情報処理学会論文誌 Vol.62 No.4 995–1007 (Apr. 2021)

IoTの柔軟な相互運用性を実現するソフトウェアアーキテクチャの提案

横山 史明[1,a)] 沢田 篤史[1,b)] 野呂 昌満[1,c)] 江坂 篤侍[1,d)]

受付日 2020年8月3日, 採録日 2021年1月12日

概要：IoT の利便性を向上させるためには，多様な環境においてアプリケーションを稼働させることのできる相互運用性が重要である．また，利用状況や嗜好に応じたサービスを提供する柔軟性も重要である．スマートホームなど IoT 環境にとっての相互運用性とは，利用者の所持する機器を最大限に活用してサービスを提供できることである．柔軟性とは，刻々と変化する利用者の状況（時間や位置など）と意思（目的や嗜好など）に合わせ，当初アプリケーションが想定していない方法によってサービスを提供できることである．IoT 製品や標準の乱立によって，相互運用性の確保は特定の製品群の中だけにとどまっているのが現状である．柔軟性に関しては，センシングや機械学習などの要素技術の利用がアプリケーションごとに行われている．これら相互運用性や柔軟性の確保が場あたり的に行われると，IoT アプリケーションソフトウェア開発の効率やできあがったソフトウェアの保守性に悪い影響が及ぶことが懸念される．本研究では，IoT における柔軟性と相互運用性の確保をソフトウェア構造の問題ととらえ，スマートホームでの動的適応を可能とするソフトウェアアーキテクチャを定義する．柔軟で相互運用可能なアプリケーションを保守しやすく構築する基盤としてこのアーキテクチャを提案することで，上述した問題の解決を図る．

キーワード：ソフトウェアアーキテクチャ，スマートホーム，IoT，相互運用性，柔軟性

A Software Architecture Enabling Flexible IoT Interoperability

Fumiaki Yokoyama[1,a)] Atsushi Sawada[1,b)] Masami Noro[1,c)] Atsushi Esaka[1,d)]

Received: August 3, 2020, Accepted: January 12, 2021

Abstract: Assuring interoperability among various devices is vital for realizing useful IoT environment in smart homes. Providing flexible services which can adapt users' preference is also important. Due to different standards or products emerging out of different organizations, interoperability in a smart home is limited to the case when we use the specific set of products that conform to one specific standard. For flexibility concern, each product adopts different approach to sensing and learning its user's preference. Without any systematic approaches applied to IoT application development to ensure interoperability and flexibility, we are not able to speed-up application production and resulting products may have poor maintainability. In this study, we propose a software architecture enabling flexible IoT interoperability in smart homes. Based on our architecture, we can easily realize maintainable and flexible IoT applications which can utilize home appliances which conform to different standards.

Keywords: software architecture, smart homes, IoT, interoperability, flexibility

1. はじめに

IoT の応用先の 1 つにスマートホームがある．家庭における IoT の導入シナリオとして，利用者が好みの製品を購入しながら，すでに家庭にある製品とネットワーク上で連携させることで，徐々に IoT の適用範囲を拡大させてい

[1] 南山大学
Nanzan University, Nagoya, Aichi 466–8673, Japan
[a)] m19se701@nanzan-u.ac.jp
[b)] sawada@nanzan-u.ac.jp
[c)] yoshie@nanzan-u.ac.jp
[d)] eatsushi@nanzan-u.ac.jp

く，というものがあげられる．特にこのようなシナリオのもとでは，将来追加導入されるものも含めて，利用者が自身の保有する製品を有効に活用して，IoT の恩恵を享受できることが望まれる．

本研究の目的は，スマートホームなどの IoT システムにおいて，柔軟な相互運用性を持つアプリケーションを開発するための工学的基盤を構築することである．ここで，相互運用性とは，ISO/IEC25010 [13] が「製品または構成要素が情報を交換し，交換された情報を使用することができる度合い」と定める特性である．また，柔軟性とは，「初期的に定義されていたものを超越した利用状況において，製品またはシステムを使用することのできる度合い」と定める特性である．すなわち，本研究では，スマートホームなどに適用される IoT アプリケーションにとって，

- 特定のハードウェアやネットワーク，基本ソフトウェアに依存することなく，情報を交換しながら稼働することのできる相互運用性
- 時間や利用者の位置といった状況や，利用者の目的や嗜好に（たとえそれらが当初想定していなかった状況であったとしても）適応したサービスを提供することのできる柔軟性

が重要であるととらえ，これらの特性を持つソフトウェアの開発を容易にすることを目的とする．

IoT 製品や標準の乱立によって，相互運用性の確保は特定の製品群の中だけにとどまっているのが現状である．互換性が保証された製品群で家庭を一気にスマートホーム化する方法以外では，家中の家電製品を有効活用して利便を享受することは難しい．柔軟性に関しても，特定のアプリケーション内で利用者の状況や嗜好を考慮することはできるが，それをより広い範囲へ展開し利便性を享受することは難しい．

本研究では，IoT における柔軟性と相互運用性の確保をソフトウェア構造の問題ととらえ，スマートホームでの動的適応を可能とするソフトウェアアーキテクチャを設計する．相互運用性や柔軟性を確保することは，アプリケーションの提供するサービスの中身とは異なる関心事である．アーキテクチャ設計にあたり，IoT 機器を相互に連携させるための論理と，利用者の状況や嗜好に適用させるための論理とを明確に分離した基本構造を定義する．提案するアーキテクチャでは，この基本構造を具体化することで，IoT 機器間の多層にわたる連携と適応のための構造を定義する．これにより，スマートホームにおける柔軟な相互運用を可能とする．

提案するアーキテクチャに基づいて IoT 機器間の動的適応が可能であることを示すために，簡単なメッセージ通知アプリケーションを実装する．この実装を通じ，柔軟な相互運用が可能であること，すなわち，利用者の状況に応じてメッセージ送信先の IoT 機器を変更し，その機器に合わせた形態のメッセージをその機器の機能に合わせた方法で通知できることを確認する．

2. IoT における相互運用性と柔軟性

2.1 IoT アプリケーションの相互運用性と柔軟性

システムおよびソフトウェアの品質に関する標準である ISO/IEC25010 [13] は，相互運用性（interoperability）を，互換性（compatibility）という製品品質特性の下の副特性として，「製品または構成要素が情報を交換し，交換された情報を使用することができる度合い」と定めている．また，柔軟性（flexibility）を，利用状況網羅性（context coverage）という利用時の品質特性の下の副特性として，「初期的に定義されていたものを超越した利用状況において，製品またはシステムを使用することのできる度合い」と定めている．

本研究が対象とするスマートホームにおける IoT アプリケーションでも，相互運用性と柔軟性を ISO/IEC25010 の定義に基づいて検討する．すなわち，IoT アプリケーションの相互運用性を，特定の IoT 機器ハードウェアやネットワークプロトコル，基本ソフトウェアやライブラリに依存することなく，必要な情報を交換し，稼働することのできる性質ととらえる．また，柔軟性を，時間や利用者の居場所といったスマートホームの状況，利用者の目的や嗜好に，たとえそれらが IoT アプリケーションの初期的な想定になかったものであったとしても，適応したサービスを提供することのできる性質ととらえる．

スマートホームで求められる IoT の柔軟な相互運用について，簡単なアプリケーションを例に説明する．**図 1** に洗濯完了通知アプリケーションの概念図を示す．

このアプリケーションは利用者に洗濯の完了を通知することを目的とし，スマートスピーカを介して洗濯完了メッセージを出力するように作られているとする．利用者は家庭内を移動し，様々な活動をすることから，アプリケーションの初期的な設定であるスマートスピーカが必ずしも適切な出力機器ではない場合もある．このアプリケーションが利用者の状況や嗜好に適応して効果的に目的を果たすために，たとえば，

- 洗濯が完了した時点での利用者の位置に応じて出力
- 洗濯が完了した時間帯と利用者の就寝時間帯を考慮して出力

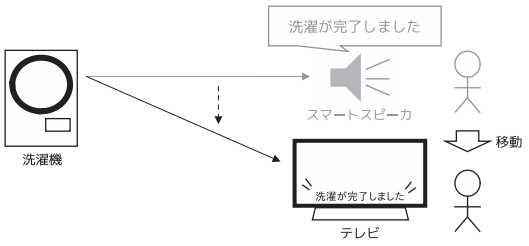


図 1 洗濯完了通知アプリケーション
Fig. 1 Notification application for IoT washing machine.

のような柔軟な対応が求められる．

これをIoT機器の連携，すなわち相互運用により実現するためには，

- ネットワーク接続レベルの変更
  センサによる利用者の位置の検出結果や時間帯に応じた，接続相手のIoT機器の変更
- メッセージングプロトコルレベルの変更
  相手のIoT機器が受付可能なメッセージ通知プロトコルに応じた，メッセージングプロトコルの変更
- 意味レベルの変更
  相手のIoT機器の機能や利用者の嗜好に応じた，表現メディアやメッセージ内容の変更

に対処可能であることが求められる．

2.2 IoTの相互運用性に関する課題

ネットワークを通じて機器どうしを連携させるために様々な標準化が行われている．oneM2M [15]では，様々な応用領域を想定したサービスアーキテクチャとプロトコル群を想定した標準が定義されている．IoT-A（Internet of Things Architecture）[2]では，IoT機器やサービスなど概念間の関係を参照モデルとして定義し，それに基づいた標準化が行われている．W3CによるWoT（Web of Things）[18]の標準化では，Webアプリケーションや Webサービスの枠組みをIoT機器に適用するための議論が行われている．

家電機器に特化した標準としては，機器をホームネットワークによって連携させるための規格ECHONET Lite [8]がある．この規格では，高性能のホームサーバにより，家庭内のすべてのIoT機器を制御する方式をとっている．この他にも，各ベンダから様々なIoT機器が提案され，家電制御やセンサ情報取得に利用されている．

前章で述べたIoT機器導入のシナリオにおいて，ネットワーク接続レベル，メッセージングプロトコルレベル，意味レベル，それぞれの変更への柔軟な適応を実現するためには，既存の標準や製品群の単なる組合せでは難しい．いずれの標準や製品群も，それぞれに閉じた範囲のIoT機器どうしが連携できることを第一義として定められているからである．標準をまたぐIoT機器の相互運用性を実現する機構が必要である．

2.3 柔軟なサービスのための共通開発基盤の必要性

情報家電の知的制御のためにオントロジを用いる試みは古くから行われてきた [21]．前節で述べたoneM2MやWoTの標準でも，ネットワーク接続レベル，メッセージングプロトコルレベルでの標準化だけでなく，IoTに関連する概念とそれを表す語彙についての標準化を行っている．oneM2Mでは，標準に従って開発されるIoTシステムにおける意味レベルの互換性を高めるために，Base Ontologyと呼ぶコア概念を拡張可能な形で定めている [16]．WoTでは，ネットワーク接続されるセンサ，アクチュエータに関して，それらを利用した観測や動作に関する特徴についての標準オントロジ（SSN）を定めている [19]．

意味レベルの相互運用性を確保するためのアプローチも行われている．井垣らは，ネットワーク家電のサービスを公開し，このサービスを用いた家電の連携を実現するための研究を行っている [12]．サービス指向アーキテクチャに基づいた実装により機器間の結合を疎にし，相互接続性や機器拡張性を向上させることで，適応的なサービスを可能にしている．

Desaiらは，IoTアプリケーションの間で意味変換を行うセマンティックゲートウェイサービスのアーキテクチャを提案している [6]．他にも，意味レベルでの連携を可能とするIoTゲートウェイのアーキテクチャがDattaらにより提案されている [4], [5]．これらゲートウェイサービスを特定領域へ応用した結果も報告されている [14]．利用者の状況や嗜好に応じたサービスを実現するために，意味レベルの相互運用性を高めるためのゲートウェイを導入するアプローチは有効である．ネットワーク接続およびメッセージングプロトコルのレベルの変更に対処する機構と組み合わせることで，柔軟に接続先や振舞いを変更するアプリケーションソフトウェアの開発が可能になる．

このようなアプリケーションにおいて，利用者の状況や嗜好に応じた変更や，接続先のIoT機器に応じた変更に関する論理は，アプリケーション本来の論理とは異なる関心事に基づいている．これらについて場あたり的な実装が行われると，アプリケーション開発効率が損なわれるだけでなく，できあがったソフトウェアの保守が困難になることが懸念される．柔軟で相互運用可能なIoTアプリケーションを開発するためには，これらの要素技術の統合を系統的に支援するための共通基盤が必要である．

3. 柔軟な適応を可能とするソフトウェアアーキテクチャ

前章では，スマートホームにおける相互運用性を確保するために，ネットワーク接続，メッセージングプロトコル，意味の3つのレベルでの適応を行う必要があることを説明した．また，これらの適応を実現するための要素技術を統合し，利用者の状況に応じた柔軟性の高いIoTアプリケーションの開発を容易にするための基盤が必要であることも説明した．

ソフトウェアアーキテクチャは，そこで定義される構造に基づいて実装されるソフトウェアの実行時の性質および開発と保守にかかわる性質を決定づけるものである [1]．本研究では，IoTアプリケーションにおいて，柔軟性，相互運用性を向上させるという課題を，それらの性質を適切に満足するためのソフトウェア構造を定義する問題ととらえ

る．すなわち，動的な適応を可能とするためのソフトウェアアーキテクチャの設計を本研究の目的とする．これにより，柔軟性と相互運用性の高いスマートホームアプリケーション構築基盤の提供を目指す．

動的適応を可能とするアーキテクチャの設計にあたり，本研究では，我々がこれまでに提案してきたPBR（Policy Based Reconfigration）パターン [9] とGoFデザインパターン [10] のAdapterパターンを組み合わせた基本構造を定義する．この基本構造を具体化することで提案するアーキテクチャを定義する．

本章では以下，3.1節でPBRパターンに触れた後，3.2節で動的適応を実現するための基本構造，3.3節で提案アーキテクチャの設計について説明する．

3.1 PBRパターン：動的適応のためのアーキテクチャパターン

PBRパターン [9] とは，インタラクティブシステムなど，利用者や外部環境に応じてその振舞いを変化させることが求められるシステムにおいて，自己適応を実現するために定義したアーキテクチャパターンである．PBRパターンが規定する静的構造を**図 2** に示す．ここで，「オブジェクト」と「コンテキスト」間の関連と「ポリシ」とを結ぶ破線は，「ポリシ」が「オブジェクト」および「コンテキスト」間の関連クラスであることを意味する．「オブジェクト」は，システムの構成要素のうち自己適応のために再構成される対象となるモジュールである．「コンテキスト」は，利用者や外部環境の状況を表すモジュール，「ポリシ」はコンテキストに応じた再構成の方針を規定するモジュールである．「再構成器」は，「オブジェクト」のインスタンス生成やリンクの張り替えなどによって「オブジェクト」の再構成を行う役割を担うモジュールである．

図 3 には，PBRパターンに従う動的振舞いを示す．ここで，関連線に付与された数字はメッセージ通信の順番を，破線は関連するメッセージ通信の横取りを意味する．システムの再構成による自己適応は，「オブジェクト」による「コンテキスト」の更新（1）をきっかけに開始する．このメッセージを横取り（2）した「ポリシ」は，そこに規定された再構成方針に基づき，更新された「コンテキスト」の値（3）に応じた再構成の指示を「再構成器」に送信する（4）．「再構成器」が対象の「オブジェクト」を再構成する（5）ことで，「オブジェクト」からのメッセージ通信先や通信方式などを切り替える（6）ことができる．

このように，PBRパターンは，「コンテキスト」，「ポリシ」，「再構成器」で示される動的適応の論理を，アプリケーション論理（「オブジェクト」間の通信）から分離させるための基本構造を規定する．この構造を適用することで，アプリケーション論理では初期的に想定していなかった構成への変更といった動的適応を可能とし，柔軟性の向上に対

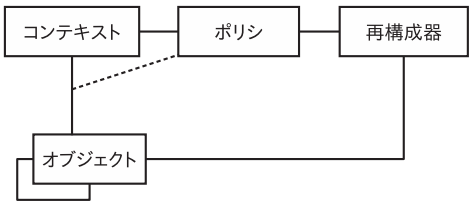


図 2 PBRパターンの静的構造
Fig. 2 PBR pattern — static structure.

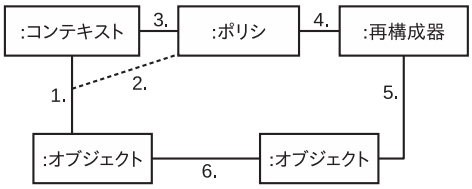


図 3 PBRパターンの動的振舞い
Fig. 3 PBR pattern — dynamic behavior.

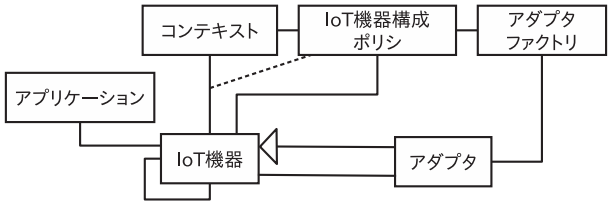


図 4 動的適応のための基本構造
Fig. 4 Base structure for dynamic adaptation.

する要求に応えることができる．

3.2 動的適応のための基本構造

これまでの議論から，本研究で定義するアーキテクチャに対する要求を次のようにまとめることができる．

- 異種IoT機器間の相互運用を可能にする適応機構を提供する．
- 利用者の状況や嗜好に応じてIoTアプリケーションの構成を動的に変更する適応機構を提供する．

これらの要求を満たすために定義した基本構造を**図 4** に示す．1つ目の要求には，異機種間の差異を吸収するためのAdapterパターンを適用することで対応する．すなわち，データや制御をやりとりする「IoT機器」に「アダプタ」を関連付け，必要なプロトコル変換などの役割を担わせる．

2つ目については，前述のPBRパターンを適用して対応する．図4では，「IoT機器」，「コンテキスト」，「IoT機器再構成ポリシ」，「アダプタファクトリ」からなる構造にPBRパターンが適用されている．これにより，センサなどの「IoT機器」から取得した情報から定められる「コンテキスト」の変化に応じて，動的にアプリケーションに参画する「IoT機器」を再構成する構造を定義できる．

基本構造に用いたAdapterパターンは，異なる「IoT機器」間の適応に関する論理をアプリケーション論理から独

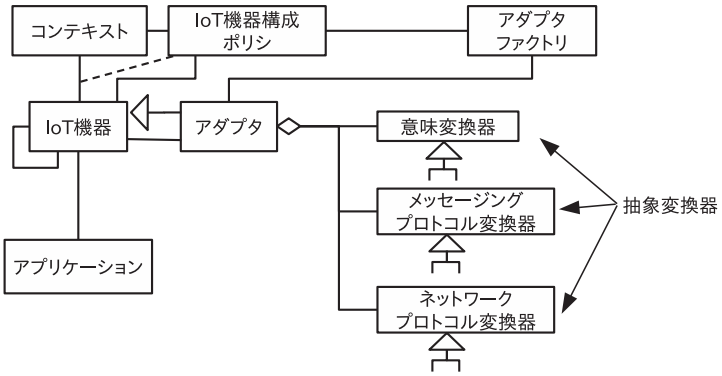


図 5 スマートホーム IoT アプリケーションのためのアーキテクチャ
Fig. 5 Architecture for smarthome IoT applications.

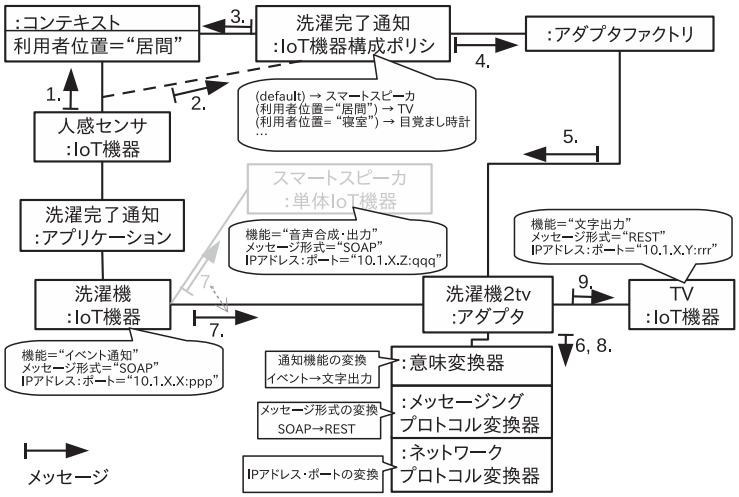


図 6 洗濯完了通知アプリケーションの動作
Fig. 6 Behavior of notification application for IoT washing machine.

立させるための設計解を与える．PBR パターンは，「コンテキスト」とそれに応じた振舞いの選択に関する論理と，構成変更の論理とをそれぞれ独立させるための設計解を与える．この基本構造では上記 2 つのパターンを適用することで，動的な再構成，すなわち柔軟性にかかわる論理を「IoT 機器再構成ポリシ」モジュールに，プロトコルなどの変換，すなわち相互運用性にかかわる論理を「アダプタ」モジュールに，それぞれアプリケーション論理から分離している．異なる関心事を明確にモジュールとして分離する構造により，IoT アプリケーション開発を容易にするための基盤を提供する．

3.3 アーキテクチャ設計

スマートホームにおける IoT アプリケーションのためのソフトウェアアーキテクチャを前節の基本構造にしたがって設計した結果を**図 5** に示す．前述したネットワーク接続，メッセージングプロトコル，意味の 3 つのレベルでの適応を実現する役割を担うモジュールを，POSA パターン [3] の Layers パターンを適用し，それぞれ「ネットワークプロトコル変換器」，「メッセージングプロトコル変換器」，「意味変換器」として定義した．各レベルの変換器は，Template Method パターン [10] に基づき，変換処理のスケルトンを提供する抽象クラスとして定義し，具体的な変換処理を具象クラスにて実装する構造とした．相互運用性を確保するために異なるレベルの適応が必要な場合は，抽象変換器のクラスを追加することで対応することができる．

このアーキテクチャに基づいて構築されるアプリケーションの振舞いについて，図 1 に示した洗濯完了通知アプリケーションを例に説明する．**図 6** に洗濯完了通知アプリケーション動作中の一時点でのメッセージ通信の様子を示す．

ここで「洗濯機」は，完了メッセージを SOAP メッセージ形式で送信するものとし，通知先は「スマートスピーカ」が指定されている．このアプリケーションに対し，利用者が居間にいる状況においては「TV」を通知先としたいという要求があるとする．このとき，変更先の「TV」は REST メッセージのみを受け付け，完了通知を音声合成して出力する機能はなく，画面に文字を出力する機能のみが搭載されているものとする．

図には説明のためにメッセージ通知を矢印で併記してあ

り，矢印に添えられた番号がメッセージの順番を表す．利用者の状況に適応した通知先変更を行わない場合の洗濯完了通知アプリケーションは「洗濯機」へ完了通知依頼メッセージを送付することで開始され，洗濯が完了するとそれを「スマートスピーカ」に通知する（図中，灰色で示したメッセージ 7）．

一方，利用者の居場所を「コンテキスト」として，それに適応した通知先の変更を実現するために「人感センサ」を用いるものとする．動的再構成は，利用者の移動を検知した「人感センサ」が起こす「コンテキスト」の変更（メッセージ 1）をきっかけに行われる．「IoT 機器再構成ポリシ」がセンサからのメッセージを横取り（2）し，変更後のコンテキスト（利用者位置= “居間”）に応じて（3），「洗濯機」が完了メッセージを通知すべき IoT 機器を「スマートスピーカ」から「TV」に切り替えるよう，「アダプタファクトリ」に再構成の指示を行う（4）．

「アダプタファクトリ」は，通知先を「TV」に切り替えるのに必要となるアダプタ「洗濯機 2tv」を生成する．これを「TV」に関連付けるとともに，「洗濯機」の完了通知先を「スマートスピーカ」からこのアダプタに切り替える（5）．アダプタでは，切り替えにともなって必要となる変換器を，意味，メッセージングプロトコル，ネットワーク接続のレベルでそれぞれ生成（6）することで「洗濯機」からの完了通知を「TV」で出力できるようにする．その後，「洗濯機」は完了通知を元々の通知先である「スマートスピーカ」に送信（灰色の 7）しようとするが，実際にはそれは「洗濯機 2tv」に送信（黒色の 7）され，構成する変換器がそれぞれのレベルでの変換を行い（8），通知先の「TV」に通知する（9）．

通信相手の IP アドレスを変更する方法や，メッセージ形式とそれにともなう通信手順を変更する方法など，以上で説明した振舞いを実現する技術には様々な選択肢がある．このアーキテクチャでは，動的適応のためのアダプタおよび変換器を，特定の実装技術には依存しない変換論理のためのモジュールとして定義している．

4. アーキテクチャに基づく実装例

提案するアーキテクチャに基づいて，異種 IoT 機器間の柔軟な相互運用が可能となることを示すために，簡単なメッセージ通知アプリケーションの実装を行った．この実装を通じて次の点について確認することで，提案アーキテクチャの妥当性を示す．

- 異なる IoT 機器間でのメッセージ通知方法の変更，すなわち相互運用が可能であること
- 利用者の状況に応じて動的な，すなわちアプリケーションにおいて当初想定しない柔軟なメッセージ通知が可能であること

本章では以下，実装したプロトタイプアプリケーションについて説明する．

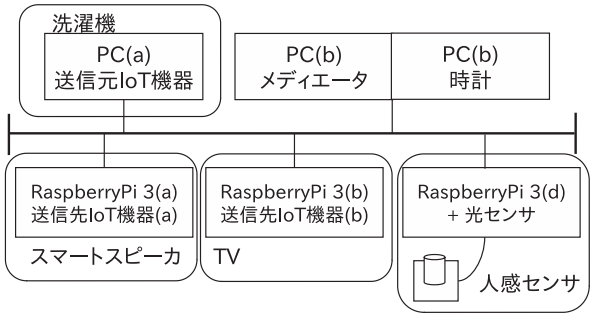


図 7 メッセージ通知アプリケーションの構成
Fig. 7 Abstruct structure of message notification application.

実装したプロトタイプアプリケーションのハードウェアおよびネットワーク構成を**図 7** に示す．これまでに説明してきた洗濯完了通知アプリケーション（図 1）を想定した構成となっているが，実際の家電機器の代わりに Raspberry Pi 3 とノート PC を用いている．これらの機器は，すべて同一ネットワークセグメント内に接続した．

ここでは，PC(a) を洗濯機に見立て，ソケット通信を用いて独自形式のメッセージ通信を行うクライアントとした．また，Raspberry Pi 3(a) をスマートスピーカに見立て，洗濯機からのソケット通信を待ち受けるサーバとした．ここで，洗濯完了通知アプリケーションは，PC(a) から RaspberryPi 3(a) へ固定的に完了通知メッセージを送付するものとして実装する．

一方，Raspberry Pi 3(b) は TV に見立て，HTTP 通信でメッセージを受けとるサーバとした．Raspberry Pi 3(c) には，光センサ（Grove Light Sensor）を接続して人感センサを模擬している．PC(b) をメディエータとしているが，これは，動的適応を仲介するために実装上導入した要素である．また，IoT 機器として時計（PC(b) を便宜上共有）も接続し，時間をコンテキスト情報として用いることができるようにしている．

これらの機器構成を用い，前章で提案したアーキテクチャに基づいて次のシナリオを実装した．

- メッセージ送信先の変更（ネットワーク接続レベルの適応）
  洗濯機（PC(a)）からスマートスピーカ（Raspberry Pi3(a)）宛に送出したメッセージの送信先を TV（Raspberry Pi 3(b)）に変更
- メッセージ形式の変更（ネットワーク接続およびメッセージングプロトコルレベルの適応）
  洗濯機からスマートスピーカ宛に送出したメッセージの送信先を TV に変更し，HTTP アダプタによりプロトコルを変換
- 機能と IoT 機器構成の変更（3 つのレベルの適応）
  人感センサの検知結果と時間帯に応じ，IoT 機器構成ポリシによって，メッセージ送信先を TV に変更する

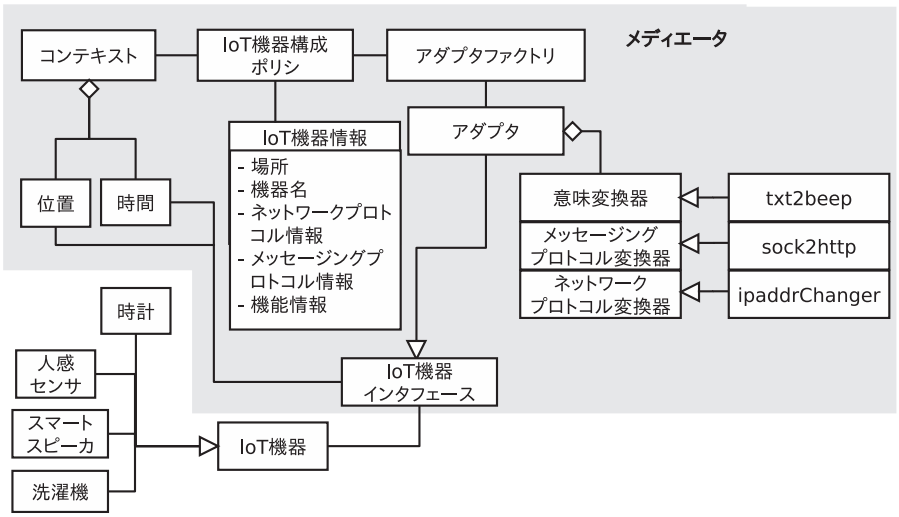


図 8 メッセージ通知アプリケーションの詳細構造
Fig. 8 Detailed structure of message notification application.

とともに，アダプタの動的生成により，メッセージングプロトコルを HTTP に，通知機能を音によるビープ機能に変更

プロトタイプアプリケーションにおいてこれらのシナリオを実現するために，前章で提案したアーキテクチャを**図 8** に示すように具体化した．この設計では，IP アドレスやメッセージングプロトコル，機能の動的変換の役割をメディエータに担わせる．IoT 機器からのメッセージの送信先をすべてメディエータに向けることとし，メディエータがコンテキストに応じた通知メッセージの送信先を動的に変更する実装構造としている．

この設計に基づいて，上記 3 つのシナリオを実装し，いずれも問題なく動作することを確認した．これらのシナリオを実装するにあたり，洗濯機とスマートスピーカの間のメッセージ通知を想定した元々の洗濯完了通知アプリケーションに対する大きな変更は必要なかった[*1]．

さらに，この環境において，新たな IoT 機器を追加してアプリケーションを稼働させるために必要な作業を明確にし，提案アーキテクチャの有効性を示すために次のシナリオに基づく実装を行った．

- メッセージ送信先の IoT 機器追加
  メディエータ（PC(b)）に SwitchBot[*2]ボットを接続し，洗濯機の完了通知先として指定する．SwitchBot ボットは，単にスイッチを押すための機器であることから，これを利用者の部屋の電灯スイッチや，呼び鈴スイッチを押すことにより洗濯完了を通知することを想定している．
- メッセージ送信元の IoT 機器追加
  メディエータに SwichBot 温湿度計を接続し，TV（Raspberry Pi3(b)）に温度を表示させる．

前者のシナリオでは，新たな送信先 IoT 機器の追加にともなう IoT 機器構成ポリシとアダプタの実装が必要となる．後者のシナリオでは，新たな送信元 IoT 機器からの情報を適切な場所へ通知するための新たなアプリケーション実装が必要となる．

前者のシナリオを実現するにあたり必要となる作業は次のとおりである．

- IoT 機器情報の追加：IoT 機器情報のインスタンスを生成し，IoT 機器構成ポリシと関連付ける．ここでは，IoT 機器構成ポリシから参照される機器情報として次を登録する．
  - 場所：（設置する部屋・位置する部屋の名前）
  - 機器名：（機器の識別名）
  - ネットワークプロトコル情報：BLE（Bluetooth Low Energy）によるネットワーク接続であることと MAC アドレス
  - メッセージングプロトコル情報：SwithBot の API によるメッセージングを受け付けること
  - 機能情報：スイッチを押す（Press）機能を持つこと
- IoT 機器構成ポリシの更新：コンテキストの値に応じて通知先を SwitchBot ボットとすることをポリシに追加する．
- アダプタの追加：3 つのレベルのアダプタとして次を追加する．
  - ネットワークプロトコル変換器：指定した MAC アドレスに対する BLE のコネクションを生成，消去
  - メッセージングプロトコル変換器：BLE コネクション上で SwitchBot ボットへの制御命令をエンコード

[*1] 実際には，実装を簡単化するために，洗濯完了メッセージの通知先として設定した IP アドレスの値をスマートスピーカのものからメディエータのものに変更した．これについては，代理 ARP 応答などを利用してメッセージを横取りするなどの方法を用いることで，アプリケーションをまったく変更せずに通知先の変更が可能になると考えている．
[*2] https://www.switchbot.jp

して送信
– 意味変換器：メッセージ通知機能をスイッチを押す（Press）機能に変更

一方，後者のシナリオを実現するにあたり必要となる作業は次のとおりである．

- 温度通知アプリケーションの作成：メディエータ上で，SwitchBot 温湿度計から温度情報を取得し，TV に対し温度通知を行うアプリケーションを作成する．
- （必要に応じて）IoT 機器構成ポリシの記述：温度通知アプリケーションが，コンテキストに応じてどの機器に温度を通知するかを記述する．
- （必要に応じて）アダプタの追加：温度通知先のアダプタを 3 つのレベルで作成・追加する．

これら 2 つのシナリオに基づき，洗濯器からの完了通知メッセージを受けたときに SwitchBot ボットがスイッチを押せること，メディエータに接続された SwitchBot 温湿度計の計測する温度データを TV に表示できることを確認した．

これら 2 つのシナリオの実現にあたり，IoT 機器情報の追加，IoT 機器構成ポリシの更新，アダプタの追加，温度通知アプリケーションの作成では，いずれも IoT 機器追加前のプロトタイプアプリケーションに対してソースコードを追加することが必要であった．一方で，通知先の IoT 機器を新たに追加する前者のシナリオでは，洗濯完了通知アプリケーションに対する変更は必要なかった．これらのうち，コーディングが必要であったコンポーネントについても，それぞれに記述すべき内容はアーキテクチャにおいて明確に定義されていることから，そのコーディングは容易に行うことができた．

これらの結果から，提案するアーキテクチャに基づくことで，IoT アプリケーションの相互運用性と柔軟性を担保できることを確認した．

一方で，スマートホームでの利用を想定する場合，利用者が簡便にメッセージ通知先の変更やその条件を指定するなど，IoT 機器構成を変更できることが望ましい．機器構成の変更にはソースコードの追加が必要となることから，本研究の提案のみではこの要求に十分に応えることができていない．コーディングが必要なコンポーネントのうち，IoT 機器構成ポリシについては，コンテキストが満たす条件とそれに応じて生成するアダプタの組合せを，たとえば表形式で記述し，これを解釈実行する枠組みを実現することでコーディングが不要となると考えている．その検討と実現は今後の課題としたい．

5. 考察

本研究では，柔軟性，相互運用性を考慮した IoT アプリケーションを構築する基盤となるソフトウェアアーキテクチャを提案している．本章では以下，提案したアーキテクチャの妥当性について考察する．

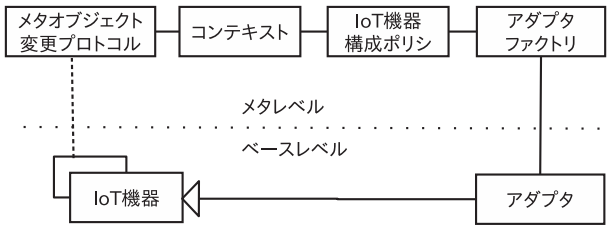


図 9 Reflection パターンを適用した動的適応
Fig. 9 Dynamic adaptation in reflection pattern.

5.1 アーキテクチャ設計の妥当性

本研究で提案するアーキテクチャには，動的適応を実現するために PBR パターンを適用した．同様のパターンとして，Reflection パターン [3] を適用することが考えられる．このパターンでは，アプリケーションをメタレベルとベースレベルに分割し，メタレベルの変更をすることで振舞いを動的に変更することを可能とする．

Reflection パターンを適用した IoT アプリケーション動的適応のための構造を**図 9** に示す．この場合，動的適応のための論理が「メタオブジェクト変更プロトコル」と「IoT 機器構成ポリシ」に，プロトコル変換などにかかわる論理はベースレベルの「アダプタ」にそれぞれ分離されていることから，PBR パターンと同様の関心事分離を実現できているといえる．一方，「メタオブジェクト変更プロトコル」は「コンテキスト」のメタ情報を保持し，「IoT 機器」の動作更新に応じて「コンテキスト」を変化させる．この点が PBR パターンを適用した場合と異なる．

Rainbow フレームワーク [11] は，自己適応のためのアーキテクチャスタイルとアプリケーションフレームワークである．自己適応計算の一般的な構造をアーキテクチャとして定義し，可変部をプラグインすることで，様々なアプリケーションにおいて自己適応の実現を可能としている．

Rainbow フレームワークを本研究の想定する IoT アプリケーションの動的適応にあてはめると**図 10** に示す構造となる．「IoT 機器」の状況が「IoT 機器監視器」によって取得され，「コンテキスト評価器」により評価される．その結果に応じて「IoT 機器ポリシ実行器」が「アダプタファクトリ」に「アダプタ」を生成させる．Rainbow フレームワークを適用し，次の 5 つの可変点を実現することでアプリケーションの構築が可能となる．

- 「コンテキスト定義」：コンテキストとして IoT 機器から取得するデータ型や属性の定義
- 「評価規則」：コンテキストの評価規則
- 「構成ポリシ」：コンテキストに応じた適応のポリシ
- 「適応処理」：適応処理論理
- 「変換論理」：ネットワークプロトコルやメッセージングプロトコルの変換に関する論理

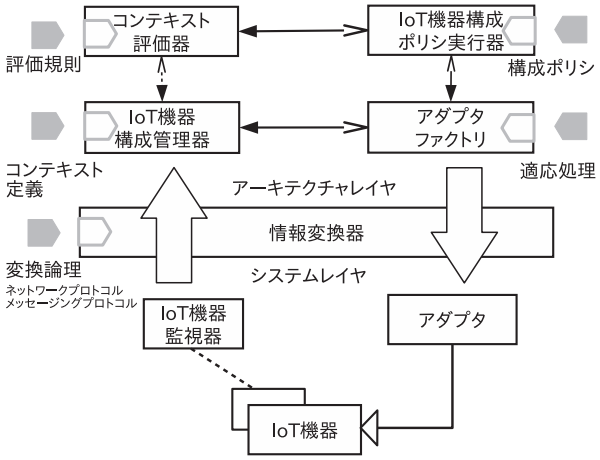


図 10 Rainbow フレームワークを適用した動的適応
Fig. 10 Dynamic adaptation using Rainbow framework.

コンテキストの「評価規則」と「適応ポリシ」が独立している点，ネットワークプロトコルとメッセージングプロトコルの「変換論理」を個々のアダプタから独立させている点が PBR パターンを用いた構造と異なる．

Reflection パターンや Rainbow フレームワークに対し，本研究のアーキテクチャでは，スマートホームでの利用を想定していることから，過度の複雑性を回避し，性能や保守性への悪影響を防ぐ設計としている．すなわち，自己反映ないしは自己適応計算を一般的なモデルとして表現することを目的とするこれらのアーキテクチャを，コンテキストに応じた適応による再構成に特化して詳細化したものであると位置づけることができる．

また，自己適応システムの構成に関して，Sykes ら [20] は，ゴール管理，変更管理，コンポーネントの 3 レイヤ参照モデルに基づく概念構造を提案している．ゴール管理レイヤでは，アプリケーションの目的に応じたプランの自動生成を行い，変更管理レイヤでは，プランに従ってコンポーネントを自動構成する．これにより，状況やゴールの変化に応じた自己適応を可能としている．

Sykes らの提案する構造を本研究で想定する IoT アプリケーションの動的適応にあてはめると，ゴールは，洗濯機の完了通知を行うなどのアプリケーションの目的に相当し，プランは，終了通知を利用可能な IoT 機器に届けるための IoT 機器構成ポリシに相当する．ゴール管理レイヤでは，IoT 機器や環境の状況，すなわちコンテキストに応じて，洗濯機の完了を通知させるためのポリシを自動生成し，変更管理レイヤでは，ポリシを満たすように IoT 機器の再構成と適応を行う．これら自動構成の枠組みに時相論理による仕様記述が必要になる．

4 章で示したように，本研究では，IoT 機器の追加のたびにポリシやアダプタ構成に関する記述を追加しなければならない．現段階では，スマートホームにおける IoT 機器追加変更の頻度を勘案し，アプリケーションの目的を満たすポリシの自動生成と，ポリシを満たす IoT 機器構成の自動生成については検討していない．しかしながら，IoT 機器の追加や削除に対応し，より柔軟な適応を実現するためには，Sykes らが提案するような，ポリシと機器構成の自動生成の枠組みは重要であり，今後の課題として関連する成果を取り込んだ発展を検討する必要がある．

変換論理を実現するためのアーキテクチャ設計では，Adapter パターン，Layers パターン，Template Method パターンを組み合わせて適用した．この設計に対し，Broker パターン [3]（Mediator パターン [10]）を適用し，ブローカが IoT 機器間の通信を仲介する構造が考えられる．スマートホームに高性能のホームサーバが存在する環境では，ハードウェアやネットワーク構成を反映した素直なアーキテクチャ設計となりうる．一方で，個々の IoT 機器どうしの動的適応と変換の論理がブローカに集中する形態となることから，保守性を考慮したそれぞれの論理の分離は，アーキテクチャに従ってアプリケーションを実現する際の課題となる．Broker パターンの適用は，4 章に示したように，本研究で提案するアーキテクチャに従ってモジュール分割した結果から，アプリケーションの詳細設計を行う際に，実現のためのパターンとして想定するスマートホーム構成に応じて適用するのが妥当である．

5.2 柔軟性と相互運用性について

4 章で説明した簡単なメッセージ通知アプリケーションでは，利用者の状況（位置，時間）に応じた，メッセージ送信先のネットワーク接続（IP アドレス，ポート），メッセージングプロトコル（ソケット通信/HTTP），意味（テキストによる通知/ビープ音による通知）の変更を，アダプタモジュールのインスタンスの動的な生成と IoT 機器間接続の再構成によって実現することができた．すなわち，アーキテクチャに基づく 3 つのレベルの動的適応が可能であることが示された．また，IoT 機器の種類や機能の相違を吸収する論理とアプリーション論理とを個別のモジュールとして独立に開発できることも示すことができた．

ここで，ネットワーク接続レベル，メッセージングプロトコルレベルの動的適応のためには，各 IoT 機器が用いるネットワーク接続，メッセージングプロトコルは既知であり，適切なプログラミングインタフェースが提供されていることが前提である．未知のものやインタフェースが提供されない場合はアダプタも生成できないが，IoT の目的は多様な機器を連携させることであるから，この前提はさほど強いものではない．

一方で，意味レベルの適応のためには，IoT 機器間の協調処理やその中での各機器の役割について，何らかの類型化が必要である．IoT を構成する組込み機器の機能には単純なものが多いので，典型的な機能とその間の変換論理をライブラリの形で提供することで最低限の相互運用性を確

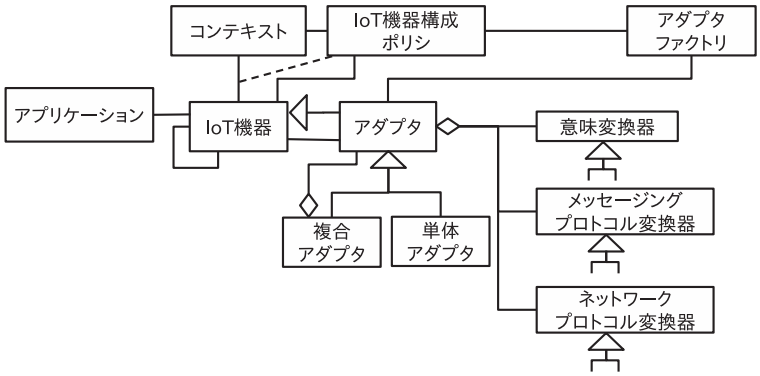


図 11 複合 IoT 機器の動的適応を対象とするアーキテクチャ
Fig. 11 Architecture for dynamic adaptation among composite IoT devices.

保することはさほど難しくないと考えている．より多様な処理機能までを対象に相互運用性を高めるためには，多鹿らが提案する情報家電システム [17] で行われているようなネットワーク家電機能の体系化が必要である．すなわち，情報の入出力，変換といった処理機能や，音声，文字，映像，画像といった情報の表現メディアを整理し，さらに，処理機能の互換性（たとえば文字通知をビープ音通知で代替することが可能など）についての知識体系が必要となる．その検討については今後の課題としたい．

図 5 のアーキテクチャ定義にあたっては，IoT 機器間のメッセージ通信や機能の変換が 1 対 1 で行われることを想定した．その一方で，複数の IoT 機器群を 1 つの多機能 IoT 機器ととらえ，それらとの相互運用を行うこともスマートホームアプリケーションに対する要求として想定できる．洗濯完了通知アプリケーションの例では，利用者の居場所の部屋に存在する IoT 機器群をひとまとまりの複合 IoT 機器ととらえ，利用者の状況に応じてそれを構成するすべての機器から完了通知メッセージを出力することや，完了通知メッセージを分割（たとえば，メッセージ着信の通知とメッセージの内容に分割する，など）し，それぞれを別の機器から出力することなどが考えられる．

特定のアーキテクチャに基づくアプリケーション開発では，通常，上述のような要求の進化に対応するために，複雑化した問題の構造を特定のモジュールで受け止める方法を取らざるをえない．すなわち，複合 IoT 機器の機能の分割や統合，さらには，構造を持つメッセージの分割や統合などを，1 カ所のアダプタに複雑な変換処理を集める形で実装する．結果として，アダプタ自身の開発コストや再利用性の面で問題が生じ，肝心の相互運用性にも悪影響を及ぼしかねない．

PBR パターンを適用したことで，提案アーキテクチャでは，ライブラリなどの形で提供される多相型のインスタンスとしての「アダプタ」を動的に選択して切り替えるだけでなく，複数のアダプタからなる複合構造を再構成することが可能となっている．つまり，上述のような問題構造に及ぶ要求の進化には，アダプタの構造を再構成して対応することができる．

利用者の状況などに応じて複合 IoT 機器を考慮した動的適応が必要となった場合には，図 4 に示した基本構造や図 5 のアーキテクチャと矛盾しない形で，複合 IoT 機器に対するアダプタを動的に構成することができる．つまり，通常の方法では 1 つのアダプタにまとめて実装される複雑な適応論理を，単体の IoT 機器のためのアダプタと複合 IoT 機器のためのアダプタに受け持たせて構造化することができる．このことは，提案アーキテクチャが，PBR パターンに基づく再構成の結果，**図 11** に示すような「複合アダプタ」と「単体アダプタ」からなるアーキテクチャへ，動的に進化できることを示している．

「複合アダプタ」を用いた動的適応時の振舞いを**図 12** に例示する．ここでは，3.3 節の図 6 に示した「洗濯機 2tv」の「アダプタ」を，居間に設置されている「TV」と「目覚まし時計」からなる「複合アダプタ」の構造に動的に切り替える様子を示している．図中のメッセージ番号は図 6 に倣って付されている．

ここでは，利用者の「コンテキスト」に応じ，洗濯完了通知を居間にある全 IoT 機器（この場合，「TV」と「目覚まし時計」）に行う必要が生じたとする．対応する「IoT 機器再構成ポリシ」に基づいて，居間全体の IoT 機器に対する「複合アダプタ」として「洗濯機 2 居間」を生成し，完了通知先をこれに切り替える（メッセージ 5）．「複合アダプタ」では「TV」および「目覚まし時計」に対する「単体アダプタ」とそれらを構成する変換器群を生成する（6）ことで，洗濯機からの完了通知（7）をそれぞれの機器に応じて変換し（8），通知先の IoT 機器に送信する（9）．

このように，「複合アダプタ」と「単体アダプタ」からなる構造を動的に構成することで，IoT 機器単体の適応のための論理と，それらを統合して適応させるための論理を明確に分離して実現することができる．

様々な規格にまたがる IoT の相互運用性を確保する仕組みを提供することを目的としたオープンソースソフト

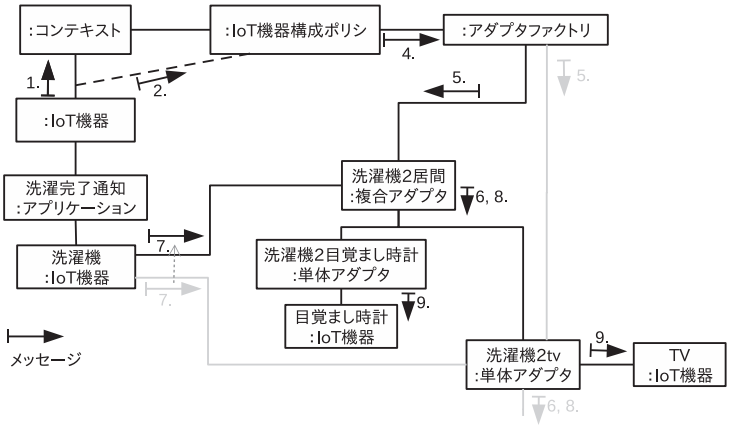


図 12 複合 IoT 機器に対する動的適応時の振舞い
Fig. 12 Behavior of dynamic adaptation for composite IoT devices.

ウェアプロジェクトとして Eclipse sensiNact [7] の活動がある．sensiNact Gateway を中心に．個々の IoT 機器がそれぞれのプロトコルでゲートウェイに接続するための下向き（southbound）ブリッジと，ゲートウェイが他の情報システムと接続するための上向き（northbound）ブリッジが配置されたアーキテクチャが定義されている．IoT 機器の相互運用性を実現するために，既存の各種プロトコルに対する下向きブリッジが用意されており，IoT 機器とゲートウェイとのやりとりは機器に適合したブリッジを組み合わせることで実現する．さらに，新しいプロトコルに対するブリッジを定義するためのガイドラインも提供されている．機器の制御や情報を可視化するアプリケーションは sensiNact Studio を用いて開発することができる．

sensiNact のゲートウェイアーキテクチャで定められているブリッジは本研究におけるアダプタに相当する．sensiNact において，ネットワークプロトコルとメッセージングプロトコルに対するアダプタは各種規格に対応するものが定義され，また拡張しやすい構造となっている．一方で，意味レベルの適応論理や，コンテキストに応じて機器構成を行うためのポリシについては，開発環境において記述する枠組みが提供されているものの，アーキテクチャ上必ずしも明確になっていない．また，複合的な構造を持つアダプタを用いた動的適応の扱いについても明らかに示されていない．これらの点で本研究とは異なっているが，両者は競合するものではない．補完することで，明確な関心事分離によるモジュール独立性の向上が期待できる．

5.3 保守性について

提案するアーキテクチャでは，利用者の状況に応じた再構成の論理と，再構成の際に必要となる適応の論理をモジュールとして分離している．柔軟な相互運用性のために動的適応を行う IoT アプリケーションの実装をこのアーキテクチャに基づいて行うことで，それぞれの論理の独立性が高まり，保守性の向上も期待できる．

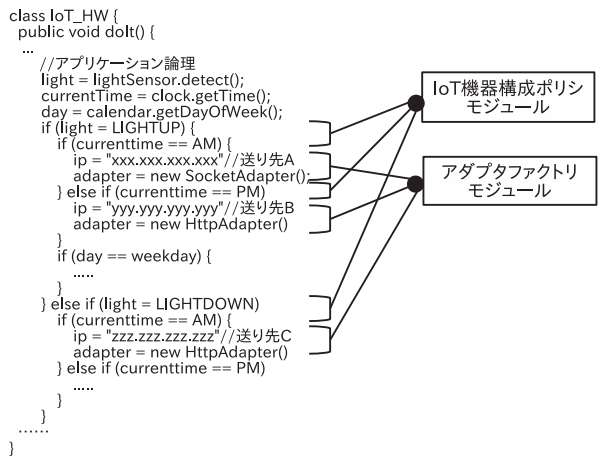


図 13 ソースコード例
Fig. 13 Source code fragments not based on the proposed architecture.

図 13 には，アーキテクチャに基づかない，すなわち，再構成の論理と適応の論理を 1 つのモジュールに混在させて記述した場合のソースコードとその中のコード片が提案アーキテクチャのどのモジュールに含まれるべきかの例を示す．ここでは，人感センサが検知する状況（`LIGHTUP`,`LIGHTDOWN` など）と，IP アドレスの付け替え，メッセージングプロトコルの変換アダプタを生成する処理などが，入れ子状の `if` 文により組み合わされている．これらの論理が関連する複数の IoT 機器に横断して実装されると，ソースコードの保守は困難になる．提案アーキテクチャにより関心事を分離したモジュール化の指針を与えることで，モジュールの独立性を高め，保守性の高いプログラムを構築することができる．

6. おわりに

本研究では，スマートホームなどで用いられる IoT アプリケーションにおける柔軟性と相互運用性の確保を可能とするソフトウェアアーキテクチャを提案した．アーキテクチャの設計にあたり，IoT 機器を相互に連携させるための

論理と，利用者の状況や嗜好に適用させるための論理とを明確に分離させながら，IoT 機器間の多層にわたる動的適応のための構造を定義した．このアーキテクチャにより，スマートホームにおける柔軟な相互運用を可能とする IoT アプリケーションの構築基盤を提供する．

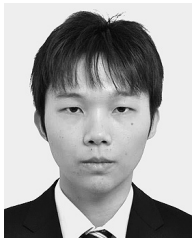
提案するアーキテクチャに基づいて，IoT 機器間の動的適応が可能であることを示すために，機器間の簡単なメッセージ通知アプリケーションを実装した．この実装を通じ，利用者の状況に応じてメッセージ送信先の IoT 機器を変更し，その機器に合わせた形態のメッセージを通知できることを確認した．また，利用者の状況に適応する論理と，異なるプロトコルを持つ機器へ適応する論理とを分離することで，IoT アプリケーションが初期的に想定していない方法でのメッセージ通知が可能であることも示した．

今後の課題としては，より複雑な事例に基づいて，アーキテクチャの適用可能性について検証を行うことがあげられる．特に，メッセージングプロトコルレベルと機能レベルのアダプタについては，より現実的で複雑なメッセージと機能変換について検討する必要がある．また，リアルタイム性やメモリ消費などの性能効率性に関して，このアーキテクチャに基づいて実装することでどの程度の影響があるかを評価することも今後の課題である．

謝辞 本研究の一部は，JSPS 科研費（基盤研究（C）19K11911，20K11759），および 2020 年度南山大学パッヘ研究奨励金 I-A-2 の助成による．

参考文献

[1] Bass, L., Clements, P. and Kazman, R.: *Software Architecture in Practice, Third Edition*, Addison-Wesley Professional (2012).
[2] Bassi, A., Bauer, M., Fiedler, M., Kramp, T., van Kranenburg, R., Lange, S. and Meissner, S. (Eds.): *Enabling Things to Talk — Designing IoT Solutions with the IoT Architectural Reference Model*, Springer Berlin Heidelberg (2013).
[3] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. and Stal, M.: *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*, Wiley Publishing (1996).
[4] Datta, S.K. and Bonnet, C.: Smart M2M Gateway Based Architecture for M2M Device and Endpoint Management, *Proc. 2014 IEEE International Conference on Internet of Things* (*iThings*), *and IEEE Green Computing and Communications* (*GreenCom*) *and IEEE Cyber, Physical and Social Computing* (*CPSCom*), pp.61–68, IEEE Computer Society (2014).
[5] Datta, S.K., Bonnet, C. and Nikaein, N.: An IoT Gateway Centric Architecture to Provide Novel M2M Services, *Proc. 2014 IEEE World Forum on Internet of Things* (*WF-IoT*), pp.514–519, IEEE (2014).
[6] Desai, P., Sheth, A.P. and Anantharam, P.: Semantic Gateway as a Service Architecture for IoT Interoperability, *Proc. 2015 IEEE International Conference on Mobile Services*, pp.313–319, IEEE Computer Society (2015).
[7] Eclipse Foundation: *SensiNact*, available from ⟨https://wiki.eclipse.org/SensiNact⟩ (accessed 2021-01-20).
[8] エコーネットコンソーシアム：ECHONET Lite 規格の特徴と概要，入手先 ⟨https://echonet.jp/about/features/⟩（参照 2021-01-20）．
[9] 江坂篤侍，野呂昌満，沢田篤史：インタラクティブシステムのための共通アーキテクチャの設計，コンピュータソフトウェア，Vol.35, No.4, pp.3–15 (2018).
[10] Gamma, E., Helm, R., Johnson, R. and Vlissides, J.M.: *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Longman (1994).
[11] Garlan, D., Cheng, S.-W., Huang, A.-C., Schmerl, B. and Steenkiste, P.: Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure, *Computer*, Vol.53, No.10, pp.46–54, IEEE Computer Society (2004).
[12] 井垣 宏，中村匡秀，玉田春昭，松本健一：サービス指向アーキテクチャを用いたネットワーク家電連携サービスの開発，情報処理学会論文誌，Vol.46, No.2, pp.314–326 (2005).
[13] ISO/IEC 25010: *Systems and Software Engineering — Systems and Software Quality Requirements and Evaluation* (*SQuaRE*) *— System and Software Quality Models* (2011).
[14] Jabbar, S., Ullah, F., Khalid, S., Khan, M. and Han, K.: Semantic Interoperability in Heterogeneous IoT Infrastructure for Healthcare, *Wireless Communications and Mobile Computing*, Vol.2017, Article ID 9731806 (online), DOI: 10.1155/2017/9731806 (2017).
[15] oneM2M Partners: *oneM2M: Standards for M2M and the Internet of Things*, available from ⟨http://www.onem2m.org⟩ (accessed 2021-01-20).
[16] oneM2M Partners: *Ontologies Used for oneM2M*, available from ⟨http://www.onem2m.org/technical/onem2m-ontologies⟩ (accessed 2021-01-20).
[17] 多鹿陽介，安次富大介，中村素典，美濃道彦，釜江尚彦：ホームネットワークに適した単機能分散型ネットワークドアプライアンスアーキテクチャ，情報処理学会論文誌，Vol.44, No.9, pp.2320–2333 (2003).
[18] W3C: *Web of Things at W3C*, available from ⟨https://www.w3.org/WoT/⟩ (accessed 2021-01-20).
[19] W3C: *Semantic Sensor Network Ontology*, available from ⟨https://www.w3.org/TR/vocab-ssn/⟩ (accessed 2021-01-20).
[20] Sykes, D., Heaven, W., Magee, J. and Kramer, J.: From Goals To Components: A Combined Approach To Self-Management, *Proc. 2008 ICSE Workshop on Software Engineering for Adaptive Self-Managing Systems* (*SEAMS*) (2008).
[21] 山田知秀，飯島 正，山口高平：オントロジーを利用した情報家電エージェント協調アーキテクチャ，第 19 回人工知能学会全国大会，1B2-03 (2005).



横山 史明

2019 年南山大学理工学部ソフトウェア工学科卒業．現在，南山大学大学院理工学研究科ソフトウェア工学専攻博士前期課程在学中．組込みソフトウェアの開発支援に興味を持つ．

沢田 篤史 （正会員）

1990 年京都大学工学部情報工学科卒業．1995 年同大学大学院工学研究科情報工学専攻博士後期課程単位取得退学．1995 年奈良先端科学技術大学院大学情報科学研究科助手．現在，南山大学理工学部ソフトウェア工学科教授．この間，2011～2012 年までスイスチューリヒ大学客員教授．博士（工学）．ソフトウェア工学，組込みシステム，ソフトウェア開発支援環境等の研究に従事．日本ソフトウェア科学会，電子情報通信学会，システム制御情報学会，IEEE Computer Society，ACM 各会員．

野呂 昌満 （正会員）

1981 年慶應義塾大学工学部管理工学科卒業．1986 年同大学大学院工学研究科管理工学専攻博士後期課程単位取得退学．1986 年南山大学経営学部情報管理学科講師．現在，同大学理工学部ソフトウェア工学科教授．この間，1988～1990 年まで米国メリーランド大学計算機科学科客員研究員．工学博士．ソフトウェアアーキテクチャ，アスペクト指向計算，プログラミング言語の意味論および処理系等の研究に興味を持つ．日本ソフトウェア科学会，電子情報通信学会，IEEE Computer Society，ACM 各会員．

江坂 篤侍 （正会員）

2011 年南山大学数理情報学部情報通信学科卒業．2018 年同大学大学院数理情報研究科数理情報専攻博士後期課程修了．2017 年より同大学理工学部ソフトウェア工学科助教，現在に至る．博士（数理情報学）．ソフトウェアアーキテクチャ，アスペクト指向計算等の研究に従事．日本ソフトウェア科学会会員．