2005年1月25日発行
第1巻　第1号（通巻1号）
ISSN 1349-8622

SEC journal

1

Software Engineering Center

SEC journal 創刊記念招待論文

ソフトウェア開発負荷見積り式の汎用化の提案
冨永　章

最新ソフトウェア技術による高信頼組込みソフトウェアの開発
片山卓也

IPA
独立行政法人 情報処理推進機構
http：//www.ipa.go.jp/

SEC journal
Software Engineering Center
No.1目次

創刊にあたって

SEC（ソフトウェア・エンジニアリング・センター）は2004年10月1日、IPA（独立行政法人 情報処理推進機構）の中に設置されました。産学官連携の拠点として、「高品質のソフトウェアを効率よく開発する手法を確立し、普及させる」ことを旗印に活動いたしますので、読者各位のご支援とご協力をよろしくお願いいたします。

活動の一環として、「SEC journal」を発行することにしました。ソフトウェア・エンジニアリングに的を絞り、学術論文のみならず、プラクティカルでエンピリカルな情報を幅広く発信したいと考えています。また、旧「ソフトウェア工学研究財団（RISE）」（当時 小長啓二理事長）から継承した基金を有効に使わせていただき、斯界の権威の先生方に選んでいただいた優秀論文を表彰いたします。ぜひとも積極的な投稿をお願いいたします。

● ソフトウェア業界への厳しい要望

さて、サービスや機器を動かし安定性や操作性を左右しているのが、今やハードウェアではなくソフトウェアであることは言をまたないところですが、昨今は、価格、品質、期間に関する要望が、大変厳しくなってきています。吉野家の牛丼ではありませんが、「早くて、安くて、美味い」ことが厳しく求められています。ソフトウェアの場合、この三拍子揃ってお客様に納められるのが、なんとプロジェクト全体の25％程度だといわれています。牛丼屋に行って、4回に3回、遅いかまずいか、料金がメニューよりも高いかだとすると、この店は確実につぶれますね。ソフトウェア業界も例外ではありません。これまでのような状況は許されない時代が来たのではないでしょうか。

● まだまだ意識が低い日本の情報サービス業

私はこの2年間、JASPIC（日本ソフトウェアプロセス改善コンソーシアム）とJISA（社団法人情報サービス産業協会）が主催するソフトウェアプロセス改善に関するカンファレンスを、時間の許す限り聴講してきました。そして、4回のカンファレンスの合計92件の論文を読んでみました。この結果、気が付いたことを列挙しますと、

- 日本のソフトウェア・エンジニアリングのレベルは低いといわれるが、これらのカンファレンスの論文はかなりのレベルに達しているのではないか。
- 聴講者と論文の著者の大半は企業のスタッフ部門の人であり、プロフィット部門の参画は少ない。まして、幹部の参加はゼロに近い。
- 論文では、技術やマネジメントの発表以外に、自社における組織の壁、部分最適化、決められたことの形骸化等、企業の組織や企業風土に関する指摘が多い。
- 発表企業や発表者がほぼ固定している。

等になります。

発表企業数は2年間で47社、ちなみに、SECのタスクフォースへの参加企業は現在59社であり、重複を整理すると両者で68社になります。恐らくこの数倍の会社がソフトウェアプロセス改善活動を活発に行っていると思われますが、日本の情報サービス産業の企業が１万社に近いことを考えると、まだまだ少ないといわざるを得ません。これが一番目の問題です。ソフトウェアにおいては、単純なミスでも大きな問題に発展することは明らかであり、インテル社がペンティアムプロセッサのたった3行のコーディングミスが原因で4億ドルの損害を被ったことは有名な話です。このように、ソフトウェア全体の品質向上のためには、コーディングを分担する企業を含めた、ソフトウェア・エンジニアリング活動の裾野の拡大が極めて重要な課題になります。

● スタッフ部門中心のソフトウェア・エンジニアリング活動

上述のカンファレンスで気がついたもう１つの大きな問題は、日本のソフトウェア・エンジニアリング活動がスタッフ部門中心に進められているのではないか、はっきりいうと、現場と遊離しているのではないかということです。確かに、ソフトウェア開発の現場は頻発するトラブルプロジェクトの火消しに追われており、このため、優秀なエンジニアが、優秀であればあるほど、目先の仕事に没頭せざるを得ない状況になっています。さらに、現場主義という名のもとに、事業部あるいはプロジェクト毎にプロセスやツール等が選定されているか、あるいは、全社的に決められていても形骸化して守られていない状況が見受けられます。多くの会社でスタッフの技術力や成果が生かされていないのです。

これでは、戦略的な全社最適による生産性の向上を目指しているインドや中国に勝つことはできません。上述のような状況を打破するには、スタッフの一層の努力とともに、日産のようなトップマネジメントの強いリーダーシップが必要ではないでしょうか。今回、SECの活動に対して59の企業から、約200人の技術者および1,000件を超える定量データを出していただきましたが、これはトップのリーダーシップが発揮された結果だと思います。これを契機に、各企業内および企業間において、個別・最適論理から脱却し全体最適を目指す動きが一層活発になってくることを期待したいと思います。

● ソフトウェア産業の発展に必要なこと

ソフトウェアの問題を解決するためには、ソフトウェア開発プロセスやツール、技法といった目に見える部分だけでなく、これと同期して、管理システム、業務プロセス、人材、研究開発等の目に見えない能力を強化しなければなりません。

ソフトウェアの問題は技術の問題であるとともに、経営の問題であることを強く認識しなければならないと思います。

独立行政法人 情報処理推進機構
ソフトウェア・エンジニアリング・センター
所長　鶴保　征城

SEC journal創刊記念　招待論文 invited paper

最新ソフトウェア技術による高信頼組込みソフトウェアの開発

Highly Reliable Embedded Software Development Using Advanced Software Technologies

北陸先端科学技術大学院大学
片山　卓也

Japan Advanced Institute of Science and Technology
Takuya Katayama

我々は文部科学省e-Society プロジェクトの一環として高信頼性組込みソフトウェア構築技術プロジェクトを推進している．本プロジェクトの目的は，最新のソフトウェア技術を産業界における高信頼な組込みソフトウェアのために活用することである．本稿では、プロジェクトの概要と現在までの成果について報告する．

We have launched “ Highly Reliable Embedded Software Development ” Project, held as a part of e-Society Project, supported by Ministry of Education, Culture, Sports, Science and Technology（MEXT）, Japan. The aim of this project is to enable the industry to produce highly reliable and advanced software by introducing latest software technologies into embedded software development. In this paper, we introduce the overview of the projects and our activities and results so far.

Key Words & Phrases ： Embedded software, Design verification, Operating environment, Real time Java
組込みソフトウェア，設計検証，オペレーティング環境，リアルタイムJava

1　はじめに

組込みソフトウェアは，家庭電気製品，自動車，携帯端末，各種制御機器等の心臓部に組込まれ，その機能や品質は製品や機器の価値を決める最も重要な要素であり，その開発技術を高く保つことは，我が国製造業の競争力維持にとって極めて重要である．

しかしながら現在では，高度なユーザインタフェース，セキュリティや通信機能等要求される機能が高級化する一方，CPU やメモリ等のハードウェア資源が高性能化してきたため，ソフトウェアの規模と複雑さが増大し，従来の開発手法が十分に機能しなくなりつつある．その結果，多額の経済損失を伴う製品のリコールや製品開発の遅れ等が生じている．

この問題を解決するには，組込みソフトウェア業界全体の技術レベルの底上げを図るとともに，最新のソフトウェア開発技術を導入し，高信頼化と高度化に対応しなければならない．本稿では，現在文部科学省e-Society プロジェクトで行われている研究開発を中心にして，高信頼組込みソフトウェアの開発への最新ソフトウェア技術の適用について述べる．

2　e-Society プロジェクトにおける組込みソフトウェア研究プロジェクト

文部科学省リーディングプロジェクト「e-Society 基盤ソフトウェアの総合開発」は2003 年度より開始され，8 大学および十数社の企業が緊密に連携し，（1）高い生産性をもつ高信頼ソフトウェア作成技術の開発，（2）情報の高信頼蓄積・検索技術の開発，を推進している．本プロジェクトでは，産業界からのニーズに基づき，大学等がもつ研究機能，人材養成機能を最大限活用し，社会の基盤となるソフトウェアの研究開発と研究者養成を一体的に推進している．

このプロジェクトの大きな特徴は，大学と企業が密接に連携し，現実問題の解決を目指して研究開発を推進していることである．大学での研究成果を現実のソフトウェア開発に適用する枠組みができた意義は非常に大きく，プロジェクト開始後実質1年半が経過した段階であるが，製品システムへの組込み等も一部には行われる段階に達している．今後の成果が大いに期待できると同時に，この貴重な機会を生かすべく研究開発を一層進めるつもりである．現在，e-Society プロジェクトにおいては，高信頼ソフトウェアの開発に関して７つの課題についての研究が行われている（http://cif.iis.u-tokyo.ac.jp/esociety/）．このうちの1つが「高信頼組込みソフトウェア構築技術」であり，(1) 高信頼組込み用オブジェクト設計技術（北陸先端科学技術大学院大学）(2) 組込みシステム向け基盤ソフトウェア（早稲田大学）(3) 組込み用実時間Java 技術（京都大学）の3つの観点から連携をとりつつ研究を進め，次世代高信頼組込みシステム技術の雛形となる一貫性のとれたシステムの実現を目指している．以下，これらの研究開発の概要とこれまでの研究成果等を紹介する．

3　高信頼組込み用オブジェクト設計技術プロジェクト

3.1　研究の概要

本プロジェクトでは，開発の上流工程にソフトウェア開発の最新の科学的工学的成果を適用することによりソフトウェアの信頼性の向上に取組んでいる[1]．特に（1）宇宙，航空，軍事といった分野のみで利用されてきた形式検証の民需分野組込みソフトウェア開発への適用，（2）アスペクト指向開発，プロダクトライン開発等のソフトウェア工学的成果の組込みソフトウェアの開発への適用，(3) オブジェクト指向技術の組込みシステムへの適用を困難にしてきた実時間制約問題の解決に焦点を当てて研究を進めている．

3.2　研究体制

本研究は北陸先端科学技術大学院大学を中心とする，以下のメンバによって行われている．

- 北陸先端大：片山卓也（教授），岸知二（産学官連携客員教授），青木利晃（助手），岡崎光隆（ポスト・ドクター），博士課程学生３名
- 国立情報学研究所：中島震（教授）
- 参加企業：日本電気株式会社

3.3　UML 設計モデルへのモデル検査技術の適用

組込みソフトウェアでは，物理世界で発生し得る様々なイベントの組合せや順序を考慮する必要があるが，そうした網羅的な確認を確実に行うために，モデル検査技術が有効である．モデル検査技術は，有限状態システムが望ましい動作をすることや，望ましくない振舞いをしないことをシステムの状態空間を調べ尽くして確認する技術である．ハードウェアの検査ではすでに有効性が確認され，実際に活用されている．

一方ソフトウェアでは，状態空間が非常に大きく設計の抽象化等を行わないと状態爆発を起こすこと，モデル検査特有のツールや言語の習得が必要なこと等が，実用化の障壁となっている．これらを解決するために，(1) UMLでの設計にモデル検査を適用するためのツールや環境の開発，(2) 検証結果や検証モデルの再利用，(3) アスペクト指向での検証モデリング等の検討を行っている．

3.3.1　UML モデル検査環境

開発中のUML 設計モデル検査ツール（図1）は，Eclipse プラットフォーム上に構築され，UMLのモデリングにはUMLプラグインを，モデル検査にはSPIN を利用している．本ツールは，UMLモデルと，UML中の定義を参照した論理式（LTL式）を与えると，それをモデル検査のための言語（Promela言語）と，Promela中の定義を参照したLTL式に変換してモデル検査を実行し，結果を再度UMLの概念に変換して表示する機能をもつ．これによりPromela言語に関する詳細を知らなくてもUMLレベルでのモデル検査が可能になっている．現在，評価バージョンが完成しており，評価のための配布が可能な状態である．

3.3.2　検証モデルの再利用

モデル検査技術の適用は時間的，要員的にコストがかかるため，いったん構築した検証の枠組みを再利用することが望まれる．民需分野の組込みソフトウェアの多くがプロダクトライン開発であることを考えると，そこでの再利用が有効であると考えられる．

プロダクトライン開発においては再利用資産を体系的

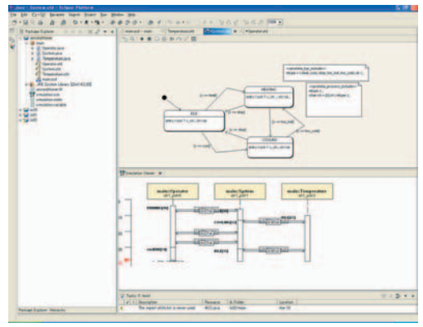

図1　UML 設計モデル検査ツール

に管理することが重要となるが，我々は検証に必要なテストシナリオ（を表現した状態モデル）や，LTL式等を再利用可能な形で管理し，それらと要求事項や設計等との間にトレーサビリティを定義することで再利用を可能とすることを検討している[2].

現在までに，検証モデル（テストシナリオに対応するモデルや論理式）を再利用可能とする方法について検討をするとともに，その再利用性を具体例に基づいて評価してきた．今後さらにツール支援の強化等実用化に向けた検討を進めていく予定である．

3.3.3 アスペクト指向による検証モデリング

UMLモデルをモデル検査等に適用するためには，検証目的に対して必要十分な厳密性をもったモデルを記述する必要がある．一方，設計はあくまで人間が行う行為であるから，設計モデルは理解容易でなければならない．この2つの要求を満たすため，アスペクト指向による検証指向のモデリングを検討している[3].

アスペクトとは，対象システムを特定の関心事や局面から射影し切出したものであり，それによりソフトウェアを関心事毎に独立性高く理解，構築，修正できる．設計検証をアスペクト毎に行うことにより，全体の検証コストを下げることが期待される．我々はいくつかの事例に関して実験を行い，この方法の実現可能性を評価すると同時に，モデル検査をアスペクト毎に増分的に実行するための理論的研究を行っている．

3.4 設計検証の役割

設計に検証技術を適用する際に問題になることは，設計文書の意味論である．一般に設計文書にはシステムの実装条件やプラットフォームに関する情報が明確には与えられない．特にUMLのステートチャート図では，イベントの配送メカニズム等の全貌に対する明確な意味論が整備されておらず，設計者は意味論の詳細を意識せずに記法を使うことが多く，検証を困難にしている．

検証のための意味論を与える際には，どのような不具合を想定しているかを明確に意識する必要がある．検証目的を一意に決めることは困難なため，そのための意味論も詳細な目的に応じて設定していく必要がある．現在，いくつかの実例の検討に基づき，現実的な枠組みの中でモデル検査を行う場合について，意味論の表現や選択法に関する研究を行っている．

3.5 UMLモデルの形式化と形式検証，動作解析

ソフトウェアの大規模化に伴い，そのコードであるプログラムを直接的に扱うことが量的に困難になり，ソフトウェアを抽象的に記述した分析モデルや設計モデルの重要性が大きくなっている．ここでは設計に科学的手法を適用する際の技術的課題として，定理証明技術の適用や協調並行動作の解析に関して触れる．

3.5.1 UMLモデルの実行と不変性検証

3.3項で述べたように，モデル検査は網羅的検査等には優れているが状態爆発等の問題をもっている．設計抽象化により状態空間を縮小できるがこれは検査結果の不正確性をもたらす．正確な検証には定理証明システム等による検証やモデル実行によるテストが必要になる．我々はUMLの実行のための操作的意味論を与え，それを関数型言語処理系SML/NJ を用いて実行する環境を作成し，特定の実行系列やテストケースに対する振舞いの解析等を行ってきた．

さらに完全な正しさを保障するために，プログラム検証と同様の手法を用いてUMLモデルのオブジェクトの属性に対する不変的な性質を保証する検証手法を提案し[4]，定理証明システムHOLを用いて実際に証明を行う環境F-Developer を用い，リアクティブシステムの検証実験を行っている．このシステムは現在一般配布が可能である．

定理証明システムでは対話的に証明を行うため検証コストが高くつく．そこで我々は証明の再利用と段階的な正しさの保証により，検証コストを削減する手法の研究を行っている[5]．また，複数のオブジェクトに関する検証ではオブジェクトの直積を構成するために複雑になる可能性があるので，必要最低限な協調関係に関する情報を用いて振舞いを近似して検証する手法[6]や，個々のオブジェクトの振舞いではなく，協調関係をベースに検証する手法の研究を行っている[7].

3.5.2 協調並行動作の解析

実時間性の保障に用いられるスケジューリング理論等の手法では特定の条件を満たす実行系列，すなわちタスクを構成する．このタスクはスケジューリング理論に起因する制約を満たす必要がある．オブジェクト指向設計では，実行系列は一連のメソッド呼出し等で表現されるが，個々のオブジェクトの定義からこれらが明確に見えず，その適用を困難にしている．

我々はオブジェクト指向開発でタスクの設計を行うためのモデルを提案してきた[8]．そこでは様々な実行の組合せから実行系列を獲得する必要があるが，それを人手で行うのは困難であるため，並行正規表現を用いて並行オブジェクトを表現し，その等価変換によってオブジェクト指向設計モデルからタスクを系統的に構成するための枠組を提案した[9]．この方法に基づいて，タスクを自動的に構成するための手法の研究を行っており[10]，得られたタスクに実時間解析やスケジューリング理論を適用することを計画中である．

4 高信頼組込みシステムのためのオペレーティングシステムプロジェクト

4.1 はじめに

携帯電話やデジタルテレビ等の組込み機器開発は現在大きな転換点を迎えている．従来の組込みシステムでは，ソフトウェアはハードウェアと比べてはるかに単純であったが，現在では立場が逆転し，ソフトウェアが組込み機器の付加価値を決定する重要な地位を占めている．それとともに，組込みシステム向けのソフトウェアの規模は巨大化しソフトウェアバグやセキュリティホールによる障害を発生する可能性があり，ソフトウェアの信頼性を向上するための新しい基盤ソフトウェアが必要となってきた．今後，組込みシステムのインターネット接続が当たり前になっていくので，組込みシステムが基幹産業の1つである日本では，ソフトウェアの信頼性の強化は政府が積極的に取組むべき研究課題である．

ユビキタス時代の到来により日常品にコンピュータを埋込み，情報を扱う様々な情報アプライアンスが出現するものと思われる．情報アプライアンスは単体で機能するのではなく，組合せることにより様々な新しいサービスを作りだす．しかし，情報アプライアンスのソフトウェアは巨大となり，ソフトウェアのバグによりシステム全体の信頼性の保証が困難となる．また，情報アプライアンスはインターネットに接続されるため，セキュリティに対する脅威からシステムを守る必要がある．組込みシステムは従来のエンタプライズシステムとは異なり，リソースの制約を考慮する必要があるため，従来の手法と異なる新しい技術の開発が必要となる．

次世代組込みシステムの開発ではバグを含む可能性がある多くの既存ソフトウェアの再利用が必要である．既存のソフトウェア資産の有効利用はコスト削減，開発期間の短縮化のために極めて重要である．そのため，C，C++言語等で記述された、バグが存在する可能性がある既存のソフトウェアを安全に動かすためのオペレーティングシステムのサポートが必要不可欠なものとなる．形式的手法は今後高信頼組込みソフトウェアを開発するための技術として徐々に使われていくと思われるが，新規のアプリケーション開発の場合も，多くのバグを含んでいる可能性がある既存のライブラリを利用する必要があるため，オペレーティングシステムレベルでの支援なくして組込みシステム全体の信頼性を向上することは不可能である．

4.2 研究体制

本研究は早稲田大学を中心とする以下のメンバによって行われている．

- 早稲田大学：中島達夫（教授），追川修一（客員助教授，現筑波大助教授），博士課程学生3名
- 参加企業：ノキア・ジャパン株式会社，松下電器産業株式会社

4.3 高信頼組込みシステムを構築するためのオペレーティングシステム

現在，早稲田大学のグループでは，組込みシステムの信頼性を向上する2つのオペレーティングシステム開発に取組んでいる．1つ目の取組みは，Linux のCPU リソース管理の強化である．2つ目の取組みは，μカーネルに基づき信頼性やセキュリティを向上するOS である．

4.3.1 組込みLinux におけるQoS サポート

現在のLinux は，悪意のあるプログラムをダウンロードして実行すると，そのプログラムがシステムリソースを占有することにより，システム全体が正しく動かなくなってしまう可能性があり得る．また，マルチメディアアプリケーション等のリアルタイムアプリケーションがCPUを占有すると，非リアルタイムプロセスとして実行されるGUI等の処理が動かなくなってしまう可能性がある．我々のプロジェクトにおいて開発中のLinuxは，オリジナルのLinux にCPUアカウンティング機構と階層型スケジューラを追加することにより，リソース占有を防ぐことを可能とする[11][12].

アカウンティング機構を導入した場合，各プロセスはアカウンティング機構が管理するアカウンティングオブジェクトにバインドされる．アカウンティングオブジェクトは周期と実行時間というパラメータをもっている．実行時間とは，各周期内で各プロセスが利用可能なCPUキャパシティである．各周期内に指定した実行時間を超過してプロセスを実行しようとした場合，次の周期が始まるまでアカウンティングオブジェクトがバインドされているプロセスの実行を中断する．これにより，アカウンティング機構は，ダウンロードしたアプリケーションのCPU の占有を防ぐことを可能とする．また，各システムにはプログラムのダウンロードを管理するマネジャプロセスが存在する．マネジャプロセスは，新しいアプリケーションをダウンロードして起動するときに，マネジャプロセスが所有するアカウントオブジェクトにバインドする．このアカウンティングオブジェクトはダウンロードしたアプリケーションが利用可能なCPU キャパシティが指定されている．アカウンティング機構は，新しく生成したプロセスを親プロセスにバインドされたアカウンティングオブジェクトに自動的にバインドする．以上により，ダウンロードしたアプリケーションがプロセスを生成することによりCPU キャパシティを占有しようとしても，指定された以上のCPU キャパシティを利用できないことを保証する．

階層型スケジューラはリアルタイムプロセスが起動中でも，ある程度のCPU リソースをタイムシェアリングスケジューラが実行するプロセス（タイムシェアリングプロセス）が使用することを可能にする．

通常，マルチメディアアプリケーションのマルチメディア処理部分はリアルタイムプロセスとして実行されるが，GUI部分はタイムシェアリングプロセスとして実行される．このとき，リアルタイムプロセスがCPUリソースを占有してしまうと，タイムシェアリングプロセスが処理するGUI のイベントを実行できなくなる可能性がある．そのため，階層型スケジューラを利用することにより，常にある一定のCPU キャパシティをタイムシェアリングプロセスに割当てることができるようになる．

現在，アカウンティング機構と階層型スケジューラのプロトタイプ実装は終了し，モンタビスタソフトウェアジャパンの協力を得て，商用の組込みLinux として利用可能となるように実装の改良を進めている．また，カーネルインタフェースに関しては日本エンベデッドリナックスコンソーシアムのリソースマネジメントワーキンググループにおいて標準化が進められている．

4.3.2 μカーネルに基づくオペレーティングシステム

組込みシステムでは，外界からのイベントにタイムリに反応するため，一定時間内に応答を返すことが重要なアプリケーションが多数ある．組込みLinux はそれらの要求に対応するため，カーネルを横取り可能にする等の改良が行われてきた．しかし，μ秒オーダの高速な応答性を達成することは難しい．そのため，Linux on ITRONのようなリアルタイムオペレーティングシステム（RTOS）上でLinux を動かすハイブリッドアーキテクチャが開発されてきた．しかし，従来のハイブリッドアーキテクチャは，RTOSやRTOS 上のアプリケーションがLinuxカーネルのカーネルスペース内で動作するため，RTOS 上のアプリケーションのバグによりシステム全体がクラッシュしてしまう可能性がある．コード量の増大によりソフトウェアのバグも増加するため，バグが存在してもシステムが動作し続けるような仕組みをオペレーティングシステムが提供することは，ますます複雑化する組込みシステムを開発する上で非常に重要なことである．

我々は，図2 に示すようなμカーネル上に複数のオペレーティングシステムを同時に動作させることを可能とするシステムの実装を現在進めている[12][13][14]．本システムでは，*TL4* マイクロカーネルというスレッド，アドレススペース，プロセス間通信等の低レベルな抽象化を提供する小型のオペレーティングシステム上でμITRON カーネルやLinux 等の従来のオペレーティングシステムを動作することが可能となる．

また，複数のμITRON カーネルを異なるアドレススペース上で実行することも可能であり，μITRONカーネル上のアプリケーションのバグによる障害を他のアプリケーションに伝播させないことを可能とする．つまり，独立に開発したアプリケーションを，異なるμITRONカーネル上で動作させることにより，お互いのバグ等による障害の影響を隔離する．

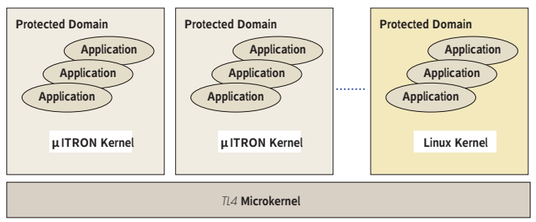

図２　μカーネルに基づくOS

複数のLinux を同時に起動し，ダウンロードしたアプリケーションの実行をその他のアプリケーションと異なるLinux 上で実行することでお互いの実行を隔離することにより，セキュリティや信頼性を格段に向上することが可能となる．例えば，ダウンロードしたアプリケーションがファイルシステムを破壊しようとしても，そのアプリケーションが動作しているLinuxカーネルが提供するファイルシステム以外のファイルシステムはアクセスすることが不可能となる．また，あるLinux カーネルに不都合が生じた場合は，ユーザが気づかないうちに，そのLinux カーネルを再起動することにより不都合から回復することを可能とする．しかし，オペレーティングシステム全体を再起動するコストは小さくない．そのため，細粒度なコンポーネント毎に再起動可能にする回復可能コンポーネントをμカーネル上に実装中である．回復可能コンポーネントを利用して，オペレーティングシステムの一部がクラッシュしても，オペレーティングシステム全体を再起動することを防ぐことが可能となる．

現状のプロトタイプシステムでは，複数のμITRONカーネルを*TL4* 上で動作することが可能である．また，簡単な性能評価の結果，μカーネルを利用するオーバヘッドは極めて小さいことが確認されている．現状では，μITRON，Linux 等の複数のOS を動作させるためのデバイスドライバフレームワークの開発や，複数のOS を実行するときのCPU キャパシティを指定することを可能とするリソースマネジメント機構の開発を行っている．

4.4 まとめ

次世代の組込みシステムを構築するためには，現在，組込みオペレーティングシステムとして広く利用されているμITRON やLinux は不十分である．どのように振舞うかが明らかではないサードパーティのアプリケーションやバグを含んでいる可能性がある大規模ソフトウェアを動作させても，システムが不安定な状態にならないようにするリソース管理機構や，ソフトウェアのバグを隔離するためのオペレーティングシステムのサポートが必要不可欠である．本節では，早稲田大学で取組んでいる2 つのオペレーティングシステムのプロジェクトに関して簡単に紹介した．以上述べた成果に関しては，今後も様々な企業と共同で開発を進めることで商用システムとして利用可能とし，可能な部分に関してはオープンソースとして公開することで組込みシステム研究の促進にも貢献することを検討している．

5 組込みシステム用実時間Java技術プロジェクト

本プロジェクトは，Java によって記述する組込み実時間アプリケーションの開発を効率化するための諸技術を開発することを目的として，実行基盤の開発，実証実験，要求仕様検証技術との統合を行うものである．平成15 年度と16 年度は主として，実時間組込みソフト固有の開発コストを軽減するための，自動化機能を備えたJava 処理系実装方式の開発と実装を行っている．この節では，このうち実行時間の大幅な改善に成功した実時間ごみ集めの実装について報告する．

Java はオブジェクト指向言語であり，アプリケーションの実行中に，データを動的に生成する．実行が進むと，メモリ領域が満杯となるために，不要なデータを回収して再利用する「ごみ集め」の処理が必要となる．通常のJava 処理系の場合は，アプリケーションを一時的に中断し，ごみ集め処理を行うが，この方法では，中断時間の予測が難しくアプリケーションの実時間性を保証できない．また，メモリ管理の方法によっては，データの生成時にも実時間性が問題となることがある．この問題に対応するために，ごみ集めの対象とならないメモリ領域を利用して一括廃棄するといった拡張機能を利用することがある．しかし，このようなプログラミングは，基本的に手作業で行われ，バグの原因となる可能性が高いうえに，作業工数が増加するという問題がある．これらを解決するために，本プロジェクトでは，拡張機能を使わないでも実時間性が保証できる実装技術を開発している．

5.1 研究体制

本研究は京都大学を中心とする以下のメンバによって行われている．

- 京都大学：湯淺太一（教授），八杉昌宏（助教授），馬谷誠二（ポスト・ドクター），博士課程学生２名
- 参加企業：オムロン株式会社，オムロンソフトウェア株式会社

5.2 基本アルゴリズム

マーク・スイープ方式のごみ集め処理を実時間化する

ために，京都大学の湯淺らのグループで開発してきた「スナップショットごみ集め」および「リターンバリア」を基本アルゴリズムとして採用した．アプリケーションの実時間性を保証するためには，ごみ集め処理を一括して行わないで，小さな単位に分割し，アプリケーションの実行中に少しずつ進めればよい．しかし，単純に処理を分割しただけでは，アプリケーションの実行によって，使用中にもかかわらずマークされないデータが生じる可能性がある．スナップショット方式ごみ集め[15][16]では，アプリケーション実行中に「書き込みバリア」という機構を挿入することによって，ごみ集め開始時点で使用中であったデータをすべてマークすることを保証する．

当初のスナップショット方式ごみ集めでは，ごみ集め開始時にルートの走査を一括して行う必要があった．ルート領域にはスタックを含み，プログラムの実行によっては，スタックの走査にかなりの時間を要し，その間だけは実時間性が保証できなかった．これを改良するために開発されたのがリターンバリア[17][18]である．スタック走査を単純に分割して実行すると，マーク処理と同様に，使用中のデータがマークされない可能性がある．そこで，走査ずみのスタックフレームにバリアを設置し，メソッド（あるいは関数）がバリアを越えてリターンしようとする際に，走査を一定量進める．これにより，スタック走査中は，アプリケーションが走査ずみのスタック領域のみを参照することを保証し，マークもれを防ぐ．

5.3 JeRTy VM

今回実装の対象としたJava 処理系は，本プロジェクトの共同研究者であるオムロン が開発し，実際に組込みシステムの基盤として応用実績のあるJeRTy VM[19][20]である．JeRTy VM では，組込みシステムに要求される実時間性を保証するために，中断・再開可能なマーク・スイープ方式ごみ集めを実装している．ごみ集めは，１本のスレッドとして実装されており，アプリケーションの実行と平行して少しずつ進められる．アプリケーションスレッドとごみ集めスレッドとの切替えは，一定時間毎にごみ集めスレッドの優先度を上げる（アプリケーションスレッドの優先度より高くする）ことによって行われる．

ごみ集めの基本アルゴリズムは，On-the-fly 方式と呼ばれるものである．この方式は，スナップショット方式と比較すると実行速度が劣るものの，スタック走査を一括して行う必要がないために，リターンバリアのないスナップショット方式よりは，実時間性に優れている．ただしJeRTy VM では，マーク処理を確実に行い，ライトバリア処理の速度を向上させるために，スタック走査を一括して行っている．On-the-fly方式もライトバリアを必要とするので，ごみ集めをスナップショットに移行する際に最も工数のかかるライトバリアの挿入作業は不要であった．そこで，今回の実時間ごみ集め実装の最大の課題は，リターンバリアの実装であった．

5.4 リターンバリアの実装

オリジナルのJeRTy は，マーク処理のための各アプリケーションスレッドのルート走査のために，スタック全体にロックをかけていた．このために，ルート走査が終了するまでアプリケーションスレッドが停止する．今回の実装では，スタックトップからフレームを一定個数ずつ走査し，最後に走査されたフレームにリターンバリアを設定することとした．この間，スタックがロックされるが，その時間は予測可能であり，一度に走査するフレーム数を調整することによって，時間の調整を行うことも可能となる．

JeRTy では，メソッドがリターンするときに必要となる戻り番地として，そのメソッドを呼出したバイト命令へのポインタをフレームに格納している．メソッド呼出し用のバイト命令は，その種類によってオペランド長が異なるので，JeRTy ではリターン時に「戻り番地」の指す命令によって，リターン後にどの命令から実行を再開するかを決定している．

今回の実装では，リターンバリアを，バイト命令へのポインタにはなり得ない特殊な値で表現した．戻り番地を格納しておく位置にリターンバリアを格納することによって，そのフレームまで走査が終了していることを表す．本当の戻り番地は，アプリケーションスレッド毎に保管用領域を用意し，そこに保存しておく．メソッドからリターンする際に，戻り番地がリターンバリアでなければ普通にリターンするが，リターンバリアであれば，ごみ集めスレッドとの排他制御を行った上で，フレームの走査を数フレーム分行い，リターンバリアの位置を更新して，予め保存されている戻り番地へリターンする．

今回のJeRTy への実装による処理系の変更箇所は，新規追加が約60 行，変更が約50 行であった．ごみ集め処理ルーチンが全体で2,400 行であることを考慮すると，極めて小さな変更であった．

5.5 評　価

実装した実時間ごみ集めの性能評価を行った．Java用のごみ集め評価に適したベンチマークは少ないので，Java で記述したLisp 処理系であるJAKLD[21]をJeRTy 上で動作させ，Gabriel のLisp 用ベンチマーク[22]も性能計測に使用した．ベンチマークテストが使用するデータに加え，JAKLD 処理系自体が内部的にデータを使用するので，ひんぱんにごみ集めを行う．ここでは，Java で記述されたマージソートプログラムSORT とGabriel ベンチマークのDIVとFFTについて，計測結果を挙げる．

プログラムの実行時間を表1に示す．リターンバリアを組込む前の「従来版」と，今回リターンバリアを実装した「改良版」における実行時間および従来版との比率を示す．表1からわかるように，改良版では，30％程度の大幅な速度向上となった．これは，スナップショット方式に移行することによって不要となったライトバリアの実行時間が大きく影響している．

表1　改良版と従来版との実行時間比較

	従来版（ms）	改良版（ms）	従来版との比（%）
SORT	160,965	116,221	72.2
DIV	32,602,733	23,960,749	73.5
FFT	96,451,432	62,644,597	64.9

ごみ集め実行にともなう割り込み禁止時間の最大値を表2 に示す．従来版が数ミリ秒から数十ミリ秒だったのに対し，改良版では１ミリ秒を下回り，大幅に減少している．これは，ルート挿入を分割して行うようになったためであり，今回の実装が，実時間処理に適した実装となっていることがわかる．

表２　割込み禁止時間の最大値

	従来版（ms）	改良版（ms）
SORT	2.6	0.2
DIV	20.0	0.4
FFT	4.2	0.5

メソッドがリターンする際にリターンバリアに遭遇した回数と，その際にフレーム走査に要した時間の合計を表3に示す．プログラム実行中に，ごみ集めは数百回から数千回のオーダで行われており，リターンバリアに遭遇する可能性は極めて低いことがわかる．また，フレーム走査のためのアプリケーション停止時間は，実時間性を損なわない範囲に収まっている．

表3　リターンバリアとの遭遇回数とフレーム走査時間

	実行回数	総時間（ms）
SORT	1	0.1
DIV	114	31.1
FFT	47	15.9

以上のことから，今回実装したJeRTy の実時間ごみ集め処理は，当初の目的を達成するものであると結論づけることができる．

参考文献

[1] Tomoji KISHI, Toshiaki AOKI, Shin NAKAJIMA,Natsuko NODA and Takuya KATAYAMA：Project Report: High Reliable Object-Oriented Embedded Software Design，The 2nd IEEE Workshop on Software Technology for Embedded and Ubiquitous Computing Systems（WSTFEUS'04），pp.144-148, 2004.

[2] Tomoji KISHI and Natsuko NODA： Design Testing for Product Line Development based on Test Scenarios，International Workshop on Software Product Line Testing,（SPLiT 2004），pp.19-26, 2004.

[3] 岸知二，野田夏子：アスペクト指向による状況モデリング情報処理学会ESS2003，組込みソフトウェアシンポジウム2003 論文集，pp.22-29，2003.

[4] 青木利晃，立石孝彰，片山卓也：定理証明技術のオブジェクト指向分析への適用，日本ソフトウェア科学会，コンピュータソフトウェア，Vol.18, No.4, pp.18-47，2001.

[5] Toshiaki Aoki and Takuya Katayama：Foundations for Evolutionary Construction of State Transition Models，International Workshop on Principles of Software Evolution 2004，pp.143-146，2004.

[6] 立石孝彰，青木利晃，片山卓也：振舞い近似手法を用いたステートチャートに対する不変性の検証，情報処理学会論文誌，Vol.44，No.6，pp.1448-1460，2003.

[7] Kenro Yatake，Toshiaki Aoki and Takuya Katayama：Collaboration-based verification of Object-Oriented models in HOL，Verification and Validation of Enterprise Information Systems，pp.78-80, 2004.

[8] 青木利晃，内藤壮司，片山卓也：オブジェクト指向組込みシステム開発のためのSes-Based アプローチ，日本ソフトウェア科学会，コンピュータソフトウェア，Vol.16, No.2，pp.67-71，1999.

[9] Mitsutaka Okazaki，Toshiaki Aoki and Takuya Katayama：Extracting threads from concurrent objects for the design of embedded systems，Asia-Pacific Software Engineering Conference 2002，pp.107-116，2002.

[10] 岡崎光隆，片山卓也：並行オブジェクトシステム動作解析のための実行スレッド自動抽出，ソフトウェア科学会DSW04，第1 回ディペンダブルソフトウェアワークショップ論文集，pp.85-94，2004.

[11] S. Oikawa，T. Nakajima：Resource Management Architecture for Future Information Appliances, Journal of Embedded Computing，Cambridge International Science Publishing，Vol.1，No.1，2004.

[12] Shuichi Oikawa，Midori Sugaya，Masatoshi Iwasaki and Tatsuo Nakajima：Using Virtualized Operating Systems as a Ubiquitous Computing Infrastructure，In Proceedings of 2nd IEEE Workshop on Software Technologies for Embedded and Ubiquitous Computing Systems（WSTFEUS 2004），pp.109-114，2004.

[13] Shuichi Oikawa，Hiroo Ishikawa，Masatoshi Iwasaki，Tatsuo Nakajima：Constructing Secure Operating Environments by Co-Locating Multiple Embedded Operating Systems，2nd IEEE Consumer Communications and Networking Conference（CCNC 2005），2005.

[14] Tatsuo Nakajima，Midori Sugaya，Shuichi Oikawa：Operating Systems for Building Robust Embedded Systems，In Proceedings of the tenth IEEE International Workshop on Object-oriented Real-time Dependable Systems（WORDS2005），2005.

[15] T. Yuasa：Real-time garbage collection on general-purpose machines，The Journal of Systems and Software，Vol. 11，No. 3，pp.181-198，1990.

[16] 湯淺太一：実時間ごみ集め，情報処理，Vol. 35，No. 11，pp. 1006-1013，1994.

[17] 湯淺太一，中川雄一郎，小宮常康，八杉昌宏：リターン・バリア，情報処理学会論文誌，41 巻SIG9（PRO 8）号，pp.1-13，2000.

[18] Taiichi Yuasa，Yuichiro Nakagawa，Tsuneyasu Komiya，and Msahiro Yasugi：Return Barrier，Proceedings International Lisp Conference，San Francisco，2001.

[19] オムロン株式会社，http://www.jerty.com/

[20] 廣野光明，栗林博：JeRTy の組込み・リアルタイム機能とその応用，システム/制御/情報，Vol.30，No. 5，pp. 61-67，1998.

[21] 湯淺太一：Java アプリケーション組込み用のLisp ドライバ，情報処理学会論文誌，44 巻SIG4 （PRO 17）号，pp. 1-16，2001.

[22] Richard P. Gabriel：Performance and Evaluation of Lisp System，MIT Press Series in Computer Science，MIT Press，Cambridge，MA，1985.

エンタプライズ系ソフトウェア開発の課題

エンタプライズ系ソフトウェア開発力強化推進タスクフォース　総合部会　副主査委員／
独立行政法人 情報処理推進機構　ソフトウェア・エンジニアリング・センター　エンタプライズ系プロジェクト　プロジェクトサブリーダ
石谷　靖

本稿では、エンタプライズ系ソフトウェア開発について、これまでの長い取組みの歴史を踏まえながら現在抱えている課題とその解決の方向をまとめる。技術的な課題もさることながら、ソフトウェア開発に関する共通の「相場観」や「ものさし」が欠けていることを背景にしたマネジメントの課題がプロジェクトの成否に大きな影響を及ぼしている。一方、これまでの研究や企業の現場での実践に数多くの有益な知見が蓄積されており、その共有と適切な活用が重要であると考えている。

1. エンタプライズ系の背景と目的

エンタプライズ系ソフトウェアとは、ビジネスアプリケーション等の、主に企業活動を対象とした顧客対応システムの開発を指している。この分野では、既に30年近くのソフトウェア工学の研究開発がなされ、展開されている一方、激しい技術の進展やソフトウェア及びその開発が「見えない」ことに起因するマネジメントの難しさをいまだ大きな課題としている。実際、ソフトウェア開発プロジェクトの失敗事例は増加こそすれ、減少する様子は見られない。特に問題と考えられるのは、長年抱えてきた原因を解決することなく、同様の失敗を繰返していることである。

昨今、システムの大規模化や社会的なインフラにソフトウェアがますます入りこんでいるところから、さらなる品質向上が求められており、ソフトウェア産業全体が抱え続けている問題を改善することによって、開発のリスク低減と効率アップの必要性が高まっている。一方、現実は、ソフトウェア開発企業でいまだその場限りのアドホックな開発やマネジメントがなされ、ムダ・ムラが残っている。また、何よりもプロジェクトの成否が属人的である場合が多い。さらに、ソフトウェア開発のさまざまな点で、ユーザとベンダ間に認識に大きなギャップがあり、ムリ・ムダの原因となっている。

例えば、初期の段階では見積等の相場観にギャップがありムリな仕事となることが多い。同じソフトウェア開発に対して、ユーザは低めに見積り、ベンダはリスク等を勘案して多めに見積る。その間のギャップは、数倍や桁レベルでの違いになっている場合もある。そしてギャップがあるまま受注し、ムリなプロジェクトになってしまうことが少なくない。また、ソフトウェアの仕上がりに対する期待にお互いギャップを残したまま、開発終盤に手戻りが発生する等のムダにより失敗につながることも多い。結果的に、ユーザはもちろんのこと、ソフトウェア開発に関わるマネージャ・技術者等、関係者全員を不幸にしている。

以前からこれらは大きな課題として認識されていたが、根強くソフトウェア業界に存在し続け、昨今ソフトウェアが社会インフラとして広がるに伴い、一層解決が求められているところである[1]。

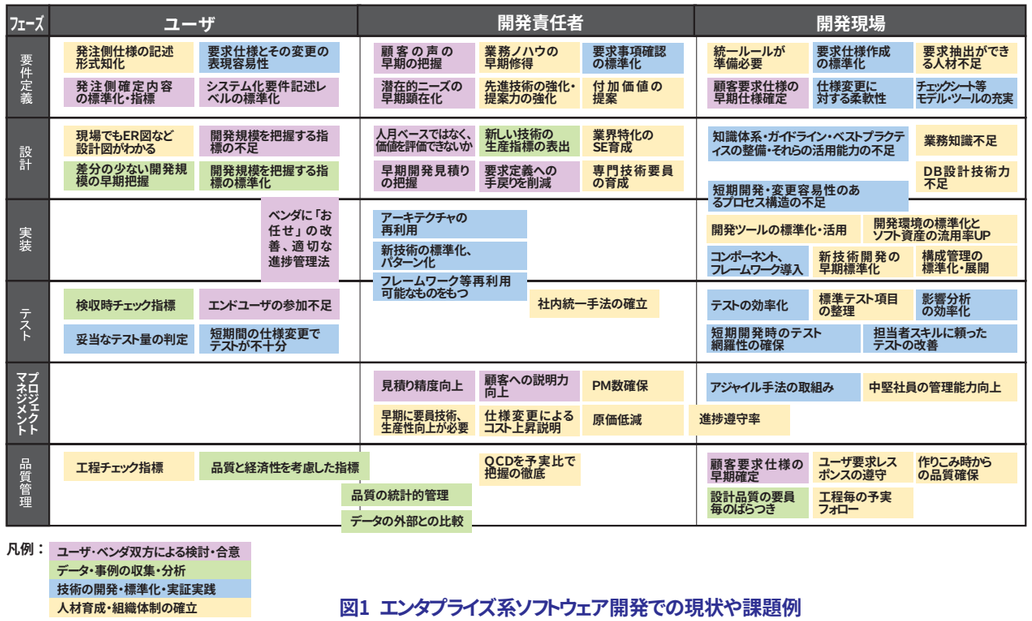

図1 エンタプライズ系ソフトウェア開発での現状や課題例

2. エンタプライズ系の現状と課題

課題としては、上記のとおり顧客とのコミュニケーションやマネジメント的な側面での課題が背後にあり、既存技術や先進技術の適切な活用がうまくいっていないことが基本的な構造である。

図 1に、エンタプライズ系ソフトウェア開発での具体的な課題例を示す。エンタプライズ系企業の参加するソフトウェア・エンジニアリング・センターの研究タスクフォースで意見を集約したものである。分類として、ソフトウェア開発への関わり方（ユーザ、開発責任者（マネージャ）、開発現場（開発者））と、ソフトウェア開発のフェーズで分類している。技術的な課題とマネジメント的な課題さらに慣習的な課題、そして人材に関する課題が全般に広がっている様子がわかる。

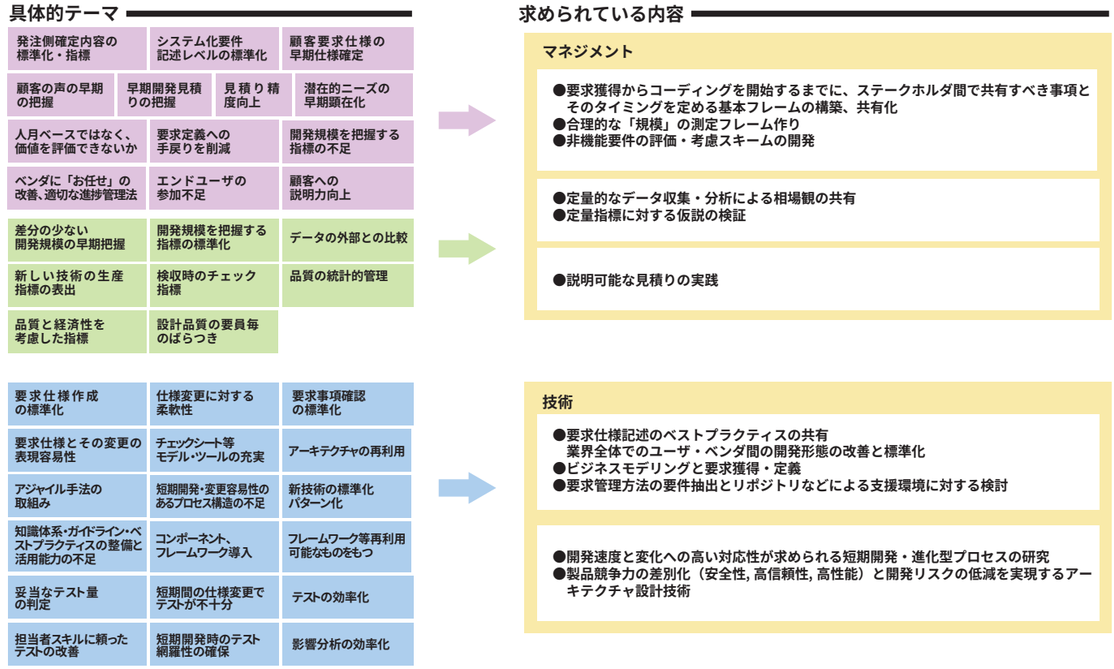

図 2 エンタプライズ系ソフトウェア開発で求められているもの

3. 求められる取組み

3.1 課題の整理

図 1の課題の中から、エンジニアリングテーマと関連付けたものが図 2である。全体的な傾向を見ると、要求・要件定義の課題、プロジェクトの全体規模の見積の課題、プロジェクトの定量的な把握の課題、短期開発への対応の課題に分けることができるが、それぞれに大きく関わる課題として、ユーザ・ベンダ間の認識のギャップが浮彫りにされる。要求定義では、ユーザ･ベンダの両者の共同的な作業が本質であり、要求を明確にすることで妥当な見積りが可能となるが、実際にはユーザのコミットが得られない、ベンダがユーザのニーズを汲取れないことが多い。また、見積りに当たっては、ユーザ・ベンダの両者が規模感等に共通の感覚を持っていないとお互いの納得が得られない。短期開発では、ユーザのライフサイクル全般を通じたコミットメントが本質である。

一方、技術的な面から見ても、要求の獲得・記述を始め、再利用技術・設計技術等の開発に直接関わるものから、見積手法、データ分析を含むマネジメント手法について「うまくいっていない」という感覚が課題として挙げられる。ここでのカギは、アドホックな方法から脱却して、再現性のある方法を定着させ、確実に実践することである。

3.2 認識ギャップ解決への取組みレベル加工

ユーザとベンダとの間の認識のギャップに関しては、定量的な相場観やソフトウェア開発における適切な役割分担を明確にし、両者で共有することが基本的に必要である。例えば作成するシステムの規模に対してユーザ・ベンダの定量的な「感触」でのくい違いをなくすことである。

また、ソフトウェア開発はユーザやベンダなどの関係者（以下「ステークホルダ」）の共同作業により、効率よく成功につなげられるということを再確認することである。ユーザ及びベンダの両者における意識変革が重要となる。

3.3 アドホックな開発に対する取組みレベル加工

アドホックな開発やマネジメントからの脱却に関しては、これまでソフトウェア工学の研究成果として提案されてきた手法や方法論の中に、適切な場面・条件下で利用すると効果を発揮するものが、既に数多くあることを改めて認識する必要がある。実際企業や組織で実践的に使われている例がある。長年の経験に基づき納得されているものもあれば、定量的なデータ等で実証されているものもある。しかしながら、このような信頼に足り、再現性のある取組み（エンジニアリングアプローチ）がソフトウェア開発の現場に欠けている。ある企業での成功例（ベストプラクティス）があっても、部分的な情報に基づいたために全く違う状況に適用してもうまくいかず、そのプラクティスが捨て去られたり、そもそもせっかくのベストプラクティスが全く知られていないことが少なくない。国内外のソフトウェア開発の現場から適切なエンジニアリングアプローチを見つけ出し、必要ならば研究開発・実証研究を行い、普及させる必要がある。

3.4 情報の偏在に対する取組み

上記のように、そもそも実践的な情報が産業全体でうまく共有されていないという課題がある。これは、産業・学界にソフトウェア工学に関する様々なコミュニティがあるにも関わらず、情報が現場までうまく流通していないということが背景にあり、それぞれを「つなぐ」仕組みが必要である。

図 3は、現状と本来のあるべき姿についてイメージを示したものである。ユーザとベンダ間の認識のギャップやベストプラクティス等の貴重な情報の偏在に関する悪循環を断切り、図3の右に示すような好循環になることが理想である。そして、この理想は個別の企業の努力ばかりでなく、ユーザ及びベンダなど関係者を含んだ産業全体としての取組みなしには実現不可能である。

4. 取組みのポイント

本項では、数多くの課題がある中で、特に産業として取組むべきものとして、次のポイントを説明する。すべて、「ムリ」「ムダ」「ムラ」の排除に通じる。

- ソフトウェア開発の「相場観」の共有
- ステークホルダのあるべき役割の理解
- アドホックな活動からの脱却

4.1 相場観の共有

(1) 共通データの不足

相場観の共有に関する課題は、ソフトウェアの経験データの蓄積が産業的にみて貧弱であることが第一にある。海外を見てみると、オーストラリアのISBSGを始め、2〜3のデータ収集の試みがある。しかし、文化的・ビジネス環境的に異なる海外のデータに頼ることは、データのコンテキストが何にも増して重要であるソフトウェア開発では望ましくない。相場観の醸成には、適度な共通項目でプロジェクトのデータセットを絞込まないと難しい。これは、技術的な課題というよりは、まずは社会インフラ的な参照データが整備されていないことが課題である。

(2) 共通データベースの構築の必要性

この課題への取組みとして、典型的なソフトウェア開発例（例：分野・アーキテクチャ・規模等プロジェクトの特性に対する工数、生産性の分布等）の相場観とともに、ソフトウェア開発の特徴を示す分析（例：要員のスキル・仕様の安定度及び生産性の関係等）を実施し、広く産業に1つのベースラインおよび議論の土台を提供する必要がある。

基本的に実際のデータに基づき議論することが重要であり、そのためには、データ収集を行い、分析することが、産業における「相場観」を築き上げる第一歩である。

また、このようなデータベースは各種ベンチマーキングにも活用可能であり、産業としての切磋琢磨、ひいては競争力強化につながる。

4.2 ステークホルダのあるべき役割の理解

(1) 上流工程の重要性

ソフトウェア開発失敗リスクの低減の1つの方法は、ベンダとユーザ間での役割分担（誰が、いつまでにどの程度のことを決めるのがよいか）について共通の認識のもとで共同的に開発を行うことである。とくに上流工程でのステークホルダとの協力関係と適切な役割分担が、プロジェクトの成否、ひいては、ユーザにとってシステムの投資効果の高さを決めるといっても過言ではない。

現状をやや極端に表現すると、ユーザはベンダに任せたがり、ベンダはユーザが決めてくれないと文句を言い、要求を仕方なく想定してユーザが考えていたものとは違うものを構築してしまったという状況が繰返されている。

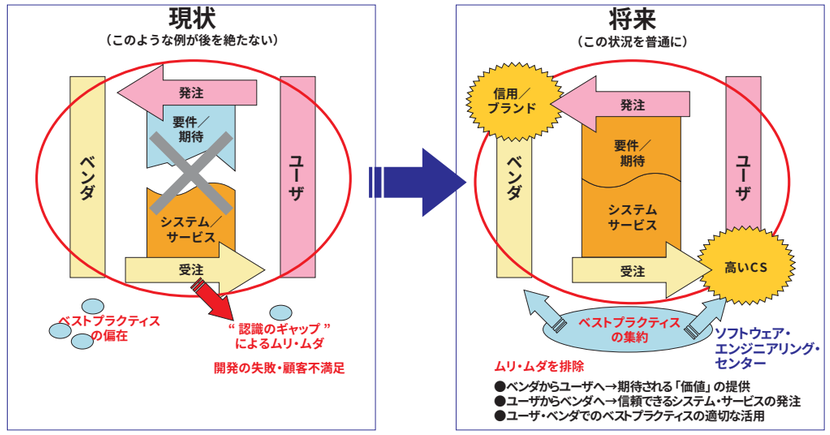

図 3 ソフトウェア開発のあるべき姿

(2) 成功する開発に向けての意識改革

課題の背景は、ソフトウェア開発が複雑な機能を実現するものであり、本来関係者の共同的で緻密な取組みが必要であるという点が理解されていないことにある。さらに根源的には、ユーザ・ベンダのそれぞれの役割として何が望まれるのかという点について、共通の基盤がないこと、すなわち、あるべき姿に関して理解にばらつきがあることを背景としている。これから生じる意思伝達上のムリ・ムダを排除し、ソフトウェア開発プロジェクトを成功させるためには、ユーザとベンダとが協力して、最も重要なポイントで共同的な役割分担を果たす必要がある。

これまでも様々な形で要求定義のあり方が示されてきているが、ユーザとベンダのどちらか一方からの観点からのものが多い。既存の成果を生かしつつ、ユーザとベンダとの共同で作成するという観点からの見直しが必要と考えている。

(3) 要求工学・設計開発技術の重要性

役割分担が共有されたとしても、更なる課題として、その表現方法は、ベンダ・ユーザ間でミスコミュニケーションを十分に防ぐ上で重要な要素である。また、ソフトウェア要求仕様の変更・増大は避けられないものとして、制御方法を確立する必要がある。

要求工学および設計開発技術の分野では、「いかに正しく要求を把握し、要件を定義するか。また、表現するか。」と「いかに効率よく品質の高いソフトウェアを開発するか。」という点について、技術的な観点からこれまでも多くの研究開発がなされている。他のテーマにも増して重要であるとともに、人間がさらに絡む分野であることから解決も困難だが、現在も活発な分野であり、ダイヤグラムを含む記述方法やユーザの要求を漏れなく適切に吸上げる方法について数多くのアプローチが提案されており、適切な条件と場面がそろえば強力なツールとなり得る。形式的なアプローチも、特に検証の観点からは根強いニーズがある。

4.3 アドホックな活動からの脱却

(1) エンジニアリングアプローチの側面

アドホックな活動からの脱却には、経験・ノウハウなどのナレッジの活用と定量データの活用の2側面から実証された再現性のあるアプローチを実践する必要がある。いずれの側面に基づいたものでも十分効果があるが、最善のアプローチは両者に基づいたものである（図4）。

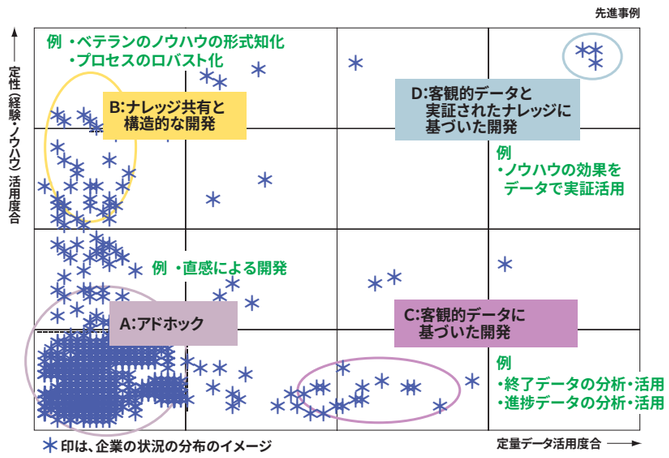

図4 エンジニアリングアプローチの2軸

アドホックな活動の最右翼は見積りである。見積りの精度を向上することは、ユーザとの間の納得感を確保し、また、開発側の進捗マネジメントで適切な目標設定を実現するという直接的な効果がある。さらに、適切な見積りとはソフトウェア開発全般の分析と把握を初期段階で適切に実施することであり、他の活動にエンジニアリングを実践する上でも大きな好影響をもたらす。このようにプロジェクトの成否を決める重要なファクタでありながら、現状は「勘と度胸」に基づいた見積りがなされ、実績から大きく外すことも少なくない。

一方、過去の定量データに基づく見積りとともに、経験豊富なPL（Project Leader）の見積りの精度がよいことからその知識を利用することも可能である。見積りは図4の両軸をもっている典型である。

(2) エンジニアリングアプローチ例（見積り）

見積手法は、既に数多くの試みが提案され、実践されているが、現場への導入の失敗例も多い。他組織で提案された手法・モデルを自組織にそのまま導入するなど、前提条件を検討することなく採用することが大きな原因の1つである。国内での実践例を見ても、見積時期、規模算定、工数算定等にそれぞれ特徴を有しており、効果のある場合とそうでない場合がある。手法の内容だけでなく、どのような前提で効果的かの情報を併せることにより、誤った導入を避けることができる。

例えば、開発側がプロジェクトの規模を見積る場合、企業にあった規模の指標（例：画面・データ量・自社にカスタマイズしたFP（Functiom Point）等）を設定し、その規模を実現するための標準的な生産性とその標準から外れる影響要因（例：ユーザの関与度合い・要求信頼性・仕様の安定度等）を把握するということが一般に実践されている。これらから、規模を示す指標の種類とそれが適切な開発分野、影響要因の種類と組織毎の要因の把握の適切な方法を抽出することが可能である。

4.4 ソフトウェア・エンジニアリング・センターでの取組み

ソフトウェア・エンジニアリング・センターでは、エンタプライズ系ソフトウェア開発強化プロジェクトを組み、上記の各種課題に対する解決策に取組んでいる。現在は、定量データの収集分析と各課題に対するベストプラクティスの発掘を中心に活動している。

定量データ分析では、企業が保有するプロジェクトデータの収集・分析を開始し[2]、さらに、見積手法・モデリングについて国内外の実践を研究・実証し、方法とともに、有効な前提条件、利用場面等を明確にする。

役割分担の意識醸成については、ガイドラインを作成する。成功や失敗に最も大きな影響を及ぼすものに絞込み、ポイントをついた情報を提供する予定である。さらに、要求工学や設計開発技術について産業の共通基盤として必要なものを洗い出している。

そしてソフトウェア・エンジニアリング・センターはこれらの情報の流通を活性化するために、各コミュニティのハブとなることをミッションと考えている。

5. おわりに

ソフトウェア開発を取巻く課題は様々であるが、いまだ基本的な活動がおろそかにされている面は否めない。そしてそれに対する処方箋は、ソフトウェア工学である。ソフトウェア工学は、ゆるやかなスパイラルを描きながらであるが、確実に進化している。「故きを温（たず）ねて新しきを知る」をまさに実践しているといえる。技術的にもコンピュータやネットワーク技術の進化がソフトウェア工学の実現を促進させている現在、改めて先人の知恵・経験を学び、新しい知恵を追加することで、ソフトウェア開発を成功させ、ソフトウェア産業を競争力ある産業にさせる好機であると信じている。

---

(1) その他大きな課題として、人材育成の問題がソフトウェア産業には根強く残っているが、機会を改めたい。

(2) JUAS（日本情報システム・ユーザー協会）との協力関係の下、ユーザ企業のデータも収集する予定

組込みソフトウェアスキル標準

ETSS Embedded (Software) Technology Skill Standard

組込みソフトウェア開発力強化推進タスクフォース　組込みソフトウェア開発スキル領域　責任者／
東海大学電子情報学部情報メディア学科　教授　工学博士／
独立行政法人 情報処理推進機構　ソフトウェア・エンジニアリング・センター　リサーチフェロー
大原　茂之

ここでは、独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センターの活動において重要な役割を果たす組込みソフトウェアスキル標準（Embedded (Software) Technology Skill Standard：ETSS）について解説する。ETSSは日本の組込みソフトウェア開発力を強化することを狙いとしている。組込みソフトウェアは製品の仕様を実現するソフトウェアである。組込みソフトウェアを開発するには、電気、機械、制御等の工学的な技術も必要であり、組込みソフトウェアは製品を構成するコアといえる。しかし、組込みソフトウェア分野の人材不足は大きな問題となっている。この問題の解決策がETSSである。ETSSを利用することによって、人材の育成、人材の確保、あるいはプロジェクトチームの構成のための最適な人材配置が可能になることを述べる。

1. はじめに

2004年10月1日に、独立行政法人 情報処理推進機構（IPA）の研究組織として、ソフトウェア・エンジニアリング・センター（SEC）が設立された[1]。SECの大きなミッションの1つには、組込みソフトウェアスキル標準の開発がある。このミッションの背景には、最近の製品の多くの機能が組込みソフトウェアによって実現される方向にシフトしてきていることがある[2]。製品によっては、組込みソフトウェアの開発力が製品の競争力に直結しているといっても過言ではない。諸外国が製品開発力を付けてきている中、日本の製品の国際競争力を強化する技術戦略上の要は、組込みソフトウェアの開発力強化にあるといえる。組込みソフトウェアスキル標準は、人材のスキル向上、目的とする組込みソフトウェア開発に必要な人的資源の調達などロジスティックスの面で寄与しようとするものである。

これまでに国が取組んできた組込み分野の人材育成としては、日本情報処理開発協会において初級・中級マイコン応用技術者試験が9年間ほど行われた。現在は情報処理技術者試験の中のテクニカルエンジニア（エンベデッドシステム）として実施されている。この間、マイコン応用技術者の育成指針、スキル標準が公開されるなど、国としても約20年弱にわたってこの分野の人材育成に取組んできたことになる[4,5]。しかし、試験だけでは人材のスキル向上や、技術者のモチベーションを向上させることは困難である。もっと積極的に組込みソフトウェアの開発力を強化する施策が必要である。SECでは組込みソフトウェアに関して、エンジニアリング領域とスキル領域という2つの側面から開発力強化に取組んでいる。その組込みソフトウェアのスキル標準は，NPO法人SESSAMEの活動が大きく貢献している[6]。以下、組込みソフトウェアの特徴、組込みソフトウェアスキル標準の体系と活用について述べる。また、とくに断らない限り本稿で使用する調査データは、文献[3]に基づくものである。

2. 組込みソフトウェアの特徴

2.1　組込みソフトウェアの広がりと重要性

組込みソフトウェアは、電子機器などの製品の機能仕様を実現するソフトウェアであると定義できる。例えば、ゲーム機本体の機能を実現するソフトウェアは組込みソフトウェアとみることができる。一方、ゲームソフトなどは、そのゲーム機の仕様に束縛されるものの、ゲーム機そのものの機能を定義していないので、そのゲーム機の組込みソフトウェアであるとはいえないことになる。

このような観点で調査した組込みソフトウェア応用製品のカテゴリを図1に示す［3］。最も大きな割合を占める産業向けの製品から家電機器や個人用情報機器に至るまで、ほとんどの製品に応用されていることがわかる。以後、このようにカテゴライズされた個々の製品の領域を組込みソフトウェアの応用ドメインと呼ぶ。

2.2　業務系ソフトウェアとの違い

組込みソフトウェアは、ほとんどの製品に組込まれるという応用範囲の広さから、汎用的な技術であるといえる。また、開発する製品のドメインによっては、機械、電気、化学、生体、制御等の知識や技術を修得し、その結果を組込みソフトウェアに反映するスキルが要求される。このような点において、組込みソフトウェアが扱う領域は、いわゆる業務系のソフトウェアとは異なることが理解できよう。

2.3　組込みソフトウェアの開発

組込みソフトウェアを含めて、ソフトウェアの開発を完全自動化できないことはゲーデルの不完全性定理から明らかである。すなわち、ソフトウェアの要求仕様そのものと、その要求を実現したコードが、それぞれ最終的な解である保障はなく、時間軸上においてソフトウェアは常に不安定な状態にあることを理解しておくことである。いつまでたってもバグを除去できないソフトウェアもあれば、バグが枯れたソフトウェアであっても新たな要求が発生してバージョンアップされることもある。詰まるところ、時間軸に対して不安定なソフトウェアという産物といかに付き合うかということが我々の課題といえるのである。

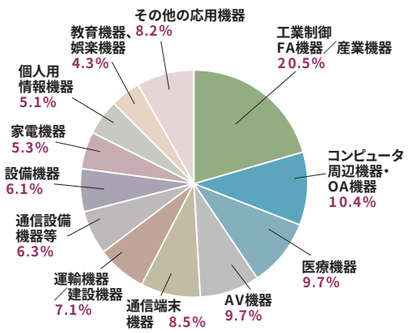

図1　開発している製品カテゴリ

しかし、ソフトウェアが不安定であるとはいっても、時間軸上の一定区間で製品の機能を保障する組込みソフトウェアの不安定さは許されない。一般ユーザが直接使用する製品においては、高い安定性をもった、言い換えると高品質の組込みソフトウェアを提供しなければならないのである。開発される組込みソフトウェアの大きさは図2に示すように1,000行未満のものから1,000万行を超えるものまで極めて広いレンジになっている。1万倍以上の開きがあるということは、組込みソフトウェアといっても、開発方法、開発環境、開発対象、人材のスキルなどが質的に異なると考えておくのが自然である。1,000行未満の場合は個人作業で対応できるが，その1万倍を超えるとなると、数百人の開発体制となる。

一方、図3に示すように開発期間は6ヶ月未満と1年未満とでそれぞれ約40％を占め、両者でほぼ80％に達する。先ほどの開発量のグラフと重ねると、不明の箇所を除いても、 50万行ないし100万行未満を1年以内に開発していることになる（図2,3）。

これだけ大量のソースコードを短期間にかつ品質を守って作成するには、個人個人の開発スキルのバランスをとったプロジェクトのチーム編成や、個人を超えたチームワークが重要となる。

3. 組込みソフトウェアスキル標準の必要性

3.1　人材の調達と要求されるスキル

組込みソフトウェアを開発するには、大量、短期間、高品質という課題を解決していくことを要求される。この要求に応える人材を各企業がどのように考えているかを示したものが図4である。

最も重要視されていることはプログラミングのスキルであり、それと匹敵してコミュニケーション能力が重要視されている。また，「非常に重要」に加えて「重要」までみると、ドキュメント作成に関するスキルが専門技術知識や開発経験を抜いて、プログラミングとコミュニケーションのスキルと同等に重要視されていることがわかる。

この図を、時間軸を固定した上でのソフトウェアの不安定要因の除去という観点から考えてもよい。学歴や資格が極端に低いのは、これらを使用してもソフトウェアの不安定要因を除去できないということであろう。同時に、知識や経験よりもスキルが重要視されていることも理解できよう。

一方、組込みソフトウェアの開発管理を行う人材に要求される重要事項を調査した結果が図5である。組込みソフトウェア開発技術者と管理者の違いが明確に示された調査結果である。また、この図からも、経験や知識よりもスキルが重要視されていることがわかる。

3.2 組込み技術者教育の課題

産業競争力を高めるための政策として各企業が要求する優先順位の調査結果が図6である。上位から第4位まで、標準化と人材教育が交互に出てきている点が興味深い。標準化と人材育成をセットで捉えている節が伺える。ある意味でこのことは必然的である。技術革新の後は、その技術を普遍化し多くの人材を歩留まりよく育成する必要が生じ、標準化が必要条件となってくるのである(図6)。

問題は、いかにしてスキルを修得させるかということであり、そのためにはスキルの標準化が必要ということである。図6の調査結果ではこの要求が第3位に挙げられている。

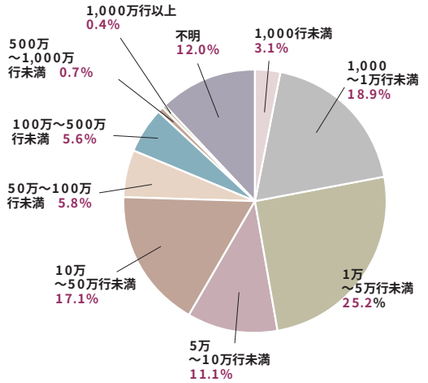

図2 既存製品の組込みソフトウェアのソースコードの行数

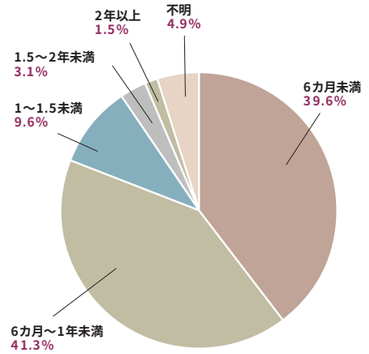

図3 平均的な組込みソフトウェア開発期間

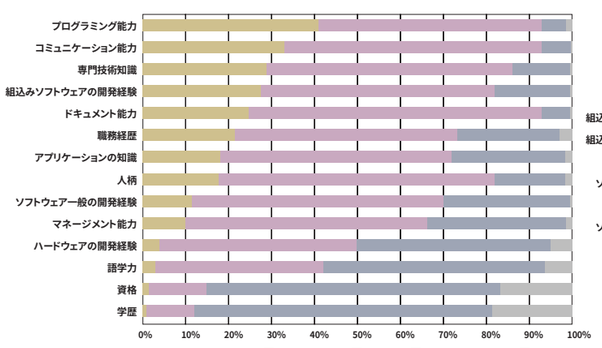

図4 組込みソフトウェア技術者を採用する場合に重要と考える要素

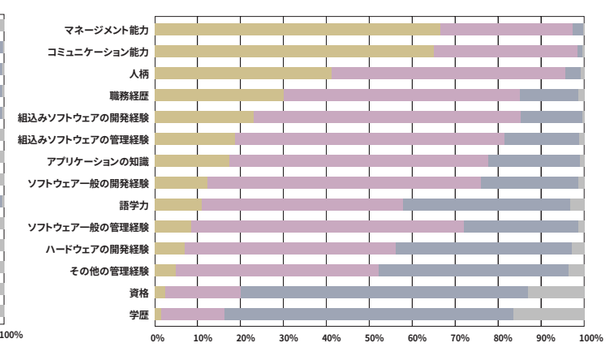

図5 組込みソフトウェア開発管理者を採用する場合に重要と考える要素

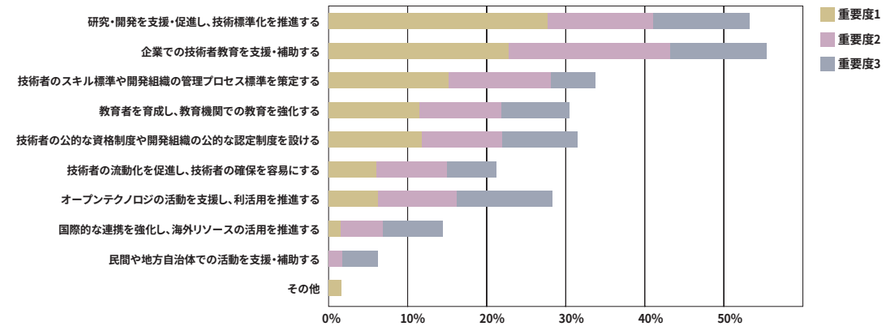

図6 組込みソフトウェア分野で我が国の今後の産業政策として重要と考える内容

4. 組込みソフトウェアスキル標準の活用

4.1 組込みソフトウェアスキル標準の構成

これまでの議論から、組込みソフトウェアスキル標準の必要性が理解されたと考える。そこで、製品のドメインに依存することなく導出した組込みスキル標準のフレームワークが図7である。このフレームワークの基本的な構造は、技術要素、開発技術、管理技術からなっている(図7)。

さらにこれらの各技術はそれぞれ、第一階層、第二階層、第三階層と細分化(詳解化)される。例えば開発技術で説明すると次のようになる。

- 第一階層にシステム設計があるとすれば、
- この第一階層を第二階層に細分化するならば、例えば、構造設計、リアルタイム設計、再利用設計等に分解される。
- この中のリアルタイム設計という第二階層は、リアルタイムタスク設計、排他制御等の第三階層に分解される。

この分類の粒度は測定するスキルの粒度に応じて対応させればよい。すなわち、育成を目的とする場合、人材の調達を目的とする場合等、その場面ごとに第一階層、第二階層、第三階層のいずれの段階で測定するかを決めればよいのである。スキルとしては、これらの項目を作ることのできるスキル、使えるスキルという二通りに分類できる。組込みソフトウェアスキル標準では、知識そのものはスキル評価の対象とはしていない。知識を測定しても、それが使えるスキルや設計できるスキルを保証するわけではないからである。ただし、知識はスキルを獲得する前提としては必要であり、人材育成という観点からは必要になってくる。

次にスキルレベルについて説明する。図8に示した項目を計測する場合、次の4段階からなる評価尺度を用いる。

(1) 設計・開発スキル(作れるスキル)を計測する場合

レベル1:上位者の指導のもとにその項目の設計・開発ができる
レベル2:上位者の指導が無くとも自立的にその項目の設計・開発ができる
レベル3:下位の技術者の設計・開発の指導ができる
レベル4:経験を体系化し先進的な設計・開発方法を工夫・開発できる

(2) 利用スキル(使えるスキル)を計測する場合

レベル1:上位者の指導のもとにその項目を利用できる
レベル2:上位者の指導が無くとも自立的にその項目を利用できる
レベル3:下位の技術者がその項目を利用できるように指導ができる
レベル4:経験を体系化し先進的な利用方法を工夫・開発できる

図8は項目に対してレベルを割当てた例である。項目としては階層のどの項目に対してでもよい。要はどの粒

度までを見たいかということを満足すればよいのである。このグラフは測定対象者のスキル特性を表している。この被測定者は、開発技術に関して広さと指導力のある高いピーク特性をもっており、要素技術もやや狭いながら指導力をもったピーク特性をもっている。しかし、管理技術に関しては、それほど高度な技術をもっているわけではない。このように各項目のレベルを明らかにしたグラフをスキルプロファイルと呼ぶ。

4.2　組込みスキル標準の使い方

実は、スキルフレームワークの項目を具体的に埋め尽くそうとすると、大きな問題が発生する。総論では、要求項目をすべて提示して、人材の育成、調達の精度を上げたいのである。しかし、ある製品を作上げるのに必要な技術項目と、各項目に要求されるスキルレベルをすべて公開することは、ある意味でその製品を開発するノウハウを流出させることになってしまう。そのため、各論としての企業の論理は、項目を明らかにできないという矛盾に陥ってしまうのである。従って、組込みソフトウェアスキル標準では、このフレームワークと、階層の詳細化の仕方と、各項目のレベル評価を定義するところまでが共通の物差しとなっている。項目の詳細化は、企業内あるいは業界内で行うことになる。

スキル標準を用いることで、次のようなシナリオを描くことができる。

(1) 人材育成のシナリオ

育成対象者：組込みソフトウェア開発技術者

育成担当者：企業内の講師、大学の教員等

育成担当者は育成対象者のスキルレベルを計測し、その対象者を育成するゴールのスキルを設定し、スキルの差分を修得させる。育成対象者は自己評価を行うとともに、育成担当者が計測したスキルとの差を理解し、自分自身のスキルレベルを明確に意識するとともに、評価に関する相場感を磨いていくことができる。

(2) 人材活用

技術者：組込みソフトウェア開発技術者

管理者：プロジェクトリーダ等

各技術者は事前に自分のスキルプロファイルを登録しておく。管理者はプロジェクトを遂行するために必要となるスキルプロファイルを設計する。次に、そのスキルプロファイルをカバーできるように、各技術者のスキルプロファイルを参照し、最適なスキルプロファイルの組合せになるように技術者を調達する。

4.3　組込みスキル標準と職種モデル

組込みソフトウェア技術の分野には、まだ明確な職種モデルが無い。統計調査でも明らかなように、組込みソフトウェア技術者のスキルとして一番要求されていることはプログラミング能力である。スキルに基づく職種モデルができると、組込みソフトウェア技術者にとって、自分自身のロードマップが明確になる。例えば、プログラマからスタートしてアーキテクトを目指したり、プロジェクトマネージャを目指すための筋道を描けることである。現在，ソフトウェア・エンジニアリング・センターのスキル標準領域のキャリア開発部会において鋭意検討中である。

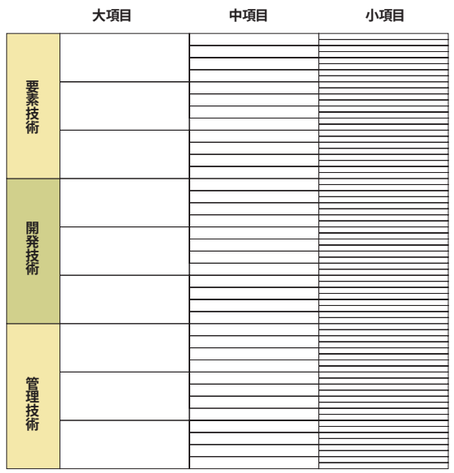

図7 スキルフレームワーク

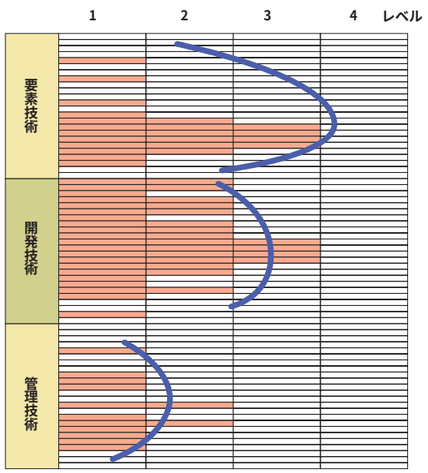

図8 スキル特性の測定例

5. おわりに

組込みソフトウェア技術は製品開発のコア技術となっている。そのような中で、製品開発の国際競争力を高めることを考えると、技術戦略的には組込みソフトウェアの開発力を強化することになる。組込みソフトウェアスキル標準はこの強化策の一環として位置づけられる。

2004年度には、本稿で解説した内容を改良してスキル標準の1.0版を開発した。その後順次キャリアモデル、教育カリキュラムを公開していく予定である。最後になるが、本稿をまとめるにあたって組込みソフトウェア開発力活性化推進委員会の皆様のご努力に敬服すると同時に感謝の意を表する次第である。

参考文献

ETSS関連の文献はまだ少ないため、基本的には経済産業省あるいはIPAのHPの公開内容を参考していただきたい。
[1] 独立行政法人 情報処理推進機構（IPA）
[2] 組込みソフトウェア開発力強化推進委員会活動報告書，http://www.ipa.go.jp/software/sec/download/200406es.php
[3] 2004年版組込みソフトウェア産業実態調査報告書，http://www.ipa.go.jp/software/sec/download/files/report/200406/es04r002.pdf
[4] IPA情報技術者試験センター，http://www.jitec.jp/
[5] 情報処理技術者スキル標準，http://www.jitec.jp/1_17skill/pdf20040
[6] NPO法人　組込みソフトウェア管理者・技術者育成研究会，http://www.ssessame.jp/

COLUMN

ITスキル標準

独立行政法人 情報処理推進機構　ITスキル標準センター
鈴木俊男

ITサービス産業ではユーザもベンダも市場のニーズの多様化、深化に伴い、求められる「人材のスキル」もまた多様化し、深化してきた。さらに、情報サービス産業に従事する人材に求められる仕事の内容の深みも出てきた。それに伴い、その仕事の内容とスキルを定義するといった“尺度”も求められてきた。そこで「パブリックドメイン」という形で仕事の内容を整理し、さらに、その指標を提示することにより、情報サービス産業に従事する人材に対してキャリアパス／キャリアアップの道筋と目標を明確にしたのが「ITスキル標準」である。また、ITスキル標準は、企業にとっては事業領域（ビジネスドメイン）に要求されるプロフェッショナル人材育成を戦略的に実施するための道具として活用される。

今日、情報システムの構築に際し、特定の人間だけで全体像を理解し推進することが困難になり、そこに携わる人材も細分化され、それぞれの仕事に対する専門性もプロジェクトの成功には不可欠な要素である。このような環境の変化に伴い、従来のプログラマ→SE→PM→コンサルタントという単一のキャリアパスで人材を育成するのではなく、専門家としてのキャリアパスの確立が必要となった。

また、ユーザの多様なニーズに対応するには1社で全てのニーズを満足させることも困難になり、企業同士の連携も一般的になってきた。企業側からみても全方位的に人材育成投資をするだけでなく、事業領域を定めて「選択と集中」により投資の効率性を求める必要性もあった。そこで日本独自のITサービス産業の共通基準を策定することが急務となり、ITサービス産業で必要とされる仕事の内容を整理し、共通の枠組として経済産業省により11職種38専門分野からなる「ITスキル標準」が策定され発表された。ITスキル標準においては各職種をハイレベル（レベル5～7）、ミドルレベル（レベル3～4）、エントリレベル（レベル1～2）の7段階の総合的な実務能力のレベルを定義している。実務能力としては全職種共通のテクノロジ、メソドロジ、プロジェクト・マネジメント、ビジネス/インダストリ、パーソナルのスキルを活用したビジネスおよび企業内外へのプロフェッショナルとしての貢献度で評価される。

2003年7月には「ITスキル標準」を補完するために教育訓練に活用するための「研修ロードマップ」を発表し、研修のフレームワークとして職種別の研修体系（体系図、コース一覧、コース概要等）によりスキル標準と人材育成の主要な要素である研修についても連動して策定した。

研修ロードマップについては、2003年7月に6職種を発表、ついで2004年8月に5職種を発表し、全ての職種が整備された。

組込みソフトウェア・エンジニアリング

組込みソフトウェア開発力強化推進タスクフォース　組込みソフトウェアエンジニアリング領域　責任者／
株式会社東芝　ソフトウェア技術センター　企画担当　参事　工学博士／
独立行政法人 情報処理推進機構　ソフトウェア・エンジニアリング・センター　組込みプロジェクト　研究員
平山　雅之

30年ほど前、ソフトウェア危機という言葉が真剣に議論されたことがある。コンピュータの急激な進化にともないソフトウェア需要が急増し、開発技術者が不足する時代が到来すると騒がれた。そして、その解決策としてソフトウェア工学とそれに裏打ちされたソフトウェア工場の考え方が脚光を浴びた。当時のソフトウェア危機は汎用コンピュータの進化と普及を背景に、主に業務システムを対象としていたように記憶している。

21世紀を迎えた現在、我々の身の回りには30年前には創造だにしなかった様々な情報システムが溢れている。そしてそれらの多くは様々なハードウェアをベースにした組込みシステムであるといっても過言ではない。そしてこうした組込みシステムの開発に関係する立場で周囲を見直してみたとき、新たなソフトウェア危機が眼前に広がろうとしている。

1. 現　状

我々が開発という立場で、あるいは一般のユーザという立場で関わりをもつ組込みシステムは多岐にわたる。その多くはマイコンを利用し、その上で動作する組込みソフトウェアによって様々な機能を実現している。これらの組込みソフトウェアは製品としての価値を生み、また、製品の差異化の源泉となっている。結果的にこれらの組込みソフトウェアはその開発規模が急激に増加し、従来の開発方法では適切な品質やコスト等を維持することが難しくなってきている。

従来の組込みソフトウェアは、様々な機器（ハードウェア）のコントロール（制御）を担う部品として、製品システムの開発の中では付属物的な扱いを受けてきた。このため、組込みソフトウェア開発はハードウェア開発者が片手間に行うことも少なくなく、所謂ソフトウェア工学等に根ざした適切な開発手法の実践がおろそかになっていた。しかし、携帯電話やデジタル家電等の事例を見るまでもなく、近年、様々な組込みシステムで不具合が発生する等深刻な状況に陥っており、従来然とした開発方法で状況を解決することは難しくなってきている。

ここでは、こうした背景を踏まえ、第2節で組込みソフトウェア開発の状況を紹介し、第3節では組込みソフトウェア開発の特質を考えてみたい。第4節では、組込みソフトウェア開発の状況や特質を考慮し、既存の開発手法の一部を紹介する。第5節では、今後の組込みソフトウェア開発力強化のために求められるソフトウェア・エンジニアリングについて論じてみたい。

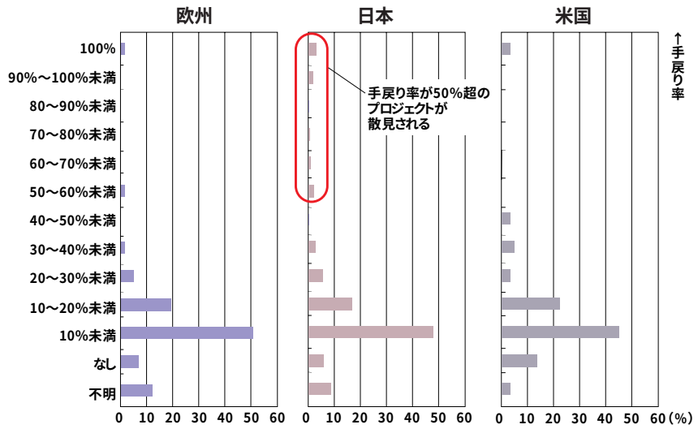

図1 組込みソフトウェアの品質トラブル発生度合い

2. 組込みソフトウェア開発の状況

ここではまず、現状の組込みソフトウェア開発の状況を、QCD（Quality、Cost、Delivery）の面から考察する。

●組込みソフトウェアの品質

我が国の工業製品は、従来、品質面での優位性がその国際競争力の源泉となってきた。しかし、昨今の様々な組込みシステム製品のトラブルを見る限り、その中核をなす組込みソフトウェアの品質は心もとない限りといっても過言ではない。図1は2004年に経済産業省で実施した組込みソフトウェア実態調査の結果の中から、組込みソフトウェアの品質トラブルに関するデータを抜出したものである[1]。図の縦軸は開発中の手戻り率を表しており、横軸はプロジェクト数を示している。この図をよると、我が国の組込みソフトウェア開発プロジェクトには、開発中の手戻りが50%を超えるプロジェクトが散見されることがわかる。これに対して欧米ではこうした開発中の手戻りがほとんどないことが読取れる。開発段階での手戻り率の高さは、開発上流工程が必ずしも十分に機能していないことの表れでもある。

●組込みソフトウェアの開発コスト

図2は新製品開発におけるソフトウェア開発費の占める割合を示したものである。近年の組込みシステムの多くは複雑な機能を実現するためにソフトウェアによる差別化を図る傾向が強く、図に示すように全開発プロジェクトの約30%近くで、製品開発費の半分近くがソフトウェア開発に充当されているといった状況にある。一方で、図3に示すように開発効率を向上するための開発支援ツールの利用状況は欧米並とはなっておらず、依然として個々の技術者の技量に依存した開発スタイルが高コスト構造を生みだしており、往々にして開発コストの超過等の事態を招いている。

●組込みソフトウェア開発の期間

我が国の組込みソフトウェアの多くはいわゆるコンシューマプロダクトと呼ばれる領域の製品に搭載されている。こうしたコンシューマプロダクトの世界は、多くの企業間での激しい市場獲得競争にさらされており、製品の市場投入タイミングが極めて重要な要因になっている。

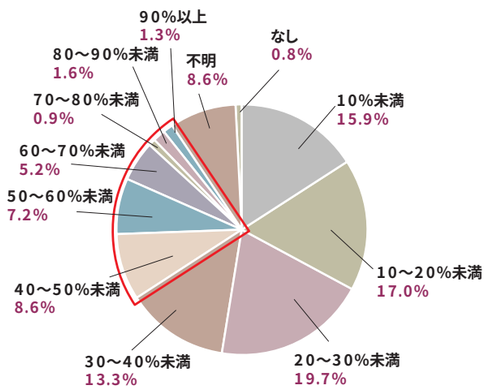

図2 応用製品の新製品開発費のうちの組込みソフトウェアの開発費の占める割合

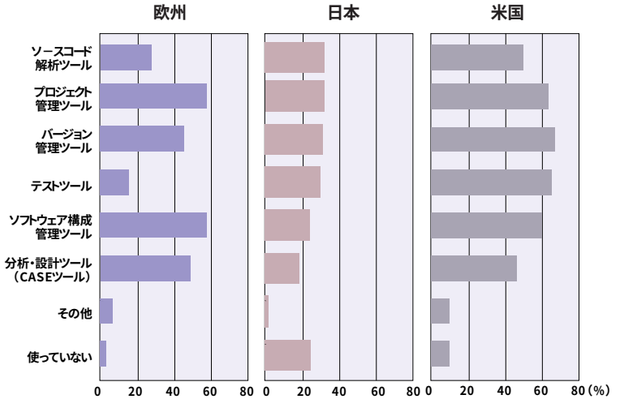

図3 開発支援ツールの利用状況

このため、その一部として開発される組込みソフトウェアも極めてタイトなスケジュールの中で開発される傾向があり、開発期間も図4に示すように半年から1年程度と短期間での開発が常態化している。そして実際の開発現場ではこうしたタイトな開発スケジュールに間に合わずに製品リリースを延期する等のケースも少なからず発生している。

3. 組込みソフトウェア開発の特質

ここではまず、組込みソフトウェア開発の特質を述べる前に、組込みソフトウェアの特徴を整理しておく。一般に組込みソフトウェアは「ハードウェア依存性」「リアルタイム性」「リアクティブ性」等の特徴をもつ。

3.1 組込みソフトウェアの特徴

① ハードウェア依存性

組込みソフトウェアは様々なハードウェアが連携して動作し、システムとしての動作を実現する。このためハードウェアで実現する部分の機能や性能等も含め、ハードウェアに依存する部分が多く、ソフトウェアとハードウェア間のすり合わせが開発のポイントとなる。

② リアルタイム性

組込みソフトウェアは様々なハードウェアの振舞いとの同期をとりながら機能を実現するために、決められた時間内での処理実行が必須となる。場合によってはミリ秒のオーダーが要求されるものも少なくない。

③ リアクティブ性

組込みシステムの多くは、センサ等を介して組込みシステムの外部環境状況を把握し、機能を実現する。そのため、こうしたセンサからの情報をもとにシステム外部の状況・状態に合わせた動作実行が求められる。

3.2 組込みソフトウェア開発の特質

このように組込みソフトウェアは従来のエンタプライズ・ソフトウェアには無いいくつかの特徴を有しており、これらの特徴を考慮した開発手法や開発プロセスが必要となっている。ここでは組込みソフトウェア開発手法や開発プロセスを整備する上での留意事項を整理しておく。

a. ハードウェアへの対応

組込みソフトウェア開発では関連するハードウェアを常に意識しておかなければならない点が極めて重要となる。これは開発技術、開発プロセス技術や開発マネジメント技術等あらゆる技術に共通して当てはまる。

● 開発技術におけるハードウェア要因への対応

例えば組込みソフトウェアを設計・実装する場合を考えてみる。この場合、どのような特性をもったどのようなハードウェアが開発しているソフトウェアに関係するかを正しく理解していないと、適切な動作を実現する組込みソフトウェアは開発できない。このためソフトウェアの設計モデリングの段階で、ハードウェアに関する情報を考慮することが必須となる。

● 開発プロセスにおけるハード要因への対応

多くの組込みシステムは、ハードウェア開発と並行する形で組込みソフトウェア開発が進められる。この場合、ハードウェア開発と組込みソフトウェア開発がどの段階でどのような接点をもち、どのようなすり合わせを行うかが開発プロセス上の重要な問題となる。このため組込みソフトウェア開発では、ソフトウェア開発のみを意識した開発プロセスではなく、ハードウェア側の開発プロセスを考慮した形で整備しておくことが求められる。

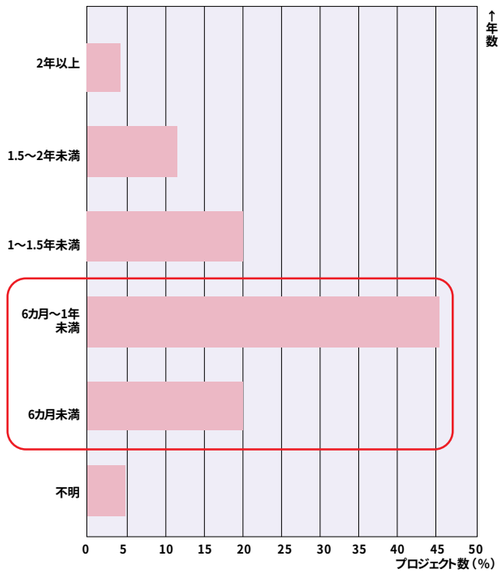

図4 組込みソフトウェアの開発期間

● 開発管理技術におけるハード要因への対応

一般に開発管理において、ソフトウェア開発とハードウェア開発の差異は「見える」か「見えないか」といった点が大きく異なっている。ハードウェア開発では開発した成果物が可視化しやすく結果的に開発管理がしやすいといわれている。一方、ソフトウェア開発では、開発した成果物（ソフトウェアアーキテクチャやソフトウェアそのもの等）は可視化が難しく開発管理が難しい。特に組込みソフトウェア開発では、ハードウェア依存性ゆえにハードウェア、ソフトウェアの両面を考慮した開発管理が必要となる。即ち、可視化しにくいソフトウェア開発を如何に可視化し、その状況をハードウェア開発者に如何に伝えて、開発の刷り合わせのタイミングをとっていくかが、とくに重要となる。

b. 実時間・実世界への対応

実時間への対応を前提とした動作や組込みソフトウェアが動作する実世界への対応動作は、組込みソフトウェア開発を難しくしている重要な要因の1つである。

組込みソフトウェアの多くは、周辺機器やハードウェアの制御といった側面を強くもっている。そして、こうした制御動作では周辺世界の実時間に対応した動作が求められ、リアルタイム性の実現や様々な並行動作の実現が重要な要素となる。このため、組込みソフトウェア開発では常にこの実時間性を意識した設計やテストが必須となる。例えば実時間性を考慮した並行動作については、様々なタイミングを考慮した振舞いの設計が中心となるが、タイミングのトリックによって思わぬ誤動作等を誘発する可能性が高い。このように、組込みシステムの実時間性は、その品質を大きく左右する要因になっている。

また、 組込みシステムはシステムを取巻く外部環境やユーザ操作等によって、様々な動作や振舞いが求められ、組込みソフトウェアを考える上で、このような実世界への対応は極めて重要な要素である。例えば、実世界への対応の中でも最も代表的な要素としてのリアクティブ性は、組込みソフトウェアの状態分岐の爆発をもたらす主要因の1つである。リアクティブ性を考慮した様々な動作バリエーションの設計は組込みソフトウェア設計で欠くことのできない作業であり、正常系動作に加え異常系動作も含めた網羅的なシナリオ分析が必須となる[4]。

4. 組込みソフトウェア開発技術の現状

ここでは実際のソフトウェア開発現場で利用されている技術の現状と課題について紹介する。

4.1 組込みソフトウェアの設計技術

エンタプライズ・ソフトウェアの設計ではOMG（Object Management Group）によって提唱されたUML（Unified Modeling Language）が広く使われるようになりつつある[2]。この潮流を受け、組込みソフトウェアの開発現場でもUMLを利用する企業が増加している。しかし、UMLはオブジェクト指向をベースとした手法であるが、組込みソフトウェア開発において、UMLを利用して設計モデリングを行った場合、そのままOOP（Object Oriented Programming）を実施するケースはそれほど多くなく、実装の段階でオブジェクト指向から従来の構造化の視点に視点変換を行っているケースも少なくない。このため組込みソフトウェア開発の現場ではUMLやオブジェクト指向の利点を生かしきれていない。

また、2003年に改定されたUML 2.0では組込みソフトウェアを意識してタイミング図等が導入されたが、組込みソフトウェアの実時間性や微妙な動作タイミング等を表記する上では必ずしも十分な記述能力を有しているとはいえない[3]。

4.2 組込みソフトウェアのテスト検証技術

ソフトウェアの品質を維持し向上させる上で、検証技術やテスト技術は極めて重要な役割をもっている。欧米では宇宙・航空産業あるいは防衛産業等の領域の組込みソフトウェア開発において徹底したV&V（Validation & Verification）の実現を目指している。一方、我が国では、e-Society 基盤ソフトウェアプロジェクトの中で高信頼組込みソフトウェア構築技術として取組まれている[4]。

また、テスト技術でも、欧米では肥大化するシステムを効率的にテストするための手法として統計的解析手法を活用したテスト項目の設計手法等が提案・実用化され

ている。一方、我が国では、いまだに網羅テストや人手によるレビューに重きが置かれており、最新のソフトウェア工学に根ざした技術の導入が遅れている。

4.3 開発マネジメント技術

一般にソフトウェアの開発マネジメントの世界ではPMBOK等が広く知られている[5]。PMBOKはプロジェクトマネジメントを行うための様々な技術を集めた知識体系として米国の非営利団体であるPMI（Project Management Institute）によって整備されたものである。開発マネジメントは開発プロジェクトの規模が大きくなった場合には必須の技術であり、既に、多くのエンタプライズ・ソフトウェアの開発マネジメントでは利用されている。組込みソフトウェアの場合、急激な機能規模の増大に伴い、その開発プロジェクト規模が肥大化したために、プロジェクトマネジメントの必要性が十分に認識されないままに開発が進んでいるプロジェクトが少なくない。このため、組込みソフトウェア開発の現場には、まず、PMBOKのような開発マネジメントの必要性を啓蒙していく必要がある。その上で、組込みソフトウェア開発のマネジメントは、第3節でも述べたように、ハードウェアを意識することが求められるため若干の開発マネジメント手法のカスタマイズが必要となってくる。

開発マネジメントを有効に機能させる上では、開発初期段階に作成する開発計画の妥当性が最も重要である。開発計画書の中では当該プロジェクトでどのようなマネジメント戦略を採用するか、あるいはどのようなスケジュールでマネジメントや開発を進めていくかを明確にし、確定することが求められる。組込みソフトウェア開発の開発マネジメントでは、このマネジメント戦略の立案に際して、 組込みソフトウェア開発の可視化と定量データに基づく開発のコントロールをどのように進めるかがポイントである。特に関連するハードウェアの開発とどのようなリンクや整合性をとっていくかについては十分な工夫が必要である。

4.4 開発プロセス技術

ソフトウェア開発プロセスに関しては、ISO/IEC 12207やSLCP（Software Life Cycle Process）等で基本的な考え方が示されている[6]。しかし、これらが対象としているソフトウェアはいわゆるエンタプライズ・ソフトウェアであり、必ずしも組込みソフトウェアを意識したものとはなっていない。組込みソフトウェアの場合、ハードウェア開発プロセスとのインタフェースや、上流でのハードウェア/ソフトウェア分割、実機ハードウェアとのすり合わせ等、プロセスやそれを構成するアクティビティ（作業）のレベルでエンタプライズ・ソフトウェア開発とは異なる点がある。このため、組込みソフトウェアの開発プロセスを設計する際にはこれらの要素を考慮しなければならず、実際の開発現場では経験的に個別のプロセスを構成して運用している。

また、組込みソフトウェアを短期間で開発するために、XP等の軽量プロセス（Light Weight Process） を導入する傾向も出始めている[7]。しかし安易な軽量プロセスの導入は、本来、組込みソフトウェア開発で実践しなければならない基本的なアクティビティ（作業）やプロセスを単純に省略してしまう等の事態を誘発し、深刻な結果をもたらす可能性を併せもっている。

このため、組込みソフトウェア開発に必須な作業を抽出し、標準的なプロセスとしての整備が急務になっている。

5. 組込みソフト開発力強化のシナリオ

以上、紹介してきたように我が国の組込みソフトウェア開発についてエンジニアリングの側面から眺めてみると、まだ多くの改善すべき点が存在している。

ここでは我が国の組込みソフトウェア開発の開発力強化を実現する上でのシナリオを考えてみたい。

5.1 近未来のシナリオ

第2節で紹介したように、我が国の組込みソフトウェア開発はQCDの面で問題が少なくない。そして第4節に見るように、組込みソフトウェア開発の中ではソフトウェア工学の成果に基づく様々な手法や技術は必ずしも効果的に利用されていないことがわかる。

このための処方箋としては

① 既存のソフトウェア工学に根ざした手法を広く紹介し、啓蒙・普及を進めていく

② 第2節で述べた側面を考慮して手法自体を組込みソフトウェア開発に適した形にカスタマイズしていく

という2つの重要な活動が求められる。

例えば、SECが2004年にドラフトとして発行した小冊子「組込みソフトウェア開発におけるマネジメントの勧め」等も有効な活動の1つである[8]。組込みソフトウェアの特質を考慮した上で、組込みソフトウェア開発におけるプロジェクトマネジメントの有用性や注意事項を整理し、マネージャや実際の開発担当者等にも広くプロジェクトマネジメントを普及啓蒙するといった位置づけをもっている。

また、 欧州の自動車業界等では自動車の制御ソフトウェアを対象としたコーディングルールとしてMISRA-Cを提唱し運用を始めている[9]。このように、ソフトウェア開発の原点でもあるソースコードレベルの品質を確実なものとする方法等も現在の組込みソフトウェア開発の中では重要であると考えられる。コーディングルール等はある意味で昔から個々の企業や開発者ごとに整備されてきたものであるが、MISRA-Cは自動車という高信頼性を要求される製品ドメインの特性を考慮して整備している。このように対象とするドメインの特徴を考慮した上で、手法や技術をカスタマイズしていくことも有効なアプローチである。

5.2 数年後に向けたシナリオ

第3節で紹介したように、我が国の組込みソフトウェア開発技術の研究や実用化は欧米に比べるとやや遅れている。この数年、 CMMI、プロダクトライン、MDAを始めとして業界の話題となっている技術の多くは海外からの輸入技術であり、ソフトウェア開発技術の面では輸入超過が続いている。しかし、上記のような海外発の技術が我が国の組込みソフトウェア開発に適しているかどうかは定かではない。即ち、我が国の製品開発の強みは「すり合わせ型開発」にあり、これが適したソフトウェア開発技術の確立が急務である。

組込みソフトウェア開発の中での「すり合わせ」の最たるものはハードウェアとソフトウェアの間のすり合わせである。設計手法、プロジェクトマネジメント、開発プロセスのいずれをとっても、このハードウェアとソフトウェアの間のすり合わせをいかに円滑に進められるか、そのための手法の整備が重要となってくる。

例えばシステム設計のレベルでは既に回路設計の上流でUMLを導入する等の研究も進んでおり、ハードウェア設計とソフトウェア設計のコンカレント化が進もうとしている。また、組込みシステムとしての動作を開発のより早い段階に効果的に確認するための協調検証技術等の研究も始まっている。また、こうしたハードウェア・ソフトウェア間の高度なすり合わせ技術以外にも、組込みソフトウェア技術者が設計の段階でハードウェア動作を適切に意識した設計ができるようにする設計モデリング手法やその上での形式検証技術の実用化等、解決すべき課題は多岐にわたる。

6. おわりに

以上、本稿では、組込みソフトウェア開発に焦点をあてて、その現状や課題点、現状の開発技術の状況等を紹介し、組込みソフトウェア開発力強化にむけたシナリオを紹介した。組込みソフトウェア・エンジニアリングについては、技術研究・開発普及・啓蒙ともに端緒についたばかりである。是非、産学官の連携のもと、我が国発の組込みソフトウェア・エンジニアリング手法が広がっていくことを期待したい。

参考文献

［1］2004年組込みソフトウェア開発実態調査報告，経済産業省
［2］J.ランボー他：UMLリファレンスマニュアル，ピアソンエデュケーション，2002.
［3］Object Management Group：OMG UML2.0,infrastructure superstructure specification, Tech.Repo，Object Management Group, 2003.
［4］文部科学省リーディングPJ：e-society基盤ソフトウェアの統合開発　平成15年度報告書，http://cif.iis.u-tokyo.ac.jp/e-society
［5］PMI：Guide to the project management body of Knowledge-2000
［6］ISO/IEC 12207：Software Life Cycle Process
［7］長瀬嘉秀：eXtreme Programming実践レポート-XPプロジェクトを実践した手法と軌跡，翔泳社，2002年.
［8］組込みソフトウェア開発におけるプロジェクトマネジメントの勧め，独立行政法人 情報処理推進機構，2004年.
［9］MISRA-C研究会：組込み開発者におくるMISRA-C 組込みプログラミングの高信頼化ガイド，日本規格協会，2004年.

所長対談：Rombach博士に聞く

ソフトウェア・エンジニアリングの産学官コラボレーションの要は

H．Dieter．Rombach博士

ドイツ・カイザースラウテルン大学情報専門学群（コンピュータ・サイエンス学部に相当）専任教授でソフトウェア・エンジニアリング担当。フラウンホーファ協会の実験的ソフトウェア工学研究所（Institute for Experimental Software Engineering：IESE）の所長兼務。また、メリーランド州カレッジパークのメリーランド大学内のコンピュータ・サイエンス学部とUMIACS（メリーランド大学・先進コンピュータ研究所）において教官を務め（1984年から1991年まで）、さらに同時期（1986年から1991年まで）のソフトウェア・エンジニアリング・ラボラトリー（SEL：NASA、ゴッダード宇宙飛行センター及びメリーランド大学の共同事業）のメンバーを経て、現職にいたっている。

● **経歴**　1975年にドイツ連邦共和国カールスルーエ大学でBS（数学）、1978年に同カールスルーエ大学でMS（数学及びコンピュータ・サイエンス）、1984年にドイツ連邦共和国カイザースラウテルン大学でPh.D.（コンピュータ・サイエンス）の学位を取得。1990年にはソフトウェア・エンジニアリングにおける研究成果が認められ、全米科学財団から「若手研究者対象・会長賞」を受賞。2000年には博士のソフトウェア・エンジニアリングにおける研究の成果、並びにフラウンホーファ研究所設立を通して行われたドイツRhineland-Palatinate州の経済発展に貢献したことが称えられ、同州から勲功章を授与されている。

● **活動**　ドイツ政府や欧州連合（EU）、欧州産業界から支援を受けている複数の研究プロジェクトのリーダ。例えば、ソフトウェア・エンジニアリングにおける革新的技術に関して、その知識を網羅したドイツ版データベース構築を目指す連邦政府支援プロジェクト（ViSEK）の主査を務めている。品質改善、ソフトウェア測定、ソフトウェアの再利用、プロセスのモデル化などソフトウェア技術一般の問題に関して、多くの企業のコンサルティングを行い、連邦政府においてはソフトウェア関連の技術顧問となっている。また、ソフトウェアの品質改善、ソフトウェア測定、ソフトウェアの再利用、プロセスのモデル化について、企業の役員向けセミナーをしばしば行っている。『IEEE Software』特別号のゲストエディタも２回（1987年9月の「ソフトウェア品質保証」特集と1994年7月の「測定を基にしたプロセス改善」特集）務めた。1992年9月にドイツ、Dagstuhlで開催された「実験ソフトウェア・エンジニアリングの課題」に関する国際ワークショップを主催。1996年にベルリンで開催された第18回国際ソフトウェア・エンジニアリング・カンファレンス（ICSE1996）では議長を務めた。

● **最近の活動**　Kluwerジャーナル『実証的（エンピリカル）ソフトウェア・エンジニアリング（Empirical Software Engineering）』の副編集長。ACMのメンバ、IEEEのフェローとしても活躍。特に、ソフトウェア工学関係の最近の役割としては、以下の2つがある。

(1)　Software Process Achievement Award（IEEE及び米国カーネギーメロン大学SEI主催）評価委員
(2)　ICSE 2006のプログラム委員長

ドイツ・フラウンホーファ協会の実験的ソフトウェア工学研究所IESEは、大学等の研究機関により開発されたソフトウェア・エンジニアリングの技術的成果の産業界へのトランスファを目的とし、客観的データに基づいた定量的な実証（エンピリカル・ソフトウェア工学）を基本的な研究姿勢として持つ。このほどカイザースラウテルン大学教授を兼ねるIESE所長のH.Dieter.Rombach博士が、SECとのソフトウェア開発ベストプラクティスの確立を目指した共同研究契約締結のために来日し、本ジャーナルのためにSEC所長の鶴保征城と対談した。産学官連携で数々の成果をあげているIESEからは、我が国のソフトウェア・エンジニアリングの発展に重要な示唆が得られると期待されている。

※フラウンホーファ協会については本誌38頁を参照。

IESE活動の原点

鶴保　SECは2004年10月1日に設立し、IPA所属が26名、タスクフォース所属が130名という陣容です。ここでは産学官連携で日本のソフトウェア競争力向上をめざし、ソフトウェア・エンジニアリングの実践強化を進める拠点として活動を展開します。そこで同様な目的をもつIESEの具体的な取組み内容をお聞かせください。

Rombach　IESEはフラウンホーファ協会配下の58の研究所の1つです。IESEのミッションはSECと共通し、産学官連携推進により、民間に向けた技術移転を円滑に行います。具体的には、イノベーティブ・ソフトウェアエンジニアリング、マネジメント・プラクティス、ナレッジ・マネジメント等のノウハウや知識に関し、実証的で測定可能な方式を培うことで実現します。現在IESEには、研究員が150名おり、米国のメリーランド大学にも25名を擁しています。ドイツでは、組込み型ソフトウェア分野を対象とした自動車業界や銀行・保険関係等の金融業界、そしてIT系製造業界等の産業分野で連携しています。

鶴保　大学とは、どのような連携を行っていますか。

Rombach　58の研究所の各ディレクタは、大学のディレクタ的な役割を兼務しています。私はカイザースラウテルン大学情報専門学群（コンピュータ・サイエンス学部相当）の専任教授兼務です。

IESEは委託研究とコンサルテーションの双方を行っています。すなわち新技術の開発と同時にコンサルテーションにより、その技術移転サービスも行っています。

鶴保　それは、まさに医学分野における大学の基礎研究、大学病院の臨床といったようなイメージですね。

Rombach　ご指摘どおりで、フラウンホーファ協会では医学における臨床試験の提供のようなことを考えています。すなわち基礎研究のリサーチデータをさらに洗練させる、ここにツールを提供する、要素をパッケージ化する等を学生達とともに行い、企業サイドでもこれらを組込んだ上でかなり現実的な再現も行っています。このような実証研究から、よい新技術と評価されれば企業でも活用されるでしょう。その結果企業内プロセス改善にもつながり、やがてパイロットプロジェクトの段階にも進めます。そしてここからロールアウトして、ベストプラクティスへといった一連の流れになります。企業は私どもの新技術評価により、技術の強みや弱みを評価できるので、技術導入後のリスクを低減でき成功の可能性もより高まります。

鶴保　征城（つるほ　せいしろ）
1966年大阪大学大学院工学研究科電子工学専攻修士課程修了後、同年4月日本電信電話公社（現NTT）入社。1989年11月ＮＴＴ ソフトウェア研究所長、ＮＴＴデータ取締役、ＮＴＴソフトウェア社長等を経て、2004年6月独立行政法人 情報処理推進機構 参与。同年10月ソフトウェア・エンジニアリング・センター所長に就任。
高知工科大学工学部情報システム工学科教授（2003年～）
奈良先端科学技術大学院大学 客員教授（2003年～）
独立行政法人日本学術振興会「基盤的ソフトウェア技術開拓」に関する研究開発専門委員会委員（2003年～）
電気通信大学 客員教授（2002年～）
社団法人情報処理学会会長（2001年～2002年）
XMLコンソーシアム　会長（2001年～）
社団法人電気情報通信学会　フェロー
社団法人情報処理学会　フェロー

物理学的な法則の欠如を補うフィードバック

鶴保　実証ベースの（エンピリカル）ソフトウェア・エンジニアリングは、大変すばらしいと思います。工学はすべてが実証的であるべきですが、ソフトウェア工学だけが実証を銘打つのは逆に不幸とはいえませんか。

Rombach　ソフトウェア・エンジニアリング分野は成長期で新しいから仕方がないことでしょう。ソフトウェア・エンジニアリングと他の違いは、我々をガイドする物理学的な法則がない点です。したがって、開発や設計を繰返し行い、実証せねばなりません。しかも実証は、産学官バラバラで行っていてはいけません。例えば大学の研究室で、あるテーマの成果が得られても、スケールアップの際に、それを実証できるかどうかは疑問です。またスケールアップ後、民間企業では、大学が得意とする基礎的測定あるいはその方法に関する知識が欠落して

いるので、大学側からの技術移転が必要となってきます。私の経験では、企業との連携による実証研究は、ビジネスケースでデータ分析後、そのフィードバックを我々サイドから企業に向けて提供し、どのような改善が必要かガイドをしないと、企業側では得るものがありません。ですからこのフィードバックが極めて重要です。

日本においては産業界と学界との距離は遠い感じがします。この距離をSECが橋渡ししなければならず、さらなる努力が必要でしょう。

R&Dと商用化を隔てる「デスバレー」

鶴保　その通りですね。一般的にR&Dと商用化との間には深い谷、いわゆる「デスバレー」があるといわれています。よい技術ができた場合、同じ会社内でもなかなか事業部サイドで商品化できません。そこでSECでは、最初からこうした問題を予測するようにしています。民間の第一線で使えるよう商用化するためには、最初から同じ場に参加し、新しい方法論なりベストプラクティスをお互いに協議しながらコンセンサスをとるようにしていく、そして技術を作った民間側の人が持ち帰って、デスバレーの谷底からはい上がっていくような場を最初から設定すべきという考え方に立っています。

Rombach　最初から民間が参加するというのは、極めて有望な考え方であり、両者の信頼関係は培われるでしょう。私の経験から付言させていただきますと、同じ産業界側でもR&D担当及び事業部担当では違いがありますので、できればユーザ側に近い事業部の方に関与していただきプロジェクトを進められるといいのではないでしょうか。R&D側ですと、デスバレーの同じ側になってしまいます。

我々のアプローチは、プロセス改善や製品開発の中に明確な目標を設定し、これをもとに研究所や民間企業から4～5人ずつのメンバを集めて1つのプロジェクトチームを作ります。そして、3～5年の期間で、プロセス改善や製品開発を行います。これにより、基礎技術の専門家と、応用技術のノウハウを持った人がコラボレーションでき、開発サイクルの短縮につながります。

新しいソフトウェア工場論

鶴保　Rombachさんはご自身の著書の中で、ソフトウェア工場論を好意的に書かれています。日本では東芝などで1990年代前後にそうした考え方が普及し始めました。しかし、現在ソフトウェア工場と銘打っているところはありません。この理由は、設計と製造を分離でき、製造をソフトウェア工場で実施すべき、という考え方がうまく機能しなかったからです。とくにお客様から発注を受けたソフトウェアは、製造をなかなか分離できません。そのうちシステム規模が小さくなってきて、ますますスペック自体がお客様サイドでも決まらない、つまりアジャイル開発でなければダメという流れです。そうなると、ソフトウェア工場という考え方は今後も成立し得ないのではないでしょうか。


右からIPA理事長 藤原武平太，Rombach博士，
IPAソフトウェア・エンジニアリング・センター 所長 鶴保征城

Rombach　確かに、要件や設計と、製造の切分けといった狭い意味でのソフトウェア工場の考え方はうまくいきませんでした。しかし、ソフトウェアプロダクトラインというソフトウェア工場の新しい実現方法ができています。システム開発において、要件からコードに至るまでの一連の流れと、製造における繰返し作業の2つの間で切分けを行うのです。こうしたソフトウェアプロダクトラインをベースとした、あるいは経験工学といった観点からのソフトウェア工場は、いまソフトウェア・エンジニアリングの世界では注目されています。

競争力低下が懸念されるソフトウェア・エンジニア不足

鶴保　日本においては、毎年約5,000人がIT専門教育を受け、業界に就職しています。一方、業界では100万人に近いSEが働いているので、5,000人というのはいかにも少ない数です。いま日本はここが弱点として問題になっています。

Rombach　ドイツもソフトウェア・エンジニアリングの専門教育を受けて採用される人は少ないですね。カイザースラウテルン大学では、3つの変革を行いました。第一にソフトウェア・エンジニアリングの教科では、プラクティカルな部分のソフトウェア・エンジニアリングを必須科目としました。単に講義を聴講すればいいのではなく、実際の実践プロジェクトに参加しなければなりません。第二に情報工学部門では、より応用分野にフォーカスした教育を行うこととしました。つまり、情報工学部門を卒業して一般的なコンピュータ・サイエンスがわかっていても、自動車業界では即戦力にならないなどのクレームが産業界からあり、アプリケーションをベースとした知識も学校で身に付けておくようにしました。第三が企業に採用された技術者が途中からソフトウェア・エンジニアリングを担当しなければならなくなったときです。この人たちを対象に2005年からスタートさせるプログラムが、インターナショナル・ソフトウェア・カリキュラムというもので、こうした人たち向けに必要な教育を提供する場としています。これは自動車会社からの要請によります。

鶴保　産業界からの要請は、数万人単位といった大量の技術者の確保ではないかと思います。

Rombach　新卒のほか既存技術者による移転を絡ませると、そうしたニーズに近づけると思います。過去20年間の流れは、トラディショナルな技術者90％に対して、残り10％がソフトウェア・エンジニアでした。しかしいまでは50％がソフトウェア・エンジニアといったように、その比率は向上しています。アプリケーションレベルでの経験を培ったものであれば、そのポテンシャルを我々が補填することで、ソフトウェア・エンジニアが大きく育つという可能性は大いにあります。

近年、海外へのアウトソーシングの問題がありますが、ノウハウや高収入を伴うインテリジェンスなタスクは、国内に留めるべきです。中国やインドは確かに教育に力を入れてはいますが、これは主としてローレベルプログラミングの部分です。この分野はアウトソースでいいのです。しかし、システム・エンジニアリングといったハイレベル分野に関しては、まだこれらの国々においてクリエイティブな教育が行われているとはいえません。ドイツでは、ソフトウェア・エンジニアリングこそ21世紀を担う産業と考えられています。これは20世紀の製造業と同様に重要な役割を果たします。かつて自動車業界において、アウトソーシングしつつも高収入を維持できたのは、プロセス・テクノロジがあり、品質を保証できたからです。ソフトウェア・エンジアリング分野でも、これらプロセス・テクノロジと品質こそが、競争のための差別化になります。

鶴保　確かに、中国やインドが毎年10万人以上の学生をIT専門家として送出し、かつ賃金も安いので、日本は今後彼らには勝てないのではないか、という懸念があります。しかし、そうではなく、まず日本には大きな市場があり、世界的にみてもハイレベルの新しいビジネスモデルもある、これをソフトウェア化すれば、中国やインドに負けないような可能性を秘めている、と考えるべきだと私は思います。今後も協力していきましょう。

ドイツ・フラウンホーファ協会 IESE

URL：http://www.iese.fhg.de/

エンタプライズ系ソフトウェア開発力強化推進タスクフォース　総合部会　副主査委員／
独立行政法人 情報処理推進機構　ソフトウェア・エンジニアリング・センター　エンタプライズ系プロジェクト　プロジェクトサブリーダ
石谷　靖

ドイツにはソフトウェア工学の最新成果を産業にスムーズに定着させることを目的にした研究所が存在する。フラウンホーファ協会実験的ソフトウェア工学研究所（IESE）がそれである。SECとも共同研究の包括的合意覚書を交わし、共同プロジェクトを実施している。以下には、IESEについてその活動と特徴を簡単に紹介する。

1　IESEの概要

IESEは、Institute Experimentelles Software Engineering（実験的ソフトウェア工学研究所）の略である(1)。IESEはドイツ・フラウンホーファ協会(2)の研究所として、フランクフルトから南西に約100kmに位置するカイザースラウテルン市にある。現在の所長であるRombach教授が中心となって1996年に設立された。当初は10名程度の研究員だったが、現在では150名弱の大所帯となっている。

他の情報先進国同様、ドイツでもソフトウェアの基盤研究が大学で行われているが、その成果を企業が取入れるまでに時間がかかりすぎる、又は全く活用されないという問題が存在している。IESEの事業拡大の背景には、ソフトウェア産業では数多くの課題があり、その解決のためにソフトウェア工学の最新成果をできるだけ早く企業に展開したいというニーズがある。

実際、IESEは現在も事業拡大傾向にあり、現在の研究員を2007年には100人増の250名体制にする予定である。2005年夏からは同市内で建設中の最新設備のビルディングに移動する予定となっている。

2　IESEにおけるコア技術と研究体制

IESEにおける研究体制は、図1に示すとおりである。

現在150名の研究員により主要なコンピテンシーとしてソフトウェア工学全般をカバーしているが、IESEの名称からも窺えるとおり、実験／実証的（Experiment）なアプローチがすべての活動の基盤となっているところが著しい特徴である。個別のコンピテンシー、たとえば、品質・プロセスエンジニアリング、ソフトウェアプロダクトライン、組織的学習などは、すべて実験／実証的なアプローチに基づいて展開されている（図2）。定量化・実証という文化が深く根付いた研究組織である。

既に40年以上もソフトウェア開発の難しさがいわれ続けており、ソフトウェア工学としてその解決のために数多くの方法が提唱されてきたが、多くは提唱されては時間が経つと消えていくことを繰返してきた。中でも、ソフトウェアは見えにくく、人間が扱うものであるため、定量的に扱うことはできない、という認識は根強く、科学的なアプローチは取り得ないと信じられているといってもよい。

そのような状況でIESEは、現場の問題解決に当たって、定量的なアプローチを活用し、先端的なソフトウェア工学手法を「エンジニアリング」として可能であることを示すことを実践しているユニークな機関である。

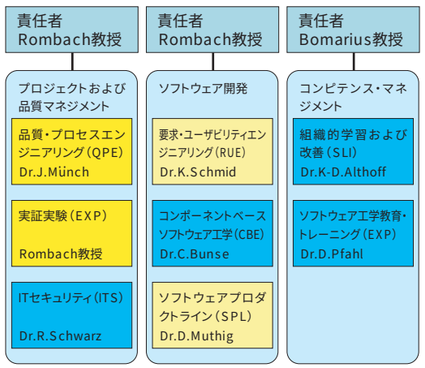

図1　IESEの研究体制　（出典）Annual Report 2003

3　実証的なアプローチ

IESEでは、データの背景となるコンテキスト（例：企業の特性、個々のプロジェクトの特性）を重要視して、個別の企業のニーズに応えるノウハウを蓄積している。図3には、その取組みのイメージを示している。ソフトウェア開発における難しさは、個別のプロジェクトがさまざまな特性をもっており、一般的な原理・原則だけの適用では問題を解決仕切れないという点にある。IESEの研究者は「プロジェクトのコンテキスト」を常に念頭に置き、ドメインを始めとするコンテキスト（背景）が類似したプロジェクトでのコンサルティングなどの経験に基づき、個別のリアルなデータについてその内容を分析（主に、差異分析）することによって、適切な解決方法を提案する。

また、解決策を求めるアプローチは非常にプラグマティックであり、既存技術（標準・規格、ツール等）は、積極的に活用する。つまり、特に必要としない限りは、できるだけ、既存の技術やツールを活用し、早急な問題解決を図る。この方針は、企業ごとの特性が様々であることや、技術進展に対して汎用性の確保・維持が難しいことを背景としている。非常に現実的な考えに基づいている。

技術普及促進の一環として、技術体系に名称等を付けてパッケージとして示し、体系化のアピールも特徴的である(3)。

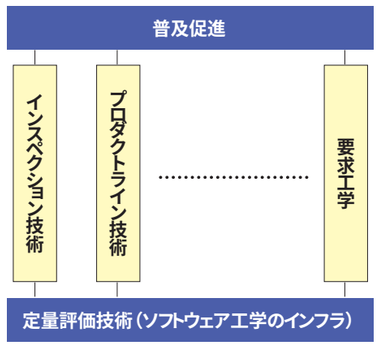

図2　IESEのコンピテンシー

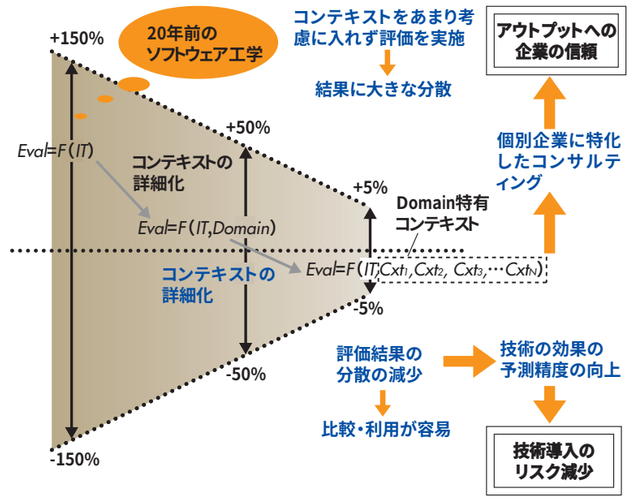

図3　実証的定量評価のアプローチのイメージ　Rombach教授との意見交換に基づく

4　ソフトウェア・エンジニアリング・センターとの共同研究

ソフトウェア・エンジニアリング・センターでは、現在IESEで開発された見積モデリング手法の日本での活用可能性の研究として共同で国内企業に試行している。IESEとの共同研究は、海外のベストプラクティスを研究する第一歩であるとともに実証的なアプローチの可能性も追求するものでもある。

(1) IESEの研究者たちは「イエゼ」と発音する。
(2) フラウンホーファ協会は、58の研究所を擁し、コンピュータ、ソフトウェア、バイオ、建築、化学工学、電気・電子工学など、エンジニアリング分野のほぼすべてをカバーしている。各分野の先進的な技術を産業に技術移転することに注力し、個別の企業の具体的な課題に対する研究実績の適用によるソリューション提供をミッションとしている。
(3) http://www.iese.fhg.de/Products_Services/

EASEプロジェクト

URL:http://www.empirial.jp/

エンタプライズ系ソフトウェア開発力強化推進タスクフォース　定量データ分析部会　委員／
独立行政法人 情報処理推進機構　ソフトウェア・エンジニアリング・センター　エンタプライズ系プロジェクト　研究員
神谷　芳樹

ソフトウェア開発の問題への取組みとして、データに基づいた科学的手法によるソフトウェア開発を目指す「エンピリカル（実証的）ソフトウェア工学」（以下エンピリカルソフトウェア工学）の研究成果に熱い視線が集まっている。こうした中、奈良先端科学技術大学院大学と大阪大学は平成15年度から文部科学省のリーディングプロジェクト「e-Society 基盤ソフトウェアの総合開発」計画の一環として、EASE（Empirical Approach to Software Engineering：ソフトウェア工学へのエンピリカルアプローチ、イーズと読む）と名付けた研究プロジェクトを、産業界からの参加を得て産学官連携ですすめている[1]。

1　目　的

「EASE プロジェクト」（以下EASE ）は、ソフトウェア開発の分野において、他の科学、工学分野と同様に、計測、定量化と評価、そしてフィードバックによる改善という手法の実践を目指している。経験的に頼りがちな現状に対し、科学的手法に基づく信頼性や生産性の高いソフトウェア開発手法の確立を目指している。そのために、産業界との協力による実データの収集を通し、定量的なデータに基づいて、ソフトウェア工学のさまざまな手法、技術、ツールを評価し、産学官の連携体制によって大学だけでの研究の限界を克服しようとしている。

2　アプローチ

エンピリカルソフトウェア工学の研究では、大学などアカデミアにとっては、研究対象としてソフトウェア開発現場を得ることが重要な要素になるが、一般にはこれは容易ではない。一方日常の課題に埋没しがちな産業界では、アカデミアにおけるソフトウェアメトリックスなどの豊富な知見を開発現場に活用していくのは容易ではない。そこでEASEではこの課題の解決を狙い、国（文部科学省）の予算を引き金に新しい産学官連携のフレームワークを実現した。研究のスタートにあたっては、それを方向づけるいくつかの要素がある。

(1)　リーダシップ

EASEにおいて、アカデミアに蓄積されたエンピリカルソフトウェア工学の知見を活用することを狙い、アカデミア側のリーダシップとしている。

(2)　リーダシップの鍵

アカデミアから産業界へは、知的な人的資源などを供給できるが、これらだけでは計画立上げは難しい。そこでEASEでは、産業界に対してEPM（Empirical Project Monitor）すなわち「ソフトウェア開発における自動的なデータ収集と解析のためのプラットフォーム」というソフトウェアシステムを新規開発し提供していくことにした。

(3)　資金、リソース供給

一般にこのような計画での資金確保は容易ではない。EASEでは前述のように、引き金資金を政府予算に求め、より幅広い活動と、また産業界側の責任を明らかにする意味も込めてマッチングファンド方式を採用している。

(4)　物理的作業場所

EASEプロジェクトでは産学連携を円滑にすすめるために大学キャンパス外の交通アクセス至便の場所に専用ラボを設けて活動拠点としている。大阪府千里中央のライフサイエンスセンタービル内に開設した「エンピリカルソフトウェア工学ラボ」である（図1）。

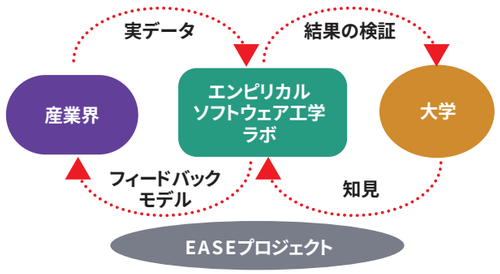

図1　ラボを介した産学連携のイメージ

　このようなラボ開設の背景に「ソフトウェア工学研究における大学病院モデル」あるいはこれを発展させた「大学工房モデル」の考え方がある[2]。

(5)　展開方法

EASEでは次の3つのStepを考えている。

Step1（第1年度）：コア大学とコア企業によるコンセプトワークとプラットフォーム（EPM α版）の開発。

Step2（第2年度）：開発したプラットフォーム（EPM α版）の少数の企業群への提供とフィードバック。提供先企業群とのデータ共有の実現。

Step3（第3年度以降）：プラットフォーム（EPM）の機能拡充と幅広い提供、これを囲むコミュニティの実現。幅広いデータ共有の実現。共有したデータの解析による新しい知見の獲得とコミュニティへのフィードバック。これを通した我が国ソフトウェア産業の競争力獲得への貢献。

3　活動の概要と利点

EASEプロジェクトは、2003年4月に開始された。ソフトウェア企業4社（NTTソフトウェア、SRA先端技術研究所、日立製作所、日立公共システムエンジニアリング）を産業界のコアメンバとし、ただちに前述のラボを開設し、プラットフォーム（EPM）の開発に取掛かった。

EPMの機能は、基本的には、ソフトウェア開発環境で基本となる構成管理、メール管理、障害追跡管理等のツール群から、ソフトウェア開発管理情報を自動的に収集し、標準形式に変換したのちリレーショナルデータベースに格納する。そしてアナライザによって、これを分析、ビジュアルに表示する（図2）[3] 。ソフトウェア開発工程の流れに沿いリアルタイムで自動的に管理情報を把握し、リスクに対応可能にすることを狙っている。

EASE開始後、半年してEASE国際フォーラムを開催する等、計画の公知活動を展開し、EPMを適用する共同研究の候補企業を確保した。

EASE開始1年後の2004年4月にはEPM α版の提供を開始し、合わせて適用企業および適用予定企業とのクローズな研究会（「エンピリカルソフトウェア工学研究会」：隔月東京で開催）を開始した。本執筆時においては、7社でEPM適用作業中である。

4　国際アドバイザ

EASEでは、これまでエンピリカルソフトウェア工学研究の分野で関係が深く、国際的に評価の高い4人の教授に国際アドバイザを依頼し、時に応じてアドバイスを得ている。

- Dr.Victor R. Basili （米国・メリーランド大学教授/フラウンホーファ・メリーランド センター長）
- Dr. Dieter H. Rombach （ドイツ・カイザースラウテルン大学教授/フラウンホーファ実験的ソフトウェア工学研究所所長）
- Dr. Barry W. Boehm （米国・サザンカリフォルニア大学教授）
- Dr. Ross Jeffery （オーストラリア・ニューサウスウェールズ大学教授）

5　ソフトウェア・エンジニアリング・センターとの連携

ソフトウェア・エンジニアリング・センターのエンタプライズ系ソフトウェアの定量データの収集と分析計画、および組込みソフトウェア開発への取組みはEASEと高い補完性をもっている。両者は今後その連携を強めて発展させていく予定である。

参考文献

［1］井上克郎，松本健一，鶴保征城，鳥居宏次：実証的ソフトウェア工学環境への取り組み，情報処理，45巻7号，pp.722-728，2004.
［2］井上克郎：実践的ソフトウェア工学における産学協力，平成12年度電機関係学会関西支部連合大会　S9-5，pp.S50，2000-11-26，2000.
［3］大平政雄，横森励士，阪井　誠，松本健一，井上克郎，鳥居宏次：Empirical Project Monitor:プロセス改善を目的とした定量的開発データの自動収集・分析システムの試作，電子情報通信学会技術報告SIGSS，Vol.103，No708，SS2003-48，pp.13-18，Mar，2004.

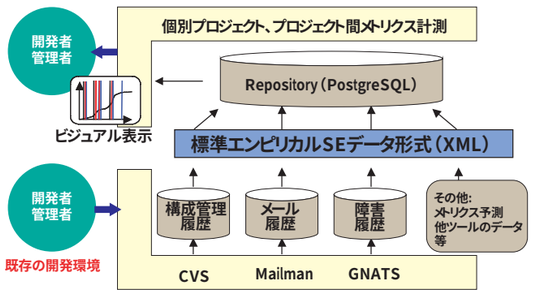

図2　EPM（Empirical Project Monitor）の構成

東京大学21世紀COE ものづくり経営研究センター

URL:http://www.ut-mmrc.jp/

東京大学大学院経済学研究科教授　21世紀COE拠点リーダー
藤本　隆宏

東京大学ものづくり経営研究センターは、日本発の「ものづくりシステム」の国際的な研究拠点を目指して2003年に設立された。国内主要16メーカとコンソーシアムを組み、「統合型ものづくりシステム」に関する知識の一般体系化を進め、この知識の産業間移転・海外発信を行いながら、現場発の新たな産業観を提起している。

1　ものづくりシステムのアーキテクチャ

現在、日本の諸産業や官界の間では、トヨタ方式に代表される統合型システム（すりあわせ型ものづくり）を競争力回復の鍵とみる論議が急速に高まっている。

また、一部の欧米企業には、ITバブルの崩壊後の反省と「基本への回帰」の一環として、日本の統合型システムを再評価する機運がみられる。自動車等「すりあわせアーキテクチャ」の製品を主な土俵とする「統合型ものづくりシステム」については、主にトヨタ生産方式、全社的品質管理（TQC）等を中心に、我が国でも生産管理論、経営組織論等の諸分野で研究が進み、80年代以降は海外でも「リーン生産方式」等の分析が行われた。しかし90年代に入ると、日本経済の低迷により、こうした日本発のものづくりシステムに対する海外の学界・言論界の興味は衰退し、かわって、モジュラー的な製品思想をもつ製品に関する研究が活況を呈することとなった。今日では、これらの2つの流れをバランスよく理解しようとする機運が高まりつつあり、その流れで「すり合わせ型製品」（例えば自動車）と「モジュラー型製品」（例えばパソコン）のものづくりシステムを包括的に理解するためのアーキテクチャ分析の枠組みが形成されつつある。

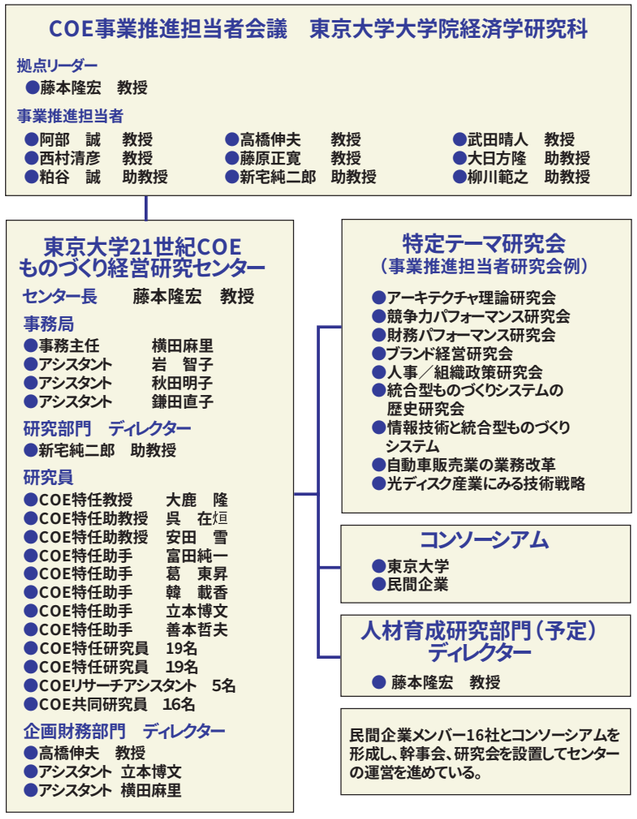

図1　センターの組織

2　新たなものづくり論の形成

こうした中、ものづくり経営研究センターでは次の４テーマを柱に複数のプロジェクトを同時並行的に研究している。

（1）統合型システムの一般体系化研究
（2）アーキテクチャ産業論研究
（3）ブランド力
（4）産業競争力の国際比較研究

特に主要テーマである「統合型システムの一般体系化研究」を産学連携して推進するために、民間企業16社（トヨタ、松下、ソニー、キヤノン等）とコンソーシアムを組み、共同研究を行っている。国立大学法人として、このようなコンソーシアムを設けての共同研究は、先進的な取組みといえよう。

ものづくり経営研究センターには若手研究者の他、ものづくり経営のプロ経験者数人に特任研究員として加わってもらっている。この方々は、東大他の若手研究者・学生と研究チームを組み、彼らの豊富な暗黙知を継承可能な形式知化する仕組みを作る予定である。これに純粋学術的な研究を加え、すべての研究活動を成果対応のプロジェクトベースで行っている。また、さまざまな分野のものづくりのベテランと若手研究者との協力により、いろいろな視点・分野での成果を予定している。

3　内外研究機関との連携

内外研究機関との連携も活発に行い、例えば、教育面を主導する高橋伸夫教授を中心に、既設の特定非営利活動法人であるGBRC（グローバルビジネスリサーチセンター）や5年一貫教育として創設した「特修コース」を連動させ、この分野で調査能力・発信能力のある若手研究者を育てる。また、東京大学の1、2年生を対象とした文理融合の「フレッシュマンビジネススクール」というコースを開設する。社会人向けには丸の内地区において、ものづくり経営に関する学者・実務家のレクチャーを常設で行う「ものづくり寄席」を開設している。毎週月・木曜日に開設し、すでに1,500名以上が来場した。

MIT等海外の先行事例でも明らかなように、こうした目的の拠点は息の長い研究が必須である。あえて「センター」と名乗ったのは、拠点形成を目的とする「COEプログラム」終了後も世界への知的発信の基地として継続させようとの不退転の決意の現れである。

4　ソフトウェア・エンジニアリング・センターとの連携

今回のソフトウェア・エンジニアリング・センターとの連携は、ソフトウェア開発力が、ものづくりシステムの1つの焦点になってきている現状を反映したものである。伝統的に経営学分野でのソフトウェア産業の研究は手薄な分野であり、その社会的重要性を考えると、今後、さらなる研究資源の投資が必要な分野である。デジタル機器や車載ソフトに代表されるように、ハード＝ソフトの二分法ではなく、産業競争力という観点からソフトウェア開発を分析し、日本の（ひいては世界の）ソフトウェア産業へ何らかのフィードバックができれば、本センターとしてもうれしい限りである。

コンソーシアム会場の様子

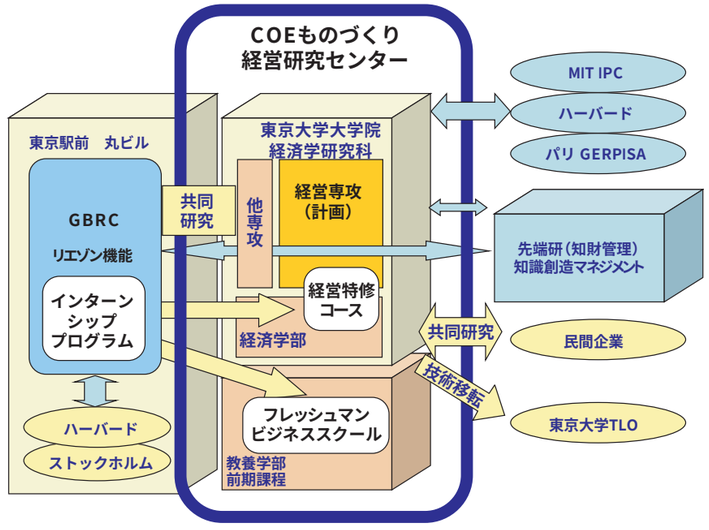

図2　内外研究機関との連携

「コンピュータの名著・古典100冊」

石田 晴久 編・著
青山幹雄・安達淳・塩田紳二・山田伸一郎　共著

ISBN: 4-8443-1828-4 発行：インプレス
A5判・254頁　定価1,575円（税込）2003年10月刊

「これは読んでおけ」と、良き先輩のアドバイス

本書は、コンピュータに関する膨大な書籍の中から、『名著』と言われるものや『古典』といわれるものを『コンピュータ名著読書推進委員会』の方々がセレクトして紹介したものであり、コンピュータに携わる人たちに「これはお勧め」といったアドバイスをしてくれる書籍といえる。

失敗を含む先人の知恵は、使えるものは使うべきだ。しかし、この知恵というのが曲者で、氾濫する情報の中から「使える知恵」を抽出することは、非常に難しい。昔の情報は、内容が限られており、また、技術的にも基本的で本質的な要素に近い粒度の情報が多く「使える知恵そのもの」といえた。しかし、コンピュータの高性能化やインターネットの普及によって、情報量が爆発的に増え続け、「知恵の抽出」は今後、より難しさを増していくだろう。

そのため、現在のようにコンピュータが進化し、情報が氾濫する時代は、本書のようなガイドブックが非常に役に立つ。このガイドによって、読者は良質の情報に効率よくアプローチできるのだ。豊富な経験と見識を持つガイド（『コンピュータ名著読書推進委員会』の方々）によって、先人の「使える知恵」を有効に使おうではないか。

本書は100冊を次の11の分類に整理している。①歴史、②人物・企業、③ドキュメンタリー、④思想、⑤数学/アルゴリズム、⑥コンピュータサイエンス、⑦アーキテクチャ/OS、⑧コンパイラ/言語、⑨プログラミング、⑩ソフトウェア開発、⑪データベース/ネットワーク。そして、1冊ごとにそれぞれ見開き2頁で紹介されている。また、書籍の価値や有効性を記述しているため、まるで尊敬する先輩が、「この本はいいよ」「これは読んでおけ」と言ってくれているような雰囲気を感じさせる。身近にこのような先輩がいないときには、この本は良き先輩としてアドバイスをしてくれることだろう。

残念ながら、紹介されている書籍のなかで、これまでに私が読んだ書籍は少なかった。特に前半は皆無に近く、これまで先人が歩んできた道のりや、思想を含む基本技術的な書籍を読んでいないことを痛感した。これは技術スキルの自己診断結果と合致するものであり、「本を読む」ことの重要性を再認識させられた。

紹介されている書籍の約7割は10年以上昔に出版された書籍である。私が読んだことのある書籍を思い出すと、最近の図表が多用される書籍に慣れていると違和感を覚える書籍が多いように思える。地味だが本質に近い「使える知恵」が記載されているので、この違和感も楽しみながら、未読の書籍にチャレンジしてみたいと思う。これからのエンジニアや現役エンジニアには、この本をお読みになり、自己診断されることをお勧めする。この本で紹介される100冊以外でも構わないが、広く先人の「使える知恵」に触れる機会を増やして欲しい。

（渡辺　登）

『組込みソフトウェアレポート2005』

情報処理推進新機構（IPA）ソフトウェア・エンジニアリング・センター／
経済産業省商務情報政策局情報処理振興課 組込ソフトウェア開発力強化推進委員会 監修

ISBN: 4-7981-0845-6　発行：翔泳社
B5判・242頁　定価2,415円（税込）　2004年11月刊

注目したいのは“人”

本書は、裏方的なイメージの強い組込みソフトウェア業界の実態をインタビューや調査によって炙り出したものである。組込みソフトウェア業界の今を垣間見ることのできる珍しい書籍である。

当然、組込みソフトウェアって何？　という疑問も解消できる。また、自動車、デジカメ等の中の組込みソフトウェア解説や、組込みソフトウェアに関する動向や技術解説もしっかりしている。これだけでも、組込みソフトウェア業界に馴染みが無い人には読み応えがあるはずだ。そして、読み終えて感じたのだが、組込みソフトウェアとは、ひと言では表現できないものであり、逆に表現する必要性もない。

ありとあらゆるものがコンピュータで制御される今、組込みソフトウェアは必須の部品である。部品というと単純なものでしかないように受け取られるかもしれないが、これは全く違う。脳をCPUに例えるならば、組込みソフトウェアは脳に蓄積される記憶や才能といったシナプスの接続モデルといったところだろうか…。 余計に組込みソフトウェアをわからないものにしてしまいそうなので、この辺でやめておく。

さて、組込みソフトウェアに関する書籍では、センサやアクチュエータの制御、リアルタイム制御等々、技術にフォーカスが当てられることが多いが、本書で注目して欲しいのは”人”である。メーカやベンダの社内で開発作業に従事し、製品が出荷されても、守秘義務の関係から表に出て製品や技術に関する話をする機会が少ないため、よっぽどのことが無い限り、組込みエンジニアがインタビューに答えることはないだろう。コンシューマ向けの機器の場合でも、商品企画や機構設計者がインタビューに答えるくらいだ。しかし、本書は珍しく10名以上のエンジニアへのインタビューが掲載されている。『ピープルウェア』的に人の問題が開発に及ぼす影響が云々ではなく、組込みエンジニアが、これまでの仕事や現在の仕事等についてインタビューに答えている。

ところで、組込みソフトウェア開発は、企業内に閉じた仕事になりがちである。社外のエンジニア、しかもトップランナ的存在のエンジニアのインタビューは読み応えがある。技術系の雑誌で取上げられるインタビューは経営者や研究者が多いので、エンジニアのインタビューが読めるのは嬉しいものだ。ゲームソフトウェアにおけるスーパークリエイタ的なヒーローが、組込みソフトウェアのエンジニアからも出てほしい。それも「あの機種のソフトウェアは、○○氏が開発したから期待できる」という感じで。パッケージ・ソフトウェアやゲーム・ソフトウェアでは、開発者が語られることがあるが、組込みソフトウェアではまだ皆無である。過酷な労働条件といったイメージが先行してしまった組込みソフトウェア開発エンジニアの職種ではあるが、本書により、世の中を便利に快適にする製品を実現するキーマンであるというイメージへと変わって欲しいと願う。

（渡辺　登）

ソフトウェア・エンジニアリング関連 イベントカレンダー

作成：SEC journal編集委員会

開催時期	開催日	イベント名	主　催	開催場所／発行者	URL
1月		SEC journal No.1発行		独立行政法人 情報処理推進機構（IPA）	http://www.ipa.go.jp/software/sec/
2月	3日（月）～4日（火）	デブサミ（Developers Summit）2005	株式会社翔泳社	青山ダイアモンドホール（東京都港区）	http://www.seshop.com/event/dev/
	22日	JASA組込みソフトウェアフォーラムin名古屋	社団法人 日本システムハウス協会（JASA）	調整中	調整中
3月	2日（水）	情報処理学会全国大会	社団法人 情報処理学会	電気通信大学（東京都調布市）	http://www.ipsj.or.jp/10jigyo/taikai/67kai/
4月		SEC journal No.2発行		独立行政法人 情報処理推進機構（IPA）	http://www.ipa.go.jp/software/sec/
5月	18日（水）～20日（金）	IPAX2005	独立行政法人 情報処理推進機構（IPA）	東京ビックサイト（東京都江東区）	http://www.ipa.go.jp/
6月	8日（水）～10日（金）	ソフトウェア・シンポジウム2005	ソフトウェア技術者協会	富山国際会議場（富山県富山市）	http://ss2005.jaist.ac.jp/
	20日（月）～21日（火）	SECソフトウェア開発力強化推進フォーラム2005	独立行政法人 情報処理推進機構　ソフトウェア・エンジニアリング・センター	経団連ホール（東京都千代田区）	http://www.ipa.go.jp/software/sec/
	29日（水）～7月1日（金）	ESEC	リードエグジビションジャパン株式会社	東京ビックサイト（東京都江東区）	http://www.esec.jp/
	29日（水）～7月1日（金）	SODEC	リードエグジビションジャパン株式会社	東京ビックサイト（東京都江東区）	http://www.sodec.jp/
7月	25日（月）～26日（火）	SPES2005	社団法人 情報サービス協会（JISA）	日本科学未来館（東京都港区）	調整中
		SEC journal No.3発行		独立行政法人 情報処理推進機構（IPA）	http://www.ipa.go.jp/software/sec/
8月	25日（木）～26日（金）	SWEST（Summer Workshop on Embedded System Technologies）	組込みシステム技術に関するサマーワークショップ実行委員会	遠鉄ホテルエンパイヤ（静岡県浜松市）	http://www.ertl.jp/SWEST/
10月	3日（月）	情報化月間	経済産業省	全日空ホテル（東京都港区）	http://www.ipa.go.jp/
	17日（月）～19日（水）	組込みソフトウェアシンポジウム2005	社団法人 情報処理学会 ソフトウェア工学研究会	日本科学未来館（東京都港区）	http://www.ipsj.or.jp/
	10月予定	ソフトウェアジャパン2005	社団法人 情報処理学会	明治大学アカデミーコモン（東京都千代田区）	http://www.ipsj.or.jp/
	24日（月）	SECソフトウェア・エンジニアリング・コンファレンス	独立行政法人 情報処理推進機構　ソフトウェア・エンジニアリング・センター	経団連会館	http://www.ipa.go.jp/software/sec/
		SEC journal No.4発行		独立行政法人 情報処理推進機構（IPA）	http://www.ipa.go.jp/software/sec/
11月	16日（水）～18日（金）	Embedded Technology 2005	社団法人 日本システムハウス協会（JASA）	パシフィコ横浜（神奈川県横浜市）	http://www.jasa.or.jp/
	16日（水）～18日（金）	SEPG japan 2005	日本SPIコンソーシアム（JASPIC）	都市センターホテル（東京都千代田区）	http://www.jaspic.jp/
12月	調整中	SPCシンポジウム（ソフトウェア生産シンポジウム）	財団法人 日本科学技術連盟	日科技連　東高円寺ビル（東京都杉並区）	http://www.juse.or.jp/

上記は変更される場合があります。参加の際に必要な詳細事項は主催者にお問合せをお願いいたします。

SEC topics

各種小冊子の提供予定

SECは活動成果物として現在提供している「組込みソフトウェア開発におけるプロジェクトマネジメントの勧め」（ドラフト版）に続いて、2005年4月以降に、各種小冊子の提供を予定しています（但し小冊子名は仮称）。

「組込みソフトウェア開発力強化推進委員会」成果物
「組込みスキル標準」小冊子
「コーディング作法ガイド」小冊子
「エンタプライズ系ソフトウェア開発力強化推進委員会」成果物
「定量データ分析」小冊子
「開発プロセス共有化ガイド（上流）」小冊子
「見積り手法ガイド」小冊子

SECのイベントへの取組み予定

2月　デブサミ（Developers Summit）2005
講演：「組込みプロジェクトマネジメントの勧め」
講演：「組込み開発スキル標準の開発と活用」

2月　JASA組込みソフトウェアフォーラムin名古屋
基調講演：「プロジェクト/ソフトウェア特性プロファイルと組込みシステム開発」
テーマ講演：「高品質実装のためのコーディング作法」
テーマ講演：「スキル標準で描く高品質ソフトウェアを作れる技術者像」
パネルディスカッション：「高品質なものづくりとしてのソフトウェア開発とは」
招待講演：「次世代自動車ソフトウェアプラットフォーム開発」

3月　情報処理学会全国大会：「組込みシステム産業の将来とそれを支える技術者育成」
講演：「組込みシステム技術動向」
講演：「組込みソフトウェア産業の実態」
パネル：「組込みシステム技術者の将来像と育成戦略」

5月　IPAX Spring：「SECの成果報告」

6月　ソフトウェアシンポジウム2005
講演：「鶴保所長による総括報告」
パネル：「実践的なソフトウェアエンジニアリングを目指して」

6月　SECソフトウェア開発力強化推進フォーラム2005
基調講演：「東京大学　坂村健教授（予定）」
成果報告：「エンタプライズ系ソフトウェア開発力強化推進委員会」
成果報告：「組込みソフトウェア開発力強化推進委員会」
調査報告：「組込みソフトウェア産業実態調査2005」

6月　SODEC：セミナー、展示ブース

6月　ESEC　：セミナー、展示ブース

SEC topics

編集後記

ソフトウェア・エンジニアリング・センター（SEC）として、ジャーナルを出そうという話が出たのは、2004年の8月のことでした。SEC自体もまだ立上がっておらず、10月の設立に向けて大忙しの最中でした。その中で、敢えて、ジャーナルを出そうと考えたのには、理由があります。ソフトウェア・エンジニアリング、特に、実証的ソフトウェア・エンジニアリング（empirical software engineering）に関しては、日本では、まだまだ模索の段階です。その為、せっかくよい研究、報告、手法、その他の成果物が生みだされたとしても、シンポジウムなどでの発表にとどまってしまい、定期刊行物に掲載し皆とシェアする場がありません。そういう場を準備しておかなければ、SECの活動も小さな輪の中の出来事で終わってしまいます。SECとして何かそのような場が提供できないか、ソフトウェア・エンジニアリングに関する最新、最高の内容が詰込まれたジャーナルを刊行したい。そのような思いで本誌を創刊いたします。幸い、その思いが通じて、優秀な執筆陣により、熱意あふれるジャーナルができたと自負しております。

また、今号では記念論文を募集いたしました。皆様のご応募をお願いいたします。

まだ、創刊号が出たばかりですが、創刊の志を忘れず、たゆまぬ努力で、真剣な討論、議論の場としたく考えております。

皆様のご指導ご鞭撻をお願い申し上げます。　（ま）

SEC journal　第1巻第1号（通巻1号）　2005年1月25日発行

独立行政法人 情報処理推進機構　2005

編集兼発行人　〒113-6591　東京都文京区本駒込2-28-8　文京グリーンコート センターオフィス16階
独立行政法人 情報処理推進機構　ソフトウェア・エンジニアリング・センター　所長　鶴保　征城
Tel.03-5978-7543　Fax.03-5978-7517
http://www.ipa.go.jp/software/sec/

編集・制作　〒101-8460　東京都千代田区神田錦町3-1　株式会社オーム社　Tel 03-3233-0641

SEC journal 創刊記念論文募集

独立行政法人 情報処理推進機構
ソフトウェア・エンジニアリング・センターでは、
下記の内容でSEC journal創刊記念論文を募集します。

論文テーマ

ソフトウェア開発現場のソフトウェア・エンジニアリングをメインテーマとした実証論文

開発現場への適用を目的とした手法・技法の詳細化・具体化などの実用化研究の成果に関する論文
開発現場での手法・技法・ツールなどの様々な実践経験とそれに基づく分析・考察、それから得られる知見に関する論文
開発経験とそれに基づく現場実態の調査・分析に基づく解決すべき課題の整理と解決に向けたアプローチの提案に関する論文

論文分野

品質向上・高品質化技術
レビュー・インスペクション手法
コーディング作法
テスト／検証技術
要求獲得・分析技術、ユーザビリティ技術
見積り手法、モデリング手法
定量化・エンピリカル手法
開発プロセス技術
プロジェクト・マネジメント技術
設計手法・設計言語
支援ツール・開発環境
技術者スキル標準
キャリア開発
技術者教育、人材育成

応募要項

スケジュール

応募締切：2005年4月15日（金）
投稿締切：2005年5月13日（金）
採否通知：2005年7月　1日（金）
ノミネート論文4件選定
カメラレディ締切：2005年　8月31日（水）
最優秀論文審査：2005年10月24日（月）

提出先

独立行政法人 情報処理推進機構　ソフトウェア・エンジニアリング・センター内 SEC journal事務局

その他

論文の著作権は著者に帰属しますが、採択された論文についてはSEC journalへの採録、ホームページへの格納と再配布、論文審査会での資料配布における実施権を許諾いただきます。
提出いただいた論文は返却いたしません。
応募時の個人情報の取扱いは下記のとおりです。
SEC内の審査事務局にて管理し、論文審査に係わる査読委員、審査委員とSECが行う広報活動（論文公募、各種イベントの案内、実態調査など依頼）で使用することを許諾いただきます。

応募様式

所定の応募書式に従って、以下の内容を記載の上、提出してください。

応募者：氏名、所属
連絡先：住所、電話番号、FAX番号、メールアドレス
論文題目
論文概要：200文字以内
キーワード
誓約書
投稿論文様式：情報処理学会の論文誌への投稿論文の様式に準拠

http://www.ipsj.or.jp/07editj/toukou/shippitsu/kaishi.html

記念論文賞

最優秀賞：1名　副賞賞金　100万円
優秀賞：3名　副賞賞金　50万円
副賞賞金は旧「ソフトウェア工学研究財団（RISE）」から継承した基金より充当します。

詳しい内容（＜応募条件＞、＜優秀論文審査委員＞、＜論文査読委員＞等）は下記のURLに示すホームページにてご確認ください。

http://www.ipa.go.jp/software/sec/journal.php

SEC journal No. 1
第1巻第1号（通巻1号）
2005年1月25日発行

編集兼発行人

〒113-6591 東京都文京区本駒込2-28-8 文京グリーンコート センターオフィス16階
独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センター
所長 鶴保 征城
Tel.03-5978-7543 Fax.03-5978-7517
URL:http://www.ipa.go.jp/
定価1,470円（本体1,400円）

IPA
独立行政法人 情報処理推進機構

ISSN 1349-8622