IoT機器に対するEXI利用時のスキーマ簡約による ROM必要量最適化

米澤 祐紀[1]　土井 裕介[1]

概要：
IoT (Internet-of-Things) はさまざまな領域での応用が期待される。そのうちの一つに、スマートグリッドなどを含む各種産業向けシステムがある。このようなシステムでは、異なるベンダの異なる機器の統合制御に、各機器・機能に応じた標準が定義される。産業向け標準仕様では、多くの場合 XML スキーマによるデータモデルやメッセージ形式の定義が行われる。一方、標準で定義される多様なデータモデルを含む XML スキーマは肥大化しがちである。このような肥大化した XML スキーマを、IoT で用いられる組込み機器で処理するのは最適ではない。
本研究では、計算機資源に制約のある組込み機器を、XML ベースの標準仕様に統合するために、仕様で定義された XML スキーマを機器にあわせて最適化する方法 (XML スキーマ簡約) を述べる。あわせて、本手法を EXI に適用した際の EXI 文法のサイズ削減効果についても示す。

キーワード：IoT, XML, スキーマ, 組込み機器, EXI

IoT Device ROM Optimization for EXI Processing with XML Schema Contraction

Yuki Yonezawa[1]　Yusuke Doi[1]

1. はじめに

1.1 背景

Internet of Things (IoT: モノのインターネット) は、さまざまな領域で爆発的な普及が期待されている。IoT の先行例の一つに、スマートグリッドシステムを代表とする各種産業向けシステムがある。そのようなシステムでは、異なるベンダの異なる機器の統合制御のために、各機器・機能に対応した標準が定義される。

データ処理において、特に拡張性や比較容易性の観点から、データには、その形式と他のデータとの関係を記述する、スキーマの存在が期待される。適切なスキーマが存在すれば、得られたデータが何であるか、二つのデータは比較可能なのか、等が自動的に判別可能である。また、あるスキーマから別のスキーマへの射影が定義できれば、個別のデータを異なるスキーマにおいて比較・評価が可能になる。

システム間でのデータ表現を共通化するために、ASN.1、JSON、W3C XML Schema といった標準的なデータ表現方法が存在する。特に W3C XML Schema は、名前空間を持ち、拡張性や再利用性に優れているため、複数のベンダ・多様な機器が混在する産業向けシステムにおいて有力なデータ表現方法の一つである。

1.2 課題

本研究で解決する課題は、通信仕様の定義に用いられる XML スキーマが、計算機資源が潤沢なサーバ・制約された多様なクライアント (組込み機器) 、という非対称の環境において、組込み機器側での XML スキーマが最適でなく、これにより実装規模が肥大化するというものである。

XML は、拡張性と可読性に優れているが、データ表現の冗長性の高さ、文法の複雑さにより、XML の利用には多くの計算機資源を必要とする。データ表現の冗長性については、XML のバイナリ版表現である、EXI (Efficient XML

[1] 株式会社東芝　研究開発センター
Corporate R&D Center, TOSHIBA Corporation

Interchange) [1] 標準により解決できる。EXI にはスキーマを利用した効率的な符号化方式と、スキーマを利用しない符号化方式が存在する。スキーマを利用しない方式を効率的に利用するには、動的に文法を生成する必要があり、一般に RAM の制約が厳しい組込み機器においてこの方式の利用は難しい。従って、スキーマを利用した符号化方式が組込み機器においては適している。

スキーマを利用した符号化方式において、XML スキーマから変換される EXI 文法を用いる。EXI 文法は XML スキーマに対応するプッシュダウンオートマトンであり、およそ XML スキーマのサイズに比例する。多様な機器を含むデータモデルを定義した標準規格において、この機器全てのデータモデルを内包する XML スキーマは肥大化しがちである。従って、組込み機器において EXI を処理する場合においても、EXI 文法のサイズが大きくなってしまい、対応困難になってしまう。

一方、個別の機器に着目した場合、その機器に必要なデータモデルはごく一部である場合が多い。ただし、実際には XML スキーマのデータ構造 (抽象化) は機器にあわせたものとは限らず、機器に必要な部分を残して不要な部分を排除する、という処理方法は自明ではない。

本研究では、個々の機器が利用する部分にあわせて、XML スキーマの抽象構造を最適化する処理を、XML スキーマ簡約と呼ぶ。特に本稿では、XML スキーマのモデルの簡約化手法の提案とその効果を評価する。

2. 関連研究

Kyusakov 他 [2] は、組込み機器向けの EXI エンコーダ・デコーダ: EXIP を作成し、ルネサス社製 M16C/65 プロセッサ上で gSOAP を用いた SOAP にもとづく通信を実現している。EXIP はストリーム型のメッセージ処理 API を持っており、省メモリ処理が可能な構造になっている。また、デコードに用いる XML スキーマは動的に生成することも、静的なオブジェクトとして持つこともできる。

EXIP は組込み機器に適した構造を持っているが、先に述べたように EXI の性質により、デコードには EXI 文法が必要になる。スキーマが大きくなった場合、EXI 文法は組込み機器に搭載するには適切でない大きさになる場合がある。これに対し、Caputo 他 [3] は、センサに用途を限定し、デコーダを削除しエンコーダのみに改造した EXIP を Contiki に搭載している。これを STM32W を搭載したマイコンに搭載している。

EIGEN[4] は、組込み機器をターゲットとした EXI 処理エンジンである。EXIP と異なり、EIGEN は組込み機器が利用するデータを構造体の形で限定し、コードの自動生成により機器毎に最適化された EXI 処理エンジンを生成する。ただし、EIGEN においては、受信する EXI ストリームについてはそのままデコードできる必要があることから、デコーダに搭載する EXI 文法は XML スキーマに由来するものであり、XML スキーマのサイズに比例したサイズの ROM が必要になるという課題が残っている。

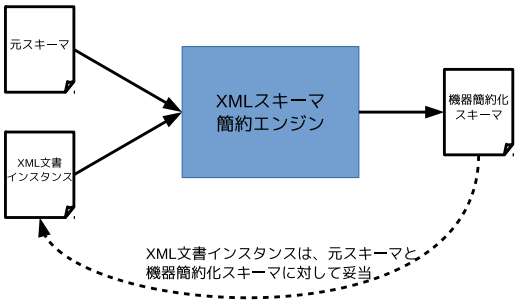

図 1 XML スキーマの簡約化

3. 機器仕様にあわせた XML スキーマ簡約

本節では、XML スキーマで規定する膨大なデータモデルや、その抽象定義から、個別の機器に合わせたデータモデルのみを残す、XML スキーマ簡約化手法を提案する (**図 1**)。ここで、仕様などにより定められた、サイズの大きい XML スキーマを元スキーマ、機器毎に最適化されたスキーマを機器簡約化スキーマと呼ぶ。

本提案手法は、機器で利用するデータを記述した XML 文書例を用い、一定の手続きにより元スキーマを簡約化し、その文書が妥当である機器簡約化スキーマを生成する。生成する機器簡約化 XML スキーマは、元スキーマの複雑さを継承しない。出力した XML スキーマは、XML の妥当性検証、EXI ストリームのデコード等を、最小限の ROM で動作させることを可能とする。

本稿が対象とする XML スキーマの簡約化処理は、次の二つである。一つ目の簡約化処理は、オプションの簡約化である。XML スキーマにおいて、データ型のうち、記述してもしなくても良い項目 (オプション項目) を定義できる。一方、個々の機器が利用する要素はある程度決まっている。従って、機器が利用する範囲にオプション項目の絞り込みは、XML スキーマ簡約として有効な手法である。また、EXI 固有の問題として、オプションの項目が連続すると、EXI 文法の状態遷移数が爆発的に増大する、という問題がある[*1]。その項目がない場合は機器にとってエラーとなるような、その機器にとっての必須項目があれば、オプション項目の必須項目への置き換えで、さらに文法の簡約化が可能である。

二つ目の簡約化処理は、派生型の簡約化である。オブジェクト指向設計において、多様な機器・機能を効率的に管理するために、中間的な抽象化モデルを定義し、個別の機器は種別毎、機器毎の派生型として定義する。構造化されたモデルの処理には、モデルの持つ上位モデルも処理する必要があり、複雑さが増大する。中間の抽象モデルを簡

[*1] 項目数 n に対して遷移数は $(n-1)!$ となる。

約化し、個々のデータモデルを具体化することで、XML スキーマは単純化可能である。

4. XML スキーマ簡約の実装

XML スキーマ簡約の実装として、XML スキーマにおけるオプションと派生型の簡約化方法を述べる。

本研究では、機器は少数のパターンの XML 文書のみを受信し、これ以外のものは処理できないと仮定する。また、各機器が受信可能な XML 文書のパターンは、少数の具体的な XML 文書 （XML 文書インスタンス) により表現できるものとする。

4.1 オプションの簡約化

本節では、対象とする XML 文書インスタンスから、元スキーマの不要なオプション項目を削除することにより、XML 文書インスタンスの妥当性を損うことなくスキーマを簡約する手法について説明する。

XML スキーマに妥当な XML 文章インスタンスから、その XML 文章インスタンスを妥当とする XML スキーマの要素を抽出することは、XML スキーマにおけるオプション項目のうち、必要な項目だけを抽出していることと同義である。XML スキーマにおけるオプション項目は、型を構成する要素のオプションとグローバルに宣言される Element の二つである。型に対するオプション項目は、Element の minOccurs 属性に 0、Attribute の use 属性に optional を指定する。グローバルに宣言される Element のうちいずれか一つが、XML 文書インスタンスのルート要素となり、利用されていない他の要素はオプション項目となる。

本稿では、次に示す手順で、オプション項目を簡約化する。

(1) XML 文書インスタンスが利用している要素・属性の一覧を作成
(2) 手順 (1) で作成した一覧に対応する元スキーマの要素・属性を記録
(3) 記録した要素・属性だけを含んだ機器簡約化スキーマを出力

次に、オプション簡約化による EXI 文法への影響について、**図 2** を用いて説明する。図 2 に示す XML スキーマでは、Note 型の name=”body” の要素がオプション項目である。この XML スキーマから生成される EXI 文法は、**図 3** に提示するようなオートマトンを生成する。なお、図中の遷移に与えられたラベル (SE:title など) は EXI の内部表現に対応する ([1] Table 4-1) 。

手順 (2) は、図 3 で示した各状態のうち、利用する状態を記録することを意味する (図中では各状態にマーク)。**図 4** は、図 2 に示す XML スキーマに対し妥当な XML 文書インスタンスである。図 4 を用いると、**図 5** に示すように、オートマトンに対応する状態を記録できる。よって、EXI

```
<xs:element name=”note” type=”Note” />

<xs:complexType name=”Note”>
   <xs:sequence>
      <xs:element name=”title” type=”xs:string” />
      <xs:element name=”body” type=”xs:string” minOccurs=”0” />
      <xs:element name=”author” type=”xs:string” />
   <xs:sequence>
</xs:complexType
```

図 2 オプション要素を含んだ XML スキーマの例

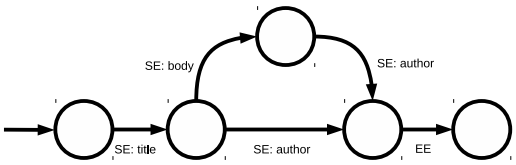

図 3 図 2 の EXI 文法に対応するオートマトン

```
<note>
   <title>こんにちは</title>
   <author>Tarou Toshiba</author>
</note>
```

図 4 図 2 の XML スキーマに対し妥当な XML 文書

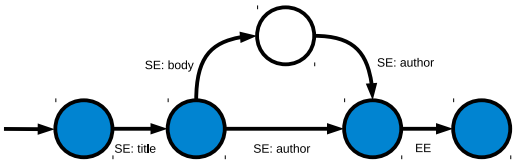

図 5 オプション簡約化手順 (2):図 4 の XML 文書インスタンスを用いて、必須な要素・属性を記録したオートマトン

図 6 オプション簡約化手順 (3):オプション簡約化による機器簡約化スキーマの EXI 文法に対応するオートマトン

文法の中から、図 4 に示す XML 文書インスタンスの必須項目が明らかになる。

手順 (3) の出力する機器簡約化スキーマは、**図 6** に示す EXI 文法のオートマトンを生成する。図 6 は、図 4 を受け入れる限度内で最適化されている。

4.2 派生型の簡約化

XML スキーマでは、抽象化モデルにもとづく型から、継承による具体的な型に派生させることができる。**図 7** の例では、C にもとづく型の具体的な型として D と E が派生している。また、C は B から、B は A からそれぞれ派生した型である。

派生型の簡約化は、ある具体的な型の元となっている抽象的な型の要素と属性を、一つの型に簡約する。元スキーマが、図 7 に示す派生関係で、かつ、XML 文書インスタンスで D だけを利用する場合、A から D を一つの型に簡約化する。

一方、図 7 に示す派生関係で、XML 文書インスタンス

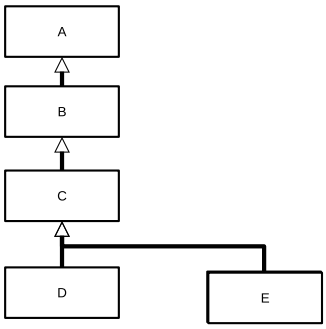

図 7 派生型簡約化の手順 (1):D と E は C から、C は B から、B は A から、それぞれ派生した型であることを示している

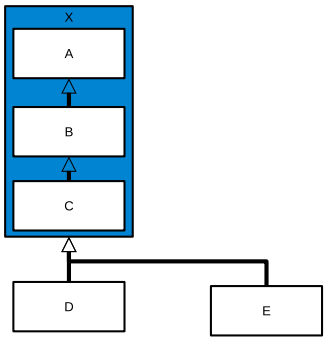

図 8 派生型簡約化の手順 (2):C が D と E の二つの型を派生しており、C が A から C、A から E の分岐点であることを示している

で D と E を利用する場合は、**図 8** に示すように、A、B、C の要素を含む抽象的な型 X と、X から派生した D と E、との三つの型を作成する。仮に、抽象的な型 X を作成せず、A から D と A から E をそれぞれ一つの型とした場合、簡約化した二つの型は独立した型となり、オブジェクト指向における構造化を壊してしまう。

以上を踏まえて、派生型の簡約化に対し、次の三つの手順で実現した。

(1) XML スキーマの xs:extension から、派生関係の調査
(2) 手順 (1) の調査結果を用いて、最も抽象化されている派生元から分岐点を検知
(3) 分岐点に至るまでの型を、ひとつの型に変換した機器簡約化スキーマを出力

次に、派生型簡約化による EXI 文法への影響を説明する。EXI 文法は、XML スキーマに定義されている各型に対応する EXI 文法を作成する。つまり、図 7 に示すような派生関係の場合では、五つの EXI 文法が必要である。

一方、図 8 に示すように派生型関係を簡約化することで、A、B、C を一つの型で表現できる (図 8)。これを EXI 文法にした場合、五つ必要だった EXI 文法を二つ削減でき、ROM サイズ削減に繋がると予測する。

5. 評価

今までに述べた設計を元に構築した XML スキーマの簡約化エンジンを用いて、オプション・派生型の簡約化の効果を評価する。

表 1 OpenADR2.0a の元スキーマで宣言・定義されている要素数

	数
Element	48
ComplexType	13
SimpleType	10

表 2 OpenADR2.0a のデータモデルにオプション簡約化手法を適用した機器簡約化スキーマで宣言・定義されている要素数

	数
Element	5
ComplexType	0
SimpleType	1

5.1 評価の条件

簡約化の効果測定には、EIGEN で作成する EXI デコーダのオブジェクトサイズを用いる。また、元スキーマと機器簡約化スキーマで宣言・定義されている要素数を比較することで、期待通りの簡約化が実施されていることを確認する。

派生型簡約の評価では、オプション簡約化後の XML スキーマに対して、派生型の簡約化手法を適用する。本稿が提案する派生型の簡約化手法は、1 対 1 の親子関係を対象とする。一方、一般的な型の構造化では、一つの型を複数の型に具体化するため、対象とする親子関係は稀である。しかし、オプション簡約化を実施した XML スキーマでは、具体化されていた複数の型が絞り込まれるため、1 対 1 の親子関係となる場合がある。

5.1.1 オプション簡約化の効果

オプション簡約化の評価には、OpenADR2.0a[5] の XML スキーマを用いた。OpenADR2.0a で宣言・定義されている要素数を**表 1** に示す。

本稿では、OpenADR2.0a の oadrResponse メッセージを利用した。oadrResponse メッセージは、クライアントである組み込み機器が、デマンドレスポンス要求に対する応答として、サーバから受信するメッセージの一つである。oadrResponse メッセージは、5 つの Element で構成するメッセージである。本稿では、OpenADR アライアンスから認証を得た OpenADR ソフトウェア (VTN[*2])[6] が出力する XML 文書インスタンスを用いて、OpenADR2.0a の元スキーマに、オプション簡約化手法を適用した。

オプション簡約化手法の適用後の XML スキーマに宣言・定義されている要素数を**表 2** に示す。表 2 中の Element 数は、oadrResponse メッセージが利用している Element 数と一致している。よって、オプション簡約化手法により、必要な要素が絞りこめているとわかる。

[*2] Virtual Top Node:デマンドレスポンス信号の発行サーバ

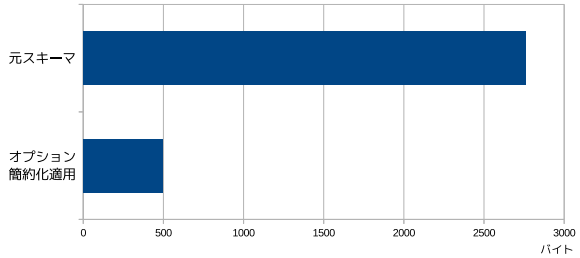

図 9　OpenADR 2.0a 元スキーマとオプション簡約化後の機器簡約化スキーマを用いた EXI デコーダのオブジェクトサイズ

表 3　ZigBee SEP2 の元スキーマで宣言・定義されている要素数

	数
Element	312
ComplexType	312
SimpleType	25

OpenADR 2.0a の元スキーマとオプション簡約化後の機器簡約化スキーマのそれぞれから生成される EXI デコーダ中の静的 EXI 文法部分のオブジェクトを**図 9** に示す。元スキーマは、2759 バイトを必要としていた。一方、オプション簡約化適用後では、495 バイトとなり、オブジェクトサイズを約 80%削減できた。

5.1.2　派生型簡約化の効果

OpenADR2.0a では、抽象化モデルからの具体化が一つ型でしか実施されていないため、派生型の簡約化評価には不適切だと考え、ZigBee SEP2(Smart Energy Profile 2) の XML スキーマと XML 文書インスタンスを用いた。ZigBee SEP2 の XML スキーマの規模を、**表 3** に提示する。

ZigBee SEP2 の XML スキーマにもとづいた XML 文書インスタンスを用いたオプション項目の簡約化により、EXI デコーダのオブジェクトサイズは、37,817 バイトから 481 バイトとなった。オプション簡約化後の XML スキーマで、宣言・定義されている要素を**表 4** に示す。表 4 で示されている Element の宣言は、XML 文書インスタンスのルート要素である。また、五つの ComplexType のうち四つは、**図 10** に示すような、1 対 1 の親子関係を成す。残った一つの ComplexType と SimpleType は、図 10 中の AbstractDevice 型の sFDI の型定義に用いられている。

次に、オプション簡約化後の XML スキーマに、派生型の簡約化手法を適用した。派生型簡約化後の XML スキーマで、宣言・定義されている要素数を表 4 に示す。派生型簡約化前（表 4）と比較すると、図 10 に提示した四つの ComplexType が、一つの ComplexType に簡約化された数と等しいため、正しく簡約化されていることがわかる。

派生型の簡約化により、EXI デコーダのオブジェクトサイズは、481 バイトから 405 バイトとなり、オブジェクトサイズを約 15%の削減できた。

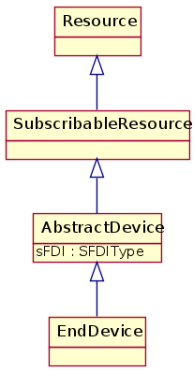

図 10　ZigBee SEP2 元スキーマへのオプション簡約化適用後の XML スキーマ UML クラス図

表 4　ZigBee SEP2 元スキーマへのオプション簡約化適用後の宣言・定義数

	数
Element	1
ComplexType	5
SimpleType	1

表 5　ZigBee SEP2 元スキーマへのオプションと派生型簡約化適用後の宣言・定義数

	数
Element	1
ComplexType	2
SimpleType	1

6.　おわりに

本研究では、標準仕様などにより定義される、さまざまな機器仕様を包含する XML スキーマを、個別の組込み機器が利用する部分のみを残して機器別に最適化する、XML スキーマ簡約化手法を提案した。本手法により、機器側が利用する XML 文書について、最適化された XML スキーマと元々の仕様にもとづく XML スキーマのそれぞれにおいて妥当性検証が可能である。XML スキーマは EXI の復号において必須であり、XML スキーマを簡約化することにより機器側の実装を軽くできる利点がある。

多様な機器を包含するデータモデルの定義において、オプションの定義による多様性の吸収と、中間的な抽象化型の定義と派生による多様性の定義が、データモデルの複雑化の要因であると仮定し、オプション簡約化と型派生簡約化のそれぞれを検討した。

ある XML 文書インスタンスに対して、オプション簡約化では、利用しないことが確定しているオプション要素・属性をスキーマから削除する、という手法を取った。さらに、利用しないことが確定した要素を削除することにより、これに対応する型も削除できるため、型派生の構造がシンプルになる、これについては、型派生簡約化により派生先と派生元をひとつにまとめることで簡約化を行った。

それぞれについて、XML を利用した電力向け仕様である、OpenADR2.0a と ZigBee SEP2 の XML スキーマを利用した評価を実施した。組込み機器で解釈する必要がある典型的なメッセージ形式を決め、これに対応する機器簡約化スキーマを生成した。当該メッセージについて、機器簡約化スキーマによる妥当性が検証できることを確認した上で、著者らによる EXI デコーダに必要となる ROM サイズの比較を行った。XML スキーマの構成に依存するが、例えば OpenADR2.0a スキーマに対してはオプション簡約化により必要 ROM 量を 2759 バイトから 495 バイトへと 80%程度削減でき、SEP2.0 スキーマについてはオプション簡約化と派生型簡約化により約 37kB から 405 バイトに必要 ROM 量を削減できることを確認した。

以上から、XML 簡約により、個別の組み込み機器に合わせた最適化と、XML スキーマ/XML 文書インスタンスによる仕様に厳格なデータ処理とを両立できる。これにより IoT 機器による XML ベースの標準の利用に向けた障壁を一つ減らすことができたと言える。

参考文献

[1] Schneider, J. and Kamiya, T.: Efficient XML Interchange (EXI) Format, W3C Recommendation (2011).

[2] Kyusakov, R., Eliasson, J. and Delsing, J.: Efficient Structured Data Processing for Web Service Enabled Shop Floor Devices, *Proc. of IEEE International Symposium on Industrial Electronics (ISIE) 2011*, pp. 1716–1721 (2011).

[3] Caputo, D., Mainetti, L., Patrono, L. and Vilei, A.: Implementation of the EXI schema on wireless sensor nodes using Contiki, *Proceedings of 2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pp. 770–774 (2012).

[4] Doi, Y., Sato, Y., Ishiyama, M., Ohba, Y. and Teramoto, K.: XML-Less EXI with Code Generation for Integration of Embedded Devices in Web Based Systems, *Proceedings of 2012 International Conference of the Internet of Things*, pp. 80–87 (2012).

[5] Alliance, O.: OpenADR 2.0 Profile Specification A Profile (2011).

[6] : 早稲田大学 新宿 EMS 実証センター Grid EMS 及び電力 DRAS，東芝レビュー，Vol. 69, No. 3 (2014).