モデル変換技術を用いたIoTアプリケーション開発と資産再利用性の検討

大木 憲二[†1] 仲道 耕二[†1] 野村 佳秀[†1] 長谷川 尚己[†1]

概要：IoT アプリは使用する機器，通信方式，開発言語などが多岐にわたり，また試行から運用フェーズに至るまで様々な構成や方式を検証しながら開発が行われるため，開発の手戻りが少ない抽象度の高い開発環境及び手法が求められる．我々は IoT 向けフローベース開発環境である Node-RED に基づくモデルを構築し，2 種類のモデル変換技術によって配備環境に適した実装を生成可能なシステムを構築している．本稿では，IoT アプリに対してモデル変換技術を適用した方式について報告すると共に，その際に生じる再利用性についての検討を行う．

1. はじめに

近年のセンサの多様化やクラウドプラットフォームの進展により，大量データの収集，分析，活用を行う IoT システムへの期待が高まっている．

IoT を活用したアプリの開発では，使用する機器，通信方式，開発言語などが多岐にわたり，またプロトタイピングからリリースに至るまでに様々な構成や方式を検証しながら開発を進めることも多く，開発初期の段階で特定の実装に依存しすぎてしまうと変更時に開発の手戻りが発生してしまう点が課題である．

そこで我々は，抽象度の高い IoT アプリのモデルを M2T（Model To Text）と M2M（Model To Model）の 2 種類のモデル変換技術によって配備環境に適した実装を生成可能なシステムを構築している．開発環境として OSS の IoT 向けアプリ開発環境として広く知られている Node-RED[1]を利用し，Node-RED から得られるフローデータに基づくモデルを構築した．

本システムを用いると，アプリ開発者はコアなロジックを記述することで，変換によって様々な環境で動作検証を行いながら開発できるようになると考えているが，一方で本システムを用いるために一から変換定義やメタモデルを用意することは時折新規開発以上に負担が掛かってしまう．そのため，様々なアセットを再利用できる仕組みや体制を整えた上で運用していくことが不可欠である．

本稿では，IoT アプリに対してモデル変換を適用した方式について報告すると共に，本方式を展開していく際に必要となる再利用性についての検討を行う．

2. IoT アプリ開発時のモデル変換の適用

2.1 変換に対する要件

我々は，Node-RED のような開発環境で実装したコアロジックを実際の環境に配備・動作する際に重要となるであろう 3 つの要件を変換要件として定めた．

(1) 機能の分割

IoT アプリは複数の機器が共調動作する分散アプリであるが，開発初期の段階でネットワーク構成を意識して多数のデバイスそれぞれに機能を作りこむことは困難である他，ロジックが各所に分散してアプリ全体の見通しが悪くなってしまう．開発時にはアプリ全体を単一のフローとして開発し，動作検証時にフローを機能分割して各機器に配備できることが望ましい（図 1 参照）．

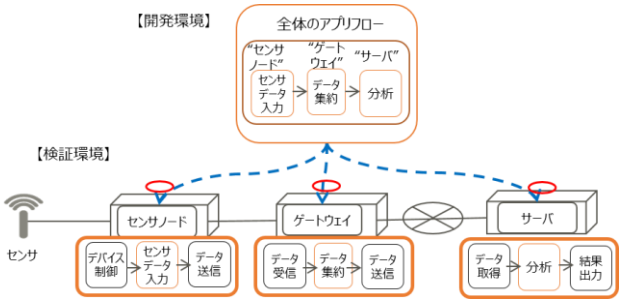

図 1 アプリフローの機能分割

(2) 多種プラットフォームや通信方式への対応

実環境のデバイスは必ずしも Node-RED が動作するとは限らないため，開発したアプリが他のプラットフォーム上でも動くことが求められる．また，各デバイス間の通信は MQTT や HTTP など様々な方式が考えられ，要件に合う方式を選択できる必要がある．

(3) 非機能要件の実装

IoT アプリに求められる品質特性に応じて，レイテンシの計測や通信の再送機能などは主に開発の後行程で必要になってくる機能であるが，開発初期の段階でこのような非機能要件は考慮されにくく，適宜このような機能を追加できることが望ましい．

2.2 2 種類のモデル変換

本システムでは M2T と M2M の 2 種類のモデル変換を用いることで，前節で述べた要件を満たすための変換定義を実装している．両者のモデル変換では Node-RED のフローに基づく共通のメタモデルを用いており，M2T では入力モ

†1 (株)富士通研究所
Fujitsu Laboratories Ltd.

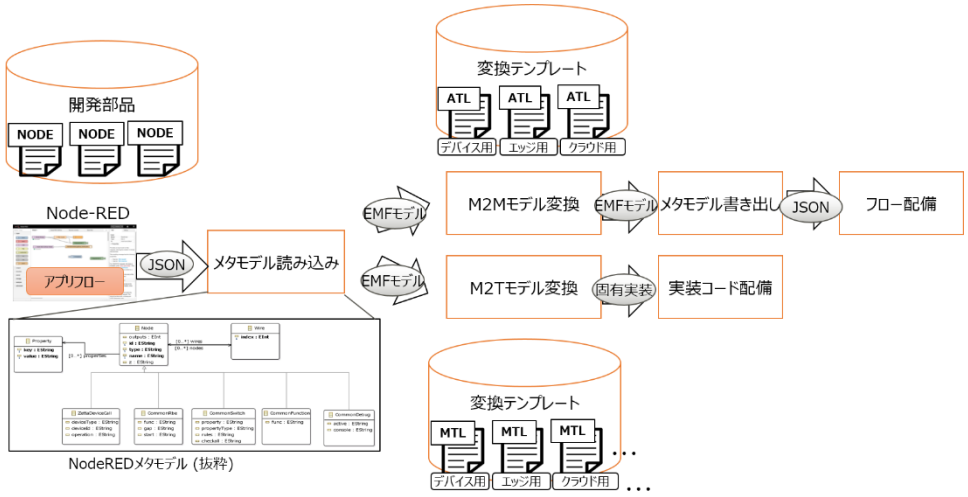

図 2 フロー変換プロセス

デルを実装コードに，M2M では入力モデルと同一メタモデルを持つ Node-RED モデルに変換する．（図 2 参照）

(1) **M2T によるモデル変換**

モデルをテキスト（実装コード）に変換する手法は，特に 2.1 節の(2)の要件を満たすために不可欠である．本システムで M2T の実装ツールとして用いている Acceleo[3]では，予め各プラットフォームと機能毎に MTL で記述したテンプレートを用意して，モデルに対して開発者の要件を満たすテンプレートを適用することで各実装に変換を行う．

(2) **M2M によるモデル変換**

モデルをモデルに変換する手法は，変換後のモデルも Node-RED 上で動かすことを前提としているものである．実行プラットフォームが制限されてしまうが，2.1 節の(1)と(3)の要件充足に注力するために取り入れたものである．本システムでは M2M の変換言語として ATL[4]を用いており，開発者は要件を満たす ATL を適用することで Node-RED 上で動作するフローに変換される．

3. 再利用性の検討

本開発環境を構成する資産の再利用性について述べる．

(1) **変換テンプレート**

M2T 変換テンプレートは 2.1 節(2)で述べた多くのプラットフォームをサポートするために品揃えが必要な構成要素である．プラットフォームごとに通信方式や非機能に対応対応したテンプレートが必要となるため，品揃えのコストが高い点が課題となる．一方で，M2M 変換テンプレートは変換後のプラットフォームを固定しているため，M2T 変換テンプレートより拡充が容易である．

また，要件に見合うテンプレートを適切に選択する手段がないといくら品揃えが充実しても展開していくことが難しい．例えば，機能・非機能要件をフィーチャーとしてモデル化し，モデルを元に適したテンプレートを判断する方法などが必要になる．

(2) **フローメタモデル**

前項の変換テンプレートはフローのメタモデルに従って記述を行うため，本メタモデルの変更は変換テンプレートにも影響が及んでしまう．また，テンプレートが想定しているドメインによって最適なメタモデルは変わってくるため，テンプレートと共にメタモデルをメンテナンスしていく必要がある．

(3) **フロー開発部品**

フロー開発部品は Node-RED の枠組みとして提供されており，新たに作成した部品の再利用する事はもちろん，外部のコミュニティで開発された部品も含めて利用することが可能である．しかしながら，これらの部品がフローのメタモデルに反映されていないと，変換テンプレートの記述が煩雑化してしまう．

本メタモデルでは新たな属性が定義されたフロー開発部品を読み込めるように任意の key と value のペアを許す属性を定義しており，テンプレート内で自由に参照が可能になっているが，新たに設定された key をメタモデル側に吸い上げていく仕組みが再利用性向上のために必要となる．

4. おわりに

本稿では IoT アプリケーション開発を行うためのモデル変換システムの方式について説明し，本システムを展開していく上で課題となる再利用性についての検討を行った．今後，このような再利用が必要となる部品の構築手法についても検討し，開発環境と併せて展開していきたいと考えている．

参考文献

[1] “Node-RED”, https://nodered.org/
[2] “Acceleo”, http://www.eclipse.org/acceleo/
[3] “ATL”,
http://wiki.eclipse.org/MMT/ATL_Transformation_Language_(ATL)