ZIPC/State-Model Converterを適用したパワースライドドアのMBD事例

～新しいアプローチでさらなる発展を遂げる! 車載ECUのモデルベース開発の現場から～

古澤　透

株式会社 両毛システムズ
組込事業部　組込事業課
チームリーダー

1993年～ 1997年　オートクルーズコントローラ用ソフトウェア開発
1998年～ 2000年　電動アシスト車椅子用ソフトウェア開発
2001年～ 2003年　パワーテールゲート用ソフトウェア開発
2004年～　パワースライドドア用ソフトウェア開発

鈴木　秀昭

株式会社 両毛システムズ
組込事業部　組込事業課

・基板機能試験器用ソフトウェア開発
・出荷検査用データベースシステム開発
・温度制御コントローラ用ファームウェア開発
・LED光量校正装置ソフトウェア開発
・モデルベース開発教科書製作
・オイルポンプ検査用HILS開発
・パワースライドドア制御モデルベース開発

課題・背景

■自動車業界ではMATLAB/Simulinkを利用したモデルベース開発が注目されている
■機能の複雑化・短納期化に伴い、上流工程での十分な検証が必要
■Stateflow（状態遷移図）だけでは、ヌケ・モレを発見することが困難

解決方法

■要求レベルからのUML分析で、ドメイン・クラス分割し、クラス間のメッセージのやりとりで全体として制御する可読性・保守性の高い再利用可能なモデルを実現
■Stateflow（状態遷移図）で設計される部品に対しては、ZIPCでの検証を適用してヌケ・モレを早期発見
■ZIPC検証後の状態遷移図をSMC(※1)で変換し、Simulinkモデルに組み込むことでモデル全体の検証をシームレスに実現
（※1:SMC：State-Model Converterの略）

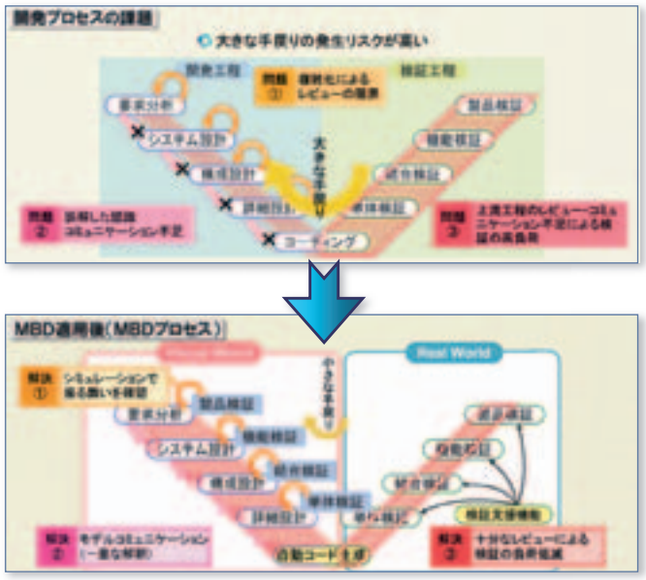
1. 導入背景

自動車業界の動向として、高度化・複雑化する要求を実現するために、電子制御の果たす役割が増している。さらには、期待される機能をもった製品の早期の市場投入が求められている。

これらの背景に対して、近年、開発の上流から下流までを「モデル」を用いて開発を行うMBD（MATLAB/Simulinkなどのモデリングツールを利用したモデルベース開発）に注目が集まっている。

MBDでは、以下の特徴により開発を大幅に効率化することができる。（図1参照）

・モデル化することで機能のつながりが認識しやすい
・メーカー・サプライヤ間や組織間のコミュニケーションが円滑となる
・シミュレーションにより設計段階から開発対象の動きを検証できる
・自動コード生成ツールや検証の自動化ツールの適用



[図1]MBDによる開発プロセスの課題の解決

MBDの特徴により開発プロセスにMBDを適用することで課題を解決し開発を大幅に効率化できることを表した図

[図2] パワースライドドア

今回紹介する事例では、パワースライドドア（図2参照）を題材にしたモデルベース開発を行う。

実開発では既存の仕様書とソースコードが存在するが、度重なる仕様変更により仕様書も複雑化しているケースも多い。

また、既存のソースコードをそのままモデル化しても、今後の複雑化した要求に対する適切な対応は、困難であると予想される。

本事例では、可読性・保守性の高い、再利用可能なモデルの実現を目標とし、要求レベルから分析を行うことにした。

また、複雑化した仕様のヌケ・モレを早期発見するためにZIPCを適用し、開発全体のフロントローディング（図3参照）を目指した。

2. 選定理由

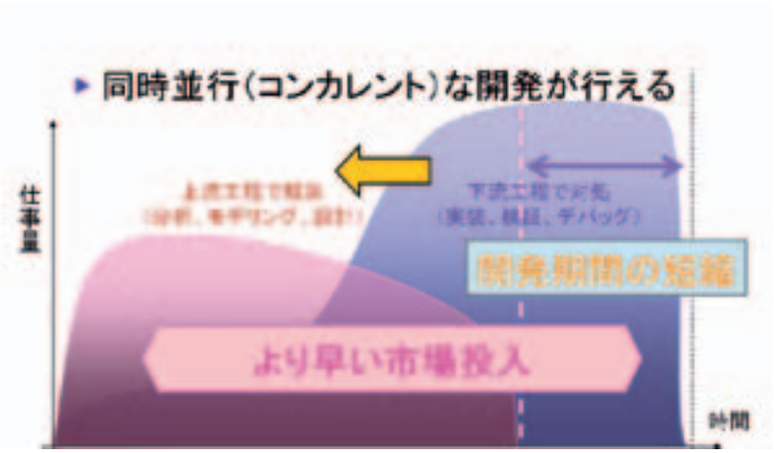
Stateflow(状態遷移図)では、モデルの概要を把握しやすい反面、ヌケ・モレに気が付かないことが課題として挙げられるが、ZIPCの状態遷移表は、未検討の部分を容易にチェックすることができ、ヌケ・モレの早期発見に有効であると思われた。

そこで、Stateflowで設計した部分の検証に対しては、SMCを使うことでStateflowからZIPCに変換し、状態遷移表形式でヌケ・モレの確認ができると考えた。

3. 改善内容

チームによる効率的な開発、より高品質で拡張可能なソフトウェアアーキテクチャ構築には、整理・整頓された、だれもが読める設計図が必要となる。そこで、はじめにシステム内部を複数の問題領域（ドメイン）に分割し、ドメインごとにしっかりとした責務分けを行った。（図4参照）

また、オブジェクト指向に基づき、UMLを使用して各ドメインの構造をクラス図でモデル化することで、モデル間の低依存度、低肥大化、高柔軟性を考慮した設計を行い、システム全体を整理・整頓されたモデルとした。



[図3] フロントローディング

設計レベルの検証を行うことにより開発プロセス全体を上流工程にシフトする。また、並行開発により開発プロセス全体を上流工程にシフトする。フロントローディングでより早い市場投入ができることを表している。

ドメイン名	責務
アプリケーションドメイン	ドアの状態と動作指令の管理
ユーザーインタフェースドメイン	ユーザーからのシステムへの入出力
メカニズムドメイン	ドアの移動メカニズムと制御アルゴリズム
プロトコルドメイン	データの受け渡し・伝達手段の実現
デバイスドメイン	ハードウェア信号の入出力と信号の変換

[図4] ドメイン分割

ドメイン名とドメインごとの責務をまとめた表

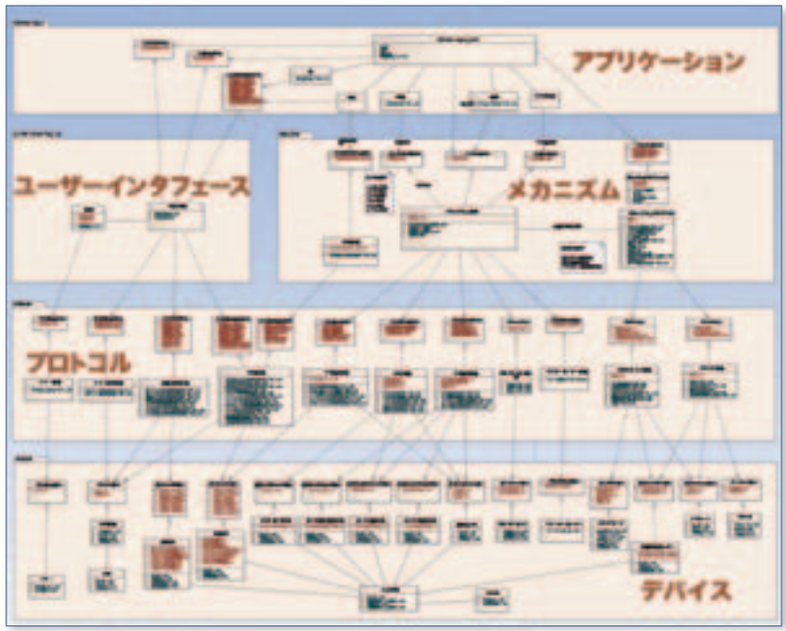
次に、Simulinkモデルの作成の効率化、スムーズな並行開発実現のため以下の定義を行った。

・ドメイン間のインターフェース定義（図5参照）
・UMLからSimulinkへのマッピングルール定義

以上のように、分析によるモデル間低依存度設計とルールの定義により、チーム員の要求仕様に対する認識レベルが統一化され並行開発が容易になり開発効率の大幅な向上が実現できた。

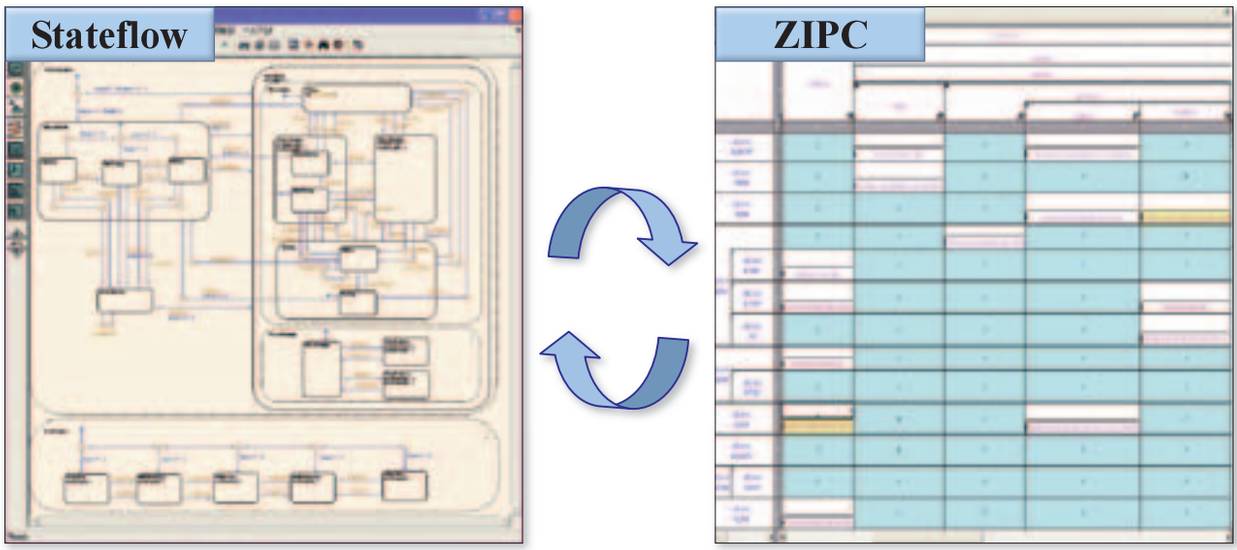
また、分析により分割されたドメインに対し、ZIPCが得意としている状態遷移設計に最も適したドメインを検討した結果アプリケーションドメインと、ユーザーインタフェースドメインを対象にZIPCを適用することにした。

Stateflowにて作成した状態遷移図モデルからSMCでZIPCに自動変換して、検証対象モデルの状態遷移表を作成し、この状態遷移表を使ってヌケ・モレの確認を実施した結果、ドアの挟み込み時の処理や前回動作条件の通知など未実装な状態遷移を発見することができた。（図6参照）



［図5］ドメイン間のインタフェース

ドメイン間のインターフェースを明確にし、
システム全体のデータの流れをまとめた。



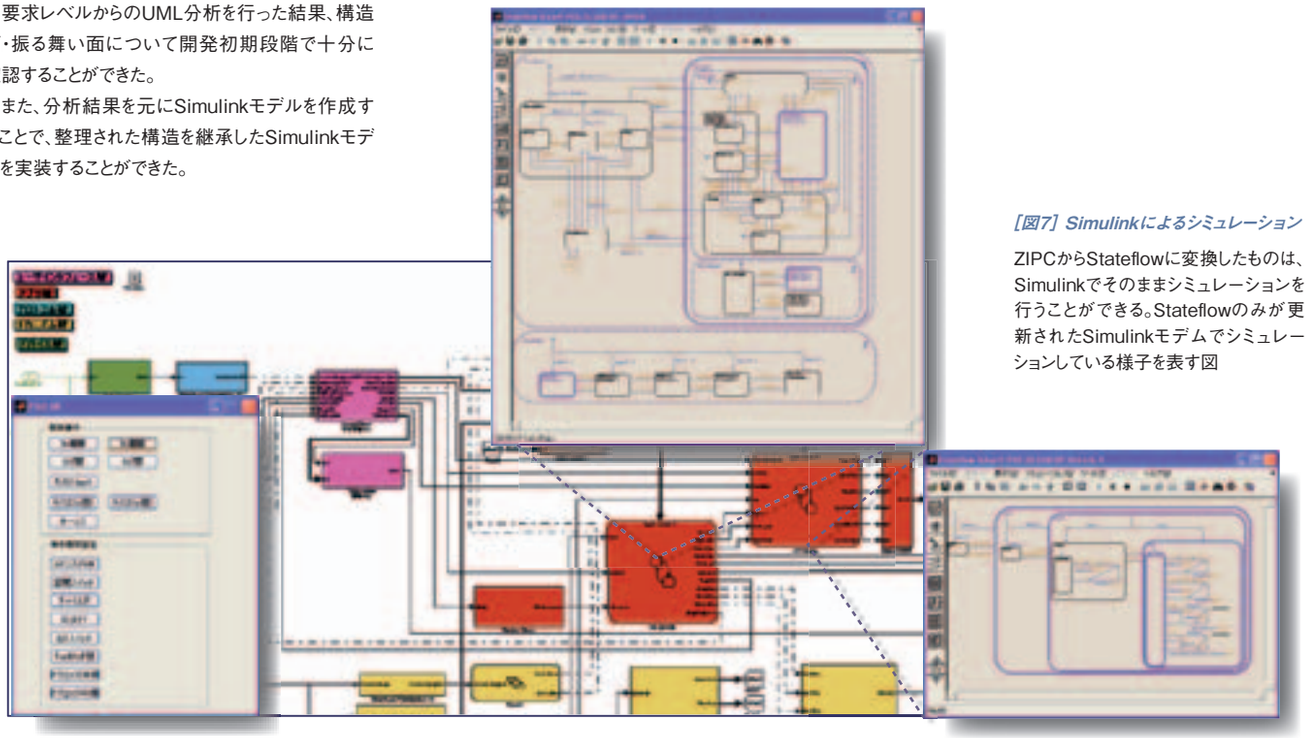
［図6］SMCによるSZ⇔ZS変換

Stateflowの状態遷移図をZIPCに変換し、ZIPCで検証したものを
Stateflowに再変換してモデルの精度を高めている様子を表す図

4. 導入効果

要求レベルからのUML分析を行った結果、構造面・振る舞い面について開発初期段階で十分に確認することができた。

また、分析結果を元にSimulinkモデルを作成することで、整理された構造を継承したSimulinkモデルを実装することができた。



[図7] Simulinkによるシミュレーション

ZIPCからStateflowに変換したものは、Simulinkでそのままシミュレーションを行うことができる。Stateflowのみが更新されたSimulinkモデムでシミュレーションしている様子を表す図

さらに、MBDを適用した効果として、開発者間でのレビュー時に共通の認識で情報を共有することができたほか、シミュレーションにより開発対象の動きを視覚的に検証できたため、コミュニケーションツールとしての効果が大きく、並行開発を円滑に行うことができたと言える。

本事例で、課題としてあげていたStateflowのヌケ・モレの早期発見については、状態遷移を中心に設計されたドメインに対してZIPCを適用した結果、設計段階での検討不足による手戻りを削減することができた。

また、上記ヌケ・モレ検討後のStateflow は、Simulinkモデルにそのまま組み込むことができるため、Simulinkでのモデル全体のシミュレーションをシームレスに行うことができた。(図7参照)

以上の結果として、MATLAB /Simulinkを使用したモデルベース開発を行うことで、モデルによるコミュニケーションと並行開発を実現し、開発プロセス全体のフロントローディングを行うことができた。

さらにZIPCを連携させることで手戻りの削減に対して更なる効果をあげることができた。

また、しっかりとした要求分析を行うことでモデルを整理することができ、整理・整頓された設計図に基づく可読性・保守性の高い再利用可能なモデルを実現することに成功した。

5. 将来の展望

本事例では、MATLAB/Simulinkでのモデルベース開発を前提にZIPCの適用を検討したが、ZIPC自身の持つさまざまな機能の一部を使用したにすぎない。

また、ZIPCも統合開発環境として上流から下流までの開発を一貫して行うことができる有効なツールであり、連携ツールを活用することであらゆる組込み開発をサポートできると考えられる。

当社としては、MBDにおけるZIPCを適用した開発手法のルール化やガイドラインの整備などを行うことで、より効率的なZIPCの活用法を確立し、ZIPCでの開発を検討しているお客様に対しても提案できるようにしたい。