熟練者の知識流通を支える開発環境

株式会社デンソークリエイト　小林 展英
株式会社チェンジビジョン　平鍋 健児

組込みソフトウェア業界にも「つながる世界」が到来し、開発に携わるエンジニアは、つながる先の製品に関する知識も獲得しながら開発を進めることが期待されている。しかしながら、一人のエンジニアの努力だけで獲得できる知識量には限界があり、ほかのエンジニアが獲得した知識を資産化して流通させることが不可欠となりつつある。本稿では、組込みソフトウェア開発に関する知識を概観し、その知識の資産化と流通に用いるモデリング言語とその記述を支える開発環境について紹介する。

1 組込みソフトウェア開発環境が求められる背景

現在の組込みソフトウェア開発では、つながる世界を想定することが当たり前となっている。従来つながりのなかった製品がつながることで、開発の仕方や製品品質に対して異なる文化を持ったエンジニアとの接点が生まれ、自分たちの開発では当たり前と考えていた確認作業が履行されないなど予期せぬ問題の発生が予想されている。こうした状況を回避するため、組込みソフトウェアの開発に携わるエンジニアには、自分の開発する組込みソフトウェアが搭載される機器に対する深い知識だけでなく、その機器がつながる様々な機器に対する幅広い知識が求められている。

一方、実際の開発現場では、上述した課題に対して、開発チームを牽引する熟練したエンジニア個人の知識に依存して対応している状況が少なからず存在している。しかしながら、つながる世界の広がるスピードを考えると、個人の努力だけで対応し続けるのには限界があり、熟練したエンジニアの分析・設計手法や再利用可能なパターンを資産化し、開発チームの誰もが容易に利用できるよう流通させることが不可欠となりつつある。図1に筆者が期待するつながる世界の組込みソフトウェア開発における開発環境のイメージを示す。本開発環境が対象とする組込みソフトウェアは、アプリケーション部分とプラットフォーム部分に大別されており、製品共通の機能を実現したプラットフォームを再利用することでエンジニアが開発する範囲をアプリケーション部分のみに限定している。更に、熟練したエンジニアが分析・設計時に用いるモデリングの仕方を「標準設計手法」として資産化し、更に実際のモデリング結果から有用なパターン（ソフトウェア構造や製品に共通して想定されるリスクなど）を抽出して「再利用可能な設計資産」として流用できる環境としている。

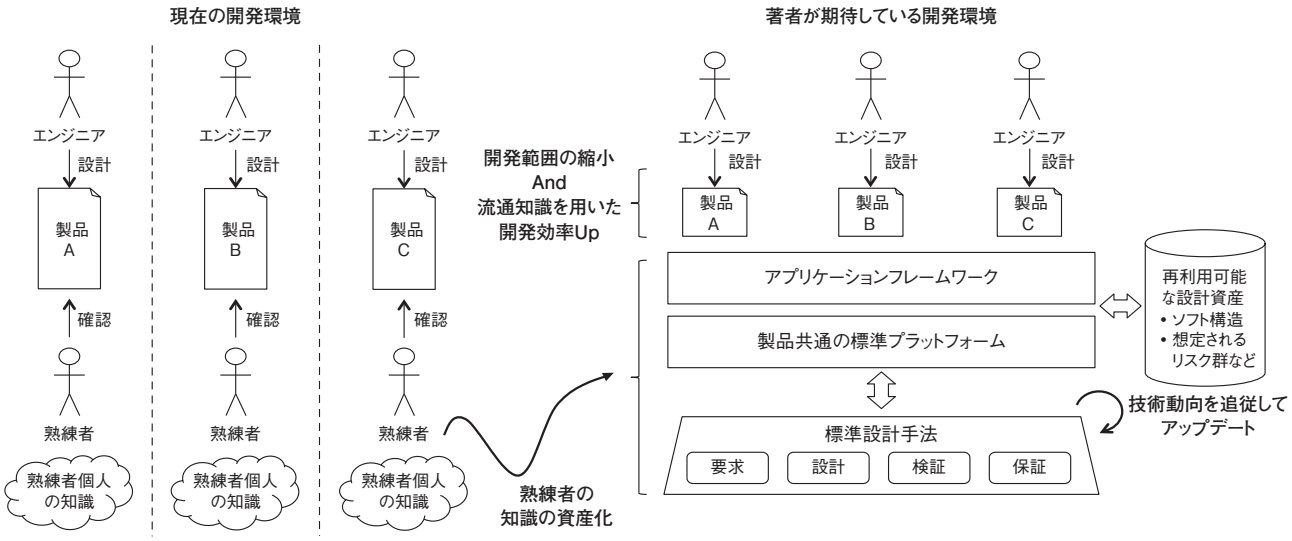

図1　熟練者の知識を活用できる組込みソフトウェア開発環境

次節以降では、つながる世界の組込みソフトウェア開発に携わる熟練エンジニアの知識流通を支えるモデリング言語について２節で概観し、更にその記述を支援する代表的な開発環境について３節で紹介する。

2 組込みソフトウェア開発に用いられるモデリング言語

組込みソフトウェア開発に関連する代表的なモデリング言語を図２に示す。情報システムに搭載されるソフトウェア開発との大きな違いは、開発対象のソフトウェアが搭載される機器に起因する性能要求、安全要求などの非機能要求を満足した上で、実世界の物の挙動を記述した制御モデルの振る舞いをソフトウェアとして実現する点である。

非機能要求を扱うモデリング言語としては、非機能要求を体系的に表現できるNFR Frameworkが存在している。安全要求に対しては、分析手法としてスタンダードなHAZOP、FTA、FMEAなどに加え、つながる時代に向けて最近注目されている、システム思考に基づく新しい手法としてSTAMP/STPAがある。また、セーフティ要件の分析結果を統合し、システムの安全性の保証のための議論構造を表現する記法としてGSN（Goal Structuring Notation）が考案され、日本ではD-Caseとして拡張されている。そのほか、ハザードや安全要求などのシステムの安全に関する情報を記述できるよう、SysMLを拡張したSafeML、ISO26262などの機能安全におけるシステムの安全設計をアーキテクチャ視点から整理し、論じるための記法SCDL（Safety Concept Description Language）なども考案されている。なお、自動車業界を事例に挙げると、これまではセーフティに注力するだけで良かったが、つながる世界の波が自動車業界にも届いた結果、セキュリティに関しても考慮する必要が出てきている。今後、SafeSecのようにセーフティとセキュリティを統合して扱える手法の考案が期待される。

一方、制御モデル設計については、制御対象となるプラントとそのコントローラをモデリングする必要があり、SimulinkやModelicaといったモデリング言語が広く使われている。両者は共に物理世界の数式や制御式を基本としており、本稿では「連続モデル」と呼ぶこととする。また、情報を扱うオブジェクト指向ソフトウェアの一般的な記述から始まったUMLが企業の情報システム構築で使われるようになり、現在では組込みシステム開発でも利用されるようになってきている。特に組込みソフトウェア開発では状態遷移図を中心としたモデルが広く使われており、本稿では「離散モデル」と呼ぶこととする。

最後に、システムズエンジニアリングにもモデルベースが適用され、汎用的に利用できるSysMLや車載ソフトウェア分野でのアーキテクチャ記述言語であるEAST-ADLなども注目されている。今後のシステムズエンジニアリングは、連続モデルと離散モデルの統合、組込みシステムと情報システムの統合、統合された文脈での安全性・セキュリティといったようにつながる世界の広がりに対応していく必要があり、その分野の研究成果としてより良い手法が考案されることを期待している。

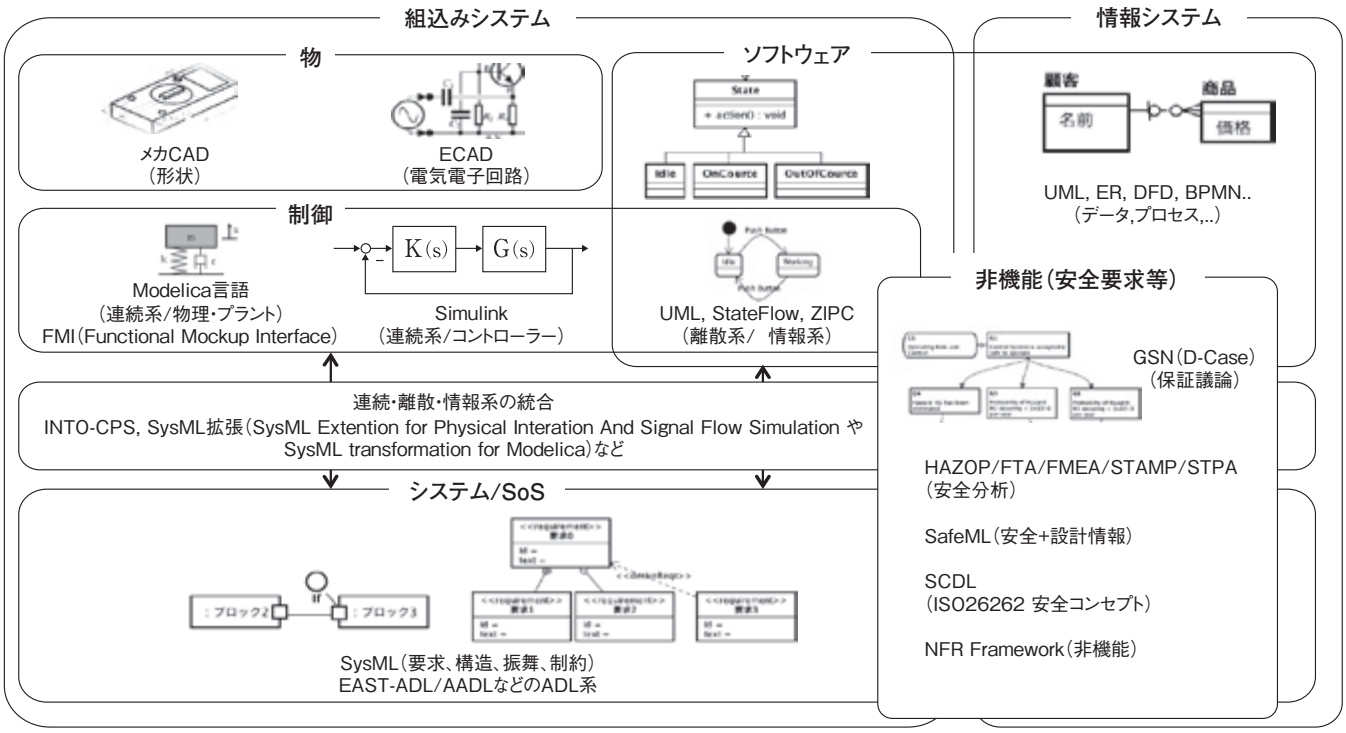

図２　組込みソフトウェア開発に関連するモデリング言語

3　組込みソフトウェア開発環境の現状

本節では、２節で紹介したモデリング言語の記述を支援する開発環境について紹介する。

ソフトウェア開発全般で利用される離散モデルをサポートする開発環境は組込みソフトウェア業界でも浸透しつつあり、astah（Change Vision）、Enterprise Architect（Sparx Systems）などが利用されている。また、状態遷移設計に関してはZIPC（CATS）も利用されている。

一方、組込みソフトウェアの特徴の一つである連続モデルのモデリングにはSimulinkが幅広く利用されており、制御対象の挙動を記述したプラントモデルのモデリングにはModelicaも利用されている。そのほかにも様々な製品が利用されているが、ツールに依存しないモデル交換、統合したシミュレーションを可能とする環境としてFMI（Functional Mockup Interface）が認知されている。近年では、例えばINTO-CPSプロジェクトにおいて、SysMLの拡張言語で離散・連続モデルで構成されるコンポーネントを設計し、その設計情報をFMIの仕組みを利用して現場が利用する様々な製品に連携させるような取り組みも行われている。また、OMGではSysMLに対する拡張として、SysML Extention for Physical Interaction and Signal Flow Simulation等が提案されている。なお、今後の組込みソフトウェアは、情報システムに搭載されるソフトウェアとの連携もますます増加すると予想される。組込みシステムと情報システムで扱う情報の質は、信頼性、情報の鮮度、セキュリティなどの面で大きく異なるため、双方のシステムが要求する情報の質が相反する可能性も高い。両者を統合したモデリング結果に基づき、適切な解を提示できる開発環境の実現が期待される。

組込みソフトウェアのもう一つの特徴となるシステムの安全性をどのようにして設計に織り込んだか、どのように安全性を担保しているかを表現できる環境については、GSNを記述できるastah GSN（Change Vision）、SCDLに対応したSafilia（GAIO）などのモデリングツールが提供されている（図３、図４参照）。しかしながら、HAZOP、FTAなどの分析手法については、ExcelやVisioなどの汎用ツールを使って記述している場合も多いと予想される。

図３　GSNの例
品質の妥当性を説明するGSN。C（前提）に基づいてG（ゴール）をS（戦略）でサブゴールに分割し、末端にSn（解決）を付けて説明する。

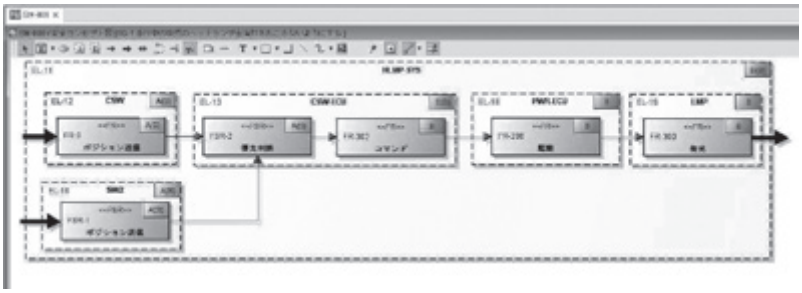

図４　SCDL（Safety Concept Descriptiono Language）の例
安全コンセプトを記述するSCDL。エレメント（作りの単位：矩形点線）の中に要求（機能の単位：矩形実線）を配置していく。各要素にはASIL（作りの手厚さレベル：右肩）が付与され、安全機能や機能間干渉の存在なども明示し、安全コンセプトのアーキテクチャを記述する。

また、完全性やトレーサビリティを重視する組込みソフトウェア開発においては、モデルからソースコードやテストケースなどを自動生成する開発環境に対するニーズは高い。連続モデルからソースコードを自動生成する開発環境としてはTargetLink（dSPACE）、状態遷移図などの離散モデルからソースコードを自動生成する開発環境としては、ZIPC、BrickRobo（富士通コンピュータテクノロジーズ）などが提供されている。テストケースの自動生成に関しては、ソースコードに基づいたテスト設計の支援ツールとしてカバレッジマスター winAMS（GAIO）などが存在しているが、UMLのような表現レベルの高いモデル表現からテストケースを自動生成する開発環境については今後の発展が期待される領域と予想している。

そのほかの自動生成の事例として、IPA（独立行政法人 情報処理推進機構）のRISE（Research Initiative on Advanced Software Engineering）が公開している保証ケースを自動生成する取り組みを紹介する（http://www.ipa.go.jp/files/000052723.pdf）。本取り組みでは、保証対象のシステム構成、システムが満足すべき安全要件、及びシステムに発生し得るリスクに基づいてGSN形式の保証ケースを自動生成する方法を提案している。これらの情報は、前述した開発環境でモデリングすることが可能であり、本提案と組み合わせることで保証ケースの作成時間を大幅に短縮することが期待できる。

最後に、自動車業界が取り組んでいる開発環境の事例について紹介する。車載ソフトウェアも一般の組込みソフトウェアと同様に、大別すると製品固有のアプリケーション部分と製品分野共通のプラットフォーム部分に分けることができる。自動車業界ではAUTOSAR（AUTomotive Open System ARchitecture）が規格化を進めており、プラットフォーム部分については、ECU（Electronic Control Unit）共通で必要となる機能がAUTOSAR BSW（Basic Software）として定められ、その振る舞いを切り替えるパラメータ群についても規格化されている。また、AUTOSAR BSWを提供するベンダーは、パラメータを設定できる開発環境を提供しており、それを利用することでECU共通の機能を実現することが可能である。また、アプリケーション部分についても他アプリケーションやプラットフォームとの接続方法が規格化されており、その規格に準拠した開発環境を利用することで、プラットフォームにそのまま搭載可能なアプリケーションを開発することが可能となっている。図５にAUTOSARを用いた開発の流れの概観を示す。

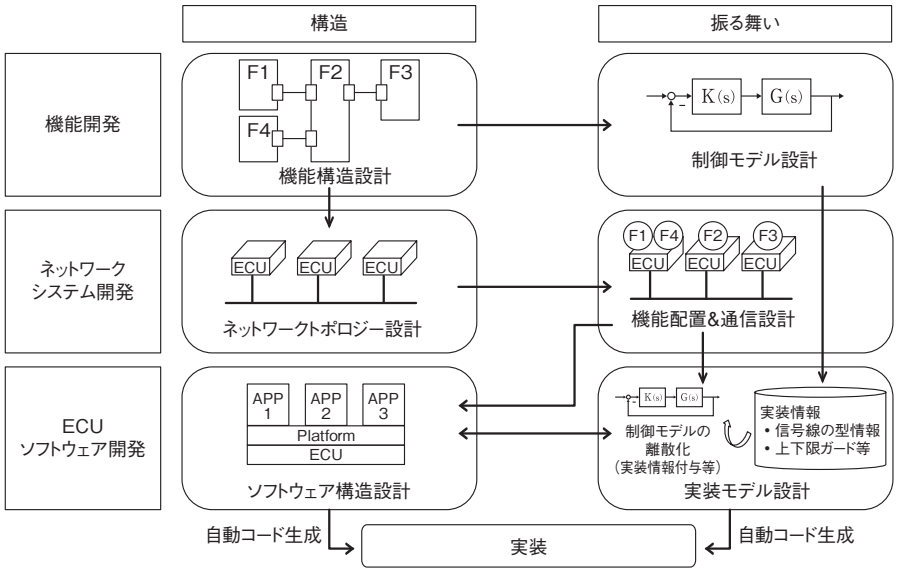

図5　AUTOSARを用いた車載ソフトウェア開発の流れの概観

Vector社が提供する開発環境を事例に挙げると、PREEvisionが制御モデル設計を除く機能開発工程とネットワークシステム開発工程を支援し、DaVinciがソフトウェア構造設計を支援している。制御モデル設計についてはSimulink（Mathworks）、実装モデル設計についてはEmbedded Coder（Mathworks）、TargetLink（dSPACE）などが利用されている。そのほか、Mentor社、dSPACE社、Elektrobit社等も自社の得意分野に合わせて開発環境を提供している。これらのツールはすべてAUTOSARが策定したツールチェーン規格に準拠しているため、車両メーカのニーズに合わせて適切な開発環境を組み合わせて利用することが可能である。これらのツール群は車載ソフトウェアを開発する上で非常に強力であるが、入力しなければならない設計情報が膨大、かつ複雑であり、AUTOSARに関する知見のないエンジニアが利用するにはハードルが高い側面もある。つながる世界において車載ソフトウェアの品質を確保するためには、車載ソフトウェアの構造と振る舞いをつながる先の製品開発に携わるエンジニアに理解してもらうことが重要になると予想される。他業界のエンジニアと設計成果を共有できる汎用的なUMLツールと車載ソフトウェア開発に特化したツールがそれぞれの得意分野で力を発揮し、お互いのモデリング結果をシームレスに連携できる環境の実現が期待される。

4　最後に

本稿では、組込みソフトウェア開発に従事するエンジニアが抱えている課題とその解決を支えるモデリング言語、及び開発環境について概観した。つながる世界を想定することが当たり前となった現代のエンジニアは、一つの機器を熟知すれば良かった時代のエンジニアと比べて、より深く、かつより広い知識に基づいて開発を進める必要に迫られていることを強調したい。こうした現場のエンジニアが抱える窮状を救い、一人一人のエンジニアが自分の製品領域の開発に集中できるようにするためには、様々な分野の熟練したエンジニアの知識が集約され、その知識群が容易に活用できる形で流通していることが不可欠である。つながる世界の組込みソフトウェア開発に携わるエンジニアの一人として、熟練者の知識が容易に活用できる世界が開発環境によって実現されることを期待している。