第7章

状態遷移表を使ってクルーズ制御ソフトを設計する

キャッツの組み込み向け上流CASEツール「ZIPC 2001」

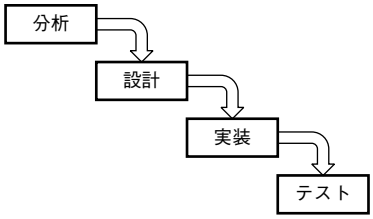
Disc 1
設計ツール

吉野由起夫

ここでは状態遷移表を利用した組み込みソフトウェアの設計を体験する．サンプル・システムはクルーズ・コントロールである．状態遷移表を作成すると，状態と要因の組み合わせが明示される．ソフトウェア開発者にすべての組み合わせを強制的に意識させることになるので，不ぐあいが見つかりやすい．ここでは，プロトタイプと組み合わせたシミュレーションや，Cコードの自動生成も行う．使用するツールは，キャッツが開発した組み込み向け上流CASEツール「ZIPC(ジップシー)」である．（編集部）

複雑かつ多様化する組み込み機器の開発において，不ぐあいによる多大な損失や開発の遅れといったトラブルが露呈してきています．そのうえ，コスト削減や人員削減によって，開発現場はさらに窮地に追い込まれてきているような気がします．こうした状況の下，ソフトウェア開発プロセスの抜本的な改革が求められています．

このチュートリアルでは，現状のソフトウェア開発プロセスのどこに問題があるのか，市販のCASE(computer aided software engineering)ツールを使うことによって何を解決できるのかを説明していきます．ここで利用するCASEツールは，設計表現として主に状態遷移表を利用する「ZIPC」です．実際にツールを操作しながら，CASEツールを用いたソフトウェア開発プロセスを理解していただきたいと思います．



〔図1〕ソフトウェア開発プロセス

組み込みソフトウェアの開発プロセスには，主に四つの工程がある．すなわち，要求仕様を分析し整理する「分析」，具体的にどのようなプログラムにするかを決める「設計」，プログラムを作成する「実装」，プログラムが正しく動作するかどうかを検証する「テスト」である．

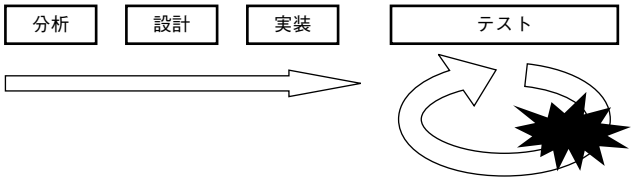
1．なぜソフトウェア開発プロセスの変革が必要か

組み込みシステムといってもさまざまな種類がありますが，開発現場は共通の問題を抱えています．

一つ目の問題はソフトウェアの規模の増大です．携帯電話を例に取ると，初めのころは電話する機能が主であり，アドレス登録などの簡単なデータ処理機能が付いているだけでした．しかしその後，インターネット接続など，さまざまな機能が付加され続けています．メール送受信機能やWeb閲覧機能の搭載は今やあたりまえとなり，最近ではゲームなどのアプリケーション・ソフトウェアをダウンロードして実行する機能や，動画表示の機能に対応したものまで出てきています．

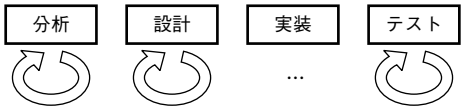
携帯電話以外の組み込みシステムでも，ネットワーク家電のようにインターネットに接続して付加価値を付けたり，ほかとの差異化を図るためのさまざまな機能が付け加わり，ソフトウェアの規模が増大しています．

二つ目の問題は開発サイクルの短さです．製品に新たな機能を追加する速度がどんどん早くなっており，開発に時間をかけていては，製品リリースのときにその機能が時代遅れになっている，ということもありえます．そのため，製品を短期間に開発し，できるだけ早く市場に投入することが要求されています．



〔図2〕従来の開発の流れ

分析，設計，実装までは順調に開発が進んでいるように見えるが，テスト工程でそれまで手を抜いていたところが露呈する．結局，開発が大幅に遅れることになる．



〔図3〕CASEツールを使用したソフトウェア開発の流れ

分析，設計の工程で検証を行うことによって設計品質を高めておくと，テスト工程では最終的な動作確認に専念できる．工程管理も容易になる．

〔図4〕
状態遷移表

列にプログラムの要因を，行にプログラムの状態を記述する（逆に記述することも可能）．要因と状態が交差する場所に処理と処理終了後の遷移を記述する．これによって，すべての要因と状態の組み合わせを明示する．

●分析や設計の手抜きがテスト工程で発覚する

ソフトウェアの開発プロセスを簡単に表すと，分析，設計，実装，テストになります（**図1**）．多くの場合，分析と設計はワープロなどで行います．また，実装やテストの工程ではコンパイラやデバッガなどのソフトウェア開発ツールを使います．

分析や設計をワープロで行っているということは，頭の中で考え，それを文章にしていることになります．開発規模が小さい場合には頭の中で考えられる範囲であるため，それなりに十分なレベルまで分析や設計を行うことができます．しかし，開発するソフトウェアの規模が大きくなってくると限界が来ます．頭の中ではすべての内容を把握しきれなくなり，問題点を見逃すことになります．また完成度が見えにくいという点も問題です．

作業の進捗が遅れている場合，完成度の見えにくい分析や設計の工程で手を抜いてしまいがちで，分析や設計が不十分なまま，実装の工程に入ってしまうことがあります．このしわ寄せは，すべてテスト工程で顕在化します．分析や設計で解決しなければならない問題を含んでいるため，テストに多大な工数が生じ，しかもどのくらいの工数で問題が解決するのかといった見通しが立たなくなることもあります．

また，テスト工程で不ぐあいが見つかったとき，プログラム・コードのみを修正し，設計書をそのままにしている例があります．この場合，バージョン・アップ時に設計書が信用できなくなるため，テストに依存した開発プロセスになってしまいます（**図2**）．

●開発支援ツールが必須になる

開発するソフトウェアの規模が大きくなるほど，分析や設計に十分時間をかける必要があります．テストに依存した開発プロセスになりがちなのは，実際に動作を検証できるのが，唯一テスト工程だけだからです．分析や設計を十分に行うためには，テスト工程と同じように分析や設計の工程でも検証を行うべきだと思います．ただし，従来の開発スタイルのままこれを行うのは容易ではありません．

分析や設計の段階できっちり検証を行う方法の一つとして，分析や設計の工程を支援する上流CASEツールを導入することが挙げられます．こうしたツールの多くは，分析や設計の段階で実際に設計書（モデル）を動作させてみて検証を行います．今後の組み込みソフトウェアの開発では，こうしたツールをうまく使いこなすスキルが必要になると思います（**図3**）．

2．使用するツールとサンプル・システム

このチュートリアルで使用するZIPCは，設計，実装，テストの工程を支援する組み込みソフトウェア向けの開発支援ツールです．現在，ZIPC 2001（Ver.8.0）が出荷されており，すでに10年以上の実績があります．

●特徴は状態遷移表とツール連携

ZIPCの設計手法は「状態遷移表設計手法」と呼ばれています．状態遷移表は，開発するプログラムの状態と要因の組み合わせを表にしたものです（**図4**）．

状態遷移表の利点はいくつかありますが，最大の利点は，状態と要因を抜き出すことによりそれらのすべての組み合わせを開発者に強制的に意識させられるというこ