2009－GN－71 (8)
2009／3／18

フィードバック情報収集機能の組み込み可能なフレームワークの開発

三村 次朗†，上田 賀一†
†茨城大学

フィードバック情報収集は高品質なソフトウェアを開発するために重要である．しかし，それ自体はソフトウェアの本質的な機能ではないので，開発中に機能を付加することはコストが高い．一方でアスペクト指向プログラミングは，オブジェクト指向技術の問題点を解決するための方法として注目を集めている．本研究では，アスペクト指向技術を用いてフィードバック情報収集機能を横断的関心事としてモジュール化した．そしてオブジェクト指向開発されたJavaプログラムに対してアスペクトを織り込むことで，フィードバック情報収集機能を効率的に組み込み可能とする方法を提案した．また，提案手法を実現するためのフレームワークを開発した．

A Framework for Adding Feedback Information Gathering Function to Software

Jiro MIMURA†, Yoshikazu UEDA†
†Ibaraki University

Gathering the feedback information is important to develop high quality software. But, it is not essential function of software. On the other hand, aspect-oriented programming is attracted as a method to solve problems of object-oriented programming. In this study, a feedback function is modularized as a crosscutting concern by using aspect-oriented technique. By weaving aspects into programs developed in object-orientation, we propose a method for embedding feedback functions efficiently. We develop a framework also to realize proposed method.

1 はじめに

近年，数多くのソフトウェアが開発されている．これらのソフトウェアは初版公開時から変化しない状況は稀であり，ユーザからの意見を元に機能を拡張したり，処理速度の向上やバグフィックスなどを経てバージョンアップされていくものである．ユーザの嗜好を上手く取り入れ，バグを取り除いて安定したソフトウェアを開発することがソフトウェア自体の完成度を高めることだと言ってもよい．そのため，ソフトウェア開発においてはフィードバック情報を如何に多く集めるかが，質の高いソフトウェアを開発するための重要な要因になると考えられる．特に，正式リリースの前段階に差し掛かっているソフトウェア開発においては，様々な環境での動作確認や，実行時例外が発生する条件などを特定するためのフィードバック情報を多く集めることが重要である．

しかしフィードバック情報収集は，ソフトウェア開発において本質的な目的ではない．品質向上などの目的のためにフィードバック情報を収集したいのであり，それ自体がソフトウェアの目的ではないからである．フィードバック情報収集機能は開発工程中の一定期間のみ利用し，ソフトウェアのリリースを繰り返す間に不必要になることもある．開発初期からフィードバック情報収集機能を組み込むことを計画していた場合は，それが不必要になった際にコードが残ってしまう．開発がある程度進んだ時にフィードバック情報収集機能を追加しようとすると，再設計や再実装などの予定していなかったコストが発生してしまう．また，開発が進んだソースコードに手を加えることで保守性の低下に繋がる [1]．よって，フィードバック情報収集機能には，再設計や再実装などのコストをかけずに実装でき，かつ既存の設計やソースコードに大きな修正を加えないという性質が必要である．

そこで本研究では，アスペクト指向言語を用いてフィードバック情報収集機能を効率的に追加・変

更する仕組みを提案する．また，それを行うための支援環境として，Eclipse[2] プラグインによるフレームワークを開発した．対象は Java によるオブジェクト指向開発における正式リリース前の比較的小規模なソフトウェアとする．開発者はどのようなフィードバック情報収集機能を組み込みたいのかといったことだけを選択すれば，目的の機能が自動的に実装される．

2 章では，アスペクト指向について説明する．3 章で提案手法について説明し．4 章でフレームワークについて解説する．また，開発したフレームワークを用いた実験について 5 章で述べる．6 章で関連研究との比較・考察を行い，7 章でまとめとする．

2 アスペクト指向

アスペクト指向 [3] は，オブジェクト指向の欠点を補う新しいモジュール化技術である．オブジェクト指向技術の普及に伴い，保守性や拡張性の高いシステムを構築することが可能になった．しかし，オブジェクト指向技術ではカバーし切れない問題が発生している．それはロギング処理に代表されるような「横断的関心事」の存在である．横断的関心事とは，処理内容が他の関心事に強く依存している関心事のことである．

そこで，依存側 (横断的な関心事をもつモジュール) が，依存先のどこで処理を行うのか，またどのような処理を行うのかを 1 つのモジュールとして定義すればよいと考えたのがアスペクト指向である．オブジェクト指向設計によるモジュール分割を対象として，上記のような別の観点により横断的な処理をモジュール化する考え方とも言える．また，モジュール化した処理を実際のプログラムに反映させることを，織り込みと呼ぶ．

アスペクト指向の Java 実装に，AspectJ[4] がある．本フレームワークおよびフレームワークを適用するプログラムは Java 言語を用いて開発するので，アスペクトの定義に AspectJ を用いる．AspectJ はジョインポイントモデルによって実現されている．ジョインポイントモデルはジョインポイント，ポイントカット，アドバイスと呼ばれる 3 つの概念から構成されている．ジョインポイントはメソッド呼び出しや例外発生などプログラム実行中フロー中の時点である．ポイントカットはすべてのジョインポイントの中から特定の条件を満たすものを選び出した部分集合である．アドバイスはポイントカットによって選び出されたジョインポイントでのプログラムの実行を変更または追加するものである．図 1 は，AspectJ でのアスペクトの定義例である．図中の 2 行目と 3 行目はポイントカットの定義である．これらのポイントカットはそれぞれ，戻り値無しで int 型の引数を 2 つ取る Figure クラスの move メソッドの呼び出し位置と，戻り値が Figure で引数無しの Figure クラスの dup メソッドを抽出している．5 行目以降はアドバイスの定義である．ポイントカットを基準にして，その前後に行う処理を定義することで，元ソースコードを変更せずにロギング処理を織り込むことができる．

```
aspect Logger {
   pointcut callMove() : call(void Figure.move(int, int));
   pointcut callDup() : call(Figure Figure.dup());

   before() : callMove() {
      System.out.println("Entering: move");
   }
   after() : callMove() {
      System.out.println("Exit: move");
   }
   before() : callMove() {
      System.out.println("Entering: dup");
   }
   after() : callMove() {
      System.out.println("Exit: dup");
   }
}
```

図 1: ロギング処理を織り込むアスペクトの例

3 提案手法

本章では，フィードバック情報の定義ついて議論する．続いて，フィードバック情報収集機能を効率的に付加するための手法を提案する．

3.1 フィードバック情報の定義

まず明示的なフィードバック情報および暗黙的なフィードバック情報について説明する．その後，本研究で用いるフィードバック情報の定義を行う．

3.1.1 明示的なフィードバック情報

明示的なフィードバック情報は，ユーザインタフェースに非常に近い部分から取得される．具体的にはメニュー操作の呼び出し履歴や画面操作の履歴である．

メニュー操作の呼び出し履歴は，ユーザビリティの改善のために利用することができる．

画面操作の履歴もユーザビリティの改善に利用可能である．画面操作の具体例としては，マウス操作や画面遷移の履歴などがある．マウスの操作は，クリックもしくはグラブした位置と，その位置からの移動経路である．画面遷移の履歴は，ユーザが何らかの入力を行い，画面が変化した場合にその結果を保存したものである．これらの履歴は，Web アプリケーションにおいては取得が比較的容易な情報であるが，一般的なスタンドアローンアプリケーションでは困難な場合がある．その場合は，画面遷移が起きる際に発生するアクションを利用して画面遷移を検知すべきである．この方法には 2 つの利点がある．1 つ目は，画面レイアウトに依存しないことである．画面遷移の間に発生するアクションを利用するので，画面レイアウトの影響は無い．2 つ目は，実装の容易さである．ア

クションは一般的にはメソッドの形で表現される場合が多く，大部分の実行環境ではメソッドの呼び出し形式は一定のパターンが存在する．そのため，アクションに対応して何らかの処理 (ここでは画面遷移が発生したことを検知する処理) を実行させることは比較的容易であると言える．

3.1.2 暗黙的なフィードバック情報

暗黙的なフィードバック情報は，プログラムの動作に関するものが多く，ユーザが意識せずに収集される．例えば次のようなものが挙げられる．

- メソッドの呼び出しログ
- メソッドの実行時間
- ネットワークの通信状態

メソッドの呼び出しログは，Java で例外が送出された時に生成されるスタックトレースログのようなものである．プログラム実行中に，どこでどのようなタイミングで例外が発生したのかを知ることができる．多くの例外は，実装段階で適切な例外処理を記述することによりリカバリ可能であるが，実行環境に依存するような例外 (スタックオーバーフローなど) をリカバリすることは難しい．この場合，例外に至るまでのメソッド呼び出し階層やオブジェクトの状態をログファイルとして保存する．そして，ログファイルを分析することでプログラム修正を行う．

メソッドの実行時間は，あるメソッドが呼び出されてから終了するまでの時間である．このフィードバック情報は，どのモジュールがパフォーマンスのボトルネックになっているかを調べるために使われる．パフォーマンスは，一般的にはプロファイラを用いて計測するが，開発者が保持する実行環境でしか計測できないため，計測環境が限られてしまう．そのため，十分な計測が行えない場合がある．

ネットワークの通信状態は，S/C 間でのレスポンス時間である．Web アプリケーションもしくは S/C 型アプリケーションの場合，ネットワーク状態がパフォーマンスを決める大きな要因となる．

3.1.3 本研究で用いるフィードバック情報の定義

本研究では，フィードバック情報を次のように定義した．

1. メソッドの呼び出し履歴と実行時間
2. オブジェクトの数と状態
3. ネットワークの応答速度

メソッドの呼び出し履歴と実行時間は，3.1.2 節で述べた暗黙的なフィードバック情報の呼び出しログと実行時間を同時に記録したものである．

オブジェクトの数と状態は，プログラム実行中に存在するオブジェクトの数と，オブジェクトの状態を記録したフィードバック情報である．異なる実行環境において，プログラム内に存在するオブジェクトの数が大きく異なっていないことを確認したり，生成可能なオブジェクトの数の統計を取ることで，多くの実行環境で安定して動作するようなオブジェクト数の調整を行うための手助けになる．また，個々のオブジェクトの状態も記録しておくので，開発者が意図していない状態になっているオブジェクトを発見することができる．

ネットワークの応答速度は，クライアントからサーバへ何らかのデータを送信し，サーバで処理した結果が返ってくるまでの時間である．ネットワークを介したプログラムのパフォーマンスは，ネットワークの応答速度に依存するので，このフィードバック情報を収集して分析することはプログラムの性能向上の手助けとなる．また，ネットワークの状態は環境によって様々に変化するので，多くの実行環境でフィードバック情報を収集することが望ましい．

3.2 フィードバック情報収集機能の組み込み方法

3.1.3 節で定義したフィードバック情報を収集するためには，プログラム中のメソッド呼び出しを検知する必要がある．2 節で述べた通り，AspectJ を用いてメソッド呼び出しの前後に任意の処理を織り込むことが可能である．この任意の処理中で，フィードバック情報収集を行えば良い．この処理を織り込むためには，アスペクトを定義しなければならない．加えて，アスペクトを定義するためには，ジョインポイントの決定，ポイントカットおよびアドバイスの定義が必要である．そこで既存のソースコードを解析し，フィードバック情報収集機能を織り込む位置を決定すると共に，ポイントカット・アドバイス・アスペクトの定義を行うという流れになる．フィードバック情報収集機能の組み込み手順は次のようになる．

1. 組み込む機能を指定する
2. プロジェクトからジョインポイントを抽出する
3. アスペクトを生成する
4. アスペクトをプロジェクトに追加する
5. ビルドする

まず開発者は，組み込みたいフィードバック情報収集機能を選択する．選択肢は 3.1.3 節で定義した機能 3 つである．この選択は，プラグインが提供する GUI を通して行われる．また，フィードバック情報収集機能のカスタマイズ設定があれば，ここで行う．具体的には，機能を組み込む範囲 (特定のクラスだけ，特定のメソッドだけなど) の絞り込みである．

次に，プロジェクト内のソースコードから，フィードバック情報収集機能を組み込むために必要なジョインポイント情報を抽出する．例えば，す

べてのメソッド呼び出しに対してロギングを行う場合，プログラム内のすべてのメソッド呼び出し位置がジョインポイント情報として抽出される．

そして，ジョインポイント情報と組み込むフィードバック情報収集機能の種類から，アスペクトを生成する．生成されたアスペクトには，ポイントカットとアドバイスが定義されている．ポイントカットは機能を組み込む範囲が，アドバイスには機能のロジックコードが記述されている．

最後に，生成したアスペクトをプロジェクトに追加し，再ビルドを行う．これで，フィードバック情報収集機能付きのプログラムが生成される．

4 フレームワークの設計

本研究では，アスペクト指向技術をフィードバック情報収集機能の組み込みに利用する．また，開発者を支援するツールを Eclipse プラグインとして実装する．提案手法の目的は，対象とするプログラムにソースコードレベルでの独立性を保ったままフィードバック情報収集機能を付加することである．本研究で実装するツールは，次の 2 つの要求を満たしている．

- 既存プログラムへの組み込み・取り外しが容易
- 元のソースコードの修正は最小限

1 点目の要求は，フィードバック情報収集機能の特徴ゆえに発生するものである．1 章で述べたように，フィードバック情報収集機能はソフトウェア開発過程において一部の期間のみ利用されるものであるから，利用したいときに簡単に組み込み可能であり，必要ないときには修正の手間をかけずに取り外しできることが望ましい．2 点目の要求は，フィードバック情報収集機能は元プログラムの関心事ではないので，フィードバック情報収集機能を追加するために元プログラムを変更するのは不適当だからである．また，オブジェクト指向開発において，クラス間の関係や継承関係の変更，インタフェースの追加実装などの設計レベルでの変更はコストが大きいのも理由の一つである．

本研究で提案するフレームワークは，図 2 に示すアーキテクチャである．図中の (a) が実装した Eclipse プラグインである．(b) は Eclipse 本体，さらに (c) は Java プロジェクトである．(a) について，開発者 (Developer) は 3.1.3 節で定義したフィードバック情報収集機能のいずれかをウィザード形式で選択する．また，選択した機能によってはいくつかのオプションを選択する．ウィザードが終了したら，(a) 内の楕円部分の処理が行われる．まず，Java プロジェクト内のソースコードから，必要なジョインポイント情報を取得する (a-1)，次に，開発者が選択した機能の種類とジョインポイント情報を用いてアスペクトを生成する (a-2)．最後に，生成したアスペクトを Java プロジェクト内に追加し，ビルドを行う (a-3)．これでフィードバック情報収集機能付きのプログラムが生成される．生成されたアスペクトは純粋なソースコードなので，後で開発者が独自に修正を加えても良い．フィードバック情報収集機能付きプログラムは，指定したサーバへフィードバック情報を送信する仕様になっている．開発者はサーバへ送信されたフィードバック情報を参照してプログラムの修正を行う．

5 実験

本研究で開発したフレームワークを用いて，既存のプログラムにフィードバック情報収集機能を組み込んだ実験について述べる．まず組み込み対象と実験環境を説明する．次に組み込む手順を説明する．最後に，得られた結果を示す．

5.1 実験環境

実験対象として，以下の仕様を持つオンラインチャットプログラムを用意した．このプログラムにはサーバプログラムとクライアントプログラムがある．

- サーバプログラムを介してマルチチャットができる
- クライアント間で同期しながら図形を描画できる
- 任意のクライアント間でファイル送信ができる

このオンラインチャットプログラムに，本研究で実装した Eclipse プラグインを用いてオブジェクト数とネットワークの応答速度をフィードバックする機能を組み込んだ．組み込み後のプログラムの実行環境を表 1 に示す．

表 1: 実行環境

	サーバ	クライアント
マシン名	MacBook	iMac
CPU	C2D 2.0GHz	C2D 1.83GHz
メモリ	4GB	2GB
仮想マシン	JRE 1.5.0_16	JRE 1.5.0_16

5.2 オブジェクト数の計測

実験で用いたチャットプログラムは，クライアント間で同期して図形描画ができる．プログラム内では，Point オブジェクトを送受信して複数のクライアントに描画させている．そこで，図形が描画されたときに生成された Point オブジェクトの数を数えておくことにした．そうすることで，無駄なオブジェクトが生成されがちな図形を判断し，次バージョンで描画アルゴリズムを改善しようと考えた．今回対象とするのは，描画可能な図形全てである．円，塗りつぶし円，四角，塗りつぶし四角，直線，自由曲線の 6 種類の図形オブジェク

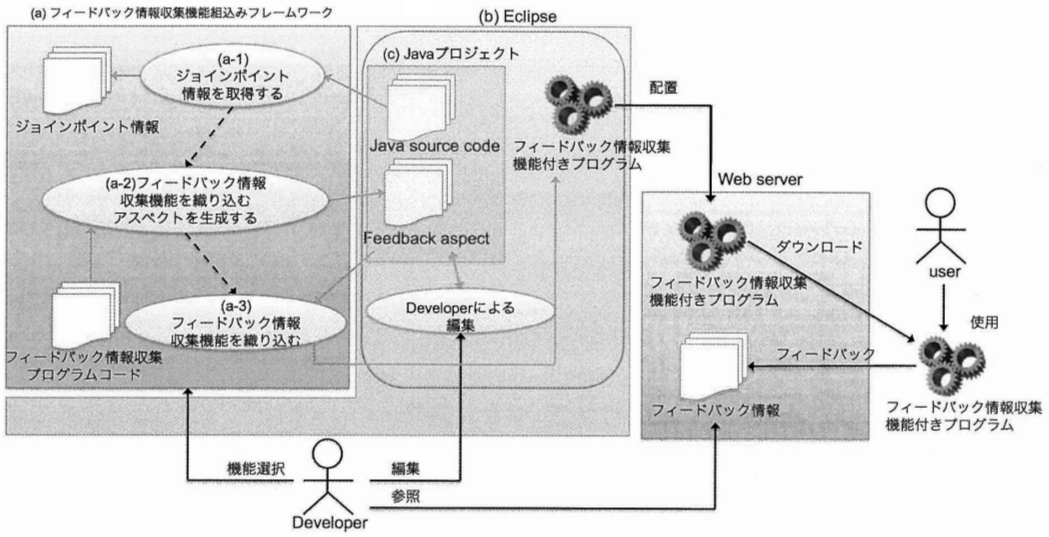

図 2: フレームワークの概要図

トが描画されたときに生成された Point オブジェクトの数を記録して報告する．

各図形オブジェクトの生成数を計測した結果を，表 2 に示す．自由曲線を描画する際に送信される Point オブジェクトの生成数が圧倒的に多く，他の図形を描画する際に新たに生成された Point オブジェクトの数は 0 である．これは，自由曲線を描画するときに，始点-終点間の途中点を記憶するために，新たな Point オブジェクトを生成していることが原因である．他の図形での Point オブジェクトの生成数が 0 である理由は，マウスをクリックしたときに，マウスの位置情報を保存しておく Point 型の変数 (startPos) が更新され，マウスを放したときは，マウスを放すイベントのイベントオブジェクトのフィールド変数からマウスの位置情報を得るため，新しく Point オブジェクトを生成しないからである．また，自由曲線を描画するとき，クライアントプログラムのキャンバス上で素早くマウスを動かすと，自由曲線の途中点がまばらになる現象が発生する．この現象はドラッグ中に途中点のみが描画されている場合にのみ起こり，マウスボタンを放すと途中点同士を直線で繋いで表現された自由曲線が描画される．しかしこの自由曲線は途中点同士の距離が離れていると，角ばった曲線になってしまう．これを改善する方法としては，途中点をその都度生成しないようにアルゴリズムを変更することなどが挙げられる．

5.3 ネットワークの応答速度

オンラインチャットプログラムは，ファイル送信機能を持つ．開発者は，大容量ファイルの送信に伴うパフォーマンスの低下を懸念し，平均的なファイル送信時間を計測したいと考えた．開発者はファイル送信を行っているメソッドを予め知っているので，そのメソッドのみを対象とした．今回対象となったメソッドは，データ送信用メソッドである．データ送信用メソッドの実行時間を計測してフィードバックすることで，ファイル送信にかかる平均の時間を調べる．

表 2: 図形描画時におけるオブジェクトの生成数

図形の種類	Point オブジェクトの生成数
自由曲線	629
直線	0
四角	0
塗りつぶし四角	0
円	0
塗りつぶし円	0

送信ファイル別の送信時間を表 3 に示す．概ね，現実的な時間でファイル送信が行われている．しかし，10 ギガバイト級のファイルを送信した場合は，メソッド呼び出し終了後の時刻が記録されていなかった．これは，writeObject メソッドが正常に終了しなかったことを示している．恐らく，オブジェクトを送信する途中で何らかの例外が発生したものを思われる．

6 関連研究

フィードバック情報収集のためのテクノロジとして，Microsoft 社製品に搭載されている WER(Windows Error Reporting)[5] や Jude[6] がある．これに関連して本研究では，フィードバック情報収集機能を効率的に組み込み・取り外し可能になる

表 3: ファイル送信にかかった時間

file size(MB)	送信時間 (msec)			
1	1002	930	994	845
10	4857	4676	4731	4431
100	29884	30198	31846	29667
1000	370473	345741	367920	374653
10000	-	-	-	-

ツールを提案している．本研究で開発したフレームワークと類似のプロダクトとしては，Smart Crash Reports[7] と ILCrashReporter[8] がある．Smart Crash Reports は MacOSX のクラッシュレポート機能を拡張したものである．Smart Crash Reports はアプリケーションへの導入は簡易だが，ユーザ環境にもインストールする必要がある．また，ILCrashReporter は，クラッシュレポート機能を付加するためのフレームワークである．ILCrashReporter はクラッシュレポート機能に特化しており，レポートの手段も電子メールのみである．加えて，ILCrashReporter は対象アプリケーションがクラッシュするのを検知して処理を行うものなので，機能をアプリケーションに組み込んでしまう本研究で提案したツールのアプローチとは異なる．類似研究として，アスペクト指向技術を用いて現行システムに大きな変更を加えずに稼動状態測定機能を組み込むことを目的としたフレームワーク [9] が研究されている．この研究では，組み込み対象のコンポーネントに特定のインタフェースを実装させることによって目的の機能を実現している．一方，本研究では，特別なインタフェースの実装などといったコストをかけずにフィードバック情報収集機能を組み込むことを目的としている．

7 おわりに

本研究では，フィードバック情報収集機能を効率的に付加可能にする Eclipse プラグインの開発を行った．

フィードバック情報収集機能を横断的関心事と捉え，アスペクト指向言語 AspectJ を用いてモジュール化した．このモジュール化はプラグイン内で自動的に行われるので，開発者が始めから行う必要は無い．

本研究で開発したプラグインとそれが生成するアスペクトは，既存プログラムに対する独立性の高さにより，既存の設計やソースコードへの修正を最小限に抑えられること可能である．

そしてオンラインチャットプログラムにフィードバック情報収集機能を実際に組み込み，ソースコードへの変更が最小限であることを確認した．またフィードバック情報としてメソッドの呼び出し履歴や実行時間，インスタンス生成数や，ネットワークの通信速度がフィードバック情報として送信されていることを確認した．

また，本研究に対する今後の課題を以下に示す．

- 収集可能なフィードバック情報の追加
  本研究で開発したフレームワークを用いてフィードバックとして収集可能な情報の種類を増やす．あるいは収集したい情報を開発者が追加・拡張可能な仕様に変更する．
- フィードバック情報の分析手法の確立
  大量に収集したフィードバック情報から対象プログラムの問題点を検出するための手法を定義する必要がある．まずは，フィードバック情報を体系的に整理するための手法の提案およびシステムの実装が求められる．
- より複雑な対象プログラムへの適用
  今回の研究で行った実験は，対象のプログラムが簡単なものであった．より複雑なプログラムに対しても同様の実験を行い，フィードバック情報収集機能の組み込み・取り外しの容易性を検証する．

参考文献

[1] 松本吉弘: ソフトウェアエンジニアリング基礎知識体系 -SWEBOK-, オーム社 (2003).

[2] Eclipse: http://www.eclipse.org/.

[3] Kiczales, G., Lamping, J., Menhdhekar, A., Maeda, C., Lopes, C., Loingtier, J.-M. and Irwin, J.: Aspect-Oriented Programming, *In Proceedings of European Conference on Object-Oriented Programming*, pp. 220–242 (1997).

[4] AspectJ: http://www.eclipse.org/aspectj/.

[5] WindowsErrorReporting: http://www.microsoft.com/japan/whdc/maintain/StartWER.mspx.

[6] Jude: http://jude.change-vision.com/jude-web/index.html.

[7] Smart Crash Reporter: http://www.smartcrashreports.com/.

[8] ILCrashReporter: http://www.infinite-loop.dk/developer/index.php.

[9] 楠和秦, 玉井哲雄: アスペクト指向技術と SNMP によるコンポーネント稼動状態測定フレームワーク, 情報処理学会研究報告, Vol. 2006, No. 75, pp. 1–6 (2006).