Vol.2010-EMB-19 No.7
2010/12/6

【中間報告】スパイラルカリキュラムによるソフトウェア工学教育

山崎　進[†1]

学部生から大学院生を対象とした，プログラミング演習を通してソフトウェア工学の原理・原則を自然と学べるようした e ラーニング主体の授業を開発している．大きな特徴は，教育工学の分野で外国語学習などを中心に適用されているスパイラルカリキュラムを用いていることである．この成果は今年秋から新規開講したオブジェクト指向プログラミング演習で活用されている．今後，教育効果の評価と他のソフトウェア工学関連科目への展開を予定している．

Software Engineering Education with Spiral Curriculum

Susumu Yamazaki[†1]

We are developing e-learning-based software engineering curriculum to enable undergraduate and graduate students learn spontaneously software engineering disciplines through programming exercises. We build our curriculum based on the spiral curriculum model, which is a concept in educational technology, and is applied to foreign language education. We are applying them to object-oriented programming tutorials and exercises that just begined newly in this autumn. We will evaluate its effects and apply to the other software engineering lessons.

†1 北九州市立大学
University of Kitakyushu

1. はじめに

1.1 背　　景

北九州市立大学国際環境工学部情報メディア工学科 (以下，本学と略記) は，通信，メディア処理，VLSI 回路設計，制御，ソフトウェアといった比較的広い範囲の専門分野の教員が集まった学科である．本学には 3 つの履修モデルがあり，その 1 つとして組込みシステムを挙げている．ソフトウェアとして組込みシステムに焦点を当てている理由は，本学教員の専門分野のうち，とくに VLSI 回路設計，制御の専門家がいるため，シナジー効果を狙うならば組込みシステムの方が有利だと判断したためである．

本学のソフトウェア関連科目を**図 1** に示す．このカリキュラムの特長は，大きくソフトウェア工学とコンピューターサイエンスの 2 つに大別されることである．この理由は，組込みシステムに関連が強いのはソフトウェア工学であること，ソフトウェア担当教員 (2 名) がコンピューターサイエンス出身で，特にそのうちの 1 名がコンピューターサイエンスの知見を生かしたソフトウェア工学の問題の解決を研究課題としていることである．そのため，大学院科目の「組込みソフトウェア」「組込みシステム開発演習」「ソフトウェア検証論」の 3 科目は，組込みシステム開発の課題を形式手法を中心としたコンピューターサイエンスの成果で解決することを狙いとしている．

ソフトウェア工学関連の科目は計算機演習 I・II で C 言語のプログラミング演習を行う

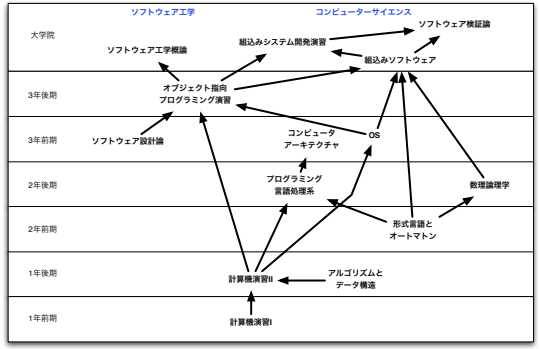

図 1　ソフトウェア関連科目の依存関係
Fig. 1　The dependencies of software-related classes

ことを出発点として，ソフトウェア設計論でオブジェクト指向モデリングを学習し**オブジェクト指向プログラミング演習** (以下，**OOP 演習**と略記) でモデリングとプログラミングの両方を学習する，PBL (project-based learning) 的な演習科目を配置している．OOP 演習での体験を問題意識として，大学院のソフトウェア工学概論で理論的な背景を体系的に学習することを意図している．本稿では OOP 演習について報告する．

OOP 演習は 2010 年秋に新規開講する 3 年次後期配当の週 90 分連続 2 コマの科目である．

1.2 OOP 演習への要求

OOP 演習に課せられた要求を次に示す．

(1) 大学院科目に対する出口
- (a) **ソフトウェア工学の重要な原理・原則を体験できること．** 大学院科目のソフトウェア工学概論で講義する理論の「ありがたみ」が実感できるようにすることが求められる．
- (b) **モデリングとプログラミングを両方学習すること．** 大学院科目の組込みシステム開発演習で扱うモデル駆動開発への布石とする．

(2) 社会に対する出口
- (a) **ソフトウェア開発の工程を一通り体験できること．** 学部で卒業し就職する学生に対しても，社会に求められる水準以上のソフトウェア開発能力を身につけさせることが必要である．
- (b) **最近のソフトウェア開発で用いられる開発環境のリテラシーが身に付くこと．** 統合開発環境や構成管理システムなど広く普及したツールを一通り扱う．
- (c) **プログラミング言語・環境に依存する知識と依存しない普遍的な知識を明確に区分すること．** ソフトウェアは日進月歩なので，長期にわたって普遍的に通用する知識は，短期にめまぐるしく変化する知識から明確に分離したい．

(3) それまでに履修する科目の制約
- (a) **C 言語の文法を一通り理解した程度のプログラミング能力の復習から出発すること．** 1 年次の計算機演習は現状では C 言語の文法レベルの学習しか行っておらず，しかも学年が離れているので学生が内容を忘れている可能性もある．
- (b) **基本的なアルゴリズムとデータ構造の復習から出発すること．** アルゴリズムとデータ構造についても同様の事情である．
- (c) **UML を使うときには解説を交えること．** ソフトウェア設計論では UML の基本的な文法を学習しているが，選択科目なので全員が履修しているとは限らない．また，UML の詳細な意味論やモデリングの実例は扱っていない．

これらの要求を全て満たすのは容易ではなく，旧来の伝統的な講義や演習では十分に達成することができない．そのため，根本的なアプローチから見直す必要がある．

1.3 アプローチ

教育効果を最大限発揮するためには，学生が自分のペースで学習できることが不可欠だと考えた．Christensen らは，教育に関して破壊的なイノベーションが起こるとしたら，学生の学習スタイルとペースに合わせた個別の学習教材が鍵となると予言している[3]．また鈴木も自習できる教材こそが教育効果を飛躍的に高めることができると論じている[7]．したがって，**学生が自習できる教材**を目指すことを根幹に据えることとした．

また，ソフトウェア開発のスキルは，プログラミング言語の文法とライブラリの語彙の修得が大きな要素を占める．そのことから連想し，外国語の学習方法として成果を挙げている**スパイラルカリキュラム**[2)5)] を採用する．

1.4 この後の構成

本稿のこの後の構成は次の通りである．第 2 節と第 3 節では，第 1.3 節で取り上げた 2 つのアプローチ，学生が自習できる教材とスパイラルカリキュラムについて，よりどころとなる文献をそれぞれ紹介する．第 4 節では，第 1.2 節で示した要求に対して応える，OOP 演習の主な学習目標を挙げる．第 5 節では，実際にスパイラルカリキュラムの考え方に基づいて配置した詳細な学習目標を中心に示す．まだ OOP 演習は開発しながら実地で適用している段階なので，最後に第 6 節で今後の予定を示し，第 7 節で中間的な総括を行う．

2. 教材設計マニュアル

教材設計マニュアル[7] は，最新のインストラクショナルデザイン (instructional design: ID) 理論の知見を教材作成の初心者にも理解できるような形で集約している．

インストラクション (instruction) は，狭義の「教えること (teaching)」にとどまらず，たとえば教材を選択する，学習者の準備状況を見極める，クラスの時間進行を管理する，教授活動をモニターするといった，より広範囲の教育関連活動も含んだ概念である[4]．ID の中心的活動は，文字通りインストラクションを設計することである．とくに e ラーニングのように事前にプログラム化された教材を開発するには ID は不可欠である．

教材設計マニュアルの基本方針を，ソフトウェア研究者や技術者に分かりやすく一言で比喩するならば，エクストリームプログラミング (XP)[1] だろう．

- 学習目標を明確化するために，学習目標を達成したか確かめるテスト問題をまず作成す

Vol.2010-EMB-19 No.7
2010/12/6

る．XP のテストファーストに相当する．

- 教材を作りっぱなしにせず，効果を評価・総括してフィードバックする．XP の反復開発に相当する．
- 教材の効果の評価のために，実際の学習者に近い協力者を得て「モニター」(形成的評価) を行う．XP のオンサイト顧客に相当する．

教材設計マニュアルの開発手順を次に示す．

(1) **出入口を決める．** 学習目標の明確化とテストの作成
(2) **中の構造を見極める．** 課題分析図の作成
(3) **教え方を考える．** 指導法略表の作成
(4) **教材を作る．** 教材の開発と形成的評価の準備
(5) **教材を改善する．** 形成的評価の実施と教材の改善

本稿では，主に学習目標に焦点を絞って議論をする．学習目標は依存関係をもつグラフ構造である．教材の分析段階で学習目標グラフの明確化が求められている．

3. スパイラルカリキュラム

スパイラルカリキュラムは Bruner が提唱した概念である[2]．**図 2** はスパイラルカリキュラムのイメージを説明している．スパイラルカリキュラムでは 1 回の授業で一通りのことができるよう，全工程の基礎的な学習目標を一通り修得させる．次の授業では基礎的な学習目標を復習・定着させながら，発展的な学習目標を追加し学習を進める．この過程でだんだん大きな例題にとりくめるようにする．これを図で表すと，スパイラル，つまり螺旋状に学習目標が広がっていくことからスパイラルカリキュラムの名が与えられている．

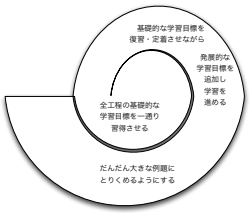

図 2 スパイラルカリキュラム
Fig. 2 The spiral curriculum

語学教育では，基礎的な例題から発展的な例題へ，文法と語彙の観点からだんだん表現力が増していくような教育が望ましい．これはスパイラルカリキュラムの特性と非常に相性がいいことから，語学教育，とりわけ外国語教育の分野でスパイラルカリキュラムが用いられてきた．具体例としては Richards と Gibson による GDM (Graded Direct Method)[5] が有名である．これは英語を母語としない人が英語を学習するための教材で，誰にでも理解容易な絵を示し，それを表す単語や文章を英語で記述した絵本である．最初は I, You などの代名詞から始まり，次に遠近感のある絵で近くと遠くを表し，それぞれの人を指差して “I am here,” “He is there,” という文章を絵で説明するという具合に進行する．1 つ 1 つの絵で会話は完結しており，先に進むに連れて新たな文法と語彙を学習する．

4. OOP 演習の学習目標

我々が定義した OOP 演習の主な学習目標を**図 3** に示す．図中のグラフ構造は頂点が学習目標で辺が依存関係を表している．上下については，下の方がより基礎的で上の方がより発展的な学習目標である．(pre) と示した学習目標は OOP 演習を受講する以前に履修する科目で学習する (はずの) 学習目標である (これらは要求 3 に該当する)．OOP 演習で特に重視すべき学習目標を赤字で示している．

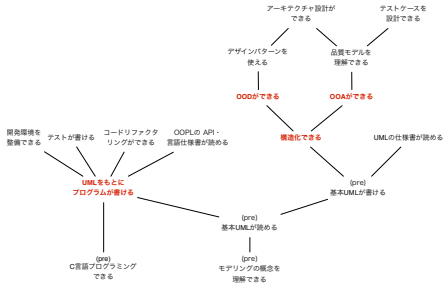

図 3 OOP 演習の主な学習目標
Fig. 3 The goals of the OOP tutorials and exercises

要求 1b に対応し，グラフ構造も左側のプログラミングスキルと右側のモデリングスキルの 2 つに大別される．プログラミングスキルの大きな目標は「UML をもとにプログラムが書ける」である．ここで特定のオブジェクト指向プログラミング言語 (たとえば Java) と明記していないのは要求 2c のため，つまり特定のプログラミング言語だけに依存しないように作るためである．プログラミングスキルの枝葉の部分の随所に要求 2b に対応する学習目標を掲げている．

右側のモデリングスキルにおいて重視すべき学習目標「構造化できる」「OOD ができる」「OOA ができる」の詳細は，要求 1a と対応関係をとるように設計する．これらの詳細な学習目標については SWEBOK 2004[6] を参考にした．なお OOP 演習の開発当初は SWEBOK 2010 は策定中であった．

5. OOP 演習の設計

スパイラルカリキュラムは第 3 節で示したような特性を持つことから，モデリングやプログラミングなどの表現力の問題を扱う OOP 演習に適用すると効果が高いだろうと我々は考えた．これにより要求 2a を満たすことを狙う．

スパイラルカリキュラムを構成するには，まず学習目標を列挙し，何かの例題に即して説明する．次に依存関係を分析し，無理なく学習が進むよう順序関係を定義する．

現在 OOP 演習は開発中であり，スパイラルカリキュラムの特性の 1 つ，**無理なく**学習目標を広げていく必要性があることから，教材の作り込みの段階で学習目標を細かく調整をする必要がある．そのため，執筆時点 (2010 年 11 月上旬現在) で教材として確定した部分だけの学習トピックのタイトルを**図 4** に示す．

この後は大まかには次のように展開する予定である．

- 調査したアルゴリズムを UML で表現して設計書を記述する
- テストファーストでモデルを開発する
- リファクタリングでイベント駆動型のプログラムに変更する
- コントローラーを状態機械で実装する

この演習は moodle を用いた e ラーニング教材として開発しており，学生は自らの理解度に合わせて進度を調整する．これにより授業の進度を全体で調整する必要はなくなった．

この演習が完了した後はグループワークに移行し，学生が考案するアプリケーションを開発する．これにより個人ごとの e ラーニング教材だけでは修得できない，実開発で行われるチーム開発を体験することができる．さらに早く終わった学生は，後から開発を始めたグループに加わって指導の立場に回る．俗に指導するには相手の 10 倍は理解する必要があるといわれるが，指導することでより深い理解をさせることを狙う．

6. 今後の予定

OOP 演習は本学において現在も実施中である．各トピックにてアンケートを行っており，学生からの意見にもとづいて調整を図っている．この結果を統計的にまとめ，OOP 演習の教育効果を測定することを予定している．この分析が完了した時点でようやく一通り完成した事例研究として世に問うことができるだろう．

2012 年度には本学の演習室がリニューアルする予定である．それに伴い，プログラミング言語・環境が大幅に入れ替わる可能性がある．これを機会に要求 2c に教材が本当に対応できるのかが問われるであろう．その評価については，後日論文化する予定である．

OOP 演習の成果は，初めてプログラミングを学習する科目である計算機演習や，より高次の学習を行う大学院科目であるソフトウェア工学概論，組込みシステム開発演習にも順次適用していく．個々の科目の効果・効率の向上が期待できることから，長期的には学習内容の充実などカリキュラム全体を見直す活動を行いたい．

(1) OOP 演習の意義と目標
  (a) OOP 演習の意義と目標
  (b) 例題「電卓を作ろう！」
  (c) 小テスト: OOP とは?
(2) 計算結果を表示する
  (a) IDE を使ってみよう
  (b) 適切なプロジェクト名をつけてみよう
  (c) 計算結果を表示する
  (d) 課題: 四則演算
  (e) 課題: 2 次方程式
(3) さまざまな場合を想定しよう
  (a) さまざまな場合を想定しよう
  (b) 版管理システムを使おう (1)
  (c) リファクタリング (1)〜メソッドの抽出を行おう
  (d) xUnit を使おう (1)
  (e) テストを設計しよう (1)
  (f) 課題: 2次方程式 (解の判別式の同値クラス)
  (g) 課題: 2次方程式 (その他の同値クラス)
(4) GUI を作ろう
  (a) GUI を作ろう
  (b) 課題: 電卓の GUI
(5) 開発方針を立てよう
  (a) 開発方針を立てよう
  (b) アーキテクチャ(1) MVC
  (c) 小テスト: MVC アーキテクチャ
  (d) アルゴリズムを調査しよう
  (e) 課題: アルゴリズムの調査
  (f) 版管理システムを使おう (2)
  (g) 図のあるレポートを作成しよう

図 4 開発済みの OOP 演習の学習トピック

7. おわりに

本報告では OOP 演習への要求を示し，アプローチとして教材設計マニュアルの提唱する「自習できる教材」とスパイラルカリキュラムの適用を提案した．OOP 演習の学習目標を分析し，執筆時点までに開発済みの教材のトピックを紹介した．最後に今後の予定を示した．

謝辞 熊本大学の鈴木克明先生から多くの助言を頂きました．鈴木先生の助けにより OOP 演習をこのような形で開発・発表できました．ここに深く感謝の意を表明します．

参 考 文 献

1) Beck, K. and Andres, C.: *Extreme Programming Explained: Embrace Change*, Addison-Wesley Professional, 2nd edition (2004).
2) Bruner, J.: *The Process of Education, Revised Edition*, Harvard University Press, 2nd edition (1976).
3) Christensen, C., Johnson, C. and Horn, M.: 教育×破壊的イノベーション教育現場を抜本的に変革する，翔泳社 (2008).
4) Gagné, R.M., Wager, W.W., Golas, K.C. and Keller, J.M.: インストラクショナルデザインの原理，北大路書房 (2007).
5) Richards, I. and Gibson, C.: 絵で見る英語 BOOK，Vol.1, アイビーシーパブリッシング (2006).
6) Society, I.C.: Guide to the Software Engineering Body of Knowledge (SWEBOK), `http://www.computer.org/portal/web/swebok` (2004).
7) 鈴木克明：教材設計マニュアル - 独学を支援するために，北大路書房 (2002).