社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

拡張可能ソフトウェアの動作情報を用いたプラグイン開発支援

清崎　大輔†　　大久保弘崇††　　粕谷　英人††　山本晋一郎††

† 愛知県立大学大学院 情報科学研究科　〒 480–1198 愛知郡長久手町大字熊張字茨ヶ廻間 1522–3
†† 愛知県立大学 情報科学部　〒 480–1198 愛知郡長久手町大字熊張字茨ヶ廻間 1522–3
E-mail: †kiyosaki@yamamoto.ist.aichi-pu.ac.jp, ††{ohkubo,kasuya,yamamoto}@ist.aichi-pu.ac.jp

あらまし　本稿は，拡張可能ソフトウェアのプラグイン開発支援システムを提案する．既存プラグインの動作をトレースすることで，拡張ポイントの選択と，その利用定義に関する実例が得られる．我々は OSGi ソフトウェアの動作情報取得手法を提案している．その手法で得られる動作情報を利用して，特定機能実行時における動的なプラグイン依存関係，及び関連するクラスやメソッドを提示する開発支援システムを実現する．実現したシステムでは，プラグインの上下関係や機能の中心となるクラスを考慮した可視化を行う．また，関連クラスのソースプログラムも提示可能である．

キーワード　プラグイン開発 OSGi

Plugin Development Support using Execution Information on Extensible Software

Daisuke KIYOSAKI†, Hirotaka OHKUBO††, Hideto KASUYA††, and Shinichirou YAMAMOTO††

† Aichi Prefectural University, Graduate School of Information Science　1522–3, Ibaragabasama, Kumabari, Nagakute, Aichi-gun, Aichi, 480–1198, Japan
†† Aichi Prefectural University, Faculty of Information Science and Technology　1522–3, Ibaragabasama, Kumabari, Nagakute, Aichi-gun, Aichi, 480–1198, Japan
E-mail: †kiyosaki@yamamoto.ist.aichi-pu.ac.jp, ††{ohkubo,kasuya,yamamoto}@ist.aichi-pu.ac.jp

Abstract　This paper proposes a plugin development support system for extensible software. Tracing behavior of existing plugins provides extension points which these plugins use and the definitions of the points. Our previous approach have presented a method to acquire execution information on OSGi-based software By utilizing execution information, our system visualizes dynamic plugin dependencies, and the associated classes and methods, considering parent-child relation in the dependencies and principal classes in an execution. In addition, our system provides a viewer of source codes of the classes.

Key words　Plugin development OSGi

1. ま え が き

近年のソフトウェア開発では，再利用性の高いプログラムの設計や生産性向上を支援するライブラリの利用に加え，各種の部品を柔軟に組み合わせることができる機能拡張のベースとなるプラットフォームに注目が集まっている．プラットフォームに機能拡張モジュール（**プラグイン**）を追加することで，プラットフォーム部を改変することなく，機能を追加することができる．このようなプラグインによる機能拡張をサポートするソフトウェアを拡張可能ソフトウェアと呼ぶ．従来のソフトウェアの機能拡張と比較して，開発者は本来実現したい機能の実装に注力できる．

より柔軟な機能拡張を目的としてプラットフォームの機能拡張の仕組みが拡張されていった．その結果，本来機能拡張を効率よく行うために用意された機能拡張の仕組み自体が大規模化し，その仕組みやベースとなるプラットフォームの理解は確実に困難になってきている．この問題に対し，OSGi アライアンス [1] は 2000 年に，組込ソフトウェアを対象としてプラグインの独立性向上とプラグインの柔軟な追加と削除を実現するための標準仕様を策定した．OSGi に準拠した拡張可能ソフトウェアの代表例は統合開発環境 Eclipse [2] である．Eclipse プラグインに代表される，プラグイン開発の活発化を背景に，我々は

2007 年に Eclipse の実行時情報（**動作情報**）を取得するための手法について提案した [3].

1.1 目　　的

本研究は，拡張可能ソフトウェアのプラグイン開発支援を目的としている．本論文では，先行研究で提案した動作情報取得手法を用いたプラグイン開発について述べる．具体的には，プラグイン開発において不足している情報を，拡張可能ソフトウェアの動作情報を用いてプラグイン開発者に提示する．そのために取得すべき動作情報と，動作情報を利用したプラグイン開発支援方法について考察する．そして，動作情報の提示機能を備えたプラグイン開発支援ツールを Eclipse プラグインとして実装する．また，先行研究で提案した，動作情報の取得手法についての評価を行う．

1.2 概　　要

本論文の構成を説明する．2 章で拡張可能ソフトウェアと Eclipse の機能拡張の仕組みについて述べる．3 章では，プラグイン開発において必要な情報について調査し，4 章では本研究で扱う動作情報の定義及び，動作情報を用いたプラグイン開発支援について述べる．5 章では考察した支援方法を実際に実装したプラグイン開発支援ツールについて説明する．そして，6 章では動作情報の提示に関する関連研究を取り上げ，本研究との比較考察を行う．7 章では本研究で実現した動作情報取得機能の取得能力と実装ツールによるプラグイン開発支援について評価を行う．最後に 8 章で本論文のまとめと今後の課題について述べる．

2. 拡張可能ソフトウェア Eclipse

Eclipse は，IBM が自社及び他社のソフトウェアを統一的に扱えるようにするために研究・開発した統合開発環境である．Eclipse は OSGi の仕様 [4] に準拠したプラグインプラットフォームを提供しており，ソフトウェア実装のベースとなるプラットフォームとして注目が集まっている．そのプラットフォームに用意された機能拡張の仕組みが拡張ポイントである．

2.1 拡張ポイント

拡張ポイントは，機能拡張用のプラグインをプラグインプラットフォームから利用するためのエントリポイントに相当する．実装レベルでは，Java におけるインタフェースである．特定の機能を拡張する場合，その機能を拡張するために用意された拡張ポイントに対し，規定のインタフェースに適合するように開発したプラグインをプラグインプラットフォームに登録する．

2.2 Eclipse のプラグイン

Eclipse におけるプラグインとは，拡張ポイントが規定するインタフェースを実装したクラスや，必要なリソースの集合である．通常は，プラグイン構成を記述したプラグインマニフェストファイルとともに，Jar ファイル形式にまとめて配付される．プラグインマニフェストファイルには，そのプラグインが使用する拡張ポイントや，そのプラグインが新たに公開する拡張ポイントの有無を記述する．

また，プラグイン A がプラグイン B を必要とするとき，プラグイン A，B 間には依存関係があるという．

表 1　Web 上の情報量 (2008/1/13 現在)
Table 1　Quantity of information on Web (2008/1/13)

拡張ポイント名	全世界	日本
org.eclipse.ui.actionSets	4,450	41
org.eclipse.ui.editorActions	1,160	19
org.eclipse.team.core.repository	25	2
org.eclipse.core.resources.refreshProviders	8	0
org.eclipse.pde.core.source	29	0

3. プラグイン開発に必要な支援

Eclipse における，プラグイン開発支援について議論する．まず，通常のプラグイン開発の手順を紹介し，その後でプラグイン開発に必要な支援のアプローチについて述べる．

3.1 プラグイン開発フロー

Eclipse のプラグイン開発は，大きく分けて次の工程で行われる．

（1） 拡張ポイントの調査
（2） 拡張ポイントの利用に関する定義
（3） プラグインの構成，依存関係定義
（4） API の調査と実装

プラグイン開発は上記工程の繰り返しであり，機能拡張を行うたびに調査と実装が必要となる．

3.2 プラグイン開発の問題点

Eclipse のプラグイン開発が困難である理由について述べる．

（1） ドキュメントの不足

ボタンのような簡単な例については Web 上にも有用な情報が載っているが，より複雑な機能を実現する拡張ポイントについてはほとんど情報が載っていない．2008 年 1 月 13 日に Google で，拡張ポイント名を検索キーとして全世界のサイト，日本語サイトを検索した際のヒット数を（表 1）に示す．調査対象拡張ポイントは，上 2 つ以外については UI 拡張でない拡張ポイント全体から無作為に選択した．

（2） API ドキュメントの限界

Eclipse 付属の API ドキュメントには，公開されているクラスやメソッドの役割は述べられているが，それらクラスやメソッドを適切に利用するための手続きまでは知ることはできない．Eclipse の API には，定型文に相当する固有のコードパターンが存在し，その手続きなしでは目的クラスのオブジェクトを取得できないことが多い．例えば，エディタを開く機能は，IWorkbenchPage クラスの openEditor メソッドを呼び出す必要がある．しかし，IWorkbenchPage クラスのオブジェクトの取得方法は API ドキュメントの該当項目だけでは分からない．

3.3 プラグイン開発に有用な情報に関する調査

Eclipse のプラグイン開発者が必要としている情報を調査するために，コミュニティサイト EclipseZone [5] を対象に，プラグイン・開発一般フォーラム (Platform Forum) の Eclipse に関する質問を，表 2 に示す 7 つのカテゴリに分類する．

表 2　分類カテゴリ一覧
Table 2　Classification category

カテゴリ	説明
依存関係	質問がプラグインの依存関係に直接関連するか，その解決法が依存関係にある．
拡張ポイント	プラグインの定義に関する質問．または拡張ポイント名やプラグイン定義が解決法として提示されている．
実装	API に関する質問．または，質問者の要求に対してコード断片が提示されている．
トラブル	プラグインが動作しないといった障害報告．
未解決	質問自体が曖昧であるなどの理由で適切な回答が得られていない（回答数 0 含む）．
プラグイン実装	プラグイン定義やクラスローダなどの，プラットフォーム実装に関する質問や回答．
その他	Eclipse の利用・設定方法，新バージョンのアナウンスなどの一般的な質問．

表 3　質問の分類 (Platform Forum)
Table 3　Classification (Platform Forum)

分類名	数	割合 (%)
依存関係	39	8.4
拡張ポイント	71	15
実装	182	39.3
トラブル	55	12
未解決	35	7.6
プラグイン実装	10	2.2
その他	71	15
合計	463	100

3.3.1　調 査 結 果

2007 年 9 月 28 日から過去 463 件のトピックを調査した結果を表 3 に示す．この調査結果から，最も多いカテゴリは実装であり，拡張ポイント，依存関係と続くことがわかる（その他カテゴリは除く）．また，質問に対する回答の多くがコード断片であることからも，Eclipse のプラグイン開発においてはドキュメントよりソースの方が有用であることがうかがえる．この結果と 3.2 節の問題点を勘案すると，実装，拡張ポイント，依存関係の 3 カテゴリについて下記の情報が不足しているといえる．

- 実装
  - Eclipse 固有のコードパターン
  - API 利用の実例
- 拡張ポイント
  - 拡張ポイントと対応づけられた機能
  - プラグイン定義の例
- 依存関係
  - プラグインの依存関係
  - 特定クラスの利用に必要なプラグイン

4.　動作情報を用いたプラグイン開発支援

本章では，本研究で扱う動作情報とプラグイン開発支援の概要について言及する．

4.1　動作情報を用いる理由

機能の充実を実現するため，利用可能なプラグインや拡張ポイントは増加している．その一方で，ドキュメントの整備は不十分である．そのためプラグイン開発者は，既存プラグインのソースプログラムを読みながらプラグイン開発を行っている．既存ドキュメントから必要な情報を取得することが困難であることから，実際に動作している Eclipse から自動的に取得して提示することが有効であると考えられる．

4.2　支 援 方 法

プラグイン開発支援において不足している情報を，プラグイン開発者が実際にどのように解決しているのかを説明し，そのあとで支援方法について述べる．

4.2.1　本研究での支援方法

本研究のプラグイン開発支援は，プラグイン開発者が行っていた調査を支援するという性質から，従来の調査方法に沿った支援方法を考える．

各種情報を得るために従来行っていた調査方法を下記にまとめた．

（1）　拡張ポイントの選定と利用定義
類似する機能を提供するプラグインをソースプログラムから特定し，そのプラグイン定義を参照する．

（2）　依存関係の定義
利用したい機能や API を提供するプラグインをソースプログラムから特定する．

（3）　実装
類似機能を実現するクラスあるいはメソッドのソースプログラムを参照する．

各調査で類似機能を参考にしている理由は，Eclipse のプラグインでは類似機能を提供するプラグインは同一拡張ポイントを利用する可能性が高いという経験則による．

本研究での支援方法について考察する．Eclipse では拡張機能はプラグインとして配布されるため，プラグインの依存関係を考慮する必要がある．すなわち，利用したい機能や API の利用に必要なプラグインや，間接的に依存しているプラグインを定義する．Eclipse 標準のプラグイン依存関係ビューでは，静的な依存関係しか提示できない．また，プラグインの定義と依存関係の定義はプラグインマニフェストファイルに記述する．そのため，(1) 拡張ポイントの選定と利用定義及び，(2) 依存関係の定義では共通の調査作業が行われる．このとき，Eclipse ではプラグイン定義の記述・閲覧は専用のマニフェストエディタで行う．そこで，これらの段階の支援として，類似機能実行時の動的なプラグイン依存関係をグラフとして提示する．そして，グラフからプラグイン定義にジャンプできるようにする．

次に，類似機能を提供するクラスの特定と，コードパターンについての情報を提示する．そのために，類似機能の実行に関

連したクラスと，その内部で実行されたメソッド呼び出しについての情報をそれぞれ可視化する．また，クラスやメソッドのソースプログラムを参照する機能を提供する．

このように，本論文でのプラグイン開発支援は，プラグイン開発に不足している情報を提示するだけではなく，その提示機能と既存の開発機能とを連携させる．

4.3 取得すべき動作情報

一般に，ソフトウェアの動作を動作情報に基づいて提示するためには，下記の情報を取得する．

- メソッド呼び出し
- スレッド実行
- オブジェクトの生成

4.2.1 節で考察した支援を実現するために，取得すべき動作情報について考察する．

4.3.1 メソッド呼び出しに関する考察

ソフトウェアの動作を把握する役割の他に，コードパターンの提示や API に関する情報を提供するためにも，メソッド呼び出しの情報を取得する必要がある．コードパターンとして利用されるのは，Eclipse が提供する特定のメソッドである．その呼び出し手順を把握するためには，そのメソッド呼び出し履歴の取得が必要である．また，Eclipse のように GUI を採用したソフトウェアはマルチスレッドで実行されるため，メソッド呼び出しが行われたスレッドを特定することも必要である．

一方，特定クラス利用時に指定すべき依存プラグインについては，実行されたメソッドのパッケージ名を基に，そのパッケージを公開するプラグインを探索することで特定できる．各メソッドが所属するプラグインの特定により，メソッド呼び出し関係を動的なプラグイン依存関係とみなすことができる．また，Eclipse 実行時に呼び出されたメソッドが，拡張ポイントで定義されたインタフェースのメソッドであるか調べることで，機能と拡張ポイントを対応づけられる．

このように，メソッド呼び出しの情報から，所属プラグインや所属クラスの特定が可能である．そして，動的なプラグイン依存関係やコードパターンについての情報もメソッド呼び出し情報から得ることができる．

4.3.2 オブジェクト生成に関する考察

実行履歴からプログラムの動作を理解する関連研究 [6], [7] では，オブジェクト生成についての情報を取得している．しかし，拡張可能ソフトウェアでは不正なオブジェクトの生成を防ぐため，プラットフォームがファクトリメソッドが提供される．また，Eclipse プラグインでは，パッケージに対するアクセス権限を定義できる．すなわち，アクセス制限属性の指定により，外部プラグインからの利用を制限できる．このように，プラグイン開発者が直接 Eclipse が提供するクラスのオブジェクトを生成する機会は少ない．そこで，オブジェクト生成の情報を取得することが，プラグイン開発支援に寄与するか検討するために，既存プラグイン AmaterasUML を対象に実態調査を行った．

AmaterasUML は，UML 図エディタと，その UML 図から Java のコードを自動生成する機能を提供するプラグインである．2007 年 12 月現在の最新版 1.3.0.1 を対象に，AmaterasUML プラグインのソースプログラム中に出現するオブジェクト生成を，その生成クラスの種類によって 5 つのカテゴリに分類する．AmaterasUML 内の全てのオブジェクト生成文 1,002 カ所を対象に，生成オブジェクトの型を基に分類した結果を表 4 に示す．表 4 より，AmaterasUML プラグインのソースプログラム中に出現するオブジェクト生成文において，AmaterasUML 内部のクラス 243 種類が生成されるのに対し，Eclipse 内のクラスは 17 種類しか生成されない．Eclipse 内のクラスを生成する文が全体の約 1 割存在する事実から，オブジェクト生成を無視して良いと結論づけることはできない．そこで，Eclipse のクラス 17 種類に対し，その役割を調査した．

その結果，Eclipse が提供する，ファイルリソースなどを表すモデルを表すオブジェクトの生成は 19 カ所であった．その他の大部分は，API の仕様上型を合わせるために使用するラッパークラスのような一時的に利用されるクラスであった．すなわち，オブジェクトの生成文 1,002 カ所のうち，Eclipse が提供する実用的なクラスのオブジェクト生成は 19 カ所 (1.9 %) である．この調査結果から，Eclipse のプラグイン開発においてはオブジェクト生成の重要度は低い．したがって，代替手段であるファクトリメソッドの情報を取得できればオブジェクト生成については取得する必要はないと判断した．

4.3.3 本研究での動作情報

本研究において動作情報とは，以下の情報を指すことにする．

- メソッド呼び出し
- スレッドの実行
- 使用されたプラグイン
- 実行時に使用された拡張ポイント

プラグインや拡張ポイントに関する情報は，メソッド呼び出し情報から得られる付随情報である．しかし，Eclipse の動作を表すという点で動作情報である．

また，プラグイン定義の実例やソースプログラムは，使用されたプラグインや拡張ポイントの情報を基に静的に取得できる．Eclipse の動作情報としてこれらの情報を取得することで，3.3.1 節で考察した内容は全て提示することができる．

5. 動作情報提示ツール

本研究で考えるプラグイン開発支援とは，従来プラグイン開発者が直接行っていた情報収集，すなわち参考になるプログラムの調査を支援することである．本章では，実装したプラグイン開発支援ツールについて述べる．

5.1 ツール概要

本ツールは，3.3.1 節で考察したカテゴリの情報を提示するプラグイン開発支援ツールである．実装言語に Java を用いて，Eclipse のプラグインとして実装した．本ツールのプラグインは，次の 3 つのプラグインで構成される．それぞれのプログラム規模を表 5 に示す．

（1） 動作情報取得プラグイン

文献 [3] で我々が提案した，AspectJ [8] を用いた動作情報の取得を行う．

（2） 動作情報取得制御プラグイン

表 4　AmaterasUML での分類結果
Table 4　Observation results in AmaterasUML

カテゴリ	説明	出現回数	クラス種類数
Java	Java 標準ライブラリに含まれるクラス	131	20
SWT/Jface	Eclipse 用の GUI コンポーネント	132	43
Eclipse	Eclipse が提供するクラス	**109**	**17**
Amateras	AmaterasUML で定義されたクラス	369	243
Draw2d/GEF	Eclipse 用描画ライブラリ内クラス	261	64
合計		1,002	387

表 5　プログラム規模
Table 5　The scale of program

プラグイン	クラス数	行数	実行行数
動作情報取得	85	8,073	5,364
動作情報取得制御	3	167	61
ビューワ	291	24,758	12,522

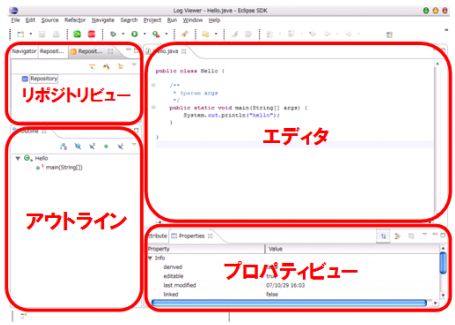

図 1　動作情報閲覧モードの画面構成
Fig. 1　Main screen of LogViewer mode

表 6　動作情報提示用のビュー
Table 6　Views to represent execution information

知りたい情報	利用するビュー
拡張ポイント，依存関係	プラグイン依存関係
実行に関与したクラス	クラス間コラボレーション
利用された API，コードパターン	メソッド呼び出し

動作情報取得プラグインを制御する GUI 部.

（3） ビューワプラグイン

動作情報を提示する各種ビューを提供する.

ツールの概観を図 1 に示す．本ツールは，動作情報閲覧モードを実装しており，Eclipse でのプラグイン開発モードと動作情報閲覧モードは容易に切り替えられる．

5.2　提示ビュー

本ツールで実装した，動作情報の提示ビューを表 6 に示す．各ビューについては後節で説明する．

5.2.1　プラグイン依存関係

プラグイン依存関係ビューは，動作情報記録期間中に利用されたプラグイン，及びその依存関係をグラフとして表示する（図 2）．円形ノードはプラグインを表し，依存先プラグインへ向かってエッジが引かれる．黄色で表示された円形ノードは実行時に利用されたプラグインを表す．水色の円形ノードは実行時には利用されなかった，プラグイン定義上の依存プラグインを表す．また，拡張ポイントを利用したプラグインには円形ノード右下に専用の印が表示される（図 2 拡大部）.

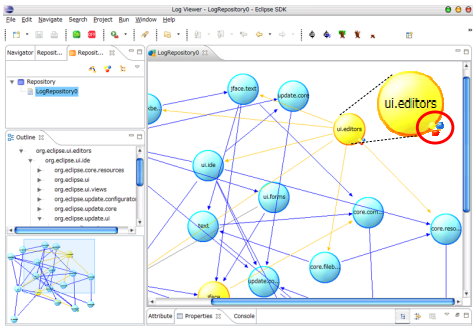

図 2　動的なプラグイン依存関係
Fig. 2　Dynamic plugin dependencies view

5.2.2　クラス間コラボレーション

クラス間のメッセージ送信を提示する．ノードがクラスを表し，エッジは他クラスのメソッド呼び出しを表す．取得した動作情報において，メソッドの呼び出し回数とエッジの数を基にクラスに重要度を設定する．類似機能の実行において中心的な役割を担うクラスと，その他クラスの関連を可視化することで，類似機能の全体像と各クラスの役割を提示する．最重要クラスを強調するため，Fisheye の考え方をエッジの長さに応用した [9]．すなわち，最重要クラスから直接利用されるクラスのエッジは長くする．一方，間接的に到達できるクラスへのエッジは，最重要クラスからの距離が離れるほど短くする．図 3 では，類似機能の中心的な役割を担うクラス周辺が空間的に広くなる．これにより，中心的な役割を担うクラスと直接関連したクラスの把握が容易になる．また，同一プラグインに属すクラスは近辺に配置される．したがって，プラグイン依存関係ビューからこのビューに切り替えても，プラグイン開発者の理解作業を阻害しない.

5.2.3　メソッド呼び出し

メソッド呼び出しビューの画面例を図 4 に示す．メソッドは四角形で表され，メソッド呼び出しをエッジで表している．縦方向が時間を表し，横方向がメソッド呼び出しの深さを表す．このビューは，メソッド呼び出しを時系列順に配置しており，プログラムの実行と 1 対 1 で対応づく．また，共通のメソッド呼び出しパターンの連続は折りたたまれる．特定クラスからの実行を追う場合や，目的メソッドを呼び出すまでの手続きを調

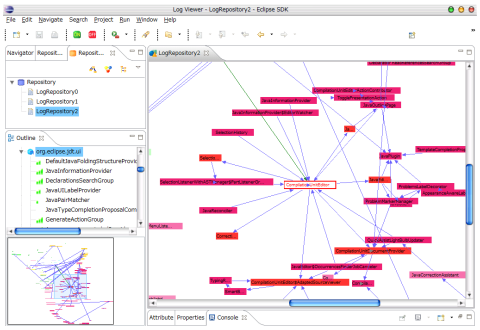

図 3　クラス間コラボレーションビュー
Fig. 3　Collaboration view

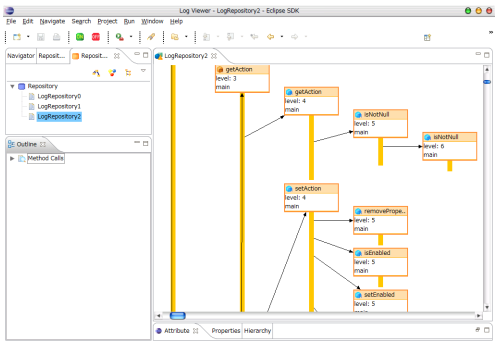

図 4　メソッド呼び出しビュー
Fig. 4　Method call view

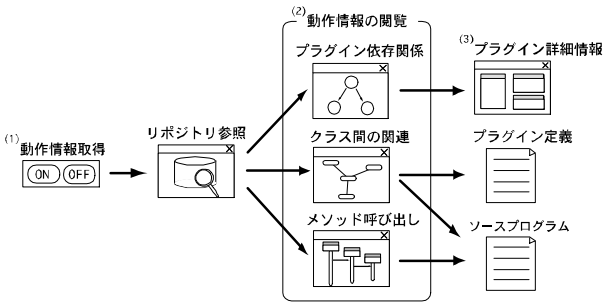

図 5　開発支援の流れ
Fig. 5　Support flow for plugin development

べる場合に有用である．

5.3　本ツールによるプラグイン開発支援例

本ツールによるプラグイン開発支援の流れを図 5 に示す．図中の (1)〜(3) について，以降の節でそれぞれ説明する．また，動作情報の利用方法は，プラグイン開発者の関心事によって異なる．ここでは，利用すべき拡張ポイントを調査する状況を例に説明する．

5.3.1　類似機能の動作情報を取得

まずはじめに，類似機能の動作情報を取得する．

動作情報の取得には，Eclipse 上に組み込まれた動作情報取得制御ボタンを用いる．動作情報の取得は，以下の手順で行う．

（1）　取得開始ボタン押下
（2）　類似機能の実行
（3）　取得終了ボタン押下

取得した動作情報はメモリ上のリポジトリに格納される．リポジトリの一覧は図 1 左上のリポジトリビューに表示される．ここでは，リポジトリをプラグイン依存関係ビューで閲覧したとする．

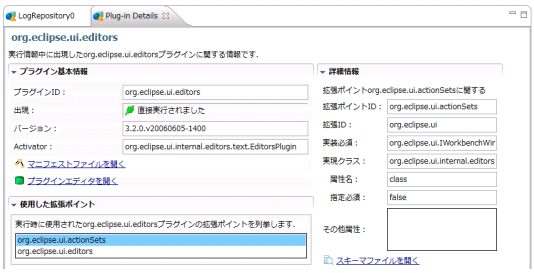

図 6　プラグイン詳細情報
Fig. 6　Plugin information in details

5.3.2　拡張ポイントを利用したプラグインの特定

プラグイン依存関係ビューから，拡張ポイントの利用を表す印が表示されているノードに注目する．プラグイン依存関係ビューでのノード配置には磁性ばねモデルレイアウトアルゴリズム [10] が利用される．磁界の設定により，依存元プラグインが依存先プラグインより上に寄せられる．そのため，GUI 関連の拡張ポイントであれば，拡張ポイントを利用したプラグインのうち，最上位のプラグインに注目することが有効である．また，注目プラグインについての詳細情報を見ることができる（図 6）．

5.3.3　Eclipse との連携

図 6 では，拡張ポイントを利用する上で必要な以下の情報が提示される．

- プラグイン ID
- 利用した拡張ポイントの ID
- 拡張ポイントの定義上，実装が必要なインタフェース

また，現在閲覧しているプラグインのプラグイン定義は，Eclipse 標準のプラグインマニフェストエディタ上に提示される．したがって，提示されたプラグイン定義を，実装中プラグインの定義にコピーすることで同一機能のプラグインを作成できる．さらに，実装作業において参考になるサンプルコードとして，拡張ポイント指定のインタフェースを実装したクラスのソースプログラムを Eclipse の Java エディタで閲覧することもできる．

このように，本ツールと Eclipse のプラグイン開発支援機能が連携することでシームレスなプラグイン開発が行える．

6.　関 連 研 究

動作情報の取得と応用という本研究のアプローチは，プログラム理解支援やリバースエンジニアリングの分野でも用いられる．

Jinsight [11] は，マルチスレッドで動作するオブジェクト指向設計のプログラムを対象としたプログラムの動的な振る舞いを可視化するツールである．開発当初はプログラム理解を支援する位置づけであったが，近年ではデバッグやプロファイリングに重点が置かれている．そして，プロファイリング結果の提

示とは別に，任意の部分実行トレースに関する情報提示が可能であり，メソッド呼び出し，オブジェクト生成，オブジェクト間の関連を視覚的に提示する．表示される情報の粒度が細かいため，特定クラスの調査のような，比較的注目箇所が絞り込まれた状況で有効である．本研究が目的としているプラグイン開発支援では，注目箇所を絞り込むために情報を可視化する点で，動作情報の応用方法が異なる．

Eclipse Monitor [12] は，クラスロード時にバイトコード操作を行うことで Eclipse の動作情報を取得し、その情報を可視化するプロファイリングプラグインである．Eclipse Universe ビューでは，プラグインのロード順序や，プラグイン間の関連についての情報が提示され，ロードされたプラグインを順に惑星として配置する．本ツールが扱う動作情報の他に，プラグインオブジェクトの生成，メモリの消費量や，オブジェクト内変数の値などの情報を取得している．しかし，プラグイン開発支援に必要な拡張ポイントの情報や，該当ソースプログラムへのジャンプ機能はない．

7. 評　　価

7.1 動作情報取得手法の評価

7.1.1 関連ツールとの比較

AJEER [13] は，Martin Lippert による Eclipse プラグインプラットフォームの AspectJ 拡張である．この研究の主な内容は，Eclipse のプラグインメカニズムを利用したアスペクトの宣言方法とその実装 [14] であり，AspectJ ライブラリとバイトコード操作用プラグインを OSGi フレームワークに組み込んでいる．独自の拡張ポイント “aspects” が用意され，プラグイン開発者が実装したアスペクト（を内包した）プラグインのプラグインマニフェストファイルに，aspects 拡張ポイントの利用定義を記述することで対象クラスのロード時にウィーブを行う．

AOSGi は OSGi のプラットフォームに対する AspectJ 対応を行う．クラスロード機能を拡張している点は AJEER と共通しているが，アスペクトの扱い方が異なる．AJEER によるウィーブの影響範囲は，アスペクトの定義のみで決まるが，AOSGi ではプラグインの依存関係で決まる．専用のマニフェストヘッダが定義され，ウィーブ対象プラグインのマニフェストヘッダを変更する必要がなくなった．AOSGi は一般公開されていないため直接利用することはできないが，Equinox Inqubator プロジェクト [15] では，AJEER と AOSGi を統合したプラットフォームの実現に取り組んでいる．

本手法と関連研究における，動作情報の取得に関する違いを表 7 に示す．

関連ツールはいずれも OSGi の仕様策定に携わる開発者らによって開発されている．そのため，情報取得の仕組みは，既存プラグインのマニフェストファイル修正が不要である点で本研究よりも優れている．しかし，本研究は OSGi 利用者の立場で AspectJ サポートを実現する手法を検討し，プラットフォームの修正を行うことなく同等機能を実現した．

また，Eclipse Inqubator が Eclipse プロジェクト下で進行中であるが，まだ一般の開発者が実際のシステム上に載せる段階ではない．したがって，現在開発者が利用できる Eclipse のロギングツールは AJEER といえる．AJEER によるロギングの際にはロギングを有効にするための設定ファイルを記述する必要がある．この設定ファイルを適切に記述するには一定水準の知識が必要であり，プラグイン開発初心者が自身のプラグイン開発の情報収集手段として使用するには無理がある．

以上から，プラットフォームの置き換えを必要とせず，OSGi の利用者としてのアプローチで動作情報の取得が実現できる本手法には十分な価値があるといえる．

表 8　評 価 環 境
Table 8　Evaluation environment

OS	WindowsXP Home Edition SP2
Java	1.5.0_07
Eclipse	3.2.0
AspectJ	1.5.3

表 9　動作情報の取得に失敗するプラグイン
Table 9　Plugins were failed to acquire execution information

プラグイン ID	バージョン
org.eclipse.core.contenttype	3.2.0.v20060603
org.eclipse.core.jobs	3.2.0.v20060603
org.eclipse.core.runtime.compatibility	3.2.0.v20060603
org.eclipse.equinox.common	3.2.0.v20060603
org.eclipse.equinox.preferences	3.2.0.v20060601
org.eclipse.equinox.registry	3.2.0.v20060601
org.eclipse.osgi	3.2.0.v20060601
org.eclipse.jface	3.2.0.I20060605-1400
org.eclipse.swt.win32.win32.x86	3.2.0.v3232m

7.1.2 情報取得能力に関する評価

先行研究で提案した手法では，Eclipse プラグインに対して，AspectJ を利用して動作情報取得機能を組み込んだ．この手法が，Eclipse の全プラグインに対してどの程度適用可能なのかを評価する．評価を行った環境を表 8 に示す．調査対象は Eclipse プロジェクトのプラグイン 87 個である．

評価方法は，各プラグインに対してウィーブを行い，Eclipse の起動や振る舞いが正常であるかを確認した．その結果，表 9 に示す 9 個のプラグインにおいて，Eclipse の実行中にエラーが起こることが分かった．なお，今後はプラグイン ID の共通部分 (org.eclipse) を省略する．

jface プラグインと swt プラグインを除く 7 つのプラグインは core.runtime プラグイン自体が依存しているプラグイン群である．そのため，先行研究の手法によって core.runtime プラグインがエクスポートしている AspectJ のライブラリは，それらプラグインでは不可視となる．しかし，その 7 つのプラグインは core.runtime プラグインによって隠蔽され，プラグイン開発者の関心の対象ではない．また，swt プラグインは SWT (Standard Widget Toolkit) と呼ばれる，Eclipse のプラットフォームとは独立した GUI ライブラリプラグインである．jface プラグインも SWT を利用した GUI フレームワークを提供するプラグインである．したがって，Eclipse プラグイ

表 7　関連ツールとの比較
Table 7　Comparison with related works

	本手法	AJEER	AOSGi
AspectJ サポートの実現	プラグイン	拡張ポイント	プラットフォーム
アスペクトの影響範囲	対象プラグインのみ	アスペクト定義による	アスペクト定義と依存関係
ウィーブ方式	コンパイル時	ロード時	ロード時
情報取得の準備作業	マニフェストファイルの設定	専用ランチャの設定	マニフェストファイルの設定
既存プラグインの修正	必要	不要	不要
プラットフォームの置き換え	不要	専用ランチャ	必要

ン開発者の関心の対象となるプラグインについては動作情報取得機能を追加できたといえる．

7.2　プラグイン開発支援としての評価

3.2 節では，ドキュメント不足とその内容の偏りを問題点として指摘した．そして，類似機能の動作情報を基に，拡張ポイントや API の例を提示するツールを実装した．そこで，本ツールがプラグイン開発の支援に寄与できているか評価するために，次の調査を行った．

（1） 2008/2/1 現在の Google において，拡張ポイント名をキーとして日本語サイトを検索したヒット数．

- 最も WEB 上での情報が多い拡張ポイントは org.eclipse.ui.views
  - ヒット数 559
- ヒット数が 0 だった拡張ポイントは 26 個
- ヒット数が 1 だった拡張ポイントは 68 個
  - 内，Eclipse のオンラインヘルプのみヒット 61 個

このように，87 個の拡張ポイントについて実用的な情報を提示できない．

（2） Eclipse のバージョン 3.2.0 において，標準構成で利用可能な拡張ポイント名と，その利用回数．

- org.eclipse.ui.views の利用回数 22
- どの拡張ポイントも少なくとも 1 回は利用される

したがって，次の 2 点から本ツールはプラグイン開発の支援が行えたと判断できる．

- Google でのヒット数が多い拡張ポイントについては十分な例を提示可能
- 動作情報を用いると，どの拡張ポイントについても少なくとも 1 つの例を提示可能

8.　まとめと今後の課題

本研究では，拡張可能ソフトウェアのプラグイン開発支援を目的とし，プラグイン開発支援に必要な情報とその情報を用いたプラグイン開発支援について議論した．そして，実際に Eclipse 上で動作するプラグイン開発支援支援ツールを実装した．動作情報の取得については関連研究の方が強力である．しかし，関連研究ではプラットフォームの置き換えが必要であり，動作情報取得実現のコストは高くなる．本研究の取得手法ではプラグインレベルでのアプローチを採用しているため応用の容易さで優れている．

また，動作情報を用いた支援という点では，現在は類似機能の実行に着目し，その動作情報をそのまま提示するといった基本的な支援しか行えていない．しかし，動作情報の貯蓄と解析を行うことで，対象ソフトウェアの動作以外の情報も提供できると考えられる．たとえば，問題に遭遇した熟練プラグイン開発者がどのような操作でその問題を解決するのかを貯蓄する．そして，その情報を解析して問題と解決法を対応づけて提示する．

さらに，本論文では本ツールのもつプラグイン開発支援の可能性について示した．実際に運用評価を行うことで，本ツールの実用性を示す．これらを今後の課題とし，さらなるプラグイン開発の支援を目指す．

文　献

[1] OSGi Alliance - http://www.osgi.org/
[2] Eclipse - http://www.eclipse.org/
[3] 清崎大輔, 大久保弘崇, 粕谷英人, 山本晋一郎, “OSGi に基づく拡張可能ソフトウェアの動作情報取得手法,” ソフトウェアエンジニアリング最前線 2007, pp.219–222, August 2007.
[4] OSGi Service Platform Release 4 - http://www2.osgi.org/Release4/HomePage
[5] Eclipse Zone - http://eclipse.dzone.com/
[6] D.F.Jerding, and J.T.Stasko, ”Using Visualization to Foster Object-Oriented Program Understanding”, Graphics, Visualization, and Usability Center, Georgia Institute of Technology, Technical Report GIT-GVU-94-33, 1994
[7] William DePauw, Richard Helm, Doug Kimelman, and John Vlissades. Visualizing the behavior of object-oriented systems. In Object-Oriented Programming Systems, Languages, and Applications Conference, pages 326–337, May 1993.
[8] AspectJ - http://www.eclipse.org/aspectj/
[9] Manojit Sarkar and Marc H. Brown, ”Graphical Fisheye Views of Graphs.” In Human Factors in Computing Systems: Proceedings of the CHI ’92 Conference. New York: ACM, 1992.
[10] 三末和男, 杉山公造: “マグネティック・スプリング・モデルによるグラフ描画法について”, 情報処理学会研究報告ヒューマンインタフェース 55-3, pp.17–24, 1994
[11] Jinsight - http://www.research.ibm.com/jinsight/
[12] Chis Laffra, Martin Lippert. Visualizing and AspectJ-enabling Eclipse Plugins using Bytecode Instrumentation. OOPSLA ’03, October 26.30, 2003, pp.70–71. ACM 1-58113-751-6/03/0010.
[13] AJEER - http://sourceforge.net/projects/ajeer
[14] Martin Lippert. AJEER: An AspectJ-Enabled Eclipse Runtime. OOPSLA ’04, Oct. 24.28, 2004, pp.23–24. ACM 1-58113-833-4/04/0010.
[15] Equinox Inqubator - http://www.eclipse.org/equinox/incubator/