APSEC2000 参加報告

鈴　木　正　人†

2000 年 12 月に開催された第 7 回アジア太平洋ソフトウェア工学国際会議 (APSEC2000) での招待講演、論文発表、パネルなどから最新の話題をいくつか紹介する

A Conference Report of APSEC 2000

MASATO SUZUKI†

This article shows some interesting topics of keynotes, session papers and panels in APSEC 2000, held last December.

1. はじめに

2000 年 12 月 5 日から 9 日にかけて、アジア太平洋ソフトウェア工学国際会議 (Asia-Pacific Software Engineering Conference,APSEC2000)[1] がシンガポール国立大学 (National University of Singapore)[2] において開催された。この国際会議はソフトウェア工学に関する国際会議の元祖とも言える ICSE(International Conference on Software Engineering) が 15 回を数えるうちに、組織の肥大化などにより会議の運営の方向性に閉息感が出てきたことなどを背景に、より時代の変化にあったテーマを取り上げ活発に議論することを目標とした国際会議を開催すべく、発展著しいアジア太平洋地域の研究者や大学、企業が中心となり発足したものである。発足直後にアジアは深刻な経済危機に陥り、参加者数が激減するなど一時は会議の存続が危ぶまれたものの、現在は (日本を除いて) 経済は概ね回復基調にあり、参加者数も順調に延びて来ているようである。

第 1 回の東京 (94) を皮切りに、オーストラリア (95)、韓国 (96)、香港 (97)、台湾 (98) といったように、アジア太平洋地域を会場として行われている。99 年には再び日本 (高松) で開催されており、2000 年はそれに続く第 7 回の開催となる。

筆者は第 5 回 (台湾) から参加しており、高松では Registration Co-chair を勤めさせて頂いた。ただしその際には、裏方の業務があったため本会議の内容をほとんど聞くことができなかったので、自由にセッションに参加することができたのは 2 年ぶりである。本稿ではそこでの話題の中から興味深いものをいくつか紹介したい。

2. 会　　場

シンガポール国立大学 (以下 NUS) はシンガポール中心部から 12km(車で 30 分) ほど西に行ったところに位置している。創立は 1905 年と、新興国シンガポールにしては長い歴史のある大学である。今の大学は 1980 年に 2 つの大学が合併してできたようで、学部生が約 2 万人、院生が約 8000 人の計 2 万 8000 人と約 6000 人のスタッフが 150 エーカー (約 60ha) のキャンパスで生活しているとのことである。島国という場所柄、北米の大学ほどの広大さは望めないが、それでも日本の首都圏の大学よりはかなり贅沢に土地を使えるようで、羨ましい限りである。

今回の APSEC2000 は、NUS の Computer Science のビルのうち、S17 という建物を主会場として開催された。セッション会場としては 200 名程度が収容可能な教室を最大 3 つ使用し、Keynote や Panel にも同じ会場が使用された。

ランチはエレベータホールでケータリングを行い、100 名ほどが入れる休憩室で各自取る形式である。3 日目夜の Conference Dinner は大学内ではなく、市中心部のホテルで開催された。参加者は 200 名程度であったと思われる (主催者側の正式発表なし)。

† 東京工業大学 大学院情報理工学研究科
Graduate School of Information Science and Engineering, Tokyo Institute of Technology

3. プログラム

プログラムとセッションタイトルを以下に示す。

12/5
- (T)'Advanced OO Modeling' by Brian Hederson-Sellers
- (T)'Beyond RUP' by Brian Hederson-Sellers
- (W)Two-State Derivation of System Architectures
- (W)On the Future of SE

12/6
- Keynote 1: Survivability Analysis of Networked Systems
  by Jeannette M. Wing
- Session 1A: Real-Time
- Session 1B: SE Education & Practice 1
- (Lunch)
- Session 2A: Formal Method 1
- Session 2B: Software Reliability
- Session 3A: Analysis & Design 1
- Panel : Formalism in Software Engineering Education & Practice

12/7
- Keynote 2: Building Formal Models for Requirements
  by Axel van Lamsweerde
- Session 4A: Testing & Verification
- Session 4B: Component Based Development 1
- Session 4C: Requirement Engineering
- (Lunch)
- Session 5A: UML/Object-Z/Z
- Session 5B: Software Maintenance
- Session 5C: Architecture Framework 1
- Session 6A: Analysis & Design 2
- Session 6B: Architecture Framework 2
- Session 6C: Component Based Development 2

12/8
- Session 7A: SE Education & Practice 2
- Session 7B: Formal Method 2
- (Lunch)
- Session 8A: Software Process
- Session 8B: Software Metrics
- Session 9A: Distributed Systems
- Session 9B: Panel 'Web-based Software Engineering'

昔から続いている Formal Method などの話題と最近の話題である UML/Component、ICSE2000 でも話題となった教育 (SE Education) に関してはパネルを設けるなど、ソフトウェア工学の様々な分野のテーマをバランスよくカバーしているという印象である。筆者の個人的興味でいえば、Software Process や Metrics などはもう少し発表があってもよいと思うのだが、他の発表でも特に Practical な研究に関しては Process または Metrics に関連するものも散見されている。

4. トピックス

4.1 Keynote

Keynote は 3 件あり、どれも 200 名程の聴衆を集める盛況であった。ここではその中から Jeannette M. Wing による"Survivability Analysis of Networked System"を紹介しよう。

Survivability とは予測不可能なイベントが発生した状態でも、システム全体が異常状態に陥らずにサービスを継続することを保証した性質である。特にネットワーク結合 (networked) されたシステムにおいて Survivability を保証することは重要な課題である。Jeanette らは**シナリオグラフ**という概念を用いて障害を意図的に加えたシステムの状態遷移を記述し、survivability が保証されることを定理証明器により証明するという手法を示している。

まずシステムを構成するノードとリンクを状態遷移機械 (STM) とみなし、ネットワーク結合されたシステムを STM の集合であらわす (ノード間の通信が STM における共有データになる)。次に各 STM のうち障害状態に至る可能性のある遷移を追加する。この過程で STM は非決定的になるが、その分岐をシナリオグラフとして記述することにより、survivability の解析を容易にすると共に、他の性質 (latency やコスト) の解析にも応用が可能となる。

この"networked"という用語がひとつのキーポイントになったようで、会場からは実際の分散ネットワーク上でのアプリケーション (トランザクション?) をどのように扱うかに関する質問があった。筆者の印象では、非決定的な部分の分岐を図で表現するところは時相論理の証明で用いる tablau 法に似ており、実際の応用範囲は定理証明器の能力に大きく影響される (状態爆発を避ける) 点でも同法と共通の問題があるように感じた。実際 Jeanette も定理証明器に関して何か言及していたようだが、残念ながら十分には聞き取ることができなかった。詳細は[3] を参照されたい。

4.2 テクニカルセッション

テクニカルセッションは S17 ビルの 3 つの講義室を利用して開催された。ここでは formal な話題として Object-Z, practical な話題として Architecture & Framework, 最近注目を集めている話題として Education & Practice のセッションからそれぞれ 1 編ずつ紹介する。

5A:UML/Object-Z/Z

実はこれは chair が急用のため筆者が代理で chair を行ったセッションである。ご存じのとおり Z は formal method の代名詞とも言われるものであるが、最近ではこれを OO の概念を取り入れて拡張した Object-Z(R Duke et.al 1995) に関する注目が集まっているようである。ここで紹介するのは UML と Object-Z を融合させた仕様記述の方法である。

”An Integrated Framework with UML and Object-Z for Developing a Precise and Understandable Specification: The Light Control Case Study” by Soon-Kyeong Kim and David Carrington Univ. of Queensland, Brisbane, Australia

仕様記述言語としての UML はその図的な表現から直感的に理解しやすいが、formal な記述が困難という問題がある。UML に欠けている formality を補うものとして、Object-Z に着目し、UML の図的表現を Object-Z に変換するための公理系を定義し、機能モデルをすべて Object-Z で記述するためのフレームワークを与えている。

変換方法としては、

- クラス図に関しては、クラスはそのままクラスに、関連は属性の間に制約をもつクラスとして表現する。
- 状態遷移図に関しては、状態変数を Object-Z のクラスにおける属性として定義する。これらの状態変数は外部から不可視とする。状態遷移はそれを発生させるイベントを定義し、イベントを Object-Z の操作として記述する。遷移により新たに発生するイベント (propergate event) については、他のオブジェクトへ伝搬する場合には (対応する Object-Z のクラスにおける) 属性を変更する操作に変換される。

と、ある意味で非常に直感的である。他に時間を表す表現などもあるが、紙面の関係で割愛する。

Kim らは信号制御システム (クラス数 3, 状態数 8) を例として、Object-Z への変換 (スキーマ数約 20、定理が約 100 行) を (部分的に) 示している。

変換方法自体は理解できるのだが、UML による記述をすべて Object-Z で表現することの有効性には疑問が残る。またある言語による仕様記述を全て別の言語 (この場合 Object-Z) による記述に変換した場合、逆変換が可能なのかがいつも問題になる。Object-Z の世界で検証した結果、仕様に誤りが発見された場合、それが UML 記述のどこに起因するのか、あるいは Object-Z で付加された仕様が誤っているか、または変換方法が誤っているかを判断するのは困難と思われる。筆者もこの点を質問したのだが、あまり明確な回答は得られなかった。残念である。

6B:Architecture/Framework

筆者はコンポーネント関連のセッションに出席するといつも感じるのだが、人によってコンポーネントと呼ぶものの内容が相変わらず不統一である。どうも混乱の原因はコンポーネントに関するよい教科書がない点にあるような気がするが、いかがなものであろうか? Heineman の教科書 (?) が出版されればこの辺の事情も変わるかもしれない。

”Enterprise Modeling using Class and Instance Models” by Rakesh Agarwal, Giorgio Bruno, Macro Torchiano Infosys Technologies, India and Politecnico de Tolino, Italy

UML に代表されるモデリングでは主にクラスに注目し、インスタンスはシナリオ記述程度にしか利用されない。しかしながら、複雑な enterprise system を構築/モデル化する際に、最初から綺麗に構造化されたクラスモデルを得ることは困難であり、現実的でない。インスタンスモデルをまず作成し、そこから得られる情報をクラスモデルに反映させることによってより正確なモデルをより簡単に得るための手法を提唱する。

インスタンスモデルを構成するオブジェクトは Association または PartOf の関係で相互接続されているものと見なすことができる。PartOf 関係が木構造になるならば、それはクラスモデルにおいては (aggregation で接続された) クラス階層によって表現される。そうでないならば ”Business Process(BP)”, ”Organization(ORG)”, ”Enterprise”というクラスを導入し、”Enterprise”を root, BP,ORG をその子とし、各ノードを ORG の子、(PartOf 以外の) 関連を全て BP の子として木構造を構築し、それに対する解釈を与えるというものである。

着眼点は面白いと思うが、最終的にできあがるクラスモデルにおいて、ORG の子となるノード (組織を表す) と、BP の子となるノード (アクティビティを表す、順序つき) にどのような関連があるかを発見し、正しい関連を構築することは容易ではない。両者を仲介する Role というオブジェクトが提唱されてはいるものの、使用方法があまり明確ではない。要するに ”うまくいく” 例だけが示されており、うまくいかない場合にどうすればよいか、そのプロセスが示されていないという印象である。実際に会場からもそのような内容の質問があった。

なおこのセッション、地元シンガポールの Nanyang

大学のグループが no show をしたために 3 件目の発表時刻がくり上がったものの、(発表者以外の) 関係者が会場にいない、というハプニングがあった。並行セッションだと必ずどこかでこのような問題が発生する。困ったものである。

1B:SE Education & Practice

Education には 2 つの立場があり、企業内での訓練 (OJT 含む) と、大学での (主に学部での)SE 関連科目の教育である。筆者は主に後者に携わっておるのだが、教えるべき内容は年々増えていくのに対して、学生の理解が追随していかないのが大きな悩みである。

新しい概念を教えるにはよい事例 (case study) を与えるのがよいことはいまさら言うまでもないが、分散環境上でのソフトウェア開発を題材とした場合、実際の例では複雑すぎ、かといって簡単な例題だとわざわざ分散開発までもない、ということになる。以下に示すのはカーネギメロン大学とミュンヘン工科大学が共同で行った、3 つの分散開発教育に関するプロジェクトの事例である。

”Transatlantic Project Courses in a University Environment” by Bernd Bruegge, Allen H. Dutoit, Rafael Kobylinski, Günter Teubner, Technische Universität München, Germany

CMU と TUM の学生が行った 3 つのソフトウェア開発プロジェクトの教育事例である。物理的に離れた場所で同時に開発を行うことにより、学生は分散開発におけるコミュニケーションの重要性とカルチャーや言語、標準の違いの克服、グループウェア、ビデオ会議、分散リポジトリなどの有効性を実体験することが可能になった。

このプロジェクトのゴールは学生に実際の (分散) ソフトウェア開発状況を体験して貰うことである。このため以下の 3 つのプロジェクトを行った。

- JAMES(Aug. 97 – Mar. 98)
  分散アプリのためのアーキテクチャ構築
- PAID(Aug. 98 – Aug. 99)
  逐次更新される分散データベース
- STARS(Aug. 99 – May. 00)
  電子マニュアルの (分散) 執筆管理

JAMES と STAR は米独で期間にオーバーラップなし、PAID はオーバーラップのある例である。いずれのプロジェクトも複数のチームに分かれて行っている。結果として、学生は以下のような能力を見につけることができた。

- クライアントは対象ドメインの知識や要求を提供し、仕様にたいして辛辣なフィードバックを与えるという意味で非常に重要な働きをする。コミュニケーションに要する時間が短いほど相互の理解の不一致は避けられる。
- 実際にチームのメンバのうち 2 名が (大西洋を越えて) 互いに相手のチームと一緒に作業を行ったが、旅費/滞在費がかかる割にそれに見合った成果はあがらなかった。
- ビデオ会議は単独ではほとんど効果がない。ただしビデオ会議のバンド幅が低いことが原因で、会議の前に双方とも十分な用意をするような習慣が発生し、それにより状況整理や理解が進むといった副次効果があった。ビデオ会議は状況報告には役立つが、意思決定をこれで行うのは無理がある。
- プロジェクト管理は 3 階層 (全体マネージャ、サイトマネージャ、コーチ) で行うのが効率がよい。各サイトマネージャが (プロジェクト開始時に) 数週間の直接対話を行うことは有意義であったが、双方の全体マネージャで対話を行っても、プロジェクトを相互非依存な 2 つに分割するだけであった。

等、かなりショッキングな内容が報告されている。興味のある向きは Proceedings を参照されたい。

双方で数 10 名の学生とほぼ同数のスタッフを使用して、約 3 年にもわたるプロジェクトが達成されたというだけで驚きである。アジア地区内で同様のプロジェクトは可能だろうか? おそらく言語の壁が米独以上にあるため、円滑なコミュニケーションは難しいだろう (中国が中心となれば可能かもしれない)。

4.3 パ ネ ル

どの国際会議でも最終パネルというのは非常に白熱するものであるが、今回のテーマは ”Web-based Software Engineering” (Chair:Karl Reed) である。

まず最初に 4 名のパネリストが Web 上でのソフトウェア開発の事例を紹介し、続いて会場から質問を受けるはずだったのだが、プレゼンテーションの途中で会場から”Programming for the Web(Web 上で動作するソフトウェアの開発)”なのか、”Programming by the Web(Web を利用したソフトウェアの開発)”なのかどちらかはっきりして欲しい、という質問が飛び出し、2 名は”for the Web”, 2 名が”by the Web” ということでその後の議論が続き、結局最後までまとまることはなかった (途中で Karl Reed もまとめるのを諦めていた?)。新潟工科大 (当時) の青山先生の i-mode を事例とした紹介が会場にも非常に強いインパクトを与えていたようである。

5. おわりに

20 世紀最後の APSEC に参加することができたわけであるが、SE 各分野における現状の問題把握と、将来にむけての問題の検討がなされており、非常に有意義な会議であったと思われる。

次回は 2001 年 12 月 4-7 日、中国マカオ特別行政区において APSEC2001 が予定されている。Call for Paper を含め、詳細は[4] に示されている。論文投稿締め切りは本報告と同じ 2001 年 6 月 1 日であるが、昨年同様若干の延長が予想される。なお本稿では触れなかったワークショップの報告が[5] にある。興味のある方は参照されたい。

参 考 文 献

1) 7th Asia-Pacific Software Engineeing Conference Home Page.
`http://www.comp.nus.edu.sg/apsec2000`
2) National University of Singapore Home Page.
`http://www.nus.edu.sg/`
3) Jeanette M. Wing, Survivability Analysis of Networked Systems, CMU-CS-00-168 October 2000.
(`http://www.cs.cmu.edu/~wing` に PS ファイルあり)
4) United Nations University/International Institure for Software Technology, APSEC 2001 Home Page,
`http://www.uust.unu.edu/2001/`
5) National University of Singapore, APSEC 2000 Workshop Report,
`http://www.service-oriented.com/apsec/`