

組み立て型のサービス開発を実現するIoT向けイベント処理基盤の提案

山岡 久俊[†] 中川 岳[†] 板倉 宏太[†] 高橋 英一[†] 金政 泰彦[†]

植木 美和[†] 松本 達郎[†] 武 理一郎[†]

[†] 株式会社富士通研究所 〒 211-8588 神奈川県川崎市中原区小田中 4-1-1

E-mail: †{yamaoka-h,nakagawa.gaku,itakura.kota,takahash,kanemasa,ueki.miwa,tatsuro,riro}@jp.fujitsu.com

あらまし ビジネスにおけるIoT活用では、現場に存在する大量の人や物から取得したデータを単一のサービスから利用するのではなく、複数のサービスを跨いで横断的に利活用することで、現場毎の個別の問題解決のみならず全体最適に繋げることが期待されている。さらに、IoTサービスは現場の状況変化に応じて要件も常に変化するため、事前設計したサービスを運用するのみではなく、サービスの運用開始後も多様な活用アイデアのトライ&エラーを繰り返しながらサービスを発展させていくことが求められている。しかし、従来のIoTを活用したサービスは垂直統合によって開発されがちであり、異なるサービス間で広くデータを利活用したり、運用開始後にサービスの設計を変更することは難しかった。そこで我々は、実世界を対象としたサービスの容易な変更・改良と、異なるサービス間のデータ連携を目的として、高頻度で生成されるIoTのデータストリームから、実世界の状況を人や物の単位でクラウド上に写像して同期・管理するオブジェクトと、プラグイン機構によるオブジェクトに対する機能拡張を特徴とするイベント処理基盤を提案する。本論文では提案するプラットフォームのアーキテクチャと、その応用例として試作した、大量の接続済み車を想定したモビリティサービスの内容について述べる。

キーワード IoT データ, ストリームデータ処理, 接続済み車

1. はじめに

従来、IoTサービスの主な役割は、現場のIoT機器から取得したデータから現場状況をより詳細・正確に把握し、可視化や最適化に繋げることであった。しかしIoTへの期待は、特定の現場の問題解決から、複数の現場/業務/業種に跨った全体最適化や新たなサービスの創出へと広がっている。このことから、我々はクラウドにアップロードしたセンサデータを分析することで生まれる価値のみではなく、現場のデータを複数のサービスが横断的かつリアルタイムに利活用することによって生まれる価値に注目している。例えば接続済み車を想定すると、運送車両の位置や積み荷内容に関するデータを複数の物流業者間でリアルタイムに共有できれば、車両同士が積み荷を交換し輸送最適化を実現できる可能性がある。また、急ブレーキ操作を行った回数や地点のデータを保険会社が利用できれば、運転状況に合わせたきめ細かなインセンティブ保険サービスに繋がる。

これまでIoTサービスを構築する場合、センサデバイスの制御処理や、センサデータをサービスから利用し易くするための抽象化処理の開発が、サービスの開発と併せて密結合・垂直統合的に行われてきた。垂直統合型のシステム開発は、全体設計に合わせて事前にシステムの構成要素間のインタフェースを定義することができ、またそれによってパフォーマンスの最適化も行える。しかしその反面、一度設計して運用を開始したシステムは、その後の動作変更が容易に行えないという問題がある。また、個々のサービスの処理やデータがそのサービスに特化した形で存在し、結果として異なるサービス同士の連携を困難に

している。

一方、AWS(Amazon Web Services)やMicrosoft Azureに代表されるPaaSを用いたマイクロサービスアーキテクチャ[2]によるサービス開発も主流になりつつある。マイクロサービスアーキテクチャは、独立して存在する機能単位を、要件に応じてその都度組み合わせてシステムを構成することで、サービスの仕様変更に対し比較的柔軟に対応できる。しかし、マイクロサービスアーキテクチャは各機能単位、あるいはその組み合わせの機能単位群の境界や役割を明確に設計・定義することが難しい[3]。特に複数の異なるサービスが連携するような状況では、各サービスが個々の都合で機能単位の設計を行った結果、機能単位の間で複雑な依存関係が生じ、結果として垂直統合型でシステムを構築した場合と同様に、運用開始後の変更やメンテナンスが困難となる可能性がある。

2. 課題

複数サービスのリアルタイムな連携と運用開始後のサービス内容の柔軟な変更を実現するためには、まず実世界に存在する大量の人や物のデータを遅延なく処理できなくてはならない。その上で、通常は各サービスの内部に存在するデータや処理を、サービスを跨いで横断的に流通させる仕組みが必要である。これらのための課題を次に述べる。

1. 実世界（デバイス）とサービスの分離

一般的に、IoTデバイスを利用するサービスの開発者は、利用するデバイスの仕様に熟知する必要がある、デバイスへのアクセス方式や状況毎の対処処理を、デバイス仕様に準拠した形で記述しなくてはならない。このための処理記述の複雑化がデ

バイスとサービスが密結合となる原因となっている。この結合を解くためには、まずデバイスの仕様や状況をサービスから隠蔽し、各々のデバイスの接続や同期処理をサービスから切り離して実行する必要がある。また、デバイスの独自形式で表現されている情報を一般的な形式に変換する必要がある。

2. サービスの多様性への対応

複数のサービスからのデバイスの容易な利用を可能にするためには、デバイスのデータを一般形に変換して保持するだけでは十分でない。一般化したデータをさらにサービスの望み形式に変換し、適切なタイミングでイベント化する必要がある。特に IoT 分野では、業種や業務の現場毎に多様なサービスが構築され、変換の方式やイベント化のタイミングがサービス毎に大きく異なる。このため、デバイスデータを参照するための固定の API を設計して提供するのではなく、サービス毎に異なるデータの使い方の要望に柔軟に対応できるようにする必要がある。

3. 運用中の処理の動的変更

現場の要望や改善施策に応じて試行錯誤しながら、上述した抽象化やイベント化の処理内容を変更することが求められる。この処理をノンストップかつ複数のモジュール間で一貫性を保ったまま行えるようにする必要がある。

4. スケール

ビジネスの現場に存在する大量の人や物から高頻度に発生するデータの保持や、そのデータのリアルタイムな変換、加工処理をスケール性を損なわずに実行できなくてはならない。

3. 提案する基盤

上述した課題に対応し、我々は大量の IoT データを容易に活用するためのプラットフォームとして IoT 向けイベント処理基盤を提案する。本基盤のコンセプトとシステム構成を図 1 に示す。

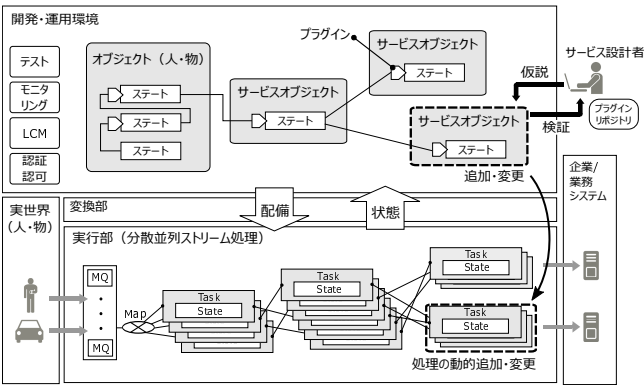


図 1 提案する基盤のコンセプトとシステム構成

本基盤の主な特徴は

- 1. オブジェクトによる実世界仮想化
- 2. プラグインによる処理の追加・拡張
- 3. 運用中な動的な処理変更
- 4. スケーラビリティ

である。これらは先に述べた課題 1~4 にそれぞれ対応してい

る。3, 4 については分散並列ストリーム処理基盤を基にして、スケール性能を保った上で、ストリーム処理を継続したまま処理内容を柔軟に変更できる機構を設計、実装することで対応した。この詳細については本基盤の実装に関する別論文 [1] で述べる。本論文では 1 と 2 について述べる。

3.1 オブジェクトによる実世界仮想化

本基盤ではデバイスから送信されてくるデータストリームを、実世界に存在する人・物単位でクラウド上に「オブジェクト」として仮想化する。オブジェクトは内部にステートを保持し、対象とする人・物の状況が実世界の状況と連動してリアルタイムに同期反映される。オブジェクトには「実世界の人・物に対応するオブジェクト」と「サービスオブジェクト」の 2 種類が存在する。

3.1.1 実世界に対応するオブジェクト

実世界に対応するオブジェクトは、人・物といった実世界の物理的存在を対象単位として、それに紐づく状態や処理をまとめて保持・管理するものである。これらのオブジェクトには、対象の人・物の状態の変化イベントが常時届けられ、その最新状態値がステートとして保存される。これらのイベントは、人や車に対応する例えばスマートフォンや車載 GPS 等といったデバイスから送信される。

実世界に対応するオブジェクトのステート階層の考え方を図 2 に掲載する。ステートは大きく分けてサービス固有ステート、一般ステート、デバイス固有ステートに分けられる。各階層の詳細について述べる。

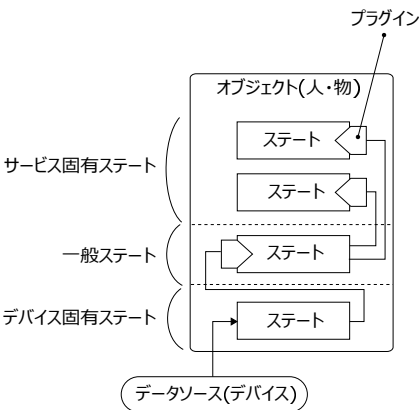


図 2 オブジェクトのステート階層

デバイス固有ステート)

デバイスが生成する人・物の状態変化イベントが常時届けられ、その最新値がデバイス固有ステートとしてステート保存される。デバイス固有ステートには、センサデータが未加工に近い形式で保存され、そのデバイスに依存した固有の表現形式のデータも含まれる。

一般ステート)

デバイス固有ステートは必ずしもサービス開発者によって理解し易いものとは限らない。そこで単位変換等の処理を行うことで、デバイス固有の表現形式を一般化・抽象化し、サービス開発者にとって理解しやすい一般ステートに変換する。一般ス

テートは特定のサービスに依存したものではなく、あくまで不特定多数のサービス開発者から理解し易いデータである。変換処理は後述するプラグインによって行う。このプラグインはサービス開発者とは関係の無い、デバイス仕様を知る者が開発することを想定している。

サービス固有ステート)

一般ステートをサービスにとって適した粒度や通知頻度に変換したり、複数の一般ステートを併せてサービスにとって必要な情報を生成した結果が保存される。このデータは後述するサービスオブジェクトから参照されることを想定している。サービス固有ステートを出力するプラグインは、サービス開発者がサービス毎に開発することを想定している。サービス開発者は一般ステートを参照すればよく、デバイス固有ステートの仕様を知る必要はない。

このような3階層構成をとることで、実世界を同期する役割を担う側はデバイス固有ステートの同期のみに着目すればよく、サービスを開発する側は、実世界に存在するデバイスの詳細仕様等を意識することなく、使い易い形でデータを利用できる。

3.1.2 サービスオブジェクト

本基盤では、実世界の人・物をオブジェクト化することに加え、それらの情報を用いて特定のサービスを行うための処理やデータも同様に、各サービスを単位として処理や状態をまとめて保持・管理できるようにしており、これをサービスオブジェクトとして定義している。サービスオブジェクトは通常のオブジェクトから送信されるイベントを受けてサービス固有の処理を行い、その結果を自らのオブジェクトのステートとして出力する。人・物といった実世界の対象物と、サービスの両方をオブジェクトという共通の枠組みで扱って稼働させることで、サービスのステートも他のオブジェクトから容易に利用可能となりサービス間のデータ連携が促進される。また、サービスも含めたスケールや整合性確保が可能となる。

3.2 プラグインによる処理の追加・拡張

オブジェクトはデータ処理を追加するプラグイン機構を有する。プラグインは特定のステートを監視し、そのステート変化をイベントとして捉えて起動し、何かしらの処理を行った結果を別のステートとして出力する。プラグインの具体的な処理としてはデータの抽象度コントロール、データの形式や型の差異吸収、複数のデータを組み合わせた新たなデータ生成などが挙げられる。上述したように、サービスは実世界のオブジェクトに対して自らのサービスに都合の良いサービス固有ステートを出力するプラグインを追加でき、これによって多様なサービスオブジェクトが柔軟に実世界のオブジェクトと繋がることできる。複数のプラグインをステートを介してパイプライン的に繋ぎ合わせることも可能である。

3.3 システム構成

図1に示した構成要素は次の通りである。

開発・運用環境

オブジェクトの定義やプラグインの開発、またオブジェクトに対するプラグインの接続設計などを行う。実世界の人・物の増減に応じたオブジェクトのライフサイクル管理、デバイスの

認証等の機能を有する。

変換部

開発環境で設計したオブジェクトやプラグインを、実行部である分散並列ストリーム処理に適した形に展開配備する。また、実行部から駆動中のオブジェクトのステート値や処理負荷状況を取得し、運用部のモニタリング機能に提供する機能を有する。

実行部

運用管理部よりデプロイされた構成に基づき、オブジェクトのステート値の保持・更新と、プラグイン処理を分散並列ストリーム処理基盤を用いて実行する。

4. モビリティサービスの試作

開発した基盤を用いて、コネクテッドカーを想定したサービスを試作した。試作したサービスのオブジェクト構成を図3に示す。

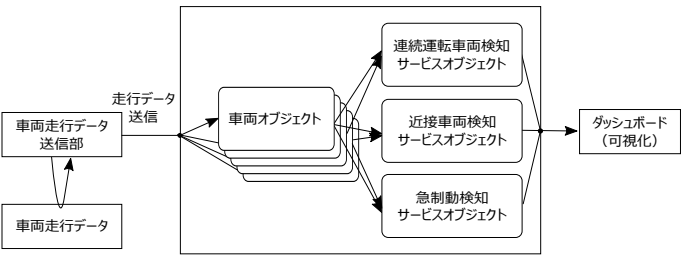


図3 試作したサービスのオブジェクト構成

図3において、基盤に対してストリームとして入力する車両走行データは実際の車両から取得したものを利用している。その仕様を表1に示す。今回の試作では2,608台分の車両データを用いて、2,608個の車両オブジェクトを生成・同期し、それらを利用するサービスオブジェクトを作成することでサービスを開発した。

表1 車両の走行データに含まれる内容

項目	頻度
位置（緯度経度）	毎秒
速度（時速）	毎秒
運転開始時刻	運転開始時のみ

4.1 試作したサービス

試作したサービスの具体的な内容は次の3つである。

長時間運転検知サービス

無休憩で規定時間を超えて連続運転をしている車両を検出し地図上にその位置を示す。運転手の過労による交通事故防止に役立てることを想定している。

近接車両検知サービス

長時間運転検知サービスが検知した運転超過車両について、それらの車両から一定の距離範囲内に位置する別の車両を検出し、地図上に描画する。近隣車両同士の積み荷交換の提案などにより、ドライバーの労働負荷軽減と輸送最適化に繋げることを想定している。

急制動検知サービス

車両の急制動運転を検出して急制動が多発しているエリアを地図上でヒートマップ形式で可視化する。道路上の落下物など交通危険個所の特定を目的としている。

4.2 開発したプラグイン

本基盤ではこれらのサービスの処理をプラグインを作成することで実装する。具体的には、車両オブジェクト側に追加するプラグイン、および、それらのプラグインが生成したデータを集約するサービスオブジェクト側のプラグインを開発することでサービスを構築する。試作したサービスにおけるオブジェクトとプラグインの関係を図4に示す。

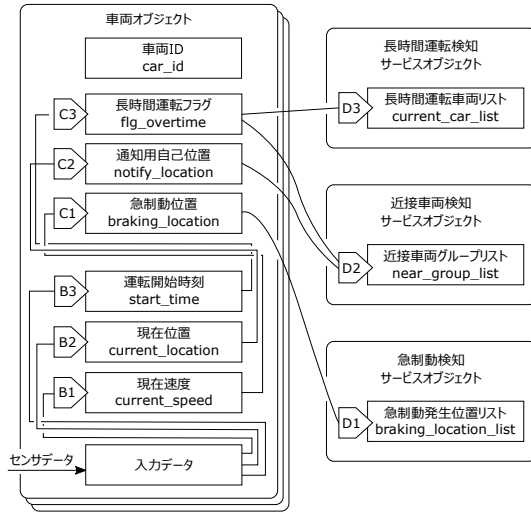


図4 試作したサービスにおけるオブジェクトとプラグインの接続関係

図4における各プラグインの役割を示す。

プラグイン B1～B3

入力データを一般的な形式のデータに変換して保存するものである。今回のケースでは緯度経度の測地系や速度の単位、タイムスタンプの形式変換を行っている。これらのプラグインは対象のオブジェクト（今回は車両）がサービスに依存せずに兼ね備える機能として、オブジェクトの設計時に作成する。これらのプラグインが出力するデータは図2内の一般ステートに該当する。

プラグイン C1～C3

B1～B3が出力した一般ステートをサービスに適した形に変換するものであり、サービス開発者が作成する。C1は急制動検知サービスが追加するプラグインであり、B1が出力する速度の変化を捉えて急制動の有無を検出し、急制動が生じた場合はその時の位置を出力する。C2は近接車両検知サービスが追加するものであり、自己位置をサービスに通知するためのものである。微小な位置変化の度に通知するのは無駄が多いため、一定距離以上移動したか、もしくは一定時間以上経過したタイミングで通知用の自己位置を更新する方式をとっている。C3は長時間運転検知サービスが追加するものであり、B3が出力した運転開始時刻と現在時刻との差から長時間運転の有無を判断し出力する。C1～C3の処理の内容を Algorithm1～3 に記す。

Algorithm 1 急制動検知プラグイン (C1) の処理

Input: 現在速度: *current_speed*, 過去速度: *past_speed*, 急制動判定閾値: *th_braking*, 現在位置: *current_location*
Output: 急制動発生位置 *braking_location*
if *past_speed* - *current_speed* > *th_braking* **then**
 braking_location $\leftarrow$ *current_location*
end if
past_speed $\leftarrow$ *current_speed*

Algorithm 2 通知用自己位置更新プラグイン (C2) の処理

Input: 現在位置: *current_location*, 現在時刻: *current_time*, 前回通知時刻: *past_notify_time*, 前回通知位置: *past_notify_location*, 通知抑制時間: *suppress_time*, 通知抑制距離: *suppress_dist*
Output: 通知位置: *notify_location*, 通知時刻: *notify_time*
if *current_time* - *past_notify_time* > *suppress_time*
or *dist(current_location, past_notify_location)* > *suppress_dist* **then**
 notify_location $\leftarrow$ *current_location*
 notify_time $\leftarrow$ *current_time*
end if

Algorithm 3 長時間運転検知プラグイン (C3) の処理

Input: 運転開始時刻: *start_time*, 長時間判定閾値時間: *th_overtime*, 現在時刻: *current_time*
Output: 長時間運転フラグ *flag_overtime*
if *current_time* - *start_time* > *th_overtime* **then**
 flag_overtime $\leftarrow$ *true*
else
 flag_overtime $\leftarrow$ *false*
end if

プラグイン D1～D3

これらはサービス固有処理を行うプラグインであり、複数の車両オブジェクトのステートを参照し、その変化イベントを集約して処理を行う。D1は車両オブジェクトから送信される長時間運転状況の変化イベントを受け取り、長時間運転をしている車両リストの内容を更新して自らのステートとして出力する。D2は車両の位置と長時間運転状況変化イベントを受け取り、長時間運転をしている車両ごとに、その近隣に存在する車両をリアルタイムに抽出し、ステートとして出力する。D3は急制動発生イベントを受け取り、その位置リストを更新して出力する。

4.3 サービスダッシュボード

試作したサービスを実行中のダッシュボードを図5と図6に示す。なお、今回全オブジェクトのステート情報を外部のデータベースに出力しており、図6に示した地図はデータベースから取得した情報に基づいて描画している。

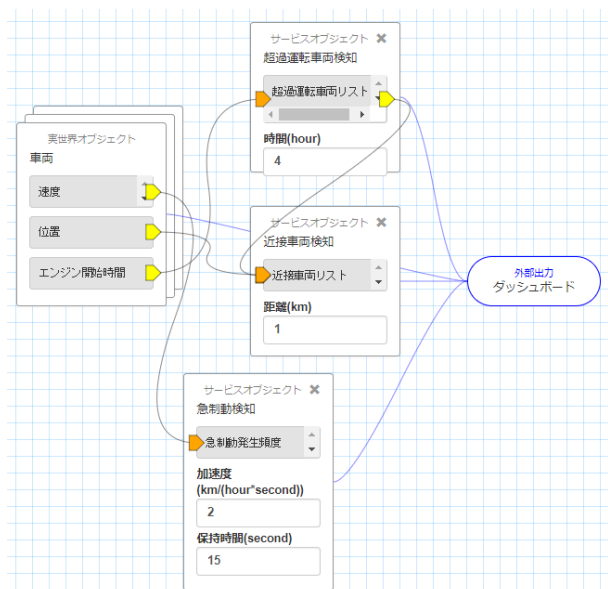


図5 ダッシュボード（プラグイン接続設計）：サービスの有効無効切替やプラグインのパラメータ変更を行う。

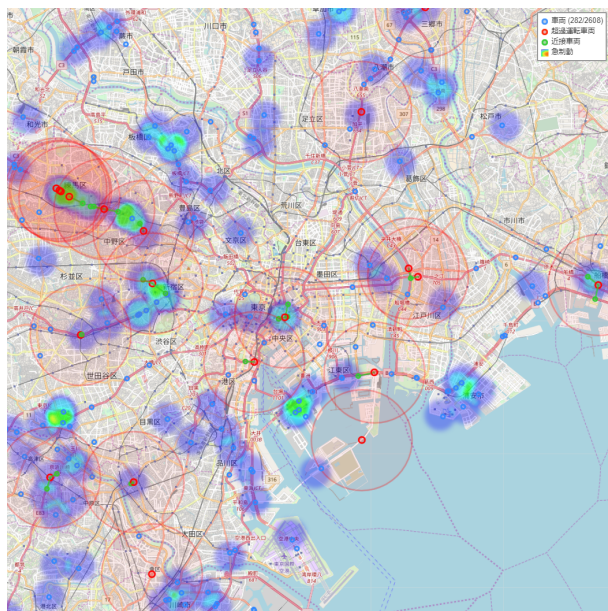


図6 ダッシュボード（地図画面）：サービスの稼働状況をリアルタイムに可視化。各点は車両を表しており、赤い点と円は連続運転車両とその近接領域。近接領域内に存在する他の車両が近接車両として検出される。ヒートマップ形式の濃淡は急制動発生頻度を示している。描画には OpenStreetMap(<https://www.openstreetmap.org/>), Leaflet(<http://leafletjs.com/>) を利用。

5. 考 察

今回試作したモビリティサービスを、本基盤上で開発した際の利点を述べる。

車両走行データの詳細仕様をサービス開発者が意識不要

車両走行データのパスや一般形への変換処理が行われた結果が車両オブジェクトのステートとして出力されるため、サー

ビスの開発者は車両オブジェクトの特定のステートの内容（今回のケースでは位置，速度，運転開始時刻）のみを把握しておけば，そこからサービスを組み立て可能であった。実際のサービス適用時には車種や搭載センサによってデータの仕様が異なることが考えられるが，この場合でもその変換処理は全て車両オブジェクトの内部で行われるため，車種の追加やセンサ仕様の変更に対してサービス側は影響を受けにくい。これは前述した課題1に対応する効果である。

車両データを多様なサービスから利用可能

車両オブジェクトに対して各サービスがプラグインを追加することで各々のサービスにとって都合の良い形でデータを受け取れた。これによって運転超過検知，急制動検知，近接車両検知サービスが，それぞれサービス側で行う処理を最低限に保ちながら，容易に車両オブジェクトのデータを利用可能であった。これは課題2に対応する効果である。

サービス間の容易な連携が可能

プラグインが出力したステートを他のサービスが利用することで，サービス間の容易なデータ連携が可能であった。今回は長時間運転検知サービスのためのプラグインが出力した運転超過情報を，近接検知サービスが利用することによって，長時間運転をしている車両に限定し，その近隣に存在する他の車両を検出するサービスを容易に組み立て可能であった。これは課題2に対応する効果である。

6. ま と め

実世界の人や機器などの事物の状態をデジタルに写像し，それを複数のサービスから組み合わせることで利用することによって，現場と連動したサービスの開発と，サービス間の連携を実現するプラットフォームを提案した。今回提案したプラットフォームは，数十万規模の大量のデバイスから頻繁に発生するデータをリアルタイムに扱いつつ，実世界の人・物が生み出すデータを複数の異なるサービスから横断的に利用できる。また，プラグイン機構により仮説と検証を繰り返しながらサービスの内容を運用中に動的に改善でき。本プラットフォームを用いてモビリティサービスを試作した結果，課題解決に対して本基盤の特徴が効果的であることも分かった。今後はプライバシーやセキュリティ機能の追加，人・物のデータモデルの検討の検討，開発運用環境の強化などを予定している。

文 献

- [1] 中川岳, 金政泰彦, 山岡久俊, 板倉宏太, 高橋英一, 植木美和, 松本達郎, 武理一郎, “Flink を用いた動的変更可能な IoT 向けイベント処理基盤”, DEIM2018-C7-2.
- [2] S. Newman, “Building Microservices Designing FineGrained Systems.”, O’Reilly Media, Feb. 2015.
- [3] Simon Brown, “Distributed big balls of mud”, Coding The Architecture, July 2014.
- [4] Blackstock, M. and Lea, “R. 2012. IoT mashups with the WoTKit. Internet of Things (IOT)”, 2012 (Wuxi, China, Oct. 2012), 159166.