グループ3報告書

R-1·グループの活動概要

1. メンバの簡単な紹介と役割分担

氏 名	会社名／所属	グループ内での役割	業務傾向／特徴 他
酒井雅裕	(株) ハイテック	グループリーダ R-1 担当　最終発表担当	パソコンの CAM アプリケーション開発
吉田恵	(株) さくらケーシーエス	R-2 担当 CRC カード	Mac 上のアプリケーション開発
渡辺清文	オリンパス光学工業 (株)	R-3 担当	パソコンによる医療用アプリケーション開発
伊藤徹	アドバンスト·ソフトウエア·インスティチュート	サブリーダ R-4 担当	UNIX 上のアプリケーション開発
井上忠則	(株) シスプラン	R-5 担当 フォーラム発表	UNIX,DOS,MAC のアプリケーション開発

2. 使用した手法名

ＣＲＣカード

3. 解いた課題（特にグループ独自で拡張した部分に関する説明)

·一度に借りられる冊数が未設定であったので、３冊と設定した。
·貸出期間は２週間に設定した。
·同一名の本の重複を認めた。(ベストセラーの本は何冊もあるため)

以上が 9/3 の午前中

·端末は１台とする。
·利用者、本には各自ＩＤ＝バーコードを所有する。
·利用者のデータベースは、すでに他のシステムによって確立されているものとする。

以上が分析フェイズでできたもの

4. 作業プロセスの概要

9/2		顔合わせの討論のみ
9/3	午前	ＣＲＣ法に関するチュートリアル 図書館問題の前提条件 …… ０．５ｈ
	午後	クラスとは論、グループ内の役割と分担 …… ２．０ｈ (中間報告) 分析フェイズ …… ５．０ｈ
9/4	午前	分析·設計フェイズ …… ３．０ｈ
	午後	設計フェイズ …… ２．０ｈ 報告書作成（クラス詳細） …… ４．０ｈ ·前提条件の分析はほとんどしなかった。 ·分析はウォークスルー形式で行った。 ·クラスの詳細を設計していて、インタフェースの疑問が発生した。

R-2･分析内容（吉田恵）

1. 着目点、アプローチの概要

Ｇ３グループでは、オブジェクト指向分析、設計で例題を解いていこうということだったので、オブジェクト指向的なアプローチをするという方針で進みだした。どういうことか簡単に言えば、細かいところに最初からこだわらずに、とにかくやってみようということであった。

特徴は分析というフェーズにおいても、ＣＲＣカードを使用することである。又、図書館問題全体に関する分析も大まかに行なった。

９月３日午前中、オブジェクト指向に関するレクチャを受けた後、昼食までの３０分ほどで、各自の図書館問題の例題に関する不明点の洗いだしを行なった。他のグループに比べて、この時間はかなり短いもののようであったが、後から考えると、この時間内でかなり不明点は消化されていたようだ。

午後（と言っても昼食、散歩後３時頃）からは、実際に分析作業に入ったが、まずそこで問題になったのは「クラスとは何か」という、“クラス”についての定義であったので、“クラス”についての、メンバ全体の意志統一をする必要があった。

そこで出た結論としては、実際にシステムが完成したと仮定して、その画面のメニューとなるものがクラスではないかということで、スタートしたが、メンバのコンセンサスはとれないまま中間報告に入った。

この時点でのＣＲＣカードには、クラス名のみが記入された状態であり、かろうじて“本”というクラスにのみデータがあるといった具合だった。とにかく思い付くままカードを作ったという状態である。

中間報告後は、午後９時よりシナリオを作成しながら、シミュレーションを行なうことにより、結果的には分析フェーズでＣＲＣカードを活用していた。

これと並行して、分析過程において不足していた、不明点の洗いだし部分も補え、ここにきてようやくメンバのコンセンサスをとることができた。

仕様を（３）までシミュレートして午前２時になりかけたので明日に持ち越した。

９月４日午前９時より残っている（４）、（５）の２つの仕様をシミュレートして、一応の分析作業を終了し、メンバの確認を得た。

Ｇ３グループの大きな特徴として、分析、設計に関してスケジュールを詳細に決定してから進みはじめるのでなく、その日の大まかな進捗スケジュールを決定した後は、とにかくやってみるという姿勢で取り組んできた。

2. 分析で困ったこと（発生した問題点）

２日目の中間報告に至るまで、とにかく思い付くままＣＲＣカードにクラス名を書いてゆき、カードが激増した。その後いろいろなＰＣの方が来られては、少しづつコメントがあり、それに流されるような格好で、カードの数が増減を繰り返した。あるいは、処理範囲が広がり過ぎたようなことが起こった。

これは、先を見通すことを完全にしないまま“とにかくやってみよう方針”だったので、よく見えないものの中を進んでいったことによるのではないかと思う。

3. 解釈／意見の相違の扱い

第一に、オブジェクト指向自体をメンバ全員が完全に理解していない状態で取り組みはじめたので、考え方を統一していなかった。

第二に、自分達が実作業を通して今まで私用してきた技法（特に構造化）とオブジェクト指向との決定的な違いを理解するのに時間を必要とした。（どうしても階層的なモジュール作成をしたがる）等々の問題があった。

それらの処理方法として、データを隠蔽しよう、カプセル化しようという構造化では、フリーにアクセスできる部分と制約しながら“本”というデータを管理する、“本管理”、“分野管理”、“作者管理”、“ユーザ管理”というクラスを設けて、それぞれの管理者が、その下にぶら下がっている、“本”、“分野”、“作者”、“ユーザ”というそれぞれのデータにアクセスしに行く。

言い換えると、データと手続きがカプセル化されており、そのデータを触ることができるのは、手続きしかないという、最終的結論に至った。

4. 作業の手戻りはどの程度あったか？（プロセスを書き留める）

まず、発生した手戻りは、クラスの追加、削除に影響する作業で、はじめに“本”というクラスと、その上の“本管理”というクラスを別々のクラスと考え、本のデータをアクセスするのは“本管理”のみということで考えたが、“本”はデータであって、クラスではないという討論の結果、“本”は“本管理”のクラスのカードの裏に書かれるデータであって、クラスとしては存在しないという結論が出た。その上で、例えば、“本管理”というクラスにしかなかったものを“作者管理”と“分野管理”の３つに分割したことである。

その原因は、初めの見通し不足によるものであった。

その対処は、実際にＣＲＣカードを使ってブレーンストーミングを行なってみるということである。

ここでも、“とにかくやってみる方針”に、のっとっていたのである。

5. その評価、問題点

【評価】

・分析の評価として、我々はやはり、一番初めの時点におけるスケジュールの見通し不足があった。又、クラスというものを、メンバが完全に理解していないまま、それでもこの場で考え込んでいても仕方がないので、とにかく進んだことに意義がある。

・分析は、必要以上にする必要はない（どこまでを必要とするのかということの区切りは難しいが)。もしも、イメージづけのためにと言う理由で、必要以上に記されていない仕様を考えたとしても、それを考えているときの考えているメンバは真剣かも知れないが、第三者的に見ると、なんとも無駄な時間を費やしているようにしか見えない。その結果として、ガチガチの仕様の設計をしかねないということである。

・考え、設計共に、柔軟性がなければならない。予想される仕様変更に、なるべくわずかな労力で、耐えうるように作る必要がある。仕様の拡張や、サービスに耐えていかなければならない。

【問題点】

・分析段階から全体が見えてくるまでに時間がかかる。

・今まで実作業でしてきたことと、どこがどう違うのか。(何がオブジェクト指向なのかがわかりにくい)

6. そこでの発見など

・ＣＯＢＯＬで作業しているメンバが一人もおらず、Ｃで作業しているメンバが多かったので、オブジェクト指向的な発想、分析方法に、違和感なくすんなりなじむことができた。

・構造化での分析、設計で壁にぶち当たったときにはじめて、オブジェクト指向の利点が理解できる。

・図書館問題ぐらいのスペックなら、構造化での分析、設計で充分といえ、又、そのほうが今までの作業でも慣れているので、考え方が流されてしまいそうになる部分がある。

7. 分析結果または主要なワークシート

ＣＲＣカードを参照してください。

R-3·設計（渡辺清文）

1. そのプロセス

Ｇ３ではＣＲＣカードを用いて設計·分析を行った訳だが、その作業方針は「とにかくカードを作っていって考える」という、シンプルで、力強く、与えられた技法の強みを最大限に生かす、効果的で分かりやすいものであった。

ＣＲＣカードを用いた分析·設計は、その行程が混然一体としており、明確に分離することは（他の手法と比較して特に）難しい。クラスの抽出がすべて終わった段階で設計もまた終了し、あとは細かな修正と、カードの内容を設計書に書き移すだけである。

シミュレーションが終了した時点でクラス名その他の見直し、決定をした。このとき９月４日（３日目）の昼食前だったのだが、ほとんど余裕で、終わったつもりでいた。

その後ＣＲＣカードをもとに設計書を書き始めた。設計書に書く途中でフォーマットの問題と、メッセージの決め方の問題が発生したので、その段階でさらにクラス定義を明確にし設計書フォーマットを再決定した。

全体として Trial & Error で問題が発生したら、その場で解決するというやり方を取った。

2. 設計で困ったこと

・データの決定

たとえば、利用者は自分の借りている本を知っているという判断から、利用者データの中には借りている本が含まれることにした。(ユーザー管理クラスのデータとして利用者が存在するのであって、利用者クラスがあるわけではない。）この時利用者データの中にあるのは本の名前か、それとも本のＩＤかという問題があった。

この問題に対しては、本のことを知っているのは本管理クラスであって、他のクラスは本ＩＤによって本管理クラスに「しゃべって」もらう、ということで解決した。つまり利用者が知っているのは借りている本のＩＤだけということである。

ここで重視したのはデータの隠ぺいということである。

・クラス名のつけ方

クラス名は分かりやすくシンプルに、というのが方針である。

しかし分かりやすさとシンプルさは必ずしも一致しない。自分の借りている本のリストを出すという機能が問題の中にはあるが、この処理を行うクラスに対するクラス名として次のふたつが考えられる。

１　リスト作成処理

２　借りている本のリスト作成処理

１は名前の短さを取り、２は分かりやすさを取った。結果としては２を選んだ訳だが、その理由は２の方が分かりやすく、１では他のリスト処理との区別がつかないというものであった。

・メッセージ名のつけ方

メッセージ名は同じ機能のものはすべて同じ名前にするという方針で決めていった。たとえばデータをあいてのクラスに教えるときのメッセージは「しゃべる」である。

ただし貸すと借りるのような対応関係にある動詞は明確に区別した。

3. 解釈／意見の相違の扱い

・名前の付け方の問題

２であげたような問題が出たわけだが、分かりやすさとデータの隠ぺいという方針が優先した。単に声がでかいものの勝ちという面もあるが、話し合いというのはそういうものなので、多少は仕方がないだろう。

4. その評価、問題点

・ＣＲＣへの記述方法

我々のクラス分けでは、大きく二種類のクラスが存在している。ひとつは協力してもらうクラスがなく、ひとつのメッセージに対してその機能が一行で書き表せるようなものである。これは本管理クラスなどが例としてあげられる。

一方、いくつものクラスと協力し、メッセージに対して一連の機能を実行することで答えるクラスである。たとえば貸出処理クラスなどがあげられる。このクラスは、本管理、ユーザー管理、バーコードリーダ、表示、など複数のクラスと協力し、本の貸出を実現する。

このふたつのクラスでは、Responsibilities の項目の一行の意味がまったく違う。前者では一行あたりひとつのメッセージに対する処理で、後者ではひとつのメッセージに対する一連の処理のひとつにすぎない。

この違いを把握していなかったので、設計書を書くときになって混乱が生じた。

この違いを区別するためには、メッセージごとに'*','#' など、何でも良いが行頭に印をつけると良いと思われる。

例：

	(Responsibility)	(Collaboration)
＊1	メッセージ1	
	…して	＊＊＊クラス
	…して	＊＊＊クラス
	…する	＊＊＊クラス
＊2	メッセージ2	
＊3	メッセージ3	
	…して	＊＊＊クラス
	…する	

・書くことの大事さ

ＣＲＣカードを使うときは、頭で考えるよりもとにかく書き出してみることが重要だ、ということでグループの考えは一致した。

・カードに書けない情報がある

クラス間の関係は記述できない。協力してもらうクラスは Collaboration の項に書けば良いが、それだけではわかりにくい。またスーパクラスーサブクラスの関係はまったく記述することができない。継承に関しては対応していないので別の技法を利用する必要がある。

・ＣＲＣ十模造紙またはホワイトボード

上の問題について簡単に解決する方法は、模造紙やホワイトボードの上にカードを並べ、マーカーで線を引いてクラス間の関係を図示するという方法がとれる。

5.そこでの発見など

・変更が簡単・柔軟

ＣＲＣ法は、大枠の構造さえ決まれば、多少の仕様変更には簡単に対処できる。

・全員参加に意味がある

分析・設計は一人でやった方が速い。しかし他の人と分業で仕事をするときは、その人達とブレストをやった方が設計書だけを渡すよりもあとの作業がはかどるのでは。

・データの隠ぺいは当たり前（の人もいる）

データの隠ぺいはオブジェクト指向言語でなくても可能であり、実際Ｃ言語で普段からそうしている人もいる。オブジェクト指向のよりすぐれた点について、もっと具体的に知りたかった。

6.主なワークシート

ＣＲＣカード参照

R-4·設計書（伊藤徹）

1. 全体構造

端末（又はバーコード）から入力されたユーザ ID を元に“一般利用者”か“職員”かを判別し、操作メニューを表示する。使用者は、表示されたメニューから自分が作業したい項目を選択し、モニターの指示に従って作業を進めて行く事によって目的を達成する。以下に簡略図を提示する。

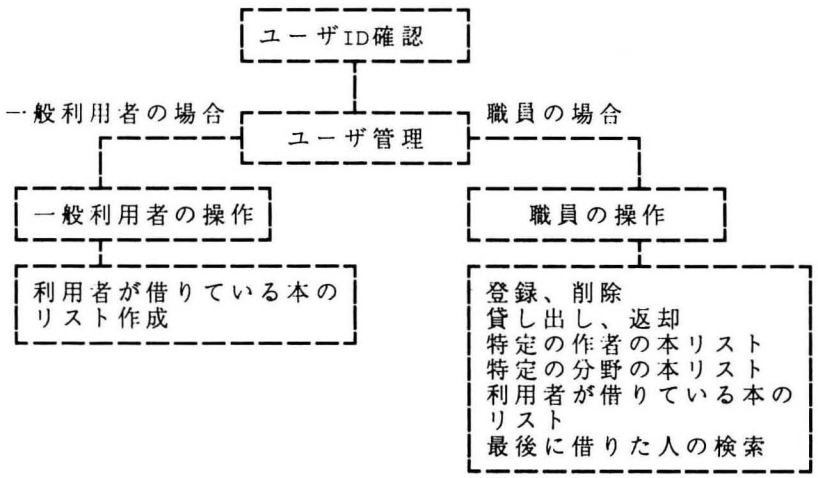


図 1

個々の処理の実現方法は以下のとおりである。表記上の記号は以下の意味を示している。

“·”は一つのクラスを示している。

“⇒”は右側のクラスにメッセージを発行し処理を依頼している事を示す。

“⇔”は右側のクラスにメッセージを発行し処理を依頼し、戻り値を受け取っている事を示している。

個々のクラスについては後記述してあるので、参照の事。

（1）本の貸出し、返却の全体構造

a. 貸出

·ユーザ ID 確認　⇒　·表示
↓　⇔　·キーボード（もしくはバーコードリーダ）
·ユーザ管理 ⟶ 一般利用者の場合
↓
職員の場合
↓
·職員の操作　⇒　·表示
↓　⇔　·キーボード
貸出を選択された
↓
·貸出　⇒　·表示
⇔　·バーコードリーダ（キーボード）
⇔　·ユーザ管理
⇔　·表示
⇔　·本管理

b. 返却

·ユーザ ID 確認 ～ ·職員の操作 は上と同じ
↓
返却を選択された
↓
·返却　⇒　·表示
⇔　·バーコードリーダ（キーボード）
⇔　·本管理
⇔　·表示
⇔　·ユーザ管理

(2) 本の登録、削除の全体構造

a. 登録

·ユーザ ID 確認 ～ ·職員の操作 は上と同じ
↓
登録を選択された
↓
·登録 ⇒ ·表示
⇔ ·キーボード
⇔ ·本管理
⇔ ·作者管理
⇔ ·分野管理
⇔ ·バーコードライタ

b. 削除

·ユーザ ID 確認 ～ ·職員の操作 は上と同じ
↓
削除を選択された
↓
·削除 ⇒ ·表示
⇔ ·バーコードリーダ (キーボード)
⇔ ·本管理
⇔ ·作者管理
⇔ ·分野管理

(3) 作者別、分野別の本のリスト表示。

a. 特定の作者の本

·ユーザ ID 確認 ～ ·職員の操作 は上と同じ
↓
特定の作者の本リストを選択された
↓
·特定の作者の本リスト
⇒ ·表示
⇔ ·キーボード
⇔ ·作者管理
⇔ ·本管理
⇒ ·表示
⇒ ·印刷 (必要なら)

b. 特定の分野の本

·ユーザ ID 確認 ～ ·職員の操作 は上と同じ
↓
特定の分野の本のリストを選択された
↓
·特定の分野の本のリスト
⇒ ·表示
⇔ ·キーボード
⇔ ·分野管理
⇔ ·本管理
⇒ ·表示
⇒ ·印刷 (必要なら)

（4）利用者が借りている本のリスト表示。

a. 一般利用者が操作する場合

·ユーザ ID 確認　⇒　·表示
↓　⇔　·キーボード（もしくはバーコードリーダ）
·ユーザ管理 ⟶ 職員の場合
↓
一般利用者の場合
↓
·一般利用者の操作
↓　⇒　·表示
·利用者が借りている本のリスト
⇔　·ユーザ ID 確認
⇔　·ユーザ管理
⇒　·表示
⇔　·ユーザ管理
⇔　·本管理
⇔　·表示
⇒　·表示
⇔　·表示
⇒　·印刷（必要なら）

b. 職員が操作する場合

·ユーザ ID 確認 ～ ·職員の操作 は上と同じ
↓
利用者が借りている本のリストを選択された
↓
·利用者が借りている本のリスト
⇔　·ユーザ ID 確認
⇔　·ユーザ管理
⇒　·表示
⇔　·バーコードリーダ
⇔　·ユーザ管理
⇔　·本管理
⇒　·表示
⇒　·印刷（必要なら）

（5）特定の本を最後に借りた人の検索。

·ユーザ ID 確認 ～ ·職員の操作 は上と同じ
↓
特定の作者の本リストを選択された
↓
·特定の作者の本リスト
⇒　·表示
⇔　·キーボード
⇔　·本管理
⇒　·ユーザ管理
⇒　·表示

2. 各モジュール説明

以下に個々のモジュールの説明を記載する。

ユーザ ID 確認

《モジュール概要》 入力されたＩＤにより職員か利用者かを区別し、それぞれの操作の選択をよぶ。

《機能》 1。 ユーザのログインＩＤを受け付けて、そのＩＤを表示し、ユーザ管理にユーザを区別させる。

受付メッセージ	処理する
関連するクラス名（メッセージ）	表示（表示する）

2。 現在使用されているユーザＩＤを返す。

受付メッセージ	しゃべる
戻り値	現在使用している人のユーザＩＤ

《データ構造》 現在使用している人のユーザＩＤ

ユーザ管理

《モジュール概要》 ユーザのデータを操作するクラス。ユーザとは職員及び利用者である。主な機能としては、ユーザのデータを教えたり、貸出、返却時の本の変更を行なう。

《機能》 1。 ユーザのデータを教える

受付メッセージ	しゃべる
引数	ユーザＩＤ
戻り値	該当するユーザのデータ

2。 本の貸出

受付メッセージ	借りる
引数	ユーザＩＤ、本のＩＤ

3。 本の返却

受付メッセージ	返す
引数	ユーザＩＤ、本のＩＤ

《データ構造》
- ユーザのデータ
- ユーザの名前
- ユーザＩＤ
- 職員か利用者か
- 借りている本の冊数
- 借りている本のＩＤ

職員の操作

《モジュール概要》 操作選択の表示後、処理の指示を受け付け、指定された処理を行なう。

《機能》

受付メッセージ	処理する
関連するクラス名（メッセージ）	表示（表示する）
	キーボード（入力する）
	登録（処理する）
	削除（処理する）
	貸出（処理する）
	返却（処理する）
	利用者が借りている本のリスト（処理する）
	最後に借りた人の検索（処理する）
	特定の作者の本リスト（処理する）
	特定の分野の本リスト（処理する）

一般利用者の操作

《モジュール概要》 一般利用者に対して自分の借りている本のリストを作成するか、終了するかを選択させる。

《機能》 1。 操作選択

受付メッセージ	処理する
関連するクラス名（メッセージ）	表示（表示する）
	キーボード（入力する）
	利用者が借りている本のリスト（処理する）

登録

《モジュール概要》 図書館への本の登録を行なう。登録用の操作画面を表示し、本の名前、作者名、分野名をキーボードより入力する。本管理、作者管理、分野管理に登録し、バーコードライタにバーコードを発行させる。

《機能》

受付メッセージ	処理する
関連するクラス名（メッセージ）	表示（表示する）
	キーボード（入力する）
	本管理（登録する）
	作者管理（登録する）
	分野管理（登録する）
	バーコードライタ（書く）

削除

《モジュール概要》 図書館への本の削除を行なう。本のＩＤをバーコードリーダから読み込ませ、本管理、作者管理、分野管理に削除処理を行なわせる。

《機能》

受付メッセージ	処理する
関連するクラス名（メッセージ）	表示（表示する）
	バーコードリーダ（読む）
	本管理（削除する）
	作者管理（削除する）
	分野管理（削除する）

貸出

《モジュール概要》 本の貸出の一連の処理を行なう。本の貸出の操作画面を表示し、本を借りる人のＩＤを入手する。３冊以上借りていないかを調べて、借りることが出来るならば、そのユーザのデータに本のＩＤを記録し、本のデータを貸出中にし、借りる人のＩＤを記録する。貸出不能ならば、そのことを表示する。

《機能》 1。 本の貸出処理

受付メッセージ	処理する
関連するクラス名（メッセージ）	表示（表示する）
	バーコードリーダ（読む）
	ユーザ管理（しゃべる，借りる）
	本管理（貸出する）

返却

《モジュール概要》 図書館への本の返却を行なう。返却の操作画面を表示し、本のＩＤをバーコードリーダから読み込み、本管理、ユーザ管理に返却処理を行なわせる。

《機能》

受付メッセージ	処理する
関連するクラス名（メッセージ）	表示（表示する）
	バーコードリーダ（読む）
	本管理（返却する）
	ユーザ管理（返す）

特定の作者の本リスト

《モジュール概要》 指定した作者の本をリストアップし、画面に出力する（プリンタ出力もサポート）。

《機能》

内容	キーボードから、作者の名前を入手し、作者管理ＤＢから、その作者の本（複数）ＩＤを入手し、これを、本管理によって名前に返却し表示（プリンタ）する。
受付メッセージ	処理する
関連するクラス名（メッセージ）	表示（表示する）
	キーボード（入力する）
	作者管理（しゃべる）
	本管理（しゃべる）
	印刷（印刷する）

特定の分野の本のリスト

《モジュール概要》 分野別の本のリストを作成、表示する。リストはプリントアウトすることができる。

《機能》 1。 分野別リスト作成

内容	操作画面を表示し、分野名の入力待ち状態に入る。分野が入力されたら、その分野の本のＩＤを得て、ＩＤから本の名前を得る。本の名前を画面に表示し、印刷するかどうかを聞く、印刷を選択されたらリストを印刷する。
受付メッセージ	処理する
関連するクラス名（メッセージ）	表示（表示する）
	キーボード（入力する）
	分野管理（しゃべる）
	本管理（しゃべる）
	いんさつ（印刷する）

利用者が借りている本のリスト

《モジュール概要》 利用者のＩＤより、その人の借りている本があればそのリストを表示する。

《機能》

受付メッセージ	作成する
引数	ユーザＩＤ
関連するクラス名（メッセージ）	ユーザＩＤの確認（しゃべる）
	ユーザ管理（しゃべる）
	表示（表示する）
	バーコードリーダ（読む）
	本管理（しゃべる）
	印刷（印刷する）

本管理

《モジュール概要》 本のデータベースに対してアクセスできる唯一のモジュール。本の登録、削除、ＩＤの設定、返却状態を管理する。

《機能》

1。 指定された本のデータを返す

受付メッセージ	しゃべる
引数	ＩＤ または 名前
戻り値	指定された本の全データ

2。 指定された本を貸出状態にし、返却予定日を設定する

受付メッセージ	貸出する
引数	本のＩＤと借りる人のＩＤ

3。 指定された本を返却状態にする

受付メッセージ	返却する
引数	本のＩＤ

4。 新しい本をデータベースに登録

内容	新しい本のデータベースを作成し、本の名前の登録、貸出可能の設定をしＩＤを返す。
受付メッセージ	登録する
引数	本の名前
戻り値	本のＩＤ

5。 データベースから指定された本のデータを削除する

受付メッセージ	削除する
引数	本のＩＤ

《データ構造》
本の名前
本のＩＤ
貸出可能状態
最後に借りた人のＩＤ
(現在借りている人)
返却予定日

作者管理

《モジュール概要》 作者ごとの本のデータを管理する。作者のデータを教えたり、本の登録、削除を行なう。

《機能》

1。 新しい本の登録

受付メッセージ	登録する
引数	本の名前、作者名

2。 本の削除

受付メッセージ	削除する
引数	本の名前、作者名

3。 作者のデータを教える

受付メッセージ	しゃべる
引数	作者名
戻り値	

《データ構造》
作者データ
作者名
その作者の本のＩＤ（本の数だけ）

分野管理

《モジュール概要》 分野ごとの本のデータを管理する。分野のデータを教えたり、本の登録、削除を行なう。

《機能》
1。 新しい本の登録
受付メッセージ 登録する
引数 本の名前、分野名
2。 本の削除
受付メッセージ 削除する
引数 本の名前、分野名
3。 分野のデータを教える
受付メッセージ しゃべる
引数 分野名
戻り値

《データ構造》
分野データ
分野名
その分野の本のＩＤ（本の数だけ）

表示

《モジュール概要》 受けとった表示データを画面に出力する。

《機能》
1。 引数で受けとったデータを画面へ表示
受付メッセージ 表示する
引数 表示データ

キーボード

《モジュール概要》 文字列の入力待ちを行ない、入力されたデータを返す。

《機能》
受付メッセージ 入力する
戻り値 入力された文字

印刷

《モジュール概要》 受けとったデータをプリンタポートに出力する。

《機能》
受付メッセージ 印刷する
引数 出力するデータ

バーコードライタ

《モジュール概要》 引数で受けとったＩＤをバーコードに変換し、出力する。

《機能》
受付メッセージ 書く
引数 ＩＤ

バーコードリーダ

《モジュール概要》 読み込んだバーコードをＩＤに変換し、返す。

《機能》
受付メッセージ 読む
戻り値 ＩＤ

最後に借りた人の検索

《モジュール概要》　指示された本を最後に借りた人の名前を表示する。
《機能》

受付メッセージ	処理する
関連するクラス名（メッセージ）	表示（表示する）
	キーボード（入力する）
	本管理（しゃべる）
	ユーザ管理（しゃべる）

R-5･全体評価（井上忠則）

1. 試用した手法の問題点は何か？

ＣＲＣカードを用いた設計手法を実際に試用したときの問題点を、ブレーンストーミングで挙げていった。

･グループ内でシナリオをにつめるのが不十分であった。
･分析にもっと時間を割くべきだった。(時間をかけることの大事さが理解できていなかった)
･オブジェクトの抽出が正しくなされているか（適当であるか）自信がもてない。
･グループ内での意見の統一がなかなかとれなかった。(時間がかかった)
･グループで作業を行った場合、うるさい。(特にグループ３)
･これがオブジェクト指向なのか？　ぜんぜんわからない。今やっているやり方とかわらない。(情報を隠ぺいするのは当たり前、今さら何を言っているのか！)

ＣＲＣカードの問題点

･オブジェクトが１つ１つ美しく分割されていても、現実問題として作成するとき全体を知っていて１部分を作ることが必要である。
･ＣＲＣカードには書き込めない情報がある。
･ＣＲＣカードと矢印で関連づけすることを行えば、もっと有効なのではないか。

ＣＲＣカードの利点

･変更が簡単に行える。
･クラスを決めた後、責任、協力者をうめていくことで、どんどん仕様を洗い出すことができた。
･仕様変更に対しても柔軟。

２．各自の現場への適用は可能か？、問題点と代替手段はあるのか？

･無理なんちゃう？(現在、担当している範囲では、すでに細かくモジュール分けされ指示されるため、応用する場がない)
･今、現在すでに似た仕事のやり方を行っている。
･現在の担当者は、理解できないであろう。
　反論　担当者が理解できないのではなく、担当者を理解させる自分自身の努力が足りない。
･適用できるのではないか。

R-6·グループ活動の感想

1. 酒井雅裕

このレポートをかいているのは、９月９日の段階である。(ああっ。締切をすぎている) すっかり名古屋は秋の風が吹いている。９月５日は夏の終わりであったのだ。

(1) 何を失ったか

参加するまで、分析·設計に対して何を思っていたか書き連ねてみれば、何が分かったのか、明らかになるであろう。

·ドキュメントを作成するのに、時間をかけたくない。(どうせ、ドキュメントとソースは乖離するのだから)

·ひとりでやるほうが、早く終わる。

·分析設計で時間がかかるのは承服できない。(ソフトの規模が大きくないので分析しなくてもできちゃう。ソフトの規模が大きいときは、とりあえず早くコーディングしないと時間がないと思う)

·一生懸命やったって、仕様変更で手戻りは起こるのだ。

(2) 何を得たか

いろいろ挙げてみると分かることがひとつある。「仕事量を増やしたくない」理由で、分析や設計から逃避しているようだ。しかし本当に分析や設計をしないことで「仕事量」は減るのだろうか。「目先の」仕事量は確かに減るが、それだけが仕事量のすべてではないことは痛い目を見てようやく分かってきた。

ユーザは「ブラウン管」を見て、「タイプライターみたいな奴」をさわってはじめて評価をする傾向にあるようだ。(当社の客だけか？) ペーパーでレビューして、承認を得て作成してみて、使い始めてはじめて「つかえない」といわれることは、往々にしてある。

しかしそのことは、ユーザのわがままもあることはあるが、使いやすいシステムを提供するために、何をユーザから聞き取ればよいのかの検証が足らないために起こってしまうのではないか。

事前に分析をして検討すれば、わかってしまうことも多いのではないか。

(3) 何が壁か

ユーザとのコミニュケーションを活発にするために、「分析·設計」が重要な役割を果たすことは分かった。(そういう役割を果たさせてしまえばよいのだということも分かった) 今の私の仕事のあり方を改善をしなくてはいけないだろうとも思う。だが、漠とした不安も少しある。

今回、ＣＲＣカードによるオブジェクト指向分析設計を学んだがその後何人かのソフトウエア技術者と話してみて、より鮮明になったことがいくつかある。

·「オブジェクト指向」の理解が技術者各人で異なる (ソフトウエア技術者は人のいうことは信じない)
ワークショップの最中にＧ３でも発生した。オブジェクト指向は何かという論議だけで、不毛に時は流れる。
昨日、ＣＲＣカードの説明を機会があって、行おうとしたところオブジェクト指向といっただけでインチキ扱いされるのは、参った。その人にいわせれば、オブジェクト指向は観念論で、技術ではない。ＡＩのように５年たったら言葉も聞かなくなる。ということでした。

·時間をかけて大丈夫か (投資効果は十分か)
はじめに時間をかけることが、本当に効果があがるか、心底から疑問に思っている技術者が結構いる。

·分析がかかる時間の割には不十分ではないか (終わりなき分析にならないか)
どんな分析法でもそうなるのではないかと思うが、仕様の漏れを気にかけているらしい。最終討論でもコメントがあったが、何も「信頼性」をないがしろにしようというわけではない。医療関係のプログラムや原発プラントのプログラムと「図書館問題」では要求される質が違うのではないかと思うのだ。

(4) 現場は動くか

ＣＲＣカードの利点を述べれば、上にあるいくつかの批判にコメントすることはできる。

ＣＲＣカードはその手軽さこそが利点である。それは、型にはまった手法を要求せず、「ブレインストーミング」で分析ができることであった。そこには、オブジェクト指向の「あるべき論」は、前提にない。(正確にはまったくないわけではない。カプセル化位は気にとめる必要があるだろう) 各人の分かる範囲からスタートでき、時間がなければきりあげることができ、深い分析が必要になったなら、分析再開が可能なのだ。

オブジェクト指向のキーワードは知識として、ワークショップに参加する前でも知っていた。だが、現実の仕様をどのようにその知識と結びつければよいのかわからなかった。

しかしＣＲＣカードが簡単にその壁を越えてしまったのには、おどろいた。各人のオブジェクト指向への理解度も吸収し、ＣＲＣカードを中心にプロジェクトが固まっていくのは、非常に面白かった。時間を投資するに見合った、仕様への理解が得られることもよくわかった。

自分の立場はＣプログラマで、おそらくはＣ＋＋プログラマに移行していくのであろうと思われる。ＣＲＣカードの分析は今の立場でも未来の立場でも両方に通用することも大きな魅力であった。

「使える」「使えない」の論議はまた今後も続くことであろう。私個人としては、オブジェクト指向の魅力と、ＣＲＣカードの利点にぜひとも、すがってみたい。

まずは、小さいプロジェクトで実験を重ねていきたい。

2. 吉田恵

・＜類は友を呼ぶ＞ということわざが、実証された。(血液型判断より)
・初めてＣＲＣカードを使用し、設計作業を行なったが、メンバの中で、誰一人この手法を完全に知る人がいなくても、設計は可能だと思った。それと同時に、討論が行き詰まった時にうまく助け出してくれる役割の人は大切で、ＰＣの方に何度もお世話になった。
・メンバがタフだったので午前９時から午前２時まで３日間作業を続けることができた。
・年齢も、出身も、さまざまな環境も異なるメンバでも、目的を持つことにより、まとまりをもって進んでゆけることがわかった。

3. 渡辺清文

・全体討論がかみ合っていなかったのが残念だった。
・他のグループの技法については良く理解できない。全グループがＯＯＡ／ＯＯＤでやっても良かったのでは。同じ手法の方が全体討論が盛り上がると思う。
・オブジェクト指向が時期尚早ではないかという事務局の判断があったそうだが、そうは思わない。また若手の会なのだから多少早まったことをしてもいいのでは。
・どこの会社も設計やドキュメント整理といった点では多少なりともいい加減にやっていることが分かったので安心した (そんなことでいいのか？)。

4. 伊藤徹

皆さん同じだとは思いますが、私も初日に会場に入った時にはそれなりに "不安"と"緊張" のプレッシャーを感じていました。我々のグループは、初日の晩に PC の部屋に "各人の部屋の冷蔵庫からビールを持って集まれ" とのお達しが渡りました。言われたとおりに PC の部屋にビール持参で行ってみると、既でに他のメンバーは出来上がっており、そこで自分達が現在携わっている仕事上の不満、問題点等を酔った勢いで話し込んでいた訳です。

話しを聞いてみると "あー、皆同じ様な問題を抱えてるんだなー" と感じられるところが多く、昼間の不安はどこへやらすっかり打ち解け、次の日にはしっかり普段の自分に戻っていました(俺だけか?)。

さて、CRC カードの設計を始めてすぐに、クラスをどこで分けるかでグループ内の意見が大きく 2 つに分かれました。(これは G4 グループでも同じ壁にぶち当たっていた様ですが...) 幸いにして ("まーいーかー" ってのもあるけど) 我々のグループはそれなりに全員の意見の統一がされ、最終目的まで達成出来た訳ですが、"人それぞれ面白い事を考えているものだ" と思いました。今回の設計では、私が支持していたクラス分けで設計がされましたが、個人的な意見を言えば、これは普通に考えれば誰でも思い付くと言えるクラス分けだっ

たと思います。もう一つの意見(クラス分け)で設計していった場合の最終的な形がどんな物になったのか? 大変興味深いものが有ります。

色んな人の意見が聞ける、かといって多過ぎては、言いたい事も言いにくくなる。(最終日の全体会議の様にマイクを使わなくてはならない様だと発言しにくい)そういった意味でも4～5人のグループでの活動はよかったと思います。欲を言うと、他のグループとの結び付きが薄かったのでは? と思われます。(他のグループの設計手法があまりよく分からなかった)

初めてワークショップという物に参加した私にとっては、全くの見ず知らずの人達と、3日間という短い期間で1つの物を作り上げるという作業は、予想以上に楽しい物でした。もちろん、結構自由な日程になっていた事、よい環境であった事、などSEA及びPCの方々に感謝する部分も大きいのですが何と言っても、グループメイトに恵まれた点が大きいと思います。設計するのも、プログラムするのも、所詮は人間です。気持ちのいい人達と気持ちのいい仕事をしていきたいものです。

5. 井上忠則

グループでの利点

·グループのメンバー全員が設計過程を体験でき、同一の理解がとれる。
·総合的に見ると、他の人(設計者以外)に説明する時間を省くことができ作成時間が短縮できる。
·メンバーのキャラクタによって、でき不できが左右されるのではないか。

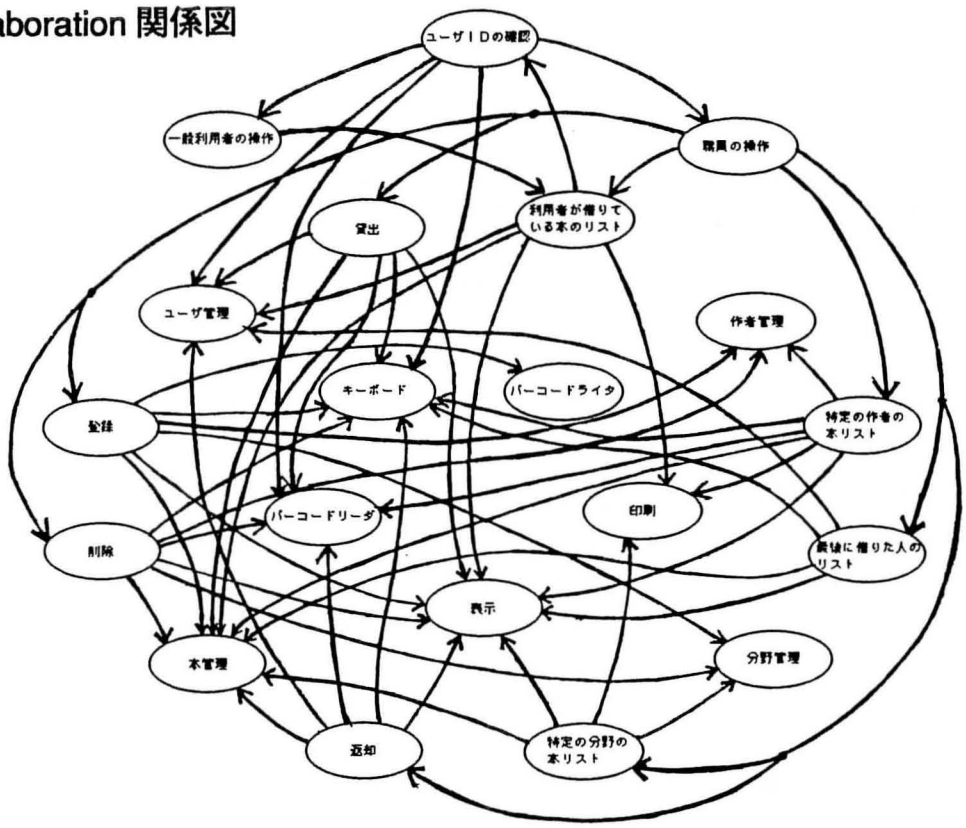
グループでの問題点

·自分の考える方向に必ずしも進まない。
·思い通りにやれないため、不完全燃焼となる。

全体を通して

·今回のワークショップでもいろいろな人がいて、いろいろな考え方で、いろいろな説明のやり方があると実感しました。
·結構、お酒を飲んでしまった。

Collaboration 関係図



ユーザＩＤの確認

Class

1992若手の会

Responsibilities	Collaboration	Data
・ユーザーのログインを受け付ける	バーコード入力	現在のユーザのＩＤ
	キーボード	
・職員か利用者かを判断する（ＩＤ）	ユーザー管理	
1）職員	職員の操作	
2）利用者	利用者の操作	
3）どちらでもない		
・現在使用されているＵＩＤを返す		

利用者の操作

Class

1992若手の会

Responsibilities	Collaboration	Data
・操作メニューを表示する		
・操作メニュー処理を受け付ける		
・ユーザーが借りている本を尋ねる	借りている本の	
	リスト作成	
・終了する		

職員の操作

Class

1992若手の会

Responsibilities	Collaboration	Data
・操作選択メニューの表示	キーボード	
・操作選択を受け付ける		
・各々の処理を呼び出す	登録・削除・貸出・返却	
	借りている本のリスト	
	最後に借りた人の検索	
	特定の作者の本のリスト	
	特定の分野の本のリスト	
・終了する		

登録 Class	1992若手の会	
・登録用操作画面を表示する		
・本に関するデータ（書名、作者等）	キーボード	
を尋ねて受け取る		
・本のデータ及び"登録"を渡す	本管理	
・本のＩＤを受け取る		
・作者名、本のＩＤ及び"登録"を渡す	作者管理	
・分野名、本のＩＤ及び"登録"を渡す	分野管理	
・バーコードライタに本のＩＤを渡す	表示・バーコードライタ	
・終了する		
Responsibilities	Collaboration	Data

削除 Class	1992若手の会	
・削除用操作画面を表示する	表示	
・削除する本に関するデータを受け取る	バーコードリーダ・キーボード	
・作者管理ＤＢから受け取ったＩＤの本を	作者管理	
削除する		
・本管理ＤＢから受け取ったＩＤの本を	本管理	
削除する		
・分野管理ＤＢから受け取ったＩＤの本を	分野管理	
削除する		
・終了する		
Responsibilities	Collaboration	Data

貸出 Class	1992若手の会	
・貸出用操作画面を表示する	表示	
・貸出す本に関するデータを受け取る	バーコードリーダ・キーボード	
・貸出者のＩＤを受け取り表示する	バーコードリーダ・キーボード	
・貸出者のＩＤを渡し、３冊以上借りて	ユーザ管理	
いないかをチェックする		
・チェック後エラーは表示する	表示	
・本のデータ及び"貸出"を渡す	本管理	
・本のＩＤ及び"借りる"を渡す	ユーザ管理	
・終了する		
Responsibilities	Collaboration	Data

返却	1992若手の会	
Class		
・返却用操作画面を表示する	表示	
・返却する本に関するデータを受け取る	バーコードリーダ・キーボード	
・本のＩＤ及び"返却"を渡す	本管理	
・返却者のＩＤを受け取り本のＩＤと	ユーザ管理	
返却者のＩＤ及び"返却"を渡す		
・終了する		
Responsibilities	Collaboration	Data

特定の作者の本リスト	1992若手の会	
Class		
・リスト作成用操作画面を表示する	表示	
・作者名のデータを受け取る	バーコードリーダ・キーボード	
・作者名及び"しゃべる"を渡す	作者管理	
・本のＩＤを受け取る		
・本のＩＤ及び"しゃべる"を渡す	本管理	
・本の名前を受け取り表示する	表示	
・印刷する場合	印刷	
・終了する		
Responsibilities	Collaboration	Data

特定の分野の本リスト	1992若手の会	
Class		
・リスト作成用操作画面を表示する	表示	
・分野名のデータを受け取る(名前、10進表記)	キーボード	
・分野名及び"しゃべる"を渡す	分野管理	
・本のＩＤを受け取る		
・本のＩＤ及び"しゃべる"を渡す	本管理	
・本の名前を受け取り表示する	表示	
・印刷する場合	印刷	
・終了する		
Responsibilities	Collaboration	Data

利用者が借りている本のリスト Class	1992若手の会	
・現在のユーザのＩＤを受け取るために	ユーザＩＤの確認	
"しゃべる"を渡す		
・ユーザのＩＤを受け取る		
・ユーザＩＤ及び"しゃべる"を渡す	ユーザ管理	
・借りている本用操作画面を表示する	表示	
・ユーザＩＤ及び"リスト作成"を渡す	ユーザ管理	
・本のＩＤから名前を受け取る		
・本のＩＤ及び"しゃべる"を渡す	本管理	
・受け取った本の名前を表示する	表示	
Responsibilities	Collaboration	Data

Class	1992若手の会	
・印刷する場合	印刷	
・終了する		
Responsibilities	Collaboration	Data

最後に借りた人の検索 Class	1992若手の会	
・検索用操作画面を表示する		
・本に関するデータを受け取る	キーボード	
・書名及び"しゃべる"を渡す	本管理	
・最後に借りた人のＩＤを及び"しゃべる"	ユーザ管理	
を渡し、ＩＤを名前に変更する		
・最後に借りた人の名前を受け取り	表示	
表示する		
・終了する		
Responsibilities	Collaboration	Data

ユーザ管理

Class

1992若手の会

Responsibilities	Collaboration	Data
・渡されたＩＤからユーザのデータを返す（名前、住所、電話等）		自分のＩＤ 自分の名前、その他
・ユーザが借りている書名を返す		職員、利用者の区別
・借りた本を登録する		借りている本のＩＤ
・返却した本を削除する		借りている本の名前
・貸出時に本のＩＤを登録する		借りている本の冊数
・職員か、利用者かを返す		

本管理

Class

1992若手の会

Responsibilities	Collaboration	Data
・書名、利用者ＩＤ等のデータを返す		本の名前
・本の状態を返す（登録済、貸出中等）		本のＩＤ、その他
・本のＩＤを決める		貸出可能状態か
・本のＩＤを返す		
・本に関するデータを受け取る		返却予定日

作者管理

Class

1992若手の会

Responsibilities	Collaboration	Data
・作者に関するデータを返す		作者のデータ
・作者を登録する		本のＩＤ
・作者を削除する		
・作者を検索する		

分野管理 Class	1992若手の会	
・分野に関するデータを返す		分野のデータ
・分野を登録する		
・分野を削除する		
・分野を検索する		
Responsibilities	Collaboration	Data

キーボード Class	1992若手の会	
・入力待ちし、入力データを返す		
Responsibilities	Collaboration	Data

バーコードリーダ Class	1992若手の会	
・本又は人の情報を表示する		
・本のＩＤを返す		
Responsibilities	Collaboration	Data

表示 Class	1992若手の会	
・与えられたデータ（レイアウト、フォーマットを含む文字列）を表示する		
Responsibilities	Collaboration	Data

印刷 Class	1992若手の会	
・印刷する		
Responsibilities	Collaboration	Data

はじめてCRCカード手法でオブジェクト指向設計をやってみて

桜井 修

このグループは, テーマに「オブジョクト指向設計」を取り上げ, この具体例として, Kentの「CRCカード手法」で共通問題の図書館問題を解いて見た。ここではPCとしてメンバーのそばで感じた点やCRCカードの問題点など列記してみようと思う。

1 参加メンバーについて

メンバーは若手5名であるが, 「設計」について十分確信をもって参加してくれたメンバーは, いないようであった。むしろ, それぞれ現場でのそれなりの対応力がついたメンバーが集まった。若手のワークショップであるからその方が良く, 議論を持ち込める力量のあるメンバーであることは確かであった。そのことは, メンバーが集まって1時間もしないうちに仕事の話やら, 最近の業界のトレンド, いま自分が悩んでいる技術的な問題点などが話し合われたことからも分かる。

そこで聞かれる「現場での設計」の雰囲気は,

- 割合いい加減な進め方でも何とかなる。
- システム設計など, 大げさに掲げるほどの作業を行っていない。
- いや, 仕様に追われて設計なんて気にかけていない。
- さらに, 仕様自身もうまく表現されていないため, 変更のためのプログラミングを行っているようなものだ。仕様通りを鵜呑みにできない。
- 分かってはいるけど, そんなことより動くものをつくらないと。

とかなりいい加減なものである。

彼らが上げる技巧的なものは, 「フローチャート」「データフロー」「構造化プログラミング」などで, 開発環境もそれほど大規模であるメンバーもいなく, 多くて5,6人程度のプロジョクトのようであった。

これだけでは, 彼らがなぜこのワークショップに参加したのか理解できないが, これは彼らが「設計」をやっていないというわけではなく, さらに軽視しているわけでもない。だた現場サイドの率直な雰囲気で, 人に言えるほどに体系的なものになっていないからであるようだ。

そう, あらためて自分の「設計」作業を省みるとそう答えることは, 私自身も同じような現場であるので理解できる。彼らは彼らなりの「設計」をもって参加してきていると確信している。

2 はじめてオブジョクト指向設計を試みると

その彼らにしてもオブジョクト指向のアプローチで設計してみることははじめてのようであった。オブジョクト指向の発想は, メンバーに違和感なく理解されていたようだ†1が, 具体的な設計作業になると戸惑いやこだわりがはじめのグループ作業のときに見受けられた。

今回は, CRCカードなるものを使って, それらのカードを定義して, 関係をもつことでシステムを設計していくアプローチをとったが, 特にその取り掛かりで彼らが悩んだ点を上げてみることにする。

1. **どこから設計しはじめたら良いか分からない**
   はじめの取り掛かりのクラスがどんなものか, 見つけられないため, 設計がすぐにスタートできない。
   これは, CRCカードのアプローチの仕方でもあるが, 分析の初期段階でシステムのシナリオを作成しておかなければならない。そのシナリオからスタートのクラスを見つけることが出来る。図書館問題においても, シナリオのはじめのクラスがユーザー待ちのための表示クラスであることが分かる。この点構造化手法であるとトッププロセスとして「図書館システムがそれぞれの処理を行う。」と安易に確定できるが, オブジョクト指向だとそれを善しとしていないため, 設計がすぐにスタートできないように見られがちである。メンバーもその点, 他グループの作業がどんどん進んでいるように錯覚してしまい中間報告の時点で焦っている様子であったが, 一向にかまう必要はない。
2. **なにが問題の中でのクラスなのか, 悩んでしまう**
   今回のように, 既存の再利用可能なクラスがない場合は, 自ら作り出して行かなければならない。データ, 装置, 表示など, ある目的の振る舞いのあるものをクラスとして定義してみて, システムにあわなければどしどし捨てる覚悟でつくっていけば良い。その間, CRCカードが汚くなっても一向にかまわない。図書館問題にお

池上通信機(株)

†1 これは本当は嘘かもしれない。オブジョクト指向とはどんなものかの議論を延々やっていたのは, うちのグループだった。PC側から議論できるたけの材料を提示できなかったからかな。ごめんなさい。討論の内容も発散傾向にありましたね。

いても, その登録者カードのIDをチェックするためのカードリーダーが必要と思えば, そのクラスを作成して, カードリーダーについてサービスできることを列記したCRCカードをすぐ作成すればよい。それがシステムの中でどう使われるかは, 設計が終了した時点で明らかになればよいのである。ちなみに, グループ作業においても, このカードリーダークラスが分析/設計段階で一度消えたが, 再び採用された経緯もあったので, 不要とおもわれるクラスでもゴミ箱に捨てないで横に外して置いたほうが良い。

なにがクラスかについては, 既存のよく設計されているものを参考にすれば理解できるだろう。我々現場のエンジニアは理屈や体裁を考えるより, まずやってみることだ。

3. **階層的なモジュール構造を連想してしまい, トップのクラスを考えようとする**

これはよくあることで, トップダウン設計のイメージをオブジョクト指向設計に持ち込んでしまっている。全体がぼんやりしている中で, 大きなクラスを決めようというのは良いアプローチとは言えない。はじめのころ便宜的にうまく行き, 全体を見通せたように錯覚させるには, 良いかもしれないが, その後のクラスを分割していく上で混乱を生じる。主要なデータ抽象クラスを作り出すことが先決であり, 初期段階で全体的にまとまっている何かを見つけるものではない。全体的にまとまるのは, クラスを見つけ出したり, もって来たりしてその関係を見つけたあとに現れるものである。

しかし, はじめて試みるときには, クラスを定義しても設計段階でそのクラスが2つ以上に分割されることは, しばしばある。はじめから, 確定されたクラスを見つけようとしてもいけないようである。

以上の点は, もしかすると「オブジョクト指向設計」を始めるときに誰でもかかることのような気がする。第3グループのメンバーも以上の点でみんなのコンセンサスが取れなくて作業が息詰まったようだ。しかし, これらの点をクリアすると, 結構作業はスムーズに進むようになる。メンバーもちょっと疑問[†2]はあるにしても, まずはやって見ようということでCRCカードに思いついたクラスを書き込んでいたことは, 印象的であった。

3 CRCカードの評価

CRCカードを用いてグループで設計を行った経験は, 始めての試みでもあった。その点, 若手のメンバーで作業をした中で, いろいろな点に気づかせてくれ, あらためてCRCカードを評価できることになった。グループで作業を行ってCRCカードについて気の付いた点を上げみることにする。

1. **カードは, クラスを複雑にしない程度の大きさであるが, はじめからその大きさにこだわり過ぎないほうが良い**

たいてい, クラスを見つける作業のはじめは, 1つのクラスに役割がどんどん詰め込み過ぎる傾向になる。第3グループのはじめのころのCRCカードを参照してみると, 小さなカードに役割がいっぱいかかれて見る気もおこらない内容である。しかし, あまり大きさにこだわり過ぎるとはじめから内容を書き込むことに躊躇してしまう。今回グループ作業で使用したカードは, Buddのテキストに出ていた大きさとほぼ同じにしてみたが, 分析の初期段階でのぼんやりしているときは, もっとラフな大きさのメモに書いてみても良いかもしれない。

CRCカード手法は, カードさえ作ってしまえばどこでも簡単に行えるが, 無理にカードにこだわる必要はない。クラス (C) 責任 (R) 協力 (C) の要素がまとまっていて, 分割, 移動, 参照, 修正が容易で, その大きさも書き込める内容の量がそれほど多くならないように規制できるものであればよい。クラスはもともと対象に対して単純なものでなくては, 抽象化されたことにならなく, カードの大きさがそれを規定しているからである。

2. **カードの並べ方は, 重要である**

シミュレーションする上でカードは, 具体的にプロセスを表現できるが, カード間の関連を示すように並べるだけのスペースがいる。Buddのテキストでは, カードそれ自体しか表現されていない。

その協力関係を表す方法は, テーブルの上にカードを並べることだ。しかし, その置く位置は, 確認のためにプロセスをシミュレーションするとき, カードからカードへ目を移動しやすいようにする。この並べ方が良くないとカードを探している間に依頼された内容も忘れてしまうことがある。テーブルに自由に線を描け, さらに消せるようなものが必要と感じた。その点, このCRCカードそれ自体は, 設計の最終段階においても全体を見通せるほど, まとまりの良いものにはならない。カードの束が残るだけだ。

3. **クラスは, どんどん定義していった方がよい**

今回のように, はじめから再利用可能なクラスがないようなときは, 問題領域にありそうなクラスをどんどん先に定義していったほうが, 設計の助けになる。さらにシナリオに添ってCRCカードを定義していくと, 勢い

†2 いやかなりあったようだ。一番声高く議論していたのはうちグループだった。うるさかったなあ。

カードの内容が多くなり，複雑になりやすい傾向にあるらしい。これもCRCカードが1枚に収まらず2枚3枚と重ねられたらすぐに分割してクラスを定義していった方があとの作業がスムーズに進む。最終段階でクラスを統合させるのはたやすいことだが，複雑になっているCRCカードから分割するのは難しい。

4. **記述方法に一定の取り決めが必要**

役割の記述方法やクラス名の決め方について，グループ作業する上で統一的なものを決めておく必要がある。一人で設計するときは，それほど気にかからない問題であるが，グループでやるときは，混乱を生じる。第3グループでもサービスを依頼する方と依頼される方との表現の違うを無くすため，「話す」「しゃべる」などの言葉の統一をおこなっていた。

5. **再利用性については，汎用的クラスが見つかりにくい**

問題領域を対象にシステムの動作シナリオに添って分析／設計作業を行っていくため，既存のクラスがあればそれを使用できるが，なければ作らなければならない。その時は，どうしても問題領域に近い概念で抽象化されたクラスをつくってしまうため，他のシステムで再利用できるかどうかの評価はできない。

さらに，クラスの継承関係も表現出来ない。繰り返えされる機能や，2つ以上のクラスで重なっている部分が生じたとき，共通のクラスを作成することでしか表現できない。

6. **シナリオの変更に対して，柔軟性がある**

CRCカードは，それがどこからのサービスに応答するか規定していないため，どのシナリオの部分からでも利用できる。そのため，動作順番に変更があっても，変更のあったカードに着目し，関連カードの並び替えを行えば良かった。カードリーダーでのＩＤチェックは，シナリオのどの部分でも利用できる。また，ある操作の終了時点でも，どのクラスへもサービスを依頼できるので操作シーケンスの変更が簡単であった。

7. **カードの関連図を残すことができない**

分析／設計作業中CRCカードは，移動しやすさや扱いやすさで利点を発揮するが，最終結果でその関連図を残すことができず，人に説明できない。今回も最終報告会で模造紙にCRCカードを貼って見せたが，すべて表現できないため，関連カードごとのまとめてしまい誤解をまねくような状態だった。さらに，最終結果がただCRCカードの束が残るだけになるため，見栄えも劣る。なにかに整理された記述法に書き換える必要がある。

以上の点は，ある部分でCRCカードの欠点を気づかせてくれる。それらをどうカバーしていくかは，実際の作業現場へ投げ込んでしまっても良いと思う。あとは現場対応でと言うわけである。それほど，この手法は柔軟性のある，型にはまっていない，どこでも応用の効くものだと確信できた。これを元にして，コンピュータ上にカードを書けるシステムを独自に開発してもよいし，現場にあったカードフォームを作成しても良い。Buddのテキストだと CRCカードの裏面に格納データを書き込むことを指示しているが，裏がえすのが面倒と感じたら，カードを大きくして新たにデータセクションを設けてもよいのである。

4 今後

参加してくれたメンバーのそれぞれのポジションペーパーから，今回のワークショップに期待するものが載っている。

- 生産性向上のための「設計」を吸収したい。
- グループとしての「設計」を考えたい。
- 仕様書にだまされながら変更の日々から抜けたい。
- 「将来」の設計のためにアンテナを張っておきたい。

などである。現場にもどって，その後のことを聞いていないので分からないが，どのように変化しただろうか。なにを見つけたであろうか。どうしても，現場へ戻ってしまうと流されて目的を達成しようとしてもいろいろな障害がつきまとう。しかしそれを解決していかないとワークショップに参加して持ち帰ったものが生かされないのである。このワークショップを機会にオブジョクト指向やCRCカードについて理解や疑問が深まることを期待したい。

ちなみに，Buddのテキストが翻訳されるらしい。しかし，このグループがCRCカードを実践した数少ない日本人であることは事実であり，いろいろな問題点を気づかせてもらったことは，私にとってもよい体験であったと思う。第3グループのみなさん，ありがとうございました。

グループ活動の感想

豊田 英利

我々のチーム取り入れた設計方法論は, グループ 4 と同じ" オブジェクト指向分析/設計" の 1 手法である CRC カードを採用し, 課題のシステム設計をした。

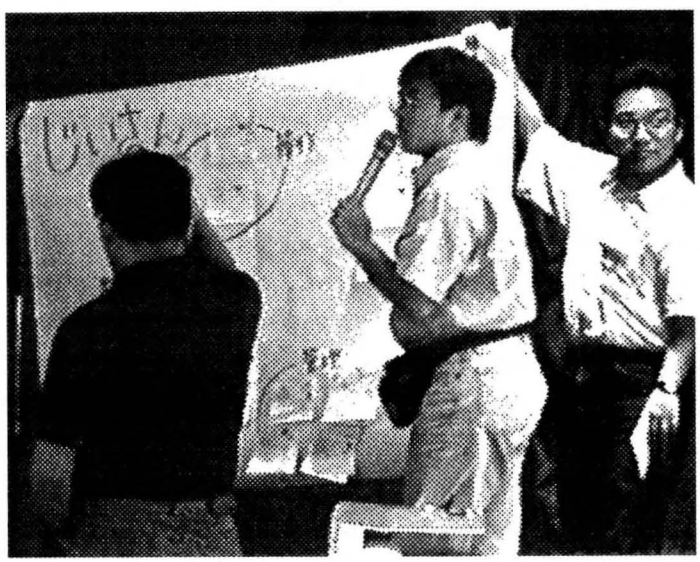
メンバー全員がオブジェクト指向は初めてであり, 2 日目の設計方法論のレクチャ後開始した。課題に関する仕様のツメを約 30 分し, とにかくやってみようと言うことで, 全員が CRC カードの作成を開始しだした。

私は, このアプローチを見て, どんなスケジュールでどこまで設計するのか予測がつかず少し戸惑った。作業が進むにつれて, 各個人が従来の設計手法でシステムのイメージを作り, それを CRC カードに記入している様に感じていた。そこで起きたことは, 処理関連の思考で, オブジェクト指向の独立した責任の委託によるクラスが見付け出せなくなり, 何をクラスにするかの議論が再三行われた。

結局, システム利用者側の立場にたってどこにどんな責任を持たせるかを CRC カードにどんどん記入していった。記入して整理してみると, やってみようから始まり, 最後には CRC カードが本来目的とする設計書が出来上がった。CRC カードを利用して設計した場合, 他の設計作業のどの部分に当たるかと考えた場合, 状態遷移図とデータフローダイアグラムぐらいまでは出来上がるのではないだろうか。CRC カードを採用してみて, 大勢の人で設計作業ができ, ユーザーを設計作業に参加させやすい方法である。また, オブジェクト指向を学習するのに最適であった。

今回, 現場で色々な経験をつんだ人が集まってグループを作り, 与えられた設計方法論で設計作業をし, 設計書をまとめた。今までの手法と違った視点から設計を考えることが出来た筈である。

今後, ここで気づいた点, 又小さな発見を現在の設計作業に生かしていって下さい。



株) セス

Group-4　CRC 設計最終報告書

R-1・グループの活動概要

1. メンバの簡単な紹介と役割分担

No. 氏名	会社名	役割	その他
001 古閑 幸一	ヒラタソフトウェアテクノロジー	R-1	少しだけ オブジェクト指向経験あり
003 岡田 安彦	オリンパス光学工業	R-3	設計に手法を用いた経験なし
010 森下 葉子	SRA	R-2(1-3)	OOP の経験はあるが、 OOA/OOD は勉強中
019 工藤 聡	岩手電子計算センター	R-4	OOP を趣味で少々
021 舟引 秀光	さくらケーシーエス	R-5	Hyper Text System の研究、開発 オブジェクト指向データベース の研究、開発 主な使用言語は、C,C++
023 門田 久好	新日鉄情報通信システム	R-2(4-7)	オブジェクト指向分析／設計未 経験

初日 (9/2) のグループ顔合わせ後，活動方針を

- コミュニケーション・ギャップをなくす。
- プロセスを明確にする。

と設定した。その為の方策として

- かくしごとをしない。
- 意見に対して必ず応答する。
- プロセスの記述を工夫する。

ことをあげた。

また目標は

- 設計結果をレビューするところまでいく

と設定し，大まかに以下のようなスケジュールとした。

- 9/3 午後要求分析
- 9/4 午前設計またはその見直し
- 午後設計またはその見直し

17:30 ～ 18:30 発表準備

以下に項目を区切って少しだけ詳細に説明する。

2.使用した手法名

CRC (Class, Responsibility, Collaboration)

3. 解いた課題 (特にグループ独自で拡張した部分に関する説明)

提示された課題の内容ではあいまいな点が見受けられたので，
顧客の立場で内容を決定し，あいまいな点，不明な点をなくしていった。
その際，実際には使いずらいと思われることでも，今回のシステムに限り
ということで決めたこともある。
(詳細は "R-2 2.分析で困ったこと" で述べる)

3.1 課題内容検討のやり方

- フリーディスカッション的に進めた。
- 入力が必要になるような部分では，それに必要なデータを決めた。

例) 貸出時
- 本のＩＤ
- 借りた人のＩＤ
- 貸出日
- 現貸出数
- 貸出期間 (本によって設定)

- モノをデータで表現してみた。

例) 図書館本とは？
- ＩＤ
- 題名
- 著者名
- 分野
- 出版社
- 版数
- 発行日
- 値段
- 貸出期間
- 登録日
- 現在の状況
  - 貸出日
  - 貸出人
  - 返却状態
- 削除日
- 削除した職員のＩＤ
- 削除理由

4. 作業プロセスの概要

9/3(木) 午後で今後の進め方等を決定した。
プロセス計画は次のようになる。

1.0 プロセスの進め方討議
2.0 分析作業
3.0 中間発表
3.5 分析見直し
4.0 設計作業
5.0 設計見直し作業
6.0 発表準備
7.0 発表
8.0 全体評価
9.0 グループ報告書作成

DFD-like に記述した。
(図 R1.1 プロセス計画 DFD 参照)

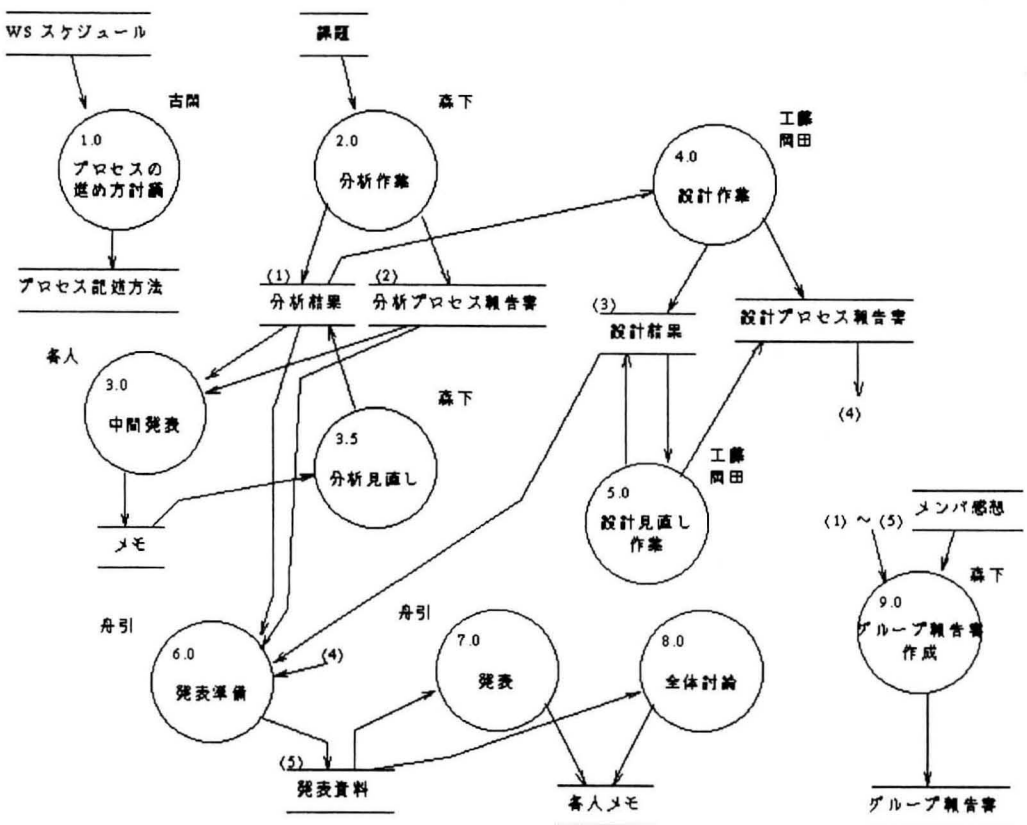


図 R1.1 プロセス計画 DFD

R-2・分析内容

1.着目点、アプローチの概要

-1 初期方針と実際の作業プロセスの概要
今回の図書館問題を読んで、各自が疑問と思った点がいくつかあった。疑問点／不明な点を抽出し、解決していくことで、機能要求を定義しなおし、我々のグループの図書館問題を新たにつくることを分析段階とした。
また、分析後の設計段階で問題が発生したら、分析の見直しというプロセスをとった。

（プロセス）
その１：各自で問題文を読む
その２：グループ内で問題文の不明点・疑問点の抽出をおこなう
その３：不明点、疑問点を解決する

-2 アプローチ（その１）とそこでの着目点
*** 各自で問題文を読む ***
10 分間、各自で問題を読み、疑問点・不明点を抽出することで、今回の問題に対して各自が抱いているあいまいなイメージを整理することをねらいとした。

-3 アプローチ（その２）とそこでの着目点
*** グループ内で問題文の不明点・疑問点の抽出をおこなう ***
アプローチ（その１）で各自がもっている疑問点・不明点をだしあって、認識しあうことをねらいとした。

-4 アプローチ（その３）とそこでの着目点
*** 不明点、疑問点を解決する***
本の登録／削除等、処理過程を順に定義していくことで不明点、疑問点を解決することをねらいとした。
また、グループ内で処理イメージの統一化をめざした。

2.分析で困ったこと（発生した問題点）

1.の作業プロセスのところで出したように、今回の問題文には不明な点がいくつかあり、図書館処理がイメージできなかった。
不明点としては次の項目が挙げられた。

[例] 不明点
→ 仮定した内容
（理由）

[1] 図書館には CD 等も借りれるところがあるが、問題文中の「本」とは何か

→紙に印刷したもので、次の属性をもつものと定義した。
ID
題名、著者名、分野、出版社、版数、発行日、値段
貸出期間、登録日、貸出日、借りた人、返却状態
本の削除日、本を削除した人のID番号、削除理由
(理由)
問題文中には「本」についての定義が"あいまい"で、問題文の処理をおこなうためには「本」の定義／属性を決めておく必要があった。

[2] 同じ題名の本は図書館でどのように管理されているか
→本の属性にID番号をつけることにより、同じ題名の本でも異なった本として扱えるように決めた。
(理由)
問題文には同じ題名の本の管理方法が書かれていないため、仮定する必要があった。

[3] 同姓同名者はどうするか
→「人」の属性にID番号をつけることで見分けることにした。
(理由)
「本」の管理方法と同様に、「人」の管理方法が記述されていない。
問題文の処理を行なうためには仮定する必要があった。

[4] 貸出期間はあるのか
→1冊の本の貸出期間を設けることにした。
(理由)
問題には記述されていないが、通常の図書館を考えると、貸し出される本には必ず期間が決まっている。
日数を決めなくても、貸出期間があることだけは仮定する必要があった。

[5] 本を借りている期間内に、追加して本を借りれるか
→数回に分けて借りても、借りている本の合計がxx冊以内で、貸し出し期間を過ぎていなければ借りられる。
(理由)
問題には1回に借りることのできる本の数が決まっていることは書かれていたが、分割して借りることができるかは書かれていなかった。
本の貸し出し時に、借りる人の資格チェックをするなら仮定する必要があった。

[6] 貸出期間があるとすれば、図書館の休日は貸出期間中に含めるのか
→貸出期間は、図書館の稼働日数で計算する。
(理由)
問題には記述されていないが、本を借りる時に既に借りている本が返却日を過ぎていないか、または、本を返却する時に返却日を過ぎ

ていないかをチェックするなら貸出期間の定義を仮定する必要があった。

[7] 貸し出し条件は何か
→ 本を貸出す時､[5],[6]のように、
・貸出し本数以内か
・既に借りている本の貸出し期間が過ぎていないか
をチェックしてから貸出しがおこなわれるが、今回のシステム内ではこのチェックは行なわない。
(理由)
問題文には書いていない処理だが、通常のシステムではチェックは必要である。だが、チェック処理を今回の問題に入れるとすると処理が複雑になりすぎるのでいれないことにした。

[8] 職員は利用者か
→ 職員も図書館の本は借りることができるので、利用者にもなる
(理由)
問題文の利用者の位置付けはあいまいである。図書館の職員の中には問題文のような処理が行なえない職員も考えられるが、今回は職員が本を借りる場合でも、一般の人が本を借りる場合と同じ処理をとると仮定した。

[9] ユーザインタフェースをどうするか
→ ログイン画面、職員が扱うメニュー画面を決めた

[初期メニュー]

User ID : Birthday:

[管理メニュー]

1. 本の貸出 2. 本の返却 3. 本の登録 4. 本の削除 5. リスト作成 6. 本を最後に借りた人の検索

(理由)
問題文には職員と一般利用者が使うマシンを分けるかどうかは記述されていなく、グループ内でも頭の中でもっているイメージがバラバラだったため。
詳しいマシン構成まで決めていってもキリがないという意見もあり、メニュー内容を決めるまでにした。

※ この処理イメージは、後で設計を行なっていて再び問題になった点である。
どのように対応していったかは“3.解釈/ 意見の相違の扱い”でまとめる。

3.解釈／意見の相違の扱い

最終的に、問題文中の各処理で扱うデータの型は次のように決まった。

<1>本の貸出時に登録するデータ
- 本の ID
- 借りる人の ID
- 貸出日（日付は自動的に入る）
- 貸出期間（本によって異なる）

<2>本の返却時に登録するデータ
- 本の ID
- 返却日（日付は自動的に入る）

<3>本の属性データ
- 本の ID
- 題名
- 著者名
- 分野
- 出版社
- 版数
- 発行日
- 値段
- 貸出期間
- 登録日
- 貸出日
- 借りた人の ID
- 返却状態
- 削除日
- 削除した職員の ID
- 削除理由

<4>図書館を利用する人の属性データ
- 人の ID
- 名前
- 住所
- TEL
- 生年月日
- 登録日
- 職員／一般利用者の識別フラグ

分析中、グループ内で解釈が異なり意見が異なったのは、主に次の２つの項目である。（→ 以降は最終的にまとまった結果）

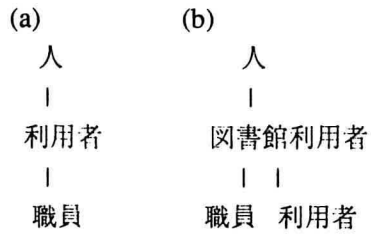
[1] 職員は利用者になるか
→ 職員も図書館の本は借りることができるので、利用者にもなる

[2] ユーザインタフェース（マシン構成も含む）はどこまで仮定しておく必要があるか
→ ログイン画面、職員が扱うメニュー画面を決めた
キーボード入力や出力装置は抽象的な定義だけ仮定しておけばいいことにした

これらの項目がそれぞれ１つの解釈にまとまるまでの経過を記述する

[1] 職員は利用者になるか
この問題を考慮するときに我々のグループではクラス階層を先に考慮した。

<<分析段階>>



「人」は図書館を利用する／しないに関係しない一般の人としたが、「利用者」クラスの位置付けをどのようにするかの論議となった。

最初は (b) の階層で話がはじまったが、
・“職員は利用者にもなる”と仮定できめた
・職員も一般利用者も同じ ID をふりわける
・今回の問題では、職員用と一般利用者用でマシンを分けるという制限を設けない
・職員と一般利用者がわかる区別をどこかでもっているだけでいいのではないか
の意見から、(a) のクラス階層で進めていこうと話がまとまった。
（人の属性データに、”職員／一般利用者の識別フラグ” をつけることによって利用者の区別をする）
この時点では「人」のクラスと「利用者」のクラスがあり、それぞれのクラスで扱うデータは次のようになった。

「人」のクラス	「利用者」クラス
・名前	・人の ID
・住所	・TEL
・生年月日	・登録日
	・職員／一般利用者の識別フラグ

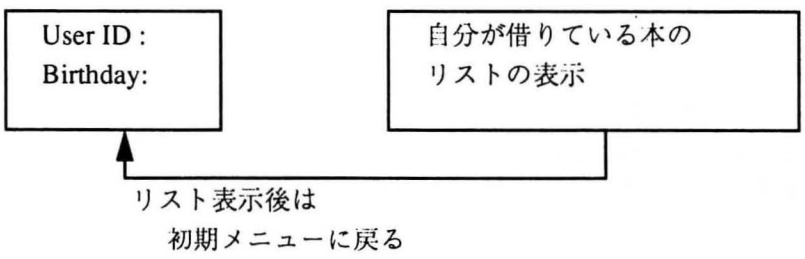
<<設計段階で変更>>
CRC カード手法を用いた設計手法で、「人」クラスはなくなった。この過程は "R-3 設計" のところで述べる。

[2] ユーザインタフェース（マシン構成も含む）はどこまで仮定しておく必要があるか
<<分析段階>>
中間発表の時、他のグループは図書館での処理を詳細に定義していた。
図書館が扱う処理で、
・図書館における本の容量
（買ったあとにおく場所がない可能性を考える必要があるのではないか）
・データベースの制限等、分析をするためには条件に制限をつける必要があるという意見がでた。
だが、
・今回の問題では容量は無限として考えても問題ない
・実際の分析の時に必要だということがわかっていればいい

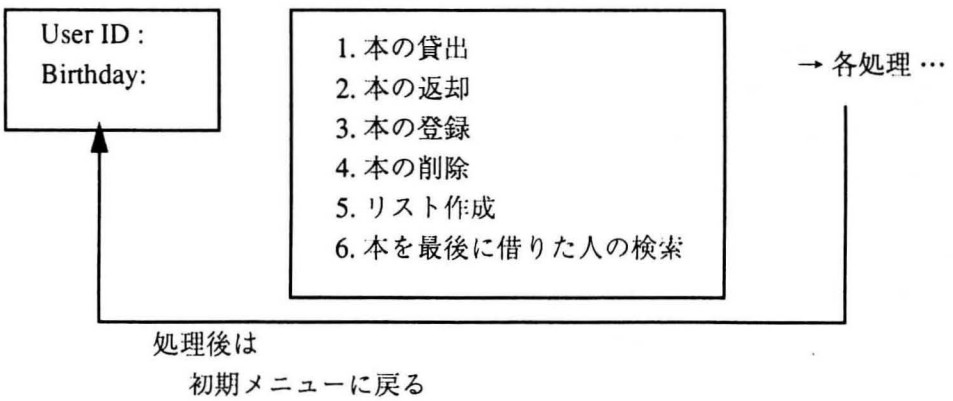
との意見が強く、入力データはキーボードでも何でもよく、ブラックボックスの扱いにした。また、メニューも必要な項目がわかればいいというように抽象的な定義ですませた。

ただ、処理の流れからユーザインタフェースの動作は最低限決めておく必要があり、職員と一般利用者が行なう 1 プロセスの処理を次のように決めた。

＜一般利用者＞
1. 自分の ID でログイン → 2. 自分が借りている本のリストを表示



＜職員＞
1. 自分の ID でログイン → 2. 管理メニューを表示する



4.作業の手戻り

作業中に発生した手戻りは、次のような事項である。

・確定した要求仕様の見直しの発生

(発生状況)

ＣＲＣカードをメンバー全員で書いて、シミュレーションを実施しようとしたところ、各自、図書館問題について様々なとらえ方をしていることが判明した。

(原因)

要求仕様をまとめる時に、図書館問題で扱う業務についてのシナリオを作らなかったため、各自バラバラの認識をしていた。

(対処)

(1) メンバー全員で、要求仕様の再確認の実施。

(2) 図書館問題をシステム化する上で必要なユーザインタフェースの画面を書き出して（主要メニューのみ)、業務の確定をした。

5.その評価、問題点

分析の評価は次の通りである。

・抽象化したレベルでの分析で済んだ。

ＣＲＣカードの大きさの制約から、具体的に記述することができないためと考えられる。

そのために、入出力装置、ハード構成などを限定しないで、分析・設計を進めても殆ど支障がでなかった。ただし、ある程度具体化する必要がある部分として、ユーザインタフェースの画面は考えておく必要がある。

・図書館で扱う業務に関するデータ項目の洗い出しが有効的であった。

図書館で扱う業務のイメージを把握することができる。また、分析・設計を通しクラスごとにデータ項目を分けていく必要があるが、クラス決定の判断要素にすることができる。

6.そこでの発見など

その他、気づいた点は次の通りである。

・野外での分析が有効的であった。

実際に図書館に行くまでには至らなかったが、図書館をイメージするのに、リラックスできる野外（岡崎公園）で実施した。その成果もあり、図書館問題の不明点、問題点が比較的スムーズにまとまった。

7.分析結果やワークシート

分析結果 … 各処理で扱うデータの型

(R-2 "3.解釈／意見の相違の扱い" で記載済)

R-3・設計

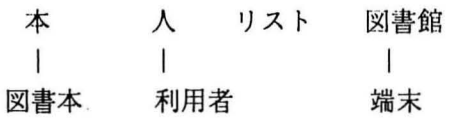
1.プロセス

我々のグループは次のような手順で設計作業を進めた。

(1) 各人がぞれぞれ思いつくクラスをＣＲＣカードに書いていく。
(2) (1) の結果を一人ずつ発表する。
　　(クラスわけを各自がどのようにしたか知るため)
(3) ＫＪ法の要領でＣＲＣカードを分類する。
(4) 分類した CRC カードのグループごとにクラスをつくる。
(5) 各クラスのデータを考える。
(6) 各クラスの責任と協力者を考える。
(7) クラスが出揃ったところでシミュレーションをする。

設計が進むにつれてクラスは次のように変化していった。

各人が書きだしたクラスをいくつかのグループに分類し、そのグループごとにクラスをつくると次のようになった。



次に、各クラスのデータを考えた。この段階で、利用者と職員の関係が問題になった。
確認者というクラスを設けてそこで利用者ＩＤのチェックをするようにした。

次に、クラスの責任と協力者を考え、シミュレーションをした。
その結果、責任を持たないクラスが幾つかあったのでそのクラスは削除した。
また、図書館コンピュータクラスを設けて、そのクラスに端末から入力された情報によってどのクラスを動かすかを決める役割を持たせることにした。

システム利用者　リスト編集　図書館コンピュータ
図書本　端末　メニュー

図書館コンピュータクラスに機能が集中しすぎているので、次のようにメニュー別にクラスを設ければよいのではないかという意見がでた。結局どちらにするかは決着がつかないまま設計作業を終了した。

システム利用者　リスト編集　図書本
初期メニュー　管理メニュー　端末

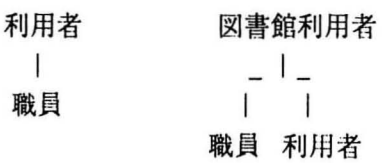
2.発生した問題点

・ユーザインタフェースのイメージが統一できていなかったので議論が進まなくなった。そこで分析に戻って、メニューなどの仕様を決める必要がでた。

・実際の利用者と、システム内での利用者の区別がはっきりしていなかったので、実際の利用者の行動（例えば「本をかりる」など）を利用者クラスの責任にいれてしまい、少し混乱した。

3.解釈、意見の相違の取扱い

・利用者と職員の関係をどうするかで議論になった。最初は次のふたつの意見がでた。



このときは、利用者にできることは職員にもできるという理由から、前者の利用者に職員が含まれるという意見を採用した。しかし、シミュレーションをした結果、職員クラスは責任を持たないことがわかったので、職員クラスを削除し、利用者クラスをシステム利用者クラスとすることにした。

・本の検索をどのクラスで行うかが問題になった。図書本クラスに検索の責任をもたせようという意見と、そうするのであれば図書本クラスに本の総数というデータを持たせなくてはならない、そうでなければ検索を行う別のクラスを用意するべきだという意見がでた。これについて、塩谷氏に質問したところ、それは言語に依るもので設計の範囲を越えてしまうということだった。
そこで、図書本クラスに検索の機能を持たせることにした。

・利用者クラスと本クラスがデータとして同じ利用者ＩＤを持っているが、これはオブジェクト指向の考え方に反しているのではないかという意見がでた。しかし、利用者ＩＤはこの双方のクラスに必要なデータであり、本クラスで利用者ＩＤを書き換えるわけでもないので、いいだろうという結論になった。

・図書館コンピュータは役割が多すぎるので、メニューごとにクラスを用意して、図書館コンピュータを削除してしまおうという意見と、図書館コンピュータは指令を出しているだけで、実際にはほかのクラスが働いているので、役割が多すぎる訳ではないという意見がでた。
この議論は最後まで決着がつかなかった。

4.評価、問題点

・議論が平行線をたどることが多く、設計が思うように進まなかった。特に図書館コンピュータの問題では意見がまとまる前に時間切れとなってしまった。しかし、全体としてみれば、シミュレーションで要求された機能をすべて満たしていることが確認できたので、ほとんど基本設計が終わりかけていると考えていいのではないか。

5.感想、発見など

・設計が進むにつれてクラスが減っていったのは、よかったのだろうか？

・分析でもっと細かい点まで問題を想定したほうが、設計が順調にいったかもしれない。

R-4・設計書

＜設計の進捗状況＞

・クラス の洗いだしは終了した。（図 R4.1 CRC カード 参照）

・クラス の設定について 2 通りの意見があり収束していない。第 4 グループとしてはこの状態で作業を終えた。

案 1 :「図書館コンピュータ」を用い、全ての処理の制御は「図書館コンピュータ」が行なう。
(図 R4.2 案 1 の場合のクラス関係 参照)

＊使用する クラス＊
メニュー, 端末機, 図書館コンピュータ, リスト編集者,
図書館の本, システム利用者

案 2 :「メニュー」クラス を 2 つに設け、それぞれのメニューに関係する部分は、各クラスに責任を持たせる。
(図 R4.3 案 2 の場合のクラス関係 参照)

＊使用する クラス＊
初期メニュー, 管理メニュー, 端末機, リスト編集者,
図書館の本, システム利用者

- 図 R4.1 CRC カード -

Class	システム利用者	Data
Responsibilities	Collaboration	利用者 ID 資格 電話番号 登録日
・利用者情報を返却する		

Class	端末機	Data
Responsibilities	Collaboration	
・図書館コンピュータに入力事項を返却する ・結果を表示する	図書館コンピュータ	

Class	図書館の本	Data
Responsibilities	Collaboration	題名，著者名，分野， 出版社，版，発行日， ID，登録者，貸出期間， 登録日，貸出日， 借用人，返却状態， 削除日，削除者， 削除理由，値段
・借りた人の ID を登録する ・返却状態を書き換える ・本データを登録する ・返却状態を調べる ・削除データを登録する ・特定の作者の本データを返却する ・特定の分野の本データを返却する ・特定の利用者が借りている本データを返却する ・最後に借りた利用者の ID を返却する		

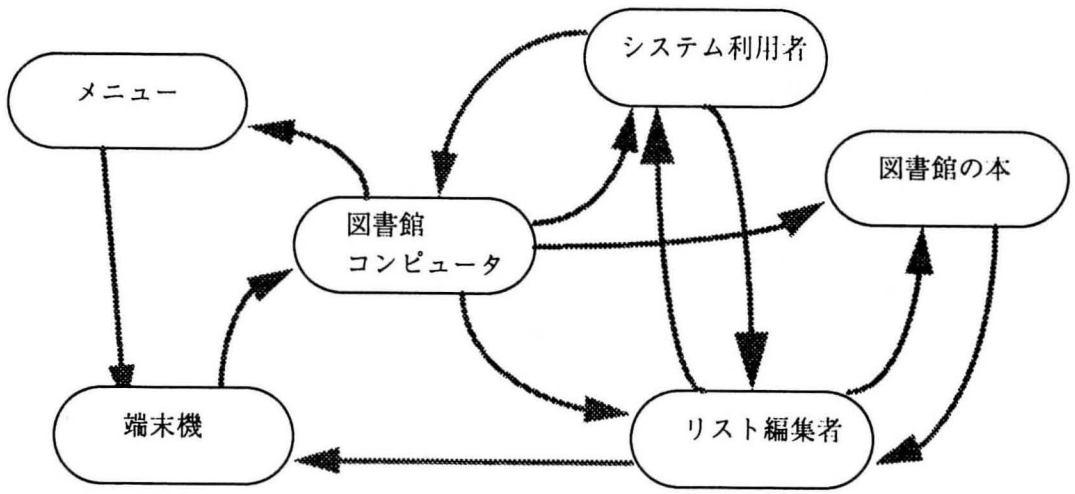
Class リスト編集者		Data
Responsibilities	Collaboration	
・特定の作者の本に関するリストを編集する ・特定の分野の本にかんするリストを編集する ・特定の利用者が借りている本のリストを編集する ・特定の本を最後に借りた人のリストを編集する ・編集結果を表示する	図書館の本 システム利用者 端末機	

Class 図書館コンピュータ		Data
Responsibilities	Collaboration	
・字句を解析し、ある特定のクラスに処理を依頼する ・システムの状態を保持し、次の状態を決定する	メニュー 図書館の本 リスト編集者	

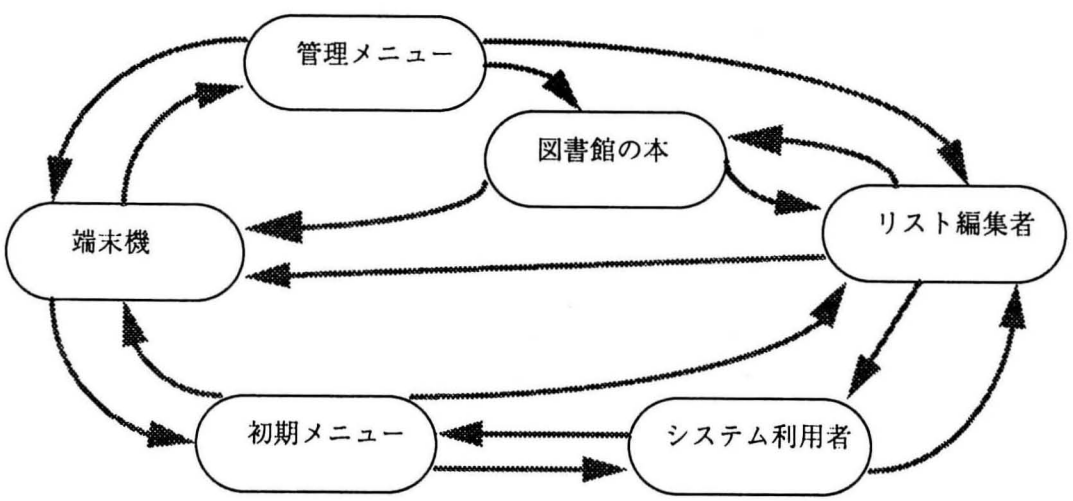
Class メニュー		Data
Responsibilities	Collaboration	
・端末機にメニューを表示する指示を行なう	端末機	第1メニュー (初期メニュー) 第2メニュー (管理メニュー)

Class 初期メニュー		Data
Responsibilities	Collaboration	初期メニュー
・メニューを端末に表示する	システム利用者	
・リスト編集を行なう	リスト編集者	
・管理メニューを表示する	管理メニュー	

Class 管理メニュー		Data
Responsibilities	Collaboration	管理メニュー
・メニューを端末に表示する	図書館の本	
・本の貸出を行なう	リスト編集者	
・本の返却を行なう		
・本の登録を行なう		
・本の削除を行なう		
・本のリストを編集する (著者)		
・本のリストを編集する (分野)		
・本のリストを編集する (利用者)		
・本のリストを編集する (最終貸出)		



- 図 R4.2 案 1 の場合の Collaboration 関係 -



- 図 R4.3 案 2 の場合の Collaboration 関係 -

R-5 全体評価

1.試用した手法の全体的評価（利点/ 難点は何か？）

（他グループの手法との比較）

・SA/SD 法との比較

SA/SD 法は、DFD によってのモデル化が中心となってくるが、オブジェクト指向法にもその流れを汲んでいるものが少なくない。DFD は、データの流れ、手続きの順が明確に記述されている分、拡張、変更に対する柔軟性が弱いといえる。これに対して、オブジェクト指向法は、一連の手続き順が明確に表現されていないが、拡張、変更に対して弾力的である。

・JSD 法との比較

JSD 法は、行動特にリアルタイムの行動に重点を置いているが故に、同期の並行処理システムのモデル化等に有効であると考えられる。しかし、オブジェクト指向法のように、属性を重視していないため、従来のデータベース設計には、やや不向きである。

（CRC カードを用いた手法と他の OOA/OOD 手法との比較）

CRC カード

＜利点＞

・１つ１つのクラスの役割とそのためには、他のどのクラスが必要となるかを綿密に確認しながら作業をすすめることが出来る。

＜難点＞

・オブジェクトの数に関する表現、継承表現が不明

・全体モデルを把握しにくい

Coad/Yourdon 法

＜利点＞

・モデルがすっきりしている為、特に分析の初期段階では扱い易い

＜難点＞

・オブジェクト間の関連に関する表現が不足

・イベント間の時間的関連に関する表現が不足

OMT 法

＜利点＞

・オブジェクトモデル、動的モデル、機能モデルを用意している為、様々な状況表現に対して、比較的弾力性がある

・特にオブジェクト間の関連に関する表現はかなり詳細である

＜難点＞

・かなり詳細な表現をとっているが故に、プログラミング言語によっては、実装段階で困難となってくる可能性がある

上記３つに挙げたオブジェクト指向手法について、共通に指摘できることは、

制約表現の未熟さである。
(ここでの制約とは、次の意味をもつものである。
1. 定義域制約
・ 属性のとる値の範囲の制約
・ キー制約
2. 関連制約
・ オブジェクト間における参照及び複合関係に関する制約
・ Composite Object におけるオブジェクトの複合関係に関する制約
)

制約そのものをオブジェクトとして表現していく等、
制約という概念を如何に機能的、動的にモデル化していくかが1つの重要な課題といえる

2.各自の現場への適用は可能か？、問題点と代替手段はあるのか？

(CRC カードが現場に適用できるかどうかについて)
・グループ内6人中、3人が使ってみても良いと答えた

(現場に導入/適用する際の問題点と改善案)
＜問題点＞
・実際に CRC カードを使うには、ある程度の人数が必要であろう
・クラス数が多くなるような大規模なシステムでは、カードを
扱う際の空間的問題が出てくる

＜改善案＞
・特に小人数で行なう場合には、クラスの役割と全体の中での位置付け
をはっきりさせるためにも、クラス間の関係（継承、全体ー部分関係
等）についての簡単な表記法を決めておく必要がある。

・また、カードが混乱しないように、ボード等の利用が有効であると考
えられる。

R-6・グループ活動の感想

<1> CRC 手法について

[個人別感想]
(古閑)
・ 人を説得するということの大変さと力のなさを痛感した。
特に考え方の背景，ベースが違う時はほとんど無力であり，有効なのは政治力かと
ちょっと悩んでしまった。

・ グループ活動のメリットの一つに，色々な意見を聞きよりよいものを作るという事が
あるが，この場合どのようにまとめあげるかがキーだと感じた。
この力が足りなかった為に PC の予想・期待に反して 設計終了と言えるグループが
少なかったように思う。

・ 設計手法についてもそうだが，グループ活動のプロセスについても現場の技術革新につながるような工夫をしたかった。
プロセス記述をやれないかと考えたが，結果的に出来なかった。(やらなかった)
プロセスについても何らかの手法を取入れて見るのは少し欲張りか？

・ プロジェクト運営なり設計作業なりで最低限持っておくべき知識と意識がグループ内にどれ位存在し，また必要と思われているのか確認出来なかった。
これは単に酒の量が少なかっただけか？

・ CRC に関しては今後現場に取入れるべく勉強を進めて行きたいと思っている。
しかしその時最低限必要なのは資料にもあった通りオブジェクト指向の概念を知っておくという事だろう。
現場に導入する時注意しなければいけないと分かったことは WS の成果の一つと言える。

(森下)
・ 各自のアプローチの方法が異なるため、意見がまとまるのに時間がかかった。

・ サブクラスはスーパークラスの属性を継承できるが、この関係を CRC 手法にどのように記述しておけばいいか悩んだ（あるルールづけが必要？)。

・ 分析をしすぎたということはないが、問題点が何かという見極めの難しさを痛感した。

(舟引)
・ 特に多人数の共同作業においては、システムの仕様（今回の場合は、図書館の規模、職員数、端末数等）に対する考えの統一が必要であると感じた。

・ 各手法に対して、やはりある程度概要をわかっていた方が良かった。

・ 互いにバックグラウンドが違う人間が集まって一つの作業を短時間で行なっていくことにより、何が分析で、何が設計かということを改めて考えることが出来たように思う。

(岡田)
・ オブジョクト指向というものが分かりかけてきた気がするし、それについて興味を持てるようになったのはいい収穫だったと思う。

・ 自分より経験をたくさん積んでいる人たちと一緒に勉強できて、いい刺激になった。

(門田)
・ G 4 グループの活動方針として、最終的な設計結果より設計過程を重視の
" コミュニケーションギャップをなくす" と" プロセスの明確化" を掲げ実施した。
時間の制約はあったものの、各自の意見を活発に交わし活動方針は達成できたと思う。
特にその中でも、構成メンバーが同世代とはいえ職務経験などの違いから様々な問題のとらえ方をし、お互いに正当性を認め合うのに時間を大分費やした感じがする。

このような事は実際の業務においては、システムの開発者としての思いはあるが、

工期などの制約により、徹底して議論することが出来なかったりする場合がある。

今回のワークショップでは、最終発表、報告書作成などの課題はあったものの
普段の業務では経験できないことを試すことができた。
また、手法に関しても理解のレベルはあるものの、初めてオブジェクト分析／設計を
試すことができたことも、今後のシステム開発において参考になった。

(工藤)

・「主張」することの難しさ、「説得」する難しさを感じた。
外部の環境に応じて己の理念を柔軟に対応させることが必要。
ワークショップはためになる。
CRC は面白い。設計手法を内蔵したすばらしいツールと判断した。
実作業への適応は難しい面もある。
自身の考えとしての「設計」は多少明確になったが、未だにはっきりとした定義づけが
できないでいる。

[各自の感想のまとめ]

<1> CRC 手法について

(1) 手法を含め、オブジェクト指向分析／設計をある程度わかってい
た方が良かった。

(2) サブクラスとスーパークラスの継承の関係をどのように記述して
おけばいいか悩んだ。

(3) 設計手法を内蔵したすばらしいツールと判断した。

(4) 今後、この手法をそのまま現場へ適用するのは難しい面もあるが、
勉強を進めていきたいと思っている。

(5) オブジェクト指向分析／設計に興味をもてるようになった。

<2> 分析／設計について

(1) 共同作業により分析／設計を行なう場合は、システム仕様に対す
る考えの統一が必要である。

(2) 分析をしすぎたということはないが、問題点は何かという見極め
が難しかった。

(3) 設計に関して、未だにはっきりした定義づけができないでいる。

(4) 分析／設計のプロセスについても、現場の技術革新につながるよ
うな工夫をしたかった。

<3> グループ活動について

(1) 構成メンバーの背景となる職務経験などの違いから、様々な問題

のとらえ方をし、意見がまとまるのに時間がかかった。

(2)「主張」することの難しさ、「説得」する難しさを感じた。

(3) 作業を進める上で、最低限もっておくべき知識と意識がグループ内にどれ位存在し、また必要と思われているのか確認できなかった。

(4) 色々な意見を聞き、より良いものを作るということもあるが、時間内に設計終了するようにまとめあげることもキーだと感じた。

(5) お互いにいろいろな背景の経験を積んでの活動となり、いい刺激になった。

若手92の感想

福良 博史

今回初めて若手の会に参加した。この会は, 岩手で開催される場合が多いそうですが, 今回の会場は愛知県岡崎市のホテル「レク・ワールド」でした。ホテル側の手違いで, 会場の部屋割りに最初戸惑ったが, 最終的には大ホールを一部屋割当られた。これがかえって良かったと思う。それは, 各グループ毎に部屋が割当られた場合には, 相互の交流が難しいのに対し, 大ホールを全グループに一部屋割当られたために, グループ毎の間仕切に屏風を使った。こうする事で相互の状況が理解しやすく, グループ間の交流と刺激を促進した。また各グループのPCも他のグループの間を見て廻る事が容易に出来たために情報交換を行いやすい環境になった。

今回のワークショップは, かなり熱の入ったものとなり, 徹夜したグループも現われた。今回の参加者は, 皆それぞれに設計についての様々な考え方, 纏め方, 発表方法など色々と有益な事/考えさせられる事などがあったと思う。

以下に私がPCとして参加して感じた事を列挙する。

1. 分析にかなり時間がかかりすぎた。

   今回の目的として考えていたことに, 出来れば, 設計完了後に, 大きな仕様変更を生じさせ, その影響によりどのようなことが設計に生じたか, を見たかった。しかし殆どのグループが, 分析に1日以上費やした。この原因は, 今回のケースとして用いた仕様に曖昧な箇所が多かったためにグループ毎に解釈がかなり異なったものとなってしまったことが主な原因だと考えられる。もう少し, 曖昧な箇所を少なくし, 分析が早く終えられるようにすることが必要であった。

   又, 今後のやりかたとしては, この種のテーマは焦点の当て方によって以下のような方法が考えられる。

   (案1) **要件を明確化するためのインタビュー方法の訓練**

   今回のワークショップで感じたことは, 同じテーマでもやはり十人十色の纏めかたがある, ということである。このためにどのようなアプローチを用いるのかによって, 要件の纏め方の品質/丁寧さなどが異なる。このような点を検討してみるのも興味深いことだと思う。この場合に, 前提として, 主催者側が, 綿密なモデル (図書館など) の組織, 運営業務, 問題点などを仮想しておき, 要件としては, 今回の仕様程度の内容を提示しておく。そして, システムの顧客 (図書館の館長など) を一人設定し, 各グループ毎に質問は, 顧客にする。顧客は, 聞かれた事のみに答えるようにする。このようにして各グループ毎に纏めた結果と事務局側で設定したモデルとの比較検討をする。

   (案2) **設計方法論を身につける**

   「設計が完了した」という事の客観的な評価が測定しにくい。本当に評価しようとすれば, プログラムを造り, 動かし, 要件に変更も加えてみる必要があるのではないか。このためには, 開発環境を限定し, 言語, DB, DC などを限定し, しかも設計者と製造者を別人格とし, 完成したモノを使ってみなければならない。そして各種のデータを集めて, 検討する。

   但し, これを行うには設備と開発環境に対する経験が必要になる。またモノを動かすところまで試行するとなるとかなりの時間が必要となる。

   (案3) **設計方法論の比較検討**

   (案2) を各種の設計方法論で実施する。

   これを行うには, 時間と人が沢山必要となる。これを解消するためには, 1年毎に異なる設計方法を実施するしかないか? これでは最終結果が何時出るか不明となる。しかし毎年行っていけば, 2年目以降は, それまでの結果を参考にでき, 比較検討も徐々にできそうな気もする。

   (案4) **開発環境の比較検討**

   詳細設計書は事務局で完成品を用意し, 各種の環境でモノを造り, 動かし, 保守を行い比較検討する。

2. 分析終了といっても, ユーザ・インターフェースがバラバラで, 相互の意志が統一されていない場合があり, お互いに誤解を生まないように意志統一する事の難しさを皆が理解したと思う。

   はじめにユーザの画面構成と画面レイアウトを決めることにより, 相互の意志が統一されるようになっていった。

3. CRCカードに, クラス名だけ記述して並べていると, 人により思惑が異なりなかなか話が統一出来ない。これは, 各クラスの役割を皆が勝手に想像してしまう事に起因している。CRCカードに各々の責任事項 (役割) を

株) ジェー・エム・エー システムズ

明記するようにした結果, 相互に各々のクラスの構成についての理解がしやすくなっていった。

4. 各人が1つのCRCカード(クラス)となり, 人間コンピュータとなり, 一つの業務を行う場合の動きを検証した。これは, 設計者全員が共通の認識を確立できるので, 設計の検証方法としてはなかなか良い方法だと思う。
5. クラスの分け方に未解決の問題がある。
   - 単に機械的に名詞を洗い出した後, クラスの候補を選出し, そこにどのような責任/役割を与えるかによって, クラスの構成が異なって来る。
   - どうすれば, 客観的に合意を得るクラス構成がとれるか?
   - どうすれば, 客観的にバランスのとれた責任の役割分担が定義出来るか?
   - 幾つかの案から一つに絞るには, 「エイヤッ」と決めるしかない?
6. responsibility-drivenの考え方は, 割合よく理解してくれていたと思う。
7. DBの設計に時間をかけたグループと, RDBを簡単に想定して, DBは何でもしてくれる, と想定したグループがある。
   この当りも本当は, 条件設定しておくほうが良かったように感じる。
8. 今回の参加者のバイタリティの凄さと真面目さ, には感心した。
9. PCとして皆の討論に直接介入せず, 進行を援助することの難しさを感じた。ともすると, 自分の考え方を押し付けたい感情に駆られる。心掛けたことは, 以下の点である。本当にこうだったか, それは参加者の評価に委ねる。
   a. 討論内容の論理矛盾の指摘
   b. 討論の進行の舵取
   c. 各種の疑問に対しての相談

その他雑感

1. 屋外でのディスカッション
   二日目に, 第4グループは, 昼食を外で採ることにした。特に目当てがあったわけではなく, 弁当を買い, 岡崎城にて食事となった。この後, 木陰のテーブルを囲んだベンチを見つけたので, 午後はここで分析の続きの討論をおこなった。ここは, うまい具合いに, 人があまり来ない所で, 後ろは小山になっており, 前面は池となっているために, 静かに気を落ちつけて議論に専念できる場所であった。この討論で, グループの皆が打ち解けて, 最初はギクシャクしていた議論が徐々に滑らかさを増していったように感じられる。
2. 山崎氏の「Z notation」の紹介
   日頃, 事務処理アプリケーションのソフトウェアの開発/保守を行っている。自分の仕事の現場では, 形式的記述という言葉は, 殆ど聞かれない。私自身は, 昔(15年位前?)何かの論文を調査していた時にVDLという仕様記述言語を使って, IBMがPL/Iの定義をしようとしていたといったような話を読んだ事があった。その時に少し論文を探してみた。しかし, その当時, あまりよく理解出来ず, 仕事が多忙を極めていたために, そのままうっちゃってしまっていた。
   それが, 今回の若手の会における特別講演(日本ユニシス 山崎氏)の話が, この形式記述についての紹介であった。しかも内容を判り易く噛み砕いた説明であった。
   内容は, 幾つかの形式的仕様記述の考え方の流れから入り, Zという形式記述の考え方を利用して, 図書館問題を例にして, どのように記述していくのかを示していただいた。
   この山崎氏の話から, 以下のような印象を受けた。
   (a) システムの設計への導入
   うまく記述できれば, 未だインタビューしていない内容の類推が可能となる。(逆に言うと, 冗長な質問はしなくてすむ)
   複雑なシステムの記述を行うことにより相互の矛盾を発見することが可能となる。
   (b) 設計が完了したという事の評価
   設計が完了したか否かの評価に利用できるかもしれない。単一の仕様の記述だけでは, 難しいと思うが, 既存の類似の仕様の記述が保存してあれば, それと, 現在開発中の仕様との比較による評価が可能かもしれない。
   (c) 設計の再利用
   形式記述で表現したものを, 類似のシステムの仕様記述に容易に再利用することができるかもしれない。
   (d) このような考え方をSE/プログラマの誰もが身につけるような文化を醸成することがソフトウェアの品質向上に欠かせないと感じた。
3. 岡崎の味噌煮込みうどん
   このうどんは, 名物とのことで, 土産に買って帰り, 家で煮て食べた。うどんが, 硬く, 味噌はからかった。半分程食べた後, 味噌煮込みうどんをそのままフライパンで焼きうどんにした所, 非常に香ばしくうどんも柔らかくおいしく食べられた。

「若手の会」に参加して

後藤 浩

1 はじめに

「若手の会」にはメンバとして過去2回参加している。今回は, 諸々の経緯でプログラム委員として参加した。
CRC カードを用いた, オブジェクト指向設計を試みる第4グループを担当したが, グループの世話係に徹し, 討論に加わることや, 問題解決に向けての方向付けを示唆するなどプログラム委員としての本来の役目を果たすことができなかった。残念でならない。
(初めから力不足であったという声も聞こえてくる...)
そんな訳でここでは, 「若手の会」の4日間を時間の経過で振り返り, 感想を記す。

2 集合まで

プログラム委員の集合は11時であった。
新幹線[†1], 名鉄, タクシーを乗り継ぎ, 会場の「レク・ワールド岡崎」に着いた。1階がパチンコ屋だったのには驚いたが, 「レク・ワールド」の「レク」は「レクリエーション」のレクだと知り, 妙に納得してしまった。

3 講演/PC パネル/グループ分け

講演は, プログラム委員長でもある SRA の塩谷さんの「オブジェクト指向分析/設計外観」が行われた。
PC パネルは, 10名のプログラム委員が順番に担当するグループで採用した設計方法論についてのアウトラインと方針を述べた。日本ユニシスの山崎さんの「プログラムを書いていた頃には, 設計方法論など無かった」という発言と, 私自身が数十人を前にし, ひどく緊張したことが印象的であった。
グループ分けは, 第3希望までを募り, プログラム委員で調整した。なんとなく CRC カードのグループの人気が高かったように記憶している。ともあれ, 第4グループのメンバ6名がここで決まった。

4 グループ討論

グループ分けが終了し, 夕刻から顔合わせのグループ討論が始まった。PC を含む8人の自己紹介が終わる頃には, 和やかな雰囲気の中...っというわけにはいかず, 私とよく目の合う位置に座ったヒラタソフトウェアテクノロジーの古閑さんがこの席の議長ということになった。
4日間のおおまかなスケジュールと方針を決めた。4グループの方針は「隠しごとをしない」(?) になった。
グループ討論で印象的だったのは, SRA の森下さんの「プログラムを書くためになぜ"設計手法"を使わないといけないのか」という問いかけであった。明確な答がみつからいまま現在に至っている。

5 グループ紹介&パーティ

場所を移し, グループ紹介を兼ねた立食パーティを行った。各グループとも特色あるグループ紹介であった。4グループは各人が川柳でワークショップに望むことやテーマに関することを表現した。さくらケーシーエスの船引さんの「オブジェクト, しこう (指向/思考/試行) しなけりゃ, ただのもの」に拍手を贈った。

6 グループ別設計方法論レクチャ/自由行動 (設計作業)

明けて9月3日の午前中は, グループ別の方法論のレクチャを行った。4グループは3グループと合同で CRC カードを用いた設計手法のレクチャを受けた。講師は, ジェーエムエーシステムズの福良さんが行った。
グループ別自由行動は, 岡崎城へ足を運んだ。メンバの方々の意気込みを感じさせられたのは, 作業の時間を惜しむ余りにホカ弁屋の弁当を境内に持ち込むことになったことだ。結局, 鳩におびえながら弁当を啄んだ。
夜ともなれば若い2人が語り合うにふさわしいと思われる屋根付きの恰好の場所に, 昼間から若手の6人は場所を陣取った。そこで数時間, 設計作業に勤しんだ。(でいるように見えた)

株) シスプラン

†1 新幹線の切符は新橋のチケットショップで買った。JR の正規料金よりも若干安く買えたことは, ご存知の通り。往復で弁当代位にはなります。

7 中間発表/最終発表/全体討論

中間発表は3日の夕方に行った。参加メンバの全員が発表の機会を与えられるという形式で行われた。この形式は効果があったように思う。

最終発表, 全体討論についてはあまり覚えていない。報告書(SEA MAIL)が完成し, それ(実はこれ)を改めて手にしたときに記憶を新たにしたい。

8 おわりに

「若手の会」が終わり, 今年も夏が終わった。[†2] 最後に全体的な感想を記す。

メンバの方々に関しての感想であるが, 良くも悪くも真面目であったと思う。もっと脱線した話題で盛り上がるとか, 突拍子もないことをやるとかあってもよかったと感じるのは私一人でしょうか。

3グループにメンバとして参加した同僚の井上さんが, 今の仕事で, CRCカードを使って設計作業をしている。頼もしいというか, 嬉しい気持ちになります。私は, CRCカードで見積書を書くわけにはいかない...

9 おまけ

ここでは, 「若手の会」開催の前後に私が取り組んだことをまとめる。[†3]

9.1 PCになった訳

SEAの環境分科会で活動していて, 92年の「若手の会」は環境分科会のメンバ中心で会の運営を行うことになったのことがいきさつであった。

91年の11月頃だったように思うが, 分科会後の会食の席[†4]で, 「若手の会」の話題で盛り上がったのとを覚えている。この時点で, 個人的にはやる気になった。

9.2 社内に対してのコンセンサス

「若手の会」に参加するにあたって会社に対して, 数日間留守にする点と, 数万円の費用を使う点を認めて頂くために, 規則に従い申請書を提出した。

目的に記した項目は,

- 同年代の参加メンバと活動を供にし, コミュニケーションを行うことにより, 視野を広げたい。
- この程度の規模のイベントを運営者側で経験できる機会を失いたくない。

だった。

岡崎城の天守閣にも登ったことだし, 視野はそれなりに広がったと思うが, 運営者側の経験という点については, 良くも悪くも大変であった。

9.3 プログラム委員によるミーティング

開催資料の作成, 参加メンバの選考, 日程の調整などを行った。

9.4 予習

環境分科会の活動の一貫として, 図書館問題をCRCカードを使って設計した。

9.5 資料作成

開催当日の資料を作成した。私の担当は, CRCカードの例の部分だった。

9.6 反省会

「若手の会」が終了し報告書発行の目処が立った92年に暮れに, 反省会[†5]を行った。

9.7 報告書発行

「若手の会」の報告書としてSEA MAILが発行される。私が担当したのは, ポジションペーパの体裁合わせとこの部分である。この作業を通して, TeXと出合い, 少しだけTeXのことが分かり, 結構好きになった。

---

†2 なぜかもうすぐ来年である。

†3 年が明けた。紙面の都合でもう少し長い原稿を書くようにという指示がありました。

†4 環境分科会はSEAの会議室で行うことがほとんどで, この日もそうであった。散会後は, 四谷近辺の飲食店で雑談を交わすことが慣例になっている。この日は美食家が集まる中華料理屋の隣のすこし汚いモツ屋だった。

†5 この日は四谷のモツ鍋屋になった。流行に関係なく, みんなモツが好きなようである。

Ｇ５　ＪＳＤ　設計最終報告書

［R-1］グループの活動概要

1．メンバの簡単な紹介と役割分担

氏名	会社名／所属	グループ内での役割	業務傾向／特徴
西岡　啓和	さくらＫＣＳ	Ｒ３担当 運転手	データベース （ＲＤＢ、ＯＯＤＢ）
富岡　雅己	ジェーエムエーシステムズ	Ｒ２担当	工程管理
江谷　典子	富士ゼロックス情報システム	Ｒ１担当	ヒューマンインターフェース
岩城　弘二	富士ゼロックス情報システム	Ｒ４担当	ＯＯ、ソフトウエア工学 開発支援
水谷　公一	高岡製作所	Ｒ５担当	Ｃ言語、開発支援
東宮　祐一	ネックストファンデーション	Ｒ１担当	テクニカルライタ

2．使用した手法名

ＪＤＳ法（Jackson System Development）

3．解いた課題（特にグループ独自で拡張した部分に関する説明）

(1)共通問題の解決について

「一回にある決った数以上の本」の仕様について、「一人当り最大数制限」と解釈した。

(2)共通問題の拡張について

・検索リストに貸し出し可否を入れる
　リスト出力は貸し出しを前提とする場合が多いので可否を示した方がよい。

・本による貸し出し期限の設定
　貸し出し期限を設けないと永遠に帰らない新刊、人気本は回転を早くしたい。

・返却されない場合の処理
　返却されないことが起こり得る。

・ＡＮＤ／ＯＲ条件による検索
　自由に検索を行い対象をしぼりたい。

・予約、待ち人数の照会
　確実に本を手に入れたい。

・購入希望図書受け付け
　利用者のニーズに答える。

・貸し出し平均日数のリスト出力
　貸し出し状況を把握しニーズに答える。

4．作業プロセス

-1.作業分担を決めた。
-2.ジャクソン法の概要について参考文献の読み合わせを行う。
-3.ジャクソン法のシステム開発手順に従い、各段階ごとに分析および設計を行った。
-4.共通問題の拡張を行った。
-5.文献の不明点、疑問点は適時、山崎氏にご指導いただいた。
-6.ジャクソン法を理解するのに手間どったため、思い通りのスケジュールにはならなかった。

［R-2／R-3］分析内容及び設計

1．着目点，アプローチの概要

第５グループは、JSD法を用いて分析設計を進める。JSD法の作業は以下の６つの段階からなる。

(1) 実体行動段階
(2) 実体構造段階
(3) 初期モデル段階
(4) 機能段階
(5) システムタイミング段階
(6) 実現段階

これらの段階を通して、分析工程と設計工程に相当するものを明確に分けて示すことはできない。また、実際の作業もこれらの各段階毎に分析、設計を行なった。そこで、当報告書では、上記段階に沿って分析内容、設計をまとめた形式で報告する。

［各段階での作業概要］

(1)　実体行動段階

まず、モデル世界で現われる実体とその行動の選択を行う。ここで注意すべきことは、入れ物的なもの（本棚，倉庫など）は実体として扱うべきではないという点である。また、システムの動作のうち、実体の変化を伴わないものは、実体の行動とはしないという点である。

(2)　実体構造段階

実体の行動は、時間的に順序づけられているので、その順序を表現する。表現にはジャクソン構造図を用いる。

(3)　初期モデル段階

ここで実世界とモデル世界との対応（結合経路）を記述する。また、実体の行動の仕様を記述する。

(4)　機能段階

システムの出力を作り出す為に機能を使用化する。必要に応じて新プロセスを追加する。

(5)　システムタイミング段階

タイミング及び動機プロセスについての検討がなされる。業務時間の制約とか端末の応答速度等をインフォー

マルに記述する。

(6) 実現段階

実際に発見する機能の環境を想定して、その実現方法を検討する。

2．分析で困ったこと（発生した問題点）

メンバー全員がＪＳＤ法を知らなかったため、最初のとっかかりである実体や行動として何を採用してよいものか全く判らない状況であった点、困ったことと言えよう。

当初、何を実体や実体の行動とすべきか判らず、後から考えれば実体とすべきでないものを実体として捉えたり、実体の行動として実現すべきでないものを行動として考えていたりした。実体や行動として考えたものを以下に列挙する。これらのうち結果的には誤った（不都合のある）ものもあった。

・最初に実体として考えたもの

本、利用者、職員、図書館、書庫、本棚

・上記のうち結果的に実体としなかったもの

図書館，書庫，本棚　（入れ物は実体とは捉えない）

職員　（サービス機能を提供するものを実体と捉えない）

・また、各種検索リスト作成の機能を実体における行動と捉えていた。

（サービス機能は、後の段階で出てくるスケジューラによって扱われるプロセス）

以上に挙げたように、最初のうちは何を実体や行動とすべきかどうかメンバー間、メンバー個人の意識の中で混沌とした状態であった。この状態からＧ５のメンバーを救ってくださったのが他ならぬ、山崎さんのアドバイスであった。

機能段階においても、まず最初に「機能」という言葉が何を意味するものなのか理解できなかった。機能として、実体行動の一部を詳細化するものとの誤解も生じた。

システム・タイミング段階では特に問題となる項目は挙がらなかった。

実現段階においては、この段階で最終的に表されるべき形態がメンバ間でイメージしにくかった。また、今回の作業では、この段階でのハード面の考慮はしなかった。

3．解釈／意見の相違の扱い

我々の場合、メンバーの考えに食い違いが出るときというのはメンバー各人のＪＳＤ法の理解不足による場合が多かった。そういうときは参考文献などを調べる作業はするものの、実情はとにかく山崎さんに御教授願うしか手立てがなかった。上記の問題点の項で挙げた実体や行動の選択では各人各様の捉え方があった。全体的にオブジェクト指向的な感覚で考えていたようであった。以下に、実体や行動の採択で意見の食い違いの見られた点を列挙する。

・入れ物は実体とするのか否か？
・各リスト出力の機能はどの実体の行動と考えるのか？

主に、これらの点でメンバー各人のイメージに差があったように見受けられた。この解釈の違いも、山崎さんのアドバイスなしでは解決できなかった。（４を参照のこと）

機能段階においては上記の問題項目でも述べたように、機能というものについての議論が噴出した。また、この段階で属性を具体的に捉えようとしたが、それはこの後の段階で行なえば良いものであった。

実現段階では、今回の活動作業として基本設計書を作るという目的の為にこの段階までの作業が必要かどうかの議論が出た。結果的に、Ｇ５では実現の概要を示すシステム実現図（ＳＩＤ）として書くまでとした。

4．作業の手戻りはどの程度あったか？（プロセスを書き留める）

ＪＳＤ法の前半(1)から(2)の段階で、特に大きな手戻りは無かったと言えよう。これらの段階では以下の３つのポイントに留意して実体と行動の選択を行うことが重要である。これらのポイントは全て山崎さんのアドバイスによるところが大きい。

(1) 入れ物は実体にはならない。

図書館問題では、例えば、”図書館”、”書庫”、”本棚”など入れ物を示すものは実体としては捉えない。よって、実体としては”本”、”利用者”が採択される。

(2) サービス機能は実体の行動とはしない。

ここでのサービス機能とは、実体の状態変化を伴わない機能を指す。図書館問題での、各種リスト出力機能は実体である”本”や”利用者”の状態変化を伴わない。よってリスト出力機能は実体の行動とはしない。リスト機能は、後の機能段階で機能プロセスにより実現される。

(3) 実体の構造図（文）を考える際には、実体の生成から消滅までの期間を考える。

実体の行動の構造は、実体の存在に沿って考える。これより。あらゆる実体には発生と消滅の行動を伴うことが判る。

ＪＳＤ法の第一段階である”実体行動段階”で以上の点を考慮して実体とその行動の選定を行うと、以降の各段階での作業が格段に進めやすくなる。これらのポイントを知らずに、Ｇ５の作業を進めたとしても、ＪＳＤの６つの段階をワークショップ会期内に経験することはできなかったであろう。ここでのアドバイスがなければ、この報告書が存在していなかったかも知れないと思うと

改めて山崎さんに感謝いたします。

5．その評価．問題点

我々Ｇ５は、山崎さんに手取り足取りＪＳＤ法の進め方について教えていただき、おかげでＪＳＤ法の作業概要をおぼろげながら理解できたた。しかし、逆に失敗の経験をできなかった点が悔やまれる。例えば、入れ物を実体として採択し、作業を進めるとどういった結果になるのか。但し、この結果は報告書の消滅を想像させてもくれる。

山崎さんのお話は非常に説得力がある。恐らく山崎さんは、失敗を重ね分析／設計作業方針の裏付けを身に付けられたのではなかろうか。我々Ｇ５のメンバーはＪＳＤ法の方法論の裏付けを修得していない。今後、実際にＪＳＤ法を用いたり、人にＪＳＤ法を伝えるならば、ＪＳＤ法の進め方の裏付けを学ぶ必要がある。

以上、Ｇ５の作業の中で我々がＪＳＤ法の作業で失敗の経験をできなかったことも、問題点と考えられるであろう。また、指向錯誤しているだけの時間的余裕がなかったともいえる。

6．そこでの発見など

機能拡張の作業を通じてモデルの変化の小ささに我々は驚きすら覚えた。逆に、機能拡張にたえられるモデル構成の重要さ、即ちＪＳＤ法での実体と行動の選択の重要性が今回のワークショップでの我々Ｇ５の発見の一つと言えよう。

また、実現段階で取り扱われるスケジューラは、単に通信文を一手に引受けてディスパッチするだけの役割ではなく、通信文と共にコントロールをも持ち込むものであって、これは昨今のウィンドウ・システムと似ているとの発見も挙げられる。

7．分析結果または主要なワークシート

(1) 実体行動段階での結果

以下に採択された実体とその行動を挙げる。

実体：本 行動：登録される、貸し出される、返却される、削除される	実体：利用者 行動：登録、貸りる、返却する、抹消
登録される：本が図書館の蔵書として登録される。 属性：日付，本情報 貸し出される：本が利用者に貸し出される。 属性：日付，利用者 返却される：本が利用者から返却される。 属性：日付，利用者 削除される：本が図書館の登録から抹消される。 属性：日付，理由	登録：図書館に利用者として登録する。 属性：日付，氏名，性別，住所 貸りる：利用者が本を貸りる。 属性：日付，本 返却する：利用者が本を返却する。 属性：日付，本 抹消：図書館の利用者登録から抹消する。 属性：日付，理由

以上，実体として”本”と”利用者”の２つが採択された。

(2) 実体構造段階の結果

実体構造段階の結果は時間順序を明示した構造図である。以下にＧ５で得られた構造図を示す。

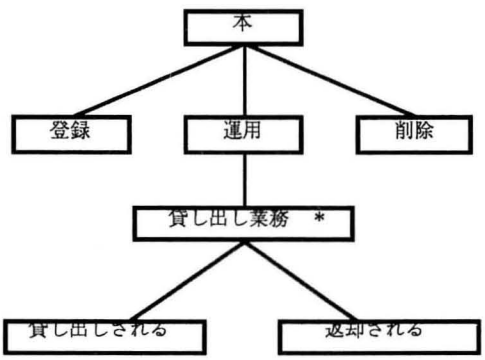


図２－１　実体 ”本”の構造図

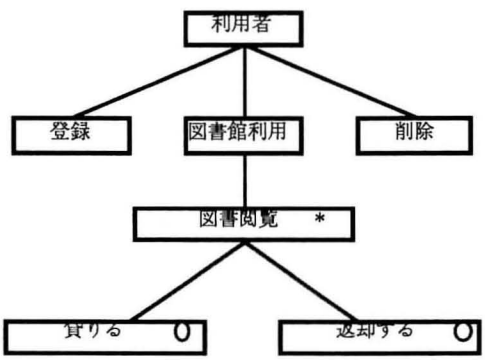
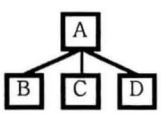
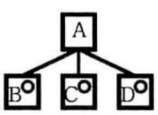


図２－２　実体 ”利用者”の構造図

上記構造図におけるＪＳＤ記法は以下に示すものである。



Ａは連接要素



Ａは選択要素

Ａは反復要素

（3）初期モデル段階での結果

初期モデル段階では、まず実世界のプロセスとシステム内部のモデルプロセスの関係をシステム仕様図（ＳＳＤ）を用いて表す。以下に，Ｇ５の初期モデル段階で得られたＳＳＤを示す。

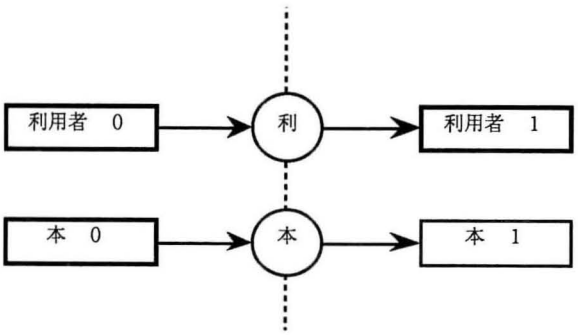


図２－３　初期モデル段階でのシステム仕様図（ＳＳＤ）

図2-3で添え字０は便宜上、実世界のプロセスを表し、添え字１は同じくモデルプロセスを表している。

初期モデル段階では、さらに構造文を書くことによりモデルプロセスを仕様化する。以下に、利用者１、本１の構造文を示す。

```
利用者１の構造文

利用者１ seq
  read利;
  登録; read利;
  図書館利用 itr while (借りる ,返却する)
    図書閲覧 sel (借りる)
      借りる; read利;
    図書閲覧 alt (返却する)
      返却する ; read利;
    図書閲覧 end
  図書館利用 end
  削除;
利用者１ end
```

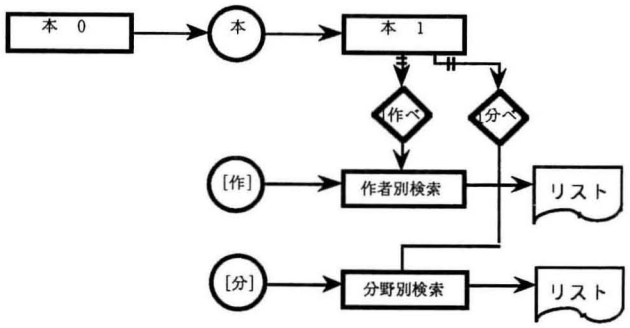
```
本１の構造文

本１ seq
  read本;
  登録; read本;
  運用 itr while ( 貸し出される, 返却される)
    貸し出し業務 sel (貸し出される, 返却される)
      貸し出される; read本;
    貸し出し業務 alt ( 返却される)
      返却される ; read本;
    貸し出し業務 end
  運用 end
  削除;
本１end
```

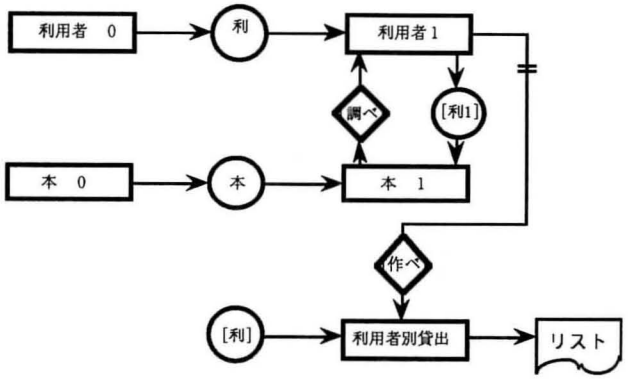
（4）機能段階での結果

初期モデルＳＳＤ（図２－３）を用いて、以下に示す機能がどのようにモデルプロセスに組込まれ結合されるかを示す。今回の実作業では、機能の詳細仕様を示す構造文は時間的制約の為作成しなかった。

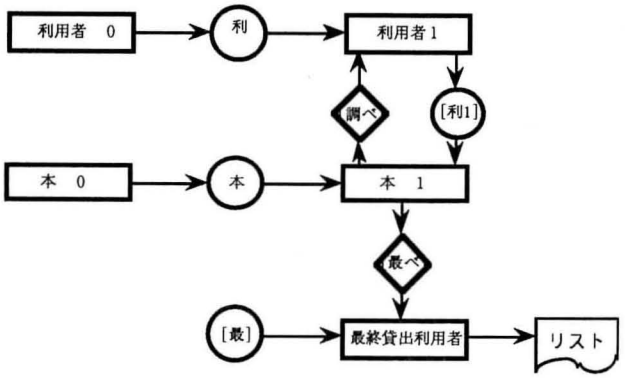
（ａ）特定の作者の本、あるいは特定の分野の本に関するリストの作成



（ｂ）特定の使用者が借りている本のリストの作成



（ｃ）特定の本を最後に借りた人の検索

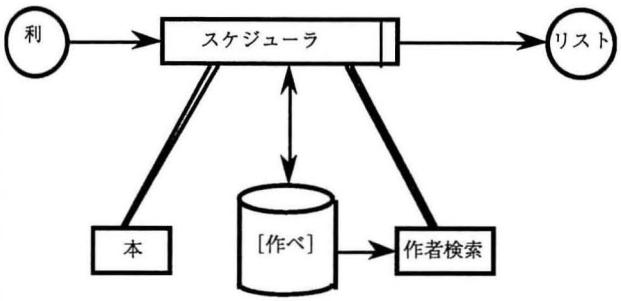


（5）システムタイミング段階での結果

本の貸出、返却等の行動が発生する場合、その都度システムが利用されるものと考えると、各機能の出力に時間の遅れは発生しない。

（6）実現段階での結果

機能段階で得られたＳＳＤのうち、特定の作者の本を検索する機能に対する実現の概要をシステム実現図（ＳＩＤ）として示すと以下のように表される。



同様に、他の機能に対してもＳＩＤが表される。

以上の作業で得られた最終的なＳＳＤ（システム仕様図）、ＳＩＤ（システム実現図）を次の項目であるＲ－4設計書において提示する。

8．システムの拡張

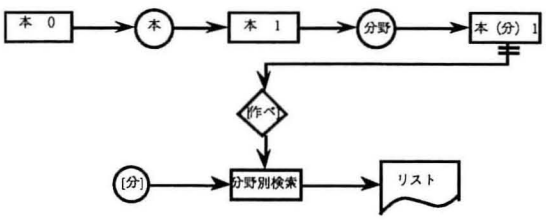
以上で、共通問題として提示された図書館問題に対する実現段階前の全ての段階を終了する。ここでは、これをVer.1と呼ぶことにする。

つぎに、Ver.2として通常の図書館において最低限必要であろうと思われる機能を付加してみた。

(1)検索処理の改善

1) 検索処理の改善の為、インデックス機能を果たすプロセスを追加する。
2) 検索リスト出力に貸し出し可否情報を入れる。
3) AND/OR条件を可能にし、適切な検索が行える様にする。

以下に、この改善についてのＳＳＤ図を示す。



このように、プロセス「本１」から分野ごとのプロセス「本（分）１」を新たに作成するだけで、システムの高速化がはかれる。

(2)貸出し日数の設定

新刊、人気本などの貸出し日数を通常のものより短く設定したり、特殊な本に対して貸出し禁止状態を設定できる様にする。

(3)予約

すでに、貸し出されている本の予約、予約者の待ち人数照会などを行えるようにする。

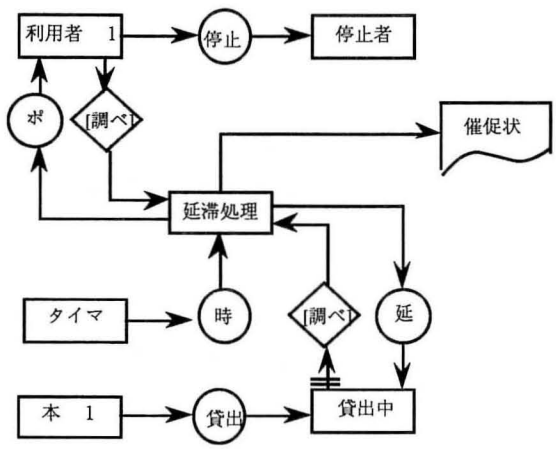
(4)購入希望図書の受け付け

利用者が希望する購入図書のリクエストを受け付ける様にする。

(5)返却延滞時の処理

1) 返却延滞を行なった利用者に対し、ペナルティ・ポイントを与える。
2) ペナルティ・ポイントが一定の期定数を越えた場合、利用禁止期間を与える。
ペナルティ・ポイントは、利用禁止期間経過後にクリアする（但し、全ての延滞本の返却を行なっている場合）。
3) ペナルティ・ポイントのカウントは一年(1月1日)毎にクリアする（但し、全ての延滞本の返却を行なっている場合）。
4) 初めての延滞のみ返却期限を一週間自動延長し、それ以後は、返却催促状を発行すると共にペナルティ・ポイントをカウントアップする。

以下に、この改善についてのＳＳＤ図を示す。

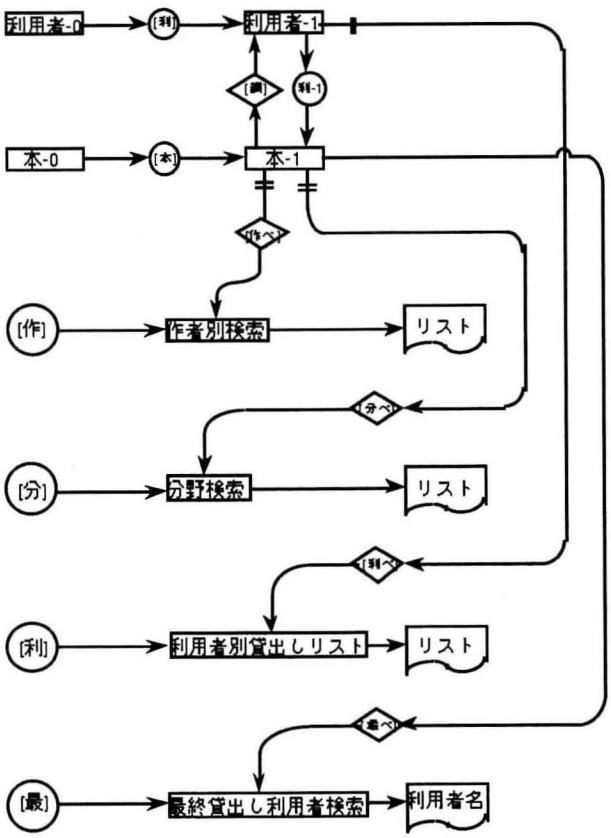


ここでは、貸出中の本に関する情報を迅速に得るため、新たに「貸出中」プロセスを作成する。延滞処理プロセスは、タイマから起動され、定期的に「貸出中」プロセスを調査し、状況に応じた処理を行なう。

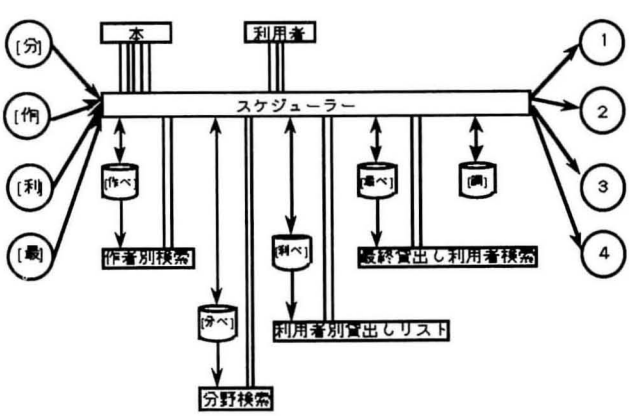
［R-4］設計書

ここでは、先の項で示されたＳＳＤ、ＳＩＤを最終的にまとめた形式で示す。

SSD（System Specification Diagram：システム仕様図）



SID（System Inplementation Diagram：システム実現図）



1) 特定の分野の本に関する問い合わせ結果リスト
2) 特定の作者の本に関する問い合わせ結果リスト
3) 特定の利用者が借りている本の問い合わせ結果リスト
4) 特定の本を最後に借りた人の問い合わせ結果リスト

［R-5］全体評価

1．使用した手法の全体的評価

［利点］

・オブジェクト指向の利点を持ちつつ、更にタイミングを記述できる。
・機能拡張が容易にできるようだ。
・無限個のプロセッサを仮定したことによって、複雑な問題でも無理なく記述することができると思われる。
・オブジェクト指向と異なり、「実体」の選び方の方針がある程度与えられている。
・機能にしばられず、システムを構築できる。

［難点］

・現在の通常の環境で実現するには、スケジューラ、コルーチンを考えなくてはならないという煩わしさがある。
・設計書がそのままドキュメントになりえないような気がする。
・余り普及していないようであり、手法を取得するのが難しい。
・状態ベクトル結合が瞬時に終了するとか、データ列結合が無限であるなどという非現実的な仮定がある。

2．各自の現場への適用は可能か？

以下の理由により、職場へ適用するのは難しいように思われる。

・スケジューラ、コルーチンを実現するのが容易でない。
・非現実的な仮定をいかにして実現するか？と言う問題点がある。
・職場のグループ内に浸透させるのに、かなりの労力と時間が必要になると思われる。

［R−6］グループ活動の感想

［江谷（@富士ゼロックス情報システム）曰く］

・年齢や業界経験値が似ていたので問題に対する視点のやり方にばらつきが少なく限られた時間内に方法論のノウハウを知るのには良かった。
・お互いの粘り強さが相乗効果となって最後まで（最終日前夜　朝５時まで）活動のまとめができた。
・はじめて出会った人たちがいかに自己開示をし、グループ活動を行うのかを観察できて面白かった。
・山崎氏のセミナー参加のおかげで、「ＪＳＤ法」の本にはでていないノウハウが判ってよかった。
・限られた時間で手法のノウハウを知るためには、実戦的ではないがグループの向かっている方向が失敗しないように誘導することが大切であると考えていた。（他の参加者の中には、山崎氏がいらっしゃらにと何もできなかった。とか、失敗しなかったので技法が身につかなかったという意見に対する反論）

「東宮（@ネクストファンデーション）曰く」

もちろん、一つのことを習得するにはグループなどではなく、一人で勉強し、不明点のみ詳しい人に聞くというのが最良の方法であると思われます。わざわざグループを組んでやるなどというのは、小学校だか中学校だかでやることであり、大人がやることではないと思ってます。今回は、既知の手法により問題について意見を戦わせるという以前に、手法自体が不明という状態でしたから、なおさらです。

とまぁ、冷静に考えるとこうなるのですが、実際の所は楽しかったのですから、そんなことはどうでもいいですね。欲を言えば、もう少し酒を飲んだりできればよかった。ちょっとストイックになり過ぎたかとも思います。

それと、このグループではリーダーを置かなかったのですが、やはり設定した方がいいですね。どうもグループの方向性が行きあたりばったりになったような気がする。リーダーというとみんな躊躇するから、これもアミダクジで決めればいいんじゃないのかな。

うーん、こんなもんです。ろくな感想がでなくてすいません。

「冨岡（@ジェームスエーシステムズ）曰く」

若手の会'９２@岡崎にＧ５のグループのメンバーとして参加したことの感想として，一番に山崎利治さんに感謝の気持ちを捧げたい．もし山崎さんのお力添えが無かったらＧ５の作業状況はどうなっていたのだろうか．おそらくは分析／設計の結果と言えるものは何一つ無いままに今回のワークショップを終えることとなっていたであろう．山崎さん，初歩の初歩から丁寧に教えていただきまして誠にありがとうございました．

他のグループで２日目の夜に徹夜をしていたグループがあったが，後から思うと我々も作業のピークをワークショップ会期の前寄りに持ってくるべきであったと思う．さらに，きちんとしたスケジュールを立てずに作業を行っていた点，ワークショップという場の有効利用としては失敗だったと悔やまれる．スケジュールを立てなかったために，ワークショップの作業以外のグループで楽しむ時間を充分につくれなかった．結局，食事と岡崎城に出掛けたぐらいであった．（私の参加したＧ５がシステムの動作を時間軸に沿って記述するＪＳＤ法のグループであったことを考えると運命の皮肉を感じるが）

しかし，最終日の深夜を越えた辺りからの我がＧ５の集中力は目を見張るものがあった．各人が誰ともなくやるべきことを自然に分担して作業し，それまでの２日間での作業以上のものが明け方前までの数時間で行われた．久しぶりに密度の濃い時間を過ごせて嬉しく思えたものである．前夜の夜通しの作業のため，最終日の全体討論ではＧ５のメンバーは，ほとんど思考力ゼロに近い状態であったため積極的な発言が無かった点が残念である．これも，会期中のスケジュールを立てなかったことの影響である．

脈絡無く書き並べたが，以上'９２若手の会グループ活動の感想とする．

「西岡（@さくらケーシーエス）曰く」

"グループで活動する場合に何が重要になってきますか?"と聞かれたとすれば、"グループ内のメンバ間に先ず共通の認識をはっきりさせて、そこからスタートし、その認識が崩れてばらばらにならないよう維持しながら、その認識そのものの内容を進めていくように気を使うことではないでしょうか"というような取り留めのない返事を私はすると思います。

しかし、共通認識そのものが絶対的に大事であると言っているのではありません。このグループ内の共通認識というものは、各メンバの個人的なものから 形成されることは間違いないでしょうが、その形成の仕方、それと個人の関わり方によっては全くグループで活動する意味がなくなってしまう場合もあると思えます。

と言うのは、いくらグループ内の共通認識が大事であると言ったとしても、それに各メンバが個人的な考えを適当(?)に合わせるような状態であれば、個性的なメンバ(?)が集まってグループを作っていること自身意味 ありません。グループ内のメンバは、合体る必要はなく、むしろ逆で、個人として存在していなければならないと思います。何に対して個人的な意見、考えがあるのかと言った、その"何"という部分にグループ内の共通認識があってこそ、自分とは違った個性を持ったメンバとグループを組んで活動するメリット、おもしろさが生まれてくると私は思っております。

今回の活動で、このような自分勝手な信念(?)をさらに自分勝手に深めてしまったとも言えそうです。

設計方法、設計手法といったもの対し、私は個人的にある偏見のようなイメージを、遥か昔の学生の頃から現在に至るまでずっと持ち続けていました。ここでまた取り留めのない話しになりますが、"認識する"ということに関して自分なりに思っているところがあり、まずそこから始めます。

同じ物、同じ現象を見ても人はそれぞれ違った印象を受け、違った反応をするような場面があると思います。その違いは、お腹が空いたので何か食べたいといった本能的な部分ではなく、何を食べたいかという意識的な部分に出てくるのではないでしょうか。意識的な所で違いが出てくる一つの理由として私は、それは個人ひとりひとりが意識の中に枠のようなものを持っていて、その構成が個人的に違うからだと思っています。

その枠は、個人が生まれた時からの、住んでいる地域、家庭、学校や個人が見てきたこと、教えてもらってきたことなどのような環境、それと同時にその人自身の持つ性質のようなものから形成されるものであると思っていますので、当然、このような枠を持つ人は個人的に異なったものになるということになります。(枠を持たない人はいないとも思ってます。)

設計手法ということに話しを戻すと、これは上述した意識上の枠と大した違いはないものと思ってしまいます。

簡単に言うと、設計を行なう際その考え方の枠組を与えているものでしかないようなイメージしか持てません。そのように思ってしまう為に、本当に設計と言う実際の作業にこのような枠組を作ってしまって良いのかどうか分からないでいました。確かに手順として簡潔に与えられていれば、それに従っていれば良く、作業もし易いかも知れませんが、でもそれは、結局一つの枠組の中に閉じた作業でしかないんじゃないかと言ったような疑問がずっと私にはありました。

今回私が参加したグループでは、ＪＳＤ法という手法で作業を行なっていたわけですが、その途中で幾度も山崎利治さんからアドバイスを頂いきました。もちろん、その内容はＪＳＤ法についての説明ではありましたが、そのアドバイスを伺う度に、私としては不思議な印象を受けました。

その印象とは、"何か、全然ＪＳＤ法に拘ってないなあ"と言う感じがしたということです。こういう言い方はおかしいかも知れませんが、一つの手法に拘ったからこそ、その方法を把握しているように思えるのに説明を伺っていてもそんな気配は全然感じられませんでした。

この印象は私にとって、とても有難かったものです。何事も自分の都合に合わせて解釈する傾向の強い私ではありますが、これで長年の疑問にも自分なりの考え方ができたように思えます。枠組として設計手法というものがあるという認識はあまり変わってはいませんが、その枠を付けることには意味があると今は思います。その枠はきちんと付けないといけないもので、その枠内での考え方もきちんと認識する必要はある。でも、その枠に拘る必要はないし、逆にその枠に捕らわれるようなものであればかなりの制限にもなり得る。　要は、その枠も使い方しだいといったようないいかげんな言いように近付いてしまったかのようにも思えますが、少なくとも設計手法というものに対する私個人のイメージは、150 度くらい方法転換したような気がしています。

最終日、各グループ最終発表の後の全体討論の時間は、私にはほとんどつまらなく思いました。私自身、積極的に自分の意見を出していたわけではないのであまり偉そうなことは言えませんが、あのような場で自分の意見を出し合うことが大事といっても、ただ人前で話しをする度胸付けのの練習のようなものであるなら、どこかの会社の新人研修で人通りの多い路上で歌をうたわせるような類とあまりかわらないという気がします。

田原～さんのようにパフォーマンス付きのではなくとも、時々ＰＣの方がされていた中途半端な形でなく、はっきり進行役を設定していた方がその場の討論会もおもしろくなっていたと思います。せっかくＰＣの方から次の新しい質問をされても、それに応えることなくまた前の話題を話し始るような討論が楽しいとは私には思えません。

今回のワークショップで私が参加したグループは、PC の渡邉さん、アドバイザの山崎さんに大変お世話になりました。また、グループ内の各メンバにも大変お世話になりました。

我がグループは、一緒に歌を唱うことも、酒の勢いで絡み合うこともありませんでしたが、終始一貫"ＪＳＤ法って何？"いうことで共通認識はほぼ完璧に確立されていたと思います。他のグループから見て我がグループがどういうふうに映っていたのか興味のあるところですが、我がグループはいたって平和でありました。

「水谷（@高丘製作所）曰く」

終わってみればあっという間で、この４日間短かったなという印象があります。思い起こしてみれば、一生懸命ジャクソン法を取得しようと躍起になっていただけで残念ながら他グループの方々とは余り交流できなかったと思います。しかし、自グループ内は１つの目的に向かって和気合々と進めることができたのではないでしょうか。討論でも同じ職種の仕事をしていてもこんなにも考え方が違うのかといい意味で驚くことばかりです。だから課題作成について朝５時まで討論したということができたと思います。（半面、他の班の進捗と比べ、仕方がなかったという意見もあるが・・・）　とにかく、自分としては珍しく真面目に参加していたと思います。不満を言えば、グループでもう少し遊ぶ機会（岡崎観光等）があってもよかったのではないでしょうか。

山崎さんには大変お世話になりました。山崎さんがいなかったら自分達の班はここまでできなかっただろうと思います。設計を進めていくうえで行き詰まった時、山崎さんがいらっしゃって教えてくれるという、まさに我が班のｍａｎコマンドである山崎さんに心よりお礼を言いたいと思います。

せっかくグループ内外のメンバーと知り合えたのだから今後いろんな面で交流を深めて行きたいと思います。とりあえず「ＳＥＡ秋のフォーラム」参加して、再びみんなと再開したいと思っています

[岩城弘二]

1.作業の進め方について
今回非常に反省している事は、きちんと予習をしてくるべきであったというあたりまえの事である。特に我々が選択したjsd法は難解であり、４日間で方法論をマスターできるようなものではなかった。しかも我々の中には予習をしてくるものはおらず、その為方法論的にグループを引っ張っていくことができる者が一人もいなかった。山崎氏の指導がなければ悲惨な結果になっていたのであろう。
だが、おそらく他のグループでも予習してくる者などはあまりいなかったのではないかと思われる。それにも関わらず、独自の方法論をあみだすなど、興味ある結果を出すグループも存在したわけである。考えてみれば、4日間でその方法論の提唱者が苦労して生み出したものを

マスターしようとした事自体無茶であった。
それよりも、今回のワークショップでは、設計方法論の難しさと重要性や問題点を認識すること。そして他の会社の人々と協力することにより、実際の業務上で生じる問題を解決するためのヒントや指針を得ることのきっかけを作るということのほうが意義があったのではないか感じている。

2.方法論に関して
私がJSD法を選択した理由の一つは、現在自分が使用しているオブジェクト指向とJSDを比較し、そしてJSDに有利な点が存在するなら、それをオブジェクト指向方法論に取り込むことはできないかということを探ることであった。しかし残念ながら、まだJSD法のほんの入り口に入ったににすぎず、評価できるまでには至っていない。
ワークショップを終わった時点での感触では、JSD法の利点は再利用性と拡張性にあると感じている。設計手順に沿って作業を進めていくと（きちんと理解して進めた訳ではないが）、核となる部分に後から自由に機能を加えることができるという事には感心した。しかし、再利用性や拡張性に関してはオブジェクト指向でも利点の一つとなっており、まだこの時点ではどちらが優れているかは評価は出来ない。
もしJSD法の利点をオブジェクト指向方法論に取り込むことができるとすれば、それはJSD法でのモジュールとなる実体などの取りだし方を、オブジェクト指向で言うクラスを抽出するうえでの指針として利用できる可能性があるという事。そしてオブジェクト指向ではあまり考えられていないプロセスという概念を取り入れるということである。この２点は、オブジェクト指向方法論を考えるうえでも参考となるものではないかと期待している。

グループ活動雑感

渡邉 雄一

1 はじめに

自分がメンバとして参加したのが 第2回のそれであるから, 随分と月日の経つのは早いものである。今回は JSD のグループを担当したが, 自分の力量不足もあって本来の PC の責務が果たせたとはとても思えない[†1]が, いろいろと感じることも無かったわけではないので, 10 年前の自分を思い起こしながら 感想を記す。

2 グループ 5 の特徴

このグループは, JSD をやってみたいという人と 他の手法を希望したにも関わらずグループ分けの都合上こちらに押し込められてしまった人が, 交じった形になった。

JSD をやりたいという 3 人は, いずれもオブジェクト指向についての知識も経験も持っている人達なので, 彼らは それらとの JSD との差を探る目的でこのグループに加わっていた。

他の 3 人はオブジェクト指向に関心はあったり, 知識としては基本的なことは持っていたにしても, 経験という点では十分とは言えないといった具合だった。だが, 全員 JSD は初体験という状況だった。また, 今回の構成の中でも比較的平均年齢が高かったグループだと思われた。そのせいか, 全般におとなしい。

手元に, 初日の一番最初の顔合せをしたときの自己紹介のときのメモを見ると

岩城さん OOx の仕事をしている

西岡さん

- OODB の仕事をしている
- 設計法を勉強してみたい
- 正当法でアプローチしたい

水谷さん

- PC98 を自宅で使用している
- C++ を買ったが 十分に使用していない
- JSD は本が厚い (ハードルが高い)

富岡さん

- 3 年目
- 工程管理をしている
- 仕様書からいきなりコーディングしていたので設計のプロセスを経験していない
- object 指向を勉強したかった
- 実務ではデータ中心の設計をしている

東宮さん

- Object 指向が第 1 希望
- C++を 2 年ほど使っていた
- JSD を選んだ理由はない
- 設計ということをまともにやったことがない
- 1 つでも良いからかじってみたい

とある。あれ, なんで**江谷さん**のが無い[†2]のだろう?。席を外していたのかな?。

3 なぜ JSD ?

JSD を担当したいと言った理由は, JSD はモデルの形成に力点を置いており, そのことの良し悪しを, 普段の自分のプログラミングという仕事と比較してどのように思うのかを討論するのが適当と思ったからである。その意味では JSD 以外に, モデルに拘る手法があればそれでも良かったのだが, JSD そのものに興味を持つ人が半分を占めたために, 事実上 'JSD 勉強会' となってしまった。メンバの主たる興味は JSD がどんな手法であるかを理解し, それがある程度出来た上で, 問題を解きながら不明な点を再確認し, 体得する[†3]というように見えた。勿論, 私が彼らの疑問に対して何かの答えを持ち合わせていることもなく, また ある意味で その必要は無いと思っていた。

私の方針では 兎も角, PC としてメンバと一緒になって JSD で図書館問題を一緒に解いて行くことが重要だと考えていた。解いて行く過程でどんなことを感じて何が大きなハードルなのかが解るのだけでもワークショップとしては意義があろうと思っていたからだ。だがメンバはことの本質が解らないのに それを進めるも何もあったものではないという風であった[†4]。が, JSD に関しては, 原典を訳された山崎さんに手取り足取り教えていただいて, 苦労することなく行間を知り得た [1]。手法そのものの理解は, まず実体とその行動の選定を何を基準にどのような箇所でやるかにかかっていて, 特に実体の選定は, オブジェクト指向におけるクラスと通ずる

株) アスキー

†1 毎回 同じ様な感想を書かねばならないのは本当に心苦しい

†2 岩城さんのが短いのも気がかりだなぁ

†3 それって贅沢過ぎない?

†4 その気持ちも解らなくはないけれどもね 8-)

ものがあることが体得出来た点は個人的には大きな収穫だった。もちろんそれは, 単に私一人の感想に留まらずグループメンバ全員も同様の気持ちであったであろうことは, 彼らの報告書 (以下 G5 レポートと略す) の中から容易にわかる。

4 JSD の壁

今回のワークショップの狙いの 1 つには, 他の手法との比較ということがあったが, 残念ながら, 実際にその作業を十分に行なうことは出来なかった。ここでは, 私が 1 番感じた CRC カードにおけるクラスの選定の勘所と JSD での実体の選定の違い[†5]について記しておきたい。

4.1 闇雲に実体を決めてみる

Budd の本[3] には『まずクラスを探せ!, クラスは名詞を探すと良い』と書かれている。私の場合は, JSD で**実体** (entity) を定義するのに, このクラスの概念に沿って作業を進めてみた。具体的には, 問題に記載された中から**名詞**を抜き出すと,

図書館, 本, 作者, 特定の分野の本に関するリスト, 利用者が借りている本のリスト, 図書館の職員, 図書館の利用者

などが容易に見つかる。そこでこれらをとりあえず**実体**と考えて闇雲に それに対する**行動** (action) を定義してみた。まず, **図書館の職員**という実体に対して行動を洗い出すと,

貸し出す : 本を利用者に貸し出す
　属性 : 題目, 作者, 分野, 利用者 ...
返却 (処理) する : 本を利用者から返却される
　属性 : 題目, 作者, 分野, 利用者 ...
本の登録 : 本を図書館に登録する
　属性 : 題目, 作者, 分野
本の削除 : 本を図書館から削除する
　属性 : 題目, 作者, 分野
リストの作成 (作者) : 特定の作者の本に関するリストの作成
　属性 : 作者, 題目 ...
リストの作成 (分野) : 特定の分野の本に関するリストの作成
　属性 : 分野, 作者, 題目 ...
リストの作成 (借りている本) : 特定の利用者が借りている本のリストの作成
　属性 : 利用者, 題目 ...
最後に借りた人の検索 : 特定の本を最後に借りた人の検索属性 : 本, 利用者

と羅列された。同様に**利用者**という実体に対して行動を洗い出すと,

借りる : 本を借りる
　属性 : 題目, 作者, 分野, ...
返却する : 本を返却する
　属性 : 題目, 作者, 分野, ...
リストの作成 (自分の借りている本) : 利用者自身が借りている本のリストの作成
　属性 : 利用者 (自分), 題目 ...

と羅列された。で, 取り敢えず実体の洗い出しをこの位で止めておいて次の**実体構造段階**に進んで ジャクソン構造図を**図書館の職員**に関して考えてみるが, なんともそれらしく書くことが出来ないのである。苦し紛れに 図.1 のようなものを書いてみるが, なんともし難い。具体的には, オブジェクト指向の場合ならば, それぞれの実体をクラスと思ってそれらの協調であることがなされると考えても良いだろうが, JSD の場合に 1 つの構造図に全体を収めるということは, クラスとは別のなんらかの階層を導入し概念を縮約する必要があるように (ぼんやりと) 思われた。

で, ここで山崎さんから頂いたアドバイスは「一生を考えなさい」であった。ここで図.2 を書いてみた。で, この絵を書きながら悩んだのは, 職員のしなければいけないことという観点[†6]で見ると, 貸出期間ということがオーバーラップしていて, それを図の上でどのように表現すべきか, そしてそれをモデルの中にどのように登場させるのか? といった要らぬ思いを巡らせていた。

4.2 そこで考えたこと

で, 悩んだのは, 問題に隠されたことがらをどの程度システムの記述に登場させたら良いのだろうか? ということである。Budd の本の 銀行のキャッシュディスペンサ (ATM) の例では, 顧客が持っているカードは無視され 操作のためのディスプレイがクラスに登場していた。そのアナロジーからすると, この段階で**図書館利用カード**とか**貸出し券**とかを登場させた方が良いのかと考えたわけである[†7]。

ところが結果的には, 問題に直接記載されていないが Budd のそれのアナロジーとは まったく違ったものが登場してくる。それは, **利用者の登録/削除**といったことだ。
もちろん, これは図書館利用カードの発行/抹消ということに置き換えても良いのかもしれない。が, そんな具体的なこ

†5 というか それを考えている際に思い巡らせたことなど

†6 そもそもそれが良くない。ところがこの絵は不正確だが, 太線の構造だけを見ると本の一生の Jackson 構造図になっているから不思議だ!

†7 こだわりだせば切りがない. 本の整理/識別の為には 本 ID を考えなければならないのだろうか? といったことまで思いは巡る!!!

とにとらわれることなく, 問題の本質を捉える抽象度の高いモデリングの例を目の当たりしたことに, 驚嘆していた。
特に現実にモデリングをしない (データベースの構造にデータ項目を直接反映させながら) システムの設計などをしていると, そのようなもの (例えば 図書館利用カード) の存在を見落とすと, 結果としてシステムの再構築に匹敵するような改造をすることとなる。
だが 多少脱線するようだが, 良くよく考えてみると, それを現実世界で直接表現するためには, 何からの仕掛けが必要になる。そして そのメカニズムを実現するための介在物を登場させることにより, 問題が自然と複雑化してゆく。それをユーザが納得出来る程度まで具体的にモデリングしないと本当の意味でのプロの設計とは言えないような気がする。例えば その介在物として 図書館利用カード を取り扱うこととしたときに, その 発行/一時停止/紛失届け/再発行/抹消などが考えられる†8からである。

4.3 実体にふさわしいものとは

少し話しが横道にそれたが, 上に述べたような初心者の葛藤は, G5 レポートには, そのことはごく簡単にしか書かれていない。が, JSD にふさわしい実体を 如何に的確に定義し それが持つアクションを洗い出すか?という 1 番最初の壁を越えられるか否かが, JSD を習得できるか否かの大きな別れ目といってもよさそうだと感じた。そしてその勘所 (着眼点)†9は G5 レポートに記載されている通りである。

また, 不勉強でこのワークショップの後で読んだ山崎さんの [2] にその勘所を氏の明解な筆で説かれている†10。

5 おわりにかえて

最近, 自分の歳を感じる場面が少なくない†11。特に今回感じたのは,

- **ちゃんとした技術資料を丹念に読む気力が失せている**
  まあ疲れていたせいもあるのだろうが, 最近のプログラマの欠点として手元に処理系や環境があるものは, 実際に試すことで体得して知識として吸収するが, 抽象的な記述を熟読してその本質を知ろうとすることがおざなりになっていると反省している
- **試行錯誤を躊躇してしまう**
  単に気力が失せていたからかもしれない
- **酒に弱くなった**
  もともと強くはないが, 本当に最近は弱くなった。特に日本酒は深酔いするなぁ～

ってなことかなぁ。

だがある意味では, 今回のグループは やや若さがそがれているのかなっと思われる場面も少なくなかった。みんなと酒浸りになるような状況†12 を作り出せなかったことは陳謝するにしても, CSP を勉強してきて皆を論破してやろうとか, 自分の抱えている仕事/アプリケーションに沿った内容に話/議論を発展させて, その延長にグループのテーマを持って行こうとするような人はいなかった†13。

ちょっと前なら, そんなことを残念に思う気持ちもあった†14けれど, もう自分の残りの人生で出来る仕事の大きさを計れる歳になってきた私にとっては, 他人のことを構って要られるほど暢気にはしていられない。

10 年前を振り返って, 自分がメンバとして参加したときは, 良くも悪くも技術的な成果を求めずに ある種の精神論を体得にしたに過ぎなかった。それは ワークショップというそのものを体験したことがそうであった。だから, 今回参加してくれた人達が, 後日なんからの形で振り返って貰えるものがあったならそれで良いと思うのは甘いだろうか?

参考文献

[1] M. Jackson (大野, 山崎 [監訳]) : システム開発 JSD 法, 共立出版, 1989
[2] 山崎 利治 : マイケルジャクソンのシステム開発法, JSD 法の実際 (1), bit, Vol.22, No.2,(pp.216-222) Feb. 1990
[3] Timothy Budd, An Introduction to Object-Oriented Programming, Addison-Wesley Publishing Company,Reading,MA,1991.

†8 このような検討は まだ十分に行なえていない. しばらくは自分自身で葛藤する必要があるのだろう
†9 重複するかもしれないが, 手元にメモしてあるのは,「行動 (action) から実体 (entity) を探す」,「入れ物は実体 (entity) にはならない」,「サービス機能は後回し」,「物 (の集合) ではなくてコピー (1 つの物) を考えよ」
†10 但し, これを読んだからと言ってすぐに Jackson 法が体得出来るわけでは無い!. 今回 岡崎で自分が体験したような試行錯誤を経ずして, それを掴むことは至難の技のように思える.
†11 世間で言うところのプログラマ定年説なのか?
†12 それこそ悲惨な状況かな?
†13 オブジェクト指向と JSD とを比較することが最初から目的の人がいたのだから, 自分たちのやりたいようにやっていたのかもしれない。そうだとしたらとってもシタタカ!
†14 今でも全く無い訳では無いが

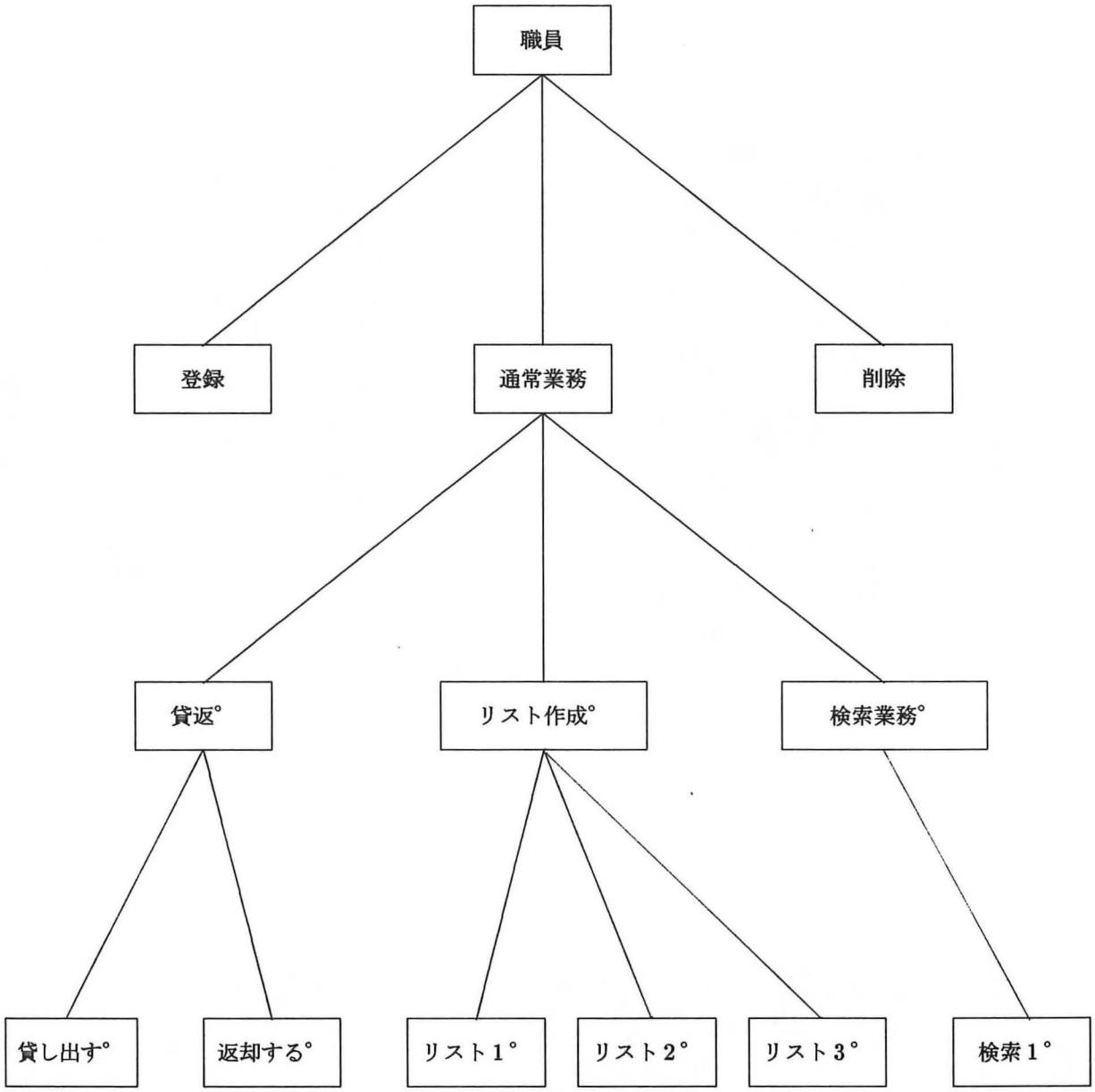


図 1. 闇雲に作った構造図

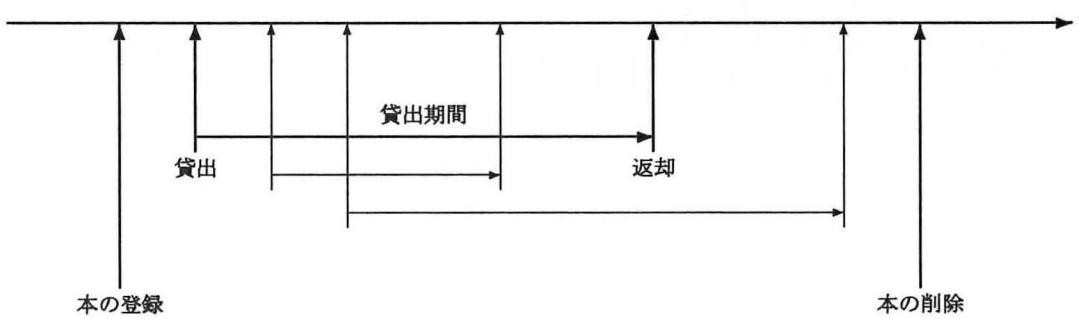
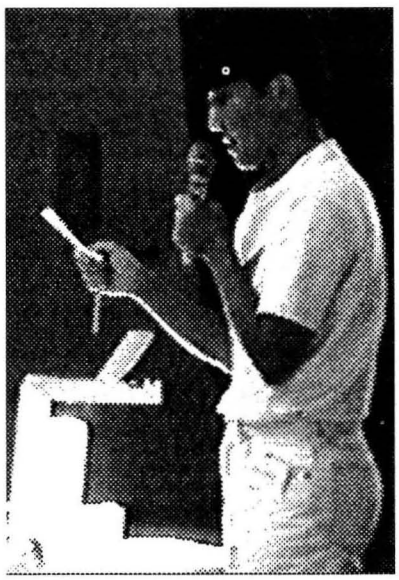
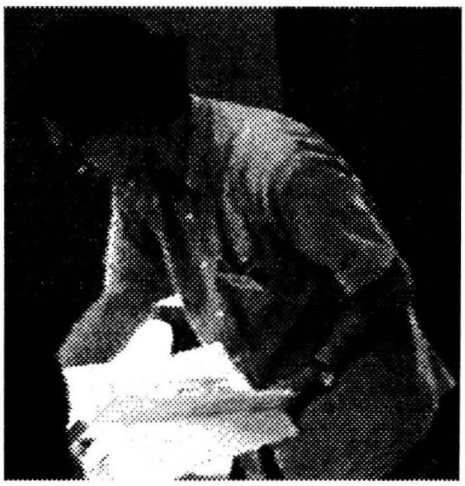
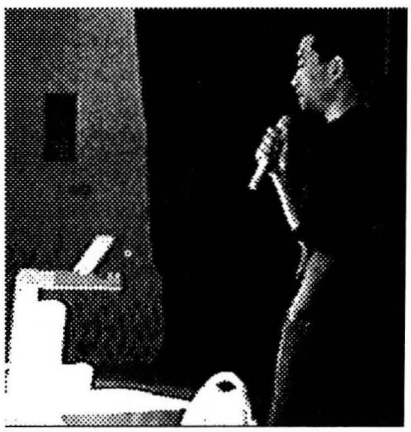


図 2. 職員の一生

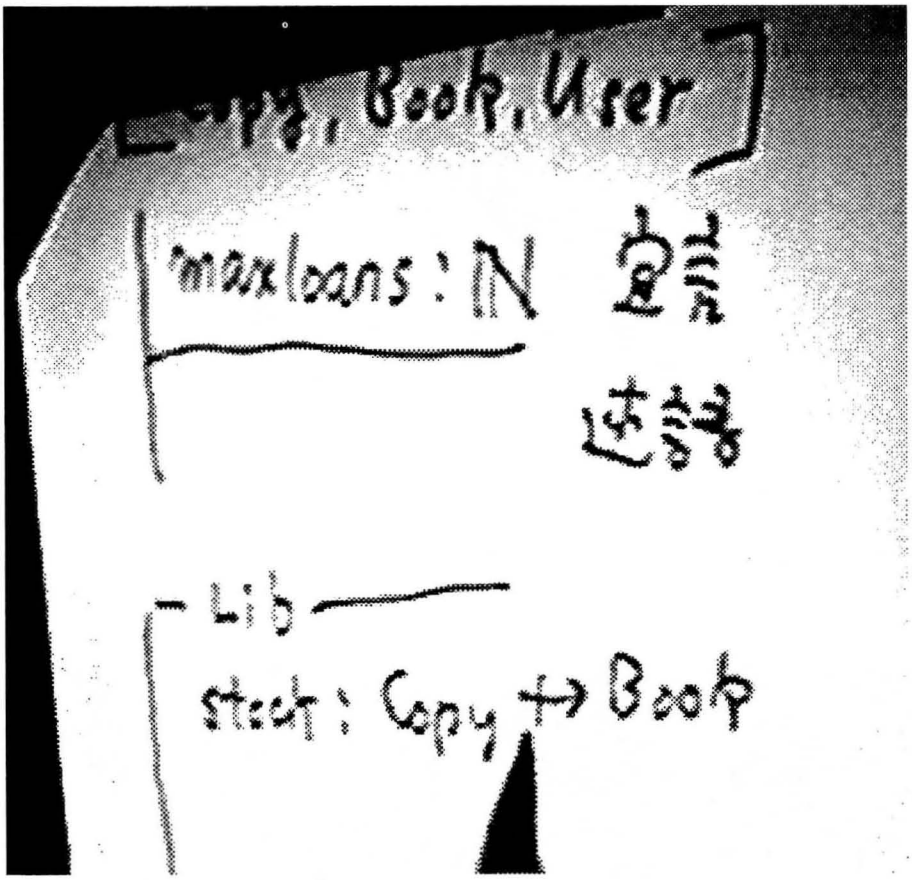






4. 講演記録

- オブジェクト指向分析/設計概観 (塩谷 和範) 155
- 形式的ソフトウェア開発 (山崎 利治) 170



オブジェクト指向分析/設計概観

(株) SRA
ソフトウェア工学研究所

塩谷　和範
shioya@sra.co.jp

[目次]

オブジェクト指向の考え方

オブジェクト指向概観

オブジェクト指向分析

オブジェクト指向設計

まとめ

1. オブジェクト指向の考え方

1.1.　ソフトウェア危機[*1]

ソフトウェアは、

ますます巨大化複雑化する　-> 巨大プログラミング
継続的な進化が要求される　-> 拡張性再利用性
柔軟さと堅固さが要求される　-> 高信頼・互換性
ハードウェアより長く使われる　-> 独立性・携帯性

しかしながら、

ソフトウェアは手早く簡単には作れない (人に依存)

それゆえに、

もっと沢山のソフトウェア技術者が必要である

更に、

既存ソフトウェアの再利用ができ、
信頼性の高い、堅固な、
ソフトウェアを開発する為の手法が
継続的進化を実現する為に必要である。

---

*1 現在は不況の為、"沢山のソフトウェア技術者"が余っている状況であるが、本質的に優秀な技術者が必要であるという事実は変わらない。これからは、ユーザ自身がソフトを作成することになるであろうし、ユーザからの要求も厳しいものになるだろう。これに対処する為には、新しいパラダイムが必要となる。オブジェクト指向はその一つである。

1.2. より良いソフトウェアを作成するには？

二つの重要な概念抽象化とカプセル化。

(1) 抽象化 (Abstraction)

問題を単純化し、
モデル化(一般化)し易く、
理解し易く、
分解を助ける。

(2) カプセル化 (Encapsulation)

複雑さを管理し抽象化を促進する。

1.3 抽象化の変遷

(1) 伝統的な見方:

Data + Algorithm = Program [Wirth]

(2) 構造化分析/設計 (SA/SD)

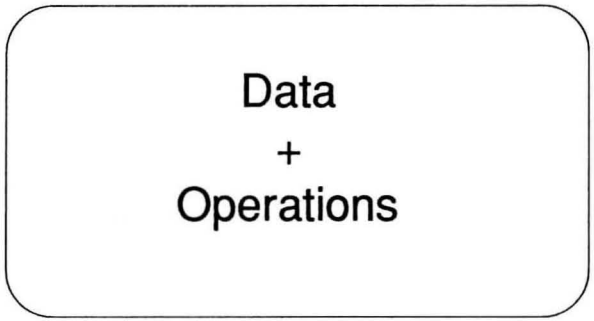
・機能に着目し構造化する

- 制御構造の抽象化
- Module (手続きのモジュール化)

(3) 抽象データ型設計 Abstract Data Type (ADT) Design

・あるデータはどの型が包含するべきか?

- データの抽象化 (Data Abstraction)
- データとアルゴリズムのカプセル化 (Encapsulation)
- 情報隠蔽 (Information Hiding)



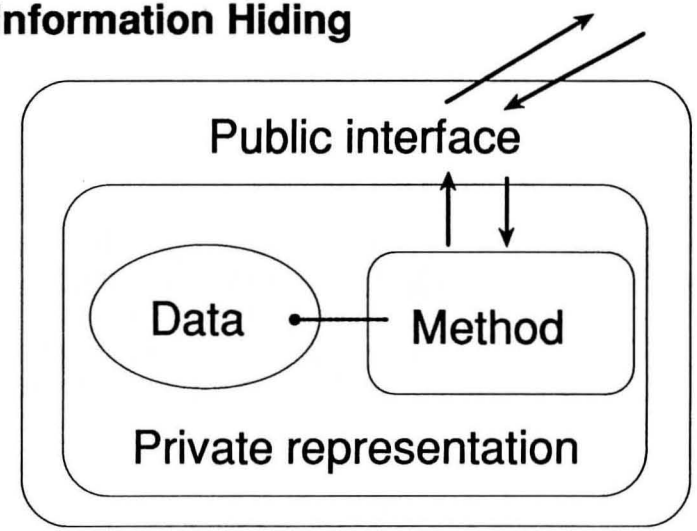
(4) オブジェクト指向設計

・何がオブジェクトか?

(記憶領域 + 処理装置) + 通信装置 => Object

[所真理雄]

Information Hiding



- データとアルゴリズム(Methods)の抽象化
- データとアルゴリズム(Methods)のカプセル化
- 情報隠蔽 (Information Hiding)

--

オブジェクトはインタフェースを通してのみ操作可能

Message Sending (message送信による結合)

Message Passing

1.4 "Obejct-Oriented" Key Concepts:

HP研究所による定義 [Sny89],[SHO89]

Object は抽象化を実現する
Object はClientsにサービスを提供する
Clients はオブジェクトからのサービスを要求する
Objects はカプセル化されている
Methods は対応するデータを操作する為にある
Methods は要求(Request)に基づき実行される
Request は総称的である (Polymorphism)
Object はその実現を共有することが出来る (Class)
複数のobjects(instances) は実現 (classes)を共有する
Inheritance(継承)は実現を組織化する為の機構である

2. オブジェクト指向概観

SA/SDとオブジェクト指向言語との不整合
(Grand Canyons -- Peter Coad : OOPSLA'91)
一貫したソフトウェア開発のライフサイクルが必要。

OOA --> OOD --> OOP(OOI) --> OOT

OOA: OO Analysis
OOD: OO Design
OOP: OO Programming
OOI: OO Implementation
OOT: OO Testing

OOA:	
Shlaer and Mellor	[SM88]
Coad/Yordon	[CY90]
OMT:Object Modeling Technique	[RBP91]
ObjectOry	[Jac87],[JL91]
OORASS	[RN89]
OOD:	
Booch	[Boo91]
Coad/Yordon	[CY91]
CRH(CRC):Class,Responsibility,Collaboration	[BC89]
Responsibility Driven Design	[WW89][WWW90]
Demeter System	[LBS91]

3. オブジェクト指向分析: Object-Oriented Analysis

Shlaer and Mellor オブジェクト指向システム分析
- Sally Shlaer & Steve Mellor

Object-Oriented Systems Analysis:
- データ中心による世界のモデル化

Models & 手順：

(1) 情報モデル (Information Models) (Data)
概念上の存在物(entities)と属性つまり
データと、それらの関係(relations)の認識
・ER(A)モデル[Che76]による分析

(2) 状態モデル (State Models) (STD)
オブジェクト(Data)と関係の間にある
動的な振る舞い方の分析
・オブジェクト通信モデル
・状態遷移図(状態、イベント、アクション)

(3) プロセスモデル (Process Models) (DFD)
各状態についてアクションプロセスを
DFD で記述する。
・Data Flow分析

特徴：
構造化手法の流れを組むデータ中心分析手法
各段階の役割が明確、分析不十分の時は前段に戻る。

Tool Support:	
TeamWork	- CADRE
ObjectMaker	- Mark V System, Ltd.

Coad/Yordon法 - Peter Coad (Object International, Inc.)

OOA (Object-Oriented Analysis)

手順：
(1) クラスとオブジェクトの認識
(2) 構造の認識
　一般化 - 特種化構造 (is-a関係)
　全体 - 部分構造 (has-a関係)
(3) Subjects (オブジェクトの組織化)の認識
(4) 属性の定義・配置
　インスタンス接続の認識
(5) サービスの定義
　オブジェクトの状態の認識、状態遷移図作成
　サービスの抽出、メッセージ接続の認識
　サービスの記述、サービスチャート作成

特徴：
記法が簡単
分析・設計のガイドラインを持つ
言語に依存しない。

Tools Support:
OOATool
　- Object International, Inc.
ObjectCast, ObjectView
　- 富士ゼロックス情報システム（株）
ObjectMaker
　- Mark V System, Ltd.

OMT: Object Modeling Technique
- J.Rumbaugh (GE) [RBP91]

Models:
オブジェクトモデル (Object Model)
　オブジェクトとクラスの静的構造
動的モデル (Dynamic Model (STD))
　オブジェクトの状態と振る舞い
機能モデル (Function Model (DFD))
　機能分解

手順：
(1) 分析とモデル化
　・問題の記述を要求者と供に行なう。
　・何をしなければいけないかを分析しモデル化
(2) システム設計
　システムの構造全般にわたる決定。
　・サブシステム化
(3) オブジェクト設計
　実装方法を含む詳細化
　・データ構造とアルゴリズムの決定
(4) 実現
　・言語実現/ DB-SQL/ 非コンピュータ利用の決定
　・指針に従い言語毎に展開する。

特徴：

・要求定義／モデル化の重要性を強調。
・SA/SDから移行し易い。
・モデルが比較的単純。
・言語独立。
・分析／設計が一貫して行なえる。
・適用例を含む解説本が出版されている。 [RBP91]

Tools Support:

OMT Tool - GE Labo.
ObjectCast - 富士ゼロックス情報システム（株）

ObjectOry
- Ivan Jacobsen (Objective Systems, Sweden)

ソフトウェア工場の概念を開発プロセスに導入し、
分析から設計、テストまで工業的に行なう事が目的。
更に、既存システムをre-engineeringして
OOアーキテクチャに変更する方法を提案。

Re-Engineering =
Reverse engineering
+ delta
+ Forward engineering - [JL91]
(delta: Changes of the system.)

Models:
Use model
Use Caseと呼ぶユーザとシステムとが交わす
トランザクションの列の定義、エンティティ、
外部インタフェース定義を含む状態遷移図類
似のオブジェクト分析モデル
Use Case = Responsibility

手順：

分析過程
(1) 既存システムの改善対象部分(Block)を識別する。
(2) Blockに相当する分析モデル (Use model) の定義
a) Actorの識別
b) Use Case(使用状況)の識別
c) Entityの識別
d) InterfaceObjectの識別
e) ActiveObjectの識別
f) FunctionObjectの識別

g) 各オブジェクトの仕様化

設計過程
(3) 各Use modelを既存部分の実現部と対応させる
(4) 交換部と残りの部分のインタフェースを評価する
(5) 新サブシステム及び既存部との間のインタフェースを設計する
Componentと呼ぶソフトウェア部品を利用。
(6) 既存システムに新サブシステムへのインタフェースを付加する

テスト過程
(7) テスト計画、テスト識別、詳細化
(8) テスト実行
各ブロックのテスト
各UseCaseのテスト
システム全体のテスト

特徴：

大規模システムの段階的リ・エンジニアリング
ソフトウェア工場の概念を開発プロセスに導入
ObjectOry手法とツール及びサポートをライセンス。

Tools Support:

OrySE (ObjectOry Support Environment)
- Objective System

OORASS: Object-Oriented Role Analysis , Synthesis and Structuring
- Trygve Reenskaug (TASKON Work Environment)

組織化されたオブジェクトの協調構造の作成支援

Role (役割):
なぜ (Why) そのオブジェクトが必要か?
Type (型):
何が (What) そのオブジェクトの外的性質か?
Class:
どのように (How) 実現するか?

Models:
役割モデル (Role model) 図
オブジェクトの振る舞いのカプセル化

手順：
(1) 役割(Role)分析による役割モデルの記述

対象エリア (Domain) の識別
トリガと応答 (Stimulus - Response) の識別
オブジェクトの認識
オブジェクト相互作用シナリオ作成
(CRC法の利用: クラス部は役割として記述)
役割図 (Role diagram) 作成
メッセージ契約 (Message Contracts) の決定

(2) 合成(Synthesis)による複合オブジェクトの定義

オブジェクトは、複数の役割を持つ
このような複合役割 (Composite Role) を合成する

協調構造 (Collaborator structure)
　Role Model
内部構造 (Internal structure)
　オブジェクト階層 (Hierarchy)
継承構造 (Inheritance structure)
　一般 - 特殊化、合成(Synthesis)

(3) 構造化(Structuring)によるオブジェクト協調の規定

オブジェクトは、複数の役割を持つ。
これら複数のRole Modelの役割記述から
オブジェクト仕様 (Object Specification) を合成

(4) クラスの実現 (Class Implementation)

外部からの視点:
　役割 (Role) と契約 (Contract) により、
　メッセージと協調者(Collaborator)を規定
内部からの視点:
　クラスはオブジェクトの内部構造と振る舞い
　を規定する

ObjectProviders :
　契約の指定によりオブジェクトを作成
　ObjectMatchers :
　契約に合致するオブジェクトを見つけだす。
Semantic Types :
　Metamodelを作成する。
Metamodel :
　一般化した再構成可能なオブジェクト構造
　の記述。アプリケーションに特有な型定義。

特徴：

Role (役割)概念の導入
(一つのオブジェクトは複数の役割を持つ)
複合役割の分析と構造化による役割合成。
クラス実現の自動化。ObjectProvider, Metamodel他

Tools Support:

CASEツール、toolmaker, modulemaker (開発中)
　- Senter for Industriforskning, Oslo, Norway.

4. オブジェクト指向設計 OOD: Object-Oriented Design

Booch法 - Grady Booch (Rational)

Model:
静的な意味分析 (Static semantics)

論理的視点 (Logical view)
クラス構造図 (kind-of(is-a)階層)
オブジェクト構造図 (part-of階層)
物理的視点 (Physical view)
モジュール構成図
プロセス構成図
動的な意味分析 (Dynamic semantics)
状態遷移図
タイミング図

手順:
(1) クラスとオブジェクトの認識
Shlaer/Mellor, Coad法と同様の観点で
対象領域の語彙から抽出
(2) クラスとオブジェクトの意味付け
(3) クラスとオブジェクトの関係付け
(4) クラスとオブジェクトの実現

特徴:
言語を特定しないがAda, C++, Object Pascalに重点。
設計偏重、実現言語によっては詳細に過ぎる記述。

Tools Support:
Rational Rose - Rational
ObjectMaker - Mark V System, Ltd.

CRC(CRH) : Class, Responsibility, Collaboration
- Kent Beck, W.Cunningham

"CardCASE" : OO brain storming - Peter Coad

オブジェクトを、
・クラス (Class)
・責任 (Responsibility)
・協力 (Collaboration)
の3要素に分けて一枚のカードに記述し分析する

手順：
(1) クラスの識別と命名
(2) クラスの責任の記述
(3) 責任を果たす上で必要な協力
(他のオブジェクト名)を記述
(4) カードを広げて協力関係を検討する事により、
責任の分割・集約、クラスの分割・集約など
再構成/再定義を行なう。古いものは捨てる
(5) 幾つかのクラスをまとめて上位クラスを定義する
(6) 全体を見回して、責任の重複・欠落・未参照が
無くクラス構造が単純化されていれば終了

特徴：
簡単・明瞭な手法。どこでも簡単に行なえる。
最も安価なツールであるインデクスカードを使用。
言語非依存。
現在のところCASEツール支援無し。再利用が難しい

Tools Support:
CASEツール無し。CRC Card のみ使用。

Responsibility Driven Design
- R.Wirfs-Brock, B.Wilkerson [WW89]

図形表現を用いたCRC手法に基づく手法。

手順：
要求定義段階

クラス (Class) の識別
責任 (Responsibilities) の識別
クラスの外部に対しての振る舞い
協調関係 (Collaborations) の識別
他のクラスに依頼する要求

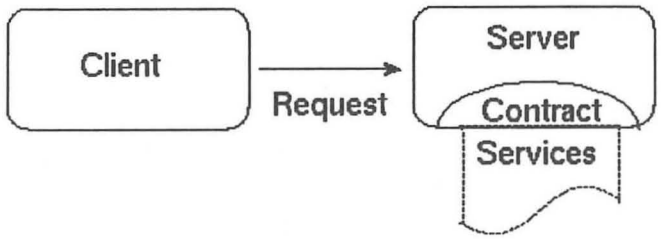
分析段階 (予備設計)

構造の分析
契約 (Contract)によるClient - Server関係
契約はクライアントが依存する責任の結合体

サブシステムの分析
サブシステムは、クラスのグループ化。
(役割によるグループ化)
サブシステムは概念的存在

プロトコルの定義

Client-Server Contract



Client
クライアントはサーバーにサービスを要求する。

Server
サーバーは、クライアントに対しサービスを提供する。

Contract
契約は、サーバーが提供するサービスの一覧

Resopnsibility Driven Design

Designing Object-Oriented Softwareからの引用

特徴：

・Client-Serverモデルによる"契約 (Contract)"という
責任の概念 (Responsibility) を追加
・サブシステムの概念

Tools Support:
?

Demeter System
- Karl Lieberherr at Northeastern University.

Models:
Demeterの律法 (Law of Demeter) [LHR88],[LR88]
クラス構造と変数の使用法を制限することにより、
クラスの内部構造に依存しないコードを作成する。

・他のオブジェクトの一部を参照したり、
そのオブジェクトの命令を直接実行してはいけない。
替わりに、その命令の実行を依頼する。

手順：
・クラス辞書（クラス構造の定義）を中心に、
グラフィカルなクラスエディタでクラス記述を行なう
・Demeterの律法に基づき分析・再構成を行なう
・コード作成／テスト

特徴：

Demeterの律法により、
良いプログラミングスタイルのソースコードを
作成する為のCASEツール

・テスト環境を備える
・作成するソースコードは、コンパクトで読みやすい
・クラス辞書は言語に依存しないが、クラスモジュー
ル中の Method定義は特定の言語で記述されている
・現在Flavors, C++をサポート

Tools Support:
Demeter System - Northeastern University

まとめ

まとめ

オブジェクト指向は、

・現実世界をモデル化しシステム化するための有効
・特にシミュレーション向き / プロトタイピング向き。

だが、

・OOPL（言語）によって効果が上がるわけではない。
・考え方（モデル化能力）が大事、
　言語は二次的。非OOPLでも実現可能 [RBP91]。
・再利用は簡単ではない。（良い設計が必須）

しかしながら

・良い方法論と言語を組み合わせると効果的。
・良い開発・デバグ環境が必須。
・プロジェクトの特性に応じた最適なOOツールを選定
　すれば効果が上がる。

最後に、

・オブジェクト指向(OOA/OOD/OOI/OOT) は、
　ソフトウェア危機への打開策になる。
・OO手法を導入する前に
　先ずオブジェクト指向の概念を理解する事が必須
・最初は小さな試行プロジェクトに適用し経験者を
　育て、次のプロジェクトのリーダとする
・一貫したオブジェクト指向開発を支援する
　統合OO-CASEツールが必要。
　（センスがない人にも専門家にも）

[参考文献]

[BC89] Kent Beck and Ward Cunningham.
A laboratory for teaching object-oriented thinking.
In OOPSLA'89 Proceedings, pages 1-6, Oct 1989

[Boo91] Grady Booch.
Object Oriented Design with Applications.
Benjamin/Cummings, Redwood City, California, 1991

[Bud92] Timothy Budd. -- CRC法による分析と各種OOLによるコーディング例
An Introduction to Object-Oriented Programming.
Addison Welsley, 1992. ISBN 0-201-54709-0

[Che76] P.P.Chen.
The Entity-Relationship MOdel - Toward a Unified View of Data.
ACM Transaction on Database Systems. Vol. 1, No.1, March '76

[Cox86] Brad J. Cox. -- OO概念とObjective-CによるOOプログラミング解説
Object Oriented Programming.
Addison Wesley, 1986.

[Cox86J] Brad J. Cox. 前川守 監訳
オブジェクト指向のプログラミング　ソフトウェア開発技法の進化
トッパン、1988.

[PW91] Lewis J.Pinson, Richard S.Wiener.
Objective-C Object-Oriented Programming Techniques.
Addison Wesley, 1986.

[CY90] Peter Coad and Edward Youdon.
Object-Oriented Analysis (second edition).
Youdon Press, 1990,1991.

[CY91] Peter Coad and Edward Youdon.
Object-Oriented Design.
Pentice Hall, 1991.

[GR83] A. Goldberg and D.Robson.
Smalltalk-80: The Langauge and its implementation.
Addison Wesley, 1983.

[Jac87] Ivar Jacobsen.
Object-Oriented Development in an Industrial Environment.
In OOPSLA'87 Proceedings, pages 183-191, Oct 1987.

[Jac92] Ivar Jacobsen.
Object-Oriented Software Engineering.
ACM Press, Addison Wesley, 1992

[JL91] Ivar Jacobsen, Fredrik Lindstrom.
Re-engineering of old systems to an Object-Oriented architecture.
In OOPSLA'91 Proceedings, pages 340-350, Oct 1991.

[LM91] Jo A.Lawless, Molly M.Miller.
Understanding Common LISP Object System.
Prentice Hall, 1991.

[LP90] Wilf R.Lalonde, John R.Pugh.
Inside Smalltalk Volume I, II.
Prentice-Hall, 1990, 1991.

[LBS91] Karl J. Lieberherr, Paul Bergstein, and Ignacio Silva-Lepe.
From Objects to classes: algorithms for optimal object-oriented design.
Software Engineering Journal, 1991. (Accepted for publication)

[LHR88] Karl J. Lieberherr, I.Holland, A.J.Riel.
Object-Oriented programming: an objective sense of style.\
In OOPSLA,88 proceedings, 1988.

[LR88] Karl J. Lieberherr, A.J.Riel.
a CASE study of software growth through parameterized classes.
J. of Object-Oriented Program, 1,3 (August/September 1988).

[Mey88] Bertrand Meyer.
Object-Oriented Software Construction.
Prentice Hall, 1988.

[Mey88J] Bertrand Meyer著、二木厚吉監訳、酒匂寛、酒匂順子訳
オブジェクト指向入門
ASCII出版、1990

[Mey92] Bertrand Meyer. - Eiffel version 3
Eiffel: The Language.
Prentice Hall, 1992.

[RBP91] J.Rumbaugh, M.Blaha, W.Premerlani, F.Eddy, W.Lorensen.
Object-Oriented Modeling and Design.
Prentice Hall, 1991.

[RBP91J] J.Rumbaugh他、羽生田栄一監訳
オブジェクト指向方法論OMT モデル化と設計
プレンティスホール・トッパン、1992.

[RN89] T.Reenskaug, E.Nordhagen.
The Description of Complex Object-Oriented Systems: Version 1.
Senter for Industriforskning, Oslo, Norway, 1989.

[RS89] T.Reenskaug, A.L.Skaar.
An Environment for Literate Smalltalk Programming.
In Proceedings of OOPSLA'89, Oct 1989.

[Rum87] James Rumbaugh.
Relations as Semantic Constructs in an Object-Oriented Language.
In OOPSLA'87 Proceedings, Oct 1987.

[Sav90] Dusko Savic.
Object-Oriented Programming with Smalltalk-V.
Prentice-Hall, 1990.

[SM88] Sally Shlaer and Steve Mellor.
Object-Oriented Systems Analysis.
Prentice Hall, 1988.

[Sny89] A.Snyder.
The essense of objects. Rep. STL-89-25.
Software Technology Laboratory, Hewlett-Packard Laboratory.

[SHO89] A.Snyder, W.Hill, W.Olthoss.
A glossary of common object-oriented terminology. Rep. STL-89-26.
Software Technology Laboratory, Hewlett-Packard Laboratory.

[Str86,89] B.Stroustrup.
The C++ Programming Language. Second Edition
Addison Wesley, 1986,1991.

[WJ90] R.Wirfs-Brock, R.Johnson.
Surveying Current Research in Object-Oriented Design.
CACM Sept 1990.

[WW89] R.Wirfs-Brock, B.Wilkerson.
Object-Oriented Design: A Responsibility-Driven Approach.
In OOPSLA'89 Proceedings, 1989.

[WWW90] R.Wirfs-Brock, B.Wilkerson, L.Wiener.
Designing Object-Oriented Software.
Prentice Hall, 1990.

[青木91] 青木淳
Smalltalkソフトウェア開発(1) - (12)
SuperAscii 1991/8月号 - 1992/5月号

[駒井92] 駒井孝彰　さくらソフトウェアサービス（株）
オブジェクト指向の事務処理分野への摘要
InterAI Feb./92

[OO-CASE製品]

CDE　C++開発環境　統合型OOSD CASEツール /X11
- IDE(Interactive Dvelopment Environment)
  595 Market Street, 10th Floor, San Francisco, CA 94105 USA
  TEL:+1-415-543-0900, Fax:+1-415-543-0145, E-mail:sales@ide.com
- （株）SRA CASEマーケティング部
  TEL: 03-3234-2623, Fax:03-3234-8048

Rational ROSE:　Boch法を支援する統合型CASEツール /X11
- Rational.
  3320 Scott Boulevard, Santa Clara, CA 95054-3197, U.S.A.
  phone: +1-408-496-3700

TurboCASE 4.0:　汎用CASEツール /Mac
デモ版（MAC用バイナリ）が下記住所から$15で入手できる。
- StructSoft, Inc.
  5416 156th Ave.S.E., Bellevue, WA 98006, U.S.A.
  Fax: +1-206-644-7714, phone: +1-206-644-9834

Teamwork: OOA,OOD 統合型CASEツール, Ada Code Generator /X11
- Cadre Technologies Inc.
  Teamwork Division
  222 Richmond Street, Providence, RI 02903 USA
  TEL: +1-401-351-CASE
- 横河ヒューレットパッカード（株）TEL: 03-331-6111

OOATool: OOA Tool /Mac, (Code Generator開発中)
- Object International, Inc.
  9430 Research Blvd., IV-400, Austin, TX 78759-6535
  TEL: 1-800-926-9306, +1-512-343-4549, Fax: +1-512-343-4569

ObjectCast: Objectworks\Smalltalk用 OO分析/設計支援環境
ObjectView: Objectworks\Smalltalk用 OO分析支援環境
- 富士ゼロックス情報システム（株）
  〒160 新宿区西新宿3-16-6 水野ビル TEL: 03-3378-8011

ObjectMaker: OOA,OODツール /C, C++, Ada /X11, Windows, Mac
- Mark V Systems
  16400 Ventura Boulevard, Enrico, CA 91436, U.S.A.
  Fax:+1-818-995-4267, Email:objmaker@hermix.UUCP
- （株）PSI
  TEL:0263-26-8220, 03-3490-2556, Fax:0263-26-4029

[OO-Framework試作品/PDS]

BETA:　Simulaの後継オブジェクト指向言語環境
pre-release版をftpするに、support@mjolner.dkに問い合わせる。
Mjolner Informatics ApS Science Park Aarhus, Gustav Wiedsvej 10,
DK-8000 Aarhus C, DENMARK, E-mail:mjolner@mjolner.dk
Fax: +45-86-20-12-22, phone:+45-86-20-20-00,

Demeter:　Demeterの法則による分析/設計を支援する統合型CASEツール
X対応前のデモ版 (Sun3/4用バイナリ）が同大学から ftp で 入手できる。(129.10.8.150 pub/demeter)

Modula-3: Modula-2の後継オブジェクト指向言語環境 (サポートなし)
バージョン1.6のソースコード及びドキュメントが
gatekeeper.dec.com (pub/Dec/Modula-3)からftpで入手できる。