特集論文

ドメイン固有言語を利用した設計手法

佐藤美和*
岡　藍子*

Model－based Approach to the Design of Domain Specific Language

Miwa Satou, Aiko Oka

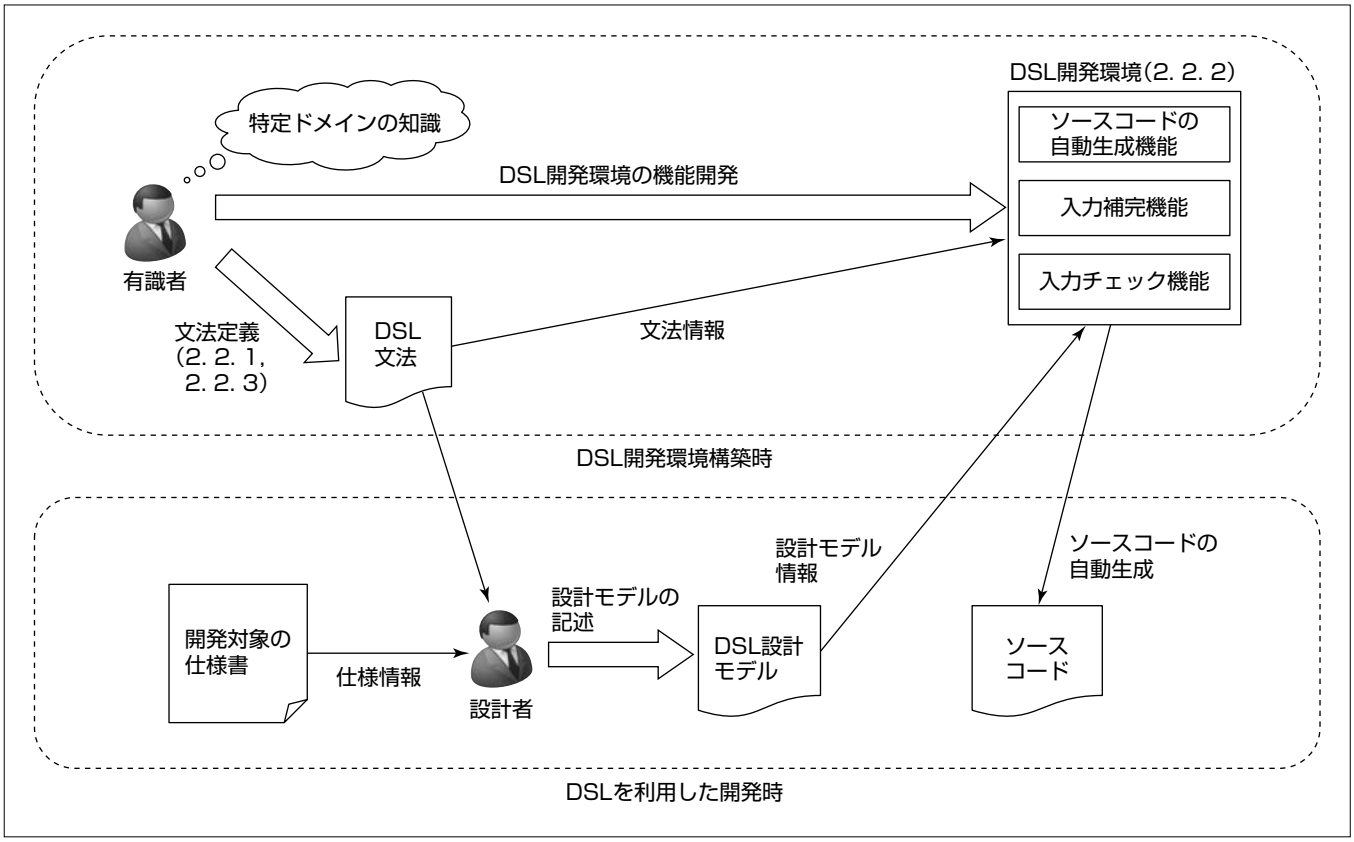
要　旨

大規模通信システムの開発では，システムを構成する装置が多く，各装置を並行して開発する。このため，各装置の組合せ試験は開発終盤にしか行えず，試験で不具合が検出されると，原因究明，ソフトウェア改修，再試験と手戻りが発生する。対策として各装置の通信応答を模擬した通信シミュレータによって，各装置間の試験を事前に実施しているが，装置数が多い場合，通信シミュレータ開発の負荷が課題となる。

今回，大規模通信システムでの装置間の通信応答を模擬した検証用通信シミュレータの通信パケット解釈処理部分の設計にドメイン固有言語(Domain Specific Language：DSL)を適用した。DSLを用いて，特定領域(ドメイン)のふるまいに特化した設計モデルを記述し，設計モデルからソースコードを自動生成する。

適用事例では，必要最低限の要素に当たるデータ名称，データ長，入力値範囲の制約など，通信インタフェースの仕様の約80％をDSL文法として定義した。また，DSLを用いて記述した設計モデルからソースコードの自動生成を実現することで誤りが混入する可能性を低減した。さらに，あらかじめ決められた値や通信インタフェースの仕様を表現する要素だけを記述できるように入力補完機能などを加えることで，誤った値や内容が記述された場合には警告メッセージを表示するので，意図しない設計の防止が可能である。

今回構築したDSL開発環境では，通信パケット解釈処理部分の開発工数を70％削減，仕様解釈の誤りに起因するシステムテスト工程での不具合件数0件を実現した。



DSLによる開発スタイル

DSLによる開発スタイルは，DSL開発環境構築時とDSLを利用した開発時の２段階で構成している。DSL開発環境構築時には，特定ドメインの知識を基にDSL文法を定義し，DSL開発環境の機能(ソースコードの自動生成など)を開発する。DSL開発環境構築後は，設計者がDSLで設計記述すれば，ソースコードが自動生成される。

*設計システム技術センター

1. まえがき

携帯電話やデジタル家電，自動車など多機能なシステムでは，複数のハードウェアと複数のソフトウェアが連携して動作する。このようなシステムは多くの開発人員と開発期間を要することから大規模組み込みシステムと呼ばれ，複数企業で分担開発される場合が多い。

大規模組み込みシステムのソフトウェア開発では，設計，実装，試験の開発工程を時系列に分割して管理を行うウォーターフォール型開発手法が採用されており，各企業で開発した装置を組み合わせた試験は開発終盤のシステムテスト工程で行われる。この開発手法は前工程が問題なく完了していることを前提とするため，開発終盤で不具合が検出されると，原因を究明して不具合が混入された工程まで戻って修正する。この修正に伴い，不具合が混入した後の工程の開発もやり直す必要があるため，手戻り工数の削減が課題である(**図1**)。

本稿で取り上げる大規模通信システムの開発では，この課題を解決するために，開発装置と通信を行う対向装置との通信を模擬する検証用の通信シミュレータが活用されている。シミュレータの活用によって，他装置との通信機能試験を上流工程で実施し，後工程の試験不具合の発生率の低減を図ってきた。しかし，通信システムの規模が大きくなることで，システムを構成する装置数が増加し，装置間ごとのインタフェースの仕様が複雑・多様化してきている。これに伴い，対向装置ごとに開発を要する通信シミュレータの開発規模の拡大，仕様解釈の誤りによる不具合混入が問題となっている。

これらの問題を解決するために，特定領域のふるまいに特化した設計モデルを記述するDSLと呼ばれるコンピュータ言語を導入した。ソフトウェア開発でのDSLの活用は，Martin Fowler等によって提唱されている[1]。特定領域のふるまいを記述するためのDSLの文法(以下“DSL文法”という。)を定義し，DSLによる記述結果である設計モデルからソースコードを自動生成する機能を作成しておくことで，開発工数削減と品質向上を実現した。

2. 通信シミュレータ導入での課題と対策

2.1 課　題

大規模通信システムは，複数のハードウェアやソフトウェアで構成し，複数の企業で開発している。他社システムや並行して開発している装置との組合せによる通信機能試験は，開発終盤や現地でしか行えない。試験で発生した不具合の多くは，通信インタフェースの仕様を記述するときの曖昧性に起因しており，原因究明，ソフトウェア改修，再試験によって大きな手戻りとなっている。

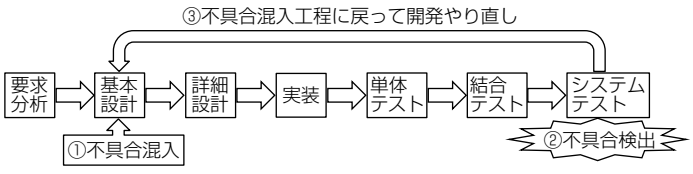


図1．不具合検出による手戻り

そこで，開発装置と通信を行う対向装置との通信を模擬する検証用の通信シミュレータを導入し，通信機能試験の上流工程での実施を図ってきた。しかし，通信シミュレータはデータ通信を行うための信号構成や規約を定義した通信インタフェースの仕様が装置ごとに異なるため，大規模通信システムを構成する全装置を模擬する通信シミュレータを開発する必要がある。通信機能試験を上流工程で実施するためには，通信シミュレータ開発の省力化と期間の短縮が必要である。

2.2 対　策

通信シミュレータは，パケットの送受信処理，解釈処理，表示処理で構成している。このうち，送受信処理，表示処理は，通信インタフェースの仕様に依存しないためソースコードの流用が可能だが，解釈処理は，通信インタフェースの仕様が異なる装置ごとに設計・実装が必要である。そこで，通信インタフェースを対象ドメインとしたDSLの文法を定義し，DSLで記述した設計モデルから解釈処理のソースコードを自動生成するDSL開発環境を構築した(**図2**)。設計者は，設計を補助する入力補完機能，入力チェック機能，DSLの図式記述を活用して設計モデルを記述することで，ソースコードの自動生成が可能となる。

2.2.1 DSL文法の定義

自然言語による仕様の記述には曖昧性が含まれ，仕様解釈の齟齬が発生するおそれがある。そこで，記述の曖昧性を排除するため，通信インタフェースの設計モデルの記述に特化したDSLの文法を定義し，設計者は，DSLで設計モデルを記述することにした。DSL文法は，実装を意識した明確な設計モデルの記述ができるように，製品やシステムの暗黙知も含め，通信インタフェースの仕様を明確に理解している有識者が定義する。DSLで記述された設計モデルからソースコードが自動生成される。このDSL文法を通信インタフェース文法という。**図3**に，設計モデルを記述するための通信インタフェース文法と，通信インタフェース文法を用いて作成した設計モデルの例を示す。通信インタフェース文法は，信号種別，送信先アドレス等のデータ名称，数値型や文字列型などデータの内容を解釈するのに必要なデータ型，データ長や値の範囲等の制約事項など，通信インタフェースの設計モデルを表現する要素で構成する。

2.2.2 DSL開発手法の定義と機能

(1) ソースコードの自動生成機能

DSLで記述した設計モデルからソースコードを自動生成する機能は，通信インタフェースの仕様に依存しない処

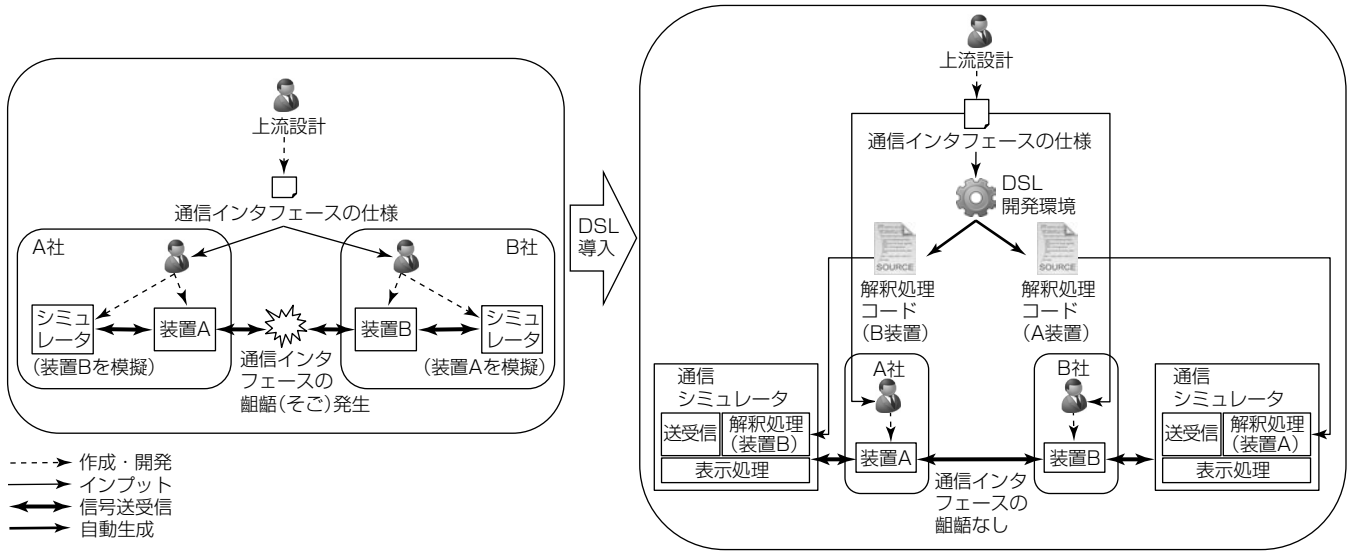


図2．通信シミュレータへのDSL導入イメージ

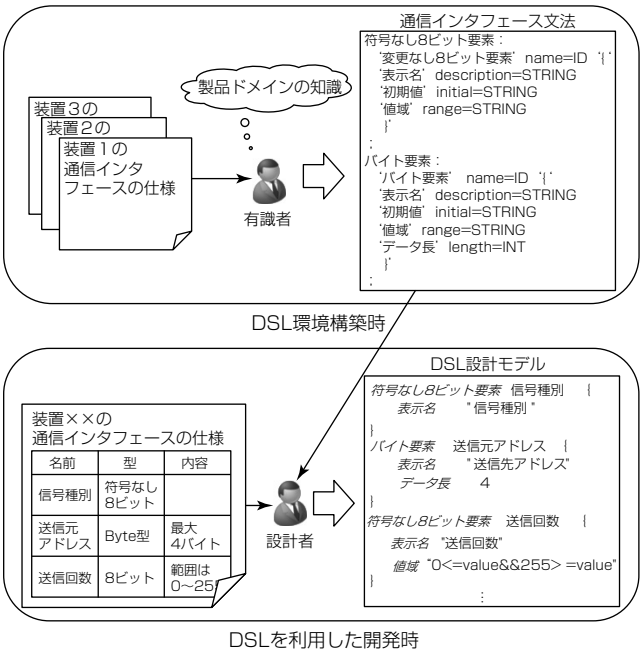


図3．DSLによる仕様記述例

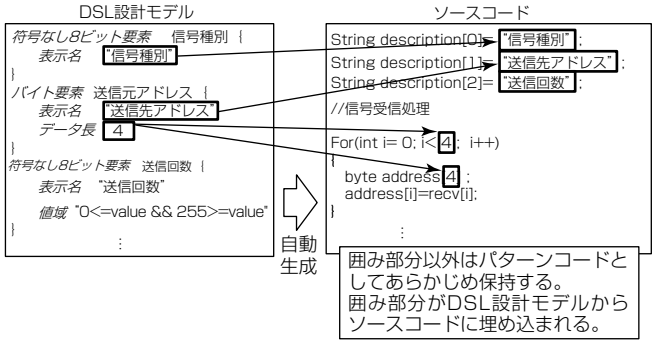


図4．ソースコード自動生成の仕組み例

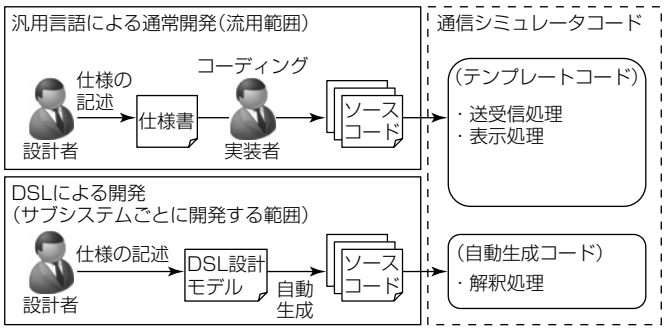


図5．DSLを利用した通信シミュレータの開発

理をパターンコードとしてあらかじめ保持しておき，仕様ごとに異なるパラメータ値等の情報をDSL設計モデルから直接パターンコードに埋め込むことで実現する(**図4**)。自動生成によって，開発工数が削減されるだけでなく，仕様解釈の誤りや実装時の人的ミスの可能性が大幅に減少する。自動生成対象のソースコードは，設計モデルによる変動要素が多い箇所とする。変動が少ない箇所は，通常の汎用言語による実装を行い，流用部分としてテンプレート化し，自動生成コードとテンプレートコードを組み合わせて製品コードとする(**図5**)。通信シミュレータでは，通信インタフェースの仕様の設計情報が多く含まれる解釈処理を自動生成対象とし，通信インタフェースの仕様に依存しない送受信処理，表示処理をテンプレートコードとした。

(2)　設計を補助する入力補完機能

入力補完機能は，任意の設計モデルの記述に対して入力候補となり得る内容を事前に設定しておき，DSLで設計モデルを記述する際に，入力候補を表示することで設計誤りを未然に防ぎ，設計を効率化する。例えば，通信インタフェース文法の構成要素であるバイトオーダー(メモリ配列への格納順序)には，最上位データから格納するビッグエンディアン，最下位データから格納するリトルエンディアンがあり，必ず一方を指定する必要がある。設計者がバイトオーダーを記述する際に，候補としてこれら2種類を表示し，ど

ちらか一方を選択すると選択した文字列が自動で入力される。

⑶ 設計を補助する入力チェック機能

入力チェック機能は，任意の設計記述に対してエラーとなり得る内容をあらかじめ設定しておき，DSLで設計モデルを記述する際に，誤った値や内容が記述された場合には警告メッセージを表示することで，意図しない設計を防ぐことが可能となる。例えば，符号なし整数８ビット型の場合には０～255しか入力できないため，０～255以外の場合に，“符号なし８ビット型は０～255である”という警告メッセージを表示し，早期に誤り発見を促す(**図６**)。

2.2.3 DSL手法の利用容易化に対する開発環境の構築

DSLで記述した設計モデルからソースコードが自動生成されると，実装時の人の介在が低減され，開発の効率化が見込まれる。しかし，通信インタフェースの仕様は仕様書に図表で記載されており，DSLを用いて設計モデルを記述するには人手で行う必要がある。大規模通信システムを構成する装置が数十個であれば，DSLを習得した技術者１人で通信シミュレータの作成は可能であるが，装置が数百個の場合１人では開発期間がかかりすぎるという問題がある。この場合，通信シミュレータを複数人で作成することになるが，DSLを初めて使う技術者はDSL文法を習得する必要があり，特にプログラミングを不得手とする技術者にはDSL導入の敷居が高くなる。そこで，設計モデルの記述を，DSLの代りにブロック図を用いてできる機能を実現することで設計モデル記述の容易化を図った。通信インタフェースの仕様のデータ型ごとにブロックを用意し，ブロックにデータの名称・説明，データ型に合わせた値の範囲を定義する“プロパティ”，ブロック間をつなぐ“シンボル”，ブロック間の“接続可否の定義”によるブロック図式を定義した。

図７に通信インタフェースの設計モデルをブロック図で記述した例を示す。信号のデータ部分を示すデータ部のブロックの下に左から順にデータ型のブロックをおき，シンボルでつなぐことによって信号のデータ部分を表現する。

2.3 効　果

今回，大規模通信システムでの装置間の通信応答を模擬する通信シミュレータ開発用に，通信インタフェースの仕様の約80％をカバーするDSL文法を定義し，DSL開発環境を構築した。これによって，通信パケット解釈処理部分の開発工数が約70％削減された。また，通信インタフェースの仕様をモデル化したことで，設計と実装が標準化され，仕様解釈の誤りに起因するシステムテスト工程の不具合件数０件を実現し，開発終盤での不具合検出による手戻り発生を抑止した。

2.4 今後の課題

装置間の連動する部分の自動試験で，送受信データのテストデータを人手で作成しており，自動試験環境構築に時間と人手がかかっている。今後はテストケース定義と通信インタフェース仕様からのテストデータ自動生成による試験工程の工数削減が課題である。

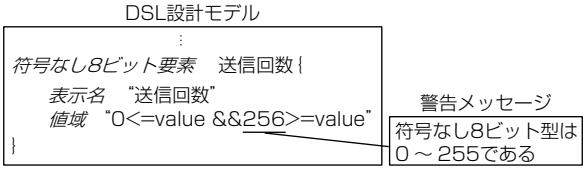


図６．入力チェック機能による警告メッセージ例

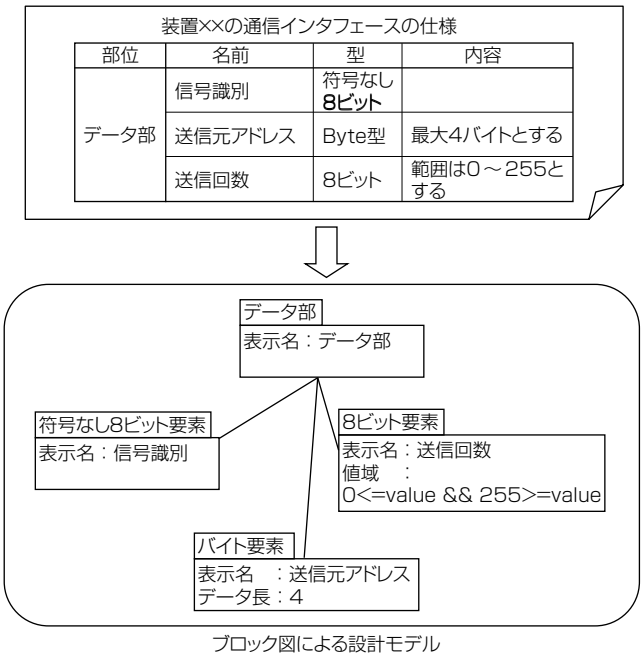


図７．ブロック図による記述例

3. むすび

大規模通信システムの開発では，システムを構成する装置を並行開発しており，各装置の組合せ試験は開発終盤に行うため，不具合発生時の手戻りが大きくなる。対策として，各装置間の通信を模擬した通信シミュレータを導入しているが，装置数が多い場合，通信シミュレータの開発が負荷となる。そこで，DSLを利用したソフトウェア設計手法と通信シミュレータにDSLを導入した事例について述べた。

今後も，システムの大規模化及び要求される品質の高度化に伴い，特定ドメインを対象とした設計・実装のモデル化はますます重要になってくる。DSLを設計だけではなく試験にも活用できるよう進化させ，開発工数削減と品質向上に役立たせていく。

参考文献

⑴ Fowler M.，et al.：Domain-Specific Languages，Addison-Wesley（2010）