# ① ソフトウェア工学の共通問題とは

岸　知二（早稲田大学）　細合晋太郎（九州大学）

## それは酒屋から始まった

ソフトウェア工学分野の人には，酒屋の共通問題をご存じの方も多いと思う．本誌 1984 年 9 月号の「共通問題によるプログラム設計技法解説」[1] 以降 3 回の特集で，ソフトウェア設計技法，プログラム技法の比較に使われた問題である．

酒屋の共通問題の詳細については，本特集の「酒屋問題再考」を見ていただくとして，本稿ではこの共通問題を通してソフトウェア工学という分野を眺めるとともに，共通問題というものが持つ今日的な意義について考える．

## 設計技法と共通問題

まず設計技法やプログラム技法と共通問題とのかかわりについて考えてみる．

小さなプログラムはともかく，実際に企業等で作られるほとんどのソフトウェアはいきなりプログラミングすることは不可能であり，まずソフトウェアをどう作るか，内部の構造を決めてから，そこで必要となるプログラムを作るという手順を踏む．たとえばワードプロセッサであれば，ユーザインタフェース部，ファイル入出力部，印刷部など，必要となる構成要素（モジュールと呼ぶ）を決め，それらが互いにどうかかわるかを決める．この作業（モジュール化と呼ぶ）の良し悪しは，たとえば新たな機能を追加しやすいかどうか，ある機能追加をしたことによる影響波及が捉えやすいかどうかといったソフトウェアの特性を大きく決定づけるため，ソフトウェア設計上の大きな課題となる．

ソフトウェア設計技法やプログラム技法とは，端的に言えばモジュール化の考え方や記述方法などの体系である．たとえば構造化技法ではシステムを入力から出力への変換機能のネットワークとして捉えるし，オブジェクト指向手法では協調動作するオブジェクトの集合と捉えてモジュール化をする．前掲の特集は，こうした技法の特徴を明確にするために同一の共通問題をそれぞれの技法で解いて見せたもので，連載を通じて 15 の技法での解法が示されている．

モジュール化の議論は 1970 年前後から始まったが，その後も常にソフトウェア工学の 1 つの中心的課題であり続けている．オブジェクト指向，アスペクト指向，あるいはコンポーネントウェアなどの技術の発展は，いずれもよりよいモジュール化の追及の歴史と捉えることができる．酒屋問題が長くコミュニティで取り上げられてきたのは，こうしたソフトウェア工学の重要課題にかかわっているからであろう．

## ソフトウェア工学の特徴

この酒屋の共通問題を通して，ソフトウェア工学という分野の持ついくつかの特徴が浮かび上がってくる．

この問題は酒屋の在庫管理システムの設計やプログラムを求めているが，システムの使われるビジネス状況や実現技術などについては具体的に書かれていない．またどういう変更を考慮せよとか，性能要求など解に対する具体的な要件も与えられていない．問題の末尾には「あいまいな点は，適当に解釈して下さい」とまで書かれている．

設計やプログラムの技法は，絶対に優れている唯一の方法があるわけではない．対象分野，設計やプ

ログラムを行う人のスキルやバックグラウンド，開発プロジェクトの置かれている技術的，組織的，あるいはビジネス的な状況など，さまざまな要因によってそこに適した特徴を持った技法が選ばれる．共通問題は解の優劣を決めるためのものではなく，各技法の持つ特徴をくっきりとさせることが目的なのである．モデリング技術，再利用技術，プロジェクト管理技術など，ソフトウェア工学の扱う多くの問題はこうしたソフトな問題，つまり明確に優劣を決めることのできる目的，評価軸，評価尺度が設定しづらいものなのである．

また，この問題は酒屋という題材を用いた現実問題を指向した設問となっている．小さな例題をきれいにモジュール化してみせても，多くの人は現実の問題に適用できるとは信じない．現実問題に対して有用な技術であることを主張するには，ある程度リアリティを感じさせる問題でなければならないのである．モジュール化に限らず，ソフトウェア工学の多くの技術は，小さな問題への適用結果から，それを大きな問題に適用したときにどうなるかということを推察することが困難なものが多い．つまりスケーラビリティに関する理論や考え方が不明瞭なのである．

さらに，前掲の特集で扱われた 15 の技法には，ソフトウェア工学の教科書に掲載されるような著名な研究者の提唱する技法から，特定の企業の製品レベルの技法までが混在している．例えて言えば，特定のレストランの具体的なメニューと麺類一般とを比較しているような違和感がなくはないのだが，ソフトウェア工学では，技術の名称と製品の名称とが明確な区分をされずに使われることがままある．ソフトウェア技術は実務での利用が急速に発展してきたこともあり，要素的な技術や理論の発展と，実用品・製品の開発がある意味同時進行し，その両者の関係性が必ずしも明確になっていないのである．

もちろんこれらは必ずしもソフトウェア工学だけの特徴ではないかもしれないが，ソフトウェアを取り巻く技術・ビジネス環境の急速な変化や発展などとあいまって，この分野の性格を示すものとなっている．

## ソフトウェア工学研究の評価

こうした特徴は，ソフトウェア工学研究の進め方，特にその評価方法に影響を与えている．端的に言えば，評価が難しいのである．

今回の特集は，ソフトウェア工学研究会が主催したワークショップでの「ソフトウェア工学研究の評価」に関する議論がきっかけとなっている[2]．典型的には研究開発中の技術を論文で発表する際に，その有効性や妥当性を主張するための，枠組みが十分に確立していないのである．

たとえば論文に対するネガティブ評価の 1 つの典型は，提案が現実問題にスケールするかどうかが不明だという指摘である．ある意味重要でもっともな指摘ではあるが，困ったことに何を示せばスケールすると認められるかその基準がないため，ともすれば評価が査読者の主観に強く依存してしまう危険性があるのである．一方，産業界での研究の価値はビジネスへの貢献という観点で見られる．工学であるから役立つことが最終的な目標であることに異論はないが，新しい技術を本当に役立つものへと育てるために，その途中段階をどう評価するのかという視点が不十分なのである．その結果，自動検証やプログラム解析などコンピュータ科学に基づいた定量評価が相対的に容易な分野の論文が増加する一方，コンセプト論文や他の分野の論文が採択されづらくなり，また現場との距離が乖離するという問題も出てきている．

## 技術の特性評価の重要性

ソフトウェア工学といってもさまざまな技術があり，それに応じてその性格も評価の方法も違い得る．ワークショップでも性格に応じた研究の整理が議論された[3]．**図 -1** は，そうした議論の過程で示された研究の分類の例である．

ここでは，研究指向か現場指向か（横軸）と技術指向か実証指向か（縦軸）という 2 軸で整理し，研究を 4 つに分類している．

- **カテゴリ 1（研究・技術指向）**：新たなコンセプトや技術の提案が目標となる分野である．
- **カテゴリ 2（研究・実証指向）**：提案技術の実証実験が目標となる分野である．
- **カテゴリ 3（現場・技術指向）**：既知の技術を利用して現場で有効なソフトウェアを開発することが目標となる分野である．
- **カテゴリ 4（現場・実証指向）**：現場の知識や工夫の知識化が目標となる分野である．

この分類でいえば，本稿で議論している評価の問題が顕著なのは，カテゴリ 1 である．将来の実用技術の種となる潜在性はあるが，現時点では実用的な適用成果のないものをいかに評価するかが重要となる．いきなりスケールして役立つ技術が生み出されるわけではないため，このカテゴリでの研究を健全かつ効果的に識別し，価値あると考えられるものを育て，他のカテゴリの研究へとつながる評価の枠組みを確立することが重要である．

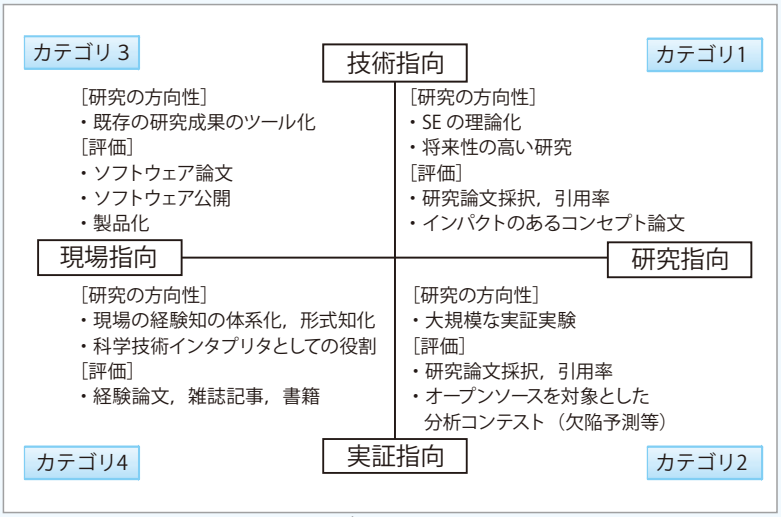


**図 -1　ソフトウェア工学研究の分類** [3]

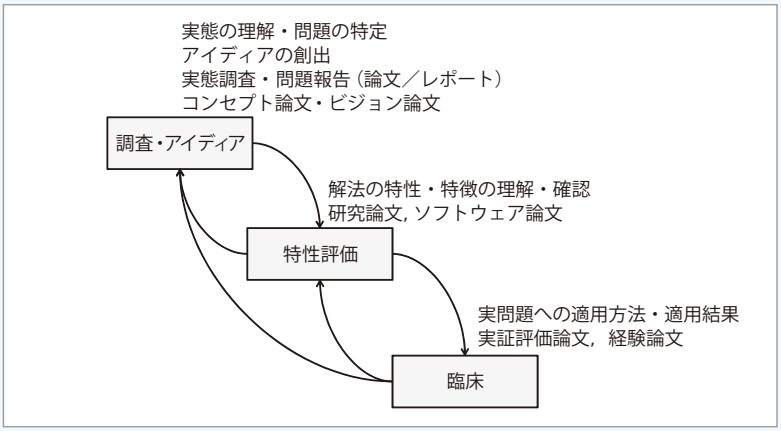


**図 -2　ソフトウェア工学研究のステップ**

ワークショップではたとえば新薬などの開発ステップとの対比でこの議論がなされた．新薬はラボ実験されたものがいきなり臨床に使われるわけではなく，その前に動物実験等で有効性等を評価するステップが存在する．たとえば遺伝子の特性を揃えたノックアウトマウスなどが利用され，制御された環境下で特徴を評価する．

**図 -2** は，これをソフトウェア工学に例えたものである [4]．ここでは以下の研究ステップが示されている．

- **調査・アイディア**：問題の特定と，アイディアの形成にかかわる段階である．調査や事実に基づく問題の妥当性は問われるが，アイディアそのものには客観的な評価指標が存在しない．
- **特性評価**：アイディアを具体的な理論や技術として整理する段階である．評価においては同種の技術との特性の違いを明確にすることが重要である．ここでは必ずしもスケールやリアリティを要求しない．
- **臨床**：理論や技術を実問題で評価する段階である．

ソフトウェア工学の特徴を踏まえると，この特性評価の確立が重要となる．

## 共通問題のタイプと意義

共通問題は，こうしたソフトウェア工学研究の評価に利用されてきた．ソフトウェア工学分野で，共通問題と呼ばれているものは，酒屋問題だけでなく国内外に多く存在するが，それらは大きく以下の 3 つに分類することができる．

- **ベンチマーク型（分析比較型）**：複数の解法の存在する問題を共通問題として提示し，それらの解法の特性を比較するものである．酒屋問題はこれに該当する．共通の問題を解くことにより，提案する解法がどういう特徴を持つかを，他の解法との比較の中で相対的に評価することができる．
- **コンテスト型（結果比較型）**：1 つの目的を達成するための解法を，その目的の達成度合いによって比較するものである．ベンチマーク型が特性の分析的な比較や特性間トレードオフの議論が重視されるのに対して，最終的な目的の達成度，達成への総合力，バランス力といった工学的観点を重視して評価する点に特徴がある．ET ロボコン[5]などが該当する．
- **グランドチャレンジ型（課題達成型あるいは裾野効果型）**：解決されることにより，その分野にとって大きな進歩につながるような，一定の意義と難易度を持った問題を提示し，解を求めるものである．提示された課題を解くことそのものが大きなチャレンジであり，またそれを解くためにはさまざまな副課題や関連課題の解決が必要となることで，研究の裾野を広げる効果を持つものである．形式手法のグランドチャレンジ[6]などが該当する．

こうした共通問題を研究のステップに応じて活用することで，研究の評価さらにはその発展に寄与することができる．典型的にはベンチマーク型は特性評価で，コンテスト型は特性評価から臨床へ進む段階で，またグランドチャレンジ型は調査・アイディアを含んだ研究ステップ全体にかかわると考えられる．

## 新しい共通問題の必要性

ここまで述べてきたソフトウェア工学研究の評価の難しさの指摘や，研究への共通問題への活用という議論は，ソフトウェア工学のコミュニティで以前よりなされてきたものである．しかしながら共通問題の議論を再度行い，具体的な共通問題について見直すことは意義があると考える．

まず酒屋問題から 30 年がたち，その間にソフトウェア工学を取り巻く状況が大きく変わっていることである．新しいビジネス環境や技術環境は，ソフトウェアに対して新しい要求を課しており，そうした時代に適した共通問題は何か，ここで見直すことは意味がある．

また共通問題を通してソフトウェア工学が解くべき課題を浮き彫りにする意義である．酒屋問題が扱ったモジュール化の問題は依然として重要な課題であり続けているが，それ以外にも新たな課題が台頭している．コミュニティの方向性を示し，また分野外の人にソフトウェア工学を正しく理解し位置づけてもらうためにも，今ソフトウェア工学でどういう課題が重要なのかを示すことは意義深い．

ソフトウェア工学は若い学問ではあるが，時代の要請の中で短期間に拡大し，また激しい変化の中で発展してきた分野である．そうした中で，実務でのニーズに応じながら，評価を含め研究の方法論を模索し続けている分野でもある．ソフトウェア工学分野以外の読者にもソフトウェア工学を知っていただくひとつの切り口として，共通問題を眺めていただければ幸いである．

**参考文献**

1）山崎利治：共通問題によるプログラム設計技法解説，情報処理，Vol.25, No.9, p.934（Sep. 1984）.
2）ウィンターワークショップ・イン・倉敷 2010 論文集，IPSJ, Symposium Series, Vol.2010, No.3.
3）鵜林尚靖：ソフトウェア工学研究のための評価フレームワーク，ウィンターワークショップ・イン・倉敷 2010 論文集，pp.151-152（2010）.
4）平山雅之：ソフトウェア工学分野の技術評価フレーム，ウィンターワークショップ・イン・琵琶湖論文集，pp.133-134（2012）.
5）http://www.etrobo.jp/2013/（2013 年度の情報）
6）Hoare, T. : The Verifying Compiler : A Grand Challenge for Computing Research, In Journal of the ACM, Vol.50, No.1, pp.63-69 (2003).

（2013 年 7 月 1 日受付）

**岸　知二**（正会員）kishi@waseda.jp

1982 年京都大学・情報工学専攻修了．1982 年 NEC 入社．2002 年北陸先端大・情報科学研究科博士課程修了．博士（情報科学）．現在，早稲田大学経営システム工学科教授．ソフトウェアアーキテクチャ，ソフトウェアプロダクトライン，ソフトウェア設計，設計検証の研究に従事．

**細合晋太郎**（正会員）hosoai@qito.kyushu-u.ac.jp

2007 年北陸先端科学技術大学院大学情報科学研究科博士前期課程修了．2013 年同博士後期課程単位取得退学．現在，九州大学大学院システム情報科学研究院 学術研究員．