ソフトウェア工学 82－3
（1991. 12. 3）

プロトタイピングによるマイコンソフトウェア開発手法

五十嵐真悟　西村朋子
株式会社東芝 システム・ソフトウェア技術研究所

近年、複雑化・高機能化の一途を辿る組込み用マイコンソフトの生産性向上が望まれている。これらのソフト開発は性能・リソースの制約が大きい、あるいは設計変更などに伴う後戻りが多いなどの問題点が存在する。これに対し我々は、開発の上流工程においてよりよいソフトウェア・アーキテクチャを設計し、これに沿って開発を進めることが重要であると考えソフトウェアの方式設計の評価を主眼としたプロトタイピング導入したマイコンソフト開発手法を提案する。本発表では、この開発手法の概要と実システムに対する適用結果について報告する。

DEVELOPMENT OF EMBEDDED MICRO COMPUTER SOFTWARE WITH PROTOTYPING

Masato Igarashi　Tomoko Nishimura
Systems & Software Engineering Laboratory,TOSHIBA Corporation
70,Yanagi-cho,Saiwai-ku,Kawasaki 210,Japan

Over the last few years, there has been a trend toward improving the productivity of embedded micro computer software. There are many problems in developing this type of software such as severe restrictions on performance and resources and frequent backtracking due to design modifications. Despite these problems, we think it is important to improve the method of designing software architecture in the upper phase of development. We propose a design method for embedded micro computer with prototyping. In this article, we describe the outline of our design method and report the result of applying this method to a real system.

1 はじめに

近年、組込み型マイコンの適用範囲は拡大の一途をたどり、特に家電製品などの分野では多くの製品にマイコンが組み込まれている。

一方、家電製品のような組込み型マイコン用ソフトウェア開発には、一般のアプリケーションソフトウェア開発とは異なった以下のような特徴を持つと言える。

(1) MPU の制約
コスト面の問題から 4～8bit クラスの低位のマイコンを用いる場合が多く、MPU の性能やメモリ容量などの制約が厳しい。

(2) リアルタイム処理
対象システムの性質上、リアルタイム処理が必要となる。

(3) ハードウェアが必要 ソフトウェアの動作を確認するために、テスト用のハードウェアが必要である。

(4) 頻繁な仕様変更
ハードウェアを含めて新規開発となる場合が多く、ハードウェアの仕様変更などが頻繁に行なわれる。

(5) 小規模な開発
数人月程度の開発規模の小さなものがほとんどである。

このような特徴の元でのソフトウェア開発は、生産性の向上を図る上で以下のような点が問題となる。

I. MPU の厳しい制約下の元でリアルタイム処理を行なおうとするため、 ソフトウェアの構造が複雑になり、再利用が困難である。

II. テスト用のハードウェアの開発スケジュールに、ソフトウェアの開発スケジュールが依存してしまう。

III. ハードウェアの仕様変更が頻繁に行なわれるため、工程の後戻りが多くなり、開発効率が悪化する。

IV. 小規模な開発であるために、開発者個人の能力に依存した開発になりがちであり、 開発の分業化がなされていない。

我々は、以上のような問題の解決手段として、マイコンソフトウェア開発の上流工程に注力することが重要であると考える。つまり、与えられた要求をマイコンの持つ制約下で実現するために、設計の段階で十分に検討を重ねた後に下流工程に進むことにより、下流工程での設計変更などに伴う工程の後戻りなど開発効率悪化の原因を抑制することが可能となる。プロトタイプソフトウェア作成のコストを考えた場合、基本設計レベルでのプロトタイピングが開発の効率化に最も効果的であると考えられる。
このような理由により、我々は組込み型マイコン用ソフト開発の基本設計の段階にプロトタイピングを導入した開発手法を提案する [1]。

2 設計型プロトタイピング

2.1 プロトタイピングによる問題解決

我々の提案する設計型プロトタイピング手法は、開発の上流においてプロトタイプソフトウェアを作成し、これを評価することで前述の問題を解決しようというものである。マイコンソフトウェアの基本設計のレベルでプロトタイプソフトウェアを開発し、これを動作させることにより、ソフトウェアに対する要求を確認し、開発下流での仕様変更などにともなう工程の後戻りを抑制することが可能となる。

また、プロトタイプソフトウェアの段階では、テスト用のハードウェアを用いずに動作の確認が可能であるため、テスト用ハードウェアの完成を待つことなく、ソフトウェアの開発を進めることができる。

また、迅速かつ容易にプログラム作成を行なうことが可能な環境でプロトタイプソフトウェアを開発することにより、与えられた要求をマイコンの持つ制約下で実現しつつ再利用が可能な設計を、十分に検討した後に下流工程に進むことを容易に行なうことが可能である。

このとき、この設計に基づきターゲットソフトウェアの部品化を行なえば、部品毎に開発の分業化を行なうことが可能となる。

さらに、プロトタイプソフトウェアにおいてハードウェアを隠蔽するような設計を行なうことで、ハードウェアの変更への対応が容易になる。
このようにプロトタイピングを行なうことで、第1節で述べたようなI～IV の問題点を解決することが可能となる。

また、プロトタイプソフトウェア開発の工数を考えた場合、基本設計レベルでのプロトタイピング

が開発の効率化にもっとも効果的であると考えられる。

以上のような理由により、我々は組込みマイコン用ソフトウェア開発の基本設計の段階にプロトタイピングを導入した開発手法を提案する。ソフトウェア開発におけるプロトタイピング手法とは、ソフトウェアの要求定義の段階で用いられるだけではなく、ソフトウェアシステムの設計段階でも有効であると考えられている [2] が、家電製品等の小規模なマイコンソフトウェアの設計にプロトタイピングを応用した例はあまり見られない。

2.2 プロトタイプソフトウェアの実現

我々が提案する手法は、プロトタイプソフトウェアとターゲットとなる組込みマイコン用ソフトウェアの双方を同様の内部構造を持つように作成し、プロトタイプソフトウェアの設計仕様を通して、ターゲットソフトウェアの設計仕様を評価しようというものである。

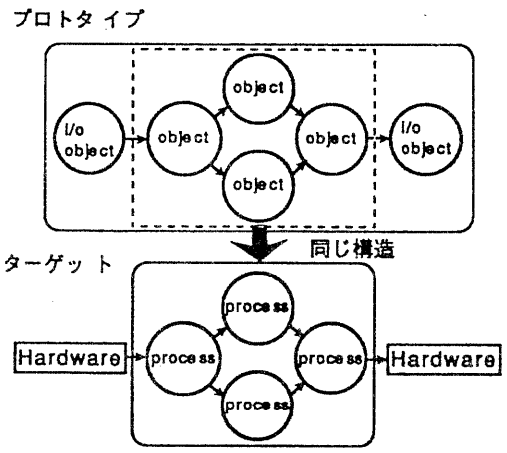

図 1: プロトタイプとターゲットの構造

また、プロトタイプソフトウェアは、外部仕様の確認などを行なうためにも、ターゲットのシステムの入出力機器も含めた全体をシミュレートするようにも作成する。

この手法は、以下のような流れでマイコンソフトウェア開発を行う。

1. プロセスの分割などの基本設計を行う。
2. 1. にも基づき、プロトタイピング用高級言語を用いて、プロトタイプソフトウェアを迅速にも開発する。
3. 2. のプロトタイプを動作させ、システム全体のシミュレーションを行なうことにより、外部仕様の確認および基本設計の評価を行う。
4. 設計にも問題があれば、これを修正し、1. にもフィードバックする。
5. 1. ～4. のループを繰り返し、適切な仕様を得たところでターゲットソフトウェアの開発にも移る。

本手法では、プロトタイピング用高級言語として Smalltalk を採用している。Smalltalk はオブジェクト指向プログラミングシステムであり、データの抽象化や継承の概念を持ち、また、豊富なクラスライブラリや優れたユーザ・インタフェースを持つため、迅速かつ容易にもプロトタイプソフトウェアを作成することが可能である。

プロトタイプソフトウェアの開発において、基本設計によって得られた一つのプロセスを、Smalltalk 上の一つのオブジェクトとして実現する。また、プロセス間の情報のフローは、プロトタイプソフトの上ではオブジェクト間のメッセージ通信として実現する。

また、設計されたシステムが共有のデータストアを持っているような場合、プロトタイプソフトにおいては、これも一つのオブジェクトとして実現する。これにより、共有データストアへのアクセスを明示化することが可能となる。

このような形でプロトタイプソフトウェアを実現した場合、動作時のオブジェクト間のメッセージ通信を解析することにより、各プロセス間の関係などを動的に分析することができる。

このような形でプロトタイプソフトウェアを実現した場合、動作時のオブジェクト間のメッセージ通信を解析することにより、各プロセス間の関係などを把握することができる。また、プロセス分割のこの情報を基本設計にフィードバックすることにより、ターゲットシステムの動作状況により即したプロセスの分割が可能となる。

ターゲットソフトウェアの開発においては、上流からの仕様を受けて、プロトタイプソフトウェアのオブジェクト群と同様の構造でプロセス群を実現する。

つまり、プロトタイプソフトウェアは完全に使い捨てとなるわけではなく、図 1 のようにその骨格をターゲットソフトウェアに受け継ぐことができる。

3 プロトタイプソフトウェアの構成

Smalltalkにおいて、アプリケーション作成時のオブジェクトの構成法として、MVCモデル[3],[4]という考え方がある。これは、一つのシステムを構成するオブジェクトをModel(シミュレーションの対象)、View(モデルをある視点からとらえて表示する)とController(モデルに変化を起こさせる)の三つの要素に分けて作成するという考え方である。

本手法においては、プロトタイプソフトウェアにおいて入出力装置も含めたシミュレーションを行なってるが、ハードウェアのシミュレートしている部分とターゲットソフトウェアの機能をシミュレーションしている部分がMのオブジェクト群に混在している場合、この構造をそのままターゲットシステムで実現してしまうと、ターゲットソフトウェア内でハードウェアに依存する部分が大きくなり、ハードウェアの変更などに対応することが困難になる。

このため、我々はこのプロトタイピング手法向けにMVCモデルを応用し、シミュレーションの対象を構成するモデルに属するオブジェクトの中で、ターゲットソフトの機能を実現するオブジェクトをTargetという要素と考え、これをMから分離して、MVCモデルをTMVCモデルに拡張する。つまり、Mには、入出力機器などを含めたターゲットシステムの全体が含まれるが、Tにはそれらの入出力機器に関するオブジェクトは含めない(図2参照)。

このようなオブジェクトの構成をとることにより、

- ハードウェアの隠蔽
  機能のオブジェクトと入出力機器のオブジェクトを明確に分離することで、プロトタイプソフトウェアにおけるハードウェアの隠蔽を行なう。
- オブジェクトの部品化
  オブジェクトを役割毎に分類することで、オブジェクトの部品化を促進する。
- ターゲットソフトウェアの部品化
  Tに属するオブジェクトと対応した形でターゲットソフトウェアの部品モジュールを作成し、これをライブラリ化することにより、ターゲットソフトウェアの合成も可能になる。

などの効果がある。

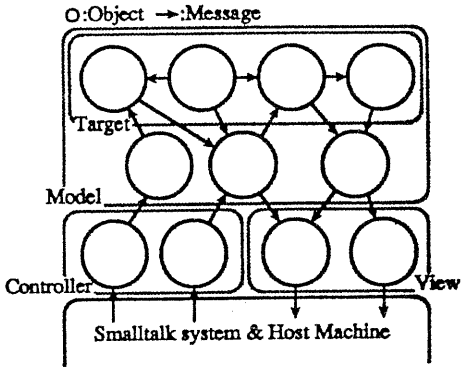

図2: TMVCモデル

4 プロトタイプソフトウェアの自動合成

家電製品など製品分野においては、ある製品の仕様に若干の機能を追加あるいは削除することによってシリーズ製品を作成する、といった開発が行われることが多い。このような場合にはあらかじめ考えられる機能を単位として、ソフトウェア部品を作成しておけば、この中から必要な機能だけを選択し、プロトタイプソフトを容易に合成することが可能である。

本プロトタイピング手法は上で述べたような考え方のもとに、各製品分野毎にプロトタイプ用部品ライブラリを作成し、ここから製品に必要な機能を選択することによって、プロトタイプソフトを合成するための環境を実現している。

プロトタイプソフトウェア合成は以下のような手順で行なう。

(1) 対象の製品に搭載される可能性のある機能について、それを実現するためのプロトタイプ用部品オブジェクトを開発し、システムに登録する。

(2) ユーザの要求(この場合、ある機能を搭載するか否か)をシステムに入力する。

(3) プロトタイプを合成する。

(4) プロトタイプソフトウェアを動作させ、外部仕様や設計などについて評価する。

(5) 問題があれば、(1)～(3)にフィードバックする。

このような手順のものとで、プロトタイプソフトウェア合成のためのシステムを、第3節のようにいくつかのツールと合成のために必要な各種データに

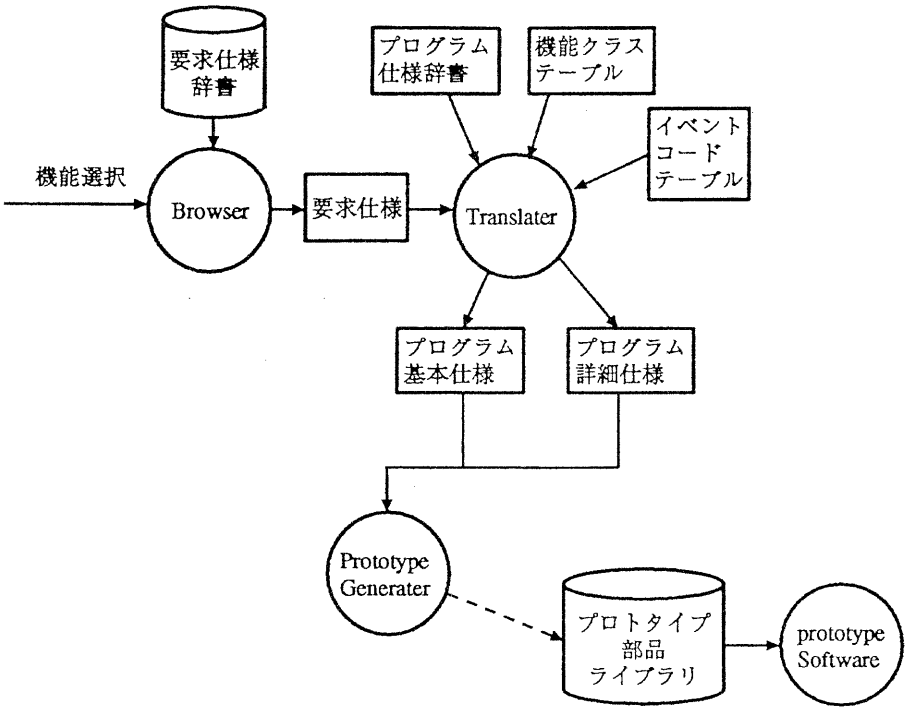

図 3: プロトタイプソフトウェア合成システム

よって構築した。次節以降でこれらのツールやデータについて説明を加えていく。

4.1 要求仕様辞書

前述した通り、プロトタイプソフトウェアの合成はソフトウェアに対するユーザの要求をあらかじめ用意された機能の中から選択する、といった形式で行なう。

このあらかじめ用意された機能に関するデータを要求仕様辞書と呼ぶ。要求仕様辞書は、階層的な"辞書(dictionary)形式の構造でシステムに格納されている。ここで辞書構造とはある見出し(key)とそれに対応する値(value)の組からなり、この見出しにより値の検索や登録を行なうことができるデータ構造である。

要求仕様辞書は対象システムに搭載が予想されるような機能に関するデータを集積したもので、対象製品毎あるいはその機能毎に階層化/構造化されたデータ構造を持つ。

このように分野/製品依存のデータを集中することにより、合成システムを構成する各ツール(図3のBrowserあるいはTranslater)のコードから分野/製品依存のデータやオブジェクトに関する記述を取り除くことができる。

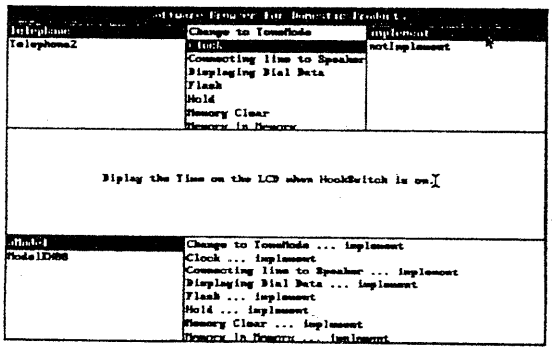

図 4: 機能選択用ブラウザ

4.2 機能選択用ブラウザ

機能選択用のツールは、前節で述べた要求仕様辞書に記載された機能の中から、ユーザが求める機能を選択するためのものであり、図4のようなブラウザ形式のユーザ・インタフェースを持つ。

ブラウザによって要求仕様辞書から選択された情報を、その対象製品の要求仕様と呼ぶこととする。この要求仕様は、各製品に搭載が予想される機能とその搭載の可否あるいは機能に対するパラメータなどを格納した辞書構造のデータで、次節で述べるトランスレータの入力情報となる。

4.3 トランスレータ

トランスレータは、ブラウザからの出力情報である要求仕様を、プロトタイプソフトウェア合成のために必要なプログラムの仕様に変換する。

具体的には、単に製品の機能とその選択の状況からなる要求仕様を、その機能を実現するために必要な情報に変換し、プロトタイプ合成を行なうツールへの入力情報を作成する。

トランスレータの入力情報としては、要求仕様以外に以下のようなデータが必要である。

機能クラステーブル プロトタイプソフトウェアの各機能及び各入出力装置のシミュレーションを実現するためのオブジェクトのクラス名一覧が、辞書形式で格納されている。

プログラム仕様辞書 予想される全ての機能に対して、その機能を実現するために必要な情報(その機能を構成するオブジェクトを生成/実行するために必要なデータ、機能実現に必要な状態名及びイベント名の一覧あるいは要求仕様の矛盾をチェックするための機能間の関係に関する情報など)が格納されている。

イベントコードテーブル 予想される全ての機能を選択した場合に発生する全てのイベントのイベントコードが辞書形式で格納されている。

トランスレータは、これらの情報を元に、以下のような処理を行なう。

(1) 仕様のチェック
プログラム仕様辞書内に格納されている、機能間の関係情報を用いて要求仕様の一貫性についてチェックを行なう。要求仕様に矛盾があるような場合には、これを警告し、仕様の選択に戻る。

(2) プログラム基本仕様の作成
要求仕様に矛盾がない場合には、トランスレータはプログラム仕様の作成を行なう。
トランスレータによって作成される情報は、プログラム基本仕様とプログラム詳細仕様の二種類である。 プログラム基本仕様は、プログラム仕様辞書から要求仕様に含まれる機能に関する部分を抜き出して編集したものであり、データの構造はプログラム仕様辞書と同じである。

(3) プログラム詳細仕様の作成
最後にトランスレータはプログラム詳細仕様を作成する。 プログラム詳細仕様は、プログラム基本仕様、機能タスクテーブルおよびイベントテーブルから作成され、選択された要求仕様を実現するために必要な機能クラス名、そのオブジェクト、入出力装置クラス名、イベント名あるいは状態名などから構成される。

トランスレータによって作成されたプログラム基本/詳細仕様は、次節で述べるプロトタイプジェネレータの入力となる。

4.4 プロトタイプジェネレータ

本節では、前節で述べたようなプログラム基本/詳細仕様から、実際にプロトタイプソフトウェアを生成するプロトタイプジェネレータについて述べる。

プロトタイプソフトウェアにおいては、ブラウザから選択されたアプリケーションの各機能を実現するために、個々の機能オブジェクトを用いている。つまり、選択された機能を実現するためには、機能を記述するクラスからインスタンスを生成し、実行時にはこのインスタンスに何らかのメッセージをおくって機能を実現することになる。

よって、プロトタイプソフトウェアの合成は、各機能を実現するためのインスタンスの生成及び初期化を行ない、さらにこれらのインスタンスを駆動するメカニズムを作成する、という2つの処理が必要である。

本システムにおけるプロトタイプジェネレータは、インスタンスの生成、初期化と駆動メカニズムの作成を行なうために、以下のような手順を採用した。

(1) 必須機能オブジェクトの組込み
選択され得る全ての仕様に共通するような基本的な機能を実現するオブジェクトを取り出し、システムに登録する(インスタンス生成はプログラム詳細仕様作成時に行なわれている)。

(2) イベントコントローラの生成と初期化
システムに対する入力イベントからシステム全

体の状態遷移を制御するイベントコントローラのインスタンスを生成、初期化しシステムに登録する。

(3) タイマの生成と初期化
時間に関する処理を行なうためのタイマに関するオブジェクトを生成 / 初期化し、システムに登録する。

(4) 入出力装置の生成
入出力装置のシミュレーションを行なうためのオブジェクトを生成 / 初期化し、同時にプロトタイプソフトウェアのユーザ・インタフェースの初期化を行なう。

(5) 機能タスクオブジェクト群の組織化
各機能を実現するオブジェクトを初期化し、それらの機能オブジェクト間あるいは機能オブジェクトと入出力装置間の関係付けを行なう。

(6) 状態遷移テーブルの生成とインストール
イベントコントローラで用いる状態遷移マトリクスを生成し、これをシステムに登録する。

本システムにおいては、ブラウザから必要な機能を選択し、これを実現するプロトタイプソフトウェアを作成する、という方針を採っている。状態遷移マトリクスについてもこの方針に基づき、あらかじめシステムに登録された最大仕様でのイベント / 状態からなる状態遷移マトリクスを作成しておき、選択された機能によってこの最大仕様のマトリクスから不必要な部分 (その機能の組合せでは起こらない状態 / イベントに関するマトリクスの各行 / 列) を取り除くという手順で、仕様を満たす状態遷移マトリクスを作成している。

状態遷移マトリクスの各要素には、ある状態であるイベントが起きた場合にシステムが起こすアクションと次に遷移すべき状態が記載されているが、本システムにおいては、アクションは次に動く機能オブジェクト名とそのオブジェクトに対して送るメッセージ (起動すべきメソッド名) が記載される。

以上のような手順により、プロトタイプソフトウェアは生成 / 初期化される。

プロトタイプジェネレータは、プロトタイプソフトウェアを生成した後にプロトタイプソフトウェアに制御を移す。

プロトタイプソフトウェアは、ユーザ・インタフェース画面上に表示されたキーなどをマウスでクリックすることにより、実際のハードウェアでのキーを押す動作などのシミュレーションを行っている。押されたキーのオブジェクトはプロトタイプソフトウェアへの入力としてそのキーのキーコードなどの情報をイベントコントローラにメッセージとして送る。イベントコントローラは現在の状態と受けとったメッセージから、状態遷移マトリクスに記述された機能オブジェクトにメッセージを送り、メソッドを起動する。そのメソッドが終了した後、プロトタイプソフトウェアは次の状態に遷移し、更なるイベントを待つ。

実際に作成したプロトタイプソフトウェアの実行時画面を図5と図6に示す。この例における対象システムは多機能電話であり、図5はその最大仕様を実現したものである。また、図6は最大仕様から、時刻表示やダイヤル表示などの表示に関する機能及びダイヤルメモリ機能などを全て取り去った仕様である。

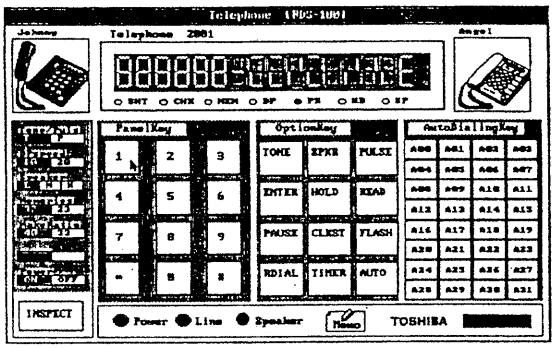

図 5: 多機能電話のプロトタイプ (最大仕様)

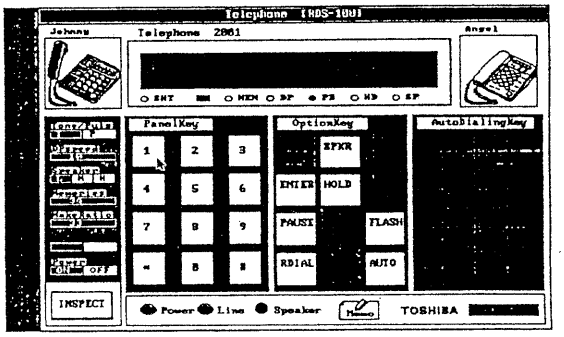

図 6: 多機能電話のプロトタイプ (表示機能なし)

これらのようなプロトタイプソフトウェアを動作させることにより、選択した機能がプロトタイプ上

でどのように動作するか、あるいはある入力に対するプロトタイプの反応が仕様通りに実現されているか、などの外部仕様の確認を行なうことができる。

また、第2.2節で述べたように、プロトタイプソフトウェアはターゲットソフトウェアと同様の内部を持つように作成している。つまり、各機能オブジェクト間のメッセージ通信を解析することにより、ターゲットソフトウェアの各プロセス間の関係を、プロトタイプソフトウェアによって確認できる。

5 ターゲットソフトウェアの自動合成

プロトタイプソフトウェアにより、動作の確認を行ない、同時に基本設計を十分に検討した後、ターゲットソフトウェアの開発に進む。

新規開発の場合、プロトタイプソフトウェアによる基本設計を元にターゲットソフトウェアを開発する。このとき、再利用が可能なように部品化を行なう。

既にターゲットソフトウェアに関しても最大仕様を実現できるような全ての部品が揃っている場合、プロトタイプソフトウェア合成時に選択した仕様と同様の仕様でターゲットソフトウェアを自動合成することも可能である。

前節で例示した多機能電話は、最大仕様を実現するようなターゲットソフトウェア部品をも開発したので、プロトタイプソフトウェアと同様の仕様によるターゲットソフトウェアの自動合成も可能である(開発にはアセンブラ言語を使用した)。

部品化することによるプログラムサイズのオーバーヘッドは、最大仕様において、部品化を考慮せずに同様の仕様で開発されたプログラムに比べて、約4%の増加に留まり、プロトタイプソフトウェアと同様の構造でターゲットソフトウェアを開発し、再利用可能な部品を開発する、という手法が十分に実用に適することが確認できた。

6 おわりに

以上、本稿ではプロトタイピングを用いたマイコンソフトウェア開発手法の概要を述べた。本手法は、冒頭で述べたようなマイコンソフトウェア開発における問題点を以下のように解決する。

I. プロトタイプソフトウェアで設計を十分検討することにより、再利用可能な設計を行なうことが可能である。

II. プロトタイプソフトウェア上でハードウェアのシミュレーションを行なうことで、テスト用のハードウェアがない場合でもソフトウェアの動作が確認できる。

III. ハードウェアを隠蔽するような設計を行なうことで、ハードウェアの変更に対する対応が容易になる。

IV. プロトタイプ及びターゲットソフトウェアを部品化が可能となるため、開発の分業化が可能となる。

また、ある製品分野において、利用されうる機能に関するソフトウェア部品を用意することにより、機能に関する要求からプロトタイプソフトウェア及びターゲットソフトウェアを自動合成する手法とその有効性について述べた。

今後の課題としては、以下のような項目が挙げられる。

- プロトタイプ部品作成の支援
  現在は、プロトタイプ用の部品は直接Smalltalkでプログラミングしているが、より容易にプロトタイプソフトを作成するために、部品作成を支援するツールあるいは簡易言語を開発する。
- 設計の評価
  プロトタイプソフトのオブジェクト構成が適切であるか否かの評価を行うための基準の策定する。
- 他の組込み用ソフトウェアへの適用
  他の組込みマイコンソフト開発に本手法を適用し、有効性を確認する。

参考文献

[1] 五十嵐, 西村:「プロトタイピングによるマイコンソフトウェア開発」情報処理学会第43回全国大会論文集(1991)

[2] 伊藤, 本位田, 内平:「プロトタイピングツール」啓学出版(1987)

[3] 小林:「オブジェクト指向とSmalltalk」CQ出版社(1989)

[4] 舘野他:「基礎からのSmalltalk-80」工学社(1987)