# 63

# ドメインスペシフィックモデリング（DSM）言語によるシステムモデリング事例[1]

**Domain-Specific Modeling Enables Full Code Generation**

## 1. はじめに

### 本末転倒？　目的と手段が入れ替わる。

始めてのモデル駆動開発に、UML や Simulink を採用することは多いと思います。ただ本来の目的である開発効率やプロセスの改善が、いつの間にかツールを使うことが目的になっていないでしょうか？　手段であるツールを使いこなすのは言うまでもありませんが、それが高じて陥りがちなワナにはまっていませんか？

（1）汎用に過ぎる　“あらゆる用途に使えないのなら有用ではない”と考える

例えば下図のような Polar 社のスポーツウォッチ製品専用のモデリング言語なら、製品のコンセプトを用いてモデリングが容易になることに加えて、機能の制約やルールを用いて、フォーマルかつ高品質にモデル化できます。そして資産を上手く再利用するジェネレータを作って、モデルから完全なコードやドキュメントを自動生成することができます。

これは開発チームの熟練者秘伝のレシピを取り入れたドメインスペシフィックモデリング（DSM）言語で、開発の抽象度を上げて目覚ましい効率改善を得るということです。Simulink もある種のドメイン固有言語ですが、制御系全般が対象で広範過ぎます。UML では汎用に過ぎます。あらゆる課題を解決しようとするよりも、実用面で効果を挙げることを目指すことで大きな成果が得られます。

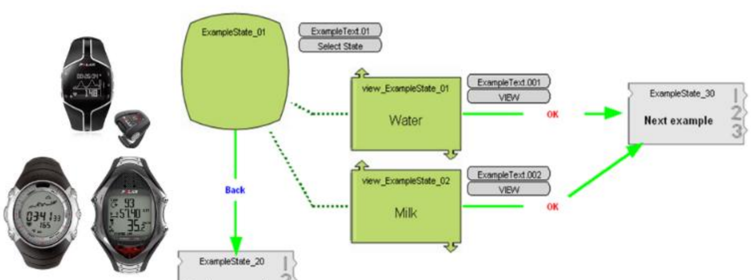


図 63-1　製品のコンセプトとルールでモデル化する

[1] 事例提供: MetaCase 社 Juha-Pekka Tolvanen
翻訳: 富士設備工業株式会社 電子機器事業部 小山 美樹 氏、浅野 義雄 氏

**(2) 手段に固執　“ツールありき”**

ドメインスペシフィックモデリング言語とジェネレータの開発や、そのドメインの持続的な成長に伴うメンテナンスには、相当な工数と費用が掛かることや、スケーラビリティやコラボレーションに課題があるとの報告が散見されます。これは、モデリング言語開発に従来からのプログラミングを用いるファイルベースのツールを採用した場合の話であり、モデル言語開発専用ツールでは、そのような課題はありません。製品のライフサイクル及び開発の進化と共に成長できるドメインスペシフィックモデリング言語が、短期間で現実的な費用で開発されています。

専用ツールなら 1～2 週間、 それ以外（UML ベースなど）は 10～50 倍の期間を要する。既存の手法やツールに固執せず、専用ツールを利用することにより優れた開発環境を手に入れることができる。

## 2. DSM 概要

ドメインスペシフィックモデリングは、ドメイン固有のコンセプトを用いてソリューションを定義することで、プログラミングのそれと比較して抽象レベルを引き上げる。そして最終製品（コード等）を、高い抽象レベルで定義されたモデル（＝ 仕様）から生成できるので、開発の生産性や、市場投入までの期間、品質等を飛躍的に改善する。この自動化の仕組みは、単一企業の単一ドメインの要件だけを満たすモデリング言語とジェネレータによって得ることができる。本稿では、ドメインスペシフィックモデリングについて、産業界で実践された事例を合わせて紹介する。加えて、ドメインスペシフィックモデリング言語とコードジェネレータの開発手順も解説する。

ドメインスペシフィックモデリング（Domain-Specific Modeling =DSM）は、アプリケーションドメインのコンセプトを用いてソリューションを定義することで開発の抽象度を上げる、モデリングの取組みである。これはドメインスペシフィックモデリング言語（ドメインの抽象レベルに特化したセマンチックに従ったモデリング言語）の使用によって実現され、ソリューションをコードで実装するための技術的課題に比較して、開発者はソリューションそのものに集中することができる[2],[8]。DSM の一事例を次に紹介する。図 63-2 は、DSM を用いた自動化システムの仕様であり、暖房装置の構造と振舞いの両方をモデル化している。

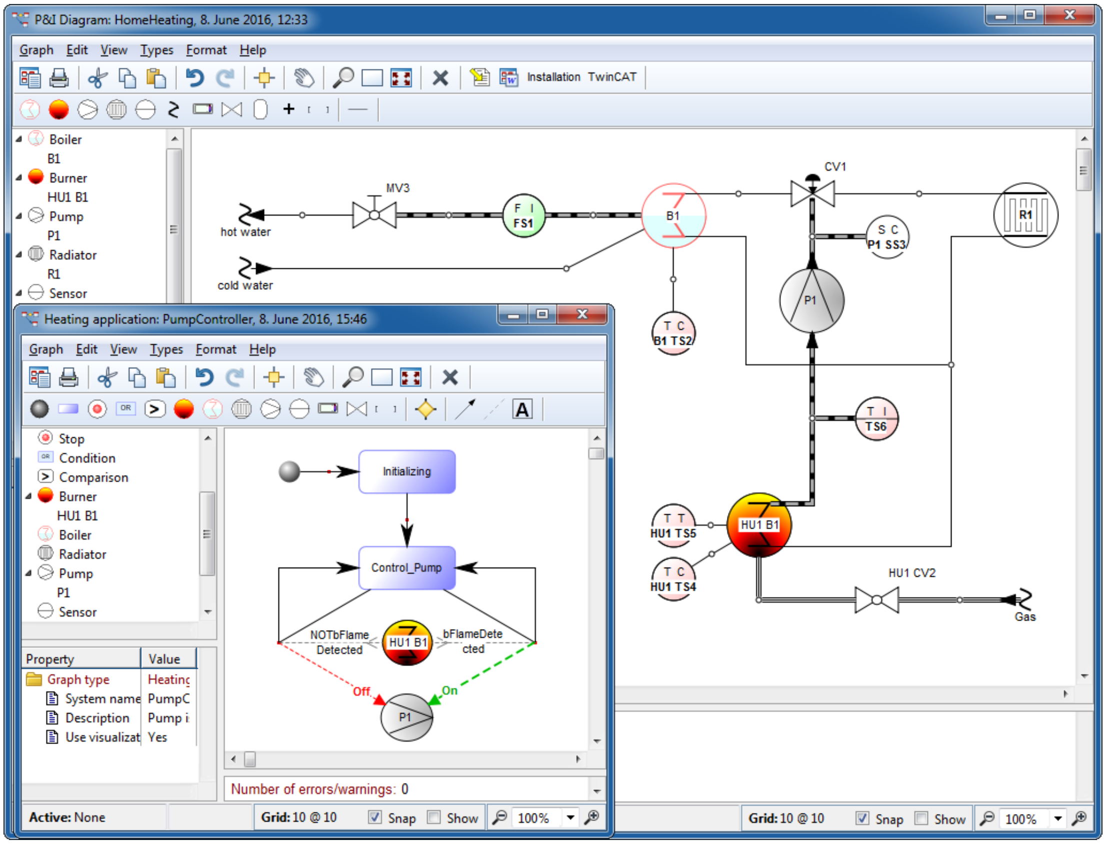


図 63-2　DSM of heating system: Model of the system structure (right), model of the application behavior (left)

モデルエディタのツールバー内にドメインスペシフィック言語のコンセプトが見える。背面のエディタ上にはパイプで接続されるボイラーやガスバーナー、ポンプ、センサー等の様々な機器がある。このように暖房システムが、配管や機器構成など既知のドメインコンセプトで仕様化される。前面にある、もう一つのモデル言語エディタにあるのは、同じシステム内のポンプの振舞いの視点である。この言語ではステートやトランジション、様々な機器ごとのアクション（ポンプの起動設定等）、といったコンセプトも使用する。モデリング言語にはドメインのルールが組み込まれているので、間違った設計を事前に排除することや（例えば熱交換器にポンプを直接接続してしまう）、不完全な状態を指摘することもできる（ポンプをパイプに配管し忘れている）。

これらのモデリング言語は、ソースコードよりも高度なレベルで扱われ、開発者は生成されるコードを知る必要が無いにも関わらず、モデルから暖房システムの完全なコードを自動生成することができる。このようなコード生成が可能となるのは、ドメインの専門性のおかげである。それはモデリング言語とコードジェネレータの両方を単一の限られた狭いドメインの要件にのみ適合させることである。それは一企業のドメイン、あるいは単一のソリューションプロバイダーのドメイン等であるが、複数の企業間で共有されるような業界標準を広範にカバーするドメインもある。またジ

ェネレータはコード以外の成果物も生成できる。ハードウエアの割当てやデバイスの構成、仕様書やテストケース、インストールガイドなど、あらゆる成果物を同一のモデルから生成できる。

## 3. 産業界の実績

産業界の様々な企業の実績から、ドメインスペシフィックモデリングによって色々の成果が得られることが報告されている。生産性が向上すること、利害関係者間のコミュニケーションが改善されること、新しい開発者でも即戦力に出来ること、設計の早期段階でエラーを排除できるので品質を改善すること[8]、などである。中でも生産性の向上は最も多く主張されている。図 63-3 に、5 つの公開事例を要約する。これは人手によるコード実装から DSM に移行したことで、生産性がどの程度向上したのかを図示している。

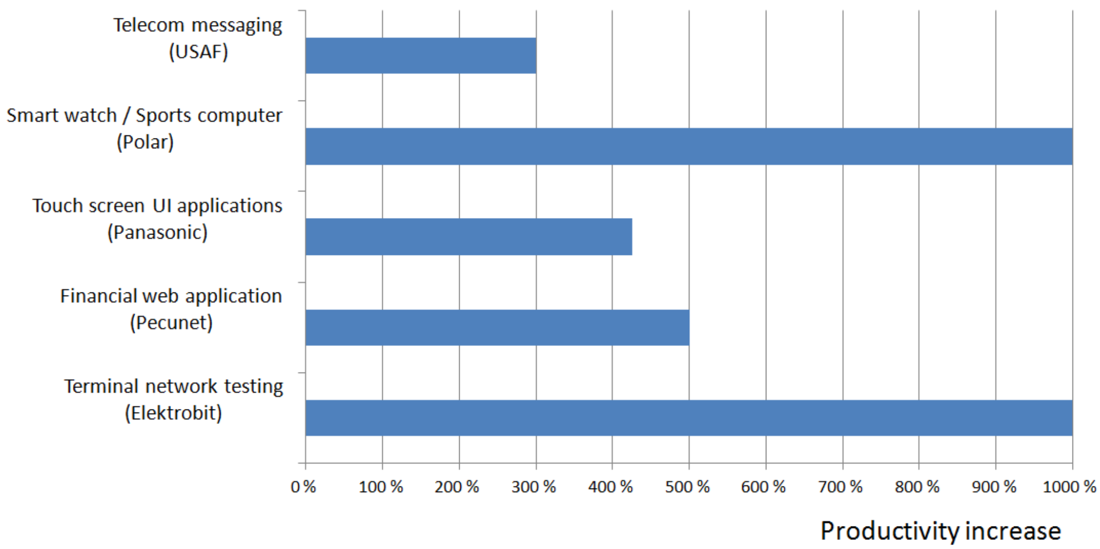


**図 63-3　Reported productivity increase – compared to earlier manual practice**

**人手に頼ったコード実装に比較して生産性が改善された**

米国空軍（USAF）の防衛情報通信システム（message translation and validation system）の事例（Kieburtz et al. [3]）では、ドメインスペシフィック言語とコードジェネレータを、初期の開発や保守工程などを含む 130 以上の開発タスクへ実験的に採用した。そして手作業によるコード実装に比較して、自動生成されるコードの生産性は 3 倍で、エラーは半分になった。また各タスクや担当者間の成果に殆ど違いが無かったこと（99%以上同じ）も特筆される。

Polar 社の事例[4]では、スポーツコンピュータのアプリケーション開発用の DSM 言語とコードジェネレータを作った。代表的な機能の開発を通じて DSM の評価を行ったことに加えて、大規模な製品機能全体の開発を通じて、10 倍以上の生産性向上を得ることができた。

パナソニックの事例[7]では、タッチスクリーンのアプリケーションの人手による実装と、DSM

言語とジェネレータでコード生成することの両方を比較して、4倍以上速くなることを確認した。Pecunet 社では金融関連の製品と契約内容を構成する DSM 言語を作成し、エンジニアでは無い金融システムの担当者によるモデルから、Web アプリケーション用の Java コードを自動生成することで、5倍の生産性向上を得ることができた[2]。

Elektrobit 社では、VoIP 端末と接続網のシステムテストに DSM 言語を採用した。テストのカバレッジが改善され、人手による実装に頼った従来のテストに比較して、同じ期間で 10 倍のテストが実施できるようになった。

# 4. 事例紹介（３種）

DSM 言語を適応した実例として、スマートウォッチの開発、無線通信システムのテスト、車載システムのアーキテクチャ設計と安全分析といった、３つの異なる領域の例を紹介する。最初の事例でコード生成について、２つ目の事例でテスト生成について、そして３つ目の事例では安全分析に主眼を置いて説明する。全ての例でモデルは高い抽象度レベルのドメインのコンセプトで表現されて、それらからジェネレータがソースコードや、テストケース、安全分析データを生成する。

## 4.1. コンシューマ機器と UI アプリケーションの開発

スポーツ関連デバイスの一流ブランドである Polar 社は、ランニング、フィットネス、クロストレーニング向け心拍計や、GPS 対応サイクリングコンピュータ、耐久トレーニング向けスポーツウォッチを提供している。心拍や消費カロリー、距離、高度、ペダルレートやサイクリングパワーなどの測定と、そのデータの解析や視覚化、記録や練習日誌化などの機能を有する。
Polar 社は、仕様書からデータをコード内にコピーするといった手作業を排除することや、コードを人手で実装するのではなく自動生成させることで、アプリケーション開発の生産性を改善することを求めていた。そして DSM 言語が、この目的に沿うと判断された。また新しい開発者を容易に即戦力にすることや、コード品質の改善も求めていた。

これらの目的を満たすために Polar 社は、スポーツコンピュータの UI アプリケーションの DSM 言語を作った[4]。このドメインが選択された理由は、最も大きなソフトウェアであり、開発工数の 40~50%を費やしていることであった。UI アプリケーション開発の効率を改善することは、全開発期間に対して最大限の効果を与えることになる。図 63-4 に、最もシンプルなモデルで、この DSM によるモデリングを例示する。メニューから次のメニューや表示への遷移の選択肢を仕様化している。実際には数十のエレメントがモデル図内に存在して、ひとつのアプリケーションは数十のモデル図があり、何十ものアプリケーションで製品が構成される。単一のモデル図にあるひとつのエレメントは、他のモデル図にリンク、参照され、また再利用される。あるいはサブグラフにリンクされて更なる詳細を定義することもできる。そしてモデル図に対してジェネレータを実行して、完全なコードが自動生成される。

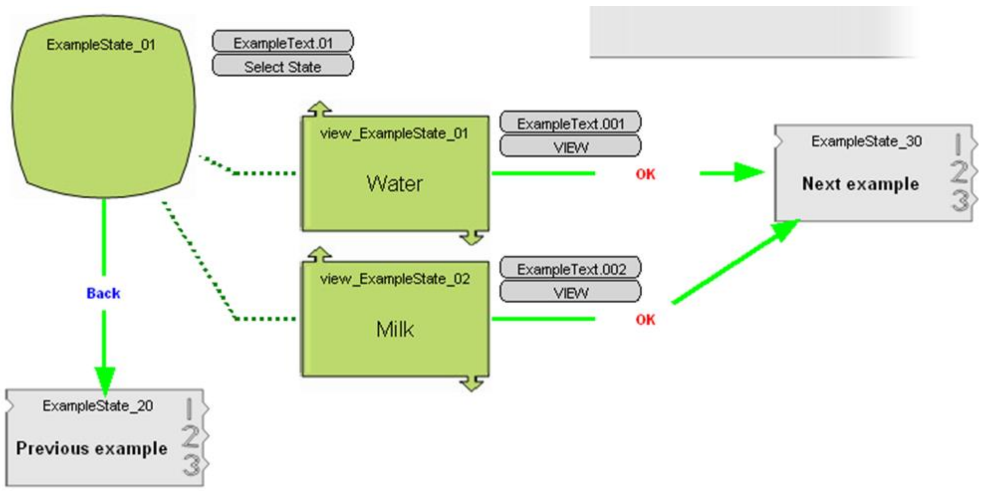


**図 63-4　Sample of UI specification using domain concepts like panel, navigation path, menu, display**

Polar 社の DSM 言語は実用性で評価された。研究・調査段階では、6 人のエンジニアがスポーツコンピュータの代表的な機能を個々に開発して、平均 900%の生産性向上を記録した。そしてスポーツコンピュータの大きな部分での評価結果は、1000%以上の開発効率改善を得ることができた。生産性の著しい向上である。また Polar 社は、生成されたコードの品質や、この DSM 言語を用いたアプローチが容易であるかについて、使用した開発者に意見を求めた。その結果、ジェネレータにより生成されるコードはマニュアル実装されたコードより良いこと、またこのモデルベースの取組みでは、新しい開発者が容易に即戦力になることを確認した。DSM 言語とコードジェネレータで、Polar 社は期待通りの成果を得ることができた。

## 4.2. 通信システムのテスト開発

車載機器や通信システムを提供する Elektrobit 社は、軍需用の VOIP 通信ネットワークと無線端末の開発とテストを目的に DSM 言語を採用した[5]。Elektrobit 社が DSM を採用した理由は、テストスクリプトの実装や、その実行を人手に頼ったままでは、多くの時間を費やしてしまうためである。特に多くの端末がネットワークに接続されて非常にタイトな時間制約内で双方に呼び出し合うこと、しかもそのシーケンスを変化させるといった複雑さを扱うことは容易では無い。人手によるテスト実行も可能ではあったが、呼び出し時のタイトな時間制限や境界値の試験などへの課題を克服するために、通信プロトコルのテストに広く使われている TTCN-3[2]をベースにしたテスト自動実行環境を採用していた。ただ自動実行ができても、多くの端末間の呼び出しシーケンスをタイ

[2] Testing and Test Control Notation Version 3：通信プロトコルや Web サービスのテストに特化したプログラミング言語

トな時間制約内で実施するテストの開発や、それらテストスイート[3]のメンテナンスは容易では無い。TTCN-3 に対する特別な知識と経験が必要となるが、そのためのリソースや予算を得ることも容易では無い。

このような課題を解決するドメインスペシフィックモデリングでは、テスト担当者は TTCN-3 の知識を必要とすることなく、個々のテストケースとテストロジックを指示するモデルを記述する。そしてモデルからジェネレータが、用意されたテストのフレームワークに TTCN-3 形式のテストスクリプトを自動生成する。

図 63-5 のモデルはテストの小さな例である。左図では通信端末である Device1 が Device3 を呼び出すことでテストケースが開始される。この例では、もし Device3 が一定時間内に応答しなければ、呼び出しを無効にして End Calls に遷移させるオプション（図中の時計の 2s）も備える。呼び出しシーケンスに加えて、右図では、このテストケースに対するネットワーク上の構成をモデル記述している。ネットワーク上の端末とその特性は、テストロジックと分離することができるので、別のテストを目的にテストの設定を変更することが容易である。またテスト実行中、モデルはアニメーション機能で結果をハイライトできるので、どのようにテストが実行されているかの確認も行い易い。

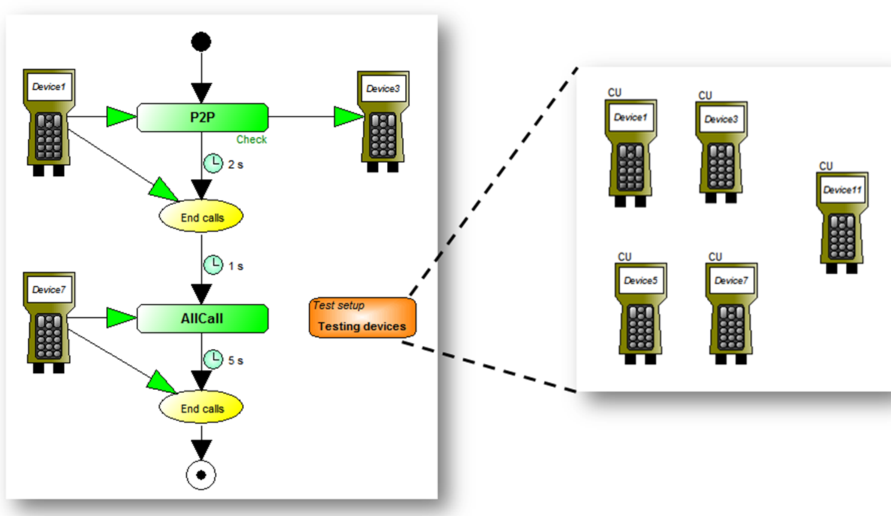


**図 63-5　Sample model for network testing case**

この２つの DSM 言語でなら、テスト担当者はテストケースとテストロジックの両方を馴染みのあるドメインコンセプトで指定できる。TTCN-3 形式のテストケースの実装は、人手に頼った手法に比較して、DSM 言語では 1000％も早くなることを確認した。また Elektrobit 社は、テストケースの開発速度を加速できたこと以外の効果も確認した。開発担当者や、その他の利害関係者までが、

[3] ここでは TTCN で書かれた多数のテストケースをまとめたものを示す

DSM 言語によるテストケースの視覚化と自動生成の取組みは、わかり易いと評価した。グラフィカルなテストケースは、何をするのか理解しやすいためである。またテスト実行中、その進捗が視覚化される。これでテストケースの分析に要する時間も節約できる。

## 4.3. 車載システムモデリングと安全性解析

自動車業界では OEM とサプライヤーの両方が、ソフトウェアを対象にした AUTOSAR や、電気・電子システムの機能アーキテクチャ設計を目的にした EAST-ADL（SysML のコンセプトを参考に車載システム向けに開発された）など、様々なドメインスペシフィック言語を開発してきた。この章では、Volvo、Fiat、Daimler、Continental、Bosch により共同開発された EAST-ADL[4] モデリング言語について紹介する。図 63-6 は EAST-ADL で記述された 2 種類のアーキテクチャモデルである。左側はファンクショナルアーキテクチャで、右側はハードウエアアーキテクチャを定義している。また双方の割り当てをマトリクス形式で定義できる、専用のモデリング言語（下側）も用意されている。この例でも、高い抽象レベルでモデリングされていて、ECU、センサー、ノード、基本ソフトウェアコンポーネントなど、自動車の部品をコンセプトに用いる。そして、これらのモデルから Simulink モデルや AUTOSAR の XML、要件トレーサビリティの情報等が自動生成できる。

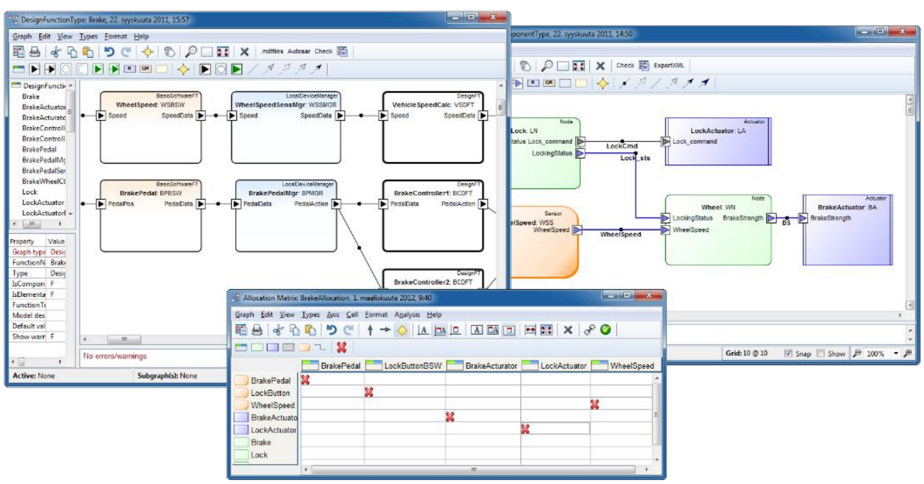


図 63-6　**Functional architecture (left), hardware architecture (right), and allocation modeling (bottom)**

[4] http://www.east-adl.info

アーキテクチャモデルは様々な解析にも役立てられるが、とりわけ安全性解析に活用される。安全性解析には、多くの労力と費用を費やすので重要である。車載システムの安全解析を支援するには、ISO 26262 のような機能安全規格がアーキテクチャモデルを介してサポートされる。そしてドメインスペシフィック言語には、機能安全規格と同じコンセプトを直接採用する。「Item」「Hazard」「HazardEvent」「SafetyGoal」などである。これによって機能安全規格に則して、安全要件を作りこむことができる。図 63-7 の左側に、パワーウインドゥの安全性モデルの例を示す。この例で、ハザードとして障害の検出（Obstacle Detection）が記述されている。このようなモデルから、ジェネレータは FTA（Fault Tree Analysis）や、FMEA（Failure Mode and Effects Analysis）などの解析の基になるデータを自動生成する。これらに対応する専用ツール の機能を活用することで、モデル変換の仕組みでエラーモデルを構文解析して、それに相当する解析用モデルを生成できる。図 63-7 の右側では例として、FMEA のツールである HiPHoPS へのモデル変換結果[9]を示す。モデルからの自動生成を通じて解析が実施されることで、解析に必要な多くの工数が削減されて、またその品質が大きく改善された。このモデルベースの取組みによって、50%以上の削減効果があったことが報告されている[6]。このモデル変換の取組みなら、安全設計技術者による安全設計が、実際に計画されているアーキテクチャ上で全ての側面を考察できる。

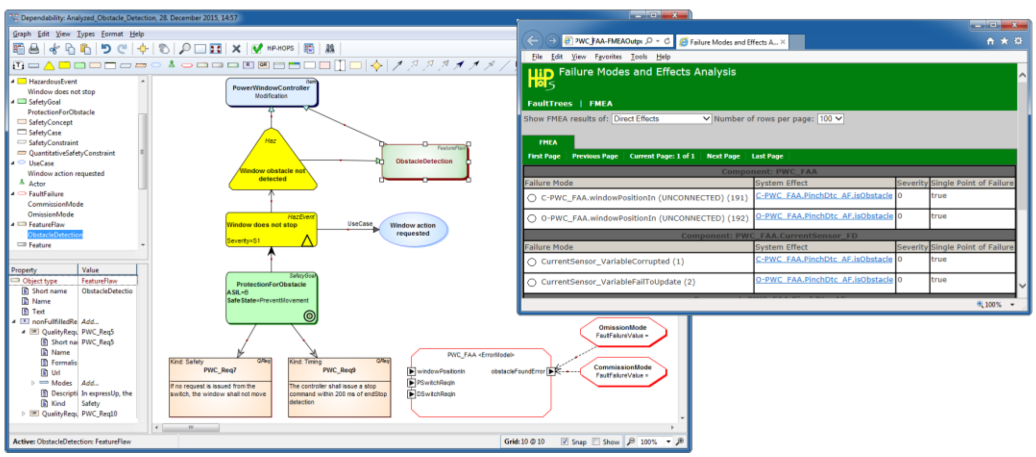

図 63-7　Safety model according to ISO 26262 and produced FMEA results

## 5. DSM の開発手順

上述した生産性と品質の飛躍的な改善には、ドメインスペシフィックモデリング言語とジェネレータを作ることが先決であるが、その開発は 1～2 名の担当者のみで取組むことが賢明である。それ以外の開発者は、モデリング言語を使用することに徹することである。またモデリング言語とジェネレータをツール化するための工数やリソースも検討する必要があるが、上述した企業が採用した専用ツールなら、ドメインスペシフィックモデリング言語に応じたツールが自動で生成される。それゆえツール化のための工数も、モデリング言語変更時のツールメンテナンスも必要が無い。こ

のことは持続的に変更され進化する製品及びプロダクトラインの開発に欠かすことのできない、非常に重要な要件であるが、実はあまり知られていない。

以下に、ドメインスペシフィックモデリング言語とジェネレータ開発の、主要な４つのステップについて説明する。には、図 63-8 この概観を示している。

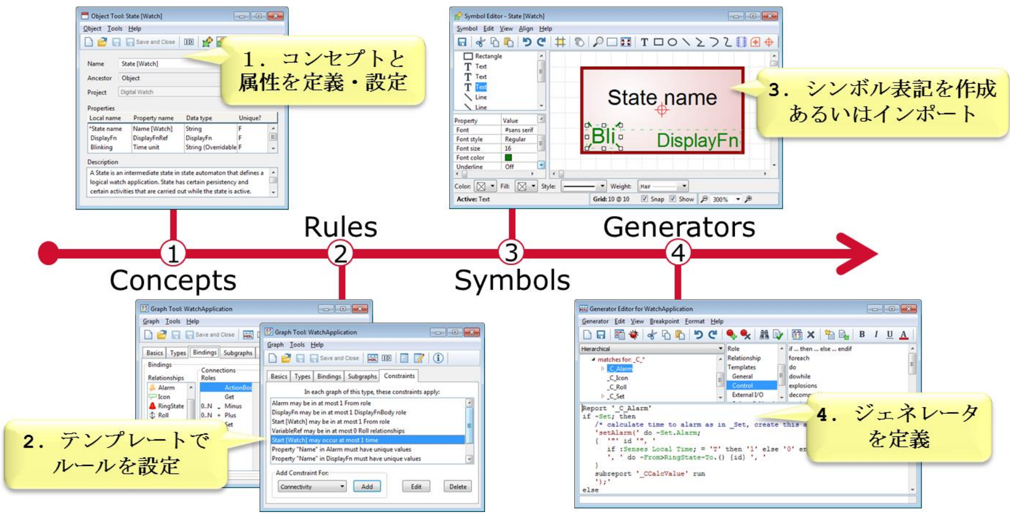


図 63-8　Steps for defining domain-specific modeling languages and generators

## Step 1: まずはコンセプトを定める

ドメインスペシフィックモデリング言語の開発は、コンセプトを定めることから始まる。コンセプトは対象となるドメインに応じて、製品の部品、製品アーキテクチャ、製品系列の特性（プロダクトラインのバリエーション）、モデルから自動生成される成果物、などから得られる。抽象レベルを可能な限り高くすることで、生産性と品質の改善を最大限に得ることができる。言語の適正な抽象レベルは、問題空間（problem domain）と同じであるか、あるいはできるだけそれに近いレベルである。そうすることで、モデリング担当者は既に馴染みのあるドメインのコンセプトを用いてモデル化できるし、またそれは利害関係者が読んで理解しやすいので、双方の意思疎通を大いに図ることになる。

モデリング言語の定義は、メタモデルに形式化される。ここでプロパティや接続等の、言語の様々なコンセプトも定義される。一般にドメインの主要コンセプトは、モデリング言語のオブジェクトとして定義され、それ以外のコンセプトはオブジェクトのプロパティや、コネクション、サブモデル、他のモデル言語上のモデルへのリンク等になる。図 63-9 で、ある言語コンセプトの定義例を示す。この「State」 エレメントは、「State name」「DisplayFn」「Blinking」「Documentation」 の４つのプロパティを持つ。

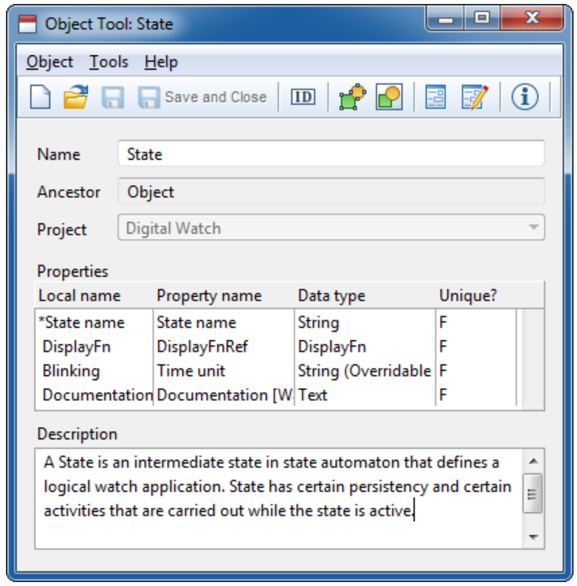


図 63-9　Definition of metamodel element ‘State’.

## Step 2: ドメインのルールや制約をモデル言語に持たせる

モデル言語にドメインのコンセプトを用いることに加えて、モデルがドメインのルールに沿うようにすることが望まれる。これにより設計の早期段階でエラーを未然に防ぐことや、モデリングよる仕様化の労力を軽減することができる。また自動生成やモデル変換のためのジェネレータの定義にも活用される。コンセプトを定義した次の段階として、このようなルールがメタモデルに追加される。図 63-10 では、state names がユニークであることを制約として定義している。適切なルールをモデリング言語に定義することで、モデリング担当者は設計ガイドラインを頻繁に参照することから解放される。

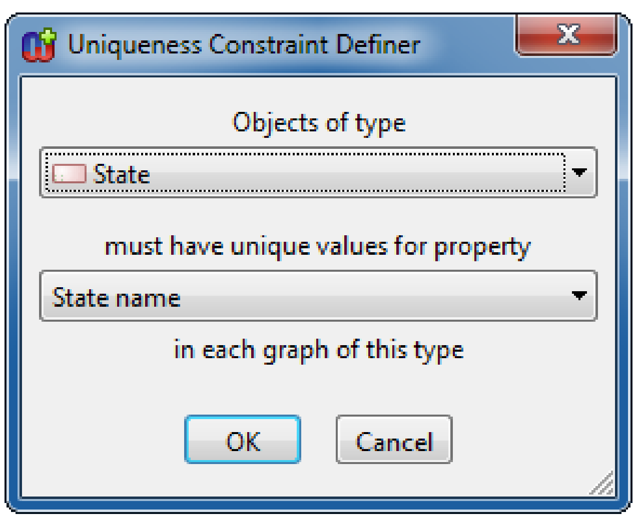


図 63-10　Choosing rules

## Step 3: モデルの表現や見た目を設定する

モデリングには視覚的な表現が必要となる。一般にはダイアグラムであるが、マトリクスや表形式、あるいは単なるテキスト形式のこともある。このステップでは、視覚的に編集されるべきモデリング言語のエレメントの表記が定義される。この資料内の各図に、異なるグラフィカルなモデリング言語の表現が紹介されているが、それらに使用されるシンボルやアイコンは、それぞれ異なった言語のコンセプトを象徴している。良い表記はドメインの描写を真似ることや、開発者間で既に馴染みのある約束事から得ることができる。それによってモデリング作業が簡単になり、それを読んで理解することやメンテナンスすることも容易になる。

図 63-11 は、モデリング言語のコンセプトである'State'のシンボル定義の例である。ここでシンボルにはステート名（State name）に加えて、条件付きのアイテム（図中 Bil は表示の点滅、DisplayFn は状態表示）が赤枠四角の中に示されている。視覚的な表記によって、担当者のモデルによる仕様化は支援され、またモデルの不完全な箇所を示唆することもできる。

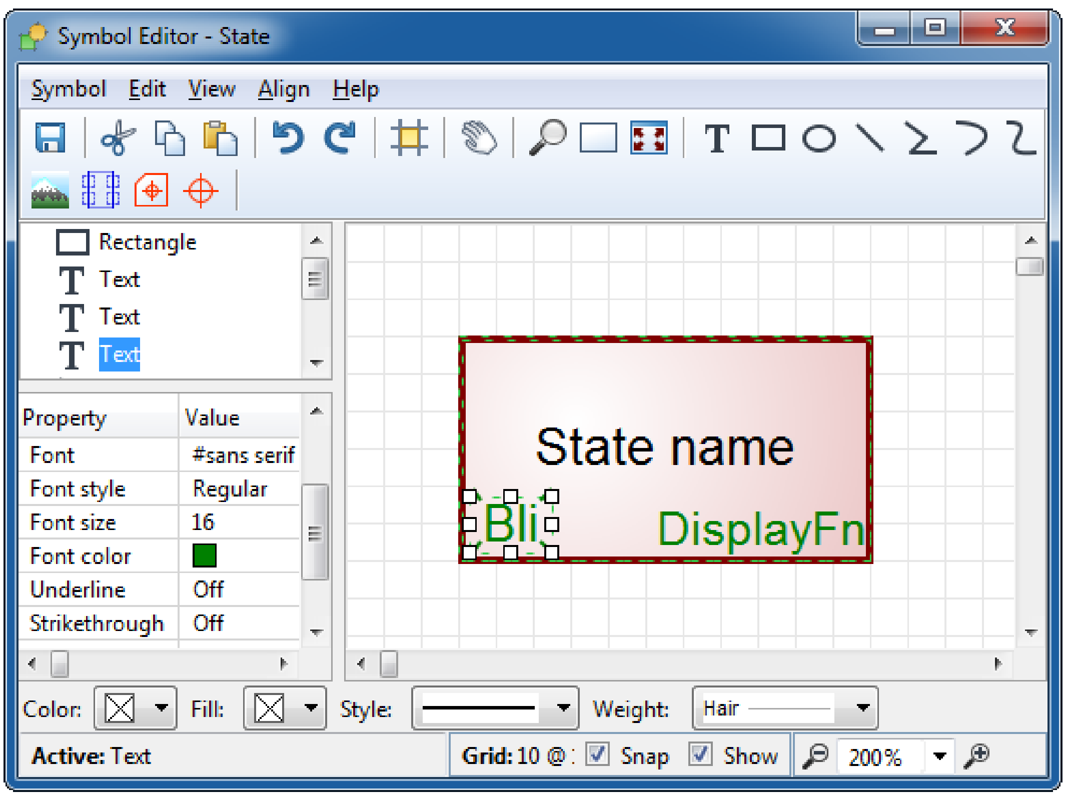


図 63-11　Drawing notational symbols

モデルの各エレメントに表記が追加されると、モデリング言語のユーザに提供してテストが開始される。MetaEdit+では、モデリング言語のコラボレーション開発をサポートしているので[10]、

モデリング言語が編集される中でも、モデリング担当者が使用してテストすることができる。これによってユーザであるモデリング担当者からのフィードバックに迅速かつ容易に対応することができる。

## Step 4: モデルから C コード等への変換機能を設定

ドメインスペシフィックモデリングの究極の目的は、モデルを変換してソースコードやドキュメントなどの様々な成果物を自動生成することである。モデリング言語を定義して何らかのモデルができれば、最後のステップとしてジェネレータを定義する。ジェネレータの開発は、モデル内のコンセプトをソースコード等の成果物にどのようにマップするかということにつきる。もっとも単純なケースでは、モデリングの各シンボルから、モデリング時に設定された引数を伴った、特定の決まったコードを生成させる。より一般的なケースでは、ジェネレータはシンボル間のリレーションシップや他のモデリング情報なども考慮に入れる。

図 63-12 はコード生成の為のジェネレータ定義例を示す。ジェネレータのハイライトされる部分から、どのように State あるいは Start がアクセスされるかを見ることができる。この例では、イベントトリガの起点となるボタンを基にステート間の状態遷移を頼りにして、C コードの case 文が生成される。

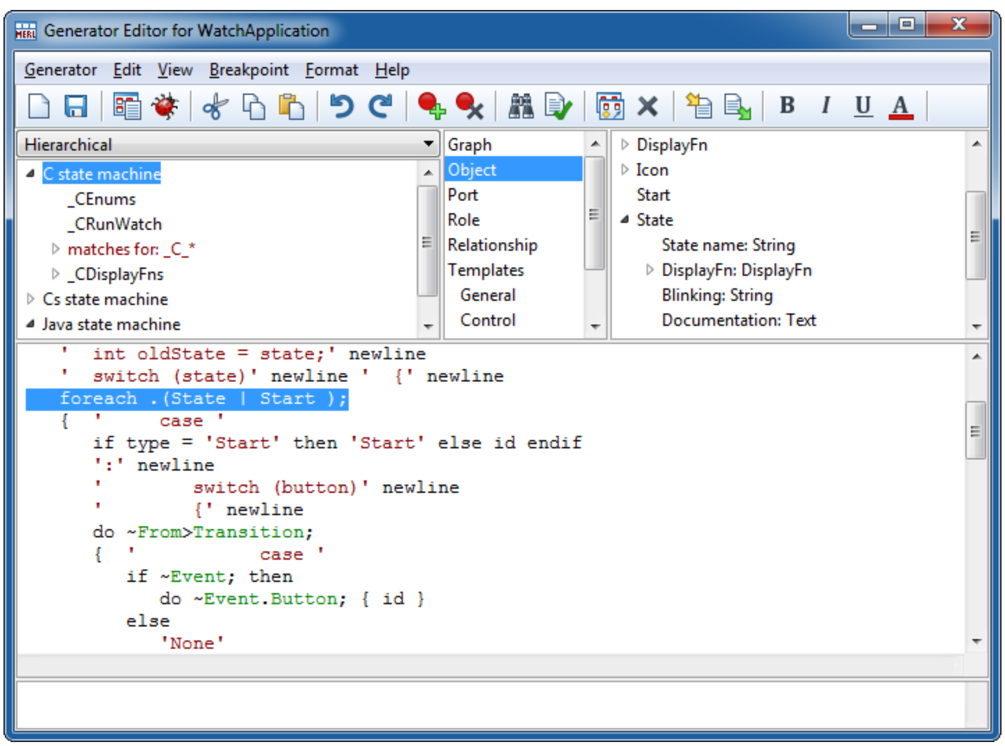


図 63-12　Defining a generator

ジェネレータの用途はソースコードの自動生成に限らない。コード生成時にビルド用のスクリプトを同時生成させることもできる。またテストケースや、各種ドキュメント、何らかの構成設定、インストール手順書、Simulink 等の各種モデリングツールとの相互変換等、様々な用途に活用されている。これにより生成される全ての出力は、相互に最新状態に一貫性を持って維持される。そして各成果物は、個別に編集することや、更新する必要が無いので、生産性の向上に大きく寄与する。

### 実践的な導入について

実践的に活用するには、インクリメント、かつイテレーティブに、製品のライフサイクル及び開発の進化と共に持続的に成長させる能力が DSM を開発するツールに求められる。実は DSL や DSM の考えは以前からあるものの、実践活用が上手くいかない理由が、ここにある。

DSM 言語開発の最善な取り組みは、インクリメンタルに行うことである[1],[2]。小さな限られた範囲でモデリング言語を作ってみて、それを用いて一部をモデル化して、モデリング言語に変更を加えて、更にモデル化してみて、といったイテレーティブな手順でリスクを最小限に抑えながら、早期段階からテストが可能で、組織上の変更に対する敷居も下げながら、従来のコーディングからモデルベース開発に移行することができる。そしてモデリング担当者がモデリング言語を早い段階から試してフィードバックすることも可能になる。モデリング言語開発の初期段階ではイテレーションは止まることは無い。そして一旦使用されるようになれば、ドメインの進化（製品への持続的な変更など）とのギャップを埋めるべく、そのモデリング言語も拡張されて進化することになる。モデリング言語開発の専用ツールなら、ドメインスペシフィックモデリング言語とジェネレータを開発するための工数と労力が大いに削減できるだけでなく、モデリング言語への変更に伴う既存モデルのアップデートが自動化されるので、その開発プロセスがアジャイルに継続できる。

## 6. まとめ

ドメインスペシフィックモデリングは、対象ドメイン固有の問題空間上のコンセプトやルールを直接用いてプログラミングから抽象度を上げることと、その高い抽象度のモデル（仕様）からコードやコンフィグレーションなど様々な成果物を自動生成することで、生産性を飛躍的に改善することが、多くの企業から報告されている。

DSM 言語の開発は、多くの組織にとって初めての経験であり敷居が高いと見る向きもある。でも実は、既に慣れ親しんだドメイン固有のコンセプトや表現を開発に用いている。それらは仕様書や議事録内のスケッチ、顧客との間で交わされる用語集などにあり、新しくこれらを作る必要は無い。実績があって、評価されてきた既知のコンセプトや用語から、直接システムを生成しようとする試みである。

そして先進のツールを採用することにより、ドメインスペシフィック言語とジェネレータの開発は数日から数週間で事足りるので、パイロットの取組みや、新しい課題への対処も容易に行える。

何よりも重要なことは、言語には継続的な変更や追加が欠かせないということである。それゆえ、進化を支援できるツールが要求される。上述した事例で活用された専用ツールには、言語の進化をサポートする仕組みが備わっており、モデリング言語の変更は容易であり、それら変更は既存のモデルを破壊することなく、自動的に反映させることができる。そしてドメインスペシフィックなモデリング言語とジェネレータを用意すれば、モデル化を開始することで、上述した生産性と品質の飛躍的な向上を得られるようになる。

参考文献

[1] Asano, Y., ジェネレータとモデリング言語のコツ うまくいっている組織のアプローチ, Embedded Technology, 2015 (http://www.fuji-setsu.co.jp/files/ET2015DSM.pdf)

[2] Kelly, S., Tolvanen, J.-P., Domain-Specific Modeling: Enabling Full Code Generation. Wiley, 2008.

[3] Kieburtz, R. et al., A Software Engineering Experiment in Software Component Generation, Proceedings of 18th International Conference on Software Engineering, Berlin, IEEE Computer Society Press, March, 1996.

[4] Kärnä, J., Tolvanen, J.-P, Kelly, S., Evaluating the use of domain-specific modeling in practice. Proceedings of the 9th OOPSLA workshop on Domain-Specific Modeling, 2009.

[5] Puolitaival, O.-P., Kanstrén, T., Rytky, V.-M, Saarela, A. Utilizing Domain-Specific Modelling for Software Testing, The 3rd International Conference on Advances in System Testing and Validation Lifecycle, 2011.

[6] Ruede, E., Walker, M., Papadopoulos, Y., Securius, P., Parker, D., Experience from Application of Semi-Automatic Fault Tree Synthesis to Reliability and Availability Analysis of Ship Systems, 22nd International offshore and polar engineering conference, 2012.

[7] Safa, L., The making of user-interface designer a proprietary DSM tool. In 7th OOPSLA workshop on domain-specific modeling (DSM), 2007.

[8] Sprinkle, J., Mernik, M., Tolvanen, J-P., Spinellis, D., What Kinds of Nails Need a Domain-Specific Hammer?, IEEE Software, July/Aug, 2009

[9] Tolvanen, J.-P., Luoma, J., Chen, D.-J., Reaping the benefits of architectural modeling in embedded design. Embedded, 2014, http://www.embedded.com/design/prototyping-and-development/4437065/Reaping-the-benefits-of-architectural-modeling-in-embedded-design

[10] Tolvanen, J.-P.,コラボレーション開発の技術革新, http://www.fuji-setsu.co.jp/files/Collaborative_modeling_and_metamodeling_JP.pdf