ソフトウェア工学 103−25
（1995．3．10）

管理主体の特性に応じた粒度で格納するリポジトリの構築法について

佐藤 友康　　森保 健治

NTT ソフトウェア研究所

ソフトウェア開発において，リポジトリの構築が重要になっている．一方，CASE ツールを利用する環境においては，CASE ツールのデータもプロダクトの一部である．従ってドキュメントと同様にリポジトリで管理することが望まれるが，情報の性質や粒度が異なるため，これらの統合管理が問題になる．

本稿では，プロダクトの管理を目的に，様々な粒度の情報を，その性質に適応して統合的な管理を実現するリポジトリ構成法について提案する．ドキュメントとCASE ツールのデータの管理エリアを設け，ファイル単位，あるいはそれらをまとめた情報単位で管理し，エリア間で互いに関連を持つデータには波及解析を行うことで各エリア間の一貫性を保つことを実現する．

キーワード：リポジトリ，CASE ツール，情報粒度，開発環境，PCTE

Management of software design information decomposed into fit granularity

Tomoyasu SATO　Kenji MORIYASU
NTT Software Laboratories
3-9-11 Midori-cho Musashino-city Tokyo, Japan, 180

To share many documents of software development, it is effective to introduce repository to development environment. On the other hand, since CASE tools are often used to design efficiently, CASE design information also be managed in repository.

This paper describes a method of integration management of both documents and CASE design information with suitable granularity. The repository is divided by Document area and CASE-Semantics-Information area. In Document area, documents and CASE graphic information are stored, and in CASE-Semantics-Information area, CASE semantics information are. To maintain the consistency all over these areas, objects in Document area are linked with others in CASE-Semantics-Information area which are related each other, and change propagation is available with the links.

Keywords: repository, CASE, information granularity, software development environment, PCTE

1．はじめに

ソフトウェア開発において，多くの設計書や仕様書のようなプロダクトが作成される．プロダクトは開発作業に携わる多くの人から参照される．そこで，プロダクトの効率的な管理が重要になっている．これを実現するために，ソフトウェア開発におけるバックボーンとしてのリポジトリの存在が重要である．リポジトリは複数の開発者から長期的にわたって共有されるため，以下のような項目を満足することが重要である．

- オンラインでの登録／参照／取得が可能
- 設計書やファイルなどのような任意の単位で取得が可能
- 同様に任意の単位で再構築が可能
- 複数のシステム開発で利用可能

一方，ソフトウェア開発には生産性向上のために CASE (Computer Aided Software Engineering) ツールが利用されている[1]．CASE ツールのデータは，プロダクトの一部と考えられ，文書などのドキュメントと同様にリポジトリで管理することが望ましい．この時，情報の性質や粒度が異なるため，ドキュメントとCASE ツールのデータの統一的な管理が問題になる．文献[2]では，情報の管理を必要以上の細粒度で管理を行っても性能が低下するのみであり，既存ツールの機能／方法論を尊重し，ツールのリポジトリへの統合化を容易にするためにも，ユーザやツールの利用形態に応じた適切な粒度で管理することが望ましいと述べている．

リポジトリで管理する情報には，開発において作成されるプロダクトと，開発工程や，その進捗を管理するプロジェクトがある．本稿では，プロダクトの管理に焦点を絞り，ドキュメントとCASE ツールデータを，それぞれの性質に適応して統括管理するリポジトリの構成法について提案する．

2．作業モデル

リポジトリの構築には，プロダクト管理のために作業形態のモデル化を行い，プロダクトの適性粒度を明らかにすることが必要である．そのためにプロダクトの論理構成の明確化とその作成を支援するための管理手法の明確化する．さらにセキュリティのためのユーザ管理が重要である．これらの３点を表現するためのモデルは以下の通りである．

- プロダクト構成モデル
- プロダクト管理モデル
- ユーザモデル

2．1　プロダクト構成モデル

プロダクトとは，ソフトウェア開発作業において作成される仕様書／設計書などのドキュメントやCASE ツールのデータを指す．

通常，仕様書は章や節から構成され，図１のような木構造になっている．例えばレコード仕様書は，システムデータモデルを定義する章などから構成され，章は実体関連図などを書く節から構成される．さらに節にはプロダクトの実体である文や図が入る．このように，プロダクトは文書と章／節のような構成要素，そして文／図であるコンテンツから

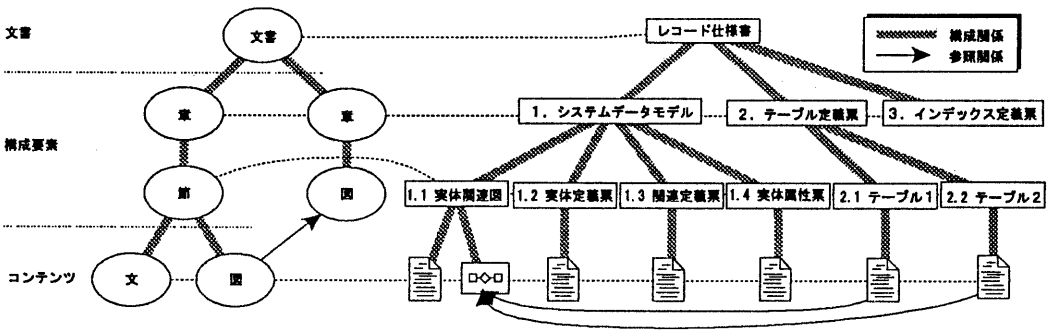


図１．一般的な仕様書の構成

構成される．通常，仕様書の作成には，開発者が章や節単位で分担し，各担当部分を作成する．また，利用ツールはファイルが最小単位である．これに従えば，章／節や，実際の作成単位であるファイルが最適な管理粒度であると言える．

ドキュメント構造は，木構造に加え，各ドキュメント間での関連が存在する．図１の例では，テーブル１やテーブル２の設計に，実体関連図を参照している．実体関連図のデータであるエンティティを取り出し，テーブル定義に利用する．このような場合，両者と参照元である実体関連図に関連を持たせると，波及解析の面でも有利となる．しかし，単純に関連づけることは無秩序なデータ管理になる．そこで，関連を意味付けし，各データの存在性を表現する．図１のように，木構造の表現のために，コンテンツが構成要素を，構成要素が文書を構成する．このような場合，構成関係にあると言える．また，ドキュメント作成には，他のドキュメントのデータを取って，それを元に作成するということが多い．先ほどの実体関連図とテーブル設計の関係の場合，データを取った側はデータを渡した側とは参照している関係にある．従って，このような場合，参照関係であるとする．

両者の関係の特徴は，構成関係は，木構造形成のため，上位ノードが存在しなければ下位ノードが存在することができないという制約条件がある．図１の例では１章のシステムデータモデルが存在しなければ，１章の下の１節の実態関連図などは存在できない．参照関係は構成関係とは異なり，他のプロダクトのデータを参照した時に関連を持つ．そのため，参照元が存在しなくても参照先は存在可能である．参照関係はデータを参照している関係から，波及解析に利用することができる．

CASE ツールのデータは，各ツール独自に管理方式を導入しており，管理方式を統一的にモデル化することはできない．従って，各ツールの管理方式に合わせることとする．

さらにCASE ツールのデータ管理における細粒度での管理は，データリソースの意味的管理を行う上で重要である．このような管理もツール毎に依存するため，各ツールが行うこととする[2].

２．２　プロダクト管理モデル

プロダクトの作成過程では，仕様変更などによりプロダクトの変更が行われる．このためたくさんの版のプロダクトが派生し，そのプロダクトを参照して行う後続の作業において混乱を招くことになるのみならず，そのプロダクトの作成途中にも以前の版との違いを把握することが困難になる．そこでプロダクト管理モデルは，変更を管理することによる作業支援のためのモデルである．

プロダクトは一般的に，図２に示すように作成／修正／テスト／ライブラリ登録の作業を繰り返し行われる．作成後，インフォーマルにコメントをもらったり，テスト結果の修正要求などを吸収すると，プロダクト開発者が担当する範囲で多くの変更作業が行われていることになる．そこで，開発者の担当範囲において管理を行う必要がある．さらにテストに合格したプロダクトの管理も必要である．テスト合格後は，プロダクトを管理する人（管理者）の手元にあるため，管理者に合わせて管理する必要がある．このように変更の管理は開発者と管理者の両方の視点に対応した管理が必要である．管理者の管理単位はプロダクト構成モデルでの文書に対応するため，文書単位で管理することが適当である．一方，開発者にとってはファイル単位での管理が適当であると言える．

前述のようなプロダクト作成過程を支援するためには，バージョン管理が有効である．このとき，開発者と管理者の両方の視点でバ

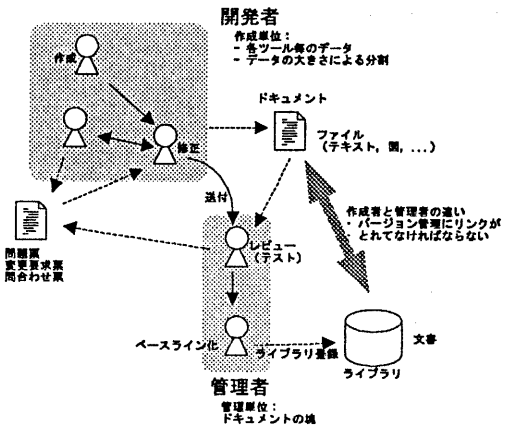


図２．プロダクト作成過程

ージョン管理を行えば，ファイル送付時の確認の点などで有効になる．そこで，ファイル単位と文書単位の多段階にわたるバージョン管理を導入する．

2．3　ユーザモデル

前節までに述べたモデルに基づき，開発作業に携わる人の担当を明確化しておくことが作業の効率化につながる．ユーザモデルはユーザとユーザの役割分担について表現する．

ユーザモデルには以下の4つのリソースが存在する．

(1)ユーザ

プロジェクトに関わるすべての人を指す．

(2)立場

立場にはプロジェクト全体を統括するプロジェクトリーダ，プロジェクト内での担当部門を統括／管理するサブリーダ，そして実際に開発を行う一般開発者が存在する．上位の立場は下位の立場の権限を制限することができる．これらの立場は図3のように木構造で表現できる．尚，サブリーダは，組織により数段階に階層化されることがある．

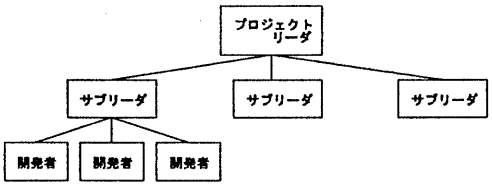


図3．立場

(3)権限／役割

権限とは，ファイルのアクセス権のような，実行を行える資格のことである．権限がユーザに与えられて役割となる．権限には，プロダクト管理モデルで述べた，作成者の作成権，管理者の管理権がある．さらにプロダクトを参照するための参照権がある．

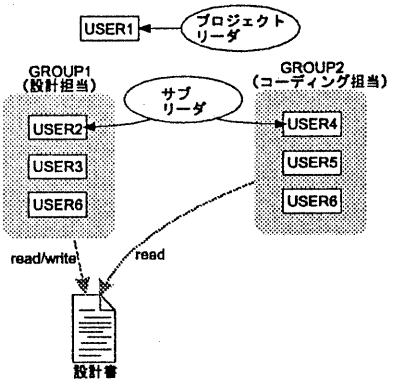
(4)グループ

開発作業において，ユーザにはプロセス設計やデータ設計などのような，担当が振り分けられる．プロジェクト内のユーザの担当の明確化のために，ユーザのグループ化が必要である．通常，グループは設計作業の担当部門単位で分けられる．但し，さらに細かく設定されることがあり得る．例えば，1部門内においてプロダクトの作成やプロダクトの管理をするために，プロダクトに対する作成権（書き込み権）や管理権をグループ単位で与える場合などがそれに相当する．このように，グループ分けには，担当分野の定義やグループによるアクセス権の定義が可能などのメリットがある．

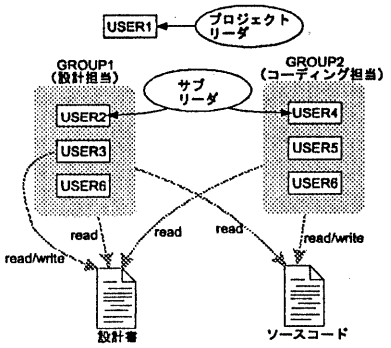
これらのリソースは独立に存在するが，互いに関連するものである．関連づけによってユーザに様々な作業を行う権利が与えられたり，各立場が割り当てられる．

立場の割り当てによって，各ユーザはプロジェクトリーダ，サブリーダ，一般開発者となる．

各ユーザに役割を割り当てると権限を得ることができる．ユーザへの役割を与えるのは，通常は部門での管理者（サブリーダ）である．



(a)設計書作成時



(b)ソースコード作成時

図4．ユーザの権限，役割

そこでユーザに権限を与えるのは，上位層の立場のユーザによって行われる．

ユーザ，グループ，立場，及び権限は，互いに独立の関係にある．そこで１ユーザに複数の立場や権限が与えられたり，１ユーザが複数のグループに属することがある．例えば，図４に示すように，USER1 にプロジェクトリーダという立場を与える．USER1 は USER2, USER3, USER6 に設計担当として，グループ化(GROUP1)し，USER2 にサブリーダの権限を与える．また，USER4, USER5, USER6 はコーディング担当として，GROUP2 を作成し，USER4 をサブリーダとする．この時USER6 は両方のグループに所属していることになる．作成担当のユーザは設計書を設計するため，GROUP1 には設計書へのread/write 権が与えられる．コーディング担当のユーザは設計書を参照しながら作業を行うため，GROUP2 には参照が可能なread 権のみが与えられる．作成が完了すると，設計書の管理が必要になる．その管理にサブリーダUSER2 が USER3 に管理権を与える．このとき，仕様書が完成した時点でUSER2 は USER3 にのみread/write 権を残し，他のGROUP1 のユーザからwrite 権を消す（プロダクトの凍結）．一方，プログラムコードにはGROUP1 には read 権のみが与えられ，GROUP2 には read/write 権のみが与えられる．

このように，ユーザ，立場と役割から構成されたユーザモデルによりユーザの役割，権限の明確化を行う．

３．リポジトリの構成

冒頭で述べたように，リポジトリで管理するプロダクトには仕様書や設計書のようなドキュメントとCASE ツールのデータがある．そこで，これらを前述した粒度で管理する方法について述べる．

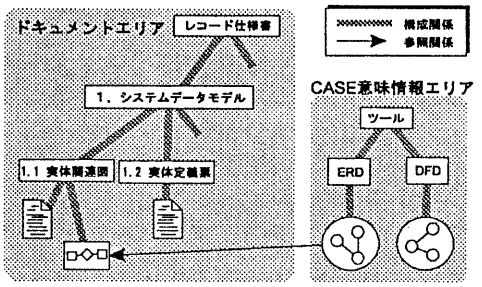


図５．プロダクトの管理

ドキュメントとCASE ツールのデータは，性質の異なるソフトウェア設計情報である．一般に，ドキュメントは表示情報であるのに対し，CASE ツールのデータは図形情報である表示情報と，図形の意味を示す意味情報の２種類の情報からなる．意味情報はドキュメントとしては扱えない．CASE ツールの情報は，意味情報のみを管理し，図形の出力は対応するツールで行えば良いと言うことも考えられる．しかし，CASE ツールなどは，DFD などの記述機能にチューンされているため，図のカスイタマイズや，ドキュメントへの組み込みという点では自由度が低い．このため，図形情報を他のツールでも読み込みが可能な形式で管理されることが要求される．そこで図５に示すように，性質の異なる各情報に対応してリポジトリの内部をエリアに分割し，ドキュメント（CASE ツールの表示情報を含む）を管理するドキュメントエリアとCASE ツールの意味情報を管理するCASE 意味情報エリアを設ける．各エリア内には対応した情報を格納する．以下では各設計情報の管理単位のことをオブジェクトと呼ぶ．以下に各エリアについて説明する．

３．１　ドキュメントエリア

ドキュメントエリアは図５のように，プロダクト構成モデルやプロダクト管理モデルに基づき，ドキュメントを管理する．ドキュメントの管理単位は，一般開発者が管理するファイル単位，管理者が管理する文書がある．一般にドキュメントの最小構成単位はファイルであり，ドキュメントエリアではこのファイルをプロダクト構成モデルに従って木構造を構成して格納される．

CASE ツールのデータは，EPS ファイルのような図形情報をドキュメントとして管理する．

ユーザモデルにおける権限は，オブジェクトへのアクセス権で表現される．例えばあるプロダクトの作成，編集を行う権利を与えられている開発者はwrite 権が与えられ，参照

する権利を与えられている開発者にはread 権が与えられる．つまり，ユーザやグループに対してアクセス権が与えられる．

プロダクト管理モデルは，ドキュメントエリアの構成要素，テキスト，図のバージョン管理によって実現される．

3．2　CASE 意味情報エリア

利用するCASE ツールが持つ独自のデータを格納する．CASE ツールによっては独自にディレクトリ作成を行ってデータを保存することもあるが，この場合は，ディレクトリ構造なども再現して格納する．

ドキュメントエリアのデータに対し，対応するCASE 意味情報エリアのオブジェクトも同期してバージョン管理を行う．このためには，CASE ツールの意味情報とそれに対応する表示情報を関連づけ，ドキュメントと同期をとって管理する．プロダクト管理モデルに基づくバージョン管理はドキュメントオブジェクトでのバージョンに合わせて，それに対応するCASE 意味情報エリア内のオブジェクトもバージョンが設定される．

変更による影響波及解析は，テキストや図にリンクされている参照リンクを元に行う．CASE ツール間でデータを変換する場合，図間のリンクたどり，その図に関連するCASE ツールの表示情報を解析して変換を行う．このようにすることで，ドキュメントとCASE ツールの情報の一貫性を保証する．

3．3　システム構成

システム構成図を図6に示す．リポジトリの周辺に各種ツールにリポジトリとの仲介役を担うカプセルをかぶせて統合する．インテグレートツールにはワープロと構造化分析／設計[3]を支援するCASE ツールを用いる．通常，CASE ツールの設計情報は，図単位でファイルに持つ．従って，ドキュメントエリアに図形情報を格納し，CASE 意味情報エリアに対応するオブジェクトをファイル単位で作成し，そのオブジェクトにファイルを格納する．

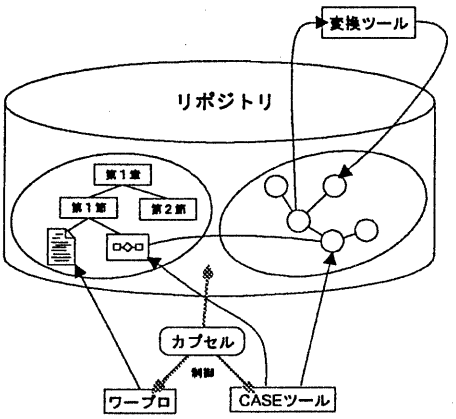


図6．システム構成

構造化分析／設計で作成するデータフロー図(DFD)と実体関連図(ERD)では，DFD のデータストアがERD のエンティティとして利用されるため，これらの間でデータ変換を行い，DFD のデータをERD の作成に利用できるようにする．データ変換はデータ変換ツールを用いて行う．

変換元と変換先の関連は，ドキュメントエリアのオブジェクト間において参照リンクを張り，それに対応したCASE 意味情報エリアのオブジェクトに対して変換を行う．CASE 意味情報エリアの変換元と変換先のオブジェクト間には関連づけない．なぜならドキュメントエリアでのリンクで変換元から変換先が検索可能であるからである．このような参照リンクはすべてドキュメントエリアで張ることによってドキュメント同士，ドキュメントとCASE ツールデータ，あるいはCASE ツールデータ同士のようなデータタイプによるリンクの差別化をなくす．これにより，リポジトリはデータの関連を解析する際にデータタイプを意識せずにオブジェクトの管理が可能である．

カプセルは，リポジトリのオブジェクトの格納／検索を行ったり，ツールの起動を行う．ドキュメントエリアの各オブジェクトにはそのコンテンツが作成されたツール名やデータタイプの属性を組み込んでおく．カプセルは，オブジェクトデータを取りだし，オブジェクトに対応するツールを起動する．編集終了時には，設計情報をリポジトリ内の適当なオブジェクトに格納する．

各オブジェクトには，オブジェクト管理モデルに対応してバージョン管理を行っているが，バージョンの検索はデフォルトでは最新バージョンを検索する．必要に応じて特定の

バージョンを検索を行えるようになっている．

リポジトリを構成するには，データモデルが必要である．現在，リポジトリのデータモデルとして，リポジトリのオペレーションも整備されているものを利用すると比較的実現が容易になる．そこで本稿では，ECMA (European Computer Manufacturers Association) / NIST (National Institute of Standards and Technology)で策定された，リポジトリの標準規格であるPCTE (Portable Common Tool Environment)[4][5][6]を採用する．PCTE を採用することにより，オブジェクトの関連にはプロダクト構成モデルで述べた構成関係と参照関係の表現が可能となる．また，オブジェクト単位でのバージョン管理が可能なため，プロダクト管理モデルの実現も可能となる．

これらを考慮したスキーマを図７に示す．図はPCTE の表記法にほぼ従っている．エンティティが重なって表現されているものは，そのオブジェクトに対してバージョン管理を行うことを示す．通常，開発者は作業領域においてプロダクトの作成作業を行い，完成すればライブラリに登録される．各領域にはドキュメントエリア，CASE 意味情報エリアを持つ．各オブジェクトは，作成ツールやオブジェクトのデータタイプを管理している．このようにしてオブジェクトがどのようなものであるかを判別可能にしている．

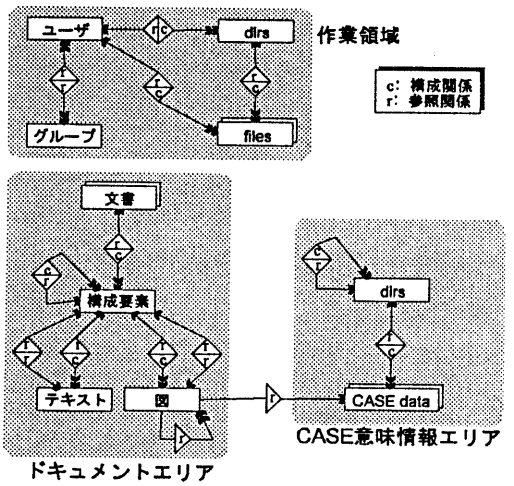


図７．リポジトリスキーマ

４．効果

DFD から ERD を作成する場合を考える．DFD の設計が完成し，ライブラリに登録される（初版）．前述のように，DFD のデータストアからERD のエンティティを作成するため，データ変換ツール（DFD→ERD）を利用する．変換をかける際に，ドキュメントエリアに ERD の表示情報のオブジェクトを作成し，同時にCASE 意味情報エリアにERD に対応する意味情報のオブジェクトが生成される．この時，ドキュメントエリアにおいて，DFD と ERD の間に参照リンクが生まれる．DFD に設計変更が生じると，DFD はバージョンアップされ（２版），同時に参照リンクを通じてERD に変換をかければよいことがわかる．さらに，ERD のドキュメントエリア内のオブジェクトとCASE 意味情報エリア内のオブジェクトはリンクがあるため，CASE 意味情報エリア内の２つのオブジェクト同士でデータ変換をかければ良いことがわかる．このように，ドキュメントとCASE ツールデータの一貫性の確保が実現される．

本構成においては，１章で述べた要求事項は満足可能である．プロダクト構成モデルによるプロダクトの構成を管理することで，任意の単位で取得が可能である．例えば，編集のためにファイルを取り出したり，あるいはプロダクト管理のために文書を取り出すことが可能である．

さらに，バージョン管理などのような管理機能に焦点を当てると，ツールによりバージョン管理機能を持つツールもあれば，持たないツールも存在する．機能的には存在しても，ツールによって機能実現レベルが異なる場合が多い．このような機能を共通的に提供するため，ツールに依存しなくなることにより，ユーザの管理負担の軽減となる．

これまでは，データ変換時はデータ変換ツールがCASE ツール個別に対応していたため，ユーザが変換ツールを利用CASE ツールにより使い分けなければならないという負担があった．本稿の構成では，リポジトリのデータにツールを対応させており，データによって自動で変換ツールの選択が可能になるので，

このような負担がなくなり，スムーズに変換作業が行える．

また，作業形態に応じた，適切な粒度でデータ管理を行っているため，性能低下を防ぐことができる．

5．関連研究

本稿では適性に応じた粒度で情報を管理し，ドキュメントとCASE ツールのデータの統合的な管理を実現している．これに関連する研究を以下に述べる．

KyotoDB[7]は，ソフトウェアエンジニアリングデータベースと称し，ソフトウェア開発保守におけるプロセスとプロダクトを統合的に管理することを目的として，Project，Plan，Product，Coordinator という４つのクラスのオブジェクトの複合構造により，構成複合，参照複合，継承複合で関連を示すことで実現している．

文献[8]では，DB2 をベースとしてリポジトリを構築し，市販CASE ツールの統合環境の構築法について述べている．

文献[9]では，integral version managementの実装について述べている．本稿においても，作成者と管理者の２つの観点からバージョン管理を行う点で，参考になる文献である．

6．まとめ

リポジトリの構築法については，これまで長い間にわたり，研究されてきたが，その多くがCASE ツール間のデータ共有のために細粒度でのデータ管理を目標としてきた．本稿で提案したリポジトリは，ソフトウェア開発の中で最も多く作成されるプロダクトであるドキュメントの管理を行うという点でリポジトリを越えるものといえるであろう．

本稿では，設計情報の特性に応じた粒度で設計情報を管理するリポジトリの構築法について述べてきた．バージョン管理には，２つの視点に基づいて行う方式を検討したが，各視点に合わせ，多段階にバージョン管理を行うとトレースが複雑となり，旧バージョンの復元などが困難になることもある．今後はこのような問題点について取り組む予定である．

謝辞

本研究の機会を与えていただいた黒田幸明研究グループリーダ，様々なご助言をいただいたNTT ソフトウェア株式会社の梶原清彦担当課長，宮崎裕司設計主任，そしてご協力をいただいた多くの方々に感謝いたします．

参考文献

[1]原田: CASE のすべて，オーム社，1991

[2]佐藤，森保，他: PCTEの性能に基づく効果的な利用法について，情報処理学会第49 回全国大会講演論文集(5), pp.131-132, 1994

[3]DeMarco, Tom: Structured Analysis and System Specification,YOUDON,Inc.,1978

[4]Lois Wakeman, Jonathan Jowett: PCTE - The Standard for Open repositories, Prentice Hall, 1993

[5]ECMA: STANDARD ECMA-149 Portable Common Tool Environment (PCTE) abstract specification, 1993

[6]ECMA: STANDARD ECMA-158 Portable Common Tool Environment (PCTE) C programming language binding, 1993

[7]鰺坂，松本: ソフトウェアエンジニアリング・データベースKyotoDB の設計と実現，情報処理学会論文誌，Vol.33, No.11, 1992

[8]L.Slusky: Modeling of I-CASE Platform, Information Software Technology, Vol.33, No.8, 1991

[9]E.Lippe，G.Florijn: Implement Techniques for Integral Version Management, ECOOP'91, pp.342-359, 1991

ウィンターワークショップ・イン・沖縄'95 開催報告

鯵坂恒夫 (京都大学), 平川正人 (広島大学), 荒野高志 (NTT),
菅沼 明 (九州大学), 中谷多哉子 (富士ゼロックス情報システム), 藤岡 卓 (三菱電機),
深澤良彰 (早稲田大学), 田代秀一 (電子技術総合研究所),
松本健一 (奈良先端大学), 杉山安洋 (日本大学), 青山幹雄 (富士通)

あらまし

第102回研究会は「ウィンターワークショップ・イン・沖縄'95」と題して，1995年1月26,27の両日，沖縄県宜野湾市の沖縄ハイツ（沖縄勤労福祉センター）で開催された．合宿形式のワークショップは93年夏の大雪山に続いて2度目である．予想を大幅に上回る32件の講演，74名の参加を得た大盛会となり，何はともあれ「ソフトウェア工学危機」は杞憂ではないかと元気づけられる研究会となった．

ワークショップの全体テーマは，昨年9月の100回記念シンポジウム「変革期のソフトウェア工学」を受け，「変革期を越えるソフトウェア工学を求めて」という挑戦的なスローガンである．何が変革かというと，開発対象の変化と開発方法の変化である．講演募集ではそれぞれの細目として次のような例を示していた．

- 開発対象の変化に対応するエンジニアリング
  - クライアント・サーバシステムを対象とするもの
  - マルチメディアデータを扱うシステムを対象とするもの
  - 超高信頼性が要求されるシステムを対象とするもの
  - 遺産的 (legacy) システムを対象に進化的保守開発をするもの
- 開発方法の変化に対応するエンジニアリング
  - オブジェクト指向技術－その実践やより精細な分析 / 設計法など
  - マルチエージェントモデルによる開発方法
  - リポジトリとオープンCASEを活用した開発方法
  - ソフトウェアプロセスや協調作業の定式化を活用した開発方法
  - ドメインモデリングに基づく開発方法

集まった論文は残念ながら（予想通り）必ずしも両者のバランスが取れず，かなり「方法」の方に偏ったものとなり，プログラム編成に苦労することとなった．ひとことで言えば，あいかわらずオブジェクト指向技術が絶大な勢力を誇っている．しかし，プログラム理解や波及解析といったlegacy対応のもの，クライアント・サーバをキーワードとするものなども集まり，対象の変化に対応する芽も感じられる．セッション構成は，古典的ながら上流（実世界をソフトウェアシステムに写像するモデリング），中下流（ソフトウェアプロダクトの解析と加工）の大分類に，オブジェクト指向，プロセス，並行制御などの技術分野を重畳したもので，以下の通りである．

セッション1 ソフトウェアプロダクトの解析と加工 (1) — 仕様／プログラム理解，保守，リエンジニアリング（司会：鯵坂恒夫，菅沼 明）

セッション2 実世界をソフトウェアシステムに写像するモデリング (1) — オブジェクト指向分析 / 設計（司会：中谷多哉子）

セッション3 ソフトウェアプロダクトの解析と加工 (2) — オブジェクト指向再利用技術（司会：藤岡 卓）

セッション5 実世界をソフトウェアシステムに写像するモデリング (2) — 仕様化，ドメインモデリング（司会：深澤良彰）

セッション6 ソフトウェアシステム構成要素間のコミュニケーション — エージェントモデル，並列オブジェクトモデル，クライアント・サーバシステム（司会：田代秀一）

セッション7 ソフトウェアプロセス — プロジェクト管理，プロセス支援環境（司会：松本健一，杉山安洋）

ひとつ番号の飛んでいるセッション4は，合宿ワークショップの目玉ともいえるナイトセッション（司会：平川正人，荒野高志）である．開発対象の変化と開発方法の変化に対応するエンジニアリングと題し，

- どうすれば役に立つソフトウェア工学を作り上げられるか
- 今後30年のソフトウェア工学変遷のシナリオ — どういう技術がどう開発を変えていくか
- 情報社会への変革期にソフトウェア工学はどう対処すべきか
- パッケージソフトが支配する時代への対応
- 現実と理論のギャップを埋めるには
- オブジェクト指向をいかにして役に立たせるか

といった話題例を用意して望んだが，実際の議論については該当セクションを参照されたい．

ワークショップの進め方については，論文はポジションペーパで議論がほとんどという「純」ワークショップと，講演が中心で議論は質疑応答程度という通常の研究会あるいはシンポジウムとの中間的な性格を狙い，講演：議論＝6：4とした．件数が多いこともあって，論文1件につき，講演15分，議論10分，計25分とややあわただしいものとなった．議論の時間は一部プールして各セッションの最後に総合討論を，というようなことも目論んでいたが，各講演に対する議論がそれぞれ活発に巻き起こり，このパターンはどのセッションでも実現しなかった．

セッション1

ここに多く集まったのは，とくに日本ではややもすると開発に覆い隠されてしまいそうなメンテナンス分野の発表である．リ(バース)エンジニアリング，波及解析，仕様記述，プログラム理解，プログラムスライシング，テスト系列生成，プログラムの細粒度リポジトリといった話題が並んだ．議論されたポイントをいくつか拾ってみると，次のとおりである．

- 作業者のスキルとプログラムやドキュメントに対する理解度
- フェーズ分割によるリエンジニアリング・プロセスの改善
- 非構造的/不整構造プログラムのスライシング
- 形式的仕様記述の理解性向上
- 並行システムのテスト系列自動生成
- プログラム論理構造の柔軟な可視化
- プログラム・リポジトリの意義と環境構成論的位置づけ

プログラムのレベルでは形式性が確立されていて抽象度も低いのだから，構文的に扱えば問題なかろうと思われがちである．しかし，そのすぐうしろには意図や意味という厄介なものが渦巻いており，それを解きほぐして理解する（水平トレース），さらには抽象度を高めていく（垂直トレース）のは大変なことである．性急に一般性，汎用性を求めることなく，とりあえず狭い適用領域をひとつひとつこなしていき，そのなかから一般原理を少しずつ見つけていく努力が重要であろう．遺産的システムの処置に手を焼くことは今後ともまちがいないから，この分野にはもっと精力が注がれてしかるべきである．

セッション2

オブジェクト指向分析/設計を中心にセッション2は構成された．オブジェクト指向分析/設計は，最近になって，やっと方法論が統合の方向に向かい始めたところである．現実のソフトウェア開発に適用するには，まだ多くの課題を抱えている．セッションでは，これらの課題に対する研究が集められ，内容は次に示すように多岐に渡った．

- 各方法論によって生成される成果物の相違
- 方法論をメタモデル技法を用いて整理し，部品化することで，問題領域や環境に対して適切な方法論を再構成し，適用する技術と環境
- オブジェクト指向を適用したソフトウェアプロセスでは，ビジネスアプリケーションにオブジェクト指向を適用
- OMT法の3つのモデルからプロトタイプを生成し，顧客要求とモデルの検証をおこなう環境
- 日本語要求記述から自動的に抽出されたオブジェクトを基に，OMT法の3つのモデル間の整合性を検証

方法論の検討に関する2件の論文を除いて，発表された論文は，オブジェクト指向を小さな問題に対して適用し，課題への対処を検討したという研究が多かった．これらの研究では，オブジェクト抽出は名詞からという方法が一般的に取り上げられているが，それでどれだけ問題領域のオブジェクトを完全に網羅できたのか，自動化することで問題は発生しなかったのか，といった検討も必要であろう．また，OMT法の3つのモデル間の整合性を検証し，モデルを完成した後は，利用者の要求の矛盾点や要求と仕様の不整合をどのように発見し，モデルを完成させていくかといった研究が必要かもしれない．プロトタイプによって顧客の要求確認を行うという研究は，そのひとつの例だった．

さらに，方法論の整理を進めるためには，方法論によって定義されるモデルの違いや強み／弱みといった特徴を認識することが重要である．また，問題領域に適合した独自の方法論を作るためには，メタモデルの検討も不可欠である．この分野に関する今後の研究も多いに期待できる．

セッション3

再利用という括りで構成された本セッションには，

- 上流からのトップダウン的な再利用を目指したモデルの提案
- 再利用時に自ら変化する部品の作成方法
- 部品を組み合わせるための方式と視覚的プログラミング手法
- そうして作成されたオブジェクト指向プログラムのテスト
- 部品作成者と利用者双方へ指針を与えるメトリクス

という，多様ではあったがバランスの取れた論文が集まった．

既存の多くのオブジェクト指向方法論は新規開発を想定しており，オブジェクト指向技術に寄せられる最大の目的である再利用をいかに進めるべきかについて述べてはいない．しかし，実際のソフト生産で毎回一から作るものは少ないのが現状であり，上記のような様々な視点からのアイディアを実践，検証し，それらを整理した再利用ベースの開発の方法論が求められる．

速く書くことには限度があるが，書かないことによる生産性の向上にはまだまだ努力の余地がある．そういう意味からもオブジェクト指向技術に期待できる．

セッション5

ソフトウェアとは，何らかの意味で，実世界をシミュレートするものである．よって，ソフトウェアを開発する際には，実世界をどのようにソフトウェア上へ写像するのかが，開発の容易性，できあがったソフトウェアの信頼性，保守性などに大きな影響を及ぼす．即ち，このモデル化（モデリング）技術が重要な位置を占める．

より良いモデル化をより容易に行うために要素技術として，以下が挙げられよう．

1. モデル化手法：構造化分析，オブジェクト指向等，モデル化の中心（視点）をどこに置くか．
2. モデル表現法：1. によってモデル化されたものをどのような形式で表現するか．通常は，モデル化手法に付随して，表現法も提案される．
3. モデル化関連の支援ツール：ユーザ要求抽出ツール，図式化支援ツール，モデル理解支援ツール，後工程のための情報生成ツールなど．
4. モデルの標準化
5. モデル化の基盤理論
6. 特殊な性質（並行ソフトウェア，オンラインソフトウェア等）をもつソフトウェアへの対応
7. ドメインとしての扱い：一つ一つのシステムのモデル化ではなく，同種のシステムをまとめて考えることの促進．

本セッションは，1. のモデル化手法の観点の内，オブジェクト指向に属するもの（セッション2に切り出されまとめられている）を除いた研究4件から構成されている．多くのテーマをもつ，広い領域に対して，4件の発表だけであるので，この領域のすべての側面をカバーしているとは言えない．しかし，発表の内容を見れば，前述の要素技術が深く関連していることがわかるであろう．これらの技術は，より良いモデル化のためには，欠くべからざるものであり，今後の更なる発展を期待する．

セッション6

広域ネットワークの利用が日常的なものとなってきた昨今，分散し，協調動作するオブジェクトあるいはエージェントによって構成されるソフトウエアシステムへの需要は急速に高まっており，開発対象の変化という意味で「ソフトウエアの変革期」を実感させる好例となっている．

本セッションでは

- 分散した情報サーバおよび情報中継サーバとクライアントからなるシステムの構成法
- 簡略化，統一化されたインタフェースを持ったオブジェクトを連結することによってシステムを構成する手法
- システムの変化に柔軟に対応しうるエージェント間通信方式とその仕様記述法
- 並列オブジェクトシステムの振るまいのモデル化手法

について発表が行なわれた．

ネットワーク上に分散した多数のオブジェクトの連携によって構成されるシステムでは，個々のオブジェクトの仕様にあわせ，オブジェクト相互がどのように係わるかについての仕様（コミュニケーション仕様、全体構成の仕様等）の記述が必要となる．また，システム全体の振るまいを分析するためのモデル化手法も必要である．

ネットワークが大規模化・広域化した場合，システム全体の状態（オブジェクトがどこにあるのか，どんなオブジェクトがあるのか等）の全てを完全に把握することが困難となる．そこで，不完全な情報に基づき，状態の変化にも柔軟に対応しうる仕様記述法の検討が進められている．

また，コミュニケーション法の統一化を図るなどで仕様自体の簡略化を図るアプローチについても発表された．

今回は，このような論点について具体例をまじえつつ，基礎から応用までバランス良く議論を行なうことができた．

セッション7

ソフトウェア開発の技術的，及び，管理的側面について議論する時，ソフトウェアプロセスという概念の果たす役割は小さくないものとなりつつある．本セッションの前半では，ソフトウェアプロセスの概念に基づくプロジェクト管理とプロセス開発支援に関する次の4つの発表が行われた．

- プロジェクト管理ツールとプロセス管理ツールの連携の実現
- 教示的生産管理システムの提案
- ソフトウェアプロセス評価支援システム「SPATS」について
- 開発プロセスを構成する要素間のインタラクションに関する考察

ソフトウェアプロセスの概念に基づくプロジェクト管理においては，組織レベルでの管理フレームワークの確立，及び，開発作業レベルでのプロセス記述とそれ基づく作業支援や進捗・品質データの収集・評価支援が必要となってくる．

管理フレームワークの一つであるプロセス評価フレームワークはこれまでにもいくつか提案されている．しかし，それらの比較検討は十分ではない．ソフトウェアプロセス評価支援システム「SPATS」は，特定のプロセス評価フレームワークに依存しないシステムである．このシステムにより，既存のフレームワークの比較が可能となり，より優れたプロセス評価フレームワークの構築が期待される．また，教示的生産管理システムとは，一般的な工業製品に対して用いられてきた生産管理技法をソフトウェア開発に適用するためのモデルである．ソフトウェアと他の工業製品を同一視してよいかどうか，セッション参加者による議論は結論に至らなかったが，管理フレームワークを考える上で示唆に富んだ意見が交わされた．

開発作業レベルでのプロセス記述においては，開発作業を正確に記述できるだけでなく，プロセス記述からの実作業の逸脱やプロセス実行中の記述変更に対応できる柔軟なメカニズムが必要となる．プロセスの構成要素間のインタラクションを形式的に表現するモデルは，開発者間で頻繁に行われる情報のやりとりをより正確に，しかも，現実に即した柔軟な形式で表現する手段として提案されたものである．このモデルの持つ記述上の柔軟性は，そのまま開発支援の柔軟性へと繋がる可能性を持つ．また，プロジェクト管理ツールとプロセス管理ツールの連携の実現は，プロセス記述に基づく作業支援や進捗・品質データの収集・評価支援をより効率良く行うための試みである．この種の支援を既存のツールやシステムを用いて実現することは可能である．しかし，必要なツールやシステムを統合し，一つのシステムとする試みはまだ始まったばかりである．実際のソフトウェア開発で利用されている2つのシステムを対象とした，この試みの意義は大きい．

セッションの後半は，プログラム編成の関係上，

- 統合型 CASE の評価法の提案と実存の CASE の評価結果の報告
- プロセス統合のためのユーザインタフェースに関する新モデルの提案
- マルチメディアの内容の生産性に関する考察

といったセッションのタイトルにとらわれない広い範囲の発表が集まる結果となった．ワークショップ最後のセッションでもあり，発表者も聴衆も疲れてはいたが，疲れを表に見せずに活発な意見の交換がなされた．

数多くの CASE が市販されるようになった現在，ユーザにとっては，どれを選択するかの判断が難しい．CASE の発展とともに，その統一的な比較方法の研究や評価が活発になされるべきであり，それは我々ソフトウェア研究者，技術者の使命の一つであろう．また，ソフトウエアプロセスの実行に人間が参加できることは，プロセスにとって必要不可欠の要素である．プロセス自身の研究や，その実行メカニズムの研究に加えて，プロセスと人間のインタフェースを研究していくことは，今後の大きな課題である．また，ソフトウェアの開発

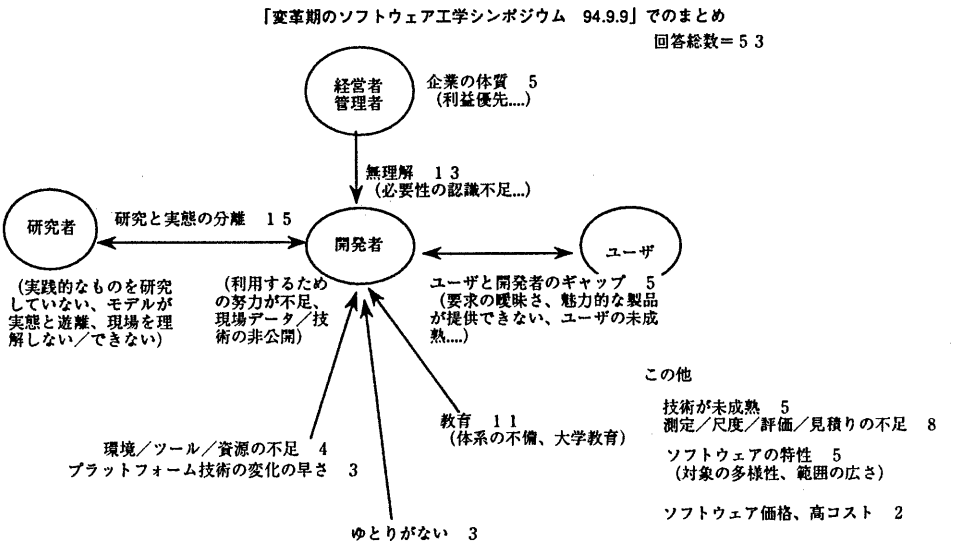


図 1: ソフトウェア工学実践の阻害要因

も，今までのプログラムだけを開発していれば良い時代は終わり，マルチメディアの内容の開発方法が問われる時代になった．マルチメディアソフトウェアエンジニアリングは，非公式のプリワークショップチュートリアルで取り上げられた話題でもあり，最後の発表は，これに呼応して，同じ話題でワークショップの締めくくりとなった．

ナイトセッション

ナイトセッションは参加者の生の声を聞き出す絶好の機会である．今回，そういった効果が最大限得られるように，事前にウォーミングアップを兼ねて，電子メールを使っての意見交換の場を設けた．一足先に沖縄行きのネットワークパスに搭乗いただいた３氏は，都合の良いことに，それぞれ企業，大学教官，学生という異なった立場におられ，そこでナイトセッション当日の議論を興すきっかけ作りをお願いした．加えて，今回のワークショップでは前日からチェックインする参加者が多いという点を活かし，プレ・ナイトセッションという企画が用意された．100 回記念シンポジウムのチュートリアルで取り上げられた「マルチメディア・ソフトウェア工学」をベースにした問題提起を受けて，適度な飲み物と気軽な雰囲気の中，参加者の生の声が聞き出せたと思う．「今後の研究を進める前に，これまでの成果について十分な評価をすることが必要ではないだろうか」という声が大きかった．さらに駄目押し的に，参加者全員にアンケート形式で，ソフトウェア工学の抱える問題点，有望と思われる技術項目，ナイトセッション向けトピックスなどを訊ね，本番に備えた．

ウォーミングアップ（懇親会の後ということで，胃袋的にも）十分な中，ナイトセッションがいよいよ開始された．放っておくと話が発散する恐れもあったので，企画側で予めストーリを決めておいた．まずは「ソフトウェア工学について」，その目的，達成度，変革期という位置づけの可否，といった話題を取り上げた．その議論を受けて後半では，ソフトウェア工学の実践に向けての阻害要因は一体何だろうか，という質問を投げかけた．前述の企業，大学教官，学生を代表した三氏（津田道夫氏：日立製作所，新城靖氏：琉球大学，小元規重氏：電気通信大学大学院）には，この場面で登壇願った．特に津田氏の，ビジネスや情報処理技術の変化にソフトウェア工学は対応すべきであるという意見は，今回のセッションタイトルにもぴったりであった．また，企画側では，94 年 9 月の「変革期のソフトウェア工学シンポジウム」でのパネル「理論と現場のギャップを埋めるには」での議論をまとめて紹介した（図１参照）．そこでは，特に研究者と実践者の意識のギャップ

表 1:「今後 10 年間に役立つ / 普及すると思われる技術」アンケート

1 位	部品化 / 再利用	22 票
2 位	オブジェクト指向	21 票
3 位	開発プロセス	11 票
4 位	開発支援環境 /CASE ツール	10 票
4 位	CSCW/ グループウェア	10 票

を指摘するものが多かったため，3 氏の発表とからめて，そこを議論の出発点としようと試みたが，企画側の思惑とは裏腹に，ボルテージが上がってくるといよいよ議論が道路からはみ出してきて，セッションも最後の方になるとガイド役を諦めざるを得なくなってしまった．

さて全体を通しての議論を大雑把に整理すると，まずは，ソフトウェア工学の教本となるべき優れた教科書を用意すべきだという意見に支持が得られた．これに関連して，ソフトウェア工学を捉える軸を 3 つ設定するとしたら，それらは一体何か，という問題提起があった．上流から下流へと向かう開発工程，技術（プロダクト）と管理（プロセス），という 2 つについては参加者の合意が得られたようだが，最後の 3 軸目については決定打は見つからなかった．

事前にとった今後有望な技術というアンケート結果（表 1 参照）に示唆する通り，参加者の興味はオブジェクト指向の議論を沸騰させた．「オブジェクト指向は SA/SD の単なる焼直しにすぎない」という根本的疑問派や，「オブジェクト指向は GUI 以外に役に立つか」「所詮，再利用できる部品はオブジェクト指向以外の printf 関数ぐらいのものだ」といったオブジェクト指向の効果を疑問視する意見も出た．また，従来技法をスキーのボーゲン，オブジェクト指向をパラレルにたとえ，「オブジェクト指向は真剣にソフトウェア作りをしてきた人には良い道具だが，マトモにものを作れない人にとっては役に立たない」といった議論もあった．この背景には「オブジェクト指向は今だアンステーブルである」という共通認識がある．オブジェクト指向に限らず，好評を博した技術は万能であるかのように錯覚されるきらいがある．しかしながらそれは誤解であって，それぞれに向き不向きがあるので，その点を十分に理解した上で適用することが重要である，といった主張には耳を貸す必要があろう．

全員の合意が取れるところまではいかなかったが，工房（家内工業）と工場との対比，それらの工学との位置づけ，役割りについての議論には興味深いものがあった．ビジネスや周辺技術の変化に対応するためには従来の工場的な考え方だけではだめであるという意見，工房という考え方に工学が入り得る / 得ないという議論，工房も工場も本来どちらも必要なはずだが，日本では特に画一的になっているという意見などがあった．また変革期というキーワードに対しては，ソフトウェア工学はこれまで 20 数年の歴史的積み重ねがあるにもかかわらず，「奇跡（実際に役に立つこと，宗教には不可欠）」は一度も起こっておらず，若手技術者が夢を持ち得ない状況にあるといった指摘もあった．プレ・ナイトセッションで議論された評価の必要性や，実践と理論とのギャップといった問題とも絡み，今後常に頭に置いておくべき事柄のひとつであろう．また，工学という枠内にだけ留まっておくのではなく，新しい応用（例えばマルチメディア）を意識して，異分野との交流に積極的に取り組む必要性についても意見があった．

総数 47 で複数回答あり．ちなみに，94 年 9 月「変革期のソフトウェア工学シンポジウム」のアンケート結果は，オブジェクト指向，部品化 / 再利用，開発支援環境 /CASE ツール，CSCW/ グループウェア，プログラム合成 / 自動プログラミング，開発プロセスと続く．参加者が変わっても，ほぼ上位を占める項目は同じである点は注目に値する．

今後に向けて

今回のワークショップは，「変革期を越えるソフトウェア工学を求めて」をテーマとした．もとより，変革期を越えることは一朝一夕には困難である．しかし，問題提起や新しい成果の発表など活発で，かつ，本質的な議論ができたのではないかと思っている．このような議論を通して，このワークショップが，今後のソフトウェア工学のあり方を議論し，研究，開発へのフィードバックの契機となることを期待する．ここで生まれた多くの小さな波が成長を続け，変革期を越える大きな潮流になって欲しい．

参加者の方々へのアンケート調査によると，このようなワークショップの開催には全員の方から良かったとの回答を得た．特に，合宿形式で行うことにより，セションのみならずナイトセション，懇親会，食事などの時間に討論したり話しあう時間が多かった点が好評であった．また，今後取り上げるべきテーマとしては，やはり，オブジェクト指向の声が多く聞かれた．しかし，ソフトウェア工学の教育，理論と実践のギャップ，企業内でのソフトウェア工学の実践，技術移転，ソフトウェア工学の体系化など，ワークショップの議論を踏まえた興味深い提案もあった．

ソフトウェア工学研究会では，この合宿形式のワークショップを，年1回，継続的に開催していく予定である．アンケートでは，ワークショップとして取り上げるべきテーマや運営について多数の意見を頂いたので，今後の企画，運営に活かしたい．

なお，次回は，7月20(木)～21日(金)に立山厚生年金休暇センターで開催の予定です．発表申込みの締切りは4月21日(金)，原稿の締切りは6月末です．