情報処理学会論文誌 Vol.62 No.3 867–876 (Mar. 2021)

futaba：スマートビルのための ビッグデータ・プラットフォーム

粕谷 貴司[1,2,a)] 江崎 浩[1]

受付日 2020年5月19日, 採録日 2020年12月1日

概要：本研究では，スマートビル専用のデータ・プラットフォームである futaba を提案する．リアルタイムのビッグデータ処理技術であるラムダ・アーキテクチャと Web of Things による RESTful API を用いることで，リアルタイム制御とバッチ処理が両方が可能でスマートビルの多くのユースケースに対応できる高い汎用性を得た．また，スマートビルに有用なセマンティクスの提案を行い，プラットフォームのデータモデルに組み入れることで，サードパーティの新規参入障壁を下げ，アプリケーションの再利用性を向上させた．加えて，PaaS の採用と機能最適化により，毎分 5 万件のデータを想定した場合でも，ビッグデータ処理部が月額 1 万円以下で運用可能であることを確認した．

キーワード：スマートビル，ビッグデータ，セマンティックウェブ，web of things

futaba: Bigdata Platform for Smart Building

Takashi Kasuya[1,2,a)] Hiroshi Esaki[1]

Received: May 19, 2020, Accepted: December 1, 2020

Abstract: We propose futaba, a data platform dedicated to smart buildings. We obtained a high level of versatility by using the Lambda Architecture and the RESTful API with Web of Things, that enables both real-time control and batch processing for many use cases of smart buildings. We also proposed semantics applicable for smart buildings and incorporated them into the data modeling of the platform, lowering the barrier to entry for third parties and improving the reusability of applications. In addition, by adopting PaaS and optimizing functions, it was confirmed that the big data processing unit could be operated for less than 10,000 yen per month even assuming 50,000 data items per minute.

Keywords: smart building, big data, semantic web, web of things

1. はじめに

近年，建設業界において，設計・施工・維持管理業務のデジタル化が進んでおり，スマートビルと呼ばれる高度な制御機能を有した建物も増えてきている．スマートビルについては多様な定義が存在しているが，クラウド，IoT（Internet of Things），人工知能（AI: Artificial Intelligence）などを用いて，省エネや快適性・利便性の向上などを実現するビルという理解がされている．スマートビル増加の理由としては，技術革新に加え，3 次元形状や部材の属性情報なども含んだ総合データベースといえる BIM（Building Information Modeling）の普及，震災による省エネ・BCP 需要の高まり，人材不足などの社会的要請がある．

スマートビルの普及が進む一方，実際にビル設備とクラウド・IoT・AI などを連携させようとした場合，多くの課題が顕在化する．具体的には，多種多様なデバイスへの対応が困難であること，収集されたデータに対してメタデータやセマンティクスを与える慣習がないこと，システム運用に多額のコストがかかることであり，スマートビルの普及と新規参入を防ぐ大きな障壁となっている．そのため我々は多様なユースケースに適合する汎用性を持つ，スマートビルの専用データ・プラットフォームであ

[1] 東京大学
The University of Tokyo, Bunkyo, Tokyo 113–9656, Japan
[2] 株式会社竹中工務店
Takenaka Corporation, Koto, Tokyo 136–0075, Japan
[a)] kasuya@hongo.wide.ad.jp

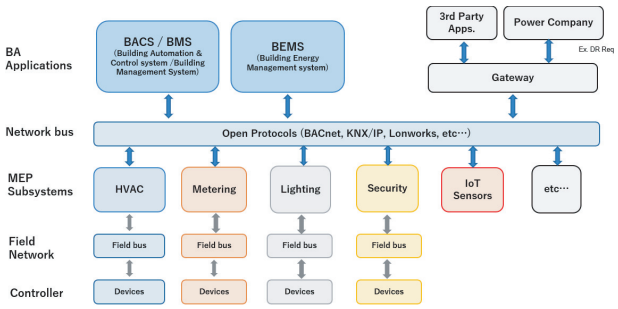

図 1 建物設備制御システムの構成イメージ
Fig. 1 Building automation and control system.

る futaba（Facility Unified digital-Twin Architecture for Building Automation）を提案する．

本論文では，2 章でビルの設備制御の中心である BACS（Building Automation and Control System）を説明し，3 章で関連研究について述べ，4 章で本研究の目的を述べる．5 章で対象とするスマートビルの要件分析を行い，6 章で futaba の設計要件について述べ，7 章でそれぞれの実装と評価について述べる．8 章で futaba によるアプリケーション例について述べ，9 章で本論文の結論を述べる．

2. BACS

BACS とはビル向けの中央監視制御システムを示すものであり，ISO 16484 シリーズで定義されている．具体的には，ビルにおける電気，ユーティリティ，照明，空調，給排水衛生などの諸設備を中央監視室で一元的に監視，制御および管理する機能を有し，設備の稼働状況を人的または自動的に常時監視し，またトラブル発生時に適切に対応し，良好なメンテナンスを図るためのシステムである [1]．

図 1 に BACS の構成を示す．おおむね 1 万 m^2 以上のビルでは，空調・照明などのサブシステムをモニタリング・制御する BACS が設置され，管理員が集中監視業務を行っている．一般的にサブシステムはそれぞれ構築ベンダが異なり，スタンドアロンで動作するよう構築される．それらがゲートウェイ機器を通じて BACnet [*1] などの専用プロトコルでシステム間の通信を行う構成となっている．BACS は閉域網による構成を前提としていたが，近年インターネット上のシステムとの連携事例が増えてきており，柔軟な連携の対応が求められている．なお，BACS で扱われるデータは「ポイント」と呼ばれ，デバイスの状態値である計測・計量・警報・状態・設定など表している．

3. 関連研究

BACS にインターネットなどの技術を適用して，高度化を図る研究としては U.C. Berkeley の SDB（Software Defined Building）プロジェクトで，多くの研究がされている．

Dawson-Haggerty らは，センサや BACS といった物理デバイスの情報をシンプルかつ効率的に交換するための技術である sMAP（Simple Measuring and Actuation Profile）を提案している [2]．sMAP はインターネットを介した運用を前提とした RESTful なウェブサービスであり，データモデルとインタフェースを規定している．デバイスの現在値の取得だけではなく，メタデータの取得や，WebSub によるデータ取得も可能である．

彼らはさらに，BACS をよりプログラマブルにすることを目的に，ハードウェアの抽象化を行い，アプリケーションの移植性を高め，耐障害性を高めるために BOSS（Building Operating System Services）を提案した [3]．ビル用 OS を目指し，sMAP をベースとした命名規則とセマンティクスを用いて分散環境におけるリソースの抽象化を行い，それらを柔軟に扱うために独自のクエリ言語を開発し [4]，データのバッファリングやリソースの発見機能，排他的制御を実現するためのトランザクションやアクセス制御機能を提供している．時系列データ処理のためのサービスを有しており，クレンジングなどの前処理の機能を組み込むことによって，アプリケーションの開発効率の向上を狙った．

Fierro らは BOSS を拡張した XBOS（An Extensible Building Operating System）を提案している [5]．XBOS はスマートビルのためのより高度なビル用 OS であり，特徴的なのは，ビルシステムのライフサイクル管理のためのプロファイル（Building Profile）を定義していることである．Building Profile は空間階層のセマンティクスや，ビルシステムの記述に必要なメタデータを含むが，独自記法を採用しており，拡張性に乏しい．そのため，Balaji らは BACS に特有な語彙と関係性を定義したメタデータ・スキーマである Brick [6] を提案した．

Fierro らはさらに，上記の技術を組み合わせ，Mortar と呼ばれるデータ・プラットフォームを開発した [7]．実在するビルのデータ（90 棟）を集めており，それぞれ Brick によるメタデータが付与されたうえで公開されている [*2]．Mortar は，スカラー値の保存の適したスケーラブルな時系列データベースである BTrDB [8]，データモデルストレージとして Brick に最適化された HodDB [9]，Go 言語で実装したクエリプロセッサなどで構成される．アプリケーションの実行環境としては GRPC による API を提供しており，クエリプロセッサを介してデータ取得が可能である．

SDB プロジェクトによる，BACS の高度化の貢献は大きいといえるが，課題としては，スカラー値以外は未対応であること，システムの運用にサーバクラスタの稼働が必要があるため運用コストが高くなること，大量のポイントを有するビルの場合，人手でのデータモデリングが困難であることがあげられる．

[*1] https://www.iso.org/standard/71935.html

[*2] https://mortardata.org/

ほかにも，SOAP を用いたストレージを中心としたアーキテクチャによってエネルギー管理システムを構築した研究 [10]，中小ビルのエネルギーマネジメントシステムの構築を目的とした BEMOSS [11]，独自のドメインモデルに基づいたアーキテクチャを提案している BaaS（Building as a Service）[12]，Lilis らによるメッセージ指向ミドルウェア [13] があるが，プロジェクトごとに利用しているメタデータやオントロジが異なるために，横断的なデータ分析や BACS の専門業者以外の新規参入が難しい，またはビッグデータ処理や AI 適用のユースケースを想定していないなどの課題がある．

一方で，Jara ら [14] は，IoT の相互接続性を向上させるための仕組みが，Restful なアーキテクチャに基づいた WoT（Web of Things）であると述べている．WoT とは実空間上のオブジェクトである Thing に対して，インターネットの技術により情報取得や操作などの機能を提供する技術仕様であり，W3C で規定されている[*3]．WoT をスマートビルに適用する事例としては Bovet らの取り組みがある [15]．ローカルの分散環境下のデバイスに WoT のミドルウェアを組み込むとともに，独自のセマンティクス拡張を施したものであるが，BACnet を用いた既往の中央監視制御システムとの連携については考慮されていない．

本研究の提案は，位置情報などの多次元データといったスカラー値以外のデータに対応し，多様なユースケースに対応するための柔軟なテレメトリ・スキーマ定義の導入，安価に運用を可能にするための PaaS を前提としたビッグデータ処理基盤，BIM とポイントリストを用いたデータモデリングおよびセマンティクス付与，および WoT による API の提供である．

4. 本研究の目的

本研究の目的は，幅広いユースケースに適合する汎用性を持つ実践的なスマートビル専用のデータ・プラットフォームの構築である．前述の課題とプロジェクト経験から設計指針として以下を定めた．

多様なユースケース対応する高い汎用性 スマートビルにおいては，即応性が求められるもの，1 日ごとのバッチ処理が必要なもの，複数ビルを制御するものなど，多様なユースケースが存在する．それらに対応することに加え，データ種別としてスカラー値以外の情報も扱える高い汎用性が求められる．

再利用性の高いアプリケーション スマートビルのアプリケーションは，IoT/AI を用いるなど BACS の専門会社以外のサードパーティとの協業が多い．BACS では，ポイントの意味や関係性を表すセマンティクスが与えられることがほとんどないため，データの理解が困難で参入障壁が高く，かつ構築したアプリケーションの再利用性が乏しい．そのための適切なメタデータと合理的なデータモデルを付与し，共通化した API を提供することで，開発した機能モジュールの再利用が可能な構成とする必要がある．

建設プロジェクトへ適用可能 既往研究では，コストや保守性などを考慮していないことが多い．たとえば，HodDB や BTrDB は拡張性を確保するため，サーバクラスタを必要とするが，オンプレミス環境であってもそれらの保守・運用のコストと高額になる．ビル管理業務は費用削減がつねに求められているため，運用コストは可能な限り抑えて提案する必要がある．また，保守性の向上のために責任範囲の明確化が重要であり，プラットフォームを介したシステムの不具合が BACS に影響を与えない構成とする必要がある．

5. スマートビルの要件分析

本章ではスマートビルのユースケース分析を行う．実際のプロジェクトで構築した特徴的なユースケースを紹介し，それらの分析から得られた要件と，スマートビルに必要なセマンティクスについて述べる．

5.1 想定ユースケース

5.1.1 IoT デバイスを用いた快適制御

IoT を使った BACS の事例として多いのが，快適性や利便性の向上のための無線センサの活用である．粕谷らは，無線環境センサにより細かい温湿度などをとらえ，Bluetooth ビーコンによって執務者の位置情報を取得するとともに，ウェアラブルデバイスによって計測した心拍数，加速度から計算した代謝量をリアルタイムに計算している [16]．それらの情報と執務者の不快感申告も考慮したうえで，風量制御が可能なパーソナルファンと天井の放射パネルの温度を制御することで，快適な温熱環境の提供を目指したウェルネス制御システムを構築した．

5.1.2 強化学習による高度な設備制御

一般的な BACS では，ビル管理員や入居者の目標値設定や，熱源機器などはスケジュールに従って動作する．熱源機器の温度設定や運転スケジュールなどは，竣工時に決められた後にチューニングされることは稀で，されたとしても人手による制御では，省エネと快適性を両立させるような高度な制御の実現は難しい．そのためビル制御への AI 適用事例が増えており，たとえば強化学習を使って自律的にパラメータの最適化を行いながら制御する試みがある [17]．ここではバッチ処理で抽出したデータをもとに制御対象である目標値を予測する AI と，決められた目標値から報酬を定め，それに従って行動（制御）をリアルタイムに決める強化学習 AI の 2 つがあり，それぞれが独立して学習を行う構成となっている．なお，強化学習によって

[*3] https://www.w3.org/WoT/

最適な設備の制御値を求める研究は Han らの調査 [18] に詳しい．この調査では，限定的な環境や制御対象での評価が多いが，文献 [17] のように多くの特徴量を前提とした制御も，実用段階に入ってきたといえる．

5.2 機能分析と設計要件

上記ユースケースに加え，プロジェクト対応経験から，データ・プラットフォームに対して以下の設計要件を得た．

制御ロジックの変更容易性 スマートビルではチューニングのために，頻繁な機能やロジック，パラメータの変更が想定される．それらを現地の設備改修で行うことは，コストや事務的な手続きにおいて負荷が大きく，クラウドから容易に変更をデプロイできる構成が望ましい．そのため，遠隔更新が可能なゲートウェイを開発するとともに，パラメータ変更のための API を提供することとする．

リアルタイム性の高い通信プロトコル デバイスの遠隔制御のために，ウェブアプリケーションを提供することが多いが [19]，たとえばポーリング型のアーキテクチャを採用した場合，十分なリアルタイム性が確保できず満足度が著しく損なわれることがあった．このため，IoT に親和性が高い Pub/Sub 型のプロトコルである MQTT（Message Queuing Telemetry Transport）を採用する．これは BACS と IoT の連携評価を行った先行研究 [20] の評価による．

ビッグデータ保存のために安価なストレージ ユースケースで必要となる BACS のデータは 1 棟あたり数千〜数万ポイントとなり，取得周期はおおむね 10 秒〜300 秒である．また複数棟のユースケースも考慮すると，それらは必然的にビッグデータとなる．安価な運用のため，Apache Hadoop 用の分散ファイルフォーマット（HDFS: Hadoop Distributed File System）を用いたビッグデータ処理基盤を構築する．

共通機能の提供 電力負荷予測のユースケースでは，日本の電力計測の慣習に合わせて 30 分ごと（1 日だと 48 次元）のデータ抽出が多いが，見える化システムなどは 1 時間ごとのデータ抽出が多い．こうした多様な粒度のデータ抽出への対応が必要である．また，ビル設備のデータ分析に特有な処理として，積算値差分や欠損値補間がある．積算値は最大値までカウントすると 0 に戻る仕様があり，それらの計算間違いが問題になることもある．これらの機能もプラットフォーム側の機能として盛り込むこととする．

実用的なセマンティクスの導入 サードパーティの参入障壁を下げ，アプリケーションの移植性を高めるために必要である．詳細は次節で述べる．

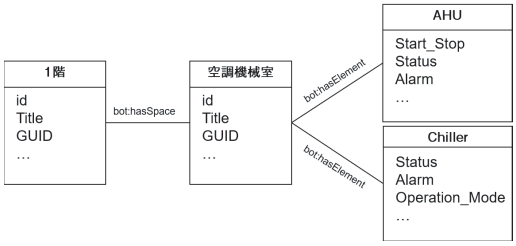

図 2 スマートビルのアプリケーションに必要なセマンティクス
Fig. 2 Suitable semantics for smart building applications.

5.3 必要なセマンティクス

建設ドメインにおいて，多様なメタデータ・スキーマやオントロジが提案されている [21]．Brick は BACS において最も参照されているオントロジであり，十分な記述能力を持ついえるが，我々の対応したユースケースにおいては，ここまで詳細なセマンティクスを利用するようなユースケースはほとんどない．たとえば，空調機（AHU）がどの VAV（可変風量の吹出口）に対して空気を提供しているといったことは feedsAir という語彙で記述できるが，それら静的関係性は，AI による遠隔制御やモニタリングといったユースケースにおいてはほとんど利用されない．

図 2 はユースケースから導出した，スマートビルに必要なセマンティクスである．空間階層と空間に包含されるデバイスやセンサがあり，それらに紐づくポイントがプロパティとして記述される．これによって，クラウドシステムでは対象の空間を指定することで，内包されるデバイスなどの取得が可能になる．利用するオントロジは，空間階層の記述に適した BOT（Building Topology Ontology）[*4] とデバイスの記述に適した Brick を採用し，データモデルは RDF（Resource Description Format）によって記述する．これにより，ポイント単位でのエンジニアリングから脱却し，オブジェクト指向のモデリングにより，開発者に対してより直感的なインタフェースを提供する．

こうしたモデリングは，BACS に詳しい設備エンジニアが行う必要があるが，オントロジにも精通している必要があり，大型ビルの場合はポイント数も多く負荷もかなり大きい．そのため，我々は BIM のデータ交換用のフォーマットである IFC（Industry Foundation Classes）から，空間階層と包含される構成要素を抽出し RDF に変換するツールを開発することでモデリングの効率化を試みた．

IFC には，BIM でモデリングされる形状情報と，空間階層および設備機器や部材の属性情報の双方が記録されている．開発したツールは，IfcSite（敷地），IfcBuilding（ビル），IfcBuildingStorey（フロア），IfcSpace（部屋）といった空間階層を探りながら，空間が内包する IfcElement の派生クラス（設備機器など）と属性情報を抽出し，BOT と Brick を用いて RDF に変換し，同時に抽出した IfcSpace

[*4] https://w3c-lbd-cg.github.io/bot/

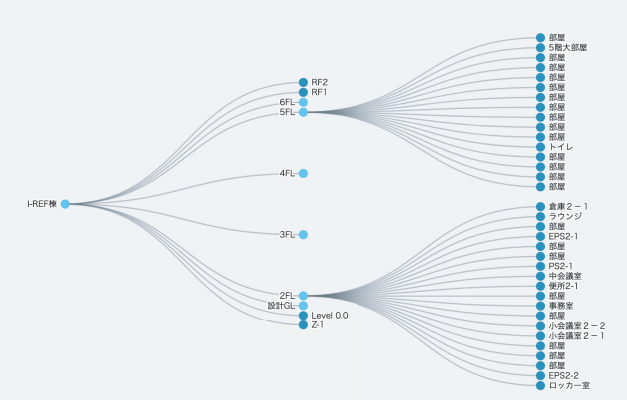

図 3 IFC から抽出した空間階層構造
Fig. 3 Space graph extracted from IFC.

```
inst:Entrance a bot:Space ;
    rdfs:label "Entrance" ;
    bot:hasElement inst_device:CO2-W-3,
        inst_device:MS-W-2,
        inst_device:MS-W-3,
        inst_device:TEP-C-3-RT .

inst_device:CO2-W-3 a inst_class:CO2-W-3 ;
    rdfs:label "CO2-W-3" ;
    rdfs:comment "CO2 濃度" ;
    brick:hasPoint inst_point:CO2-W-3 .

inst_point:CO2-W-3 a brick:Sensor ;
    rdfs:label "CO2 濃度" ;
    inst:topic "takenaka.co.jp/Site_Name/EQ_House/-/Entrance
                /Utility/Sensor/WebCNT3/CO2-W-3/CO2_Level/R" .
```

図 4 建物メタデータの例（抜粋）
Fig. 4 Example of building metadata.

の形状出力も行う．なお，IFC から RDF への変換ツールとして IFCtoLBD [*5]があるが，日本語対応などに課題があるため，IfcOpenShell [*6]と rdflib [*7]を用いて開発した．

図 3 にツールを使って出力した RDF の可視化例を示す．建物，フロアといった空間階層が抽出されていることが見てとれる．なお，ポイントと構成要素の突合には，ISO 16484-3 に定義されている BACS ポイントリストを用いる．ポイントリストにデバイス名などをユニークな ID として付加し，ID を BIM に埋め込むことで上記ツールの出力に含有させ，それらを突合するスクリプトによって，ポイント情報を含んだ RDF（建物メタデータ）を生成する．

図 4 に生成した建物メタデータの例を示す．Entrance という空間が 4 つのデバイスを有しており，そのうちの 1 つのデバイスが CO2-W-3 というセンサのポイントを有していることが記述されている．

[*5] https://github.com/jyrkioraskari/IFCtoLBD
[*6] http://ifcopenshell.org/
[*7] https://github.com/RDFLib/rdflib

6. futaba の設計

前章の要件をもとに futaba を設計した．図 5 にシステム・アーキテクチャを示す．リアルタイム・ビッグデータ処理のためのアーキテクチャの 1 つとして知られるラムダ・アーキテクチャ [22] をベースとしている．

futaba では，ゲートウェイなどを介して取得するデータ（テレメトリ）をリアルタイム処理のためのホットパス，バッチ処理のためのコールドパスに分岐し，それぞれをスピードレイヤ，バッチレイヤで処理し，サービスレイヤを介してデータの利活用を行う．サービスレイヤでは，WoT に準拠した RESTful API を提供しており，リアルタイムデータの取得，メタデータの取得，制御コマンドの発行，蓄積されたデータ抽出などが可能である．オープンシステムと PaaS を前提としており，保守性の向上と運用コストの低減可能な構成を目指した．なお，futaba は日本語の二葉（双葉）からとられており，特徴的なラムダ・アーキテクチャ，デジタルツインを想起させることを意図して命名した．以降でそれぞれの機能について述べる．

6.1 スピードレイヤ

6.1.1 データ入力部（ゲートウェイ）

BACS で取得できるポイントデータを，MQTT などに変換して Data Ingest に送信するためには，ゲートウェイ（GW）によるプロトコル変換が必要である．BACS では BACnet 以外にも多様な通信規格が用いられるため，対応プロトコルの追加・変更と遠隔更新を容易にすることを目的として，OSGi [*8]や Docker [*9]を用いて開発する方針とした．

6.1.2 データ処理部

建物から受信したテレメトリを事前に定義したルールに基づき前処理を行い，現在値の保存などのリアルタイムデータ処理を行う．データの受信は Data Ingest から行い，たとえば MQTT Broker を介してテレメトリを取得する．テレメトリには**表 1** に示すスキーマを定めた．データは values が格納されるが，多様なデータタイプに対応するため，value のみを必須とし，ほかは自由に決められる仕様とした．図 6 にビーコンによる位置情報の例を示す．value には userId が転記されており，バッチレイヤでの抽出時に，補間処理の対象データとして扱われる．data は文字列として抽出され，必要に応じてクライアントで処理される．

Message Broker からの取得したテレメトリは，Ingress Function と呼ばれるコンポーネントによって，必要な前処理が行われる．たとえば異常値の判定や，事前に定義されたスキーマ外のデータの除外などである．処理済みのデー

[*8] https://www.osgi.org/
[*9] https://www.docker.com/

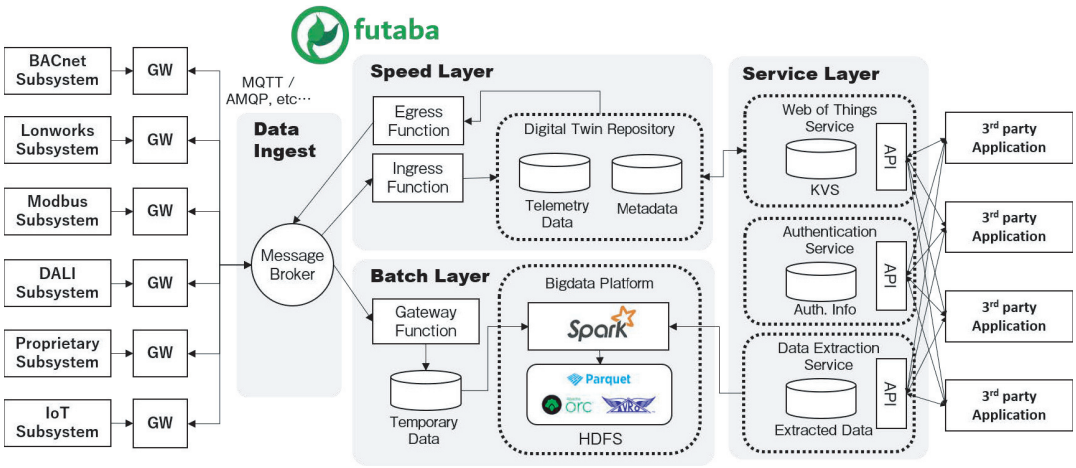

図 5 futaba のアーキテクチャ
Fig. 5 System architecture of futaba.

表 1 テレメトリのスキーマ
Table 1 Schema for telemetry.

メタデータ	説明
pointID	ポイント識別子
eventTime	イベント発生時刻
topic	トピック名
values	値情報（任意型）
sequenceNumber	シーケンス番号．欠損値の検出に用いる．

```
{
  "pointID": "73AC698E-B03F-4F64-B365-8C009EF02777",
  "eventTime": "2020-03-06T18:05:00 999+0900",
  "topic": "takenaka.co.jp/Site/Building/Floor/Space
           /Location/Beacon/BLE/Area_4/Location/R",
  "values": {
    "value": 31,
    "data": {
      "areaId": 4,
      "userId": 31,
      "locationTime": "2020-03-06T18:00:00+0900",
      "x": "-12.5068",
      "y": "-22.1004",
      "floor": "1"
    }
  },
  "sequenceNumber": 1234
}
```

図 6 テレメトリの例
Fig. 6 Telemetry example.

タは，Digital Twin Repository と呼ぶサービスに保存され，この際に対象のポイントデータの ID と，5.3 節で示した RDF をもとにして，サービスレイヤにおけるエンドポイントでの参照値にマッピングする．公開されたエンドポイントから，ビル内設備の制御も可能であり，ポイントへの制御指示は，Egress Function によって，不正制御などを防ぐためのフィルタリングなどを施され，同様に Message Broker を経由して，ビル設備の中に伝達される．

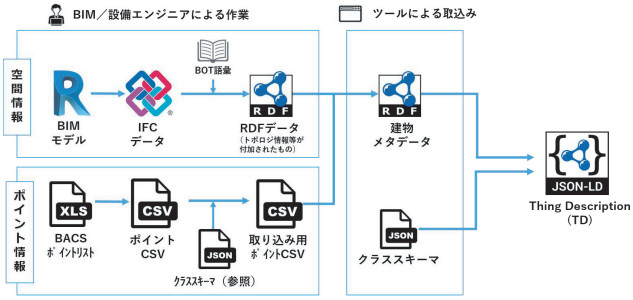

図 7 Thing Description の生成
Fig. 7 Generation of a thing description.

6.2 バッチレイヤ

Message Broker から取得したテレメトリは Gateway Function によって，一時保管領域にファイルなどで格納される．その後，Apache Spark を用いて 1 日 1 回～数回のバッチ処理で HDFS に変換し，処理済みの一時ファイルは削除する．Apache Spark Stream を用いる構成も可能ではあるが，サーバクラスタを起動させ続ける必要があり，クラウドでの稼働を前提とすると運用コストが増加するため，この構成とした．データ抽出のリクエスト時のパラメータで抽出周期や期間を指定し，データ補完処理や異常値検出，積算値差分なども行う．

6.3 サービスレイヤ

スピードレイヤで得られたリアルタイムのデータを公開するための WoT Service と，バッチレイヤで蓄積されたデータを抽出するための Data Extraction Service，それらのサービスを利用するための認証を行う Authentication Service から構成される．WoT Service は，5.3 節で述べた建物メタデータと，対応するクラス・スキーマと組み合わせて URI と Thing Description（TD）を生成し，WoT Service は TD をもとにエンドポイントを生成する（図 7）．

Data Extraction Service は，バッチレイヤで保存された

データ抽出のための非同期 API を提供している．リクエストがあった時点で，サーバクラスタが起動することで，稼働時間を最小限にし，コスト最小化を実現する．

Authentication Service は，上記の 2 つのサービスを利用するため，OAuth などによる利用者認証 API を提供する．

7. 実装と評価

設計要件に基づいて，futaba の実装を行った（図 8）．

7.1 スピードレイヤ

図 9 に開発した BACnet ゲートウェイを示す．ReadProperty や WriteProperty といった BACnet サービスを使って，収集対象のポイントの値を定期的に収集，書き込みを行っており，それらを MQTT に変換して MQTT Broker に発出している．OSGi で動くバンドルとして設計・実装されているため，1 つのゲートウェイにモジュール化された複数の機能を実装することができる．

スピードレイヤの実装は先行研究 [20] の基盤を利用する．ECL2.0 *10 上に構築されており，VerneMQ *11 で 5 棟のデータが集約され，1 分あたり平均 16,773 件のデータを処理している．データは原則として VPN を介して収集しているが，一部の IoT 機器などは，インターネットを経由する構成があるため，認証・暗号化を行う外部連携用の Broker を経由して連携させている．Ingress / Egress Function は簡易的に NodeRED *12 で実装しており，エンドポイントに公開するために必要な名前空間の変換などを行う．建物メタデータは静的な運用を想定し，事前に生成したものをサーバ上に保存している．

7.2 バッチレイヤ

バッチレイヤは，安価で保守性の高い運用のため Microsoft Azure の PaaS を用いて構築した．Scala で実装したゲートウェイを介して Azure Event Hubs にデータを送り，PaaS の機能で 15 分ごとにキャプチャデータをファイル保存しており，それらを Azure Databricks を用いて，1 日ごとのバッチ処理で Delta Lake に変換して保存する（図 10）．Azure Databricks は Apache Spark の分散実行基盤であり，バッチジョブを実行する際に，必要な計算資源を用意し，使用が終了すると解放するため，大量の計算資源を使用しながら，課金額を最小限に抑えることが可能となる．

現状の実装では，サードパーティの要望に応じて，抽出用のスクリプトをスケジューラに登録し，抽出したデータを API を介して取得する実装となっている．データの永続化のための毎日のバッチ処理時間が 35 分，月額費用

*10 https://ecl.ntt.com/news/service/ecl2/
*11 https://vernemq.com/
*12 https://nodered.org/

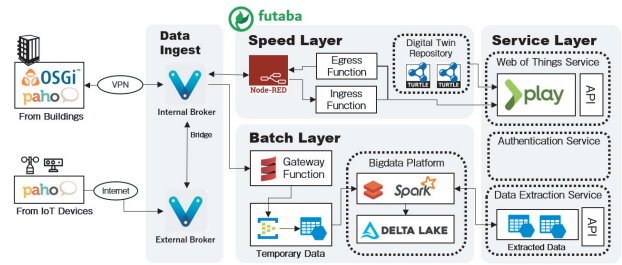

図 8 futaba 参考実装
Fig. 8 Reference implementation of futaba.

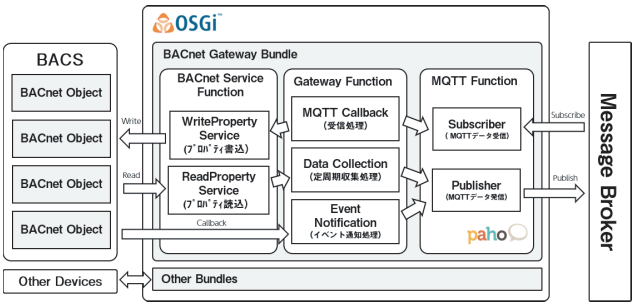

図 9 OSGi による BACnet ゲートウェイの実装
Fig. 9 Implementation of BACnet gateway with OSGi.

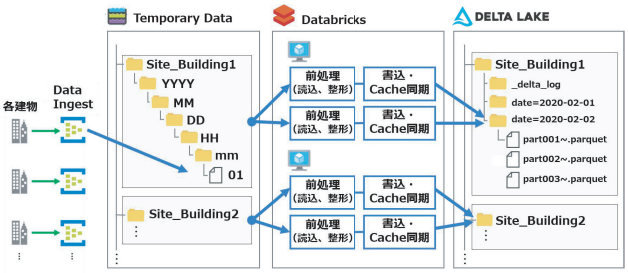

図 10 バッチレイヤでのデータ保存
Fig. 10 Store data in batch layer (after optimization).

で 2,400 円程度かかっているが，機能最適化検討の結果，Standard_L4s（32 GB メモリ，4 コア，29 ノード）での処理で 13 秒程度となり，月額 877 円程度に抑えられることが分かった．Dalta Lake フォーマットによる圧縮効率も高く，たとえば 5 万件/分のデータを想定した場合においても，250 MByte 程度となる．Azure Data Lake Storage Gen2 では，1 TB を保存しても 8,000 円程度といえ，十分なコスト効率であるといえる．

7.3 サービスレイヤ

サービスレイヤでは，WoT Service 用のサーバを Mozilla の参考実装 *13 を参考に，Play Framework *14 で実装した（図 11）．起動時に建物メタデータを解析し，そこから Thing を生成し，ThingsContiner に保持する構成になっている．Thing からメソッドを介して TD が生成できるようになっており，サードパーティは TD とクラス・スキーマ

*13 https://iot.mozilla.org/framework/
*14 https://www.playframework.com/

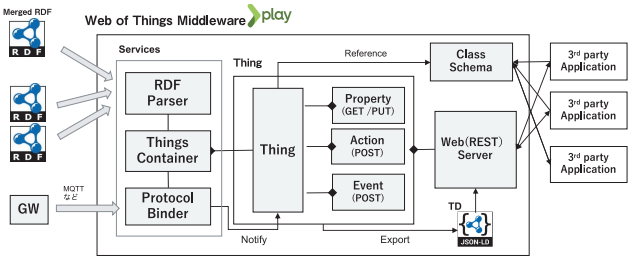

図 11 WoT Service の参考実装
Fig. 11 Reference implementation of WoT service.

図 12 Thing Description 例（ブラウザからアクセス時）
Fig. 12 Thing description (Accessing from a browser).

を参照しながら，RESTful API のエンドポイントにアクセスする．**図 12** は，エンドポイントに対してブラウザでアクセスした際に得られる TD の例である．properties 以下に，デバイスが保持するポイントの情報が書かれている．この例では，links をたどって当該プロパティのエンドポイントを特定した後，HTTP の GET でデータの取得，PUT でデータ書き込み，POST でデータ購読用のエンドポイントを生成が可能である．なお，参考実装においては，Authentication Service の実装は行っていない．

7.4 評価

本節では，4 章で述べた設計指針に対する評価を述べる．

多様なユースケース対応する高い汎用性 futaba はラムダ・アーキテクチャを採用しているため，リアルタイムおよびバッチ処理が必要なユースケースに適合する．リアルタイム処理については，ポーリングだけでなく，イベント購読用のエンドポイントの提供により，イベント駆動型のアプリケーションにも対応する．そのため，5.1 節で述べたシステム以外にも，たとえば即時的な制御が必要なパーソナルファン制御 [16] にも適用が可能である．バッチ処理についても，任意の時間間隔でデータ抽出ができるために，負荷予測やデータ分析 [23] といったユースケースにもよく適合する．柔軟なテレメトリ・スキーマを導入したことで，スカラー値以外にも対応した．近年は位置情報以外にも画像処理によって得られる特徴量ベクトルなども IoT のデータとして扱うことがあり，そのようなユースケースにも対応可能である．

再利用性の高いアプリケーション BOT や Brick による簡易なデータモデルを提案・採用し，BIM やポイントリストから生成した RDF をもとに，WoT のエンドポイントを生成している．メタデータとクラス・スキーマを公開することで，データの構造や意味についての理解が容易になり，BACS の専門業者以外の新規参入が容易となったといえる．

また，WoT による RESTful な疎結合型のアーキテクチャを採用し，ユースケースに適合した API を提供したことで，制御やモニタリングといったアプリケーション機能のモジュール化が容易になり，再利用性が高まったといえる．

建設プロジェクトへ適用可能 オープンシステムと PaaS を前提したことで，7.2 節に示したように，十分に安価な運用が可能であることを検証した．

保守性向上のため，BACS と futaba のデータ連携はゲートウェイを介して行う構成とし，遠隔制御の際にはルールエンジンである Ingress/Egress Function や認証機構によって不正制御防止がなされる構成としている．インターネット接続やプラットフォーム側に不具合があった際も，現地の BACS が問題なく動作するような機能配置としており，責任範囲の明確化を実現している．そのため，実際の建設プロジェクトにおいても十分に適用可能な構成といえる．

8. アプリケーション

本章では，futaba の参考実装を利用して開発したシステムについて述べ，ユースケースの適合性について評価する．

対象ビルの EQ House は延床面積 88 m^2 の展示施設・住居で，電気自動車が普及した際の新しい住居の在り方を提案するコンセプトハウスであり，多くのセンサを備えている（**図 13**，**表 2**）．小規模の建物であるが，全ポイント数は 1,304，空調関連だけで 854 あり，一般的な中小ビルと同等以上のポイントを有し，家よりもビルに近い設備構成となっている．空調関連のポイントに関連するデバイスは 96，それらに対応するクラスは 20 である．**図 14** にそれらを構造化して，可視化した RDF を示す．Building が Space を持ち，Space には Device が配置され，Device が Point を持つことが見てとれる．これらのデータは現地のゲートウェイを経由して MQTT でクラウド環境に発出され，また外部からの制御を受け入れることもできる．

EQ House では，5.1.2 項で述べた強化学習による省エネと快適性を指標とした設備制御を試行している．提示した

図 13 EQ House 外観（井上登写真事務所）
Fig. 13 Exterior of EQ House.

表 2 EQ House の計測ポイント
Table 2 EQ House measurement points.

気象ステーション	風向・風速・雨量・気圧・日射量
多機能人感センサ	在/不在・人数・表面温度・照度
表面温度計	パネル表面温度
CO2 センサ	CO2 濃度
スマートウォッチ	加速度センサ・心拍・評価など
電力量計	電力量
マグネットセンサ	扉の開閉状況
輝度カメラ	輝度・明るさ感
カメラ	画像解析による人・行動検知
音声認識用マイク	音声認識による会話・語彙抽出
PV・蓄電池関連	充電量，放電量など
空調機関連	風量，送風温度，熱源の効率など

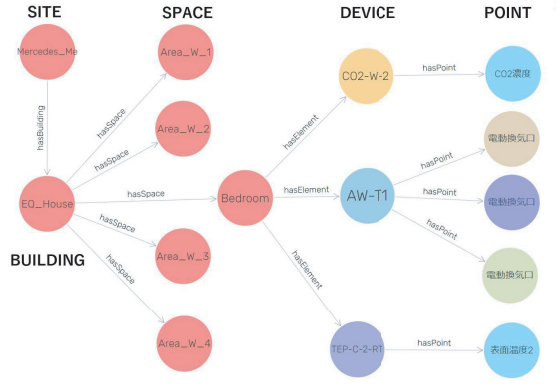

図 14 EQ House におけるセマンティクス（抜粋）
Fig. 14 Semantics of EQ House (excerpt).

要件をもとに開発した AI は 696 の計測ポイントをバッチ処理によって定期的に抽出し，予測用 AI の学習に利用している．同様にリアルタイムのデータも取得しながら，179 の制御ポイントを対象とした強化学習による制御を行う．

図 15 は，温熱快適性の評価指数である PMV（Predicted Mean Vote）を，試験的に環境悪化させた状態から，簡易的な AI 制御によって快適範囲である ±0.5 の範囲まで回復させている様子を示している．なお，これらは強化学習の

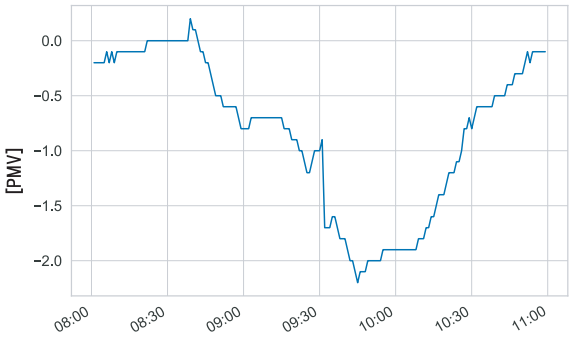

図 15 Archiphilia Engine による PMV の最適化
Fig. 15 Optimizing PMV with Archiphilia Engine.

前段階の教師あり学習モデルによる制御試行のデータである．制御ポイントの大部分を占める照明制御などは，温熱環境に無関係といえるため制御対象外とし，空調関連の 18 ポイントに限定している．ここでの PMV の計算は室温，湿度などが支配的であり，これらを制御対象とする空調システムの時定数は 15〜30 分である [24]．図中の PMV の反応はそれより遅いといえるが，通常冷房では 1 時間程度予冷運転をかけるため，実用上は問題ないといえる．

本節で述べたシステムは 2019 年 3 月から継続的に運用され，学習および制御アルゴリズムの改善を続けている．EQ House ではほかにも，主として演出を目的として，音声入力による指示やカメラによる人物・姿勢検出をトリガとした環境制御を実装している．これらの制御データも，他の設備情報と同様に MQTT を介してプラットフォームに送信・保存され，モニタリング可能であるとともに，クラウドからの制御モード変更などのアプリケーションも運用している．十分な汎用性を持つといえるだろう．

9. まとめ

本研究では，スマートビル専用のプラットフォームである futaba を提案した．ラムダアーキテクチャと WoT の仕様に基づく RESTful API を用いることで，多様なユースケースに対応した高い汎用性を得た．また，スマートビルに有用なセマンティクスの提案を行い，それらをデータモデルに組み入れることで，BACS の専門業者以外の新規参入障壁を下げ，アプリケーションの再利用性を向上させた．加えて，PaaS による機能最適化により，たとえば毎分 5 万件のデータを想定した場合でも，バッチレイヤ部が月額 1 万円以下で運用可能であることを確認した．さらには，実物件において参考実装を適用し，その汎用性を検証した．

本論文では参考実装を示したが，2019 年度の国立研究開発法人新エネルギー・産業技術総合開発機構（NEDO）の助成事業において，futaba の実用化・事業化を目指し再実装を進めている．これらの成果により，高度なスマートビルの普及と，多様な業種の新規参入による新たな市場の誕生と拡大を期待している．

謝辞 本研究における調査などに多大な協力をいただいた東大グリーンICTプロジェクトBIM基盤WGのメンバに深甚の謝意を表します．また，この成果の一部はNEDOの助成事業の結果得られたものです．

参考文献

[1] 豊田武二：ISO16484-1 BACSプロジェクトの概要，電気設備学会誌，Vol.33, No.2, pp.92–95 (2013).

[2] Dawson-Haggerty, S., Jiang, X., Tolle, G., et al.: SMAP – A Simple Measurement and Actuation Profile for physical information, pp.197–210 (01 2010).

[3] Dawson-Haggerty, S., Krioukov, A., Taneja, J., et al.: BOSS: Building operating system services, *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation* (*NSDI 13*), pp.443–457, Lombard, IL (2013).

[4] Krioukov, A., Fierro, G., Kitaev, N. and Culler, D.E.: Building Application Stack (BAS), *Proc. 4th ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, *BuildSys '12*, pp.72–79 (2012).

[5] Fierro, G. and Culler, D.E.: XBOS: An Extensible Building Operating System, Technical Report (2015).

[6] Balaji, B., Bhattacharya, A., Fierro, G., et al.: Brick: Towards a Unified Metadata Schema For Buildings, *Proc. 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*, *BuildSys '16*, pp.41–50 (2016).

[7] Fierro, G., Pritoni, M., Abdelbaky, M., et al.: Mortar: An open testbed for portable building analytics, *ACM Trans. Sensor Networks*, Vol.16, No.1 (2019).

[8] Andersen, M.P. and Culler, D.E.: BTrDB: Optimizing storage system design for timeseries processing, *Proc. 14th USENIX Conference on File and Storage Technologies* (*FAST 2016*) (2016).

[9] Fierro, G. and Culler, D.E.: Design and Analysis of a Query Processor for Brick, *ACM Trans. Sensor Networks*, Vol.14, No.3-4, pp.1–25 (2018).

[10] Ochiai, H., Ishiyama, M., Momose, T., et al.: FIAP: Facility information access protocol for data-centric building automation systems, *2011 IEEE Conference on Computer Communications Workshops* (*INFOCOM WKSHPS*), pp.229–234 (2011).

[11] Khamphanchai, W., Saha, A., Rathinavel, K., et al.: Conceptual architecture of building energy management open source software (BEMOSS), pp.1–6 (2014).

[12] Vicari, N., Kurt, G., Yalcin, B.O., et al.: Building as a Service – BaaS Deliverable D05 – BaaS Reference Architecture, pp.1–144 (2016).

[13] Lilis, G. and Kayal, M.: A secure and distributed message oriented middleware for smart building applications, *Automation in Construction*, Vol.86, No.June 2017, pp.163–175 (2018).

[14] Jara, A.J., Olivieri, A.C., Bocchi, Y., et al.: Semantic Web of Things: An Analysis of the Application Semantics for the IoT Moving towards the IoT Convergence, *Int. J. Web Grid Serv.*, Vol.10, No.2/3, p.244–272 (Apr. 2014).

[15] Bovet, G. and Hennebert, J.: Distributed semantic discovery for web-of-things enabled smart buildings, *2014 6th International Conference on New Technologies, Mobility and Security – Proc. NTMS 2014 Conference and Workshops* (2014).

[16] 粕谷貴司，田中規敏：IoTを活用した建物制御システム，電気設備学会誌，Vol.38, No.8, pp.466–469 (2018).

[17] 粕谷貴司：最新のIoT/AIを活用したスマートビルの実施例：EQ HouseにおけるAIによる無人制御への挑戦（特集AI・IoTを活用した建築・設備における新たな価値の創出），BE建築設備，Vol.70, No.12, pp.24–30 (2019).

[18] Han, M., Zhang, X., Xu, L., et al.: A review of reinforcement learning methodologies on control systems for building energy (2018).

[19] 粕谷貴司，岡野健二，茂手木直也，畠山英之，矢内雅浩，小松孝司，大野翔平：ビッグデータリアルタイム解析による建物管理システム，空気調和・衛生工学，Vol.89, No.12, pp.65–70 (2015).

[20] 粕谷貴司，近藤正芳，茂手木直也，松岡康友，矢野 雅，秋山貴紀，境野 哲，貞田洋明，堀越 崇，畠山英之：スマートシティのためのMQTTプラットフォームの検証，情報科学技術フォーラム講演論文集 (2014).

[21] Bhattacharya, A.A., Ploennigs, J. and Culler, D.E.: Short Paper: Analyzing Metadata Schemas for Buildings: The Good, the Bad, and the Ugly, *BuildSys@SenSys* (2015).

[22] Marz, N. and Warren, J.: *Big Data: Principles and best practices of scalable realtime data systems*, Manning Publications (2015).

[23] Kasuya, T., Takai, T. and Esaki, H.: Building Activity Profiling: Explainable and Predictive Modeling for Building Automation, *2020 International Conference on Artificial Intelligence in Information and Communication* (*ICAIIC*), pp.242–247 (2020).

[24] 中村政治：IT時代の計測・制御技術の動向 (5) IT時代の空調制御理論と制御系設計ツール，空気調和・衛生工学，Vol.80, pp.533–542 (2006).

粕谷 貴司 （正会員）

2007年慶應義塾大学大学院政策・メディア研究科修士課程修了．2008年（株）竹中工務店入社．2013年より同社情報エンジニアリング本部．ビル設備に対する情報エンジニアリング業務や研究に従事．2017年より東京大学大学院情報理工学系研究科に社会人博士として所属．2020年博士課程修了．博士（情報理工学）．

江崎 浩 （正会員）

1987年九州大学大学院工学研究科電子工学専攻修士課程修了．同年（株）東芝入社．1990年米国ニュージャージ州ベルコア社．1994年コロンビア大学・客員研究員．1998年東京大学大型計算機センター助教授．2001年同大学大学院情報理工学系研究科助教授．2015年同大学院同研究科教授，現在に至る．博士（工学，東京大学）．MPLS-JAPAN代表，IPv6普及・高度化推進協議会専務理事，WIDEプロジェクト代表，JPNIC副理事長．