研究論文

鳴門教育大学情報教育ジャーナル　4，47－55，2007

失敗ソースコード学習と学生主体組織による プログラミング授業に関する研究

皆月昭則*

プログラミング言語を使用したモノづくりは，単にコードを書くということではなく，作り込みに着目させる必要がある。作り込みとは，モノを向上・発展させてプログラミングすることで，モノづくりにおける質的向上過程と言える。質的向上策には，前年度学生の成果と失敗事例を理解させる必要がある。失敗事例には，正確な文法による美しいサンプルコード学習とは異なる学習が可能であり，作り込みに必要な構想力を生み出すとも考えられる。本研究の第１主題はプログラミング言語のサンプルコード学習における問題点を抽出して，改善に向けた授業実践について述べ，第２主題は授業の最終課題におけるモノづくりの構想支援と組織環境，モノづくりテーマの選定方法，プログラミング授業実践と社会貢献に関する成果を述べる。

〔キーワード：プログラミング教育，Java，構想力，前工程，後工程〕

1．はじめに

多くの社会科学系（文系）の学生において，プログラミングの授業は他の科目群の中でも数少ない実習的な科目であり，モノづくりを実践する機会である。また，プログラミング言語を学んでモノづくりをする授業は，多くの学生が興味関心をもつ科目であると言える。しかし，プログラミングの授業について，学生の感想を抽出すると，学習内容や授業展開において，十分な満足感が得られていないことが明らかである。過去のプログラミング授業による15週終了後のアンケートの結果では，プログラミング言語の基本文法はおおむね理解できたが，最終課題のシステムづくりがスキル不足で達成できなかったなど失望的な要因が多くあげられる。失望的要因の異なる解釈では，プログラミング授業は実用的なスキルを得られるのではないのかという受講学生の期待が達成できなかったことになる。授業展開の指摘では，サンプル・プログラムのコード入力だけでは，最終課題をつくるためのスキルなど，自由な作り込みに対応していないことになる。この問題点は，プログラミング言語の基本文法に関するサンプル・プログラムのコード入力学習と実際の作り込みスキルが異なることを示す。授業では，最終課題となるシステムづくりに関して，共通テーマあるいは自由テーマを設定するが，そのテーマに沿ったシステムの作り込みができないことが起きている。作り込みができない学生によると，授業で示されたサンプル・プログラムの文法や解説が理解できないということではなく，サンプル・プログラムの学習イメージから一脱することができないため自由な作り込みができないようである。プログラミング授業において自由テーマで自由設計の最終課題ができないということは，実用的なスキルを求める授業の位置づけを欠くことになる。

よって，プログラミング授業は，どのようにしたら実用的な作り込みスキルになるのかを考えていくのも昨今の中長期的な課題である。現状でプログラミング授業における期待は，30週あるいは15週前後の授業時間では困難であるという意見もあるが，サンプル・プログラムの解説と入力演習の繰り返しでは，授業時間を長くしても作り込みスキルの会得は難しいと考えられる。よって，「プログラミングができるようになる」ことを最終課題で確認し，それを実用的スキルとするのであれば，従来の指導内容と展開において何か工夫をすることが必要であると考える。本稿は，従来のプログラミング授業の指導内容と展開を否定することではなく，語学の授業のように基本文法も重要視する。プログラミングは，あらかじめ決められた文法にしたがってプログラムすることであるが，構造化も同時に着目しなければならない。すなわち，プログラムが正常に実行できない状態であれば，エラー箇所の文法の修正だけでなく構造全体を見直す必要がある。従来の授業で解説するサンプル・プログラム

* 釧路公立大学　経済学部（minaduki@kushiro-pu.ac.jp）

は，エラー箇所や構造上の問題が含まれていないため，学生が単にコードを入力することで実行が可能になるものである。コード入力の繰り返し授業で学生の視点は，プログラムコードの実行構造が左上から数行程度であると理解する。正確なサンプル・プログラム学習は必要であるが，一連のシステムとプログラム全体構造を構想させるためにはコード入力やコード修正など視点切り替えも必要である。

社会科学系の学生にとって，プログラム授業というモノづくりの機会は，理工系学生に比較して少ないため，どのようなプログラミング言語を選択して学ぶ（使えるようにする）のかが重要である。例えば，プログラミング言語で Fortran，Visual Basic，C++，Java，C# など数種類の言語を選定する場合，その選定における重要な条件は，どのような最終課題を学生に課して，どのくらいの時間で作り込みが終了できるのかを考慮する必要がある。例えば，C++ というプログラミング言語は，実装するコンピュータの性能を最大限に発揮するようなソースコードの記述と構造化が可能であるが，使えるようになるまでは Java 言語に比較して多くの時間を要する。Java というプログラミング言語はソースコードの記述が単純明快で読みやすいという特徴がある。そして，C# というプログラミング言語は，Java に比較して文法がやや複雑になるが，C++ 言語の特性も含み，さらには，.Net Framework という Windows アプリケーション向けの特性を備えている。よって，プログラミング言語のさまざまな特性を考慮した上で，本稿では，前期 15 週の授業で Java 言語を選定し，後期 15 週の授業で C# 言語を選定している。本稿の授業実践では，一言語よりも二言語のプログラミング言語を選定学習することで，各プログラミング言語の特性に応じたモノづくり（システムづくり）できると考える。

社会科学系のカリキュラムでは，モノづくりに関する方法論に多くの時間をとれない。モノづくりに関する方法論とは，プログラミング言語を習得し，構造的なシステムとして組み合わせていくことであるが，そのプロセスには「構想力」というものが関係すると考えられる。構想力は，プログラミングコードを入力ながら，システムの構造を逐次カタチづくる能力ではないのかと考えられるが，構想力そのものを形式的に伝えることは困難である。

構想力が，どのような力の組み合わせで生じるのかという説は，我々の社会におけるモノづくりの種別によって異なるが，そのような力の生じる起源は作り込み環境や経験であろうし，2 つ以上のこと（力）を同時に組み合わせて作るということも実際のモノづくりから推察できる。よって，本稿の授業実践例では，プログラミング言語の理解と構想力が発揮される授業内容と展開例を次節に述べる。

2．「構想力」と失敗ソースコード Tips

プログラミング授業内容は，入門的な事柄から始めて応用的な内容になるが境界が明確ではない。例えば Java 言語のサンプル・プログラムの入力では，「文字を表示してみよう！ Hello, Java」が入門のサンプル・プログラムであり，この授業開始時点は学生の興味と関心が高く，コンソール出力でも，アプレット出力でも大半の学生が実行できる。しかし，このようなサンプル・プログラムを小出しにする授業は，作り込みの流れの全体から「前工程」と呼ぶことができる。このようにサンプル・プログラムを小出しにする「前工程重視型」授業は，従来から確立されている展開法であり，加えて構想力を発揮することでプログラミング言語は自在にあつかえると考えられる。すなわち，複数のサンプル・プログラムを組み合わせて作るモノを考えることが，構想力であり，それは，システムの構造的な流れに着目した「後工程」を自ら作り込むことでプログラミング言語の実用度は高められる。ただし，従来の授業では，後工程を作り込めない学生もおり，後工程の作り込みは，システム全体の構想的な問題と大きく関わる。よって，前工程と後工程をバランスよく理解するための授業環境（支援サーバ）が必要である。Java 言語の場合，構築・実装されたモノとは，ソースコードのオブジェクトクラスとして一連の流れで読み解くことが可能であり，数字や文字以外のデータはすべてオブジェクトクラスであり，語学の学習の例に比較すると，初期段階で長文読解をするように長いセンテンス（コード群）になる。語学の授業と異なる点は，読解に用いる長文（ソースコード）は，誤りのある（失敗あるいは良くないソースコード）も支援サーバで提示することである。以下，授業実践した「前工程」と「後工程重視型」の環境における授業展開で重要な観点を列挙して述べる。

2．1　前工程を重視する Hello, Java 型と実用度の問題

Hello, Java という文字列を表示してみようというソースコードは多くのプログラムの模範的なイメージであり，C# 言語の授業では Hello, C# でも可能である。Hello に続くサンプルは，四則演算の方法，変数・条件文・配列などのサンプルも類似した出力イメージになる。基本文法が正確なサンプル・プログラムは整然と実行することが可能であるが，構造や構想的な問題を残さない。授業の最終課題が「できない学生」の多くがサンプル・プログラムに偏重した思考になっていることが，学生のできない理由のひとつとしてあげられる。実際の作り込み工程では，きれいで整然としたプログラムコードが記述できることはなく，煩雑なプログラムになることが多い。サンプル・プログラムの整然としたイメージに偏重した学

生の作り込み意識は，構想的なイメージを妨げる可能性もある。よって，本授業実践では，前年度学生が取り組んだ煩雑で誤りのあるソースコードを授業で示して考察することでも偏重意識は変えることができると考えた。そこで失敗したソースコード集（失敗事例集）を閲覧するための環境を構築して授業で用いた。失敗したソースコード集はWebサーバからの利用が可能であり，プログラムの一連の構造を考察することが可能になり，多くの学生の興味や関心を高めることにもなった。これによって構造化の際の「木を見て森を見ず」のような視点は改善できると考えられる。また，コーディングにおいて，「木」を確実に１本１本植えながらも，最終的にどのような「森」にするのかを考えさせる事例として，失敗事例のソースコードに着目させることは構想力向上の学習にもなる。

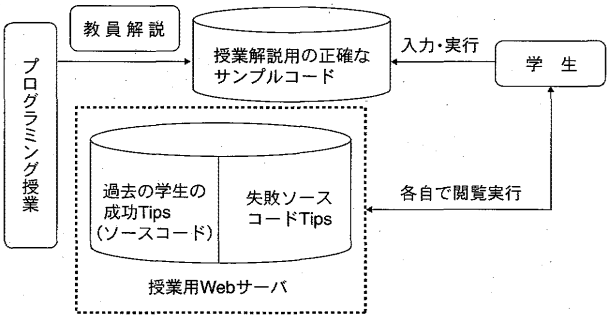

図１　授業Webサーバ（失敗ソースコード集）

プログラミング授業で得たモノづくりの「構想力」は，中間と最終（定期）試験で評価することが難しく，実際に最終課題で何かを作り込ませてデモンストレーションさせることが必要である。すなわち，試験などで，検定や資格試験のような制御文に着目した問題の解答では，構想力の評価は難しい。プログラミング授業の評価を決めるとすれば，構想力を結集した最終課題が「実行できるモノ」になっているのかを評価することであり，実際の製品やモノの評価は，使えるシステムとして作り込みによる品質の向上が大きな評価を占めている。大学における学問分野や科目のすべてに，実用度を求めてはならないが，少なくとも，プログラミング言語を含む情報系の授業は，英語などの語学のように実用度がどのくらい定着するのかを期待されているのではないだろうか。プログラミング言語の習得と構想力をのばして，実用度を高めるための先行的な取り組みでは，IT企業の研修制度の変容が見られる。IT企業では，OJT(On-the-Job Training)方式によって社員の構想力を向上させる取り組みをしており，プログラミング言語の基本文法などの研修期間中の初期の短い時間で学び，研修期間中の長い時間を現場のプロジェクトメンバーに参画させる。OJTは作り込み組織へ参画させることで，研修生の知識偏重イメージをなくすことでもあると考えられる。また，OJTにおける作り込み組織への参画は，個人の成果を他者（同僚あるいはユーザ）から評価することが可能である。大学におけるプログラミング授業は，OJTと同じにする必要性はないがモノづくりや作り込み組織からの評価環境は参考にすべき点が多い。よって，本稿の授業実践では，最終課題に取り組むための組織編成や最終課題における成果の評価をOJTにおけるモノづくり研修を参考にしながら，授業展開の枠組みを図２のようにして授業実践した。

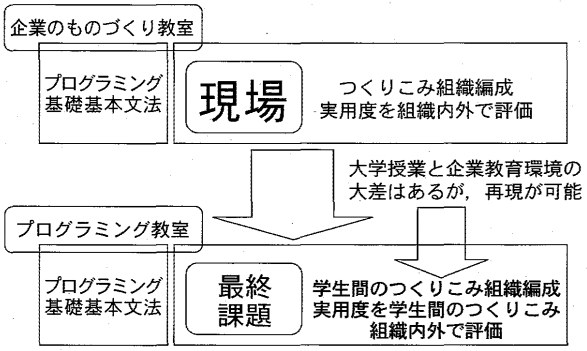

図２　本稿で述べるプログラミング授業展開の特徴

プログラミング言語を駆使して構造的にシステムを組むには，構想力が必要であり，Java言語を用いる場合，システムの作り込み工程における前工程（図３）から後工程（構造的自由設計部分）まで一貫した処理が必要である。すなわち，前工程を作り込みながら後工程でどのようなデータを受け取り，どのように処理するのかなどを構想する必要がある。構想的に作り込むことは，前工程で使用する型宣言や変数の名前も後工程に影響する可能性を考慮して，制御文なども含めて一連のシステムの流れを構想することが必要である。このような構想的な作り込みは前工程のサンプルコード学習偏重では難しく，図３で示すような前工程の確実な作り込みから後工程を整合させる必要がある。構想的な作り込みでは，授業の早い時期にプログラミング言語における長いソースコードに慣れ親しむことで，後工程における値の受け渡しや全体構造に着目させることができる。よって，本稿で述べる授業実践では，長めのソースコードを授業に利用している。

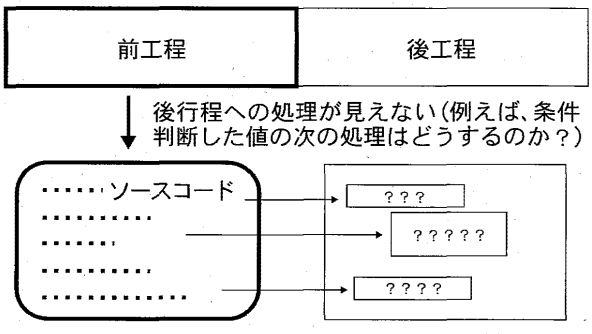

図３ 前工程重視 Hello, Java 型

2．2　後工程重視 See, Java 型

本稿では，Java 言語を使用したプログラミング授業を後工程に着目させながら進める授業展開を See, Java 型と呼ぶ。これは前節で述べた前工程重視型のように Hello, Java という文字を表示させる文法とサンプルソースコードの解説に加えて，構造化など後工程の作り込み事例を示すことで全体構造に着目させる授業展開法である。

この授業展開法の See の意味するところは，作り込みにおいて「木を見て森を見ず」状態に陥らないように，「森を構想しながら木を適切に植える」という視点である。後工程プログラムの具体的イメージを示すと，図４のようになり前工程と厳密に区別する境界はないが，Hello, Java のようなサンプルに比較するとコードが長い。よって，長いソースコードにおける構造を読み解くことで開発者（前年度学生）の作り込み意図を理解させることも可能になる。作り込み意図の例では，ソースコードにおける階層的な処理構造や制御文への値の受け渡しなどの工夫を理解させることが可能である。

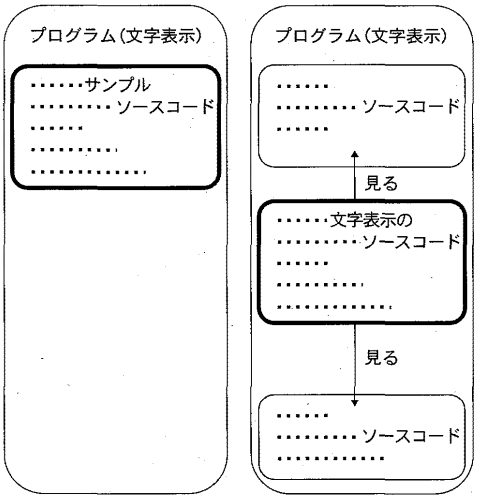

図４　前工程と後工程ソースコードの比較イメージ

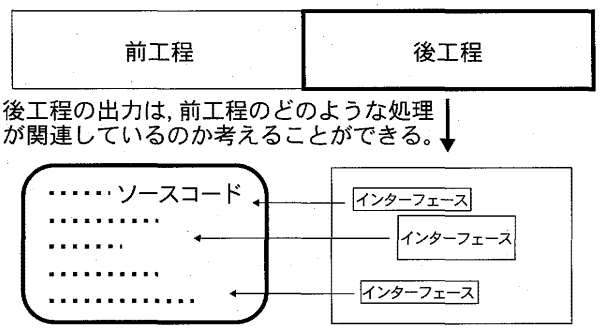

図５　後工程重視 See, Java 型

3．後工程重視 See, Java 型授業実現の授業環境

本稿で述べている授業実践の学生に課す最終課題は，授業の初回時にテーマを選定・決定させて，授業終了 15 週目までの４ヶ月にわたり取り組ませる。授業および最終課題の支援では，前年度から前々年度の授業で学生が作り込んだ後工程関連ファイル（システム・ソースコード・クラスファイル・Reademe ファイルなど）をプログラミング授業専用のサーバで利用できる。サーバの閲覧ページ例は，図６のような 2006 度の授業学生ファイルから 2005 年度以前の前・後工程に関するファイルと失敗ソースコード Tips などを閲覧することが可能であり，具体的な作り込みイメージや構想における流れを理解することが可能である。

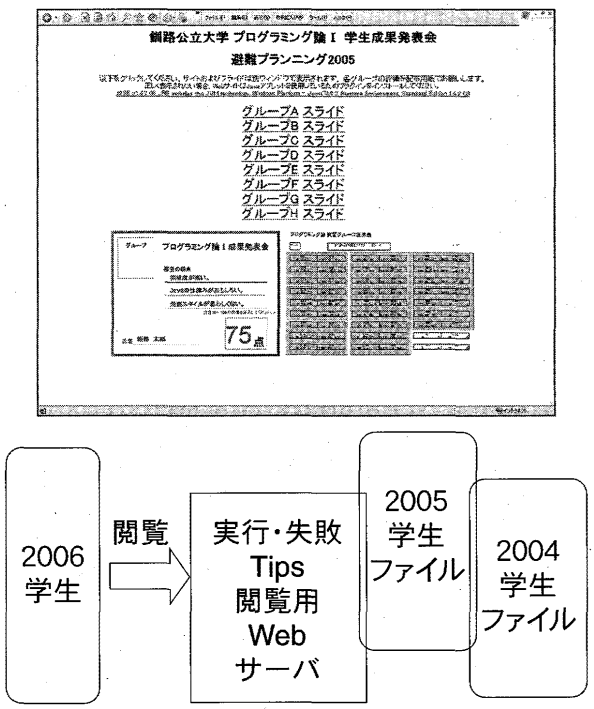

図６　作り込み支援サーバのページ遷移のイメージ

3．1　支援サーバのコンテンツと多年度進行型授業展開

本稿で述べたプログラミング授業の支援サーバのページ遷移は，前年度と前々年度の作り込みの工夫や苦労の痕跡を実行・失敗 Tips の閲覧でたどることも可能であり，失敗 Tips ソースコードには実行できなかった経過や Readme など学生の注目ページも多い。この支援サーバは，年度毎学生に共有ファイル閲覧を許可しており，コンテンツの追加的書き込み・読み込みを年度に該当する授業学生に対してアクティブ・ディレクトリ機能で管理している。授業実践における最終課題テーマは，３年間継続するテーマであり，支援サーバ内を前々年度のページまで閲覧すると，後工程における構造化が十分ではなく，意味のない作り込み機能も見られるが，前工程で何を作りそして，後工程で何を構造化したかったのかなどが Readme で理解できる。例えば，構想的に不十分であると評価するモノ（オブジェクト）は，前工程における基本機能の作り込みに時間を要したことも Readme されている。特に，初年度に取り組む学生は，図７のように基本機能の作り込みで参考や引用するモノがないため，構

想的な自由度が狭いモノになっていることもある。しかし，初年度の学生による作り込みの成果には，構造的な見直しをして，さらに構想を発展的にすることも可能であり，次年度学生に対する課題にもなる。したがって次年度学生は，後工程の作り込みとして構想的な作業が中心となり，構造的な見直しと，前工程の基本的な制御や入出力などのしくみを理解して改良をおこなう。これは，各機能がクラス化されているため前工程の作り込みの負担が軽減されると同時に改良するための作り込みにおいて一層の構造的理解を深めることになると考えられる。よって，最終課題は，初年度から少なくとも３年間の工期を経て各機能が発展的に構造化されて，同時に多年度にわたる学生の構想力が集められた作り込みである。図７に示したように，学生の作り込みにおいて前・後工程に費やす時間配分は，前工程の基本的なクラスの作り込みが年度の進行で発展して深まることは，後工程からの構造的な見直しに時間をかけることが可能であり，年度の３年目には，後工程に大半の時間を用いる。しかし，次年度の一部の学生は，前年度学生の作り込み成果を利用しないでゼロから作り込みを開始する場合もある。すなわち，優れている成果（モノ）であれば，さらに次年度の学生によって利用されて発展的なモノになることを示す。この授業展開では，後工程に費やす時間が増えるほど，構造を深く理解し複雑な作り込みをしようと考えている評価もできるが，十分に発展させることができないまま終わる学生もいる。多年度の作り込み方式が全面的に有効であるとは言えないが，学生の優れた作り込み成果が多く見られる。

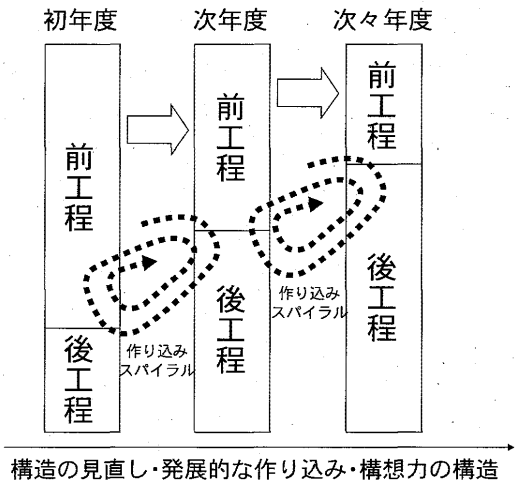

図７　多年度にわたる作り込みの時間配分変化イメージ

３．２　前年度学生によるミニ講義による後工程の促進

プログラミング授業の年次進行にともない後工程における発展的な作り込みには，構造の見直しと構想力が必要になる。発展的な作り込み構想では，新たな知見が加わり新たな機能も作り込まれることがある。新たな機能では，前年度の作り込み全体を分析するなど，新たな機能の作り込みのために前年度学生の意見も求めることも見られる。また，図８のように前年度学生による作り込みの構想案をミニ講義で再現するなど，前年度学生間の交流も多くなり，現学生の作り込み組織は前年度学生も参画して拡大的になる。例えば前年度の学生のミニ講義による資料は図８のようになる。前年度学生の作り込み案が再現されることで，現学生の作り込み構想は，次第に幅を広げて授業で解説した教員のサンプル・プログラムによる作り込み構想を拡大していくようになる。よって，後工程の作り込み構想は，図９のような拡大構想で進行する場合がある。この拡大構想は，図７に示した作り込みスパイラル現象と呼び，前工程と後工程のリソースを多年度にわたり統合したモノづくり進行方策でOJTにはない大学独自の展開となった。

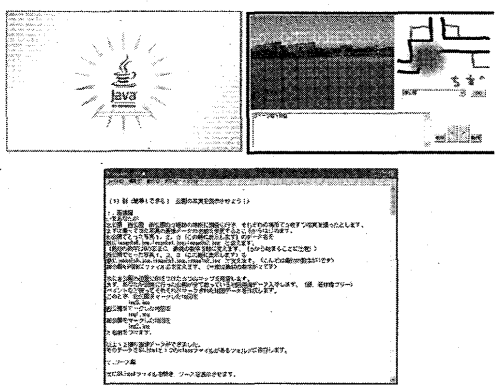

図８　前年度学生のミニ講義資料とサーバ内のReadme

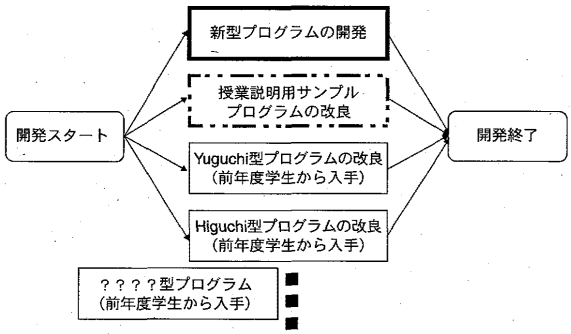

図９　最終課題の作り込み方策の拡張選択の機会イメージ

４．学生の作り込み主体組織と学生相互の成果評価

プログラミング授業における最終課題では，５人の学生を１グループとした組織編成をする。グループによる作り込みでは，組織内の各メンバーがどのくらいの貢献をしたのかが成果（評価）に影響してくると考えられる。各メンバーの貢献度を評価するには，多くの観点を作り込み期間中に見ていく必要があるが，評価の観点や評価法は組織編成されたグループ毎に決定させることにした。グループの組織編成では，各グループの４ヶ月間における独自の目標が決定される。本授業における単位認定の

評価方式は，2つの評価点を参考にしており，第1評価は，グループの成果発表による評価点と，第2評価は，グループ内の学生相互の評価点を参考にしている。2006年度の授業の場合は，45人の受講学生において5人1グループの組織編成で9グループをつくり評価した。評価に際してグループでは，プログラムの説明・デモンストレーションと社会調査の成果発表をして，そのグループの発表に対する評価を他の8グループの学生（40人）が100点法で評価する。そして，発表終了後は，そのグループ内の学生どうし（5人）で，4ヶ月間の期間を振りかえりながら，100点法で評価する。いづれの評価も評価観点を記述してグループ内部と外部評価の平均を教員側で集計して授業における単位認定の参考にしている。

4．1　作り込みグループの結成フロー

最終課題の作り込みグループは図10のような流れで部分的な合意形成から組織編成されていく。例えば2006年度の作り込み組織では，45名の受講生に対して9グループを組織編成するために，まず，受講生の中から9名の仮リーダを教員が抽選する。抽選においては，仮リーダ資格条件は問わない。仮リーダの役割は，1グループを組織編成するために残りの36名の学生の中から，まず，1人を抽出することである。仮リーダ以外の36名の学生には，5㎝×10㎝の紙（以下アピールタグと呼ぶ）に自己紹介コメントを記入させる。自己紹介アピールタグの内容は，学生番号と名前は記入させないで，それら以外は自由に記入させる。すなわち，グループの結成過程は，仮リーダを含む5人がアピールタグと面接による合意形成で順番に決定されていく。プログラミング授業に限らず，何かの課題に取り組むためのグループ編成の方法は，組織力を発揮させる上で学生の合意形成による編成が好ましいと考えられる。本授業の編成方法は，仮リーダ（1人目）の観点が2人目の選出時に強く影響するが，3人目の選出には，仮リーダとの合意形成が必要であり，以後，選出学生との合意形成で5人が決定される。

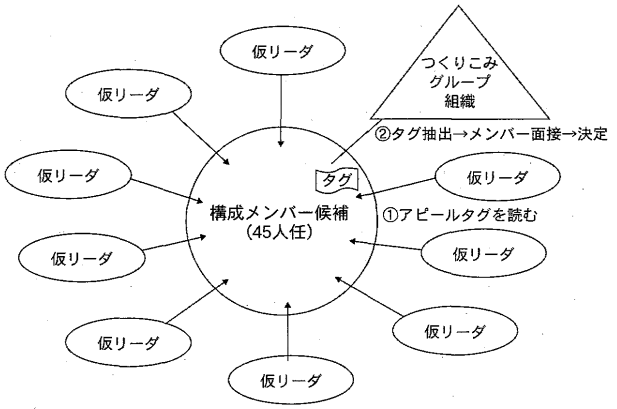

図10　学生グループ組織編成の抽出決定イメージ

図10のような方法で作り込みグループ組織を編成していくと図11のような9グループが結成される。この9グループ毎に，コンピュータ演習室の座席位置が決定される。本授業実践では，9グループの最終課題として経済学部に適したテーマを選定して，社会調査にもとづく情報発信システムの構築を目標にした。社会調査の対象エリアは図11のように，大学の立地する市内をグループの個数（2006年度は9つ）にゾーニングして，東西南北方向の約10㎞圏に定めて社会調査の範囲がグループで重複しないように決定させた。

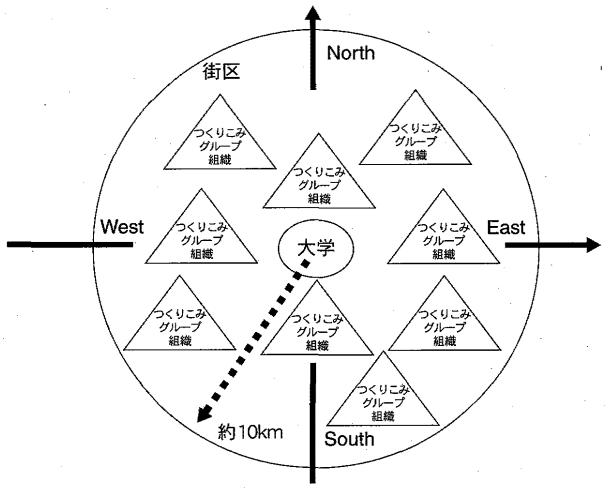

図11　学生グループの社会調査範囲をゾーニング

4．2　作り込みグループにおけるSECI改効果の期待

学生の合意形成によって組織編成されたグループの作り込み支援には，Web/ファイルサーバを情報共有サーバとして使用させている。本授業実践では，プログラミング言語の基本文法以外に作り込みにおける組織論を意識させるため，ウォータ・フォール型～エクトリーム・プログラミングにおける一連の作り込み組織論を講義する。また，作り込み創造論としてブレーンストーミングやKJ法も講義する。そして，両論あわせて知識創造過程の一般論を図12のように改変して講義する。図12のSECI改モデルは，グループによる作り込み組織の合意形成のための情報－知識変換のイメージとして解説している。解説では本来のSECIモデルのような暗黙知と形式知の定義を明確に意識させないが，作り込みの会議ノートや

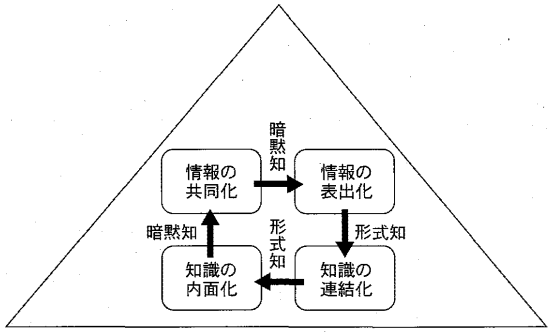

図12　作り込み組織の情報－知識変換用SECI改モデル

合意形成内容をマイクロソフト Visio によるブレーンストーミングと KJ 法を活用してまとめるように指導した。

5．後工程重視 See, Java 型授業テーマの選定

プログラミング授業の最終課題のテーマ選定には，ゲーム等のシステム開発も有用であるが，グループの合意形成や社会調査，後工程の作り込み構想過程に着目させるような組織的モノづくりを選定した。本授業のテーマの幅は図 13 のように大きな範囲になるが，プログラミング授業のテーマとして選定するには，テーマの大小規模と深さの関係に注意が必要である。すなわち，小規模テーマでも深い（長い）改良を積み重ねていくような最適化理論や処理アルゴリズムの開発であればテーマの幅を広げる必要はないと考える。また，本稿の授業テーマのように作り込みが浅い場合でも，社会調査などが大規模に要するのであれば作り込みを深めることは必要なく，図 13 のような四角錐を構成する各辺の合計の長さが等しければテーマに取り組む時間は両型で等しいと考える。すなわち，本稿の最終課題のテーマでは，作り込み以外に幅広い範囲の情報を調査して知識変換など合意形成による時間を要する。

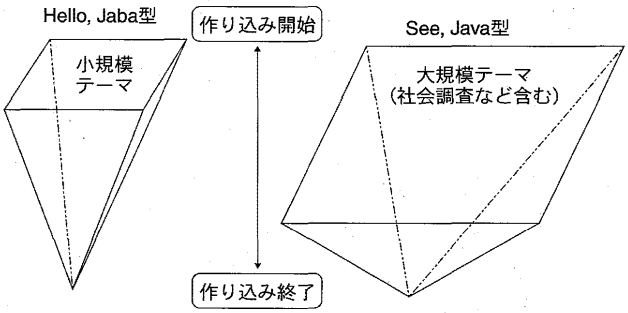

図 13　テーマの規模と作り込み深さ（長さ）の考察

図 14 のように，本授業の最終課題テーマは単年度で深い作り込みは難しいが，多年度の継続テーマのため，調査範囲も広がり，深い作り込みも可能であるため，図 14 のような深い作り込みが 3 年後には達成されている。これは，該当学生と前年・前々年度学生の共同参画で構想力が相互に結びつき，深い作り込みが達成されることによる。以下，次節に学生の作り込み成果を紹介する。

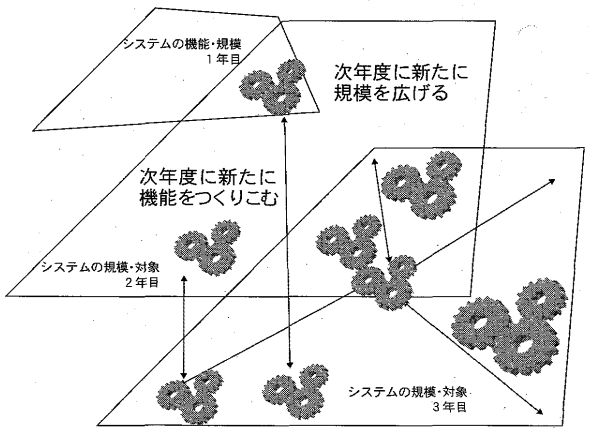

図 14　多年度にわたる深いつくりこみ

5．1　地域情報を活用したポータルサイトの作成

2003 ～ 2004 年度学生は，広く地域情報を活用するテーマを設定した。テーマにもとづく社会調査としては，地域の商業施設や観光施設を取材してポータルな情報発信システムを構築した。ポータルな情報発信の成果例を一部紹介すると，地元の商店と観光情報が一体となったものや，観光案内用のルート案内を含む機能（図 15 における少年の体の部分をクリックしたイベントログをシステムがカウントして，どのような地域の観光にユーザの関心があるかなど収集分析する）を 2 年間にわたり作り込んだ。

この社会調査を含む最終課題テーマの意義は，学生が地元の街を歩き，さらに地元の商店主と関わりあうことで経済学部の学生の能力を活かした地域貢献として多くの注目を集めた。

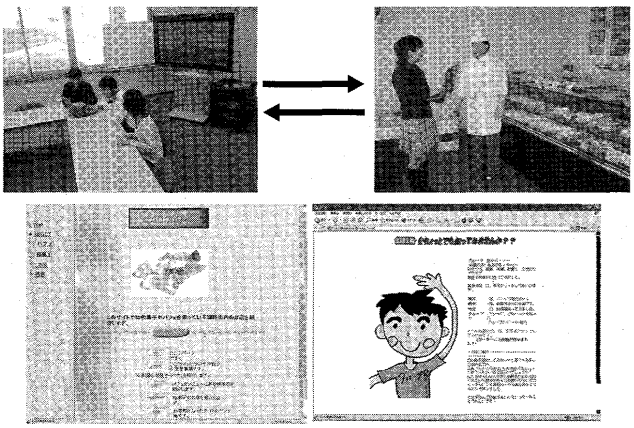

図 15　学生の社会調査の様子と情報発信システム

5．2　安全・安心な街づくり－避難行動プランニング

2005 ～ 2006 年度学生は，2003 ～ 2004 年度学生と同様に地域情報を活用した大規模テーマであり，「被災時における避難場所までの行動プランニングを考える」ということを選定した。本学の立地は，過去の 10 年を振り返るだけでも度重なる大地震にみまわれ，将来も大地震発生の確率が高い地域であることから，学生にも身近で関心の高いテーマである。テーマの規模としては，2004 年度よりも大規模で，かつ，新しいテーマであることから，前年度成果に関するファイルが使用できないことを考えて，プログラミング言語のソースコードを読み解くための参考として授業支援サーバに過去の成果を 1 リンクのみ設けた。しかし，教員の考えとは異なり多くの学生がそのリンクで過去のファイルを閲覧した。閲覧した学生によると，前年度とのテーマは違うが，システムの作り込み過程・失敗ソースコードやユーザイン

ターフェースのデザインなどを参考にしており，テーマが異っても，作り込みや構想過程の学習に過去の作り込みリソース利用されたと推察される。

社会調査に関しては各グループが市内の担当区域をゾーニングして93箇所すべての避難場所の特性を調査している。調査例では，被災時における住民の安全が確保される屋外避難場所までの経路シミュレーションをJavaで作り込んでおり，安全性や避難誘導について詳細に検証している。最終課題の成果の一部を示す図16では，避難場所までの経路や弱者の視点を分析して，避難場所までのシミュレーション機能をJavaアプレットで実証実験している。このシミュレーションには，実際のクルマ椅子を用いて避難場所までの経路を走行し，その画像を保存して，Java機能を用いて地図上に同期することで，詳細に再現したシミュレーションである。また，図17のような避難経路の環境変化の検証では，昼と夜の風景の違いによる画像を収集して，避難経路の問題点を考察している。紹介例で示したように，地域の大学に通いながら，地域のことを考えるテーマは，学生の情報収集能力や地域貢献をおこなうきっかけにもなった。また，学生の住んでいるところで土地勘を活用した避難プランニングは，行政や自治体の防災担当者の見落とされがちな箇所も検討しておりプログラミング授業の学習の成果を地域貢献に活用している。

これらの取り組みは，プログラミングの授業の単位取

図16　2005学生の避難場所案内サイト

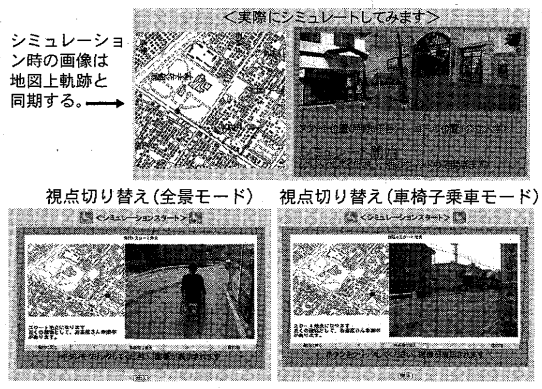

図17　2005学生の避難場所案内サイトの作り込み機能

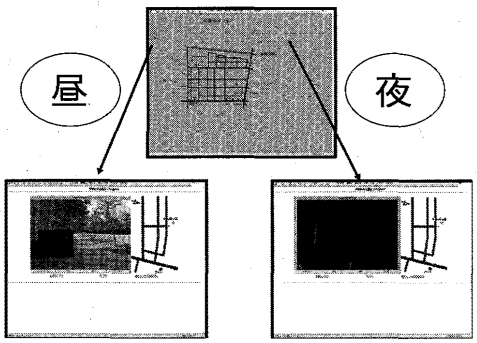

図18　2005学生の避難場所案内サイトの作り込み機能

得で終わるだけでなく，安全な街づくりの社会貢献として地域住民をはじめ行政・自治体に高く評価されている。

6．授業をきっかけに生まれた学生サークル

2003～2005年度までの授業展開における最終課題のテーマでは，多くの地域情報を収集し分析してまとめられた。これらの成果は，学内ネットワークで公開されており，入学生の防災意識の向上に閲覧活用されている。そして，これらの学内ネットワークで公開されている有用な成果を，地域住民のための情報として整理して学外向けに公開したいという要望が授業後から学生達のあいだで高まり，図19に示すようなWebサイトが作られて2006年2月に「地域情報研究部」という学生サークルの結成と同時に，現在も運営されており，日々，地域で活動している。

他の教科もプログラミング授業と同じく，多くの成果

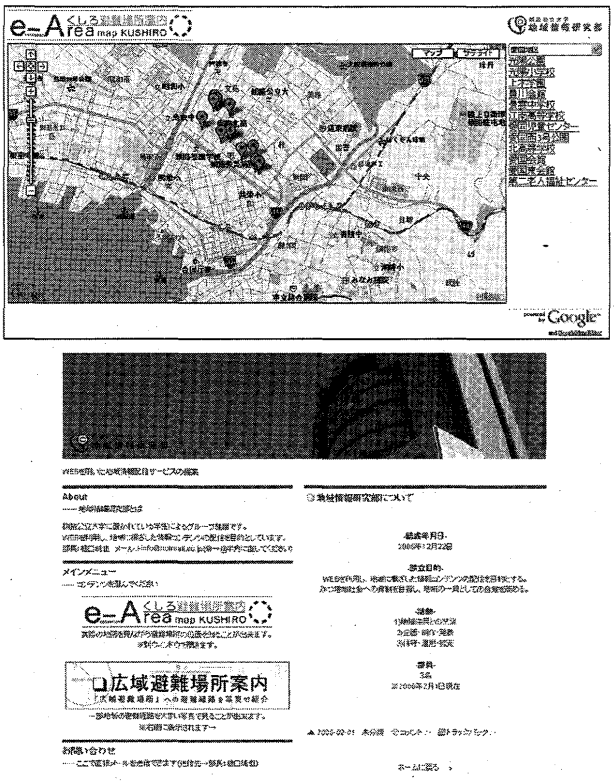

図19　公開された避難場所情報と地域情報研究部のサイト

や期待がある。他の教科の成果では，授業と試験を受けて単位を取得して完結することであることが一般的であり，学生は単位を取得することは大学卒業に必要不可欠であり，教員は単位取得の筋道を作るだけで良いと考えていた。しかし，授業終了後あるいは単位取得後も継続的に授業での成果を社会あるいは地域のために活用するという学生の思いは，地域情報研究部の発足で一部の教員の意識も変わったと考えられる。このような展開は異例ではあるが，今後，プログラミング教育あるいは情報教育のめざす一目標として据えることも可能である。つまり，プログラミング授業や情報リテラシ教育のテーマは「社会や地域で貢献するきっかけづくり」としても活用することが可能であると考えられる。

7. ま と め

本稿で述べた授業実践における展開方法は，前工程と後工程という表現を用いたが，過去のプログラミング教育の方法論を否定することではない。プログラミング言語によるモノづくりに限らず，どのように作り込むのかという本質的な方法論は，学生が自ら学び作りあげていくと考えられる。過去の授業実践では，冗長的に基本に着目させるようなサンプルソースコードの入力で始まりそして終わる授業展開をしてきたが，昨今の学生の期待を保持することは難しい。本稿では，「See」を切り口に，過去の学生成果のソースコードの失敗事例も見せながら，テーマの選定や規模，作り込み工程を考察した。

また，本授業の作り込みグループの組織構成では，時間をかけて複雑な方法を実践したが，ソフトウェアを作り込む企業では，プログラミング能力だけでなく，作り込み過程を発展的に推し進めるような組織構成が必要であり，どのような組織化がいいのかを管理者が日々悩んでいる。この悩みは，組織が良くないと，その成果も良くないことが多いと考えられる。企業では，各ソフトウェアの作り込み工程に適切なメンバーが組織化されており，各作業はシェアしており，近年では，エキストリーム・プログラミングなど，開発を短期で終わらせるような組織構成もIT企業で導入されている。大学のプログラミング教育では，そのような企業事例に等しくあわせる必要性はないが，失敗してもいいからという意識と，構想的な作り込み方法と組織で，モノづくりを経験させることは今後も重要である。つまり，モノづくりをする日本の位置づけを将来にわたり維持するためにも重要である。近年，日本のソフトウェア開発は欧米だけでなくアジアにアウトソーシングする割合が高くなっていることは，モノづくりを推進する人材が日本で育っていないことを意味する。これまで，ソフトウェア開発というモノづくりは，理工学系の分野出身の人材が必要不可欠と言われてきたが，実際，JavaやC#言語の仕様を駆使することで，人文系出身者でも高度で柔軟な構想力で高度なモノづくりが可能であると考えられる。このような背景からIT系の企業は文理の専攻分野を問わない学生を今後も求めてくると考えられる。

参考文献

[1] 野中郁次郎・竹内弘高・梅本勝博, 知識創造企業, 東洋経済新聞社, 1996

[2] 高橋誠, 新編創造力事典, 日科技連出版社, 2002

[3] 結城浩, Java言語プログラミングレッスン上・下, ソフトバンクパブリッシング, 2000

[4] 鈴木正人, ソフトウェア工学, サイエンス社, 2003

[5] 山田健志, オブジェクト指向ワークブック, 翔泳社, 2003