ZIPCの開発上流からの活用事例

松下電工株式会社
モノづくり革新センター 開発プロセス革新グループ 課長
坂上　博信

1．はじめに

弊社工場で開発する商品は、大半がソフトウェアにより制御される家電製品であり、そのプログラミングの作業を外部に委託する場合もある。その場合は、設計仕様書を基に、仕様要求を伝える事となる。従って、仕様要求を正確に伝える為に作成する設計仕様書の完成度が、完成されるプログラムの性能、品位、コスト、納期を大きく左右する。その一方で、要求仕様を正確に伝える為の設計仕様書を短期間で作成することも、重要な課題となっている。

また、メカを制御するソフトウェアの開発においては、メカ設計者と協調して行う事も重要であると考えている。このメカ・ソフトの協調設計を推進する為に、富士通製の「VPS」を「ZIPC」と連携して活用する事も検討中である。

本稿では、弊社工場で設計仕様書作成にも活用している「ZIPC」及び、検討中の「VPS」「ZIPC」連携について記載する。

2．ZIPCの導入

2.1．VPS(IO Connector)及びZIPCの導入

数年前にまず、メカの３次元（3D・CAD）設計データを有効活用する事からスタートした。

3D・CAD設計データをVPSに取り込む事により、動くモデルとして扱う事が可能となる為、その動くモデルを制御するソフトウェア開発に活用を考えた時に、ZIPCにめぐり合えた経緯がある。

2.2．ZIPCの展開

当社では、ZIPCの導入をスムーズにする為、商品群毎に雛形となるサンプルを作成して、それを基に実際の商品設計に活用する方法を選択した。設計者がZIPCを使用して、すぐに効果を上げるには、このサンプルを活用する事が、得策と考えたからである。

2.3．ZIPCの活用

ZIPCにて仕様設計をはじめる場合、エディタは、状態遷移図（STD）エディタと状態遷移表（STM）エディタの２種類があるが、当社では、開発ステージ毎にそれぞれのメリットを考慮して使い分けている。

もちろん、ZIPCのシミュレーション機能を活用、或るいは、富士通製のVPSと連携してシミュレーションを行う為には、STMによる設計が必要である。開発の初期段階では、全体の仕様を把握する場合は、STDがやはり理解し易いと考える。

弊社では、STDにて仕様の概要を記述して、全体の仕様を把握した後に、順次STMにて仕様を記述している。ZIPCの機能の一つとして、STDからSTMへ相互に自動変換出来る機能があるが、どちらを先に作成をして、変換を行ったとしても、自動変換後の修正作業は困難に感じた。

この為、どちらを先に作成しても、自動変換機能を使用せず、最初から作成し直す事が、殆どである。この開発の進め方の検討を行うと共に、ツールの改善が必要であれば、対応をお願いしたい内容である。

また、状態遷移表の欠点は、表が大きくなってしまう事である。ZIPCでは、「拡張階層化状態遷移表設計手法」を採用して表の階層化により、その欠点を補う事を推奨している。しかし、当初、弊社の開発部門にてSTMによる仕様記述を行うとした場合、仕様の詳細迄を記載しなくても、すぐに＜100イベント × 100ステート＞以上の大きなSTMになっていた。

この様に、作成した大きなSTMを分析しながら階層化する方法の習得は、困難に感じた。

そこで、キャッツ殿と相談して、当社の開発の流れに合った方法を用いる事とした。

方法は、以下に述べる。

３．分析STMの作成

キャッツ殿からの提案は、下記の手順であった。

初めに、「図１　V字開発モデルの開発フロー」の様に、“ 分析STM作成 ” → “ 抽象STM作成 ” →“ 詳細STM作成 ” の各ステップに分けて、「図２　ZIPCの活用フロー」と「表１　各ステップ目的作業手順」の様に、各ステップで行う作業の目的を明確にして、作業内容の整理を行った。

その上で、最初の目標を、抽象STM迄を作成する事とした。

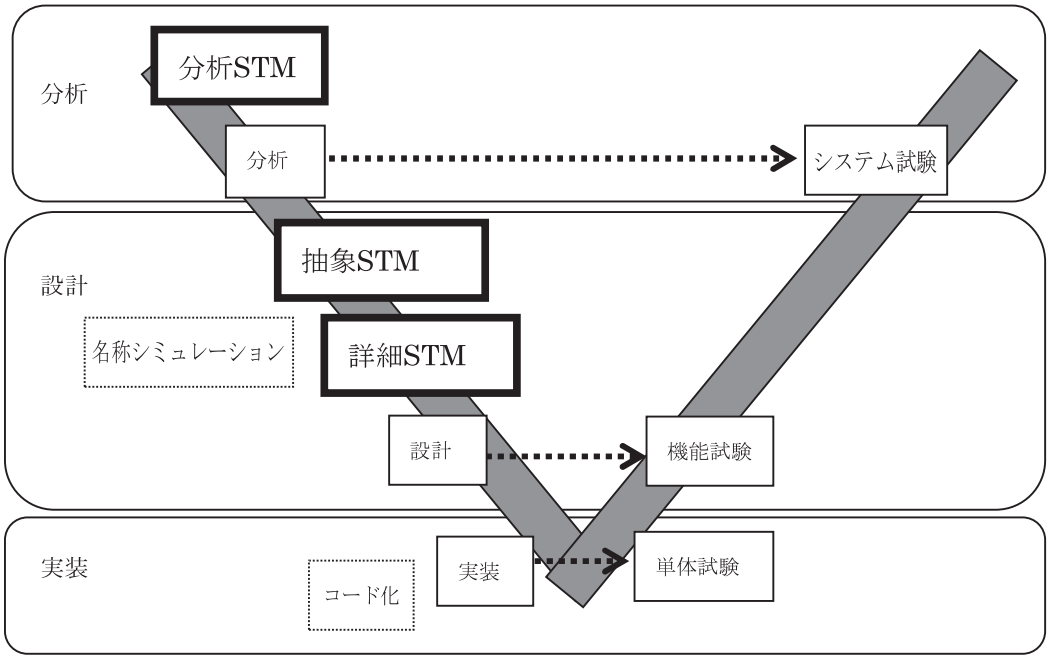

図１　V字開発モデルの開発フロー

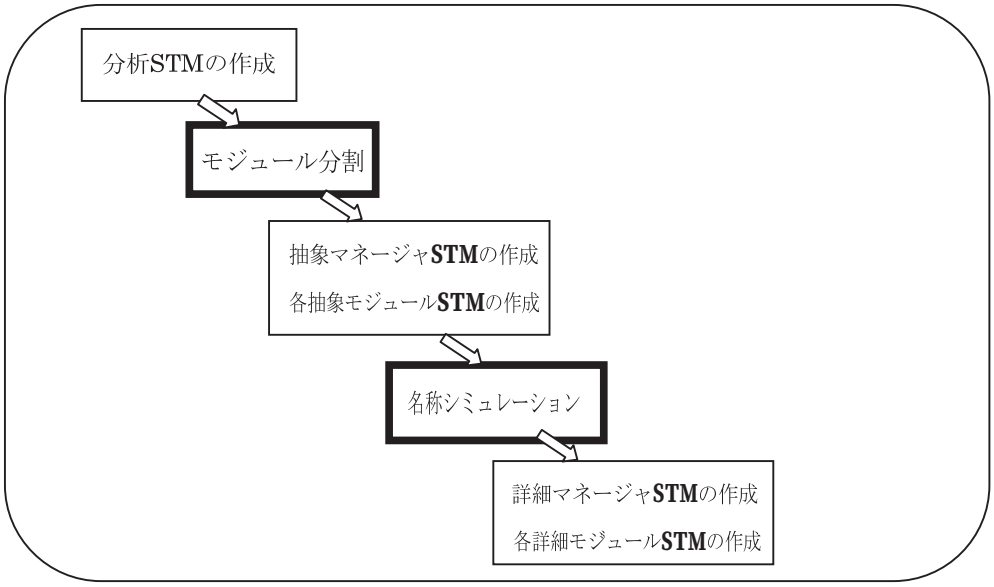

図２　ZIPCの活用フロー

表１　各ステップ目的作業手順

		目的・ポイント
step1	分析STMの作成	全体の把握と分析 モジュール分けの為にも、必要
step2	モジュール分割	機能に着目してモジュールを分ける
step3	抽象マネージャ**STM**の作成 各抽象モジュール**STM**の作成	STM全体の粒度を揃えることが重要になる
step4	名称シミュレーション_1	各モジュールを個別にシミュレーションする シミュレーションにより問題点を洗い出す
step5	抽象STMの階層化	マネージャSTMと各モジュールSTMのリンクを行う
step6	名称シミュレーション_2	シミュレーションにより問題点を洗い出す
step7	抽象マネージャ**STM**の作成 各詳細モジュール**STM**の作成	シミュレーションによる動作確認で問題が無い事を確認出来てから、初めて詳細な仕様を表現したSTMの作成を行う

3.1．分析STM

まず、分析STMの作成である。実際は、この分析STMの作成の前に、設計者が仕様の概要を確認する意味で、状態遷移図（STD）の作成も行っているが、省略している。

この分析STMを作成する大きな目的は、仕様の全体を設計者が把握して、どの様にモジュール分割するかを決定することである。

この時、注意すべき事は、STMを記述する時は、設計対象の仕様を「なにを」するのかの目的を出来るだけ抽象的に記述する事を意識して、表を＜20イベント × 20ステート＞程度に可能な限り収めるという事である。

筆者も実際に作業をおこなってみて、同感であった。

3.2．モジュール分割

次に、モジュール分割である。

本来は、プログラマーがソフトウェアの構成を仕様の追加・変更が発生した場合に、ソフトウェアを再利用することを考慮して行うものであるが、弊社では、外部にプログラミングを委託する場合に、設計仕様書を作成する開発者が、設計仕様書をプログラマーに仕様を理解しやすい形で作成する事を目的とする。

以上の様に、分析STMを活用して、モジュール分割を考える。言い換えると、モジュール分割を行う為に、分析STMを作成すると考えるべきであろう。

モジュール分けでの注意点は、下記が上げられる。

・モジュールとは、システムを機能毎にまとめたもの。
・ユーザ視点のモード分析とは異なるので、注意する。

モジュール分けを行うと、「図３　モジュール分け」に示す様に、システム全体の動作をコントロールしたり、各モジュールがどの様な状態にいるのか把握する役目のマネージャSTMが必要になる。

一方、各モジュールSTMは、マネージャSTMからの指示により、動作を開始し、その指示による動作を終了したら、完了した事をマネージャSTMに通知する役目になる。

これは、「表２　マネージャSTMと各モジュールSTMの責務（役割）比較」に示す関係になる。

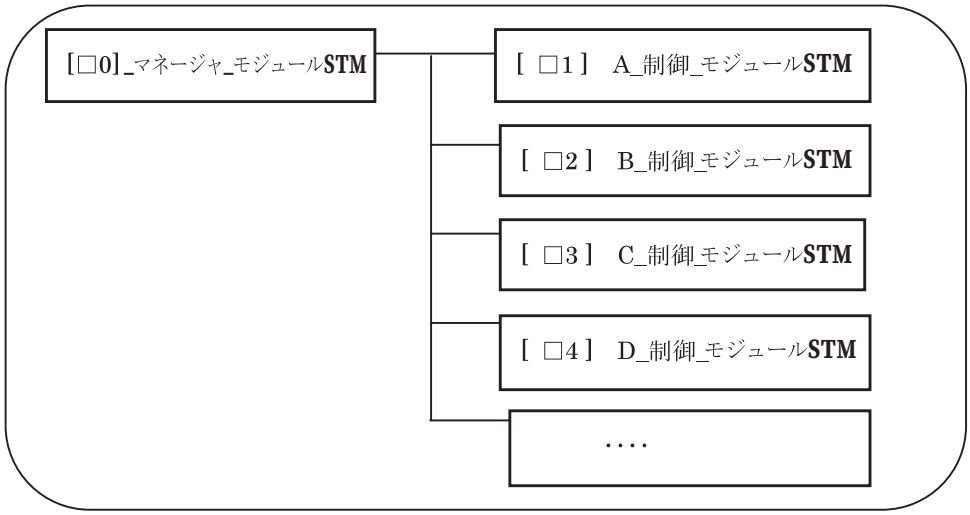

図３　モジュール分け

表２　マネージャSTMと各モジュールSTMの責務（役割）比較

	マネージャSTM	各モジュールSTM
責務 （役割）	システム全体の動作をコントロールしたり、状態を把握するための**STM** ・システム全体としての方向性を決定 ・どのような作業を行うかを考える ・どのような順序で行うかを考える	マネージャモジュールから指示された通りの作業を行う**STM**

マネージャSTMと各モジュールSTMの作成時の注意点として、キャッツ殿より下記の様なアドバイスを受けた。

「図４　マネージャSTMと各モジュールSTMの好ましくない関係」の様に、各モジュールSTM間で、状態等を変化させる様な制御を行うと、マネージャSTMは絶えず、各モジュールSTMの状態を把握（管理）する必要が生じる。

その様な仕組みにならない様に、「図５　マネージャSTMと各モジュールSTMの適切な関係」のような関係性を保つ必要がある。

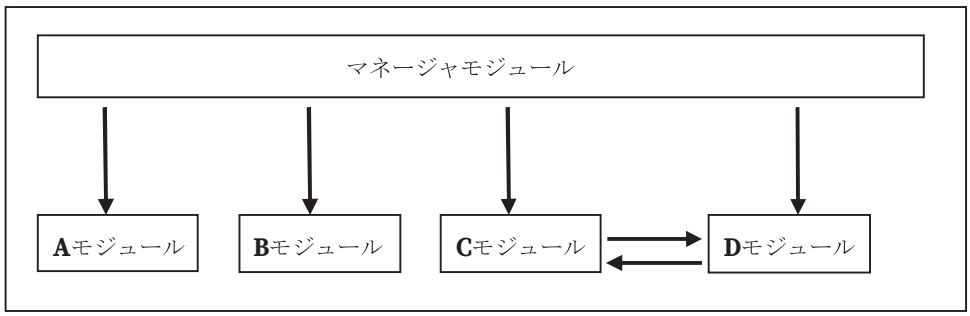

図４　マネージャSTMと各モジュールSTMの好ましくない関係

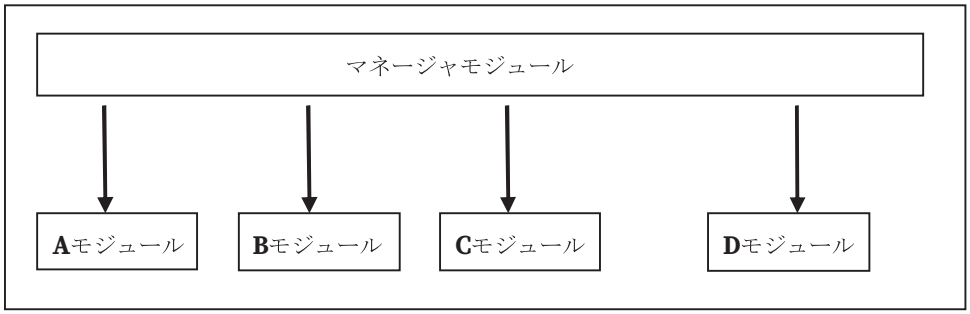

図５　マネージャSTMと各モジュールSTMの適切な関係

３.３．各モジュールの作成

次に、各モジュールの作成である。

STMの作成、検討の初期段階では、「図６マネージャ抽象STMの記述方法例」の様に、マネージャSTMの処理セルには、“モジュール＊呼出”として記載する。

□0 マネージャー	S	電源オフ状態	電源オン状態		
			待機状態	動作選択状態	
E		0	1	2	3
電源オフ	0				
電源スイッチ	1				
	2			モジュールA呼出；	
	3		モジュールA呼出；	モジュールB呼出；	
	4			モジュールC呼出；	

図６　マネージャ抽象STMの記述方法例

この状態で、マネージャSTMはもとより、各モジュールSTMをZIPCのシミュレーション機能を使用して仕様の動作確認を充分に行って、完成度を高めてから、マネージャと各モジュールSTMをリンクさせて全体の動作確認をする様にした。

ZIPCは、抽象度の高いレベルで記述した仕様でも、動作確認が行えるという事に着目し、ZIPCのシミュレーション機能は、頻繁に活用した。このシミュレーション機能を活用しないと、Excelで状態遷移表を作成することからのメリットとの違いが明確化しずらい。

STM作成は、最初から、詳細に及ぶ仕様を考えながら、記述をおこなっていくと、行き詰る恐れがある。この為、まずは、抽象度の高いレベルでの作成を考える事を、絶えず念頭において作成する事が重要である。

詳細化は、抽象度の高いレベルで動作確認により完成度を高めた後に行い、その後、詳細化を行ったSTMを基に作成されるソフトウェアプログラムの性能・品質が高まる事を実感した。

また、STMの仕様書を再利用する事により、コスト・開発期間の削減に貢献する事も確信した。

４．ZIPCと富士通製VPSとの連携

次に、富士通製VPSはメカの３次元（3D・CAD）設計データを動くモデルとして扱う事が出来る。このVPSとZIPCを連携して、メカ・ソフトの協調設計に活用する事も視野に入れて、開発フローの改善を推進する計画である。

以下が、ZIPCとVPSの連携により、動くモデルベース開発を行った場合に期待される効果である。

・開発のフロントローディング化（開発のピークを、上流にシフトさせて問題の早期発見）
・実機デバッグ機の削減（試作メカの作成費用コスト削減）

５．ZIPC適応で得られた効果

商品開発の上流である、設計仕様書の作成の初期段階から、ZIPCによるSTM設計を行う事により、下記の効果が得られた、或いは得られる事が判った。

・設計者とプログラマーとのコミュニケーションギャップの改善
・仕様書（状態遷移表）のフォーマットの統一
・専用エディタでSTMを作成する為、作成時間の削減が出来る。
・表による分析を行う事で、仕様の整理がしやすい。
・STMを再利用しやすい
・プログラミングの属人的要素を、STMにより「見える化」する事により、プログラムの構造を関係者で確認し改善出来る。

以上の様に、ZIPCの自動コード生成機能を使用しなくても、STM専用エディタ及び、シミュレーション機能を活用する事で、充分な効果が得られる事を実感した。

ZIPCを適応し、充分な効果を得る為には、現状の開発方法での問題点を把握して、最初に、その解決に役立つかの事前検討が重要であることを痛感した。

６．終りに

今後も、各開発部署での問題点を筆者の所属する様な、支援部隊が把握し、開発者との信頼関係の下に改善の取組は続けたい。

キャッツ殿には、ZIPCの効果的な活用方法について、多大なアドバイスを頂いた事を、紙面をお借りして御礼申し上げる。