九州大学学術情報リポジトリ
Kyushu University Institutional Repository

Object Pascalプログラムに対するリファクタリング支援ツールの開発

秦野, 克彦
九州大学大学院システム情報科学府情報工学専攻 : 博士後期課程

乃村, 能成
九州大学大学院システム情報科学研究院情報工学部門

谷口, 秀夫
九州大学大学院システム情報科学研究院情報工学部門

https://doi.org/10.15017/1525462

出版情報：九州大学大学院システム情報科学紀要. 7 (2), pp.125-130, 2002-09-26. 九州大学大学院システム情報科学研究院
バージョン：
権利関係：

KYUSHU UNIVERSITY

九州大学大学院
システム情報科学紀要
第7巻　第2号　平成14年9月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.7, No.2, September 2002

Object Pascalプログラムに対する リファクタリング支援ツールの開発

秦野克彦* ・ 乃村能成** ・ 谷口秀夫**

Development of a Tool to Support Refactoring for Object Pascal Programs

Katsuhiko HATANO, Yoshinari NOMURA and Hideo TANIGUCHI

(Received June 14, 2002)

Abstract: Software flexibility and extendibility reflect program architecture. For example, it is difficult to add functions and to maintain programs because of their complexity. We need to improve a software carefully if we want to keep it in high quality. We can use "Refactoring" which is the technique to reorganize a program without changing its functions. Refactoring methods allow existing programs to change easily against future modification and maintenance. We have to find out where we apply refactoring and which refactoring method we should use. In this paper, we propose a tool to support refactoring for Object Pascal programs using software metrics.

Keywords: Refactoring, Software metrics, Software maintenance, Object Pascal

1. はじめに

要求に合わせて機能変更や機能拡張を行い，ソフトウェアは改版されていく．ソフトウェア開発や保守にかかる費用の半分以上が，機能変更と機能拡張への対応に充てられており[1]，ソフトウェアの機能変更や機能拡張の工数を少なくすることは非常に重要である．ところが，ソフトウェアを繰り返し改版していくと，設計当初のプログラム構造の統一性は崩れていくことが多い[2]．このため，機能変更や機能拡張の工数が増加してしまう．

そこで，ソフトウェア構造を見直すためにリファクタリングが有効である．リファクタリングとは，ソフトウェアが提供する機能を変更することなく，プログラムの内部構造を変更することである．このため，既存のプログラムにリファクタリングをうまく適用して，機能変更や機能拡張の工数を少なくすることができる．

しかし，リファクタリングを行うためには機能変更や機能拡張の工数の増加を招くプログラムの構造的欠陥を検出する必要がある．さらに，検出した構造的欠陥を解消する適切なリファクタリング手法を選択し施す必要がある．従来，こうした検出や選択は難しく，リファクタリングに関する知識や経験を必要とした．

そこで，本論文では，ソフトウェアメトリクスを利用したリファクタリング機構に基づく，Object Pascalプログラムに対するリファクタリング支援ツールについて述べる．ここで，ソフトウェアメトリクスとは，ソフトウェアの構造を数値化できる尺度のことである．ソフトウェアメトリクスを利用して構造的欠陥を検出することを支援し，構造的欠陥を解消する適切なリファクタリング手法を選択することを支援する．これらにより，作業者の知識や経験に大きく左右されることなくプログラムを改善することができる．

平成14年6月14日受付
* 情報工学専攻博士後期課程
** 情報工学部門

2. リファクタリングによるプログラムの再構成

2.1 従来のリファクタリング機構

従来から，リファクタリングを利用して，機能変更や機能拡張の工数削減を目指すことが行われている．この様子を**Fig. 1**に示す．**Fig. 1**は，既存プログラムをリファクタリングにより再構成する様子を示している．**Fig. 1**において，不具合の兆候とは，機能変更や機能拡張の工数の増加となるようなプログラムの構造的欠陥であり，何らかの原因を伴う．例えば，巨大なクラスが存在するという不具合の兆候には，クラスの責務が大きすぎるという原因がある．Fowlerらは，不具合の兆候を「不吉な匂い」と呼び，22個を示している[2]．リファクタリング手法とは，プログラムを再構成する操作である．例えば，メソッドの移動やクラスの抽出といったものがある．以降に，**Fig. 1**に基づき，リファクタリングを利用した従来の機構における作業手順を説明する．

(1) 既存のプログラムの構造を分析し，不具合の兆候を検出する．
(2) 不具合の兆候を生み出す原因を検討し，不具合の兆候を解消するリファクタリング手法を選択する．

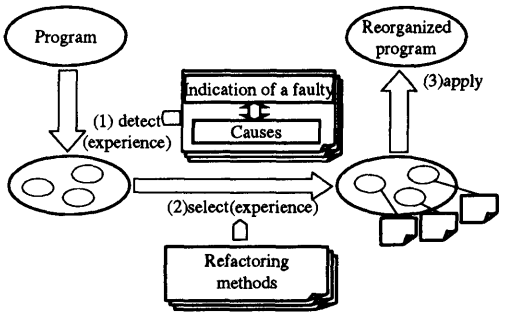

Fig.1 The conventional mechanism to reorganize existing program by refactoring

(3) リファクタリング手法を既存プログラムに適用し，プログラムを再構成する．

上記の手順において，手順(1)(2)が適切に行われるか否かは，作業者の知識や経験に大きく依存する．このため，再構成されたプログラムについて，リファクタリングがうまくできたとは限らない．つまり，従来のリファクタリング機構は，作業者の知識や経験に大きく依存するため，以下の問題がある．

(1) 不具合の兆候をうまく検出することは難しい．

(2) 不具合の兆候に対し，適切なリファクタリング手法を選択することは難しい．

2.2 ソフトウェアメトリクスを利用したリファクタリング機構

従来のリファクタリングの問題点を解決するため，ソフトウェアメトリクスを利用したリファクタリング機構を提案した[3]．ソフトウェアメトリクスとは，ソフトウェアの構造を数値化できる尺度であり，代表的なものとして，C&Kメトリクスがある[4]．提案した機構を**Fig. 2**に示す．**Fig. 2**では，「構造的欠陥検出基準」と「リファクタリング手法選択基準」という2つの知識を用意する．「構造的欠陥検出基準」とは，ソフトウェアメトリクスの測定値をもとに，当該プログラムの改善の要否を判定する基準である．「リファクタリング手法選択基準」とは，リファクタリング手法とソフトウェアメトリクスの値との増減関係である．つまり，リファクタリング手法をプログラムに適用したとき，ソフトウェアメトリクスによる測定値は，減少，増加，不変のいずれかであり，この関係を示すものである．以降に，**Fig. 2**に基づき，機構の作業手順を説明する．

(1) ソフトウェアメトリクスにより既存プログラムを数値化し，測定結果を得る．

(2) (1)で得た測定結果を利用し，「構造的欠陥検出基準」からプログラムの構造的欠陥を検出する．

(3) (1)で得た測定結果を利用し，「リファクタリング手法選択基準」から構造的欠陥を解消するリファクタリング手法を選択する．

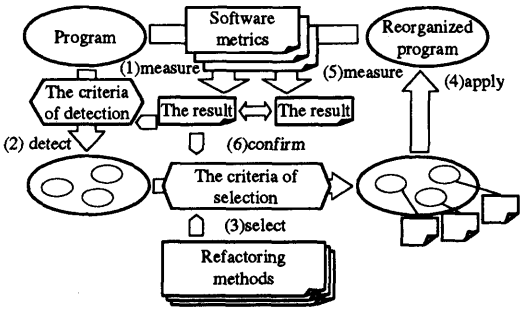

Fig.2 A mechanism to support refactoring using software metrics

(4) リファクタリング手法を既存プログラムに適用し，プログラムを再構成する．

(5) 再構成されたプログラムをソフトウェアメトリクスにより数値化する．

(6) ソフトウェアメトリクスの数値の変化により，リファクタリングによって構造的欠陥を解消できたことを確認する．

以上のことから，作業者の知識や経験に頼ることなくリファクタリングを行える．具体的には，「構造的欠陥検出基準」を用いることで，不具合の兆候を生む構造的欠陥を機械的に検出でき，「リファクタリング手法選択基準」を用いることでリファクタリング手法を同様に機械的に選択できる．また，ソフトウェアメトリクスを作業手順の最初と最後で利用することにより，プログラムの構造的欠陥を定量化でき，リファクタリングの効果を把握しやすくなっている．したがって，本機構により，先に示した従来の機構の問題を解消できる．

2.3 関連研究

リファクタリングに基づくeXtreme Programmingと呼ばれるソフトウェア開発手法が提唱されている[5]．この開発手法では，リファクタリングに関する十分な知識や経験を必要としている．

リファクタリングに関する初期の研究ではリファクタリング手法を適用する過程に焦点があてられており[6),7]，リファクタリング手法を適用する過程を自動化するツールついての研究がなされている[8]．これに対し本論文は，リファクタリング手法を適用するまでの過程に焦点をあてている点が異なる．

また，リファクタリング手法を適用するまでの過程をInvariantsを用いて支援する研究がなされている[9]．Invariantsとは，プログラム中で使用される変数間に暗黙的に存在する不変式のことである．プログラムが満たしておくべき固有の前提条件を各リファクタリング手法は持っている．プログラムからInvariantsを検出し，各リファクタリング手法固有の前提条件に一致するものを，

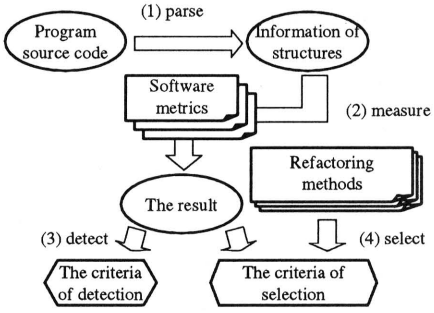

Fig.3 The action process

適用するべきリファクタリング手法として提示している．しかしながら，Invariantsの検出には，対象プログラムのソースコードに加えて，適切なテストケースを必要とする．

3. リファクタリング支援ツールの開発

3.1 開発目的

ソフトウェアメトリクスを利用したリファクタリング機構では，対象プログラムの規模が大きくなるほど，測定結果を得る工数が増加する．そのため，構造的欠陥を検出する工数やリファクタリング手法を選択する工数も増加する．そこで，工数削減のためにソフトウェアメトリクスによる測定を自動化し，構造的欠陥の検出とリファクタリング手法の選択を支援するツールを開発する．

3.2 実装内容

プログラミング言語Object Pascalで書かれたプログラムに対するリファクタリング支援ツールDelphiCKを開発した．本ツールは，**Fig. 2**に示した，(1)(5)測定，(2)検出，(3)選択に関わる機能を実現したものである．ソフトウェアメトリクスとしてC&Kメトリクス[4]，リファクタリング手法としてFowlerのリファクタリング手法[2]を用いている．構造的欠陥検出基準のために，C&Kメトリクスのしきい値を定めた．また，リファクタリング手法選択基準のために，Fowlerのリファクタリング手法とC&Kメトリクスの尺度について測定値の増減関係を調べた．本ツールの動作過程を**Fig. 3**に示し，動作を説明する．

(1) ツールは，Object Pascalプログラムのソースコードを構文解析し，測定に必要な情報を持つプログラムの構造情報を生成する．

(2) 生成されたプログラムの構造情報を対象に，C&Kメトリクスに基づいて測定結果を得る．C&Kメトリクスは，複雑度を表す6つの尺度から成る．これら6つの尺度はクラスを測定単位としているので，1つのクラスに対して6つの測定値が定まる．測定値は0以上の整数値を取り，値が大きなほどクラスが複雑であることを意味する．

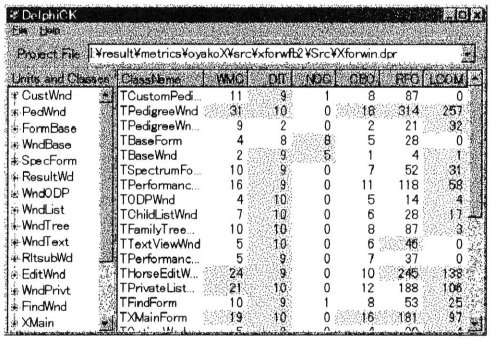

Fig.4 The detected result of structural faults

(3) C&Kメトリクスによる測定値は，値が大きいほど改善を必要とすることを意味する．そのため，定めたしきい値よりも大きな測定値の場合，その測定値を構造的欠陥とみなすことができる．これが構造的欠陥検出基準である．測定結果を利用して，構造的欠陥検出基準から構造的欠陥を検出する．

(4) プログラムに対するリファクタリング手法の適用はソフトウェアメトリクスによる測定値を増減させる．そこで，Fowlerリファクタリング手法とC&Kメトリクスの6つの尺度との測定値の増減関係を調べた．測定値が大きなほど構造的欠陥とみなすため，検出された構造的欠陥の測定値を減少させるリファクタリング手法が，選択すべきリファクタリング手法とみなせる．これがリファクタリング手法選択基準である．測定結果を利用して，リファクタリング手法選択基準から構造的欠陥を解消するFowlerのリファクタリング手法を選択する．

3.3 リファクタリングを支援する機能

3.3.1 構造的欠陥検出機能

本ツールは，構造的欠陥検出基準を利用して，構造的欠陥を検出する機能を持つ．C&Kメトリクスによる測定値は，値が大きなほど改善を必要とすることを意味する．そのため，定めたしきい値よりも大きな測定値を取る箇所を構造的欠陥とみなしている．この機能の実行結果を**Fig. 4**に示す．**Fig. 4**はプログラムに対して構造的欠陥を検出した結果である．本ツールは，測定対象のプログラムを構成するファイル群を左側に列挙する．そして，C&Kメトリクスによるプログラムの測定結果を右側に列挙する．測定結果を画面上に出力する際，不具合箇所を示す数値に対して色を付けて区別する．**Fig. 4**では，網かけになっている部分が構造的欠陥であることを示している．

3.3.2 構造的欠陥改善順序の提示機能

複数の構造的欠陥を検出した場合，どの構造的欠陥から改善すればよいかわかれば便利である．そこで本ツールは，構造的欠陥の改善順序を提示する機能を持つ．こ

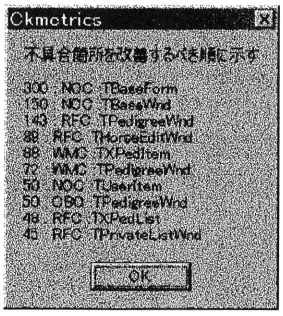

Fig.5　The recommended order that should improve structural faults

の機能を実行した結果を**Fig. 5**に示す．**Fig. 5**は，検出された構造的欠陥を改善するべき順に提示している．例えば，300:NOC:TBaseFormは，改善優先度300で尺度NOCについてクラスTBaseFormを改善するべきことを示す．

本ツールは，構造的欠陥改善順序を示すために改善優先度を用いている．改善優先度は値が大きなほど，優先的に改善するべきことを示す．そのため，改善優先度の値の大きいものから順に示す．(1)式で，$P_{c,m}$は改善優先度を意味し，cはクラスを意味し，mはC&Kメトリクスの尺度であるWMC，DIT，NOC，CBO，RFC，およびLCOMのうちのどれかを意味する．$V_{c,m}$はクラスcに対する尺度mについての測定値である．t_mは尺度mのしきい値である．

$$P_{c,m} \stackrel{\text{def}}{=} \begin{cases} V_{c,m} & (t_m = 0 \text{ の時}) \\ \frac{V_{c,m}-t_m}{t_m} & (\text{それ以外の時}) \end{cases} \tag{1}$$

3.3.3　リファクタリング手法選択機能

本ツールは，リファクタリング手法選択基準を利用して，Fowlerのリファクタリング手法から構造的欠陥を解消するリファクタリング手法を選択する機能を持つ．本ツールは，測定値が大きなほど構造的欠陥とみなす．そのため，検出された構造的欠陥の測定値を減少させるリファクタリング手法を提示する．この機能の実行結果を**Fig. 6**に示す．**Fig. 6** は，クラスTBaseFromの尺度NOCに対する構造的欠陥に対して，これを解消するリファクタリング手法を選択した結果である．クラスTBaseFormのNOCの値を減少させるリファクタリング手法として，階層の平坦化とフィールドによるサブクラスの置き換えを本ツールは提示している．

3.3.4　リファクタリング手法の解説機能

リファクタリング手法を選択した場合，提示されたリファクタリング手法の詳細が確認できれば便利である．そこで本ツールは，提示されるリファクタリング手法に

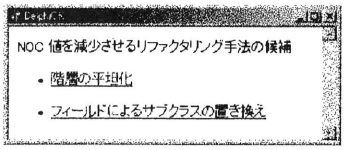

Fig.6　The selected result of refactoring methods

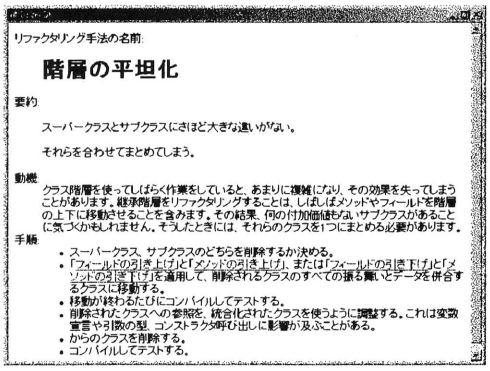

Fig.7　Details of a refactoring method

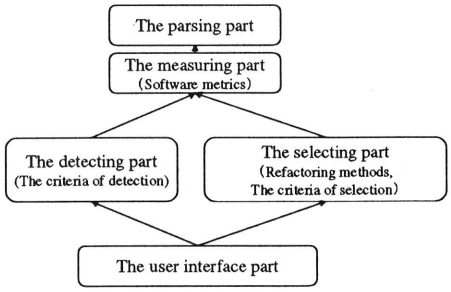

Fig.8　モジュール構成

ハイパーリンクを施した．ハイパーリンクを辿ることで，関連するリファクタリング手法の解説を閲覧できる機能を持つ．この機能を実行した結果を**Fig. 7**に示す．**Fig. 7**は，階層の平坦化というリファクタリング手法の解説を閲覧している．解説はリファクタリング手法に対する，名前，要約，動機，そして手順である．

3.4　モジュール構成

本ツールのモジュール構成を**Fig. 8**に示し，各モジュールの機能と役割を説明する．

構文解析部　構文解析部は，プログラムの構造情報に対するアクセスインターフェースを定義する．構文解析器を通じてプログラムのソースコードからプログラムの構造情報を抽出することにより，記述されたプログラミング言語の差を吸収する．

プログラム測定部　プログラム測定部は，ソフトウェアメトリクスによる測定結果に対するアクセスインターフェースを定義する．採用したソフトウェアメトリクスに従ってプログラムの構造情報を数値化す

ることにより，測定結果を生成する．

構造的欠陥検出部 構造的欠陥検出部は，構造的欠陥検出基準を定義する．さらに，構造的欠陥検出基準に従って，測定結果の数値群を構造的欠陥か否か判定し，ユーザーインタフェース部に通知する．

リファクタリング手法選択部 リファクタリング手法選択部は，リファクタリング手法選択基準を定義する．ここで，リファクタリング手法をプログラムに適用したとき，ソフトウェアメトリクスによる測定値を増減させる関係にある．リファクタリング手法選択基準に従って，この増減関係からリファクタリング手法を選択し，ユーザーインタフェース部に通知する．

ユーザーインタフェース部 ユーザーインターフェース部は，構造的欠陥検出部とリファクタリング手法選択部に問い合わせ，問い合わせ結果をユーザに示す．

これらモジュールにおいて，プログラミング言語に依存する部分を構文解析部に局所化し，また，ソフトウェアメトリクスに関わる部分をプログラム測定部に局所化した．これにより，次の2つのことを実現できた．

(1) 記述されたプログラミング言語に依らず，構造的欠陥を検出可能で，構造的欠陥を解消するリファクタリング手法を選択可能．

(2) 測定に用いるソフトウェアメトリクスに様々なものを採用可能．

4. 性 能 評 価

本ツールの性能評価を行うために，3つのObject Pascalプログラム(A:3070行,B:5850行,C:9549行)に対して(1)構文解析に要する時間，(2)測定に要する時間，(3)検出に要する時間，(4)選択に要する時間を測定した．測定には，PentiumIII 1000MHzで主記憶512MBを持つ計算機を使用した．この結果を**Table 1**に示す．(1)構文解析，(2)測定，(3)検出，について，3つのプログラムの規模が大きくなるほど処理時間を要している．しかし，最大規模のプログラムCに対して5秒ほどで処理を終えており，総じて十分短い時間内で処理を終えていると言える．これにより，本ツールを使用することでリファクタリングに関する十分な支援を期待できる．なお，**Table 1**の中の0.00という値は，処理時間が短すぎて有意な値を得られなかったことを意味している．

5. お わ り に

本論文では，Object Pascalプログラムに対するリファクタリング支援ツールの開発について述べた．本ツールにより，作業者は，構造的欠陥の検出やリファクタリング手法の選択に手間をかけることなくリファクタリング作業を行える．

Table 1 The processing time(sec) for three programs

Program	A	B	C
(1)parse	1.30	1.51	1.52
(2)measure	1.31	2.13	3.41
(3)detect	0.10	0.11	0.11
(4)select	0.00	0.00	0.00
Total	2.61	3.75	5.04

従来のリファクタリング作業では，作業者は構造的欠陥を定量的に把握しておらず，作業の成否は作業者の知識や経験に依存するものであった．このため，プログラムの構造的欠陥の箇所を検出することが難しく，構造的欠陥を解消する適切なリファクタリング手法を選択することが難しい，という問題がある．そして，この問題を解消するソフトウェアメトリクスを利用したリファクタリング機構について述べた．しかしながら，ソフトウェアメトリクスによる対象プログラムの測定は，対象プログラムの規模に応じて工数が増加するという問題がある．そこで，工数削減のためにソフトウェアメトリクスによる測定を自動化し，構造的欠陥の検出とリファクタリング手法の選択を支援するツールDelphiCKを開発した．

DelphiCKは，Object Pascal言語で記述されたプログラムを対象とするリファクタリング支援ツールである．本ツールは対象プログラムの測定にC&Kメトリクスを用いて構造的欠陥を検出する．そして，構造的欠陥を解消するFowlerのリファクタリング手法を提示することで，リファクタリング手法の選択を支援する．また，構造的欠陥の改善順序を提示する機能およびFolwerのリファクタリング手法を解説する機能を持つ．さらに，本ツールのモジュール構成を述べた．本ツールは，プログラミング言語に依存する部分と，ソフトウェアメトリクスに関わる部分をそれぞれ独立したモジュールとして持つ．こうしたモジュール構成により，記述されたプログラミング言語に依らずリファクタリング作業を支援することができ，測定に用いるソフトウェアメトリクスに様々なものを採用することができる．

最後に，3つのObject Pascalプログラムを対象に本ツールの性能評価を行った．対象プログラムの規模が大きくなるほど処理時間を要しているが，約9500行のプログラムに対して全ての処理を約5秒で終えており，総じて十分短い時間内で処理を終えていることがわかった．残された課題としては，より有効なリファクタリング手法の適用を可能にするために，適用するリファクタリング手法を選択する規則の検討と実装がある．

参 考 文 献

1) Horowitz and M.Barry: Strategic Buying for the Future, Washington DC: Libey Publishing(1993).
2) M.Fowler: Refactoring: Improving The Design of Existing Code(1999), 児玉公信, 友野晶夫, 平沢章, 梅沢真史訳: リファクタリング: プログラムの体質改善テクニック, ピアソン・エデュケーション,(2000).
3) K.Hatano, Y.Nomura, H.Taniguchi, K.Ushijima: A Method to Support Refactoring Using C&K Metrics, Proceedings of Pan-Yellow-Sea International Workshop on Information Technologies for Network Era, pp.112–118(2002.3)
4) S.R.Chidamber and C.F.Kemerer: A Metrics Suite for Object Oriented Design, IEEE Transaction on Software Engineering, Vol.20, No.6, pp.476–493(1994).
5) K.Beck: Extreme Programming Explained: Embrace Change, Addison-Wesley(1999).
6) W.F.Opdyke: Refactoring Object-Oriented Frameworks, Ph.D. diss., University of Illinois at Urbana-Champaign(1992).
7) W.F.Opdyke and R.E.Johnson: Refactoring: An aid in designing application frameworks and evolving object-oriented systems, Proceedings of SOOPPA '90: Symposium on Object-Oriented Programming Emphasizing Practical Applications, September(1990).
8) D.Roberts, J.Brant and R.E.Johnson: A Refactoring Tool for Smalltalk, Journal of Theory and Practice of Object Systems(TAPOS), Vol.3, No.4, pp.39–42(1997).
9) Y.Kataoka, M.D.Ernest, W.G.Griswold and D.Notkin: Automated Support for Program Refactoring using Invariants, Proceedings of International Conference on Software Maintenance (ICSM'01), pp.736–743(2001).