Vol.2017-DPS-172 No.26
Vol.2017-SPT-26 No.26
Vol.2017-EIP-78 No.26
2017/11/30

データペトリネットによるIoTシステムのオーケストレーション問題の定式化と考察

山口直郁[1,a)]　Mohd Anuaruddin Bin Ahmadon[1,b)]　山口真悟[1,c)]

概要：Internet of Things（IoT）は多くの異なるデバイスとサービスにより構成されている。一般的に、いろいろなデバイス同士をオーケストレーション（結合）する段階において別の環境へと移した場合にも相互運用性を持たせることは難しい。これはIoTサービスがデータの解釈やサービスの相互運用性の問題により再利用ができないことに起因している。今までのサービスオーケストレーションはメッセージパッシングのプロセスのみに注目されており、データの取り扱いについての検討はなされていない。今回、我々はコンセプトとして「design once, provide anywhere（たった1度の設計で、どこでも使える）」を掲げている。まずは本研究での問題の定式化を行い、それを解決するために用いた手段として、(i)IoTサービスをデータ指向型のサービスモデルとして設計。(ii) サービスのオーケストレーションの補助としてコンテキストオントロジーを利用する。という2つを提案し、考察した。最終的に筆者らは、さまざまなオントロジーを持ったIoTサービスを本アプローチで構成できたことを例を用いて示す。

キーワード：Internet of Things, データペトリネット, オントロジー, サービスオーケストレーション

Consideration and Formalization of Orchestration Problem for IoT System Based on Data Petri Net

Naofumi Yamaguchi[1,a)]　Mohd Anuaruddin Bin Ahmadon[1,b)]　Shingo Yamaguchi[1,c)]

Abstract: The Internet of Things is composed of many heterogeneous devices and services. In general, the phase of orchestrating different devices in order to allow interoperability in different environment is difficult. This is because most IoT services are not reusable because of data interpretation and service interoperability problem. In this research, we embrace the concept of design once, deploy anywhere for IoT services. We proposed two major methods which are (i) modeling IoT services with data-aware service model and (ii) semantical approach using context ontology to support service orchestration. Finally, we showed that IoT services with various ontologies can be composed based on our orchestration method.

1. はじめに

Internet of Things（IoT）は多くの異なるデバイスとサービスにより構成されている。一般的に機器を利用するときにはそれらの管理を行うデバイスから1つずつ役割の指定を行う。IoTサービスはそれらを組み合わせて一連の動作環境を構築したものである。ここでのサービスとは指定されたデバイスを利用する仕組み（もしくは仕様）のことを言う。このサービスには、同じようなモノを利用したい人がわざわざ自分で一から作ることなく、インターネット上から設計図をダウンロードして必要な機器をそろえることで簡単に利用できるようにするという目的がある。しかしこの設計図を使用するときに問題が発生する、人によってデバイスの種類が違うのだ。IoT環境内では各デバイス間の相互運用性が達成されることでサービスが提供される。しかしながら、サービス内でのプロセスが曖昧な状態での

[1] 山口大学
Yamaguchi University
[a)] u079ff@yamaguchi-u.ac.jp
[b)] anuar@yamaguchi-u.ac.jp
[c)] shingo@yamaguchi-u.ac.jp

Vol.2017-DPS-172 No.26
Vol.2017-SPT-26 No.26
Vol.2017-EIP-78 No.26
2017/11/30

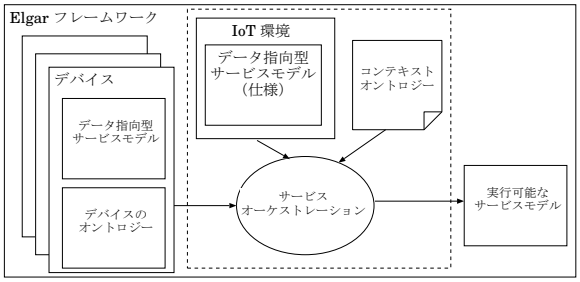

図 1: Elgar フレームワークの全体像

実行は好ましくない。例えばそのデバイスが温度計だったとしよう。その環境内で想定されている温度の単位は摂氏度（℃）である、だが実際に接続した機器は温度をケルビン度（K）で返すものだったのだ。これによる弊害は単純にサービス自体が実行できないだけではない。サービスが予期せぬ動きを行い、使用者の身に危険を及ぼすかもしれない。実際、別のデバイスへの変更や動作環境の移行の際にはサービスの構成を元から設計し直す必要がある。さもなければ、サービスの動作も安全性についても保証はできない。その解決のためにはサービス内で取り扱うデータについての解釈やサービスの仕様の定義を行うべきである。

IoT サービスの管理とオーケストレーションに関して、関連研究が行われている。Lee らは [1] 中で、デバイスのオブジェクト化によるサービスのオーケストレーションをサポートするサービスプラットフォームを提案した。しかし、主にオブジェクトの管理に焦点を当てており、正式なオーケストレーションは行われていなかった。その後、筆者らの研究チームは [2] 中にてクラシックなペトリネット [3] に基づいたオーケストレーションの手法を提案した。彼らは、オーケストレーション方法がコアサービスの動作を保持していることを示した。しかし、そこでもまたデータの相互解釈に関してサービスの相互運用性を達成する必要があった。その後、文献 [4] 中でコンテキストオントロジーをサポートするためのデータ指向型サービスの構成を行う「Elgar フレームワーク（図 1 参照）」を提唱し、これを用いたサービスのオーケストレーション手段についての提案を行った。

本稿で筆者らは、[4] に基づいてコンテキストオントロジーの持つアドバンテージについて述べていく。導入である本章の後は、サービスの仕様の視覚的なモデルとして使用する”データペトリネット”の定義を 2 章で行う。続く 3 章では (i)IoT サービスのデータについての考慮を行ったサービスモデルとしての設計、(ii) サービスのオーケストレーションの補助としてコンテキストオントロジーを利用する。という 2 つの手段を紹介する。また、設計の流れを例を用いて示した。

2. データペトリネット

データペトリネット（以下 DPN と呼ぶ）は 6 つの要素 $DPN = (N, V, U, R, W, G)$ により構成される。それぞれ、ペトリネット [5], [6] N=(P, T, F, ℓ), 変数 V, 変数 $v \in V$ の値域を定義する関数 U , 変数からトランジションへと読み込まれる値 R, トランジションから変数へと書き込まれる値 W, 各トランジションのガード関数 G. ガード $G(t)$ が「true」を返すときトランジション $t \in T$ は発火可能であり、ガードが存在しなければ常に $G(t)$=$true$ である。本稿では、DPN のサブクラスをデータワークフローネット (以下 DWF-net と呼ぶ) と定義している。N が入力プレース p_I と出力プレース p_O をそれぞれ 1 つずつ持つワークフローネットであればその DPN は DWF-net である。

DPN の発火シーケンスのトレースは (t, ϕ) と表される。ここで $t \in T$ かつ ϕ はいくつかの値 $v \in V$, すなわち v^w=x, v^r=y を差し引いた関数である。 v^w=x (v^r=y) は $v \in V$ からの書き込み（読み込み）x (y) を示す。トレースを σ=$\langle t_1\{\phi_1\}, t_2\{\phi_2\}, \cdots, t_n\{\phi_n\}\rangle$ と表記する。有効な発火 (t, ϕ) は [3] 中の図 4 のような状態変異の規則を満たす。

DPN のマーキングは (M, A) の組で表され、D=$\bigcup_{v \in V} U(v)$ である。 M はペトリネットのワークフロー (P, T, F, ℓ) である。A は各変数の値を示している。すなわち、$A : V \rightarrow D \cup \{\perp\}$ であり $A(v) \in U(v) \cup \{\perp\}$ である。もしも値が存在しなければ $A(v)$= $\perp$ となる。M_0=$[p_I], \forall v \in V : A_0(v)$= $\perp$ における最初のマーキングを (M_0, A_0) と表す。

3. Elgar フレームワーク

当フレームワーク（図 1）は環境全体と、相互に接続されたデバイスという 2 つを重要な要素として焦点を当てている。我々の目的は、IoT サービスレイヤー上のサービスフレームワークを構築し、環境内のサービスとの互換性を持たせることである。このことから、データ同士のオーケストレーション手段を考慮したサービスモデリングを行うこととする。フレームワークに入力するのは、サービスモデル自身とオントロジーの記述を含んだプロファイルである。フレームワークのコアとして、サービスの仕様を視覚化した IoT サービスモデルとサービス間のデータの互換性をサポートするためのコンテキストを設定している。本フレームワークは、さまざまなデバイス間でのサービスの構成をサポートすることが目的であり、コンセプトとして、「design once and provide anywhere（たった 1 度の設計で、どこでも使える）」をかかげて構成を行った。

Vol.2017-DPS-172 No.26
Vol.2017-SPT-26 No.26
Vol.2017-EIP-78 No.26
2017/11/30

3.1 データ指向のサービスモデル

IoT サービスモデルには 2 つのタイプがある。1 つ目は、環境内のサービス全体を記述したユーザー定義のサービスモデルである。2 つ目は、デバイスによるサービスの処理を記述したデバイス仕様である。IoT デバイスであるセンサーやアクチュエータは共有されたデータを基に動作を行う。このため、プロセスのみではなく、データについても取り扱うことのできるモデルを使用する。

定義 1 (サービスモデル):サービスモデル D は DWF-net (DPN, M_0, A_0) であり、サービスを表す。 □

本稿では IoT サービスの実例としてスマートファームサービスを提案する。このスマートファームの概要図を図 2 に示す。また、図 2 のサービスの各デバイスの DWF-net を図 3 に示す。この時、それぞれのデバイスのサービスモデルはサービスのデータフローとプロセスを表している。例として使用するこのサービスにはデバイスとして、観測機 D_1、温度センサー D_2、湿度センサー D_3、肥料機 D_4、空調設備 D_5、スプリンクラー D_6、タイムスタンプ D_7、収穫用機械 D_8 が使用される。これらのデバイスにより構成されたスマートファームサービスを図 4 に示す。スマートファームサービスの仕様を下に記述する：

> スマートファームサービスは野菜などの育成から収穫までを一括で行うサービスである。タイムスタンプと観測機によって得られた成長度のデータにより育成と収穫を管理しする。育成の場合は、温度調整、水やり、肥料の管理（散布）を行う。温度調整は、温度センサーからの温度データ（単位℃）を利用して空調機による適温の維持をする。水やりは土の中の湿度計の湿度データ（単位%）に応じてスプリンクラーによる水分量の調整を行う。肥料については、観測機からの成長度のデータを参照しての管理を行う。これらをタイムスタンプのデータと観測機からの情報で適正な頃合いまで育成処理を繰り返す。収穫の際は収穫用の機械により収穫を行い、取れた量（単位 kg）を計測する。

定義 2 (サービスの仕様):S_{spec} は DWF-net(DPN_{spec}, M_0, A_0) でありサービスの仕様を表す。 □

図 2: スマートファームの概要図

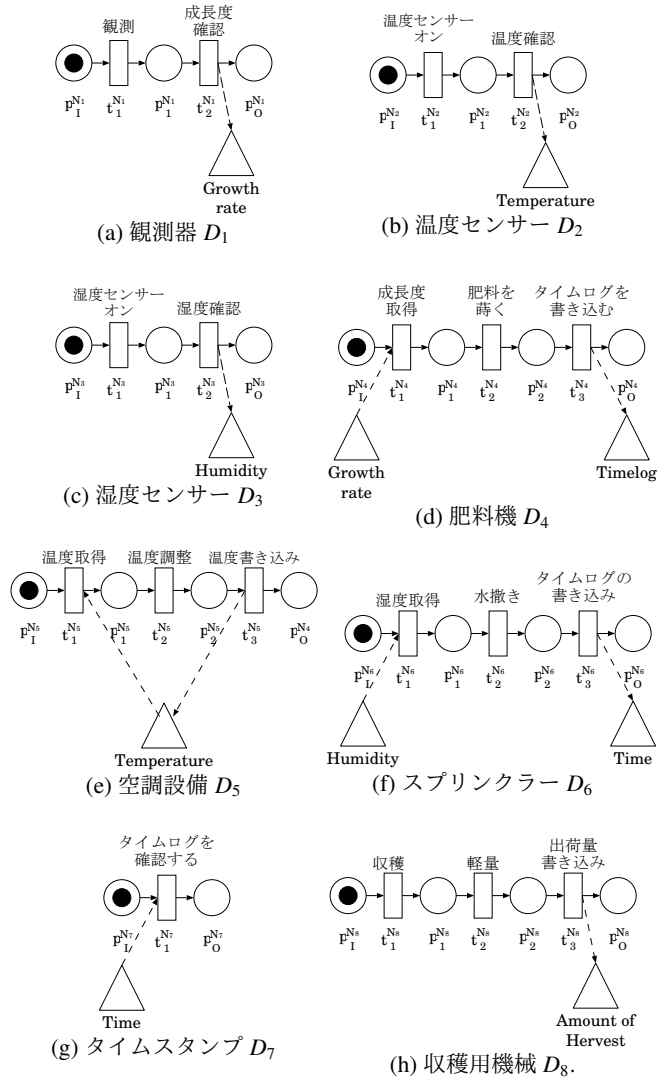

図 3: スマートファームファクトリーサービス内におけるデータ指向型のサービスモデル.

3.2 オントロジーをベースとした IoT 関係の記述

IoT ネットワークにおいて、デバイスの相互運用性は重要なものである。問題の 1 つとして異なるデバイス間のデータの解釈 (型や単位など) の違いがある。サービスのオーケストレーションを行うとき、初めに他のデバイスやサービスがデータについて解釈できる必要がある。本フレームワーク内で、我々は関係性のコンテキストを記述している。コンテキストの記述は IoT-Lite[7], [8] を用いている。これは、エンティティとサービスを表現するための軽量オントロジーをベースとしたリソース記述フレームワーク（RDF）のメタモデルである。筆者らはこれをオーケストレーションの補助として参照している。まず、サービスモデルとオントロジーの記述を以下のように定義する。

Vol.2017-DPS-172 No.26
Vol.2017-SPT-26 No.26
Vol.2017-EIP-78 No.26
2017/11/30

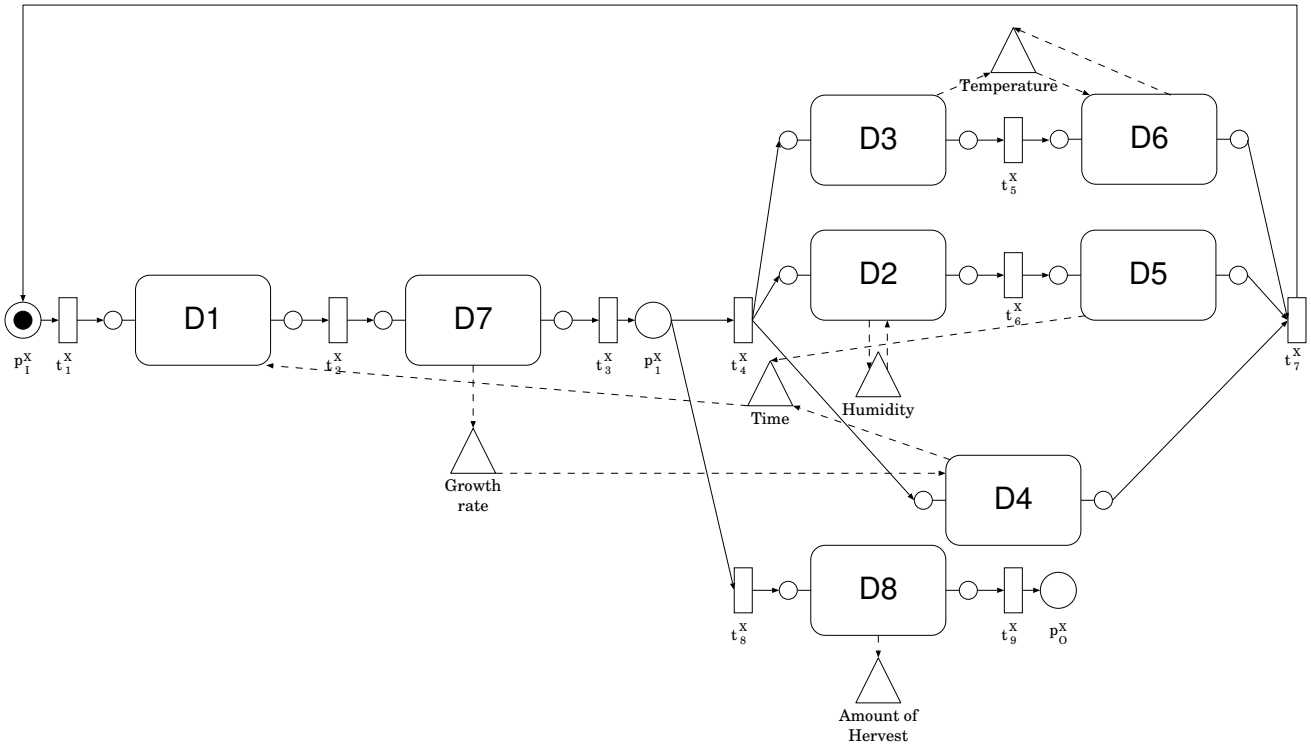

図 4: スマートファームファクトリーサービスのためのサービスの仕様 S_{spec}

定義 3 (サービスプロファイル):サービスモデル D とそのオントロジーの記述 δ の組 (D,δ) をサービスプロファイル D とする。 □

オントロジー記述 δ は、IoT 環境におけるエンティティの関係を記述する。 オントロジー δ は、サービスモデル D のエンティティとプロパティの関係の表現をサポートするための重要な記述である。筆者らは、図 6〜8 にデバイスのオントロジ記述の例を示す。 仕様 S_{spec} のオントロジーの一部を図 5 に示す。図 4 に示すサービス仕様 S_{spec} は、図 3 に示す $D_1, D_2, \cdots, D_8$ から調整する必要があるが、これらのデバイスは異なるメーカーによって製造されてもよいし、より新しいモデルに置き換えられていてもよい。 仕様が若干異なっている、もしくはデバイスによって提供されるサービス間で処理されるデータが仕様を満たさない場合も存在する。したがって、次の問題を定義できる。

定義 4 (データ指向型サービスのオーケストレーション問題):
入力：サービスプロファイル (D_1,δ_{D_1}),(D_2,δ_{D_2}),...,(D_n,δ_{D_n}), サービス仕様のプロファイル (S_{spec},δ_{spec}) に対し
出力： (S_{spec},δ_{spec}) に基づいて構成されたサービスモデル S_{DD}=$D_1\circ D_2\circ\cdots\circ D_n$ は S_{DD}=S_{spec} であるか？

定義 4 は入力としてサービスプロファイル (D_1,δ_{D_1}),(D_2,δ_{D_2}),...,(D_n,δ_{D_n}) とサービス仕様のプロファイル (S_{spec},δ_{spec}) が与えられる。また、出力として図 4 のように (S_{spec},δ_{spec}) に基づいて構成されたサービスモデル S_{DD}=$D_1\circ D_2\circ\cdots\circ D_n$ を出力する。ただし、これは S_{DD}=S_{spec} を満たさなくてはならない。$D_1\circ D_2\circ\cdots\circ D_n$ は S_{spec} に基づいて結合された DPN を表す。

4. データ指向型サービスオーケストレーション

定義 4 における与えられた問題を解決するために、筆者らはオントロジーを用いたオーケストレーションに焦点を当てる。(S_{spec},δ_{spec}) の仕様は設計上正しいと仮定する。 よって、 (S_{spec},δ_{spec}) に基づいて S_{DD}=$(D_1,\delta_{D_1})\circ(D_2,\delta_{D_2})\circ\cdots\circ(D_n,\delta_{D_n})$ を構成すればよい。このとき、デバイスのタイプと (S_{spec},δ_{spec}) で指定されたデータ型に注目する。

例えば、(S_{spec},δ_{spec}) が、温度センサ TEMPERATURE を必要とするサービスを指定している。この時、(i) QuantityKind が Temperature である。(ii) Unit が degree_celcius である。この二つを満たしていることが条件である。 次に、S_{DD}=S_{spec} を得るために、サービスのオントロジーの記述を比較する必要がある。

筆者らは、RDF を基にした SPARQL[9] を使用してオントロジ記述を比較するためのセマンティカルアプローチを行う。SPARQL は、RDF でフォーマットされたファイルに格納されたデータを操作および検索するために使用され

Vol.2017-DPS-172 No.26
Vol.2017-SPT-26 No.26
Vol.2017-EIP-78 No.26
2017/11/30

```
@prefix iot-lite:
    <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#>.
@prefix ssn:
    <http://purl.oclc.org/NET/ssnx/ssn#>.
@prefix sf:
    <http://<uri>/smart-farm#>.
@prefix m3-lite:
    <http://purl.org./iot/vocab/m3-lite3#>.
@prefix fiesta-res:
    <http://platform.fiesta-iot.eu/srd/registry/v1/>.
...
sf:SmartFarmFactory  a  ssn:System.
sf:smart-sm  a                  ssn:System;
        iot-lite:isSubSystemOf  sf:SmartFarm.
sf:service#iot-smart-farm
 a                  iot-lite:Service;
 iot-lite:endpoint  "http://<uri>/iot-temp".
 iot-lite:endpoint  "http://<uri>/iot-humid".
...
fiesta-res:IoT-TEMPERATURESensor
 a                          ssn:SensingDevice;
 iot-lite:hasQuantityKind   m3-lite:Temperature;
 iot-lite:hasUnit           m3-lite:degree_celcius
 iot-lite:isExposedBy       sf:service#iot-smart-farm;
 iot-lite:isSubSystemOf     sf:smart-f;
 ssn:onPlatform             sf:GreenHouse.
fiesta-res:IoT-HUMIDITYSensor
 a                          ssn:SensingDevice;
 iot-lite:hasQuantityKind   m3-lite:Humidity;
 iot-lite:hasUnit           m3-lite:percent
 iot-lite:isExposedBy       sf:service#iot-smart-farm;
 iot-lite:isSubSystemOf     sf:smart-f;
 ssn:onPlatform             sf:GreenHouse.
 fiesta-res:ClimateController
 a                          ssn:ActuatingDevice;
 iot-lite:hasQuantityKind   m3-lite:Voltage;
 iot-lite:hasUnit           m3-lite:Volt
 iot-lite:hasQuantityKind   m3-lite:Temperature;
 iot-lite:hasUnit           m3-lite:degree_celcius
 iot-lite:isExposedBy       sf:service#iot-smart-farm;
 iot-lite:isSubSystemOf     sf:smart-f;
 ssn:onPlatform             sf:GreenHouse.
...
```

図 5: IoT-Lite を用いたオントロジーの記述の一例、コンテキスト記述 δ_{spec}.

る。 各 RDF ファイルは、各デバイスの国際化リソース識別子（IRI）に格納されている。 SPARQL クエリの例を図 9 に示す。図 9 は、温度センサーの数値と単位データを取

```
fiesta-res:IoT-HUMIDITYSensor
 a                          ssn:SensingDevice;
 iot-lite:hasQuantityKind   m3-lite:Humidity;
 iot-lite:hasUnit           m3-lite:percent
```

図 6: 湿度センサーのオントロジー δ_{D_3}.

```
fiesta-res:IoT-TEMPERATURE
 a                          ssn:SensingDevice;
 iot-lite:hasQuantityKind   m3-lite:Temperature;
 iot-lite:hasUnit           m3-lite:kelvin
```

図 7: 温度センサーのオントロジー δ_{D_2}.

```
 fiesta-res:ClimateController
 a                          ssn:ActuatingDevice;
 iot-lite:hasQuantityKind   m3-lite:Voltage;
 iot-lite:hasUnit           m3-lite:Volt
 iot-lite:hasQuantityKind   m3-lite:Temperature;
 iot-lite:hasUnit           m3-lite:degree_celcius
```

図 8: 空調設備のオントロジー δ_{D_5}.

得するクエリを示している。 データを取得する温度と度合いは、各サービスモデルから取得した各データと比較される。Apache Jena の RDF API を使用して SPARQL クエリを実行できる。Apache Jena は、SPARQL を実行して RDF 形式を読み込んで操作する Java フレームワーク API である 。

オントロジーの記述に基づいたサービスオーケストレーションを提案する。ネットの構成は、S_{spec} のサブネットである各サービスモデルが S_{spec} に従って構成できることは明らかである。

《データ指向型サービスオーケストレーション》
入力 : サービスモデル $(D_1, \delta_{D_1}), (D_2, \delta_{D_2}), \cdots, (D_n, \delta_{D_n})$, サービスの仕様 $(S_{spec}, \delta_{spec})$
出力：IoT サービスモデル S_{DD} もしくは D_n のための違ったエンティティが必要なのではないか？

1° S_{spec} からペトリネット N_D、N_{spec} のみを生成するためにすべての変数ノード $v \in V_D$ と $u \in V_{spec}$ を取り除く。
2° 構成されたデバイスのサービスモデル $N_{D_1} \circ N_{D_2} \circ \cdots \circ N_{D_n} \to N_{DD}$. $N_{DD} \neq N_{spec}$ であるならば、不正なデバイスとしてエラーを出力し停止する。
3° *“SELECT ?fiesta-res WHERE ?iot-lite:hasUnit”* といったコンテキストの記述クエリーし、すべてのデバイス

Vol.2017-DPS-172 No.26
Vol.2017-SPT-26 No.26
Vol.2017-EIP-78 No.26
2017/11/30

```
PREFIX <http://purl.oclc.org/NET/UNIS/
       fiware/iot-lite#>.
SELECT ?iot-lite:hasQuantityKind
       ?iot-lite:hasUnit
FROM   <http://device.uri/iot-lite/smartFarm.rdf>
WHERE
  {
   ?fiesta-res:ClimateController
    a ssn:ActuatingDevice;
  }
```

図 9: SPARQL クエリーの例

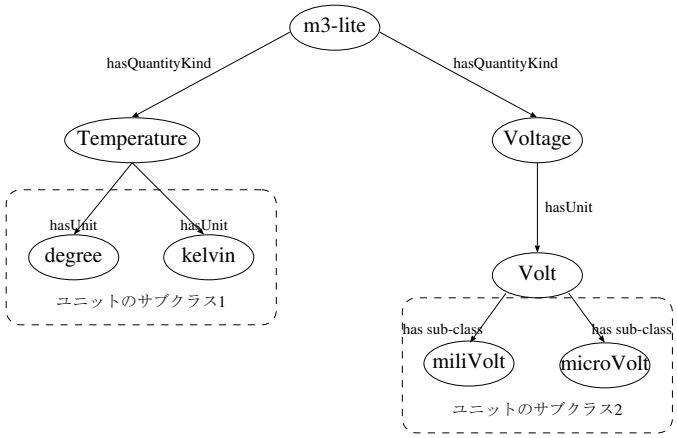

図 10: オントロジーのクラス図の一例

を含んだ D_{iot} を返す。

4° もしも *''iot-lite:hasUnit"* が S_{spec} と関連しているならば D_{iot} に基づいた N_{DD} 中の $v \in V_D$ を $t \in T_D$ へ繋ぎなおし、 S_{DD} の出力を行い停止する。

まずは、図 3 のような一連のデバイスのサービスモデルと各デバイスのオントロジー $(D_1, \delta_{D_1}), (D_2, \delta_{D_2}), \cdots, (D_8, \delta_{D_8})$ が与えられる。また、図 4 のように仕様 S_{spec} 必要がある。そのために、《データ指向型サービスオーケストレーション》を適用する。ステップ 1° では、$D_1, D_2, \cdots, D_8$ と S_{spec} からデータを取り除きペトリネット N_D、N_{spec} を生成する。ステップ 2° では構成したデバイスのサービスモデル $N_{D_1} \circ N_{D_2} \circ \cdots \circ N_{D_8} \rightarrow N_{DD}$ が得られる。 $N_{DD} \neq N_{spec}$ であるならば、不正なデバイスとしてエラーを出力し停止する。ステップ 3° では、図 9 のようなコンテキストの記述クエリーをし、すべてのデバイスを含んだ D_{iot} を返す。このクエリーでは"hasQuantityKind" や"hasUnit" に注目する。ステップ 4° では、もしも *''iot-lite:hasUnit"* が適合しているならば、D_{iot} に基づいた N_{DD} 中の変数ノードを結合し、S_{DD} の出力を行い停止する。

上記の手順では、作成されたモデル S_{DD} が S_{spec} と等しくなるように、S_D のデバイスサービスモデルを構成する。最初にデータを考慮しないプロセスフローの接続を行う。この後にコンテキストオントロジーに基づいて各変数同士を接続する。この時、デバイスの持つ QuantityKind や Unit を比較する。この時使用されるオントロジーのクラス図の一例を図 10 に示す。図 10 のように各オントロジー記述内（今回は m3-lite[10]）にはサブクラスなどが設定されている。hasQuantityKind:Temperature であればその Unit のサブクラスとして摂氏度 degree_celcius とケルビン温度 kelvin を記述してある。本来、データとして返される値の Unit が違うため任意の結果が得られない、もしくは機器の故障などのトラブルに発展する可能性がある。この時、オントロジーの記述を行っているためそれぞれのユニットに対する互換性をとることができるようになる。hasQuantityKind:Voltage についても同様だが、この場合は Unit そのものにサブクラスが存在している。これらのオントロジーの記述により各データの柔軟性を高め、やり取りするデータの性質をより容易に合わせることができるようになる。この一連の流れの後、要求していた仕様 S_{DD} を得ることができる。

5. おわりに

本稿では、コンテキストオントロジーとデータペトリネットを用いたサービスのオーケストレーション手段について説明した。オントロジーについても QuantityKind を比較する際の許容範囲の設定やユーザー定義のオントロジーの適用範囲などいくつかの問題が存在している。
今後の課題として、デバイスに対するオントロジーの自動的な記述や環境そのものに対する結合条件などについての検討を行っていく。

謝辞 本研究の一部は（株）インタフェース社の援助を受けた。

参考文献

[1] N.K. Lee, H.W. Lee, W. Ryu, “Considerations for web of object service architecture on IoT environment,” Int. Journal of Smart Home, vol.9, no.1, pp. 195–202, 2015.

[2] M. A. B. Ahmadon, S. Yamaguchi, “On service orchestration of cyber physical system and its verification based on Petri net,” 2016 IEEE 5th Global Conference on Consumer Electronics, Kyoto, pp. 1–4, 2016.

[3] M. de Leoni, W.M.P. van der Aalst, “Data-aware process mining : discovering decisions in processes using alignments,” Proc. of the 28th Annual ACM Symposium on Applied Computing, SAC ’13, Coimbra, Portugal, pp.1454–1461, 2013.

[4] M. A. B. Ahmadon, ”Elgar Framework: Context-Aware Service Orchestration with Data Petri Net,” 2017 IEEE 6th Global Conference on Consumer Electronics, Nagoya,

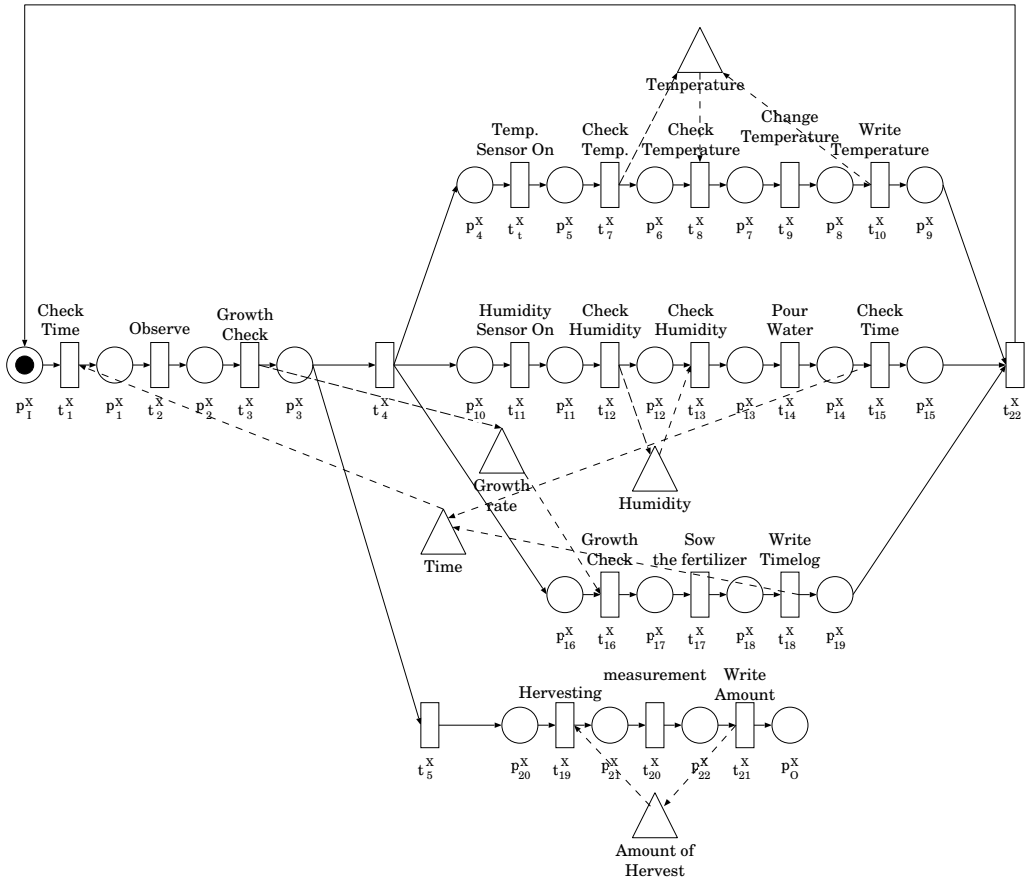

図 11: オーケストレーション結果

pp.812–815, 2017.

[5] Shingo Yamaguchi, Mohd Anuaruddin Bin Ahmadon, Qi-Wei Ge, ‘’Introduction of Petri Nets: Its Applications and Security Challenges,” in Handbook of Research on Modern Cryptographic Solutions for Computer and Cyber Security, Hershey, PA, USA, chapter 7, pp.145–179, 2016.

[6] T. Murata,“Petri nets: Properties, analysis and applications,” Proc. of the IEEE, vol.77, no.4, pp.541–580, 1989.

[7] M. Bermudez-Edo, T. Elsaleh, P. Barnaghi and K. Taylor, “IoT-Lite: A Lightweight Semantic Model for the Internet of Things,” 2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress, Toulouse, pp. 90–97, 2016.

[8] IoT-Lite Ontology - World Wide Web Consortium（オンライン）, <https://www.w3.org/Submission/iot-lite/>

[9] D. Boldt, H. Hasemann, M. Karnstedt, A. Kreller and C. v. d. Weth, “SPARQL for Networks of Embedded Systems,” 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), Singapore, pp. 93–100, 2015.

[10] m3lite - Linked Open Vocabularies（オンライン）, <http://lov.okfn.org/dataset/lov/vocabs/m3lite>