第3章

システム・レベル言語による組み込みソフト開発

村上晋一郎

ここでは，組み込みソフトウェア開発者向けに，SpecC 言語を利用してソフトウェアの仕様を決めていく手順を紹介します．すなわち，SpecC 言語で記述した仕様モデルをソフトウェアとして実装した場合の例です．SpecC 技術を利用すると，早期にプロトタイプによるテストを実施するスパイラル・モデルと呼ばれる開発フローを実現できます．これによって，早い時期に仕様もれなどを発見できます．実際に「車のヘッド・ライト制御」をモデル化しながら，開発手順を説明します．　（編集部）

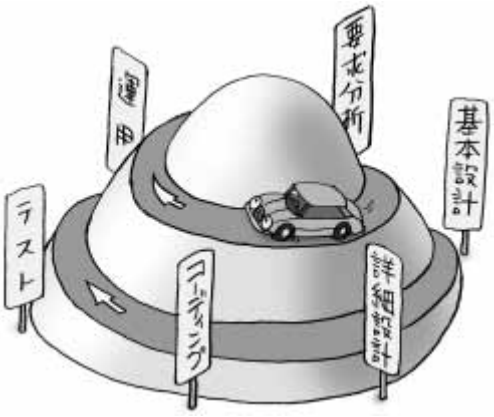
いまの世の中は「なにをしたいのかはっきりさせること」が大切な時代です．そして重要なのはそのプロセスです．

一昔前までは結果重視で，「なにができたのか」といったことばかり取り沙汰されてきました．結果を重んじるのはいまも昔も変わりはありませんが，その過程でできあがった生成物にたいへん意味があり，重要視されてきています．なぜなら，世の中に存在するいわゆる仕事といったものは昔に比べて規模が膨れ上がり，1人では処理しきれないものがほとんどだからです．一つの仕事にチームで取り組む，人事の異動によって引き継ぎが発生するといったことを考慮に入れながら，ゴールへと進んでいかなければなりません．

チームで仕事をこなす場合，お互いのコミュニケーションが重要であり，標準化された中間生成物をもとにコミュニケーションが確立されると考えられます．引き継ぎが発生した場合でも標準化された中間生成物があると，それがない場合よりもはるかに楽です．このような状況では，中間生成物がどれだけ優れているかで，結果が変わります．「終わりよければすべてよし」の時代は終わり，結果だけを管理していてはチーム内におけるコミュニケーションがばらばらになり，コストは膨れ上がり，納期が遅れるといったことになります．しかし，悲しいことに中間生成物を作成するだけでは問題は解決しません（「その件についてはあの人じゃなきゃわかりません」症候群に対する特効薬はないのか？）．

重要なのは「だれでも意味を理解できるもの」を生成することです．チーム内でバラバラの記述で作成された中間生成物

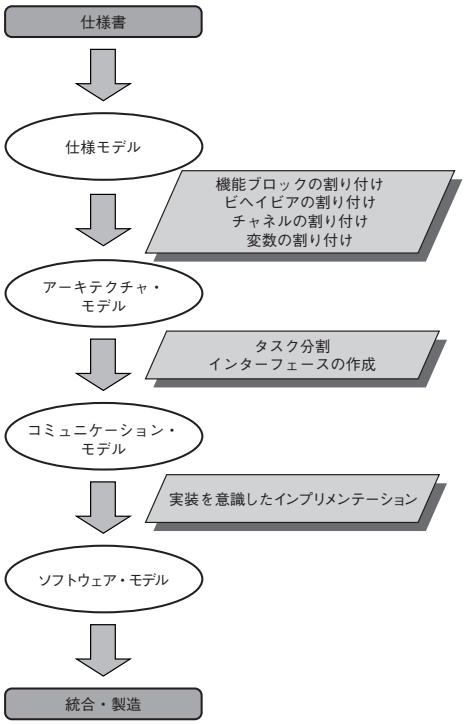
(a) ウォータ・フォール・モデル



(b) スパイラル・モデル

〔図1〕ソフトウェア開発の二つのモデル

(a)のウォータ・フォール・モデルでは，川の水が上流から下流へ向かって流れるがごとく，設計が進められていく．この設計手法では，下流から上流への設計の手戻りは許されない（「許されない」というのが手法であり，このモデルの特徴である）．(b)のスパイラル・モデルでは，上流の局面においてプロトタイプを作成する．そのプロトタイプをもとに実際に動作するようすをデザイン・レビューでチェックし，目的とするシステムが正しい方向に向かっていることを確認する．



〔図2〕SpecCによる組み込みソフトウェアの開発フロー

機能ブロック，ビヘイビア，チャネル，変数を割り付けることでアーキテクチャ・モデルが構築される．さらにリアルタイムOSを意識したタスク分割を行い，タスク間インターフェースを定義したものがコミュニケーション・モデルとして構築される．分割されたタスクごとに実装を意識したインプリメンテーションを施して，ソフトウェア・モデルが構築される．SpecCの各モデルについては，本特集第7章を参照．

は，その人でなければ理解できません．チーム内で統一されていないフォーマットもしかり．他の部署から配属された新人にはさっぱり意味がわかりません．

中間生成物のほとんどはただの文書（ドキュメント）です．業界標準のルールで記述された中間生成物は，それだけでも威力を発揮することは理解いただけると思いますが，いまやディジタル社会です（表現が古い？）．中間生成物はさらに進化を遂げて，動的モデルであることが望まれます．いままでは中間生成物を人が見て，それを理解するために頭の中で実際にシミュレーションしていました．この作業をパソコン上で行うことで，さらにすばやく理解することができます．

このような問題を解決するための手法の一つとして，SpecC言語が存在します．SpecC言語は，組み込みソフトウェア，および電子回路のシステム仕様を明確にするための言語です．ここでは組み込みソフトウェア開発者向けに，SpecC言語やソフトウェア開発ツール「ZIPC」などを利用したソフトウェア開発手法について解説します．

1 組み込みソフト開発とSpecC技術

ソフトウェアの開発フローには，要求分析→基本設計→詳細設計→コーディング→テスト→運用→保守といった，いわゆるウォータ・フォール・モデルがあります（**図1**(a)）．SpecC技術から見ると，要求分析は仕様モデルの作成，基本設計はアーキテクチャ・モデルおよびコミュニケーション・モデルの作成，そして詳細設計・コーディングはソフトウェア・モデルの作成に相当すると考えられます（**図2**）．なお，SpecC技術の理論的な概要については，本特集第7章を参照してください．

〔図3〕コミュニケーション不足が招くエラー

要求分析の際の言葉の取りちがいが原因となって，問題が発生しやすい．

ウォータ・フォール・モデルとは文字どおり「滝のように水が上から下へ向かって流れていく」ようすをモデル化したものです．この考え方は，上流の工程を経て下流の工程を実施することを前提としており，その逆はタブーとされています．この技法のよいところは，各工程の責任範囲が明確になり，各工程がある程度ルーチン・ワーク化されることにあります．各フェーズがしっかりと実施されていれば，ウォータ・フォール・モデルはとても有効な手法ですが，要求分析フェーズでのもれがテスト・フェーズで発見されると，最悪の場合，システムを最初から作り直す必要が出てきます．

これは，システム仕様を検討する際に，完成されるだろうモデルを想像しながら必要と思われる機能を抽出してレビューすることに問題があります．人によってイメージがあいまいで，たとえば仕事を発注する側と仕事を受ける側の間で，かなりイメージのとらえ方に差が出る傾向があります（とくに現場の技術者と全体を統括するマネージャでは価値観に差がある．言葉は同じでも頭のなかのイメージにギャップがある）．

そこで考えられているのが，スパイラル・モデルと呼ばれる開発フローです（**図1**(b)）．スパイラル・モデルには厳密には何種類かあるようですが，ここではプロトタイプを意識したスパイラル・モデルを説明します．通常，システムの検証は完成間近になってから行われます（上流工程ではすべて静的な設計書ばかりで，動くものがなに一つない！）．

たとえば，エンジニアのA君は少しどきどきしながら，やっとできあがったシステムを初めてユーザに見せることになりました（どきどきしているのは過去の経験からか？）．システムを操作しはじめて3分後，後ろで腕組みをしたユーザが一言．