IoT アプリケーションにおけるコンポーネントの動的再構成の仕組みの適用についての考察

2013SE093 小嶋拓樹　2013SE119 宮田一朗　2013SE255 山崎智輝
指導教員：野呂昌満

1　はじめに

ICT 技術の発展に伴い，IoT アプリケーションは身近なものとなっている．IoT アプリケーションにおいて，システムを取り巻く外部環境はデバイスが移動することにより変化する．

事前に定義されたコンポーネントを静的に組み合わせる場合，さまざまな状況に対応することや，実行時にプログラムを組み換え，振る舞いを変更することは困難である．全ての適切な振る舞いをコンポーネントとして事前に定義し，それらの組み合わせを動的に変更できるようにすることが求められる．

本研究の目的は，コンポーネントの動的再構成の仕組みを考案することである．動的再構成を可能にすることにより，アプリケーションの動きを変更させることを可能にする．動的再構成の実現可能性を確認する方法として，デザインパターンを用いる．デザインパターンを用いた動的再構成の仕組みを考案することにより，外部環境の変化に柔軟に対応することができると考えた．

既存の IoT アーキテクチャを定義し，スマートデバイスの視点から改版を行なう．改版したアーキテクチャからコンポーネントの動的再構成可能な箇所を特定する．動的再構成可能なコンポーネントを特定し，動的再構成の仕組みを考案する．提案する仕組みは外部環境の変化に応じて再構成することを可能にする．一般に，外部環境の変化に応じて再構成を行なう仕組みはコンテキスト指向技術で表現することができる．コンテキスト指向技術を用いて，モデル化を行なう．実現可能性を確認する方法として，デザインパターンを用いて試作を行ない，動的再構成の妥当性を確認する．

全ての適切な振る舞いをコンポーネントとして事前に定義し，それらの組み合わせを動的に変更できるようにすることにより，アプリケーションの静的領域を節約することができる．また，コンポーネントの組み合わせによってアプリケーションの動きを一部変更することができる．

2　関連研究

2.1　コンテキスト指向技術

コンテキスト指向技術とは，プログラムの実行時に，システムを取り巻く外部環境の状況すなわちコンテキストに応じて，実行時にソフトウェアを再構築し，振る舞いを変化させることが可能である [3].

3　動的再構成を可能にした仕組み

既存の IoT アーキテクチャを定義し，スマートデバイスにおける視点から改版を行なう．改版したアーキテクチャ上で，動的再構成が実現可能な箇所を特定する．動的再構成の仕組みをコンテキスト指向技術を用いて定義する．

3.1　IoT アーキテクチャと動的な再配置が可能な箇所

IoT はインターネットを通じて，Sensor や Actuator，Application を連携する [6]．図 1 は，IoT におけるアーキテクチャである．不特定多数の Cloud や Service が Application を利用する．Gateway はインターネットに接続することができない Sensor を繋げる．Sensor が移動体を検知し，Application や Gateway に繋がる．図 1 の点線は Sensor が Gateway に繋がるものだけではなく，移動体を検知することにより，Application に繋がるものもあることを示す．

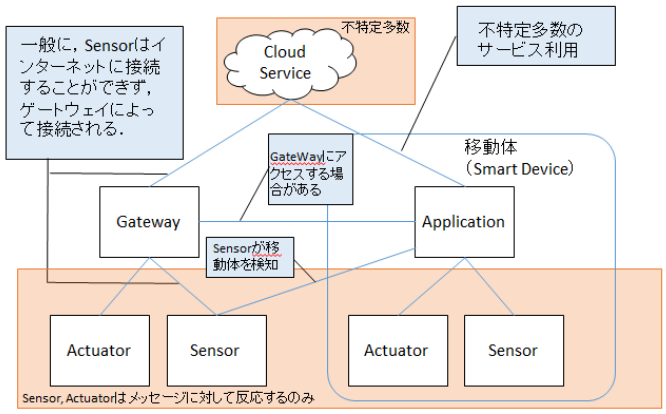

図 1　IoT アーキテクチャ

Cloud や Service は自ら書き換えを行なうことができないことから，動的再構成の実現が困難だと考えた．我々は，Gateway と Application にコンポーネントの動的な再配置が可能であると考えた (図 2).

本研究ではメッセージの暗号化を事例とする．動的再構成の実現可能な箇所にコンテキスト指向技術を導入する．Gateway から送られるメッセージをコンテキストとし，コンテキストに応じて暗号化を行なうかを Layer Activator が検知する．Layer Activator は暗号化が必要な場合には，処理を行なう Layer を活性化し実現する．

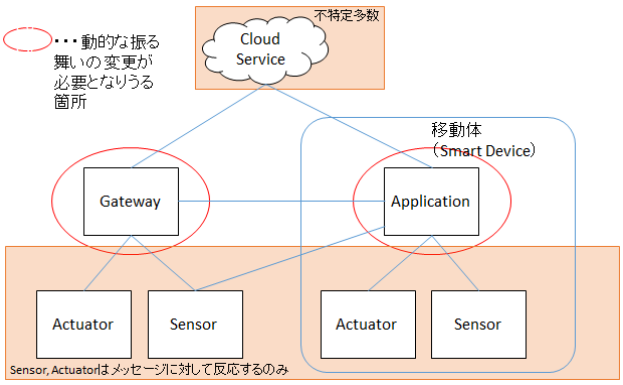

図 2　動的再構成が可能な箇所

3.2　動的再構成の仕組み

動的再構成の仕組みの概要を説明する．コンテキスト指向技術を導入し，事前に Context と Layer を定義する．事前に Layer を定義することにより，Context に応じた組合せを実現することができる．Layer Activator は必要となる Layer を活性化させる役割を持つ．活性化された Layer を Object に織り込むことで動的再構成を行なう (図 3)．

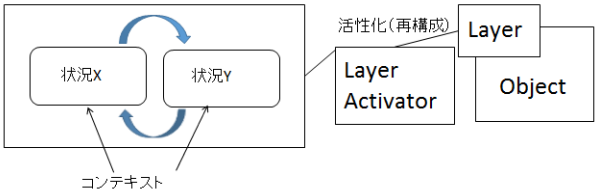

図 3　動的再構成の仕組み

それぞれのコンポーネントの役割を記述する．

• Context

Gateway から Application に送られるメッセージ．

• Layer Activator

特定の状況になったことをきっかけとし，対応する Layer を活性化し Object に織り込む．

(Aspect Weaver の役割を担う)

• Layer

メッセージに対応した適切な処理を事前に定義する．

4　デザインパターンを用いた動的再構成の仕組み

本研究では，言語に依存しないコンポーネントの動的再構成を可能にすることを目的とし，デザインパターンを用いる．デザインパターンを用いることにより，インスタンスを生成し，動的にコンポーネントの組合せの変更が実現可能であると考察する．オブジェクト指向では，メッセージ記述を変化させることはできないので，コンテキスト指向を用いる．メッセージ記述を変化させることを目的とし，Hook Operation パターンを用いる．操作の前後に Hook をつけておくことにより，前後の処理を変更することができ，メッセージに対しての処理を動的に変更することができる．クラスメソッドを表現するために，Prototype パターンを用いる．Prototype パターンは，メタ情報の追加や削除を行なうことができるので，動的な変更を柔軟に行なうことができる．メタ情報には，振る舞いに関するパターンである State パターンや Strategy パターン，Command パターンを定義する (図 4)．この Layer 構造を基に，動的再構成を行なう．

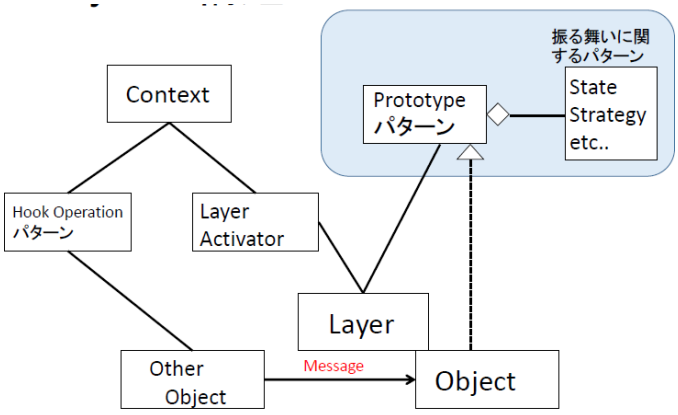

図 4　Layer の構造

次のデザインパターンを用いることで，メタオブジェクトへの操作およびそのメッセージング記述の分離を表現する．

●Prototype パターン

クラスメソッドをインスタンス側に定義するためのパターン．

●振る舞いに関するパターン

クラスメソッドにより，振る舞いに関するパターンを変更する．オブジェクトはパターンに基づき動作することで動的再構成を表現する．

-State パターン

状態をクラスとして表現することにより，状態の変化に応じて振る舞いが変化する場合に用いる．

-Strategy パターン

アルゴリズムを定義し，使用するクライアントとは独立して，アルゴリズムの変更を可能にする．

-Command パターン

要求自体をオブジェクトとすることにより，仕様化されていないオブジェクトの要求を作成できるようにする場合に用いる．

-Template Method パターン

クラスに定義されるインスタンス構造のうち，変更可能な部分を Template パターンとして定義しておくことにより，コンポーネントの動的再配置を可能にする．

• Hook Operation パターン

メッセージを送る側に対し，操作の前後にフックをつけておくことにより，前後の処理を変更できる．

5　事例

IoT アプリケーションの一例である対話型アプリケーションにおいて，動的再構成が必要となる場面を挙げる．本研究では，メッセージの暗号化を考えたさいの動的再構成の仕組みを考案する．メッセージの暗号化を考えた場合，静的に組み込むとメッセージに対応するコンポーネントを複数用意しなければならない．動的にコンポーネントの組み合わせを変更することにより，特定のメッセージに対してのインスタンスを変更することにより対応することができる．

Command パターンを用いてメッセージをオブジェクトとして取り扱い，メッセージの詳細をカプセル化する．コンテキスト指向技術を用いて特定の状況になったことをきっかけとし，暗号化処理を変更する．

5.1　メッセージの暗号化の処理

異なるデバイスとの協調により，コンテキストが変化し適した暗号化方式の Layer が活性化される．コンテキストは暗号化方式を決定づける外部環境の情報である．

本研究の事例では，どの Fog に接続しているかにより振る舞いを決定させる．コンテキストの変化により，特定のきっかけになったことを Layer Activator が検知し，Layer を活性化させる．活性化する Layer は暗号化方式を定義したものである．Layer は振る舞いが変化する対象のオブジェクトの prototype パターンで定義されたクラスメソッドに対しメッセージを送る．メッセージに応じてどの暗号化処理を行なうかのインスタンスを変更する．

図 5 はメッセージの暗号化の処理を表したものである．Encryption 1，Encryption2 は Command パターンを用いて表したものである．Command パターンを用いることにより，メッセージをオブジェクトとして扱え，詳細をカプセル化することができる．これにより，インスタンスの変更を用意に行なうことが可能になる．

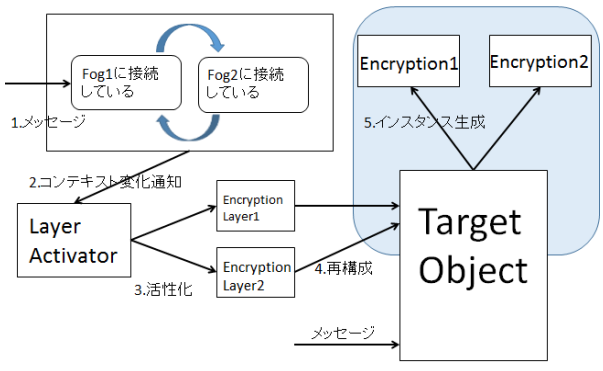

図 5　メッセージの暗号化を考えた場合の処理の流れ

5.2　メッセージの暗号化を考える場合の静的構造

メッセージの暗号化を事例として動的再構成を行なうさい，Prototype パターン，Command パターンと Hook Operation パターンを用いてクラス図を記述した．メッセージが送られてきた場合，特定のメッセージに対して，インスタンスの変更を行ない，メッセージを暗号化する Layer を活性化する．活性化された Layer を動的に織り込む．図 6 にメッセージの暗号化を考えた場合のクラス図を示す．

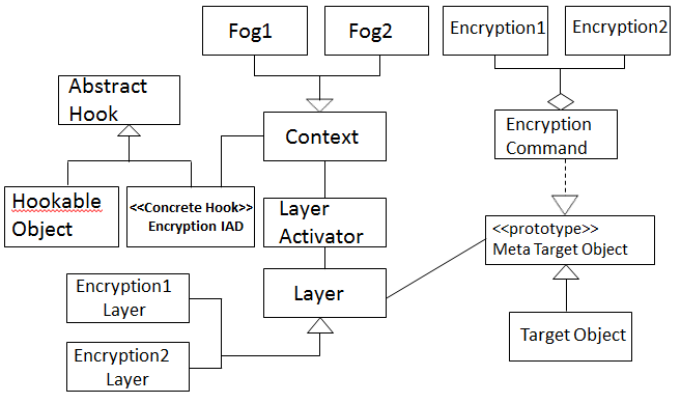

図 6　メッセージの暗号化を考える場合

それぞれのコンポーネントの役割を記述する．

- Abstract Hook
  preHook のインタフェースの共有を行なうもの．
- Hookable Object
  メッセージを受けたことを起因とし，Concrete Hook に preHook で処理を行なう．
- Concrete Hook
  送られてきた preHook の処理の実装を行なうもの．
- Abstract Hook
  preHook のインタフェースの共有を行なう．
- Context
  Fog から送られるメッセージ．
- Layer Activator
  Context の変化に起因し，再構成させる．
- Layer
  Context に応じた処理が記述されたもの．
- Encryption Layer
  複数ある暗号化処理を定義したもの．
- Meta Target Object
  クラスメソッドを表現することを目的に prototype パターンを用いて定義されたもの．
- Target Object
  メッセージの振り分けを行なうもの．
- Encryption Command
  振る舞いに関するパターンを記述したもの．Command パターン用いてメッセージをオブジェクトとする．
- Encryption
  暗号化手続きを記述したもの．

6　提案する動的再構成の仕組みの妥当性を確認

コンポーネントの動的再構成が可能である場面に対し，我々の提案する仕組みを用いて対応する．動的再構成を行なう仕組みをコンテキスト指向技術を用いて場合分けを行なう．実現可能性を確認する方法としてデザインパターンを用いて定義した．

メッセージの暗号化を考えた場合，静的に組み込むと外部から送られてくるメッセージに対して複数のコンポーネントを定義しておかなければならない．動的に再構成を行なうことにより，特定のメッセージに対し，インスタンスを変更を行なうことで対応することが可能であると考えた．

Layer の動的再構成が可能であるパターンを定義することができれば，さまざまな状況に対して，事前に定義したコンポーネントを組み合わせ再構成することができる．それにより，アプリケーションの静的領域の節約や動作の一部を変更することが可能になる．

7　おわりに

ICT 技術の発展に伴い，IoT アプリケーションは身近なものとなっている．IoT アプリケーションにおいて，システムを取り巻く外部環境はデバイスが移動することにより変化する．

事前に定義されたコンポーネントを静的に組み合わせる場合，さまざまな状況に対応することは困難である．外部環境が常に変換する状況にあることからコンポーネントを動的に再構成できるようにすることが求められる．

既存の IoT アーキテクチャを定義し，スマートデバイスの視点から改版を行なった．改版したアーキテクチャからコンポーネントの動的再構成可能な箇所を特定した．動的再構成可能なコンポーネントを特定し，動的再構成の仕組みを考案を行なった．一般に，外部環境の変化に応じて再構成の仕組みを行なう仕組みはコンテキスト指向技術で表現することができる．コンテキスト指向技術を用いて，モデル化を行なった．実現可能性を確認する方法として，デザインパターンを用いて試作を行ない，動的再構成の仕組みの有用性について考察した．

全ての適切な振る舞いをコンポーネントとして事前に定義し，それらの組み合わせを動的に変更できるようにすることにより，アプリケーションの静的領域を節約することができる．また，コンポーネントの組み合わせによってアプリケーションの動きを一部変更することができる．

今後の課題として，実現可能性を確認した動的再構成の仕組みをデザインパターンを用いての実現を行なう．振る舞いに関するパターンを洗練し，より外部環境の変化に対応することができるよう洗練を行なう．

参考文献

[1] Bass, L. ：Software architecture in practice, AddisonWesley，2007

[2] Erich Gamna, Richard Helm, Ralph Johnson, John Vlissides，オブジェクト指向における再利用のためのデザインパターン 改訂版，ソフトバンククリエイティブ，1999

[3] Hirschfeld, R., Costanza, P. and Nierstrasz, O. “ Context-oriented programming ”, Journal of Object technology，Vol. 7，No. 3，2008

[4] Vilet,H.V.：Software engineering: principles and practice,Wiley，2007.

[5] 江坂篤侍，野呂昌満，沢田篤史, “ インタラクティブソフトウェアの共通アーキテクチャの提案 ”，情報処理学会研究報告，ソフトウェア工学報告，vol.2015-SE-187,no.32，pp.1-8，2015-03-05.

[6] 江坂篤侍，野呂昌満，沢田篤史, “ コンテキストアウェアネスを考慮した組み込みシステムのためのアスペクト指向アーキテクチャの適用と実現 ”