ソフトウェア工学演習コース 活動報告

Report on Software Engineering Practice Course

主査 野中 誠（東洋大学経営学部）

今年度より「ソフトウェア工学演習コース」が新たに開講された。本稿では、その主な狙い、例会での演習概要、有志により実施された例会以外の活動、例会の議論に端を発したレビューに関する問題認識について報告する。

The software engineering practice course was newly opened in this year. This article reports on the primary aims of this course, summaries of each practice in regular meeting, other activities held by members interested, and a problem recognition about software review that was derived from regular meetings.

1. ソフトウェア工学演習コースの主な狙い

近年のソフトウェア産業において頻繁に指摘される問題点として、「ソフトウェア工学を理解していないソフトウェア技術者が多い」「ソフトウェア開発がエンジニアリングとは程遠い状態にある」などが挙げられる。ソフトウェア工学という言葉が 1968 年に提唱されてから 35 年以上が経過したが、技術の蓄積が現実のソフトウェア開発に適用されていない、または役立っていないとするならば、これは産業全体にわたる大きな問題である。ソフトウェア工学に関する基本的な技術を正しく理解し、その適用を試みるとともに、最新の研究動向にも目を配りながら、開発・管理技術をより高度なものへと進化させることが求められる。

一方で、「大学でのソフトウェア工学研究・教育が実践的でなく、役立たない」という批判もよく耳にする。もちろん、技術の純粋な発展のためには、現実の組織に関わる制約から離れたところでの基礎研究も必要である。しかし、研究者自身も、研究内容や取り組みの有効性を実務家に伝え、産業の発展に貢献するとともに、新たな問題点を研究へとフィードバックさせる努力も求められよう。

このような問題意識から、本コースでは、主にアカデミックな観点でソフトウェア工学研究・教育に携わっている講師を招き、実践的かつ先進的なソフトウェア工学手法に関する講義と演習を企画した。また、受講者の知識定着を高めるためにも、演習というスタイルを重視した。本コースの演習を通じて、受講者が、次の点を十分に理解できることを本コースの狙いとした。

(1) ソフトウェア工学の知識・手法を確実に習得する。
(2) 実践的かつ先進的なソフトウェア工学手法を学習する。
(3) 手法の有効性や適用場面などについて議論し、理解を深める。

このような主旨に賛同いただけたためか、総勢 20 余名の受講者を得ることができた。コース立ち上げ初年度ということもあり、演習内容やコース運営など、いくつかの問題点もあったが、熱心な受講生に支えられて無事にコースを終えることができた。

以降、例会での演習概要、有志により実施された例会以外の活動、例会の議論に端を発したレビューに関する問題認識について報告する。なお、以下の報告は、受講者の分担執筆による貢献が大きい。

2. 例会での演習

2.1 演習の概要

本コースでは、ソフトウェア開発技術およびマネジメント技術に関する以下の演習について，それぞれ個別に以下の講師(敬称略)を招いて実施した。各演習内容について、その概要を紹介する。

第 1 回：PSP (Personal Software Process) ：**野中 誠 (東洋大学経営学部)**

ソフトウェア技術者にプロセス測定の必要性や測定方法を教育する演習コースである PSP (Personal

Software Process) [1]に関して、その概要、主要なトピックス、講師の PSP 教育経験などについて議論した。演習では、個人のソフトウェア開発におけるプロセスデータを測定し、そのデータからプロセスの状態を把握・評価する方法を扱った。また、PSP で導入されているプロセス・メトリクスを紹介し、その有用性や活用方法について議論した。さらに、講師が新人ソフトウェア技術者向けに 2 日間で PSP のエッセンスを体験させている取り組みについて紹介し、その成果について議論した。

第 2 回：ゴール指向要求分析：海谷 治彦 (信州大学工学部)

近年の要求工学のトピックスであるゴール指向要求分析をテーマに、顧客ニーズをゴールという観点で捉えて要求定義を行う技法を扱った。まず、IEEE 830[2]に示された要求仕様書(SRS: Software Requirements Specification)の品質特性が紹介された。この品質特性に基づき、音楽プレーヤーの SRS を対象に「SRS として望まれる特性は何か」という観点から評価し、SRS を修正・完成させる演習を行った。

次に、ゴール指向手法の一つである KAOS[3]を用いて、音楽再生ソフトの要求分析を行った。ここでは、初期要求からのゴール識別を行い、ゴール識別における一般的なパターンを理解した。また、要求と期待の識別や Agent(達成責任)の識別を行った。さらにゴール間の Conflict(衝突)の抽出を行い、分析方法を理解した。最後に、演習で作成したゴールグラフを発表し、討論を行った。

第 3 回(1)：機能規模測定法 (IFPUG 法)：西山 茂 (NTT アドバンステクノロジ(株))

機能規模測定法の 1 つである IFPUG(International Function Point Users Group)法[4]の演習を行った。演習では、GUI(Graphical User Interface)ベースのシステムを対象に、IFPUG 法によって機能規模 FP(Function Point)を測定した。FP 測定にあたって必要な「複雑さの評価」と「システム特性係数の算出」について受講者の理解が深まり、FP 法が分かりやすく、有用な手法であると理解できた。

また、機能規模ベースの見積り手法に関する議論も行われた。IFPUG 法をうまく適用できれば、経験則(定性的)で規模(LOC: Lines of Code)を見積もるよりも、見積り精度の向上が期待できる。ただし、入力と出力が変わらない性能チューニングといった開発の場合には IFPUG 法の適用は難しいことが示された。

第 3 回(2)：PBR (Perspective-Based Reading) レビュー手法：野中 誠 (東洋大学経営学部)

実証的(empirical)ソフトウェア工学の分野でよく取り上げられているレビュー手法の有効性評価に関して、受講者に実際にレビューを実施してもらった。演習では、6 ページ程度の要求仕様書を対象に、ユーザ、設計者、およびテスト担当者それぞれの観点から、個別に用意したチェックリストを用いてレビューを行った。このように、レビューを特定の観点から実施する方法は PBR(Perspective-Based Reading)と呼ばれており、限定的な範囲ではあるが有効性が確認された手法である。

第 4 回：コーディング作法とコードレビュー：平山 雅之、大野 克己 (IPA/SEC)

午前中の特別講義に引き続き、午後の演習では、SEC(Software Engineering Center)版コーディング作法[5]を利用したソースコードのレビューを行った。架空のプロジェクトを想定し、限られた時間の中で重点レビュー対象のソースコードモジュールを絞り込むために、どのようなメトリクスを用いればよいかをグループで検討し、選択したメトリクスを用いてモジュールを測定した。その後、絞り込んだモジュールに求められる品質特性を考慮した上で、SEC 版コーディング作法の中からチェック項目を選択し、バグを検出するという演習を行った。演習を通じて、システムに求められる品質特性にあわせたコーディング作法を利用することが、ソースコードの品質向上につながることを理解した。

第 5 回：設計パターン：鷲崎 弘宜 (国立情報学研究所／総合研究大学院大学)

午前中の特別講義の内容を受けて、設計パターンの適用に関する演習を実施した。演習では、UML (Unified Modeling Language)で設計したクラス図に対して、様々な観点から洗練をかけていく作業を行った。その過程で、Gamma らが提唱した 23 個のいわゆる GoF(Gang of Four)設計パターン[6]のいくつかに

ついて、どのパターンをどのように適用すれば良いか、どのようなメリットが得られるのか等を学んだ。演習を通じて、設計パターンの良い点、悪い点、使い勝手、経験談などについて議論した。

第 6 回：プログラム検証：中島 震 (国立情報学研究所／総合研究大学院大学／JST)

ESC/Java[7]の講義と演習を行った。午前中に学んだ SPIN(モデル検査ツール)とは違い、Java のプログラムを直接的に検証するツールであり、普段 Java で仕事をしている受講者にとっては直接的に有意義な演習であった。また、Java に馴染みの薄い受講者にとっては言語の面などから多少厳しい演習ではあったが、その背景にある DbyC(Design by Contract: 契約による設計)の有効性を体感できた。

形式的手法は普段あまり触れることがないため、受講者にとって貴重な機会となった。演習で扱った内容は、人間の代わりにコンピュータがモデル(設計)やプログラムのバグを見つけてくれる技術である。開発コストをアップさせないで品質をアップできる可能性に魅力があり、今後の勉強の動機付けになった。

第 7 回：品質要求定義：東 基衛 (早稲田大学理工学術院)

午前中の講義では、品質要求、品質測定技術、品質モデル、および品質評価が体系化されることが重要という解説がなされた。これを受けて、午後の演習では、インターネットによる通信販売を課題として、4 つのグループに分かれて実際に品質要求分析と定義を行った。その方法は、システム機能要求の定義、外部品質要求の選別と優先順位付け、品質要求基準の設定という流れを、10 段階からなるステップに分けて行うというものであった。要求品質の仕様化と品質要求基準の設定方法を、より具体的に理解することができた。ISO/IEC 9126-1[8]の品質特性は、その概念は広く理解されているものの、実際に利用した経験を持つ人は少ない。今回の演習は、品質特性を利用する良い機会になったといえる。

2.2 SWEBOK との対応

以上に紹介した各演習を SWEBOK(Software Engineering Body of Knowledge)[9]の副知識領域と対応づけると、図 1 のように示すことができる。演習コース全体としては、SWEBOK の知識領域のレベルである程度包括的に実施できたといえる。ただし、個別の演習内容はそれぞれの知識領域において深掘りされた内容が多かったため、副知識領域のレベルでは、特定の領域について重点的に扱うこととなった。

演習で扱わなかった副知識領域については、受講者自らが書籍などを通じて学ぶなどの努力をすることによって、より包括的な理解が深まることであろう。

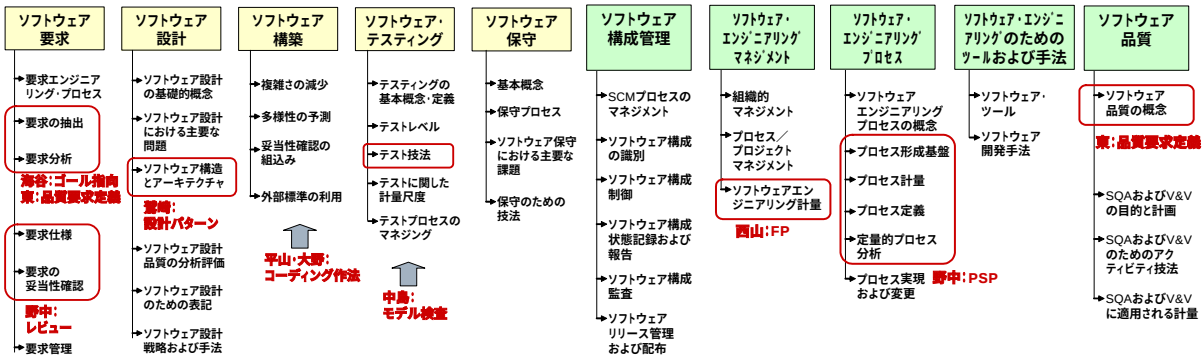

図 1 演習内容と SWEBOK との対応

2.3 演習内容を実務で役立てるには

本演習コースでは、必ずしも実務に直結した技法を紹介することを意図しておらず、通常の開発業務とは異なる視点や考え方を学んでもらいたいという意図がある。しかしながら、その中からいかに実務への適用を図り、ソフトウェア開発をエンジニアリングとして一段高いステージに押し上げるかは、やはり重視すべき課題である。ここでは、各演習を次の 4 つで分類し、実務に適用するという観点から考察する。

(1) 先端的で実務とのギャップがある技術

ゴール指向要求分析やプログラム検証が、この分類に相当する。最先端の研究であるため興味は尽きないが、残念ながら、実務で広く利用できる段階に至っているとは言い難い。ゴール指向要求分析は、その考え方の重要性は共有できたが、実務において生じる数多くのゴールを、すべて整合性を保った状態で関連づけるのは、作業工数の面からも非現実的である。また、プログラム検証は、ツールの実現により身近になったものの、DbyC を一般のソフトウェア技術者が使いこなし、通常のデバッグ作業よりも高い効率を発揮できるには、まだハードルが高い。これらの技術は、まずは頭の中の棚に整理しておくことが重要で、今後さらに研究が進んだ段階でも、棚から引き出すことで類推できることが重要である。今回の演習では、そのための基礎知識を得るのに大いに役だった。

(2) 受講者の実務内容とのギャップがある技術

設計パターンがこれに相当する。本年度の受講者の多くは、C 言語などの非オブジェクト指向言語を実務で用いている。オブジェクト指向開発を実践している受講者(ただし少数)にとっては、設計パターンは示唆に富んでいた実務に対する工夫が得られた。しかし、オブジェクト指向開発とは縁遠い受講者にとって、設計パターンの内容を実務に直接結びつけることは困難であろう。ただし、対象世界を抽象化して捉えることの意義、拡張性を考慮した設計の重要性、技術者間でコミュニケーションする上での共通言語としてパターンが役立つことは十分に理解できた。

(3) 実務で実践していて、工夫の着眼点が得られたもの

品質要求定義、要求仕様書レビュー、コーディング作法は、いずれの受講者も実務で実践している取り組みであり、その実践方法を工夫するためのよい着眼点が得られた演習であった。これらは、実際の活動に対するヒントが多々あり、実務に応用しやすい。そのため、演習で紹介された方法をさらに発展させるためにどのような工夫をすればよいかなど、より発展した議論が行えた。

(4) 理解しやすい（または既知の）技術だが、適用できていない技術・手法

IFPUG 法、PSP は、多くの受講者がその概要をすでに理解していたり、手法の主旨を容易に理解できる演習であった。しかし、どちらも組織に定着させて実践するには手間のかかる技術・手法であり、導入による効果が現れるまで信念を持って行い続けるのが難しい。特に、組織での運用にあたっては、各担当者が測定をいやがる事が多く、その壁を乗り越えることが大変であろう。

特に、今回の受講者の約半数が組込みソフトウェア関係であったため、データベースの存在を前提とした IFPUG 法では実務への応用が利きにくい。そのため、講演の中で紹介されていた COSMIC-FFP 法の方が、実務への適用が行いやすいと思われる。

3. 例会以外での有志による活動

1 章でも述べた通り、本コースは熱心な受講者によって支えられた。時間や距離的な制約などから、すべての受講者が参加できたわけではないが、有志による活動も活発であった。以下に、それら活動の概要を紹介する。

3.1 Pressman 著『実践ソフトウェアエンジニアリング』の輪読

有志により、Pressman 著『実践ソフトウェアエンジニアリング』[10]の輪読を実施した。また、各章末にある演習問題を分担して実施し、その結果をメーリングリストに投稿し、議論を行った。当初は本コースのテーマとも関連が深い 6〜15 章、22 章、26 章を読破する目標でスタートしたが、メーリングリストのみでの活動には限界が有り、6〜9 章で挫折してしまった。

挫折の原因としては、主に以下が挙げられる。

・参加者の時間の確保が困難であった

・演習問題の回答に対して、メーリングリスト上では十分な議論が行えなかった
・Pressman の書籍が辞書的であり、紹介されている技法の詳細まで理解しきれなかった
・更には演習問題が抽象的で、明快に解答できないものも多かった

ということで、上記の反省を踏まえ、やはり face to face での議論が必要と考えた。もっと早く実施すべきではあったが、2 月の最終例会前に臨時会として勉強会を実施することにした(執筆時点は臨時会の開催前)。単なる演習課題の答え合わせではなく、議論を通して理解を促し、更に深く学ぶきっかけとなるように締めくくりたい。

3.2 相互プレゼンテーション

その他、臨時会として、「ソフトウェア工学及びその関連技術で、影響を受けた、本、論文、セミナーについて」というテーマを設定し、各自が発表する機会を設けた。各者各様の着眼点で、他の受講者に伝えたいという意欲を持ってのプレゼンテーションとなった。以下に、紹介されたトピックスとその概要を、影響を受けた受講者の談として、その伝導者とも呼ぶべき講師名(敬称略)とともに紹介する。

(1) ソフトウェアテスト:西 康晴 (電気通信大学)

テストについての鉄則を伝授された。常にテストで悩んでいるテーマ(終了条件、テスト技術／プロセス)や、例え話(テストの達人)などに影響を受けた。テストを起点にソフトウェア開発全体が改善されていく事を知り、テストを前向きに捕らえる事ができるようになった。

(2) コードコンプリート:Steve McConnell

ものすごいボリューム(950 ページ)の本だが、読みやすく内容が具体的で非常に分かりやすい。入社 7 年目にしてソフトウェア開発部門へ異動になったが、この本のおかげで乗り切れた。ソフトウェア業界は参考書籍が非常に多い。特にこの本はコーディングテクニックについて見事に知識が体系化されている。これは他の工業製品やサービス分野を見渡しても稀な事で、恵まれていると共に技術革新が早く常に勉強が必要だが、努力すればどんどん向上できる事を物語っている。

(3) SE に求められる技術:清水 吉男 ((株)システムクリエイツ)

清水氏の講演や書籍より、設計製造技術(設計とは複雑さの解消)、仕事を進め方(作業を分割し少しの空き時間でこなす)、形式知化する事の重要性(OJT を盾に暗黙知のままにしていないか)、知識労働者の自覚等について、強い影響を受けた。氏の教えを実践しようと日々努力している。

(4) 実践ソフトウェアエンジニアリング教育:松尾谷 徹 ((有)デバッグ工学研究所)

単体テストからシステムテストまでの観念とテスト手法(同値分割、CFD 等)について 6 日間演習教育を受講した。今まで整理できずに悩んでいた、テストの概念と解決すべき問題の仕組みを理解でき、ソフトウェア工学に惹かれるきっかけとなった。

(5) リスク管理について:日経 BP ソフトウェア品質向上セミナーでの事例

プロジェクトには必ずリスクが存在する。このセミナーを通じて、システム提案時から監視を開始し、定量化する手法を知った。これまでプロジェクトを行ってくる中で不安に感じていたことは、リスクだったという気づきがあった。そこから、リスクチェックシートを作成し、部門に展開した。現在、部門でのリスク管理を始めようとしている。

4. レビューの有効性を高めるための方策に関する議論

4.1 問題認識

2.3 節(3)で述べた通り、品質要求定義、要求仕様書レビュー、コーディング作法などは、実務において

工夫の着眼点が得られる演習となった。しかし、実際の開発現場では、品質要求定義やレビューなどの活動を確実に定着させることは容易ではない。上流段階で品質を確保することの重要性は広く理解されているものの、品質確保のための工数やコストが割に合わないと認識される場合が少なくない。その結果、組織のノウハウが盛り込まれたチェックリストが、実務において十分に活用されていない、あるいは無視されている場面が多々ある。まさに、「総論賛成、各論反対」といった行動をとる組織またはソフトウェア技術者が少なくない。

一方、品質確保のためにテスト工程を重点的に実施することの重要性を認識し、そのように行動している組織またはソフトウェア技術者は多い。テスト工程の充実化は、品質保証の目的からすれば当然の取り組みである。しかし、上流工程で品質確保のための活動を行っていれば回避できたはずの問題のために、非効率的なテスト作業を余儀なく場面も多数見受けられる。ソフトウェア工学分野における一つの常識として、「上流段階で欠陥を除去する方が、下流工程で欠陥を除去するよりも効率的である」という認識があり、これを支持する実証データも多数報告されている(例えば[1][11])。しかし実際には、上流工程での見逃した欠陥を下流工程で拾っているという組織が多いのが現実である。実証データを示しただけでは、組織や個人の行動を変えさせるだけの動因になり得ていない。

上流工程でのレビューが有効に実施されていないという問題の所在は、ソフトウェア工学の範疇を超えて、ソフトウェアビジネスの産業構造やワークモチベーションなど多岐にわたる。しかしここでは、いたずらな議論の発散を避け、レビューの目的を明確化することと、それに対応した適切な手段を適用することの必要性を主張する。このような整合性を確保する努力をした上で、モチベーションなどの組織論の問題に取り組むべきであろう。

なお、以降の議論はあくまで議論ベースによるものであり、厳密な検証は行っていない。その点をあらかじめご理解いただきたい。

4.2 問題点の整理

先の問題認識について、より具体的な内容に触れながら、問題点とその対処の基本方針を整理する。

(1) チェック項目に関する知識の欠落

第一に、チェック項目に込められた人間の知識が、十分に共有されていないという問題が挙げられる。品質に関する問題意識の高い組織は、組織固有のノウハウが盛り込まれたチェックリストを保有している。しかし、改善を積み重ねるにつれて、チェック項目数は増加の一途を辿り、技術者が短期記憶で扱える許容範囲を超えてしまう。また、チェック項目は要約した抽象的な記述となっている場合が多く、着目ポイントが何であるのか、どのような問題の防止に役立つのかなど、チェック項目の背後に潜む情報が十分に伝達されない。

(2) ソフトウェア開発活動における目的の多様さ・不明確さ

第二に、レビューなどの活動の目的が明確に設定されていない、または目的が共有されていないという問題が挙げられる。ソフトウェア製品の品質要求は、ISO/IEC 9126-1 の品質モデルに示されている通り、多様な側面で表される。ソフトウェア製品毎に品質要求は異なるため、本来的には、プロジェクト毎に品質要求が明確化されるべきである。また、教育目的のためにレビューを実施したり、再利用資産の形成のためにレビューを実施するなど、ソフトウェア製品自身とは異なる目的を持って組織は行動する。

(3) 目的と手段の対応付けの不明確さ

そして第三に、目的と手段のリンク付けが不十分という問題が挙げられる。すなわち、これまでに指摘した二つの問題を結びつける努力をしなければならない。これらがすべて揃って初めて、適切な手法が合理的に盛り込まれた開発標準を規定でき、技術者自身も、その開発標準の意義を理解した上で実践できる。当たり前ともいえる議論ではあるが、それが十分に実践できていないところに問題がある。

4.3 問題の解決に向けて

(1) チェック項目に関する知識の補足

第一の問題を解決するために、それぞれのチェック項目が持つ役割や効果などを明確化した上で、膨大なチェック項目を分類整理あるいは統合する必要があるそれぞれのチェック項目が、どの品質特性の確保または向上に貢献するのか、明確化する努力が求められる。例えば、表 1に示したように、組織が保有するチェック項目がどのような点で貢献するのかを整理することにより、なぜこのチェック項目を用いなければならないのかなど、よりリッチな情報を得ることができる。

表 1 チェック項目の目的の明確化例

チェック項目	レビュー戦略策定時の観点					
	教育	保守	正確性	性能	異状処理	コスト
適切なモジュールを再利用しているか	○	○				○
OSの違い(Windows2000,2003,XP)を意識しているか	○	○				
再利用性を考慮した設計になっているか	○	○				
使用するリソースが全て洗い出されているか	○	○	○	○		
利用可能な資源の範囲内で動作可能な設計になっているか	○	○		○		
求められている性能が発揮される設計になっているか	○			○		
セキュリティ面での脆弱性に問題がないか	○		○			
プログラムの起動順を考えているか(同時起動に付いて考えているか)	○		○			

SEC 版コーディング作法では、ISO/IEC 9126-1 のうち 4 つの品質特性と関連づけて、作法を分類整理している。このように、膨大なチェック項目を分類する際には、その目的別に分類整理し、チェック項目の目的を明確化することが必要である。

(2) ソフトウェア開発活動における目標の明確化

第二の問題として述べたように、プロジェクトには、ソフトウェア製品の品質特性の多様性に加えて、組織意図の多様性など、様々な目的や意図が入り交じっている。また、目的や意図は、プロジェクトの局面によって動的に変化する。このような目的や意図を把握した上で明確な目標を確立しなければ、その時点で実施すべき活動が何であるのかを判断することは難しい。技術者に対して、安易にモチベーションコントロールを試みるのではなく、合理的な判断が下せるように目標を明確化する必要がある。具体的には、ISO/IEC 9126-1 に示された品質モデルなどを用いて品質要求定義を行うことで、そのソフトウェア製品として達成すべき品質要求が明確化でき、ひいては目標の明確化に結びつくであろう。

また、設定される目標は、バグのなさだけでなく、多様な品質特性について議論すべきであろう。かつての ZD (Zero Defect) 活動と同様、多くのソフトウェア組織では、ソフトウェア製品の潜在バグをできるだけ減らすために多大な努力を費やしている。現状、諸外国のソフトウェア製品が、バグ密度という意味で品質が決して高くないこともあり、バグの少なさは日本のソフトウェア組織の競争力となる一因であろう。しかし、バグの少なさという特性は、成熟した製品分野であれば、一般に「当たり前品質」に該当する特性である。すなわち、その特性をある一定以上高めたところで、顧客の満足度に結びつかない。もちろん、この特性が要求された水準以下となると、顧客の不満足要因に結びつくため、不必要な特性というわけではない。しかし、製品の総合的な魅力を高め、顧客の満足度を高め、組織の競争力を高めていくためには、バグの少なさ以外の品質特性にも目を向けるべきであろう。

(3) 合理的な手法の選択･適用･実践

第三の問題に関しては、チェック項目やレビュー手法などが有効性を発揮できる範囲を、正しく認識しておくことが必要である。例えば、移植性や保守性に関する欠陥(または不適切な箇所)をテストで摘出することは極めて困難だが、信頼性や効率性(性能)に関する欠陥をレビューで摘出することは困難である。このように手法の有効範囲を評価する活動が、実証的(empirical)ソフトウェア工学研究として世界中で取り組まれている。こうした研究成果に目を向けて、適切な手法を適切な場面で適用できるだけの力量が求められる。

また、設定された目標に対して、実践すべき手法を体系的に取り入れる方法も必要である。例えば、品

質機能展開 (QFD: Quality Function Deployment) の応用によって、ソフトウェア製品の品質要求を達成するために必要な活動を重点的に実施するような開発プロセスをデザインするなど[12]、目的と手段を対応付けた上でプロジェクトを進めることが求められる。または、同一組織が開発するソフトウェア製品はおおむね類似の品質要求となることが多いことから、組織で 1 つまたは複数の開発標準を、同様の考え方によって規定し、これを各プロジェクトに展開するなどの努力が求められる。

さらに、この開発標準に組織固有のノウハウを盛り込む努力が求められる。品質要求を満たすために、どのようなチェック項目を重点的に実施すべきかを熟慮し、組織固有のノウハウを盛り込んだ、品質要求の確保に貢献するチェック項目を、組織の資産として作り込んでいくべきであろう。

ただし、この考えに従うと、組織が新たなドメインに入り込んでいくと、新たなチェックリストや開発標準の作成が求められる。組織標準が増えていくと、それぞれの標準が独立の道を歩んでいき、結果として、組織内で統一感のない、バラバラな標準が乱立する。その結果、部門を超えたコミュニケーションが出来なくなるなどの弊害も生じかねない。この点については注意が必要である。

5. おわりに

再三述べた通り、熱心な受講者に支えられて演習コースを終えた。受講者各位は、各組織にて、その熱意を持って「ソフトウェア開発をエンジニアリングと呼べる状態に」するべく努力されることを願う。また、次年度も本コースに参加して議論の深耕をしたり、演習コースに対して改善提案したり、他の分科会にて新たに取り組むなど、日科技連へのフィードバックにご貢献いただければ幸いである。

次年度も、演習メニューを改善した上で本コースを実施する。本稿が、この演習コースに対する興味に結びつき、次年度以降の演習コースへの参加につながれば幸いである。その延長線上として、日本のソフトウェア産業の発展に少しでも貢献できれば、著者として望外の喜びである。

謝辞 本稿の、特に 2 章と 3 章に執筆にあたって、次の受講者の方々に草案を分担執筆いただいた。ここに厚く御礼申し上げます(敬称略、五十音順)。また、毎回の演習をご指導いただいた講師の皆様にも、この場を借りて厚く御礼申し上げます。

伊藤 拓也(日本電気株式会社)、猪塚 修(横河システムエンジニアリング株式会社)、岡 英仁(日本電気通信システム株式会社)、小池 利和(ヤマハ株式会社)、高橋 千弘(アンリツエンジニアリング株式会社)、田中 宏幸(日本電子株式会社)、早坂 哲(アルプス電気株式会社)、麓 博之(ジャパンシステム株式会社)、諸 葉子(TIS 株式会社)、渡辺 康之(日本電子株式会社)

参考文献

[1] Humphrey, W.S., *A Discipline for Software Engineering*, Addison-Wesley (1995).
[2] IEEE Std 830-1998, “IEEE Recommended Practice for Software Requirements Specifications,” IEEE Computer Society (1998).
[3] http://www.objectiver.com/download/documents/KaosTutorial.pdf (2003).
[4] ISO/IEC 20926:2003, “Software engineering -- IFPUG 4.1 Unadjusted functional size measurement method -- Counting practice manual” (2003).
[5] コーディング作法ガイド(0.8 版), IPA/SEC, http://sec.ipa.go.jp/download/200504eb.php (2005).
[6] Gamma, E., Helm, R., Johnson, R. and Vlissides, J., *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley (1995).
[7] http://secure.ucd.ie/products/opensource/ESCJava2/
[8] ISO/IEC 9126:2001, “Software engineering -- Product quality -- Part 1: Quality model” (2001).
[9] http://www.swebok.org/
[10] Pressman, R.S.著, 西 康晴, 榊原 彰, 内藤 裕史訳, 実践ソフトウェアエンジニアリング -ソフトウェアプロフェッショナルのための基本知識-, 日科技連出版社 (2005).
[11] Boehm, B. W., *Software Engineering Economics*, Prentice-Hall (1981).
[12] 野中誠, 組込みソフト産業の実態と開発の課題:組込みソフトウェア特性に基づくプロジェクト構築, 情報処理, Vol.46, No.6, pp.684-690 (2005).