Vol. 36 No. 5　　情　報　処　理　　May 1995

解　説

ソフトウェアプロセス

# 7. プロセス支援環境†

鯵　坂　恒　夫††

## 1. はじめに

ソフトウェアプロセスの分野で支援環境といえば，最近よく目につくようになったPSEE (Process-centered/sensitive Software Engineering Environment；プロセス対応ソフトウェア開発保守環境）である．プロセスの分析と記述を主とする研究のあとを受けて，プロセス実働(enaction）の舞台としての環境がいかにあるべきかを求めている．

しかし，Pがつく前のSEEはプロダクト中心の見方で開発されてきたため，プロセス支援環境が実際のソフトウェア生産に使えるようなものになるためには，今いっそうプロセス中心とプロダクト中心の調和をはかる必要がある．実際，プロセス実働を制御するイベントや条件は，多くの場合，プロセスによって参照，作成されるプロダクトにその源がある．

本稿ではまずPSEE研究の現状を概観したあと，プロセスの定義と実働（人間系の活動）および支援環境（機械系の活動）について種々提案されている参照モデルを分類，整理し，双方の活動に対するサービス間の相互作用について述べる．

## 2. プロセス対応ソフトウェア開発保守環境

PSEEに求められるサービス機能は，最も粗くいえばプロセスの定義と実働の支援である．定義支援の方は，

- ●プロジェクト，チームや単位作業の構成，すなわち，プロセス階層構造（WBS：Work Breakdown Structure)
- ●ツール起動などプリミティブなプロセス要素
- ●各プロセス要素の始動，終了条件
- ●各プロセス要素に対する割込みイベント

などの計画，定義をガイドしその無矛盾性を確認する機能である．手続き型，関数型，ルール型などこれまでに種々研究開発されているプロセス記述言語に従ってサービス機能を考えることができる．

一方，実働支援の方はまだ模索段階であるものが多いようである．通常のプログラムから類推すれば，まず第一に定義記述を機械的に解釈実行するプロセスエンジンの機能ということになる．しかし，もとより実働の主体が常に厳密な振舞いをするとは限らない人間であるから，それだけではある程度のシミュレーションと，自動的に進行できるごく限られた部分に効果を与えられるにすぎない．そこでPSEEに特有の視点や機能として，以下のような項目が検討されている．

（1） プロセスのモニタリング

まずプロセスの定義，実働を，通常のプログラムの作成，実行とは異なる観点から理解しなければならない．すなわち，プロセスの定義とは，個々の作業およびそれらが相互に協調して達成すべきプロジェクト全体の目標を設定，計画したもの，そしてその実働とは，定義記述を文字どおり解釈実行するのではなく，実際のプロセスの進行が計画に沿っているかどうかを監視（モニタリング）し，計画との偏位を少なくするよう制御することと捉える．そのためにまずプロセスの状態とその変化を認識し，作業者に伝達する必要がある．プロセスの状態には，プロセスの実行中，特定の事由による中断，特定のイベントの待ち状態などがある．これに加えて，プロセスを監視しそれを制御できるようにするためには，作業の経過時間（消費コスト）とともに，プロダクトに関する指標（進捗度）の計測が重要である．

† Process Support Environments by Tsuneo AJISAKA (Department of Information Science, Faculty of Engineering, Kyoto University).
†† 京都大学工学部情報工学教室

（2） プロセスの制御とそのポリシー

モニタリングの結果を受けて実働にフィードバックをかける．この制御には，規定を強制しようとする(enforcement)強いものから，選択の幅のあるもの，単に参考のためのガイダンスを与えるものまで，その強さに幅がある．構成管理のような管理的タスクには強制的な強さ，技術的タスクにはガイダンス，というように作業の性質の他，役割や個人的資質によってもそのポリシーを変えられることが望まれる．制御の強さは時間にも依存し，たとえば締切は1カ月前には単なる通知，近づいてくるとより強くはたらき，越えると他の作業を禁止する（規定違反措置に対するポリシーの一例）ような強力なものになる．

（3） プロセスの変更

ソフトウェアプロセスに変更，特に実働中の変更はつきものである．計画—モニタリング—制御の枠内でカバーしきれない事態が発生すると，特定のプロセス要素の順延や中止，履歴情報に基づくロールバック，資源（人員）割当ての変更，あらかじめ定義された救援プロセスの起動など，インスタンスレベルの変更（定義は変更せずその運用による変更）がまず起こる．それでも対処できない場合にはプロセス計画（定義）の変更が動的に必要となる．その場合，すでに実働中のプロセス要素への影響波及を解析したうえ，整合を保ちつつ変更を伝搬させることができなければならない．これは別の見方をすると，定義が部分的で不完全なプロセスを，資源やパラメタと動的に遅延結合して実働させる(実働時にそれらを決定する)ことを許すことになる．このような試練を経た優良なプロセスは再利用の対象になりうる．

以上の項目はプロセスの管理的側面にかなり偏っている．これが実働の主役である人間の目障りになるようではいけないし，さらにPSEEは実際に作業する人間に技術的側面で利便を感じさせるものでなければならない．そのためには，プロダクトと切り離されたプロセスの「実働」ではなく，その結果が種々のプロダクトに反映される影響全体を捉えた「実効」(performance)の支援を考えなければならない．

PSEEのアーキテクチャについては，特に複数のプロセスの同期，協調のためのしくみとして次の三つが考えられる．

- D型：(論理的) 集中データベース方式
- B型：ソフトウェアバス方式
- A型：エージェント方式

データ共有によって同期をとるD型は，プロセスのモニタリングによるデータ記録とデータの重複排除が容易である．一方プロセス間通信によるA型は不完全情報の扱いに有利であるが，イベントの検知や記録のために専用エージェントの介入が必要になる．通信プロトコルを統一するB型はその中間的性質を持つ．しかし，一般に支援環境の設計においては，このようなアーキテクチャからのアプローチに先立ち，環境による支援サービス機能の体系を，アーキテクチャに独立なものとして確立することの方が重要である．

以上，本章の内容は文献1)を参考にしながら整理した．なお，PSEEの文献案内として文献2)がある．

## 3. ソフトウェア開発保守環境におけるツール統合と人間の統合

プロセスから環境への展開が試みられる一方，環境の方もプロセスを取り入れようとしている．ソフトウェア開発保守環境の最近の研究開発は，ECMA TR/55*3)に集約されるように，工程間の連係による効果を重視して，ツールを統合する開放的技術を求めてきた4)．ユーザインタフェースの統合に始まり，ソフトウェアリポジトリにつながるデータ統合，さらにツール間の通信を円滑にする制御統合の必要性が認識され，個々に技術が開発されている．そこへ第4の統合，プロセス統合を導入しようとするのだが，ツール統合との相対的位置付けが明らかでなく，唐突な印象を拭えない．

プロセスを捉えるためには，モデルの要素として，環境，ツールに加え人間を登場させなければならない．この三つの実体の間の関連として統合サービスを配置してみると，**図-1**のようになる**．この図を環境から右に見ると，データ統合

* ECMA (European Computer Manufacturers Association) TR/55 (Technical Report 55).

** この図は統合サービスの概念的位置付けを示したものであって，アーキテクチャを示すものではない．実現にあたって，特にUIサービスは，支援機能が人間と相対するところに必ず求められるから，人間を包むような位置にあると考えた方がよい．

表-1 統合サービスの器と中身

| | 器 | | 中身 | |
|---|---|---|---|---|
| データ統合 | DV | TR/55<br>PCTE<br>IRDS | DC | CDIF, IEEE-P 1175-STL<br>PCTE Common Schema<br>IRDS Content Module |
| 制統統合 | CV | TR/55, BMS, CORBA<br>Software Bus | CC | サービスグラム<br>Software Bus |
| UI 統合 | UV | ウィンドウシステム | UC | 表現情報 |
| プロセス統合 | PV | TR/55 プロセス管理<br>企業/ワークフローモデル | PC | ISO-SLCP<br>PSE |
| CW 統合 | WV | 半構造化メッセージ交換システム | WC | 活動/コミュニケーションモデル |

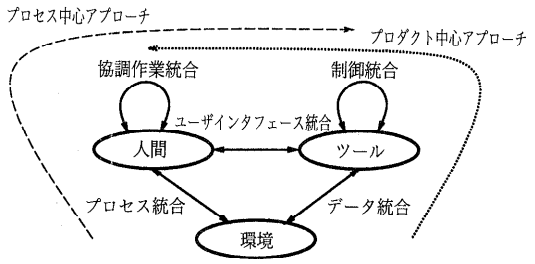


図-1 プロセス支援環境における統合サービス

やデータを扱うツールの統合機能を優先するプロダクト中心の方向性，左に見るとプロセス統合による対人間系の支援機能を優先するプロセス中心のアプローチが見える．さらにこの図の対称性から，第5の統合，協調作業(CW)統合の存在すべきことがうかがえる．

プロセス支援環境が完成度の高いものとなるためには，第4，第5の人間系の統合と，それ以前のツール系の統合が調和よく連係しなければならない．つまり統合機能の統合，メタ統合を実現しなければならない．このような統合機能間の相互作用を明らかにするため，まずは5つの統合機能をその器と中身に分けて詳細に検討してみよう．「中身」サービスとはデータやメッセージの意味的構成をかたち作るもの，「器」サービスはそれを蓄積したり伝送したりする機能である．どの標準，モデル，あるいはシステムがどの種のサービスをカバーしているか，**表-1**にまとめる．

### 3.1 データ統合—器 (DV)

TR/55 のオブジェクト管理サービス群がこのカテゴリーのサービスの概要を与えている．名前，関係，スキーマ，複合構造，バージョン，メタデータなどの定義系サービス，質問，トランザクション，データ導出，アーカイブなどの手続き系サービス，アクセス制御，並行制御，分散同期などの制御系サービスなど，合計21のサービスがあげられている．PCTE(Protable Common Tool Environment)[5),6)]はその一つの実現仕様であって，リンクに重点をおいた拡張E-Rモデルに基づく．IRDS(Information Resource Dictionary System)[7)]のサービス群も同様の機能構成であるが，値指向の関係データモデルの枠が強い．

### 3.2 データ統合—中身 (DC)

このカテゴリーに入るものは，DV サービスによって統合されるデータの意味的定義である．したがってそれらは言語や規則によって記述される*．CDIF(CASE Data Interchange Format)[8),9)]がこの分野の主導的標準である．しかし，その個々のメタモデルは，データフローや状態遷移といった特定の方法論やビューに基づいているため，開発保守の任意の活動をカバーしようとすると，網羅的なメタモデル集合が必要になる．

IEEE タスクフォース P 1175 による CASE ツール相互接続参照モデル試験規格[4),10)]に定義されている STL(Semantic Transfer Language) も，CDIF と同様ツールデータの簡潔な意味的記述を与えるものである．STL 自体はやはり特定の方法論やビューに引きずられているが，この参照モデル前半部で与えている CASE データの主題情報（要求記述，設計記述，プログラムなど）の分類は優れた体系である．すなわち，主題情報から表現情報，局面情報，概念情報，関係情報を分離識別することにより，対象問題の持つ本質的概念

* 中身サービスは一般にこのような対象記述になるが，「サービス」としては，そのような意味的定義を与えたり，参照したりするアクションであると考えればよい．

や関係を，特定の仕様やプログラムの形にするためにバイアスのかかった記述から弁別することができる．リポジトリの内部モデルとしては，このような中立的普遍的意味モデルを持つことにより，変更波及解析や一貫性管理に有利であると考えられる．

ISO/IEC JTC 1/SC 7/WG 11 では，CDIF や STL の成果に加え，PCTE の共通スキーマや IRDS の内容モジュールの開発作業も含めて，これらを調和させ，より優れたモデルを作る努力が続けられている．もしその一部が十分共通的であると認められれば，それは DC の範疇を離れ，構造的記述とともに意味的記述の枠組みを定める DV サービスの一部として固定できることになる．

### 3.3 制御統合—器（CV）

TR/55 のコミュニケーションサービス群がこのカテゴリーのサービスの概要を与えている．ブロードキャストやマルチキャストによるメッセージ伝送，プロセス間通信，リターンコードやデータ共有による制御伝達などがあげられている．HP の BMS(Broadcast Message Server) や OMG(Object Management Group) の CORBA (Common Object Request Broker Architecture) などがすでに製品として実装されている．

### 3.4 制御統合—中身（CC）

CV サービスによって伝送されるイベント，コマンドやメッセージの意味が定義される．ANSI に提案中のメッセージ交換アーキテクチャ[11]に示されている「サービスグラム」はこれをねらっているが，メッセージの型として識別されているのは要求/応答と通知という大枠だけであり，さらに詳細なメッセージ型の検討が必要である．

ESF(Eureka Software Factory) の Software Bus では，各ツールが提供できる，あるいは要求する抽象サービスを，その操作や対象データについて記述できる 3 レベルの言語を用意している[12]．その記述は，TCP/IP 上の RPC(Remote Procedure Call)/XDR(eXternal Data Representation) を用いた Software Bus 独自の CV 機能によって解釈実行される．

### 3.5 ユーザインタフェース統合—器（UV）

X ウィンドウシステムは，このカテゴリーのサービスを提供する現状で最も支配的な実プロダクトであろう．TR/55 の UI サービスの記述でも，例として X が数多く登場する．

### 3.6 ユーザインタフェース統合—中身（UC）

X を含む多くのウィンドウシステムで，UV と UC の分別が明確でない．後者は P 1175 でいう表現情報の定義と操作に徹するべきものであり，一方前者はディスプレイサーバやウィンドウ管理機能を提供するものである．X についていえば，Xlib は明らかに UV に属するが，ツールキットやウィジェットはグレイゾーンにある．

UC と DC の定義もツールの中で渾然となっていることが多い．これはツールのプラットフォームに対する接続性やツール自体の拡張性などを阻害する．環境の実現局面で解決しなければならない問題の一つである．

### 3.7 プロセス統合—器（PV）

プロセスの器とは何か．能動的要素であるプロセスに対しては，受動的要素であるデータやメッセージの場合ほど明らかでない．それはプロセス記述をしまっておく場所ではない．格納された，オンラインでないプロセスは，プロセス記述に特有の DC 定義と組み合わされた DV サービスによって扱われる．

活動状態にあるプロセスに対する器は，そのプロセスを実働させている組織そのものである．したがって，企業モデルや組織内のワークフローモデルが，プロセス支援のためのこの種の統合に有効である．TR/55 のプロセス管理サービス群は，プロセスの計算機支援機構，すなわちプロセスエンジンに要求されるサービスという観点から，このカテゴリーの概要をまとめたものと考えられる．プロセスの定義，実働，可視化，監視，トランザクション，資源に係るサービスが定義されている．

### 3.8 プロセス統合—中身（PC）

プロセス要素の意味的定義がここに収まる．ISO 標準のソフトウェア・ライフサイクルプロセス (SLCP)，およびそれをもとに日本で開発された「システム開発取引の共通フレーム」は，ソフトウェア開発保守の過程におけるプロセス要素あるいは作業項目の標準的集合を与えている[13]．

米国海軍の委員会組織である NGCR* で作ら

* The Project Support Environment Standards Working Group (PSESWG), Next Generation Computer Resources, US Navy.

れたPSEモデル（Reference Model for Project Support Environment）[14]もISO-SLCPとよく似たプロセス要素群を定めている．購入プロセス，供給プロセスという契約の観点はPSEにはなく，またPSEはTR/55の続編という位置付けで各要素を抽象サービスとして記述しているという違いはあるが，開発，保守，管理，支援の各プロセス群の構成は両者にほぼ共通している．

ISO-SLCPやPSEの定義の大部分はPCのカテゴリーに落ちるが，TR/55のプロセス管理サービスにあたるPVサービスの定義も一部含まれていると考えられる．SLCPの第7章—組織に関するライフサイクルプロセスや，PSEの第6節—プロセス管理サービスがそれである．

### 3.9 協調作業統合

協調作業統合の器（WV）と中身（WC）は**図-2**に示すようなOSI風の階層を形成する．下位層（WV）のエンティティは物理的プロセス実働者としての人間，上位層（WC）のエンティティは「役割」である．役割は論理的，仮想的なエンティティであって，個々の役割はPCで定義される個々のプロセス要素に対応する．

役割層では「人間性」はすでに下位の層でフィルタリングされていると考えられる．これは，OSIのトランスポート層が，ネットワーク層の「ノイズ」のないコミュニケーションを提供するのと同様の構成である．たとえば，自然言語で自由に書いたメッセージの部分は人間層だけで流通し，役割層では切り捨てられる．役割層では構造化されたフィールドだけが認識される．人間は複数の異なる役割を同時期に受け持つこともあるので，これら2層間では多重化/逆多重化や結合/分解などのサービスも提供される．

人間性のない役割層ではしたがって，そのプロトコル要素がCCに似たメッセージ型ないしフィールド定義（あるいは発話行為performative）になる．その設計，操作には，プロセスモデルと調和した活動/コミュニケーションモデルが必要である．CCとWCの最も顕著な違いは，WCで出されるすべての要求が必ず他のプロセス要素（役割）の起動と実働を引き起こし，場合によっては新たなプロセス要素を生成することもあるのに対して，CCで出される要求にはそのようなことがありえないことである．たとえば，設計評価の作業中に，設計の変更要求や要求仕様のレビューを求める要求が出たとすると，これはそれぞれ設計あるいは要求分析の作業の再起動を引き起こす．これはすなわち意思決定であるから，人間だけが行えツールにはできないことである．そしてこれがPVによるルーチン的なプロセス解釈実働にダイナミズムを与えるものである．

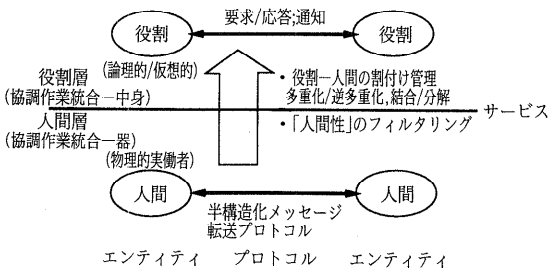


**図-2** 協調作業統合の階層モデル

## 4. プロセス支援環境におけるメタ統合

表-1の10種のサービスを組み立てると，プロセスを統合した包括的環境の構造を設計できる．これらの環境サービスを通常のプログラミングにおけるライブラリ関数のように見なすと，環境内の制御の構造を形作る仕事であるといえる．しかしこれについては解説できる認められた成果が十分あがっている段階になく，一つの概観を示すにとどめる．ここではサービス群の相互作用を認識するのが目的で，アーキテクチャ設計をするのではないので，UIサービスは除いて考える．環境のアーキテクチャ設計においては，UIサービスは人間の目の前に必ず登場し，環境側から見ればいわば人間をカプセル化するような機能である．

プロセス支援環境として最も中心的な相互作用は，プロセスエンジン（PV）によるプロセス記述（PC）の解釈実働と，人間の意思決定による割込み（WC）である．PVを主語にしてこの相互作用を見ると，次のような動作が含まれる．

- ●資源管理：物理的プロセス実働者としての人間やツールとプロセス要素（役割）との結合を管理する．
- ●ルーチン的なプロセス解釈実働：始動条件の評価，下位プロセス要素の起動，終了条件の設定など．
- ●割り込みの検知と処理：これにより実働中のプロセス要素の中断やプロセス定義の変更，新た

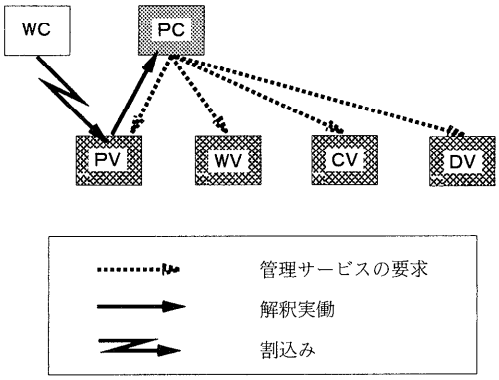


図-3 器サービスを中心とする相互作用

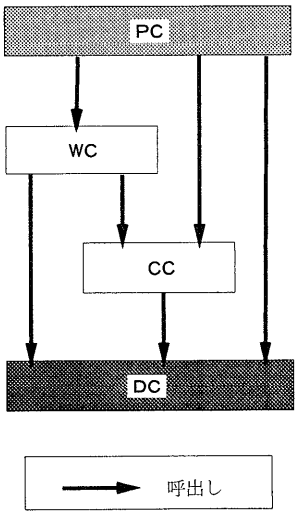


図-4 中身サービスを中心とする相互作用

なプロセス要素の起動などが起こりうる．

中身サービスが主として技術的支援にあたるのに対して，PV の資源管理の例にも見られるように，器サービスには管理機能を支える側面がある．たとえば DV の一つである PCTE では構成管理や変更管理を支援しており，また CV は利用可能なツールの登録，削除などの管理機能を持つ．WV では役割と人間との対応，すなわちエンジニアリングチーム構成が管理されるだろう．以上のような器サービスを中心とする相互作用を**図-3** に示す．

一方，**図-4** は PC に端を発する中身サービスの連鎖を示している．PC で定義される各プロセス要素の内部では，通常のプログラムのように，目的とするアクションの実行や，プロセス内部の論理構造（選択，反復，同期など）を制御するため，他のサービスが呼び出される．たとえば，あるツールに作業プロダクトをブラウズ/エディットするよう要求する CC 機能，ツールデータを用意する DC 機能，レビューなど担当者間のコミュニケーションを可能にする WC 機能などである．また，CC や WC がメッセージの内容を作成するためには DC 機能が必要であるし，WC は CC を呼び出して，協調作業に関連する作業プロダクトを同時に参照させる場面も多いだろう．

## 5. おわりに

ここに示したサービスクラスの分類とそれらの相互作用は，プロセス支援環境設計のための一つの指針（ガイドマップ）になることを期待している．特に，種々の支援機能を設計するにあたり，不必要なサービスの重複を避けるのに役立つと思われる．最初から完全な統合環境を形作るのは望めないことであるから，必要なサービスを個別に開発し環境に逐次差し込むことによって，既存の機能に影響を及ぼすことなく，包括的環境全体を段階的に構築していくことが可能でなければならない．（連合的環境[15]）

各サービスクラスについて現在参照できる標準やモデルだけでは，さらに詳細な相互作用規則を定めることは難しい．これはクラス間のサービスの粒度に大きな差があるからである．現状の PC や WC のサービス定義は，ツールフレームワーク（データ，制御，UI 統合）サービスを直接利用するにはあまりに粗い．

この問題にあたるアプローチは二つ考えられる．一つはプロセス系サービスの内容を詳細化することである．その骨格として，アクションやメッセージの型階層を定める「タスクオントロジー」の体系が必要と思われる．器についてはある程度満足なサービスが用意できているので，より「中身指向の」[16]ソフトウェア工学が求められる．

もう一つは，プロダクトの側からのアプローチである．中ぐらいの粒度のプロセス要素にも対応できるよう，広い範囲の抽象度，粒度をカバーでき，かつ特定の方法論に依存しない中立的統一的プロダクトモデルを求めることによって，粒度に関してシームレスな環境をめざす．

## 参考文献


1) Penedo, M. H. and Riddle, W.: Process-sensitive SEE Architecture (PSEEA) Workshop Summary, ACM SIGSOFT Softw. Eng. Notes, Vol. 18, No. 3, pp. 78-94 (1993).
2) Garg, P. K. and Jazayeri, M.: Selected, Annotated Bibliography on Process-Centered Software Engineering Environments, ACM SIGSOFT Softw. Eng. Notes, Vol. 19, No. 2, pp. 18-21 (1994).
3) ECMA and NIST: Reference Model for Frameworks of Software Engineering Environments, Draft Edition 3 of Technical Report ECMA TR/55 and NIST Special Publication 500-201 (1993).
4) 鯵坂恒夫：開放型CASEプラットフォーム，コンピュータソフトウェア，Vol. 10, No. 2, pp. 4-12 (1993).
5) ISO/IEC: 13719 Portable Common Tool Environment (PCTE) Part 1 Abstract Specification (1995).
6) 鯵坂恒夫，沢田篤史，満田成紀：Emeraude PCTE，コンピュータソフトウェア，Vol. 10, No. 2, pp. 65-77 (1993).
7) ISO/IEC: 10027 Information Resource Dictionary System (IRDS) Framework (1990); 10728 Information Resource Dictionary System (IRDS) Service Interface (1992).
8) EIA: Interim Standards 81-83, CASE Data Interchange Format (1991).
9) 篠木裕二，西尾高典，吉川彰弘：CDIF-CASEデータ交換形式，コンピュータソフトウェア，Vol. 10, No. 2, pp. 13-25 (1993).
10) IEEE Computer Society's Task Force on Professional Computing Tools: A Standard Reference Model for Computing System Tool Interconnections, P1175 Trial-Use Standard (1991).
11) Michaels, K.: Defining an Architecture for Control Integration, Proc. Software Engineering Environments Conference (SEE'93), pp. 63-71, IEEE Press (1993).
12) 鯵坂恒夫：ESF (Eureka Software Factory) プロジェクトの活動状況について，コンピュータソフトウェア，Vol. 11, No. 2, pp. 48-53 (1994).
13) 村上憲稔：ソフトウェアライフサイクルプロセス，Vol. 36, No. 5, pp. 421-430 (1995).
14) NGCR: Reference Model for Project Support Environment, Version 2.0 (1993).
15) Brown, A. W., Morris, E. J., Zarrella, P. F., Long, F. W. and Caldwell, W. M.: Experiences with a Federated Environment Testbed, Sommerville, I. and Paul, M. (eds.), Software Engineering—ESEC'93, Lecture Notes in Computer Science, pp. 175-196, Springer-Verlag (1993).
16) 溝口理一郎：知識の共有と再利用研究の現状と動向，人工知能学会誌，Vol. 9, No. 1, pp. 3-9 (1994).


(平成6年9月19受付)

**鯵坂 恒夫**（正会員）

1956年生．1980年京都大学理学部物理学系卒業．1985年同大学院工学研究科情報工学専攻博士課程研究指導認定退学．同年京都大学工学部情報工学科助手．1990年より同助教授．工学博士．ソフトウェア工学，データ工学，情報システム工学に関する研究に従事．ソフトウェア設計自動化，プログラム自動生成などの研究を経て，最近はとくに開放型CASE環境やソフトウェアプロセスに関する研究を行っている．93年度文部省在外研究員として英国Durham大学計算機科学科に滞在．日本ソフトウェア科学会，システム制御情報学会各会員．