SES2012 基調講演

ソフトウェア・研究・人生

情報・システム研究機構 統計数理研究所 丸山宏
email: maruyama@acm.org, Twitter: @maruyama

コンピュータ科学は面白い学問である。 科学と言いながらあまり科学らしくない。ソフトウェアには工学があるが、結局ソフトウェア開発者のやっていることはアートだ。 他人には（特に上司には）なかなか理解してもらえない。我々はなぜそんなコンピュータ科学に魅了されるのだろうか。 この講演では、私自身の人生を振り返り、それを考えてみたい。

１．プログラミングに魅了されて

1.1. 動いた時の喜び・動かす喜び

私は同年代の多くの理系少年と同様、小さい頃はラジオいじりが好きであった。小学校６年生のときにアマチュア無線の免許を取り、真空管の受信機を組み立てたりしていた。だから、大人になったら電子工学の専門家になるのだろうなと思っていた。

高校のときに、親友の一人から「面白いからプログラムを書いてみないか」と誘われた。私の通っていた高校には、数学研究部があり、中古の OKITAC4300C というミニコンがあった。8K 語のコアメモリを持ち、FORTRAN JIS3000 が動いていたものである。そこで最初に書いたプログラムは対数表を印字するものであったが、なんと、最初のコンパイルでエラーもなく、動いてしまったのである。これが私がプログラミングにハマるきっかけとなった。

その後、Intel の 8080 のデータシートなどを取り寄せて、自分のコンピュータを持つことを夢見ていた。大学生になって、初めて購入したコンピュータが、日立のワンボードコンピュータ、H68/TR というものである。6800 プロセッサ、1KB のメモリ、入出力には 14 桁 7 セグメントの表示器と電卓風のキーボードを使っていた。このコンピュータには、後にメモリや、ビデオインターフェース、デジタルカセットインターフェースなどを増設し、多くのプログラムを書いた。使った言語は主にアセンブラであったが、ビル・ゲイツの書いた BASIC なども走らせていた。

大学院に入るころには、マイコンショップでアルバイトをしてお金を溜め、当時あこがれの的であった SOL-20 というコンピュータを購入した。インテル 8080 プロセッサに、5.25 インチのフロッピーディスクドライブが外部記憶だった。このシステムでは、UCSD Pascal

の他に、Lisp も動いていて、その上で Prolog の処理系を実装したりした。

ずっと時代は下って、2006 年に IBM 東京基礎研究所の所長になったときには、ソフトウェアだけでなくハードウェアもいじれることを部下に示すために、N ゲージ鉄道模型のデジタル制御システムを作って会社のイノベーション・デーでデモした。これは、PIC マイクロプロセッサを車両に載せて、線路の極性を変えることで送られてくる信号をデコードして制御するものである。

このように自分で何かを作り、それが思い通りに動いた時には大きな喜びを得るものだ。2007 年に世界学生プログラミングコンテスト（ICPC）を東京でホストしたとき、私はそのオープニングスピーチで、プログラミングの楽しさを以下のように表現した。「プログラミングとは自分が自分の小さな世界の神になったようなものです。自分の指示によってその市民たちが動きまわり、生活する。その世界が繁栄するかどうかは、自分次第なのです。」

1.2. 人間臭さ

1983 年に就職したときには、同期に 25 人の研究者がいたが、それぞれ異なるプログラミングスタイルを持っていて、大変面白く思った。

現在富士フイルムソフトウェアに勤務する小坂一也氏は、メインフレーム用のスクリプト言語 REXX で、オブジェクト指向風のプログラミングをしていた。当時 Smalltalk-80 が話題になったばかりで、誰もオブジェクト指向プログラミングができる環境になかったが、彼は普通のプログラミング言語の名前付けなどのコンベンションだけで、オブジェクト指向プログラミングを実践していたのである。

「プログラムは動けば良い」と割り切っていたのは、今は電気通信大学に勤務する沼尾雅之氏である。彼のコードはほとんどコメントがなく、他人には読みにくいものであったが、動くコードを作るスピードは目を見張るものがあった。

対照的なのが、マンチェスター大学で博士号を取って入社した細川馨氏で、彼の書くソフトウェアは構造、アルゴリズム、コード字配りにいたるまで大変に美しいものであった。ただし、私は彼の作ったソフトウェアを使った記憶がない。綺麗なコードは書くが、人に使ってもらうことにあまり興味はなかったのかもしれない。

私が XML の仕事をしはじめたときの最初の部下の一人が、今は Google に勤める田村健人氏である。彼は大変信頼出来るプログラマであった。ソフトウェアの開発に関して彼が「やります」と言ったらそれはできたも同然と思ってよかった。

一方、現在は弁理士の仕事をしている今村剛氏は、仕事は速かったが「とりあえず、できました」という報告は怪しかった。最初のケースは通るのだが、それ以外のケースではクラッシュする、などということがよくあったからである。彼に対しては、十分なコンティンジェンシーの時間をプランしておくことが必要だった。

1.3. 気持ちのよいソフトウェア

IBM 東京基礎研究所の中で、私が最も尊敬していたコンピュータ・サイエンティストの一人が小野寺民也氏である（すくなくとも彼が素面のときには)。彼は C++が広く普及する以前に、C 言語をオブジェクト指向に拡張した COB (C with Object)という言語を設計した。Java のようにオブジェクト指向でガーベジコレクションを持つ言語であったが、同時にポインタなど C 言語ネイティブな機能も使えて小回りの効く言語だった。さらに、彼自身が実装した効率の良い、安定した処理系（コンパイラ＋ランタイムライブラリ）があった。これは私にとっては大変「気持ちのよい」ソフトウェアであり、私はこの上で後述する日本語構文解析システムなど多くのプログラムを書いた。

残念ながらこのような「気持ちの良さ」は、自分で使ったことのないマネジメントにはなかなか伝わらない。大艦巨砲主義の PL/I が幅を利かせていた IBM でこのような言語が認められるのは難しかったのだろう。

ソフトウェアの良さというのはなかなか上司にはわかってもらえない。わかってもらえなくても、自分はプロとして良いソフトウェアを開発するべきだ、と説いているのが Robert Martin の"Clean Coder"[1]である。これは、プログラマーがプロとして守るべき規範(code of conduct)とは何か、に関する本だ。この中には、ボスが聞いたら卒倒しそうなことも書かれている。「コードを見たら書き換えなさい」というのもその一つだ。ソースを見たら、常にそのコードをより美しくして、チェックインしなさい、コードの修正を恐れてはいけない、コードの修正を恐れるとそれは死んだコードになり、いざというときに怖くて触れないコードになってしまうからだ、ということなのである。もちろん、このようなプラクティスが使える前提としては、テスト駆動開発とか、連続ビルドなどがあるのだが、それでも「コードを見たら書き換える」というルールを自分に課することで、プロとしての自分を磨いていくことができるのだろう。

プログラミングはアートである。アートには、工数やコストで管理するマネジメントスタイルは似合わない。残念ながら現状ではそのようなマネジメントがほとんどであるようだ。上司に理解されなくても、誇りを持って、良いソフトウェアを作る、そういうプログラマーでありたいものだ。

２．研究

2.1. コンピュータ・サイエンスの醍醐味

コンピュータ・サイエンスというのは、サイエンスと言いながら研究の対象となるのはコンピュータという人工物であり、その意味では自然科学ではない。真理を探求するというよりも、計算の原理に基いて新たにどういう価値を創造していくのか、という面にその面白さがあるのだろうと思う。

ある原理・理論を作ったならばそれをソフトウェアとして実装して価値に結びつける道のりは、コンピュータ・サイエンスの世界ではうまくやれば非常に短いものになる。私が

大きく影響を受けた教科書の一つは、Aho らによる Compiler[2]であるが、この本には、形式言語理論という原理と、その原理に基いて作られた実用的ツールである lex,yacc などが紹介されている。

観測された自然をうまく説明できなければならない物理法則とは違って、コンピュータ・サイエンスの原理はゼロから作ることができる。自分が神になって原理から組み立てる世界だ。その世界の上で、自分の原理に従って作ったプログラムが、実際の現実世界の問題解決に役立つ、それはなんと素晴らしいことだろうか。

2.2. Research That Matters

私は同じことがやりたかった。機械翻訳の日本語処理のために、制約依存文法という形式的な理論を構築した。そして、この理論に基いて効率的な構文解析を行うアルゴリズムを設計し、前述の COB でその構文解析器を実装した。さらにこの日本語構文解析システムを日英機械翻訳システムに組み込んで、あるお客様のもとへ持ち込んだ。1990 年のことである。

このお客様は自動車会社で、日本の研究所で設計した図面を、米国の工場で生産するために英語に翻訳する必要があった。私は毎日翻訳を行なっている翻訳者の隣に席をもらい、半年間にわたって、翻訳者の知識を機械に教え込む、という作業を行なった。残念ながらしかし、このシステムは実際には使われなかった。翻訳精度が 100%にならないからには、人間の翻訳者を減らすことができないこと、また機械翻訳システムを使うためには、エンジニアが手書きで書いた日本語を新たにコンピュータに打ち込む作業が発生することなどが原因だった。

1996 年から 1997 年には米国ソフトウェア事業へ出向し、当時爆発的に起こりつつあったインターネット関連の技術評価を行った。そこで出会ったのが XML である。XML が企業間のデータ交換、いわゆる B2B に使われるという予測を聞き、XML の上でのセキュリティ、また XML データの交換のためのプロトコルが必要と考え、多くの提案を行なった。その結果、私たちのチームは、IBM のグローバルチームの中で一定の地位を占めることができ、XML や Web サービスの標準化や技術開発をすすめることができた。自分たちがやっていることが、確かに世の中に出ていってインパクトを与えている、という手応えがあった。

これらの経験から、特に企業における研究は世の中にインパクトを与えるものであるべきだと考え、2006 年に東京基礎研究所の所長になったときには、それを"ResearchThat Matters"[3]と表現して、研究所の標語にした。この言葉は、IBM 社員の価値理念の一つである "Innovation That Matters" をもじったものであるが、考え方としては私のオリジナルではなく、たとえば米国の科学技術投資のあり方を議論した"パスツールの象限"[4]にも見られる考え方である。

2.3. 問題探しの旅

Matter する研究をするには何をしたらよいのだろうか。一番大事なことは、良い問題を見つけることである。CPU の金出先生の「素人のように考え、玄人として実行する」[5]は、簡潔な、わかりやすい問題設定の重要性を教えてくれる。あるとき、金出先生の講演の後の懇親会で、問題をどのように見つけるか、アドバイスを聞いたところ、「まず、どういうデモをするかを考えなさい」とおっしゃった。ソフトウェアの研究においては、多くの場合その価値を伝えるのに、デモの力が絶大である。これは心に留めておきたいと思う。

2003 年から 2004 年にかけて、私はコンサルティング部門に出向した。研究部門がビジネスに貢献しなければならない、というのが主な目的であったが、私にとっては、お客様の現場で何が起きているのかを実際にこの目で見ることのできる、貴重な経験になった。これ以降、いろいろな技術、研究プロジェクトをお客様の視点で見る力がついたと思う。

研究者は研究所の中にとどまっていてはいけない。「マターする」研究を行うには、現場に出て、耳を傾けて、初めて良い問題が見つかるのだ。

2.4. パラダイムシフト

しかし、単に目先の問題だけを追っていてもいけない。科学技術の世界では、時折、全く新しいアイディアが生まれ、世の中を大きく変えていく。ノーベル物理学賞を受賞した Albert Michelson は 1894 年に「重要な物理法則はすべて発見された。今後は物理学において重要な発見がなされることはないだろう」と述べたそうだ。彼だけでなく、多くの物理学者が当時そのように考えていたそうである。もちろんこれは、プランクの量子論やアインシュタインの一般相対性理論の生まれる直前のことである。多くの人がそんなことは不可能だと思っていても、パラダイムシフトは起きるのだ。コンピュータ・サイエンスは比較的若い学問であるためか、動きが速い。コンピューティングにおけるパラダイムシフトは、S/360 の汎用機アーキテクチャ、マイクロプロセッサによるパーソナルコンピュータ、インターネットと Web による 3-Tier モデルなど、次々に起きてきた。現在はクラウドコンピューティングと、スマートフォンが注目を浴びている。それでは、次に来る重要なパラダイムシフトは何だろうか。

Preferred Infrastructure の岡野原大輔氏は、データの大部分がネットワークのエッジ部分で格納・処理される「エッジ・ヘビー・データ」の時代が来ると予測している[6]。これにはまだ多くの技術的チャレンジがあるが、大きな流れになるかもしれない。

研究者として我々は、このような大きなパラダイムシフトを敏感に予知し、願わくばそれをドライブしていく存在でありたい。そのためには、常に新しい動きを察知し、アイディアを交換して刺激しあうことが大切だと思う。

３．人生について

コンピューティングにおけるパラダイムシフトと同様に、人生においても変節点が訪れる。

私の場合それは、2006 年 2 月 5 日、日曜日の早朝であった。当時の IBM Research のディレクターである Paul Horn が自宅に電話してきて、東京基礎研究所の所長にならないか、と言ったのである。驚いたが、私の人生のポリシーの一つは「チャンスは与えられたらつかまえるものだ」というものである。だから私はイエスと言ったのだった。結果的にそれが正しい判断だったかは今でもわからない。所長になって多くのことを学べた反面、ここでイエスと言っていなければ今でも IBM に残っていただろうに、と思うこともある。もちろん、イエスと言ってしまった以上、前に進む以外に道はない。人生における大きなパラダイムシフトの瞬間であった。

それまでの私のキャリアでは、時々ラインマネージャーになったこともあったが、基本的にはエンジニアだった。ラインマネジメントを専門にやるようになって、興味の対象が技術から「人」にシフトしたのを感じた。

すべてが（バグも含めて）自分の指示した通りに動くプログラミングと違って、リーダーシップとは生身の人間を動かすことである。しかし、人は必ずしも指示した通りに動くわけではない。IBM での 26 年間に受けた研修の中で、一番強いインパクトを受けたのは、初めて出席した役員研修、コンサルタントの大久保寛司さんによるセミナーであった。その中で特に印象に残っているのは、「人は説得されて動くのではない、納得するから動くのだ」という言葉である[7]。

実は社会の変化とともに、リーダーシップのスタイルも変わりつつあるのをご存知だろうか。それを端的に表しているのが、最近特に IT 業界に多い、女性の CEO の出現である。HP の Meg Whitman、IBM の Genni Rommety、Xerox の Ursula Burns、そしてつい最近 Yahoo!の CEO になった Merissa Mayer などである。20 世紀における大量生産型のビジネスにおいては、強いリーダーシップの下にトップダウンに命令を伝えるスタイルのマネジメントが良いとされていた。プロフェッショナルがチームを作って問題を解いていく 21 世紀型の知識社会では、よりエンパワメントに比重をおいたマネジメントが求められる。ソフトウェアの開発チームにおいても、同じことが言えるのではないだろうか。

エンパワメントを考える際に、避けては通れないのがダイバーシティの考え方である。ダイバーシティとは、男女共同参画のことだけではない。外資系も含めて、日本の会社では「日本人だけの会議」を好む傾向にあるようだ。しかしそれではもったいない。様々なバックグラウンドの人々が集まっていてこそ、様々な角度から問題を検討することができ、新しいイノベーションが起きるのだと思う。ダイバーシティの重要性を頭で理解しても実践するのは難しい、というのはよくわかる。私がいつも気をつけているのは、「自分には理解できないものをひとまずは『優れたものだ』と仮定を置いて考えること」である[8]。

ソフトウェア開発にしろ、組織マネジメントにしろ、結局相手にしているのは人だ。自分の周りの人々をリスペクトできるかどうか、それが問われているのだと思う。

4. 終わりに

私はコンピュータが好きだ。同時に人も好きなのだと思う。私の Twitter アカウントの自己紹介には、"Computer Scientist and Friend of Many"と書いてある。それを誇りにして生きていきたい。

参考文献

1. Robert C. Martin, *The Clean Coder: A Code of Conduct for Professional Programmers,* ISBN-13: 978-0137081073, Prentice Hall, 2011.
2. Alfred Aho, Ravi Sethi, Jeffery Ullman, *Compilers: Principles, Techniques, and Tools*, ISBN-13: 978-0201100884, Addison-Wesley, 1986.
3. 丸山宏, 企業の研究者を目指す皆さんへ – Research That Matters, ISBN-13: 978-4764903821, 近代科学社, 2009.
4. Donald Stokes, *Pasteurs Quadrant: Basic Science and Technological Innovation,* ISBN-13: 978-0815781776, Brookings Inst Pr, 1997.
5. 金出武雄, 素人のように考え、玄人として実行する―問題解決のメタ技術, ISBN-13: 978-4569624570, PHP 研究所, 2003.
6. 丸山宏, 岡野原大輔, “Edge-Heavy Data: CPS・ビッグデータ・クラウド・スマホがもたらす次世代アーキテクチャ,” GICTF 総会 特別講演, http://www.gictf.jp/doc/20120709GICTF.pdf, 2012.
7. 大久保寛司, 「自分が変われば組織も変わる―コミュニケーション上手になる」, ISBN-13: 978-4761261023, かんき出版, 2003.
8. 丸山宏, “ダイバーシティとは何か,” 大学共同利用機関法人 情報・システム研究機構 男女共同参画コラム, http://www.rois.ac.jp/danjyo/essay_02.html, 2011.