ソフトウェア工学の研究と教育を振り返って

随　筆

井　上　克　郎*

Evolution of My Software Engineering Research and Education

Key Words：Program Analysis, Code Clone, enPiT

はじめに

筆者は40年近くにわたり、大阪大学でソフトウェア工学に関する教育や研究を続けてきた。2022年3月末で定年退職し、4月より名古屋市にある南山大学理工学部ソフトウェア工学科において、引き続きソフトウェアに関する教育・研究を続けている。本稿では、筆者の過去を振り返り、ソフトウェアやソフトウェア工学との接点を紹介するとともに、思いや反省などを紹介する。なお筆者の記憶違いなどによる間違いがあるかもしれないが容赦願いたい。

ソフトウェアへの入り口

ソフトウェアに触れる前に大いに興味を持ったのが電子機器である。いわゆるラジオ少年で、小中学生時代には、月刊誌の「ラジオの製作」を熟読し、真空管やトランジスタの受信機を作ることに熱中した。当時の当然の成り行きとしてアマチュア無線の免許を取得して、中学、高校のアマチュア無線クラブに所属し、送信機を作製して電波を出したりした。ただし、まだマイコンが普及する前の時代でありコンピュータを触わる機会もなく、ソフトウェアは興味の範囲外であった。

ちょうど高校3年生の頃、雑誌か何かで、プログラマブル電卓の比較的詳しい解説記事があって、キー入力を命令列として記憶、実行でき、計算値による条件ジャンプを行うことによりループの実行、そして関数値の収束計算ができることが紹介されていた。多分ヒューレットパッカード社のHP-65プログラマブル電卓の解説記事であっただろうと思われるが、詳細は不明である。単純な筆者は、「プログラムは素晴らしい、何でも計算できる、これからはコンピュータの時代だ！」と感化され、情報工学科へ道を選択した。

1975年に大阪大学基礎工学部情報工学科に入学でき、導入教育ではFACOM-45Sという大型計算機を使いFortranでプログラムを書き始めた。入力はパンチカード、出力はラインプリンタで、それなりにプロっぽい環境なので面白かったが、HPの電卓の逆ポーランド記法のようなマニアック感やワクワク感はあまり無かった。そこで当時普及し始めたマイコンチップ（MC6800）を買って、ワンボードマイコンを作って動かしたりもしたが、トグルスイッチとLEDを並べただけの入出力では、ごく小さな機械語プログラムをハンドアセンブルして入力するのが限界だった。ただこのワンボードマイコンで計算機のハードとソフトの底辺をじっくり学ぶことができ、後々いろいろな場面で役立った。

大学4年生になって、嵩忠雄先生の研究室に配属され、研究室のミニコンPDP-11/20を使うようになったが、これは大変魅力的だった。紙テープによるブートやテレタイプASR−33の入出力だったが、アーキテクチャやOSが洗練され、何かと感心することが多かった。PDP-11/40もあり、グラフィック端末が繋がれ、ライトペンを使った月面着陸ゲームが動くのには大変驚かされた。卒業研究は、山村三朗先生の指導のもと、HOP（Handai Okayamadai Processor）と名付けられたマイクロプ

* Katsuro INOUE
1956年8月生まれ
大阪大学大学院基礎工学研究科博士後期課程物理系専攻情報工学分野（1984年）
現在、南山大学　理工学部　ソフトウェア工学科　教授　工学博士
専門／ソフトウェア工学
TEL：052-832-3111
E-mail：inoue599@nanzan-u.ac.jp

ログラム方式の計算機のセルフアセンブラの開発で、Threaded Code 方式と呼ばれるサブルーチン呼び出し系列の記述を用いて開発し、卒論の締切までに無事開発を終えることができた。これが自分で作った実用的なプログラムの最初のものであった。このアセンブラを利用して HOP 上では、8080 エミュレーターが開発され、CP/M が移植されて CP/M 上でいろいろなソフトウェアが稼働するようになった。

ソフトウェア工学の研究の入り口

大学院の博士前期課程に進学し、研究テーマをどうするかということになった。当時の嵩研究室での大きな研究テーマは、代数的仕様記述言語 ASL を用いた形式的仕様記述によるプログラムの検証や段階的詳細化であった。筆者も ASL を用いた記述の詳細化に取組んだりもしたが、（筆者には）難しい割には実用性が高く感じられなかった。ちょうどその頃、Turing 賞受賞者の John Backus が受賞記念論文で FP という純関数型言語を提案しており、これが ASL の実行系として使えるのではないか、ということで、谷口健一先生の指導のもと、筆者が FP の実行系を作ることになった。FP は意味定義が式の書換え規則で定義されており、それに忠実に従った FP の式（プログラム）の書換え処理系をインタプリタとして実現した。このインタプリタによって FP プログラムの実行は可能になったが、式の書換え処理に時間とメモリを費やすのであまり実用的なプログラムを稼働させることはできなかった。とりあえずこのインタプリタの開発を修士論文としてまとめ博士後期課程に進学した。

後期課程のテーマとしては、引続き ASL の実行系の開発を選んだ。今度は ASL の部分言語として ASL/F という関数型言語を定義し、そのコンパイラ開発を関浩之さんと行った。このコンパイラでは既存の各種最適化を取り入れるとともに、必須な引数を事前判定し評価して渡すという最適化を考案し、効率的な実行が行われるように工夫した。その結果、最適化を施すことで、実行速度が一桁速くなるとともに、使用メモリ量も一桁減り、実用的な時間とメモリ量で ASL/F プログラムの実行が可能となった。見掛けは効率が悪そうなプログラムも、最適法をいろいろ工夫することで飛躍的に効率が向上することがわかった。この研究で、プログラム分析や最適化の威力を認識することとなり、その後の研究の方向性に大きな影響を与えた。

1984 年最適化の効果などを博士論文としてまとめ、学位を頂いた後、嵩先生の親友であるハワイ大学の W. Wesley Peterson 先生に誘って頂き、ハワイ大学の ICS 学科で助教として働くことになった。授業や研究の合間に、ウィンドサーフィンを習ったりアマチュア無線をしたりと楽しい思い出がたくさんある。

2 年間ハワイ大学に在籍した間に、関数型言語のゴミ集めの最適化手法の研究を行った。通常のゴミ集めでは、各セルがスタック等から到達可能でないことを調べて不要の判定を行うが、ある条件が成立する場合は、不要セルの到達性を調べる必要がなく、実行中にいきなり回収することができることを発見した。同様の研究は知られてなく、面白い結果だったので、ACM TOPLAS に投稿し、採録された。

1986 年 8 月、ハワイ大学から帰国後、大阪大学基礎工学部情報工学科の鳥居宏次先生の研究室で助手として働き始めた。研究室では、鳥居先生を中心として、菊野亨先生、松本健一さん、楠本真二さんらが、実証的なソフトウェア工学の研究を活発に行っていた。そこでソフトウェア開発工程をプログラム記述化するという Osterweil のプロセスプログラミングに着目し、ASL/F でソフトウェアプロセスを書き、実行する処理系の開発を飯田元さん、荻原剛志さんらと行った。この成果は、1989 年のソフトウェア工学国際会議 ICSE-11 Pittsburgh 大会で発表しデモンストレーションもする機会を得ることができた。ただ、ソフトウェアプロセスの研究は抽象的な記述の議論が中心で具体的な実開発プロセスとの結びつきが弱く、研究成果の普及は容易ではなかった。そのような理由もあり、徐々にソフトウェアプロセスの研究から遠ざかっていった。

その当時興味を覚えたものが（プログラム）スライスである。1981 年に Weiser が ICSE-5 で発表したプログラムの影響範囲を特定する技術で、その理論的な美しさと実用性に大変興味を惹かれた。さっそく学生らと新しいプログラムスライス法の研究に取り掛かり、再帰関数を含むプログラムのスライスや、軽量の動的情報を用いたスライス削減方法などの研究を行い、ICSE などで発表を行った。

スライスの研究は理論的に面白く、また、プログラミング言語やその要素ごとに新たに研究すべき課題があるので多くの研究者が取り組んでいたが、実用的なスライスシステムの開発・維持には、ポインタや配列の処理、関数呼び出しの取り扱い方など難しい問題があり、大学の研究としては継続する困難さも感じることが多かった。

コードクローン研究

1999年ごろ、大手SIer企業の人との雑談で、大規模で20年近く運用しているシステムの保守で困っている、という話を聞いた。プログラムの一部をコピーして他の場所で複写（ペースト）して利用しているが、その複写箇所を記録している台帳の管理が追いつかず、不完全なものになってしまった、何か自動的にコピー＆ペーストの箇所を調べるツールは無いか、ということだった。さっそく文献などを調べてみると、この問題は、コードクローン分析と呼ばれており、いくつかのプロトタイプが開発されているが、実用的なツールは無いということがわかった。そこで我々自身でコードクローン分析ツールを開発してみようとなった。これに興味を持って精力的に取り組んだのが、博士後期課程の学生だった神谷年洋さんであった。彼は文字列検索の本を参考に、短期間でツールの中核部分を作成し、コードクローンを分析して出力できるようにした。その結果は非常に面白く、より規模の大きなプログラムやいろいろな言語のプログラムを分析してみたくなり、技法を駆使してツールの効率化を図るとともに、高速でメモリを多く持つ高価なワークステーションを導入して研究を加速させた。そして、このツールをCCFinderと名づけてその詳細や分析結果を論文にまとめてIEEE Transactions on Software Engineering誌に2002年発表することができた。

CCFinderは実用的に使えそうだと分かったので、ドキュメントやメーリングリストを整備し、他の研究者や企業で簡単に利用できるようにした。ワークショップを開催してCCFinderの使い方やコードクローン分析手法の普及に努めた。このような活動の結果、多くの研究でCCFinderが利用されるようになり、先に述べた論文の被引用数はGoogle Scholarによると2022年7月1日現在で2,177となっており、発表から20年経過した今でもその数は増加している。また、ソフトウェア工学分野の研究論文で被引用数の大きな論文の第24位としてランクされた。そして、この研究のきっかけとなった会社を含め数多くの企業でCCFinderが導入され、システム分析や品質管理などに用いられるようになった。そのうちの10社ほどとは共同研究をすることができ、知見を深く共有することができた。

我々の研究室でもCCFinderの開発をきっかけにコードクローンに関する研究が一気に盛んになった。卒業論文、修士論文は言うに及ばず、博士論文でもコードクローンを主テーマにしたり、研究の一部で利用したりするようになった。また、オープンソースソフトウェア（OSS）に対するコードクローン分析の研究をきっかけに、ソフトウェアレポジトリの採掘（Mining Software Repositories MSR）の研究に興味を持つようになり、MSR研究のコミュニティの立ち上げに加わるようになった。さらに、コードクローンの研究から派生して、OSSのライセンス／著作権分析の研究やソフトウェアエコシステムの研究を立ち上げることができた。

このように大きく研究を発展させることができた理由としては、コードクローンという研究対象がコピー＆ペーストという身近にある現象であり実務家を含めて誰でも直感的に理解しやすい、コードクローンの検出にはいろいろなアプローチがあり研究者の創意工夫が入れやすい、PCやワークステーションの性能向上やインターネット普及によって大規模な分析が容易に行えるようになったこと、などが考えられるが、優秀な共同研究者や学生に巡り合えたことも大きい。

ソフトウェア工学の教育について

ソフトウェア工学に関する教育として最初に関わったのは、大阪大学基礎工学部情報科学科3年生に対するプログラム設計の授業である。この授業では、それまでプログラミングやプログラム言語などの授業だけ受けてきた受講生に対して、要求獲得や設計など、開発の上流工程の認識と経験を深めるためのものである。しかし上流工程には決まった方法や正解があるわけでもないので納得感に乏しく、非常に教え難い。また適切な教科書も見当たらなかっ

た。結局、教科書を自分で書いていくつかの基本的な手法を教えるとともに、グループワークの議論を通じて理解を促すことを行った。この授業は、保守など下流工程を専門としている筆者としてはとても勉強になった。一方、上流工程の教育法はもう少し深く研究される必要があると感じた。例えば設計における必須な教育項目、それを伝えるための表現形式、一般性を失わずに具体的な理解を促進する方法、多人数・遠隔での教育法などは、研究テーマとしても、また、実践的にも重要であろう。また、定番となる教科書も強く望まれる。

筆者は、文科省の受託事業である「先導的ITスペシャリスト育成推進事業」や「情報技術人材育成のための実践教育ネットワーク形成事業（enPiT)」、「成長分野を支える情報技術人材の育成拠点の形成(enPiT2)」の運営に深く関わってきた。これらの事業では、複数の大学で強く連携して実践的な情報技術教育を行い、高い好評を受けることができた。これは関係していただいた各大学の教職員の皆様の高い情熱と努力の結果である。また、このような教育ネットワークの運営には個人個人の繋がりが重要で、（コロナ禍以前は）シンポジウムや運営委員会を通じて多くの人と交流し繋がることができたのもよかった。

組込み適塾は、関西経済連合会を母体とする組込みシステム産業推進機構が毎年開催する社会人向け人材育成プログラムで、筆者は2008年度の設立からそのカリキュラム開発や運営に関係してきた。このプログラムでは、関連企業から参加する運営委員が、毎年、実施した講座の評価や見直しを行っており、時代に必要とされる新しい講座も次々と開講され、高い評価を得ている。このように多数の企業がエフォートを出し合って教育プログラムを長期にわたって開発、運営しているのはあまり例がなく、今後の社会人のリカレント、リスキリング教育のモデルとなろう。

おわりに

筆者がソフトウェアに触れるきっかけとともに、関係したソフトウェア工学に関する研究や教育を振り返ってみた。研究に関しては、比較的うまくいった事例を紹介したが、うまくいかなかった事例も多々あった。例えばバイオブームの時期にゲノム情報学との融合研究を目論んでバイオ技術を少し勉強したが、結局、素人が参画して実施できる面白いテーマを見つけることができなかった。コードクローン検出アルゴリズムは、ゲノム情報学で深く研究されているが、それぞれの分野での検出対象や目標には違いがあり、そのまま適用することは難しかった。

ソフトウェア工学の研究に関しては、Σプロジェクト、e-Societyなど過去比較的大きな投資が政府から施されていたように思うが、近年はあまり見かけない。一方、最近は量子計算機や機械学習分野には大規模な投資が行われている。直接の成果が可視化し難いソフトウェア工学やソフトウェアに、大規模な投資することは容易ではないのであろう。しかし、投資が無くなると、人材を継続的に育成することが困難になり、研究のみならず教育のレベル低下にもつながるので、ソフトウェア関連の研究投資を行ってもらいたい。

教育に関しては、ソフトウェア開発人材の逼迫から、比較的いろいろな投資が行われてきている。今はもっぱらDX人材、AI人材という名目の高等教育や社会人教育に投資されているように見える。また、小学生に対するプログラミング教育も大きな投資であり、中学、高校の情報教育の実質化や大学の情報入試とうまく連携すると、大学新入生の事前知識は大きく変わってくると思われる。この変革を歓迎するとともに引き続き注視していきたい。

筆者のこれまでのいろいろな活動に関わり、支援していただいた方々は非常に多くこの場で全員を紹介することはできなかったが、皆様に深く感謝いたします。