C-040

AOPを用いたOSGi Frameworkのバンドル開発手法の検討

A Consideration on Bundle Development Method for OSGi Framework Using AOP

福谷 治 †　小玉 哲平 ‡　小板 隆浩 †　佐藤 健哉 ‡
Osamu Fukutani　Teppei Kodama　Takahiro Koita　Kenya Sato

1 はじめに

近年，PC や携帯電話を始めとし，AV 機器や家電機器などの様々な機器がネットワークに接続されるようになり，お互いの機能を利用しあう機器連携が行われるようになった．そして，機器連携を実現するためのミドルウェアとして OSGi Framework[1] がある．

OSGi Framework は，Bundle というソフトウェアモジュールを起動することで，他の機器を動作させ，機器連携を行う．また，Bundle は単独で動作することは少なく，Bundle 同士が連携してサービスを実現している．しかし，Bundle 連携を行うには Bundle の実装時に利用する他の Bundle を決めておく必要があり，サービスを利用しあう Bundle 間の依存性が高い．ここでは，ある Bundle があるサービスを実現するために，他の Bundle のサービスを必要とし，呼び出しやサービスの利用を行う時，依存性があるとする．また，Bundle 間でデータの要求や提供をする数が多いほど，片方の Bundle でデータが追加，変更された際に，もう片方の Bundle でもデータの追加，変更をする箇所が増えるので，依存性が高いとする．一方，ソフトウェアにおけるオブジェクト間の依存性を軽減させる手法として AOP(Aspect Oriented Programming) がある．本稿では AOP を OSGi Framework の Bundle 開発に用い，Bundle 間の依存性を軽減させる手法について検討を行い，開発コストの削減を目指す．

2 OSGi Framework

OSGi Framewoek は，UPnP[2] や Jini[3] と言った異なる技術仕様の差異を吸収し機器連携を行う OSGi(Open Service Gateway initiative) が標準化する SOA(Service Oriented Architecture) ベースのミドルウェアで，ネットワークを介してダウンロードされる JavaVM 上で稼動する Bundle というソフトウェアモジュールを管理する．OSGi Framework では，Bundle の提供するサービスを OSGi サービスレジストリとして登録，リスト化し，サービスの管理を行い，Bundle を操作し，組み合わせることで，各機器に合わせた様々なソフトウェアを動作させることができる．ここで複数の Bundle がお互いのサービスを利用しあうことを Bundle 連携と言う．

しかし，従来の OSGi Framework における Bundle 連携では実装時に他のどの Bundle を利用するかが決められており，また各 Bundle は単独で動作することは少ない．また，他の Bundle と組合さなければ動作が出来ない場合が多いため，各 Bundle と他の Bundle は何度もデータのやりとりをすることになり，依存性が高くなる．依存性が高い場合，サービスを利用する側の Bundle が変更された際に，サービスを提供する側の Bundle も変更しなければ動作しないことが問題となる．また，その際の Bundle のソースコードの変更コストが大きいことも問題となる．そこで，AOP を用いて依存性を軽減し，ある Bundle が変更されても他の Bundle の変更が最低限で済み，開発コストが削減された OSGi Framework を開発する．

3 AOP

AOP は，アスペクトと呼ばれる独自のクラスを使用し，オブジェクト指向プログラミングではプログラム上に散在していた横断的関心事と言う似た様な処理をアスペクトとして分離する機能を実現したプログラミングである．アスペクトはポイントカットとアドバイスと言うコードから成り立っており，ポイントカットがいつ処理を実行するか，アドバイスがどの様な処理を実行するかを表現している．アスペクトにより，オブジェクト指向プログラミングにおける処理の呼び出しコードを明示的に記述する必要がなくなり，クラス間の依存性が軽減したプログラミングが可能となる．AOP では，アスペクトを適用することを Weave と言う．

従来のオブジェクト指向プログラミングでは，各機能を呼び出すコードを各アプリケーション内に記述する必要があったため，呼び出す箇所が多ければ多いほどコード量が増加し，開発コストが高くなると言う問題点があった．しかし，AOP では Weave によって機能の呼び出しコードをアプリケーション内に記述する必要がなくなり，機能の追加や変更が行われても，そのコードのみを追加，変更するだけで済み，全体のコード量も大幅に削減できるので，開発効率も向上する．また，各クラスの本来の責務以外の処理を記述する必要がなくなったので，プログラム全体の見通しもよくなる．

4 提案手法

本稿では，OSGi Framework における Bundle を AOP を用いて開発する手法を提案する．AOP を OSGi Framework に適用することで，従来 OSGi Framework では Bundle 開発時にサービスを利用する側で明示的に記述していた機能呼び出しのソースコードを書く必要がなくなる．また，サービスを提供する側の Bundle にアスペクトを実装し，サービスを利用する側の Bundle を実行中の特定のタイミングで Weave することで，サービスを利用する側の Bundle はサービスを提供する側の

† 同志社大学理工学部情報システムデザイン学科
‡ 同志社大学大学院工学研究科情報工学専攻

Bundle を意識することなくサービスを利用することができ，Bundle 間の依存性を軽減させて動作することができる．

従来の OSGi Framework では Bundle A が Bundle B を利用する場合，図 1 の様な動作をしている．

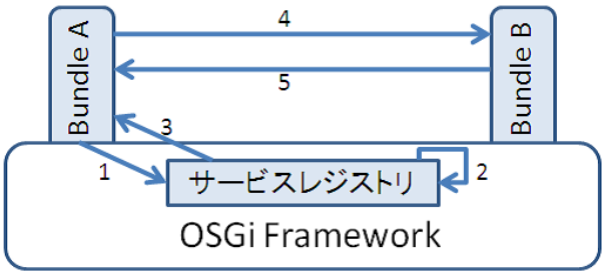

図 1　従来の OSGi Framework における Bundle の動作

1. Bundle A は OSGi Framework 上のサービスレジストリに利用したいサービスがないか OSGi Framework の API を用いて問い合わせる．
2. サービスレジストリには，その OSGi Framework が起動している全ての Bundle が所持しているサービスが登録されており，要求されたサービスが OSGi Framework の API により検索される．
3. サービスレジストリは要求されたサービスを Bundle B が所持しているというデータを Bundle A に返す．
4. Bundle A はサービスレジストリから返されたデータにより，Bundle B にアクセスしてサービスを要求する．
5. Bundle B は要求されたサービスを Bundle A に返す．

このように，従来の OSGi Framework では Bundle A は Bundle B にサービスを要求し，返されるという二つの動作が要因となり，Bundle B が変更されると，Bundle A でも変更する箇所が多く，高い依存性を持っていた．しかし，AOP を用いて，Bundle B を開発する事で依存性を軽減できる．Bundle A が実行中に Bundle B のサービスが必要になったタイミングをポイントカット，Bundle A に要求された Bundle B のサービスを返す処理をアドバイスとして，Bundle B で提供するサービスを AOP を用いて実装する．そして，Bundle B のサービスをアスペクトとして分離することで図 2 の様な動作になる．

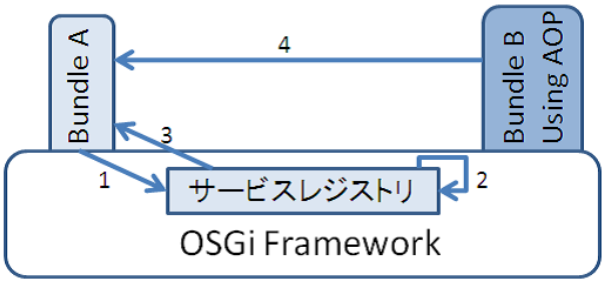

図 2　Bundle B を AOP で実装した OSGi Framework における Bundle の動作

1. Bundle A は OSGi Framework 上のサービスレジストリに利用したいサービスがないか OSGi Framework の API を用いて問い合わせる．
2. サービスレジストリには，その OSGi Framework が起動している全ての Bundle が所持しているサービスが登録されており，要求されたサービスが OSGi Framework の API により検索される．
3. サービスレジストリは要求されたサービスを Bundle B が所持しているというデータを Bundle A に返す．
4. Bundle A が実行中に Bundle B のサービスが必要になったタイミングをポイントカット，Bundle A に要求された Bundle B のサービスを返す処理をアドバイスとして実装したアスペクトを Bundle B は Weave する．

このように，Bundle B を AOP を用いて実装することで，サービスを利用する Bundle A はサービスを提供する Bundle B にサービスを要求することなくサービスを利用することができる．これにより，従来の OSGi Framework と比較して Bundle A から Bundle B へのサービスの要求という動作を省くことができる．従って，Bundle A から Bundle B に対する依存性の要因である二つの動作の内一つを省くことができたため，依存性も軽減できる．依存性の軽減により，Bundle A は Bundle B を意識せず利用できるようになったので，Bundle B のサービスが追加，変更された場合でも，Bundle A ではプログラムの追加，変更が最低限で動作させることができ，開発時におけるコスト削減ができる．削減できる開発コストはアスペクトとして実装し，分離する Bundle のサービスの利用，更新頻度にもよるが，頻繁に利用，更新されるサービスをアスペクトとして分離することで，従来と比較してより少ないコードの変更で開発を行うことができ，より大きな開発コストの削減効果が得られる．

5　まとめ

本稿では OSGi Framework の Bundle 開発に AOP を用いる手法について検討した．従来の OSGi Framework では Bundle 開発の際に，サービスを提供する側の Bundle は実装時に決められている必要があり，単独で動作をすることが少ない各 Bundle 同士は依存性が高く，Bundle の追加や変更があった際に，上手く動作しなくなる問題があった．そこで AOP を用いて OSGi Framework の Bundle を開発し，Bundle 内のサービスをアスペクトとして分離することで依存性を軽減させ，サービスを利用する Bundle はサービスを提供する Bundle を意識することなくサービスを利用し，Bundle 同士を連携させることができる．また，Bundle の追加，変更があった場合，他の Bundle のソースコードを変更する必要性がなくなり，開発コストも削減できる．

参考文献

[1] OSGi Service Platform Core Specification,Release4,OSGi Alliance,2005.
[2] Universal Plug and Play device Architecture Version 1.0,2000. http://www.upnp.org/resources/documents/CleanUPnPDA101_20031202s.pdf.
[3] Jini Network Technology,Sun Microsystems. http://www.sun.com/software/jini/.