修士論文

スマートホームにおける
データカタログを用いたデータ自動選択システムに関する研究

XIN Tao

主指導教員　丹 康雄 教授

北陸先端科学技術大学院大学
先端科学技術研究科
（情報科学）

令和 2 年 9 月

Abstract

Following the popularization of the connected device and home appliance, the market of smart home service is changing to an open mode from the mode that all develop by device makers. It will produce many of service developers to acquire data from device makers, and develop services and provide to end-users. But device makers do not have a uniform data description model and a uniform interface. Furthermore, the number of data will be billions, lead to discover the useful data by people is difficult. Therefore, in this research, we proposed a system that can use the data catalog to get data from heterogeneous architecture device clouds, provide by machine- readable interface and human-readable interface. And the system has a data auto-select mechanism to solve the problem that people difficult to discover useful data efficiently from billions of data. And validate the proposed system is effective.

概要

IoT 家電デバイスの急激な普及により，身周りに生成しているデータが多くなり，数多くのデバイス向けにサービス提供とデータ収集のため，各メーカーが自社のデバイスクラウドを構築する状況に至っている．その数多く且つ構成の異なるデバイスクラウドに収集されているデータは統一な記述方式や提供方法がなく，メーカーを越えたデータの流通が難しくなっていた．これに対し，RESTfulAPI を用いた集中型データ連携プラットフォームが典型的なプラットフォームとして立ち上げられ，IoT データ取引市場が形成されづつ，情報流通を始めつつある．

しかし，データの検索と選択は人間が行うことが想定されるため複数のクラウドからの膨大なデータ量とデータ提供要件変化の高い発生頻度により，有効に利活用することが難しい．これに対し，各メーカーの異なる構成のデバイスクラウドからのデータが利用でき，使用者による操作を必要とせず，デバイスによるデータ提供要件に変化を自動的に対応ができるシステムが求められている．

本研究では，従来のデータ選択の不便を補い，ヒューマンリーダブルインタフェースとマシンリーダブルインタフェース両方持つデータ自動選択システムを提案する．また，リソースを記述するために，データ取引標準モデルのなりつつある JEITA スマートホームデータカタログを用いたマシンリーダブルな JEITA スマートホームデータカタログオントロジーを作成し，システムに実装する．マシンリーダブルインタフェースと近似度計算に基づいた自動選択によってデータ利用者側の操作が簡易になり，データ連携プラットフォームのデータ選択方式の欠点を補うことができる．

システムの実装と評価により，提案システムはマシンリーダブルインタフェースを用い，自動且つ連続的な情報取得ができる．また，人間手動でデータ選択と比べ，９９％以上の時間を削減できる．

目次

図目次

表目次

第 1 章

はじめに

本章では，本研究の背景，目的，本文の構成について述べる．

1.1　研究背景

1980 年代末に M.Weiser らによって提唱されたユビキタスコンピューティングは，その後，2005 年ごろから言われ始めた Web2.0 から BigData への流れとつながり，2015 年頃からは IoT: Internet of Things と呼ばれるようになり，発展を続けている．近年の IoT デバイスの低価格化により，急激な普及がすすんでおり，2025 年に 416 億の IoT デバイスが 79.4ZB のデータを生成すると予測されている [1]．その数多くのデバイス向けにサービス提供とデータ収集のため，各メーカが自社のデバイスクラウドを構築する状況に至っている．

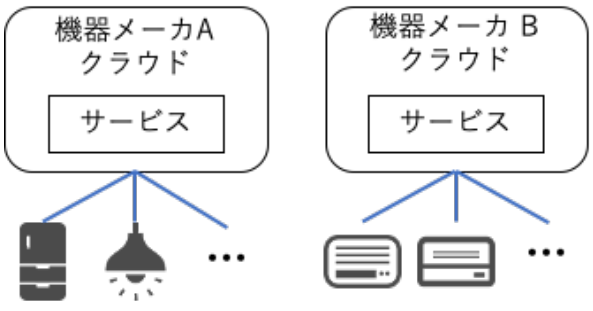


図 1.1　スマートホームにおける従来のサービス提供方式

図 1.1 のように、各機器メーカの独自発展してきたクラウドは該当メーカのデバイスと繋がり、自社のサービスが設置されており、提供する．そのうえ、2019 年 10 月から行わ

れていた「Life up プロモーション」[2] は実ユーザと契約することにより，家電機器の利用に関する実データの収集と利用ができ，新しいサービスの創出や既存サービスの改善など様々な用途でデータ利活用を行う．これにより，生活データの活用が始められている．

サービスの品質を向上させ、種類を豊富させるために、第三者のサービス提供事業者を参加させ、サービスの開発、提供、保守を行う．図 1.2 はその様子を示す．機器メーカクラウドとサービス提供事業者が複数存在しており、一つのサービス提供事業者が複数の機器メーカクラウドにサービスを提供することができ、一つの機器メーカクラウドが複数のサービス提供事業者にデータ提供ができる．

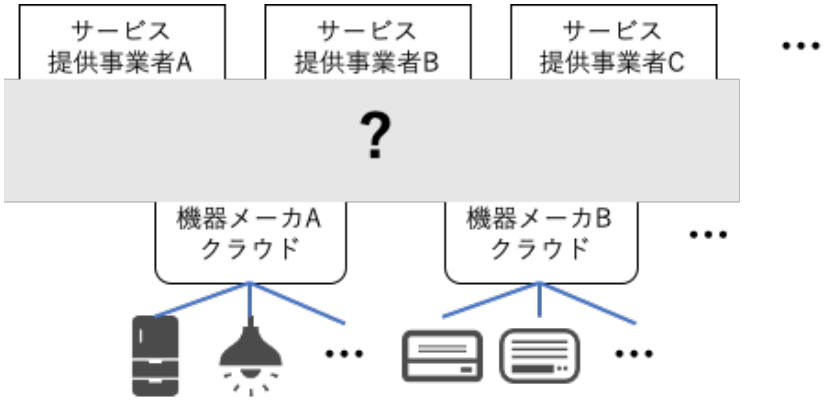


図 1.2　スマートホームにおけるのこれからなり得るサービス提供方式

とはいえ、その数多く且つ構成の異なるデバイスクラウドに収集されているデータは統一な記述方式や提供方法がなく，サービス提供事業者からのデータの発見と取得が難しくなっていた．これに対し，IoT デバイスのデータを流通させるために，RESTfulAPI を用いた集中型データ連携プラットフォームが立ち上げられ，IoT データ取引市場として情報流通を始めつつある．

こうした状況を受け，各社のデバイスクラウドにあるセンシングデータやデバイスデータを活用するために，スマートホームデータカタログが JEITA(電子情報技術産業協会) によって提案されている [3]．これにより，メタデータを説明する際の記載方法の標準化が可能になり [4]，メタデータを用いたクラウド上データの検索やクラウド間のサービス連携を行うことが今後可能になると考えられる．

しかし，現状，データの検索と選択は人間が行うことが想定されており，複数のクラウドからの人間が選択できないデータ量且つ利用中データセットより適合なデータセットの出現を発見できないことにより，有効に利活用することが難しい．これに対し，各メーカーの異なる構成のデバイスクラウドからのデータが利用でき，使用者による操作を必

要とせず，流通中のデータセットの変化を自動的に対応ができるシステムが求められている．

1.2 研究目的

本研究では，現状を踏まえ，各メーカーのデバイスクラウドをベースとしたデータ流通プラットフォームを提案する．各メーカーの既存クラウドによって分散型カタログデータベースが形成され，共通的なデータ記述方式とマシンリーダブルインタフェースを用いて検索を行う．こうしたデータ流通プラットフォームにより，デバイスの詳細情報を利用者に読ませる必要がなく，異なる構成のデバイスクラウドによるデータ連携ができる．

また，データ連携プラットフォームからデータを手動で選択するしかないという欠点を補い，より適合な新規データセットの発見と利用に自動的に対応ができるメカニズムを提案する．利用者はヒューマンリーダブルインタフェースを通じてニーズを記述する．システムは利用者のニーズ記述を受け，自動選択と切り替えメカニズムを用いて連続的な情報提供を行う．こうしたメカニズムによってデータ利用者側の操作が簡易になり，データ連携プラットフォームのデータ選択方式の欠点を補うことができる．

提案プラットフォームにより，データ取得とデータセットの情報が変化した際に人間の作業をシステムで実行することができ，サービス提供事業者のデータ取得が簡易になることで，新サービスの創出やサービスの改善に支援し，更にスマートな生活に貢献できる．

1.3 本文の構成

本文は，本章を含め，６章で構成される．章毎の概要は以下のように示す．

- 第１章
  - 本研究の背景と目的を述べ，本論文の構成を説明する．
- 第２章
  - 情報モデルに関する調査を行い，セマンティク技術を用いたプラットフォーム間相互操作における情報モデルのレイヤ構造を検討する．
- 第３章
  - 前章に調査したモデル中の典型的なモデルと既存のデータ連携流通プラットフォームについて検討をする．
- 第４章

  - 提案システムの位置付けからシステム内部の動き，リソース記述モデル，ニーズ記述モデル，アルゴリズムについて説明をする．
- 第５章
  - システムの実装について説明を行う．
- 第６章
  - システムの動作確認とその結果，または提案システム自動選択機能の効果評価を行う．
- 第７章
  - 全文をまとめ，今後の課題について検討する．

第 2 章

情報モデルに関する調査

2.1　調査の目的

IoT の社会実装が進展するにつれ，様々な機器やサービスを組み合わせて利用する実例が増加しており，データ形式を共通化する重要性が急速に高まっている．

従来，このような要求はサービス (アプリケーション) 分野ごとに生じており，それぞれの業界内でデータ形式を共通化しようとする試みは多数行われていたが，本格的な IoT 時代を迎え，分野横断の連携も視野に入り，より一般論としての共通化が求められるようになってきた．

本調査はこうした状況をふまえ，今後，もともとは異分野で使われてきたデバイスやマイクロサービスが連携して新たな分野の要求を満たすような状況を実現するための，データモデルの共通化に向けた現状調査とその整理を行うものである．

次の節では，今までに取り組まれてきたデータモデルやオントロジに関する取組みを調査し，その概要をまとめる．続く第三節ではこれらのモデルの間の関係について言及する．

2.2　調査内容

2.2.1　概要

本調査では下記のモデル/プラットフォーム/規格について調査を行なった．

- oneM2M
- universAAL

- OCF
- SASREF
- NGSI-LD
- W3C WoT-TD
- BIG IoT
- FIESTA-IoT
- W3C SSN/SOSA
- M3/M3-lite
- OMA LWM2M(+IPSO)
- ECHONET lite

各モデルの標準化状況，シンタックス，主な記述内容について表 2.1 に示す．
各モデルの概要を次のセクション以降に記載する．

組織&モデル名	標準	標準になっている部分	シンタックス	記述中心	定義
oneM2M	ITU-T ETSI	全部	OWL/XML	Device	システムが識別可能な エンティティー
universAAL	IEC	アーキテクチャ	OWL/XML	PhysicalThing	物理的なもの
NGSI-LD	ETSI	全部	JSON-LD	Entity	実世界に存在する物理 あるいは概念的な物事
ETSI-SAREF	ETSI	全部	OWL/XML	Device	具体的なデバイス
OCF	ISO/IECJTC1	フレームワーク	OAS(SWAGGER) JSON Schema	(Device)	oneIoTA に登録した デバイスモデル
ECHONET lite	IEC ISO/IECJTC1			(Device)	具体的なデバイス
W3C SOSA/SSN	OGC/W3C	全部	OWL/XML	Sensor Actuator Sampler	実デバイスやエージェント
W3C WoT	W3C	全部	JSON Schema	Thing JSON-LD	フィジカルやバーチャル エンティティー
OMA LWM2M(+IPSO)	OMA			(Device)	具体的なデバイス
FIESTA-IoT ontology			OWL/XML	(oldssn:)Device	実デバイスやエージェント
BIG IoT information model			OWL/XML	Offering	提供できるサービスと アクセス
M3/M3-lite			OWL/XML	(oldssn:)Device	実デバイスやエージェント

表 2.1　概要一覧

2.2.2　oneM2M

組織，標準

標準化機構	コード
ITU	ITU-T Y.oneMeM(ITU-T Y.4500)
ETSI	TS 118
TTC	TS-M2M
TTA	TTAT.MM
ATIS	ATIS.oneM2M
TSDSI	TSDSI STD T1.oneM2M

表 2.2　oneM2M 関連標準

oneM2M は ITU の国際標準である．
oneM2M に参加している各国の標準化団体により oneM2M は標準化された．

目的

ネットワーク中のデバイスを登録，発見，リモート制御することを基本とする．
セマンティック相互運用性を有する．
他のオントロジーにマッピングされるための最小オントロジーを有する．
この最小オントロジーでは最小限の数の概念，関係，制限などが規定される．

情報モデル

ここでは oneM2M プラットフォームの基本オントロジー oneM2M Base Ontology(TS 0012) について説明する．

■モデルの特徴

- Device 中心の記述．
- Service と Function でデバイスとのインタラクションを記述する．
- 操作に関する記述は RESTful ベース．

■定義した概念 (図 2.1)

- **Thing**: oneM2M システムで識別可能なエンティティ．

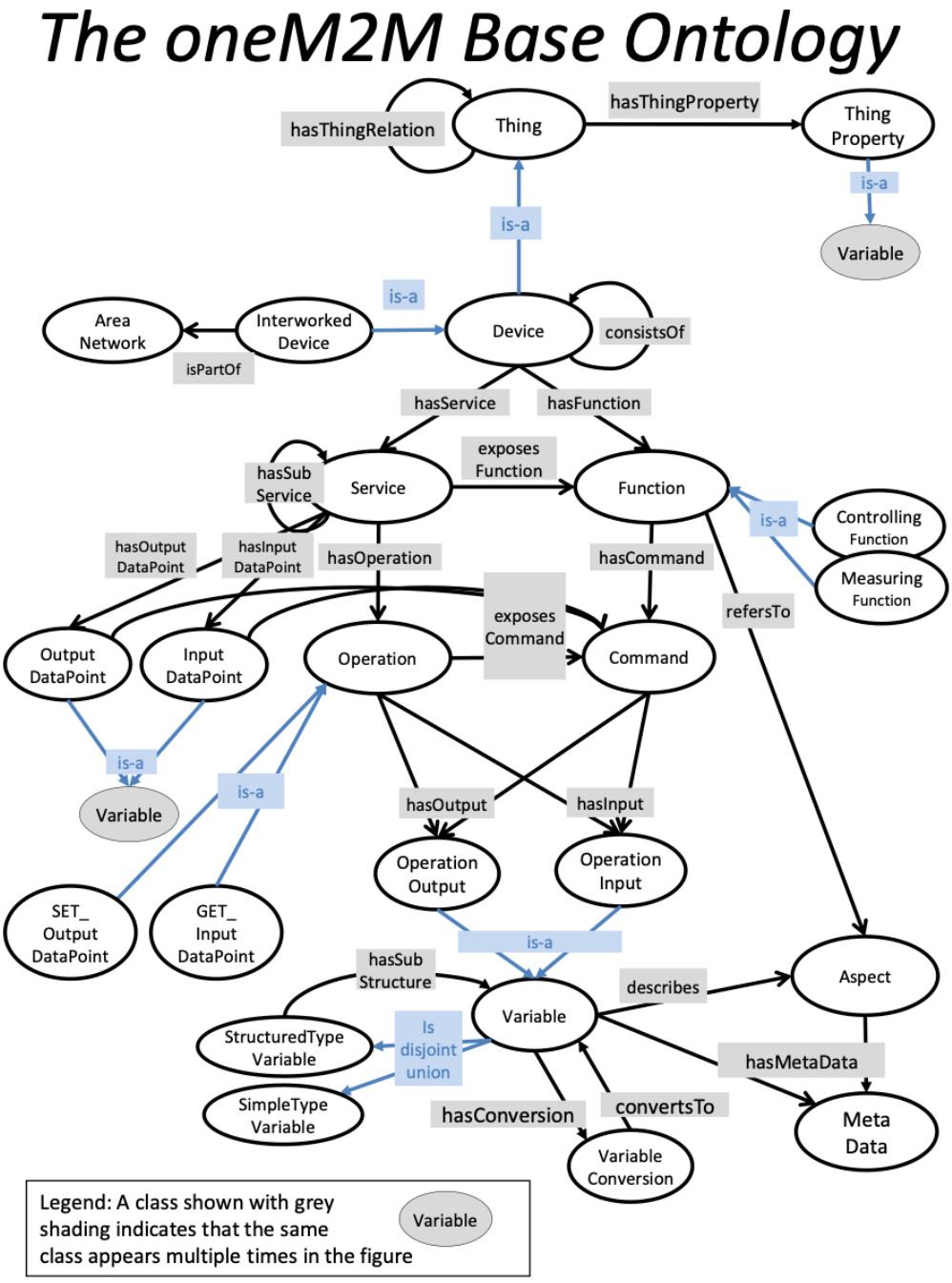


図 2.1　oneM2M Base Ontology[5]

- **ThingProperty**: Thing の属性，システムに検索，更新されることが可能.
- **Variable**: ThingProperty, OperationInput, OperationOutput, InputDataPoint, Output-

DataPoint，OutputDataPoint のスーパークラス．メンバークラスは時間に伴って変化しているデータを持つエンティティ．データは実世界のある属性を記述する．
  - **SimpleTypeVariable**: Variable のサブクラス．単純な XML タイプ変数を含む.
  - **StructuredTypeVariable**: Variable のサブクラス．構造化変数を記述する．他の変数型のセットで構成される．
- **VariableConversion**: ある変数値の範囲から別の変数値の範囲への変換ルールを記述する．
- **MetaData**: Thing の値あるいはある側面のデータ．
- **Device**: Thing のサブクラス．Function を実行し，タスクを完成するもの．サブデバイスを持つことができる．Device はアドレス可能 (addressable).
- **Function**: Device の設計されるタスクを完成するための機能．ヒューマンリーダブルに「デバイスは何をする」が記述.
  - **ControllingFunction**: Function のサブクラス．制御と影響する実世界の側面.
  - **MeasuringFunction**: Function のサブクラス．計測と検知する実世界の側面.
- **Aspect**: 実世界のある側面を記述する．物理的または非物理的なエンティティあるいは品質.
- **Command**: Function の動作をサポートする操作を表示する．オペレーションがネットワークに出すコマンド.
- **Service**: ネットワーク中の Function を表示する．ネットワーク内の Function を発見可能，登録可能，リモート制御可能にする.
- **OutputDataPoin**: Service の Variable，RESTful Device が設定する．Device は自動的に OutputDataPoin の値を更新する.
- **InputDataPoint**: Service の Variable，RESTful Device が設定する．Device は自動的に InputDataPoint を読み取る.
- **Operation**: 他のデバイスとのデータ交換.
  - **GET_InputDataPoint**: Operation のサブクラス．InputDataPoint のデータを取得するために，Device によって提供される操作.
  - **SET_OutputDataPoint**: Operation のサブクラス．OutputDataPoint のデータ更新をトリガーするために，Device によって提供される操作.
- **OperationInput**: デバイスのサービスへの Operation の入力のタイプを記述する．OperationInput は入力の全ての可能な値を表す．全ての可能な値:データ型と範囲または列挙されたインスタンスのリスト.
- **OperationOutput**: デバイスのサービスへの Operation の出力のタイプを記述する．OperationInput は出力の全ての可能な値を表す.

- **Area Network**: 物理属性，通信プロトコル，プロファイル
- **Interworked Device**: Area Network の一部.

2.2.3 universAAL

組織，標準

アーキテクチャは IEC 国際標準 IEC PAS 62883 となっている.

関連プロジェクト:

- ドイツ EMBASSI と DynAmITE プロジェクト (1999-2006)
- EU FP6 PERSONA と FP7 universAAL プロジェクト (2007-2013)
- EU FP7 CIP priject ReAAL(2013-2016)

目的

AAL システムのサービス間における共通モデルを与える.

情報モデル

基本オントロジーがない，各規格の uAAL モデルを導入して使う [6].

この部分は基本情報モデルと三つの制御バスについて説明する.

■特徴

- 基本モデルは PhysicalThing 中心で記述する.
- 基本モデルの中に PhysicalThing，Device，Resource しか定義していない，モデルによって具体的な記述は違う.
- 受信と発信はミドルウェアが行うので，モデルのなかに関連記述がない.

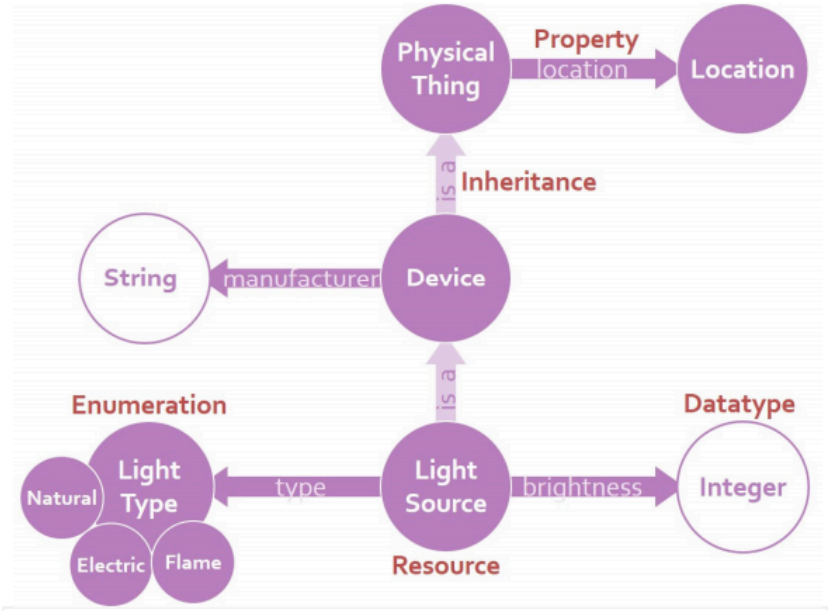


図 2.2　universAAL モデル [7]

■uAAL オントロジーの基本概念 (図 2.2)

- Resource:概念の表示方法，メッシュのなかのノード.
- Property:概念の間のリンク.
- Datatype:Boolean，Integer などのような基本データフォーマット.
- Enumeration:Resource インスタンスの集合.

■バス

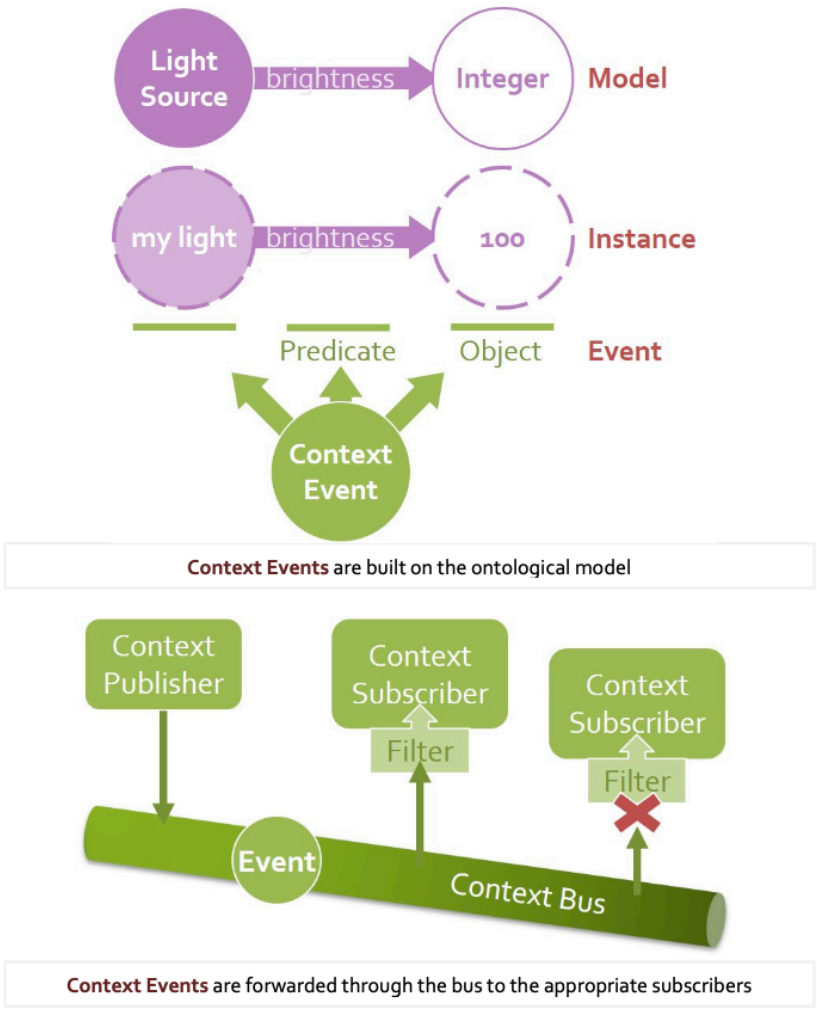


図 2.3　uAAL の Context バス [7]

- **Context バス** (図 2.3):
  コンテキストコミュニケーションはイベントベースである
  - Context Event: コンテキストバスで適当な加入者に届くコンテキスト情報.
  - Context Publisher: Context Event を送り出すプログラム.
  - Context Subscriber: フィルタで興味あるイベントを選択し，受け取るプログラム.

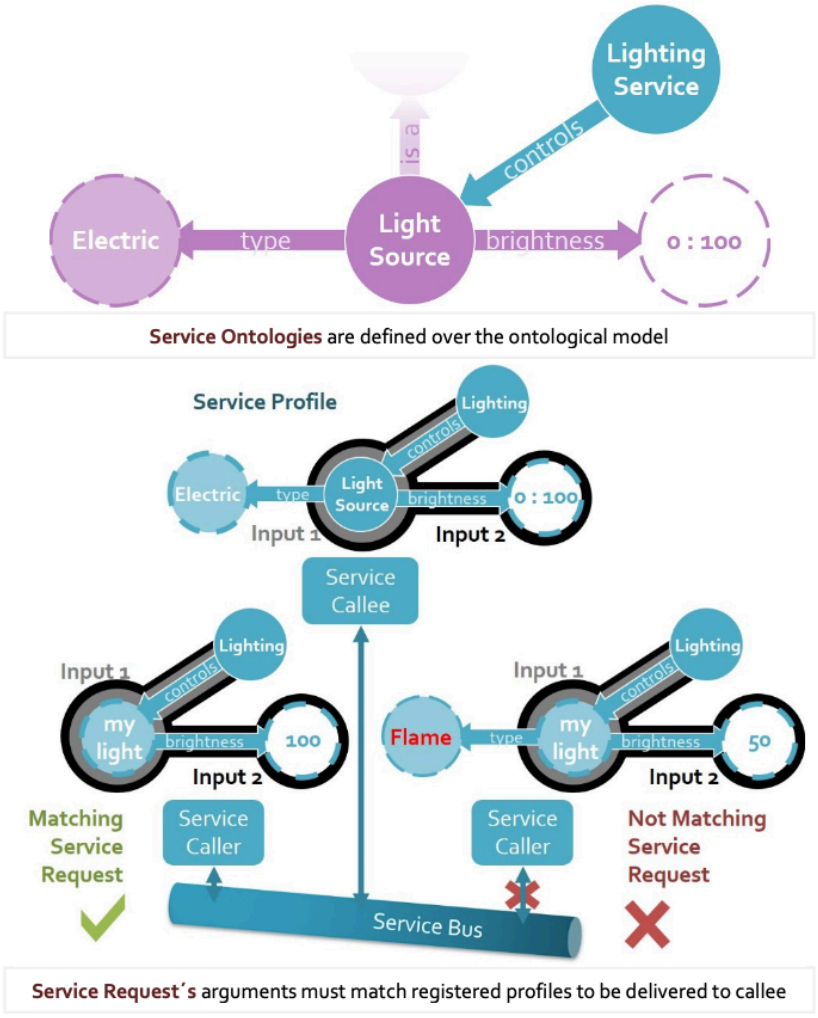


図 2.4　uAAL の Service バス [7]

- **Service バス** (図 2.4):
  - Service Ontology:要求者と提供者の間の共用モデル.
  - Service Callee:Service Ontology を提供するサービス．Service Profile で実現.
  - Service Profile:メソッドと同様，実行する操作を表す.
  - Service Caller:サービス実行を要求するプログラム．Service Request で実現.
  - Service Request: Service Profile の対応，Service Caller が実行する操作を宣言する.

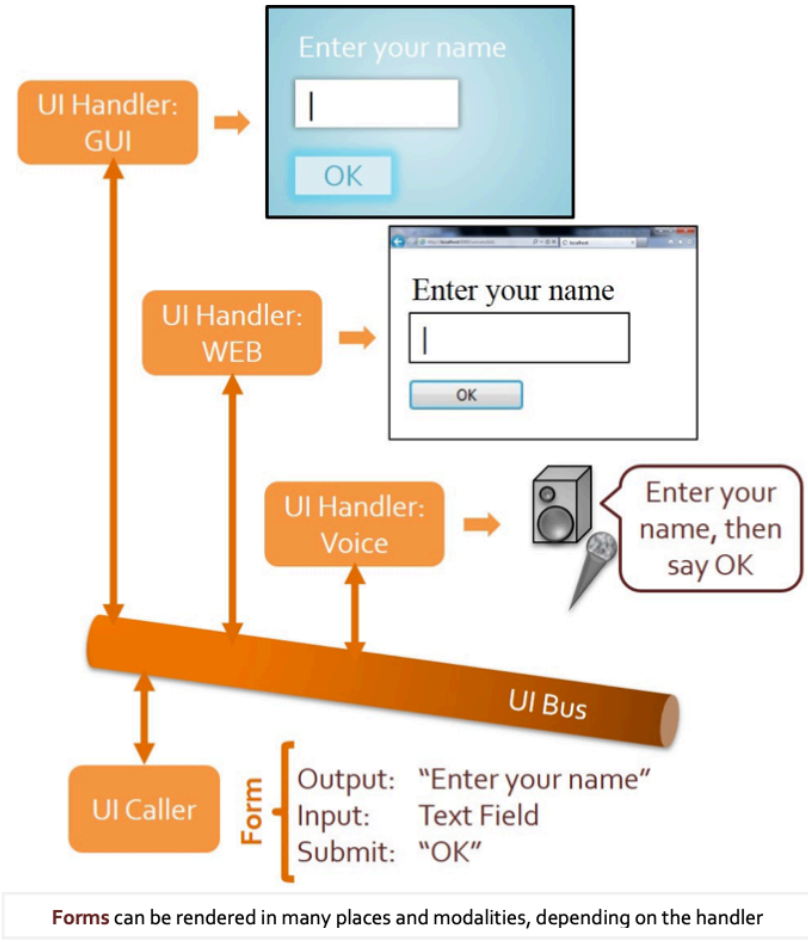


図 2.5　uAAL の UI バス [7]

- **UI バス** (図 2.5):
  UI Caller は Form を作成し，UI Handler に送り，ユーザに表示する．
  ユーザの操作が終わると UI Callers に送り，処理する．
  - UI Caller: ユーザと直接インタラクションをするプログラム．
  - Form: ユーザインタラクションコンポーネント (文字入力，選択肢，ボタンなど) のテキスト記述．
  - UI Handler: UI Caller 送った Form を GUI，音声出力，Web 画面などに変換するプログラム．

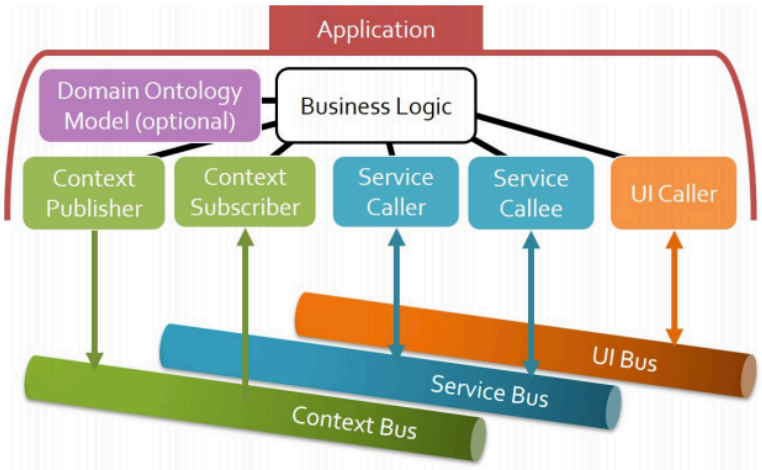


図 2.6　universAAL のバス [7]

- **アプリケーションとマネージャ** (図 2.6):
  - ビジネスロジックなどの構造以外は「uAAL wrappers」が必要
  - uAAL wrappers が含むコンポーネント:
    - Context Publisher
    - Context Subscriber
    - Service Caller
    - Service Callee
    - User Interaction Caller
  - アプリケーションインフォメーションドメインの記述手法はオントロジー.
  - アプリケーションは自分のオントロジーを定義することが可能，既存オントロジーの再利用が推奨される.
  - ワークフローやルールのスクリプトもアプリケーションになり得る.

2.2.4　OCF

組織，標準

OCF: **O**pen **C**onnectivity **F**oundation

フレームワーク (図 2.7 緑の部分) は ISO/IEC JTC1 国際標準 (ISO/IEC 30118)

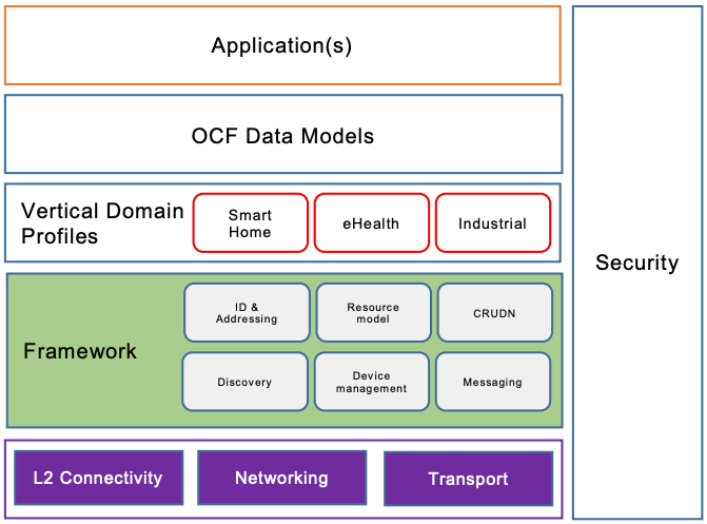


図 2.7　OCF 機能ブロック図 [8]

目的

異なるフレームワーク間の共通モデルを与える.

デバイスレベルの規格ので，特定の何種類デバイスにサービスを提供する.

情報モデル

オントロジーがない，oneIoTA というモデルハブで具体的なデバイスモデルを提供する.

■モデルの特徴

- デバイスレベルの規格.
- oneIoTA モデルは各センサ，デバイスの実例まで定義した.
- デバイスモデルのシンタックスは Open API Specification の swagger に従うので，RESTful で 操作する.

■モデル

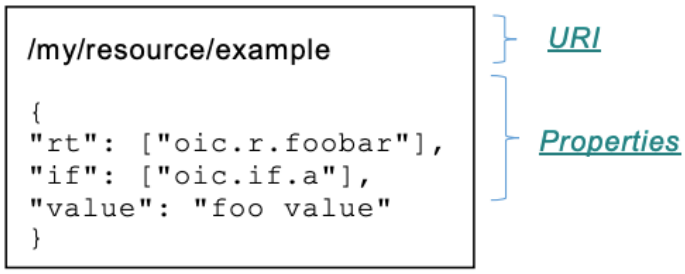


図 2.8　OCF リソースモデルの例 [8]

- **OCF リソースモデル** (図 2.8): URI と属性の集合.
  リソースモデルとコモンデータモデル一緒に OCF の基本相互操作性を提供する.
  ネットワーク上の制御と可視化をするために必要な物理デバイスやデバイス上のソフトウェアを記述可能.
  - **Common Properties**:
    - **rt**:Resource Type. リソースのタイプを定義する.
    - **if**:Resource Interface. リソース関連のインタフェースのリスト.
    - **n**:Name. 分かりやすい名前.
    - **id**:Resource Identity. ユニークなインスタンス識別子.
- **OCF Data Model(oneIoTa Model)**[8]: Bridge で他のデータモデルに変換する.
  外部モデルにマッピングする Derived Model は外部組織から提出し，OCF がモデルを審査する.
  マッピングできるモデルは一つずつ対応するマッピング記述がある.
  - **フォーマット**:
    - OIC(Open Internet Consortium)
    - OAS(Open Api Specification)
  - **三種類のモデル**:
    - **Native Models**: スイッチ，温度計，センサのような基本資源. それを使ってデバイスを組み合わせる.
    - **External Models**: 外部モデル. OCF はそのモデルを直接使わず，Derived Model でマッピングする.
    - **Derived Models**: 派生モデル. External Model と Native Model のマッピング情報を記述する.
  - **ブリッジできるモデル**:

* AllJoyn Interface [9]
* BLE [10]
* OneM2M Module [11]
* UPlus [12]
* Zigbee Cluster [13]
* Z-Wave [14]
* EnOcean [15]

2.2.5 SAREF

組織，標準

SAREF:Smart Appliance REFerence [16][16]

ETSI 標準 ETSI TS-103-264

ETSI SmartM2M Ontology

目的

マシン間のインタラクションを記述し，モデル間マッピングの中間モデルを与える．(図 2.9)

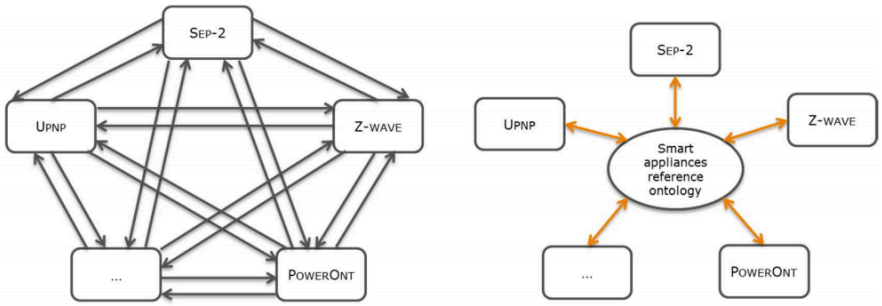


図 2.9 SAREF の位置付け [16]

情報モデル

ETSI-SAREF は領域抽象モデルであり，特定のプラットフォームに縛られない．

■モデルの特徴

- Device 中心の記述.
- Service と Function でデバイスとのイントラクションを記述する.
- 20 個以上の規格の共通概念とパタンからまとめられた.

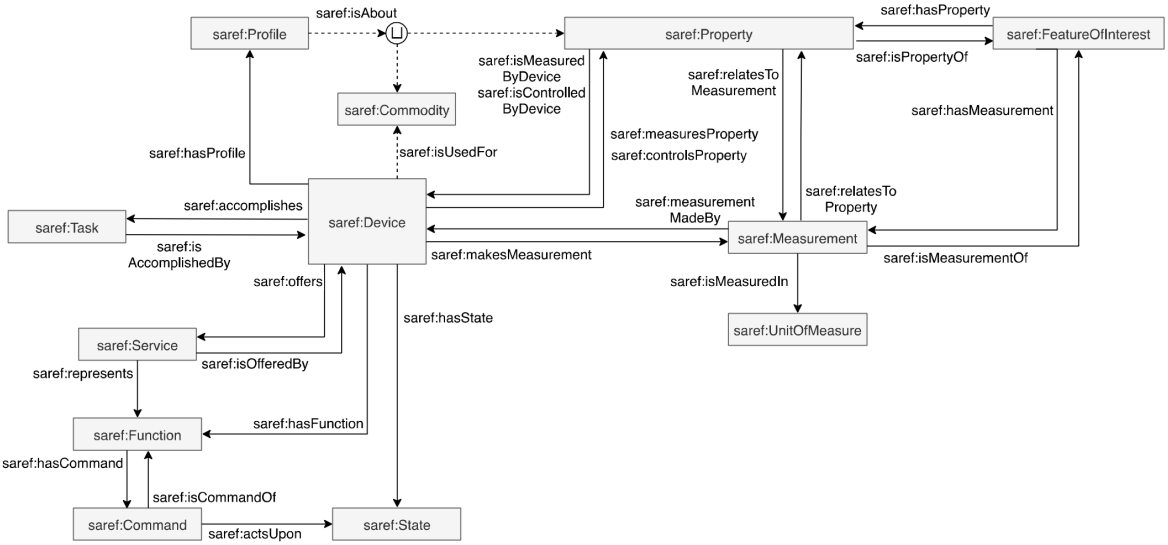


図 2.10　SAREF オントロジー [16]

■SAREF オントロジー (図 2.10) 定義した概念

- **Command**:デバイスが実行できる機能
  (例えば:OnCommand, OffCommand, PauseCommand, GetCommand, NotifyCommand, SetLevelCommand)
- **Commodity**
  (例えば:Electricity, Gas, Water)
- **Device**
  (例えば:Switch, Meter, Sensor)
- **FeatureOfInterest**
- **Function**
  (Actuating Function, Event Function, Metering Function, Sensing Function)
- **Measurement**:プロパティーの測定値
- **Profile**
- **Property**

(例えば:Energy, Humidity, Light, Motion, Occupancy, Power, Pressure, Price, Smoke, Temperature, Time)

- **Service**:ネットワーク機能
  (例えば:Switch On Service)
- **State**
  (例えば:On Off State, Open Close State, Start Stop State, Multi Level State)
- **Task**
  (例えば:Cleaning, Comfort, Lighting, Safety, Entertainment, Energy Efficiency)
- **UnitOfMeasure**:Measurement の単位
  (例えば:Currency, Energy Unit, Power Unit, Temperature Unit).

■メイン内容

- デバイスの能力
- デバイスの機能とそのネットワーク操作

■引用モデル

- 20 個の領域特定モデルを参考した (例: W3C ® SSN ontology, Echonet, EnOcean ® , SEP2, UPnP ®など)
- W3C ® Time
- W3C ® WGS84
- Ontology of units of Measure (OM) 測定単位のオントロジーのインスタンス

■oneM2M Base Ontology にマッピング (図 2.11) 記述パタンがほぼ一緒のため，マッピングしやすい.

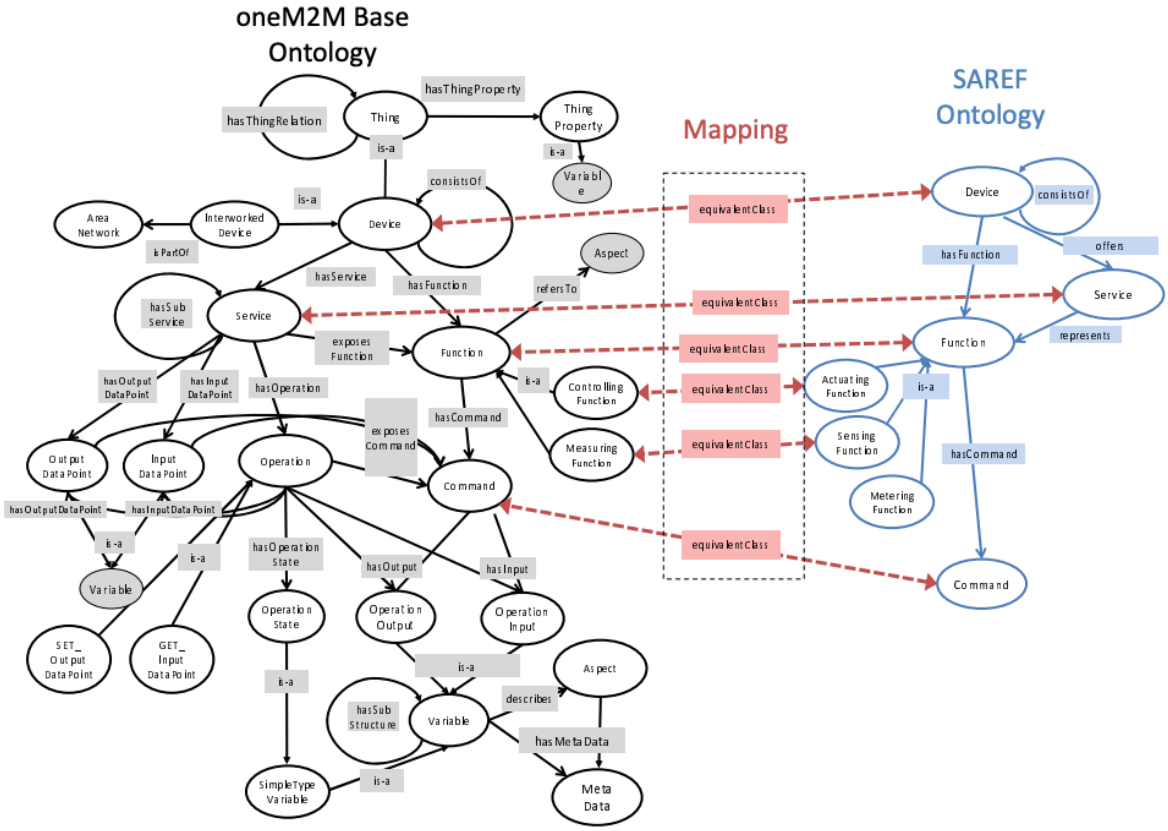


図 2.11　SAREF を oneM2M にマッピング [16]

SAREF	Mapping	oneM2M
saref:Device	owl:equivalentClass	oneM2M:Device
saref:Service	owl:equivalentClass	oneM2M:Service
saref:Function	owl:equivalentClass	oneM2M:Function
saref:SensingFunction	owl:equivalentClass	oneM2M:MeasuringFunction
saref:ActuatingFunction	owl:equivalentClass	oneM2M:ControllingFunction
saref:Command	owl:equivalentClass	oneM2M:Command

表 2.3　SAREF oneM2M Classes mapping[16]

SAREF	Mapping	oneM2M
saref:offers	owl:equivalentProperty	oneM2M:hasService
saref:hasFunction	owl:equivalentProperty	oneM2M:hasFunction
saref:represents	owl:equivalentProperty	oneM2M:exposesFunction
saref:hasCommand	owl:equivalentProperty	oneM2M:hasCommand
saref:consistsOf	owl:equivalentProperty	oneM2M:consistsOf

表 2.4　SAREF oneM2M ObjectProperty mapping

■他領域に拡張 (図 2.12)　IoT 領域の抽象モデルとして様々なサブ領域に拡張できる.

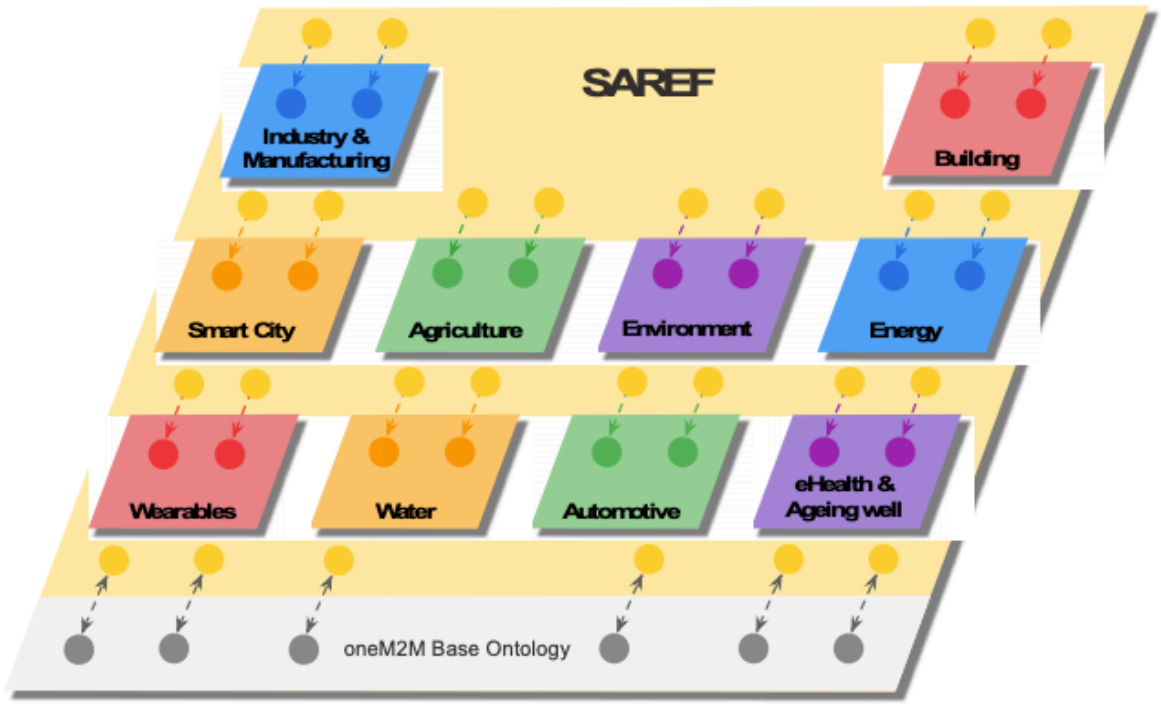


図 2.12　SAREF の拡張 [16]

- SAREF4ENER エネルギー (ETSI TS 103 410-1)
- SAREF4ENVI 環境 (ETSI TS 103 410-2)
- SAREF4BLDG 建築 (ETSI TS 103 410-3)
- SAREF4CITY SmartCities(ETSI TS 103 410-4)
- SAREF4INMA 生産製造 (ETSI TS 103 410-5)
- SAREF4AGRI 農業と食物供給 (ETSI TS 103 410-6)

2.2.6 NGSI-LD

組織，標準

NGSI-LD: **N**ext **G**eneration **S**ervice **I**nterface - **L**inked **D**ata

ETSI ISG CIM(Industry Specification Group cross cutting Context Information Management) と ETSI GS CIM 規格.

FIWARE-IoT の IoT ブローカは NSGI をコンテキスト記述モデルにしている.

目的

コンテキスト管理，コンテキスト標準化.

システム間の相互操作.

アプリケーション間のデータ交換を容易にする.

具体的な実現と抽象的な標準の間のギャップを補い，各具体的な領域に拡張できる.

情報モデル

■モデルの特徴

- コンテキスト管理のためのモデルので，特定のプラットフォームに縛られない.
- Entity 中心の記述.
- Entity はセンサデバイスや情報機器など以外, 実世界に存在するものも記述できる.
- Entity 間の関係を記述できる.
- Entity の操作に関する記述がない.

■モデル全体のレイヤ構造 (図 2.13 情報モデルは三階層に分けられ，中心から core meta-model，cross domain ontology，domain specific ontology.

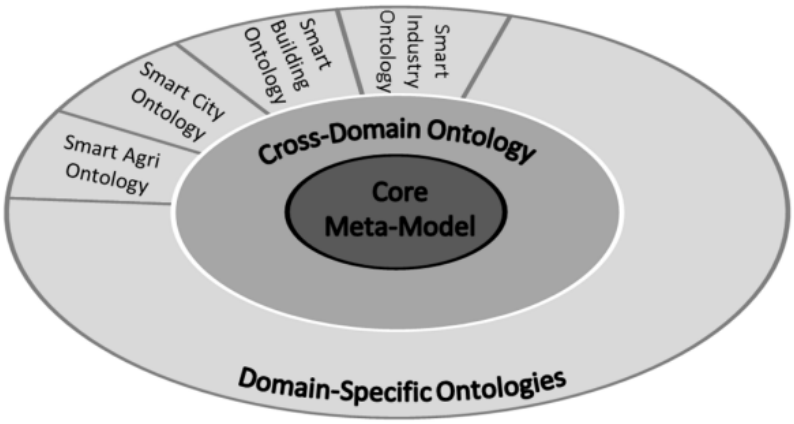


図 2.13　NGSI-LD モデルの構成 [17]

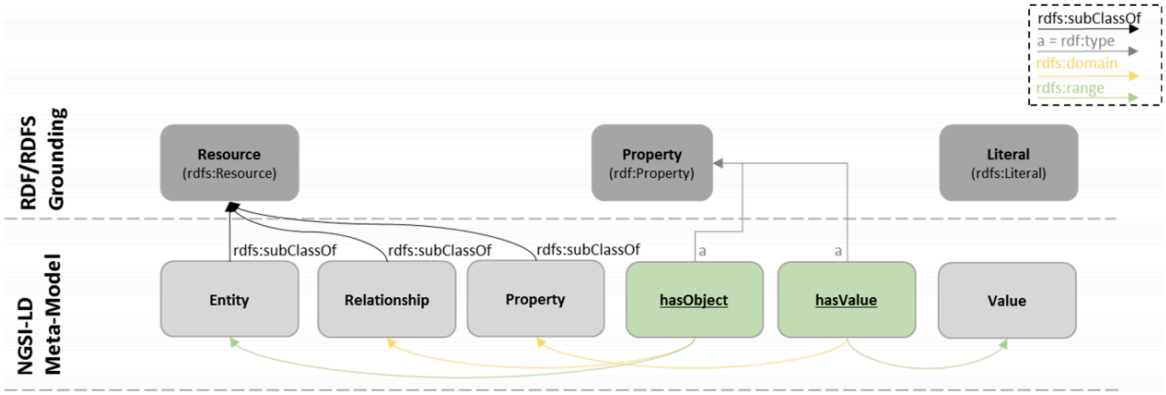


図 2.14　NGSI-LD core meta-model[17]

■NGSI-LD meta-model(図 2.14)

- NGSI-LD Entity: 実世界の中に存在している物理的あるいは概念的な物事の情報.
- NGSI-LD Property: Entity，Value，Relationship，Property の特徴記述.
- NGSI-LD Value: Json 値あるいは Json タイプの値.
- NGSI-LD Relationship: 対象の間の有向リンク．対象は Entity，Property，Relationship 中の一つ.

■ETSI ISG CIM cross-domain ontology(図 2.15)

特定領域の概念を定義していない.

領域特定オントロジーを cross-domain ontology の上に定義する.
位置情報の記述は IETF 標準の GeoJson(RFC7946) を利用する.

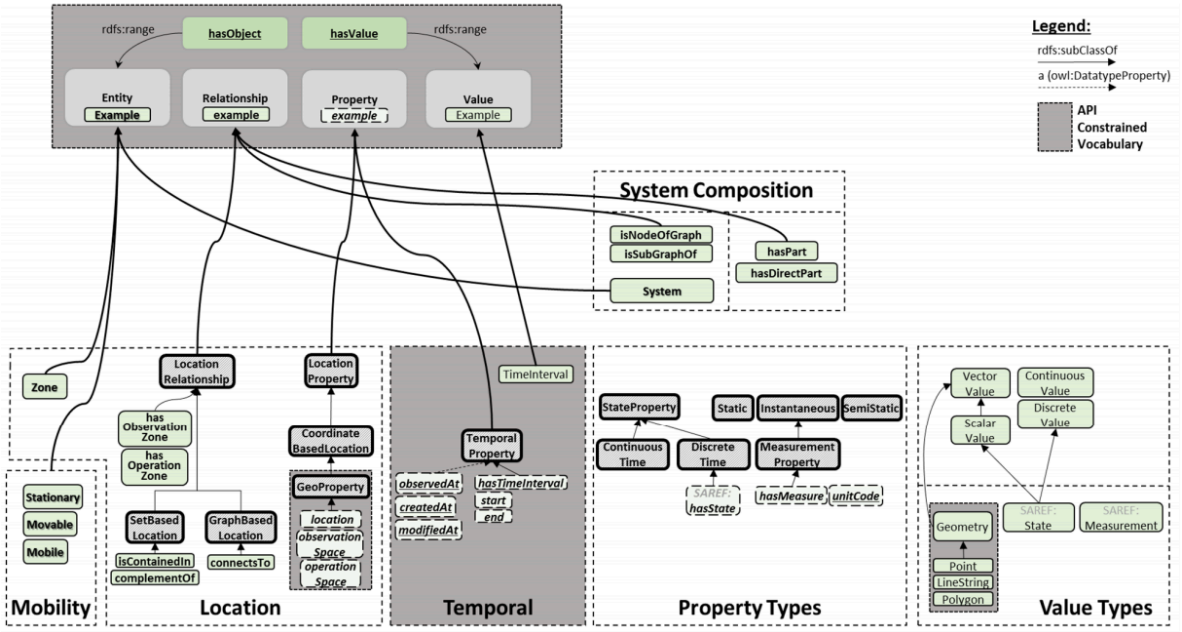


図 2.15　NGSI cross-domain model[17]

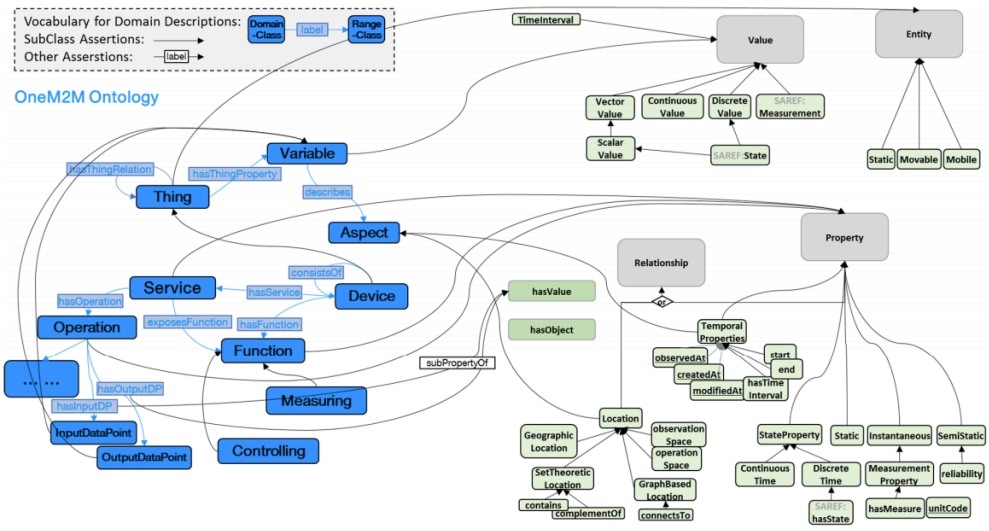


図 2.16　NGSI Mapping to oneM2M[17]

■mapping to oneM2M(図 2.16)

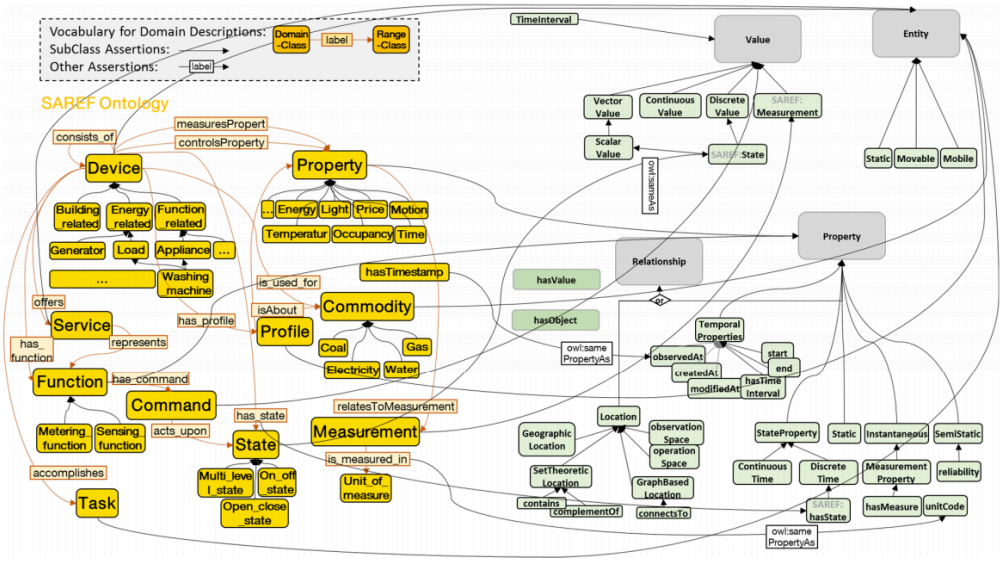


図 2.17　NGSI Mapping to SAREF[17]

■mapping to SAREF(図 2.17)

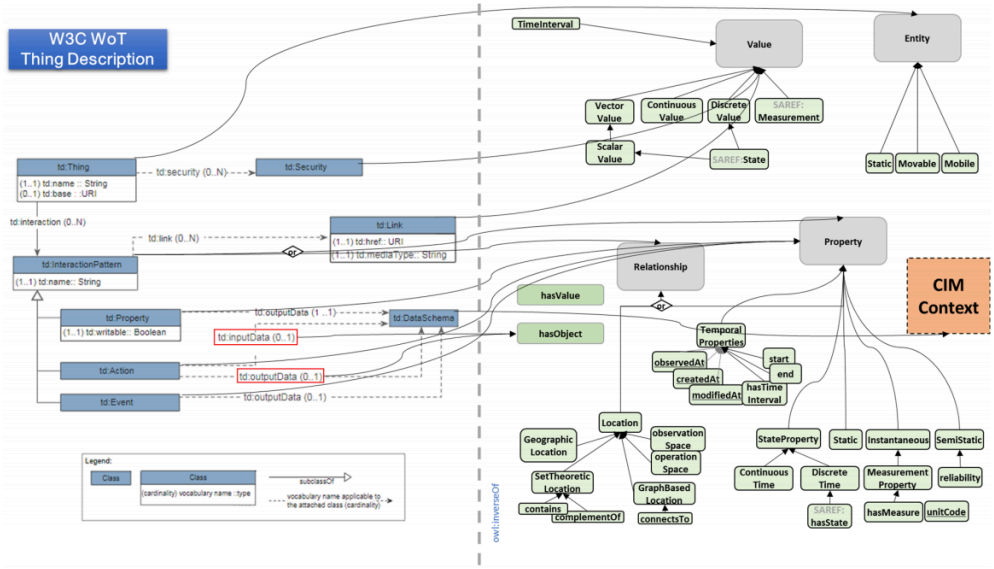


図 2.18　NGSI Mapping to WoT-TD[17]

■mapping to WoT-TD(図 2.18)

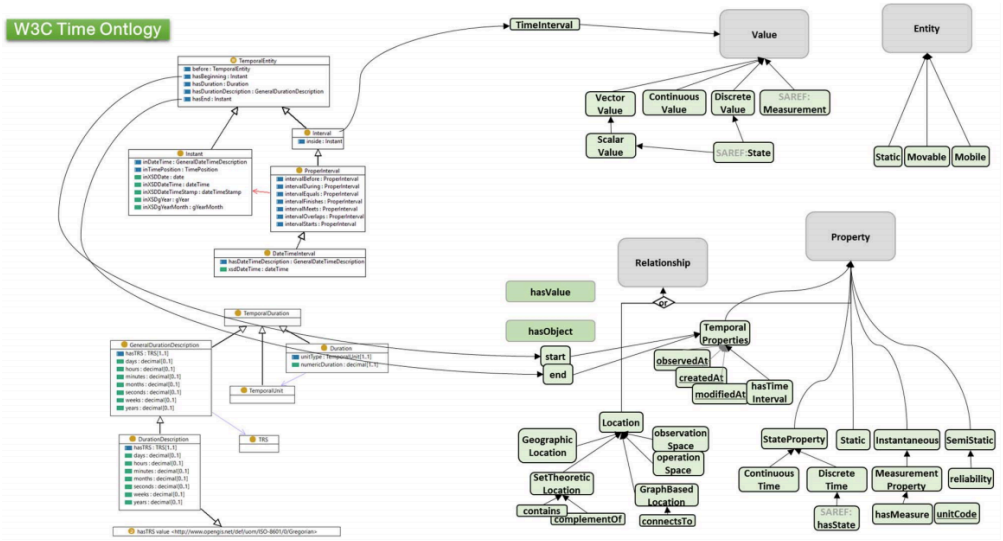


図 2.19　NGSI Mapping to W3C Time Ontology[17]

■mapping to W3C Time Ontology(図 2.19)

2.2.7　W3C WoT-TD

組織，標準

WoT-TD: **W**eb **o**f **T**hings - **T**hings **D**escription[18]

W3C 標準化中.

W3C WoT のリソース記述言語.

目的

物，インタラクション，データ交換の記述

情報モデル

■モデルの特徴

- Thing 中心の記述モデル.
- ものの操作方法を記述するモデル.
- ものの固有属性を記述するため，他のモデルを引用するべき.
- 持続的なプログラムに関する記述がある.

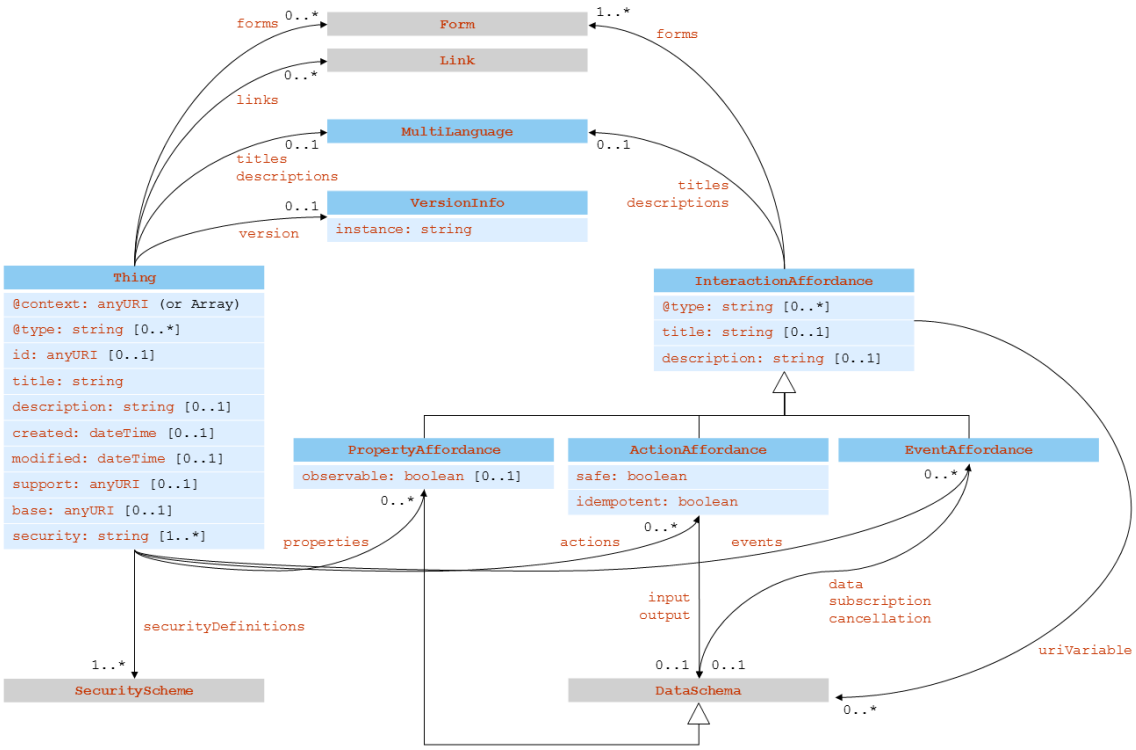


図 2.20　TD core vocabulary[18]

■コア語彙 (図 2.20)

- **Thing**: フィジカルとバーチャルエンティティの抽象，バーチャルエンティティは一個あるいは複数個の Thing より組み合わせる．「@context」属性で他のモデル (SAREF など) を導入できる．
- **InteractionAffordance**: Thing のメタデータ，インタラクションの記述，W3C-WoT では Property，Action，Event の三種類のインタラクション能力が定義されている．
  - **PropertyAffordance**: センシングデータと制御パラメータの記述．InteractionAffordance と DataSchema のサブクラス
  - **ActionAffordance**: Thing の呼び出し可能な機能の記述．状態を操作する (e.g., toggling a lamp on or off) や Thing の機能 (process) を起動する (e.g., dim a lamp over time)，InteractionAffordance のサブクラス
  - **EventAffordance**: 利用者に非同期で送る事件の記述 (e.g., overheating alerts)，InteractionAffordance のサブクラス

- **VersionInfo**: TD ドキュメントバージョンの記述．TD のコンテキスト拡張によって他のバージョンも記述できる．
- **MultiLanguage**: ヒューマンリーダブルの言語タグ．

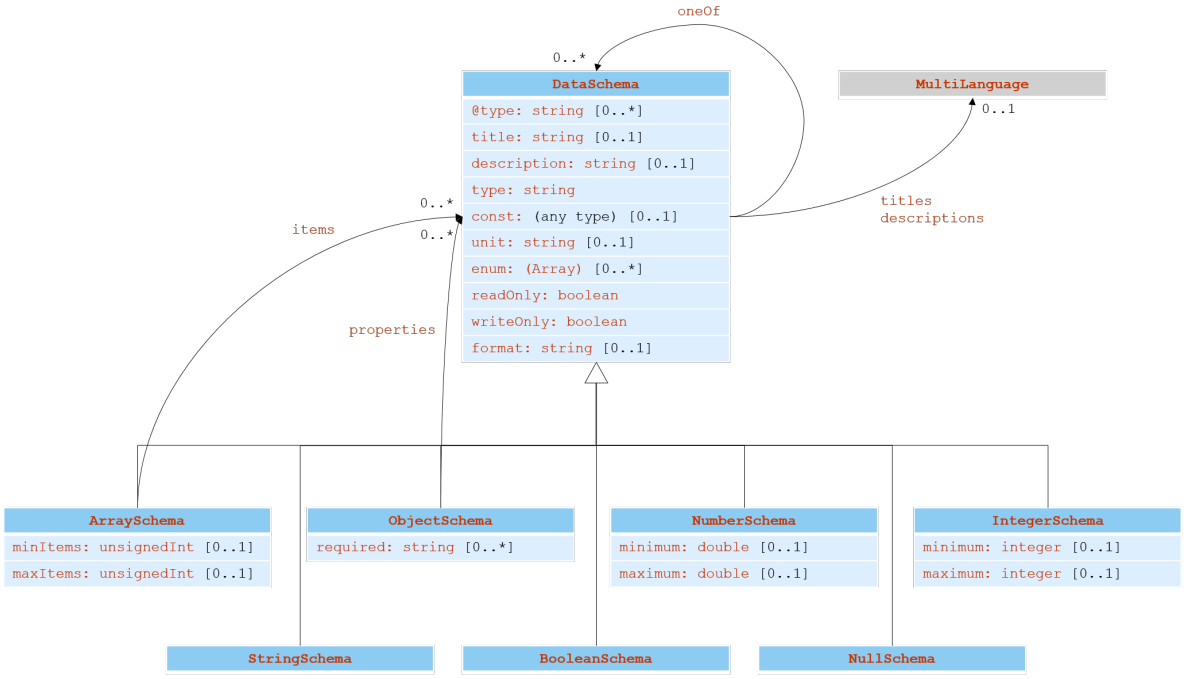


図 2.21　Data schema vocabulary[18]

■データスキーマ語彙 (図 2.21)

- **DataSchema**: データフォマットを記述するためのメタデータ．
  - **ArraySchema**: Array 型データのメタデータ (最大最小アイテム数，アイテムの特徴)，DataSchema のサブクラス
  - **BooleanSchema**: Boolean 型データのメタデータ，DataSchema のサブクラス
  - **NumberSchema**: Number 型データのメタデータ (最大値，最小値，double type)，DataSchema のサブクラス
  - **IntegerSchema**: Integer 型データのメタデータ (最大値，最小値 integer type)，DataSchema のサブクラス
  - **ObjectSchema**: Object 型データのメタデータ (再帰的な定義，必須なタイプの定義)，DataSchema のサブクラス
  - **StringSchema**: String 型データのメタデータ，DataSchema のサブクラス
  - **NullSchema**: Null 型データのメタデータ (null を表す)，DataSchema のサブクラス．

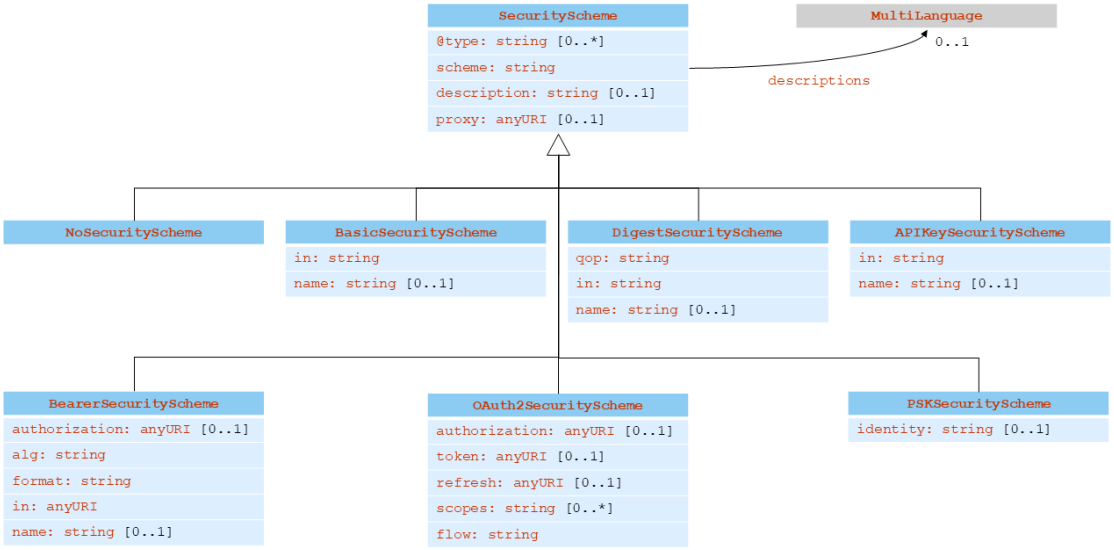


図 2.22　WoT security vocabulary[18]

■セキュリティー語彙 (図 2.22)

- **SecurityScheme**: セキュリティメカニズムの記述.
  - **NoSecurityScheme**: 身分認証が不要.
  - **BasicSecurityScheme**: Basic Authentication([RFC7617]) を利用する．暗号化されていないユーザーネームとパスワード.
  - **DigestSecurityScheme**: Digest Access Authentication([RFC7616]) を利用する．BasicSecurityScheme より，中間者攻撃を防ぐ追加機能がある.
  - **APIKeySecurityScheme**: API key authentication を利用する．API キーで身分認証をする．アクセストークンが不明確で，標準トークン形式を使っていない場合に用いる.
  - **BearerSecurityScheme**: Bearer Token([RFC6750]) を利用する．OAuth2

から独立して使用される状況．format:jwt-[RFC7519], jws-[RFC7797], cwt-[RFC8392], jwe-[RFC7516]

– **PSKSecurityScheme**: Pre-shared key authentication を利用する．
– **OAuth2SecurityScheme**: OAuth2 authentication を利用する．[RFC6749] と [RFC8252] に適合の場合

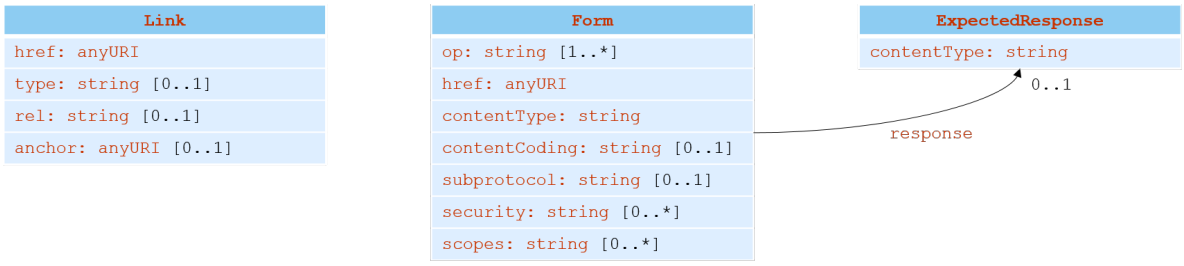


図 2.23　Hypermedia controls vocabulary[18]

■ハイパーメディアコントロール語彙 (図 2.23)

- **Link**: "*link context* has a *relation type* resource at *link target*"
- **Form**: "To perform an *operation type* operation on *form context*, make a *request method* request to *submission target*"
- **ExpectedResponse**: 予想される応答メッセージを記述する通信メタデータ

2.2.8　BIG IoT

■組織，標準　BIG IoT: **B**ridging the **I**nteroperability **G**ap of the **I**nternet **o**f **T**hings[19].
EU H2020 BIG IoT プロジェクト．
データマーケットプレースに関する検討がある．

■連携プラットフォーム

- ATOS
- BOSCH
- SIEMENS
- SEAT
- CSi

- econais
- vmz
- wolfburg
- AALBORG UNIVERSITY
- Insight
- TU Clausthal
- UNIVERSITAT POLITECNICA DE CATALUNYA BARECALONATECH

目的

共通 API の定義，プラットフォーム間の相互操作.

情報モデル

■モデルの特徴　Offering 中心のモデル.

マーケット向け，提供できるものに関する記述.

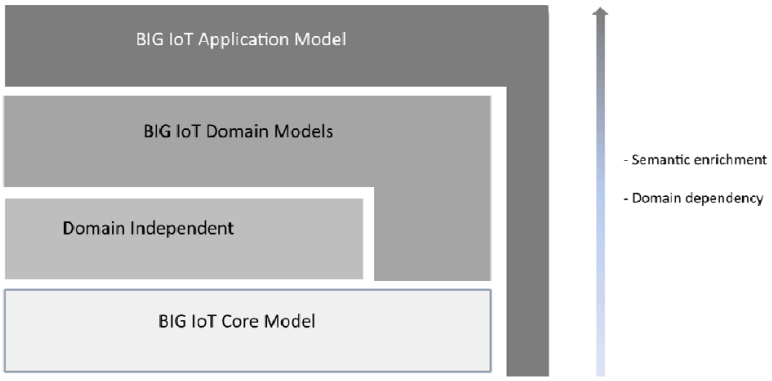


図 2.24　BIG IoT モデルのレイヤ [20]

■BIG IoT モデルのレイヤ (図 2.24)

- BIG IoT Core Model: D3.2b[21]
- Domain Indepentent Model: D3.2b[21]
- BIG IoT Domain Model: D4.2b[20]
- BIG IoT Application Model: D4.2b[20]

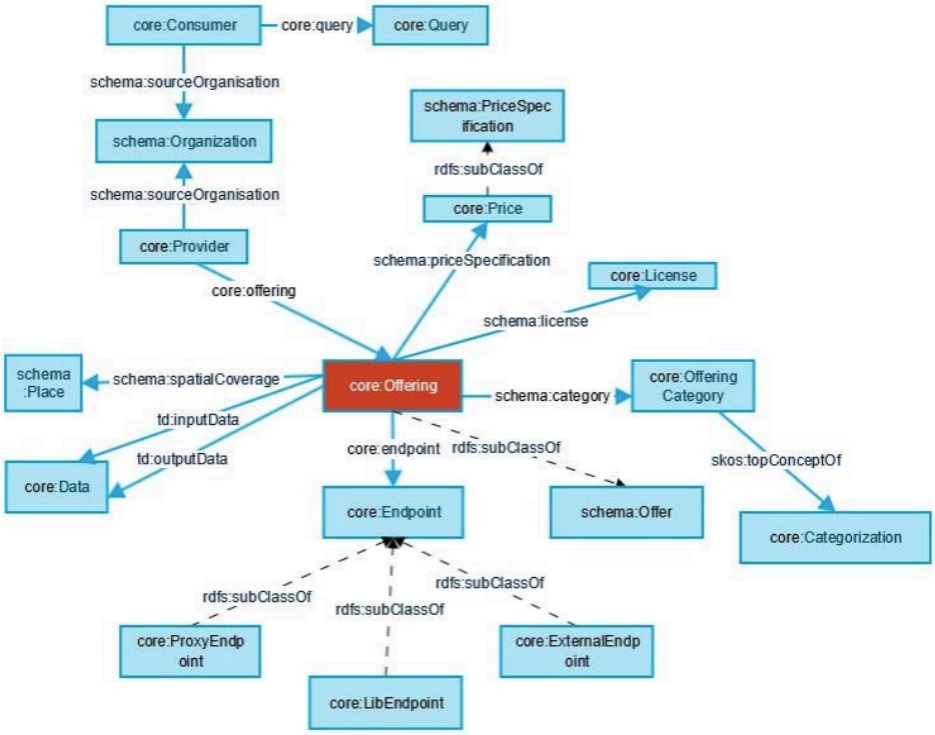


図 2.25　BIG IoT semantic core model

■BIG IoT Semantic Core Model(図 2.25)

- **役割**:

  BIG IoT 中の Offering(提供者が提供できるサービスとアクセス) と OfferingQuery を記述するコンテキストの語彙とストラクチャを定義する.
- **メイン概念**:
  - core:OfferCategory : Offering の分類
  - core:Data : Offering の入力と出力
  - core:Endpoint : Offering をアクセスためのインタフェース
- **モジュール化**:
  - Provider module: プラットフォームやサービス実例など，core:Provider クラスで記述
  - Organization module: 提供者の組織記述，schema:Organization クラスで記述
  - Offering module: コア概念，提供者が提供する製品?商品 (offering) の記述，schema:Offer のサブクラス，core:Offering クラスで記述

  - Consumer module: IoT リソースをアクセスするアプリケーションやサービスの実例，core:Consumer クラスで記述
  - Query module: クエリのアブストラクション，Consumer の興味ある属性を記述する (Offering type, input&output data, price, license, ...)，core:OfferingQuery クラスで記述

■BIG IoT Semantic Domain Models　製品の説明 (Offering Descriptions (ODs)) と製品のクエリ (Offering Queries) を記述するために，ドメイン依存語彙 (domain dependent vocabulary) とドメイン非依存語彙 (domain independent vocabulary) が必要.

- Domain Independent model:
  特定ドメインのアプリケーション (parking, smart city, or air quality など) に依存しない情報モデル
  - schema.org
  - W3C SSN Ontology
- Domain Dependent model:
  特定ドメインのアプリケーションに依存する情報モデル
  - M3, M3-lite
  - MobiVoc:Open Mobility Vocabulary，モビリティ関連の記述語彙
  - DATEX II:DATEX 規格のモデル，トラフィック管理のため
  - OCPI:Open Content Provider Interface，システム統合用
  - Pollution Ontology:ISO 37120，環境指標
- Offering の NonFunctional 属性モデル:
  - Location:
    * WGS84 Geo Positioning
    * The GeoNames Ontology
  - Time:
    * Time Ontology
  - サポートしている二つのドメイン:
    * Mobility (Parking, Traffic, Bus, Charging,..)
    * Environment (Air Quality, Pollution, ..)

■BIG IoT Application Model

- Offering の分類
  - mobility:Parking
  - mobility:Traffic
  - mobility:Transportation
  - mobility:Charging
  - mobility:Environment
- 入出力のデータモデル
- Offering のドメイン依存メタデータ

2.2.9 FIESTA-IoT

組織，標準

FIESTA-IoT: Federated Interoperable Semantic IoT Testbeds and Applications
EU H2020 FIESTA-IoT プロジェクト.
独自に概念を定義しない，既存オントロジーの融合.

目的

IoT プラットフォーム (テストプラットフォーム) の相互操作.
IoT プラットフォーム相互操作のテストベッド.

情報モデル

この部分は FIESTA-IoT ontology について説明する.

■モデルの特徴

- Device 中心の記述.
- IoT-lite に基づく.
- 既存オントロジーの融合.
- プロジェクトのアーキテクチャは IoT-A Architecture Reference Model (ARM) をテンプレートにし，コア概念も [22]ARM モデルと一致いている.

■オントロジー

- コア概念:
  - Resource: 計算性能を持っているエレメント.「Physical Entity」のアクセスや

機能の実行ができる．

  - Virtual Entity: 「Physical Entity」を表す計算やデータ要素．
  - IoT service: 定義されたインタフェースを通じて IoT リソースとインタラクションするソフトウェアコンポーネント．

- 引用モデル:
  - SSN:センサ，デバイスの記述
  - IoT-lite:リソース，エンティティー，サービスの記述
  - WGS84:位置情報の記述
  - Time:時間の記述
  - DUL:物理的，ソーシャル的コンテキストの記述
  - M3-lite:分類
  - QU:計測単位，数量種類

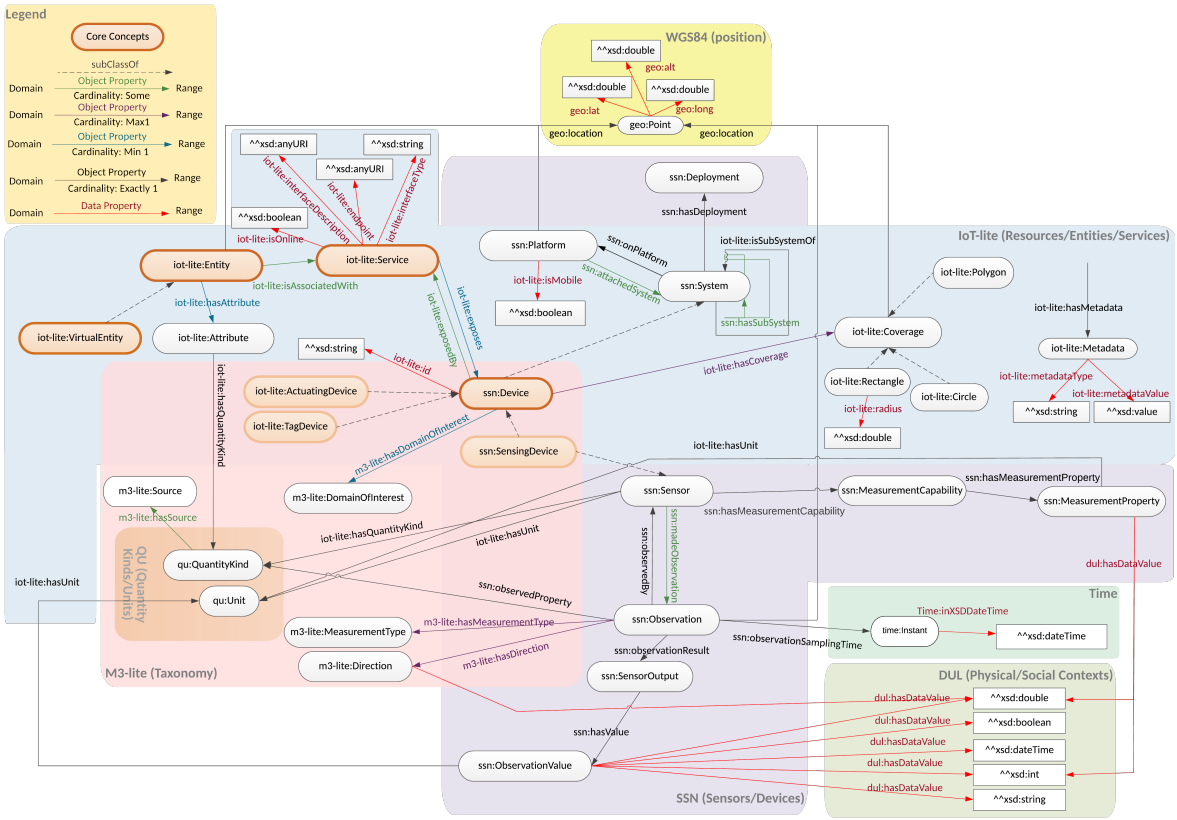


Figure 1: FIESTA-IoT Ontology

図 2.26　FIESTA-IoT Ontology[22]

- 図 2.26 の重要部分:

- リソースのグラフのルートは実際の所有者，ssn:Deployment で表す.
- GEO オントロジーでデバイスの物理的な位置を記述する．(geo:lat と geo:long)
- Platform: 他のエンティティをアタッチできるエンティティ (例えば，post, buoy，fish など).
- Device: (ssn:Device に基づく)Resource 記述のコア．システムが持っているコンポーネント．FIESTA-IoT にとって, Device は ActuatingDevices, TagDevice, SensingDevices も可能.
- Service: 公開リソースやデバイスの要素．全ての Device は Service を継承するので，Device と SensingDevice に適用されない場合もある.
- IoT-A のルールにより，Virtual Entitie と Device と Service はモデルのコア.
- SensingDevice: 実世界をセンシングするデバイス．FIESTA-IoT 連合プラットフォームに設置された物理センサ.
- SensingDevice/Sensor を物理現象にマッピングすると「sensing」になる．M3-lite 分類法で記述する.
- SensingDevice の属性は物理属性以外，計測頻度，センサ精度，正しさなども含む.
- 図の下の部分は SensingDevice/Sensors が生成したのデータを注釈する．timestamp，location，計測値，QuantityKind/Unit で measurement/observation を記述する.

- **Mapping to oneM2M base ontology[22]**

Class in Base Ontology	**Mapping relationship**	**Class in FIESTA-IoT**
oneM2M:Thing	owl:equivalentClass	ssn:Sensor
oneM2M:ThingProperty	owl:equivalentClass	m3-lite:QuantityKind
oneM2M:Device	owl:equivalentClass	m3-lite:SensingDevice
oneM2M:Service	owl:equivalentClass	ssn:Observation
oneM2M:OperationOutput	owl:equivalentClass	ssn:ObservationValue
oneM2M:MetaData	owl:equivalentClass	m3-lite:Unit

表 2.5　Classes mapping between oneM2M Base ontology and FIESTA-IoT ontology

Object property in BaseOntology	**Mapping relationship**	**Object property in FIESTA-IoT**
oneM2M:hasThingProperty	owl:equivalentProperty	iot-lite:hasQuantityKind
oneM2M:hasService	owl:equivalentProperty	ssn:madeObservation
oneM2M:hasMetaData	owl:equivalentProperty	iot-lite:hasUnit

表 2.6　Object properties mapping between oneM2M Base ontology and FIESTA-IoT ontology

2.2.10　W3C SSN/SOSA

組織，標準

SSN:Semantic Sensor Network

SOSA:Sensor, Observation, Sample, and Actuator

OGC/W3C 標準である．

SOSA は SSN のコアとして含まれている．

SSN-DUL のマッピングモジュールを提供し，DUL の語彙定義と一致できる．

目的

センサ，アクチュエータ，サンプラとその観測，実行，サンプリングを記述する

情報モデル

セマンティックセンサネットワークを記述するモデル，IoT 領域の抽象モデル．

■オントロジー モジュール (図 2.27)　現在の SSN オントロジーは図 2.27 のように，幾つのサブモデルを提供する．owl:import を利用して低レベルのモデルを導入 (拡張) する．
垂直:高レベルモデルは低レベルモデルの上に構築される．低レベルモデルはロジカル的に高レベルモデルと独立しいる．
水平:モデル間相互的な依頼関係.

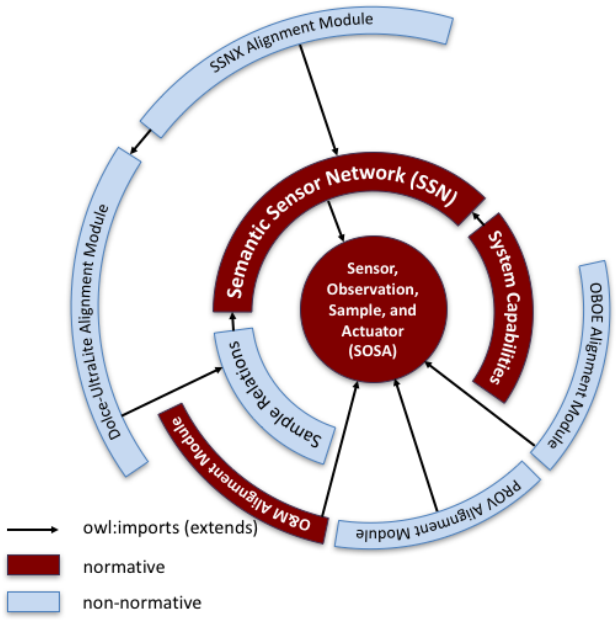


図 2.27　SOSA と SSN モデルのモジュール [23]

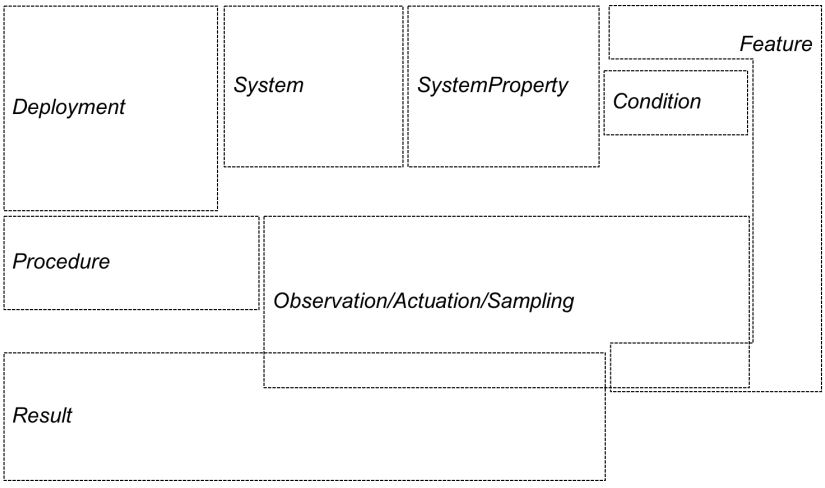


図 2.28　SSN OntStructure Overview[23]

■コアモジュール (図 2.28)　SOSA によって Schema.org の domainIncludes と rangeIncludes が利用される

Observation Actuator Sampling の三つの視点があり，視点によって記述内容が異なる．「Observation, Actuator, Sampling」と「Result」と「Procedure」と「Deployment」は SOSA

のモジュール.

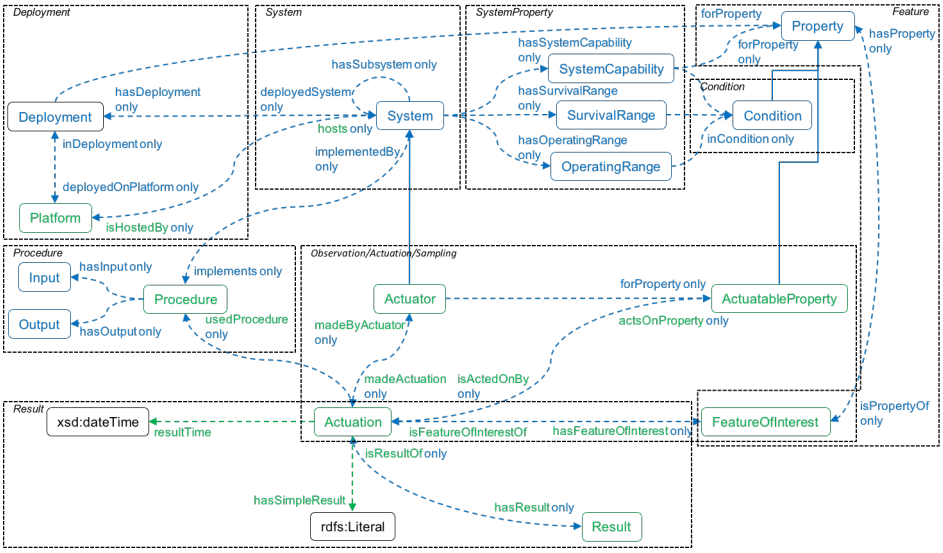


図 2.29　SSN OntStructure Actuation[23]

■SSN OntStructure Actuation

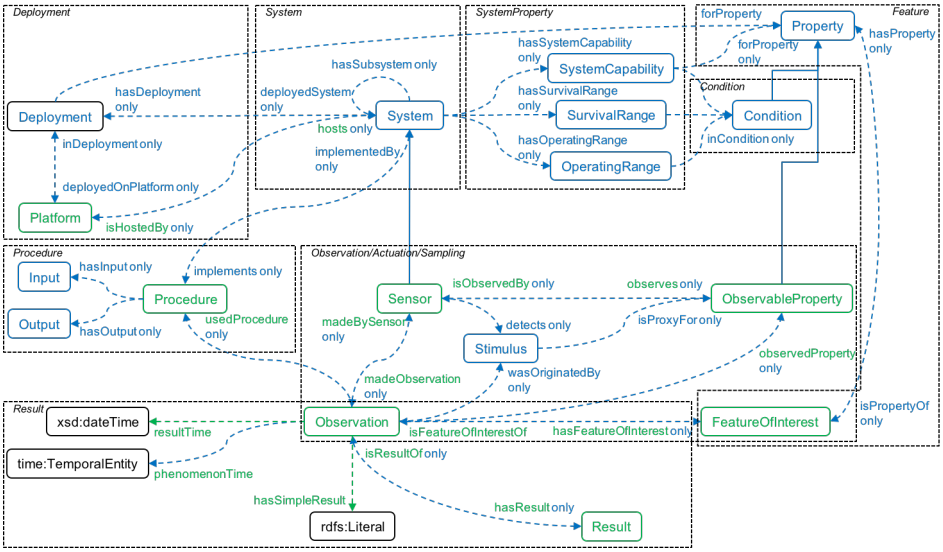


図 2.30　SSN OntStructure Observation[23]

■SSN OntStructure Observation

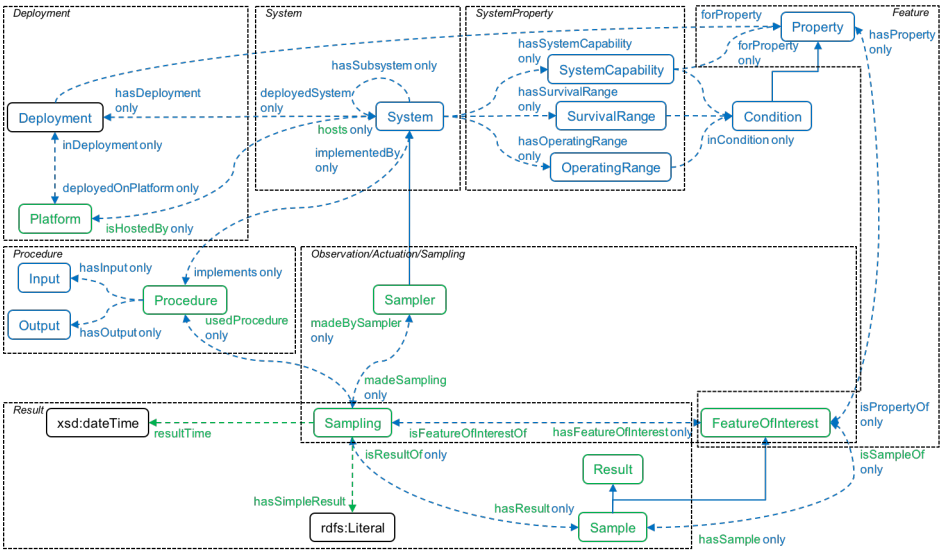


図 2.31　SSN OntStructure Sampling[23]

■SSN OntStructure Sampling

■マッピング

- **Dolce-Ultralite Alignment Module**:

SSN/SOSA class	関係	DUL class
sosa:Procedure	subclass of	dul:Method
sosa:Sensor	subclass of	dul:Object
sosa:Observation	subclass of	dul:Event
ssn:Property	subclass of	dul:Quality
ssn:Stimulus	subclass of	dul:Event
ssn:System	subclass of	dul:Object
sosa:Platform	subclass of	dul:Object
ssn:Deployment	subclass of	dul:Event

表 2.7　Dolce-Ultralite Alignment Module(Classes 部分)[23]

- **O&M Alignment Module**:

  O&M モデル:OGC Observations and Measurements(ISO 19156:2011)

sosa-om:ActuationProcedure	Actuation procedures or recipes
sosa-om:ObservationProcedure	Observation procedures or recipes
sosa-om:SamplingProcedure	Sampling, sample preparation or processing procedures or recipes

表 2.8　Utility Classes[23]

SSN/SOSA class	関係	O&M class
iso19156-om:OM_Observation	equivalent class	sosa:Observation
iso19156-om:OM_Process	equivalent class	Union of (sosa:Sensor or sosa-om:ObservationProcedure)
iso19156-sf:SF_SamplingFeature	equivalent class	sosa:Sample
iso19156-sf:SF_Process	equivalent class	Union of (sosa:Sampler or sosa-om:SamplingProcedure)
iso19156_sp:PreparationStep	subclass of	sosa:Sampling
sosa:FeatureOfInterest	subclass of	iso19156_gfi:GFI_DomainFeature
sosa:Actuator	subclass of	iso19156_gfi:GFI_Feature
sosa:Platform	subclass of	iso19156_gfi:GFI_Feature

表 2.9　O&M Alignment Module(Classes 部分)[23]

2.2.11　M3

組織，標準

M3:Machine-to-Machine Measurement

よく使われる分類用モデル.

目的

曖昧な定義をコンテキストで明確する.

デバイスの名付けを統一する.

相互操作の実現.

明示的なセンサメタデータ記述.

新しい知識を推論するための推論エンジン.

ドメイン知識の再利用.

M3 の命名法を更新するためのフレキシブル，簡単な方法を提供する.

情報モデル

W3C SSN オントロジー ssn:ObservationValue, ssn:FeatureOfInteres, ssn:Sensor の拡張.
具体的なデバイス種類までの記述がある.

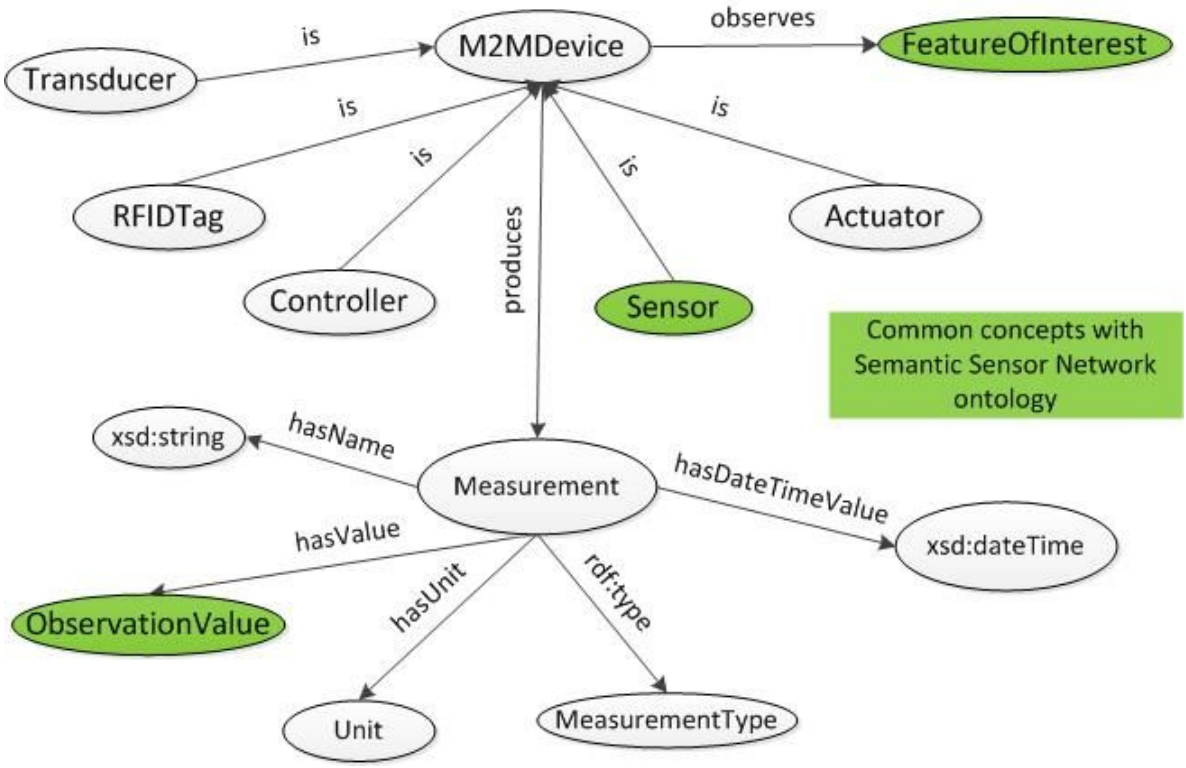


図 2.32　M3 ontology[24]

- ssn:FeatureOfInterest を拡張し
  12 個領域の observations，sensors，units を分類した
  分類された領域:
  - building automation
  - health
  - weather
  - agriculture
  - environment
  - emotion
  - transport

  - energy
  - tourism
  - location
  - city
  - tracking good
- オントロジーネットワークの構築

  関連したオントロジー :
  - IoT オントロジー (OpenIoT, SPITFIRE15)
  - セマンティックセンサネットワークオントロジー (W3C SSN, ontoSensor)
  - ユニットオントロジー (QUDT, QUDT_unit, Measurement Unit Ontology (muo))
  - ロケーションオントロジー (WGS84)
  - ドメインオントロジー:
    * スマートホーム (Smart Home Weather ontology (shw), etc.)
    * 天気 (Ontology for Meteorological sensors (aws), Smart Home Weather ontology, etc.)
    * など

2.2.12 M3-lite

組織，標準

EU FIESTA プロジェクトの一部.

分類法 (Taxonomy) オントロジー，M3 の軽量化バージョン.

目的

中身は各サブ領域，センサ種類，計量単位，計量種類が書かれている.

FIESTA-IoT のニーズを満足する．FIESTA-IoT と関係ない属性とクラスがない.

センサ，計測，ドメイン，ユニットの定義.

情報モデル

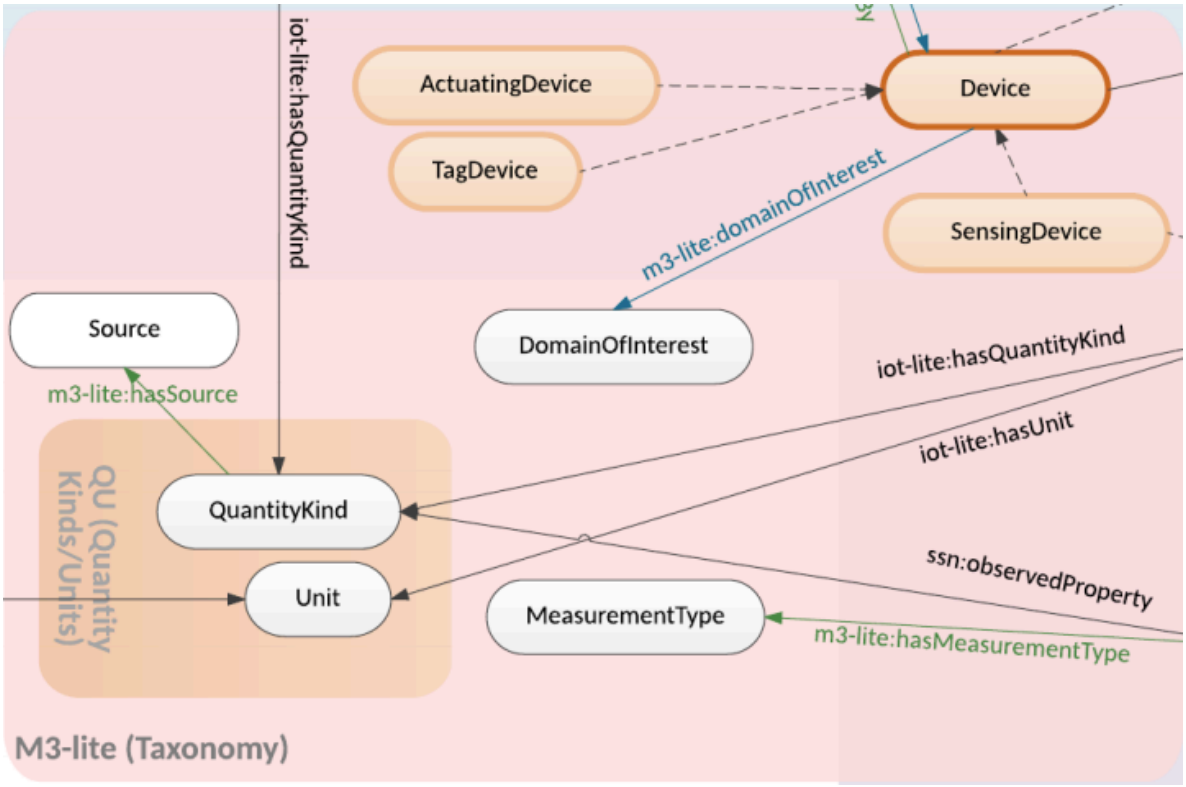


図 2.33　M3-lite モデル [22]

- **分類**:
  - Resource Type(例えば温度計).
  - Quantity kind．デバイスの感知できる物理現象の記述.
  - Domain of Interest．異なる IoT 領域の記述 (例えばスマートホーム).
  - Units．デバイスが生成したデータの単位，物理現象と関連している (例えば摂氏).
- 精確な W3C SSN 拡張，以下の内容に**統一な分類法**を提供する:
  - ssn:Device．デバイスは ActuatingDevice，SensingDevice，TagDevice.
  - ssn:Sensor．ssn:Sensor は独自の分類法を持つ (ssn：SensingDevice).
  - qu:QuantityKind.
  - qu:Unit.
- **既存モデルの再利用**:
  他の IoT オントロジーと関連できる．語彙の意味を一致している．オントロジー間

の相互操作が簡易になる.

- SSN
- SWEET_unit
- foot_smart_product
- home
- IoT-lite
- mo
- movie
- muo
- naturopathy
- ontoSensor
- OpenIoT
- QU
- qu_rec20
- qudt
- qudt_unit
- SHW(Smart Home Weather)
- spitfire
- txn(taxonconcept)
- ucum

2.2.13 OMA LWM2M (+IPSO)

組織，規格

OMA: **O**pen **M**obile **A**lliance

LwM2M: Low Weight Machine to Machine

IPSO: **I**nternet **P**rotocol for **S**mart **O**bject

記述内容

デバイスレベルの規格.

オフィシャルなオントロジーがない.

■**OMA LwM2M の構成**　OMA LwM2M は三部分を規定した:

LwM2M Client

LwM2M Bootstrap-Server

LwM2M Server

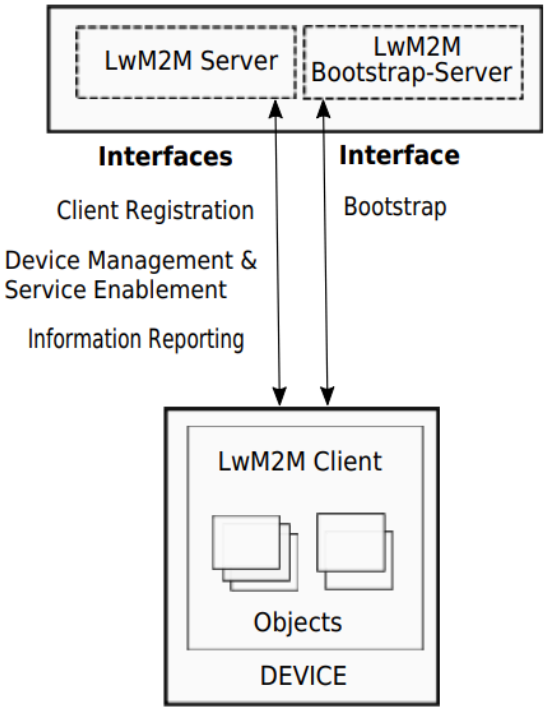


図 2.34　OMA LwM2M 全体像 [25]

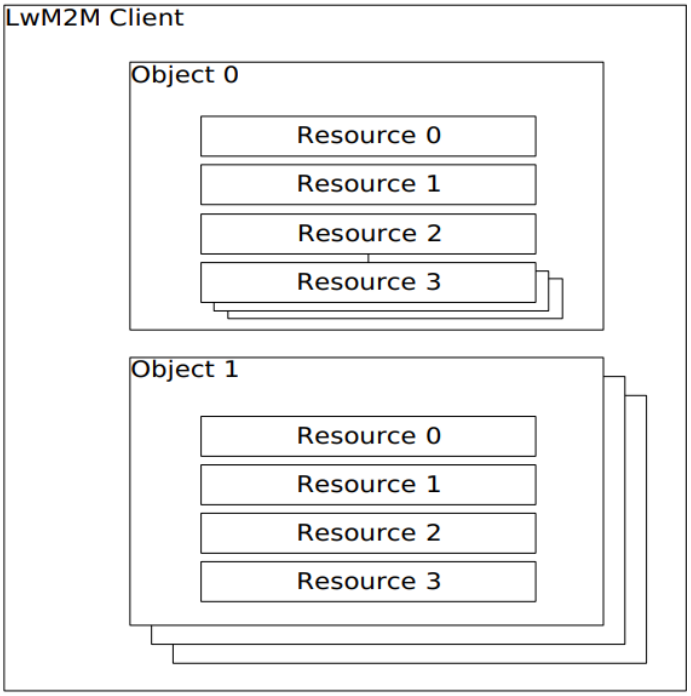


図 2.35　OMA-LWM2M Client[25]

■**OMA LwM2M Client の構成**　一つの Client は複数の Object を持つ，一つ Object は複数の Resource を持つ.

各 Object と Resource に ID が付けられる．ObjectID と ResourceID の意味は規格書で規定された.

指令のフォマット:　Object ID/ Object Instance ID/ Resource ID

Object ID: Object 種類
Object Instance ID:同じ Client の具体的な Object
Resource ID:Object の属性

操作は Read，Write，Execute 三種類.

2.2.14　Echonet lite

組織，標準

国際標準 ISO/IEC 14543-4-3 と IEC 62394
ECHONET Lite 規格デバイスの情報モデル.
オフィシャルなオントロジーがない.

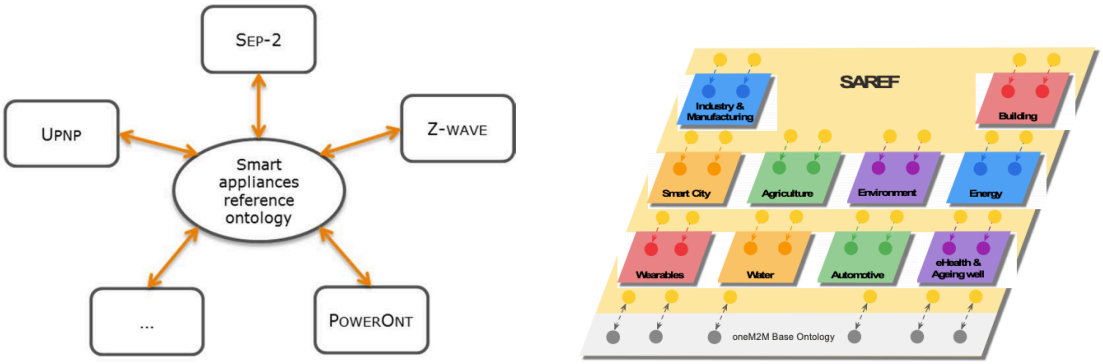
記述内容 [26]

家電，住宅設備を中心に，具体的，かつ詳細な情報モデルを定義している.

ユニークなアドレスがついている通信機器はノードと呼ばれる．ノードが複数の EOJ を含むことができ，一つの EOJ は複数の EPC を含む.

- EOJ:デバイスの種類
- EPC:デバイス持っている属性
- 各属性とインタラクション (get，set，notify)

2.3 議論



(a) (b)

図 2.36　SAREF の目的と SAREF 利用上の位置

2.2.5 節でも述べたように，ETSI による SAREF は情報モデル間のマッピングを行うための中間モデルを指向したもの (図 2.36a) で，様々なオントロジの基本となる部分を有している．これに，サービス (アプリケーション) 分野 (領域) ごとに必要とされる詳細モデルを追加した分野ごとの拡張モデル (図 2.36b) を作成し，それらの共通部分としての位置づけを有するのと同時に，oneM2M のようなプラットフォームにおける基本オントロジとのマッピングを可能とするエントリポイントとしての位置づけも有している．図 2.37 に，その様子を示す．

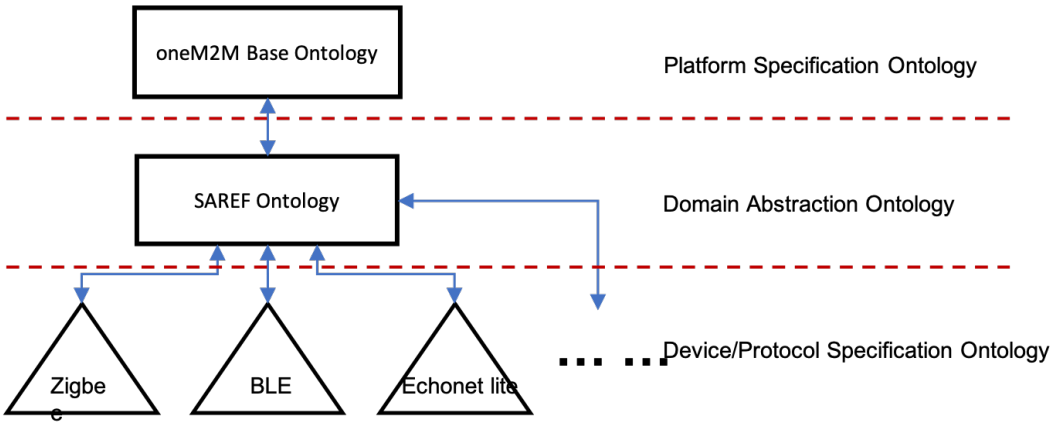


図 2.37　情報モデル角度のレイヤ

ここで，最下層はデバイスが使われる分野や，通信に使われる技術といった個別のモデル群を表しており，中間の層はこれらの間で共通の部分を抽象化した中間モデルである．最上位の層は，サービスを実現していくためのプラットフォームが持つモデルを表しており，中間モデルはこちらとの橋渡しも行うものとなるが，何れにおいても主眼がデータモデルそのものとなっていると考えられる．

一方，FIWARE[27] では，図 2.38 のようなレイヤ構造を有している．

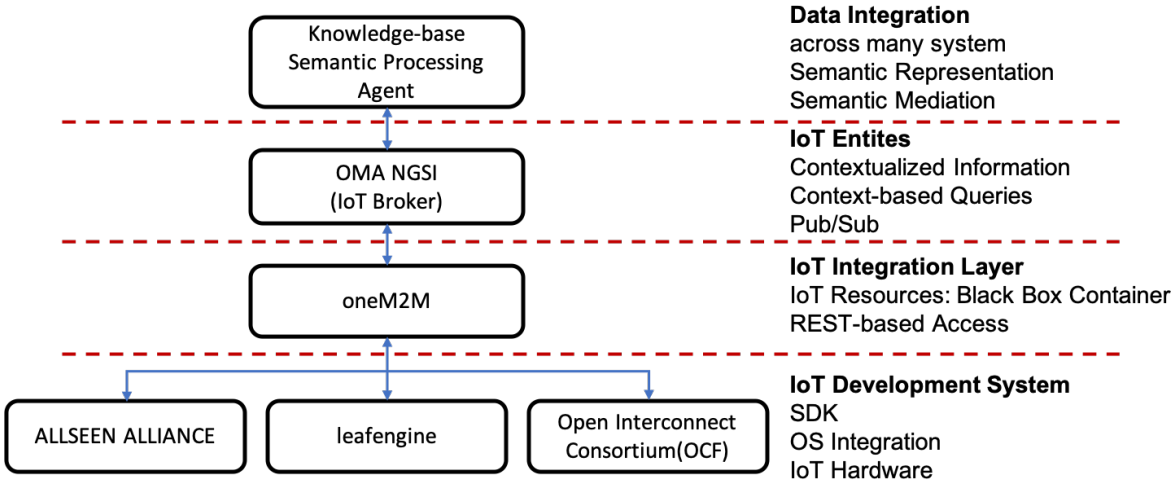


図 2.38　プラットフォーム角度のレイヤ

こちらでは，サービスの実現に向け，上位の立場から具体的なデバイス群を捉えているような構造となっている．最下層は図 2.37 における最下層と類似しているものの，データモデルそのものというよりは，機器からの情報取得や操作の API として捉えているものと考えられる．最下層に位置するそれぞれのエコシステムをまとめるプラットフォームとして，oneM2M のモデルによる統合層が下から二番目の層として存在し，最上位層の知的処理を実現するために，複数のプラットフォームからの情報を活用できるようにする IoT 情報のブローカーが上から二番目の層として位置している．この層では単なるセンシング情報といったレベルではない，コンテキストを持つ情報が扱われる．

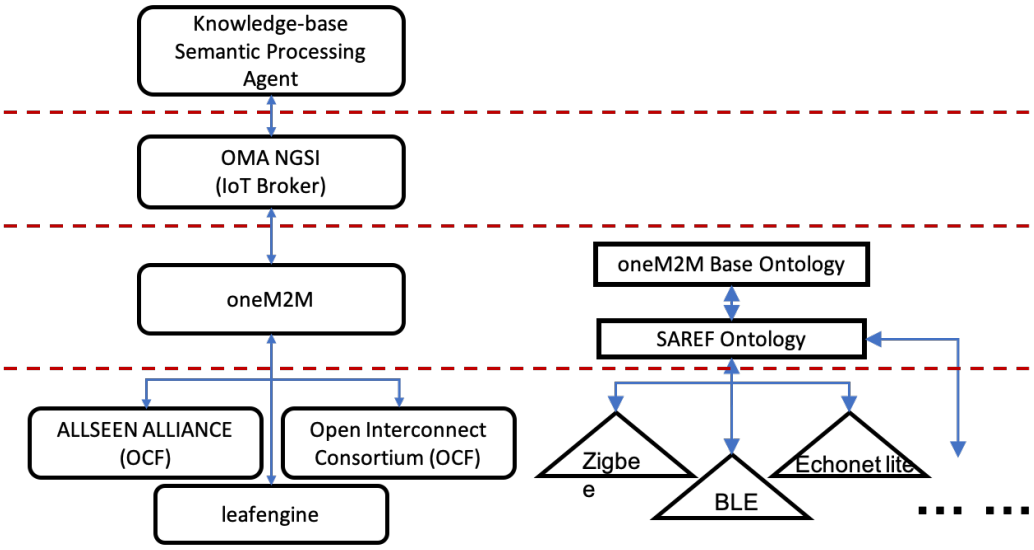


図 2.39　レイヤ構造の比較

図 2.37 と図 2.38 を整合させつつ統合したのが図 2.39 である．上位層が対象とする抽象度の高い情報処理の側面と，下位層が持つ，具体的な機器や物理量に対応する処理の側面の両側からの要求が中間の層で整合される形になるため，下から二番目の層に位置づけられる処理には多少意味合いの異なるものが集まってくることがわかる．

計算機ネットワークにおけるプロトコルも，かつての OSI 7 層モデル，TCP/IP の 4 層モデルから，概ね 5 層程度のモデルへと変遷し，現在でも有力な技術の登場のたびに揺らいでいるのと同様，図 2.39 のような図も今後，様々な変容をみせるものと考えられる．

この図 2.39 の構造に沿って，本調査で調査した規格を並べるとすれば，以下のように位置づけられるものと思われる．

- Data Integration
  - (アクセスと処理だけを実行するため，調査範囲外)
- IoT Entities
  - FIWARE (NGSI-LD)
  - W3C WoT-TD
- IoT Integration Layer
  - oneM2M Base Ontology
  - universAAL ontology
  - BIG IoT Information Model
  - FIESTA-IoT Semantic Model

  - ETSI-SAREF Ontology
  - M3/M3-lite Ontology
  - W3C SSN/SOSA Ontology
- IoT Development System / Device Layer
  - OCF (oneIoTa model)
  - ECHONET lite
  - OMA LwM2M(+IPSO)

2.4 調査結果のまとめ

この調査では，様々なデバイスやサービスを組み合わせてより高度な IoT サービスを実現可能とするために必要となる，データモデルの共通化手法に関し，現在までの取組みの調査を中心とした検討を行った．

具体的な機器が提供するプリミティブな機能と，抽象度の高いサービスの実現を AI などの高度な情報処理技術を使って行うためのサービス開発との間には大きなギャップがあり，概ね上位と下位の二段に分けた層構造が必要になると考えられる．

下位の層においては，SAREF や W3C SSN のような，具体的な機器やデータをもう一段抽象化するしくみが必要となる．一方で，IoT プラットフォームとしてサービス実現のための層に必要な機能を実現していくためには，コンテキスト記述やインタラクション記述モデルのような高度な抽象モデルが必要となるものと考えられる．

今後，具体的なシステム開発例が増えるにつれ，こうした層構造も細分化したり統合するといった変化をみせるようになるものと思われる．

第 3 章

データ連携手法に関する調査

本章では，世の中に存在し，利用されているデータ連携手法をまとめ，データ利用者の視点から既存手法の問題点を検討する．

3.1　独立型

独立型のデータ連携手法では，各機器メーカのデータ提供クラウドが独立しており，プラットフォームを使わないデータ連携方式である．

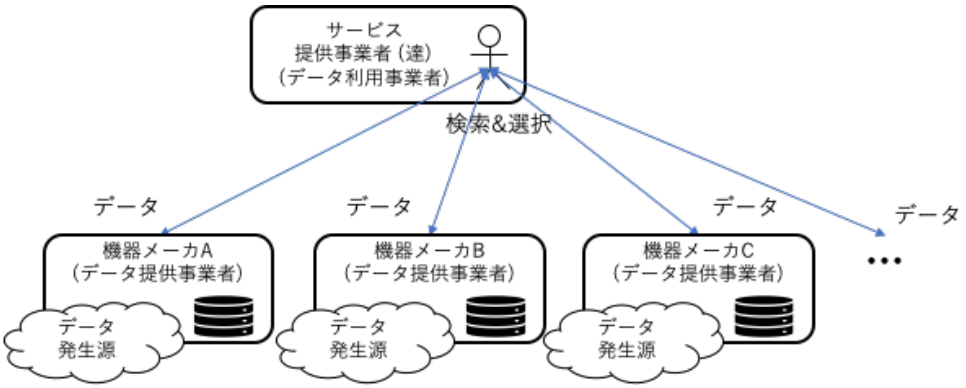


図 3.1　各機器メーカから直接取得

図 3.1 は独立型データ連携手法の動きを示す．図のように，機器メーカのクラウドが多数存在し，各自のデータ発生源と繋がっており，データを収集して蓄積する．サービス提供事業者 (データ利用事業者) は手動で各機器メーカのクラウドをアクセスし，クラウドに実装された様々な検索手段で利用したいデータを検索し，選択と利用をする．

こうしたデータ連携手法では，機器メーカが独自のクラウドを持つために，データ提供

事業者が最大限の自由度を持っている．一方，機器メーカのクラウドが多数存在しているので，データ利用事業者は多数且つ異なるインタフェースやデータ記述方法を直面しないといけない．そのため，これから大量なデータが流通される場合，データ利用事業者が複数の機器メーカクラウドから所望のデータを獲得することが難しい．また，大量なデータ情報から検索してきた結果は人間手動で選択するので，結果の数が一定的な数量以上になると人間による手動選択ができなくなり，データ流通に悪影響を与える．

3.2　集中型

集中型のデータ連携手法では，データとデータのメタデータ両方蓄積するマーケットプラットフォームを中心とするデータ流通手法である．

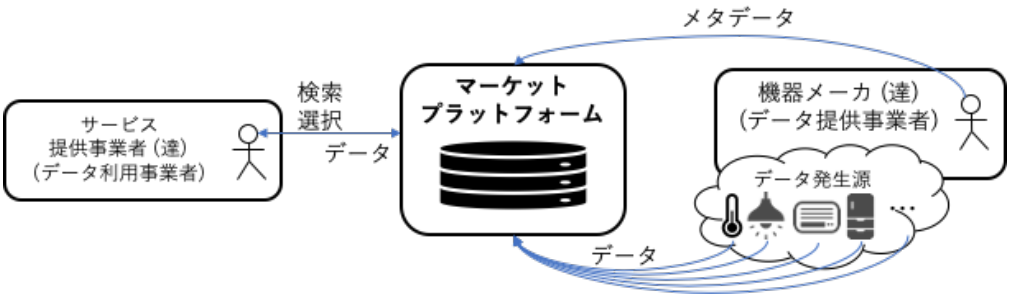


図 3.2　集中型プラットフォームを用いたデータ連携

図 3.2 はマーケットプラットフォームを用いた集中型データ連携の動きを示す．機器メーカはマーケットプラットフォームの設定要求に従い，機器デバイスのデータ宛を設定し，機器デバイスが動き始めてからデータが自動的にマーケットプラットフォームに送り出す．流通したい機器データを利用事業者側が発見できるように，機器メーカがプラットフォームに機器データのメタデータをプラットフォーム規定の記述方式でマーケットプラットフォームに登録する．データ利用事業者側はマーケットプラットフォームにアクセスし，プラットフォームにある複数の機器メーカのデータを同一なインタフェースで検索することができ，取得できる．

こうしたデータ連携手法では，同一なインタフェースと検索方式で複数の機器メーカのデータを発見できる．また，機器メーカがわずかの操作でデータ流通を実現でき，メーカ側のクラウドが必ずしも必要とせず，機器メーカの負担が小さい．一方，プラットフォームの連携により，データの数が増加し，データ利用事業者は検索してきた結果から適するデータを探すことがさらに難しくなる．そのうえ，マーケットプラットフォームに新しく

登録されたデータ記述がある場合，人間手動で発見しかできず，十分なデータ利活用が難しい．

3.3　カタログ型

カタログ型のデータ連携手法は複数の機器メーカが持っているデータの概要記述をカタログに整理し，データ連携プラットフォームでデータのメタデータだけ流通し，データそのものが別途で様々な手段でデータ利用事業者に伝送する手法である．データカタログとは，データの所在や内容等の概要情報を項目別に記入する書式の総称である．

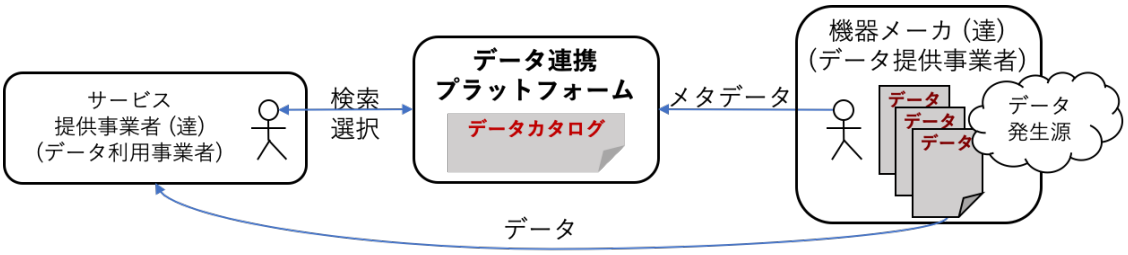


図 3.3　データカタログを用いたデータ連携

図 3.3 はデータカタログを用いたデータ連携の様子を示す．データ提供事業者となる機器メーカはデータ発生源から取得したデータを蓄積し，データのメタデータをデータ連携プラットフォームのカタログに登録する．利用者はプラットフォームにあるカタログを検索し，メタデータに従ってデータそのものを取得して利用する．

こうしたデータ連携手法では，データのメタデータを中心とするので，データ提供事業者がデータの提供方式などを柔軟で行うことができる．一方，データ連携プラットフォームの連携効果により，複数の機器メーカがデータをカタログに登録することができるため，データ利用事業者の検索クエリが大量な結果を取得することが可能ので，既存の人間手動で選択する方式では非常に不効率である．また，カタログの更新を配慮し，利用中データよりデータ利用事業者のニーズに適するデータが登録される場合，人間の手動操作による発見は非常に不効率である．

3.4　まとめ

本章に述べたように、プラットフォームを利用しないデータ連携を含み、サービス提供事業者とデータとデータ情報の提供側の間に手動で行う検索と選択はデータ利用事業者に

対しする主要な問題点である。

具体的な問題点は以下のように示す:

- 流通中データセットを検索した結果の数は人が読めないほど多いので人間が手動で選択するのは困難
- 新しく流通に入ったデータセットに対し、発見できない、効率良いの情報取得が難しい
- より相応しいデータセットがある場合、発見できず、利用できない

第 4 章

システム提案

本章には，提案システムの全体像からシステムのリソース記述モデル，ニーズ記述手法，データカタログを用いた自動選択メカニズム，インタフェースについて説明する．

4.1　システム全体像

提案データ自動選択システムはカタログ型データ連携方式を土台として自動選択の機能，マシンリーダブルとヒューマンリーダブルなインタフェース，マシンリーダブルなリソース記述モデルとニーズ記述モデルをデータ連携プラットフォームに加えるシステムである．こうしたデータ自動選択システムは人間の手動操作が必ずしも必要とせず，既存データ連携プラットフォームの人間手動で選択をする問題点とカタログに新規登録されたデータセットが人間操作しか発見できない問題点を解決できる．

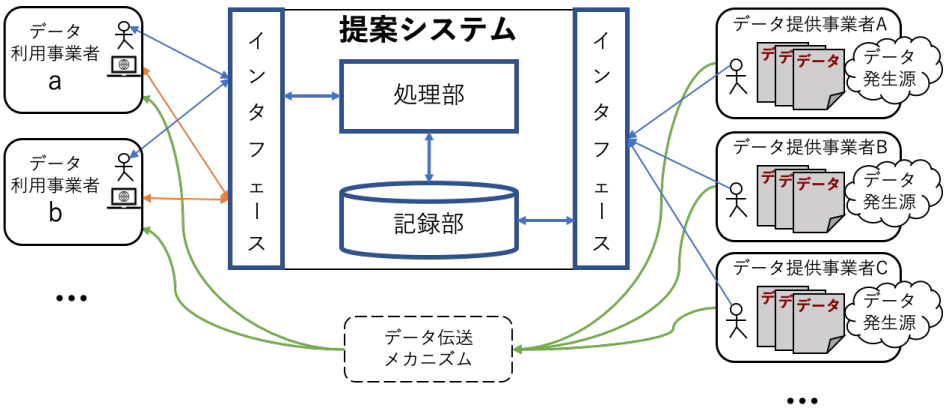


図 4.1　提案システムの全体像

システムの全体像は図 4.1 のように，データ利用事業者となるサービス提供事業者が図の左側にあり，データ提供事業者となる機器メーカが図の右側にあり，中間が提案データ自動選択システムであり，データそのものを転送するためのデータ伝送メカニズムがシステム外にある．データ提供事業者は提供しようとするデータセットを提案システムのリソース記述モデルに従って整理し，システムの提供者向けインタフェースを通じて提案システムに登録する．提案システムの利用者向けインタフェースはマシンリーダブルインタフェースとヒューマンリーダブルインタフェースから構成されるため，データ利用事業者がヒューマンリーダブルインタフェースを通じてカタログを閲覧したり，検索と選択することかでき，マシンリーダブルインタフェースを用いてプログラムによるニーズと一番適するデータセットの情報を自動的に取得できる．利用者向けインタフェースからクエリをする時，提案システムは記録部に蓄積されているカタログを検索し，その結果を処理部の自動選択経由してインタフェースに返す．

本章における以下のセクションはシステムを構成する四パーツについて説明する:

- セクション 4.2
  - 記録部とデータ提供事業者 (機器メーカ) 向けインタフェースに利用されるリソース記述モデル
- セクション 4.3
  - 処理部のメインとなる近似度計算を用いたデータ自動選択
- セクション 4.4
  - サービス提供事業者 (データ利用者) 向けインタフェースに存在する利用者のニーズを記述でき，フレキシブルなニーズ記述モデル
- セクション 4.5
  - システムのインタフェース

4.2 マシンリーダブルなリソースモデル

4.2.1 JEITA スマートホームデータカタログ

JEITA スマートホームデータカタログは生活データを活用したスマートホーム関連市場のあらゆる参入事業者が共通で理解できるメタデータ記述方法として提案される．

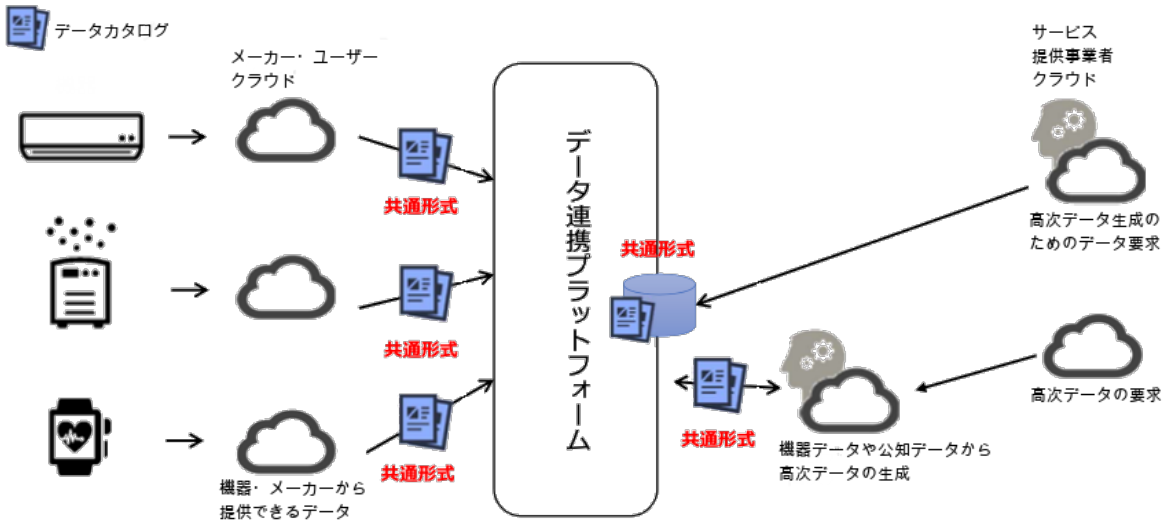


図 4.2　JEITA データカタログの役割 [3]

図 4.2 は JEITA スマートホームデータカタログの役割を示す．左からデバイス，機器メーカ，データ連携プラットフォーム，サービス提供事業者 (データ利用者) がある．機器メーカは自社のデバイスと繋がっており，提供できるデータを整理し，その概要を共通形式でデータ連携プラットフォームに登録する．サービス提供事業者はデータ連携プラットフォームに使われる共通形式に従って検索と選択をし，別途様々な手段でデータを取得し，利用する．そのように，JEITA スマートホームデータカタログは機器メーカ (データ提供事業者)，データ連携プラットフォーム，サービス提供事業者 (データ利用事業者) 三者の間の共通形式としてデータ流通を促進する．

JEITA スマートホームデータカタログはスマートホームにおける高度な記述能力を持つため，利用者がジオグラフィカルな位置情報や具体的なセンサ番号などに縛られず，取得したいデータの観測対象や観測特性などを利用してデータ発見ができる．その他，データセット取引のための値段や有効期限や利用制限なども記述されている．

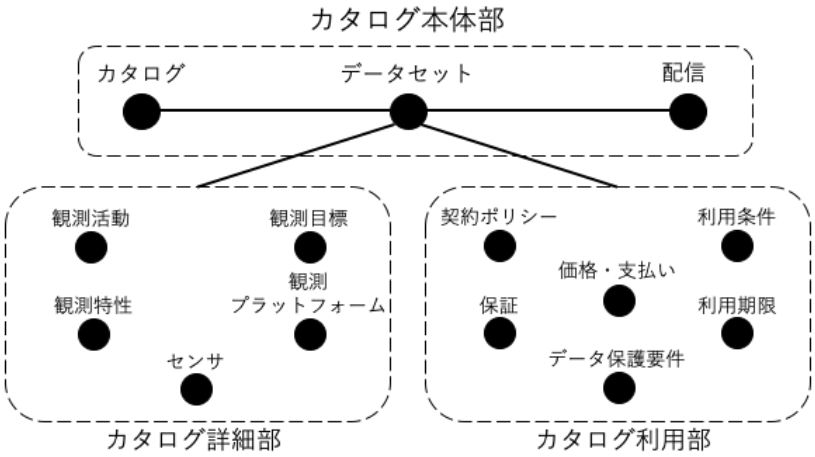


図 4.3　JEITA スマートホームデータカタログのイメージ (クラスレベルまで)

JEITA スマートホームデータカタログ大枠なイメージは図 4.3 のように示す．カタログは本体部と詳細部と利用部の三部分に分けられ，カタログ本体部の DataSet が実際に提供できるデータを表示し，検索の目的となる．左下にあるカタログ詳細部はデータの観測に関する情報が記述されている．右下にあるカタログ利用部はデータ流通に関する契約要件と利用要件が記述されている．

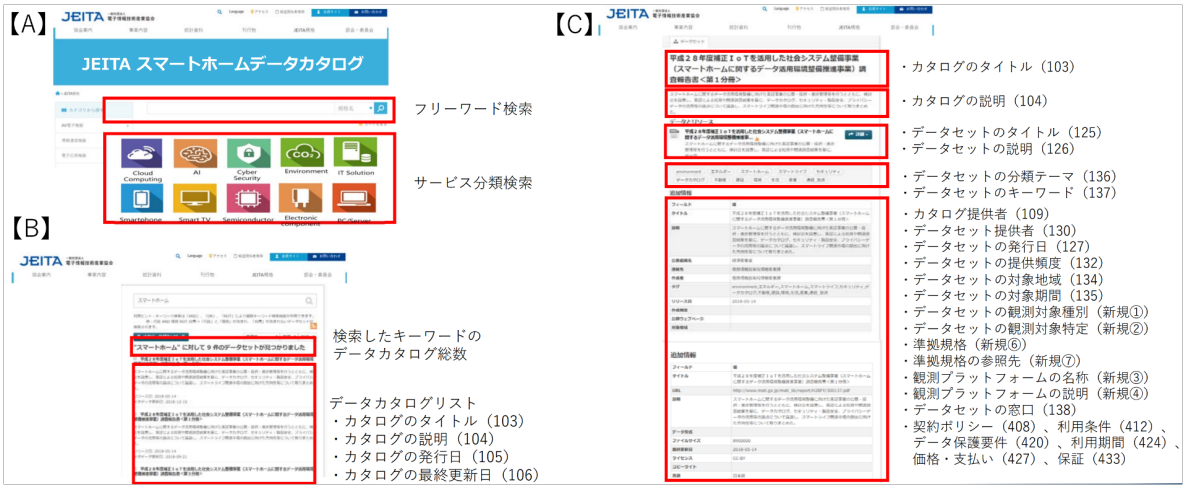


図 4.4　JEITA スマートホームデータカタログの利用イメージ

図 4.4 は JEITA スマートホームデータカタログ定義書に予想した利用シーンを示す．カタログは人間が読むためのものと設計され，分類選択やフリーワードなどの方法を用いて検索し，その結果の概要を並んで表示する．結果をクリックすると該当結果のカタログ

に記述されている全部の内容がテーブルで利用者に表示される．

4.2.2 JEITA スマートホームデータカタログオントロジー

分類選択やフリーワード検索は語彙の意味を配慮しないので，自動選択の目的に利用すると不適切な検索結果が選ばれる可能性がある．故に，マシンリーダブルな，語彙を配慮できる JEITA スマートホームデータカタログの定義形式が必要である．

JEITA スマートホームデータカタログの定義は自由記述などの機械の読めない定義が多いので，マシンリーダブルな，セマンティック的な形で利用できるようにするためには一定程度の改造を加えることが必要である．

クラスの日本語名	クラスの英語名
カタログ	Catalog
データセット	Dataset
配信	Delivery
カタログ詳細部	CatalogDetail
観測活動	ObserveAction
観測対象	ObserveTarget
観測特性	ObserveFeature
観測プラットフォーム	ObservePlatform
センサ	Sensor
カタログ利用部	CatalogUsage
契約ポリシー	ContractPolicy
利用条件	TermsOfUse
価格・支払い	PricePayment
保証	Warranty
利用期間	ExpirationDate
データ保護要件	ProectRequirements

表 4.1　JEITA スマートホームデータカタログ項目名の日本語と英語対照 (クラスレベルまで)

マシンリーダブルなモデルを定義しやすいために，JEITA スマートホームデータカタログの各項目名を一意で英語に翻訳して利用する．翻訳された英語名を利用し，以下の JEITA スマートホームデータカタログオントロジーを定義する．英語の名称と元々の日本

語名称の対照は表 4.1 で表示される．

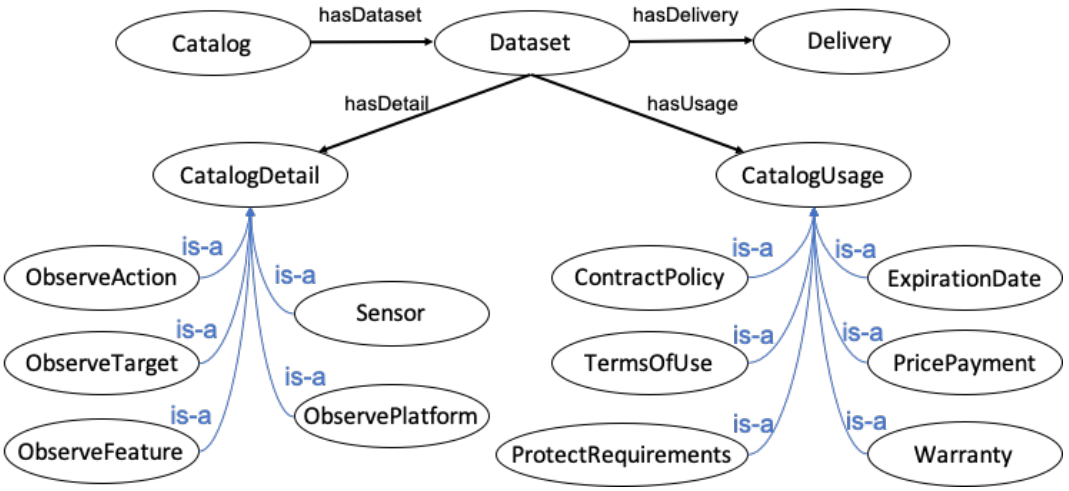


図 4.5　JEITA スマートホームデータカタログオントロジーのグラフ表現 (クラスレベルまで)

図 4.5 は JEITA スマートホームデータカタログ項目定義書によって作られたオントロジーの概要を示す．Catalog クラスはあるデバイスクラウドにある流通できるデータセットの集合体とし，hasDataset で複数の Dataset を有することを表す．Dataset は詳細情報 (CatalogDetail)，利用条件 (CatalogUsage)，配信方式 (Delivery) の情報を持っており，中に CatalogDetail と CatalogUsage は抽象クラスであり，下にある具体的な記述クラスが抽象クラスを継承する．

こうしたオントロジーによってフラットなリソース記述方式で，観測の視点からリソースを記述し，加えデータセットの利用規定に関する記述があり，データ流通の契約に関する情報も記述できる．また，「観測」に関する定義は幅広く使われているオントロジーの観測に関する記述と共通できるため，マッピングモジュールを利用して SSN などデバイス中心の標準モデルとマッピングできる．

4.3　近似度計算を用いたデータ自動選択

自動選択における近似度計算は C.Perera 等により提出されたランキング用近似度計算手法 CPWI(Comparative-Priority Based Weighted Index)[28] を土台として数値の許容範囲を加えた計算手法を提案する．

$$(CPWI) = \sqrt{\sum_{i=1}^{n} \left[W_i \left(U_i^d - S_i^a \right)^2 \right]} \tag{4.1}$$

CPWI は式 4.1 のように示す．W がデータ利用者によって設定された評価パラメータの加重値，U がデータ利用者の理想値，S が実際に評価されたいるデータセットの評価パラメータ数値である．添字については $_i$ が評価パラメータの番号，^{d} が理想値の設置していない場合使うデフォルト値，^{a} が評価対象の番号．式にある各パラメータはリソース記述モデルに数値で表せる項目と定義される．

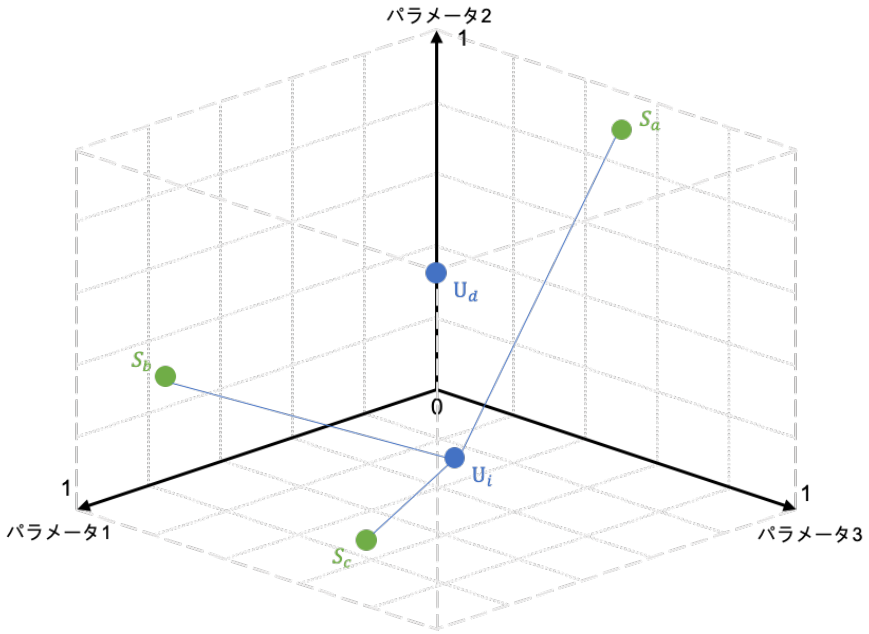


図 4.6　CPWI 計算のイメージ

図 4.6 のように，U_d はデフォルトの理想値，各パラメータの最大値と設定されている．U_i は利用者が設定した理想値，S_a と S_b と S_c は実際に評価対象となるものである．デフォルト理想値 U_d は理想値の設定されていない場合を配慮するための設定ので，実際に計算をする時 U_d と U_i が一個しか利用されない．CPWI は評価対象の各パラメータの正規化値を計算し，理想値との加重値付きユークリッド距離を計算する．

しかし，理想値と重みだけの近似度計算が自動選択の目的に用いられた結果は許容範囲外の数値が選択され易い．例え図 4.7 と表 4.2 が示したように，パラメータだけ見ると S_2 と S_3 が U に近いが，加重値に偏りが大きな数値が与えられた場合は，加重値大きなパラメータが理想値に近くと他のパラメータが無視されることに至る．S_1 が CPWI の計算によって理想値に一番相応しいものとなるが，P_1 の値が理想値に離れすぎ，実は S_3 の方が一番相応しい．

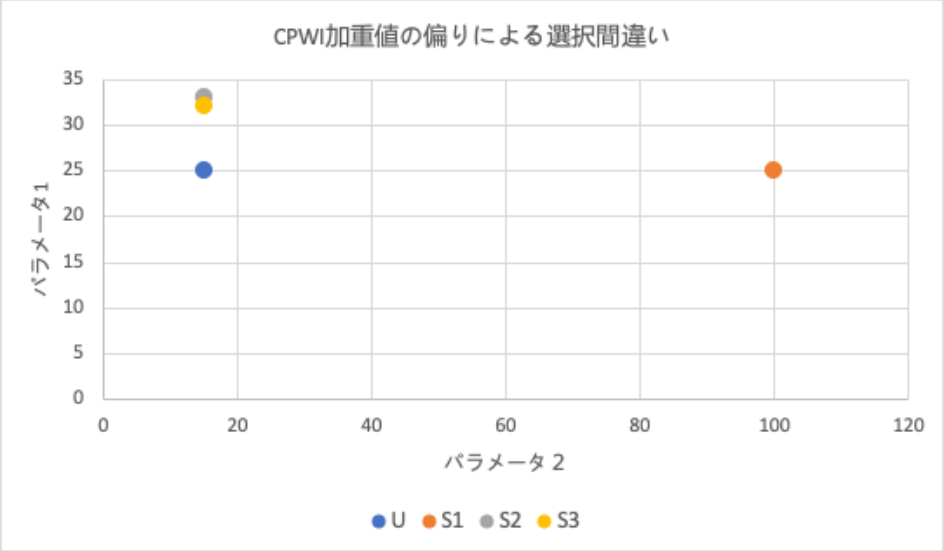


図 4.7　CPWI 加重値を配慮しない場合のイメージ

	加重値	理想値	S_1	S_3	S_3
P_1	3	15	100	15	15
P_3	6	25	25	33	32
結果	-	-	1.732050808	2.449489743	2.143303525

表 4.2　CPWI に偏り大きな加重値と偏り大きなパラメータを与えた場合の判断

本研究は CPWI に許容範囲の制限を加えた近似度計算手法を提案する．

$$CPWI = \begin{cases} \sqrt{\sum_{i=1}^{n} \left[W_i \left(U_i^d - S_i^a \right)^2 \right]}, & \text{if } U_i^{min} < S_i < U_i^{max}; \\ +\infty, & \text{otherwise.} \end{cases} \tag{4.2}$$

こうした計算により，利用者が各評価パラメータに対して加重値と理想値と許容範囲の設定を使い，利用者の許容範囲内にある選択肢の近似度順値の小さい順で並び替えでき，一番近似のデータセットの情報を利用者に送ることが可能と考えられる．

図 4.7 と表 4.2 の数値にパラメータ許容範囲を設定してから図 4.8 と表 4.3 のようになる．許容範囲によって範囲を超えたパラメータを持つ評価対象は理想値との重み付きユークリッド距離を無限大となり，最終結果にならない．

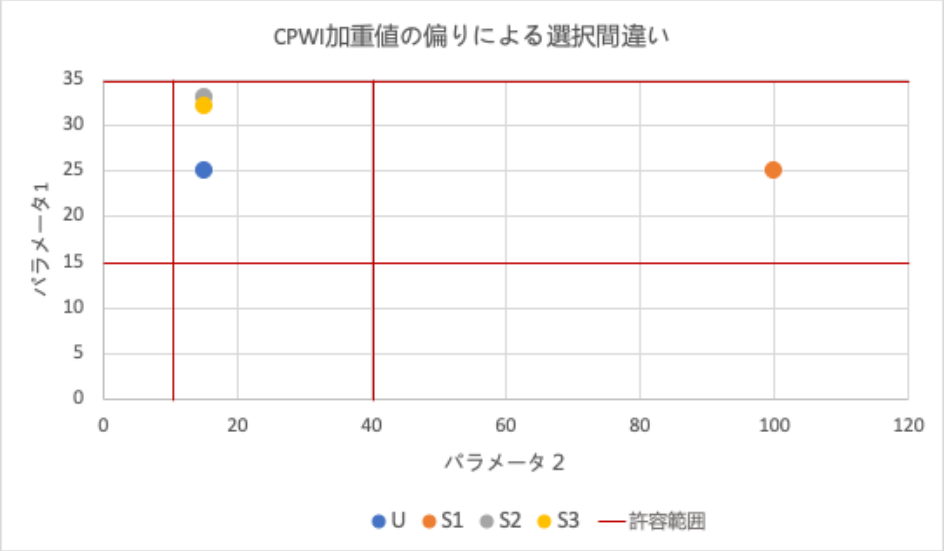


図 4.8　CPWI に許容範囲設定したイメージ

	加重値	理想値	S_1	S_3	S_3
P_1	3	15	100	15	15
P_3	6	25	25	33	32
結果	-	-	∞	2.449489743	2.143303525

表 4.3　CPWI に大きな加重値と許容範囲を与えた場合の判断

4.4　ニーズ記述モデル

利用者のニーズを記述ために柔軟な記述手法が欠かせない．柔軟な記述をするために簡単なニーズと複雑なニーズを両方記述可能な構成が必要と考えられる．各属性と属性間の関係を論理式で表示するニーズ記述モデルを提案した．

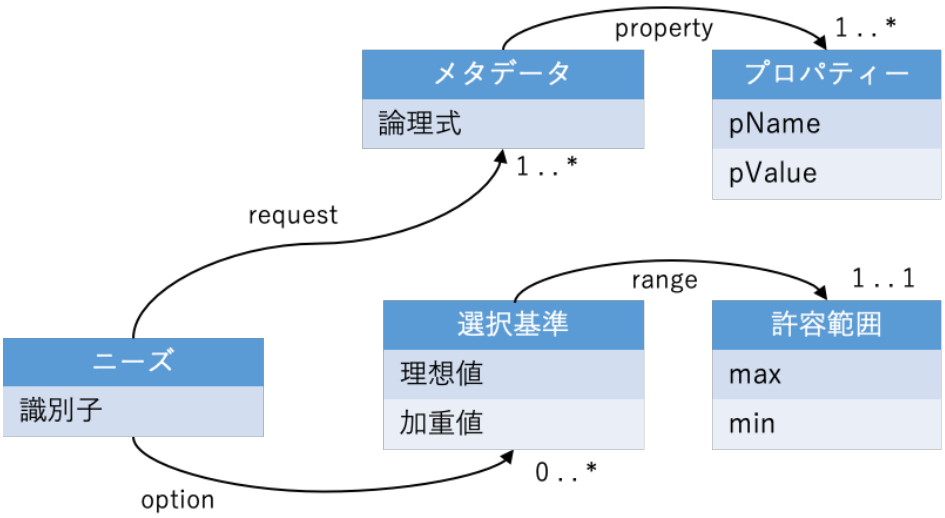


図 4.9　ニーズ記述モデル

図 4.9 はニーズ記述モデルの構造を示す．提案ニーズ記述モデルは大きく二部分にからなる．上にあるリクエスト部は，メタデータとメタデータ間の関係を記述する部分である．中にプロパティーを記述する部分はリソース記述モデルにあるプロパティー名と要求値が一ペアとしてカストマイズな記号を付く．論理式の部分については，JSON Schema の語彙でツリー構造を記述可能な特性 [29] を用い，[allof, anyof, not] で論理記号の [∧(AND), ∨(OR), ≠(NOT)] を表示し，プロパティー部分にツリー構造で論理式を表す．例えば：

$$(A \wedge (B \vee C)) \vee (D \wedge (E \vee F) \wedge G) \tag{4.3}$$

をツリー構造にすると，図 4.10 のようになる．

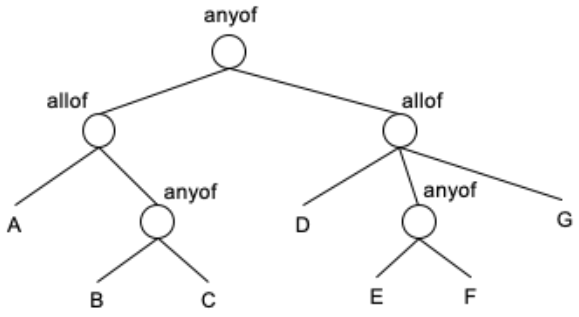


図 4.10　論理式をツリーで表す

こうしたツリー構造がしたのように JSON で表示できる．

```
{
    "anyof":[
        {"allof":["A", {"anyof":["B", "C"]}]},
        {"allof":["D", {"anyof":["E", "F"]}, "G"]},
    ]
}
```

オプションの部分は提案の自動選択手法に合わせ，理想値，加重値，許容範囲が記述されている．

4.5 インタフェース

提案システムのインタフェースはマシンリーダブルインタフェースとヒューマンリーダブルインタフェースから構成される．

マシンリーダブルインタフェースは REST API の POST を利用し，JSON でシリアライズされたニーズ記述モデルを用いてデータ利用事業者のニーズを設定する．マシンリーダブルインタフェースの戻り値は JSON でシリアライズされたデータ利用事業者のニーズと一番適しているデータセットのメタデータ，その中身は JEITA スマートホームデータカタログオントロジーと同様.

ヒューマンリーダブルインタフェースはヴェブブラウザに依存し，グラフィカルなユーザインタフェースである．条件設定としては，既存のフリーワード入力やタッグ分類選択などを利用せず，JEITA スマートホームデータカタログオントロジーの定義に従い，中の項目と存在いている値を選択し，論理式作成し，設定する．また，人間に読むためのインタフェースと定義されるので，処理部の処理結果の数量もカスタマイズで定義できる．クエリ結果は利用者に設定された数量のデータセットの情報をニーズと近い順に並び，ニーズに設定された項目以外の情報をテーブルで表示する．

第 5 章

実装

本章には，システムの実装について説明する．

5.1　システム実装の概要

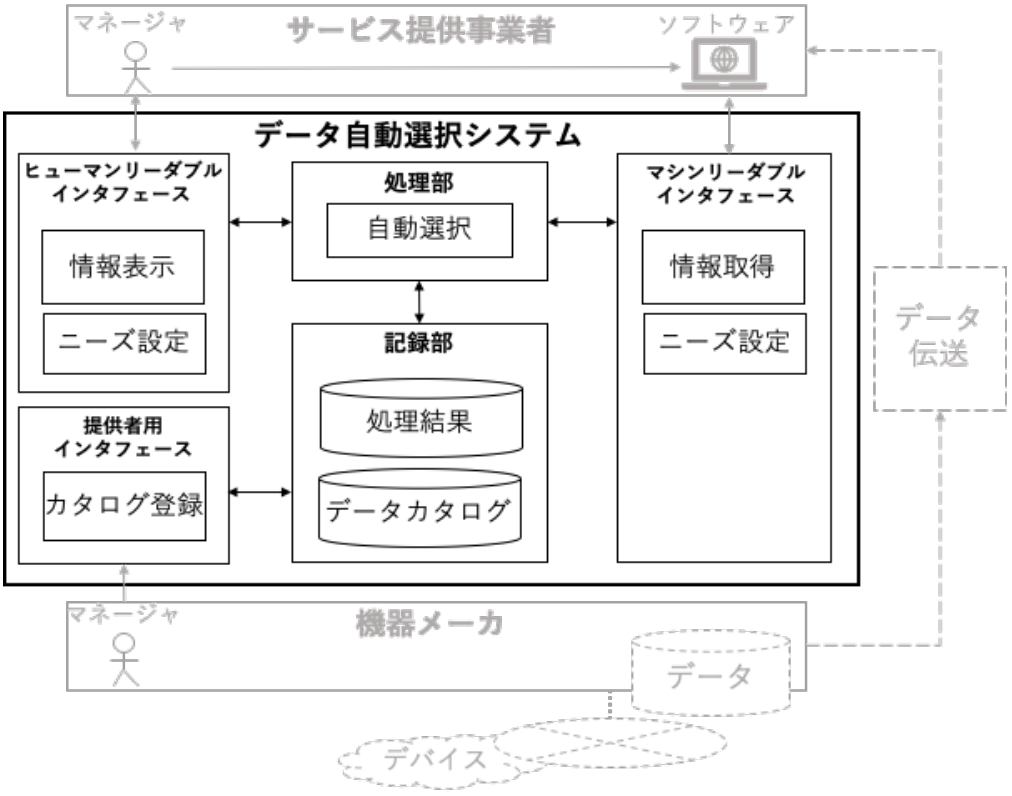


図 5.1　提案システムの実装概要図

提案システムの開発は Python を用い，システムの REST インファフェースを提供するために Flask という軽量化フレームワークを利用する．

図 5.1 システムの実装概要を示す．記録部に処理結果とデータカタログの二個のベースが実装される。カタログデータベースはリソースの記述蓄積し，セマンティック的なデータセット記述を扱うために，Apache の Fuseki TDB(Triple DataBase)[30] を利用する．また，処理結果に関する情報を記録するために NoSQL の MongoDB を用いる．マシンリーダブルインタフェースとヒューマンリーダブルインタフェースは同様にニーズ設定と情報を利用者側に返す機能を持つ。

提案データ自動選択システムの流れは次にようになる．

1. データ提供事業者が流通させようとするデータセットを整理し，JEITA スマートホームデータカタログの形式でシステムに登録する．
2. データ利用事業者がヒューマンリーダブルインタフェースでニーズを設定し，データセットを検索し，利用する．
3. 複数のデータ提供事業者により，データカタログにデータセットの情報を更新続ける．
4. 新しくデータカタログに登録されたデータセットの状況を把握するために，データ利用事業者がエージェントにパラメータを設定し，一定時間間隔でニーズに一番相応しいデータセットの情報を取得し続ける．

リスト 1 番目のデータセット登録は事前準備としてシステムにある程度の数のデータセット情報を蓄積させる．3 番目と 4 番目は非同期で動き続いており，情報を交換する．

5.2 リソースモデルの実装

モデルはベースモデルとし，リソースの記述と組み合わせ流ことができる．提案システムにおいては，自動選択の評価パラメータと JEITA スマートホームデータカタログオントロジーと組み合わせ，図 5.2 のようになる．

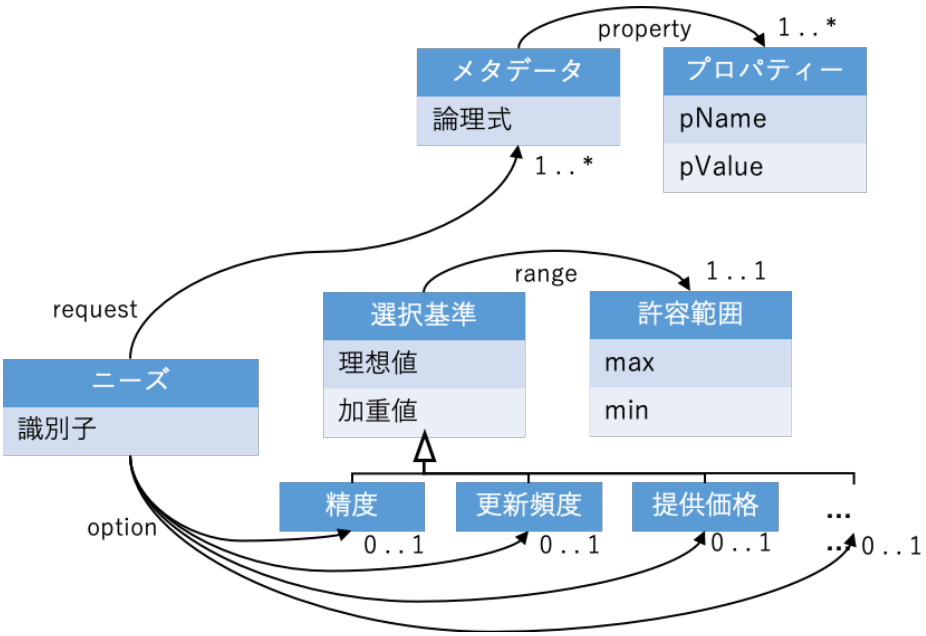


図 5.2　提案システムにおける記述モデル

JEITA スマートホームデータカタログオントロジーに記述方式と記述変換を合わせ，リクエスト関係と繋がっている部分は論理式と JEITA スマートホームデータカタログオントロジーに定義された内容で構成する．オプション部分には近似度計算を用いたデータ自動選択計算と合わせて評価パラメータの精度，更新頻度，提供価格三基準が記述される．

こうした要求記述では，人間とマシンが両方読むことができ，シンプルなニーズと複雑なニーズを記述できる．

5.3　プログラム上の実装

提案システムの実装における環境は以下のように示す．

- OS：macOS Catalina version 10.15.5
- メモリ：16GB
- プロセッサ：1.6GHz Dual-Core Intel Core i5
- 開発言語：Pyhton 3.7.5
- システム開発環境：vsCode

5.3.1　検索と記述変換

TDB の検索は SPARQL という検索言語を使う必要があり，ニーズ記述モデルのリクエスト部分を SPARQL[31] に変換することが必要である．

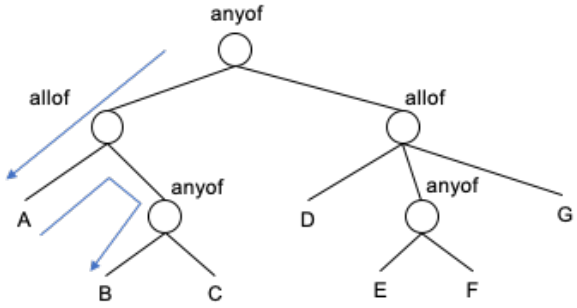


図 5.3　論理式のツリー表現の変換

図 5.3 のように，ツリー構造で表現した論理式を深さ優先探索を用い，一つのノードを解析し，各子ノードに対して再帰的な方式でトラバースをする．語彙の変換については，JSON schema の allof は ∧ と相当するので中の内容を SPARQL に直接並び，anyof は ∨ と相当するので中の内容を SPARQL の UNION に入れ，not は ! と相当するので中の内容を SPARQL の FILTER に入れる．実装されたアルゴリズムは 1 のように示す．

Algorithm 1 論理式で記述されたプロパティーを SPARQL に変換

Require: $releDict$ 論理式, $propertyDict$ プロパティー

Ensure: $sparqlQuery$SPARQL 検索文字列

```
1:  function releDecode(releDict, propertyDict, part)
2:      relKeyList ← releDict.keys
3:      newpropList
4:      newdecodedList
5:      for all key ∈ relKeyList do
6:          rellen ← releDict.length
7:          counter ← 0
8:          for all elem ∈ releDict.get(key) do
9:              if elem.type == None then
10:                 decodedList.add(releDecode(elem, propertyDict, part))
11:             else
12:                 if elem.type == String then
13:                     propList.add(elem)
14:                 end if
15:             end if
16:             counter + +
17:             if counter == rellen then
18:                 return combineParts
19:             end if
20:         end for
21:     end for
22: end function
```

5.3.2 近似度計算

近似度計算については，検索結果のデータセットリストにある各パラメータを設定された許容範囲と判断し，許容範囲外のパラメータを持つデータセットをリストから削除する．次に，各パラメータの最大値と最小値を計算し，データセットのパラメータを正規化し，CPWI 計算で加重値付きユークリッド距離を計算する．実装したアルゴリズムは 2 のように示す.

Algorithm 2 近似度計算の処理

Require: $dsList$ 検索結果リスト，$option$ 評価基準

Ensure: ds データセット情報

function ApproximationCalculation($dsList, option$)
    $dsListInRange \leftarrow rangeFilter(dsList, option)$
    $p_i^h \leftarrow gethightest(dsListInRange)$
    $p_i^l \leftarrow getlowest(dsListInRange)$
    $dsListNormalized \leftarrow getnormalized(dsListInRange, p_i^h, p_i^l)$
    $distList \leftarrow newlist$
    if $dstaset \in dsListNormalized$ **then**
        $dist \leftarrow \sqrt{\sum_{i=1}^{3} \left[W_i \left(ideal_i - p_i\right)^2\right]}$
        $distList.add(dist)$
    end if
    $dsListSorted \leftarrow sortBtDist(dsListInRange, distList)$
    return $dsListSorted[0]$
end function

第 6 章

評価と結果

本章では，提案システムについて動作確認をし，評価とその結果を示す．

6.1　動作確認

6.1.1　環境設定

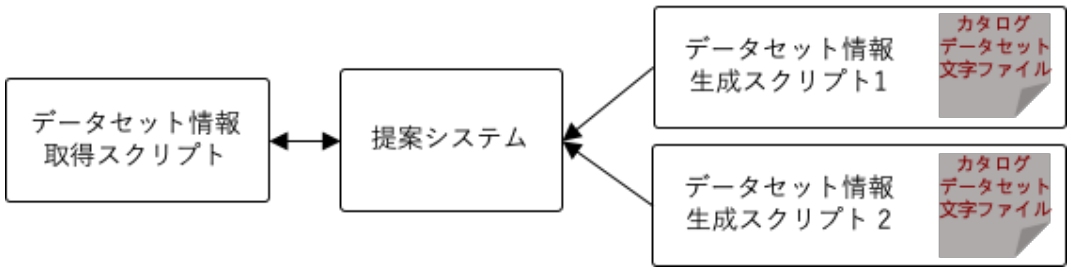


図 6.1　システム動作確認のイメージ

システムの動作確認は図 6.1 のようにする．一番右にあるデータセット情報生成機構はデータ提供事業者のデータセット情報登録の機能を実行するプログラムであり，一番左にあるデータセット取得スクリプトはデータ利用事業者の行動をし，一定的な時間間隔でデータセットの情報を取得する．

初めに，事前準備として iHouse の実デバイスに基づいてカタログを作成し，システムに登録する．データセット情報生成機構は確率分布に従う数値の時間を経過し，ポワソン分布に従う数値をデータセットの精度/提供価格/更新頻度/有効期間に付き，データセットの情報をカタログに登録する．

データセット取得スクリプトにニーズを設定し，マシンリーダブルインタフェースを通じてデータセットの情報を取得し，取得したデータセットの情報を記録する．

6.1.2 シナリオ

提案システムは精度，更新頻度，提供価格の三パラメータに対して理想値，加重値，許容範囲の三パラメータを加え，データセットが利用者の取得したいものであるかどうかを計算する．

動作確認をするため，データセット取得スクリプトに以下のような評価パラメータを設定する．

	理想値	加重値	許容範囲 (最小値)	許容範囲 (最大値)
精度	0	6	0	0.5
更新頻度	45	2	30	100
提供価格	0	1	0	35

表 6.1　動作確認のためのパラメータ設定

表 6.1 のように，スクリプトの取得したいデータセットは：

精度の理想値は (±)0，重みは 6，許容範囲は [0, 0.5]

更新頻度の理想値は (±)45，重みは 2，許容範囲は [30, 100]

提供価格の理想値は (±)0，重みは 1，許容範囲は [0, 35]

と設定する．

データセット提供者となるデータセット情報生成機構①とデータセット情報生成機構②はポアソン分布に従い数値をパラメータと生成数量，登録間隔に設定する．

データセット情報生成機構①の設定

データセット登録動作の時間間隔設定設定:$\lambda = 15$

データセット生成数量の設定:$\lambda = 500$

生成精度の分布設定:$\lambda = 50$ で生成した数値を 1000 で割る

生成提供価格の分布設定:$\lambda = 400$

生成更新頻度の分布設定:$\lambda = 320$

生成有効期間の分布設定:$\lambda = 2500$ で生成した結果を生成時点のタイムスタンプに加える

データセット情報生成機構②の設定

データセット登録動作の時間間隔設定設定:$\lambda = 35$

データセット生成数量の設定:$\lambda = 500$

生成精度の分布設定:$\lambda = 200$ で生成した数値を 1000 で割る

生成提供価格の分布設定:$\lambda = 30$

生成更新頻度の分布設定:$\lambda = 50$

生成有効期間の分布設定:$\lambda = 2500$ で生成した結果を生成時点のタイムスタンプに加える

6.1.3 結果

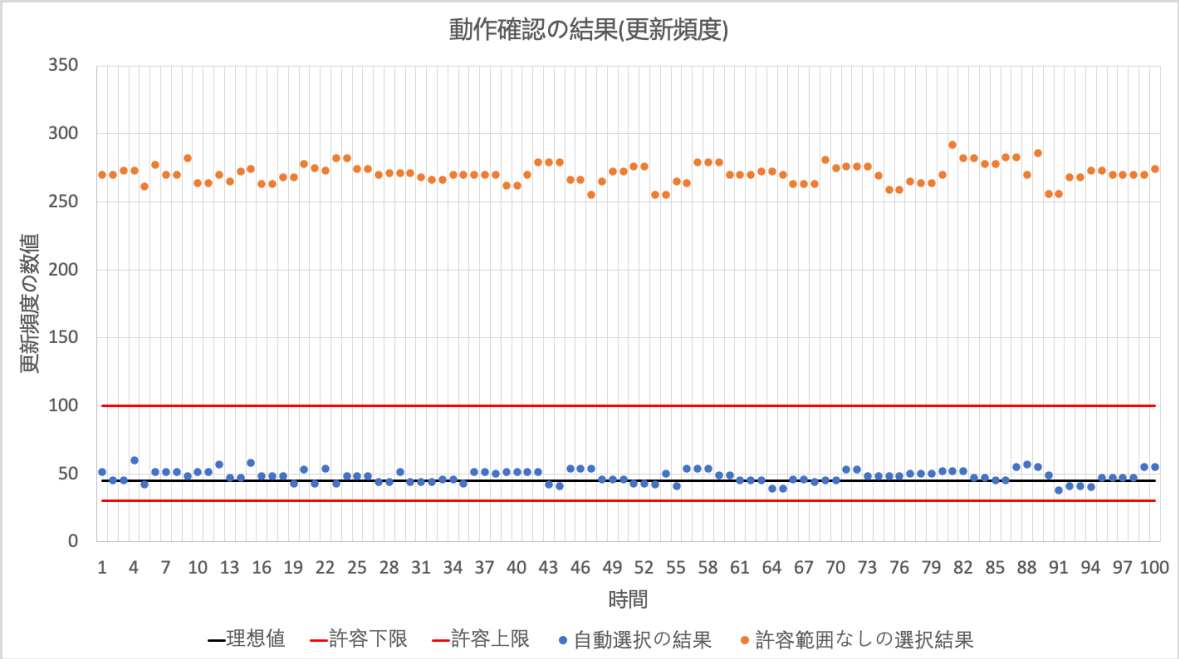


図 6.2　動作確認結果 (提供頻度，重み=2)

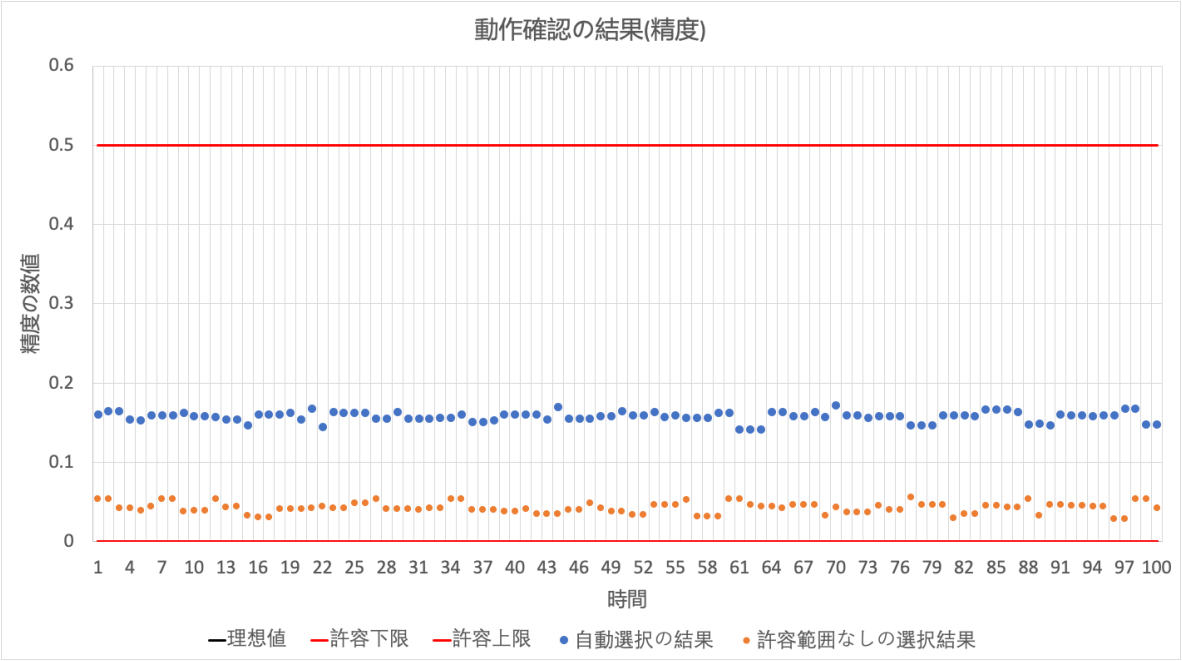


図 6.3　動作確認結果 (精度，重み=6)

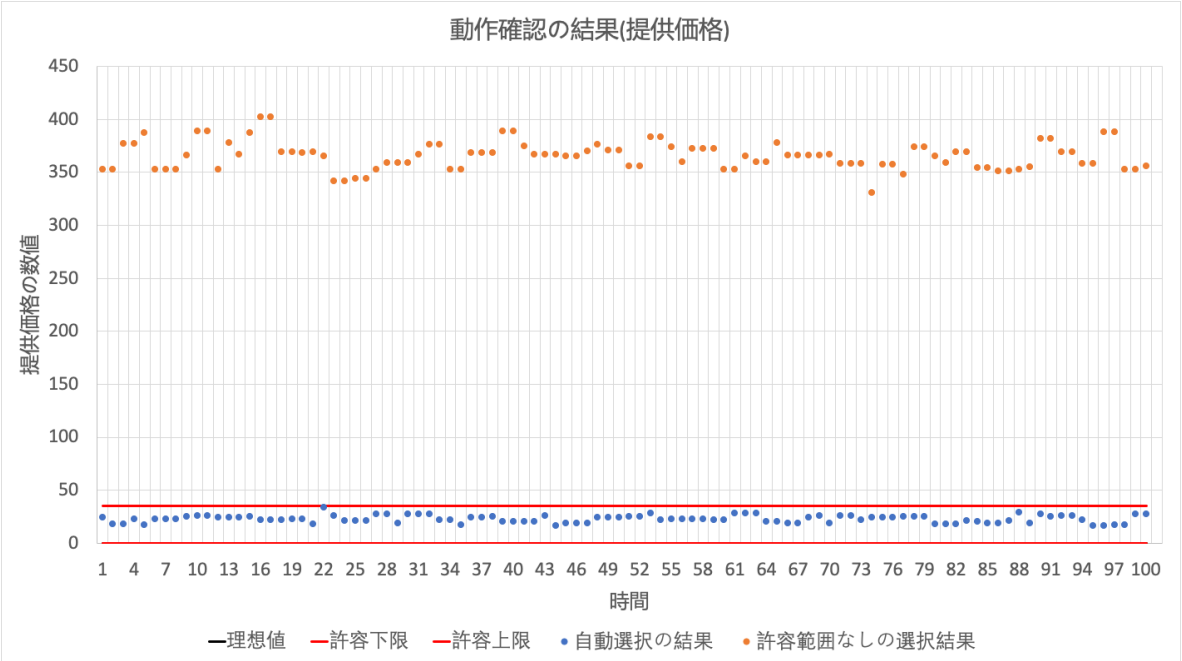


図 6.4　動作確認結果 (提供価格，重み=1)

図 6.2 と図 6.3 と図 6.4 はシステムの動作確認の結果を示す．図の中に，紅色の線は許容範囲の上限と下限，黒色の線はパラメータの理想値．

結果から，許容範囲を利用する場合は重みの偏りによって選択結果が重みの高いパラメータしか重視することを解決でき，自動選択の結果が全て許容範囲内にある．

6.2　自動選択と人間選択の比較評価

提案データ自動選択システムを利用すると利用しないの場合，データ選択における時間的なコストを用いてシステムの効果を評価する．

6.2.1　環境設定

図 6.5　自動選択と人間選択の比較評価システムイメージ

比較評価の評価用システムは図 6.5 が示す．左側にあるのはデータ利用者事業者の役割を果たすデータセット情報取得スクリプト，右にあるのはデータ提供事業者の役割を果たすデータセット情報生成機構である．左側のスクリプトは提案システムのマシンリーダブルインタフェースにニーズを送り，データセットを取得する．右側のデータセット情報生成機構は短い時間間隔で大量なデータセットを提案システムに登録する．

6.2.2　シナリオ

データセット情報生成機構はポアソン分布に従い数値をパラメータと生成数量，登録間隔に設定する．データセット情報取得スクリプトは 10 秒間隔で 100 回マシンリーダブルインタフェースに要求を送り，情報を取得する．また，自動選択機能と同時に選択前の結果の数を記録する．この二つの数値により，提案自動選択システムを用いた情報取得の効率上昇について評価する．

6.2.3　結果

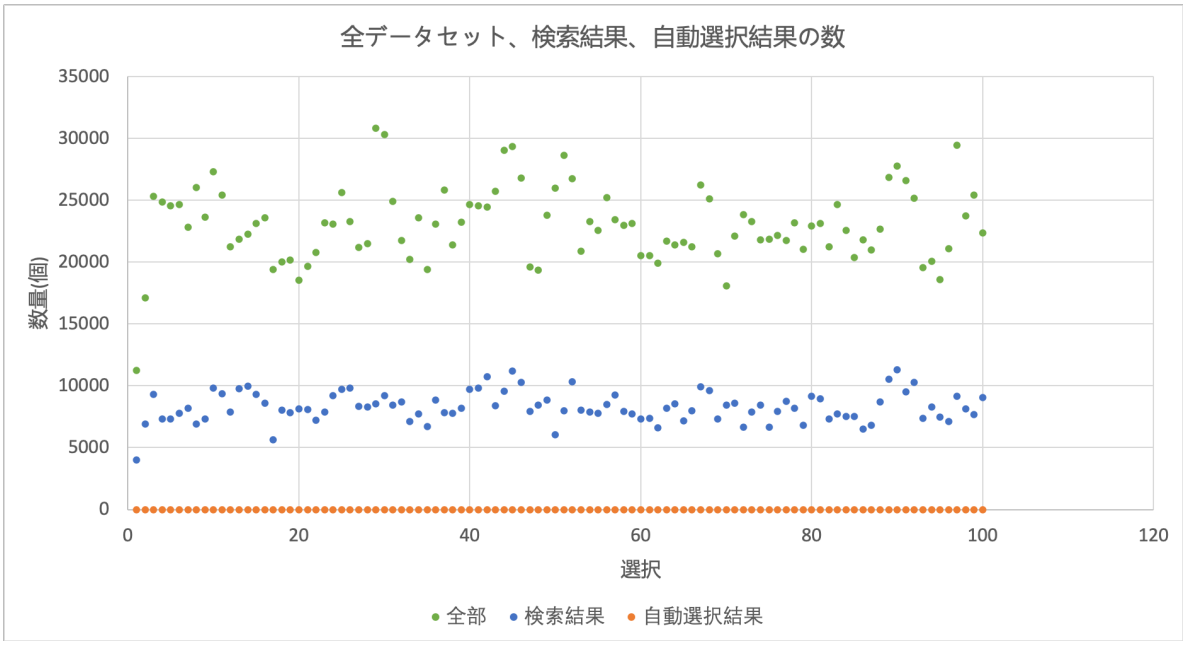


図 6.6　自動選択を利用すると利用しないの比較評価 (結果の数)

図 6.6 は提案自動選択システムを利用する場合と利用しない場合利用者側に返した結果の数量をしめす．一番上の緑色の点は取得時点のカタログに登録されたデータセット情報の数量，したに青色の点は提案自動選択システムを利用しない場合に返したデータセット情報の数量，一番したにある橙色の点は提案自動選択システムを利用する場合に返したデータセット情報の数量．検索結果の数量から提案自動選択システムは取得したデータセットの数を大幅に削減することができる．

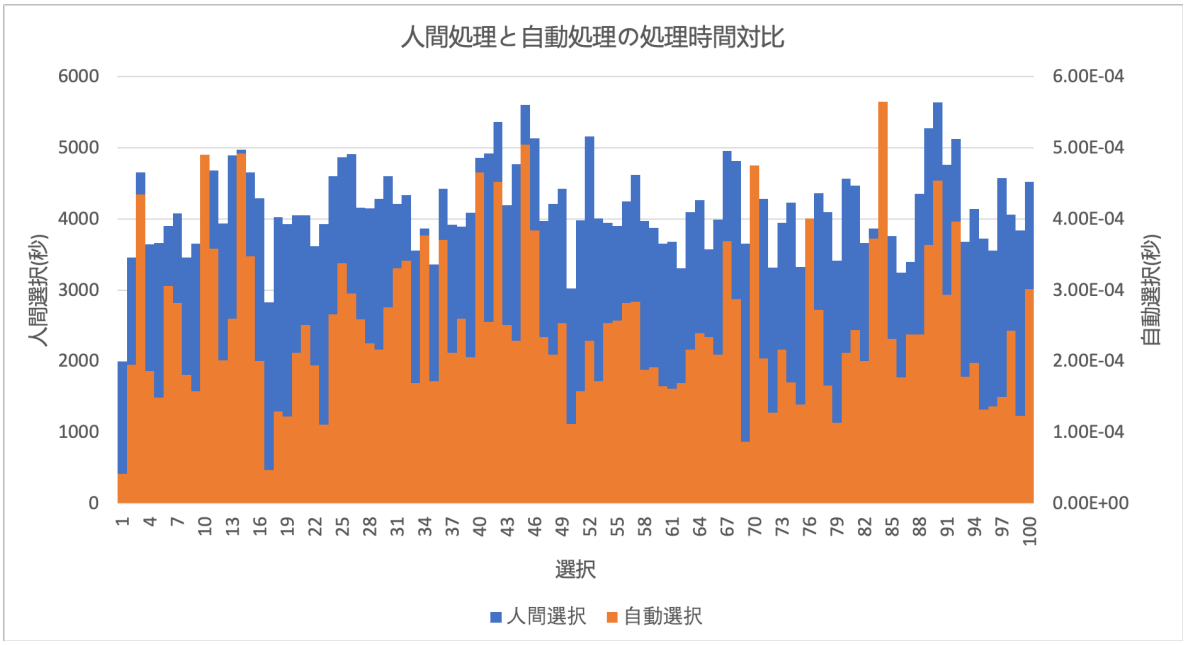


図 6.7　自動選択と人間選択の比較評価 (処理時間)

図 6.7 は図 6.6 のデータセット情報を処理するための時間コストである．人間選択の処理時間は 1 データセットあたり 0.5 秒と計算し，青色の棒で表示され，左側の軸でその数値を表す．橙色の棒が表示するのはシステムによる記録された自動選択にかかった時間，右側の軸で数値を表す．軸だけ見ると 10^4 の差が存在する．

こうした結果により，提案データ自動選択システムは自動選択機能を用いて 100% 近くの処理時間を削減でき，手動選択の不効率の問題点を解決できると考えられる．

第 7 章

おわりに

7.1 まとめ

本研究では，データ流通プラットフォームに大量なデータによる，検索結果が人間で選択できないの数になることを解決するために，データカタログを用いたデータ自動選択システムを提案した．近似度計算に基づいた自動選択手法を用いてパラメータと理想値，加重値，許容範囲で自動的に利用者の要求と一番近い結果を選び出し，利用者に返す．

また，関連研究として情報モデルに関する調査をし，プラットフォーム間の相互操作のための情報モデルの構造と構成を明らかにし，目的や実際の使い道に基づいて相互操作のための情報モデルの階層関係について検討をした．情報モデルに関する調査から明らかにした構造によりデータ流通むけのリソース記述モデルが JEITA スマートホームデータカタログに基づき，マシンリーダブルな JEITA スマートホームデータカタログオントロジーを構築し，システムのリソース記述モデルとして実装した．フレキシブルなニーズ記述と自動選択の基準を記述するため，具体的なモデル縛れないの自動選択ニーズ記述モデルを提案し，提案システムの情報モデルと自動選択手法と合わせてシステムに実装した．

最後に，提案データ自動選択システムの動作を確認し，人間操作によるデータセット選択との比較評価により，提案システムの有効性を示した．

7.2 今後の課題

本研究は大量なデータセットを自動的に選択するシステムを提案し，有効性を証明したが，システムが情報を集まり，RESTfulAPI で提供するようなアーキテクチャに離れておらず，オントロジーやセマンティック的な技術の利点を十分に生かしていない．本研究の

上に，セマンティック的な情報モデルを生かし，数多くのデータ提供事業者クラウドが異なる情報モデルをもち，ある共通的な検索モデルを用いて分散型なデータカタログを構成する．こうした分散型的なアーキテクチャでは，データセットの記述を集まる必要とせず，クエリが来ると連携プラットフォームに記録されているインデックスを利用し，関連のあるデータ提供事業者クラウドに伝送し，結果を収集する．データ連携の効率と便利性が一層向上できると考えられる．

更に，より大規模なデータ連携を目指し，複数のデータ連携プラットフォームを相互に発見可能，取得可能，利用可能な構成に関するシステムの動き，利用者と提供者の動きを明らかにする上，モデリングし，相互利用可能なアーキテクチャを研究する．

謝辞

本研究を遂行するにあたり，多くの方からご助力を頂きました．

主指導教員の丹康雄教授には，日頃の研究議論から，情報通信関連分野全般の認識を深める機会をいただき，ご指導を賜りました．心より感謝申し上げます．

副指導教員のリム勇仁准教授には，日頃のゼミを通して様々な視点から議論ができました．深く感謝申し上げます．

審査委員を引き受けてくださったリム勇仁准教授，篠田陽一教授，BEURAN, Razvan Florin 特任准教授には，新たな視点から示唆に富んだアドバイスをいただきました．厚く感謝申し上げます．

最後に，留学生活を支えてくださった両親に感謝いたします．ありがとうございました．

参考文献

[1] IDC. *The Growth in Connected IoT Devices Is Expected to Generate 79.4ZB of Data in 2025, According to a New IDC Forecast*. URL: https://www.idc.com/getdoc.jsp?containerId=prUS45213219. (accessed: 15.06.2020).

[2] *LIFE UP* プロモーション - 「つながる」と、あなたの一日はこう変わる*!* URL: https://lifeup.cyber-physical.jp/. (accessed: 15.06.2020).

[3] 一般社団法人 電子情報技術産業協会 (JEITA) スマートホー ム部会スマートホームデータカタログ WG. スマートライフ 実現向けた *jeita* スマートホームデータカタログ項目定義書 *v1.0*. URL: https://www.jeita.or.jp/japanese/pickup/category/190314.html. (accessed: 15.06.2020).

[4] 沖電気工業株式会社. *IoT* を活用したインフラモニタリングへ向けたセンサー情報モデル標準化の取り組み. URL: https://starbed.nict.go.jp/research/pdf/20200127_CyrealWS/CyrealWS_02-1.pdf. (accessed: 1.07.2020).

[5] oneM2M. *TS-0012-Base_Ontology-V3_7_3*. URL: http://member.onem2m.org/Application/documentapp/downloadLatestRevision/default.aspx?docID=29554.

[6] *The universAAL Primer - Advanced*. URL: http://depot.universaal.org/Documents/The%20universAAL%20Primer%20-%20Advanced.pdf. (accessed: 25.03.2020).

[7] *The universAAL Primer - Basic*. URL: http://depot.universaal.org/Documents/The%20universAAL%20Primer%20-%20Basic.pdf. (accessed: 25.03.2020).

[8] *OCF Core Specification*. URL: https://openconnectivity.org/specs/OCF_Core_Specification_v2.1.1.pdf. (accessed: 25.03.2020).

[9] *OCF Resource to AllJoyn Interface Mapping Specification*. URL: https://openconnectivity.org/specs/OCF_Resource_to_AllJoyn_Interface_Mapping_v2.1.1.pdf. (accessed: 25.03.2020).

[10] *OCF Resource to BLE Mapping Specification*. URL: https://openconnectivity.org/specs/OCF_Resource_to_BLE_Mapping_Specification_v2.1.1.pdf. (accessed: 25.03.2020).

[11] *OCF Resource to OneM2M Module Class Mapping Specification*. URL: https://openconnectivity.org/specs/OCF_Resource_to_OneM2M_Module_Class_Mapping_Specification_v2.1.1.pdf. (accessed: 25.03.2020).

[12] *OCF Resource to UPlus Mapping Specification*. URL: https://openconnectivity.org/specs/OCF_Resource_to_UPlus_Mapping_Specification_v2.1.1.pdf. (accessed: 25.03.2020).

[13] *OCF Resource to Zigbee Cluster Mapping Specification*. URL: https://openconnectivity.org/specs/OCF_Resource_to_Zigbee_Cluster_Mapping_Specification_v2.1.1.pdf. (accessed: 25.03.2020).

[14] *OCF Resource to Z-Wave Mapping Specification*. URL: https://openconnectivity.org/specs/OCF_Resource_to_Z-Wave_Mapping_Specification_v2.1.1.pdf. (accessed: 25.03.2020).

[15] *OCF Resource to EnOcean Mapping Specification*. URL: https://openconnectivity.org/specs/OCF_Resource_to_EnOcean_Mapping_Specification_v2.1.1.pdf. (accessed: 25.03.2020).

[16] ETSI. *ETSI TS 103 264 V3.1.1*. URL: https://www.etsi.org/deliver/etsi_ts/103200_103299/103264/03.01.01_60/ts_103264v030101p.pdf.

[17] *ETSI GS CIM 006 V1.1.1 (2019-07)*. URL: https://www.etsi.org/deliver/etsi_gs/CIM/001_099/006/01.01.01_60/gs_CIM006v010101p.pdf.

[18] W3C. *Web of Things (WoT) Thing Description*. URL: https://www.w3.org/TR/wot-thing-description/#sec-vocabulary-definition.

[19] *BIG-IoT*. URL: http://big-iot.eu/. (accessed: 30.03.2020).

[20] *Deliverable 4.2.b: Semantic Model for the Application Domain*. URL: http://big-iot.eu/wp-content/uploads/2016/04/D4.2.b_Semantic_Interoperability_Design_for_Smart_Object_Platforms_and_Services_Final_.pdf. (accessed: 15.03.2020).

[21] *Deliverable 3.2.b: Semantic Interoperability Design for Smart Object Platforms and Services*. URL: http://big-iot.eu/wp-content/uploads/2016/04/D3.2.b_Semantic_Interoperability_Design_for_Smart_Object_Platforms_and_Services_Final.pdf. (accessed: 15.03.2020).

[22] *Semantic Models for Testbeds, Interoperability and Mobility Support, and Best Practices V1*. URL: http://fiesta-iot.eu/wp-content/uploads/2017/11/D31-Web-compressed.pdf. (accessed: 20.03.2020).

[23] W3C. *Semantic Sensor Network Ontology*. URL: https://www.w3.org/TR/vocab-ssn/.

[24] *M3 ontology*. URL: http://sensormeasurement.appspot.com/?p=m3. (accessed: 30.03.2020).

[25] *Lightweight Machine to Machine Technical Specification*. URL: http://www.openmobilealliance.org/release/LightweightM2M/V1_0-20170208-A/OMA-TS-LightweightM2M-V1_0-20170208-A.pdf. (accessed: 15.04.2020).

[26] **第１部** *ECHONET Lite* **の概要**. URL: https://echonet.jp/wp/wp-content/uploads/pdf/General/Standard/ECHONET_lite_V1_13_jp/ECHONET-Lite_Ver.1.13_01.pdf. (accessed: 15.04.2020).

[27] *Internet-of-Things Standards*. URL: https://www.soumu.go.jp/main_content/000445921.pdf. (accessed: 15.04.2020).

[28] C. Perera et al. "Sensor Search Techniques for Sensing as a Service Architecture for the Internet of Things". In: *IEEE Sensors Journal* 14.2 (2014), pp. 406–420.

[29] Felipe Pezoa et al. "Foundations of JSON Schema". In: *Proceedings of the 25th International Conference on World Wide Web*. WWW '16. Montréal, Québec, Canada: International World Wide Web Conferences Steering Committee, 2016, pp. 263–273. ISBN: 9781450341431. DOI: 10.1145/2872427.2883029. URL: https://doi.org/10.1145/2872427.2883029.

[30] Apache. *Apache Jena Fuseki*. URL: https://jena.apache.org/documentation/fuseki2/.

[31] W3C. *SPARQL 1.1 Overview*. URL: https://www.w3.org/TR/sparql11-overview/.