スマートビルのためのオントロジを用いたアクセス制御フレームワークの提案

瀧崎 尚[1] 木戸 善之[1,2,4] 増田 欣之[3] 都島 良久[3] 山本 松樹[2] 下條 真司[1,2]

概要：
IoT や AI の活用により快適性・利便性の向上を図るビル（スマートビル）の開発が進んでいる．不特定多数の人が出入りするビルでは，セキュリティ上ビル内の IoT システムに対するアクセス管理が重要である．現状ビル内のアクセス制御システムはそれぞれのビルで独自に開発されるが，IoT 機器の膨大な数や設置場所の変更の多さから，システム開発の負担が大きくなる．そこで本研究ではスマートビルを対象としたアクセス制御を備えた IoT システムを自動で構築する機能を持つ，**アクセス制御フレームワーク**を提案する．提案フレームワークでは Web オントロジとして Brick を，アクセス制御モデルとして RBAC を採用する．Brick により作成したビルモデルから，IoT システムと RBAC システムの一部を自動で構築するためのプログラムを開発した．評価では，提案フレームワークを大阪大学箕面キャンパスに実際に適用しその有効性を検証した．提案フレームワークによりシステムを自動で構築できることを確認し，ビルごとの実装を極力抑えることができることを確認した．

A Proposal of Access Control Framework for Smartbuildings Using Web Ontology

NAO TAKIZAKI[1] YOSHIYUKI KIDO[1,2,4] YOSHIYUKI MASUDA[3] YOSHIHISA TOSHIMA[3]
MATSUKI YAMAMOTO[2] SHINJI SHIMOJO[1,2]

1. はじめに

Internet of Things（IoT），クラウド，AI の普及に伴い建設業界のデジタル化が進んでおり，スマートビルと呼ばれるさまざまな機能を備えたビルの開発が増えている．スマートビルとは「IoT や AI などの情報技術を利活用し，快適性・利便性の向上や，高度な省エネを実現するビル」とされている [17]．ビル内には照明や空調機を制御するシステムから，温湿度センサや CO2 センサなどの環境センサのデータを取得するシステムなど，スマートビルを管理，制御する IoT システムが多種存在する．

これらの IoT システムをサイバー空間から手軽に操作することで，快適性・利便性を向上させられる．しかしこのような IoT システムは室内の空調管理や個人情報よりアクセス制御 [24] をすることから，第三者により不正な操作がなされた場合，個人情報の漏洩につながる恐れがある．不特定多数の人が出入りするビルにおいては，そのセキュリティ，特にアクセス権限については適切に管理する必要がある．またスマートビルに訪れるユーザに対して与えるべきアクセス権限が異なり，ユーザが居住者であるかゲストであるか，ゲストでも営業による訪問，納品業者，設備（エレベータ等）管理会社の役職によって権限は異なる．よってユーザ情報の属性に応じてアクセス権限を動的に変更する必要がある．

スマートビルにおけるアクセス制御システムは各々のビルで開発されることが多い．開発者はビル内のセンサデバイスなどの機器がまとめられたリストを参照し，IoT システムやアクセス制御システムを設計，開発することになる．しかしスマートビルにおいては，機器の数が膨大であることや，機器の設置位置が頻繁に変化することから，システムを開発する負担が非常に大きいという課題がある．この課

1 大阪大学大学院情報科学研究科
2 大阪大学サイバーメディアセンター
3 ダイキン工業株式会社
4 岡山理科大学情報理工学部

題がスマートビルの普及と新規参入を防ぐ障壁となっているため，開発者の負担を減らすためにシステム構築のある程度の自動化が求められている．そこで本研究ではスマートビルにおけるアクセス制御にフォーカスを絞り，オントロジを用いた**アクセス制御フレームワーク**を提案する．

本稿では，2 章で関連研究について述べ，3 章で提案手法について述べる．4 章にて評価を行い，5 章で結論を述べる．

2. 関連研究

2.1 Web of Things

一般的にビル内の IoT システムはマルチベンダシステムになっており，IoT を制御するプロトコルを標準化するために，W3C によって Web of Things（WoT） [4] が提唱されている．

WoT は，HTTP や Javascript を用いて IoT システムを開発することで，アプリケーション層を Web 技術で統一し，システムの相互連携を図るアプローチである．WoT ではセンサや機器など実空間上のオブジェクトを「Thing」として扱い，Thing の振る舞いを表す Thing Description（TD）が記述される．

WoT システムの構成を図 1 に示す [19]．各機器は WebAPI を持ち，対応する Thing が存在する．Thing は技術的には Web サーバであり，機器にアクセスするためのエンドポイントを提供し，TD を出力するという役割を持つ．Thing は開発者が手動で生成しなければならず，スマートビルのような多数の機器がある場合にはその構築負担が大きくなる．また各 Thing を集約する役割として WoT Gateway がある．ユーザは WoT Gateway を経由して各 Thing にアクセスし，機器を操作あるいは，データを取得することができる．

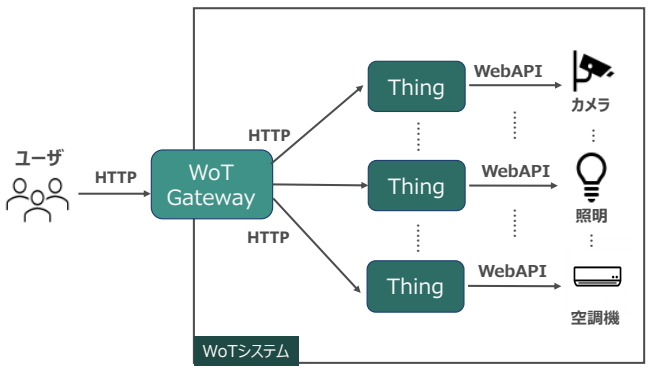

図 1 WoT システムの構成

2.2 スマートビルでのアクセス制御

スマートビルにおけるアクセス制御に関する研究はいくつか存在する．

Bandara[14] らは，スマートビルにおける API 対応のデバイスのためのアクセス制御フレームワークを提案している．機器が取り外されたり移動したりするなどのダイナミックなスマートビル環境においては，既存のアクセス制御の仕組みを適用することは困難であると述べている．提案フレームワークでは，ダイナミックな環境に対応するために，セキュリティマネージャを IoT システムのサードパーティとして設計し，セキュリティマネージャによってアクセス制御を行なっている．しかしこの研究では，ユーザ，機器の数の多さについては言及されておらず，本研究の目的を達成することはできない．

Bindra[15][16] らは，大規模なスマートビルを対象に分散型のアクセス制御システムを提案している．提案手法では，Brick を用いたビルモデルをもとに，各部屋，各経路のセキュリティコストを算出し，アクセス可否の判断に利用している．また，ブロックチェーン技術のスマートコントラクトを利用してビルユーザのアクセス権限を，分散型で柔軟に管理している．しかしこの研究では，ユーザは静的なものであるとし，ユーザの移動などの動的な部分は想定外とされている．またユーザ，機器の数の多さについても言及されておらず，開発や管理負担についても考えられていない．

Sasaki[23] らは，スマートビルにおけるアクセス制御では，ユーザの位置に基づいたアクセス制御が必要であると述べ，ユーザの位置を詐称されないアクセス制御の仕組みを提案している．提案手法では，各部屋に設置されたビーコンからユーザに対して，秘密鍵で署名されたトークンを送信する．ユーザはトークンを認証サーバに送信し，認証サーバは公開鍵によってトークンを検証する．これによりユーザの位置情報の詐称を防いでいる．しかし，この研究では具体的な実装手法が説明されていないだけでなく，スマートビル特有のユーザ，機器の数の多さには触れられていない．そのため本研究とは異なるものである．

3. 提案手法

本研究では標準化に従い WoT を用いる．WoT では各機器を Thing として扱い，開発者は手動で Thing を生成する．1 つの Thing を生成するために約 100 から 500 行のソースコードが必要である [11]．表 1 に，スマートビルに設置された制御対象となる機器数を記す．スマートビルでの制御対象の多くは空調機器，人感センサ，照明器具，施錠機器である．WoT を適用するスマートビル環境では，IoT 機器の膨大な数の機器分の Thing の実装が必要となり，設置機器数に比例して工数も増大することとなる．

スマートビルに限らず通常のテナント，オフィスビルには約数百人もの多数のユーザが出入りする．例えば大学キャンパスのビルであれば管理者・教員・学生・来訪者などが考えられる．多数かつ多様なユーザに対して与えるべきアクセス権限が異なり，柔軟なアクセス制御が求められ

表 1　スマートビルの例

名前	所在地	機器数
Intel SRR3[6]	バンガロール（インド）	約 9000
Cisco RBC WaterPark Place[10]	トロント（インド）	約 2000
ダイキン工業 研究開発センター [12]	大阪（日本）	約 400
大阪大学 箕面キャンパス	大阪（日本）	約 300

ている．

例えばユーザが管理者であれば，ユーザの位置によらず常に機器にアクセスし管理できる必要がある．一方でユーザが来訪者の場合，ユーザがビル内のある共用部屋に存在すれば利便性向上のためにその部屋の機器へのアクセスを許可してもよいが，部屋外・ビル外にいればセキュリティ上そのアクセスは拒否した方がよい．このようなアクセス制御システムを構築する場合，多数のユーザを管理し，多数の機器とユーザの位置関係まで把握する必要があるため，その構築負担が大きくなる．

そこで本研究では，センサなどの振る舞いのインターフェースとして WoT を用いる環境下でのアクセス制御フレームワークを提案する．要件として以下に列挙する．

- ビル情報管理には汎用オントロジを用いる．オントロジとは概念や言葉の意味を定義するスキーマであり，オントロジによってビル構造などを統一的な方法で表現することができる．代表的なビルオントロジとしては，Brick Schema[3]，Project Haystack[9]，IFC[5] などがあげられる．本研究では，他と比べ豊富な語彙，建物と機器の関係を表す表現を持つという理由から，オントロジとして Brick Schema を採用し，ビルモデルを作成する [13].
- ビルモデルからシステムを自動的に構築するプログラムを開発する．これによりどのビルにも適用可能で，開発者の負担を軽減するフレームワークの実現を目指す．
- アクセス権限を管理する負担を軽減するために，アクセス制御モデルとして Role Based Access Control（RBAC）を採用する．RBAC とは抽象的なグループ（ロール）を作り，それをユーザに割り当てることでアクセス制御を実現する手法である [18].

3.1　提案フレームワーク

アクセス制御フレームワークの全体図を図 2 に示す．アクセス制御フレームワークではまず，(1) converter によりビル内に設置された機器がまとめられたデバイスリストからビルモデルを作成する．その後，(2) WoT Manager によってビルモデルから WoT システムを自動で構築する．また (3) RBAC Manager によりビルモデルから RBAC システムの一部を自動的に構築する．以降に上記の 3 つの動作について詳しく説明する．

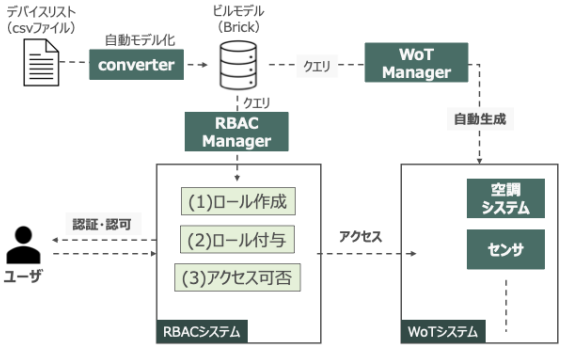

図 2　提案フレームワークの全体図

3.1.1　ビルモデルの作成

本研究ではビルモデルの作成のために Web オントロジである Brick を利用する．Brick とはビルのためのオントロジであり，空間やデバイス，センサなどの語彙の定義や，それらの関係性を記述するための語彙が用意されている．またデータ形式として RDF（Resource Description Format）が利用されており，モノとモノの関係は主語・述語・目的語で表される．例えば「センサ A は部屋 A に設置されている」という関係を Brick によって記述すると，「*sensor-A brick:hasLocation room-A*」となる．

このように Brick を用いることで，エクセル表などでは表現できない「モノとモノの関係」を表現することができ，データにセマンティクスを与えることができる．またあらかじめ定義された語彙を用いるため，それぞれのビル構造やリスト記述の差を吸収することができる．つまり Brick をもとにシステムを構築することで，システムの再利用性を向上させることができる．

上記のようなモデリングを，開発者が表 2 のようなデバイスリストを参照して行う必要がある．しかしモデリングのためにはオントロジに精通している必要があるほか，大型ビルの場合には機器の数が多くモデリングの負荷は非常に大きくなる．そこで本研究ではビルモデル作成における開発者の負担を軽減するために，自動モデル化の機能を持つ converter を開発した．converter は csv で記述されたデバイスリストを読み込み，Brick の語彙を用いてビルモデルを作成する．実際に converter により作成したビルモデルの一部を可視化した図を示す（図 3）．

	id	name	type	maker_name	location	…
1	EEE-0001	AIR-01	空調機	ダイキン	会議室 01	…
2	EEE-0002	AIR-02	空調機	ダイキン	会議室 02	…
3	EEE-0003	TEMP-01	温度センサ	…	食堂	…
⋮	⋮	⋮	⋮	⋮	⋮	⋮
1000	EEE-1000	CO-01	CO2 センサ	…	食堂	…

表 2　デバイスリストの例

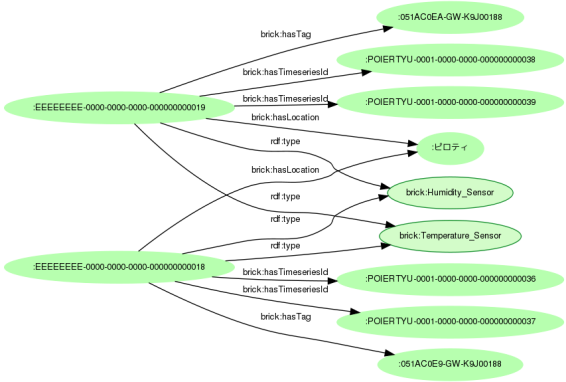

図 3　Brick を利用したビルモデル（一部抜粋）

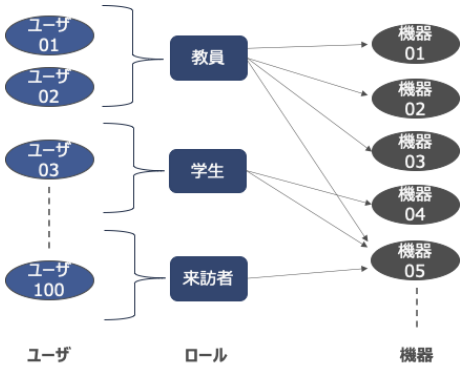

図 4　従来の RBAC における関係図

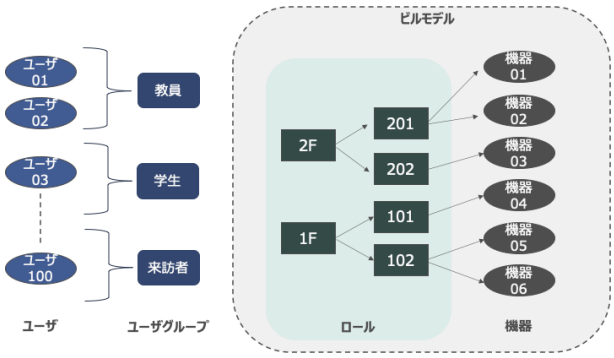

図 5　提案の RBAC における関係図

3.1.2　WoT システムの構築

提案フレームワークでは，Brick を用いたビルモデルから WoT システムを自動で構築する（WoT Manager）．具体的に WoT Manager は，大規模な WoT システムを構築する上で構築負荷が大きいとされる Thing の生成を自動で行う．以降に WoT Manager の動作について説明する．

WoT Manager はまずビルモデルに SPARQL クエリ [22] を実行することで，Thing の生成に必要な情報を取得し，Json ファイルを生成する．Json ファイルにはそれぞれの機器の名前，種類，設置場所などの詳細な機器情報だけでなく，Thing を公開するエンドポイントなどのサーバ情報も含まれる．Json ファイルの生成後，WoT Manager はそれぞれの機器の種類を読み込み，対応するテンプレートのプログラムファイルを選択する．その後テンプレートのプログラムファイルが Json ファイルに記述された情報を読み込み，Thing を起動，生成する．テンプレートのプログラムファイルとは機器ごとに用意される，Thing の生成に必要なプログラムファイルのことを指す．テンプレートのプログラムファイルの形式は，機器や Web API の種類によって異なる．

上記の手順で WoT Manager は Thing を自動で生成する．このようにビルモデルを利用することで，ビルの環境によらず統一的な方法で WoT システムを自動で構築することができる．

3.1.3　RBAC システムの構築

本研究では，多数のユーザと機器の管理負担を軽減するためにアクセス制御モデルとして RBAC を採用する．RBAC ではあらかじめロールを作成し，それをユーザに割り当てることでアクセス制御を実現する．以降に本研究の RBAC システムにおけるロールの設計方針と RBAC Manager の動作について説明する．

図 4 に従来の RBAC システムにおけるユーザ，ロールなどの関係を示す．従来の RBAC では多数のユーザを管理するために，管理者や教員などのユーザの属性をロールとし，抽象的なグループ単位でアクセス制御を行っている．しかし機器の数が多いスマートビル環境では，どのロールにどの機器の権限を割り当てるかを管理するのが大きな負担になる．

そこで提案フレームワークではユーザの属性をユーザグループとして捉え，部屋・階層などの建物構造を利用したロールを作成する（図 5）．またこのロールは包含関係を表現可能であり，例えば図 5 において，ロール *2F* はロール *201*，*202*（機器 01〜機器 03 の権限）を含む．このように建物構造を利用したロールを作成することで，一つ一つの機器を管理するのではなくグループ単位での管理が可能になり，管理負担を軽減することができる．開発者は，ユーザグループとロールの関係（ロール割り当てポリシ）を記述するだけで，ユーザにロールを割り当てることができる．

RBAC Manager はビルモデルに対し SPARQL クエリを実行し，部屋・階層などのビル構造に関する情報を取得し，構造名のロールを作成する．その後，開発者が事前に定義したロール割り当てポリシを読み込み，ユーザグループに対しロールを割り当てる．これにより，該当ユーザグループに属するユーザに権限が割り当てられることになる．例えば図 5 において，ユーザグループ「教員」に対しロール *201* を付与すれば，ユーザ 01 はロール *201* を持ち，機器 01 と機器 02 へのアクセス権限を持つことになる．

上記の手順で RBAC システムにおけるロール作成・付与を自動的に行う．一方でロール付与後のアクセス可否を判断する機能やユーザの位置を特定する仕組みについては，

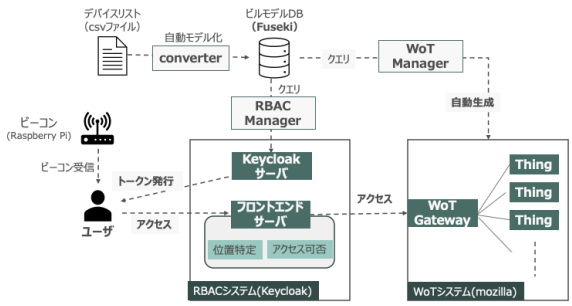

図 6 プロトタイプシステム

開発者自身で実装する必要がある．

RBAC Manager は WoT Manager と同様にビルモデルを利用することで，ビルの環境によらず統一的な方法で RBAC システムの一部を自動で構築することができ，開発者の負担は小さくなる．またロールの設計段階で，ユーザのグループ化だけでなく機器をグループ化することにより，機器の数が多いスマートビル環境において，機器の管理負担を軽減することができる．

3.2 実装

本研究では提案フレームワークを大阪大学箕面キャンパスに適用し，アクセス制御機能を備えた WoT システムのプロトタイプを実装した．実装したプロトタイプシステムの構成を図 6 に示す．ユーザはアクセス制御を備えた RBAC システムを介して WoT システムにアクセスする．RBAC システムと WoT システムはクラウド上の計算機（Google Compute Engine：AMD EPYC 7B12，4GB メモリ）に構築し，ユーザが用いる機器としてノート PC（Macbook Pro：Intel Core i5，8GB メモリ）を使用した．

ビルモデルを格納するデータベースには，RDF データベースとして広く普及している Apache Jena Fuseki[1] を使用し，WoT システムの構築には GUI など豊富な機能を備えているという理由から WebThings Gateway(Mozilla IoT)[8] を使用した．また RBAC システムの構築には，オープンソースのソフトウェアで Docker 環境にも対応しているという点で，導入が容易である Keycloak[7] を利用した．以降にユーザの位置を特定する仕組みと，RBAC システムの具体的な動作に分けて説明する．

3.2.1 ユーザの位置を特定する仕組み

プロトタイプ実装ではユーザの位置を特定するために BLE（Bluetooth Low Energy）ビーコンを用いた．BLE ビーコンとは近距離無線通信を利用して，UUID など個体を識別するための情報をアドバタイズする発信機のことを指す（以降ビーコンと記す）．本研究では Raspberry Pi 4B をビーコンとして使用し，ビーコンの通信プロトコルとして広く利用されている iBeacon[21] を使用した．発信する電波の強度は手動で設定可能であり，部屋に入れば電波を受信でき，部屋外であれば受信できないように事前に設定した．受信側の Macbook Pro には，node.js のモジュールである node-beacon-scanner[2] を使用しビーコンの電波を受信する仕組みを実装した．

ビーコンは各部屋に設置されており，ビーコンの UUID や設置場所はビルモデルに記述されている．つまり UUID をもとにビルモデルにクエリを実行すれば，ビーコンの設置場所（ビル内のユーザの位置）を特定することができる．ユーザの位置特定は RBAC システムのフロントエンドサーバによって実行される．

ユーザの位置を特定する仕組みとして，ユーザが接続するネットワークにより特定する手法がある．部屋ごとにアクセスポイントを設置し，ユーザがアクセスしてきた際にそのネットワークを確認することで位置を特定するというものである．しかし，アクセスポイントの電波強度はビーコンほど自由に設定できない．そのためユーザが実際にいる部屋のネットワークではなく，別の部屋のネットワークに接続される場合があり，この手法ではユーザの位置を正確に特定できない可能性がある．そこで本研究では，より細かい粒度でユーザの位置を特定可能な，ビーコンを用いた手法を選択した．

3.2.2 RBAC システム

本実装ではセンサのみを管理する WoT システムを構築した．そのため RBAC システムでは，読み取り権限のみを管理し，書き込みや操作権限の管理は対象としていない．ロールの生成・割り当ては RBAC Manager によって Keycloak を利用して事前に行われる．実装ではキャンパスにおけるユーザグループとして，管理者・教員・学生・来訪者の 4 つを生成した．ユーザグループに対するロールの割り当ては以下の表のようになる，管理者・教員により多くの権限を与える一方，学生・来訪者には限定的な権限を与えた．また実際にアクセス可否を判断する場合には，フロントエンドサーバによってロールの確認に加え，位置情報や時刻の確認が行われる．

ユーザグループ	割り当てるロール
管理者	全ての部屋
教員	在籍するフロア，共用スペース
学生	共用スペース（会議室，講義室）
来訪者	会議室

表 3 ユーザグループと割り当てられるロール

RBAC システムのフロントエンドサーバは，ユーザからのアクセスに対しアクセス可否を判断する，アクセス可否の判断にはロール・位置・時刻の 3 つの要素を用いた．ただしユーザグループによって，アクセス可否に用いる要素が異なり，表 4 のようになる．

アクセス可否を判断するためのポリシ（以降アクセス可否ポリシと記す）は ECA ルールによって記述される．

ECA（Event Condition Action）ルールとは，ルールの実行のトリガになる事象（Event），ルールが実行されたときに確認される条件（Condition），実行される動作（Action）の 3 つを 1 つのルールとして記述するものである [20]．実装した ECA ルールの一部を下記に示す．

- Event
  学生ユーザから部屋 A（共用スペース）にある機器の情報を要求される
- Condition
  以下の 3 つを同時に満たすとき
  (1) ロール：部屋 A のロールを持つ
  (2) 位置：部屋 A にいる
  (3) 時刻：9〜18 時
- Action
  ユーザのアクセス（読み取り）を許可する

ユーザグループ	確認項目		
	ロール	位置	時刻
管理者	○	×	×
教員	○	×	○
学生	○	○	○
来訪者	○	○	○

表 4　ユーザグループと確認項目

ユーザは RBAC システムを介して WoT システムにアクセスすることとなる．実装した RBAC システムは HTTPS サーバで構成されており，ユーザと RBAC システムの通信は全て暗号化されている．図 7 にユーザがアクセスする際に行われる動作フローを示す．ただし場合によっては (3), (6) の動作は省略されることがある．

(1) ユーザは Keycloak サーバに対し HTTP POST リクエストを送信する．その際にユーザはユーザネームとパスワードを提供し，認証される必要がある．Keycloak サーバによって認証が完了すれば (2) へ移る．
(2) Keycloak サーバはユーザの認証後，アクセストークン（Json Web Token）をユーザに送信する．アクセストークンは有効期限やロールなどが含まれた文字列をエンコードしたものである．また悪意をもった攻撃者によるアクセストークンの改ざんを防ぐために，非対称署名アルゴリズム RS256 によって電子署名されている．
(3) ユーザは周囲のビーコンによってアドバタイズされた情報を受信後，UUID を取得しリストを作成する．周囲にビーコンがない場合は空のリストが作成される．ただし位置の特定が必要でないユーザグループに属するユーザの場合，このステップは省略される．
(4) ユーザはフロントエンドサーバに対して HTTP GET リクエストを送信する．その際に，Keycloak サーバから受信したトークンと，周囲から取得した UUID のリストを送信する．
(5) フロントエンドサーバはユーザからのリクエストに対し，アクセストークンを検証し有効期限が切れていないか，改竄されていないかを確認する．アクセストークンの検証後，アクセス可否ポリシに基づきアクセス可否を判断する．実装したアクセス可否ポリシは前項で述べたようである．ロールはアクセストークンから，時刻は HTTP リクエストのヘッダから，位置は 3.2.1 項で説明した仕組みで特定できる．位置特定が必要な場合（今回の例では学生と来訪者のみ）は (6) へ移る．ポリシによりアクセスを許可された場合は (7) へ，拒否された場合は (8) へ移る．
(6) 取得した UUID のリストをもとにビルモデルにクエリを実行し，位置を特定する．クエリ結果が NULL の場合は，ユーザはビル外にいると判断する．ただし (3) と同様にこのステップは省略される場合がある．
(7) アクセスを許可された場合，フロントエンドサーバは WoT システムから対象の機器のデータを取得する．
(8) アクセスを許可する場合は対象の機器のデータを，拒否する場合は HTTP ステータスコード 403 Forbidden error をユーザに送信する．

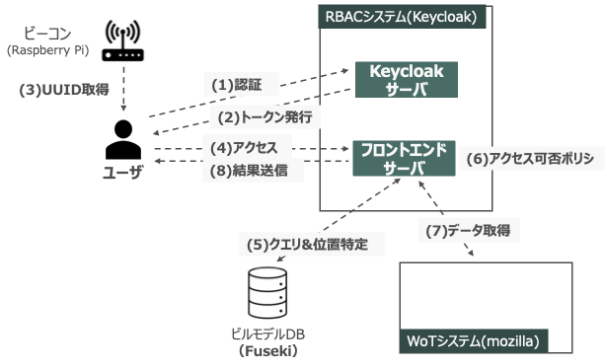

図 7　動作フロー

4. 評価

4.1 開発者の負担に関する評価

提案フレームワークでは，WoT Manager により WoT システムは自動で構築される．WoT Manager はビルモデルにクエリを実行後，WoT システムの Thing の生成に必要な情報を取得し，Json ファイルを生成する．生成後にテンプレートのプログラムに Json ファイルの情報を挿入し，Thing を生成する．

実装では 47 個のセンサの WoT システムを構築した．その際に必要なプログラムコードを，手動構築の場合と提案フレームワークを用いた場合で比較する．今回用いたセンサの Thing の生成のために，手動構築の場合は 1 つにつき

119 行のプログラムコードが必要であり，47 個の場合には 5593 行（119*47）必要となる．それに対し，提案フレームワークでは開発者の介入なしに WoT システムを構築することができる．またビルモデルによってそれぞれのビル構造の差，記述の差を吸収することができ，ビルによらず統一的な方法でシステムを構築可能である．

手動構築の場合は必要なコード数は機器の数に応じて比例的に増加するが，提案フレームワークでは一定である．そのため, 特に機器の数が膨大な大規模ビルの場合に，提案フレームワークは有効であると考えられる．

提案フレームワークでは, RBAC Manager により RBAC システムにおけるロールを自動で生成する．生成されるロールはビルモデルから抽出した部屋・階層などの建物の構造情報を利用したものであり，膨大な数の機器を抽象的なグループで管理している．

RBAC システム構築において，従来の RBAC では一つ一つの機器を管理しており，各ロールにどの機器の権限を割り当てるかを記述する必要がある．そのため機器の数が増えれば増えるほど必要な記述は増え，その構築負担は大きくなる．それに対して提案手法では，機器を抽象的なグループ単位（ロール）で管理し，ユーザグループとロールの関係を記述することで RBAC を実現する．そのため機器の数が増えても必要な記述は変わらず，構築負担は変化しない．

ビル内に新たに機器が設置される場合，従来のシステムであれば一つ一つの機器を管理しているため，どのロールに権限を与えるかを新たに記述する必要がある．それに対し提案手法では機器をグループ単位（ロール）で管理しており，新たな機器の Thing がビルモデルから生成された時点でグループ（ロール）に権限は追加される．例えば部屋 201 に機器 07 が設置された場合，Thing が生成された時点でロール *201* に機器 07 への権限が追加される．そのため新たに機器が追加された場合でも，RBAC システムにおいては追加の作業は必要ない．従来の手法では追加の作業は手動で行われるが，提案手法ではビルモデルに結びついて自動で行われるため，管理の漏れや誤りが少なくなるといえる．

提案フレームワークでは，建物の構造を利用したロールによって機器を管理している．そのため，デフォルトでは同じ部屋にある機器に関しては権限を分けることができず，例えば図 5 において「機器 01 の権限を与えるが，機器 02 の権限を与えない」といったことをする場合には，追加の実装が必要である．そういった場合にも対応するために，建物構造を利用したロールだけでなく，新たなロールを追加し拡張する必要がある．例えば機器の種類に関するロールを作成し，建物構造のロールと組み合わせることで，「部屋 A にある温度センサへの権限を与えるが，空調機への権限は与えない」といったことが可能になる．このように複数のロールを組み合わせてアクセス可否を判断することで，柔軟なアクセス制御が可能になると考えている．

4.2 セキュリティに関する評価

プロトタイプシステムでは不正アクセスを防ぐために，ロールを含んだアクセストークンを利用した．WoT システムにアクセスするためには，RBAC システムを通して，認証後にアクセストークンを取得する必要がある．攻撃者はトークンを盗むために，RBAC システムとユーザとの通信内容を解析する可能性があるが，RBAC システムの各サーバは HTTPS サーバであり通信内容は暗号化されているため，アクセストークンを盗まれる危険性は低い．

また攻撃者によるアクセストークンの改ざんを防ぐために，アクセストークンに電子署名をしている．発行されたアクセストークンは発行元のサーバ（Keycloak サーバ）の秘密鍵で署名される．アクセストークンの正当性はフロントエンドサーバの公開鍵で検証される．そのため攻撃者がアクセストークンを改ざんしアクセスした場合，RBAC システムは改ざんを検知しリクエストを拒否する．

またプロトタイプシステムでは，ロールの確認に加え，ユーザの位置や時刻を考慮したアクセス制御を実現している．これにより，万が一攻撃者がアクセストークンを取得しても，位置情報や時刻に関する条件を満たさない場合にアクセスを拒否することができる．またアクセストークンに有効期限が設定されているほか，不正を検知した場合にはアクセストークンを失効できるため，より安全といえる．このようにロールと複数のコンテキスト要素を組み合わせた認可により，セキュリティレベルを向上させている．

以上の議論からプロトタイプシステムは実用に向けて十分なセキュリティレベルをもっていると結論づける．しかし現在はユーザネームとパスワードのみによって認証しているため，パスワードが漏洩した場合にはアクセストークンが盗まれ，場合によっては不正アクセスされてしまう可能性がある．そのため今後は，追加の認証機能（ワンタイムパスワードなど）や不正アクセスを検知する仕組みの検討が必要である．

4.3 性能評価

性能評価ではアクセス制御のオーバヘッドを計測するために，以下の二つの場合においてクラウド上の計算機からセンサデータを要求するリクエストを送信した．Path 1 では，ユーザが WoT システムに直接リクエストを送信してからデータを取得するまでの時間を測定した．Path 2 は RBAC システムを介して WoT システムにアクセスした場合である．Path 2 では図 7 における，(5) アクセス可否の判断，(6) クエリ&位置特定，(7) データ取得の時間をそれぞれ測定した．

	Path 1	Path2		
項目	データ取得	(5) アクセス可否の判断	(6) 位置特定	(7) データ取得
時間 [ms]	7.3	7.3	11.3	6.4

表 5 計測結果

- Path 1：ユーザ ⇒WoT システム
- Path 2：ユーザ ⇒RBAC システム ⇒WoT システム

10 回の平均の計測結果を表 5 に示す．表から分かるように，ユーザが直接 WoT システムにアクセスした際のデータ取得までの時間（Path 1：データ取得）と，フロントエンドサーバが WoT システムからデータを取得する時間（Path 2：データ取得）の時間はほとんど同じである．そのためアクセス制御のオーバヘッドは，(5) と (6) にかかる時間（18.6 ms）に近似でき，オーバヘッドは非常に小さいといえる．

5. おわりに

本研究では，スマートビルにおけるアクセス制御システム構築の負担が膨大であるという問題に対して，システム構築の負担を軽減するアクセス制御フレームワークを開発した．提案フレームワークを用いたプロトタイプ実装では，作成したビルモデルから WoT システムと RBAC システムの一部の機能を自動的に構築できることを確認した．

今後の課題としては，より大規模なビルでの適用と管理対象機器数増加へのスケーラビリティが挙げられる．またユーザ多様性を受け入れる柔軟なロール拡張の検討なども必要となる．

謝辞 本研究の一部は，JSPS 科研費 20H00584, 18K11355 の助成を受けたものです．

参考文献

[1] Apache Jena - Home. https://jena.apache.org/, 2021.
[2] futomi/node-beacon-scanner - GitHub. https://github.com/futomi/node-beacon-scanner, 2021.
[3] Home - BrickSchema. https://brickschema.org/, 2021.
[4] Home - Web of Things (WoT) - W3C. https://www.w3.org/WoT/, 2021.
[5] ifcOWL - buildingSMART Technical. https://technical.buildingsmart.org/standards/ifc/ifc-formats/ifcowl/, 2021.
[6] Intel SRR3 — U.S. Green Building Council. https://www.usgbc.org/projects/intel-srr3, 2021.
[7] Keycloak. https://www.keycloak.org/, 2021.
[8] Mozilla IoT. https://iot.mozilla.org/, 2021.
[9] Project Haystack: Home. https://project-haystack.org/, 2021.
[10] RBC WaterPark Place — Oxford Properties. https://www.oxfordproperties.com/lease/office/88-queens-quay-west, 2021.
[11] WebThingsIO/webthing-node - GitHub. https://github.com/WebThingsIO/webthing-node/blob/master/example/multiple-things.js, 2021.
[12] テクノロジー・イノベーションセンター｜ダイキン工業株式会社. https://www.daikin.co.jp/tic, 2021.
[13] Bharathan Balaji, Arka Bhattacharya, Gabriel Fierro, Jingkun Gao, Joshua Gluck, Dezhi Hong, Aslak Johansen, Jason Koh, Joern Ploennigs, Yuvraj Agarwal, et al. Brick: Towards a Unified Metadata Schema for Buildings. In *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*, pages 41–50, 2016.
[14] Syafril Bandara, Takeshi Yashiro, Noboru Koshizuka, and Ken Sakamura. Access Control Framework for Api-Enabled Devices in Smart Buildings. In *2016 22nd Asia-Pacific Conference on Communications (APCC)*, pages 210–217. IEEE, 2016.
[15] Leepakshi Bindra, Kalvin Eng, Omid Ardakanian, and Eleni Stroulia. Flexible, Decentralised Access Control for Smart Buildings with Smart Contracts. *Cyber-Physical Systems*, pages 1–35, 2021.
[16] Leepakshi Bindra, Changyuan Lin, Eleni Stroulia, and Omid Ardakanian. Decentralized Access Control for Smart Buildings Using Metadata and Smart Contracts. In *2019 IEEE/ACM 5th International Workshop on Software Engineering for Smart Cyber-Physical Systems (SEsCPS)*, pages 32–38. IEEE, 2019.
[17] Buckman Alex H and Mayfield Martin and Beck Stephen BM. What is a Smart Building? *Smart and Sustainable Built Environment*, 2014.
[18] David Ferraiolo, Janet Cugini, et al. Role-based Access Control (RBAC): Features and Motivations. In *Proceedings of 11th Annual Computer Security Application Conference*, pages 48–241, 1995.
[19] Dominique Guinard, Vlad Trifa, Friedemann Mattern, and Erik Wilde. From the Internet of Things to the Web of Things: Resource-Oriented Architecture and Best Practices. In *Architecting the Internet of Things*, pages 97–129. Springer, 2011.
[20] Dennis McCarthy and Umeshwar Dayal. The Architecture of an Active Database Management System. *ACM Sigmod Record*, 18(2):215–224, 1989.
[21] Nic Newman. Apple iBeacon Technology Briefing. *Journal of Direct, Data and Digital Marketing Practice*, 15(3):222–225, 2014.
[22] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and Complexity of SPARQL. *ACM Transactions on Database Systems (TODS)*, 34(3):1–45, 2009.
[23] 佐々木美穂 and 越塚登. Smart building の機器操作における, location-proof 機構を使用したアクセス制御の実現. In *IEICE Conferences Archives*. The Institute of Electronics, Information and Communication Engineers, 2018.
[24] 粕谷貴司 and 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**, 38(8):466–469, 2018.