NECエレクトロニクスの車載ソフト開発環境

ＮＥＣエレクトロニクス株式会社
自動車システム事業部　マネージャ
橋本　一也

1980年代の乗用車排出ガス規制から始まったクルマの電子化は進展し、今や自動車一台に50個以上ものECU（電子制御装置：Electronic Control Unit）が搭載されるようになってきました。クルマの機能開発のテーマも、「走る」「曲る」「止る」から「安全」「環境」「快適・利便」に移ってきました。これらの開発テーマは人間に関わりがより近くなってくるため、機能を実現するためにはより一層複雑な制御を行ったり、いくつかのECUと連携して制御する必要がでてきています。このような複雑な制御動作を実現する場合には、ハードウェアにも増してソフトウェアがその処理の主体になります。そのため、制御ソフトウェアの規模は年々大規模化、複雑化が進んでいます。

このような問題に対し、解決を期待され、注目を向けられている手法がモデルベース開発です。モデルベース開発は、従来からあった考えですが、制御理論の進展、モデリング手法やシミュレーション手法の高度化、またそれを支援する設計ツールの充実、といったことにより、実用的なものになってきました。

本稿では、モデルベース開発を中心に、車載ソフト開発への適用と、それに必要な技術やツールについて考えてみます。

尚、本稿で述べる「モデルベース開発」や「MILS／SILS／PILS／HILS」などの定義や応用事例は、各社毎の定義があり、また実装の方法により位置付け・意味付けが異なってくることもありますので、注意していただくよう予めお断りしておきます。

１．モデルベース開発とは

まず、モデルベース開発とはどういったものでしょうか？

そもそもモデリングとはモデルを作ることですが、モデルとは「既存システムや設計・製作すべきシステムを表現するための有用な特徴を表現し、そのシステムについての知識を取り扱いやすい形で与えるもの」（Eykoff “ System Identification-Parameter and State Estimation ”）ということです。

典型的な制御系モデルをベースに一般的な制御について見て行きましょう。

●フィードバック制御とシーケンス制御

図１は制御系システムで使われるフィードバック制御の典型的な構造を示しています。

制御器は、目標値と制御量との差を計算し、その差が小さくなるよう制御対象を操作します。この制御器にマイコンが使われます。制御系システム外からも制御に影響を与えます。これを外界・外乱と呼び、また、制御対象と外界・外乱を合わせてプラントと呼びます。

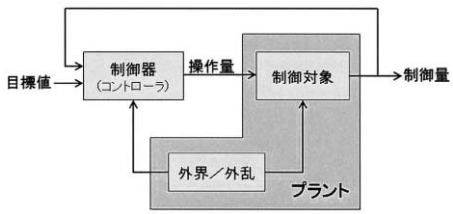

図１　フィードバック制御の典型的な構造

目標値だけ与えれば、制御器がその目標値に近づくように制御して行くため、制御対象についてほとんど知らなくても自動的に制御できるのがフィードバック制御の特徴です。ECUシステムではこのフィードバック制御が多く利用されています。

一方、「シーケンス制御」は、一連の決められた流れを自動的に行うものであり、マイコンが

重要な役割を担っています。メカのコントロール、UI部分や通信プロトコルなどにその例を見ることができます。このような制御では制御ロジック中心の設計になりますが、制御ロジックのヌケ・モレをチェックできるキャッツ社製ZIPC / ZIPC++などのツールが効果的に使えます。

●モデル検証

さて、このフィードバック制御を行うソフトウェアの検証はどのように行われるのでしょうか。フィードバック制御では時々刻々データ（制御量や目標値など）が変化してゆきます。そのフィードバック制御システムの検証を容易に実現するのがモデルベース開発環境です。

モデルベース開発は制御しようとしている対象の動作を表現するモデルを作成し、そのモデルに基づいて制御系を設計する手法です。図2にモデルベース開発のプロセスを示します。手順は、まず、ドメイン知識や実験データを基に、工学的問題モデルを駆使し、モデルを作成します。モデルは、制御器、制御対象、外界すべてについて作成します。作成したあとは、モデルを実行するシミュレータなどを使って確認を行い、モデル・パラメータの理論値を求めます。次に、このモデルをベースに制御器上の制御ソフトウェアを開発します。モデルからプログラムを生成するツールもあります。また、制御系のふるまいが求めた理論値に近づくように実装用のパラメータをチューニングして行きます。確認が終わったら実装し、実測値によりさらに詳細にパラメータのチューニングを行います。

このパラメータの数は数百～数万のオーダーになるといわれており、それらを全て正確に把握することは難しく、実機（実ECU）により実際の動きを見て決めていく試行錯誤的なやり方では対応できません。理論的にモデルを求めて、より上流で検証して行くモデルベース開発が必須となってきています。

●モデルベース開発のメリット・デメリット

ここで、モデルベース開発のメリット・デメリットを整理しておきましょう。

モデルベース開発では、各工程でそれぞれのレベルでの検証を行い、より上位レベルで検証することにより、下位工程での不具合による後

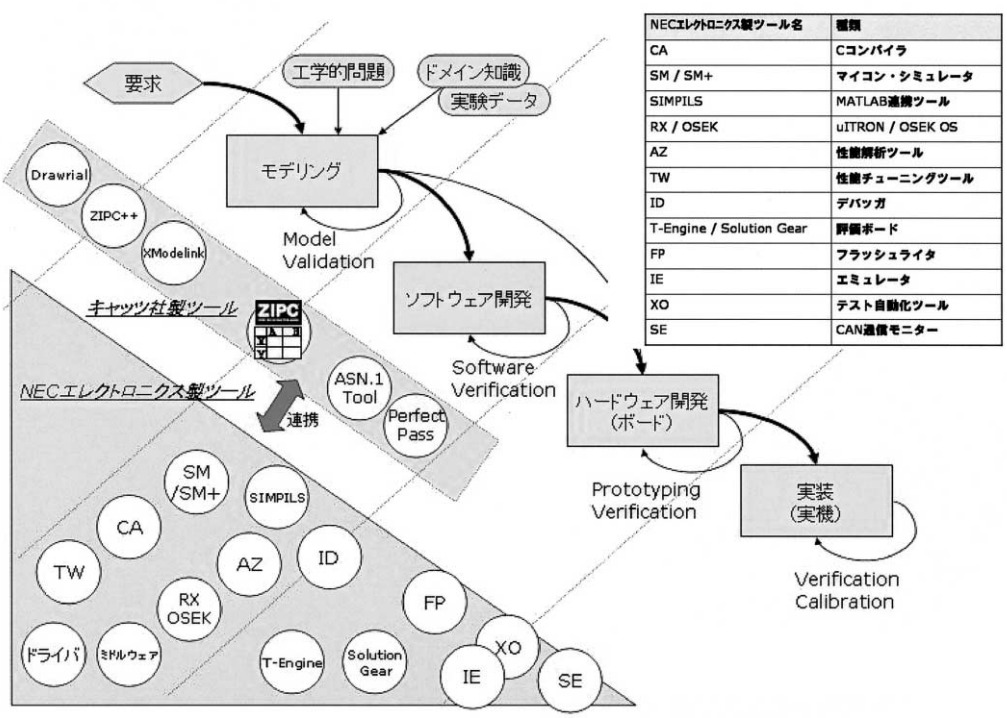

図2　開発プロセスと開発ツール

戻りを少なくすることができます。より上位レベルで数万個もあるパラメータのうち大多数を決める場合もあります。このことは、開発TATの短縮や品質向上に大きく貢献します。

また、モデルベース開発では、各レベルでのデバッグを容易に実現できます。マイコン組込システムの実機では、実行を中断させると制御対象に悪影響を及ぼすことから、デバッグ時などで実行停止させることができないことが多いのですが、モデルならそれが可能です。任意の場所でプログラムをブレークさせ、状態を確認するようなデバッグが可能です。

他に、モデル言語での記述は見た目にわかり易く、開発関係者全員の理解が統一できるので、設計資産化の助けになるのもメリットのひとつです。このモデルを「動く仕様書」と呼ぶこともあります。

デメリットは、制御対象のモデル作成が難しい点と、環境がまだ充実していない点です。

複雑な動きをする制御対象や人・路面状況・気候などの複雑な条件をモデル化するのは非常に難しいため、モデル作成に比較的大きな工数が必要である場合があります。実測値などからモデルを作成するシステム同定手法もありますが、細かい条件をモデリングするのは非常に難しいことといえます。作ったモデルは次の開発でもできるだけ使えるように流用率を高めるなどの工夫が必要です。

もうひとつのデメリットは、上位の検証工程で作ったプラントモデルやテストベクタを含むテスト環境（テストベンチなど）が、下位工程でそのまま使えるようになっていない点です。そのため、各検証フェーズでそれぞれのテスト環境を新たに構築しなければならず、作業効率が悪くなります。上位で作成したテスト環境を下位の検証環境でも使えるシームレスな環境が望まれます。

当社では、メリットをより効果的に使えるように支援し、デメリットは改善できるよう取り組んでいます。

２．モデルベース開発の適用の現状

では、車載ソフト開発ではどのようにモデルベース開発環境を使っているのでしょうか。

図２に示したように、キャッツ社製ツール群と当社製ツール群を連携してモデルベース開発の全てをサポートできればよいのですが、車載ソフトを開発するには不十分です。

図３は、一般的に使われるＶ字型プロセスと検証環境を示しています。ECUひとつの開発プロセスを表しています。開発プロセスの各フェーズで、検証環境が異なります。

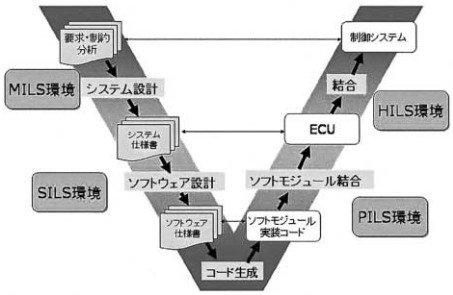

図３　Ｖ字型プロセスと検証環境

図４は検証環境を構成するコンポーネントの関連を表しています。ターゲットプログラム、制御器およびプラントの組み合わせを示しています。何を検証するかにより、検証の形態が異なってきます。

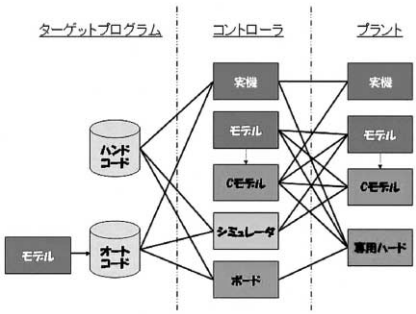

図４　検証環境の構成コンポーネント

例えば、ターゲットプログラムのハンドコード（実機上に搭載される制御ソフトそのまま）をデバッグしたい場合、制御器としての実行プラットフォームをマイコンシミュレータで、プラント側をモデルという組み合わせで検証環境を実現することができます。

ここで、Ｃモデルは、ホストマシン（ＰＣ）上でコンパイルされて、ホストマシンやそれに換わる専用マシン（例えばHILSハードウェア）で動作するモデルのことです。また、ターゲットプログラムは、マイコン用コンパイラでコン

パイルされたコードのことで、モデルを参照しながら人手で作成されたものをコードをハンドコードと呼び、モデルから自動生成されたものをオートコードと呼びます。

クルマ業界では、HILSやSILSという言葉が使われていますが、これはモデル検証の形態のことです。ここでは次のように定義します。

●MILS（ Model-In-the-Loop Simulation ）

制御器やプラントをモデルで表現し、モデルを実行するシミュレータなどを使ってシミュレーションを行う検証環境です。

まず、システム設計では、システム全体を、実システムから観測された入出力データに基づいて得られた数式モデルや、理論から求められた数式などで表されるモデルを使って検証して行きます。

●SILS（Software-In-the-Loop Simulation ）

制御器をソフトウェア（ Cモデル ）で実装し、制御対象や外界はモデルで検証を行う形態です。ソフトウェア設計では、ソフトウェア中に実装される状態遷移などのアルゴリズムを用いて、Cコード化しても前記MILS検証で検証済の動作が同じかどうか検証します。

●PILS（ Processor-In-the-Loop Simulation ）

ターゲットプログラムを使った検証環境です。制御器をターゲットプログラムとその実行プラットフォーム（マイコンモデル）で実現した検証環境です。マイコンモデルには、マイコンシミュレータやマイコン搭載ボードなどが利用されます。

●HILS（Hardware-In-the-Loop Simulation ）

HILSは、制御器に実機（実ECU）を用いて実現し、制御対象や外界はプロトタイピングのモデルで検証を行う形態です。

従来から使われているのは、HILSであり、関連した製品ツールもいくつか存在しています。

現状では、必ずしも、MILSから始まって、SILS→PILS→HILSと段階的に検証していくことにはなっていないようです。MILS検証後、実機検証という開発パターンもあります。「動く仕様書」としてMATLAB/Simulinkなどモデル記述が定着し、開発環境が整備されると、MILS、SILS、そしてPILSといった流れの開発が普及してゆくことでしょう。

３．車載ソフトの特徴

さて、車載ソフトには、他のソフトと異なる特徴があり、この点に対応した開発環境も考えなければならない点があります。

●車輌内LAN

クルマの電子制御が進んだことで、各ECU間のデータ量は拡大の一途をたどっています。また配線に要するワイヤ量が膨大になってきており、車輌内LANの採用が進んでいます。このLAN上に、ECUが数十個接続されていますが、機能の高度化によりECU単体の機能では実現できず、複数のECUと連携してひとつの機能を実現しなければならなくなってきました。従って、ECUの制御ソフトも他のECUの機能と連携してリアルタイム処理を行わなければなりません。「分散システム」ではなく、所謂、「分散リアルタイムシステム」です。

●ソフトの標準化

先に述べたように、制御動作はハードウェアからソフトウェアが主体になってきています。それゆえ、自動車メーカの独自性もソフトウェアに移ってきています。しかしながら、ECUの数が増えるにつれソフトウェア開発コストも膨大になってきているため、メーカとしては可能な限りソフトウェアの再利用を進める必要があり、独自性に関わらない部分については、自動車メーカ（OEM）や電装メーカ（サプライヤ）などの関連メーカが協調して開発することで、開発コストの削減および技術開発の促進を図る必要がでてきました。

欧州では既に取組みが進んでおり、車載ソフトの国際標準化団体AUTOSAR（Automotive Open System Architecture）を立ち上げ、活発な活動を行っています。AUTOSARは、自動車メーカと、主な電装メーカが中心となって、クルマのソフトウェア、電子アーキテクチャの共通化を目指すコンソーシアムです。AUTOSAR

で決めていく領域は、

・ソフトウェア・アーキテクチャ
・開発プロセス
・実装方法

であり、車種、OEM、サプライヤを超えたソフトウェアの流通や再利用を実現することが目標です。

また、日本国内では、2004年９月に、JasPar（Japan Automotive Software Platform and Architecture）が発足しました。APIやミドルウエアの標準化や車輌内LAN規格の標準化を図ることを目的としています。FlexRay関連事項を当面の活動としています。

各社が独自で進めてきたクルマの電子化が、競争領域でない部分で標準化される中、今後の活動が注目されます。

●高信頼性

人間を乗せて走るクルマには、極めて高い信頼性が求められます。信頼性の高い車載ソフトを完成させるためには、クルマ用の特徴的なツールを駆使して検証して行かなければなりません。例えば、複雑な動きをするクルマを制御するための理論とその理論を検証するためのモデリングツールや複雑な動作を特定するパラメータをチューニングするための適合ツールなどです。

また、車輌内LANを使った分散システム構成の中で複雑なリアルタイム処理の信頼性を確保するのは非常に難しい作業となります。分散リアルタイムシステムの信頼性向上は、今後の課題のひとつになります。

４．当社の取組み

従来の開発プロセスを変更するのは過去の資産の関係で非常に難しいことですが、分散リアルタイムシステム開発プロセスの適用とオートコード技術（後述）の導入により車載ソフトの開発プロセスの変革が可能になると思われます。

この開発プロセスの変革に備えて、当社の開発環境への取組みを説明します。車載ソフトだけではなく、それ以外のマイコン組込みシステムに使えるものもありますが、ここでは車載ソフト開発用について、特に車輌内LANに焦点をあてた車載ソフト開発環境について説明します。

車載ソフトの特徴をサポートする当社の取組みとしては大きく３つあります。

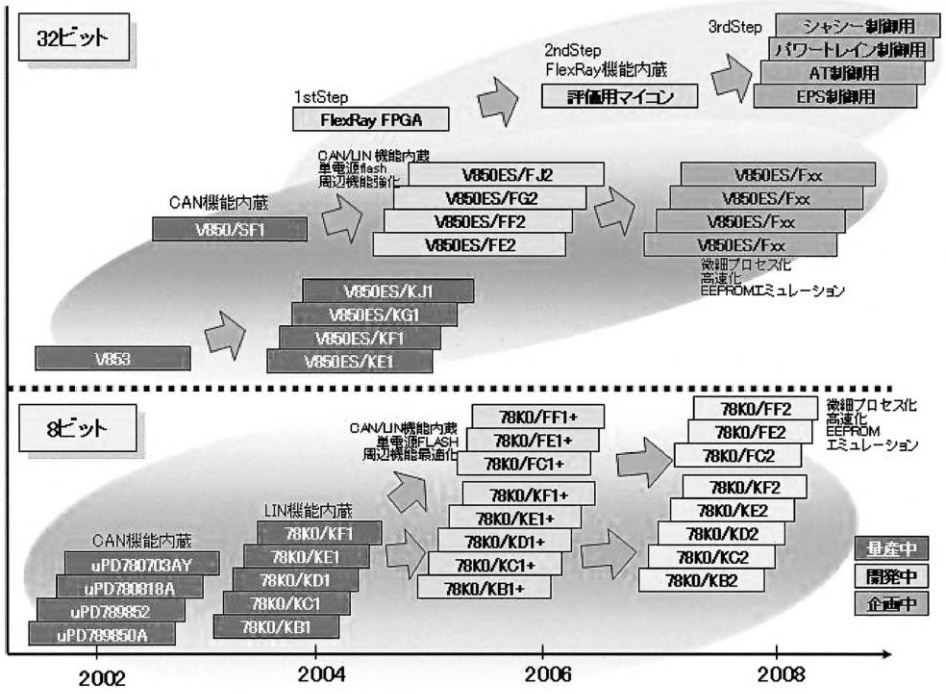

図５　NECエレクトロニクス製車輌内LAN機能搭載マイコンロードマップ

（1）車輌内LAN機能内蔵マイコンの提供

図５は、当社の車輌制御系のLAN機能内蔵マイコンのロードマップです。業界標準で使われているCAN（ Controller Area Network ）をはじめ、LIN（ Local Interconnect Network ）のコントローラを内蔵したマイコンを豊富なラインナップでご用意しています。

最近ではクルマ以外でもこのCANやLINが使われてきており、今後の普及が期待されます。

（2）標準化への対応

CANはイベントドリブンで、データを送りたい時に送信する仕組みですので、複数のECUが一斉にデータを送信した場合は、遅延してしまい、全体の性能に影響することもあります。高性能化／複雑化してきた機能の実現において、CANでの対応は限界が見えてきました。そこで、最近注目されているのが、FlexRayです。FlexRayは時分割でデータを送信するため、遅延なしでデータの送受信ができ、システム全体の信頼性を上げることができます。特に、従来油圧などの機械的な機構で実現していたブレーキやハンドルの操作を電気的な機能に置き換えるX-by-Wire技術を実現するためには、さらに多くのデータを高い信頼性で通信する必要があるため、次世代LAN規格として注目されています。

当社ではFlexRay技術開発にも積極的に取り組んでいます。FlexRayコンソーシアムをはじめ、ソフト標準化を行っているAUTOSARやJasParにも参加し、積極的に活動を行っており、標準化への貢献を果たしています。そして、標準化された次世代プラットフォームをいち早くお客様に届けられるように努力しております。

（3）車載ソフト開発ツールの整備

●分散リアルタイムシステム開発環境

図６は、分散リアルタイムシステムの開発プロセスとツールを表しています。

分散リアルタイムシステム開発では、従来と比べて、「マッピング」と「スケジューリング」のフェーズが重要になってきます。

マッピングとは、ソフトウェアコンポーネントとハードウェア資源との対応付けを行う工程です。スケジューリングは、例えば、FlexRayなどは時分割で通信スケジューリングを行う際に、いつデータを送信すべきかなどのスケジュールを予め設定しておくことです。

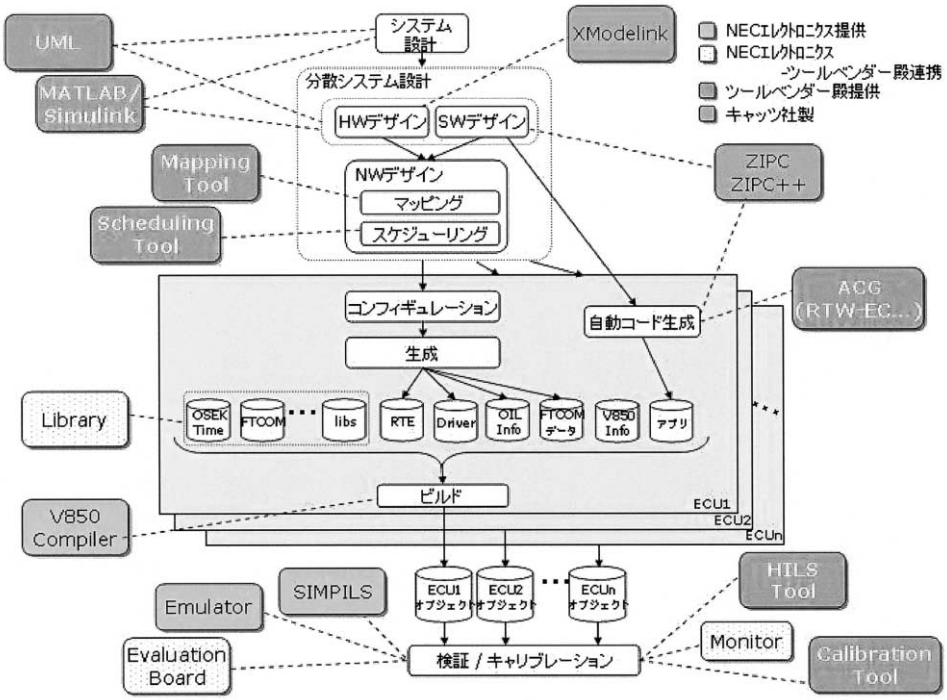

図６　分散リアルタイムシステム開発フロー

このマッピングとスケジューリングの情報を基に各ECUが連携できるようにECU毎に必要な情報ファイルを生成します。

このように分散リアルタイムシステム用ソフトを開発するためには、新たなツールも必要になってきます。当社では、それぞれの分野で強力なツール持つベンダー殿と協力して整備を進めています。

●MATLABとのコラボレーション

MATLAB/SimulinkはThe MathWorks社の製品であり、クルマ業界では標準的に使われているツールです。当社では、このMATLAB/Simulinkを基盤ツールとして位置づけ、その開発環境の整備を行っています。このMALAB/Simulinkと当社のマイコンシミュレータSM/SM+シリーズとを連携させる環境がSIMPILS™です。SM/SM+マイコンシミュレータは、マイコンが持つ周辺機能までシミュレートしており、精度良いシミュレーションが可能となります。プラント側をMATLAB/Simulinkで記述し、制御器側をマイコンシミュレータで実装することにより、閉ループ（フィードバックループ）のシミュレーションができます。図７は、SIMPILSを使ったトランスミッションのデモ画面です。制御器となるシミュレータ用のSimulinkブロック（V850ブロック）を配置し、V850マイコンの端子に相当するポートをプラント側と結線して行きます。また、CANブロックを介してCAN通信を可能としており、実機上の制御ソフトをそのまま実行させることが可能です。

この機能は、

- システム設計時の検証データを使いながら、制御ソフトを含めた系の検証
- 実機上の制御ソフトをそのまま実行させ、デバッグ
- 回路シミュレータなどを併用して、ハードウェアの回路検証／アルゴリズム検証

などに利用できます。

また、このSIMPILSの仕組みを使って要求記述・分析ツールUMLとも連携を推進中であります。利用シーンに合わせて、ご利用いただけるように検討しています。

５.「安全」「環境」「快適・利便」を実現するために

以上、モデルベース開発について概観し、車載ソフト開発環境の構築に向けて当社の最近の取り組みを述べてきました。

しかし、安全性を保証しようとすると、これでは足りません。安全なソフトウェアを作るためには、確実に意図したとおりのコード生成が必要となります。つまり要求通りに確実に動作することを保証しなければなりません。

これからの課題のひとつは、モデルとコードの一致性をどう保証するか、です。保証されたコードを生成するための効率の良い方法として、大きく２つのアプローチがあります。ひとつは、モデルから保証したコードを自動生成し、モデルからコードへの変換を保証する方法、もうひとつは、形式検証によりアルゴリズムを保証する方法です（図８）。

●オートコード生成技術

オートコード技術の進展により、実用的なオートコード生成ツール（ACG: AutoCode Generator）が出てきました。

モデルを作成・検証した後、そのモデルを参照しコードを人手で作成する、または、自動生成したコードを検証するのでは、作業効率が良くありません。できれば、モデルから保証されたコードを自動生成させたいところです。

保証されたコードを出力する製品レベルのツールも一部あるようですが、まだこれからの

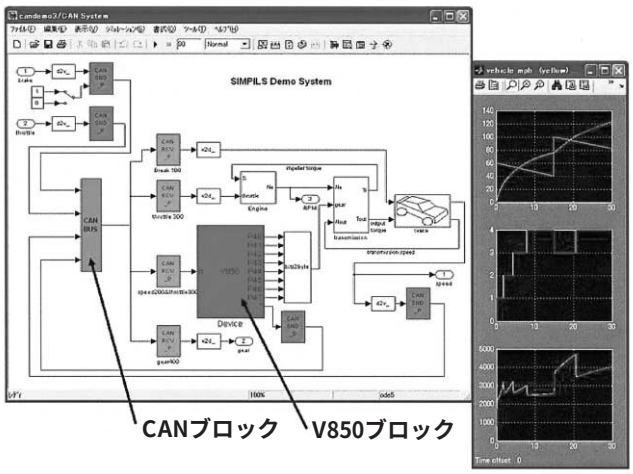

図７　MATLAB/Simulink連携環境

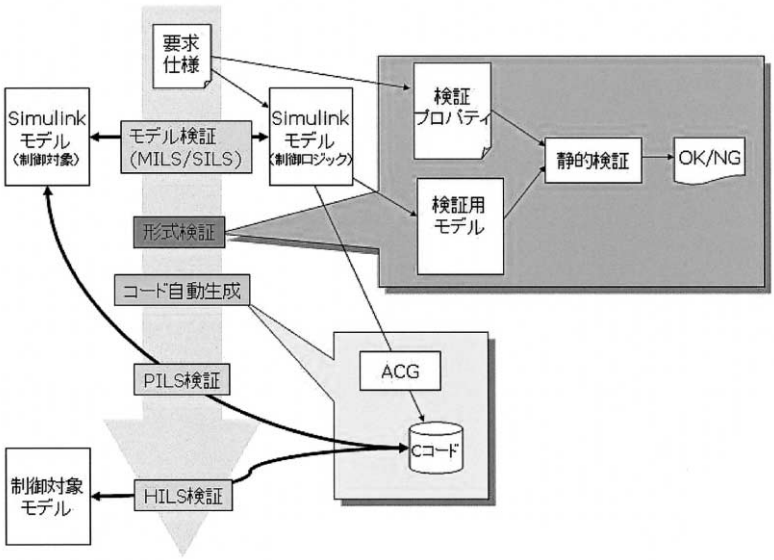

図8　モデルベース開発検証フロー

技術です。今後充実してくることと期待しています。

●形式検証

生成したコードが要求通りであることを保証しなければなりません。つまり、要求仕様通りにモデルで表現でき、コードが生成されなければなりません。要求仕様に対するモデルの保証です。

現在、キャッツ社は九州大学とFLEETS（福岡知的クラスター研究所）との産官学共同で状態遷移モデルの形式検証の研究を進めていると聞いています。複数の状態遷移表を組み合わせたときの不可セルへの未到達性を数学的に証明する研究です。この研究成果はZIPCの状態遷移表からとり得る全ての経路（テストケース）を抽出するキャッツ社製ツールのPerfectPassと組み合わせると強力なツールになることと思います。今後の進展が期待されます。

上記のような新しい技術の進展により、機械語からアセンブラ、アセンブラからＣ言語、と進展してきたように、数年後には、Ｃ言語からモデル記述という記述のステップアップは現実味を帯びてきました。制御モデルや制御ソフトは、制御開発者が作成し、電気回路モデルは回路開発者、電磁界モデルはモータ開発者、また、マイコンモデルは半導体メーカ、のそれぞれのモデル開発専門家が作ったモデルで、検証に必要な精度で、かつ必要な速度で、最適に組み合わせて検証して行くモデルベース開発の世界がすぐそこまで来ています。

NECエレクトロニクスでは、モデルベース開発に耐えうる世界最高レベルの性能を持つマイコンを含むプロダクトラインを整備するとともに、キャッツ社をはじめ強力なツールを持つツールベンダー殿と連携し、最先端の開発環境をいち早く提供して、クルマの「安全」「環境」「快適・利便」の実現に向けて貢献できるように努力して行きたいと考えています。