OSGi に基づく拡張可能ソフトウェアの動作情報取得手法

清崎大輔[†]　大久保弘崇[††]
粕谷英人[††]　山本晋一郎[††]

Eclipse や Firefox に代表される拡張可能ソフトウェアの役割が重要になりつつあるが，ソフトウェアが大規模かつ複雑になり，プラグイン開発は容易ではない．例えば，Eclipse には標準で 169 もの拡張ポイントと，89 のプラグイン jar ファイルが存在する．また，プラグインの依存関係も解決しなければならない．本研究の目標は，拡張可能ソフトウェアのプラグイン開発を，ソフトウェア理解の観点から支援することにある．最終的には，Eclipse の内部動作を可視化して提示することで理解を支援しプラグイン作成を容易にすることを目指すが，その一歩として，Eclipse の動作情報を取得する方法について報告する．

Extract execution information for the extensible software based on OSGi

Daisuke KIYOSAKI,[†] Hirotaka OHKUBO,[††]
Hideto KASUYA[††] and Shinichiro YAMAMOTO[††]

The extensible software such as Eclipse and Firefox has been increasingly worthful. However because scale of software becomes larger, it is difficult to develop the plugins. For example, there exists 169 extension points and 89 plugin-jar files in Eclipse. What is worse, resolving dependencies of plugin is required. Our purpose is to support plugin development for extensible software in terms of software understanding, in concretely to visualize Eclipse's inner execution. At first step, this paper proposes a method to extract Eclipse's execution information.

1. はじめに

1.1 背　景

近年，ソフトウェアのオープンソース化が積極的に行われている．数々のソフトウェアがオープンソースとして公開されることで，ソフトウェアの開発者は有益なソフトウェアを自由に利用・改良することができるようになった．こうしたオープンソースソフトウェアの中でも特に注目を集めているのが，機能の拡張をサポートした拡張可能ソフトウェアである．

拡張可能ソフトウェアとは，プラグインのような拡張機能モジュールを取り込むことで機能の拡張や追加が行えるソフトウェアのことである．そのプラグインの開発には，ベースとなる拡張可能ソフトウェアの理解が必要である．しかし，ソフトウェア自体が大規模になってきていることに加え，機能拡張のための仕組み自体も確実に大規模化してきていることから，そのプラグイン開発は困難となってきている．

1.2 目　的

本研究が掲げる目標は，拡張可能ソフトウェアの動作や機能拡張の仕組みを可視化して提示することによって，プラグイン開発をソフトウェア理解の観点から支援することである．その初期段階として，統合開発環境 Eclipse[1)] を対象とした理解支援に活用可能な動作情報の取得方法について報告する．

2. Eclipse とプラグイン開発

2.1 Eclipse

Eclipse は，オープンソースの統合開発環境であり，拡張可能ソフトウェアの代表例である．機能の拡張や追加をサポートし，プラグインプラットホームを提供している．そして，プラットホームとプラグインモジュールとのインタフェースとして，**拡張ポイント**が用意されている．この拡張ポイントは，Java 言語におけるインタフェースとして実装されており，拡張機能を実装したプラグインは，Eclipse から拡張ポイン

† 愛知県立大学大学院 情報科学研究科
Graduate School of Information Science and Technology, Aichi Prefectural University
†† 愛知県立大学 情報科学部
Faculty of Information Science and Technology, Aichi Prefectural University

トを経由して実行される．

このように，Eclipse 利用者は拡張したいポイントに対し，実装したプラグインを登録することで機能を自由に追加することができる．

2.2 プラグイン開発の現状

Eclipse におけるプラグイン開発は，使用する拡張ポイントを選択するところから始める．しかし，Eclipse に標準で用意されている拡張ポイントだけでも 169 存在する．これらの拡張ポイントの役割を調べ，実装したい機能に必要な拡張ポイントを選択するという作業には，とても大きなコストがかかる．

次に，使用すべき拡張ポイントを決定した後，実際にプログラムを記述することになる．拡張ポイントには実装が義務づけられたインタフェースがあるため，開発者の関心事はその実装方法や機能全体におけるインタフェースの役割になる．しかし，プラグイン開発に関する資料も拡張ポイントにより量や質に大きな差があり，プラグイン開発のノウハウ習得に大きなコストを要する．

これらのコストは，Eclipse を利用中に直前に使用された拡張ポイントやクラスを知ることができれば軽減できる．そこで，本研究では実際に対象ソフトウェアを動作させながら動作を追跡することを考える．その実現のために，Eclipse 実行時の動作情報を追跡する仕組みを新たに提案する．

3. 動作情報とその取得

3.1 動 作 情 報

本研究での動作情報とは，次の 4 つイベントとそれに付随する情報を指す．

- メソッド呼び出し
  所属クラス，ソースファイル名，引数，可視性
- オブジェクト生成
- スレッドの実行
  スレッド名，実現クラス，実行開始，実行終了
- ユーザーイベント

実際に Eclipse の機能を使用する際に実行される，メソッドやスレッドに関する情報をはじめ，機能実行のトリガとなる利用者の操作についての情報を扱う．これらの情報はプラグイン開発固有の情報ではないが，拡張ポイントに関する情報の取得・提示において必要である．したがって，動作情報は可能な限り取得し，必要に応じて後からフィルタ処理を行う．

3.2 動作情報の取得方法

動作情報の取得方法としては，取得対象となるソフトウェアに手を加えることなく動作情報が取得できることが理想的である．そのためには，対象ソフトウェア外部からそのソフトウェアの動作情報を取得する仕組みが必要になる．

3.2.1 AspectJ と JVMTI

動作情報の取得に利用可能なツールとして，AspectJ[2] と JVMTI[3] を紹介する．

AspectJ では，ロギングに代表される機能単位で分割されたモジュールへの，横断的な作用を要する機能について，織り込み（**ウィービング**）という仕組みを用いてモジュール化する．ウィービングとは，既存のバイトコードに特定のコードを埋め込むことを指す．また，このとき AspectJ で実装されたモジュールをアスペクトと呼ぶ．AspectJ は次の特徴を持つ．

- アスペクトと対象ソフトウェアは独立
- 任意の場所に直接コードをウィーブ可能

したがって，AspectJ を使用し動作情報取得用のコードをウィーブすることで，Eclipse のソースファイルを変更することなく動作情報を取得できる（プル型）．

一方，JVMTI は JDK[4] に含まれるツールであり，開発ツールおよび監視ツール用のネイティブプログラミングインタフェースである．JVM 内の様々なイベントに対してコールバック関数を設定でき，その関数内でイベントに対する処理を適切に振り分けながら解析を行うことができる（プッシュ型）．これは JVM の情報を直接利用することになるため，AspectJ よりも詳細な情報を取得することができる．

しかし，Eclipse のような拡張可能ソフトウェアにおいては，全てのメソッド呼び出しにおいて詳細な情報を取得すること以上に，拡張の仕組みに関連した情報を重点的に取得できることの方が重要である．また，特定部分の情報をダイレクトに取得することは，情報爆発抑制への貢献が期待できる．したがって，本研究では動作情報の取得に AspectJ を使用する．

3.2.2 AspectJ による動作情報取得

メソッドとコンストラクタに対して動作情報取得用のコードをウィーブすることで，オブジェクト生成やスレッドに関する情報も取得することができる．特にスレッドに関しては，Java の Thread クラスのメソッド呼び出しを網羅することでスレッドの開始や終了などを把握することができる．

次に，拡張ポイントに関する情報取得用コードのウィーブを行う．Eclipse 内部の API と拡張ポイント定義ファイルを併用し，拡張ポイントの定義上実装が義務づけられたインタフェースを中心に取得情報を拡大させる．

そして，取得した動作情報を視覚化することで，その有効性を評価する．任意の場所にコードをウィーブ可能な AspectJ の利点をいかし，不足していると判断された情報についてはウィーブ場所を追加する．一方，不要な情報については情報取得の省略を行う．

4. プラグイン管理技術

Eclipse はプラグインプラットホームと，プラグインの集合からなるソフトウェアである．その管理技術として，Eclipse のバージョン 3.2.0 から OSGi[5] が採用されている．Eclipse のプラグインプラットホームとプラグインは，それぞれ OSGi のサービスプラットホームと Bundle に相当する．

4.1 OSGi

OSGi は，そのプラットホームである Framework とその上でサービスとして動作する Bundle で構成される，Java 言語で実装されたサービスプラットホームであり，OSGi アライアンスによって標準化されている．Bundle は Framework 上で動作する機能を構成するソフトウェアモジュールであり，通常は Java のパッケージ群が格納された jar ファイルである．OSGi では Framework と各 Bundle の接続規定を定めている．

4.2 Bundle マニフェスト

各 Bundle の構成や接続方法を規定するために，Bundle ごとにマニフェストファイルが用意される．マニフェストファイルには，OSGi マニフェストヘッダ列を記述する．

Export-Package ヘッダ

Bundle 内に存在する Java のパッケージは，エクスポートすることで外部の Bundle に公開することができる．また，各 Bundle は固有のクラスパス空間を持っているが，他の Bundle と接続する場合には，接続先 Bundle のクラスパス空間も統合される．ヘッダの値として，エクスポートするパッケージ名をカンマで区切って列挙する．

Require-Bundle ヘッダ

ある Bundle が提供する機能の実行を行う際，関連する Bundle がプラットホーム上に接続されていなければならないことがある．各 Bundle は自身の機能を実行する上で必要な Bundle（**必須 Bundle**）を明示的に宣言することができる．ヘッダの値として，必須 Bundle の ID とバージョンを記述する．このとき，必須 Bundle がエクスポートしているパッケージを使用可能になる．

4.3 プラグインの依存関係

Eclipse において，プラグイン A の実行に必要なプラグイン B があるとき，プラグイン A はプラグイン B と依存関係にあるという．プラグインの依存関係は，プラグイン開発において非常に重要である．

しかし実際は，依存関係にあるプラグインとは，OSGi のマニフェストファイル内で Require-Bundle として指定された必須 Bundle のことである．

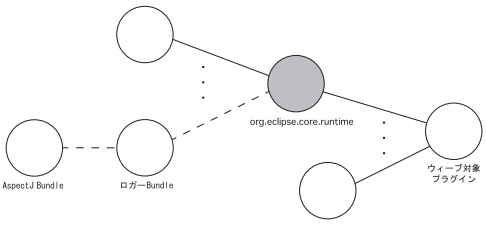

図 1 動作情報取得用の Bundle 構成
Fig. 1 Bundle construction for getting information

5. 情報取得システム

AspectJ により Eclipse に対して単純なウィーブを行うだけでは，動作情報を取得することはできない．これは OSGi では Bundle ごとに固有のクラスパス空間を持つため，動作取得用パッケージが Bundle から不可視となるためである．

5.1 動作情報取得の枠組

Eclipse に対する動作情報取得を実現するため，OSGi に準拠する形式で Eclipse の動作情報を取得する実験を実施した．動作情報取得用の Bundle 構成を図 1 に示す．Bundle を円形，依存関係を実線で表している．

図 1 中の AspectJ Bundle とは，AspectJ のランタイムライブラリを保持する Bundle であり，AspectJ のパッケージを全てエクスポートしている．これは，ウィーブされたコードの実行には AspectJ のランタイムライブラリ (aspectjrt.jar) が必要となるからである．

また，ロガー Bundle は情報取得用のロガークラスなどが格納されており，必須 Bundle として AspectJ Bundle を指定している．さらに，実際にコードがウィーブされるプラグインから，ロガークラスを利用可能にするため，ロガー Bundle は AspectJ Bundle を再エクスポートしている．再エクスポートとは，必須 Bundle として指定した Bundle がエクスポートしているパッケージを，そのまま他の Bundle にエクスポートし直すことである．

ウィーブ対象プラグインとロガー Bundle 間に依存関係を持たせることで動作情報の取得が可能になる．依存関係の追加にはマニフェストファイルの変更が

表 1 実装プログラムの規模
Table 1 a scale of implemented program

	取得部	可視化部
ソースファイル数	56	125
クラス数	58	133
行数	3,376	9,461

必要である．そこで，プラグインの依存関係を調査した結果判明した，どのプラグインとも依存関係を持つ org.eclipse.core.runtime プラグインに着目する．Eclipse の中心的なプラグインが，ロガー Bundle を再エクスポートすることで，他のプラグインのマニフェストファイルを変更することなく動作情報の取得機能を追加することができる．

その結果，ロガー Bundle が AspectJ Bundle を再エクスポートし，org.eclipse.core.runtime プラグインがロガー Bundle を再エクスポートすることで，ウィーブ対象となるプラグインから動作情報取得用 Bundle 群との依存関係をを隠蔽することができる．このことを，org.eclipse.core.runtime 以降の依存関係を破線とすることで表している．

5.2 実　　装

取得した動作情報を確認するために，動作情報を表示するビューワを Eclipse プラグインとして実装した．Java 言語によって実装し，描画ライブラリとして GEF[8] を利用した．実装システムは情報取得部と情報可視化部に分かれる．それぞれの規模を表 1 に示す．情報取得部は 5.1 節の仕組みそのものであり，ツールにより自動生成されたプログラムはカウントしていない．

実装したプラグインは，動作情報の取得開始・終了を指示するコントローラ部とビューワ部から構成される．実際の画面を図 2 に示す．

動作情報の取得は，コントローラ部の情報取得開始ボタンを押すことで開始され，動作情報の知りたい操作を終えた後，取得終了ボタンを押す．このとき，動作情報は動作の種類（メソッド呼び出し等）に応じて，それぞれ特定のクラスとしてオブジェクト化され，メモリ内に記録される．ソフトウェアの動作に関する情報を記録し，ソフトウェア理解に活用することは有用であり，関連研究として 6), 7) がある．ビューワ部は，メモリ内に記録された実行情報（リポジトリ）を選択することでその内容を表示する．図 2 から，取得したメソッド呼び出しに関する動作情報が確認できる．また，右下のプロパティビューには，メソッド呼び出しについての情報が表示される．このビューワ部は Eclipse の動作を提示するツールのプロトタイプとなる．

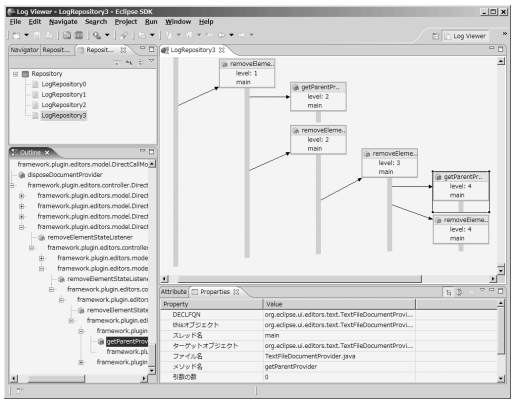

図 2 実装したビューワ
Fig. 2 implemented viewer plugin

6. おわりに

本研究では，Eclipse に代表される拡張可能ソフトウェアの理解支援を目標としており，本稿では Eclipse の動作情報取得方法を提案した．また，動作情報の取得とその表示ツールのプロトタイプを Eclipse のプラグインとして実装した．今後の課題は次の 3 点である．

- 拡張ポイントに関する情報取得
- 取得すべき情報の考察
- 情報の表示方法の実装と評価

より効果的な動作情報の提示を実現するため，膨大な情報になると予測される動作情報に対し，必要な情報を絞り込む方法について考察する．その後，その方法を実際にビューワとして実現することで動作情報の取得や表示方法の有効性を評価する予定である．

参 考 文 献

1) Eclipse: `http://www.eclipse.org/`
2) AspectJ: `http://www.eclipse.org/aspectj/`
3) JVMTI: `http://java.sun.com/j2se/1.5.0/docs/guide/jvmti/`
4) JDK: `http://java.sun.com/j2se/1.5.0/ja/`
5) OSGi Alliance: `http://www.osgi.org/`
6) 谷口 孝治, 石尾 隆, 神谷 年洋, 楠本 真二, 井上 克郎: “Java プログラムの実行履歴に基づくシーケンス図の作成”, 日本ソフトウェア科学会 FOSE 2004, ソフトウェア工学の基礎 XI, pp.5–16, 2004
7) 大森 隆行, 丸山 勝久: “追跡性を考慮したソースコード変更の抽出”, 情報処理学会研究報告, Vol.2007, No.33, pp.159–166, 2007
8) The Graphical Editing Framework (GEF): `http://www.eclipse.org/gef/`