ソフトウェア工学工房

井上 克郎
大阪大学大学院情報科学研究科
〒 560–8531 豊中市待兼山町 1–3
inoue@ist.osaka-u.ac.jp

1 はじめに

ソフトウェア工学の分野では，その諸技術の善し悪しを定量的に評価するためには，実際の開発現場での適用を通じた有効性の評価を行うことが必要不可欠である．そのためには非常に手間がかかるため，経済的に割に合わないという認識が一般的であった．結果として，定性的な評価，主観的な議論が中心となり，科学的な評価とは言い難いものが多くなっていた．しかし，近年，可能な限り定量的なデータに基づいて技術を評価しようとする実証的ソフトウェア工学 (Empirical Software Engineering) が提案され，注目を集めている．実証的ソフトウェア工学を実践している組織は世界中に存在する．著名なものとしては，カーネギメロン大学ソフトウェア工学研究所，フラウンホーファー財団実証的ソフトウェア工学研究所，オーストラリア国立情報通信技術研究所 (NICTA) 等がある．実証的ソフトウェア工学において，いかに産学連帯を進め，よりよいソフトウェア開発を行うための諸技術を評価し，広めるかということは我が国のソフトウェア分野において最も重要な課題の一つであると考えられる．

実践的なソフトウェア工学を効率よく実行するためには，医学部や歯学部が持つ大学病院を参考とした組織が考えられる．医学部や歯学部（学部と呼ぶ）は教育組織であり，大学病院（病院）とは形式上独立している．しかし，通常は，学部の講座に対応して病院の診療科が設けられており，講座の長が診療科の長を兼務している．その他の学部の教員も病院の職員を兼務している．病院では，外部からの患者が，先端の診察や治療を受けるために毎日多数やってくる．大学の教員は，実世界の多数の患者を見ることにより，現場での知識の吸収を行ない，さらに新たな研究テーマを発見，展開することができる．また，新たな薬や治療法の効果を判定するための環境も病院では提供している．病院では，このような活動を行なうために，そのための専門の職員，建物，そして，予算が与えられるのが普通である．結果として，学部と切り離して，独立な活動が比較的自由にできる．もし，医学部，歯学部に病院がなければ，その教育・研究の効果が上がらないことが予想される．

本 COE プログラムでは，このような病院のモデルに近い枠組みを，ソフトウェア工学における教育・研究に当てはめ，ソフトウェアにかかわる種々の問題を，現実的に解決・対応する組織（ソフトウェア工学工房 [12]）を考える．具体的には，大学から研究者や学生，企業等から研究員らが対等な関係で，実践的な問題や研究テーマについて問題解決を図り，技術を身に付けること目的とする．

本報告では，これまでにソフトウェア工学工房として実施した活動内容についてまとめる．

2 活動概要

平成 14 年度から平成 18 年度にかけて実施した活動内容は以下の通りである．

(1) **社会人対象のソフトウェア工学工房セミナー:**
合計 9 回のセミナーを実施し，最先端のソフトウェア工学技術を現場の開発者へ紹介した．その際には，講義だけでなく，様々なツールを実際に試用してもらった．セミナーを通じ複数の企業と共同研究を実施してきている．このプログラムには，毎回 20〜90 名程度，総計 337 名が参加した．

(2) **大学院生対象の実践的講義:**
コンピュータサイエンス専攻の講義を通じて，実践的なソフトウェア工学技術を紹介した．なお，平成 18 年度は文部科学省の「魅力ある大学院教育」イニシアティブプロジェクト「ソフトウェアデザイン工学高度人材育成コア」と

連携し，1 年間を通じて技術スキル，マネジメントスキル，ヒューマンスキルに対する講義を実施した．この講義は，総計 270 名が受講した．

(3) **社会人ドクター学生の受け入れ:**
ソフトウェア工学工房セミナーを通じて，4 名の社会人ドクター学生を受け入れ，うち 1 名は既に博士の学位を取得して修了した．

3 社会人対象のソフトウェア工学セミナー

実施したセミナーは次の 9 回である．なお，本原稿執筆は第 8 回と第 9 回のセミナー実施前であるため，この 2 回については計画を述べている．

(1) 第 1 回セミナー (2002/11/15): コードクローン検出ツール CCFinder の利用体験

(2) 第 2 回セミナー (2003/3/17): Code Clone の紹介と利用体験

(3) 第 3 回セミナー (2003/6/20): Code Clone の説明とツール利用体験

(4) 第 4 回セミナー (2004/7/21): ソースコード検索システム SPARS-J の利用体験

(5) 第 5 回セミナー (2005/3/15): コードクローン検出技術とその応用

(6) 第 6 回セミナー (2005/12/13): コードクローン検出技術とその応用

(7) 第 7 回セミナー (2006/3/13): コードクローン検出技術とその応用

(8) 第 8 回セミナー (2007/2/7): コードクローン検出技術とその応用

(9) 第 9 回セミナー (2007/2/9): コードクローン検出技術とその応用

セミナーは，大阪で 5 回，東京で 4 回実施している．以降，セミナーで実施したコードクローン分析技術とソースコード検索システム SPARS-J について説明する．

図 1: 第 6 回セミナーの様子

3.1 コードクローン分析技術

コードクローン (code clone) とは，ソースコード中に存在するコード片で，他のコード片と一致または類似しているものを意味する [2]．コードクローンは,「コピーとペースト」によるプログラミングや意図的に同一処理を繰り返して書くということにより，プログラムテキスト中に作りこまれる．レガシーシステムに対する変更や拡張においても，コードクローンは作りこまれることが多い．実際に，約 20 年間保守しながら使われている，ある大規模ソフトウェアシステム（約 100 万行のサイズであり，2000 個のモジュールから構成されている）では，約 1000 個のモジュールに何らかのコードクローンが存在していることが確認されている [7]．

我々は，これまでに新しいコードクローン検出手法を提案してきている．提案手法では，プログラムコードを字句解析したうえで表面的な違いを吸収するための正規化を行うことで，プログラムコードをはじめとしてさまざまなテキストから，効果的にコードクローンを検出することが可能である．更に，提案手法に基づいてコードクローン検出ツール CCFinder を開発した [2]．CCFinder は，Java，C/C++，Fortran，COBOL 等のプログラムから高速にコードクローンを検出する．例えば，100 万行規模のソースコードに対して，PC/AT 互換機で，数分から数時間でコードクローンを検出することが可能である．

また，コードクローンによるソースコード分析をサポートするために，統合コードクローン分析環境 ICCA (Integrated Code Clone Analyzer) を

開発してきている．ICCA は，ソフトウェア開発・保守の際における次のような課題を解決することを目標にしている．(a) 詳細な内容が分からないレガシープログラム，あるいは，外注プログラムの品質をチェックする，(b) 検出したコードクローンの集約，(c) デバッグや機能拡張を行うときに編集すべき箇所の一覧の検出．ICCA では，コードクローン検出ツール CCFinder で検出されたコードクローンを対象として，上述の (a)〜(c) の課題を支援するために，3 つのサブコンポーネントを用意している．

3.1.1 Gemini

Gemini コンポーネントは，CCFinder が検出したコードクローンの情報をグラフィカルにユーザに提供する [11]．Gemini の解析はファイルベースの解析とクローンベースの解析に分かれている．ファイルベースの解析では，各ファイルにどの程度クローンが含まれているのかというような，ファイルを単位とした解析を行う．一方，クローンベースの解析では，特徴的なクローンがソフトウェアのどの部分に含まれているのかというような，クローンの集合を単位とした解析を行う．詳細は 3.2 節で述べる．

3.1.2 Aries

Aries コンポーネントは，コードクローンの集約支援を行う [3]．集約支援を行うために，Aries は CCFinder が検出したコードクローンに対して 2 つの処理を施す．1 つ目の処理は，プログラミング言語の意味に照らして構造的なまとまりのある部分の抽出で，2 つ目の処理はメトリクスを用いた特徴づけである．この 2 つの処理を通じて，適切な集約方法をユーザに提示する．

図 2 は構造的なまとまりであるクローンの例を示している．A と B それぞれの灰色の部分は，その部分が A と B の間の最大長のコードクローンであることを示している．コード片 A ではいくつかのデータがリスト構造の先頭から順に連続して格納されている．一方コード片 B では，リスト構造の後方から順に連続してデータが格納されている．これら 2 つのコード片には，リスト構造を扱う共通のロジック（for 文）が含まれているが，コード片の最初と最後には，偶然クローンとなった部分（代入文）も含まれてしまっている．集約を目的とした場合，灰色の部分全体よりも for 文のみをコードクローンとして抽出する方が望ましい．このような場合，灰色で示されたコードクローンから構造的なまとまりを持った部分，つまり for 文の部分のみを抽出する．

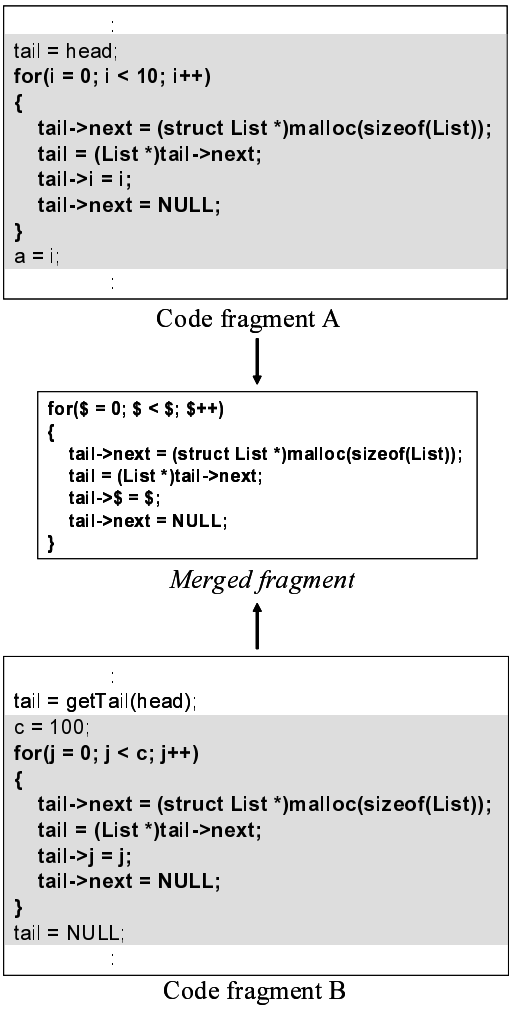

図 2: コードクローン集約の例

次に，抽出された構造的なまとまりを持つコードクローンのプログラム上の位置（例えば，クローンを含むクラスが継承木中のどの位置にあるか）やコードクローンの周りのコードに対する結合の度合いによって，適用できるリファクタリングパターンを提示する．これまでに様々なリファクタリングパターン [1] が提案されており，以下の 7 種類のリファクタリングパターンがコードクローンを除去するために用いることができる．

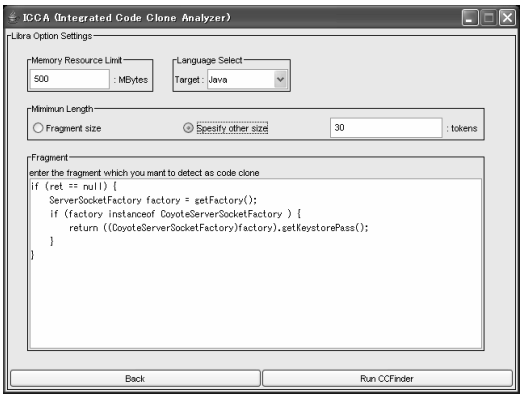

図 3: Libra 入力画面例

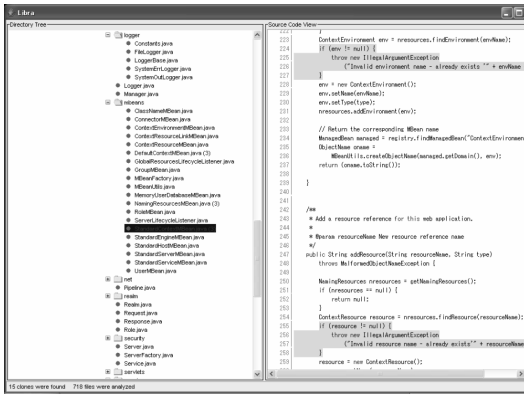

図 4: Libra 出力画面例

- Extract Class,
- Extract Method,
- Extract SuperClass,
- Form Template Method,
- Move Method,
- Parameterize Method,
- Pull Up Method.

3.1.3 Libra

Libra コンポーネントは，コードクローンに対する修正支援を目的としている [9]．コードクローンに対するデバッグや機能追加時に，Libra コンポーネントを使うことによって，漏れなく修正を行うことが可能である．Gemini コンポーネントを用いても同様の支援は行うことは可能であるが，Gemini コンポーネントは，CCFinder の検出したコードクローン全てをユーザに提供するため，不要な情報も含まれる．デバッグ時や機能追加時は，すべてのコードクローン情報は必要なく，修正箇所を一箇所突き止めたら，後はその部分とのコードクローンのみを表示すれば十分である．

図 3 はユーザがコード片の入力を行う画面を示す．Libra はここで入力されたコード片と指定されたソースコード間のコードクローンのみを検出する．図 4 は GUI のスナップショットである．ウィンドウの左側に対象ソースコードのディレクトリツリーが表示される．指定されたコード片とクローンになっているソースコードは青色でハイライトがかかり，そのソースコード内のコードクローン数も表示される．ディレクトリツリーでファイルを選択すると，そのソースコードが右側のソースコードビューに表示され，コードクローンの部分はハイライトがかかって表示される．

3.2 Gemini の機能

我々は，ソフトウェア工学工房セミナー等を通じて，これまでに国内外 100 箇所以上に ICCA を配布してきている．開発現場での適用を通じて，様々なフィードバックを得た．最も多くあった要望が，Gemini を使用したコードクローン分析法に関するものであった．

Gemini は GUI を通じて，クローン散布図やコードクローンに関するメトリクスのグラフを表示・操作する機能や個々のコードクローンのソースコードを表示する機能を持つ．

3.2.1 クローン散布図

クローン散布図の簡単なモデルを図 5 に示す．散布図の原点は左上隅にあり，水平軸，垂直軸は，それぞれソースファイルの並び（f1 から f6）に対応している．両軸上で，原点から順にそれぞれのソースファイルに含まれるトークンが並んでいる．座標平面内に点がプロットされている部分は，その両軸の対応するトークンが一致することを意味する．したがって，散布図の主対角線は，両軸の同じ位置のトークンを比較することになり，全ての点がプロットされることになる．一定の長さ（CCFinder で設定される最小一致トークン数）以上の対角線

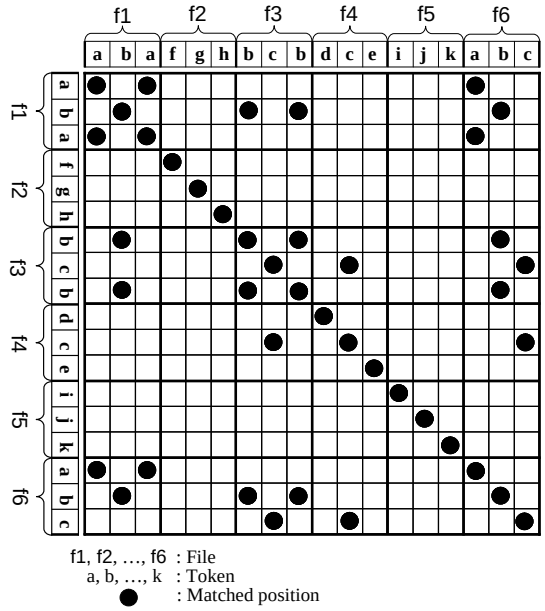

図 5: クローン散布図モデル

分が，検出されたクローンペアである．図 5 では，f1 から f6 までのファイルが，それぞれ 3 つのトークンを含んでいる．ファイルの並びはファイル名の辞書順となっている．検出されるクローンペアは f1 と f6 に含まれる“ ab ”というトークン列と，f3 と f6 に含まれる“ bc ”というトークン列である．

3.2.2 メトリクスグラフ

メトリクスグラフは検出されたコードクローンをメトリクスを用いて特徴づける．特徴づけされたコードクローンはクローンセット単位で表される．メトリクスグラフは多次元並行座標表現を用いている．このグラフで用いられているメトリクスは，$RAD(S)$，$LEN(S)$，$RNR(S)$，$POP(S)$，$DFL(S)$ の 5 つである．ここでは，$RNR(S)$ を除く 4 つのメトリクスについて簡単に説明を行う．$RNR(S)$ については 3.2.4 節を参照されたい．

RAD(S): クローンセット S 内のコード片が含まれるファイル集合 F が，ファイルシステムの中でディレクトリ構造的にどれだけ分散しているかを表す．ディレクトリ構造を表す木構造を考え，F 内の全てのファイルに共通の親ノードの中で最も下位層に存在するノードまでの距離を求め，S 内でのその最大値を $RAD(S)$ として定義する．

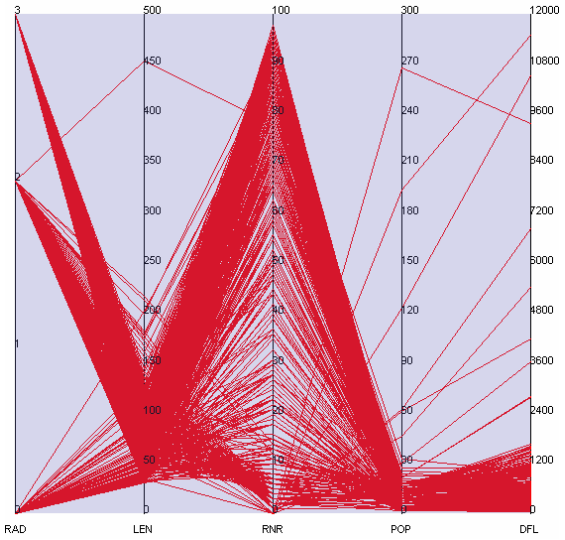

図 6: メトリクスグラフ

LEN(S): クローンセット S 内に含まれるコード片のトークン数の平均値を表す．

POP(S): クローンセット S 内のコード片単位の要素数である．$POP(S)$ が高いということは，同形のコード片が多く存在することになる．

DFL(S): クローンセット S に含まれるコード片に共通するロジックを実装するサブルーチンを作り，各コード片をそのサブルーチンの呼び出しに置き換えた場合の減少が予測されるトークン数を表す．

5 つのメトリクスはメトリクスグラフ上の縦軸となっており，それぞれにメトリクス名のラベルがついている（図 6）．このグラフではクローンセット毎に 5 つの縦軸上の点を結ぶ折れ線が引かれている．ユーザはこれら 5 つの座標の上限と下限を変更することで任意のクローンセットを選択することが可能である．

3.2.3 ファイルリスト

ファイルリストでは，ユーザは定量的に特徴的なファイルを選択することができる．以下のメトリクスがファイルを定量的に特徴付けるために用いられている．

NOL(F): ファイル F の行数を表す.

NOT(F): ファイル F のトークン数を表す.

NOC(F): ファイル F に含まれているコードクローンの数を表す.

ROC(F): ファイル F がどの程度重複化しているかを表す.

NOF(F): ファイル F がコードクローンを共有しているファイルの数を表す.

ファイルリストはテーブル形式で実装されており，各行に 1 つのファイルがそのメトリクス値と共に表示される．ファイルリストは行のソーティング機能があり，各メトリクス値の昇順，あるいは降順にファイルを並び替えることが可能である．また，ファイルリストはクローン散布図と連携しており，ファイルリストにおいて選択されたファイルがクローン散布図において強調表示される．

3.2.4 フィルタリングメトリクス: RNR(S)

ここではフィルタリングメトリクス $RNR(S)$ について述べる．$RNR(S)$ はクローンセット S 中に含まれるコード片がどの程度非繰り返しであるかを表すメトリクスである．例えば，以下のトークン列を考える．

$$x\ a\ b\ c\ a\ b\ c^*\ a^*\ b^*\ c^*\ y.$$

*がついたトークンはそれが繰り返しトークン列に含まれていることを表している．このトークン列を入力として与えた場合，CCFinder は以下の 2 つのコード片をクローンとして検出する．

F1. $x\ \underline{a\ b\ c\ a\ b\ c^*}\ a^*\ b^*\ c^*\ y,$
F2. $x\ a\ b\ c\ \underline{a\ b\ c^*\ a^*\ b^*\ c^*}\ y.$

コード片 $F1$ は 6 トークンから成り，そのうち 1 トークンが繰り返しトークンである．一方コード片 $F2$ も 6 トークンから成り，そのうち 4 トークンが繰り返しトークンである．

この場合，これらのコード片からなるクローンセット S_1 の $RNR(S_1)$ は，

$$RNR(S_1) = \frac{5+2}{6+6} = \frac{7}{12} = 0.58\dot{3}$$

となる．

これまでの経験から $RNR(S)$ の値が 0.5 未満の場合は，C 言語であれば連続した printf や scanf であったり，Java 言語であれば連続した import 文であったりと，目で確認を行ってもあまり意味のないコードクローンであることがわかっている．メトリクス $RNR(S)$ を用いることでこのようなコードクローンを取り除くことが可能となる．また，メトリクス $RNR(S)$ はメトリクスグラフで用いられいているだけでなく，クローン散布図においても用いられている．クローン散布図では $RNR(S)$ が 0.5 未満のコードクローンは青く描画され，他のクローンと区別がつくようになっている．

3.3 ソースコード検索システム SPARS-J

近年，大規模で複雑な大量のソフトウェアが開発され，様々な場所において様々な目的で利用されている．プログラムソースの公開と再配布を許可しているオープンソースソフトウェア開発コミュニティは，ソフトウェアプロダクトが増加するにつれて開発基盤としての価値だけでなく，ソフトウェア資産を有する大規模なライブラリとしての価値も高まっている．これらのソフトウェア資産の中には新たな開発作業において活用することができるアイデアや，少しの修正を加えるだけで開発に用いることが可能なソフトウェア部品が存在していると考えられる．

そこで，大規模なライブラリの可視化によるソフトウェア資産の有効活用が重要となる．有効活用の一例としてソフトウェア部品の再利用が挙げられる．一般にソフトウェア部品 (Software Component) は再利用できるように設計された部品とされ，再利用の単位として用いられる．再利用は高品質なソフトウェアを一定期間内に効率良く開発するために有効な技術の一つとして知られているが，その効果を最大限に得るためには，部品を含むライブラリに関して十分な知識を再利用者が持たねばならない．しかしながら，多数の開発者が参加する開発プロジェクト内でライブラリに関する知識の共有を行なうことは非常に困難でコストのかかる作業である．現在は自然言語文書用に開発された全文検索システムを用いてライブラリを構成し，開発者が望む情報や部品を適宜検索するという方法

が主である．しかし，ソフトウェアは自然言語文書とは違い依存や類似といった関連をもっているため適切な運用は難しい．

我々は，利用実績に基づいたソフトウェア部品の解析・検索システム SPARS (Software Product Archive, analysis and Retrieval System) の提案を行っている [8]．利用実績の評価に当たっては，(1) ソフトウェアを構成する部品間には相互に利用関係がある，(2) 一般に，時間が経過し，多くのプロジェクト開発で再利用などが行われるに連れて部品の利用関係は変化していく，(3) 十分な時間が経過した状態のもとで，被利用数が多い部品は重要であり，また，重要な部品から利用されている部品も重要である，という考えに基づいて Component Rank 値（CR 値）を提案している．SPARS システムでは，CR 値に基づいて部品の評価を行い，検索結果の順位付けをする．

これまでに SPARS システムの一つとして，本論文では Java ソースコードを対象とした，SPARS-J を構築し，その有効性を評価している．SPARS-J は，依存や類似といったソフトウェア部品特有の特性を考慮しながら，大規模なライブラリの構築を自動的に行なうシステムである．キーワードとトークン種類を検索キーとした全文検索を行ない，ソフトウェアのソースコードを効率良く検索することが可能である．さらに，検索結果表示の際にソフトウェア部品に関する詳細情報を併せて提供する．このシステムを用いることで，ライブラリの知識が無い開発者も有用なソフトウェア部品やそれに付随する有益な情報を容易に入手することができる．SPARS-J の画面例を図 7 に示す．

3.4　セミナーの成果

コードクローン分析に関しては，本セミナーを通じて国内外 150 以上の組織にコードクローン分析ツールを配布している．また，それらの配布先組織との共同研究も実施している．第 5 回セミナーではシンガポール大学のスタン・ジャザベック教授 (Professor Stan Jarzabek) に招待講演をお願いしたた．ジャザベック教授はソフトウェア再利用や保守に関する研究で世界的に著名な研究者であり，我々のコードクローン検出ツールを利用した論文を多くの国際会議や論文誌で発表している．また，

図 7: SPARS-J の画面例

同セミナーではコードクローン適用事例報告を行い，幾つかの企業でのツールの現場利用や，ツールの使い勝手，機能面における改善要求等について議論した．これらのフィードバックを受けて，新たな研究を実施することができた [10, 5, 6]．

ソースコード検索システムについても，参加企業に対してツールの配布を行っている．特に，サントリー株式会社のソフトウェア開発部門とは SPARS-J システムの開発現場への導入・機能改良を目的とした共同研究を行った．また，企業から受託研究員を受入れ，彼らの組織で開発に利用している，COBOL 言語と C 言語を対象とした SPARS システムの開発について共同研究を実施した．

4　大学院生対象の実践的講義

コンピュータサイエンス専攻博士前期課程の講義「ソフトウェア開発論」,「ソフトウェア保守工学」で実践的な講義を実施した (図 8)．ソフトウェア工学における最新技術の試用のため，ソフトウェア工学工房の予算で購入したラップトップ PC を貸し与えて，学生に実際にツールを使い，作業を行わせた．使用したツールは，統合開発環境 Eclipse，構成管理ツール CVS，アスペクト指向開発環境 Aspect-J である．また，社会人ゲストスピーカーによるソフトウェア開発現場におけるプロジェクト管理や見積等についての講義を実施した．

平成 18 年度は文部科学省の「魅力ある大学院教

図 8: 大学院講義での授業風景

育」イニシアティブプロジェクト「ソフトウェアデザイン工学高度人材育成コア」と連携し，1 年間を通じて技術スキル，マネジメントスキル，ヒューマンスキルに対する講義を実施した．詳細については Web ページ [13] を参照されたい．

5　まとめ

大学から研究者や学生，企業等から研究員らが対等な関係で，実践的な問題や研究テーマについて問題解決を図り，技術を身に付けること目的としたソフトウェア工学工房プロジェクトの活動について報告した．

COE プロジェクトとしてのソフトウェア工学工房は終了するが，今後も同様の活動を継続することを検討している．大阪大学中之島センター，イノベーションセンター（田町）を利用したセミナーの開催，大学院生に対する実践的な講義の開講を行う．本研究科では，現在，文部科学省先導的 IT スペシャリスト育成推進プログラムの支援を受け,「高度なソフトウェア技術者育成と実プロジェクト教材開発を実現する融合連携専攻の形成 IT Spiral」を実施している．IT Spiral では，情報通信技術，特にソフトウェアの高度な技術者育成を目標とし，ソフトウェア工学分野で教育・修得すべき内容をより豊富にかつ体系的・実践的に教育課程に取り込むため，関西圏の情報系 9 大学院に分散している該当分野の卓越した専門家群を結集し，融合連携型専攻を構築する．特に重要視する実践的教育については，参画企業と協働して，教科書的例題ではなく現実の開発プロジェクトそのものを教材として開発し，適用することを目的としている．本ソフトウェア工学工房プロジェクトで得られた知見や方法を IT Spiral でも引き続き活用していく予定である．

また，学外においても，ソフトウェアエンジニアリングセンターにおける活動や文部科学省のリーディングプロジェクト「e-Society 基盤ソフトウェアの総合開発」EASE（Empirical Approach to Software Engineering：ソフトウェア工学へのエンピリカルアプローチ）プロジェクトを通じて，我々のソフトウェア工学技術の普及・教育をはかる予定である．

既に述べた通り，本活動を通じて，これまでに社会人博士後期課程に 4 名の入学があり，一部の学生は学位を取得している．今後も，受託研究員や社会人博士後期課程への受け入れ，ソフトウェアの開発現場での課題解決や成果のとりまとめを通じて技術レベルの向上を目指していくことを考えている．

参考文献

[1] Fowler, M., “Refactoring: Improving the Design of Existing Code,” Addison Wesley (June 1999).

[2] Kamiya, T., Kusumoto, S., and Inoue, K., “CCFinder: A Multilinguistic Token-Based Code Clone Detection System for Large Scale Source Code,” IEEE Transaction on Software Engineering, Vol. 28, No. 7, pp.654–670 (July 2002).

[3] 肥後芳樹, 神谷年洋, 楠本真二, 井上克郎, “コードクローンを対象としたリファクタリング支援環境,” 電子情報通信学会論文誌 D-I, Vol. J88-D-I, No. 2, pp. 186–195 (2005 年 2 月).

[4] 肥後芳樹, 楠本真二, 井上克郎, “コードクローン分析ツール Gemini を用いたコードクローン分析手法,” 電子情報通信学会技術研究報告 (SS2005-30), Vol. 105, No. 228, pp. 37–42 (2005 年 8 月).

[5] 肥後芳樹, 吉田則裕, 楠本真二, 井上克郎, “産学連携に基づいたコードクローン可視化手法の改良と実装,” 情報処理学会論文誌 (2007 年, 掲載決定).

[6] Higo, Y., Kamiya, T., Kusumoto S., and Inoue, K., “Method and Implementation for Investigating Code Clones in a Software System,” Information and Software Technology (2007, to appear).

[7] 井上克郎, 神谷年洋, 楠本真二, “コードクローン検出法,” コンピュータソフトウェア, Vol. 18, No, 5, pp. 47–54 (2001 年 9 月).

[8] Inoue, K., Yokomori, R., Yamamoto, T., Matsushita, M., and Kusumoto, S., “Ranking Significance of Software Components Based on Use Relations,” IEEE Transactions on Software Engineering, Vol. 31, No. 3, pp. 213–225 (Mar. 2005).

[9] 泉田聡介, 植田泰士, 神谷年洋, 楠本真二, 井上克郎, “ソフトウェア保守のための類似コード片検索ツール,” 電子情報通信学会論文誌 D-I, Vol. J86-D-I, No. 12, pp. 906–908 (2003 年 12 月).

[10] 川口真司, 松下 誠, 井上克郎, “版管理システムを用いたクローン履歴分析手法の提案,” 電子情報通信学会論文誌 D, Vol. J89-D, No. 10, pp. 2279–2287 (2006 年 10 月).

[11] 植田泰士, 神谷年洋, 楠本真二, 井上克郎, “開発保守支援を目指したコードクローン分析環境,” 電子情報通信学会論文誌 D-I, Vol. J86-D-I, No. 12, pp. 863–871 (2003 年 12 月).

[12] ソフトウェア工学工房 URL, http://sel.ist.osaka-u.ac.jp/kobo/index.html.ja.

[13] ソフトウェアデザイン工学 URL, http://sel.ist.osaka-u.ac.jp/topics/SD2006.