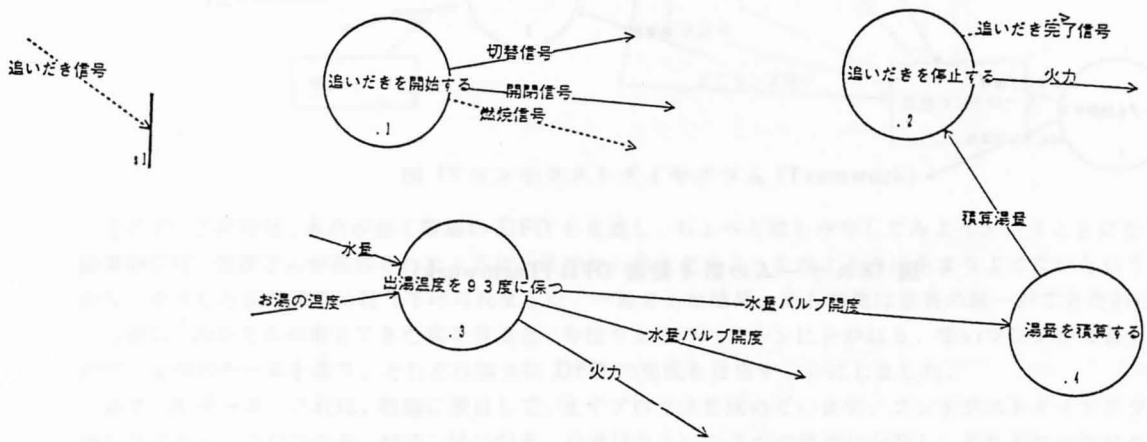


図 21 A チームの第 2 階層 DFD: 追いだきする (Teamwork)

一方 B チームのほうは，データを中心に考えて，火力だとか出入量だとかを処理するようなプロセスは何か，たとえば弁を開閉する・センサを監視する・湯釜を制御する(温度を制御する)というようなプロセスを作っています．このグループでは，DFD と並行してコントロールも考えています．

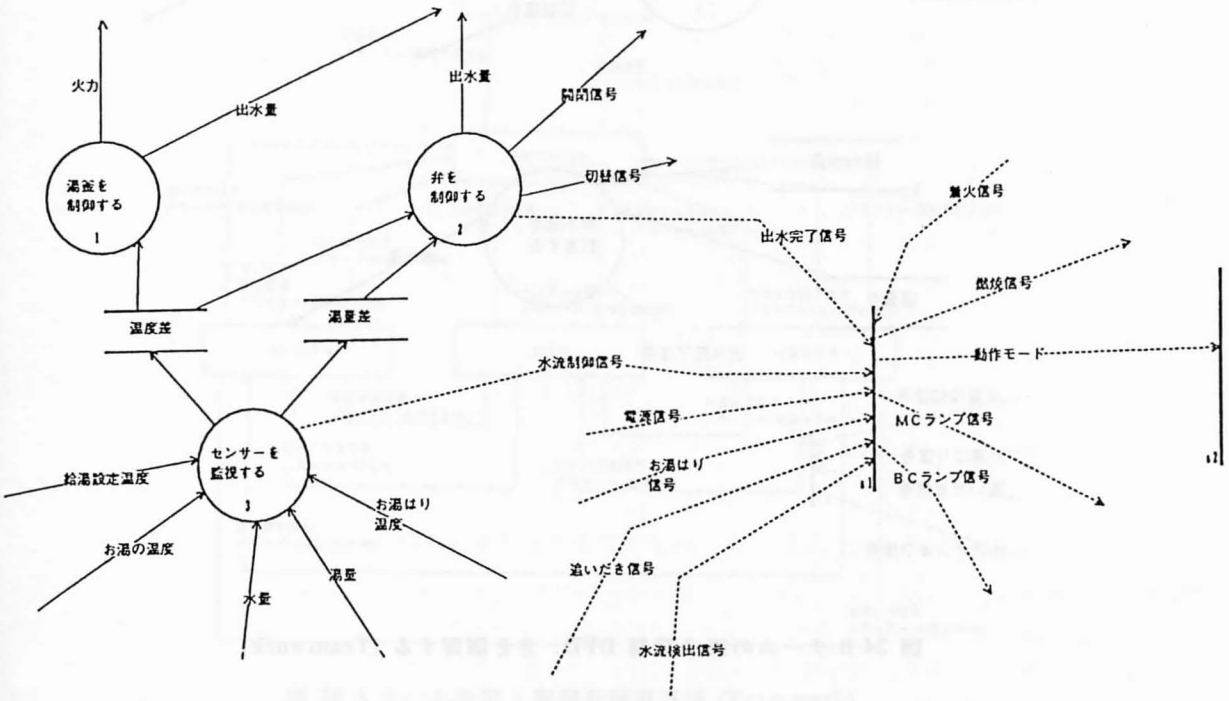


図 22 B チームの第 1 階層 DFD (Teamwork)

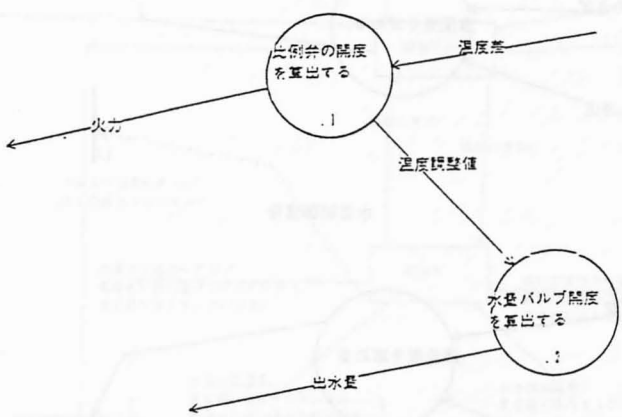


図 23 B チームの第 2 階層 DFD: 湯釜を制御する (Teamwork)

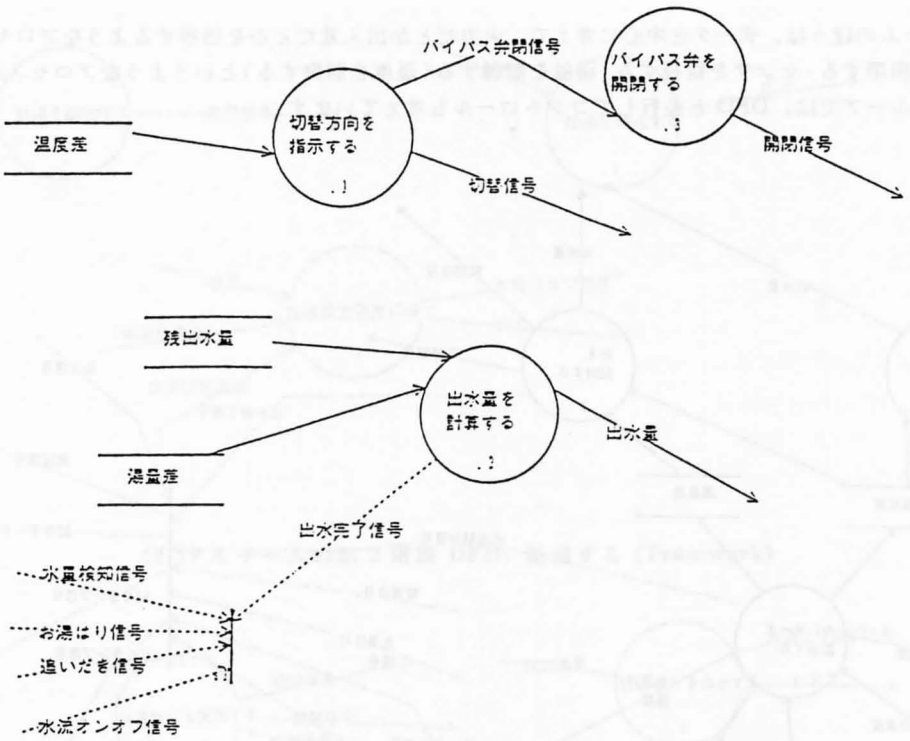


図 24 B チームの第 2 階層 DFD: 弁を制御する (Teamwork)

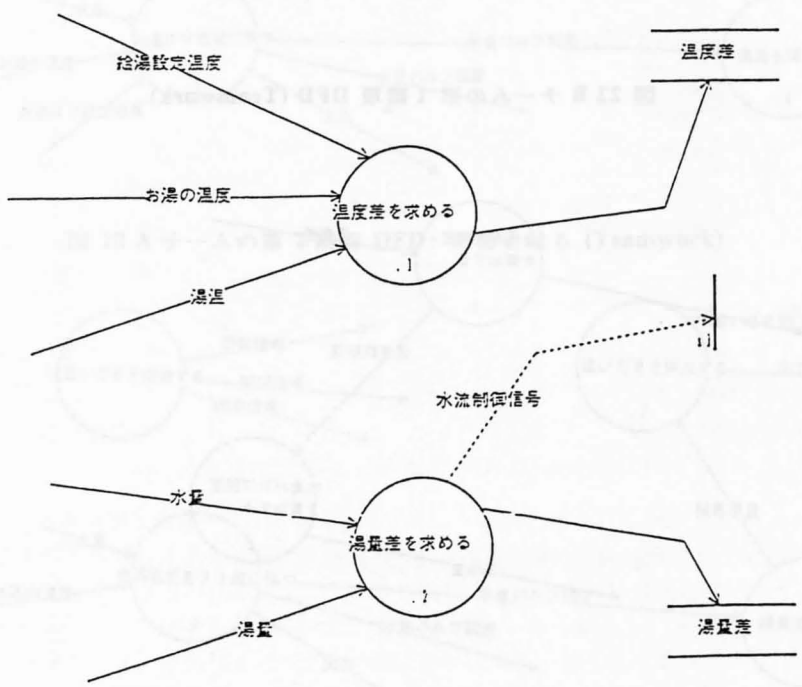


図 25 B チームの第 2 階層 DFD: センサーを監視する (Teamwork)

それから，状態遷移図は，図 26 (A チーム) および図 27 ～ 28 (B チーム) のようになっています．A チームのほうは，だいたい第 1 階層（1 番上のレベル）で制御がほとんど終わってます．一方，B チームのほうの状態遷移図を見ると，2 階層目で重要な制御が行われるような感じになっています．

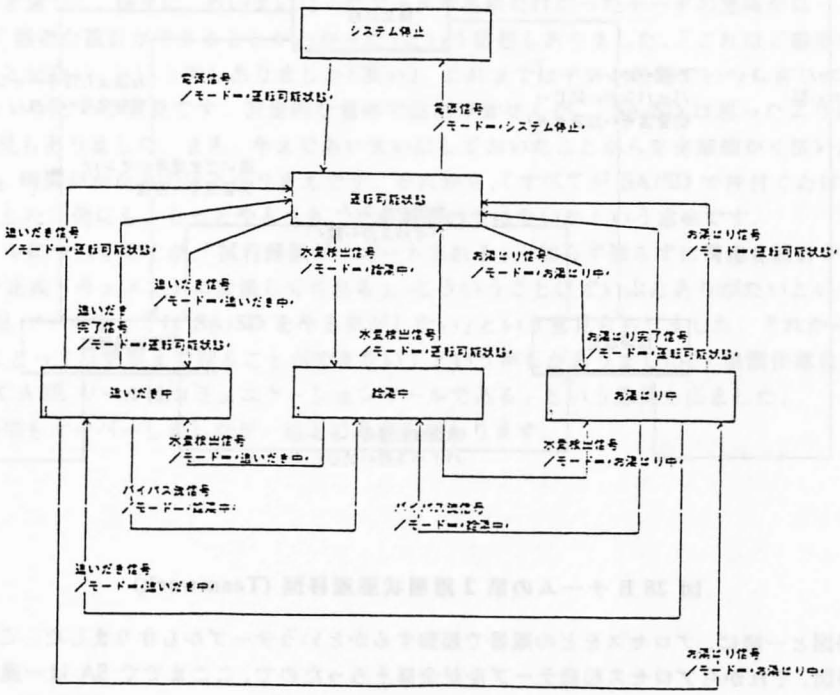


図 26 A チームの第 1 階層状態遷移図 (Teamwork)

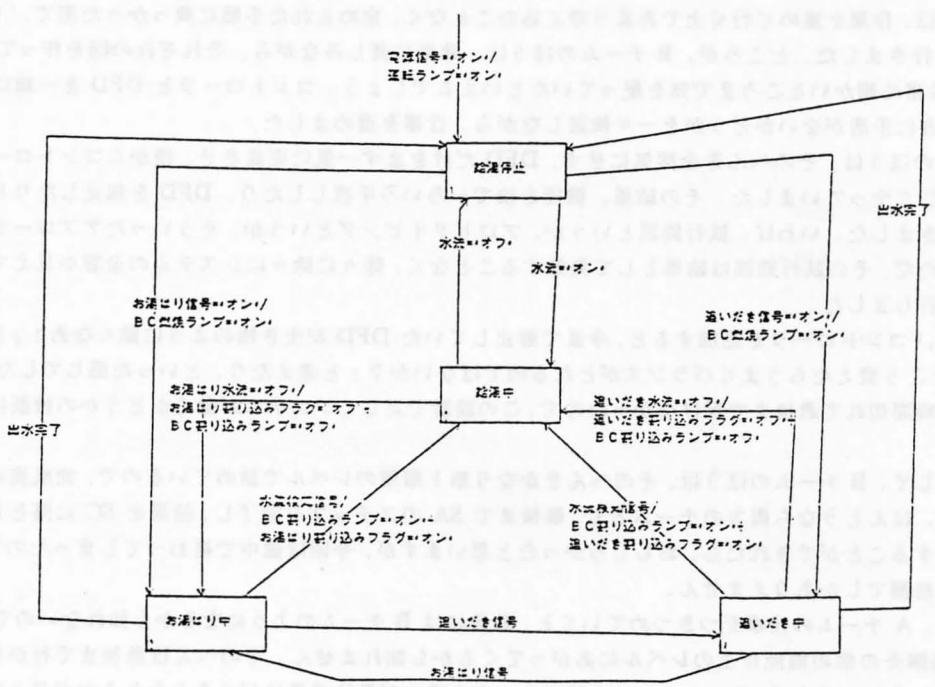


図 27 B チームの第 1 階層状態遷移図 (Teamwork)

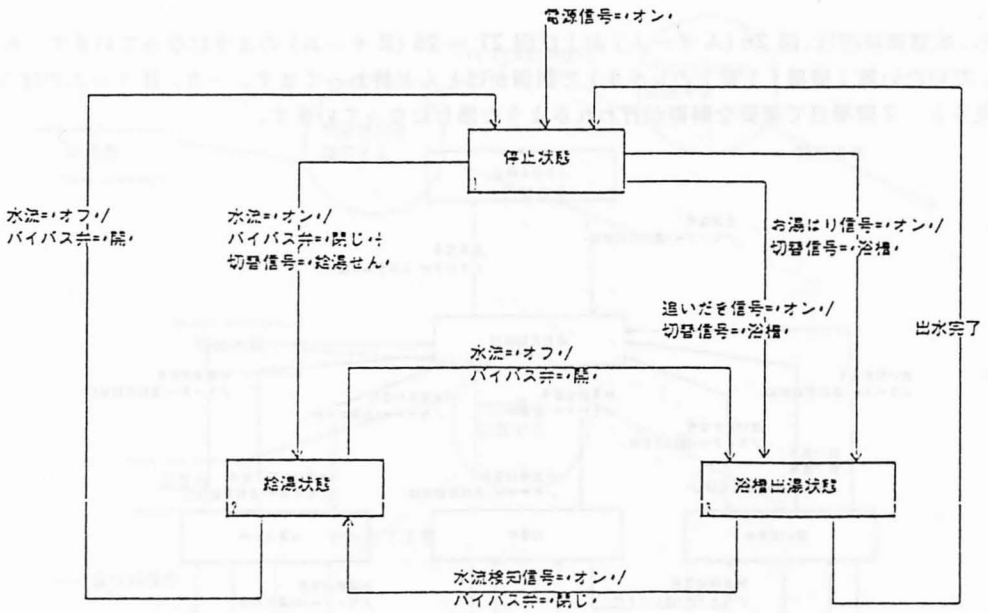


図 28 B チームの第 2 階層状態遷移図 (Teamwork)

この状態遷移図と一緒に，プロセスをどの順番で起動するかというテーブルも作りました．これで，データフローと状態遷移図，それからプロセス起動テーブルが全部そろったので，ここまでで SA は一通り終わったということで，ミニスペックと DD はまだ空のまま，ツールを使ってストラクチャチャートを出してみました．

さて，以上の作業を振り返って，２つのチームの間の違いを比べて見ると，だいたい次のような感じになります．

A チームは，作業を進めて行く上であまり考え込むことなく，定められた手順に乗っかった形で，「機械的に」先へ進んで行きました．ところが，B チームのほうは，非常に苦しみながら，それぞれの図を作っていました．１つには，非常に細かいところまで気を配っていたといえるでしょう．コントローラと DFD を一緒に作っていたので，双方に矛盾がないかどうかを一々検証しながら，仕事を進めました．

A チームのほうは，そのへんを全然気にせず，DFD だけをまず一気に完成させ，後からコントローラを入れるという感じでやっていました．その結果，機能を後でいろいろ手直ししたり，DFD を修正したりということが結構出てきました．いわば，試行錯誤というか，プロトタイピングというか，そういったアプローチだったといえます．ただ，その試行錯誤は結果として発散することなく，除々に除々にシステムの全容が見えてくるような感じで進行しました．

たとえば，「コントローラを追加すると，今まで静止していた DFD が生き物のように動くなあ！」と感心したり，「ここをこう変えたらうまくバランスがとれるのではないか？」と考えたり，といった感じでした．しかし，残念ながら時間切れで最後まで行けなかったので，この設計で正しくシステムが動くかどうかの検証は終わっていません．

それに対して，B チームのほうは，そのへんをかなり第１階層のレベルで詰めているので，完成度が高いような感じです．ほんとうなら両方のチームとも，最後まで SA のステップを完了し，結果を SC に落としてどうなるかを比較することができれたら，おもしろかったと思いますが，今回は途中で終わってしまったので，これ以上の議論は推測でしかありえません．

たとえば，A チームの仕事をつきつめていくと，最後には B チームのようになるかも知れないのです．また，バルブの制御その他の機能が上のレベルにあがってくるかも知れません．そのへんは最後まで行かなかったのでなんともいえないですが，２つのチームのアプローチの違いが最終成果にどうあらわれるかが見えなかったのがちょっと残念です．

以上，それぞれのチームの活動内容を報告しましたが，最後に，全員で方法論とツールについて少し意見を交

換しました．時間がずいぶん押し詰まってから討論に入ったので，十分な議論はできなかったのですが，次のような意見が出ています．

まず方法論についてですが，「次第にデータの意味がはっきりしてくる」という意見がありました．これは，設計のプロセスを通じて，徐々に，あいまいだったデータや名前だけだったデータの意味がはっきりしてくるということです．「厳密な設計ができることがわかった」という感想もありました．「これほど厳密な設計はいままで現場やったことがない」という声もありました(笑い)．これまではテスト段階でいつも苦しんでいたが，それが防げのではないかという意見です．否定的な意味ではありませんが，「SA/SD は思ったよりに時間がかかるなあ」という意見もありました．まあ，今まであいまいにしておいたことがらを全部細かく洗い出そうとしているわけですから，時間がかかるのはあたりまえです．それから，「すべてが SA/SD で片付くわけではない」という意見もありました．他にも，もっとやるべきことがあるのではないかという意味です．

最後にツールについてですが，「試行錯誤がサポートされる」，「知らず知らずに構造化設計ができている」，「ラフスケッチを正式ドキュメントに変換してくれる」，こういうことはずいぶんありがたいという意見が圧倒的でした．「CASE ツールなしでは SA/SD をやる気がしない」という意見もありました．それから，「たった２日間では初心者にとっては効果まで計ることができない」という声もがありました．「協調作業自然に支援される」，「その意味で CASE ツールはコミュニケーションツールである」という意見も出ました．

ちょっと時間をオーバーしましたが，以上で発表を終わります．

7. パネルディスカッション

報告者
古川 勝也・太田 勝
（静岡大学）

落水浩一郎 (静岡大学):

昨日のアンケートの結果を３項目の論点に整理してみました．それをもとに，こちらに並んでいらっしゃるプログラム委員・グループの代表・ベンダの方々に私が代表質問する形で，パネルを始めさせていただきたいと思います．

最初の論点は「方法論の教育法」です．必ずしも十分ではありませんが，今回のワークショップで，ここに集まったすべての方々が CASE ツールに直接手を触れるチャンスを持ちました．CASE のねらいは，一定の手順・枠組みの中で，共通の言語を用い，コンピュータの助けを借りながら仕事をしようということです．ところが，昨日のアンケートの回答でも，また，先程の最終報告でも，方法論が１つのネックになっているように感じられます．方法論が役に立つか立たないかという議論は無意味だと思うので，角度を変えてずばりお聞きします：方法論を使いこなすためには，何をどの程度訓練すればよいのでしょうか？

２番目の論点は「チーム作業の支援」です．お互い共通の言語でしゃべれることは CASE ツールの一つの効果でしょう．しかし，より基本的な問題があります．設計は１人でやるのか？それとも何人かで分担するのか？もちろん，どちらの場合ももあり得ます．１人で設計を行った場合には，その内容または結果を他人に伝達することが必要になります．CASE ツールはそのときどの程度有用なのでしょうか？記法が標準化されていることは１つの利点ですが，それ以外に何かを期待できるのでしょうか？一方，複数で設計する場合には，仕事を分割して分担するわけですが，CASE ツールはそのとこをいったいどういうふうに支援してくれるのでしょう？

最後の論点は「CASE ツールの効果」です．ずばり成功例と失敗例を知りたい．また，さしあたりどんな分野で効果があるのかも教えて戴きたい．ドキュメンテーション，リバースエンジニアリング等のキーワードが，このワークショップを通じて議論されてきましたが，ここでは，ベンダの方々から，御経験にもとづく情報を提供していただきたいと思います．

以上３つの論点について，壇上に並んでいる全員にお聞きすると，１人５分としても２時間かかります(笑い)．そこで，積極的に情報を提供してくださる方にどんど発言いただく形で進めたいと思います．

青木淳 (富士ゼロックス情報システム):

オブジェクト指向の方法論について，何をどのように訓練すればいいかを，若干お話したいと思います．

まずみなさんに認識していただきたいことがあります．たとえば，今回の演習で使われた図書館問題を考えてみましょう．StP グループがこの問題に挑戦し，その結果，さきほどの最終発表で示されたコンテクストダイアグラムができあがりました．これをもとに図書館システムを設計することは，やろうと思えば，どんな方法でもできるでしょう．しかし，ほんとうは，設計ができたあとが問題なのだと思います．オブジェクト指向の考え方でこの図書館問題を分析すると，結果は図１のようになります．

オブジェクト指向特有の抽象化プロセスの結果として，問題に依存したオブジェクトの上位に，いくつか抽象的なオブジェクトが生まれてきます．これらが，次にまた違うシステムを作るときの財産になるというのが，オブジェクト指向のメリットです．つまり，オブジェクト指向による分析・設計は，まずじっくりキャスティングをして，それから映画をとるような感じです．そういう考え方に慣れてほしいと思います．

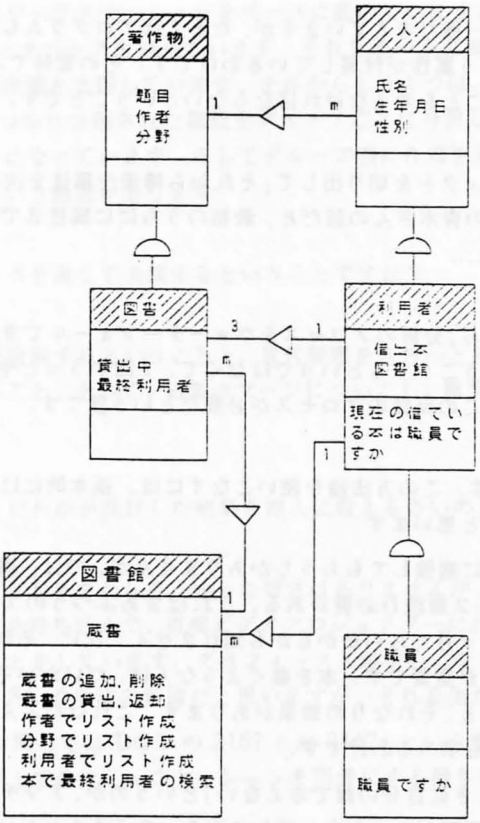


図１図書館システムのオブジェクト指向分析

それはどうすればやれるようになるかといえば，やはり，ヒュマンインテグレーションを考えられる人が，オブジェクト指向には向いていると思います．これができない人は構造化手法でやっていてくださいということになります．

天池学 (カシオ計算機):

ヒューマンインテグレーションとおっしゃいましたが，具体的にはどういうことですか？

青木淳:

登場してくるオブジェクトの役割を決めて，きちんとエンカプセレーションし，それらの間での円滑なコミュニケーションを考えなければなりません．それは，ちょうど，複数の人間を集め，チームを組んで協調作業を計画するモデルと非常によく似ています．そういうことをうまく考えられる人が，オブジェクト指向に向いているということです．

天池学:

なるほど，キーワードは協調ということですか．

平尾一浩 (ヒラタ・ソフトウェア・テクノロジー):

青木さんのチームにいたのですが，抽象化で非常に悩みました．いまの図書館システムの例で，たとえば「書籍」を抽象化して「著作物」という概念を出されましたが，一歩まちがえると，「紙」とかいうような変な抽象化をしてしまうと思うんです．そのへんの思考のポイントっていうか，何かあれば教えていただきたいんですが....

青木淳:

抽象化で出てくるクラス階層には，その人が考えている概念のカテゴリみたいなものが，正確にマップされてしまう．そのさい注意しなければいけないのはやっぱり協調の問題であって，関係者がそれを妥当だと思うかどうかにかかっています．「紙」という抽象化は，問題ドメインを経済的に表現していないので，明らかにまちがい

です．この例では図書を著作物の一種だ考えていますが，たとえばプログラムも著作物の一種であり，それをだれが作って，どういう題目でという属性が付属しているわけです．その意味でこの抽象化は非常に経済的であり，応用範囲が広い．そういったことを考えなければならないということです．

平尾一浩:

自然な思考の流れの中でオブジェクトを切り出して，それから構造や属性を決めていくときに抽象化をするのがふつうですよね？しかし，いまの青木さんの話だと，最初のうちにに属性まで先に考えておかないと，抽象化が難しいような気がするんですが....

青木淳:

そうです．その通りです．だから，分析のプロセスをウォーターフォールで考えてはいけません．オブジェクトを完全に全部洗い出してからどうこうするというではなくて，１回すうっとやってみる．で，何かが足りないことがわかって元に戻る．まさにこの反復のプロセスが必要だという話です．

佐原伸 (SRA):

SA/SD の教育に関して述べます．この方法論を使いこなすには，基本的には，日本語能力そして抽象化能力があることが，たぶん要求されると思います．

そうした能力は，それぞれの人に勉強してもらうしかありませんが，では，構造化技法を何日で学べるかといえば，まずテキストブックを１～２冊読む必要がある．これはまあふつうのエンジニアなら１週間で読めるでしょう．もちろん人によっては１ヶ月くらいかかるかも知れません．次に，それらの本の中の演習問題を，ツールを使ってトレースしてみることが重要です．本を書くような人は，だいたいそれなりにできる人ですから，その人の考えた解法をたどってみると，それなりの効果があります．これはたぶん１～２日でできるでしょう．当然一緒に CASE ツールの使い方も学べるわけです．

逆説的にいえば，「問題を解くときに自分の頭で考えない」というのが，アマチュアとしての正しいやり方です．つまり，教科書のどこかに図書館問題があったなと探してみる．そうするとたいてい出てくるんですね．その解答をなぞることで，かつてそれを解いた人のレベルにまでは行けるわけで，それ以上何か新しいアイデアが必要になった場合に，初めて自分でもう少し考えてみる．

すばらしい設計をするのには，たぶん，さきほど青木さんがいわれたような適性がないとだめでしょうが，ほどほどの設計をする程度のレベルになら，一生懸命やれば１～２週間，仕事のかたわらやって１～３ヶ月ぐらいで到達できるだろうと，私は思います．もちろん，大きな問題にアタックするときには，方法論がよくわかってる人の講義を聞く(あるいはセミナーを受ける)という形で力を補うことが必要です．そうやって行くうちに，もう６か月くらいたてば結構ベテランになってきます．

オブジェクト指向に関しても一言いわせていただくと，私は実はいまオブジェクト指向を勉強中なのですが，さっきの青木さんの意見には反対です．この方法論も，Smalltalk の中にすでに用意されているクラス階層を勉強したり，すでに解かれた例題を勉強することによって，かなり早くそこそこのレベルに到達できます．最終に作ったものがすばらしいか，さほどでもないかというところで，能力の差が出てくると思います．Smalltalk 中のクラス階層を知っていれば，だれでも，そこにコントローラというものがあるのだから，「この制御スイッチはコントローラの下に入れればいい」という程度の判断はできるでしょう．ゆっくり勉強して３ないし６ヶ月，６ヶ月たてばかなりのベテランといえる感じになると思います．

青木淳:

OOA/OOD を習得するための期間ですが，私の例でいうと，何もないところから独学で約２年かかりました．いま，そういう人を育てるのには，やはり３ヶ月はかかるでしょう．つまり，よいナビゲーターがついたとして３ヶ月かな？という感じです．

落水浩一郎:

では，次の話題に移ります．大規模システムを作る場合とか，複数人で作業するときに，CASE ツールがサポートしてくれる機能は何かということですが，どなたか....？

田中耕市 (東陽テクニカ):

Teamwork の場合，基本的にはワークステーションをベースに動いていますので，ネットワーク上で一元化された（分散ではなく集中型の）データベースを持っています．それをチーム全員で共有して作業するというのが基本です．そういう意味で，協同作業を支援しています．それからもう１つは，モデルが途中まで（ある階層まで）できたときに，そこからいくつかに分割された機能をグループごとに分担して詳細化するさいに，作業用のモデルを簡単にコピーできるようになっています．そしてグループ別に作業を進め，最終的にそれぞれの部分が完了したらそれらをマージするという機能があります．

落水浩一郎:

情報格納庫というかデータベースを通じて支援するということですね？

田中耕市:

はい．データベースを一元的に管理するということと，世代管理をきちっと行うこと，それから派生的にモデルをコピーして作業を進められること，そして，結果のマージについても，衝突を回避しながらマージするというような機能が用意されています．

落水浩一郎:

伝達についてはいかがですか？だれかが設計した結果を他人に教えるさいの支援機能はありますか？

田中耕市:

それについては，日本ではまだお客さまからそういった要求はありませんが，アメリカの場合には，ある大会社で，会議にワークステーションを持ち込んで，直接ビデオプロジェクターに投影して討論し，変更箇所が出たらその場で直していくといったことをしています．グラフィックスが中心になっているので，そうした場合変更箇所が見つけやすいというのも，SA の手法の特徴だと思いますが，それを生かしているということです．

それぞれのツールが，文書化に関しては DOD の 2167 とか 2167a という標準形式にしたがっており，そうしたスタンダードも，また，紙の上でのコミュニケーションを円滑にする働きをすると思います．

荻原剛 (大阪大学):

共同作業や意思伝達については，一般には，数人で作業をやって行く中で，何か問題が起こったときに，ごちゃごちゃ集まってどうしようかって相談をすることの支援（いわゆる CSCW）が，話題の中心になっている感じですが，作業の進行とか進捗状況自体の管理まで CASE ツールで支援できているのか？あるいはするつもりがあるのか？そのときどんな支援機能を考えているのか？そのへんについて，ベンダの方々から少し情報をいただけたらうれしいのですが....

桜井麻里 (SRA):

現状では，StP の場合でいうと，チーム・メンバの作業の進捗状況を見ようと思ったら，データベースの内容を見に行き，そのメンバが担当したモデルがいまどこまでできているかを調べる以外にやり方はありません．もちろん，作られたモデルや図の一覧とかは出ますけども，作業の進捗を正確に把握するには，データベースの内容を調べる以外にないと思います．

落水浩一郎:

モデルっていうのはプロダクトのモデルですか？

桜井麻里:

はい，そうです．作られた DFD とかがディスクに入っていますから，管理者がエンジニアと同じツールを使ってそれを見て行く以外には方法はないと思います．それから，さっきの作業分割の話ですけども，分割はもちろんいつでも可能ですが，概観図というか，全体のコンテクストダイアグラムのところは，やはりだれかが統一見解を強引に出さなければいけません．ツールの話ではなく，どうしても基本のモデルをきちんと決める必要があります．

落水浩一郎:

では次に，成功例と失敗例について，これは，ベンダ全員にお答えいただきましょう．

田中耕市:

たぶん，落水先生が満足される答えにはならないと思いますが，現材，具体例として私どもが資料をお出しし

ているのは，英国のドレッシーという会社の例です．この会社は，軍事システムを作ってる会社ですが，マイクロプロセッサを 10 台ぐらい並列で処理するようなシステムを開発し，出荷後に問題になったバグがわずか 3 つぐらいに押さえられたというケースです．これは，ヨーロッパの雑誌に報告されています．

それから，たぶんハートレーだと思いますが，構造化手法の成果として取り上げていた例として，ボーイング 737 のケースもあります．そのシステムはやはり，マイクロプロセッサが 150 個ぐらい組み合わせたシステムで，SA/SD を使って非常にうまく行きました．CASE ツールががまだ導入される以前の成功例です．ですから，CASE ツールを使えば，生産性・信頼性がさらに向上するというので，ボーイング社には，すでにいろいろな会社のツールが，それぞれかなりの本数入ってるそうです．

落水浩一郎:
日本での例はありませんか？

田中耕市:
私どもの場合にはまだありません．

渡部健次 (ソニーテクトロニクス):
何をもって成功／失敗というかですが，CASE ツールがシステムのライフサイクルの中に根づいて，企業の中でずっと使われていくことが大切だと思います．B-737 みたいなワンショット的なケースもさることながら，やはり，ある企業に CASE が導入されたとして，その後継続的に利用され運用されて行く，そうすれば，CASE ツール自体が１つの系統的な手法を持っていますから，ライフサイクル全体の中で一定の効果を生み出す，そういうことが基本だといえましょう．逆に失敗というのは，ワークステーションがあり，CASE もあるだけで機械があるだけだれもさわらないという状態です．

そういった意味では，われわれの場合は，コーディングなどの原状の作業を大事に考えています．あちこちで再三申し上げていますけども，現在あるコードから構造図に落とすという単純な機能をベースに，データフロー図を仕上げる．そして，そのデータフローに新しい周辺部を加えて設計仕様・構造図を再構成するというサイクルが組めます．

落水浩一郎:
それは，原状から CASE の世界に移って行くための１つの手段として，すでにあるコードを仕様の世界に逆変換し，そこで新しいコードを開発するというリバースエンジニアリングのアプローチだと考えられますが，成果は保証されますか？

渡部健次:
それはやはり問題意識ですね．問題意識を持って，こういった手法でやっていこうと考える，こういう方なら必ず成功します．

落水浩一郎:
ユーザーの責任にされるのはいいんですけれども，一方，導入したら必ず成功するという保証がないと，ユーザーとしてはなかなか踏み切れない．そういうジレンマがあると思いますが....

渡部健次:
SA/SD は方法論としてはすでに確立されていて，アメリカなどでもかなりの実績があるわけですから，その点についててはまったく疑いを持っていません．

桜井麻里:
いま２人の方がいわれたことを受けて，お話したいと思います．ワークステーション上の CASE ツールは，日本では２年前位から使われ始めていますが，それをを使ったソフトウェア・プロダクトがうんぬんという形での成果はまだでていないと思います．現在は，まだ導入・試用段階です．

StP のユーザの場合もさまざまです．うまく使っているところもあれば，全然使わなくなったところもあります，どういう場合にうまく使われないかというと，まず，問題意識も何なくてただ単に買ってみただけのところとか，担当者がたった１人だけで周囲から孤立しているところです．そういったユーザの場合は，最初いろいろ聞いてくるのですが，そのうちまったく質問が来なくなってしまう（笑い）．

うまく行っているユーザ，とにかく積極的に使って下さっているところは，まず複数（3人位）の担当者がいます．そうすると，わからないことがあればお互いに相談できるし，それでもわからなければベンダに問い合わせるといった環境になります．特に，方法論上の問題や分析の進め方などに関して，よい指導者がいるとか，仲間どうしで相談しあえるということが大切でしょう．試験導入とはいっても，最低2～3人の人がは使えるような環境設定と周囲の理解，そして問題意識．それだけ整えば，技法的には確立されていますし，みなさんも今度のワークショップで実感されたと思いますが，いままでできなかった設計時点でのテストとか，誤りの発見とかができるわけですから....

佐原伸:

成功例ですが，われわれがが自分自身で StP を使った時には成功しました（笑い）．それから，まったく問い合わせも質問も来ないユーザーの中に，アメリカで StP を作った先生の弟子だったという人がいて，これはある製薬会社なのですが，ここも明らかに成功しています(笑い)．つまり，方法論をよく知っている人が使えば大丈夫．他のユーザーで成功しているのは，基本的にワークステーションやツールを使うことに慣れているところ，そういった会社の場合は，比較的スムースに CASE の世界に入れるようです．

青木淳:

オブジェクト指向を使った成功例ですけども，オブジェクト指向で作られたシステムと従来のものとの違いをよく見て下さい．明らかに，みなさん強烈なインパクトを感じるはずです．それが成功例です．失敗例はまだ出てきていません(笑い)．私自身がこれからやろうとしているのは，構造化のアプローチが行き詰まったあたりからオブジェクト指向が出てきたわけですから，いずれはオブジェクト指向もさらに上位のより強力な方法論に包含されて行くだろう．そのあたりを追求して行きたいと考えています．

小林透 (NTT):

私はいま，CASE ツールについて調査しているのですが，プロダクトはアメリカ製が圧倒的に多く，また導入しているユーザもアメリカのほうが多い．日本で CASE を導入しているユーザはごく少ない．その原因としては，文化の違いその他いろいろ考えられると思いますが，どうしてこうなのかということに関して，御意見をお聞かせ願えますか？

佐原伸:

日本ユーザが外国で成功したものでないと買わないということが1つですね．それから，ワークステーションの普及率が日本の方が低いということもあります．つまりそもそも CASE を導入する基盤ができていない．また，方法論についていえば，従来からあるような要求仕様書を作って下流工程に流れて行くという開発手法でこり固まってしまっているということですね，これまではそれで何とかうまく行っていたのですが，もうそろそろかなり危なくなってきた．しかし，大半はまだそれに気づいていない．

実は，アメリカでも CASE はそれほど普及してはいないのです．軍関係とか，航空宇宙産業とかを除けば，日本のコンピュータ・ユーザの主流である事務処理系のユーザに対応するあちらの MIS コミュニティでは，まだほとんど採用されていない．

渡部健次:

日本国内でどうして CASE が爆発的に普及しないかといえば，それは，仕事がきちんと分業化されていないからだと思います．1人の人間が設計からコーディングまで手がける，それはまあ極端な例ですが，割とそういう傾向にあるんじゃないでしょうか？それが，分業化されて，たとえばアナリストという職種が確立されれば，その人が何かの出力を出さなければならなくなり，そのために必要な支援ツールを捜すと思うんです．構造化手法の CASE ツールは，そうしたアナリストのための道具です．日本ではまだソフトウェア開発の分業化が進んでいない．それが1つの原因だと考えます．

青木淳:

もっと人間的な話をすると，日本ではプログラマが会社を辞めないからです．アメリカだと非常に転職率が高いですから，各個人の頭の中にたまったノウハウを外部に取り出して，固定化しておかなければどうしようもない．ところが日本は終身雇用制度で，システムの生き字引きみたいな人が必ずいる．困ったときはいつでもその人を頼ればよいという状況がある．多分こんな所につきるでしょう（笑い）．

桜井麻里:
たしかにそのことは大きいと思います.

天池学:
社会的な問題ということですね.

桜井麻里:
それに関連して,日本のソフトウェア技術者は,自分の考えを人に説明するのに慣れていないということもあげられます.その必要性がないからだと思うんですけれども....

藤原千尋 (日本ラッド):
いまのことに関する質問ですが,他の分野,つまりソフトウェア開発以外の分野では,たとえば QC とかで割と成功して,逆にアメリカに技術を輸出するというパターンもありますよね? SA や OOA についても,今後日本的な手法が開発され発展して行く可能性は感じられませんか?

青木淳:
やっぱり,いつかは B29 を撃墜しようと思っているわけですか?(笑い)

佐原伸:
QC は別に日本独自の技術ではなく,もともとアメリカから来たものでしょう?ただ,こちらではアメリカ人よりしつこくやっただけの話だと思います.CASE については,完全に日本語化されたツールがまだできていないので,それができさえすれば,方法論自体別に変える必要はないと思います.いま,SmallTalk を使っていて困るのは,メソッドの名前をどうつけるかというところです.つまり,英語力がもろに効いてきて,方法論以前に英語で悩んでしまう(笑い).完全に日本的なオブジェクト指向言語なら,もっと設計雅やりやすいだろうなと感じることがあります.その意味で,文化的な差を乗り越えた CASE ツールが必要だと思います.現状では,たとえば StP の場合にも,日本語化されているとはいえ,ある部分では英語を使わなければいけないところがあったりする.もっと完全な日本語化が望ましいと思います.

PWB グループのだれか:
青木さんにうかがいます.導入の成功例・失敗例について,オブジェクト指向で作られたものを見てほしいといわれましたが,具体的には SmallTalk 上のアプリケーションと考えてよいのでしょうか?

青木淳:
SmallTalk 自身が SmallTalk で書かれていますんで,SmallTalk がみなさんに与えるインパクトはその通りです.それ以外では,オブジェクト指向で作られたユーザーインタフェース,たとえばマックとか X-Window とか,そういったものをごらんになってください.

PWB グループのだれか:
そうしたユーザーインターフェースの設計には,オブジェクト指向は大変すぐれていると思いますが,もっと泥くさいソフト,たとえば最後はどうしてもアセンブラでプログラムしなければならない場合,OOA/OOD で設計したものと最終インプリメンテーションとの整合性はどうなるのでしょうか?

青木淳:
私は,たとえば C でプログラムを書くときも,オブジェクト指向風に書きますよという考え方です.明日からアセンブラで書けといわれたら,アセンブラでオブジェクト指向風に書いていくでしょう.

落水浩一郎:
まだまだ議論はつきませんが,時間が来ましたので終わりにします.

8. 各グループの討論経過

1. PWB グループの討論経過

報告者
下津 直武
(静岡大学)

0. 参加メンバ

問題解決者	木村 茂和	日商エレクトロニクス
	池川 哲夫	アスキー
	飯塚 宣男	カシオ計算機
	藤井 康範	MHI エアロスペースシステムズ
	似内 均	岩手電子計算センター
	青柳 和久	東北 SRA
	館山 純	SRA
	奥村 吉彦	CSK
	小林 透	NTT
インストラクタ	足立 太郎	PFU
コンサルタント	菅野 卓矢	岩手電子計算センター

1. ツール説明

前日の解説では説明が十分でないということで，PWB のデモ（病院の受け付けシステム）を見て PWB に対する理解を深めることになった．足立さんのデモ操作を見ながら，参加者が質問をするという形式で行なわれた．以下のような質問が出た：

(1) UIM の時計は処理時間と関係があるのか？
(2) UIM と EOPM 間のリンクのはり方はどうするのか？
(3) 遷移に関係あるものだけリンクするのか？
(4) UIM の細かい設定（数字の比較，入力条件の検査）は可能か？
(5) 開始から終了までのチェックはどのエディタでも可能か？
(6) IOPM において並列同期のアニメーションは可能か？

また，全体的なツールの印象として，「ユーザの見えないニーズを生み出してしまわないか？」という意見が出された．

2. 課題の決定

課題は多数決の結果，「ガス給湯システム」に決定した．

3. 課題の理解

全員で問題を一通り読んだ後，どのように課題を解いて行くかについて議論がなされた．インストラクタの足立さんから「最初は FSM を用いて解析するのがよいのでは？」という提案があり，その方向で分析を始めることになった．そのさい，次のような意見が出た：

(1) FSM で順序関係を記述する能力は？
(2) コントローラに関する機能を洗い出そう．
(3) 主要機能（給湯・お湯はり・追いだき・追いだき中の給湯）を出発点にしよう．
(4) 機能間に何か関係はないのか？

問題中の 4 つの大きな機能（お湯はり・追いだき・給湯・お湯はり，追いだき中の給湯）を出発点として考える

という方針に決まった．最初に「給湯」の機能から分析を始めた．そのさい，次のような意見・質問が出た：

(1) FSM に階層構造は？
(2) FSM の文法がわからない．
(3) 条件を FSM でどうやって書くのか？
(4)「給湯」という仕事をいくつかの機能に分割するほうがよいのではないか？
(5) 状態遷移に機能とオブジェクトを加えるという方向で考えてはどうか？
(6) FSM にこだわらずにあげてみよう．

結局，「状態の遷移だけ取り出してその間の関連づけをしてみよう」ということになった．

4. 状態遷移図の作成

4.1 状態の認識

とりあえず，FSM による分析を断念し，給湯機能の状態を並べてみて状態遷移図 (EOPM) を作ることにした．給湯，お湯はり，追いだきの 3 機能に関して，次のような状態を認識した：

□ 給湯
・電源 ON ・電源 OFF ・給湯栓を開く ・燃焼ランプ点灯
□ お湯はり
・電源 ON ・電源 OFF ・お湯はりスイッチ ON ・切替弁が浴槽側 ・バイパス弁開く
・バーナー点火 ・燃焼ランプ ON ・お湯はりスイッチ OFF ・湯量設定オーバー ・設定温度内
・設定温度オーバー ・追いだきスイッチ OFF
□ 追いだき
・追いだきスイッチ ON ・切替弁が浴槽側 ・バイパス弁開く ・バーナー点火 ・燃焼ランプ点灯
・システム停止 ・出湯 93C ・出湯 93C 外 ・追いだきスイッチ OFF

4.2 各機能の状態遷移図の作成

3 つの機能に関して，それぞれグループに分かれて状態遷移図を作ることになった．そのさい「給湯」グループでは次のような議論があった：

(1) 電源 ON は前提ではないのか？
(2) 比例弁とは何か？
(3) 状態を初期状態と点火状態の 2 つにくまとめるのがよいのではないか？
(4) 状態遷移図は階層化してもよいものなのか？
(5) わずかな時間内の事象はどう表現すればよいのか？
(6) 水量センサーは湯量の調節になぜ必要なのか？
(7) 温度に関して「高・適・低」という状態を作るのか？
(8) 温度を下げるためには水量を多くしなければならない．
(9) 給湯の優先度がもっとも高い．
(10) フレームロッドとは着火を検出するものなのか？

4.3 各状態遷移図を EOPM で記述する

各グループの状態遷移図が一応完成した段階で，EOPM エディタを使って状態遷移図を書くことになった．参加者が本格的に PWB を操作するのはこの時が初めてだったので，次のような点が扱いにくいとの意見が出た：

(1) 全体像を把握できないのでわかりにくい．
(2) 1 ページにすべての状態遷移を書いたほうがわかりやすのではないか？
(3) 図の配置の最適化機能が欲しい．

(4) 紙に印刷する機能がない.
(5) 階層のツリーを一気に見る方法が欲しい.
(6) EOPM エディタは他のエディタに比べて使いにくいのではないか?
(7) 事務処理系の場合, IOPM エディタは必要ないのではないか?

このような意見に対して, 足立さんから, 今回のツールは開発バージョンなのでまだ実現されていない機能(印刷機能等)もあるが, 製品では実現する予定であるとのコメントがあった.

状態遷移図を打ち込んだ後, アニメーション機能を用いて動作の確認を行った. ここまでの結果を中間発表で報告した.

4.4 状態遷移図の検討

次に, 中間発表の OHP で示した個々の状態遷移図について検討した. 最初に「給湯」に関して検討した. この時, 以下のような議論があった.

(1) スイッチは外部イベントなのか内部イベントなのか?
→ 外部からのイベントも書くべきだ.
(2) 自分自身に戻る遷移を書かないのはおかしいのではないか?
(3) 適温に保つ機能は内部状態だが, スイッチが押されたというのは外部イベントではないか?
(4) 割込み状態という状態は書かなくてよいのか?
→ 矢印が割込みを示している.
(5) 状態遷移図ですべての割り込みを記述するのは難しいのではないか?
(6) 給湯準備中という状態はあいまいではないか?

また,「スイッチが押されていても何もしないという仕様が明記されている以上, そのイベントも書くべきだ」という意見に対し, 足立さんから,「下位レベルの状態遷移図で書けばよいのではないか」というコメントがあったが, それに対して,「下位レベルに落とすとかえってわからなくなる」との意見も出た.

次に,「お湯はり」に関する状態遷移図の議論では,「お湯はり中から他のイベント(追いだき)への状態変化もこの時点で書くのか?」という質問に対し,「現段階ではまだ書かなくてもよいのでは」という意見がだされた.

最後に,「追いだき」に関する状態遷移図の検討を行った. そこでは次のような意見が出た:

(1) 3 つの状態遷移図の中では一番わかりやすい.
(2) 電源 ON の状態は電気がコントローラに通じているだけで別に何もしていないのではないか?
(3) 電源 ON と待機中を分ける必要はない.
(4) 燃焼準備中での外部イベントも書いた方がよい.
(5) 燃焼準備状態という 1 つのイベントにできないのか?
(6) 各ハードの状況をいちいちイベントとして書くのは意味がない.
(7) 追いだき給湯中という状態は,「給湯」とは別の状態か?
→ 状態遷移図としては別にしたほうがわかりやすい. プログラミングの立場で考えると同じだ.

また, 点火しなかった場合に関して次の意見が出た:

(1) 点火しなかったら戻るというループでいいのではないか?
(2) 点火しなかったら別の状態を作ったほうがいいのではないか?
(3) 点火しなくても給湯栓を開くという状態に移ってもいいのではないか?

,ti +1 以上のような議論を経て, 最終的に 3 つの状態遷移図をまとめて, 1 つの図とし, その図を EOPM エディタを用いて書いた.

5. ユーザインタフェースの作成

状態遷移図が完成した後，この図を起動するようなユーザインタフェーズを作ろうということになった．ユーザインタフェース構築には UIM エディタを使用した．まず，黒板を使って給湯システムの操作パネルの設計を行った．この時次のような意見が出た：

(1) ボタンは複数にリンクして使えるのか？
(2) たいしたことのないものでも画面にでるとよく見える．
(3) システムの流れと関係ない作業が多い．
(4) UIM と EOPM のリンクはバブルとボタンをリンクするのか？

6. ペトリネット作成

ユーザインタフェースを作成して動作を確認した後，次に何をやるかということになって，IOPM エディタを使って「適温に保つ」という処理をペトリネットを用いて記述してみようということになった．この際，次のような質問がでた．

(1) トークンとは何か？
　　→ アニメーションで動いている黒い丸がトークンの動きを示している．
(2) ペトリネットをどう使えばよいの？
(3) 同期処理や並列処理がこのシステムの場合あるのか？
(4)「適温に保つ」という処理以外でも IOPM が利用できのではないか？
(5)「適温に保つ」という機能はどのモデルを使って表現するのが適当か？
(6) フローチャートになってしまうのでは？

結局，質問は出たが議論は発散してしまい，IOPM エディタを使うことは断念した．新しい作業を行うには時間が残り少かったので最終報告の準備に入った．

2. OOA/OOD グループの討論経過

報告者
小澤 一彰
(静岡大学)

0. 参加メンバ

問題解決者	井上 忠則	シスプラン
	遠藤 晋	アスキー
	高橋 秀行	SRA
	田口 秀成	新日鉄情報通信システム
	平尾 一浩	ヒラタ ソフトウェア テクノロジー
	前 雄介	中央システム
	佐藤 琢美	岩手電子計算センター
	松本 幸三	CSK
	Dabin Matthieu	静岡大学
インストラクタ	青木 淳	富士ゼロックス情報システム
コンサルタント	佐原 伸	SRA

1. 問題選択

問題選択にあたって，対象を制御系・事務系のいずれにするかが論じられた．しばらく話し合いが行なわれた後，青木さんのから次のようなコメントがあった．

「オブジェクト指向に向いているとされる制御系の設計の確認，または事務系の仕事がオブジェクト指向でできることの確認のどちらでもよい」

話し合いの結果，制御系の問題「給湯システム」を対象として選び，事務系の問題である「図書館システム」については，青木さんから解説してもらうことで合意ができた．

2. Smalltalk80 & GrapherGear についての詳細な説明

問題解決に入る前に，青木さんから Smalltalk80 および GrapherGear の操作法についての詳細な説明が行なわれた．そのさい，参加者から次のような質問が出た：

(1) 既存のオブジェクトに，新しいメソッドをつけることは可能か？
(2) 機能を付加していったとき，誤りの所在が自分が定義したのではない部分にあったとき，それは発見可能か？　さらに，それは修正できるのか？
　また，ソースには見えない部分はあるか？(要するに Smalltalk システムのすべての既存のオブジェクトを拡張・修正できるのか？という質問)
(3) オブジェクトは，マルチタスクのように常時動いているが，デバッガは，そのオブジェクトの動きを止めて見るものなのか？

青木さんの答えは，(1),(2) に対しては Yes，(3) に対しては No であった．

3. 給湯システムのオブジェクト指向解析

青木さんが説明した方法にしたがって，実際に給湯システムのオブジェクト指向解析を行なっていくことになった．そのさい，Macintosh II-fx に加え，佐原さんのご厚意により Portable Mac を利用させていただくことになった．以後，グループを２つの班に分けて作業を進めることになり，記録係はパニックにおちいった．そこで，班を特定せず，おもしろそうな話をしている方に耳を傾けるという方針でのぞんだ．

3.1 オブジェクトの認識作業

まずオブジェクトの認識を行なった．最初は問題文や図中で目につく言葉をあげ，それがオブジェクトであろうということで，ツールに登録していった．一通りめぼしいオブジェクト候補がリストアップされつくすと，次のような疑問が出てきた：

(1)「浴槽の排水」はオブジェクトかどうか？
(2)「給水」はオブジェクトかどうか？
(3) 物理的なものだけでなく，データもオブジェクトではないか？
(3-a) 水や湯はオブジェクトか？ そうだとしたら水と湯は異なるオブジェクトか？
(4) 湯の温度はオブジェクトにできるか？ もしできるならセンサはいらないのでは？
(a)「温度」と「センサ」は同じものではないか？

青木さんは参加者にすべてをまかせ，議論が出つくしたところでコメントを与えるという接し方であった．この議論の終りには次のようなコメントがあった：

「本来属性として定義されるべきものまでオブジェクトになっている．(オブジェクトという) カプセルの中に入れるべきものまで外にでている」

その結果，軌道が修正され，ある程度話が整理されたように思う．

オブジェクトの認識の作業が一段落した時点で，オブジェクト認識の方法についての討論があった．最初に，次のような疑問が出た：

(1) 次のどちらのアプローチをとるのがよいのだろう？
  (a) 問題を理解してから解析を行なう．
  (b) 問題は軽く読むにとどめ，先に思いつくオブジェクトを選定する．

これに対して「最初からこの例題のようにシステム全体がきちんと与えられることは少ないのではないか？だから，あまり問題を詳しく理解しない方がよい」という声が聞かれた．すなわち，問題を完全に理解しないまま解析を始める方が，現実に近い感覚になってよいだろうということである．結局 (b) のアプローチをとることになり，作業が続行されることになった．

その他に提案された方法は以下の通りである：

(2) オブジェクトの認識だけでは，それらの間の関係がわからないので，ツール上でオブジェクトを描く場所までは決められないのではないか？
(3) とにかく名詞をあげてみて，その後オブジェクトでないと思われるものをけずる方法でよい．

この過程で認識されたオブジェクトを表1に示す．

3.2 オブジェクト間の関係づけ

一通りオブジェクトが出そろったと思われたところで，オブジェクトの選定作業に一応の決着をつけ，オブジェクト間の関係を定義する作業に入った．オブジェクト間の関係 is-a と has-a に関し，以下のような疑問が出た(ただし is-a は一般化と特殊化を反映したクラスの分類を表現し，has-a は全体と部分を反映した集団を表現する関係である．討論の中で「あるクラスのスーパークラスを見つける」とは，「あるクラス」を一般化したクラスを見つけ，あるいは新しく導入し，両者間に is-a 関係を定義することである)：

(1) 二つのオブジェクト，バルブ A とバルブ B が共通のスーパクラスとしてバルブを持つ場合：
  (a) バルブ A に対するメッセージの受信者はバルブ，バルブ A のどちらか？
  (b) バルブ A とバルブ B が違う機能を持つ場合，クラス－インスタンスの関係で処理できるのか？
(2) is-a, has-a の関係の引き方は？また is-a,has-a の切りわけは？(ツールを使用しながらの討論のため，is-a 関係の認識はエディタ上での「線を引き方」という表現になっている)
(3)「上にひっぱる」ということの意味とやり方は？

(3) の質問に対し，青木さんが以下のような説明を述べた：

「いろいろな状態を表すオブジェクトとして State がある．その下には，特に二値状態を表すオブジェクト BinaryState がある．さらに，この下に例えばバルブのような on-off で表せるようなオブジェクトがあると考えられる」

1班	2班
コントローラ	コントローラ
メインコントローラ	メインコントローラ
風呂コントローラ	風呂コントローラ
出湯サーミスタ	サーミスタ
水量センサ	水量センサ
フレームロッド	水流スイッチ
燃焼ランプ	水量バルブ
運転ランプ	切り替え弁
割り込みランプ	給湯栓
運転スイッチ	バイパス弁
給湯水流スイッチ	湯釜
お湯はりスイッチ	浴槽
水量バルブ	水
切り替え弁	
給湯栓	
バイパス弁	
比例弁	
点火プラグ	
バーナー	
湯釜	
浴槽	

表1 認識されたオブジェクト

これは，抽象化のやり方について，非常に本質的な示唆であった．ここで議論はクラス階層を意識した方向に足並みが揃ってきた．青木さんによって示されたオブジェクト間の関係の例（図 1）を参考にしながら，たとえば以下のような問いかけがあった：

(4) 終端的（センサなど）なもののスーパークラスを見つけられないか？
(5) 温度は量であらわされるか？
(6) 水のサブクラスに，それを計るものがあるのでは？
(7) 図 1 のゲージと並んで計測器というクラスがある．
(8) スイッチやツマミ，ＬＥＤのスーパークラス（すなわち is-a の関係の親）は入力（出力）装置かそれともコントローラかどちらがよいのだろう．
(9) コントローラーの下に入力装置がある．
(10) xx ゲージ（たとえば音量ゲージ）は，図 1 のどこにつけられるか？

ここで，青木さんから次のようなコメントがあった：

「is-a 関係では子は親との違いだけを記述すればよい」（したがって新しいオブジェクトは，最も性質が近いオブジェクトの子となるように is-a 関係を結べばよい）
(11)「コントローラ」が何か（センサ，スイッチなど）を持っているのではないか？
(a)「コントローラ」は「入力装置」「出力装置」を持っている
(b) センサ，スイッチ，LED が「入力装置」「出力装置」ではないか？
(12)「コントローラ」は「計測器」か？
(13) センサやスイッチが「入力」「出力」と分けられるなら「バルブ」はどうするか？

ここまで議論が進んだところで全体による中間発表が行なわれた．このグループは，２つの班が独自に定義したオブジェクト関係図をそれぞれ発表した．

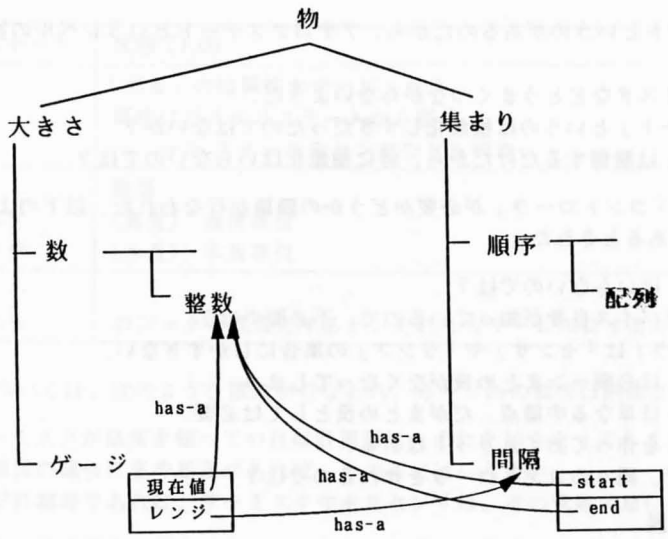


図１オブジェクト間の関係の例

3.3 両グループの成果の擦り合わせ

ここまでの作業は２つの班がそれぞれ独自に行なってきたが，ここで成果の擦り合わせを行ない，一本化することになった．以下，それぞれの図についての発言を挙げる．

(1) 第１班の図はセンサ，バルブという分類でなく，
(2) 第２班の図は，終端部の上に，統一された上位概念センサ，バルブを用いている．「バイナリステート」という２値動作に注目した分類になっている．

この２つの点が両班の成果の最も大きいちがいで，他はほぼ同一の結果であった．第２班の参加者もオブジェクト「バイナリステート」に納得したため，第１班の結果を採用することで合意した．以下は，第１班の成果に対する第２メンバのコメントや疑問である：

(3) フレームロッド等は is-a と has-a を持っている．
(4) この時点ではコントローラなど，まだ不足部がある．
(5) バイナリステートとその下で，持ち上げているレベルが違い過ぎるのではないか？
(6)「計測器」は全て（デバイスが異なっても）値を返すだけである．
  (a) どんな値を返すのか？
  (b) フレームロッドは「バイナリステート」の下にあるのではないか？
(7) サーミスタの働きは二値動作ではないか？

これらの疑問に対し，サーミスタは温度の計測だけを行ない，on-off はコントローラが行なうということになり，サーミスタおよび水温センサは「バイナリステート」ではないことが確認された．この議論を皮きりに，オブジェクトの抽象化の議論が再燃した．

(8) 温度や水量を表わすオブジェクトがいるのではないか？
  (a)「大きさ」の下に温度や水量があり，サーミスタや水温センサはそれを has-a している．
    ＝＞大きさと温度や水量の関係は？
(9) サーミスタや水量センサではなく，「計測器」が温度や水量を has-a するのでは？
  (a)「計測器」が「数」を has-a するのでは？
(10)「大きさ」の下に「数」や「計測器」がつくのでは？

以上のような抽象化の議論に対して，「整数という話（抽象化）は設計に関係ないのでは？」，「こんなところまで論じていたら仕事にならない．どこで止めるのか？終りを認識する必要がある」という意見が出された．

(11) 数と計測器は is-a 関係か？そうでないのか？
(12) 正しいかどうかは別として，少なくとも一段階は抽象化されているので，もういいのではないか？

(13) バイナリステートというのがあるのだから，アナログステートというレベルの抽象化が必要になるのでは？
(a) しかしサーミスタなどとうまくつながらないようだ．
(14)「バイナリステート」というのは抽象化しすぎだったのではないか？
(15)「コントローラ」は制御するだけだから，特に抽象化はいらないのでは？

続いて，オブジェクト「コントローラ」が必要かどうかの議論が行なわれた．以下のような発言が行なわれ，「コントローラ」は必要であるとされた．

(16)「コントローラ」はいらないのでは？
(a) 制御動作はデバイス自身が知っているので，不必要では？
(b)「コントローラ」は「センサ」や「ランプ」の集合にしかすぎない．
(17)「コントローラ」は必要＝＞まとめ役がなくなってしまう
(18)「コントローラ」は単なる中継点．だがまとめ役としては必要．
(19)「コントローラ」を作っておくメリットはある．
(a) 少しの変更で，種々のコントローラを作れるのでは？

3.4 図書館システムの解説

ここで青木さんが，参考のために「図書館システム」について解説を行なった．以下は，それに関する質問である：

(1) 職員と図書館はつながらないのか？
(2)「サービス」が「利用者」から「図書館」へ継承されているのか？
(3) メッセージの送受は has-a に対して行なわれるのか？

3.5 オブジェクト関係図の完成

青木さんの説明を参考に，オブジェクト関係図の決定が行なわれた．

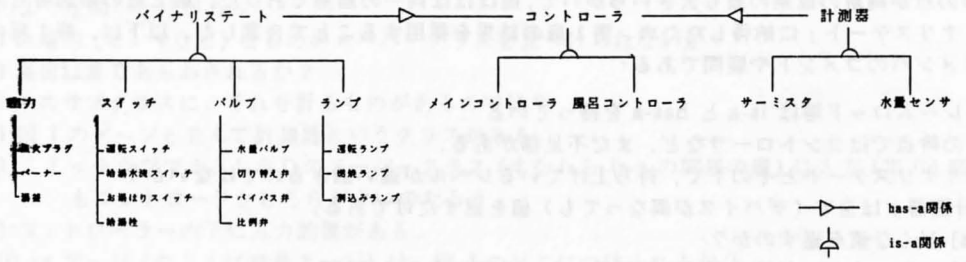


図 2 オブジェクト関係図

以下は図 2 に示されるオブジェクト関係図についての発言である：

(1) これではコントローラはバイナリステートの機能しか使えない？
(2) スイッチ，バルブ等は「入力」「出力」として分けられる？
(3) バイナリステートから下は趣味の問題で，それほど重要ではない？
(4) 湯釜はバーナ等の組合せだから入らない？

3.6 オブジェクトの属性とサービス

オブジェクトの属性とサービスの決定を行なっていった．残念ながら時間の都合ですべてのオブジェクトについてはできなかった．

バイナリステート	状態 (1,0)
ランプ	「光る」のは属性かサービスか？ 属性はバイナリステートから継承 ＝＞ スイッチやセンサなども同様
計測器 サーミスタ 水温センサ	数量 (温度) 温度単位 (水量) 水量単位
コントローラ	ロジックを属性とするか，それともサービスにするか？

計測器については，次のような議論が行なわれ，カッコ内の属性は削除された．

(1) サーミスタが温度を知っていれば計測器は，それを知らなくてもよいだろう．
(2) 計測器の属性に量や範囲があれば，その下にはいらない．
(3) 数が計測器であれば，サーミスタや水量センサは，その意味（単位など）を持つ必要がある．

ここで時間切れとなり，最終報告を行なうためのまとめの作業に入った．

4. 終わりに

最後に報告者からの感想を述べる．まず，参加メンバのほとんどがオブジェクト指向の方法論は初体験だったためか，ツールを使って例題を解いて行くというよりは，オブジェクト指向の勉強会といった雰囲気であった．そのためか，最初から最後まで方法論に振りまわされる感じになってしまった．

しかし，その過程で，第一線のソフトウェア技術者の方々が，言葉によっていかにコミニュケーションを行なうかを見聞きすることができ，たいへん興味深かった．

3. CASE Bench グループの討論経過

報告者
古川 勝也
(静岡大学)

0. 参加メンバ

問題解決者	阿部 和弘	富士ゼロックス情報システム
	藤原 千尋	日本ラッド
	泉田 昭義	岩手電子計算センター
	思坊田 和典	日本電気ソフトウェア
	松浦 光哉	中央システム
	上田 賢一	SRA
	斉藤 基	富士通ビーエスシー
	財田 雅史	ケーシーエス
	大岡 俊之	東北コンピューターサービス
インストラクタ	渡部 健次	ソニー・テクトロニクス
コンサルタント	天池 学	カシオ計算機

1. CASE Bench についての詳細な説明

渡部さんから CASE Bench の操作法についての説明が行われた．そのさい，参加者から次のような質問があった：

(1) 階層が進んだ時点で，上の階層を見ながら書くことはできるのか？
(2) バブル（プロセス）の数が 1000 個というように多くなってもよいのか？
(3) undo や remove の機能はどうか？
(4) データが大量になったようなときの整理はどうなのか？

渡部さんの答えは，(1) と (2) については Yes，(3) と (4) については No であった．参加者の大勢は「使ってみないとまだ何ともいえない」ということであった．

2. 問題の選択

問題選択にあたって，何問解くか？また，どの問題を選択するかが話し合われた．最初に渡部さんから次のようなコメントがあった：

「２問解くのは時間的にも難しい，問題は制御系と事務系とあるが，CASE Bench は制御系の問題に向いている」

これを受けて藤原さんから，制御系の問題をやってみたいという希望が出たが，日頃仕事で事務系のものを行っている人が多かったため，結局，多数決で事務系問題が選択され，もし時間が余ればもう１問制御系の問題をやってみようということになった．事務系の２つの問題のうちどちらにするかは，多数決により酒屋倉庫問題と決定した．

3. 酒屋倉庫問題の解析

CASE Bench を使って酒屋倉庫問題を解くにあたり，データフロー図を使った経験者が２人と少なかったため，渡部さんから少しコメントいただいた：

・データフロー図（DFD）におけるプロセスが酒屋の仕事に割りつけられる．
・データがシステムの内部のものか外部のものかを，問題文から識別する．
・データの区分をした上で，まずコンテクスト・ダイアグラムを作る．

3.1 データの洗い出しと分類

まず，酒屋問題におけるデータの洗い出しと，それらのデータのシステムに対する内外の区分が行われた（この時点で，システム化対象範囲についても議論された）．最初に，倉庫への品物の入庫情報をどうするかという

問題が議論されたが，話がまとまらず，リーダーの大岡さんの意見で，問題文から忠実にデータの抽出をしようということになった．問題で与えられた図ではデータ不足だという人もいたが，とりあえずデータを書き出して行くことになり，それぞれが考えた結果を泉田さんが次のようにリストアップした：

・積荷票　・出庫依頼（票）　・出庫指示書
・在庫不足リスト　・空コンテナ通知　・在庫不足連絡

このリストに対し，システム化対象範囲も考慮して，「電話連絡はどうおさえるのか？」という問題や，「受付係は在庫の有無の状況を把握し管理する必要があるから在庫係を兼ねており，倉庫係はただ肉体労働をするだけだ」といった意見が出て，電話については出庫依頼票に含まれるとし，受付係と倉庫係の仕事の範囲については先にいった意見にしたがううこととして，話が進められた．

このデータ・リストに対し，一応の形を見る上での簡単な DFD を（すべてのデータを使って）作ろうということになった．渡部さんからそのやり方について，先にバブルを固定しそれにデータの流れをつける方法と，データの流れを考えからその交点にバブルをつけて行く方法の２通りがあるという注意があった．

こうして作られた第 0 バージョンの，簡単な DFD について，データの確認をも含めて以下のような意見が出た：

・入庫時の納品書はデータに入れなくてもよいのか？
・納品書＝積荷票と考えてよいのではないか？
・コンテナをどこの部所で管理するか？
・コンテナへの商品の出し入れは，どちらを中心に考えるのか？
・コンテナ中心で考えるのが問題に忠実では？
・在庫確認は商品について行うものではないか？
・受付係で在庫管理しないと空コンテナ通知が出せない？
・入庫通知，入庫依頼がない？

4. データフローダイアグラム (DFD) の作成

データの分析が完了したところで，次にツールを使ってのデータフローダイアグラム (DFD) の作成を行った．その過程での討論を以下に記す．

4.1 コンテクストダイアグラムの作成

作業を始める前に，渡部さんから，ガイドラインについて，以下のような説明があった：

(a) エクスターナルの決定：
　　エクスターナルに名前をつけて行く
(b) データフローを書く：
　　データの流れを書き入れて行く
(c) プロセス（バブル）を入れる

説明が終った時点で，以下のような参加者からの質問や意見が出た：

(1) ガイドラインはどうやって定めたのか？
(2) プロセスになぜ名前をつけないのか？
(3) ○や□はその中に書く名前の長さによって自動的に拡大縮小してくれるほうが便利だ．

渡部さんの回答は以下の通り：

(1) 図を描く上でのわかりやすさを考えて定めた．
(2) もちろん名前をつけることはできる．しかし，無理な名前をつると後で支障が出るので，慎重に！

こうしてできあがったたコンテクストダイアグラムについては，中間報告参照．

4.2 0 階層のダイアグラムの作成

この段階では，最初に泉田さんによって提案された DFD をツールによって書くだけであった．参加者から，ツールの使い方に関してコメントまたは質問がいくつか出た：

(1) エディタで: 文章を打った後 [space] を入れれた方が便利だ.
(2) エディタで: プロセス等の名前を英文字にするには?
(3) いきなりツールを使うことはできなかったのか?
(4) データストアをどの階層で書けばよいのか?

これらの質問に対する渡部さんの回答は以下の通り:

(2) 英文字のときは無変換を選択する.
(3) 人数等のことを考え, ある程度の前置きがあった方がよいと考えた.
(4) 必要なときに必要な階層で行えばよい.

できあがったた 0 階層ダイアグラムについては, 中間報告参照.

4.3 データディクショナリの作成

データディクショナリ (DD) は DFD に対してデータを定義づけるもので, ここでは未定義のデータに対し (エディタを用いて) 定義づけをして行った.

例: コンテナ=コンテナ番号+ ....

まず, 渡部さんから「データディクショナリを書けば DFD の間違いもわかり, 不必要なプリミティブが削れる」というコメントがあった. 参加者の疑問は次のようなものであった:

・DD での入力が DFD での階層化に影響があるのか?
・依頼者名簿は何? また, 依頼者の Tel No. は?

ここまでの議論と作業によって DFD および DD が一応の完成をみた. 中間報告では, これらの内容についての報告を行った.

5. データフローダイアグラムの再設計

参加者から中間報告における DFD に不満足の声が多かったため, 再設計を行うこととなった. それにさいして, 渡部さんから次のような注意点が上げられた:

・データストアの位置確認.
・データベースとファイルの区別しっかりと.
・サブシステム化がよくなかった.
・基本的にはデータの流れを書き, その接着剤としてバブルを考える.
・データの整理.
・プロセスの名前の確認.

渡部さんのコメントの後で討論が始まり, 中間報告の内容について, 以下のような疑問点があげられた:

(1) 第 0 階層のプロセスは昨日の3つでよいか?
(2) それを変えるとコンテクストダイアグラムはどう変化するか?
(3) 機能とデータの関係が不明瞭では?

(3) を受けて, 渡部さんから ER モデル 1 の説明があった.「ER モデル 1 についての検討は後ほど」ということであったが, 参加者の関心は強く, いろいろな意見が出た.「実際に作ろうとしたとき, ER モデル 1 はすぐにできるものなのか? すぐにできるものならやってみたい」という声あった. それに対する渡辺さんの返答は次の通り:

「ふつうは ER モデル 1 から DFD を作る. それぞれのダイアグラムは, ER で書くとモデルがだいたい近いものになる. データベースが ER モデルに置けるかどうかが問題です. 皆さん頭の中ではやっているのでしょうが, それを実際にやっていくのは大変ですよ」

そうしたやりとりの結果, リーダの大岡さんが「CASE ツールを使う上ではこういった作業が一番大切と思うので, 大変だけど, データを ER 図で正規化して行きましょう」と主張し, 渡部さんも「ERD のほうは大岡さんを中心に昨日のデータディクショナリで一応やりましょう. エンティティの関係づけをしっかりつけて ER モデルを完成できれば一番よい」ということで, DFD と ER モデルの2グループに別れて作業を始めた.

5.1 コンテクストダイアグラムの検討

コンテクストダイアグラムについては，すべての参加者の意見として中間報告時の C/DFD C ONTEXT でだいたいよいのではということであったが，下の階層を検討して行く上で少し変化があった．

5.2 0 階層ダイアグラムの検討

0 階層については，中間報告時の C/DFD 0 に関する討論として，以下のような話合いが行われた：

(1) 在庫不足リストは名前がよくない．
(2) 新しい 0 階層を作る上でバブルは２個ではないか？
(3) 入庫と出庫を管理するバブルがもう１個必要な気がするが？
(4) 入庫のほうはこれ以上分けても仕方がないのではないか？

このような議論の末，0 階層の DFD は入庫の受付と出庫の受付をうけるプロセスが２個となった．できた DFD について 1 個のバブルから出入りするデータフローが多いようだということであったが．とりあえず，ツールによる DFD の作成となった．

そこで，まず渡部さんから「ここで，プロセスの名前を考えてゆくのが大切です．上の階層の分け方が適切でないと下では困難になってきますよ」とコメントがあった．

また，参加者の議論は次の通り：

(1) 窓口の受付を行なうプロセスを作ってはどうか？そうすれば 1 個のプロセスの受けるデータがへる．
(2) 取引先からの，いくつかのデータを窓口にもってくるのはおかしいので，取り除いたほうがよい．
(3) 商品管理のプロセスと窓口のプロセスでは，出庫命令とか不足報告があるはずだ．
(4) データの名前だけの参照か，深い属性のある参照か，を確認してみてはどうか？
(5) データストアの取引先と依頼者名簿は，0 階層ではいらない．1 階層で必要なのではないか？（その結果，データストアの取引先と依頼者名簿は第 1 階層に移すことになった）

参加者からのツールに関する質問は次とおり：

(1) 全体のプライマリの状況を見る方法はないのか？
(2) プリンタへの出力が少し弱いのではないか？

これに対する渡部さんの回答は以下の通りであった：

(1) WINDOW をマルチに切って見るしかない．
(2) シェルコマンドでやるしかない．

この時点で，図 1 および図 2 のような DFD が一応完成した．

これらの DFD に対して次のような検討が行われた：

(1) 図 1 の"出庫受付"バブルは，窓口業務と商品管理の２つに分けたらどうか？
(2) それだと図 2 は別にいらなくなるが....？

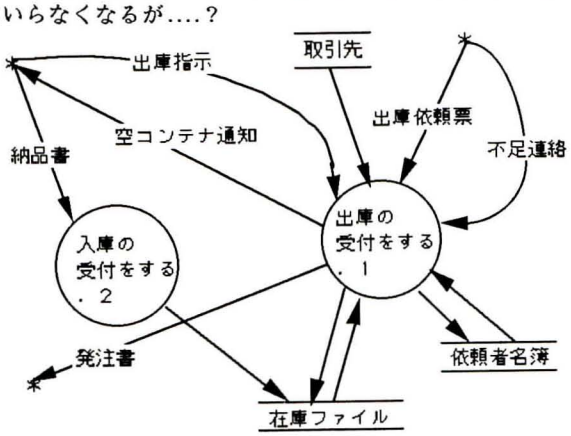


図 1 C/DFD 0 － 酒屋倉庫之入出庫を確認する

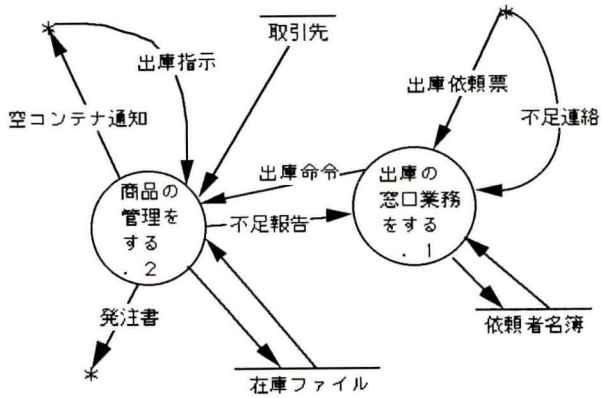


図 2 C/DFD 1 － 出庫の受付をする

検討の末，図 1 を図 3 に変更することとなった．

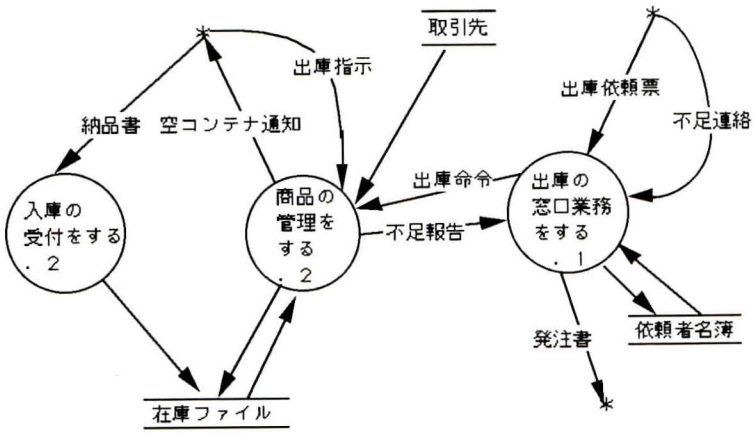


図 2 C/DFD 1 － 出庫の受付をする

この時点で渡部さんから，「図 3 の"商品管理"バブルは，もっと階層をつけて分けた方がよい」というコメントがあった．それについて話しあっているうちに，議論が行き詰まり，ツールを離れて，図 3 について全員で検討した．次のような意見が出た：

(1) 出庫指示に，空コンテナ指示を知らせるニュアンスの空コンテナ搬出マークがあるので，空コンテナ通知はいらないのではないか？
(2) たしかにそれはいえる．
(3) "入庫受付"バブルに，"商品管理"バブルへのインターフェースがないのは，問題の仕様に反するような気がする．これでは入庫したことのチェックが抜けている．
(4) 在庫ファイルは出庫リストを含んでいるのでは？
(5) 在庫ファイルには"商品管理"で対応できるので，"入庫受付"からのアクセスはいらないのでは？
(6) そのアクセスを消すと入庫受付の仕事はほとんどなくなる．しかし，入庫受付を重視すると商品管理の仕事がへってしまう．
(7) ２つのバブルを統合しては？
(8) 1 組の入出庫をバブル１個で対応するというのはちょっとどうか？２つのバブルを統合すると，そのバブルの仕事が膨大になるし，「在庫商品があるか？」という客からの問い合わせに対して，何がいくつあるかがわからないとだめなのでは？客の依頼してきた商品がなかったとき，それをいかに早く用意して出すかを考えることが必要なのではないか？
(9) この段階では２つののバブルを統合しておき，下の階層で分けてはどうか？

こうした討論の結果，結局２つののバブルを統合することになった．

(10) 0 階層において，プロセスの名前をどのレベルまで確定したものにするのか？たとえば，入力出力系の

関係を論理的に分けて見ては？

(11) この時点で名前を決め，論理的にしてみますか？

(12) この時点ではこのままにしておき，もっと下の段階まで行ってから考えればよいと思う．

さらに，商品管理バブルの下の階層について検討していった．これについては，泉田さんと藤原さんからそれぞれ案が出され，検討の結果，泉田案で行くことになった．

泉田案中の"出庫管理"バブルについて，さらに下に階層をのばした．この階層については，藤原さんから出された案を採用することで意見の一致をみた．

話がまとまったところで，ツールを用いて，最終報告に見られるような DFD が作成された．そして，中間報告のときと同様に，データディクショナリが作成された．これについては，一度経験していたので，特に議論はなかった．

5.2 ER モデルの作成

ER モデル 1 に関しては，エンティティごとの関係を確定できないものなどを，それぞれ独立に書いて行くことで作成し直すことになった．

このグループからの作成終了時のコメントは以下の通り：

うまく正規化されないとエンティティどうしの関係が作れない．そこの所でひっかかって，結局のところ中断という形になっている．DFD の方で階層が進むと，データの量が増えて，それを考えるとまとまらなくなってしまうこともありそうで，短時間ではどうしょうもない．

6. モジュール構造図の作成

DFD, ERD, DD が完成したので，ツールを用いてモジュール構造図を作ることとした．この作業は比較的簡単であった．これについての渡部さんのコメントは次の通り：

レベルの複雑度は，１つのモジュールの呼ばれる回数への依存度が高い．あるモジュールについて複雑度が大きすぎたら，階層を下げてそこに制御のモジュールを置く．

この機能に関しては，参加者全員便利であると感じた．

ここで時間切れになり，最終報告の準備にかかった．

4. StP グループの討論経過

報告者
中尾 博司
(静岡大学)

0. 参加メンバ

問題解決者	満田 豊 (B グループ)	シーイーシー
	西垣 郁夫 (A グループ)	東京電力
	清水 洋子 (A グループ)	東芝
	三上 修 (B グループ)	岩手電子計算センター
	山島 利彦 (A グループ)	日本エムアイシー
	浜 恭士 (B グループ)	SRA
	阪本 善通 (B グループ)	SRA
	柳瀬 健一 (A グループ)	ケーシーエス
	築山 秀夫 (A グループ)	ケーシーエス
インストラクタ	桜井 麻里	SRA
コンサルタント	荻原 剛	大阪大学

1. 問題の選択

まず，実際に解決する問題の選択を行なった．選択の基準に関して，

(1) SA/SD の手法で解けるものはどれか？
(2) 事務系の問題か？ 制御系の問題か？
(3) １問だけか？ ２問やるか？

という疑問があげられた．(1) に関しては，今回用意された問題はどれも SA/SD で解くことができるという助言が得られた．また，(2) に関しては，多数決の結果，事務系の方を希望する人数が多かった．(3) に関しては，「まず簡単な問題を全員で取り組み，うまくいったところで２問目をやってもよいのではないか？」という意見が支持された．

そこで，次に事務系の２つの問題（図書館管理および酒屋倉庫）を読みくらべ，どちらの問題を希望するか，自己紹介を兼ねて発言することになった．この時，「酒屋倉庫問題はこれまでにいろいろな手法で解かれているので，それらとの比較ができる」という助言もあったが，結果として「図書館管理」の問題が選択された．

2. 問題解決の方針

次に，この問題をどういうやり方で解いて行くかについて，議論が行なわれた．実際に SA/SD の手法を使ったことのある人はいなかったが，「ともかくツールがあるのだから，それを使いながらやってみよう」ということになった．さらに，今回は端末が２台あるので，「２つにグループ分けをして作業を進め，後でそれぞれの報告しあう」というとになった．

3. A グループの作業

3.1 問題の解き方の説明

まず，桜井さんから，SA/SD の基本的な作業の進め方として，

「システムの外に何があるか，どういう入力や出力があるかという点から，システムの概観図を作成して行くべきである」

というアドバイスがなされ，それにしたがって作業を進めて行こうということになった．

3.2 疑問点の洗い出し

そこでまずグループ全員で問題文を詳細に読んでいったが，その時点で生じた疑問点は以下のようなものである．

・問題に関するもの

(1) この問題文にはあいまいなところがたくさんある．これをほんとうにユーザ要求として解決して行ってもよいのか？
(2) 問題文の中で３番目の要求は，本当は図書館の利用者にも使える機能のはずである．これはこのまま解いてよいのか？（これにちうては，結局このまま解決して行こうという合意が得られた）

・作業の進め方に関するもの

(1) 四角（外部エンティティ）と丸（システム）はどちらを先に書くのか？

・ツールの使い方に関するもの

(1) 日本語の入力方法は？
(2) 辞書にない字がある場合は？

3.3 コンテキストダイアグラムの作成

疑問点を解決したところで，コンテキストダイアグラムの作成に取りかかった．この時点で問題になったのは，以下のようなことがらである（議論は実際にツールのいろいろな機能を利用しながらなされた）:

・エンティティの作成に関するもの

(1) 本の実体は何か？
(2) 本は（データベースの）属性とキーとして整理できるのではないか？
(3) "作者"というのは外部エンティティになるのではないか？
(4) "本箱"というエンティティを用意したが，実際には不要で，システムと利用者だけではないか？

・コンテキストダイアグラムの書き方に関するもの

(1) コンテキストダイアグラムはどのレベルまで書けばよいのか？
(2) 外部エンティティをこのコンテキストダイアグラムにどうマッピングするのか？
(3) コンテキストダイアグラムは，システムとそれに対するリクエスト，応答として考えればよいのではないか？
(4) 入出力は E-R 図で別に書くのか？
(5) 入出力データは１本にまとめるのか？それとも別々に書くのか？
(6)（入出力を）１本にまとめるのと別々に書くのとでは，単に見た目が違うだけではないのか？
(7) コンテキストダイアグラムで書く基準は何か？
(8) "本を借りる"というのは，線で表現するのか？システムの概観図の中にあらわれるのか？
(9) トランザクションに焦点をあてて考えるのか？

・問題の解釈に関するもの

(1) 本の追加や削除は考えなくてよいのか？
(2) 実際に利用する場合は，ユーザも職員に操作してもらうのではないか？

この結果作成されたコンテキストダイアグラムについて，以下のような疑問が出された:

・利用者と職員を区別する必要性について

(1) 利用者と職員を同じものとして考えているが，次のレベルで複雑になるのではないか？
(2) 利用者と職員を区別しておくと，システムへの入力データの識別が容易になるのではないか？
(3)「利用者と職員を区別する」問題は，結局端末が１台だけなのか，２台あって利用者と職員が別々に使えるのかという，システムの稼働する環境を決定する問題ではないか？

・出力について

(1) 出力するものはリストだけでよいのか？検索したものがあったかなかったかの確認はしなくてよいのか？

出力に関しては，「在庫リストは必ず何らかの応答を返すはずである」という意見が出された．また，次に行なう作業として，「データ定義とデータフローダイアグラムはどちらを先に作成すべきなのか？」という質問も出

た．結局，データフローダイアグラムの作成を先に行なうことになった．

3.4 データフローダイアグラムの作成

データフローダイアグラムの作成に関して，以下のような疑問が出された：

・プロセスの書き方に関するもの

(1) プロセスの分け方の基準は？
(2) ２本以上のデータが揃ったら次のプロセスを起動する，という記述はできるのか？

(2) に関しては，No という答だった．また，桜井さんから「使わないデータは渡さないこと」というアドバイスを得た．

「Reject 等の矢印はデマルコの記法で書けるのか？」(荻原)
「書けない」(桜井)

こうしてデータフローダイアグラム 0 を作成し，「次のレベルを記述するか？データ定義を先にやるか？」を相談していたが，メンバーの１人からデータフローダイアグラム 0 に対して独自の案が提案され，ひとまずそれと現在の案を皆で比較・検討することになった．そして２つの案をマージした結果，入力データは１本とし，それを分解するプロセスを挿入することに決定した．その結果，コンテキストダイアグラムは修正され，また以下のような疑問も生じた．

・「データの分解」というプロセスに関するもの

(1) データの分解とは，プロセスなのか？コントロールなのか？
(2) データの分解とは，データを変換しているのか？変換していないのか？

・入力データに関するもの

(1) もともと入力されるデータとして，必要なのはどれか？

・出力に関するもの

(1) 要求１，２はどんな出力を返すのか？(これに関しては，「メンテナンス結果では？」という意見が出された)
(2) 要求３，４，５を職員が行なった場合，結果としては何を返すのか？
(3) 出力のリスト情報の内容はどこまで記述すればよいのか？

・ファイル (データベース) に関するもの

(1) 本と人のデータベースは１つにするか？別々にするか？
(2) ファイルはメモリ上に実現するのか？ファイルを作成するのか？
(3) 参照するだけのファイルというのも入れておくべきなのか？
(4) ファイルへの登録はどうするのか？
(5) そもそも，ファイルの意味は何か？

・ファイル（データベース）の検索に関するもの

(1) リスト要求と検索とは意味が違うのか？
(2) 職員と利用者の検索は，それぞれ行なえることの権利が違うだけなのか？
(3) ファイルへの同時交信も考えるのか？アクセスは１本化するのか？

・データフローダイアグラムの書き方に関するもの

(1) プロセスとファイルの間のデータフローにはネーミングの必要性があるのか？
(2) 検索の時の矢印の向きはどちらか？
(3)「検索のキーを送るが，実際にはファイルの内容を見るだけ」という場合の，"情報を引き出す"という命令は点線で書くのか？

・プロセスに関するもの

(1) 出力がないプロセスは存在するのか？

(2) プロセスの整合性は持たせるべきなのか？

・その他の疑問

(1) トランザクションと，それに必要な情報との関係は？
(2) (SQL のような) 設計言語でやり方が変ってくることはないか？

以上のような話し合いをしながら，データフローダイアグラムを作成したところで，B グループとの打ち合わせの時間になった．

4. B グループの作業

4.1 問題の分析 ＋ コンテキストダイアグラムの作成

B グループでは，問題の分析を行ないながらコンテキストダイアグラムを作成した．

まず，中心に図書館を置き，さらに利用者のエンティティを置くことに関しては全員の同意を得られた．しかし"職員"の扱いに関しては，A グループと同様に，「利用者との違いは何か？」という議論になった．

その他この作業中に出た疑問点は以下の通りである：

・問題の内容に関するもの

(1) このシステムを利用する人は，職員だけなのか？利用者も直接システムを利用するのか？
(2) 問題には，図書館の中で利用されるはずの書類の存在が明記されていない．コンテキストダイアグラムには書く必要がないのか？

・コンテキストダイアグラムの書き方に関するもの

(1) バブルには何を書くのか？トランザクションか？
(2) "職員"の外部エンティティが２つ存在したりすることはないのか？
(3) どの程度まで詳細を書けばよいのか？
(4) データフローの数だけ線を書かなければならないのか？

・ツールの利用法に関するもの

(1) StP ではコントロールが書けないのか？

4.2 データフローダイアグラムの作成

こうして作成されたコンテキストダイアグラムをもとに，次にデータフローダイアグラムを作成した．この時以下のような疑問点が出た：

(1) ５種類の機能要求をここで分割して処理すべきなのか？
(2) 本に関するデータベースファイルが必要になるのではないか？
(3) (利用者かどうかの判定を行なうために) ユーザ名ファイルも必要ではないか？
(4) 出力はリストだけでよいのか？
(5) 利用者履歴をとっておく必要があるのではないか？

(1) に関しては，さらに下のバブルで処理することで同意が得られた．また，(3) については，ユーザ名ファイルという概念を導入することによって，さらに以下のような提案がなされた．

・利用者の種類によって，できることに区別をつければよいのではないか？
・ユーザ名ファイルを用意し，それに従ってユーザが職員か利用者かの判断をするバブルを先頭に入れてはどうか？

この提案を整理し，ユーザ名ファイルを導入する上で，以下のような疑問が生じた．

・ユーザ名 (ID) の登録に関するもの

(1) ユーザ名 (ID) はどこで登録されるのか？それとも ID のない人はいないと仮定するのか？
(2) 登録系は別にあると仮定するのか？

・ユーザ名ファイルの内容に関するもの

(1) ユーザ ID と名前の両方の項目が必要なのか？

・ユーザ名ファイルの利用目的に関するもの

(1) ユーザ名ファイルは利用者と職員の区別に使うのか？それとも，そのユーザがこのシステムを使えるか使えないかの区別に使うのか？

ファイルを導入すると決定した上で，以下のような疑問が出された：

・SA/SD におけるファイルの扱いに関するもの

(1) ファイルの参照系と書込系は１つにまとめるのか？
(2) 同じファイルは１つにした方がよいのか？

・今回の問題に依存したもの

(1) 要求３はマスタファイルで，要求４，５は別のファイルで処理した方がよいのではないか？
(2) １つのデータベースで全部の処理ができるのではないか？
(3) 検索のキーはどれでも使えるようにした方がよいのではないか？
(4) 要求１と２の処理は分けてやった方がよいのではないか？
(5) 要求３，４，５はデータベースのでき方に依存するのではないか？

こうしてデータフローダイアグラムを作成していったが，この時点で要求３，４，５を処理するバブルが１つのものになっていた．要求３，４，５はよく似ているため，この時点では１つにまとめ，下のレベルで分割すればよい，と考えたからである．

さらに，「出力は１の結果 ＋２の結果 ＋３，４，５の結果となる」ということで全員の合意が得られたが，ここで荻原さんから，「トランザクションに対して，３種類の別の出力を作る場合，コンテキストダイアグラムの時点で３種類出力があるようにすべきである」というコメントを受けた．

そこでもう一度コンテキストダイアグラムに戻り，出力部分について詳細に書きなおした．このようなレベルの上下間のバックトラックは頻繁に起こるものなのか，という疑問が出されたが，これに対する荻原さんの答は，「下から上へと，データ構造から考える方法もある」ということであった．

また，「バブルを分ける細かさの基準は？」という質問も出されたが，これは「機能がある程度まとまっていれば１つにする」という答であった．

4.3 データ構造の作成

データフローダイアグラムが一応完成した時点で，データ構造を作成することになった．この時点で問題になったのは，次の点である：

(1) 貸出・返却はどうやって入力されるのか？たとえば，１つの要求が入力されてから本の名前が入力されるのか，それとも同時に入力されるのか？

しかし，データ構造の定義が完成しないうちに，A グループとの打ち合わせの時間になってしまった．

5. 中間報告へ向けての話し合い

ここで，A,B 両グループが集まって，これまでのそれぞれの成果をお互いに発表しあった．

5.1 A グループの発表

コンテキストダイアグラム

・プロセスは１つ．
・外部エンティティは職員と利用者の２つ．
・データの線の本数は，最初は各データごとに１本ずつにしていたが，線がたくさんになることを避けて，１まとまりにした．

データフローダイアグラム 0

・データを１まとまりにしたので，まずそのデータをふりわけるプロセス（バブル）を挿入した．

今後の作業

・データの設計を行なう．

5.2 B グループの発表

コンテキストダイアグラム

・プロセスは１つ．
・外部エンティティは最初利用者と職員の２つにわけていたが，後で１つにまとめた．
・入力は１本，出力は種類ごとに分けてある．
・要求１は１つの入力であるが，さらに分ける必要がある？
・要求３，４，５に対しては入力が１本であるが，分ける必要がある？

データフローダイアグラム 0

・利用者の判断用のプロセスおよびデータベースを用意した．
・データベースの検索は，その内容でかわってくるはずだが，どういう形にするか？
（id，分類，作者，名前，user_id,....）
・出力はそれぞれのプロセスから行なわれる．

データ定義

・少し進めてある状態．

5.3. グループ全体での討論

まず，今後の方針として「今のままで行くか，A,B を一本化するか？」という問題があった．これについては，「データフローダイアグラム 0 ぐらいまではお互いに同意したものを作って，その後また担当を決めて分かれた方がやりやすいかもしれない」という意見もあり，結局それぞれのよいところを吸収して，１つにすることになった．

ただし，２つのループの明確な違いとして，「一般利用者と職員は分けるべきか？」という問題がここでも議論された．「一般利用者と職員はシステムを使う上では同じ」という意見もあったが，荻原さんに「職員だけに返す値はどこかで覚えておく必要があり，面倒かもしれない」というコメントを受けた．「利用者と職員を分けた理由は？」という質問に対しては，「その方が素直だから」という意見があった．が，結局「B グループのデータフローダイアグラム 0 は，A グループのデータフローダイアグラム 1 に入るだろう」という意見が出た段階で，中間発表の時間となった．

6. SA/SD に対するコメント (1) と質疑

中間報告会の後，桜井さんからさらに SA/SD 方法論に関する次のようなコメントがあった：

構造化のポイント

・１枚の図でわかるようにすること．
・名前のつけ方はあまり抽象的なものにしない（内容を連想できるように）．
・周囲からの矢印は少ない方がよい．
・無理してまとめようとはしないこと．

さらに，これまでの作業を通して，以下のような Q&A が桜井さんと StP グループのメンバーの間で行なわれた．

質問 1: StP は今のところワープロと同じようだ．抜けをチェックする機能などはあるのか？
回答 1: 名前のないフローは実体をわかっていないことが多い，入出力フローがあるかどうかをチェックコマンドがある．まず自分でデータ辞書登録を行ない，その後チェックを起動する．チェック内容としては：
(i) 上の階層と入出力の数が同じか？
(ii) 図自体のシンタックスのチェック（フローの中身の定義を含む）
(iii) 図とデータ辞書の日付のチェック

質問 2: ER 図のチェックは？

回答 2: できる．ただしデータ辞書は作ること．

質問 3: プロセスが見えないと書けない図もあるが...？
回答 3: 入力と出力から，どういうプロセスが必要かを考える．

質問 4: 今回のシステムは DB の書き換えになるが...？
回答 4: DB はユーザに見せる必要がない．

質問 5: データフローはどこまで詳しく書くのか？
回答 5:「何をしたいのか？」から，どこまで必要かがわかる．注意しなければならないこととして，How をなるべく書かないこと．

質問 6: データフローからソースコードを書きたいが...？
回答 6: やってはいけない！

質問 7: DB 等が最初に決まっている場合は？
回答 7: まず DB は考えず，要求定義をする．
(i) 現状調査: データフロー図作成．
(ii) 制約の排除，ゆがみをとる．論証モデル作成．
(iii) これから作るシステムが何かをデータフロー図へ．
(iv) 物理的なことを考えたモジュールを作る（物理モデル）．代替案を作って現状とのかねあいを考える．

質問 8: 実現されていないが必要なチェックは？
回答 8: 意味のチェックはできていない．バブルの中でとんでもないことをしていても，外があっていれば何もしない．また，実現可能性のチェックもできていない．

質問 9:（StP の機能として）複数のプロセスをまとめて１つ下のレベルへおとせるか？
回答 9: 無理．ただし，バッファを使ったカット & ペーストはできる．

質問 10: マクロがあれば便利では？
回答 10: きまりきった動作をマクロ定義している．簡単な機能を提供し，それを組み合わせると便利になるかも？

質問 11: レベルの上のものから下へは結果を反映できるが，下から上へは？
回答 11: ツールの基本姿勢として，
(i) 上から下へ
(ii) 勝手になおさない
というものがある．上下どちらが正しいかは，ツールは判断できない．

さらに，桜井さんから以下のようなコメントがあった:

今までは一気に物理モデルを書いていた．これではマシンが変わるとついて行けない．確かに SA/SD では，分析の時間が今までの３～４倍になっているが，手戻りを短かくしている分だけ，全体で見ると短縮化できているはずである．また，ツールのメリットは再利用や検索を支援するところにある．

こうしてグループ討論１日目は終わった．

7. SA/SD に対するコメント (2)

グループ討論２日目は，まず全体の方針として，前日の結果を１つにまとめ，それから分担して今後の作業を行なうという方針を決定した．しかしその作業に対し，荻原さんと桜井さんから，以下のような種々のコメントがあった:

バブル，データフロー，ファイルにつける名前には，よしあしの判断基準がある．それは，「見た時や直す時に内容が理解できるか？」ということである．

そもそもバブルは，何か処理をするのが基本である．だから，「処理する」という名前のバブルは，具体的に何をするのかが不明である．だからバブル名としては，「何を何する」という名前が望ましく，「処理する」や「扱う」という名前は望ましくない．同様に，「受けとったデータをファイルへ書く」というのも，「データ」と書いてはあるが，内容が不明なため，意味がない．

そこで実際に作業を行なう上で，まずデータフローから具体的に考えて行くべきである．データの中身（名前，本の ID,...）が何か？ということについて，その塊をイメージすることが大切である．同様に，貸出・返却が具体的にどういうものかわかるようにする必要がある．こうすることで，データフローダイアグラムについて共通の概念が生まれ，たとえば，「本を管理するにはどういう情報が必要か？」などの議論を行なうことができる．

質問: 名前がつけられなければ，羅列でもよいのか？
答: よい．

ファイルの名前に関しても同じことがいえる．「データベース」とか，「本棚群」という名前も不適である．データベースの項目には，著者名などが含まれていることがわかるような名前をつける必要がある．また，「本棚群」という名前をつけると，実際の本棚のイメージを思い出してしまい，実際に本をやりとりしているかのイメージを持たせてしまう．これは誤解を生じさせる原因となる．だから，本の属性を記録しておくのか，それとも有無だけを記録しておくのか，あるいは両方記録しておくのかがはっきりするような名前づけが必要である．

そもそも，データフローダイアグラムを書く上で一番むずかしく重要なのは，名前づけの作業である．これを慎重に行なうことで，ユーザ要求を反映させることができる．本来，データフローダイアグラムは，ユーザ・分析者間の相互理解のための手法であり，仕様書作りでもある．したがって，ユーザが理解できるように作成することで，手戻りを減らすことができる．

8. 両グループの作業の対応づけ

結局，名前づけも含めて作業をやりなおすことになった．コンテキストダイアグラムでは，システムの扱う範囲を（イメージとして）決めておくことが重要である，という桜井さんのコメントに対し，両グループのコンテキストダイアグラムは表現が違うだけである，ということで合意が得られた．

また，データフローダイアグラム 0 に関しては，A グループにおける「データのふり分け処理」をはずした．

この結果，両グループ間の違いはデータベースの検索の部分だけということになった．この部分に対しては，要求３を「利用者の検索」と「職員の検索」に分け，全体として４パターンの検索を考えることで合意が得られた．

そしてそれぞれのデータ名のつけ直しをしたが，桜井さんから「本来，データ名をつけ，割りふってから処理を決めるべきであり，逆だとデータ名がつけづらい．また，そのようなやり方では，処理と処理を結ぶデータ名になってしまい，これは本末転倒であるから，最初からやり直した方がよいのではないか？」という提案があった．また，最終的にシステムから得られる出力が決まっていなかったこともあって，再度コンテキストダイアグラムの段階から作業をやり直すことになった．

9. SA/SD に対するコメント (3) と質疑

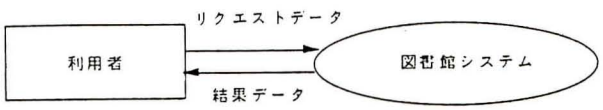
作業のやり直しについては，荻原さんと桜井さんから「最終的にシステムからどういう出力を得たいか？そしてその出力を得るためには，図書館データにどういう項目が必要か？」を考えるとよいだろう，というコメントがあった．そこで全員でこれらの検討を始めた．

その結果，要求３，４，５から出力としてあきらかに必要なものが４種類あり，また要求１，２については，出力を出すための手段をして考えればよいだろう，ということになった．

ここで，メンバーの１人から「コンテキストダイアグラムとデータフローダイアグラム 0 は図 1 のようになるのではないか？」という提案がなされた．

しかしこの図に対し，桜井さんから「これは従来の機能による分け方である．概観図には，リクエストデータという言葉しかない．これを定義する必要がある．また，データフローダイアグラム 0 のプロセス１は入出力が同じだが，何をしているのか？」というコメントがあった．さらに，コンテキストダイアグラムやデータフローダイアグラムの書き方，また SA/SD について，以下のようなコメントがあった:

提案されたコンテキストダイアグラム案



提案されたデータフローダイアグラム０案

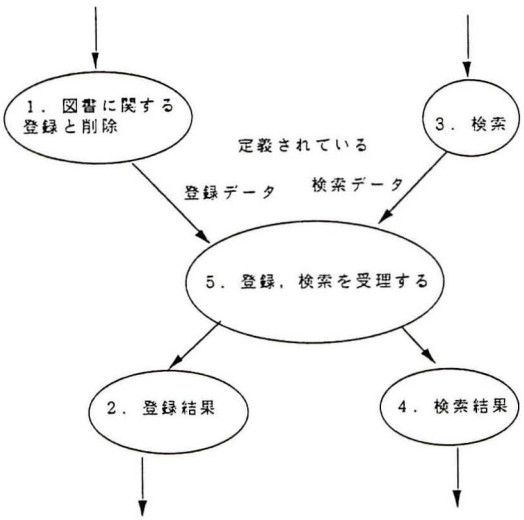


図１修正ダイアグラムの提案

(1) バブルとは機能分解ではあるが，データのやりとりに注目することが必要である．
(2) 一枚一枚の図の中で完結していることが大事である．
(3) コンテキストダイアグラムは，システムの入力として何をどこからもらい，出力として何をどこへ返すかを考えるためのものである．その後で，データや構造をしっかり書く．
(4)「本当に細かいところは後で決める」のと，「一般的な名前をつける」のとは別である．要求の段階で出ているものは，記述すべきである．
(5) コンテキストダイアグラムで外部エンティティとの間に矢印を引く時点で，項目の名前まで原則としてはすべてあきらかにしなければならない．ただし項目の集まりが１つの名前であれば，一緒にしてもよい．
(6) 概観図はシステムの境界を決めるものである．その中に出てくるものを，すべて洗い出す．
(7) 今回の問題では，ER 図は人に関して，本に関してまとめる．検索に関してまとめてはいけない．
(8) 論理モデルでは，細かなエラーは考えなくてよい．「メインは何か？」を考える．
(9) 登録系と検索系はまったく無関係ではない．本の情報どうしの内容が重複していれば，同じにしてもよい．
(10) 同じ情報なら，媒体の区別はしない．
(11) コンテキストダイアグラムの作成は，要求分析をしていることになる．
(12) SA と SD の間にはギャップがある．SA = What?, SD = How?
(13) 図を作成するのは，討論するためである．

これらの指摘に対し，以下のような質問が出た：

質問 1:「人や本」は要求から見つけるのか？
回答 1: その通り．あとはユーザとの論議で見つける．

質問 2: コンテキストダイアグラムの段階で DB が出てくることもあるのでは？
回答 2: それはありうる．その場合は DB を作るのが目的である．

質問 3: この問題では，本の情報は検索の内容に依存するが，どうする？
回答 3: とりあえず，登録のメインの処理は何かを考える．あとはバックトラックで作業を進める．

質問 4: 処理が成功したかどうかを返すため，データフローを引いた方がよいのか？
回答 4: ユーザによる．

質問 5: 今回の問題は，入力はデータ指向であるのに対して，出力は処理指向ではないか？(リストというのは茫然としている．１冊の本と本全体の情報とは別)
回答 5: ある程度仕方がない部分もある．

質問 6: 分析は本来少人数でやるべきではないのか？
回答 6: １人か２人でやるのがよい．ユーザとのやりとりで，説明をする時にコンテキストダイアグラムを使う．今回は分析者とユーザの立場を往復しているようなものである．

10. 作業のやり直し

結局，コンテキストダイアグラムの作成のやり直しを行なった．その際，時間がなくなってきたので，登録・削除に関する機能・情報は削除することになった．その結果作成されたコンテキストダイアグラムは，全員の合意を得ることができた．

次に，データ構造の定義を行なった．これは桜井さんの「データ構造の定義作業を分担してやるなら，本と人間に関して分けた方がよい」というコメントから，A グループが「最後に借りた人に関する情報」，B グループが「本に関するリスト」の定義をすることになった．ここでは，特に問題となるようなことはなかった．

続いて両者の結果を比較した．その際，問題になったのは以下のような点である：

(1)「現在借りている人」と「最後に借りた人」の関係はどうなるのか？
(2)「本に関するリスト」の中に「返却の有無」「最後に借りた人」が入るが，「最後に借りた人」が入力と結びつかない．どうするか？

さらに，データベースの定義を行なった．ここでは，「本が貸し出し中かどうか」の判断をどう扱うか，などが議論の対象になった．しかし，終了時間が近づいたため，詳細の定義を適当に済ませ，StP のチェック機能を実際に利用したにとどまった．

以下，まとめに入ったが，その結果は最終報告を参照されたい．

5. Teamwork グループの討論経過

報告者
関谷 和愛
(静岡大学)

0. 参加メンバ

問題解決者	知念 徹	(三菱電気コントロールソフトウェア)
	富永 伸哉	(ヒラタ ソフトウェア テクノロジー)
	佐藤 啓太	(富士ゼロックス情報システム)
	刀根 利光	(シーイーシー)
	渡辺 瑞枝	(SRA)
	森 彰三	(富士通ビーエスシー)
	吉田 一知	(岩手電子計算センター)
	林 明賢	(SRA)
	大篭 高之	(日本システム)
インストラクタ	田中 耕市	(東陽テクニカ)
コンサルタント	佐藤 千明	(長野県協同電算)

1. 問題の選択

どの問題がよいか希望をとったところ，かなりの人が３番を選択した．これはメンバーの多くが制御系の仕事をしているためである．別の問題を選んだ人からも特に反対意見がでなかったため，制御系問題である３番（ガス給湯システム）に決定した．

2. 問題の理解

続いてツールの説明が行なわれる予定だったが，ツールはあくまでも手段であり，先にどんな問題かを頭に入れておいた方がよいという意見が出たため，ここで問題を読むことになった．次のような意見が出た:

(1) 問題ですでにかなり詳細に設計されている．
(2) まず存在するハードウェアの略図をかいてみればわかりやすい（視覚的アプローチが重要）．

(2) に関しては，この時点から CASE ツールを使っていかなければ意味がないという意見が出たため，次のツール説明の後にしようということになった．

3. ツールの説明

田中さんから Teamwork の操作法の説明が行なわれた．そこでは，日本語処理やグラフィックス関係などの質問が出た．基本操作以外にも以下のような質問があった:

(1) チェック機能は十分か？（図の間違い，シンタックス，データの一貫性等）
(2) メモ用のお絵書きツールは付随しているか？
(3) ツールを使って問題を解く時，どのくらいの人数が適当か？
(4) 多人数だとかえって意見の不一致などで効率が悪いのでは？
(5) どのような役割分担が適切か？

4. コンテキスト・ダイアグラム (CD) の作成

入出力情報をあらいだし CD を作成していった．この際，以下のような意見や質問が出た．

(1) スイッチ類を BC(バス・コントローラ) としてまとめてよいか？
多数のランプ信号を１つの矢印でまとめては？
入力をまとめて考えておいて，あとで詳細化してはどうか？
→全てあらいだした方がよい．
「湯釜」としてまとめてあるのはどうするのか？
→ DD で詳細に定義する．

CD は詳細にしたほうがよいか，まとめた方がよいか？
→場合によってやりやすい (見やすい) 方にする．
(2)「ガス」や「水」は入力か？
→これらは物理的なもので「情報」ではない
(3) 同じ名前のデータがあってもよいのか？
TOOL 上では大丈夫か？
→ Yes. まったくく同じ意味ならよい．
消火も含むので，「点火信号」より「燃焼信号」の方がよい
名前の定義はあとでかさならないようにした方がよい
「シグナル」と「信号」という名前を使っているが，統一すべきだ

作成が終了した時点でツールによるシンタックスのチェックを行ない OK となった．

5. データ・フロー・ダイアグラム (DFD) の作成

DFD 作成にあたり，全体的な質問として次のようなものが出た．

(1) 初期処理は？
→機能ではないのでバブルにはならない．
(2) 全体を制御するものは？
→コントロール仕様でかけばよい（状態遷移表など）．

5.1 主要バブルの決定

まず，主要な機能をもとにいくつかのバブルを作り，そのあと１つどこかのバブルを選んで，そこからデータの流れを追って書いていこうということになった．主要バブルとして次の３つをの洗いだした（A 案）:

1. 給湯（給湯せんに給湯する）
2. お湯はり（浴槽にお湯をはる）
3. 追いだき（追いだきをする）

ところが，ここでデータの入出力を書き込もうとしたところ，どうもうまくいかない（たとえば「水量」というデータはすべてのバブルにはいってしまう）．そこで，バブルの設定が悪いのかもしれないということになり，次のような別の考え方にもとづく主要バブルにしてみては，という意見が出た（B 案）:

1. 温度調節
2. 水量監視
3. 弁の切替え

ここで非常に長時間にわたる議論が行なわれた．以下に主要な意見を記す:

- A 案はシステムの外側から見た機能による分類である．それに対し，B 案はシステムの内側から見た(すなわちモジュール化などを意識した）機能分類である．
- A 案のバブルが B 案のバブルをコントロールするようにしてはどうか？
- A 案でやっても結局下の階層で B 案のバブルが出てくるのでは？
- 第１段階のバブルの認識が一番難しい．
- ２つの観点で作ってみてはどうか？
- モジュールとバブルの関係は？
- 始めにバブルから考えたのは間違いだったのではないか？（データの流れを中心に考えればよかった）

結局，意見がまとまらなかったので，全員がとりあえず DFD を書いてみてそれから考えるということになった．

5.2 作業方針の決定

各自の書いた DFD を全員で分析した結果，いずれも先ほどの A 案か B 案のどちらかに分類できることがわかった．ちょうど人数も半々に分かれたので，マシンも２台あることだし，２班に分かれてやってみようということになった．班の編成は次のようになった．

第１班（温度調節，水量監視などに分ける）

刀根，吉田，林，富永，（田中）
第２班（給湯，お湯張りなどに分ける）
渡辺，大篭，森，佐藤，知念

5.3 DFD の作成

２班に分かれてからは，それぞれの考えにもとづいて，作業は順調に進み出した．作業中に出た質問や意見を以下に記す：

第１班
- ・システムが今どんな状態になっているかが不明（お湯張り，給湯など）．
  →コントロール・フローがしっかりしなければならない．
- ・火力と水量の調節は分けては考えられない．
  →温度調節という１つのバブルにする．
- ・データのチェックが手作業よりはるかに楽だ．
- ・どこまでバブルを細かく分けたらよいのか？
- ・バブルはすべて同時に動いていると考えるのか？
- ・点火はどのレベル（階層）におけばよいか？

第２班
- ・せっかくツールがあるのだから議論ばかりしていないでどんどんやってみよう．
- ・作業は順調に進んでいるが果たして動くのか？
- ・制御系はデータ・フローを書くのが難しい．
- ・バブルを中心に考えたので，バブル同士が全然つながらない．
- ・作った後で，同じ機能のバブルがあるから上の階層にあげようということが，よく起こった．結局，この「共通部品を上にあげる」という手順を繰り返すうちに，第１班の考え方に近づいてきた．
- ・ツールがあるおかげでバックトラックが簡単だ．

感想としては，第２班の方が作業が順調に進んでいたようだ．第１班はバブルの認識（どこまで細分化するか，どれをまとめるか）に苦労していた．そして２班の作業が進むうち，DFD が１班のものに近くなっていった．

ここまでで，２班ともに一応 DFD ができあがった．

6. コントロール・フローの作成

最後に，リアルタイムデータフローの制御部分であるコントロールフローを作成した．まず田中さんから制御のための各図表類 (PAT,DT,STD) の作り方の説明があった．ここで：

- ・どんな順序で作るのか？
- ・どこから考えていくのか？
- ・DFD の階層との関係は？
- ・階層ごとに作るのか？

といった基本的な質問が数多く出た．これは，DFD に比べ制御関係は非常に複雑だからだろう（シンタックスなど）．

作成途中では，以下のような質問や意見が出た：

- ・システムの異常への対応は？（エラー処理）
- ・DFD に書いてある信号，STD に書いてある信号，それらが各レベルに展開したりしているのでとてもわかりにくい．

とにかく，このようにしてなんとか制御部分まで完成した．ここでツールを使って各種の図表を出力したり，エラーをチェックしたところで時間切れとなり，まとめにはいった．