解　説　組込みシステム開発の現状

2. 組込みシステムの設計手法—オブジェクト指向設計を中心にして—

Design Methodology for Embedded Systems Software—Focusing on Object-Oriented Design Approach—by Akira KAWAGUCHI, Tomoji KISHI and Hiroshi MONDEN (NEC Microcomputer Software Laboratories).

川口　晃[1]　岸　知二[1]　門田　浩[1]

1 NECマイコンソフト開発環境研究所

1. はじめに

近年，マイコン組込みシステムの高機能，短納期化が進んでおり，その上に組み込まれるソフトウェアにも同様なことがいえる．一方，ビジネスソフトウェアなどの分野では再利用性，拡張性に優れたソフトウェア構築のための方法論が重要視されてきている．本稿ではこれらの現状を踏まえ，組込みソフトウェアの特徴を意識しながら，上流，中流設計方法論の解説を行う．とくにオブジェクト指向設計方法論については組込みシステムへの適用可能性を論じるとともに，リアルタイムOS（以下RTOSと表記する）を用いた8ビットマイコンシステム設計に応用した事例を紹介する．組込みシステムの特性を考慮して方法論を適用し，効率的なタスク分割を行えば，8ビットマイコンシステムであっても実用的な性能とコードサイズが得られたことを述べる．

2. 組込みシステムの特徴

組込みシステムといっても，数千行のアセンブラで記述されるコンパクトなものから，本格的なOS上に数十万行の高級言語で記述される大規模なものまで多種多様である．しかし通常組込みシステムを議論する際には，以下のようなソフトウェア面の特徴を備えていると考えることが多い．

（1）　実時間性

組込みシステムはハードウェアを制御したり，さらにそれを通して外部の環境を操作するものが多い．こうした自然法則の支配する世界との接点をもつシステムにおいては，相対的に早い遅いといった議論では不十分であり，何秒，何ミリ秒といった実時間との関係が問われる．適切な実時間で外界とやりとりができなければ，炊飯器も，ロボットも，化学プラントも正しく制御することは不可能である．

しかしながら実時間制約を満たすようにシステムを設計することは容易でない．実時間を予想するために各命令の実行時間を仮定しそれを累積するなどの方法がとられることもあるが，入出力などに関係する処理時間は見積もりが難しい．さらにシステムが複雑になるとソフトウェアのマクロな振舞いが予想しづらくなる．たとえば複数の処理が割込みで起動される状況を考えると，特定の処理が連続的に実行されるのか，中断されるのかなどによって，処理時間が大きく変わってくる．したがってミクロな処理時間の積み上げだけでなく，マクロな振舞いが制御できなければ実時間性を保証することができなくなる．

（2）　リアクティブ性

組込みシステムは，外部環境からの何らかの事象に対してアクションを起こすという形で実現されるものが多い．ここで事象とはシステムの振舞いに影響を与える外界のできごとであり，どのような事象が起こったら何を行うという形で振舞いが定義されるシステムをリアクティブなシステムと呼ぶ．

こうしたリアクティブなシステムでは，内部に状態を保持する必要があることが多い．たとえば電話機の受話器を取り上げてから，番号ボタンを押すという手順を処理するためには，受話器が取り上げられた時点でその状態を内部に覚えておき，次にボタンが押された時点で内部に覚えている状態と照らし合わせて振舞いを決定する．こうしたときによく用いられるのがオートマトン的な状態遷移を管理するという方法である．こうした状態遷移の設計，とくに複数のタスクがそれぞれ状態をもって協調しあうシステムの設計には高度

なスキルが求められる．

（3） 厳しい実装環境

組込みシステムはハードウェアと一体化されるため，そのコストはハードウェアとソフトウェアの両方のコストで決定される．ソフトウェアだけを考えればゆとりのあるCPUやメモリを利用した方が開発しやすくコストダウンにつながることもあるが，ハードウェア的には少しでも安い部品を使うことが有利であり，とくに量産品などでは余裕のない環境でソフトウェアを実現することが求められる．そのため性能を確保したり，オブジェクトサイズを削減したりという努力が不可欠となる．こうした実装環境のゆとりのなさが開発の方法を制約し，特徴づけている．

3. 分析・設計手法の現状

ソフトウェアに求められる要求を整理し，ソフトウェアが果たすべきサービスを定義する作業が分析であり，それをどのようなしかけで実現するかを決める作業が設計である．こうした分析や設計をどのような考え方で進めればよいのか，分析や設計の結果をどのように記述すればよいのかなどを示すものが分析・設計手法である．

分析・設計手法として有名なものとして，1970年代半ばに提案された構造化手法[1]があげられる．構造化手法は機能を階層的に分割するという考え方と，その分割結果をデータフロー図によって記述するという点に特徴がある．この手法は本来ビジネスアプリケーションを対象としたものだったが，1980年代になってそれをリアルタイムのシステム向けに拡張したリアルタイム構造化分析[2]が提案されている．

こうした方法論とは別に，エンジニアリングの分野では状態遷移設計の手法や，タスク設計などの手法[3]も提案されている．またこれらの手法に沿った開発を支援するツール群も多く存在する．組込みシステムの分野では状態遷移設計を支援するツールなどが比較的多く使われている[4]．

4. オブジェクト指向の適用

ソフトウェア分析・設計の手法として近年注目を集めているのがオブジェクト指向手法である．とくに1991年のOMT法[5]の発表をひとつの契機として広く知られるようになった．近年さまざまなソフトウェアがオブジェクト指向の技術で提供されるようになっており，こうした手法の適用も増加している．

オブジェクト指向での分析・設計においては，将来の拡張や再利用の範囲を想定しながらそのソフトウェアの備えるべき骨格を見いだし，それをソフトウェアの基本構造に反映するように開発を進める．

しかしながら組込みシステムの領域では実時間性が求められたり，厳しい実装環境でソフトウェアを実現することが求められているため，単に美しい論理構造を実現するというだけでは不十分であり，求められる性能やサイズを確実に達成できなければならない．オブジェクト指向の手法がもつメリットを生かしつつ，厳しい実装制約を達成できるかどうかが，この領域に対してオブジェクト指向の手法が適用できるかどうかの1つの鍵となる．

5. オブジェクト指向での開発事例

本章では，コードレスホンシステムの組込みソフトウェアに対してプロトタイピングを行った事例の紹介を行う．本プロトタイピングでは，ユースケースなどの一般的なオブジェクト指向分析手法をベースに，筆者らが研究中の組込みシステムの実装制約に対処するための設計手法を適用した．ターゲットシステムの概要を図-1(a)に示す．

コードレスホンシステムのプロトタイピング

（a）ターゲットシステム

- ○CPU 8bit, 5MHz
- ○メモリ ROM:60KB, RAM:2KB
- ○頻繁な機能拡張，再利用
- ○多機能
- ○リアルタイムOSを使用

（b）実装結果

- ○実装した4種類のユースケースは全て正常動作
- ○外線着信ユースケースの場合
  - μトランザクション 8
  - オブジェクト 12
  - タスク 11→7（タスクのマージ後）
- ○RAMサイズ
  - 製品版の6倍→1.1倍（タスクのマージ後）
- ○ROMサイズ
  - 製品版の1.6倍→1.1倍（タスクのマージ後）

図-1 プロトタイピングの概要

5.1 分　　析

本プロトタイピングでは基本的に，OOSE[6]に従って分析を行った．

●分析手法

まず要求分析では，要求仕様を表現するユースケースを定義した．ユースケースは，システム外部にあるもの（アクタと呼ぶ）からのイベントに対して，システムがどのように振る舞うかを文章でまとめたものである．またシステムに関連するオブジェクトを見出し，それらの静的な関係を整理してユースケースごとに分析モデルを作成した．

●分析モデル

図-2(a)は無線送信ユースケース（親機，子機端末間で無線を用いて音声以外のコマンド，ステータスの送信を行う．）の分析モデルを示している．ここでPLL，RFはそれぞれ電波の位相同期，発信を行うオブジェクトである．分析モデル

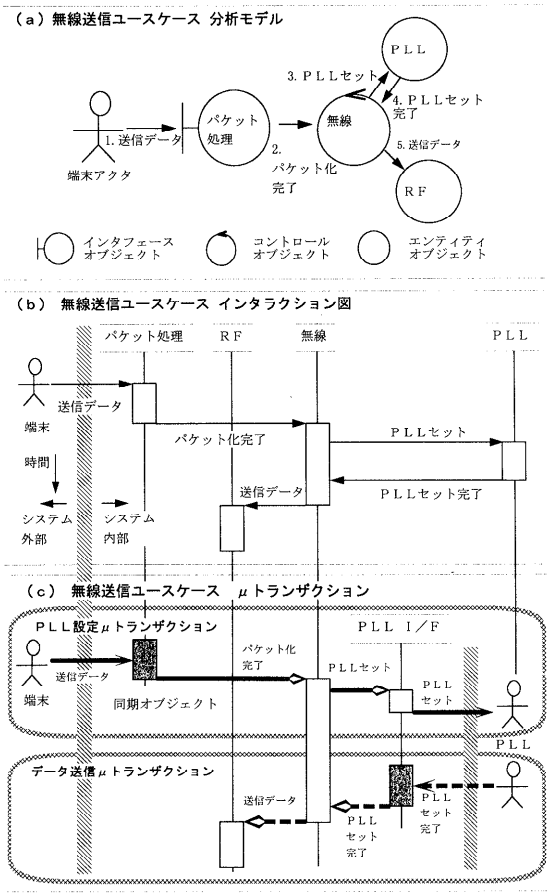


図-2　アクタとμトランザクションの抽出

では，ユースケースの処理をインタフェース，コントロール，エンティティの各オブジェクトに機能ごとにまとめ，親機，子機を端末として一般化するなど，再利用性，拡張容易性に対する工夫を盛り込んでいる．

次にシステムの振舞いをより明確に示すため，オブジェクト間のメッセージのやりとりを時間を追って規定したインタラクション図，**図-2**(b)を作成した．

●分析結果の組込みシステムへの適用

組込みシステムでは実時間性を要求されることが一般的だが，図-2(a)，(b)に示す分析結果は，オブジェクトの動作タイミング，動作時間など実時間性にかかわる部分の分析が目的ではなく，そのまま設計に用いるのは難しかった．またシステムの実時間性はその実現方法に左右されることも多く，実現方法が確定しない分析段階ではシステムの実時間性を完全に把握することは難しい．

5.2 ソフトウェア設計

前節で述べたように，分析モデルそのままでは組込みシステムの設計に適用することが難しかった．そこで本プロトタイピングでは，分析モデルを出発点としてシステムの実現方法の検討を開始し，その過程で次第に明らかになってくるオブジェクトの動作タイミングや動作時間などの実時間性に関する情報に基づいて，実時間性を扱うために適切な設計単位を求めていく手法を適用した．この設計単位をここではマイクロトランザクションと呼ぶ．（以下μトランザクションと表記する．）本節ではユースケースの実時間性について詳しく検討し，μトランザクションを用いた設計例を紹介する．

●アクタとのインタフェース

まず端末アクタがデータを送信する場面を調べる．この場面で，端末アクタはシステムと非同期に動作するのでシステムはデータ送信のタイミングを確定できない．そこでシステムはこの場面に切り替わるタイミングでパケット処理オブジェクトを活性化して送信データを待ち受ける必要がある．ここではこのパケット処理オブジェクトのように，各々の場面でアクタからのイベントを待ち受けるオブジェクトを同期オブジェクトと呼ぶ．同期オブジェクトの待ち受け時間は通常無視できない．（結果として同期オブジェクトには並行性

が要求される場合が多い.) このようにシステムに対して非同期に動作するアクタとのインタフェースは，実時間性に関して重要な設計ポイントになる．そこで本プロトタイピングの設計では，新たにシステムに対して非同期なアクタとして扱うべきオブジェクトがないかに着目して設計を進めた．

●μトランザクション

たとえば上記の視点で無線送信ユースケースの設計を進めるうちに，PLLを実現するハードウェアの処理時間がソフトウェアの動作上無視できないことがわかってきた．仮りにPLLセットコマンドを送出後，セット完了をただ漫然と待つ設計の場合，ほかのアクタからのイベントを読みこぼしたり，ほかの計算処理が不必要に遅れたりする．そこでPLLの動作がシステムに対して非同期だとみなし，PLLを新たなアクタと考えて，前項の端末アクタと同様に実時間性に関して慎重な設計を行うことにした．

この結果，PLLからのセット完了イベントを待ち受けるPLLインタフェース（PLL I/F）が同期オブジェクトとして追加され，無線送信ユースケースが**図-2**(c)の上部の実線で表される部分と，下部の鎖線で表される部分に分割された．このユースケースを分割してできる，アクタからのイベントとそれに同期して続くクラス間のメッセージの連鎖をμトランザクションと呼んでいる．μトランザクションへの分割は，設計が進みシステムに対して非同期とみなすべきオブジェクトが現れるたびに行われる．

図-3にPLLをアクタとみなして無線送信ユースケースを分割したPLL設定，データ送信のμトランザクションを示す．ともにアクタが発生するイベントを受信する同期オブジェクトと，イベントに同期したシステムの振舞いを定義する同期インタラクションからなる．

このμトランザクションは，設計を進める過程で明らかになった実時間性に関する以下のような情報の組合せを規定している．

○アクタ，およびアクタとみなすオブジェクト．
○アクタ，およびアクタとみなすオブジェクトからのイベントを待ち受ける同期オブジェクト．
○アクタからのイベントに同期する同期インタラクション．

一方，μトランザクションは，分析段階で施された再利用性，拡張容易性などに関する工夫を反映している．

●μトランザクションを抽出するヒント

今回のプロトタイピングでわかったμトランザクション抽出のためのヒントを下記に示す．

○ユースケースに複数のアクタが関係する時には，それらがお互いに非同期に振る舞うので，複数のμトランザクションに分割される．
○アクタが複数の非同期なイベントを発生する時（たとえばコードレスホンシステムでは，電話回線アクタは外線着信と外線着信終了イベントをお互いに非同期なタイミングで発生する）には複数のμトランザクションに分割される．
○ハードウェアに何らかの制御を行ったり，演算時間の予測できない計算を行うなどシステムに対して非同期とみなすべき処理を行う場合，その部分でμトランザクションに分割される．

なおμトランザクションへの分割方法は，分析結果から一意に定まるのではなく，システムの実現方法に依存している．したがって設計が進むに従って分割が進み，新たなμトランザクションが定まることとなる．

●アクタの振舞いとコーディネータの設計

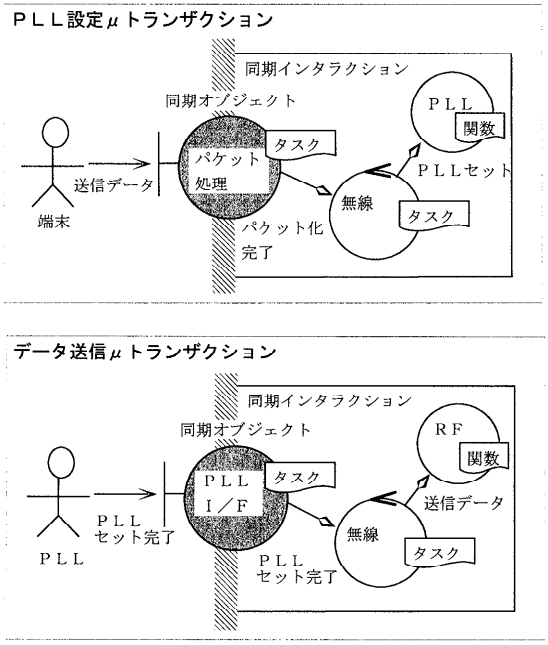


図-3 μトランザクションとタスクマッピング

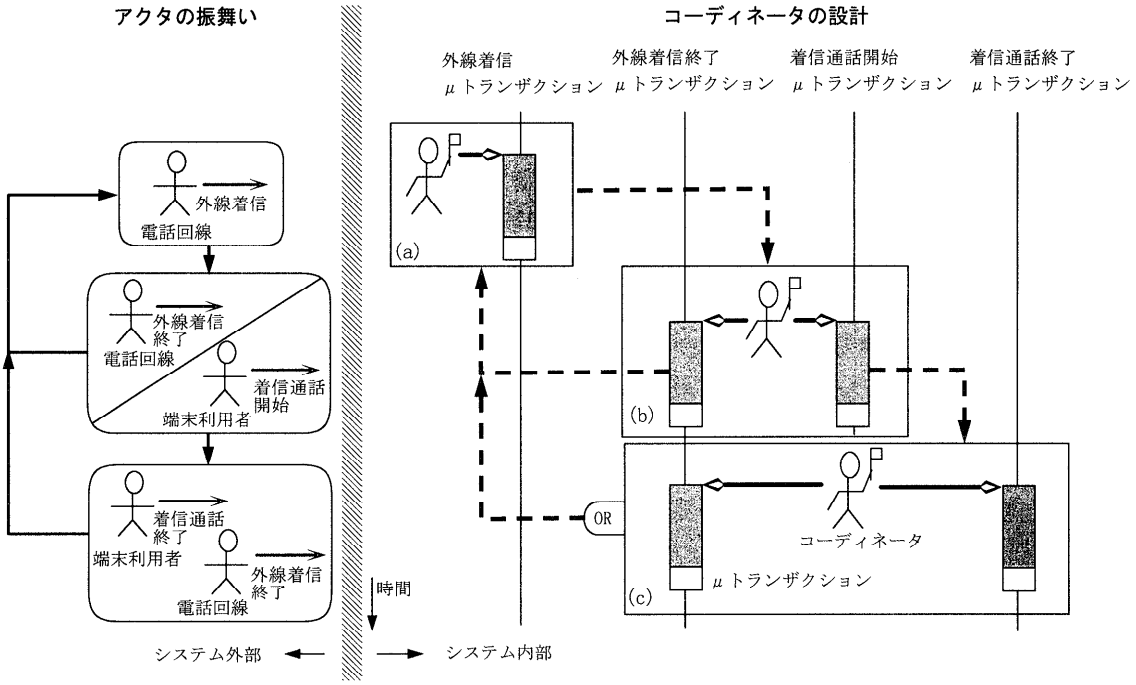


図-4 アクタの振舞いとコーディネータの設計

μトランザクションが抽出されれば，それらの関係を再整理し，どのようにμトランザクションを活性化するかを設計する．ここでアクタの振舞いとともに次々に移り変わる場面に応じて適切なμトランザクションを活性化するものをコーディネータと呼ぶ．今回はμトランザクションがアクタのイベント単位で定まる点を利用し，どのようにμトランザクションを活性化すればシステムに要求される振舞いを実現できるかを考えてコーディネータを設計した．図-4の左側の部分は外線着信ユースケースのアクタの振舞いを示し，右側はこれに対するコーディネータの設計を示している．たとえば外線着信イベントの発生後にシステムに要求される振舞いは，電話回線，端末利用者の各アクタの振舞い（外線着信終了あるいは着信通話開始イベント）に応じて適切な状態に移行することである．このシステムの振舞いを実現するようにコーディネータの設計を行った．なおプロトタイピングを行う過程で，コーディネータの設計には何種類かの標準的な設計（たとえば図-4(a)，(b)，(c)の部分）が存在することがわかった．本プロトタイピングのコーディネータの大部分は，このような標準の設計から最適なものを選択し組み合わせることで設計されている．

5.3 ソフトウェア実装

今回の実装ではRTOSを用いてできるだけ一般的な実装を心掛けた．RTOSを用いた実装ではタスクマッピング，タスク実行構造（タスク起動方法，タスク優先度など），タスク間通信などの実装を行う必要がある．ここではタスクマッピング，タスク実行構造の実装について紹介する．

●μトランザクションによるタスクマッピング

上流の分析結果から，直接タスクマッピングを行うのはスキルを必要とする作業であるが，今回はユースケースからμトランザクションを導き出すことで，μトランザクションがもつ実時間性に関する情報を利用できた．図-3にPLL設定μトランザクション，データ送信μトランザクションのタスクマッピングを示す．2個の同期オブジェクト（パケット処理，PLL I/F）は，必要な場面および受信するイベントの実時間特性が異なるので独立したタスクとした．また無線オブジェクトもタスクとした．これは無線オブジェクトはコントロールオブジェクトなのでタスクにしておく方が将来の拡張性が高いと判断したからである．またPLL，RFはそれぞれパケット化完了，

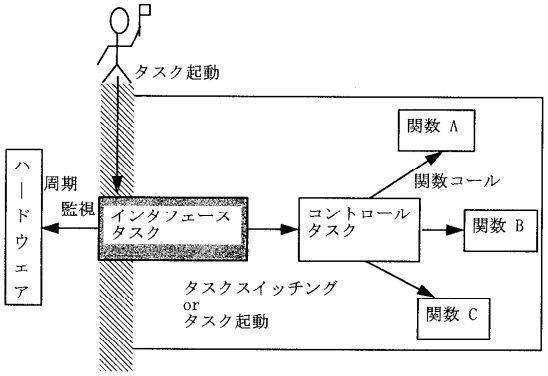


図-5 μトランザクションの実装例

PLLセット完了メッセージに同期してコールされる関数とした．

さらに実装を行う過程で，今回のプロトタイピングが問題なく動作するレベルで実時間性を満足する標準的なμトランザクションの実装があることがわかった．**図-5**に標準的な実装の例を示す．これは，同期オブジェクトにインタフェースタスクによるイベントの周期監視，同期インタラクションにコントロールタスクからの関数コールを用いた例である．今回のプロトタイピングでは，ほとんどのμトランザクションをこのような標準的な実装で実現している．

＊メモリ制約

このようにタスクマッピングしたソフトウェアをシステムに組み込んだが，メモリ容量が不足して実装できなかった．これは従来の製品版ソフトウェアに比べてタスク数が増加し，RTOSが使用するタスク管理用スタック領域が増加したためである．そこで以下のような簡単なルールにしたがってタスクのマージを行った．なおマージは手作業で行ったが，下記のルールに明らかに従うタスクに限定したため容易に行うことができた．

○同周期で実行されている，あるいは実行しても支障のないタスク同士をマージする．

○同タイミングで実行されている，あるいは実行しても支障のないタスク同士をマージする．

○ただしマージ後の実行時間，タスク優先度などに問題のないタスクにかぎる．

この結果，メモリ使用量を製品版ソフトウェアの10%増しに抑えられ実装が可能になった．

●コーディネータの実装

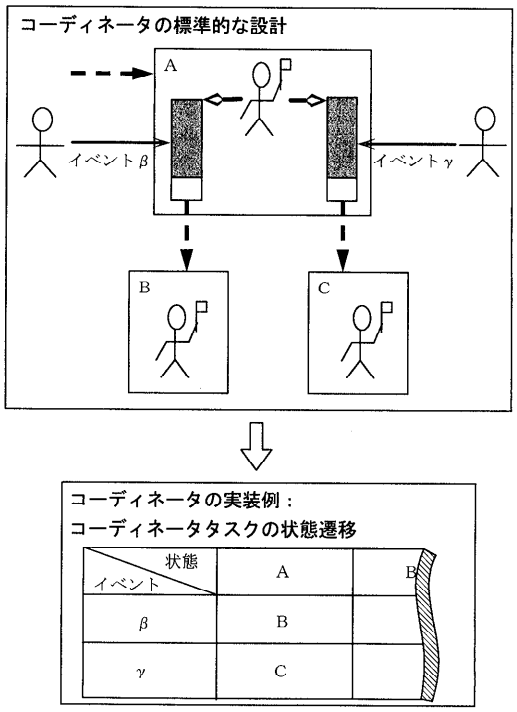


図-6 コーディネータの実装例

コーディネータの実装は，コーディネータの標準的な設計に応じた標準的な実装で対処できた．たとえば**図-6**は，コーディネータによるイベント受信後の状態の移行をユースケース単位で用意されるコーディネータタスク内の状態遷移の形で実装する例である．今回のプロトタイピングではコーディネータの大部分をこのような標準的な実装を用いて実現し，実時間性を満足することができた．

5.4 プロトタイピングのまとめ

以上の手法を用いてコードレスホンシステムのプロトタイピングを行った実装結果を**図-1**(b)に示す．またこのプロトタイピングで次のようなことがわかった．

●上流のオブジェクト指向分析で得られた論理構造を，比較的開発者のスキルに依存しない手法を用いて，実時間性やメモリ制約などの実装環境へ適応させながら，設計，実装に反映することができた．

●アクタからのイベントとそれに同期して続くクラス間のメッセージの連鎖であるμトランザクションを設計の基本単位と考え，プロトタイピングの設計をμトランザクションの抽出と，

μトランザクションのコーディネータの設計に系統立てると，アクタの振舞いとμトランザクションのもつ実時間性に関する情報をタスクマッピング，タスク実行構造の設計に活用できた．またこの枠組みに則った標準的な設計，実装を行うと，設計段階で実装コードの特性をある程度見通すことができた．

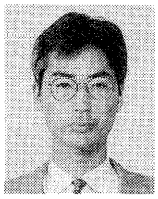
6. おわりに

以上，本稿では，組込みソフトウェアの特徴と上流，中流設計方法論の概要を述べるとともに，オブジェクト指向設計方法論を組込みソフトウェアへ適用した事例を紹介した．

参考文献

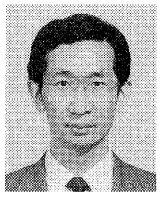
1) Yourdon, E. and Constantine, L. L.: Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design, Prentice-Hall (1975).
邦訳：原田 実，久保未沙共訳：ソフトウェアの構造化設計法，日本コンピュータ協会 (1986).
2) Hatley, D. J. and Pirbhai, I. A.: Strategies for Real-Time System Specification, Dorset House (1988).
邦訳：立田種宏監訳：リアルタイム・システムの構造化分析，日経BP社 (1989).
3) Gomaa, H.: Software Design Methods for Concurrent and Real-Time Systems, Addison-Wesley (1993).
4) 沼田健治，奥村晃子：マイコンソフトウェア用統合CASE環境，NEC技報，Vol. 50, No. 3 (1997).
5) Rumbaugh, J. et al.: Object-Oriented Modeling and Design, Prentice-Hall (1991).
邦訳：羽生田栄一監訳：オブジェクト指向方法論OMT，トッパン (1992).
6) Jacobson, I. et al.: Object-Oriented Software Engineering, Addison-Wesley (1992).
邦訳：西岡利博，渡邊克宏，梶原清彦監訳：オブジェクト指向ソフトウエア工学OOSE，トッパン (1995).

(平成9年8月11日受付)



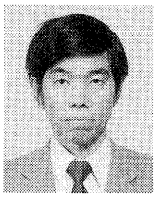
川口 晃（正会員）

1987年大阪府立大学大学院工学研究科航空工学専攻修士課程修了．同年NEC入社．衛星通信用アンテナ制御装置の開発業務に従事，94年より現職．リアルタイム組込みソフトウェアのオブジェクト指向開発技法，ソフトウェアアーキテクチャの研究開発と，ソフトウェア開発部門への支援業務に従事．現在，NECマイコンソフト開発環境研究所開発プロセス技術部主任．



岸 知二（正会員）

1982年京都大学大学院工学研究科情報工学専攻修士課程修了．同年NEC入社．1986年，米国カーネギーメロン大学客員研究員．オブジェクト指向開発技法，ソフトウェアアーキテクチャ，構成管理，ソフトウェア開発環境などの研究開発と，ソフトウェア開発部門への支援業務に従事．現在，NECマイコンソフト開発環境研究所開発プロセス技術部技術課長．IEEE CS会員．



門田 浩（正会員）

1973年京都大学大学院工学研究科電子工学専攻修士課程修了．同年NEC入社．組込みシステム開発の基本技術，方法論の研究開発に従事．現在，NECマイコンソフト開発環境研究所所長．電子情報通信学会会員．