③ 共通問題ショートエッセイ

基般

鵜林尚靖（九州大学大学院システム情報科学研究院）
野田夏子（芝浦工業大学デザイン工学部）
滝沢陽三（茨城工業高等専門学校電子情報工学科）
松本　明（和歌山大学大学院システム工学研究科）

本稿は，本会ソフトウェア工学研究会主催ウィンターワークショップ 2014（茨城県大洗町，2014 年 1 月 23 日，24 日）の「ソフトウェア工学の研究評価と共通問題」セッション参加者によるショートエッセイ集である．参加者の中には，教育・研究者だけでなく，現役の学生も含まれている．ソフトウェア工学研究における評価の難しさ，ソフトウェア考古学のすすめ，教育視点から見た共通問題のあり方，など参加者のそれぞれの想いをオムニバス形式で執筆した．

ソフトウェア工学と言語ゲーム（鵜林）

ソフトウェア工学の共通問題に関する特集を組むきっかけとなった理由の 1 つに「ソフトウェア工学研究の評価は難しい」「評価に使える共通の問題はないのか」という研究者の切実な想いがある．どうしてソフトウェア工学の研究は難しいのか，私なりに考えてみたことを紹介したい．

プログラミング言語理論やアルゴリズムなどの理論研究とソフトウェア工学などの実学の相違点は，実証が求められるか否かにある．特に「有用性」が本質的な評価尺度となる．有用性は人間に依存する尺度であり，これを真に行うためには，多数の開発者を巻き込んだ大規模な実証実験を行うしかない．しかし，コストや労力の面で非現実的な場合が少なくなく，ソフトウェア工学に関する評価は残念ながら妥当性を欠いているものが多い．何をどこまでやれば実証したことになるのであろうか？　非常に難しい課題である．有用性の評価という立場から見た場合，多くの研究は砂上の楼閣のような存在なのかもしれない．オブジェクト指向を例に考えてみよう．オブジェクト指向技術は，現在，多くの人が有用だと認める．しかし，本当にそれが有用かどうかが実証できているかと言うと実はできていない．一方，オブジェクト指向を対象とした理論研究は多数ある．それらの論文は「現在のオブジェクト指向言語には○○のような問題があり，それを解決するための理論を提案する」といったストーリーで書かれる．オブジェクト指向の有用性を脇に置くか，あるいは有用だという前提の上で研究が行われる．しかし，オブジェクト指向が有用でなければ，本当は，このような研究に意味はないであろう．砂上の楼閣になってしまう．オブジェクト指向は一例に過ぎない．これに類することは数多く存在する．ソフトウェア工学以外の分野では砂上に楼閣を建ててもクレームがつかないのに，なぜ，ソフトウェア工学ではそれが許されないのであろうか？

ソフトウェア工学とはソフトウェアの作り方に関する学問領域である．作り方には，手法名とそれが意味するもの，すなわち「有用性が保証された開発手順」がなければならない．意味とは，本来どのようなものであろうか？　Ludwig Wittgenstein の言語ゲーム（「哲学探究」）[1] では，語の意味とは「言語の中でのその使用（use）」だと言う．我々はオブジェクト指向のことをよく知っていると思っているかもしれないが，各自がどのような使用方法（以下，用法）をしているかをお互いに見比べながら（一種のゲームを興じながら），何となく理解しているような気持ちになっている可能性が高い．オブジェクト指向の「使用」が有用性をも含めた意味を規定している．

ソフトウェア工学のプラクティス（分析，設計，テスト，デバッグなどの手法）はソフトウェア開発のための言語であり，開発技術を学ぶことは用法の習得であると解釈できる．用法が用法足り得るのは，それが無意識あるいは暗黙の了解のもとで使われる場合である．先の例も，オブジェクト指向自体が「使用」により意味づけられており，その上の研究はオブジェクト指向の有用性をあえて議論しなくても問題視されないのである．楼閣と砂の間には「用法」という土台が存在する．理論研究の多くはプログラミングパラダイムが定着した後に行われるので，用法としてすでに開発者間で意識の共有ができている．一方，ソフトウェア工学の研究は「新たな用法」を提供することであり，新規性やオリジナリティは用法の「新しさ」に依存する．ここに大きなジレンマが存在する．研究として提案したものは，まだ開発者間で意識が共有されてなく，実は用法ではないのである．用法を提案しつつ，それは用法の条件を満たさないという自己矛盾をソフトウェア工学研究は抱えている．用法になるには長い年月を経て人々の間で使用される必要がある．ソフトウェア工学の研究コミュニティはこの問題を克服していかなければならず，用法の新しさではなく，用法を育てていくことに研究の価値を見いだしていく時期に差し掛かっているのかもしれない．

ソフトウェア開発史あるいは ソフトウェア考古学のすすめ（野田）

学生と話をしていて驚いたことがある．彼らの多くは 2000 年問題を知らないという．確かに学部生であれば，小学校に上がったか上がらないかという年齢だったであろうから，驚くには当たらないのかもしれない．しかしあの当時あれだけ大騒ぎした出来事が，次世代を担う若者に知られていないということには，単に驚きだけではなく，少々の不安も感じる．あのときに直面した問題，それをどう解決したか（あるいはできなかったか）という経験が，次代に継承されていないことになるからだ．

2000 年問題というときの「問題」と，共通問題の「問題」は，本来の意味合いは少し違う．前者は Problem，困ったことという意味での問題だし，後者は Question，解答の対象としての問いという意味での問題だ．しかし，Problem は Problem として放置しておくことはできず，いかにしてそれを解決しようかという，皆にとって共通の Question でもあったはずだ．当時，2000 年問題は IT 業界全体がかかわる相当大きな問題であった．年が 2 桁で表現されていてそのままでは問題を引き起こすプログラムをどう修正するかということであるが，単にプログラミング上の問題というだけではなく，開発成果物全体において問題にかかわる個所や影響が及ぶ範囲をどのように特定するか，限られた時間の中でどのように優先度をつけて対処するか，それでも問題が起こったときにどう対処するか，リソースをどのように手当てするか等々，ソフトウェア工学が扱うさまざまな分野に及ぶ問題でもあったと思う．いわばグランドチャレンジ型 [2] の共通問題であったと言えるかもしれない．

しかし，このチャレンジを終えて無事 2000 年を迎えて以降，これをきちんと整理して記録するといったことはなされていないように思う．問題の本質は何だったのか，それに対してどんな試みが行われ，何が有効であったのか．振り返られることなく，歴史に埋没しようとしている．2000 年問題自体は，1 つの個別具体的な問題であり，解決してしまえば 2 度起こることではない．しかし，問題の本質を見極め，一般化をすれば，今後も生き続ける共通問題になるのではないか．グランドチャレンジとしての使命を終えた問題が，今度はさまざまな解法の特性を比較検討するためのベンチマーク型 [2] の共通問題になる．

過去を振り返れば，2000 年問題以外にもこのような問題はありそうに思う．私たちはこれまで過去を振り返り分析・整理するということをあまり行ってこなかったが，そろそろそのような試みを始める時ではないだろうか．科学史という学問分野はあるが，いまだその中にない，ソフトウェア開発史を作

るのだ．あるいは，ソフトウェア考古学だろうか．考古学というにはあまりに歴史がないと思われるかもしれないが，何しろこの分野はドッグイヤーである．しかも，未整理の膨大なものの中から問題を発見するのは，まさに発掘のイメージだ．これまで，作るという視点で共通問題を考えてきたが，共通問題を発掘するのも良いだろう．

ソフトウェア開発史，あるいはソフトウェア考古学，いずれにせよ，過去を振り返り，整理し，過去に学ぶことも，未来のソフトウェアのために大切なことだと思う．ソフトウェアにかかわる技術，特にソフトウェア工学はまだ若い技術分野だと考えられてきたが，もう過去の振り返りが必要なほどには成熟してきたのだから．

教育現場における課題設定に基づく共通問題の考察（滝沢）

前回の特集では，共通問題と教育現場について，ソフトウェア開発教育とソフトウェア工学の類似性，すなわち，ソフトウェア開発教育の方法そのものの評価手段としての共通問題という観点で論じられていた[3]．ここでは，PBL（Problem Based Learning）やプロジェクト実験などの実践教育における課題設定を，共通問題の観点で捉えたい．

実践教育において，課題設定それだけでは共通問題としては不十分であり，学生らが課題に沿って提案する問題点や解決方法が共通問題に相当する役割を果たしている．利用者視点の強い学生らが自ら（共通）問題を設定するという過程があり，さまざまな分野における共通問題の開発の足がかりとなる可能性がある．この可能性について，筆者らが担当している科目の事例を踏まえて検討したい．

担当する PBL では「ファイル管理」という，学生にとってもなじみのある事柄を課題に設定している．ファイル管理の問題点と解決方法を明確にしたのち，解決方法に基づくツール開発を進める．課題設定としては「枯れた当たり前の問題」[3] と言え，研究・開発経験のない（少ない）学生らにとっては「何をすればよいか分からない」という状態には陥りにくい．また，新しい視点に基づいたさまざまな共通問題の提案が期待できる．たとえば，デスクトップ上のファイル整理を問題点として認識し，あまり参照していないファイルについては半自動的に移動あるいは削除する解決方法が提案されたことがある．これは，スケジューリングなどの時刻管理を意識した分野共通の問題への発展が期待できる．

一方，そのままではシナリオや基本設計が実現可能性の低いものとなる可能性もある．開発者（指導教員）視点による軌道修正を経て，適切な（共通）問題として洗練されると言えるだろう．また，今回とりあげた PBL は，進捗管理がウォーターフォールモデルに沿った指導である．5 ～ 6 人のグループ単位で 1 テーマを週 4 時間× 5 週にわたって進め，6 週目に報告書を提出させるとともに，成果発表会を行うというスタイルである．フィードバックを想定した開発手法の評価に適した共通問題を導くには，それに見合った進捗管理を導入しなければならないだろう．

筆者らが担当するもう 1 つの科目にプロジェクト実験がある．1 テーマを週 5 時間× 4 週にわたって進め，報告書を提出させるとともに，成果発表会を行う．この実験は，課題設定は情報工学分野（ICT システム開発）であるが，学生グループは情報工学以外の分野を専攻する学生を含む混成方式である．このため，利用者視点のシステム提案だけでなく，設計・評価も利用者視点が強い．これは，共通問題開発の観点では，長所とも短所ともなる．

プロジェクト実験におけるシステムの提案は，どのようなコンピュータシステムでも良いこととする一方，事前に示した開発手法や記法，評価方法に沿った，より詳細な成果物の作成に重点を置いている．このことにより，就職・進学支援のための情報共有システムといった，学生にとってより身近な提案内容が，共通問題として活用しやすい形式で作成されている．一方，たとえばシステム利用場所の快適さの追求など，開発・評価ではあまり重要ではない側面も強調されることがある．PBL の事例同様，開発

者視点をどのように取り入れるかが，より適切な問題設定を生み出すための大きな課題だろう．

共通問題に取り組む学生に対する動機付け（松本）

ソフトウェア工学における共通問題と同様に，ソフトウェア開発教育においても共通問題は重要である．また，能力の向上にあたり，取り組む学生の姿勢も重要である．

本誌 2013 年 9 月号「ソフトウェア開発教育における共通問題」[3] において，学生が取り組むソフトウェア開発教育の共通問題は，学生の興味を引き，そして学生に対するインセンティブが存在するものであるべきだと述べられている．しかしながら，ソフトウェア開発の経験の乏しい学生にとっては，興味が湧きにくいためにモチベーションが上がらない，効果的なインセンティブでない場合は学習意欲の維持が難しい，といった課題がある．そのため，共通問題に取り組む学生のモチベーションを向上させ，かつ学習意欲を維持させる仕組みを上手く取り入れる必要があるといえる．

ここでは，「共通問題に取り組む学生に対する動機付け」にゲーミフィケーションの活用を提案する．ゲーミフィケーションとは，ゲーム的要素をゲームとは異なる分野に応用することで，利用者のモチベーション向上を図る手法である．近年，ゲーミフィケーションは分野を問わず広く導入されており，教育分野においても積極的に取り入れられている．以下では，一例として教育用開発環境におけるゲーミフィケーションの導入について考える．

初めて教育用開発環境で開発に取り組む学生は，開発環境の使用方法が分からなかったり，開発の進め方がイメージしにくい．最初に使用方法や開発の進め方を視覚的に分かりやすく教えてくれるチュートリアル機能があると，つまずくことなく，開発を進めていくことができる．また，チュートリアルを最後まで完了することでポイントや特別なアイテムを獲得できる，といったインセンティブの付与によって学習意欲の創出につなげる．さらに，開発を進めていくごとに獲得できるボーナスポイント，開発に遅れが生じてしまった際にポイントが減るペナルティ，同じ開発環境で開発に取り組む学生のポイントランキングによる競争などの要素を取り入れて，学生の学習意欲を維持させる．

1 つのアイディアとしてゲーミフィケーションの活用について述べたが，共通問題に取り組む学生に対する動機付けの工夫は多面的に考えていかなければならない．

参考文献

1）ヴィトゲンシュタイン：論理哲学論考，藤本隆志，坂井秀寿 訳，法政大学出版局（1968）．
2）岸　知二，細合晋太郎：ソフトウェア工学の共通問題とは，情報処理，Vol.54, No.9, pp.878-881（2013）．
3）権藤克彦：ソフトウェア開発教育における共通問題，情報処理，Vol.54, No.9, pp.898-902（Sep. 2013）．

（2014 年 8 月 1 日受付）

鵜林尚靖（正会員）ubayashi@acm.org

1982 年広島大学理学部数学科卒業．1999 年東京大学大学院総合文化研究科広域科学専攻博士課程修了．博士（学術）．（株）東芝，九州工業大学を経て，2010 年より九州大学教授．2013 年より本会ソフトウェア工学研究会主査．

野田夏子（正会員）nnoda@shibaura-it.ac.jp

東京女子大学大学院理学研究科数学専攻．北陸先端科学技術大学院大学情報科学研究科博士課程修了．博士（情報科学）．NEC 勤務を経て，2013 年より芝浦工業大学准教授．

滝沢陽三（正会員）takizawa@ece.ibaraki-ct.ac.jp

1992 年茨城大学工学部情報工学科卒業．1997 年茨城大学大学院工学研究科博士後期課程修了．同年より茨城工業高等専門学校教員（2007 年より准教授）．

松本　明　s151045@center.wakayama-u.ac.jp

2014 年和歌山大学システム工学部情報通信システム学科卒業．現在，和歌山大学大学院システム工学研究科システム工学専攻博士前期課程在籍．