第3章

杉浦 英樹，石田 栄子，小林 稔央

組み込みソフトウェア開発現場のトラブル・シューティング

実際の開発業務では，どのような問題が発生しうるのか

前章では，「コンセプト」，「要求分析」，「設計」，「実装」，「テスト」，「ハードウェアとソフトウェアの統合とシステム・デバッグ」の各工程に分けて，組み込みソフトウェア開発者が，何を，どのように考えて仕事を進めるのか，ということを紹介した．しかし，実際の製品開発では，それほどスムーズに事が運ぶわけではない．ここでは，実際の開発現場で起こりがちなトラブルとその対処法について解説する．（編集部）

組み込みソフトウェアの開発現場で働こうとするなら，まず，ソフトウェア開発の方法を勉強するでしょう．最近ではソフトウェアの作り方についての参考書が増え，独学でも勉強しやすい環境になってきました．さらに，組み込みソフトウェア開発を前提にした解説書も増え，その気になれば明日からでも組み込みソフトウェアを開発できるのではないかという錯覚に陥ります．

ところが，これらの書籍は，新たに一からソフトウェアを作成することを前提にしたものがほとんどです．これに対して実際の職場では，制御対象を制御するソフトウェアがすでに存在し，そのソフトウェアを分析し，利用しながら，変化する要求に対応することがほとんどです．つまり，参考書と実際の開発現場の間には，悲劇的と言えるくらいの大きなギャップがあるのです．参考書を勉強してから，組み込みソフトウェア開発の職場に入ってきた人は，その違いにがっくりしてしまうかもしれません．

でも，落胆する必要はありません．先輩が残した負の遺産は負の遺産として，しっかりと見極めればよいのです．もしかすると，それが負の遺産であるということに気付かずに，皆さんも先輩をまねてしまうことがあるかもしれません．そうならないように，組み込みソフトウェアはどう作るべきか，どうすればもっと良くなるかということを常に考えるようにしてください．

例えば，情報処理推進機構（IPA）ソフトウェア・エンジニアリング・センター（SEC）から「コーディング作法ガイド」というプログラムの書き方についての参考書が出ています（**写真1**）．このガイドには，今までの経験に基づいた間違いの少ないソース・プログラムの書き方のコツが書かれています．自分たちが作成しているプログラムは良いものなのか悪いものなのか，どう書いたらよいのかの参考になるでしょう．

製品開発現場の要求分析や設計の工程においても，理想（理論）と現実の違いが存在します．開発現場のエンジニアのスキルによるところもありますが，開発手法や管理技術について，理論や理屈でいくら理想を説いても，実際の開発現場ではさまざまな問題が発生するものです．それらのギャップがなぜ起こるのか，また，どうすればそのギャップを埋められるのかを考えることが，組み込みソフトウェア・エンジニアの仕事といえます．

この章では，組み込みソフトウェア開発の現場で，「理想どおりにいかない」，「本質とかけ離れている」といった問題や，ソフトウェア開発で使われるツールの制約や癖について，参考書や書籍にはない視点で紹介していきます．製品開発の現場の実態が，少し見えてくると思います．

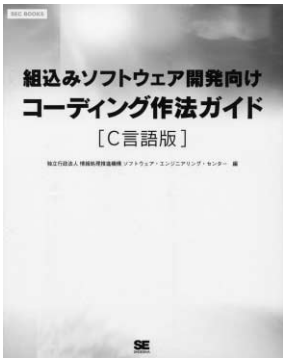

写真1
プログラムの書き方についての参考書

IPA SEC監修の「組み込みソフトウェア開発向けコーディング作法ガイド［C言語版］」（翔泳社，ISBN4798111899，1,800円）．以下のWebサイトから無償で入手することもできる（要ユーザ登録）．http://sec.ipa.go.jp/index.php

KEYWORD ── 要求分析，設計，実装，テスト，システム・デバッグ，ユースケース，オーバフロー，0 div，チャタリング，コンパイル環境，JTAG ICE，フルICE

1. 開発工程で起こった問題とその対処法

以下では，要求分析，設計，実装，テストの各工程で発生した問題点とその対処法について説明します．

要求分析工程の問題点(1)

要求仕様を入手しないと作業が進められない

●現場で起こった問題

メカ(機構系)を制御する組み込みシステムのソフトウェア開発では，メカとエレキ(電子系)の制御仕様を要求仕様としてソフトウェアを作成します．制御仕様を満たすことがソフトウェア作成の目的です．開発は，要求分析から始まり，設計，実装，テストという工程をリリース(後工程へのソフトウェアの引き渡し)ごとに繰り返します．

ただし，要求分析を開始する段階で制御仕様が入手できているとは限りません．また，制御仕様を入手できたとしても，その仕様が5日後には変わっていた，ということも少なくありません．このような状況下では，ソフトウェア担当者から「仕様が決まっていないので作業が始められません」という声が上がってきます．

このような考え方をしていると，制御仕様が確定するのを待ったまま作業が進まず，開発初期からスケジュールに遅れが発生してしまいます．

●問題解決へ向けての考え方

要求元の作業を待っていては，スケジュールが遅れてしまうばかりです．ソフトウェア開発者は，分かる範囲で自ら制御仕様を作成し，積極的に仕様決めに対する活動を行う必要があります．現実には再利用開発(過去の設計資産を流用する開発)が多いので，流用元の仕様やソース・プログラムからシステムの概要を調べることは可能です．

また，組み込みシステムのソフトウェアを開発する場合，「要求仕様は変わるものだ」と考えていた方が現実的です．要求の変化にすばやく対応できる分析を行うためには，制御部材や制御目的に対する知識を深めることが大切です．

自分自身で自分の作業を進められる状況を作ることで，スケジュールどおりの作業を目指しましょう(**図1**)．

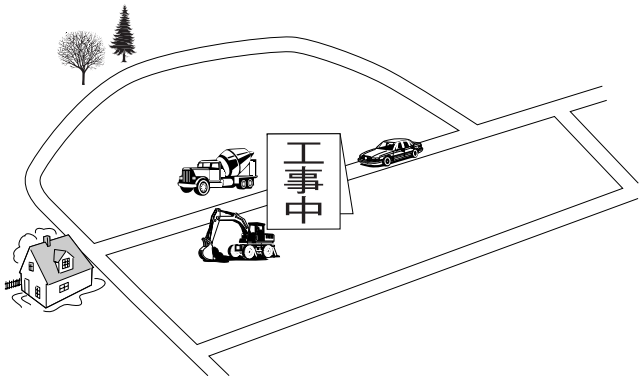

図1　自分自身で自分の作業を進められる状況を作る

待っているだけでは進まない．地図を見て，目的地に早くたどりつけるコースを見つけた方が早い．

要求分析工程の問題(2)

要求の抜け・漏れを要求分析工程で発見できない

●現場で起こった問題

要求分析の工程では，システムが「何を」するものなのかを整理し，制御内容を明らかにしていきます．分析工程を終了し，設計，実装と作業を詳細化していくと，そこで初めて，どう動作させるべきか分析できていなかったケースに気付くことがあります．また，実際にシステムを動作させてから，特定のケースに対する制御が明確になっていなかったことに気付き，新たに仕様を追加することもあります．

このようなとき，ソフトウェア担当者は，「ソフトウェア・バグでなくてよかった」と胸をなでおろすことでしょう．ですが，もし，要求分析のときにすべてのケースに気付いていれば，仕様変更は必要なかったはずです．たとえ要求元がそのケースを見落としていたとしても，ソフトウェア開発者が見落としに気付かなければ，「十分な分析ができていなかった」ということもできます．

では，どうしたら，要求分析のときに要求仕様の抜け・漏れを見つけられるのでしょうか．

●問題解決へ向けての考え方

要求仕様の抜け・漏れ防止には，ユースケース・シナリオによる分析が有効です．ユースケースに対して事後条件を定義し，基本系列(事後条件を満たすケース)と代替系列(事後条件を満たさないケース)を分析していくと，さまざまな状況に気付くことができます．