搭載電子装置の組込みソフトウェアの開発技法と信頼性確保

Software Development and Quality Assurance for Embeded Electronics

名古屋誘導推進システム製作所　大川祐司*1　栗田浩司*2

ミサイルのオートパイロット電子装置やヘリコプタのエンジン制御装置に組込むマイクロコンピュータのソフトウェアは，非常に高い信頼性が要求されている．この信頼性確保のため，プログラム設計工程にソフトウェア開発手法の一つである構造化手法をサポートするCASE（Computer Aided Software Engineering）ツールを，試験工程にソフトウェアを含む製品の機能・性能をダイナミックに評価できる閉ループシミュレータを導入した．CASEツールによりソフトウェア開発の早い段階で設計不適合が多く検出され，また，閉ループシミュレータによりシステム試験などの後工程での不適合が大きく減少し，ソフトウェア品質向上に効果を上げている．

Software of micro-processors embeded in the autopilot of a missile and gas-turbine engine controller requires high reliability. For satisfying this reliability, we introduced the CASE tool, supporting Structured Analysis and Structured Desigh for software development, and a Closed-loop simulator, which evaluates the function and performance of the equipment. We decreased the number of software problems using these tools.

1．まえがき

当社では，ミサイルのオートパイロット電子装置やヘリコプタ用ガスタービンエンジン制御装置などに組込むソフトウェアを開発している．これらのソフトウェアは，その性格上，非常に高い信頼性が要求されている．そのため，ソフトウェア開発の工程と作業標準を明確に定め，信頼性の確保に努めてきた．しかし，ソフトウェア開発後のシステム試験などの後工程での不適合は，なかなかゼロにはならない．また，近年の開発作業効率化の方針に従い，効率的にソフトウェアの信頼性を向上することが急務となってきた．

この課題解決のために，ソフトウェア設計の工程にCASE（Computer Aided Software Engineering）ツールを，また，ソフトウェア試験の工程に閉ループシミュレータを導入した．CASEはソフトウェアの設計・製作を計算機で効率的に行うためのツールである．閉ループシミュレータは，対象ソフトウェアの周辺の外部環境をソフトウェアで模擬し，電子装置の機能・性能をダイナミックに評価・検証するものである．本報では，ソフトウェアの信頼性向上に大きな効果のあったCASEツールと閉ループシミュレータについて，導入のねらい，具体的な利用方法，その効果などを述べる．

2．搭載型組込みソフトウェアの特徴

当社では，多種にわたる電子装置を設計・製作してきたが，近年手掛けてきた代表例としては，次のようなものがある．

(1) ミサイルのオートパイロット電子装置

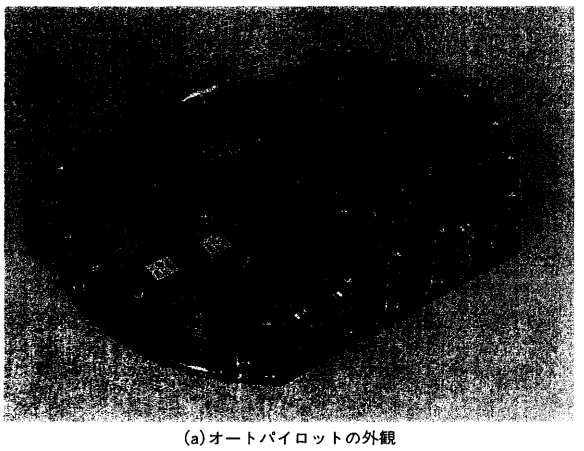
ミサイルを目標に導くために，入力データを基に操舵(だ)翼を適切に制御する装置（図1参照）．

(2) ヘリコプタ用ガスタービンエンジン制御装置

ガスタービンエンジンの始動，負荷変動に対する応答，安全側への制御，停止等を自動で行う，冗長機能を持った制御装置．

これらのミサイルやヘリコプタに搭載する電子装置に組込むマイクロコンピュータ（マイコン）のソフトウェアを，搭載型組込みソフトウェアと呼んでいる．

この搭載型組込みソフトウェアは以下のような特徴を持っている．



(a)オートパイロットの外観

(b) オートパイロットの位置

図1　ミサイルのオートパイロット　ミサイルのオートパイロット電子装置の外観と位置を示す．
Autopilot for missile

(1) ミサイルのように高速で運動する物体を制御するためには，その制御周期は非常に短く，姿勢制御では1kHzの制御ループもある．

(2) 非常に高い信頼性が要求されている．

(3) 電子装置の高機能化に伴いソフトウェアへの依存度が高まり，組込みソフトウェアも大規模化，複雑化している．また，他の電子装置とのインタフェース，特に高速通信インタフェースが多い．

3．組込みソフトウェア開発上の問題点

このような特徴を持った組込みソフトウェアを高い品質で開発するため，当社では，ソフトウェア開発の工程が明確に定義され，各工程での作業標準が定められている（図2参照）．この作業標準

*1 誘導・電子機器部ソフトウェア設計課長
*2 誘導・電子機器部ソフトウェア設計課

	要求分析（システム設計）	プログラム基本設計	プログラム詳細設計	プログラム製作	プログラム試験	ハード/ソフト結合試験	システム試験
作業内容	製品に対する機能・性能を基にソフトウェアに対する要求仕様を明確にし，プログラム開発仕様書を作成する．	ソフトウェアに要求される機能をタスクに分割し，タスク間の関連，タスクの起動タイミング，タスク内構造，データベース等を設計する．	プログラムモジュール単位での処理を設計する．	プログラミング（プログラム・コードの作成）を行う．	プログラム単体または結合した状態で，機能・性能が要求どおりであることを開発計算機上で検証する．	実機上でハードウェアとソフトウェアを含めた機能・性能を検証する．	ほかの電子装置と組合せてシステムとしての機能・性能を検証する．
作業標準	計算機プログラム開発仕様書作成要領	計算機プログラム基本仕様書作成要領	計算機プログラム詳細仕様書作成要領	コーディング規約設定指針（マニュアル）	プログラム単体試験手順書/成績書作成要領 プログラム結合試験手順書/成績書作成要領	プログラム結合試験手順書/成績書作成要領	
出力文書	計算機プログラム開発仕様書	計算機プログラム基本仕様書	計算機プログラム詳細仕様書	プログラムコード	プログラム単体試験手順書/成績書 プログラム結合試験手順書/成績書	プログラム結合試験手順書/成績書	システム試験手順書/成績書
レビュー	ウォークスルー実施要領，計算機プログラム所内審査実施要領						
変更管理	製品用ソフトウェア形態管理実施要領						

図2　ソフトウェア開発の各工程と作業標準　ソフトウェア開発の各工程での作業内容，準拠すべき作業標準，出力文書等を示す．
Software development process and standards

に従ってソフトウェア開発を行っているが，ソフトウェア開発後のシステム試験などの後工程での不適合は，なかなかゼロにはならない．この原因を開発工程別に見ると，次のようになる．

3．1　要求分析工程での問題点

搭載電子機器は，それ自体の機能が複雑であるだけでなく，ほかの電子機器とのインタフェースも極めて複雑であるため，開発の初期段階では細部の要求仕様にはあいまいな部分が多い．また，要求仕様の設定者であるシステム設計者の意図とソフトウェア開発者の理解との間には，どうしてもミスマッチが残る．これらがプログラム設計工程以降で表面化し，ソフトウェアの品質を低下させる可能性がある．

3．2　プログラム設計工程での問題点

複雑なソフトウェア構造・処理内容を明確にするため，また，長期間のプログラム保守を考慮して細かい部分まで記述するため，ドキュメントの量が非常に多い．したがって，これらのドキュメントのレビューや整合性の確認に十分な時間を掛けてもチェック漏れが生じることがある．また，ドキュメントが手書きであると，作成自体に時間が掛かるだけでなく，類似システムへの流用も簡単にはできない．

3．3　プログラム試験工程での問題点

搭載電子装置は動的システムのため，それに組込むソフトウェアもシステム全体の挙動やほかの電子装置の動きとの整合性を考慮すべきである．したがって，単なる入力に対する出力の確認だけでは，十分な評価とはいえない．

3．4　ハード/ソフト結合試験工程での問題点

搭載電子装置はハードウェアと一体化したマイコンシステムのため，製品としての機能・性能を検証するためには，製品を閉ループの中に組込んでリアルタイムでシミュレーションすることが望ましい．しかし，試験環境が大掛かりになるなどの理由により簡素化するため，完全な検証とならないケースがある．

3．5　開発工程全般に対する問題点

ソフトウェアに仕様変更は避けられないが，設計書が手書きであると迅速な対応が難しい．プログラムコードのみを修正して設計書の修正が漏れると最終製品に不整合が発生し，大きな問題の原因になり得る．

これらの問題点に対し，3.1節要求分析工程，3.2節プログラム設計工程及び3.5節開発工程全般に対する打ち手としてCASEツールの導入・利用を，3.3節プログラム試験工程及び3.4節結合試験工程の問題点に対する打ち手として閉ループシミュレータの開発を行った．

4．CASEツールによるソフトウェアの設計・製作

4．1　CASEツール導入のねらい

(1) 要求分析工程に対する解決策

要求仕様の設定者とソフトウェア開発者との間で，CASEツールのアウトプットを利用したウォークスルー(注)を行う．これにより仕様の不明確な部分を明らかにし，そこを重点的に検討することで効率的に要求分析が行えるようにした．

(注) 開発の各工程での成果物を作成直後に関係者でレビューすること．

(2) プログラム設計工程に対する解決策

CASEツールを利用してプログラム設計書を作成することにより，整合性のとれたドキュメントを簡単に作成することができる．また，類似システムに対しプログラム設計結果の流用が容易となり，品質の向上と生産性の向上を図ることができる．

(3) ソフトウェア開発工程全般に対する解決策

CASEツールを利用することにより，ドキュメントの修正が容易となるため，仕様変更時にプログラム・コードの修正だけでなく設計書の修正も迅速に行える．したがって，設計書の直し漏れによる不適合がなくなる．

4．2　CASEツールの利用方法

導入したCASEツールを**表1**に示す．このCASEツールは，ソフトウェア開発手法の一つである構造化手法（構造化分析，構造化設計）をサポートし，計算機上でこの手法に沿ったドキュメントの作成を可能にしている．我々はこのCASEツールを搭載型組込みソフトウェア開発において以下のように利用した．

(1) 要求分析，プログラム基本設計

要求仕様を受けて，必要な機能をデータ・フロー・ダイアグラム（DFD）を利用して書き表した（**図3**参照）．

(2) プログラム詳細設計

プログラム・モジュールの階層を定義した後，各モジュール

表1　導入したCASEツール
Introduced CASE tool

商品名	TEAMWORK
開発元	米国　CADRE社 〔日本での代理店　(株)東陽テクニカ〕
動作環境	EWS (SUNまたはHP)
主な機能	・構造化手法（構造化分析，構造化設計）をサポート（リアルタイム拡張を含む） ・データフロー図，構造図，プロセススペック，モジュールスペックなどの作成支援 ・作成ドキュメント間の整合性チェック

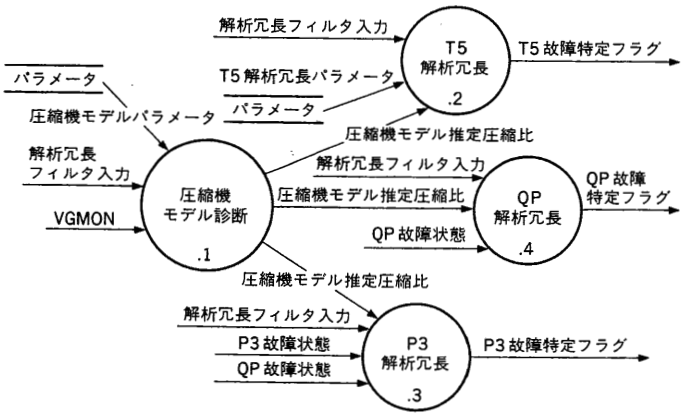


図3　データフロー図の例　CASEツールの機能のうち，データフロー図を使って要求仕様を分析した結果を示す．
Example of data flow diagram

の設計内容をCASEツールを使って記述した（図4参照）．また，この設計書からC言語のプログラム・コードを自動的に生成するツールを開発した（図4参照）．

4．3　CASEツール利用上の留意点

CASEツールの良い面ばかり述べてきたが，CASEツールは導入すればそれだけで効果が出るものではない．言換えれば，効果の出るように利用しなければならない．我々のソフトウェア開発で得たCASEツール利用のノウハウの主なものを以下に示す．

(1) 自分たちのソフトウェア開発の流れの中で，CASEツールを適用する範囲，適用方法を明確にする．我々の場合，CASEツールを用いたプログラム設計の手順を明確にし，各フェーズでの作業のノウハウをマニュアル化した．その具体例は，基本設計でのデータフロー図はタスクを意識して書く，データの定義は必要最低限にするなどである．

(2) プログラム設計のうち，導入したCASEツールの不得手な部分や，自分たちの開発方法に合ない部分には違うツールや他の方法を考える．我々の例では，シーケンスのタイミングやマン・マシン・インタフェースの定義には別のツールを利用した．また，データベースの設計には，CASEツールとデータの互換を意識しながら，表計算ソフトをベースに独自のツールを作成するなどの工夫をした．

5．閉ループシミュレータの開発

閉ループシミュレータは，対象製品である制御装置に接続する外部環境の挙動を模擬することにより閉ループを構成して，製品をダイナミックに評価・検証するものである（図5参照）．この閉ループシミュレータにはソフトウェア・シミュレータと，リアルタイム・シミュレータがある．

```
void    ＮＰＴＲＭ制限()
{
/*********************************************************************
1.ＮＰＴＲＭ（ＮＰのターゲット値）の急激な変化を押える。
*********************************************************************/
/*-------------------------------------------------------------------*
*                ＮＰＴＲＭ変化率制限                               *
*-------------------------------------------------------------------*/
if((長周期ＮＰＴＲＭｆｉｌ-長周期制限後ＮＰＴＲＭ前回値) < ＮＰＴＲＭ変化率下限)
then    /*-----下限リミッター-----*/
    長周期制限後ＮＰＴＲＭ=長周期制限後ＮＰＴＲＭ前回値+ＮＰＴＲＭ変化率下限;
else
  if((長周期ＮＰＴＲＭｆｉｌ-長周期制限後ＮＰＴＲＭ前回値)>ＮＰＴＲＭ変化率上限)
  then  /*-----上限リミッター-----*/
    長周期制限後ＮＰＴＲＭ=長周期制限後ＮＰＴＲＭ前回値+ＮＰＴＲＭ変化率上限;
  else  /*-----変化率制限内-----*/
    長周期制限後ＮＰＴＲＭ=長周期ＮＰＴＲＭｆｉｌ;
  endif
endif
/*-------------------------------------------------------------------*
*                次回のための設定                                   *
*-------------------------------------------------------------------*/
長周期制限後ＮＰＴＲＭ前回値 = 長周期制限後ＮＰＴＲＭ;
  return;
}
```

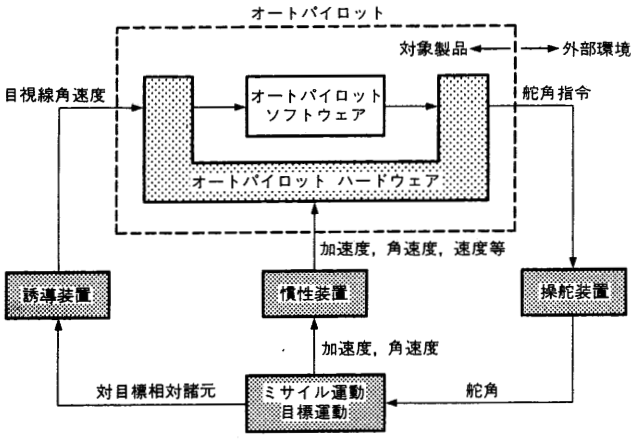
詳細設計書

自動コーディング
ツール

```
void    inNPTRML()
{
/*********************************************************
1.ＮＰＴＲＭ（ＮＰのターゲット値）の急激な変化を押える。
*********************************************************/
/*-----------------------------------------------------*
*              ＮＰＴＲＭ変化率制限                    *
*-----------------------------------------------------*/
if((ltfv_NPTRMfil-ltfv_NPTRMlimzen)<pafc_RNTD)
{           /*---下限リミッター---*/
    ltfv_NPTRMlim=ltfv_NPTRMlimzen+pafc_RNTD;
}else{
  if((ltfv_NPTRMfil-ltfv_NPTRMlimzen)>pafc_RNTU)
  {         /*---上限リミッター---*/
    ltfv_NPTRMlim=ltfv_NPTRMlimzen+pafc_RNTU;
  }else{    /*-----変化率制限内-----*/
    ltfv_NPTRMlim=ltfv_NPTRMfil;
  }
}
/*-----------------------------------------------------*
*              次回のための設定                        *
*-----------------------------------------------------*/
ltfv_NPTRMlimzen = ltfv_NPTRMlim;
  return;
}
```

プログラムコードの例

図4　詳細設計書と自動コーディングツールの出力例　CASEツールを使ってプログラム詳細設計を行った例と自動コーディングツールの出力の例．
Example of M-SPEC and output of autocoding tool



ソフトウェア・シミュレータの場合 ▨ と ▩ の部分を模擬する．
リアルタイム・シミュレータの場合 ▩ の部分を模擬する．

図5　閉ループシミュレータの概念図（オートパイロットの場合）
Concept diagram of closed-loop simulation

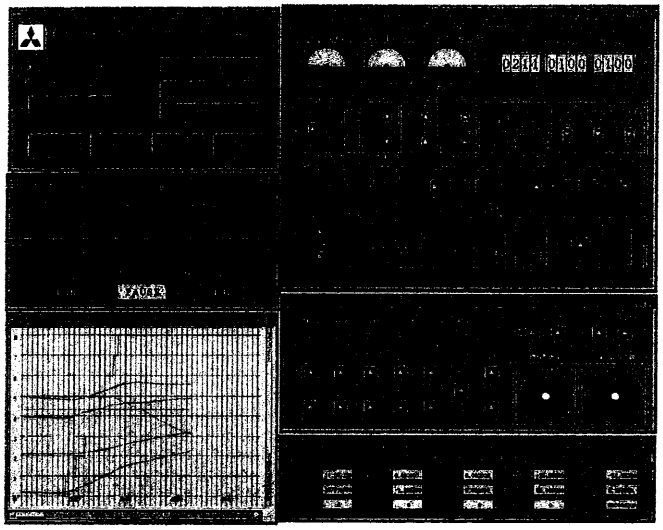


図6　ソフトウェア・シミュレータの画面の例　ソフトウェア・シミュレータを使い，プログラム試験を行っている際の画面例．左下にシステムの動きがトレンドグラフで表示される．
Example of software simulator display

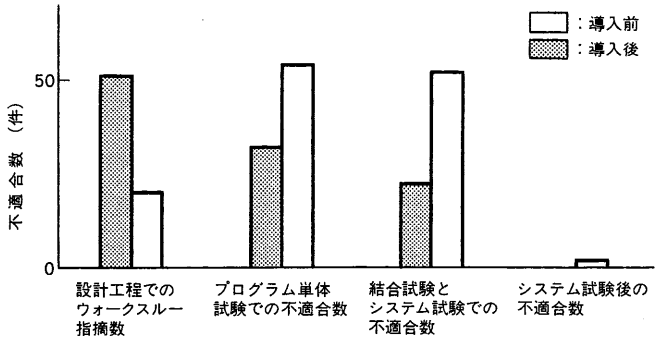


図7　ツール導入前後のソフトウェア不適合の数　CASEツールと閉ループシミュレータを導入する前後での，ソフトウェア不適合の数．プログラム規模で数を補正した．
Number of software troubles before/after using tools

表2　ツール導入後のソフトウェア不適合の数
Number of software troubles after using tools

	設計ウォークスルーでの指摘数	プログラム試験での不適合数	システム試験での不適合数	システム試験後の不適合数
Aプロジェクト（C言語/14Kステップ）	137	111	5	0
Bプロジェクト（アセンブリ言語/6Kステップ）	22	29	1	0
Cプロジェクト（アセンブリ言語/8Kステップ）	14	47	1	0

5．1　ソフトウェア・シミュレータ

ソフトウェア・シミュレータは，3.3節の問題点に対する解決策として開発した．ハードウェアと組合せて試験を行う前に，ソフトウェアとしての完全性を検証するためのものである．静的試験ではなかなか表面化しないソフトウェアの不適合を，閉ループでダイナミックに検証することにより摘出することを目的としている．

具体的には対象ソフトウェアに対する外部環境として，ミサイル運動やエンジンの動き，及びほかの電子装置や対象製品自身のハードウェアの動きを模擬したプログラムを製作し，これと組込みソフトウェアを結合して，計算機上で閉ループシミュレーションを行った（図6参照）．

5．2　リアルタイム・シミュレータ

リアルタイム・シミュレータは，3.4節の問題点に対する解決策として利用した．ミサイルシステムやエンジンシステムと組合せる前に，電子装置としての完全性を検証するためのものである．電子装置の動作速度に合せて外部環境も同じ速度で模擬するリアルタイムの閉ループシミュレーションにより，様々な動作状況での機能・性能の確認を目的としている．

具体的には高速演算・高速入出力が可能なリアルタイムミニコンを導入し，電子装置を閉ループの中に組込んだHWIL（Hard-Ware In the Loop）シミュレーションを行った．また，実際の試験が難しい異常発生時の処理について，異常状態を模擬するシナリオ機能を組込むようにし，試験を可能とした．

6．実施例と効果

CASEツールと閉ループシミュレータを導入する前の開発事例と導入後の開発事例での不適合数の比較を，図7に示す．また，この二つの方法を導入したほかのプロジェクトでの不適合数を，表2に示す．

6．1　CASEツール利用の効果

CASEツール導入後は開発の早い段階で不適合が多く発見され，開発の後半での不適合数が減少している．これは，CASEツールを利用することにより，要求仕様の設定者とソフトウェア開発者の間で十分な意志疎通が可能となったためである．したがって，後工程に持越す不適合が減少し，ソフトウェアの品質向上に大きな効果があったと考えられる．

6．2　閉ループシミュレータの効果

システム試験以降に持越す不適合が大幅に減少した．これは，通常状態だけでなく，異常発生時の処理についてもダイナミックに検証できるためであり，関係者の厚い信頼を得ている．

7．ま　と　め

本報では，ミサイルやヘリコプタなどに搭載する電子機器に組込むソフトウェアについて，その信頼性を効率的に向上させるために導入したCASEツールと閉ループシミュレータについて紹介した．

プログラムの設計工程にCASEツールを導入して，要求仕様を早期に明確にし，要求仕様の設定者とソフトウェア開発者との間でウォークスルーを十分に行うことでソフトウェア品質を向上させた．特に類似システム開発時には，設計結果の流用が容易になり，信頼性だけでなく生産性も大きく向上した．また，プログラムの試験工程に閉ループシミュレータを用いることにより，ソフトウェア自身や電子装置の機能・性能をダイナミックに評価・検証することで，後工程に持越す不適合を大幅に減少させた．

今後は，これまでのCASEツール利用で蓄積したノウハウを生かして適用方法の改善を図り，プログラム設計の更なる品質向上に努力していきたい．また，現在では閉ループシミュレータの適用が困難である画像処理等についてその適用方法を検討し，ソフトウェアを含む電子装置の更なる信頼性向上に努めていきたい．