スマートビル
システムアーキテクチャ
ガイドライン

独立行政法人情報処理推進機構
デジタルアーキテクチャ・デザインセンター
スマートビルプロジェクト

2023年（令和5年）4月21日 パブリックコメント版

改定履歴

改定年月日	改定箇所	改定内容
2023年4月21日	-	パブリックコメント版発行

目次

1. はじめに 1
1.1. 目的 1
1.2. スコープ 1
1.3. 本ガイドラインの見直し 1
1.4. 用語一覧 2
1.4.1. スマートビルのデータ 2
1.4.2. 要請の程度を表す語句 3
2. システムアーキテクチャ 4
2.1. スマートビル全体像 4
2.2. システム構成 5
2.3. 協調領域 6
2.4. 接続パターン 7
2.4.1. 連携GWを経由する場合 8
2.4.2. 連携GWを経由しない場合 9
2.5. フィールド層 10
2.5.1. 構成要素 10
2.5.1.1. ビルOS連携ゲートウェイ(連携GW) 10
2.5.1.2. 中央監視設備 11
2.5.1.3. 管理対象機器 11
2.5.2. ネットワーク 11
2.6. データ連携層 12
2.6.1. ビルOS 12
2.6.2. データフロー 14
2.6.2.1. データの保存 14
2.6.2.2. データの提供 14
2.6.2.3. コマンドの送信 15
2.6.2.4. システム連携 16
2.6.3. 機能説明 16
2.6.3.1. データ送受信モジュール 16
2.6.3.2. データ管理モジュール 18
2.6.3.3. 建物デジタルツインモジュール 19
2.6.3.4. データ連携モジュール 19
2.7. アプリケーション層 21
3. データモデル 23
3.1. 建物データモデル 23
3.1.1. 背景と現状課題 23
3.1.2. データのセマンティクスと空間モデルの必要性 23
3.1.3. モデリングの基本方針 24
3.1.4. 協調領域として規定する範囲 24
3.2. 建物データモデルの概念モデル 25
3.2.1. 概念モデルの構成要素 25
3.2.2. 概念モデルの要件 26
3.3. 建物データモデルのオントロジー 31
3.4. 建物データモデルの生成 32
3.5. 識別子 33
4. インターフェース 34

4.1. 概要 34
4.2. 基本方針 34
4.3. 関連技術 35
4.3.1. Azure Digital Twins....35
4.3.1.1. システム構成におけるADTの位置付け....35
4.3.1.2. 機能....35
4.3.1.3. 参考情報....38
4.3.2. Web of Things....38
4.3.2.1. システム構成におけるWoTの位置付け....38
4.3.2.2. 機能....38
4.3.2.3. 標準APIの仕様例 39
4.3.2.4. 参考情報....41
5. 非機能要求 42
5.1. 概要 42
5.2. 参照すべきガイドラインなど....42
5.3. 検討すべき事項....43
5.3.1. 非機能要求項目....43
5.3.2. 非機能要求検討プロセス....43
6. Appendix....44
6.1. ビルシステムの代表的な通信規格....44
6.2. 建物データモデルの補足事項....44
6.2.1. 監視点の導出背景....44
6.2.2. 監視点の実装の考え方....45
7. リファレンス 46

1. はじめに

1.1. 目的

本ガイドラインは、スマートビルの設計において必要な情報が得られるように、システム概念や技術的な仕様についてまとめている。

1.2. スコープ

本ガイドラインでは、スマートビルの全体システム構成、各構成要素、要素間の関係性、それぞれの有すべき機能などを中心に解説する。また、スマートビル構築に不可欠なデータモデルやインターフェースについても個別の検討事項として取り上げている。

各記載内容はスマートビル総合ガイドラインを前提としているため、基本的な概念については当該ガイドラインの参照が必要である。

1.3. 本ガイドラインの見直し

本ガイドラインは、関係機関・企業における現時点の技術・知見等を取りまとめたものであり、今後の運用実態や新たな技術・知見等の創出を踏まえ、より良いガイドラインとなるよう適宜見直しを行う。

1.4. 用語一覧

本ガイドラインにて用いる主要な用語について、以下のとおり定義する。総合ガイドラインで定義した用語も前提として用いているため、適宜そちらの用語一覧も参照すること。

表 1　用語の定義

用語	説明
BACS	Building Automation and Control Systemの略称。ビルにおける電気・照明・熱源・空調・給排水衛生・防災・防犯などの各設備を統合的に監視制御し、効率的で安全な設備管理や運用を実現するためのシステムやエンジニアリング技術の総称を表す。
GW（連携GW）	2.5.1.1.ビルOS連携ゲートウェイ(連携GW) 参照のこと。
MSI	スマートビル構築・運用ガイドライン参照のこと。
コントローラ	フィールド側に設置されている各種設備をローカル通信により物理的に集約管理したシステム。基本的には設備ベンダごとに用意されており、設備を監視/操作する機能を有する。 スマートビルにおけるコントローラとしては、集約した設備情報をBACnetのようなオープンプロトコルを活用してビルOS連携GWに中継する役割を担う場合や、設備ベンダが管理するクラウドシステムに通知する役割を担う場合が想定される。
セマンティック/セマンティクス	セマンティック/セマンティクスは、データの意味情報を表す。セマンティックWeb [1]に代表されるように、共通に定義されている語彙の利用やメタデータの付与によって、データの意味情報を特定の表現形式で記述することで機械判読可能にする。
テレメトリデータ	主として機器によって計測されたデータで、クラウドサーバー等の遠隔地に送信されるものを表す。
データモデル	現実世界の対象をデータ集合として表現するために、対象物を目的や用途に応じて適切に抽象化して、関係や構造を特定の表現形式で記述したもの。特に本ガイドラインでは概念データモデル、論理データモデル、物理データモデルによる3層スキーマの区分を前提として用いる。

1.4.1.　　スマートビルのデータ

スマートビルは種々多様なデータと関連し、建物が直接管轄するような施設、設備のデータを始め、外部システムやアプリケーションの持つ個人データとも連携を行う。スマートビルにおいて活用される代表的なデータについて、表 2に整理を行う。具体的なデータのやり取りや必要な機能については、2.6.データ連携層を参照のこと。

表 2　スマートビルの代表的なデータ（建物関連データ）

データ種別	説明	例
建物施設データ	建物施設全体の基本的な情報が表現された諸データを表す。	竣工図、建材データ、BIMデータ、設備台帳データなど。
建物アセットデータ	建物のアセットに関して記述された諸データを表す。	空間データ、機器/センサのメタデータ、各種計測値のメタデータなど。
設備稼働データ	建物が保有する機器/センサ等に記録された状態値等の諸データを表す。	空調照明等の設備機器データ、電力等のインフラデータ、モビリティデータ、IoTセンサによる環境測定データなど。
建物経営データ	建物の経営に関わる諸データを表す。	賃料や固定費等のPMデータ、テナントのPOSデータ、契約データ、ビル管理用のBMデータ、不動産データなど。
建物サービスデータ	建物で提供されているサービスについて表現された諸データを表す。	商品や食堂メニュー等のテナントサービスデータ、広告やイベント情報等のコンテンツデータ、周辺施設のサービスデータなど。
利用者データ	ビルユーザーや来訪者個人に関わる諸データを表す。	入退勤怠データ、部門データ、決済データ、購買履歴データ、アプリ内個人データなど。

1.4.2.　要請の程度を表す語句

本ガイドラインでは、スマートビルのシステムアーキテクチャに対する機能要件を記載しているが、各要件に対する要請の程度は、以下の語句を用いて表現している。

表 3　要請の程度を表す語句

用語	説明
必要	スマートビルの設計原則を満たすために必要である要件を表す。
推奨	目的や用途に応じて選択的に採用する要件を表す。

なお、本ガイドラインでは達成すべきビジョンや設計原則を念頭にアーキテクチャを検討している。そのため、現時点では明確に仕様が定まっていない機能要件についても、将来的な機能の必然性を鑑みて「必要」と要請している場合がある。また、現時点において「推奨」としている要件に関しても、今後のスマートビルを取り巻く状況の変化等に応じて、将来的に「必要」に変更する可能性もある。

2. システムアーキテクチャ

2.1. スマートビル全体像

未来社会におけるスマートビルのあるべき姿を実現するためには、スマートビルの相互運用性や拡張性の確保が不可欠である。一方従来のビルでは、アプリケーションとビルが連携する仕様は事業単位で作りこまれるケースが多く、建物間の相互接続や、相互連携による高度なデータ利活用を実現することは非常に難しい。

データやアプリの相互連携を促進するためには、スマートビルを建物とアプリからなる二元的なシステムとして捉えるのではなく、データ連携を保証する機能が介在した、3つの階層を持つシステムとして捉える必要がある。データ連携層が介在することで、データ管理やネットワーク通信、インターフェースの作りこみなど、データやアプリの相互連携に必須となる機能をビルのローカルシステムやアプリケーションから適切に委譲することが可能となり、疎結合なシステムとして全体を構成することができる。図 1ではスマートビルを構成するアプリケーション層、データ連携層、フィールド層の3階層間の関係性について概略図を表している。

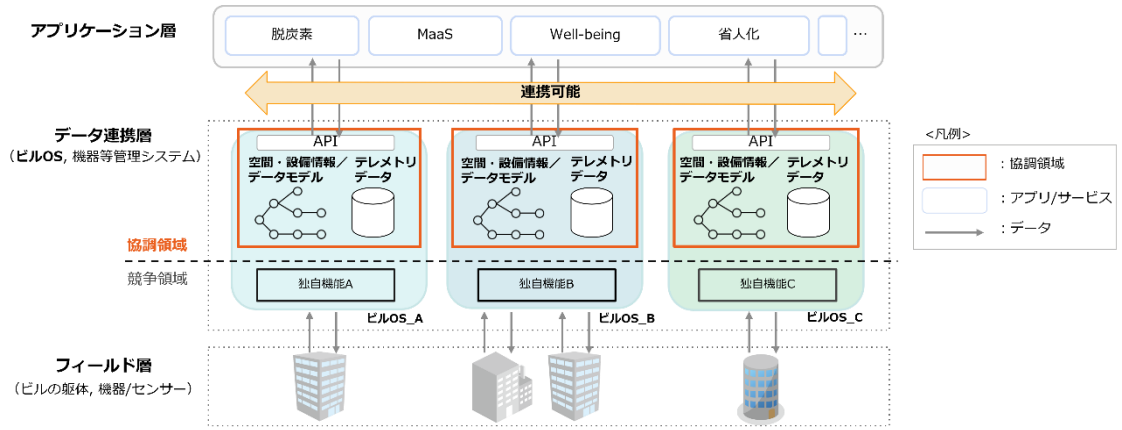

図 1　スマートビル3階層の関係性

1)　アプリケーション層

アプリケーション層はビルと連携するアプリやサービスなどが属する領域であり、建物利用者を中心とした特定のユーザーに対して価値を提供する。通信は主にデータ連携層に対して発生し、設備稼働データなど動作に必要な建物関連データ（1.4.1.スマートビルのデータ参照）を授受している。サーバはクラウド上にあることが多いと考えられるが、フィールド側（ビル内）に設置されていても構わない。

2)　データ連携層

データ連携層はフィールド層とアプリケーション層を行き交うデータを仲介する領域であり、データ連携層のシステムであるビルOS（2.6.1.ビルOS 参照）によって アプリケーションやサービスによるデータ利活用を促進する。ビルOSを中心に協調領域が設定されており、アプリへのデータ提供などの階層をまたぐ一部の通信インターフェースやデータモデルが標準化されている。（詳細は2.3.協調領域を参照）そのためアプリやサービスは共通の仕様で様々な建物関連データを取得することができ、複数のビルOSや建物をまたいだアプリケーションの展開やデータの連携が可能である。さらにビルOSはアプリからデータを

受け取り、それを他のアプリに橋渡しすることで、有用なデータの流通をアプリ間で促すこともできる。

なお、データ連携層のシステムには一部独自の機能として競争領域を有することが考えられるが、設計原則を満たすために、その機能はシステムのサイロ化を助長するものではなく、3層間の相互運用性を阻害しないようにする必要がある。

3） フィールド層

フィールド層はビルの躯体や照明、空調、計量等の設備サブシステム、中央監視やSCADA[1] をはじめとするBACS、それらを繋ぐ基幹ネットワーク（統合ネットワーク）、その他の監視カメラ、デジタルサイネージ、ロボット、IoTセンサなどが属する領域である。これらのアセットによって周辺環境をデータ化するとともに、デバイスの制御を介してユーザーに直接的な作用を行う。通信は主にデータ連携層に対して発生し、設備稼働データ等を送信するほか、データ連携層から指令を受け付けて、デバイスの遠隔制御を実行することが可能となっている。なお、デバイスの制御指令に関する一部の仕様は、データ連携層における協調領域としてインターフェースが標準化されている。このため、アプリやサービスはデータ連携層を介し、共通の仕様でデバイスの制御を実行できる。

本ガイドラインではスマートビルをこの3階層に分けて検討を行い、それぞれのシステム構成や機能を中心に取りまとめている。システム構成や機能については、スマートビルの提供価値を実現するユースケースを前提におき、アーキテクチャの考え方によって検討、導出を行った。

2.2. システム構成

協調領域を考慮した 例を図 2に示す。

フィールド層では、照明、空調をはじめとするビルの設備システム、IoT機器、中央 監視設備、それらをつなぐ通信ネットワーク、さらにデータ連携層への通信を束ねるビルOS連携ゲートウェイ（以下、連携GW）などから構成される。連携GWはビルの設備稼働データの大半を一元的に収集してデータ連携層に送信する。また、建物によっては連携GWが設置されていないケースもあり、その場合は各種設備やIoT機器は独自にアプリケーション層の設備クラウドやIoTシステムと通信を行い、当該アプリケーション経由でデータ連携層と接続される。詳細は2.4.接続パターン及び2.5.1.1.ビルOS連携ゲートウェイ(連携GW) を参照のこと。

データ連携層は基本的にはビルOSによって構成され、ビルOSは各層との通信の仲介やデータの保存等を担う複数のモジュールによって成立する。詳細は 2.6.1.ビルOS を参照のこと。ただし、本ガイドラインで記載するデータ連携層のシステム構成やビルOSの内部構成、データフローは一例であり、この通りに実装することを強制するものではない。一方で後述する機能要件に関しては、データ連携層やビルOSがどのような構成であっても、「必要」と定義している要件は何らかのモジュールで満足するように実装することが求められる。

[1] Supervisory Control And Data Acquisition　産業制御システムの一種であり、システム監視とプロセス制御を行う。対象プロセスは、製造・生産等の生産工程や、水道・ガス・送電網等のインフラ、さらにはビルや空港等の設備である。

アプリケーション層は省エネアプリ、就業者アプリなど様々なアプリケーション、サービスによって構成され、これらの連携するシステム群はビルOSを介してデータの取得やフィールド層のデバイスの操作を実行する。

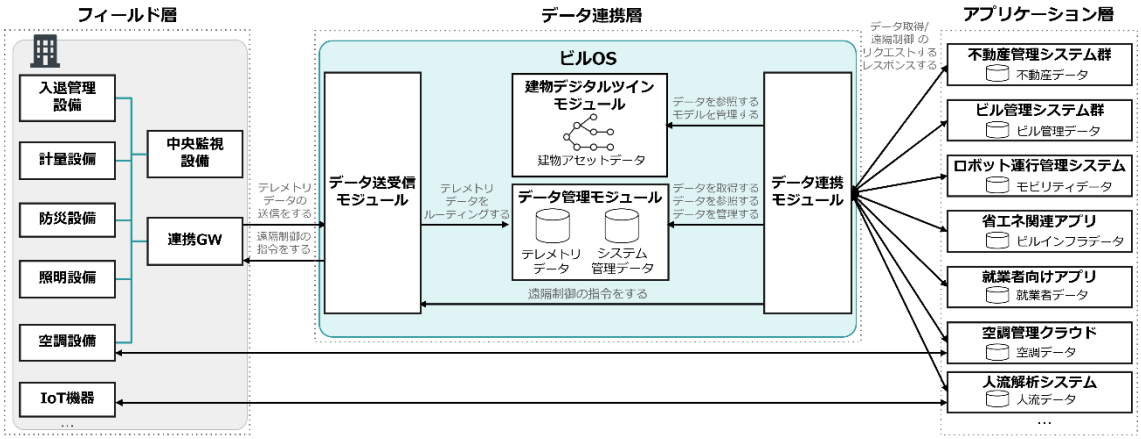

図 2　スマートビルのシステム構成図

2.3.　協調領域

協調領域はシステムを全体で最適化するために、ステークホルダー間で共通的に検討・実装すべき領域である。 システムアーキテクチャでは、スマートビルの構成要素に対して適用される標準的な仕様などが該当し、例えば特定の通信仕様やデータモデルの仕様などを対象に規定される。また、ビルOSや連携GWなど各構成要素はスマートビルの設計原則（スマートビル総合ガイドライン 3.設計原則）を満たすための機能要件を抽出しているが、このうち標準的に備えるべきものを「必要」と定義しており、これも協調領域に相当する。これらは、今後スマートビル実現のための詳細検討や市場ニーズの変化等に応じて、「必要」「推奨」の要請程度を見直す可能性や、「必要」「推奨」の要件を新規に追加、又は削除する可能性に留意すること。

協調領域の全体像は、導出したシステム構成（2.2.システム構成 参照）を前提として以下の図 3に整理している。

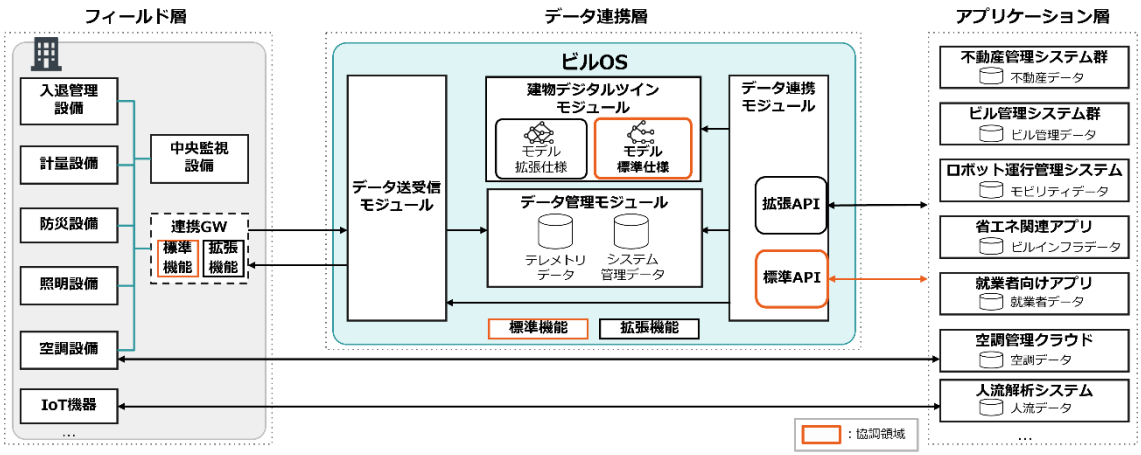

図 3　スマートビルの協調領域

1） 標準API

標準APIはビルOSの基本的な機能を受け持つAPIに対して適用される。標準APIによって、アプリケーションは統一的な仕様でビルOSにアクセスすることが可能で、例えば複数のビルOSがそれぞれ管轄しているビルに対してデータの要求を一様に実行できる。標準APIの基本的な方針については、4.2.基本方針を参照のこと。対象となる機能や具体的な仕様については将来的な検討事項とする。

2） 標準データモデル

標準データモデルはデータ連携層における建物データモデルを対象としており、主に概念モデルの要件を規定する。標準データモデルが存在することにより、データのセマンティクスが担保され、アプリケーションは取得したデータが意味する内容を解釈できる。即ち、データを二次的に利用する際の価値が高められる。詳細は3.1.建物データモデルを参照のこと。

3） ビルOSの標準機能

データ連携層のビルOSにおいて、準拠すべき機能要件に対して適用される。ビルOSが標準機能を持つことで、アプリケーション層へのデータ連携機能や、認証や権限に基づく基本的なセキュリティ機能が担保される。詳細は2.6.3.機能説明を参照のこと。

4） 連携GW の標準機能

フィールド層の連携GW において、準拠すべき機能要件に対して適用される。連携GWが標準機能を持つことで、ビルOSとフィールド層との基本的なデータ授受が担保される。詳細は2.5.1.1.ビルOS連携ゲートウェイ(連携GW) を参照のこと。

2.4. 接続パターン

スマートビルのシステム構成としての接続パターンを記載する。また、それぞれの責任分界点の考え方も記載する。

スマートビルのシステムアーキテクチャは3層で構成されるが、基本的なシステム構成としての接続パターンは、データをビルOSに集約管理する構成と、データを複数のクラウドシステムで分散管理する構成の2種類が考えられる。集約管理は、フィールド層の各種設備に関するデータが連携GWを介してビルOSに集約するシステム構成となる。分散管理は、連携GWを経由せずに、フィールド層の各種サブシステムが独自の設備クラウドやIoTシステムを介してビルOSとデータ連携するシステム構成となる。この場合、ビルOSのブローカー機能(2.6.3.4.データ連携モジュール 参照)によってデータ連携を実現することが考えられる。

いずれのパターンにおいても、アプリケーション層のアプリケーション/サービスは、ビルOSが提供する標準APIを介して標準化されたデータにアクセスして利活用することが可能である。

責任分界点の考え方としては、例えばアプリケーションに不具合が生じた際、その欠陥がアプリケーションではなくビルOSや連携GW、又はローカル設備に起因していることも考えられる。アプリケーションの機能は一義的にはアプリ・サービス事業者が責任を持つ一方で、実際には接続パターンや事象に基づき、システム全体で責任を判断しなくてはな

らない。このため、 MSIなど適切に問題の切り分けを行う役割が必要となる。上記のMSIの切り分け機能の分担については、都度契約によって明らかにする必要がある。本項では、原因が特定されている場合の責任分界点について整理を行う。

各層の構築責任も含めた接続パターンはスマートビル化されるビルの構成により相違があり、特にフィールド層の構築パターンによって類型化される。

2.4.1. 連携GWを経由する場合

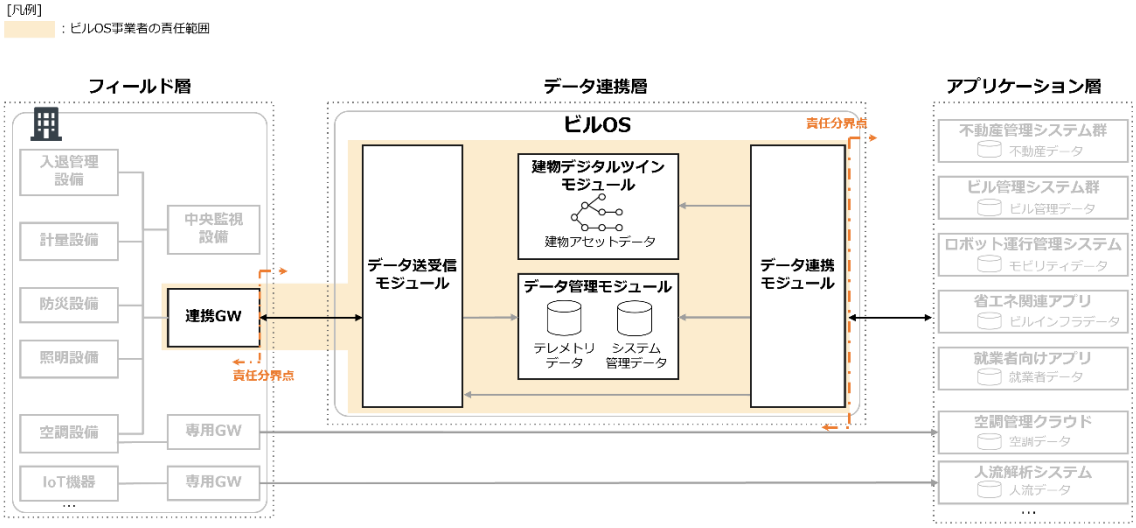

図 4　責任分界点のイメージ（連携GWを経由する場合）

1）　システム構成

ビルのフィールド層に設置された設備サブシステムと、データ連携層に存在するビルOSは、連携GW（2.5.1.1.ビルOS連携ゲートウェイ(連携GW) 参照）を介して接続される。また、データ連携層と各種アプリケーションは、ビルOSのデータ連携モジュール（2.6.1.ビルOS参照）が備えるAPIにて接続される（図 4）。

2）　責任分界点

フィールド領域の事業者とビルOS事業者の責任分界点はビルのフィールド層に設置された連携GWとなり、フィールド領域のBACSとの間に境界が存在する。ビルOS事業者等のサイバー面での事業担当者は連携GWの開発保守について責任を負う。また連携GWはアプリケーションからの制御信号に基づいて各種設備の制御を実施するケースもあり、問題を切り分ける際の分界点となる。

データ連携層と各種アプリケーションは、ビルOSのデータ連携モジュール（2.6.1ビルOS参照）が持つAPIに責任分界点が存在する。連携用のAPIについては、ビルOS事業者は提供するAPIの仕様やその開発保守についての責任を持ち、個々のアプリ・サービス事業者はAPIのレスポンスデータを利用したアプリケーションの開発保守について責任を持つ。また、システム連携（2.6.2.4.システム連携 参照）のケースなど、ビルOSとアプリケーションにおけるクライアント・サーバーの立場が逆転する場合は、それぞれその立場に従った責任を負う。

2.4.2. 連携GWを経由しない場合

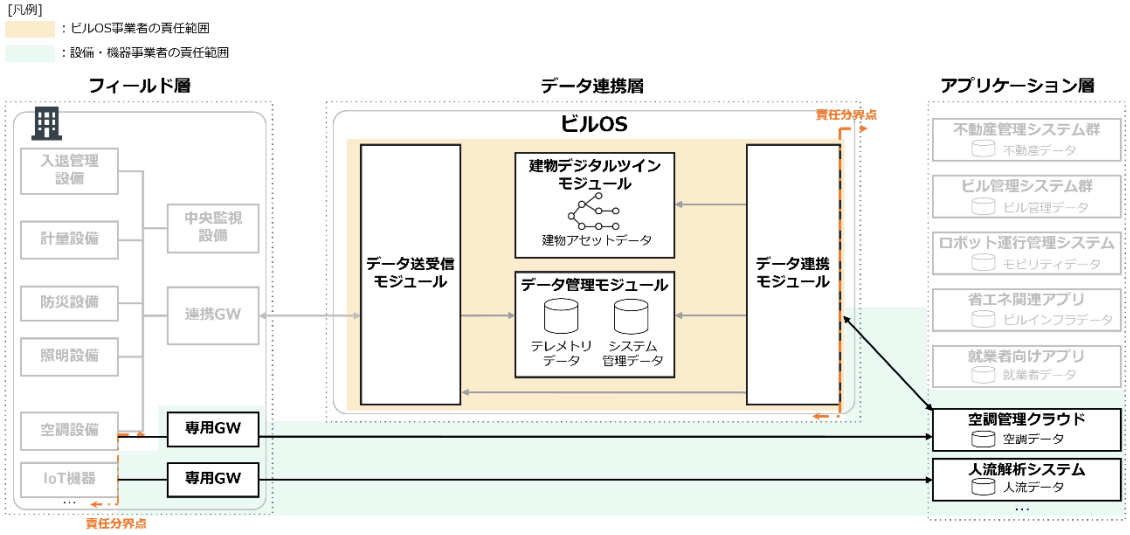

図 5　責任分界点のイメージ(連携GWを経由しない場合)

1)　システム構成

フィールド層の空調機器やIoT機器などのサブシステム毎に専用のGWが設置され、連携GWを経由せずに独自の設備クラウドやIoTシステムを介してビルOSとデータを連携する。また、ビルOSと各種アプリケーションは、ビルOSのデータ連携モジュール （2.6.1.ビルOS）が備えるAPIにて接続される（図 5）。専用GWを通過する場合はビルOSと異なるシステムを経由するが、システム間でセキュリティの構成やレベルに大きな差異が生じないようにすることが求められる。

なお、アプリケーション領域とデータ連携領域の接続において、アプリケーションが有するソフトウェアGW等を介し、データ送受信モジュールをインターフェースとしたデータ授受が行われる場合もある。

2)　責任分界点

本パターンの責任分界点はフィールド側のデータ伝送部となり、サブシステムの事業者が専用GWや設備クラウド/IoTシステムの開発保守について責任を負う。なお、図 5においては、それらのシステムはアプリケーション層と位置付けて表現しているが、システム連携(2.6.2.4.システム連携 参照)のケースのようにフィールド層のデータを仲介するなど、データ連携層の機能に類似した振る舞いをすることも考えられる。

また、それぞれのアプリケーションとビルOSがデータを交換するAPI部分にも責任分界点が存在し、連携用のAPIについては、ビルOS事業者は提供するAPIの仕様やその開発保守についての責任を持ち、サブシステムの事業者はAPIのレスポンスデータを利用したアプリケーションの開発保守について責任を持つ。さらに、システム連携のケースなど、ビルOSと設備クラウド/IoTシステム間のクライアント・サーバーの立場が逆転する場合は、それぞれその立場に従った責任を負う。

2.5. フィールド層

2.5.1. 構成要素

フィールド層を構成する主要なフィジカルアセットについて以下にまとめる。その他の構成要素については、モデルビルを参照のこと。

2.5.1.1. ビルOS連携ゲートウェイ(連携GW)

連携GWはフィールド層とデータ連携層のインターフェースであり、データの集約や送受信など、下表に整理した機能群を持つ。

表 4 ビルOS連携ゲートウェイの機能一覧

項番	機能	必要/推奨
1	設備稼働データ送信機能	必要
2	コマンド送受信機能	推奨
3	遠隔アップデート機能	推奨

1） 設備稼働データ送信機能

ビル内の設備稼働データを一元的に収集し、データ連携層のビルOSに送信する機能が提供される。収集されたデータはビルOSに対応したプロトコル(HTTP、MQTT等)に変換されて送信される。

各設備のデータ収集において、ビルの各コントローラと連携GW間の通信方式は、BACnet [2]に対応していることが望ましい。一方各コントローラと設備間の通信方式は既定しないため、他のオープンプロトコルを採用することも考えられる。なお、コントローラで規定されたコンポコントローラステム構成に適用すると、連携GWはB-OWS(BACnet Operator Workstation)、それぞれのコントローラはB-BC（BACnet Building Controller)となる。既設のBACnetネットワークにビルOSを接続する場合は、アプリケーション側からの制御と、中央監視装置等によるローカル側の制御が矛盾なく整合するように、BASベンダ等と連携して協議することが求められる。

データの送信間隔は、例えば空調設備や計量設備などのモニタリングについては、60秒間隔で送信するケースなどがあるが、一義的に推奨とする間隔は定められない。一般に送信間隔を短くするとデータの精度が向上する一方で、トラフィックの増大によってシステム負荷が高まり、データの維持管理に伴うコストも飛躍的に増大する。そのため機器の特性、ユースケースやシステム要件等に応じて個別に送信間隔を設計する必要がある。

また、データ欠損防止のため、ネットワークの切断時には送信予定であるデータを一時的に保存し、復旧時にまとめて送信する機能を持つことが望ましい。ただし、そのような機能を実装する場合は、一度にまとまったデータを送信することで輻輳が発生する等のリスクが生じる可能性に留意すること。

2） コマンド送受信機能

ビルOSからコマンド（制御指令）を受信し、対象の設備にコマンドを伝送する（設定値の上書きを含む）機能が提供される。コマンドは、各設備に対応したBACnet等のプロトコルに変換された後に伝送される。

また、連携GWは責任分界点の一つとなるので、運用時のトラブルシューティング等を想定し、コマンド送受信に関する通信ログの記録機能を有することが望ましい。これは、上述の設備稼働データ送信機能においても同様である。

3） 遠隔アップデート機能

ビルOSから連携GWのソフトウェアアップデート情報を受信し、アップデートを実施する機能が提供される。なお、アップデートに際してはクラウドにフィールド領域が接続されることになるため、アップデート配信前にセキュリティ面や試験項目などについて考慮することが望ましい。

2.5.1.2. 中央監視設備

中央監視設備はフィールド側に設置される各種設備機器に対する制御、状態監視、警報監視、発停操作、スケジュール機能等を持つ。また、複数の設備間連動に関しても中央監視設備がその役割を担う事があり、ローカルで機器を一括制御できるメリットがある。昨今の大規模・中規模のビルでは一般的に導入されており、ビル管理において重要な役割を担うものと考えられている。

ビルOSが外部のアプリケーションとインターネット等を介して連携するのに対して、中央監視はクローズドなシステムとしてフィールド側のローカル通信で構築することが基本となる。昨今では中央監視設備の一部機能をクラウドに実装するものも存在するが、一般的には外部ネットワークのシステムとフレキシブルに接続することは難しく、現時点の機能のままではアプリケーションによる拡張やデータの利活用を行うことは容易ではない。中央監視設備は、ビルOSを介してアプリケーションからの指令を受け取り、ローカルの自動制御システムと連携して、機器制御の仲介役となることは可能である。

2.5.1.3. 管理対象機器

フィールド側で管理される機器としては、入退管理設備、計量設備、空調設備、照明設備等、ビルに一般的に導入される設備機器や、設備ごとのコントローラが想定される。また、IoT機器も管理対象であり、環境センサをはじめとする無線センサ、BLE [3]などを用いたビーコン、ネットワークカメラなどがある。IoT機器は、デバイスに直接的にネットワークポートを有する場合、独自の無線ネットワークを構成し、ゲートウェイを経由する場合がある。ゲートウェイを通過する場合は、データの管理を一元化できるといったセキュリティ上のメリットが存在する。連携GWと通信する場合は、統合ネットワーク等に接続されることになる一方で、LTE [4]などを介して独自SaaSにつなげるケースもある。統合ネットワーク等を経由せず、独自SaaSを介して連携される場合はクラウド間でのデータ連携が想定される。

2.5.2. ネットワーク

スマートビルにおいてはデータ連携層やアプリケーション層のシステムがクラウドサービスとして実装されるケースも多く、フィールド層のシステムは外部ネットワークと接続可能な構成とすることが大半である。

機器の統制管理においては、拡張性を担保するために統合ネットワークの採用が好ましい。ネットワークの構築方法は、統合ネットワークで一般的なスター型だけでなく、リング型などの採用も建物規模によっては事例が増えてきている。統合ネットワークを構築することでセキュリティポリシーの統一化も図れるが、例えばBACnetを使う場合、ルーティングをする際にBBMD (BACnet Broadcast Management Devices) [2] などの追加の設備が必要になるなど、エンジニアリングの難易度は上昇する。

なお、統合ネットワークを構築しない場合でも、ビルOSとの連携は可能であるが、サブシステムごとに独立しているため、拡張性を担保するための考慮が求められる。

また、スマートビルのシステムではIoT機器が接続されることを想定しているため、照明や空調などの設備を管理する制御系ネットワークや、事務処理用の通信機器を管理するOAネットワークだけでなく、IoT機器を管理するIoTネットワーク等を導入することも、ネットワーク形態の一例として考えられる。

ネットワーク全体として、外部ネットワークからの不正な通信を検知、遮断する機能が実装される必要がある。なお、通信セキュリティに関連するこの機能はファイアウォール（FW）などの専用の機器によって担保するケースも多く、何らかのネットワーク構成要素によって具備されているべき機能である。

2.6. データ連携層

2.6.1. ビルOS

ビルOSはスマートビルのアセットを抽象化し、適切にモデリングことで、多様なサービスを汎用的に提供するためのプラットフォームとして捉えられる。また他の基本ソフトと同じように、APIによってアプリケーションやソフトウェア、Webサービスの間をつなぐインターフェース仕様を提供するという側面もある。同時にビルOSは建物関連データをアプリケーションに広く連携する基盤として、システム間のデータ授受を円滑にするための機能が備えられる。

ビルOSの機能は主としてシステムリソースのマネジメントや建物関連データの保存、管理であり、デバイスの遠隔制御コマンドの送信、アプリケーションやプラットフォームとの連携、各種権限管理、設備稼働データ等の保存などを行う。これらの機能を有し、データ連携基盤として高度にデータが活用されることで、アプリケーションの開発効率やサービスレベルが向上する。またビルOSやアプリケーションのアップデートによって、スマートビルのソフトウェアを最新の状態に保ち、新しい技術により高度なセキュリティを追加していくこともできる。

ビルOSによってビル毎の違いを隠蔽して共通化した運用ができるため、ビル管理を統合することが可能となる。リモートでの制御によって設備管理者の配置の自由度が向上するなど、運営上のメリットも大きい。さらに協調領域によって統一されたインターフェース

[2] 複数の IP サブネットで構成される BACnet/IP ネットワーク全体に対して、BACnet ブロードキャストメッセージを配信するために使用するデバイス。

をビルOSが持つことで、将来的にスマートビルは、スマートシティの中で多種多様なサービスと繋がり連携する存在となりえる。

ビルOSの基本的な機能を満たすためには、アプリケーション層とフィールド層のインターフェースを備えつつ、データを管理するためのモジュールが必要となる。一例としてはデータ連携、建物デジタルツイン、データ管理、データ送受信の4つのモジュール構成が考えられる。以下の図 6では、アプリケーション層、フィールド層との関係性と共に全体の構成例を示している。なお、2.2.システム構成で記述した通り、ビルOSの内部構成やデータフローに関しては、本ガイドラインでは一例として記載しており、この通りに実装することを強制するものではない。

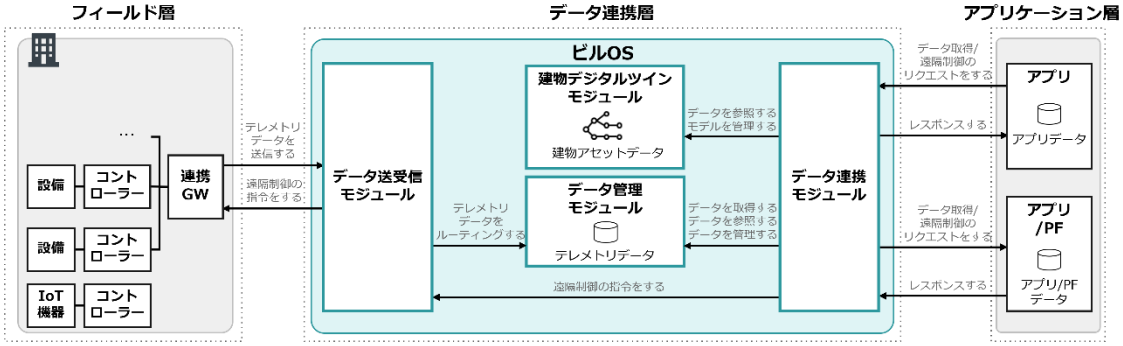

図 6　ビルOSのシステム構成例

1)　データ連携モジュール

アプリケーション層との通信インターフェースとなり、アプリケーションや設備クラウド/IoTシステムなどのプラットフォームに対するデータリクエストやレスポンスを行う。またビルOS内に存在するデータの取得や、遠隔制御の指令を仲介して各モジュールとやり取りを行う。

2)　建物デジタルツインモジュール

建物の空間情報や設備情報を中心とした建物アセットデータを保存、管理する。建物アセットデータは、構成要素間の関係性やそれぞれの要素に付与されたプロパティなど、建物データモデル（3.1.建物データモデル）に基づいて構造化されたセマンティックなデータによって構成される。

3)　データ管理モジュール

設備稼働データを中心としたリアルタイムデータや、それらを時系列に蓄積したアーカイブデータを保存、管理する。またアプリケーション由来のデータや、その他ビルOSのシステムデータなどを保存、管理する。建物アセットデータと異なり、セマンティックなデータは含まれない。

4)　データ送受信モジュール

主にフィールド層との通信インターフェースとなり、連携GWから設備稼働データ等の受信や連携GWへの制御コマンドの送信を行う。また受信したデータをデータ管理モジュールにルーティングする。

2.6.2.　データフロー

アプリケーション層、フィールド層との間に発生する主なデータフローは、データの保存、データの提供、コマンドの送信、システム連携に大別される。以下ではそれぞれのフローについて図で整理し、そのデータフローに関連するモジュールのみを強調して示している。

なお、2.2システム構成記述した通り、本項では上記に挙げた4つのデータフローに対して詳細を記述している部分もあるが、参考事例の位置付けである。また、本項ではアプリケーション層やフィールド層にまたがって通信を行うケースに限定して考慮し、システム管理者によるシステム構築時のデータフローや、保守管理におけるデータフローについては範囲に含めない。

2.6.2.1.　データの保存

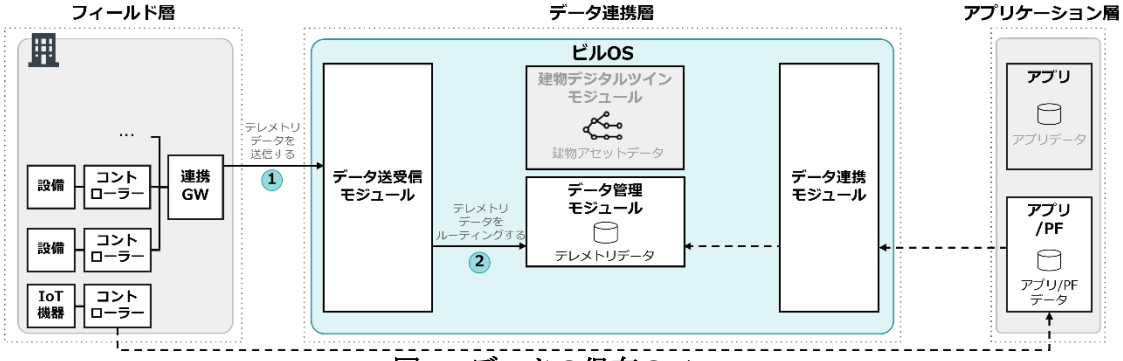

図 7　データの保存のフロー

データの保存は、主にフィールド層から受信したデータがビルOSに保存されるまでのフローを指す。まずフィールド層の連携GWからビルOSに向かって設備稼働データ等が送信され、データ送受信モジュールが受信する（①）。さらにデータ送受信モジュールからデータ管理モジュールにルーティングが行われ、データ管理モジュールにてデータの保存が行われる（②）。また、フィールド層に連携GWが無い場合のように(2.4.接続パターン)、フィールド層の設備やIoT機器のデータが、それぞれの設備クラウドやIoTシステムを介してビルOSに渡される場合もある。

2.6.2.2.　データの提供

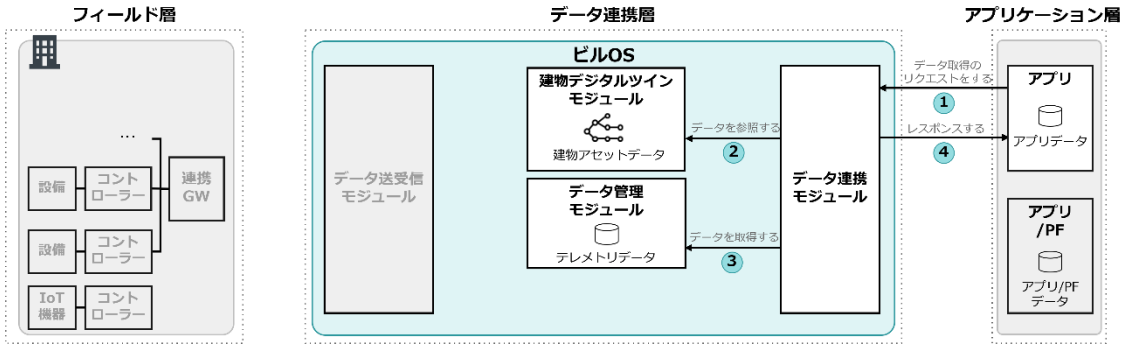

図 8　データの提供のフロー

データの提供は、アプリケーションのリクエストを契機に、ビルOSからデータがアプリケーションに提供されるまでのフローを指す。まずアプリケーションがリクエストを行い、データ連携モジュールがリクエストを受け付ける（①）。データ連携モジュールは建物デジタルツインモジュールへの問い合わせを介して取得対象となるデータを特定する（②）。その後該当のデータをデータ管理モジュールから取得し（③）、アプリケーションにデータをレスポンスする（④）。

2.6.2.3. コマンドの送信

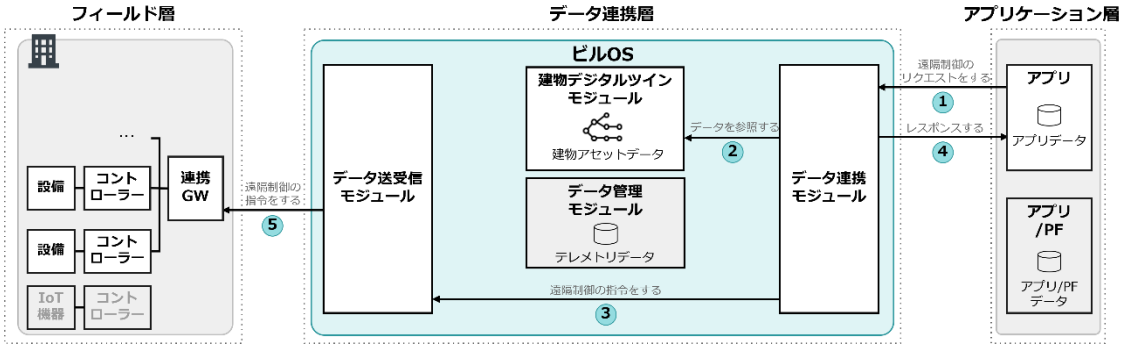

図 9　コマンドの送信のフロー

コマンドの送信は、アプリケーションのリクエストを契機に、ビルOSを介してコマンドがフィールド層に送信されるまでのフローを指す。まずアプリケーションがリクエストを行い、データ連携モジュールがリクエストを受け付ける（①）。データ連携モジュールは建物デジタルツインモジュールへの問い合わせを介して、送信対象となるデバイスを特定する（②）。データ連携モジュールはデータ送受信モジュールに向けて、デバイスの識別子とコマンドを送信する（③）。その後、データ連携モジュールは処理完了をアプリケーションにレスポンスする（④）。データ送受信モジュールはフィールド層の連携GWに向けて、デバイスの識別子とコマンドを送信する（⑤）。

デバイスの制御は完了まで一定の時間を要する処理であり、一連の処理を同期通信で行った場合はクライアント（ここではアプリケーション）の待機時間が問題になる。そのため、コマンドの送信リクエストに対する④のレスポンスは、受付完了を意味するものとして速やかに実行されることが望ましい。(詳細は2.6.3.4.データ連携モジュール参照) なお、機器制御におけるリアルタイム性については、ユースケースや機器の特性に応じた性能要件に依存しており、事例ごとに定める必要がある。

2.6.2.4. システム連携

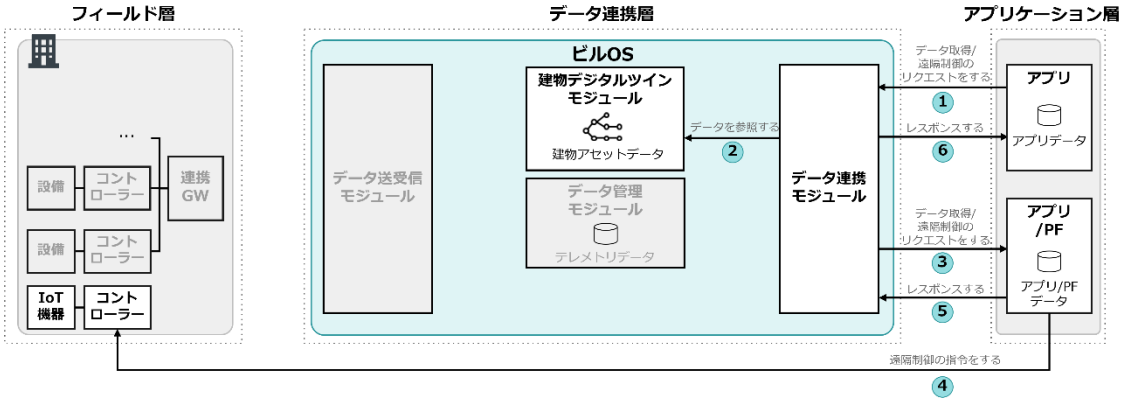

図 10　システム連携のフロー

システム連携は、アプリケーションのリクエストを契機に、ビルOSの仲介によって他のアプリやプラットフォームが管轄するデータを活用するまでのフローを指す。まずアプリケーション層がリクエストを行い、データ連携モジュールがリクエストを受け付ける（①）。データ連携モジュールは建物デジタルツインモジュールへの問い合わせを介して、取得対象となるデータや制御対象となるデバイスを特定する（②）。データやデバイスがアプリケーション層（アプリ、プラットフォーム、ストレージなど）の管轄であった場合、データ連携モジュールは対象のシステムとの通信を行い、必要なデータの取得（③）やデバイスの制御（④）のリクエストとそのレスポンスが行われる（⑤）。対象のシステムからデバイスの制御が発生する場合、IoT機器との直接の通信や、コントローラを介した通信などによって制御が行われる。最終的に、データ連携モジュールはアプリケーションにデータの返却などのレスポンスを行う（⑥）。なお、④のデバイスの制御が行われる場合、⑤のレスポンスはリクエストの受付完了を意味するものとして速やかに返却されることが望ましく、コマンドの送信におけるケースと同様である。

2.6.3. 機能説明

ビルOSのシステム構成（2.2.システム構成）において、各モジュールが持つ機能要件を以下に示す。ただし、2.2.システム構成で記述した通り、ビルOSの内部構成やデータフローは本ガイドラインで記載している通りに実装することを強制するものではないので、実際には各モジュールと機能要件の対応付けには異なる部分があると考えられる。例えば、建物デジタルツインモジュールとデータ管理モジュールを統合的に実現するケースや、より細分化したモジュール構成にするケース等が想定される。どのような内部構成であっても、下記に示す機能要件のうち「必要」と定義している要件に関しては、いずれかのモジュールで当該要件を満足するように実装することが求められる。

2.6.3.1. データ送受信モジュール

データ送受信モジュールは、主にフィールド層とのインターフェースとして機能し、デバイス通信等を行う機能群からなる。また、フィールド層に連携GWが無い場合のように(2.4.接続パターン)、フィールド層の設備やIoT機器をそれぞれの設備クラウドやIoTシステムを介してビルOSと連携する際は、本モジュールが設備クラウド/IoTシステムなどとのインターフェースとして機能することも考えられるが、外部との接続仕様の具体的な要件は今後の検討課題とする。

表 5　データ送受信モジュールの機能一覧

項番	機能	必要/推奨
1	デバイス通信（受信）機能	必要
2	データルーティング機能	必要
3	データフィルタリング機能	推奨
4	デバイス通信（送信）機能	推奨
5	連携GW遠隔アップデート機能	推奨
6	デバイス認証機能	必要

1)　デバイス通信（受信）機能

フィールド層のデバイスからビルOSに向けて送信されたデータを受信する機能が提供される。データの品質（信頼性）や接続性を担保するために、ビルOSは所定のデータ形式のデータのみを受信するように、バリデーションを行う。バリデーションは、少なくともフィールド層の連携GWや設備クラウド/IoTシステムなどとの通信において、受信データの構文チェック等が必要になると考えられるが、具体的な要件に関しては今後の検討課題とする。バリデーションの役割はビルOSや連携GWなど、スマートビルのシステム全体で適切に分担し、データの品質や接続性を担保することが求められる。

2)　データルーティング機能

受信したデータをルーティングし、適切なシステム構成要素に向けて送信する機能が提供される。データのルーティングは、主としてデータ管理モジュールの各種ストレージに向けて実施される。

3)　データフィルタリング機能

所定の閾値に従ってデータの異常値等を判定し、データのフィルタリングを実施する機能が提供される。なお、データのフィルタリングについては、連携GW（2.5.1.1.ビルOS連携ゲートウェイ(連携GW) 参照）やコントローラにおいて機能を担保することも考えられる。

4)　デバイス通信（送信）機能

ビルOSからフィールド層のデバイスに向けてデータを送信する機能が提供される。主としてデバイスへの通知、設定値変更などのコマンドの送信が行われる。

5)　連携GW遠隔アップデート機能

連携GWのソフトウェア情報を扱い、遠隔アップデートを実施する機能が提供される。

6)　デバイス認証機能

データ送信元の通信デバイスを特定し、正常な通信相手のデータのみを受領する機能が提供される。デバイスの認証ではクライアント証明書やトークンを利用した認証などが考えられる。

2.6.3.2. データ管理モジュール

データ管理モジュールは、主にフィールド層のデバイスから送信されたデータを集約し、管理する機能群からなる。また、外部のアプリケーションやプラットフォームといった連携システムのデータについても管理を行う。

表 6　データ管理モジュールの機能一覧

項番	機能	必要/推奨
1	リアルタイムデータ管理機能	必要
2	アーカイブデータ管理機能	推奨
3	連携システム情報管理機能	必要
4	連携システムデータ管理機能	推奨

1）　リアルタイムデータ管理機能

デバイスから送信された設備稼働データ等を集約し、保存、管理する機能が提供される。集約されたデータは、デバイス、監視点のIDや受信日時などのメタデータが付与され、例えば時刻や監視点のIDを指定した問い合わせに対して対象のデータを特定することが可能となる。

データのタイムスタンプは代表的なメタデータの一種であり、トラブルシューティングを始めシステムを適切に管理するために付与されていることが望ましい。タイムスタンプを付与するタイミングとしては、データの保存、又はコマンドの送信などの3層をまたぐデータフロー（2.6.2.データフロー）や、各層における構成要素間の通信などが考えられる。

2）　アーカイブデータ管理機能

デバイスから送信された設備稼働データ等を時系列のアーカイブデータとして保存、管理する機能が提供される。集約されたデータは、デバイスのIDや受信日時など必要なメタデータが付与され、データの利活用性が高められる。アーカイブデータは、主にアプリケーションが機械学習を実行するためのモデル構築や、スマートビルのサービス事業者がトラブルシューティングをするための情報源として用いられる。

3）　連携システム情報管理機能

ビルOSのデータを利活用するアプリケーションやプラットフォームなど連携するシステムの情報を管理する機能が提供される。2.6.3.4.データ連携モジュール5）権限管理機能、及び、2.6.3.4.データ連携モジュール3）ブローカー機能の対象になる情報としては、主に連携するシステムの一覧や接続先、認証・認可に関連する情報など、連携システムとの正常な接続やその管理に必要なものが含まれる。

4）　連携システムデータ管理機能

アプリケーションやプラットフォームなど連携するシステムのデータを保存、管理する機能が提供される。連携システムのデータはデータソースへ照会して取得するケースのほか、アプリケーションが作成したデータをそのアプリの要請に応じてビルOSに保存するケースも考えられる。保存したデータは別のアプリケーションに対して提供されることで、アプリケーション間のデータ利活用を促進する。連携するシステムのデータを保存する場

合は、データの提供者や、データの利用者、取得日時、提供日時などのメタデータが付与され、データの適正な利用が行われるよう管理される。

2.6.3.3. 建物デジタルツインモジュール

建物デジタルツインモジュールは、特定の概念を基に建物アセットデータを構造化し、それらをアプリケーション層から参照可能にする機能群からなる。建物デジタルツインモジュールは建物データモデル（3.1.建物データモデル）を基底として構築される。

表 7　建物デジタルツインモジュールの機能一覧

項番	機能	必要/推奨
1	建物アセットデータ参照機能	必要
2	建物データモデル管理機能	必要

1)　建物アセットデータ参照機能

建物アセットに関する構造化されたデータを、アプリケーション層から参照可能にする機能が提供される。建物アセットデータは建物データモデルの概念モデルの要件に基づいて構造化されており、空間データ、物体(デバイス)データ、監視点データから構成される。

空間データは建物自体の情報も含めた建物の空間構造に関する情報であり、敷地(土地)、建物、フロア(階)、部屋、また独自に区分けされたエリアなどからなる空間の相互関係性や、空間自体に付与されたデータ項目を参照できる。参照できる空間データはユースケースに応じたモデリングに依存するが、建物に関しては必ず参照可能な情報であり、複数の建物をまたいでグローバルに特定できる識別子を有する。なお、建物の識別子の詳細については将来的な検討事項とする。 (3.5.識別子)

デバイスデータは建物に設置されたデバイスに関する情報であり、デバイスと空間の関係性によって示されたデバイスの設置位置や、デバイス自体に付与されたデータ項目を参照できる。また、デバイスの接続、切断、発停等について識別可能であり、これらのデータを基にアプリケーションがデバイスの死活監視を実施できることが望ましい。

監視点データはデバイス等の監視点に関する情報であり、デバイスと空間との関係性によって示された監視点の所属や、監視点自体に付与されたデータ項目を参照できる。

2)　建物データモデル管理機能

フィールド層の機器構成の変更などに応じて、建物データモデル自体を作成、更新、削除する機能が提供される。建物データモデルの変更はビルOSのシステム管理者によって実行されることを想定している。また、建物に関わる一部データがアプリケーション層のシステムで管理されている場合などは、そのシステムのリクエストに応じて建物データモデルが変更される。ビルOS外のシステムで更新された情報と同期することで、建物デジタルツインは最新の状態に保たれる。

2.6.3.4. データ連携モジュール

データ連携モジュールは、アプリケーション層とのインターフェースであり、主にデータを返却する機能や遠隔制御コマンドを送信する機能群からなる。各機能はAPIとして外

部に公開するものであり、アプリケーションからリクエストを受信することで動作し、レスポンスを返却する。

レスポンスは、データ提供(2.6.2.2.データの提供)で示した例のように、ビルOS自身が保持しているデータを要求された場合は、そのデータを含む形式でレスポンスを返却する。

ただし、コマンド送信(2.6.2.3.コマンドの送信) のようにフィールド領域のデバイスを制御する場合や、大容量のアーカイブデータを取得するなど処理に時間を要するケースでは、一連の処理を同期処理として実装すると、そのユースケースで求められる性能要件を満足できなくなる可能性がある。

そのため、ビルOSがデバイスの処理完了やデータ取得の処理結果を待つことなく、アプリケーションに速やかに応答するために、非同期処理として一連の処理を実装することが考えられる。その場合、アプリケーションのリクエストに対するビルOSのレスポンスは、受付完了したことを意味するものとして返却する。アプリケーションが一連の処理の結果を確認する手段としては、アプリケーションがビルOSに対してポーリングする方法や、ビルOSからイベント通知を受け取る方法などが考えられる。

このように、ユースケースや性能要件等に応じて、ビルOSの実装方式が異なることが想定されるが、非同期処理の結果をポーリングで取得するといった、アプリケーション側の対応が必要なケースもあることに留意すべきである。

表 8　データ連携モジュールの機能一覧

項番	機能	必要/推奨
1	データ提供機能	必要
2	遠隔制御コマンド送信機能	推奨
3	ブローカー機能	推奨
4	アプリケーション認証機能	必要
5	権限管理機能	必要

1）　データ提供機能

アプリケーションによるリクエストに応じて、スマートビルの関連データ（1.4.1.スマートビルのデータ参照）をレスポンスする機能が提供される。提供されるデータとしては、主に建物アセットデータと、設備稼働データ等の動的なリアルタイムデータやそのアーカイブデータ、またアプリケーションやプラットフォームなど連携するシステムが管轄するデータなどからなる。

建物アセットデータについては建物デジタルツインの項目（2.6.3.3.建物デジタルツインモジュール）で示した通りで、建物データモデルに即して構造化された建物の情報が返却される。

動的データではリアルタイムデータやアーカイブデータがリクエストに応じて返却され、デバイスや期間の指定が可能である。デバイスは複数のデータ項目を持つ場合があるため、データ項目が指定可能で、指定された項目のみ返却されることが望ましい。また、空間が指定可能で、指定された空間と関係性を持つデータのみ返却可能とすることが望ましい。

アーカイブデータについては、受領する際のデータ形式の指定や、日次や月次で定期的にデータ取得を実行するタスクの登録が可能であることが望ましい。

連携するシステムが管轄するデータについてもリクエストに応じて返却される。アプリケーションが持つデータも含めて、ビルOSを介して使用可能であるデータについては、データの利用者に対して開示されている必要がある。

2） 遠隔制御コマンド送信機能

アプリケーションによるリクエストに応じて、デバイスへの遠隔制御コマンドを送信する機能が提供される。対象のデバイスと、実行するコマンドはどちらも指定が可能である。

また所定の閾値に従って、許可されないコマンドに対するフィルタリングが行われることが望ましい。ただし、コマンドのフィルタリングについては、ビルOSではなくフィールド層の連携GW（2.5.1.1.ビルOS連携ゲートウェイ(連携GW)）や設備において機能を担保することも考えられる。

3） ブローカー機能

アプリケーションによるリクエストを仲介し、リクエストを適切な内外システムに振り分ける機能が提供される。データ取得や遠隔制御リクエストの振り分け先を特定した後、それが連携システムの管轄であればそのシステムに問い合わせを行う。アプリケーションは連携システムのインターフェースを意識することなく、建物関連データを扱う単一の問い合わせ口としてビルOSを活用できる。

4） アプリケーション認証機能

リクエストを受領した際に送信元のアプリケーションを特定し、許可されたアプリケーションのみと通信する機能が提供される。

5） 権限管理機能

連携するアプリケーションごとに、スマートビル機能の権限について管理する機能が提供される。権限の管理により、例えばあるアプリケーションはデータ参照と遠隔制御が許可される一方で、別のアプリケーションはデータ参照のみが許可されるなど、アプリケーションの目的に応じた適切なアクセスコントロールを実現する。なお、ユーザーレベルでの操作権限の管理はアプリケーションで行うなど、権限の特性に応じてシステム全体で適切に機能を分担することが求められる。

2.7. アプリケーション層

アプリケーション層ではアプリケーションやプラットフォームが属し、それぞれ固有の価値を提供する。

基本的にこれらの要素は独自の仕様を持つが、今後一部のアプリケーションや機能がスマートビルの標準的な仕様として求められる可能性もある。例えばビルOSのシステム連携において、他のアプリケーションやプラットフォームにデータをリクエストするインターフェース仕様などはある範囲で共通化されていることが望ましい。その他公益性の高いアプリケーションが考えられるとき、その機能に必要なインターフェース仕様なども標準化の価値が高い。

このように標準アプリケーションや標準機能を持つことが業界発展に有益である場合には、協調領域を拡張してそれらを定める方針である。具体的に標準化対象とする仕様については将来的な検討事項とする。

3. データモデル

データモデルは、現実世界の対象をデータ集合として表現するために、対象を目的や用途に応じて適切に抽象化して、関係や構造を特定の表現形式で記述したものである。本ガイドラインでは、概念データモデル、論理データモデル、物理データモデルによる3層スキーマの区分を前提として記述する。

3.1. 建物データモデル

スマートビルは3層から成るアーキテクチャであるが、そのうちのデータ連携層に相当するビルOSの内部において、主に建物アセットデータや設備稼働データで構成するデータモデルを、『建物データモデル』と定義する。

3.1.1. 背景と現状課題

省エネや快適性・利便性・安全性の向上など、多様なユースケースが期待されるスマートビルにおいても、現状のビル設備管理の考え方は重要になる。特にISO16484で規定されているBACSは、国内外で普及しているビル設備管理システムの主要な考え方であり、スマートビルにおけるモデルに関しても、BACSで規定されるモデルと親和性を保つように設計することが、データモデルの受容性や汎用性の向上に繋がると考えられる。

BACSの標準プロトコルであるBACnetでは、建物内の設備機器に関するデータを、「Device・Object・Property」という階層構造でモデリングすることで、設備に関する情報を表現している。しかし現状では、「どのような機器なのか」「どの場所に設置しているのか」といったセマンティクスがデータとして表現されていないことや、データ名称などが標準化されていないので、建築・設備ドメインの有識者でなければデータの意味を理解し、データを横断的に利活用することが難しい状況である。

3.1.2. データのセマンティクスと空間モデルの必要性

スマートビルでは、建築・ビル設備ドメインの有識者に限らず、多種多様な事業者がアプリケーション開発に参入できるようにすることを一つの目的にしている。そのため、特定の知識や経験の有無に依らずにデータの意味を理解できるようにすることが必要になるので、スマートビルにおいてはデータのセマンティクスを表現できるように設計する。

また、上述の現状課題における「どの場所に設置しているのか」といった空間情報に関連するセマンティクスは、設備のプロパティ(データ項目)として表現することも可能であるが、本来的には空間と設備は全く異なる概念である。そのため、空間に関連する情報は、設備の付属情報として表現するのではなく、設備とは独立した概念である「空間」として表現することが適切である。その上で、空間と設備を互いの関係性を含めて表現する。

このようにすることで、下図に示すように、アプリケーションの開発者は「空間」単位でそれに紐づく設備や監視点のデータを参照することが可能になるので、データを直感的に扱えるようになる。また、従来のように、設備や監視点をどのように監視制御するかという詳細手段の視点だけではなく、『どのような空間にしたいか』という抽象度の高い目的指向でデータを利活用することが可能になるので、スマートビルのビジョンとして掲げる空間価値の向上やサービスの導出に繋がることが期待できる。

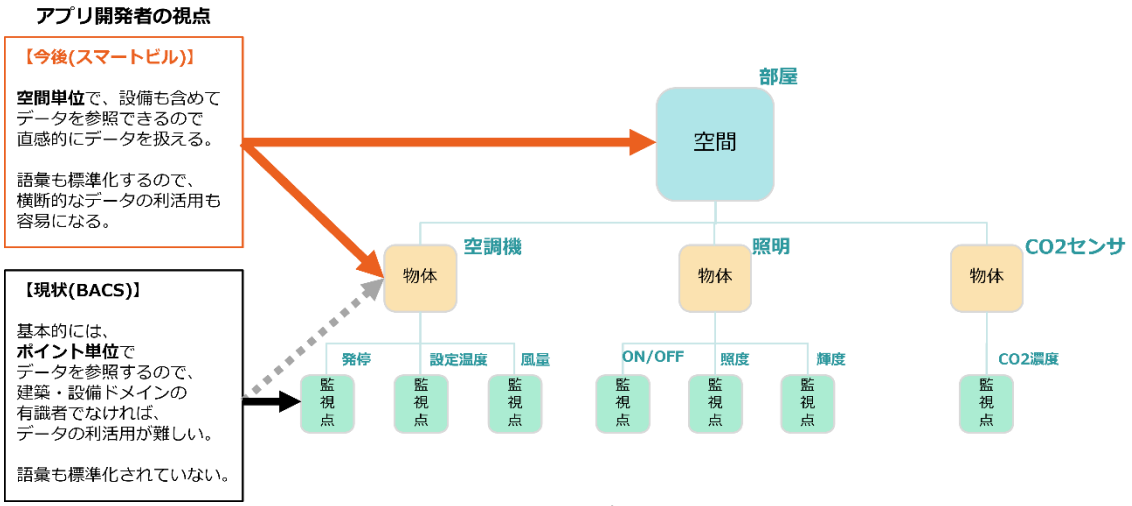

図 11　スマートビルのデータ利活用の視点

3.1.3.　モデリングの基本方針

スマートビルのビジョンやコンセプトを踏まえて特に重要な概念を抽出し、モデルの構成要素として定義する。具体的には、人・設備・ロボットを建物(空間)との関係性を含めて表現するだけでなく、建物は都市の構成要素として機能することを見据えて、建物内外の空間構成も表現可能なモデルにする。

空間をモデリングする際は、BIM（Building Information Modeling)などの建築物に関する既存の情報表現の仕組みを有効活用する。また、データのセマンティクスを適切に表現するために、既存の共通語彙であるオントロジー [5]を活用する。
(ただし、空間を表現するために必ずBIMを用いることを強制するわけではない。)

以上より、スマートビルにおけるモデリングの基本方針を以下のように定めた。

1）　スマートビルのビジョン・ユースケースに則したものであること

2）　建築・設備ドメインにおける既存の標準仕様と親和性が高いこと

（例：BACS/BACnet、BIM/IFC [6]）

3）　建築・設備ドメインで広く参照されているオントロジーを採用すること

（例：Brick [7]、Real Estate Core [8]、BOT [9])

なお、現時点ではスマートビルのビジョンやコンセプトを踏まえたユースケースの全体像を基に検討しているが、今後、ユースケースの詳細検討を行いながら、新たな概念の抽出や概念間の関連付けの詳細化等の検討を進める方針である。

3.1.4.　協調領域として規定する範囲

建物データモデルに関して、協調領域として規定する範囲は基本的に概念モデルの要件までに留める。ただし、論理モデルの要件に関しても、横断的なデータの利活用を実現す

るために標準化が必要と判断したものは、協調領域として規定する方針である。なお、実装の詳細に依存する物理モデルに関しては、協調領域として規定しない方針である。

3.2. 建物データモデルの概念モデル

3.2.1. 概念モデルの構成要素

上述の背景より、スマートビルにおいては、「設備」だけでなく「空間」に関するデータも重要になる。また、スマートビルのビジョンを踏まえると、「人」に関するデータや、将来的には配送ドローンや警備ロボットのような「ロボット」に関するデータも扱えるようにする必要がある。(以下、「設備」「人」「ロボット」等を総称して『物体』と定義する。)

したがって、建物データモデルの概念モデルは、「空間」・「物体」・「監視点」という3つの概念で構成するものと定義する。以下、それぞれの具体的な定義を記載する。

図 12 概念モデルの構成要素

1) 空間

物体を配置することができる3次元の広がりを持った領域を表す。ただし、領域自体や領域の外郭は物理的に存在するものだけが対象ではなく、住所やエリアのように何らかの体系の中で論理的に境界を設定できるものも対象とする。

したがって、任意の複数の部屋・設備をグループ化した領域や、一つの部屋を複数に分轄した領域なども『空間』として扱う。

例：敷地(土地)、建物、フロア(階)、部屋、エリア/ゾーンなど。

2) 物体

物理的な実体が存在し、データや機能を有するものを表す。大分類としては、「設備」・「人」・「ロボット」等が該当する。

例：空調機、照明、エレベータ、電力メータ、温度センサ、配送ロボットなど。

3） 監視点

物体の状態値や、物体が影響を及ぼす空間の状態値、物体を制御するための設定値に対応する動的なパラメータを表す。なお、この概念の導出背景や実装の考え方については6. Appendixを参照すること。(6.2.1.監視点の導出背景)

例：発停、設定温度、照度、現在位置、速度、消費電力、室温、CO2濃度など。

3.2.2. 概念モデルの要件

建物データモデルの概念モデルの要件を記載する。必要要件は、スマートビルの協調領域として規定する遵守すべき要件を意味する。推奨要件は、必須ではないが推奨する要件を意味する。

スマートビルのビジョンやユースケース、現状のビル設備監視の考え方や課題を踏まえて概念モデルの必要要件を導出した。また、多くの市場要望が存在するものの、実現するためには複雑な仕様になる等の課題が想定される事項に関しては、推奨要件として記載する。

なお、それぞれの要件の必要/推奨の区分に関しては、ステークホルダーの有識者と議論して判断する。また、以下に列挙している要件以外にも今後の検討や社会実装のフィードバック等を元にして、柔軟に拡張していく方針である。

表 9 概念モデルの要件

項番	要件	必要/推奨
1	空間・物体・監視点という3つの概念を、 関係性を含めて表現できること。	**必要**
2	空間同士の関係性を表現できること。	**必要**
3	空間と物体の関係性を動的に変更できること。	**必要**
4	空間・物体・監視点を表すオブジェクトは、 プロパティ(データ項目)を持つこと。	**必要**
5	任意の空間・物体・監視点をグループ化して表現できること。	推奨
6	物体同士の関係性を表現できること。	推奨
7	空間・物体・監視点を表すオブジェクトは、 プロパティとして「位置情報」を持つこと。	推奨

以下に、各要件に対する補足説明を記載する。

1） 空間・物体・監視点という3つの概念を、関係性を含めて表現できること。

建物データモデルは、空間・物体・監視点のそれぞれが関係性を持って表現できる必要がある。空間と物体の関係性は、基本的には現実世界における物理的な所有(所属)関係に

相当するが、複数の空間や物体をグループ化する「エリア/ゾーン」も空間の一種として定義しているので、論理的な関係性を表すことも可能である。

また、空間と監視点の関係性や、物体と監視点の関係性については、監視点をパラメータと定義しているので論理的な関係性を表すが、監視点を空間と物体のどちらに紐づけるのか、目的や用途に応じて柔軟に表現することが可能である。

なお、関係性を表現するための具体的なデータ構造や実装手段は特に規定しない。したがって、所有(所属)するものをオブジェクトのプロパティとして明示的に記述する場合や、後述するように、オブジェクトのプロパティとして位置情報を持たせることで、間接的に関係性を表現することも考えられる。

代表例として、ツリー構造で表現した概念モデルを図 13に示す。

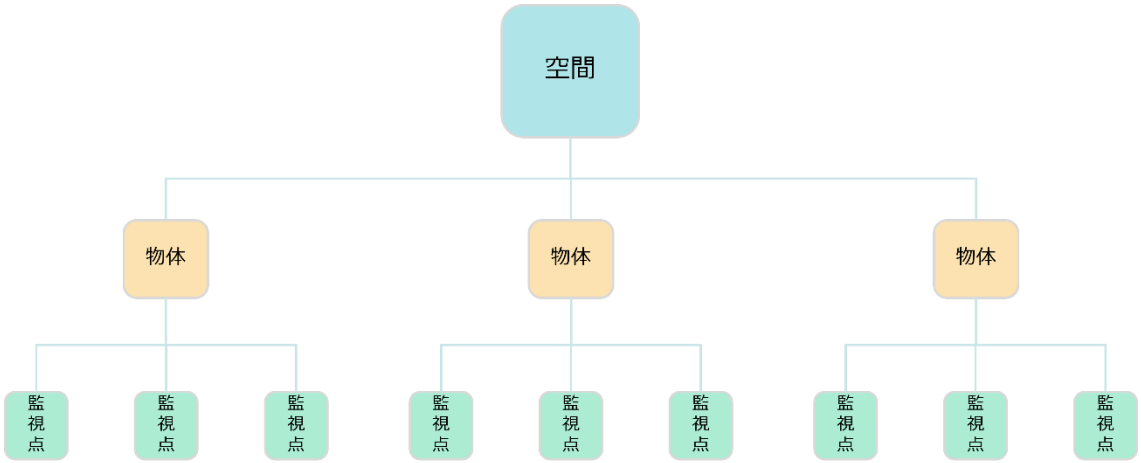

図 13　建物データモデルの概念モデル

2)　空間同士の関係性を表現できること。

建物データモデルは、空間トポロジを表現できる必要がある。例えば、建物には複数のフロアが存在し、それぞれのフロアには複数の部屋が存在するといったことを記述できるようにする。以下は、ある敷地(土地)に存在する建物に関して、建物内のフロアや部屋を含む空間階層を、ツリー構造で表現したものを図示する。

なお、建物内のフロアや部屋などの間取りは、原則、建物の竣工後には変化しないものと想定しているが、運用中に部屋の間仕切りを変更するようなケースもあり得るので、建物の特徴やユースケースに応じて空間トポロジを動的に変更できるようにすることも望ましい。

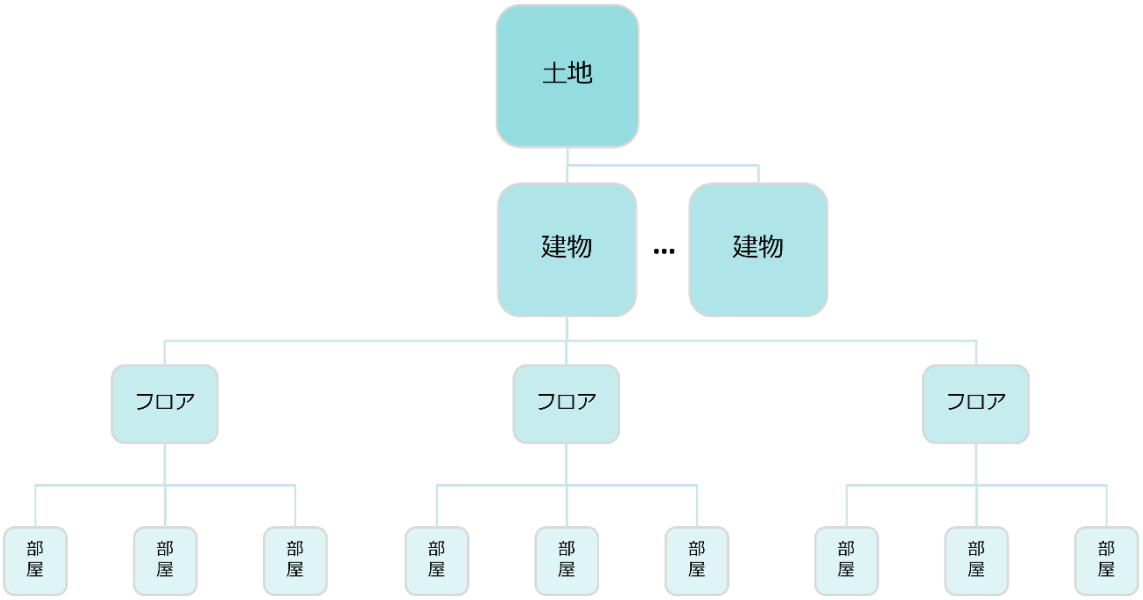

図 14　空間トポロジの具体例

3)　空間と物体の関係性を動的に変更できること。

スマートビルでは、設備だけでなく部屋の中に居る人やロボットのデータも対象であり、人やロボットは建物内の様々なフロア・部屋を移動し得るため、空間と物体(人・ロボット等)の関係性は動的に変更できるようにする必要がある。

空間と物体は互いに独立して存在することができるので、この性質を踏まえて所有関係を実装することが想定されるが、具体的な実装手段は特に規定しない。

ただし、従来通りに設備の監視だけを実現したいというユースケースも十分に存在するので、本要件の適用対象としては、移動する能力を有するものを扱う場合のみに限定する。

4)　空間・物体・監視点を表すオブジェクトは、プロパティ(データ項目)を持つこと。

「空間」に関するプロパティについて、以下に具体例を記載する。

表 10　各オブジェクトの持つプロパティの例

区分	プロパティ例
静的データ	建物の住所・名称、 部屋の種別・名称・面積・体積・最大収容人数 等
動的データ	温度・湿度・CO2濃度・照度・人数・位置情報 等

静的データは、建物の設計・施工段階で決定し、基本的には運用中に変化しない固定的なデータを示す。動的データは、運用中に時々刻々と変化し得るデータを示す。
(位置情報は、人やロボットの属性として定義することを想定して動的データの一例として記述したが、固定的に設置される設備の属性として定義する場合など、静的データの区分になる場合もあり得る。)

5） 任意の空間・物体・監視点をグループ化して表現できること

現状のビル設備監視において、ユーザー(主にビル管理者)の使用性を向上させるために、任意の複数の機器をグルーピングして、一つのグループ(「エリア(ゾーン)」)として監視するユースケースが存在する。

単独のアプリケーションだけで「エリア」を必要とするのであれば、グループ化する仕組みをアプリケーションの機能として独自に実装する方法が考えられる。その場合、「エリア」の構成要素になる「空間」や「物体」に関するデータは、アプリケーションがビルOSから標準APIを介して取得し、アプリケーション側のモデルとして「エリア」と関連付けることが想定される。

ただし、複数のアプリケーション間で、「空間」「物体」「監視点」だけでなく、「エリア」に関する情報も互いに共有したいケースが多々あると想定している。そのため、ビルOSの建物データモデルとして、「空間」「物体」「監視点」だけでなく「エリア」も関連付けて管理できるように設計することが望ましい。この場合は、ビルOSの機能としてグループ化の機能を実装し、アプリケーションに対するインターフェースは拡張APIとして公開することを想定している。各アプリケーションは拡張APIを介して「エリア」の新規作成・取得・更新・削除をビルOSに要求することで、「エリア」に関する情報を互いに共有する。

また、上述のように、ビルOSの建物データモデルとして「エリア」も関連付けて管理する場合は、任意の設備機器が部屋とエリアの両方に所属するといったように、一つの子が複数の親に所属する関係になる。したがって、実装する際のデータ構造としてはツリー構造のような階層型の構造ではなく、グラフ構造などのネットワーク型の構造で実装することを推奨する。

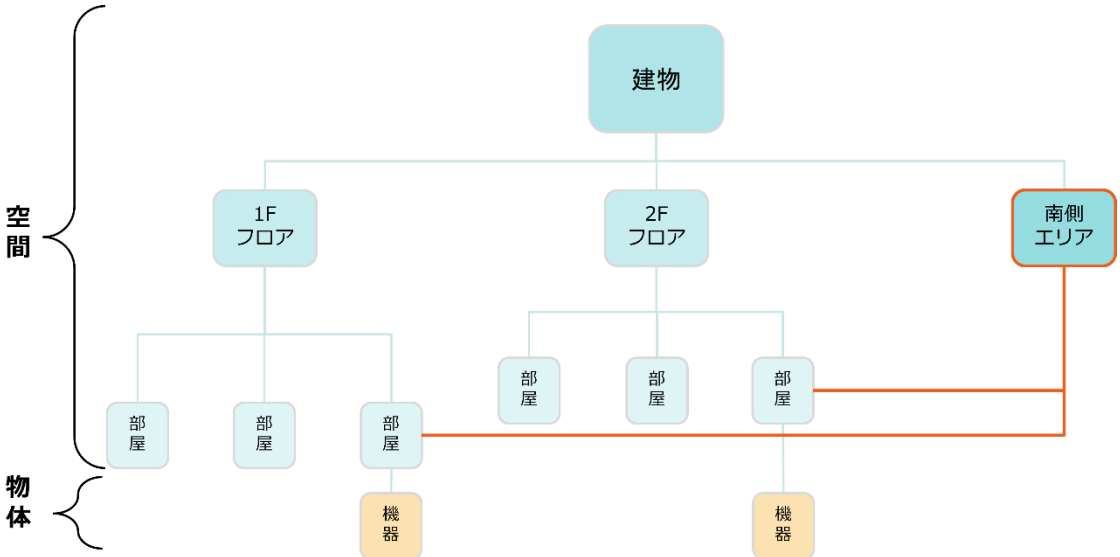

図 15　エリア/ゾーンの具体例1(複数部屋のグループ化)

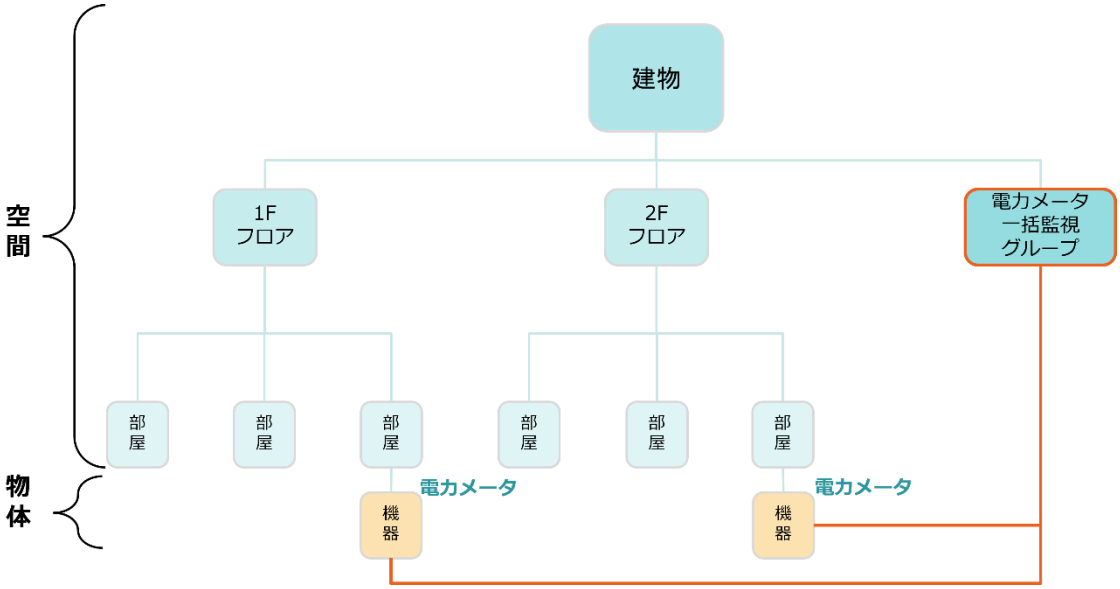

図 16　エリア/ゾーンの具体例2(複数機器のグループ化)

6)　物体同士の関係性を表現できること

例えば、設備機器の故障分析のユースケースにおいて、設備機器を構成する部品単位での横断的なデータ分析を実施したいといったニーズ等が考えられる。実現するためには、設備機器を構成部品に展開して、それぞれの部品(物体)の関係性を建物データモデルとして階層的に表現できるようにすることが有用である。また、設備によっては、互いに関連性のある複数の機器群を一つの系統として管理し、空間に及ぼす影響を系統単位で監視/制御することや、データ分析するようなユースケースも想定される。

7)　オブジェクトにプロパティとして『位置情報』を持たせること

建物データモデルの必要要件として、「空間」・「物体」・「監視点」という3つの概念について関係性を含めて表現することを規定しているが、関係性を表現するための具体的な実現手段までは規定していない。

オブジェクトにプロパティとして位置情報を持たせることは、それらの関係性を表現するための一つの実現手段になると想定している。(ここでの位置情報は、任意の座標系における物体の位置を数値表現したものと仮定する。他にも、場所コードの活用等も想定される。)

具体的には、ある共通の座標系において各オブジェクトの位置情報を参照することで、何がどこに有るかという情報をロジックで算出することができるので、間接的に関係性を表現することが可能になる。特に、配送・警備ロボットのように走行するものを扱うユースケースにおいては、位置情報は走行制御を実現するための重要な情報になる。

ただし、上述のような移動する物体を含まずに、設備だけを状態監視/操作するユースケースにおいては、「空間」と「物体」の関係性を別の手段で表現することも考えられるので、位置情報をオブジェクトのプロパティとして持たせることは必要要件としては規定せずに、推奨要件として記載する。

3.3. 建物データモデルのオントロジー

建物データモデルは「空間」・「物体」・「監視点」という3つの概念から構成されるが、実際にデータモデルを記述する際には建築物やビル設備管理のドメインで代表的な語彙基盤を参照することを想定している。現状では、あらゆるユースケースに対応可能な単一の標準的な語彙基盤は存在しないと捉えており、複数の語彙基盤を組み合わせてデータモデルを記述する想定である。

たとえば、W3C [10]が検討中のBOT(Building Topology Ontology)は、建物の空間構成に関しては記述することが可能だが、ビル設備を表現するための語彙がまだ定義されていないので、他のオントロジーと組み合わせる必要がある。Project Haystack [11]は空間構成の表現力は不足しているが、BACSに関する各種設備機器を表現するための語彙は十分に定義されているので、これらを組み合わせることで空間構成と設備の両方を表現することが可能になる。

特にスマートビルの建物データモデルと親和性が高いと捉えているのは、RealEstateCoreとBrickである。これらは、空間構成だけでなくビル設備機器に関しても豊富な語彙を定義しており、建物データモデルの必要要件を満たすことが可能なオントロジーと考えている。

なお、建物データモデルの表現には複数の語彙基盤を参照することを想定しているが、横断的なデータの利活用を実現するためには、少なくともデータ名称を標準化することが必要になる。ただし、すべてのデータ項目を標準化することは現実的には困難であるので、どの業務範囲における、どのデータ項目を標準化対象とするか、実現手段も含めて今後の検討課題とする。

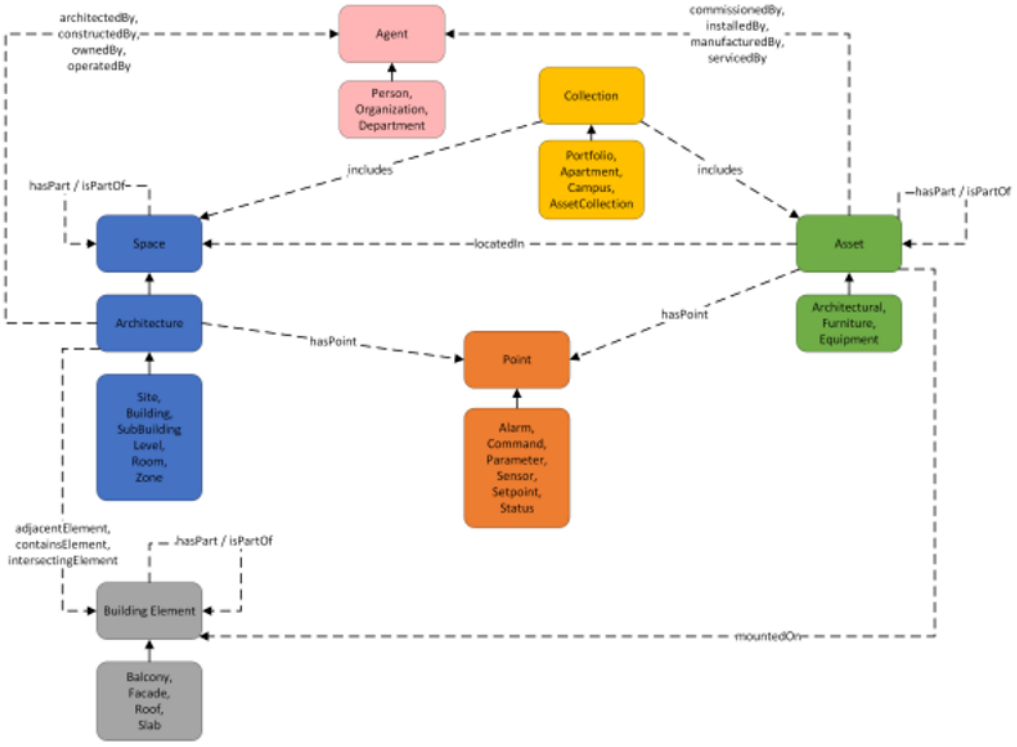

図 17　Real Estate Coreのモデル

出典：「RealEstateCore.io、RealEstateCore［8］」

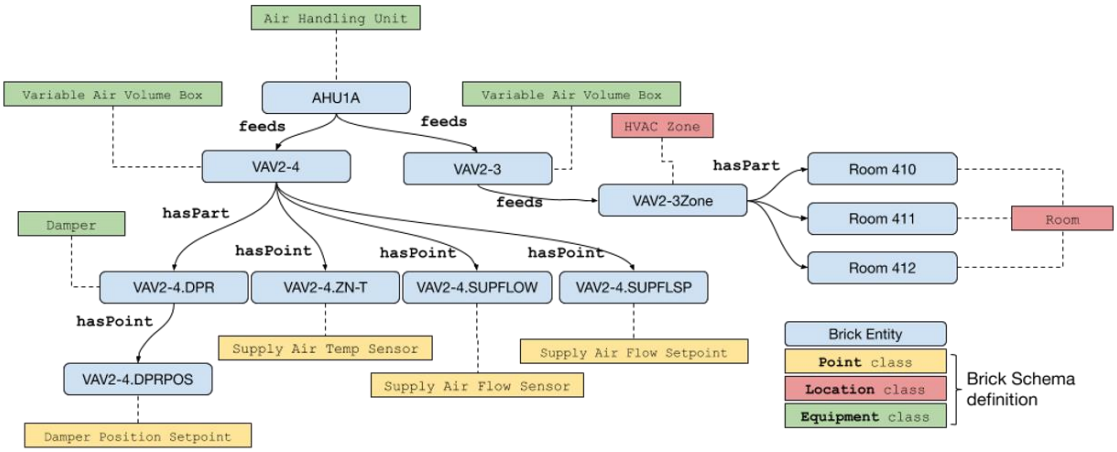

図 18　Brickのモデル

出典：「brickschema.org、Brick [7]」

3.4.　建物データモデルの生成

建物データモデルを生成するためには、設備や空間の情報がビルOSへのインプットとして必要になる。例えば、設備に関してはBACSにおけるポイントリストを活用し、空間に関してはBIM/IFCから抽出した空間・ジオメトリ情報を活用することなどが考えられる。空間と設備の関係性を表現するためには、それぞれのインプット情報を互いに突合させる必要があるが、いずれかの情報にもう一方とリンクする識別子を付与する等の手段によりマッピングを実現することが考えられる。

したがって、建物データモデルを生成するためには、ある程度の手作業は依然として必要になると思われるが、インプット情報を極力マシンリーダブルな形式で表現しておくことで、将来的にはデータモデルを半自動的に生成することは実現可能である。

今後、インプット情報の種類や具体的な表現形式、効率的な突合方法を検討し、作業工程を明確にしていくことで、スマートビルにおける建物データモデルの生成方法に関する標準仕様を策定していく必要がある。

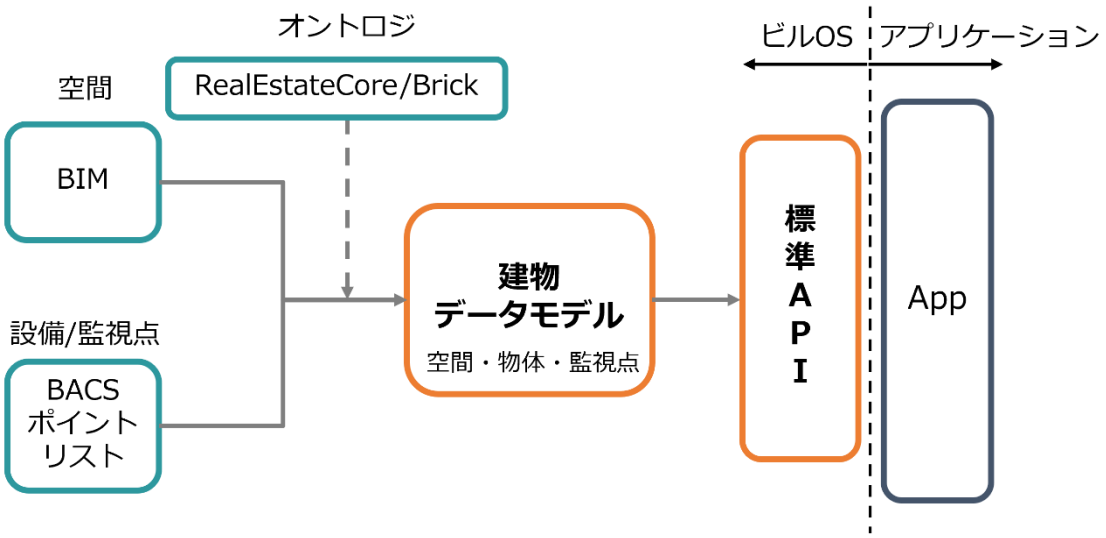

図 19　建物データモデルの生成イメージ

3.5.　識別子

建物や建物内の空間や物体を一意に識別できるようにするための識別子を定義する必要がある。建物に関しては、あらゆる建物の中から特定の建物を一意に指定することができるようにグローバル空間でユニークな識別子を定義する。また、建物内の空間や物体に関しては、建物をグローバル空間で一意に指定できる前提において、建物内のローカル空間でユニークな識別子を定義することで一意に指定できるようにする。

具体的な識別子の体系については、空間や物体などの個々の実体を識別するための識別子だけでなく、物体の種類(例：照明、空調、電気装置等の分類体系や製品識別情報等)や、属性の種類を区別するための識別子(コード)を定義することも検討する。

識別子の参考として、建築ドメインにおける大分類はIFC(ISO16739)、中・小分類は建築分類体系コード（例：Uniclass2015）が規定されており、これらの既存体系を有効活用することを一つの手段として検討中である。これらはISO12006-3に準拠するbuildingSMART Data Dictionary(bSDD)で管理されており、所定のAPIで参照することができる。また、空間や設備のプロパティ(データ項目)に関しても、bSDDにてGUIDで管理している。

4. インターフェース

4.1. 概要

スマートビルは、ビルOSが外部に対してインターフェースを公開することで、アプリケーション層やフィールド層との連携を実現する。特にアプリケーション層に対して公開するインターフェースのうち、協調領域として規定する『標準API』に関しては、以下のように基本方針を定める。

4.2. 基本方針

ビルOSが公開する標準APIの設計思想としては、原則、Web APIのデファクトスタンダードであるRESTの考え方に従うものとする。また、一般的にプロトコルはHTTP、データ形式はJSONとすることが主流であるため、標準APIもその前提で設計する。

ただし、エンドポイントやリクエスト/レスポンスの構造、認証・認可などの具体的な仕様に関しては、協調領域として規定する範囲の検討も含めて今後の検討課題とする。したがって、本章の内容は参考情報として記載する。

4.3. 関連技術

4.3.1. Azure Digital Twins

Azure Digital Twins(以下、ADT)は、現実世界の人・モノ・場所・ビジネスプロセスなどを、モデルに基づいて包括的に表現するサービスとしてのプラットフォームである。ADTでは、型(クラス)としてのモデル（クラス）を定義し、モデルに基づいて実体(インスタンス)としてのデジタルツイン（インスタンス）を生成する。モデルに沿ってツイン間の関係性（リレーション）も表現することができ、環境全体のデジタル表現（ツイングラフ）を作成することができる。

4.3.1.1. システム構成におけるADTの位置付け

ADTは、人・モノ・場所等に関する情報を、デジタルツインとして表現し、管理し、複数のデジタルツインを関連付けたツイングラフの操作をREST APIを介して外部に開示提供することができる。またツイングラフ上の関係をたどって目的の情報を検索することすることができるクエリAPIも提供している。 したがって、ビルOSの建物データモデルと標準APIを実装する一つの手段として、ADTを適用することが考えられる。

4.3.1.2. 機能

1） モデル

ADTのモデルは、JSON-LD ベースの Digital Twin Definition language (DTDL) を用いて定義される。モデルには名前 (Room や TemperatureSensor など)を定義することができ、プロパティ、テレメトリ、リレーションシップ、コンポーネントなどの要素が属性として含まれる。権限のある利用者やアプリケーションは、APIを介してモデルの追加・参照・削除が可能である。なお、 RealEstateCoreでは、DTDLで記載されたADTで利用可能な共通的なモデル群が提供されている。

(1) Property - エンティティの状態を表す
(2) Telemetry - 測定値又はイベントを表す
(3) Relationship - あるデジタルツインと他のデジタルツインの関係性を表す
(4) Component - モデルインターフェースを他のインターフェースの組み合わせとして構築

2） ツイン

ADTは、人・モノ・場所等に関する情報を、上記モデルに基づくデジタルツインとして表現し、API を介して権限のある利用者やアプリケーションに開示することができる。権限のある利用者やアプリケーションは、APIを介してツインの追加・参照・更新・削除が可能である。

3） クエリ

ADTは、各ツイン間の関係をツイングラフとして管理している。権限のある利用者やアプリケーションは、プロパティやリレーションシップをキーにツイングラフを検索し、目的とするツイン、及びそれに含まれるリレーションシップ情報を取得することができる。権限のある利用者やアプリケーションは、SQL に似たクエリ言語で記述されたクエリを特定のエンドポイントに POST することで、クエリの結果を得ることができる。

4） イベント通知

ADTは、ツインに変更があった際、あらかじめ設定された外部サービスへ通知を送ることができる。権限のある利用者やアプリケーションは、APIを介してイベント通知を設定・解除することが可能である。

5） データ履歴

ADTは、ツインやリレーションシップのライフサイクルイベント（作成・削除）があった際、またツインのプロパティが更新された際に、データ履歴機能を用いて、あらかじめ設定された時系列データベースに時系列データを保存することが可能である。時系列データベースにはADT内のツインID・ツインプロパティ・リレーションシップIDとともにデータが保存されるため、ADTでツインを特定し、そのツインIDを元に時系列データベースを検索するといったシームレスな利用ができる。また一連の時系列データを使って、ツインの生成、リレーションシップの作成、ツインプロパティの更新といった時系列のイベントを再現することが可能である。

ADTのAPI仕様について、概要及び代表的な例を以下に記載する。なお最新の詳細な仕様については、後述の公式サイト及びリファレンス ドキュメントを参照すること。

ADTのAPIエンドポイントは、https://digitaltwins-hostname/path の形で表される。なおURIパラメータでAPIのバージョンを指定する必要があるため、実際のURIは例えばhttps://digitaltwins-hostname/path?api-version=2022-05-31 のようになる。

6） HTTPメソッド

上記URIが示す通り、APIエンドポイントにはHTTPSでアクセスする。RESTの原則にのっとり、各HTTPメソッドは以下の意味を持つ。

(1) POST, PUT …… リソースの追加
(2) GET …… リソース（個別、一覧）の取得
(3) PATCH …… リソースの更新
(4) DELETE …… リソースの削除

7） リソース

エンドポイントURIに含まれるpathは、モデル、ツイン、リレーションシップ、イベント通知ルート等、ADTが扱うリソースの識別子である。表 11に代表的なリソースpathとHTTPメソッドとの対応を示す。

表 11　代表的なリソースpathとHTTPメソッド

カテゴリ	path	HTTPメソッド
DigitalTwinModels	models	POST|GET
	models{id}	GET|PATCH|DELETE
DigitalTwins	digitaltwins{id}	POST|GET|PATCH
	digitaltwins{id}/components	GET|PATCH
	digitaltwins{id}/telemetry	POST

	digitaltwins{id}/relationships	GET
	digitaltwins{id}/relationships/{relationshipId}	PUT|GET|PATCH|DELETE
Event Routes	eventroutes	GET
	eventroutes/{id}	PUT|GET|DELETE
Query	query	POST

例として、ADTに登録済みのモデルの中からid (=dtmi:com:example:Sample;1)を指定して定義内容を取得するリクエストと、そのレスポンスを以下に示す。

8） リクエスト例

GET https://digitaltwins-hostname/models/dtmi:com:example:Sample;1?includeModelDefinition=True&api-version=2022-05-31

9） レスポンス（ボディ）例

```
{
 "id": "dtmi:com:example:Sample;1",
 "uploadTime": "2022-02-28T05:30:00.1234567Z",
 "decommissioned": false,
 "model": {
  "@id": "dtmi:com:example:Sample;1",
  "@type": "Interface",
  "contents": [
   {
    "@type": "Property",
    "name": "name",
    "displayName": "Sample instance name",
    "schema": "string"
   },
   {
    "@type": "Property",
    "name": "temperature",
    "displayName": "Sample instance temperature",
    "schema": "integer"
   },
   {
    "@type": "Property",
    "name": "humidity",
    "displayName": "Sample instance humidity",
    "schema": "integer"
   }
```

```
    ],
    "@context": "dtmi:dtdl:context;2"
  }
}
```

4.3.1.3. 参考情報

以下に、ADTやReal Estate Coreに関する情報を記載した公式サイトを記載する。

1） Azure Digital Twins の公式ドキュメント [12]

2） Azure Digital Twins REST API リファレンス ドキュメント [13]

3） RealEstateCore オントロジー [8]

4） RealEstateCore 開発者向けモデリングガイド [14]

4.3.2. Web of Things

Web of Things [15] (WoT) とは、Web標準技術を活用して IoTプラットフォームの相互接続を実現するための仕組みである。WoTの中心的な構成要素として、モノ(デバイス)に関する情報を標準化した形式で表現するThing Descriptionが定義されている。

Thing Description [16]は、プログラミング言語やプロトコルに依存しないメタフレームワークであり、様々なモノをThins Descriptionとして表現することで、異なるプラットフォーム間の連携を容易に実現することが可能になる。

4.3.2.1. システム構成におけるWoTの位置付け

Thing Descriptionは、モノに関するデータを表現するだけでなく、アクセスするためのエントリポイントを定義することができる。そのため、ビルOSの建物データモデルと標準APIを実装する一つの手段としてThing Descriptionを適用することが考えられる。

4.3.2.2. 機能

Thing Descriptionは、Property・Action・Eventという3つの代表的なメタデータを、相互作用のアフォーダンスとして定義しており、利用者がどのようにモノと相互作用できるかを示す。アフォーダンスとは、物の知覚された特性、及び物がどのように利用され得るかを決定する基本的な特性を示す情報である。例えば、ドアの取手のように「ドアを開くことができること」や「どのようにドアを開くか」といったことを示唆する。 以下にそれぞれの概要を記載するが、詳細は4.3.2.4.参考情報 に記載した公式サイトを参照すること。

表 12　Thing Descriptionのアフォーダンス

項番	アフォーダンス	概要
1	Property	モノが持つ属性の現在状態を表現する。 標準ではデータ取得(Read)が可能。 オプションでデータ更新(Write)も可能。

2	Action	モノが持つ機能を表現し、外部から起動(実行)できる。 プリミティブな操作の表現も可能だが、設備やコントローラが備えている一連の動作を抽象化した機能を表現することに適している。 例えば、照明の照度を段階的に変化させるフェードアウト制御などが考えられる。
3	Event	モノの状態変化を通知する。

4.3.2.3. 標準APIの仕様例

先述のインターフェースの基本方針に準拠することを前提にして、以下に標準APIの仕様例を記載する。ただし、標準APIの仕様は検討中であるため、一例を部分的に記述することに留める。また、Thing Descriptionの具体的な実装に関しては、公式サイトを参照して検討すること。

Thing Description自体は、モノにアクセスするための具体的なプロトコルとは独立した形で、モノとの相互作用としてのアフォーダンスを規定するものである。したがって、各種センサ等の具体的なモノにアクセスするために必要なプロトコルやURIについては、forms要素を用いて、モノ側のプロトコルやURIにバインディングする必要がある。以下の標準APIの仕様例は、Thing Descriptionのformsで示されているHTTPプロトコルとURIを用いたアクセス方法を想定したものである。

表 13　標準APIの仕様例

項番	機能	対象プロパティ	HTTP メソッド	エンドポイント
1	建物内の 空間一覧を取得	空間リスト	GET	https://api.example.com/v1/buildings/:buildingID/spaces
2	空間(部屋)の 全プロパティを取得	空間ID、 空間種別、 部屋名、面積 温度、湿度、 物体IDリスト	GET	https://api.example.com /v1/spaces/:spaceID/properties
3	空間(部屋)の 任意プロパティを取得	温度	GET	https://api.example.com /v1/spaces/:spaceID/properties/temperature
4	物体(照明)の 全プロパティを取得	物体ID、 物体種別、 OnOff、照度	GET	https://api.example.com /v1/things/:thingID/properties
5	物体(照明)の 任意プロパティを取得	OnOff	GET	https://api.example.com /v1/things/:thingID/properties/onoff
6	物体(照明)の 任意プロパティを更新 (Propertyで実現)	照度	PUT	https://api.example.com /v1/things/:thingID/properties/illuminance

7	物体(照明)の 任意プロパティを更新 (Actionで実現)	OnOff	POST	https://api.example.com /v1/things/:thingID/actions/off

表 14　標準APIの仕様例（表 13の続き）

項番	機能	リクエストボディ	レスポンスボディ

1	建物内の
	空間一覧を取得
		なし	{
			“buildingID”: 1,
			"spaces": [
			{
			"spaceID": 1,
			"properties": {...},
			"actions": {...},
			"events": {...}
			},
			{
			"spaceID": 7,
			"properties": {...},
			"actions": {...},
			"events": {...}
			}
			]
			}

2	空間(部屋)の
	全プロパティを取得
		なし	{
			"spaceID": 1,
			"spaceType" :"room",
			"spaceName": "conferenceRoomA",
			"area": 25,
			"temperature": 21,
			"humidity": 50,
			"thingIDList":[thingID, thingID]
			}

3	空間(部屋)の
	任意プロパティを取得
		なし	{
			"spaceID": 1,
			"temperature": 21
			}

4	物体(照明)の
	全プロパティを取得
		なし	{
			"thingID": 10,
			"thingType": "lighting",
			"onoff": true,
			"illuminance": 150
			}

5	物体(照明)の
	任意プロパティを取得
		なし	{
			"thingID": 10,
			"onoff": true
			}

6	物体(照明)の
	任意プロパティを更新
	(Propertyで実現)
		{
		"illuminance": 170
		}
			{
			"thingID": 10,
			"illuminance": 170
			}

7	物体(照明)の
	任意プロパティを更新
	(Actionで実現)
		なし	{
			"thingID": 10,
			"onoff": false
			}

4.3.2.4. 参考情報

以下に、WoTやThing Descriptionに関する情報を記載したW3C Web of Things Webサイトや参考文献を記載する。

1） W3C Web of Things [15]

2） W3C勧告(Recommendation)

(1) Web of Things (WoT) Architecture 1.0 [17]
(2) Web of Things (WoT) Thing Description 1.0 [18]

3） W3C勧告候補(Candidate Recommendation)

(1) Web of Things (WoT) Architecture 1.1 [19]
(2) Web of Things (WoT) Thing Description 1.1 [16]

4） W3Cドラフト(Draft)

(1) Web of Things (WoT) WoT Profile [20]
(2) Web of Things (WoT) Discovery [21]

5） WoT-JP CG [22]

6） その他文献

(1) W3C WoT（Web of Things）の標準化 [23]
(2) Web of Things (WoT) Architecture 旧版和訳 [24]
(3) Web of Things (WoT) Thing Description 旧版和訳 [25]

5. 非機能要求

5.1. 概要

非機能要求は、システムに対する要求のうち、業務機能としては表現されない要求であり、可用性、保守性、セキュリティといったシステムの安定的なサービス提供のために必要なものである。

スマートビルを構築するにあたって、そのシステム要求は大きな変化を迎えている。従来のビル設計においては、空調・照明・エレベータ・防災・防犯などの各種設備システムそれぞれが独立したシステムとして構築され、外部ネットワークには接続しない閉域網での運用を前提としていた。しかし近年では、遠隔地からリモートで設備を監視/制御するユースケースや、ビル管理システムの動作環境をクラウドサービスに移行するといったニーズが高まりつつある。今後、ビル管理システムの外部ネットワークへの接続、さらにIoTデバイスをはじめとしたビル設備の多様化に伴うシステムの大規模化、複雑化といった変化が起きる中で、セキュリティ等の非機能を検討することはますます重要になる。

本項ではスマートビルの非機能要求について、参照すべきガイドラインを示すとともに特に検討すべき重要事項をまとめている。

5.2. 参照すべきガイドラインなど

1） 非機能要求グレード2018 [26]

当ガイドラインでは、多様なコンポーネントを連携する「システム基盤」を対象に非機能要求項目と、それぞれの段階的な要求レベルを網羅的に示している。スマートビルは多様なデータ連携を司るビルOSを有するため、システム基盤を持つシステムとして参考となる要素が多い。特にビルOSの各項目における非機能要求水準については、本ガイドラインを参考に関係者間で共通認識を持つことが望ましい。

2） ビルシステムにおける サイバー・フィジカル・セキュリティ対策ガイドライン [27]

当ガイドラインは、ビルシステムにおけるセキュリティ対策の汎用的な基本方針を立案することを目的として発行された。なお、このガイドラインの大前提となる考え方は、経済産業省が策定したサイバー・フィジカル・セキュリティ対策フレームワーク（CPSF）[28]に規定されている。

概略としては、ビルシステムに対して想定される様々な脅威に対して、建物内の場所や機器の種類ごとにケース分類し、それぞれの場所や機器に対して想定されるインシデント・リスク源・対策要件をポリシーレベルで整理している。また、ビル自体やビルシステムのライフサイクルに関して、設計・施工・運用管理などのフェーズごとに取るべき対応策をまとめた内容となっている。

スマートビルにおいて、様々なプラットフォームの多種多様なデータを横断的に利活用するためには、大多数のケースで外部ネットワークへの接続が必要になると想定される。また、スマートビルの設計・施工・運用管理といったライフサイクルに関しても、従来以上に複雑化することを踏まえると、それぞれのフェーズに沿った対応策が必要となる。ス

マートビルのセキュリティ対策における包括的かつ基本的な方針については、本ガイドラインを参考に定めることが望ましい。

5.3. 検討すべき事項

5.3.1. 非機能要求項目

システム基盤における非機能の6大項目としては、可用性、性能・拡張性、運用・保守性、移行性、セキュリティ、システム環境・エコロジーが非機能要求グレードにて挙げられている。

非機能要求大項目	説明	要求の例	実現方法の例
可用性	システムサービスを継続的に利用可能とするための要求	・運用スケジュール(稼働時間・停止予定など) ・障害、災害時における稼働目標	・機器の冗長化やバックアップセンターの設置 ・復旧・回復方法および体制の確立
性能・拡張性	システムの性能、および将来のシステム拡張に関する要求	・業務量および今後の増加見積り ・システム化対象業務の特性(ピーク時、通常時、縮退時など)	・性能目標値を意識したサイジング ・将来へ向けた機器・ネットワークなどのサイズと配置 ＝ キャパシティ・プランニング
運用・保守性	システムの運用と保守のサービスに関する要求	・運用中に求められるシステム稼働レベル ・問題発生時の対応レベル	・監視手段およびバックアップ方式の確立 ・問題発生時の役割分担、体制、訓練、マニュアルの整備
移行性	現行システム資産の移行に関する要求	・新システムへの移行期間および移行方法 ・移行対象資産の種類および移行量	・移行スケジュール立案、移行ツール開発 ・移行体制の確立、移行リハーサルの実施
セキュリティ	情報システムの安全性の確保に関する要求	・利用制限 ・不正アクセスの防止	・アクセス制限、データの秘匿 ・不正の追跡、監視、検知 ・運用員などへの情報セキュリティ教育
システム環境・エコロジー	システムの設置環境やエコロジーに関する要求。	・耐震/免震、重量/空間、温度/湿度、騒音など、システム環境に関する事項 ・CO_2 排出量や消費エネルギーなど、エコロジーに関する事項	・規格や電気設備に合った機器の選別 ・環境負荷を低減させる構成

出典：非機能要求グレード　01_利用ガイド解説編

5.3.2. 非機能要求検討プロセス

非機能要求グレードの02_利用ガイド利用編では、検討の段階的な詳細化のプロセスとして、モデルシステムの選定、重要項目のレベル決定、重要項目以外のレベル決定が例として示されている。システムの目的に応じ、合意を取るべき関係者の範囲を適切に設定したうえで、非機能の各項目について検討を進める必要がある。

6. Appendix

6.1. ビルシステムの代表的な通信規格

フィールド層内外システムの連携には多くの方法がある。一番単純なものだと接点連動、シリアル通信（RS-485) [29]を用いたものだと、ModbusRTU [30]や、 BACnetにマスタースレーブ方式とトークンパッシング方式を組み合わせて適用したBACnetMS/TPなどがある。他にも照明に特有の通信としてはDALI [31]、欧州や中国などでよく使われているホームオートメーション、ビルオートメーションの規格であるKNXなども国内で利用されるようになってきた。なお、統合ネットワークなどを介して接続する場合は、IP [32]化された通信方式を利用する。なお、国内で普及が進んでいるのがBACnet/IPであり、KNX/IPやMQTT [33]などのIoT向けの通信が利用されることもあるが、設備連動を行う場合、BACnet/IPに変換して共通的な通信バスを構成することが一般的である。他にもインターネットのデータベースセントリックなアーキテクチャを踏襲したIEEE1888/ISO IEC18880 [34]、更にはデマンドレスポンスの専用プロトコルとして、OpenADR [35]などがある。

6.2. 建物データモデルの補足事項

6.2.1. 監視点の導出背景

スマートビルの建物データモデルにおいて、設備・人・ロボットなどの「物体」に関する属性のうち、動的なパラメータに関しては、「物体」から切り離して「監視点」という別の概念として定義する。

1） 関連付けの柔軟性

このように独立した概念として分離する主な理由は、目的や用途に応じて、柔軟な関連付けを可能にするためである。つまり、監視点として切り出したパラメータは、空間に関連づけるのか、若しくは物体に関連付けるのか、ユースケースに応じて適切なモデリングを実現することができる。たとえば、「物体」のパラメータを「物体」自体に関連付けるのではなく、「空間」に関連付けることも可能にする。また、温度や湿度などの物理量をセンサで計測する際に、計測主体の「物体」ではなく、計測対象の「物体」や「空間」に関連付けることや、そのセンサを装着する「物体」に関連付けることも可能にする。以下にその一例を挙げる。

(1) 温度センサの計測値
『温度センサ』が計測した[温度]データは、温度センサ自体の性質を表すものではなく、計測対象である空間の性質を表している。

(2) 電力メータの計測値
『電力メータ』が計測する消費電力は、電力メータ自身が消費した電力ではなく、空調機や照明などの他の設備機器が消費した電力を表している。

さらにユースケースに依っては、同じパラメータを「設備」と「空間」の両方に関連付けて管理する場合もあり得ると想定している。

2)　従来のモデルとの親和性

BACSの代表的なプロトコルであるBACnetにおいても、「設備」に関する[発停]や[温度]などのパラメータを、設備とは独立したオブジェクトとして表現している。スマートビルのドメインにおいてもBACSの考え方は依然として重要であるので、それと親和性の高いモデルにするためには、動的なパラメータは「設備」から切り離して別の概念として定義しておくことが適切である。

6.2.2.　　監視点の実装の考え方

建物データモデルの概念としては「設備」と「監視点」を別々に分けて定義するが、実装する際は、必ずしも別々に分けて定義することは強制しない。

したがって、例えば照明の[ON/OFF]という監視点を、照明とは独立したクラスのオブジェクトとして定義するのではなく、照明オブジェクトのプロパティとしてプリミティブ型で記述することも可能である。

ただし、監視点をモデリングする際も、現在値だけでなく最小値や最大値、刻み幅、単位などのメタデータを表現することが重要になるケースが多々あると想定されるので、プリミティブ型で暗黙的に表現にするのではなく、クラスなどのユーザー定義型で直接的に表現することが望ましい。

7. リファレンス

[1] W3C, SEMANTIC WEB, https://www.w3.org/standards/semanticweb/.

[2] BACnet Committee, BACnet, https://bacnet.org/.

[3] Bluetooth SIG, Bluetooth 技術概要, https://www.bluetooth.com/ja-jp/learn-about-bluetooth/tech-overview/.

[4] 3GPP Groups, LTE, https://www.3gpp.org/specifications-technologies/releases/release-8.

[5] W3C, Semantic Web Ontologies, https://www.w3.org/standards/semanticweb/ontology.

[6] ISO, ISO 16739-1:2018, https://www.iso.org/standard/70303.html.

[7] Brick Consortium, Brick, https://brickschema.org/.

[8] RealEstateCore, RealEstateCore, https://www.realestatecore.io/.

[9] W3C, Building Topology Ontology, https://w3c-lbd-cg.github.io/bot/.

[10] The World Wide Web Consortium (W3C), W3C, https://www.w3.org/.

[11] Project Haystack, Haystack, https://project-haystack.org/.

[12] Microsoft, Azure Digital Twins のドキュメント, https://learn.microsoft.com/ja-jp/azure/digital-twins/.

[13] Microsoft, Azure Digital Twins REST API リファレンス, https://learn.microsoft.com/ja-jp/rest/api/azure-digitaltwins/.

[14] RealEstateCore, REC data modelling guides, https://dev.realestatecore.io/docs/guides/.

[15] W3C, Web of Things, https://www.w3.org/WoT/.

[16] W3C, Web of Things (WoT) Thing Description 1.1, https://www.w3.org/TR/2023/CR-wot-thing-description11-20230119/.

[17] W3C, Web of Things (WoT) Architecture 1.0, https://www.w3.org/TR/2020/REC-wot-architecture-20200409/.

[18] W3C, Web of Things (WoT) Thing Description 1.0, https://www.w3.org/TR/2020/REC-wot-thing-description-20200409/.

[19] W3C, Web of Things (WoT) Architecture 1.1, https://www.w3.org/TR/2023/CR-wot-architecture11-20230119/.

[20] W3C, Web of Things (WoT) Profile, https://www.w3.org/TR/wot-profile/.

[21] W3C, Web of Things (WoT) Discovery, https://www.w3.org/TR/wot-discovery/.

[22] W. o. T. J. CG, Web of Things JP CG, https://wot-jp-cg.netlify.app/.

[23] 芦村和幸, W3C WoT（Web of Things）の標準化, https://www.w3.org/2018/Talks/1130-ieice-ka/473-477_ieice-may_24.pdf.

[24] W3C, Web of Things (WoT) アーキテクチャ 旧版和訳, https://wot-jp-community.github.io/wot-architecture/.

[25] W3C, Web of Things (WoT) Thing Description 旧版和訳, https://wot-jp-community.github.io/wot-thing-description/.

[26] 情報処理推進機構, 非機能要求グレード 2018, https://www.ipa.go.jp/sec/softwareengineering/std/ent03-b.html.

[27] 経済産業省, ビルシステムにおけるサイバー・フィジカル・セキュリティ対策ガイドライン 第1版, https://www.meti.go.jp/press/2019/06/20190617005/20190617005.html, 2019年6月17日.

[28] 経済産業省, サイバー・フィジカル・セキュリティ対策フレームワーク (CPSF), https://www.meti.go.jp/press/2019/04/20190418002/20190418002.html, 2019年4月18日.

[29] Telecommunications Industry Association, EIA/TIA-485, https://global.ihs.com/doc_detail.cfm?&csf=TIA&item_s_key=00032964&item_key_date=870024&inp.

[30] Modbus Organization, Modbus, https://www.modbus.org/.

[31] DALI Alliance, IEC 62386 standard, https://www.dali-alliance.org/dali/standards.html.

[32] IETF, INTERNET PROTOCOL, https://www.rfc-editor.org/rfc/rfc791.

[33] OASIS, Message Queuing Telemetry Transport (MQTT), https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=mqtt.

[34] ISO, ISO/IEC/IEEE 18880:2015, https://www.iso.org/standard/67485.html.

[35] OpenADR Alliance, OpenADR, https://www.openadr.org/.

謝辞

スマートビルガイドラインの作成にあたり、独立行政法人情報処理推進機構（IPA）デジタルアーキテクチャ・デザインセンターに設置した「スマートビル将来ビジョン検討会」「標準化SG」「フィールドガイドライン検討分科会」「クラウドガイドライン検討分科会」「MSI検討分科会」へご参加いただいた方々、及び情報提供や執筆にご協力いただいた方々へ厚くお礼を申し上げます。

執筆者（五十音順）

独立行政法人情報処理推進機構	
デジタルアーキテクチャ・デザインセンター	青野 敏紀
	岡田 拓郎
	岡田 良平
	粕谷 貴司
	清國 敦史
	後藤 喬行
	中村 公洋
	那須 隆博
	野沢 直弘
	原田 晋吾
	吉田 壮志

レビュアーリスト （五十音順・敬称略）

SBテクノロジー株式会社	
エヌ・ティ・ティ・コミュニケーションズ株式会社	一宮 昇平
	中芝 孝秀
株式会社大林組	三浦 啓祐
	山口 直之
	山本 晃
シスコシステムズ合同会社	
清水建設株式会社	越地 信行
	斉藤 浩
ソフトバンク株式会社	後藤 大輝
ダイキン工業株式会社	小倉 孝訓
	北村 拓也
	桑山 忠弘
大成建設株式会社	大野 元嗣
株式会社竹中工務店	矢野 雅
株式会社東芝	佐古田 健志
東芝インフラシステムズ株式会社	会津 宏幸
日本電気株式会社	望月 康則
日本マイクロソフト株式会社	内田 訓雄
	中西 進
パナソニック株式会社	鈴木 勇至
	平松 勝彦
	吉村 祐一
株式会社日立製作所	
buildingSMART Japan	足達 嘉信
三菱電機株式会社	安達 佳明
	大瀧 尚厳
三菱電機ビルソリューションズ株式会社	朝比奈 努
森ビル株式会社	有山 清隆
	佐藤 芳紀

オブザーバーリスト（五十音順・敬称略）

大阪公立大学大学院 教授	阿多 信吾

慶応義塾大学大学院 教授	白坂 成功
芝浦工業大学 教授	志手 一哉
多摩大学大学院 客員教授	市川 芳明
東京工業大学 教授	松浦 知史
NPO法人ロンマークジャパン 理事長	富田 俊郎