01 組み込みアプリケーション・ソフトウェア開発者

山科祥悟

組み込みアプリケーション・ソフトウェア開発者とは

規模・目的に関係なく，専用のハードウェアに搭載されて特定の機能を実現するシステムを「組み込みシステム」，そこに搭載されるソフトウェアを「組み込みソフトウェア」と呼ぶ．こうしたソフトウェアを開発する技術者が組み込みアプリケーション・ソフトウェア開発者（以下，ソフトウェア開発者）である．ソフトウェア開発者は，組み込みソフトウェアで実現する機能の分析，設計，コーディング，テストを行う．組み込みソフトウェアには，設計条件にさまざまな制約が存在し，実現の難易度が高い．それゆえ，挑戦しがいのある分野であるとも言える注1．

● まずは，一つの知識を徹底的に理解・習得する

経験1～2年目の技術者は，C言語でいう関数の詳細設計やコーディング，関数単体テストなどを担当します．その作業に必要な知識として，以下の三つが挙げられます．

- 設計技法（例えば構造化設計．逐次・条件分岐・繰り返しの基本や処理単位を分割するSTS分割技法など）
- プログラミング言語（例えばC/C++言語．組み込みソフトウェア開発ではC言語が多用されている注2）
- テスト技法（同値分割，限界値分析，制御の流れに着目する制御パス・テストなど）

また，ソフトウェア工学（アルゴリズムの知識）や使用するOSのスケジューリング動作について知っていれば，奥の深いソフトウェアを開発することができます．ただし，知識を詰め込み過ぎては消化不良を起こし，それぞれの技法を有効に活用できません．まずは上述のどれか一つを徹底的に理解し，習得することが大事です．一つのことを理解しようとすれば，周辺の知識が必要となり，おのずとほかの事がらも理解できるようになります．

近年の組み込みシステムでは，携帯電話やカー・ナビゲーション・システムなどのように，GUI（graphical user interface）を備える製品が目立つようになってきました．しかし，これは組み込みシステムの中のほんの一部で，そのほとんどは目に見えないところで稼働しています注3．その見えないシステムやソフトウェアを開発するのですから，創造力と想像力，そして論理的思考力が求められます．

このように書くと，「自分はどうも…」と敬遠されるかもしれません．しかし，例えば，規則性を持ったソフトウェアには数列や帰納法の考えかたが適用できますし，ソフトウェア・テストは集合論を応用しています．理数系の基礎ができていれば，問題なく組み込みアプリケーション・ソフトウェア開発者になれます（**図1**）．

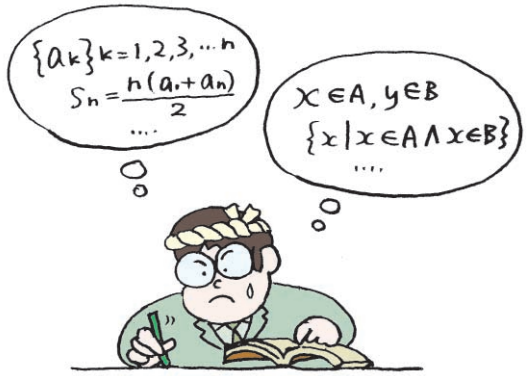

図1　理数系の基礎ができていればだいじょうぶ
例えば，規則性を持ったソフトウェアには数列や帰納法の考えかたが適用できる．ソフトウェア・テストは集合論を応用している．

● 世の中のために役に立ったことを実感できる

苦労する点は，さまざまな制約（メモリ量，CPU性能，リアルタイム性，高信頼性，安全性，価格，実行環境など）の中で品質を確保し，開発を完了させなければならないことです．また，ソフトウェア開発には技術的な制約のほかに，予算や工程の制約もあります．品質を高めるには手間と時間とお金がかかります．安易に妥協すると，顧客の評価が下がり，信用をなくします．しかし，高すぎる理想を追い求めると，開発者が疲弊します注4．

注1：決して，ほかの分野のソフトウェア開発がやさしいということでなく，制約条件がより厳しいという意味である．
注2：トロン協会が実施している「組込みシステムにおけるリアルタイムOSの利用動向に関するアンケート調査」（http://www.assoc.tron.org/jpn/research/index.html）などで報告されている．
注3：携帯電話，エアコン，テレビ，冷蔵庫など，見渡してみると，組み込みシステムに囲まれていることに気付くはずである．
注4：IPA（独立行政法人 情報処理推進機構）SEC（ソフトウェア・エンジニアリング・センター）の「2005年版組込みソフトウェア産業実態調査報告書」（http://sec.ipa.go.jp/download/200506es.php）などを参照．
注5：大きなシステムほど，稼働している状況を間近で見られる機会は多くない．現場の雰囲気を知り，システムを理解するためにも積極的に客先の現場に赴こう．
注6：http://www.ertl.jp/SWEST/を参照．

筆者は入社2年目に，システム・テストのため，顧客のもとに出張しました．それまでは，パソコンに向かって仕様書に記載された内容に忠実にコーディングし，テストしていただけで，自分の開発したソフトウェアがいったい何の役に立つのか，理解どころか想像すらできませんでした．ところが，そのソフトウェアは顧客のもとで宇宙機器の壮大な管制制御装置の一部として機能し，その役割を果たしていました．正常に動作しているソフトウェアをまのあたりにしたとき，自身の作ったものが世の中のシステムの中で役に立っていると実感がわき，非常に感動しました[注5]．その感動を何度も味わいたいがために，ソフトウェア開発を続けていると言っても過言ではありません．

やりがいは人それぞれだと思いますが，筆者の場合は，自分の設計したソフトウェアが世の中の役に立っていることを実感できたときに，「がんばって良かった」と感じます．

● 新人技術者への三つの質問

筆者は新人エンジニアに，以下の3点を質問します．

1）自分に投資していますか？

工業製品を開発する者としてしごとをするのですから，趣味のプログラミングとは違って体系的な専門知識が必要です．しかし，「しごとを教えてもらえる」という消極的な考えかたでは務まりません．自分に投資して，しごとの成果で回収する，という考えを持ちましょう．それが，自分自身のスキルを高め，より良い製品の開発につながります．

2）動機（モチベーション）を持っていますか？

ソフトウェア開発は知的労働です．入社1～2年目は知らないことが多く，勉強としごとの両立で時間の余裕がありません．そのときにこそ考えてほしいのは，「何のために勉強しているのか？」，「何のために働いているのか？」ということです．筆者は自分の作るソフトウェアで世の中をもっと便利に，豊かにしたいという気持ちを持って，18年前にこのしごとに就きました．短納期や技術的難易度の高いしごとでは辛い思いをしましたが，動機があったからこそしごとの山を乗り越え，達成感を持てたのだと思います．

3）ものづくりは好きですか？

組み込みソフトウェア開発に携わって，いろいろな人としごとをしましたが，ものづくりが好きな人たちほど楽しくしごとをして，良い製品を世に送り出しています．必要な知識，スキル，心がまえなど，いろいろ書きましたが，これがいちばん大事なことです．

筆者は先日，SWEST7[注6]に参加しました．日本の組み込み業界の著名な方から学生まで，幅広い技術者が参加していましたが，この人たちの共通項は，「ものづくりが好き」，「率先してみずからものづくりを楽しんでいる」ということでした．もし，「ソフトウェア開発が楽しくない」と感じ始めたら，一度参加してみることをお勧めします．

やましな・よしのり
三菱電機マイコン機器ソフトウエア（株）

私の推薦書

「Code Complete第2版 ―完全なプログラミングを目指して（上・下）」

Steve McConnell 著
日経BPソフトプレス 刊，
ISBN4-89100-455-X/4-89100-456-8，B5変形判，
628/545ページ，6,405円，2005年3月

「プログラマ必携のバイブル的書籍です」と，紹介者は使い古された定型の賛辞でお茶を濁したくなります．なぜなら，1行1行から得られるものがそのくらい多いからです．

この本には，より良い製品を開発するため，高品質で高信頼性のコードを書くためのヒントが満載されています．コーディングとデバッグに主眼が置かれていますが，単なる実装論にとどまらず，上位設計からテスト手法，品質保証プロセスの重要性まで言及されています．読み進み，理解が深まるとともに，ソフトウェア開発はプログラムを書く（コーディング）だけではないということ，すなわちソフトウェア開発の奥深さを知ることができます．読み返すたびに新たな発見がありますが，「驚き」が「うなずき」に変われば，それはあなたの成長のあかしかもしれません．堅苦しい印象を受けるかもしれませんが，軽快な文章（翻訳者の努力のたまもの）が理解を手助けしてくれます．

第1版発行から第2版発行までに約10年が経過していますが，10年間のソフトウェア工学発達の成果が余すところなく投入されています．考えかたはオブジェクト指向が主体となり，言語はC++，C#，Java，Visual Basicなどが取り上げられています．このようなことからも，第2版はより近代的な印象を受けます．

すべてを理解するのはたいへんですが，全体を大ざっぱに眺めた後，コーディングの現場で目次や索引を活用して必要なときに適用すると，より実践的な知識として身に付くことでしょう（とは言うものの，1回は落ち着いて目を通さないといけないのだが…）．