Vol.2016-IS-137 No.2
2016/8/25

非情報系の学生を対象としたソフトウェア開発演習の設計と改善

居駒幹夫[†1]　高橋英男[†1]　西村勝彦[†2]　平野敏行[†2]　恒川直樹[†3]　佐藤文俊[†2]

概要：非情報系の理工系大学院生を対象として実施している「実践的シミュレーションソフトウェア開発演習」の取組みを紹介する．流体力学，分子動力学の本格的なシミュレーションソフトウェアを，信頼性や保守性などのソフトウェア工学の知識も考慮して開発できるスキルを身に付けることを目標とし，大学のシミュレーションの専門家と，企業のソフトウェア工学の専門家が協働して教育設計を行った．本教育は 2009 年以来 8 年間継続，改善しており，受講者が保守性の高いシミュレーションソフトウェアを開発することを可能にしている．

キーワード：シミュレーションソフトウェア，教育，PBL

Instructional Design and Improvement of a Software Development Practicum for Students of Non-Information Engineering

MIKIO IKOMA[†1]　HIDEO TAKAHASHI[†1]　KATSUHIKO NISHIMURA[†2]
TOSHIYUKI HIRANO[†2]　NAOKI TSUNEKAWA[†3]　FUMITOSHI SATO[†2]

Abstract: This paper introduces a practicum of practical simulation software development for non-CS/SE students. Goal of the practicum is to acquire skill for developing full-fledged simulation software in the field of computation fluid dynamics and molecular dynamics, considering software engineering knowledge such as reliability, maintainability, etc. To realize the goal, academic experts in the field of simulation and corporate experts of software engineering jointly collaborated to make instructional design. This subject is being continued and improved for 8 years. Students who take the course acquire ability to develop high maintainability simulation software.

Keywords: simulation software, education, PBL

1. はじめに

シミュレーション技術を使用した情報システムは，コンピュータの実用分野としてその草創期から活用され続けている．昨今，スーパーコンピュータの演算能力の発展により，これまでは時間的に困難だった複雑なシステムのシミュレーションが可能になり，多大な時間と手間のかかる実世界での実験を，計算による仮想空間のみでシミュレーションできるようになってきた．さらに，実験結果の確認，予測だけでなく，これまで知られていなかった科学的発見の道具，さらには，工学や医学などの発明の道具としても活用されるようになってきている．

日本では，スーパーコンピュータのハードウェアとして，地球シミュレータ，京などのエポックメイキングなシステムを生み出している．しかし，シミュレーション分野で標準的に使われているソフトウェアのほとんどは欧米で開発されたものである．単に，すでに知られた事実を検証するレベルのシミュレーションの場合，他所で開発されたソフトウェアによって新たな知見が得られる場合もある．しかし，国家レベルで他国に対して優位に立てるようなシミュレーションの活用方法を考えた場合，ハードウェアとソフトウェアを組み合わせた最適なシミュレーション分野の情報システムを生み出すことが必要である．残念ながら，そのような情報システムを作る能力という観点で日本は欧米に対して競争力が不足しているのが現状である．

世界的に競争力のあるシミュレーション情報システムを開発するためには，まず，シミュレーションの対象となる分野の専門知識と，その基礎となる理学（特に物理，化学）の知識さらに，シミュレーションを実行する高性能計算（High Performance Computing）に関連するハードウェアや，ソフトウェアの知識が不可欠である．一方，シミュレーションの分野で広く活用されるソフトウェアは，全て，ソースコードの実行文行数として数万から数百万の規模を持っている．このような大規模なソフトウェアは，個人のレベルで開発することは極めて困難で，複数のソフトウェア技術者が共同して開発することが必要である．また，開発されたソフトウェアは,長期間にわたって,機能の追加，実行性能や扱えるデータ量のチューニング，バグの修正等の保守を要する．このように大規模かつ保守性を必要とするソフトウェアを開発するためには，高度な情報処理の知識が必要となる．現在の日本のシミュレーションソフトウェアの分野での大きな課題は，後者の大規模ソフトウェア

†1 (株)日立製作所 ICT 事業統括本部
ICT Business Division, Hitachi Ltd.
†2 東京大学生産技術研究所
Institute of Industrial Science (IIS), University of Tokyo
†3 東京大学分子細胞生物学研究所
Institute of Molecular and Cellular Biosciences (IMCB), University of Tokyo

を開発するための知識である．

本論文は，筆者らが 2009 年から継続している東京大学の理・工学系大学院生を対象に実施している科目「実践的シミュレーションソフトウェア開発演習」を紹介する．次章では，本演習で解決すべき課題を述べ，3 章，4 章では，科目の初期設計段階における，演習の構成，スケジュール，環境を述べる．5 章では演習を実施状況および顕在化した課題に対してどのように演習を改善してきたかについて述べる．6 章で演習の評価，考察を行い，7 章で残された課題，今後の展望を述べる．

2. 演習で解決すべき課題

情報処理を専門としない理工学系の学生を対象にすることもあり，一つの演習科目で大規模ソフトウェアを書くための知識スキルを習得させることは事実上困難である．一方，ソフトウェア開発の知識だけではシミュレーションソフトウェアの開発はできない．このような背景を踏まえ，演習の設計にあたっては，以下の三つの課題を立てた．

（1） ドメインスペシフィック
シミュレーションの分野でのソフトウェア開発の課題に対する現実的なソフトウェア工学的な解を学ぶ．演習で開発するソフトウェアも面白いだけのソフトウェアではなく，あくまでシミュレーションのソフトウェアとする．

（2） ライトウェイト
大規模なソフトウェア開発ノウハウは対象外とする．中小規模のソフトウェア開発で保守性の高いシミュレーションソフトウェアを開発できることを目標とする．

（3） プラクティカル
本科目を修了し，研究室でソフトウェア開発を行う際に，すぐに役に立つ実践的なソフトウェア開発ツールを使いこなせるようにする．

3. 演習科目の設計

本章では「実践的シミュレーションソフトウェア開発演習」の概要を説明する．他の大学や組織でも同様の科目を企画するときに参考になるように，鈴木のフォーマット[1]を参考に科目の設計内容を説明する．

3.1 科目のテーマ

科目全体のテーマは，スーパーコンピュータで動作するシミュレーションプログラムを複数人で共同開発するスキルを身に付けることである．シミュレーションや，ソフトウェア工学の知識を教えるだけでなく，それらの知識を使って，実際に動くソフトウェアを作るスキルを身に付けさせる．ただ，受講対象の学生の学部卒業時の情報処理の知識は限定されており，いきなり多人数の大規模ソフトウェアを開発するのは，時間的にも難しい．従って２，３人の学生が共同でソフトウェア開発を行い，中規模以上のソフトウェア開発に必要なスキルの基礎を身に付けさせることをテーマとした．

3.2 科目の受講者

受講者は，理･工学系の修士／博士課程の学生であるが，主に修士（博士課程前期）1 年目の学生を対象としている．

3.3 学習目標

本科目の目標は，単に受講者にシミュレーションソフトウェアやソフトウェア工学の知識を与えるだけではない．習得した知識を実際のソフトウェア開発に活用できることを目標とする．また，知識の本質部分を認知して学んだことを応用して現実の課題を解決できるようなスキルを身に付けることである．具体的に本科目ではまず，知識として，ハイパフォーマンス計算機，分子動力学，流体力学，ソフトウェア工学の概念，用語，数式などの基礎知識を理解する．続いて，習得した知識を基にソフトウェア開発関連の各種ツールを理解し，操作できるようにする．さらに，習得したソフトウェアの設計記法を使って自分が開発するソフトウェアを書き表せることができ，最終段階では，実際にスーパーコンピュータ上で動作させることを目標とする．ここで，必ずしも最終段階に至るために必要な知識は座学で全て与えることはせず，座学で教えた知識以外の知識が演習で必要になった場合でも，受講者が工夫して必要な知識を探し理解できることを目標とする．本科目で得た知識が，将来的に自分の研究環境で必要なものかどうか取捨選択ができることも期待している．

3.4 受講の前提条件

非情報系の理工学学部卒業時点でのスキルを考慮し，受講の前提条件は，以下の４点とした．

- シミュレーション対象となる物理化学の理解
- 情報処理の基礎知識（教養レベル）
- C 又は C++言語の言語文法の理解
- ソースファイルの編集とコンパイルができるだけの UNIX 系のシステムの操作

3.5 科目企画上の課題とその対応

講師

理想的には，ソフトウェア工学の知識，スキルを備えたシミュレーションソフトウェアの専門家が講師を務めるのが良いが，そのような人材は多くない．本演習では，シミュレーション系，HPC 系の講義は大学のシミュレーション系の専門家，ソフトウェア工学系の講師は産業界の技術者が分担して対応している．

膨大な知識の問題

最終的に，ソフトウェアを複数人共同で開発し，スーパーコンピュータ上で動作させるためには，シミュレーション対象の知識，スーパーコンピュータ関連の知識，プログラミングの知識，プロジェクト管理の知識など，膨大な知識が必要とされるが，限られた講義の中で必要な知識を習得するのは難しい．そこで，認知的方略[1]が重要となる．即ち，知識そのものを教えるのではなく，知識を得るために受講者がどのように工夫したら良いのかということを講義の中で示し，応用演習等でプロジェクト固有の課題に突き当たったときに自らが対応可能なようにする．

座学と演習部分の構成

本科目は，前半は演習に必要な知識を座学で教え，後半は，理解した知識を使って学生２，３人のグループが共同してシミュレーションソフトウェアを開発するという構成とする．このとき，前半で言語情報として理解された知識が運動技能として身につかないまま，後半の演習に入ると受講生が消化不良となり，負荷が後半に偏るという課題がある．このため，科目の前半部分にも，基礎演習として，座学で学んだ HPC 系，プログラミング系，開発環境系の知識を使った個人レベルのプログラミング演習を設定した．

4. 科目の講義構成，開発環境

4.1 講義構成

本科目の概略スケジュールを図 1 に示す. 講義の日数は，年度によって多少増減はあるが, 週一日 2 コマ (3 時間半) で１５週実施する．その前半部分は，ガイダンス，座学の講義，基礎演習を約 7 週間．後半の約 7 週間は，情報基盤センターのスーパーコンピュータを活用したソフトウェア開発演習を行い, 最終週に成果発表を行う. 図中の矢印は，各講義の知識がどのように連携しているかを示したものである．

前半の座学の講義は，大きく三分野，ハイパフォーマンス計算機の話，シミュレーションをする対象分野である流体力学と分子動力学の話，それに，ソフトウェア工学の話に分けられる．講義の分担は，計算機工学，シミュレーション分野の講義は大学側の講師が担当し，ソフトウェア工学系の講義は，企業側からの講師が担当する．前半部分で教える HPC とソフトウェア設計･実装の知識をもとに基礎演習として簡単なプログラムを各個人で書かせている．この段階では，スーパーコンピュータ用プログラムの開発スキルの修得を目標とし，シミュレーションソフトウェアとして必要なスケーラビリティ（スーパーコンピュータのノード数に比例した処理能力があるか）は評価対象としているが，開発するソフトウェアの信頼性，保守性は考慮させていない．

後半の応用演習では，講義で得たさまざまな知識を元に，学生がチームを組んでスーパーコンピュータで並列動作するシミュレーションソフトウェアを開発する．一チーム２～３名で，チームごとに流体力学，分子動力学のどちらかのシミュレーションソフトウェアを開発するかを選択する．開発言語は C 言語または C++言語で, 完成時の規模は, 2000～3000 行である．プログラムの骨格部分は講師側が開発したものを使用し, これを母体としてプログラムを開発する．単にコーディングをするのではなく，そのソフトウェア開発プロジェクトの計画，設計，プログラミング，テストを行い，大学側，企業側の講師が共同で受講生を指導している．応用演習においては，開発するシミュレーションソフトウェアが，動作することだけが目標では無く，シミュレーションソフトウェアとして必要なスケーラビリティ，信頼性，保守性を備えていることを評価対象としている．

最終日に，成果発表会としてチームとしての成果物のプレゼンテーションと個人のプレゼンテーションを行っている．

4.2 演習の開発環境

4.2.1 演習環境の概要

複数人の演習受講者が共同してシミュレーションソフトウェアを開発，管理するために必要な，プロジェクト管理

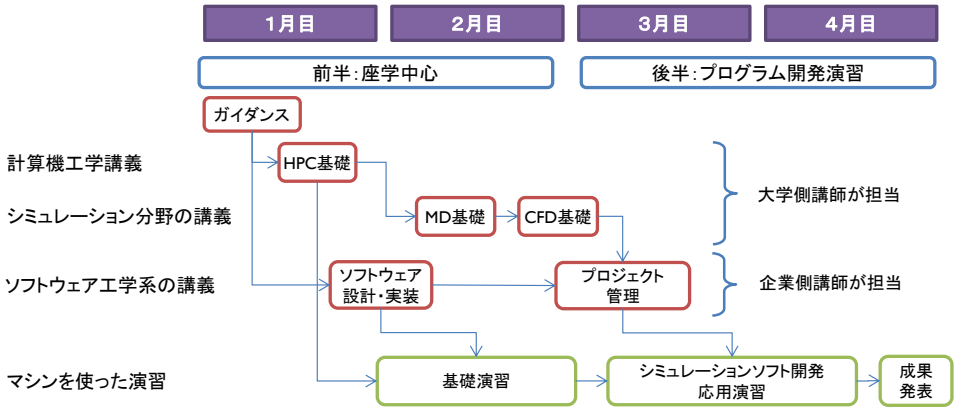

図 1　「実践的シミュレーションソフトウェア開発演習」の概略スケジュール
Figure 1 Outline schedule of the simulation software development practicum

ツール，バージョン管理システムを，演習室および研究室や自宅からアクセスできる環境を構築した．

使用するツールの選定にあたっては，情報処理系のスキルが高くない受講生が短い期間で習熟可能であること，科目修了後の受講生のスキルとなるもの（企業/学校で今後も使用する可能性の高いもの）を優先的に考慮した．続いて，運用側で導入の敷居が低いオープンソースで，世界的に実績のあるツールを考慮した．

ネットワーク構成としては，ソフトウェア開発演習という科目の性質上，演習の時間外にもマシンを使用する場合が多く，また，複数の受講者が共同してソフトウェア開発し，指導する講師も大学，企業に分散しているという本科目の特徴を踏まえ，演習時間外でも受講生や講師がコラボレーション可能な環境を構築した．

本科目で使用している開発環境の概略を図 2 に示す．

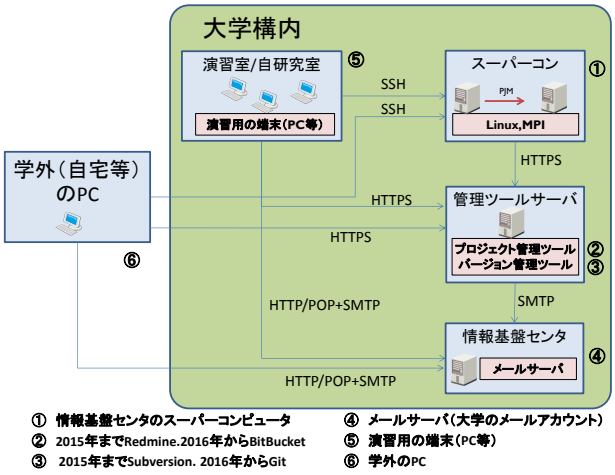

図 2　演習の開発環境
Figure 2 Development environments for the practicum

この中で，特徴的な構成は，演習科目の主催元である研究室に設置した管理ツールサーバと，その中で運用しているプロジェクト管理ツール (図中②)および，成果物のバージョン管理システム (図中③)である．これらのツールは，講義中でも講義外であっても，情報基盤センターの端末 iMac だけからでなく，研究室や自宅からもアクセスすることが可能である．例えば，プロジェクト管理ツールを使用してプログラム開発に関する作業を同じチームの他の受講生に依頼したり，見つけたプログラムのバグ情報をチーム内で共有したりすることができる．また，バージョン管理システムを使用して自分のコーディング，テストをしたソースコードをプロジェクトメンバで共有できる．こういった，複数メンバによるソフトウェア開発演習を円滑に実施できる環境を整えた．

4.2.2 使用しているツール

演習環境の方針に従い，表 1 に示すツールを採用している．

表 1　応用演習で使用しているツール
Table 1 Tools used for applied practicum

種別	使用ツール
プロジェクト管理ツール：チケット管理，コンテンツ管理，Wiki，コラボレーション等	2015 まで Redmine 2016 から BitBucket，Slack
バージョン管理システム：ソースコード，テスト，文書を格納	2015 まで Subversion， 2016 から Git
コンパイラ	gcc/g++/計算機メーカー提供のコンパイラ
ビルドツール	make（gmake）

コンパイラ，ビルドツールとして GUI ベースの統合開発環境（IDE）を採用するほうが生産性は高い場合が多い．しかしプログラミング経験の少ない受講生の IDE を習得するまでの習熟時間，スーパーコンピュータ上での開発という，シミュレーションソフトウェア開発演習特有の理由により，本演習ではコマンドラインベースのツールを採用している．

5. 演習の実施状況

本章では，演習の実施状況と，演習の実施過程で発生した問題および演習の改善策を述べる．

5.1 演習の実施概況

2009 年に演習を開始して以来，本論文執筆時点で 8 回の演習を実施している．2009 年の演習は，トライアルとして大学側講師の研究室のポスドク，博士課程後期学生を中心に実施したため，本章での報告は，2 回目の 2010 年から，8 回目 2016 年の演習の 7 回を対象とする．

表 2 に受講人数と応用演習のプロジェクト数をまとめた．受講生は，当初予定していた機械系の博士課程前期の学生の他に，マテリアル工学，応用化学，航空宇宙工学等シミュレーションを必要とする専攻の受講生が増えている．

表 2 受講生とプロジェクト数の推移
Table 2 Transition of number of students and projects

年度	‘10	‘11	‘12	‘13	‘14	‘15	‘16
人数	11	9	6	8	13	9	12
応用演習プロジェクト数	4	3	2	3	5	3	4

5.2 初期段階の課題

講師側の想定よりもさらに受講生の情報処理の知識が不足していた．具体的には，学部修了時点で，情報処理の

教養レベル（情報リテラシ，サンプルプログラムのコーディング経験）のスキルはあるが，OS の使い方，データ構造，オブジェクト指向といった基礎レベルのスキルが想定よりも不十分であった．図 3 は，本科目の 2011-2016 年の受講希望者（67 名）へのアンケートの回答の一部である．

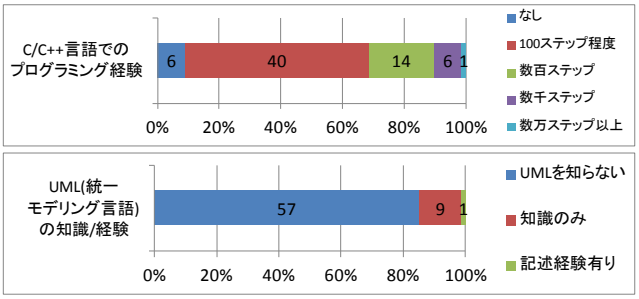

図 3　受講生に対するアンケート結果(一部)
Figure 3 Result of questionnaire for students (part)

このため，本科目の前半の座学講義の理解度が低く，後半の応用演習でもプロジェクトの企画も不十分で，成果物としてもプログラムの完成だけが目標となってしまい，特に 2011 年の演習では，本来の目標である保守性の高いソフトウェアとは程遠い成果物になってしまった．

この問題に対応して 2012 年から順次，座学，応用演習の両面で次のように改善している．

5.3 座学講義の工夫

前半部分の講義は，当初は受講者が情報処理の基礎レベルのスキルがあることを前提に，ソフトウェア開発の上流，プロジェクト管理から教えていた．2012 年の講義から，この部分を，（シミュレーションソフトウェアに特化した）情報処理の基礎レベルを中心の講義に変えた．また，演習中で教えられることは時間的に限られているため，シミュレーションソフトウェアの開発に特化した情報処理の知識をまとめた教科書「ソフトウェア開発入門 シミュレーション設計理論からプロジェクト管理まで」[2]を執筆、発行し，受講者が計算機の知識，データモデル，シミュレーションソフトウェアの処理方式などを必要に応じて自習できるようにした．

ソフトウェア工学，情報システム的な講義は，科目後半の応用演習の期間に受講者のレベルや，応用演習での進捗状況に応じて，受講者のプロジェクトが必要としている部分に特化してピンポイントで講義を行うようにした．すなわち，応用演習のプロジェクト開始時点でプロジェクト計画の講義を行い，プロジェクトの進捗に合わせて，レビュー，テスト，品質管理等の講義を短時間挟むようにした(図 4).

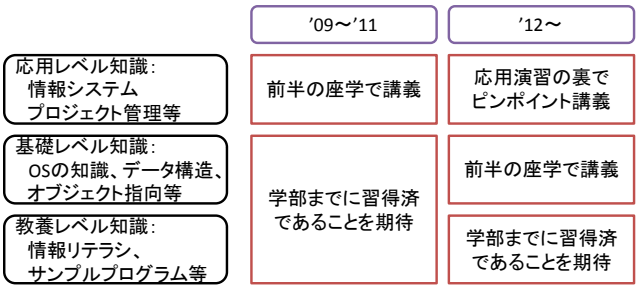

図 4　座学講義の改善
Figure 4 Improvement of classroom lecture

5.4 応用演習の工夫

当初の応用演習では，受講者がプロジェクト計画から行い，プログラムの大部分も受講者が開発していた．2013 年から，プロジェクト計画は講師側から与え，設計，コーディングについても学生がゼロから開発するのではなく，講師側から開発の母体になるドキュメントやソースコードを与え，これを参考に一部の機能を開発するようにした．すなわち，保守性の良いプログラムを受講生が自ら編み出すというよりも，講師の書いた保守性の良いプログラムをまねて、学び、自ら開発できるようにした．

開発の母体となるドキュメント，プログラムについて，受講者に十分に理解できる時間を与え，理解したことを示すドキュメントを作成させてチェックした上で，受講者が開発する部分の設計を開始させている．また，講師側から大まかな WBS およびワークパッケージを受講生側に示すが，細かな作業を誰が分担し，いつ実行するかは，受講者側で決定させている(図 5).

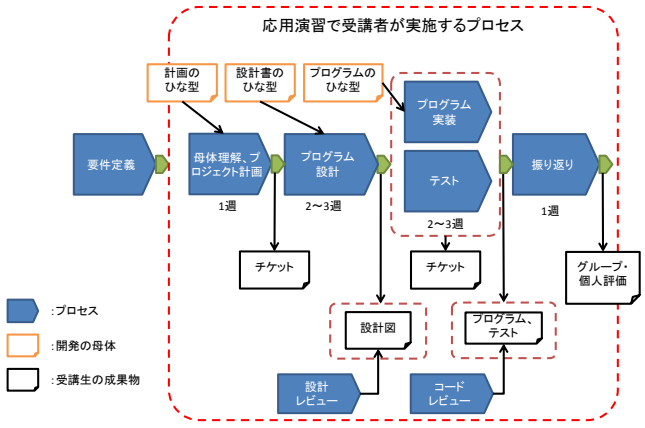

図 5　応用演習の開発プロセス概要
Figure 5 Outline of development process for applied exercise

この部分についても，教科書の続編「ソフトウェア開発実践　科学技術シミュレーションソフトの設計」[3]を執筆し，応用演習で開発するソフトウェアの対象分野である分

子動力学，流体力学や，基盤技術であるハイパフォーマンスコンピューティング(HPC)の基礎から，ソフトウェアとしてどのようにモデリングし，どのように開発するかを受講生が自習できるようにした．さらに，受講者の執筆した設計図，プログラムのソースコードは講師や他の受講者によってピアレビューを行い，プログラムの完成だけでなく，性能，信頼性，保守性の高いソフトウェアを開発できるように改善した．

6. 演習の評価

本章では演習に対する受講生評価，演習の成果物の保守性の定量的評価，演習以外で得られたその他の評価を述べる．

6.1 受講生の評価

演習最終日に受講者に演習全体および，各講義の評価を行っている．図 6 は，「本演習で習得した知識，スキルは今後のプログラム開発で役に立つか」という問いに対する受講者の評価の推移で，図 7 は，「出来上がったものに対する満足度」の評価結果である．

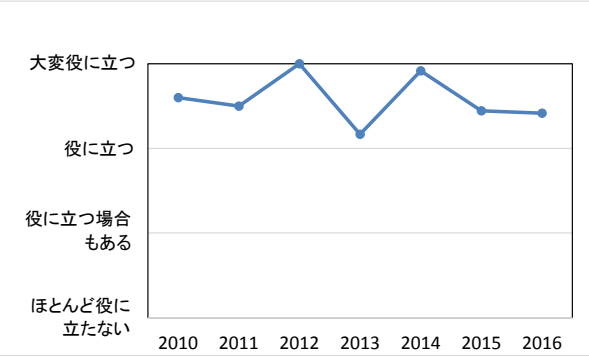

図 6　科目の有効性に関する受講生評価

Figure 6 Students feedback for effectiveness of the practicum

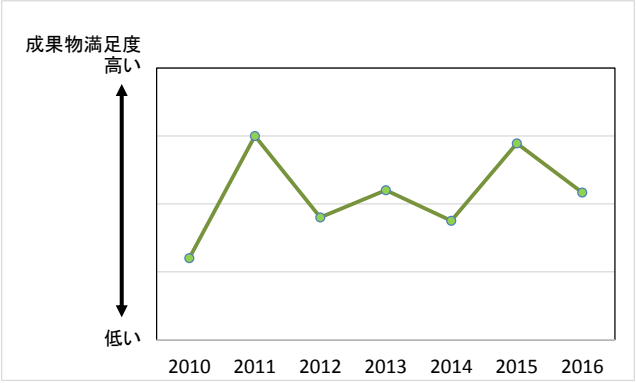

図 7　演習の成果物への受講生評価

Figure 7 Student evaluation for work of practicum

役立ち度のほうは，幸い例年高評価を得ている．成果物の満足度は，一見役立ち度の評価と反対の傾向を示している．実は，受講生の評価と講師側の評価では大きく異なり，受講者は応用演習のプログラムが動いたかどうかを判断基準としているが，本当に良いものができたかどうかという判断はできていない点は留意する必要がある．

6.2 成果物の保守性の評価

6.2.1 保守性の計測，定量的評価方法

成果物の保守性の評価は，McCabe のサイクロマティック複雑度[4]（以下，複雑度）を使用して，計測，評価する．一般に，複雑度が 10 以下であれば保守性の良い簡潔なコードで，処理対象の本質的な性質により 20 以下になることはあっても，20 よりも大きなコードは保守性に大きな問題があるといわれている[5]．なお，複雑度だけ良ければ良いプログラムとは言い切れない．しかし，経験的に複雑度が 20 より大きな関数，メソッドは，他の保守性を表すメトリクスで計測しても悪いプログラムであることが多い．このため，本論文では，「ソフトウェアの複雑度の平均」と，「複雑度が 20 以上の関数/メソッドの比率」の二つをソフトウェアの保守性を計測するメトリクスとする．

表 3　複雑度の評価基準 [5]

Table 3 Evaluation criteria of cyclomatic complexity [5]

複雑度	状態	リスク
1～4	単純な処理	低い
5～10	適切に構造化され安定した処理	低い
11～20	複雑な処理	中程度
21～50	厄介なほどに複雑な処理	高い
50 以上	エラーを起こしやすく，極度に厄介で，テストの難しい処理	とても高い

6.2.2 著名シミュレーションソフトウェアの保守性の定量的評価

実際にシミュレーション分野で良く使われている国内外で開発されたソフトウェアのうち，ソースコードが入手可能なものについて，複雑度を計測した．計測するソフトウェアの選択は，大学側の講師が行った．それぞれのソフトウェアは関数/メソッド数で数千～数万の比較的大きなソフトウェアである．複雑度の計測はソフトウェアの静的解析ツール Understand[6]を利用して算出した．計測結果を図 7 に示す．計測に使用した各ソフトウェアのソースコードは 2016 年 7 月時点の最新版である．

この結果，シミュレーションソフトウェアにより，大きなばらつきはあるが，国内製のシミュレーションソフトウェアで保守困難と言われる複雑度が 20 を超える関数/メソッドの比率が 20%ほどあるソフトウェアがあることが分かった．

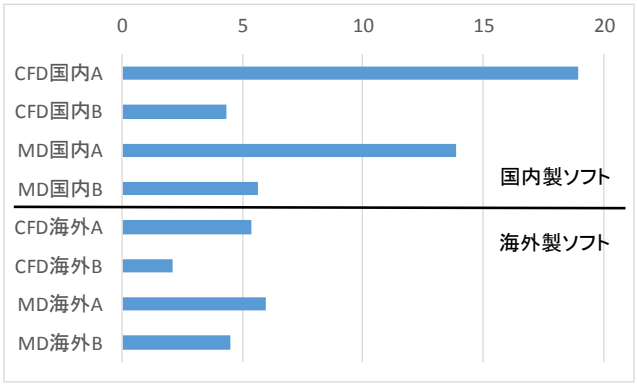

図 7 著名シミュレーションソフトウェアの複雑度
Figure 7 Cyclomatic complexity of well-known simulation software

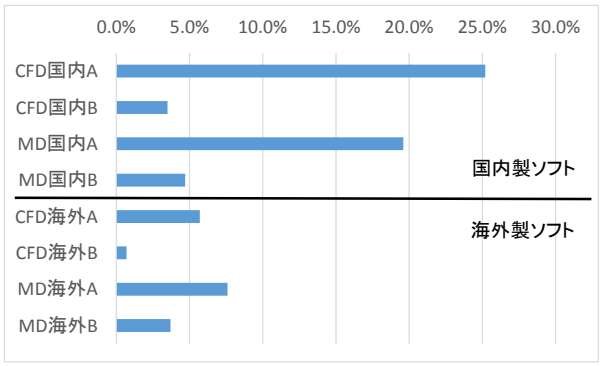

図 8 著名シミュレーションソフトウェアの保守性リスク
Figure 8 Maintainability risk of well-known simulation software

シミュレーションソフトウェアの場合，企業の業務に使われるような汎用的なソフトウェアパッケージに比べて，必然的にループ処理などが多くなり複雑度は高くなる傾向がある．しかし，現状の日本製のシミュレーションパッケージの中には，科学技術計算ソフトの本来の性質による複雑度の差を上回るような，かなり保守困難なものがある．

グローバルに普及し，長期間にわたって保守をしているソフトウェアの実態を踏まえ，複雑度が 20 を超える関数/メソッドの比率が 5%以上のソフトウェアを保守性に高リスクがあり，5%未満を保守性リスクが低いという基準で，応用演習の成果物を評価する．

6.2.3 演習成果の保守性の定量的評価

基礎演習と応用演習の成果物の複雑度の比較を図 9 に示す．複雑度の計測対象は，基礎演習および，応用演習で受講者が，コーディングした部分のみであり，変更母体の講師がコーディングし変更されていない部分は計測対象外とした．また、保守対象となるテストプログラムも計測対象にした．

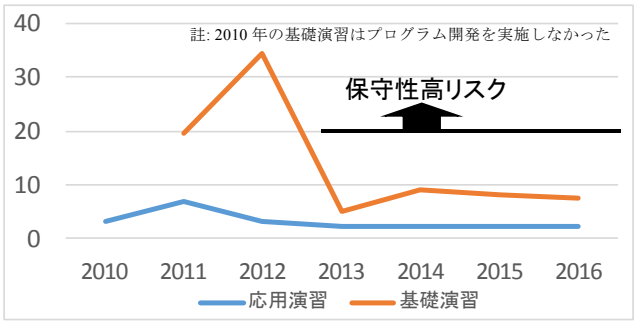

図 9 基礎演習と応用演習の複雑度比較
Figure 9 Comparison of complexity between basic exercise and applied exercise

4.1 節で述べたように，基礎演習のソフトウェア開発では保守性に対する考慮は受講生に指示しておらず，一方，応用演習においては，成果物の評価基準として保守性の良いソフトウェアを開発することを求めている（ただし，どのようなメトリクスで計測するかは示していない）．この結果から，受講生の応用演習での成果物の保守性が上がっていることが分かる．

図 10 に，応用演習での成果物の保守性リスクを示す．前項 6.2.2 で述べた基準である，複雑度が 20 を超える関数/メソッドの比率が 5%を超過したのが，2011 年の一年で，2013 年以降はシミュレーションソフトウェア本体ではほぼ，保守性に問題のある関数/メソッドはなくなっている．（2015 年の保守性が悪かった関数はテストプログラム）

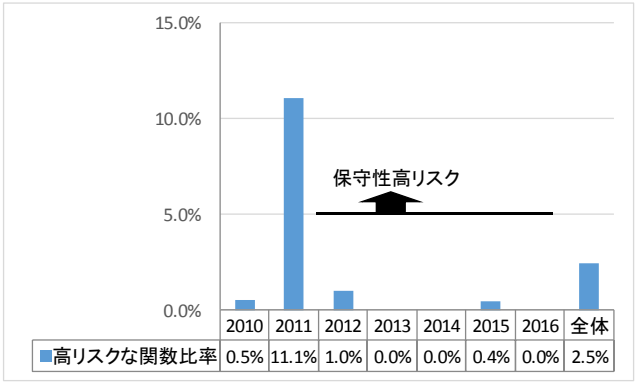

図 10 応用演習成果物の保守評価
Figure 10 Maintainability evaluation of work from applied exercise

6.3 評価まとめ

2 章で述べた本演習の科目レベルの課題が，達成されているかどうかをまとめる．

(1) ドメインスペシフィック

シミュレーションの分野でのソフトウェア開発に対応した現実的なソフトウェア工学的な解を与えるという課題

は，シミュレーション専門の大学教員と，ソフトウェア工学の専門家の企業講師の連携により，実現できたと考える．

(2) ライトウェイト

受講生が，中小規模のソフトウェア開発で保守性の高いシミュレーションソフトウェアを開発できるようにするという課題は，6 章で述べた通り達成した．ただ，受講生が研究室に戻ってから目にするソフトウェアは，昔からある保守性が極めて悪いソフトウェアであり，その規模も演習でのソフトウェアよりも大規模である．このような現状で，本演習の成果が生かせるかどうかは自明ではない．

本演習の直截的な効果ではないが，本論文の大学側著者の東京大学生産技術研究所佐藤研究室で開発保守している，分子軌道法のシミュレーションソフトウェア ProteinDF[7]の保守性改善事例を紹介する．図 11 に，本演習を始める前のバージョンの複雑度と，現在の複雑度を比較した(図 11).

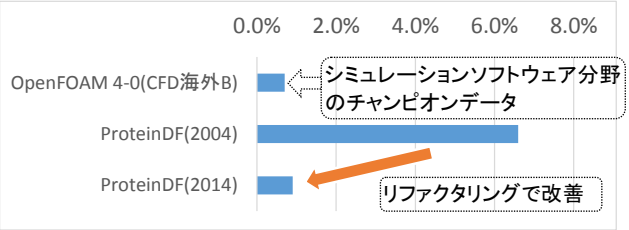

図 11 国内シミュレーションソフトウェアの保守性改善
Figure 10 Maintainability improvement of a domestic simulation software

著者の平野による性能向上・機能追加・リファクタリングが施された最新版における複雑度の結果は，図 8 に示した各種著名シミュレーションソフトウェアの中で最も良い値を示した流体力学シミュレーションソフトウェア（OpenFOAM 4-0[8]．英国で中心に開発）とほぼ同等の値となっている．この結果は，本論文で紹介した演習のクォリティを示すものであるとともに，大規模なシミュレーションソフトウェアであっても，保守性の良いソフトウェアに改善できることを示していると考える．

(3) プラクティカル

基礎演習，応用演習において 4.2 節で述べた開発環境を活用するスキルを身につけさせた．ただ，演習で身に着けたツールを研究室でも活用しているか否かについては，追跡調査ができていない．

7. おわりに

2009 年より継続的に実施しているシミュレーションソフトウェアの開発演習を紹介した．初期段階では，大学側のシミュレーションソフトウェア専門家と，企業のソフトウェア工学専門家が協働して教育設計を行い，これまでシミュレーションソフトウェアの世界では類を見なかった PBL による演習科目の教育設計を行った．さらに，8 年間の教育を通して発生した課題に対応し，受講者のレベルにマッチした教育へと進化させ，演習の当初の目標である，「本格的なシミュレーションソフトウェアを，信頼性や保守性などのソフトウェア工学の知識も考慮して開発できるスキルを身に付けること」を可能とした．

シミュレーションソフトウェアの開発が必要な非情報系の博士前期課程の学生の情報処理スキルが低いという課題に対し，本論文では演習での工夫により対処していることを報告した．この課題は，科目レベルの問題ではなく，学部時点で適切な情報処理のスキルを教えるといったカリキュラムレベルでの検討が必要である．シミュレーションに限らず，他の情報系以外の学科でも情報処理のスキルがこれまで以上に必要になっており，どのようなカリキュラムを組むかも考えていきたい．

本論文で述べた成果を以て，最初に述べたような国家的な競争力を実現することは不可能である．本演習は現在，東京大学の理・工学系の学生のみを対象に実施している．しかし，本演習での内容は，特定の大学の活動に特化した内容ではなく，他の大学やシミュレーションソフトウェアを開発している企業などでも有益である．本演習の内容，成果は，2 冊の教科書[2][3]にもまとめられており，今後は，この教育の適用範囲を広げ、ゆくゆくは国家レベルにも有益な活動になることも目指していきたい．

参考文献

[1] 稲垣忠，鈴木克明「授業設計マニュアル 教師のためのインストラクショナルデザイン」北大路書房(2011) ISBN978-4-7628-2750-1
[2] 佐藤文俊，加藤千幸編．ソフトウェア開発入門：シミュレーションソフト設計理論からプロジェクト管理まで．東京大学出版会 (2014). ISBN-10: 4130624547
[3] 佐藤文俊，加藤千幸編．ソフトウェア開発実践：科学技術シミュレーションソフトの設計．東京大学出版会 (2015). ISBN-10: 4130624555
[4] McCabe. "A Complexity Measure". IEEE Transactions on Software Engineering: pp.308–320. doi:10.1109/tse.1976.233837. ISBN-10: 4822284085
[5] リンダ・M・ライルド，M・キャロル・ブレナン．演習で学ぶソフトウェアメトリクスの基礎．日経 BP 社(2009),
[6] Understand. https://scitools.com/ (2016 年 7 月 24 日参照)
[7] ProteinDF. https://proteindf.github.io/ (2016 年 7 月 24 日参照)
[8] OpenFOAM. http://www.openfoam.com/ (2016 年 7 月 24 日参照)