レガシー開発からモデルベース開発への移行ソリューション

モデルベース開発は組込み制御開発においてQCD(Quality,Cost,Delivery)の向上を実現する手法として注目されています。多くの企業でレガシー開発からモデルベース開発への移行を検討していますが、既存のプログラムの中にノウハウが集約されているため、乗り換えた後の品質に対する不安や、移行のためのコスト、期間といった課題があります。当社は、レガシーコードの分析・検証をサービスメニューに加え、移行のコスト／期間をできるだけ抑え、安心できる品質を確保するためのソリューションの提供に取り組んでいます。

品質の見える化を実現するモデルベース開発

モデルベース開発は、設計フェーズで作成したモデルを基にシミュレーションによる検証を行いながら開発を進めていく手法であり、設計工程でシミュレーションを行うことによって設計品質の向上効果が期待できるだけでなく、設計の不具合による後戻りを低減することによって生産性向上効果も得られます。

また、シミュレーションによる検証を行うことで、設計工程でもタイミングチャートなどを出力することができ、実機と同様の動作結果の見える化が実現できるほか、カバレッジ確認ツールの活用により、シミュレーションによって検証されたテストカバレッジも確認可能です。検証工程においては、シミュレータの活用や設計工程で作成した検証データの再利用による効率化が実現でき、モデルの再利用が進めば、開発期間／コストの大幅な削減が期待できます。

当社は、車載ECU開発やFA・エネルギー分野で、自社開発したリアルタイムシミュレータ「M-RADSHIPS」(Model-based Rapid Application Development Simple & High cost Performance Simulator)を提供しています。低コストでシンプルな検証環境を可能にしたM-RADSHIPSは、設計時に作成したモデルによる正確な検証や、実機では検証の困難なレアケースの検証を実現することで、モデルベース開発の利点である開発効率と品質の向上、開発コストの削減に大きく寄与しています。

モデルベース開発への移行時の課題

レガシー開発を行っているお客様は、モデルベース開発への移行に対して、「既存のプログラムの資産をどう活用できるのか」、「移行後の品質が劣化してしまうのではないか」、「移行コストがどれくらいかかるのか」、「移行期間が長くなると、二重開発になってしまう」、といった漠然とした不安を持っています。しかし、これらは、現行のシステムが抱えている課題が原因となって不安を大きくしているものであり、もう少し本質的な部分を見ると、下記のような課題に集約されます。

・設計書がメンテナンスされていない(ソースが真)
・要求からのトレーサビリティが不完全である
・仕様・ノウハウがソースに埋まっている

設計段階では矛盾する要求に対してどこでどのように回避するかが明示されておらず、その要求を調停する仕様を見出すためにトライアンドエラーを繰り返してシステムを構築するケースがあります(いわゆるすり合わせ開発)。こうして構築されたシステムでは、すり合せた結果を仕様書にフィードバックする機会を失うケースが多く、非常に重要な仕様がソースに埋まった状態になり、要求からのトレーサビリティも不完全になってしまいます。

いったんこのような状態になってしまうと、検証についても経験と勘に頼ることとなり、さらに、仕様変更／追加や派生開発を行う際には、過去から継承されてきているモジュール、ロジックはなるべく手を入れないようにして、周りのプログラムを後付けすることで対処することが多くなります。これを繰り返すことで、ますますプログラムが複雑化してしまうことになります。これが、漠然とした不安の根本的な原因です。

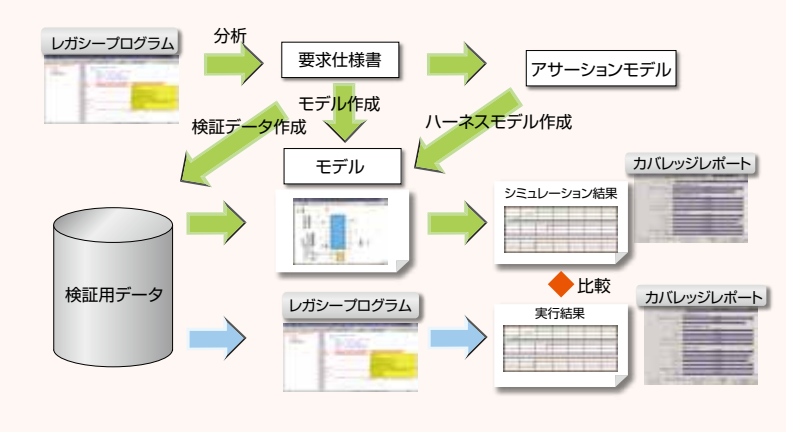

図-1 レガシー開発からモデルベース開発への移行

表-1　モデルベース開発サービスメニュー

プロセス	INPUT	内容	OUTPUT
レガシーコード分析	レガシーコード（Cソース）	メトリック分析	メトリック分析結果
		構造分析	構造分析結果
		要求分析	要求一覧
		方式／方針検討	方式検討書
要求分析	要件一覧	ユースケース分析	ユースケース図
		シナリオ作成	シナリオ
		ロバストネス分析	ロバストネス分析図
		タイミングチャート作成	タイミングチャート
システム設計	要件一覧 ユースケース図 シナリオ ロバストネス分析図 タイミングチャート	設計モデル作成	設計モデル
		プラントモデル作成	プラントモデル
		ハーネスモデル作成	ハーネスモデル
		アサーションモデル作成	アサーションモデル
		設計検証データ作成	設計検証データ
		MILS検証	設計検証結果／カバレッジレポート
レガシーコード検証	レガシーコード	レガシーコード⇒s-function	MILSモデル
		MILS検証	設計検証結果／カバレッジレポート
		結果レポート作成	レガシー⇒モデル　比較結果レポート
ソフトウェア設計～ソフトウェア検証	設計モデル	実装モデル作成	実装モデル
		自動生成	Cソース
		アサーションモデル作成	アサーションモデル
		SILS検証（BackToBack検査）	検査結果／カバレッジレポート
機能検証／システム検証	Cソース ECU 設計検証データ ハーネスモデル アサーションモデル	HILS構築	HILS環境
	設計検証データ	HILS検証	検証結果（検証ログ、タイミングチャート）
妥当性検証	設計検証データ	実機検証	検証結果（検証ログ、タイミングチャート）

になります。

当社は、モデルベース開発のサービスメニューに、「レガシーコード分析」と「レガシーコード検証」を加え、ソースに埋まっているノウハウを抽出／分析して要求仕様書を作成し、モデル化するサービスの提供を開始します(**表-1**)。レガシーコード分析においては、プログラムの構造や依存関係を分析しメトリック化するツールの活用や、分析に必要なエッセンスを取り出していくための仕組みの構築を行っており、今後当社独自のツール化も検討しています。

アサーションモデルの活用とレガシーコードからの要求抽出

一方、モデルベース開発では、設計工程でシミュレーションによる確認を行うことができるため、矛盾する要求に対する調停を行い仕様を策定する必要がある場合も、シミュレーションによって確認を行うことができ、その時点でモデル(設計)にフィードバックされるため、トレーサビリティの完全性も保たれます。

もう一つ、最近注目されている手法として、アサーションモデル(要求通りにできているかを検出し正しく制御していることを検証するためのモデル)を活用した検証があります。

作成したモデルにアサーションモデルを付加し、要求分析時に検討したユースケースやシーケンス、状態遷移を基に作成した検証データを使ってシミュレーションを行うことにより、机上では検討しきれなかった問題の発見を促すことができます。さまざまなユースケースを組み合わせた検証も容易になるため、検証の網羅性の向上や、テストカバレッジを確認するツールとの組み合わせによる品質の“見える化”を実現することができます。

レガシーコードをモデル化した場合も、この仕組みを利用してレガシーコードとモデルが完全に同じふるまいをすることが確認できます(**図-1**)。

残りの課題は、レガシーコードをモデル化する部分です。ここで注意しておきたいのは、ソースコードを忠実にモデル化することは、乱れた構造をそのままモデル化することになり、モデル化する意義を失ってしまうため避けたい、ということです。

検証の仕組みは完成されているため、少ないリスクで新しい構造にてモデル化することができるはずです。レガシーコードに暗黙的に埋め込まれてしまっている要求を抽出／分析することができれば、さらに効率よくモデルベース開発への移行ができること

機能安全のソリューションも提供へ

当社では、前述したようなV字プロセスの上流だけでなく、下流の検証プロセスにおいてもトレーサビリティを確保し、品質と効率を高めるための取り組みを進めています(**図-2**)。

自動車制御分野における機能安全規格ISO26262で規定されている検証項目では、故障注入テスト(Fault Injection Test)が推奨され、通常時だけでなく故障発生時にも正しく制御ソフトウェアが動作することを確認・検証できる仕組みが求められています。

任意のタイミングで任意の故障を起こせるような仕組みを設けるためには、シミュレータが必要不可欠であり、当社では

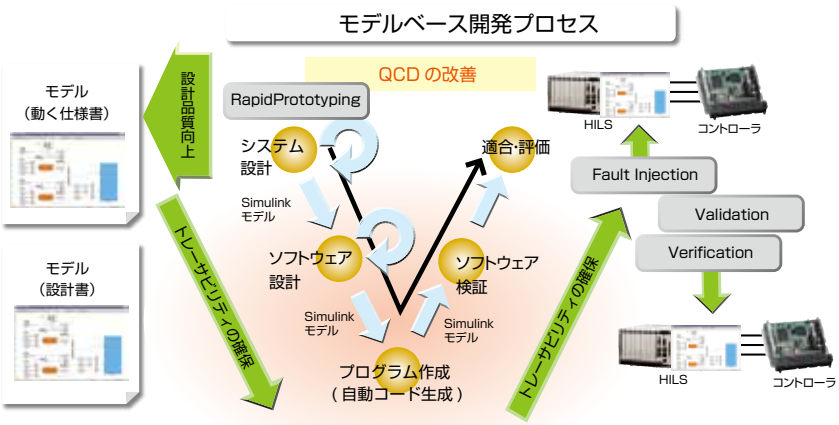

図-2　プロセス全体を通じたトレーサビリティの確保

M-RADSHIPSを使って故障を注入する仕組みの構築／提案を行っています。

最近では機能安全に対応したマイコンも登場し、マイコン内に故障が起きたとしてもそれを検出し、システムを安全に保つ仕組みを構築できるようになっています。当社では、この仕組みを検証するシステムの構築を東芝セミコンダクター&ストレージ社と一緒に取り組んでおり、将来的には製品化を視野に入れて開発しています。

(エンベデッドシステム事業部　三島隆司)