Institute of Electronics, Information, and Communication Engineers

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

信学技報
IEICE Technical Report
SS2009-13 (2009-8)

リバースエンジニアリングツールキットRemicsの試作

神谷 年洋†

†産業技術総合研究所 サービス工学研究センター 最適化研究チーム
〒101-0021 東京都千代田区外神田1-18-13 秋葉原ダイビル

E-mail: †t-kamiya@aist.go.jp

あらまし リバースエンジニアリングツールのためのツールキット Remics の現状について説明する．このツールキットは，開発者が自らのプロダクトを分析するためのツールキット作成を行うことを目標としたものであり，開発の速度およびカスタマイズ性を高めるために，いわゆるP言語のひとつであるPythonにより記述されたコンパクトなスクリプトの集まりとして実装されていて，それぞれのスクリプトはたかだか数百行の規模となっている．本稿ではさらに，ツールキットの利用例として，簡単な依存関係分析ツールを実装した例を示す．

キーワード プログラム解析，リバースエンジニアリング，ツールキット，人間中心の開発

A Prototype Implementation of Reverse-Engineering Toolkit: Remics

Toshihiro Kamiya†

† Optimized Design Research Team, Center for Service Research
National Institute of Advanced Industrial Science and Technology
Akihabara Dai Bldg. 1-18-13 Sotokanda, Chiyoda-ku, Tokyo, 101-0021, JAPAN

E-mail: †t-kamiya@asit.go.jp

Abstract This paper briefly introduces the current status of a toolkit named Remics. This toolkit aims that software developers make their own analysis tools for their product in an on-the-fly way in development processes. This toolkit is implemented as a set of compact scripts written in Python (one of P-languages), in which size of each script is up to hundred lines of code, in order to enhance development speed and customizability of the tools which are developed with this toolkit. Also, a sample dependency-analysis tool is shown as an example use of the toolkit.

Keyword Program Analysis, Reverse Engineering, Toolkit, Human-Centric Development

1. はじめに

本稿では，リバースエンジニアリングツールを開発するためのツールキットRemics [12]の現状について説明する．

著者が現在取り組んでいるRemixプロジェクト(科研費萌芽，平成21年度から3年間の予定)では，

(1) ソフトウェアの開発者が扱う様々な文書，たとえば，開発中のプロダクト(の現在のバージョン)，その過去のバージョン，関係のある他のプロダクトを，
(2) 適切な検索やマイニングのためのアルゴリズムおよび可視化手法を利用して，
(3) 開発者自身が自ら，開発プロセス中で必要となった時点で自らの必要に応じて，

分析するためのツールを自作できるようなツールキットを開発し提供することを目標としている．Remics はそのツールキットの名称である．

以降，2では，本ツールキットの現状の概略を説明する．3では分析ツールを実装する具体例を示す．4 では関連する研究・ツールを紹介する．5でまとめと課題を述べる．

2. ツールキット Remics

リバースエンジニアリングツールのためのツールキットRemics[12]は，開発者が扱う種々の文書を，文書内の文字列およびその間の関係(2項関係など)を用いてモデル化する(次ページ図1)．

ツールキット利用者が本ツールキットを容易に理解しカスタマイズできるように，ツールキットの実装技術として P 言語のひとつであるPythonを利用した．すなわち，ツールキットを，さまざまなアルゴリズム・データ構造が利用可能な(標準ライブラリや多数のサードパーティのライブラリとして提供されている)汎用スクリプト言語 Python によって記述された，相互運用可能およびカスタマイズ(修正や拡張)が容易な塊魄なスクリプトの集まりとして実装した．同時に，利用者によるカスタマイズを妨げないために， OSI(Open Source Initiative, http://www.opensource.org/) 認定のオープンソースライセンスのひとつであるMITライセンスの元に配布している[12]．

Copyright ©2009 by IEICE
NII-Electronic Library Service

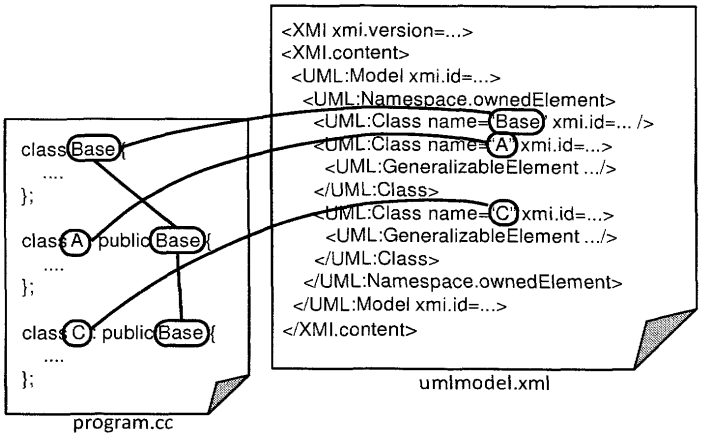

図 1 Remics の「テキストファイル中の部分文字列間の関係」モデル

2.1. Origined String

Origined string とは，前述のモデルを表現するためのデータ構造として提供されている，Remics のコンポーネントである[6]. Origined string は「自身がどこから来たかを知っている文字列」であり，通常の文字列オブジェクトとじ操作（部分文字列の生成，連結など）に加えて，その origined string オブジェクトを生成する際に用いられたテキストファイルの名前やそのテキストファイル内での位置（行，列）を問い合わせる操作を備えている．規模は，（サンプルなどを除いた）コア機能だけであれば計 700 行程度である．

Origined string は上述の「テキストファイル中の部分文字列間の関係」モデルの操作可能な表現として利用することができる．たとえば，任意のテキストファイルの一部（である文字列）の間の 2 項関係は，origined string オブジェクトのペアとして表現することができる（図 2）．

Origined string を利用することで，分析ツール開発者は，分析結果を origined string を含むデータ構造として表現することができる．これにより，たとえば，分析結果をソースコード上にオーバーラップして表示する必要があったときには，データ構造中の origined string にそのソースコード上の位置を問い合わせることができる．結果として，分析結果とソースコード上の位置の対応を管理する処理を，分析ツール開発者が記述する必要がなくなるとともに，分析ツールのソースコードにおいて，管理のためのコードが分析の他の処理のコードと絡んでしまって理解容易性が下がる，といった状況を避けることができる．

Origined string は文字列でもある（文字列オブジェクトとしてのメソッドも備えている）ため，文字列のためのアルゴリズムを実装するライブラリ，たとえば正規表現ライブラリと相互運用することができる．正規表現ライブラリに対して，origined string のインスタンスを，あたかも通常の文字列であるかのように渡すことでマッチングを行い，結果として得られた（すなわち，表現にマッチした）origined string のインスタンスに対して，その位置を問い合わせる，といった使い方ができる．

2.2. Interactive Graph Viewer (仮称)

Remics のコンポーネントのひとつとして，特に頂点や辺を多数含むようなグラフを表示することを想定した，汎用的なグラフビューアである Interactive Graph Viewer を開発中である（画面写真は 2 ページ先の図 5 を参照）．グラフ視覚化ツール GraphViz[4]によりレイアウトされた有向グラフを入力として与えると，そのグラフを画面上に図として表示し，画面上で対話的に操作することができる．具体的には，グラフの頂点をドラッグして移動したり，指定した頂点につながる（頂点に入る・から出る）辺を非表示にする操作を持つ．

規模は，動作のために最小限必要なコア機能がファイル数 5 個，計 341 行であり，補助的な機能がファイル数 2 個，計 120 行，GUI デザイナ（開発環境 SharpDevelop に組み込みのもの，http://www.icsharpcode.net/OpenSource/SD/）で生成された（テキストエディタで編集されていない）ソースコードがファイル数 3 個，計 136 行であり，全体としても 600 行以内に収まっている．

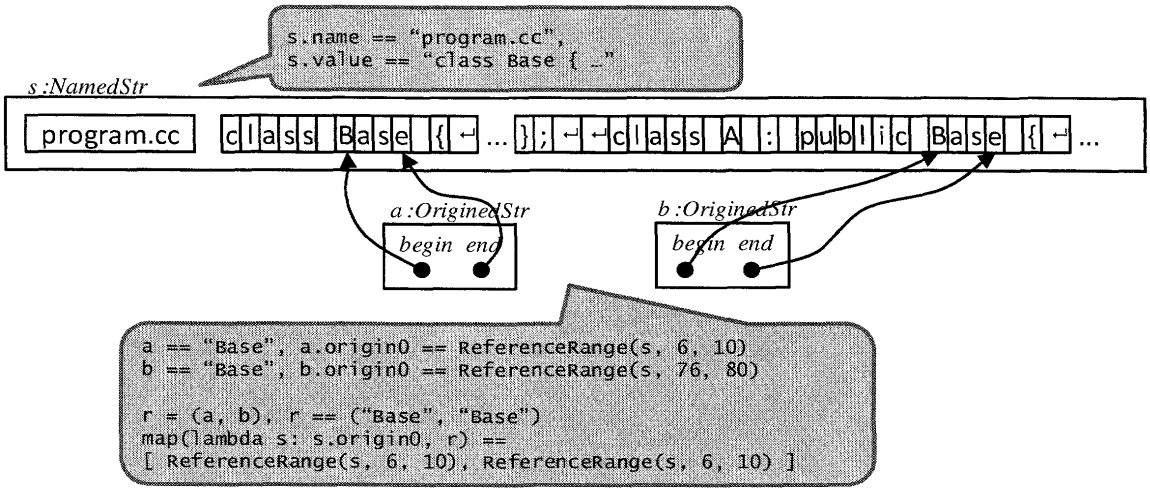

図 2 Origined String

NII-Electronic Library Service

Institute of Electronics, Information, and Communication Engineers

3. 分析ツールの実装例 depgraphcpp

ツールキット Remics を利用して，C++で記述されたソースコードの依存関係を分析するためのツール depgraphcpp を作成した．このツールは，クラス定義・利用関係を依存関係として，C++のソースファイル間の依存関係を示すグラフを生成する(図 3)．Depgraphcpp によって生成されたグラフは，Graphviz を用いて画像ファイルにすることも，Interactive Graph Viewer を用いて表示することも可能である．

規模はファイル数で 2 個，計 87 行である．ソースファイルのうち 1 つは，既発表の分析ツール depgrah[6](Python で記述されたソースコードを分析する)を修正および拡張して作成した．もう 1 つは power graph[13]の概念を援用してグラフの頂点・辺をマージすることで，グラフに含まれる情報量を減らすことなく，見かけ上グラフに含まれる要素の数を減らすためのユーティリティであり，このツールの開発中に行った予備的な実験の後で追加された．ツール depgraphcpp は，

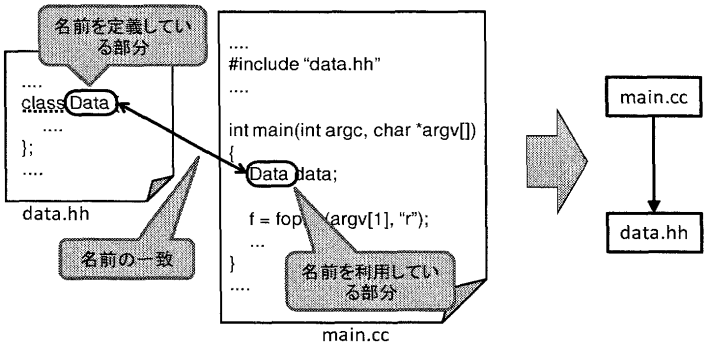

図 3 ツール depgraphcpp の処理の概略

Remics のサンプルとして配布物に同梱される予定である．

このツールに著者が開発したソフトウェアである CCFinderX (http://www.ccfinder.net/)を適用した例を図 4 に示す．分析対象となったソースファイルのうち，C++で記述されたヘッダファイル(ファイル数 54 個，計 2 万 4 千行)

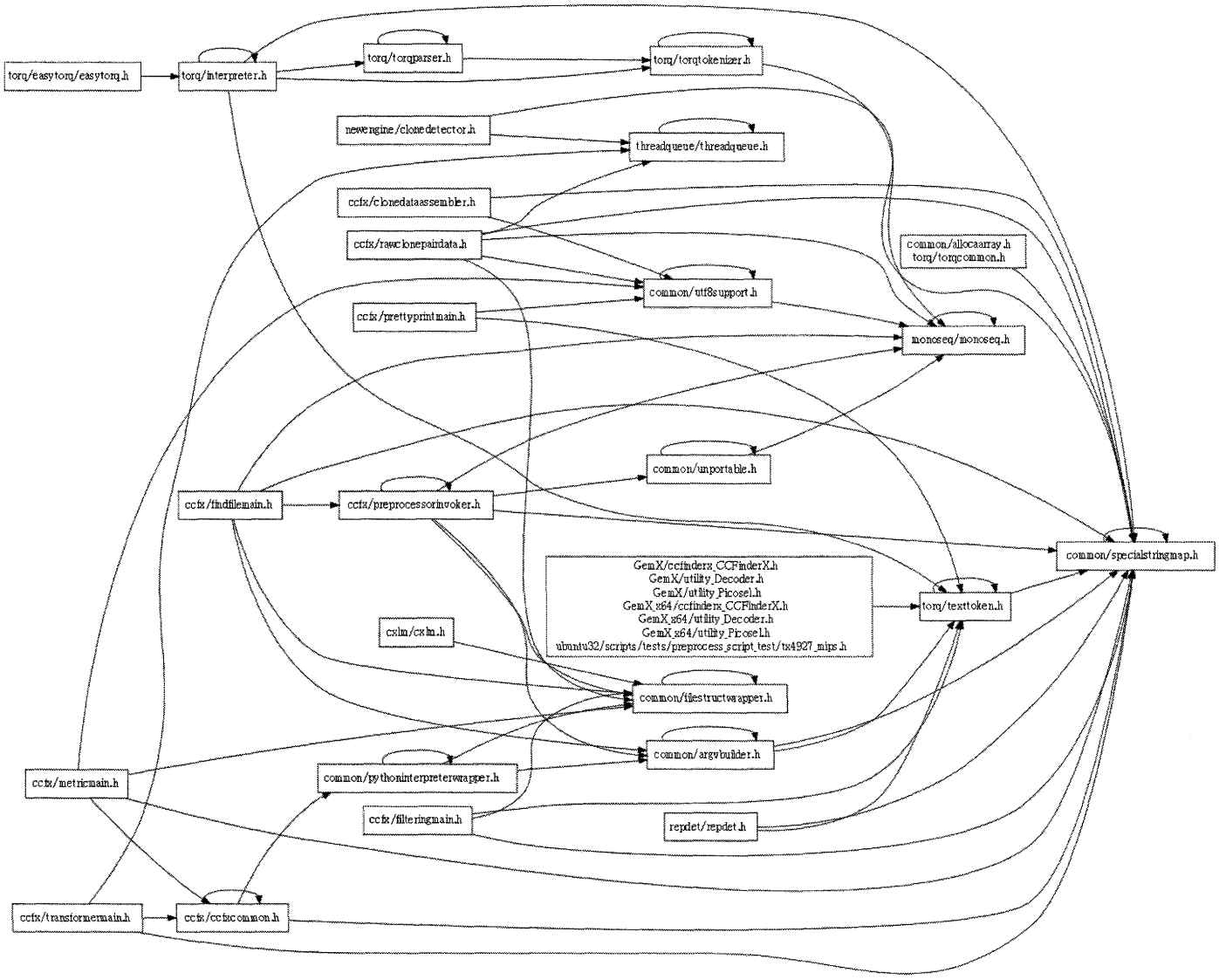

図 4 ツール depgraphcpp の適用例

NII-Electronic Library Service

Institute of Electronics, Information, and Communication Engineers

を入力とした．図中の一番大きな頂点(矩形)に 6 つのソースファイル名が記入されているのは，それらのソースファイルが同じ依存関係を持っていたために，power graph のアルゴリズムにより 1 つの頂点にまとめられたことを示している．

この例により，ツールの入力となったソースコードは小規模(数万行)であっても，依存関係のグラフを読み取るのは容易ではない場合があることが分かる．出力となったグラフの図に含まれる頂点はたかだか数十個であり，power graph アルゴリズムを用いて頂点をまとめたり，依存関係の原因になっている名前を表示しないようにしたり，ソースファイル中のヘッダファイルのみを解析の対象とする，という努力を行っているにもかかわらず，出力されたグラフの図から，全体的な構造を読み取ることは難しい．原因のひとつは，汎用的な機能のコードが紛れ込んでいることであると分かった．すなわち，入力となったソースファイルの中に，ユーティリティクラス，すなわち，アプリケーションに特化していない汎用のクラスを定義するソースファイル(図の右端の common/specialstringmap.hpp など)が存在していて，それらユーティリティクラスが多数のソースファイルから利用されているために，ユーティリティクラスに向かう数多くの辺がグラフに含まれることになった．(ユーティリティクラスは汎用的な機能を実装するため，ユーティリティクラスへの依存関係は，依存しているソースファイルを特徴づける情報とは見なされにくい．)

次に，このグラフを Interactive Graph Viewer により表示し，対話的なグラフ編集機能により，前述のユーティリティクラスに関連する辺を非表示にし，さらに頂点を移動した(図5)．分析対象のソフトウェアのおおよその構造を見て取ることができる．

4. 関連研究・ツール

文献[14]では，開発者が開発プロセス中に発する問い(調べる必要がある事柄)を分類し，Focus Point という概念を導入してこれらの問いをモデル化することを提案している．文献[16]では，プログラム中の依存関係を理解することがソフトウェアを理解し保守する際の鍵であると断じ，様々な依存関係を用いることでソフトウェアを理解する例を示している．文献[5]では，データ依存および制御依存関係を用いて，ソースコードから関心事を抽出する手法を提案し，実装したツールを定量的に評価している．

文献[1][3][7][8][15]では，C や Java といったプログラミング言語で記述されたソースコードを読み込んで，プログラミング言語の文法およびセマンティクスを取り込んだモデル(をもつデータ構造)に変換する手法が提案されている．これらとは対照的に，Remics では，異なった種類の文書(ソースコード以外のもの)を扱うことを目標のひとつとしているため，特定の言語のセマンティクスをモデルに取り込んでいない．

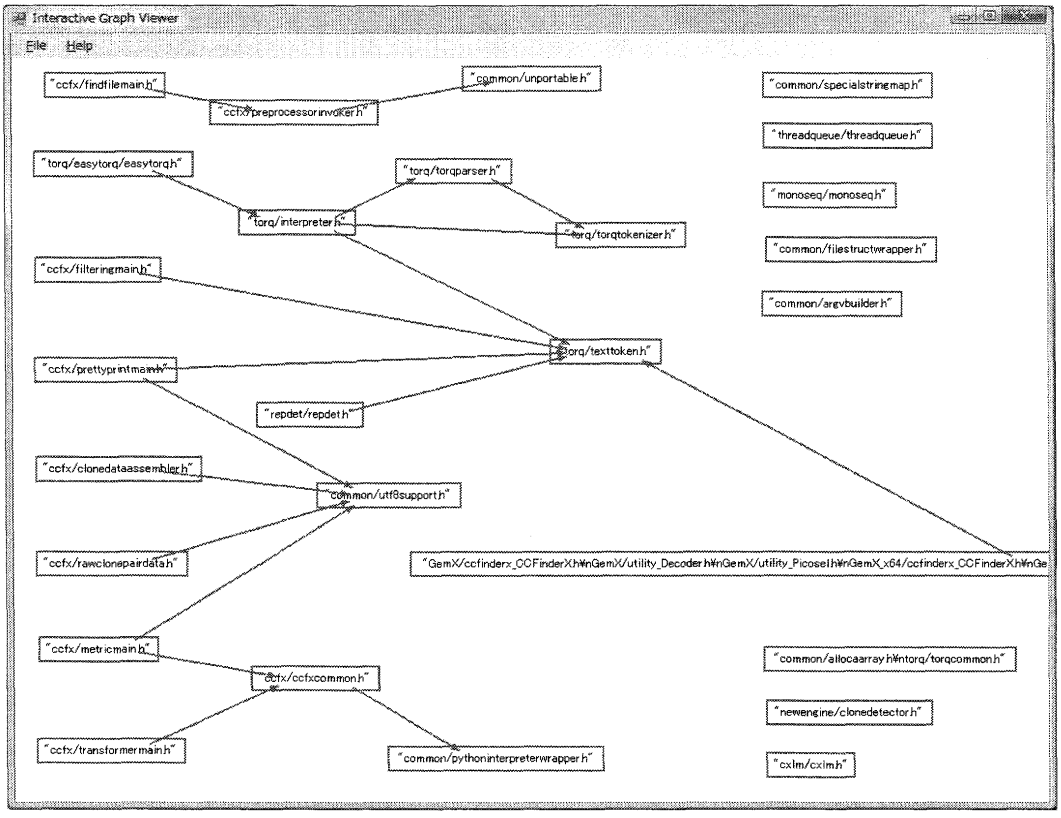

図 5 Interactive Graph Viewer により整理したグラフ

NII-Electronic Library Service

Institute of Electronics, Information, and Communication Engineers

文献[2]では．ソースコードを表現するためのデータ構造として，平文テキスト，シンボルテーブルと AST の組み合わせ，XML の 3 者を比較している．

GraphViz[4]はグラフを視覚化するためのツールである．Dot というフォーマットにより記述されたグラフを入力し，頂点や辺をレイアウトした図を，画像ファイル，SVG(Scalable Vector Graphics)ファイルなどの様々なフォーマットで出力する．多種類のレイアウトアルゴリズム(hierarchical, spring model, radial, circular)を備えていることが特徴である．OSI 認定オープンソースライセンスのひとつである CPL により配布されている．Python-graph [11]はグラフを解析するための多種類のアルゴリズム(critical path, cycle detection など)を提供するライブラリである．Python で記述され，MIT ライセンスにより配布されている．Remics は GraphViz および Python-graph と相互運用可能(技術的にもライセンス的にも)であり，これらをライブラリとして利用している．

Pajek[9][10]は汎用グラフ分析ツールである．大規模なグラフを対象に，ネットワーク分析アルゴリズムを適用し，視覚化する機能を持つ．配布はバイナリのみで行われている．

5. まとめと課題

本稿では，リバースエンジニアリングツールを開発するためのツールキット Remics の現状について，その目的や設計方針，現在提供されている，あるいは近日提供予定のコンポーネントを説明した．特に，ツールキットがコンパクトな(たかだか数百行規模の)Python スクリプト群によって記述されていること，既存のライブラリと相互運用されていることを述べた．また，ツールキットの利用例として，簡単な依存関係分析ツールを実装することで，このツールキットがどのように利用されるかを示した．

ツールキット Remics は開発の途についたばかりであり，今後コンポーネントの拡充および利用例・評価の積み重ねが必要である．

ごく短期的な目標としては，以下の点を強化する予定である．

- 既存のコードクローン検出ツール CCFinderX との相互運用性の確保
- データマイニングアルゴリズムの強化(コンポーネントとして用意するか，あるいは既存のライブラリとの相互運用性の確保)

謝辞

本研究は日本学術振興会 科学研究費補助金（挑戦的萌芽研究)(研究課題番号: 21650008)の助成を受けたものである．

文　献

[1] Michael L. Collard, “Addressing Source Code Using srcML”, IEEE International Workshop on Program Comprehension Working Session: Textual Views of Source Code to Support Comprehension (IWPC'05), taken from http://www.sdml.info/collard/papers/iwpcwspos05.pdf (2005).

[2] Michael L. Collard, Jonathan I. Maletic, and Andrian Marcus, “Supporting Document and Data Views of Source Code”, Proceedings of the 2002 ACM symposium on Document engineering, pp. 34-41 (2002).

[3] 福安 直樹, 山本 晋一郎, 阿草 清滋, “細粒度ソフトウェア・リポジトリに基づいた CASE ツール・プラットフォーム Sapid”, 情報処理学会論文誌, Vol.39, No.6, pp.1990-1998 (1998)

[4] GraphViz, http://www.graphviz.org/.

[5] 石尾 隆, 仁井谷 竜介, 井上 克郎, “プログラムスライシングを用いた機能的関心事の抽出”,コンピュータソフトウェア, Vol.26, No.2, pp.127-146 (2009).

[6] Toshihiro Kamiya, “Programmable Queries, or a New Design of Search Tools”, Proc. the 1st Int'l. Workshop on Search-driven development: Users, Infrastructure, Tools and Evaluation (Suite 2009), co-located with ICSE 2009, Vancouver, Canada (May 2009).

[7] Katsuhisa Maruyama and Shinichiro Yamamoto, “A CASE Tool Platform Using an XML Representation of Java Source Code”, 4th IEEE International Workshop on Source Code Analysis and Manipulation (SCAM'04), pp.158-167 (2004).

[8] E. Mamas and K. Kontogiannis, “Towards Portable Source Code Representations Using XML”, Proceedings of the Seventh Working Conference on Reverse Engineering (WCRE'00), p.172-182 (November, 2000).

[9] Wouter de Nooy, Andrej Mrvar and Vladimir Batagelj, *Exploratory Social Network Analysis with Pajek*, Cambridge University Press (2005).

[10] Pajek Wiki, http://pajek.imfm.si/.

[11] Python-grpah, http://code.google.com/p/python-graph/.

[12] Remics.org, http://www.remics.org/

[13] Loïc Royer, Matthias Reimann, Bill Andreopoulos, and Michael Schroeder, “Unraveling protein networks with Power Graph Analysis”, PLoS Comput. Biol. 2008 July; 4(7). http://www.ploscompbiol.org/article/info:doi/10.1371/journal.pcbi.1000108 (2008).

[14] Jonathan Sillito and Gail C. Murphy, “Asking and Answering Questions during a Program Change Task”, IEEE Trans. Software Engineering, vol. 34, no. 4, July/August 2008.

[15] srcML, http://www.sdml.info/projects/srcml/.

[16] Zhifeng Yu and Václav Rajlich, “Hidden Dependencies in Program Comprehension and Change Propagation”, 9th International Workshop on Program Comprehension (IWPSE 2001), co-located with ICSE 2001 (2001).

NII-Electronic Library Service