2007－SE－157 (4)
2007／9／27

第29回ソフトウェア工学国際会議（ICSE2007）参加報告

鷲崎 弘宜[1] 白井 明[2] 松塚 貴英[3] 吉田 尚志[4]

[1] 国立情報学研究所 [2] 日立製作所/Stanford University [3] 富士通研究所 [4]NTT データ

2007年5月に開催された第29回ソフトウェア工学国際会議（ICSE2007）に参加したので，取り上げられた主な内容を紹介する．本会議の傾向として，設計，プログラム解析およびテストに関する研究発表が多く，これらの技術領域への取り組みの活発化を伺えた．また，扱う問題領域は組込みシステムからエンタープライズシステムまで多岐にわたり，ソフトウェア工学適用の広がりが見られた．

Report on the 29th International Conference on Software Engineering (ICSE2007)

Hironori Washizaki[1] Akira Shirai[2] Takahide Matsuduka[3] Hisashi Yoshida[4]

[1]National Institute of Informatics [2]Hitachi Ltd. / Stanford University [3]Fujitsu Laboratories Ltd. [4]NTT Data Corporation

This paper reports major topics of the 29th International Conference on Software Engineering at May 2007. There are many presentations on design, program analysis and testing. Moreover, we saw the wide spread of software engineering applied to various problem domains such as embedded systems and enterprise systems.

1 はじめに

本稿では，2007年5月に米国・ミネアポリスにて開催された第29回ソフトウェア工学国際会議（29th International Conference on Software Engineering[1]: ICSE2007）において取り上げられた話題を紹介する．その紹介を通じて，執筆時点における最先端のソフトウェア工学研究の国際的傾向の一部を，概観する．また同時に，ICSEおよび類似の論文採択率の厳しい国際会議（例えばFSE[2]やASE[3]）への今後の活発な論文投稿と参加を促すことも目的とする．

ICSE（イクスィー）は，ソフトウェア工学の分野を扱う国際会議の中で最も権威が高いものと一般に認知されている．過去のICSEの様子については同様の報告[4, 5, 6]を参照されたい．今回のICSE2007は，ACM SIGSOFTおよびIEEE Technical Council on Software Engineering (TCSE)の共同で，ミネアポリス・ヒルトンホテルにて5月20日から26日まで（本会議は23日から25日まで）の日程で開催された．

全体テーマとしては，近年の一般業務から政府，社会生活および健康維持にいたるまでソフトウェアが果たす役割の増大を受けて，”ディペンダブルソフトウェアの開発”が掲げられ，同テーマに沿って基調講演等が企画された．参加者数は，併設ワークショップ等も含めた総数で約1000人であり，開催国である米国からの参加者が多数を占めた．

2 ICSE2007のプログラム概要

2.1 本会議の構成

本会議の3日間は，各日にそれぞれ行われる3件の基調講演を除いて，例年以上に並列度を上げて最大7トラックにより構成され，それぞれに扱う論文の種類や採択基準/採択率が異なる．本会議の構成トラックのうちで，以前より設置されているものを以下に示す．

(1) 研究論文: 多数の投稿より厳選された49件の研究論文発表．研究論文は335件の投稿があり，昨年の370件に比べると若干の減少はあるが，採択率14.6%は依然として厳しいものであった．

(2) 経験論文: 企業等におけるソフトウェア工学の

実践に関する15件の経験論文発表，投稿52件，採択率29%.

(3) 教育論文: 大学等におけるソフトウェア工学教育に関する13件の論文発表，投稿50件，採択率26%．2003年より設立された比較的新しいトラックである.

(4) 研究デモ・ポスター: 「動く」「見える」最新研究成果としての12件のデモンストレーション発表，投稿56件，採択率21%．また，別個15件のポスター発表もあった.

さらに，今回のICSE2007で新設されたトラックや企画を以下に示す.

(5) ソフトウェア工学の未来 (Future of Software Engineering: FoSE):

ソフトウェア工学の多様な各領域の現状と展望に関する25件の招待発表．ソフトウェア工学の研究と実践が，様々なレベルにおいて世界規模で取り組まれ知見が蓄積されつつある中で，コミュニティとして既に何をどれだけ達成し，これからどこに向かい何を実践・改善すべきかを整理する必要性を受けて開設された．短時間に各領域の全体像を把握できるため，会期を通して立ち見が出るほどの盛況ぶりであり，研究論文において幾つか閑散としたセッションが見受けられた状況と対照的であった．ICSEとしては2000年に同種の企画が実施されている.

(6) 思索のための"食事" (Food For Thought): ピープルウェアおよびアジャイル契約に関する2件パネル討論と，モデル駆動開発に関する1件の講演．全て昼食時に開催され，各討論・講演をツマミにご飯をどうぞという企画である．本会議中の昼食は参加者間で交流を深める貴重な機会であるが，そのきっかけを与える役割も果たしたと思われる．例えばTom DeMarcoを囲んでのピープルウェアに関するパネル討論では，Linda Risingがソフトウェア開発の社会的側面を成功させるカギがソフトウェアパターンにあると高らかに主張し，聴衆の喝采をあびていた.

(7) インパクト・パネル (Impact Panels): ACM SIGSOFTにおいて現在進行中の"Impact Project"（ソフトウェア工学の諸技術を整理・体系化して，もたらした効果を評価する試み[7]）の紹介，および，その1例としてミドルウェアに関する研究上のインパクトに関するパネル討論.

2.2 併設イベント他

ICSEの傾向として，ソフトウェア工学の技術や適用分野の多様化に対応して，多数のワークショップやチュートリアル，シンポジウム等を本会議と前後して開催することが挙げられる．以下の併設イベントはBoehmシンポジウムおよび博士課程シンポジウムを除いて，それぞれ特定の比較的萌芽的な分野を深く取り上げて，同分野における研究の発展や多様な分野からの参加を促すことを目的とする.

(8) ワークショップ: 36件の提案中22件のワークショップが開催され，それぞれ本会議とは独立に論文募集，査読，採択，論文/ポジションペーパ発表が行われた．各ワークショップは，最近の注目される1つの話題（ホットトピック）に特化して，比較的少人数による集中的議論によりその分野における研究の発展とコミュニティ育成を促す役割を担っている．そのため各ワークショップでは，本会議の各トラックと比較して未完成ながらも非常に斬新な手法や着眼点が披露されることも多く，ソフトウェア工学および周辺領域の今後の方向を捉える上で貴重な情報源である．ワークショップテーマは，「シナリオと状態機械」（6回目）や「ソフトウェア品質」（5回目）といった回を重ねてコミュニティとして成長しつつあるものから，「航空宇宙ソフトウェア工学」といった比較的新しいものまで多岐にわたっている．回を重ねたものについては，その十分なコミュニティ成熟に伴い，今後の国際会議への格上げが示唆されるものもあった（例えば「ソフトウェアリポジトリのマイニング」（4回目））.

(9) チュートリアル: 26件の提案中16件（全日8件，半日8件）が本会議の前後で開催された.

(10) 博士課程シンポジウム: 博士課程の学生による研究テーマや途中成果に関する16件の発表．研究の方向性や内容，手順などについて他の学生や経験をつんだ研究者/教員と議論しアドバイスを受ける貴重な機会として，毎年開催されている.

(11) 新任教員シンポジウム (New Software Engineering Faculty Symposium): 新任のソフトウェア工学関連教員（助教など）を対象としたシンポジウムであり，経験をつんだ教授陣より指導方法や研究の進め方，キャリアの積み方についての講演/アドバイスが行われた.

(12) ソフトウェアプロセス国際会議 (International Conference on Software Process: ICSP'07) ICSE2007とは独立であるが，ソフトウェアプロセスに関するシンポジウムとして開催された．昨年は同種の会議がSPW/ProSim2006なる名称でワークショップとして開催されており，プロセス関係の研究の活性化を受けて国際会議へと格上げされた格好となった.

(12) Boehmシンポジウム (Software Engineering: The Legacy of Barry W. Boehm): ソフトウェア工学で最も著名な研究者の一人であるBarry Boehmの足跡を辿る形で，主要技術の発展

経緯を紹介する連続講演の形式で開催された.

2.3 ソーシャルイベント，表彰その他

ソーシャルイベントとして，本会議前日の夜にレセプション，本会議2日目にバンケットが開催され，それぞれ多数の参加者が交流し歓談する良い機会となった.

バンケットの開始前には本会議場にて，以下の表彰が行われた．まず，ソフトウェア工学分野で重要な貢献した個人に授与されるACM SIGSOFT Distinguished Service Awardは David Notkin（University of Washington）に贈られた．また，優秀な論文に授与されるDistinguished Paper Awardは以下に贈られた.

- E. Duala-Ekoko and M.P. Robillard, "Tracking Code Clones in Evloving Software"
- S. Kim, et al., "Predicting Faults from Cached History"
- S. Nejati, et al., "Matching and Merging of Statecharts Specifications"
- A. Kiezun, et al., "Refactoring for Parameterizing Java Classes"

さらに，10年前のICSE（今回はICSE 1997が対象）における発表論文のうちで最も大きな影響を与えたものに与えられるMost Influential Paper Awardは下記に贈られた.

- A. Carzaniga, et al., "Designing Distributed Applications with Mobile Code Paradigms"

なお，本会議および併設イベントの発表論文を全て収めた予稿集は，紙媒体では用意されず，昨年同様にUSB形式で参加者に配布された.

3 本会議の話題

3.1 基調講演

基調講演は，技術，倫理，品質（ディペンダビリティ）に関する3件が企画され，"ディペンダブルソフトウェアの開発"という全体テーマに沿ったバランスのよい内容であった．基調講演の様子を図1に示す.

(1) The Architecture of the Apex Platform, salesforce.com's Platform for Building On-Demand Applications (Steve Fisher, Salesforce.com)

SaaSはサービス指向を発展させた近年の考え方である"サービスとしてのソフトウェア"（Software as a Service: SaaS）の最も代表的な成功例である顧客関係管理サービスSalesforceを支えるオンデマンドなアプリケーション開発技術の紹介があった．Salesforceは，各顧客の要求に応じてサービスを組み合わせて迅速かつ容易にアプリケーションを開発するオンライン環境（Platform as a Service: PaaS）Apexに基づいたものであり，サービス指向からSaaS，PaaSへと至るサービス提供・連携の実用化と発展を予感させるものであった.

図1: 基調講演の様子

(2) Computer Professional Ethics in Theory and in Practice (Deborah G. Johnson, University of Virginia)

コンピュータ技術者の職業人としてあるべき倫理・道徳観について，理論や実践の観点から紹介があった．今日，種々の学会が倫理規定・綱領を定めているが，そのような明文化された規定が全てではなく，責任をもって信頼できる製品を生み出すという態度や，社会に対して影響する信頼性や安全性などを包括した文化的側面などの重要性が指摘された.

(3) Limits to Dependability Assurance - A Controversy Revisited (Bev Littlewood, City University, London)

ソフトウェアの信頼性向上についての取り組みの発展経緯と展望について紹介があった．信頼性向上技術は進展しつつあるものの，信頼性に関する要求の記述困難さなどに起因して，ソフトウェアが現実の運用下において極めてディペンダブルであることを，運用前に保証することが依然として困難であることなどが指摘された.

3.2 研究論文

採択された研究論文の分野内訳を表1に示す．前年ICSE2006との傾向を比較すると，プログラム解析やテスト手法について発表が集中している点は同

様であり，継続的な研究発展の様子が伺える．他方，形式手法および実証的ソフトウェア工学を単独で扱うセッションは無くなっており，その理由として当該領域に特化した国際会議の定着や，設計等の応用技術への組み入れ等が想定される．さらに，新しくセキュリティのセッションが単独で設置されており，安心・安全な社会形成が求められる中でソフトウェアセキュリティの研究ニーズの高まりを感じさせる．

表 1: 研究論文の傾向

分野	件数
解析	9
テスト	6
デバッグ・欠陥	5
設計	5
アスペクト指向	3
アーキテクチャ	3
セキュリティ	3
モデリング	3
リファクタリング・再利用	3
クローン	3
保守	3
人的要因	3

扱われる問題領域は組込みシステムから Web アプリケーション，分散システムまで多岐にわたり，ソフトウェア工学適用の広がりと領域ごとの堅実な積み重ねを見ることができた．

3.3 経験論文

経験論文は FoSE など人気があったセッションと比べると，人が少ない傾向があった．別のフロアで行われていた影響もあるかもしれない．

(1) Agility and Experimentation: Practical Techniques for Resolving Architectural Tradeoffs

ソフトウェア開発の課題は以下の 2 点である．

- 要求が不明確である．
- 短期間で市場に出すことが求められる．

この課題を解決するために，以下の 2 点が重要である．

- 何が不明確なのかを特定し，明らかにしていく努力を続けること．
- 常に状況の変化に適応し続けること．

事例として，パフォーマンスやスケーラビリティ，および開発期間のトレードオフを挙げた．これに対して，実験を行ってパフォーマンスやスケーラビリティを実際に測定し，結果をガイドとして短期的な目標を立て，インクリメンタルに開発する方法を提案した．

感想: Agile Alliance をはじめとする Agile 開発陣営が主張する Agile 開発の長所が活かされた形の報告であった．一方で，インクリメンタルな開発は，リスクへの対処を容易にする代わりに，開発のオーバーヘッドを増す影響もある．機能や開発期間の分割単位を決定する際にはこの両者のトレードオフを考慮する必要がある．実践報告としては，このトレードオフについても論じてほしかった．

(2) Company-wide Implementation of Metrics for Early Software Fault Detection

FST (Fault-Slip-Through) と呼ぶメトリクスを定義した．まず，テストから運用までの工程を以下とする．

- UT(Unit Test)
- FT(Function Test)
- ST(System Test)
- OP(Operation)

UT や FT で見つかるべきバグが ST で見つかった場合のように，本来より後の工程で検出したバグの割合を FST to phase と呼ぶ．逆に，FT で見つかるべきバグが ST や OP で見つかった場合のように，検出するべき工程で検出できなかったバグの割合を FST from phase と呼ぶ．Ericsson の 9 つのプロジェクトでは，ST や FT の FST to phase の中央値は約 60%であった．FST from phase の中央値は UT で約 64%，それ以外は 11-12%であった．評価の際には単に割合を見るだけでなく，絶対数も考慮に入れるべきである．

感想: バグの修正コストは，早い工程で見つけるほど小さくなる．これらのメトリクスを使って適切にテストを行なえているかどうかを評価することは修正コストの最小化に寄与すると考える．

(3) Can Requirements Be Creative? Experiences with an Enhanced Air Space Management System

顧客から要求を単に聞き出すだけでなく，アナロジーを使ってより多くのアイデアを得た事例の紹介である．航空管理システムの開発において，システム境界は定めたが仕様は未確定の段階でワークショップを開いた．博物館の展示管理のアナロジーを使うことで 14 のアイデア，TV プログラム管理のアナロジーを使うことで 8 のアイデアを得た．これらのアイデアを使ってユースケースと全体のストーリーを作成し，仕様に影響があるかどうかをチェックし，仕様を確定した．

感想: 顧客との対話で，技術的な事柄の説明にアナロジーを使うことは多い．しかし，開発対象のシステムそのものを置きかえることはあまり多くなかったように思う．これが要求獲得に寄与するのであれば興味深い．

3.4 教育論文

教育論文トラックに対する投稿数は50件であり，昨年の39件に比較すると投稿数が増大していることから，教育の振り返りや知見の整理，向上といった取り組みの国際的な活発化が伺える．

例えば日本からは，トップレベルのソフトウェアアーキテクト育成のための教育プロジェクト"トップエスイー（TopSE）"[10]の取り組みが報告され，アーキテクト育成のための実践的な教育のあり方などについて議論が交わされた．

3.5 研究デモ・ポスター

発表者にさかん質問がなされており，盛況であった．ポスター会場の様子を図2に示す．

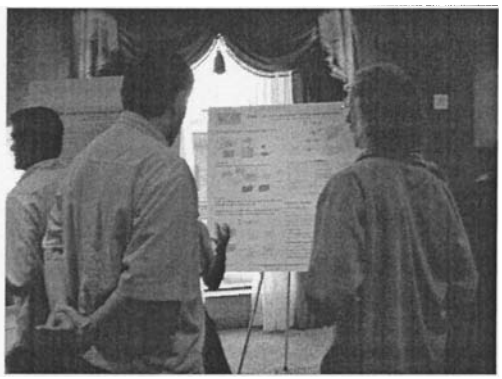

図 2: ポスター会場の様子

Crisp - A Fault Localization Tool for Java Programs

デグレードを起こした時に，正常であった時点のソースと現在のソースを比較して，どの部分の修正が問題であったかを特定するツールである．

ソース間の構文木を比較して，失敗したテストケースに関係があると判定した修正を提示する．修正はメソッドやフィールド単位に分解されており，これを正常だったソースに順に適用して再テストを行うことで，どの部分が問題かを特定する．

STRADA: A Tool for Scenario-based Feature-to-Code Trace Detection and Analysis

ソフトウェアのFeature，テストシナリオ，メソッドのトレーサビリティを管理するツールである．

例えば動画プレイヤの再生ボタンをクリックすると，再生を開始するFeatureと時間にそって再生を継続するFeatureの2つのFeatureが動作する．この場合，テストケースとFeatureは1:1対応しない．このツールは，ある操作を行った時に呼び出されたメソッドをキャプチャして記録し，操作によって呼び出されたメソッドの差集合などから可能な限り正確にFeatureとメソッドの対応関係を検出する．前述の例では，再生が継続している状態で呼び出されているメソッドをキャプチャして引くことにより，再生開始のFeatureに関連づくメソッドを特定できる．

SCL: Static Enforcement and Exploration of Developer Intent in Source Code

SCL(Structural Constraint Language)とよぶ独自言語で制約を定義することにより，設計者の意図に反した実装を検出することができる．

SCLは一階述語論理をベースにJavaの型情報を利用できるようにしている．C++も一部適用可能である．例えば，次のJavaの一般的な実装規則の違反を検出できる．

- `equals()`の引数は`Object`でなければならない．
- `equals()`は例外を送出するべきではない．
- `equals()`をオーバーライドしたら`hashCode()`もオーバーライドするべきである．

3.6 ソフトウェア工学の未来（Future of Software Engineering）

FoSEトラックより幾つかの注目する講演を紹介する．

A Future for Software Engineering?

FoSEトラック全体のプログラムの説明である．

ソフトウェア工学のトピックの種類は数十年前からほとんど変わっていない．しかし大きな進歩を遂げている．コミュニティや，ソフトウェアの適用分野は大きく広がっている．

研究者と実践者の相互作用をアセスメントするImpact Projectを行っている．今後の大きな問題は，ソフトウェアのサイズ，スコープ，複雑さの問題である．

Model-Driven Development of Complex Systems: A Research Roadmap

Model-Driven Engineering (MDE)では，問題領域とソフトウェア実装の間のギャップを埋めることが重要である．プログラマが実装言語でプログラムを書く際にも，実装言語を通して抽象モデルを構築している．しかしこれでは抽象度が低いため，複雑になってしまう．これは前述のギャップが大きいことを意味する．ギャップを小さくするためには，より抽象度の高いモデルが必要である．

MDEは抽象モデル構築のためのDSL(Domain Specific Language)やツールセット，フレームワー

クを提供することを目指す．ただし，これは簡単に実現できるものではない．

UMLを使ったモデリングから，問題領域を表すためのコアとなる小さい概念セットを見つけることは非常に難しいことが分かってきた．また，DSLを使う場合も，DSL-babel(多くのDSLを併用する状態) は避けなければならない．MDEのゴールを実現するためには，ソフトウェア工学のさまざまな分野の成果を統合する必要がある．

感想: モデル駆動開発により再利用が進んだり，ソフトウェア開発の分業化が進むと期待されているが，まだ課題も多い．抽象的な概念を開発者間で正確に共有できるように表現することが難しい点が大きな問題だと考える．実行可能，または検証可能なモデルの表現や，具体的に動作しているプログラムの可視化の研究がこの問題を解決する助けになると考える．

A Perspective on the Future of Middleware-based Software Engineering

ミドルウェアは，より広い分散システムの基盤となってきている．各時代によって以下を対象とする．

- 80年代はLAN
- 90年代はWAN
- 2000年代はOpen Network
- 今後はPervasive Network

次世代のミドルウェアは次の特性を持つ必要がある．

- ユニバーサルな相互接続性: 任意のリソースと対話できるプロトコルが必要．
- オープンな協調: お互いに知らないノードとの協調動作．
- コンテキストを考慮した適用性: システム，ユーザ，物理層などのコンテキストに，上で動くアプリケーションや，自分自身を適応させる．

また，ミドルウェアは開発プロセスの中心になる．要求に応じてミドルウェアを選定し，アーキテクチャ構築，モデリング，分析，実装，テストは全てミドルウェアを意識して行う．

感想: 分散システムが複雑化するに従って，セキュリティの確保が問題となってきている．今後ユビキタスコンピューティングの世界が実現すれば，ますます大きな問題となる．セキュリティは機能やシステムを横断する概念のため，個々のシステムでの対応よりは可能な限りミドルウェアで包括的に対応できることが望ましい．

New frontiers of reverse engineering: このセッションでは，リバースエンジニアリングの歴史を概観し，現在の状況と今後のチャレンジについてのアウトラインを示した．まず，歴史的には次のようにカテゴリわけができる．

- '80s 'スパゲティ' コード (単一システム)
- '90s 2000年問題 (OO, web, ...)
- '00s セキュリティ (パーベイシブ, モバイル ...)

次に，現在の技術状況は，次のようになっていると分析している．

- 言語自身が解析ツールを提供する (例: リフレクション)
- 統合開発環境が抽象構文木 (AST) をサポートしている (例: eclipse)
- クロス言語，クロス技術の利用
- 人的なフィードバックと自動化されたインテグレーション
- 可視化の進化: ズームからインクリメンタルレンダリングへ

そして，これからの課題を挙げていた．まず巨大なシステムへの対応が必要であること．複合化したシステムでは，システム同士の境目もあいまいで，設定や配備が動的になるため，このような「システムのシステム」に対する技術が必要になるだろうということだった．次に，連続的な (continuous) リバースエンジニアリングの必要性を挙げている．現在でもeclipseのJDTなどで一部実現されているところもあるが，コーディングスタイル，カバレッジ，リファクタリングなど，開発作業の中にリバースエンジニアリング技術が組み込まれていく必要がある．最後に，「ビルトイン」なリバースエンジニアリングの技術について言及した．これはいわゆる「自己進化」であり，リバースエンジニアリング技術を活用してソフトウェア自体の最適化などを行っていくというアプローチが必要であろうということであった．

4 併設イベント

4.1 併設シンポジウム

Software Engineering: The Legacy of Barry W. Boehm

本シンポジウムは，B. Boehmの40年間のソフトウェア工学に対する貢献（具体的には数々の論文）が書籍にまとめられ出版されたことを契機に，彼がこれまでに業績を残してきた数々のソフトウェア工学分野において，それぞれ分野で第一人者がBoehmの貢献に触れながら発表を行うというものであった．

それぞれの分野ごとに歴史，特に解決されてきた課題やそのための手法が述べられるとともに，残存する問題点や現在の研究テーマなどが述べられた．加えて，40年間の貢献とその書籍化への祝辞が述べられていた．最後には，上海でのキーノートに続き，B.Boehm自身による現存する課題の整理や今後のソフトウェア工学の方向性について述べられた．発

表者の顔ぶれ（F. Brooks, T. Demarco, V. Basili, W. Royce など）や発表があったテーマ（書籍 [8] の章構成に沿っている）にも現れているがバランスよく実際のソフトウェア開発課題をいろいろなソフトウェア工学研究テーマに結びつける手腕，そして，関係者をひきつける人間性や魅力が 40 年続く貢献を生んだとまとめられていた．発表資料などは USC のサイトで公開された．

4.2 ワークショップ

22 件のワークショップのうち，注目する 1 件の概要を紹介する．

2nd Workshop on SHAring and Reusing architectural Knowledge - Architecture, rationale, and Design Intent (SHARK/ADI'07)

本ワークショップは，アーキテクチャ設計に関する情報共有を検討している．アーキテクチャ設計は，複数の利害関係者が関わる開発において，中心的な成果物となり，より重要な役割を持つようになっている．しかしながら，現状のアーキテクチャ設計の手法ではコンポーネントとコネクタといった構造記述面に焦点があたっており，意思決定結果としての設計文書としては十分な情報を残すよう考えられていない．例えば，組織的な制約，開発プロセスの条件，ビジネス上での判断といった設計の根拠となるような情報や判断理由が文書から欠けてしまっている．

今回のワークショップでは，2 日間にわたって，現在の手段では欠落しがちな設計情報，すなわちアーキテクチャ知識を，どのように整理していくべきか，主にナレッジマネジメントをするという観点から議論された．発表には，ナレッジマネジメントツール，UML 上に非機能要求を盛り込むためのプロファイル，というような実現方法の話題や要求事項から具体的な設計に落とすまでの過程を整理し，そのときに扱うアーキテクチャ知識情報を述べる話題，そしてそのような場面での懸案事項や課題といった事柄が話題となっていた．

発表以外では，アーキテクチャ記述に関する標準である IEEE 1471：2000[9] で現在進行中の ISO と IEEE による改定作業にコメントしていくための情報集めも行われていた．ワークショップでは，特に設計の「根拠（rationalre）」を設計という意思決定結果と対にして考えることが重要であることが述べられており，そのような修正案が検討された．まだ，ワークショップ自体は 2 回目ということで，今後は，標準を含めてどのような重要情報があるのか，そしてそれを共有していくための手法について整理されていくことが望まれる．

4.3 チュートリアル

16 件のチュートリアルのうち，注目する 1 件の概要を紹介する．

Aspect oriented design in Java/AspectJ and Ruby: このチュートリアルでは，アスペクト指向の技術の基礎とデザインパターン，実際のプログラムの注意点などについて説明があった．表題にある Java/AspectJ と Ruby はサンプルに選んだもので，実際には「静的言語」「動的言語」というカテゴリわけを行って説明をしていた．つまり，以下のようになる．

- 静的言語のサンプル: Java/AspectJ
- 動的言語のサンプル: Ruby

興味深いのは，現在の多くの AOP システムはアスペクトとクラスが「非対称 (asymmetric)」である，という点である．つまり，アーキテクチャはオブジェクトモデルが担当し，アスペクトはクラスをアドバイスしたりオブジェクトモデルの接続を行ったりする．「対称 (symmetric)」な AOP というのは，アスペクトとクラスが対等にアーキテクチャを構成するという考え方だが，現時点では技術的に成熟していない．また，AOP を以下のようにカテゴリ分けしている．

- Speculative: 対象モジュールからはデータを取得するのみで，状態を変更しない
- Regulative: フロー制御を変化させるが，既存の状態やフィールドを変更しない
- Invasive: 状態やフィールドを変更するが，重要な属性の変更は行わない

5 所感

鷲崎: FoSE や Boehm シンポジウムなど，ソフトウェア工学の発展経緯（および展望）を概観する企画が多数設置され，盛況であった．2007 年は，ソフトウェア危機の認識を受けて 1968 年に NATO ソフトウェア工学会議が開催されてから 40 年目の節目にあたり，これまでを振り返る 1 つの歴史的ポイントになったということであろう．

研究論文トラックでは，テストやセキュリティ等の信頼性・ディペンダビリティ向上に関する技術発表が多数存在し，次の 40 年への 1 つの方向性が感じられた．日本国内からの発表は依然として少なく，これらの領域も含めて研究と実践のより一層の蓄積が望まれる．

白井: ソフトウェア工学分野の国際的な発展の流れにおいて，中国の勢いが増してきていることが論

文提出数の数字として現れていることが興味深い．会場でもアジア系の人間は中国の方だと思われる名前が多かった．

Java など特定の実装言語にフォーカスしたセッションや，セキュリティのセッションには人が少ない印象を持った．実装言語についてはアカデミックな話題は少ないのかもしれないが，セキュリティは身近に大きな問題となっているので，意外に感じた．

松塚: 注目する個々の技術について感想を述べる．まず AOP では，対象モジュールの振る舞いを自由に変更できるという点が強調されるが，無節操に利用すると元のプログラムの振る舞いがわからなくなり，せっかくのアスペクト指向の概念が台無しである．それを防ぐためにさまざまなデザインパターンやガイドラインを提示していたのが印象的であった．

また，リバースエンジニアリングについて，技術自体の歴史は古く，その技術単体での将来課題はあまりないのではないかと思っていたが，活用すべき場所はまだまだあるという感想を持った．

吉田: 京都で開催された ICSE 以来，2 度目の ICSE 参加だったが，今回も盛況でソフトウェア工学分野での主要な会議なのだと感じさせた．特に私にとって前回と異なるように感じられた点は，1 つは扱われているテーマがより具体的な応用を意識したものになってきたこと，もう 1 つは細分化している個別テーマをまたがった学際分野が多く見られたことであった．

前回行った時に比べ，今回は，自動車搭載ソフトウェアに対してのソフトウェア工学の話題，Self-managed システムを対象とした設計の話題，そしてインターネットとソフトウェア工学の話題といったように，より具体的な応用を意識し対象としているように感じられた．以前のように one size fits all のテーマは少なくなり，応用対象に対してより細分化・詳細化されたテーマが検討されてきたのではないかと推察する．また，同時に，現実の開発を意識した経済性・価値，人間・組織といった別分野の要素を入れたり，組込みシステムのようにハードウェアを含めたシステムに視野を広げ，分野をまたがったテーマも前回に見たときよりも多くなっているように感じた．

実際の開発現場とは，やはりある程度「離れた場所」ではあるが，「全体的な視点」の獲得や世界でも同様に存在する課題や検討されているテーマの流れといったものを，他の研究者と直接話したりすることで「肌で感じる」ためにも ICSE への参加は今後も検討していこうと思う．

6 おわりに

冒頭で述べたように，ICSE2007 は多数の参加者を集めて大成功であったといえる．今後は，2008 年はドイツ・ライプツィヒにおいて，2009 年はカナダ・バンクーバにてそれぞれ開催される予定である．特に ICSE2008 について，本稿の研究集会における発表時点では，幾つかの論文投稿が間に合う．日本からの投稿と発表，参加が増えることを期待したい．

参考文献

[1] 29th International Conference on Software Engineering, http://web4.cs.ucl.ac.uk/icse07/

[2] 14th ACM SIGSOFT Symposium on the Fundations of Software Engineering, http://www.cs.uoregon.edu/fse-14/

[3] IEEE/ACM International Conference on Automated Software Engineering, http://www.ase-conference.org

[4] 青山幹雄, 松下誠, 藤枝和宏: 第 25 回ソフトウェア工学国際会議（ICSE2003）の話題, 情報処理学会研究報告「ソフトウェア工学」, 2003-SE-142(8), 2003.

[5] 松下誠, 大場勝, 肥後芳樹, 天嵜聡介, 川口真司, 水野修, 丸山勝久: 第 27 回ソフトウェア工学国際会議（ICSE2005）参加報告, 情報処理学会研究報告「ソフトウェア工学」, 2005-SE-149(6), 2005.

[6] 鷲崎弘宜, 青山幹雄, 中川博之, 角田雅照, 吉村健太郎: 第 28 回ソフトウェア工学国際会議（ICSE2006）参加報告, 情報処理学会研究報告「ソフトウェア工学」, 2006-SE-153, 2006.

[7] ACM SIGSOFT: The Impact Project, http://www.sigsoft.org/impact/

[8] Barry W. Boehm: Software Engineering: Barry W. Boehm's Lifetime Contributions to Software Development, Management, and Research, Wiley-IEEE Computer Society, 2007.

[9] ANSI/IEEE 1471-2000, Recommended Practice for Architecture Description of Software-Intensive Systems, http://standards.ieee.org/reading/ieee/std_public/description/se/1471-2000_desc.html

[10] 国立情報学研究所・トップエスイー・プロジェクト, http://topse.jp