基般

④ 座談会
～共通問題を通して見る ソフトウェア工学の 30 年～

紫合　治（東京電機大学）　青山幹雄（南山大学）
鵜林尚靖（九州大学）　野田夏子（芝浦工業大学）　岸　知二（早稲田大学）

共通問題を題材に，酒屋問題の当事者の 1 人であった東京電機大学の紫合治先生と，その後のソフトウェア工学研究会の主査・幹事経験者で座談会を行った．出席者は以下である（敬称略）．

- 紫合　治（東京電機大学）
- 青山幹雄（南山大学）
- 鵜林尚靖（九州大学）
- 野田夏子（芝浦工業大学）
- 岸　知二（早稲田大学）　司会

岸：今日はお忙しいところお集まりいただきまして，ありがとうございます．共通問題について 30 周年ということで，今日は前回の企画のときに唯一名前が載っている紫合先生にも参加いただいて，ざっくばらんにお話しできたらと思います．紫合先生，当時のことは，覚えていらっしゃいますか．

30 年前のソフトウェア工学界

紫合：これは 84 年でしょうかね．構造化プログラムとかがはやりだして 10 年ぐらいですよね．その間に，ワーニエ法だ，ジャクソン法だ，といくつか手法ができてきて，オブジェクト指向とか，その辺がまだ新しい話題という感じでしたね．これからこのような分野でいろいろと新しくなるなという雰囲気がありましたかね．

青山：ソフトウェア工学の国際会議 ICSE が日本であったのが 82 年でしたっけ．

岸：構造化プログラミングの 3 構造で木構造図を描くような設計手法よりもちょっと上流の方法論がここで話題になっている気もするのですけれども，ソフト工学の話題というとそのあたりがメインだったのですか．

紫合：構造化プログラミングで始まって，少しずつモジュール化に流れていったという感じですかね．データ抽象化だとか，データを考えていこうという雰囲気のときにこれがあったという感じがしますね．

岸：これを共通問題で議論しようというのは何か経緯があったのでしょうか？

紫合：1 つの問題にしていろいろなやり方で解いてみると違いがよく分かるのではないかというような感じだったと思いますけどね．

青山：誰かがこれでやろうとか，具体的に言わないとなかなか進まないでしょうね．

岸：シンポジウムで同じ問題で比較してみましょうと．そのときの最終パネルに紫合先生も出ていらっしゃるのですけども．

紫合：僕が覚えているのは，方法論は形から入って心に至るというようなね，そのようなことを言った覚えはありますね．

岸：それは欧米の人にとってもそうなのですか．

紫合：欧米の人は心というよりもまず形という感じですね．

青山：だから，構造化ということをすごく大事にしますよね．

鵜林：心というとアジャイルのときに心と結構言っていましたよね．

岸：20 年先を行っていたわけですね（笑）．

成果の普及と共通問題

岸：そのときの座談会を読ませていただくと，企業

で普及しない，普及に困っているということが書かれていて．

紫合：企業で研究をしていたら，それを実用化しろとか企業の中で使えるようにしろという使命があって，なかなかうまくいかないという感じですね．そういうのはありましたよね．

鵜林：でも長い目で見たとき，抽象データの話はオブジェクト指向という形で普及していますし，いろいろなパターンとかいうのもデザインパターンとかいう形につながっています．

座談会の様子（JISA 会議室）正面左より紫合，野田，鵜林

青山：少し前に ICSE のメンバでインパクトプロジェクトというのをやっていたのですね．そこでいわれたのは，普及しようとしている人はフロントランナーなのでかなり困難があるのですが，10 年後になるとかなり普及していると．

岸:オブジェクト指向もだいぶ時間がかかりましたよね．

青山：習得が難しいと思いますね．だからそういうものをうまく理解するための手段として共通問題が使えればいいと思うのですけれども．各社がやっている方法論，表記法なんかを普及させようということで 1 つの役割があったのかなと当時思ったのですけれども．

紫合：この酒屋の問題ということですね．

鵜林：コンパクトというのと，事務処理計算での典型的な問題で，誰しもがすぐ理解できる問題だということがよく使われている理由の 1 つなのかもしれないですね．

青山：オブジェクト指向の初期のときによく使われたのに図書館問題があって，みんなが理解しやすい問題で役に立つのではないかと思うのですけれどもね．

野田：図書館問題は，88 年の IEEE ソフトウェアで，要求仕様を書くということを比較するというのでやっていたりして．

鵜林：この当時はいわゆるプログラム周りの比較，検討の共通問題なのですが，30 年たった今というのは，考えなくてはいけないことがたくさんあり過ぎて，それが共通問題はこれだとなかなか 1 つに絞れない原因の 1 つではないかと思います．

青山：技術が進歩して，ソフトウェア工学も下流から上流に進歩していると思います．プログラミングが設計，アーキテクチャの話になって，今は要求工学，そうすると，それぞれ考えるコンサーンは非常に違うわけですよね．

紫合：これは，当時のプログラム設計技法の評価とか，そういう意味での問題としては非常にいいと思いますね．

青山：その後もいくつか共通問題はつくられていますよね．

野田：やはりこう，皆さんが知っていて「これ」と指させるのは，酒屋ですよね．あとは図書館．それとこの間の特集で取り上げさせていただいたのは，話題沸騰ポット．

鵜林:検証系はありますね．SAT☆1 だとか，SMT☆2 とか，ああいうもののコンテストみたいなものとか，結果が見えるのがいいのでしょうね．

☆1 Satisfiability Problem, 命題論理の充足可能性問題.
☆2 Satisfiable Modulo Theories, 述語論理の充足可能性問題.

どう評価するか

青山：評価が比較的しやすいものだよね．ただ，組込みシステムで飛行船のコンテストなんかはずっとやっていますよね．

岸：そもそもモジュラリティというのも昔から永遠の課題だけれども，ある意味で評価軸がないという感じはしますよね．

青山：ただ，構造化設計で言うと定性的なマイヤーの 7 段階というのは昔から知られている．オブジェクト指向でも一応いくつかの指標があるのだけれども．ただし，やはりどういうものをどういう目的で設計しているかという意味的な部分が非常に大事なので，一律にはそんなに簡単には解けない問題でしょうね．

岸：先ほどの飛行船も，よく飛ぶかというのは実行時の特性ですよね．だけどもモジュラリティは開発時の特性だから，いいか悪いかというのは，非常に評価しづらいですよね．

野田：1 週間前ぐらいに突然仕様を変えたら良いのでは？　そのときに一番うまく対応できたものを評価する（笑）．

青山：2 年前ぐらいに要求工学の e ラーニングをつくったのですが，クローズな問題にすれば良いか悪いかと点数で評価ができるのだけれど，一応オープンな問題にしようと．そうすると，答えも一律でないので，教えるのが大変なのですよね．でも，そのほうがやはり本来の要求工学の課題の問題としてはふさわしいだろうという議論をしたことがあります．

鵜林：私もそのように思います．ただ，学生からは「標準解答はないのですか」という質問が多いです．やはり解が決まっていないのは，結構つらいみたいな感じですね．今の学生は．

野田：わざとあやふやな仕様を見せるとすごくいやがりますね．

鵜林：そこが，設計教育の難しいところですね．数学とか，ほかの学問と違って．

青山：必ずしも根拠が明確でないから，方法論にしても割と流行的なところがありますよね．それはやはり非常に危険なところがあって，だから，もちろん解は 1 つではないですが，科学的な根拠を我々は考えないといけないと思うのですね．

紫合：昔，MIT[☆3] に行って Barbara Liskov と話をしたことがあって，いい抽象化，いい抽象データというのはどういうことですかと聞いたら，それは言えないけれども，なんか持ってきたらいいか悪いかは言えるとか言っていましたね．

鵜林：無意識にいい，悪いとは言えるのだけれども，それをどういうふうに知識化するのかが難しいのでしょうね．

設計や抽象化の教育

鵜林：今の話を聞いて，ちょっと思ったことがあります．今までの共通問題は何か課題を解くタイプが多かったのですが，解かせるのではなく，たとえば，ある設計を見せ，「良い抽象化か，それとの悪い抽象化か」を判断させるための共通問題があってもよいかと思いました．優れた抽象化の設計とかを学生に見せて，それでどう思うかみたいな感じです．

紫合：いいですね．

鵜林：まずは見せるということ．書けと言ってもなかなか難しいので．はやはりトレーニングはしないといけないのかなという感じはしていますね．

青山：ビューの観点で言えば，基本的には機能の抽象の構造と，データ抽象と，振舞いの抽象というか，その 3 つは一応教えるのですけれども．そういうことはきちんと理解してほしいなというのはありますね．

紫合：プログラミングの話ではなくて，設計というよりもむしろ世界をどう見るかというか．

岸：問題フレームみたいな．

紫合：そうですね．いい構造というのは世界をすんなりと反映しているというような雰囲気が，ジャクソン法の昔からあったので，それはずっと生きているという感じはしますよね．

☆3 Massachusetts Institute of Technology, マサチューセッツ工科大学.

野田：たとえば，学生なんかに銀行のシステム，あれはソフトウェアでできているのだと言っても，そこにどうソフトがかかわってみたいなことはまったく想像ができない．

青山：先ほどの要求工学のｅラーニングですが，そのときに使ったのはコンビニなのですね．天気によって売り上げが変わりましたとか，そういう身近なところで誰でも理解できる．

野田：当時はこれのターゲットは誰だったのですか．共通問題を策定しようという．

紫合：あまり学生という感じはなくて，ソフトウェア工学というのは，企業の方がかなり関心持っていたという感じはありますよね．

青山：そもそもソフトウェア工学を大学で教えてなかった時代ですからね．

岸：確かに．でも，今では，UML[☆4]を知らない人は少ないですよね．

鵜林：あと最近の学生さんはPBL[☆5]とかやっていて，結構アジャイル，スクラムとかを普通に使ってやったりとか，バージョン管理をやったりいわゆるツールをうまく使いこなす土壌は昔に比べると良くなっているのかなとか思いますね．

青山：そういう意味では，研究も含めたオープンソースの活用があるので，環境は良くなっていると思いますね．だから，その環境を使いこなす技術がどうかということだと思うのですね．

岸：ソフトウェア工学というのは，心というか，うまく説明できないところが残る以上，こういういい共通問題的なものというのはまだ必要なのでしょうね．

青山：建築もコンペもやりますよね．そこはやはり優れた設計というのが出てくるというか，そういうものを評価してあげるということは必要だと思うのだよね．機械的にできることではないということだね．

野田：あと建築とかに比べるとあまりにも数字を言い過ぎるようになったのかもしれないですね．

☆4 Unified Modeling Language, 統一モデリング言語.
☆5 Project Based Learning, 課題解決型学習.


紫合治氏

青山：以前あったのは，オープンソースのいいプログラムを見せて，「これはいいね」とか思わせるというパターンなのですよね．

ソフトウェアを取り巻く状況の変化

紫合：そういう意味では，OSもコンパイラもみんなあるものというのか，今は中身を見るものではないというような感じですよね．

岸：そうですね．見えなくなっているからエキサイトしないのですよね．

青山：最近話題の自動運転なんかは，あれも本当にソフトウェアの技術ですよね．ソフトウェア工学にとっては非常に大きなチャンスであるし，面白い世界というのを感じることができる．

岸：ソフトウェア工学は一種の生産技術ですが，現場のほうがむしろすごいスピードで動いてしまっていてというところがありますよね．

青山：学生の卒論とか修論とか見ていると，アパッチのコンポーネントとか持ってきてやるわけですね．私は，これはいいと思うのですね．その上に自分でなんかつくって工夫してやるということは．でも，設計力とかはなかなかすぐには上がっていかない．

野田：何でもいいから持ってきてつなげるというのでは，やはりいろいろな問題が起こるわけなので．

紫合：たとえばDevOpsなんかやっている人たちの層が，ソフトウェア工学の層とちょっと合わないよ

うな感じが，そうでもないですか．

鵜林：作り方がちょっとずつ変わってきて，ソフトウェア工学が新しい時代のソフトウェア開発と少しずれているのかもしれないですね．

青山：DevOps なんか見ると，やはり単にプログラミングとか，設計だけでなくて，オペレーションも含めてスコープを広げていこうという．開発だけではなくて，周辺も扱わないといけないのかなと思います．

鵜林：当時の産業界というのは自分たちで方法論だとか，ツールを開発していて，それを普及したいとか，それと同時に他社はどうなっているのかという関心があったかと思います．今の産業界がそういうものを出すことにビジネスの意義を感じているのでしょうか？

青山：先日方法論をまとめた本を出したのですが，各社の方法論を 1 章ずつ書いてもらったのです．やはり上流にシフトしていると思うのです．つまり下流はアウトソーシングをしたりとか，今はやはり要件定義が一番問題の環境だと考えています．

岸：要求だとどういうところがベンチマークするときの比較の視点になるのですか．

青山：プロセスとしてどういうことをカバーできているのか．どういう技術があるのかということ．あとは成果物で何があるかという．

ソフトウェア工学が広がっている

鵜林：最近の学生 PBL では，スマホのアプリケーションを開発するケースが増えています．スマホに限定はされないのですが，最近のアプリケーションは，デザインというか見た目がやはり重要で，色の使い方とか，そういうノウハウがあって，そういうのは今までのソフトウェア工学の授業ではまったく教えてこなかったわけですよね．ただ，今後は，アーティスト的な感性とプロフェッショナルとして技術力の 2 つを兼ね備えた人材が求められていくような気がします．

青山：Web デザインの会社は両方の人材をきちんと用意して連携して仕事をするような専門会社ですけれども．ただ，大学のほうは，あまりそこは．

鵜林：プログラム大好きな人が作品としてものを出すということをやると，そういったことが気になると思います．「自分の作品」というか．

野田：名前が出ないですよね．建築の世界だとこの建築家が作ったいい建築はこれ，と指させるのですが，このソフトウェア，アーキテクト誰々みたいなのが出るわけでもないですよね．

紫合：ゲームソフトなんかはあるのですかね．誰々さんというのがあるかもしれない．

鵜林：そうですね．より作品に近いですよね．

岸：いろいろやっているけど，ソフトウェア工学の今大事な話題はこれとこれだよみたいな目立つものがあるといいかなという気もするのですけれどね．

青山：素朴な考え方をすると，ソフトウェア工学というのは基本的には現実世界をソフトウェアで写すためのモデルとしたら，すべて対象がソフトウェア工学の対象になってしまうわけですよね．

鵜林：最近の企業の事業戦略を見ていると，スマートシティだとか，社会そのものをシステム化していく方向に事業を大きく転換しつつあるように感じます．私からすると，社会システムそのものも広い意味でのソフトウェアで，ソフトウェア工学の対象のように思います．今まではコンピュータソフトウェアの開発を支援することがソフトウェア工学の大きな役目だったのですが，いま一度，どこにソフトウェア工学のアイデンティティを求めるかについて考え直さなければいけないのかもしれません．本当に，コンピュータが見えなくなってしまっているという感じですね．

青山：Nicholas G. Carr が『IT Doesn't Matter』という本を書いていますが，IT はコモデティ化してしまったので，それ自身が価値を持つわけなくて，どう使うかで価値が出てくるという言い方ですよね．ただし，それをどう作るかというのは，本質的な問題なので，そこは譲れないところだと思うのですけれども．

産業界と学会

紫合：この当時は，いろいろな方法論みたいな，それぞれの人が，それぞれがいいと言ってやっていたので，そういうのは，今はどうなのですかね．

岸：どちらかというとソリューションカタログとか，パターンに流れていって．でも最近の複雑な問題は上にシフトして，ゆり戻しがあるのかもしれないですね．

青山：たとえば，大きな例としては，NTT データの TERASOLUNA と富士通の Tri-Shaping がありますけれども，非常に壮大な方法論ですよね．いろいろな技術がたくさん入っている．問題もそれだけ複雑だということだと思いますよね．

岸：やはりそこにより要素的なものがあるわけですよね．ステークホルダ分析をどうするとか，インタビューをどうするだとか，いろいろな話はあって．

紫合：でも，共通問題になる条件の 1 つとして，その複雑さがそれほどではなくて，でも，自明ではなくてというような．

青山：そういう意味では，デザインパターンとか，そのために問題を切りだしてほどほどの抽象度でほどほどの解法を与えるというところでは同じような役割は果たしているのかもしれないね．

紫合：共通問題のもう 1 つの目的というかあれは，なんか新しいアイディアを発表するときに使うということはありますよね．

岸：論文書いても評価が甘いとか，これはスケールしていないから駄目だと言われると．少なくともこの問題をやっておけば文句言われないというものがあれば（笑）．

鵜林：萌芽研究だったらこのレベルの共通問題でここまで評価できたらいいとかあるといいですね．

岸：ソフトウェア工学は結構厳しいですよね．研究のエコシステムをつくらないと，みんながつんがつんと実はつぶされていて，良くないかなという．

青山：インパクトのときもそういう話をしていたのですよね．たとえば Richard N. Taylor が言っていましたが，ボアリングだと．過剰に評価を要求していると．だから，ほとんどイノベーターのアイディアはなくて，従来のを，ちょっと改善して評価してこうでしたと．

岸：通りやすいものね．

青山：そうすると，新しいアイディアは出てこない．やはり少し別の切り口で評価をしないとこの分野は非常に停滞すると．たとえば Amazon にしても Google にしても別に学会の評価は気にしていないわけです．新しいアイディアをつくってどんどんつくっていくような人たちをどう一緒に取り込むとかいうことをやっていかないと，非常に危険だと思うのですよね．

鵜林：ソフトウェア工学の歴史の中で，本当の意味で影響を及ぼしたのは必ずしも学術研究ではない気がします．デザインパターン，リファクタリング，アジャイルなどは今では当たり前のように普及して

座談会の様子（JISA 会議室）正面左より青山，岸

いますが，これらは書籍にはなっているけれども，初期のアイディアが学術論文として評価されたわけではありませんよね．

面白さが源泉

岸：若い人は面白いことから入りますよね．

鵜林：私が若い頃は UNIX の OS の勉強をしたりしていましたね．あの頃は新しい技術でしたから．第五世代コンピュータが話題になっていた時期ですから，Prolog を勉強したりというのもありました．いつの時代も同じですね．若い人が面白いことをやりたいというのは．

野田：面白いと思っていないですよね，若い人は．やはりソフトウェア自体が何だか見えなくなっているせいだと思うのですけれどね．

青山：学生にこれは面白いぞと毎日言い続けないと私は思っていますよ．コミュニティの中でもそうだし，外にももっと言わないといけないのではないかと．

野田：何なのでしょうね，でも，その「昔は面白かった」という時代．

紫合：やはり波があってね．たとえば，構造化プログラミングとか，またデザインパターンとか，オブジェクト指向で盛り上がって．恐らく DevOps とか，クラウドとかでまた，盛り上がるというような．

青山：80 年代は日本の会社がみんなソフトウェア工学を大事だと思って研究所を作ったりとか，海外からもたとえば Michael A. Cusumano のような非常に好意的な書き方をした論文とか本が出て，ある意味で高揚感のある時代ですね．

紫合：ただ，80 年代で，うわーっと盛り上がったのは，どちらかというと企業ですね．

青山：そうです．

紫合：それに対して 90 年代，オブジェクト指向で盛り上がったのは，結構アカデミック，そういう雰囲気はあって．

青山：今の現在の高校生から見ると，すでにスマホは使っているし，だからポテンシャルはあると思っているわけですね．ポテンシャルを引き出すためのわれわれが仕掛けをもっと持つ必要があるかなとは思っていますね．

紫合：ソフトウェアというのが付いている研究室というのは比較的人気があるというようなところはありますね．その 1 つの理由は，やはりゲーム好きみたいな話が 1 つと，何か知らないけど就職は SE ね．

青山：それだけの雇用吸収力があるし，それだけの必要性があると思いますよね．

岸：今日はソフトウェア工学の多面的なお話をしていただきまして，ありがとうございました．

紫合　治（正会員）shigo@mail.dendai.ac.jp

1971 年東京電機大学工学部電子工学科卒業．博士（工学）．日本電気（株）を経て，2003 年より東京電機大学教授．

青山幹雄（正会員）mikio.aoyama@nifty.com

1980 年岡山大学大学院工学研究科修士課程修了．博士（工学）．富士通（株），新潟工科大学教授を経て，2001 年より南山大学教授．

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．

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修了．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）．日本電気（株）を経て，2013 年より芝浦工業大学准教授．

岸　知二（正会員）kishi@waseda.jp

1982 年京都大学工学研究科情報工学専攻修了．2002 年北陸先端科学技術大学院大学博士後期課程修了．博士（情報科学）．日本電気（株），北陸先端大学を経て，2009 年より早稲田大学教授．