ソフトウェア工学 120－7
（1998. 7. 23）

# 第20回ソフトウェア工学国際会議報告

井上 克郎[1]

大阪大学大学院基礎工学研究科

内容梗概

第20回ソフトウェア工学国際会議は、1998年4月20日から26日の間、京都で開催された。本稿では、この会議を通じて見たソフトウェア工学の研究の現状について私見を述べる。

# Report of 20th International Conference on Software Engineering

Katsuro Inoue

Graduate School of Engineering Science
Osaka University

Abstract

Twentyth International Conference on Software Engineering has been held in Kyoto, Japan, from April 20 through 26, '98. In this article, we will briefly summarize this conference.

## 1 はじめに

第２０回ソフトウェア工学国際会議(20th International Conference on Software Engineering, ICSE98)が、16年ぶりに日本で開催された。この会議には、国内、国外を合わせて千名ほどの参加者があり、非常に盛大なエベントになった。この場をお借りして、この会議の準備、実行、そして事後処理などに関わった方々、御支援頂いた方々に感謝したい。また、実際に会議に参加して頂いた方々に深謝する。

筆者は、この会議の運営に深く携わっており、舞台裏の数々の事柄を経験した。それらについては情報処理７月号で詳しく述べられているので参照願いたい。また、会議で実際行なわれた発表、展示の詳細などについては、NTTデータの佐藤、端山の両氏が、新鮮な目で見た印象をソフトウェア科学の近月号に掲載する予定であるので参照願いたい。

本稿では、この会議を通じて得たソフトウェア工学の研究の現状や、それに付随するいろいろなもの

[1]〒560-8531 大阪府豊中市待兼山町１－３
Graduate School of Engineering Science,
Osaka University,
Toyonaka, Osaka 560-8531, JAPAN
06-850-6570 (Ph.) 06-850-6574 (Fax)
inoue@ics.es.osaka-u.ac.jp

についての私見を述べる。

## 2 技術発表

技術論文や経験報告、デモンストレーション、ポスター展示、基調講演、そしてICSE98-EXPOなど、数多くの発表、展示が行なわれた。このなかから幾つか印象的なものについて述べる。

### 2.1 表形式プログラムのテスト技法

"What You See Is What You Test: A Methodology for Testing Form-Based Visual Programs", Gregg Rothermel, Lixin Li, Christopher DuPuis, Margaret Burnett (Oregon State Univ.), *Proc. of 20th International Conference on Software Engineering*, pp. 198–207, Kyoto, Japan, April 1998.

この論文は、プログラム委員会で高い評価を受けた、良く書かれた論文である。Excelを始めとする表形式プログラムは通常数多くのフォールトが含まれており、それを効率良く発見するためのテスト手法について述べている。

著者らは、まず研究用の表形式言語Forms/3を用い、そのプログラムの各セルの間の関係をCRG(Cell Relation Graph)で表現する。通常のプログラムは、その文の間の依存関係には制御依存、データ依存の2種類に分けられるが、表形式言語の場合は、データ依存関係のみがセル間に存在する。ここでは、プログラムの中の全てのデータ依存関係が使われるようにテストデータを与えてテストすることを目標とする。これを実現するために、与えられた表形式プログラムの全てのデータ依存関係を（インクリメンタルに）収集する。収集した依存関係が与えられたテストデータで使われているか調べる。

次に、実際に、9つの例プログラム（大きさは式の数で１６～６０程度）それぞれに１０程度のフォールトを混入したバージョンを作り、全てのデータ依存関係を調べるテストデータを与えた結果、平均８１％のフォールトを発見することができた。

この論文の良いところは、(1)問題が身近であるにもかかわらず、あまりやられておらず新鮮なテーマである、(2)問題が比較的うまくきれいに形式化されている、(3)解法の手順が明解、(4)実際にその手法を適用してみて効果を実験的に調べている、などであろうか。この(2),(3)に関しては、通常の手続き的言語より、表形式言語の方が難しい問題がなく、形式化、アルゴリズム化が容易であることが推察される。また、(4)の実験結果が必ずしも強力であるとは言えないが、十分、現実で使えそうな印象を与えている。望むべきは、この手法を採り入れたテスト環境ツールの概要などを議論し、現実に直結する指針を与えて欲しかった。

### 2.2 D. Harelの招待講演

"On Modeling and Analyzing System Behavior: Myths, Facts and Challengers", David Harel (Weizmann Institute/ i-Logix), *Proc. of 20th International Conference on Software Engineering Vol.2*, pp. 8–10, Kyoto, Japan, April 1998.

最近、ICSEでは、１０年前に行なわれたICSEの発表論文の中から、１０年間を経て最も影響を与えた論文を一つ選び、その著者を表彰している。今回は、第１０回シンガポール大会で発表された論文の中から、David Harelらの書いた

"STATEMATE: A Working Environment for the Development of Complex Reactive Systems", D. Harel., H. Lachover, A .Naamad, A. Pnueli, M. Politi, R. Sherman, and A. Shtil-Trauring, *Proc. of 10th International Conference on Software Engineering*, Singapore, 1988.

が選ばれた。

本来、この授賞講演は、本会議２日目の午後に行なわれる予定であったが、その日の午前に予定されていた、Sun MicrosystemsのJames GoslingのJavaに関する基調講演が直前にキャンセルされたために、急きょ、Harelの講演を基調講演として、午前中に行なった。

一部、Javaの最新動向を期待していた聴衆には気の毒ではあったが、しかし、Harelの話は、ソフトウェアを本当にうまく作るために、いろいろ考えさせられる講演であった。

彼は、システムを階層的な状態図としてモデル化するStatechartとそれを実行するStatemate、Rhapsodyなどを開発してきている。彼は、複雑なシステムを設計するためには、視覚的で、簡潔で理解やすく、構造化できて、ちゃんと実行できて、動的な解析ができ、実際のシステムの一部になるコードを出力できるようなモデルが必要である、と主張している。

彼のStatechartはそれらを満たすもので、その考えは、SA/SDやOOのモデル化の中の動的な振舞いの定義にも使われてきている。さらにUMLの実行カーネルにはRhapsodyの考えが大幅に採り入れられている。また、今後、巨大になったモデルの一貫性を保証したり検証するための技術が必要であることも示唆していた。

Statechartから出発している非常に単純なモデルが、形を変え、いろいろなモデルの動的な振舞いの定義に用いられていることに驚かされた。ソフトウェアシステムのモデル法がいろいろ提案されてきたが、その動的な振舞い定義が、状態遷移だけでいいのか、という素朴な疑問もある。ひょっとして、複雑な意味定義を持つ動作定義方法は、抽象モデルには向かないのであろうか。

彼のアプローチは、簡単ではあるが、形式化がしっかり行なえ、実行、シミュレーション、コード生成、他の意味定義との親和性など、非常にバランスがいいので、現実的な動的なモデルなのであろう。

今後も状態遷移ベースの動的定義は、システムモデルの中心で、彼の研究とその製品はますます重要になると思うが、研究者としては、もう少し、状態遷移より機能に富んだ動的振舞いの定義法が出てきて、実用的に使われることを期待したい。

## 2.3 Greenの基調講演

"Automated Assistance for Software Design", Cordell Green (Kestrel Institute), *Proc. of 20th International Conference on Software Engineering Vol.2*, p.6, Kyoto, Japan, April 1998.

本会議3日目の基調講演者のCordell Greenは、Kestrel Instituteというシリコンバレーにあるソフトウェア解析の研究を行なっている研究所の所長である。彼は、長年、推論機構、証明、プログラムの意味、構成について研究してきている。その途中で、これらの推論、構成を行なうシステムREFINEを作り、それを販売する会社Reasoning社を作った。現在ここで作成しているCobolプログラム解析ツールは、巨大なプログラムを高速で解析できる。例えば2000年問題のための診断を他のツールに比べて高速かつ正確にできるようになった。

しかし、彼の興味は、Reasoning社の事業から離れて、Javaプログラムの検証系の合成、仕様上の特定の性質を保存したプログラムの進化、などの研究的な側面に向いていたので、経営を人に任せ、Kestrel研究所を設立した。

現在、この研究所では、上記のような研究の他、いろいろな人が作った仕様の交換を促進するSpecbrokerageという機構を作ったり、Reasoning社が販売している解析ツール開発環境のアカデミック使用を斡旋したりという活動も行なっている。

彼は、長年に渡って、このような研究を続けてきており、また、それを使えるようにするために、会社を作って、販売も行なっている。会社を設立するのは非常に簡単で、しっかりした企画書さえ作れば、ほぼ会社はできたようなものだと言っていたのは面白かった。また、長年、しつこく一貫したテーマをやってきて、製品化まで持っていく姿勢は、非常に参考になった。

多くの参加者は、彼の話が彼個人の研究／会社に関するものに限定されており、基調講演としては少し物足りないように感じていたが、筆者個人としては面白かった。

## 2.4 発表・展示の傾向

上記のような招待講演、基調講演を見ると、昔のソフトウェア工学の会合のように見えるかも知れない。しかし、発表や展示されている内容を見ると、現在のソフトウェア開発の方向を見ることができる。おおまかにそれを分類すると次のようになるであろう。

- オブジェクト指向技術の開花
- ネットワークを基礎とした分散システム技術の発展
- 新しいソフトウェア開発パラダイムの模索

オブジェクト指向技術は、とりたててそれを強調する時代は終り、実際にそれを使って、どれだけ良

かったかを具体的な事例やデータで示す必要が迫られていよう。オブジェクト指向に関する理論的な研究もまだ、続いているが、事例研究報告で現実の経験が数多く発表されている。

ネットワークを前提としたシステムの開発、運用が当たり前になってきており、特にそれを意識せずに、使用される場合が多い。移動コードの開発などに関する問題のように、それを積極的に研究対象としている発表もある。

オブジェクト指向技術やネットワークなど開発に関わる基盤が、過去のものと変わりつつある中、どのような開発モデルやプロセス、そしてその管理法を考えればよいのか、いろいろ模索するような研究があった。今後の発展が期待される。

# 3 ソフトウェア工学での論文の採録

今回の会議では、通常の技術論文の他、事例研究報告、ポスター・デモなどいろいろな形式で発表の場が用意されている。技術論文は４１／２０９、事例研究報告は２３／５１の採択率である。

技術論文は、過去、１／１０ぐらいから１／４ぐらいまでの間の採択率である。一方、事例研究報告は、昨年の大会から設けられたもので、比較的採択率が高い。企業での現場の経験を基にして、ソフトウェア工学の見地からそれを整理することによって、採録される確率が上がろう。技術論文、事例研究論文とも同じようにプロシーディングスに掲載され、配布され、多くの人に読まれると思われる。多くの現場の人が、少しまとめ方を工夫するなり、また、大学人と共同で整理するなりして、多くの事例研究報告が投稿されることを期待する。

事例研究論文については、まだ、歴史が浅いので、以下、技術論文についてのみ述べる。

## 3.1 論文の構成

技術論文が採録されるためには、論文のしっかりした技術的な背景のみならず、読んで理解しやすい表現技術が必要であろう。

数多くの論文が、何か面白そうなことをしているのだが、何が新しい考えなのか、わからない、などという査読委員のコメントで、不採録とされている。ポイントを整理して、分かりやすく表現することが要求されている。査読委員は、通常、１月ほどの時間で、２０～３０編ぐらいの論文を読んでそれぞれについてコメントを返す作業を強いられる。丁寧に細かい論旨のつながりをフォローしてもらって論文が面白い、ということを理解してもらうことは、期待しない方が良い。具体的、単刀直入に主義主張を表記し、それを現実のデータでサポートする必要があろう。

また、論文の表現、というのは、技術的な価値と関係の無いように見られるかも知れないが、特にソフトウェア工学の場合は、他の研究と比べて何が新しくて、どこが面白くて、というのは、一つの重要なポイントである。これがうまく書けなくては、採録はおぼつかない。

一方、内容が薄い論文を、うまく表現だけで中身があるように見せる、という技もあまり役立たない。第一線で活躍する査読者が通常３名おり、３人ともだませる、というのはなかなかない。しっかりした技術的な裏付けがあってこそ、うまい表現が生きる。

## 3.2 フィギュアスケートとスピードスケート

ソフトウェア工学の論文には、通常その価値を計る絶対的な基準がない。次章に述べるように、プログラム委員の間の議論を通じて、その論文の価値が決定される。

フィギュアスケートの評価は、複数の審査員の主観的な判断によるが、ソフトウェア工学の論文の評価も似ている。技術的な価値をうまく表現できているか、が問われている。評価基準も一応の指針があろうが、絶対的なものではなく、審査員の個々のばらつきや、時代による変化などが当然ある。いきなり無名の新人が自分なりの評価基準を主張し、自分の演技の優秀さを主張したとしても、すんなり受け入れられない時が大半であろう。

一方、スピードスケートの評価は、単に計測した時間だけである。これは、例えば新しいアルゴリズムの論文の評価に似ていよう。他のライバルのアルゴリズムに比べ、どれぐらい計算時間、使用資源を小さく出来たかのみが勝負になる。この場合、評価尺度がはっきりしており、それに沿った新しいアルゴリズムを提案できれば、無名の新人がいきなり優

勝することも可能である。

このように、ソフトウェア工学の論文は、積み重ねが大切で、「一発あてる」、ということが難しい。しかし、評価基準が固定されていないという自由さは、世の中の変化に沿ってソフトウェア技術が変わらなければならない最近の状況には、必須である。今後もソフトウェア工学は、いろいろな評価基準を飲みこんでいくであろう。

### 3.3 プログラム委員会

技術論文のプログラム委員会は、通常、プログラム委員（ICSEでは、伝統的にプログラム委員自身が査読者）全員が一同に会して、それぞれの論文について採録か不採録の決定をする。今回も９７年１１月にサンタバーバラで約４０名の委員が集まって、２日に渡って議論を行なった。この会議では、「できるだけ沢山の論文を採録して、会議の参加者を増やしたい」、という会議主催者の意向などとは全く無視されて、プログラム委員の自己主張の場となった。一つの論文を査読する３人の意見がおおむね同じようなものになる場合が３分の２、意見が分かれるのが残り３分の１ぐらいであろうか。この意見が分かれたものを論戦でどちらかに収束させる。多くの場合、その他のメンバーのコメントや、その場でざっと論文を読む、という方法で決着がつく。

採録された技術論文のうち、査読者３名全員が採録すべき、とした論文はわずかで、それらは、ほぼ議論なしに採録された。不採録、とか、不採録に反対しない、などという査読者がいる場合、それを覆すのは容易ではない。論戦も疲れてくると、論文を通すべく主張する、というのは困難になってくる。皆、楽して早く会議を終らせたい、という心理が働くのであろうか、順次審議される各々の論文の採択率が、会議の始めの方に比べ後ろの方では悪いような気がする。ある程度、論文の採録には運が働くと考えた方が良い。

### 3.4 ICSEの論文の価値

このような激烈な査読者の間の議論をくぐり抜けてきた論文は、さぞかし素晴らしい内容なのか、と思われるかも知れないが、がっかりする場合も多い。なぜこの論文が、と思うようなものも採録されていたりする。しかし、多くのものは採録の基準（新規性、技術的完全性、可読性、…）は満たされており、読んで損はしない。

ICSEでの採択論文されることに対しては、通常、高い評価を受けることが多い（日本国内での評価は固まっていないかもしれないが）。欧米では、研究の成果としてICSEで発表できることは、次の研究資金を得たり昇進したりするための基準になることがあり、研究者にとっては死活問題である。そのために、すでに著名になっている研究者も、必死になってICSEの論文書きをやっており、落されると激しく怒ったり落胆したりしている。

実用的な価値から見ると、ICSEの論文は、そのまま、現場で役に立つものはあまりない。前述のように、ICSEで採録されている論文は、技術的な側面のみならず、表現でも洗練されている。そのような論文は、現場で出てきた問題の解法を示唆しているが、そのアプローチをかなり抽象化して議論しているのが普通で、実際それをどのように現場で適用したらよいかについては、あまり議論していない。

そのような抽象化ゆえ、ICSEは研究者の会議で、実務家にとっては役にたたない議論しかしていない、というような批判もよく聞く。このような批判にこたえる意味で、毎回、プログラム委員会の始めには、すぐに役立つ経験論文も広く採録しようと試みられるが、査読した各委員のめがねにかなう論文はなかなかない。そこで、全然別のカテゴリーとして事例研究報告が作られ、すぐに現場で役立つような知見を発表してもらうことになった。今後、この技術論文と事例研究報告が、ソフトウェア工学の理論と実践の両方の側面をカバーして、ICSEがより発展することが望まれる。

### 3.5 日本人とICSE論文

中身の技術的な部分はそれなりにアイディアもあり、技術テクニックも豊富に持っている人も、それを査読者が納得するような形で英語の論文にまとめることは苦労する。筆者も、「英語や表現が悪く採録できない」、という連絡を日常的に受けとっている。

ソフトウェア工学の分野では、技術的な議論の基礎として、ソフトウェアをどう作るか、どう使わせるか、どう評価するか、などのコンセプトが非常に重要で、それをうまく読者に伝えられなければ採録されない。もし英語が母国語であれば、十倍論文を

うまく書けただろう、百倍たくさん書けただろう、など夢想してしまう。時に、そのコンセプトだけで論文を書く欧米人もいるが、我々には無理だね、と、あきらめてしまっている。

英語を母国語とする人が書いた論文で、そのコンセプトの部分が何度読んでも全然分からない場合がある。これは、英語力不足なのか、それともコンセプト自身が整理されていないのか、わからず困ってしまう。多くの場合、後者であろうことは推察できるのだが、確信が持てない。

我々が書ける論文は、比較的コンセプトをすっきりまとめて、それを証明するデータをしっかり集めて表記する、というものであろう。今回、採録された日本からの論文もそうゆう形式になっている。将来、しっかりしたデータに基づいた論文が多数、日本から投稿されるのを期待するのと同時に、新たなコンセプトの提案だけで読者を感心させるような研究者が日本から出ることを期待したい。

### 3.6 ICSEの人脈

ICSEを始めとして、ソフトウェア工学に関する会議には、長年、連続して参加している研究者や実務家が数多くいる。また、一つの会社から同じ人が何年も連続して参加し、顔が知られるようになっている。日本から定常的に参加している人もいるが、概して日本企業の参加者は一回限り、という人が多い。連続参加によって人脈が広がると思われる。

欧米の参加者は、いくつかの研究グループ、同窓グループ、なかよし集団などに属しているように見える。これらのグループの間では、お互いに競争し合ったり、あるいは協調し合ったりしている。IEEE-CSやACM SIGSOFTなどの学会のグループ、CMU卒業生や教官のグループ、Univ. of MarylandやUniv. of Mass. など大学を中心とするグループなどが目立つものである。

過去のICSEでも、このようなグループの一つの人々が運営の中核になっている場合がある。しかし極端に全部その関係の人々、と言うわけではなく、適度にバランスが保たれて運営されている。ICSEの運営委員会などでは、グループ間の考えに違いがあって、延々と長引く深夜までの議論になることもあるが、逆にそのような議論を楽しんでいるようにも見える。ICSE98でも、海外のいろいろなグループの人が準備、実行に関わってもらった。

## 4 むすび

ICSEの役割にはいろいろなものがある。

技術発表や展示に関して、純粋に技術的な情報を得る場としては、ソフトウェア工学の分野では、最高のものであろう。数多くの発表の中から、将来の技術的な方向性を見つけることが出来よう。ICSEから分離、独立して、発展していった学会も多い。プログラム委員会の白熱した議論から、ソフトウェア開発の技術の方向性が形作られているような気がする。

ICSEの役割として、これと同じくらい重要な役割には、ソフトウェア工学の研究を行なう人々の交流の場である。絶対的な価値判断がないソフトウェア工学においては、人と人による情報交換が大きな意味を持つ。本会議の会場（国立京都国際会館）は、発表会場が立派なだけではなく、広いロビー、奇麗な中庭、散策できる周辺の路など、個人的にゆっくり議論できる場が、数多くあって、非常に好評であった。

今回、この会議を組織した者の一人として、参加した人にとって、上記やその他のメリットが得られていることを期待する。組織している側は、冷静に効果を評価することは出来ないが、５年、１０年後に、日本国内のみならず海外にも、この会議のインパクトが何らかの形で残っていることを期待する。