IPA
独立行政法人 情報処理推進機構

ISSN 1349-8622

PRINTED WITH SOY INK
R100 100%リサイクル用紙を使用しています。

SEC journal No.3
第1巻第3号（通巻3号）
2005年8月5日発行

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

SEC journal No. 3

2005年8月5日発行
第1巻 第3号（通巻3号）
ISSN 1349-8622

SEC journal — 3

Software Engineering Center

プロジェクトデータ分析の指針と分析事例
古山恒夫

品質マネジメントシステムの再構築:競争優位性の獲得に向けて
野中誠

ものづくり戦略論とアーキテクチャ—ソフトウェア・アーキテクチャの測定と分析—
立本博文

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.3目次

巻　頭　言

ソフトウェア・エンジニアリング・センター

ＳＥＣ への期待

経済産業省 商務情報政策局
情報処理振興課長　小林　利典

政府は、2001年に高度情報通信ネットワーク社会推進戦略本部を設置し、「e-Japan戦略」の中で、「2005年までに世界最先端のIT国家になる」との目標を掲げました。また、世界に立ち遅れていた我が国ITインフラの整備状況やIT利活用レベルを、次のように表現しました。「我が国のIT革命への取り組みは大きな遅れをとっている。インターネットの普及率は、主要国の中で最低レベルにあり、アジア・太平洋地域においても決して先進国であるとはいえない。また、ITがビジネスや行政にどれほど浸透しているかという点から見ても、我が国の取り組みは遅れていると言わざるを得ない。変化の速度が極めて速い中で、現在の遅れが将来取り返しのつかない競争力格差を生み出すことにつながることを我々は認識する必要がある。」

それから4年余りが経ち、目標達成年である2005年となった今、私たちの身の回りでは、オフィスの業務システムから、情報家電、携帯電話に至るまで、あらゆる製品やサービスがITを利用して提供されるようになっています。

しかし、これまでの取り組みは、いわば環境整備に過ぎません。次の段階は、広く浸透したITを利用して、具体的に何をするのか、どうやって私たちの個人活動や産業活動を豊かにするのか、その中で如何に高いコストパフォーマンスを実現するのか、といったことを考えることが必要です。そのような中で、製品やサービスに価値をもたらすソフトウェアに対しての期待が、今ほど高まっている時代はなく、ソフトウェア産業に対する社会的責任が今まで以上に重くなっていると言うことができるでしょう。

このような背景の下、ソフトウェア・エンジニアリング・センター（SEC）は、2004年10月に発足しました。それから約9ヶ月が経ち、今では我が国の産業界及び学界で蓄積されていた経験やノウハウが集結しつつあります。エンタプライズ系ソフトウェア開発力強化については、1,000件を超える定量データが収集され、システム開発におけるモノサシ策定に向けた取り組みが始まり、また、ユーザ側の経営戦略とシステムを結び付ける最上流工程での基本的な考え方に関するガイドライン、見積り手法に関するガイドラインが策定されたこと等の成果が出てきています。また、組込みソフトウェア開発力強化については、スキル標準やプロジェクトマネジメント、コーディング作法についてのガイドラインが策定され、この分野に関する注目度が大幅に上がってきています。

これらの取り組みは、いわば基礎体力を付けるためのトレーニング的な性格を持つものであり、今後も継続的に実施していくことが重要です。また、これらの取り組みの成果を、いかに産業界において実装し普及させるか、経営戦略に結び付けていけるか等が、SECの今後の活動として課題であると考えています。

今後とも、SECに対して、産業界や学界の関係各位からの御協力及び御指導をお願いすると共に、SECが飛躍・発展し、我が国情報サービス産業の競争力強化、ひいては我が国産業全体の競争力強化につながることを心から願っております。経済産業省としても、引き続き最大限の支援をしてまいりたいと考えています。

経営とIT

日本では近年、様々な業界でIT化が叫ばれ、それに呼応するかのように、企業の経営者はITに対して多額の投資を行ってきた。しかし、バランスシートのスリム化が進む中、これまでブラックボックスだったIT資産にも厳しい目が向かいつつある。これまで高価なシステムに慣れてきたIT会社に対しても、これまでとは異なる姿勢が求められるようになってきた。

今回の対談では、経営コンサルタントの西浦裕二氏をお招きし、「経営におけるIT」、および「IT会社における経営」の両面について語り合っていただいた。今特に必要なのはIT会社の経営が変わること、また、業界の共通の課題について議論する場や、プラットフォームが必要との点で意見が一致した。

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データ通信株式会社取締役開発本部長、同社常務取締役技術開発本部長、NTTソフトウェア株式会社代表取締役社長を歴任し、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～）
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

鶴保　幅広い分野でコンサルティングをしておられる目から見て、今の企業経営者のITに対する認識をどうお感じになっているでしょうか。

西浦　経営者には、ITは会社の命運を左右するものとの認識は間違いなくあります。ITへの投資余力を高めることも視野に入れて経営統合が図られていますが、具体的に打つ手がなかなかつかめない経営者がまだまだ多いのが実情です。

鶴保　そもそもITを活かすための経営的な風土や意思決定のスピードが揃っていないとITが活きてきません。

西浦　年配層の経営者にはまだまだ苦手意識がありますが、最近はITに慣れた若い経営者もどんどん増えてきて、経営者の意識も変わってきています。日常生活にITが溶け込んできている現在、ITに対する「目利き」は急速に変わっていくのではないかという気がしています。

鶴保　eコマース（電子商取引）も着実に伸びています。IT産業がGDPの12％を占め、IT開発の大きな部分をソフトウェアが占めていることを考えると、ITやソフトウェアを経営に活かす、それも高信頼で、かつ低いコストで実現できる技術が今問われているわけですね。

IT資産に対する目が厳しくなる

西浦　この数年、日本の中で起きたことは、バランスシートのスリム化でした。そのためにオフバランス化、流動化、証券化ということが次々行われるようになってきました。しかし、今手が付いているのは主に目に見える資産です。これまで、経営者から見たとき、自社の情報システムはブラックボックスになりがちでした。IT資産は、人件費、リース料、外注費と、項目が全部ばらばらになっていて把握が難しい。たとえわかったとしても、その中で本当に使われているものが何か、費用対効果の点で価値のある部分がどこかを評価するのはさらに難しい。

鶴保　ITに掛かるコストは把握できるかも知れませんが、効果については定量化が難しい。これまで、情報共有システムやデータベースは各社作ってきましたが、それに基づいた意思決定をしているかどうかは疑問ですね。例えば人事異動について、キャリアデータベースに基づこうという流れがありましたが、システムを作っても、あまり使われていないのが実態です。コンビニのPOSデータベースでも、データの分析結果を商品企画や日々の販売予測に活用しているところもあれば、単に売り上げを記録するだけのところもあります。

西浦　これまではブラックボックスであることに甘んじていたのが、バランスシートのスリム化に順次手を付けていく中で、次は当然IT資産に目が向いてきます。効果を発揮していないものに対する目が非常に厳しくなるし、そういう事に対する感度のよい人が経営者になっていくでしょう。

鶴保　これまでITベンダは高価なシステムを作ってきましたが、これからはユーザがコスト意識に目覚め、IT資産に対する目が厳しくなってくるわけですね。そのような中で、淘汰というか、対応できるベンダとできないベンダに分かれてくるでしょう。

西浦　わかりやすい例が過去の銀行のシステム投資です。日本の銀行は例えて言うと、毎回、戦艦大和を建造し直してきたようなものです。これに対して、例えば米国のシティバンクのシステムは駆逐艦の集合といってもよく、その中には古い駆逐艦が長年にわたってそのまま走っている場合もあります。環境や規制の変化で、変えるべきところだけを変え、その間をミドルウェアできちんとつないでいくという考え方が徹底している。シティバンクの情報システムの強さはミドルウェアの強さです。ミドルウェアが強いから、異なるシステムを統合するのがうまく、合併や買収、経営統合に迅速に対応できる。日本では銀行それぞれの大きな船を作るので、統合するのがやたらに大変で、トラブルも起きやすくなります。

鶴保　2000年に新銀行として発足した新生銀行も、新生銀行としてシステム全体を新規に作り直したのではなくて、従来のものをできる限り活用していますね。

西浦　使えるものは使い倒して行こうという考え方です。シティバンクは日本の銀行に比べると、開発スピードがとても速い。基本的な考え方が違うといっても過言ではない。それがコストの低下につながり、システムの柔軟性につながっている。費用対効果もより見えやすくなっています。

深刻な現場力の低下と経営ポリシーの重要性

鶴保　外国との比較で言うと、日本は製造業が強いといわれていますが、この点についてはどうですか。

西浦　これまでは確かに製造業に強みがありましたが、いろいろな意味でその足元が脅かされています。1つはコストの問題で、製造コストの安い中国等に脅かされている状況です。しかしそれ以上に、現場の力が低下している問題の方が深刻ではないかと思います。よく、日本の現場は意識が高いといわれます。象徴はトヨタですが、現場の人たちが自ら改善し、何かトラブルがあれば自ら水際で感じ取って対応してきました。したがって歩留まりがものすごくよく、それが日本の製造業の強さの源だったわけです。しかし、経済が減速していった中で、多くの企業がコストを下げるために人減らしをし、正社員をパートに置き換えたりしました。その結果、現場のモチベーションが下がってきています。特にここ5年くらい、トラブルを未然に防げず、また問題が社外に簡単に流出してしまう例が目立ちます。これは、今まで現場の人たちが、自分の問題、責任としてトラブルを未然に防いでいたのが、自分の問題じゃなくなってきたのが最大の原因ではないでしょうか。

鶴保　現場の力が低下しているのは、リストラの影響でしょうか。

西浦　経営者の短期的な視点によるリストラの影響だと思います。現場の力を認識している経営者は、スリム化すべきところはスリム化しますが、人は重要な資産であり、その人に知恵が宿っているということで、安易なリストラをすることには非常に注意しています。

西浦　裕二（にしうら　ゆうじ）
株式会社ローランド・ベルガー 取締役共同会長
一橋大学社会学部卒業。住友信託銀行、ボストン・コンサルティング・グループ、シティバンクを経て、ブーズ・アレン・アンド・ハミルトン代表取締役社長。2003年10月にローランド・ベルガー 代表取締役兼CEOに就任、2005年1月より現職。コンサルタントとしては、情報通信、金融、製造、流通など幅広い分野を手がける。近年は、「企業の生まれ変わり」に関するテーマに取り組むことが多い。著書に「金融マーケティング」（東洋経済新報社）、「経営の構想力」（東洋経済新報社）。対談集に「知的財産を語る」（雄松堂出版）。1953年生まれ。岐阜県出身。

鶴保　ソフトウェア会社に関して言うと、リストラとともに、アウトソーシング（外注）の問題があります。もちろん会社によって得意不得意があり、議論の上でアウトソーシングをするのはよいのですが、単にコスト削減だけでやっていることが多い。その結果、現場でその仕事をしていた人が挫折感を味わい、弱体化につながっているのではないかと感じています。

西浦　ユーザからのソフトウェアに関する要求が厳しくなってきて、昔のように巨大で高価なシステムが売れなくなってきました。開発コストを下げる必要が生まれてきて、なるべく下請けに出していこうという流れがあります。しかし、元請けである自社にどういうコアコンピタンスが残っているのかというと、何もないのではないか。自社の存在意義が明確に認識できないような会社がソフトウェア会社に多くあるのではないかという気がしています。

鶴保　ソフトウェアの場合は、ねじや釘を外から調達するようなわけにはいかない。ソフトウェアのどの部分を外に出すのかは本来慎重な議論があるべきですが、それを抜きにして、設計は外に出さないが、製造は外に出してもよい等と、あまり根拠なく決めているようですね。しかし、東京大学の西村清彦教授の研究のように、これまでの常識とは逆に外注比率が高くなると生産性（利益率）が下がるという結果もあります。IT業界はこういう結果にもっと注目すべきだと思います。

西浦　切り出すときの基準、きちんとしたポリシーがあって、自分たちに残すべきコンピタンスがこれだという考え方がはっきりあればよいのですが、短期的なコスト削減や利益効率のために安易に外に出したりダウンサイズしたりしているのが気になります。

鶴保　スーパーマーケットの売場等、正社員をパートに置き換えるのが常識のところでも、正社員を残して成功している例もありますね。東京や神奈川でチェーン展開している食品スーパーのオオゼキが典型的な例ではないでしょうか。

西浦　正社員と外注に関しては、どちらの比率を高くするのが正しいということではなく、我々はこうしていくのだという経営者の理念が明確であればよいと思います。オオゼキは、徹

底的に地域に溶け込んでいくこと、溶け込んでいくためにはお客様の顔を覚えなくてはいけない。覚えるためには各店にコミットする正社員のほうがより望ましいという考え方で、正社員比率が非常に高い。そのことによって、地域ごとには非常に評価の高いスーパーとなっています。要は経営者がどういう理念で経営しているかということです。

ソフトウェアの価値をどう測るか

鶴保　ソフトウェアの価値をどう測るかという問題もありますね。ソフトウェアのシステム作りはよく建築にたとえられますが、基本的な作り（構造）や結果の生産物が決まっていればソフトウェアもハードウェアと同じように価値が測れます。しかし、ソフトウェアは一品料理的なものが多くあり、建築よりむしろ彫刻に近いのではないかという議論もあるくらいで、そういう場合には価値を測ることは難しい問題ですね。

西浦　価値を測りにくいという点では、私が携わっているコンサルティングの仕事も同じことが言えます。ただ、戦略策定のコンサルティングの場合は、効果が出てくるのは何年も先になる場合があります。ソフトウェアは実際にオペレーションの合理化という観点であれば、どうオペレーションが変わったかは測定しやすいはずです。ソフトウェアの価値をどう測るかは非常に大きな課題で、1つひとつの会社では答えが出ないのではないでしょうか。何がしかのモノサシ、考え方がこれから出てくることを期待しています。

鶴保　すべてのソフトウェアを価値で測るのはおそらく難しいでしょう。人事管理、社員管理、顧客管理等をシステム化した場合、それをやったらどれだけROI（return on investment）があるのかというのかは測りにくい。必要経費的な側面はある程度認めざるを得ないのではないでしょうか。

西浦　いくつかのタイプに分類する必要があるのかも知れません。情報系と勘定系では当然違うし、情報系でも何の目的かということで違う。いくつかのタイプに分類した上で、価値を測る考え方、共通のモノサシが出てくると議論がわかりやすくなります。ユーザの側も、そう簡単に答えが見つかるとは思っていませんが、ソフトウェアの価値について語ることのできる会社や経営者と付き合っていこうと考えるようになるはずです。ソフトウェア会社から提示された価値の枠組みに賛同するかどうかはユーザの問題ですが、それも語れずにわが社は何ヶ月で作ります、総額いくらですというだけでは納得されなくなるのではないでしょうか。

鶴保　例えば、これからeコマースをやろうという場合、月々1,000万の利益が上がってくれば1,000万のリターンがあり、1億円の利益があればそれだけのリターンがあるということになります。同じeコマースでもリターンは大きく違います。1億円くらいのリターンがないとIT化できないというのではなく、1,000万円のリターンだったら、ROIを出すために500万円のシステムを作らなければならないと思います。それは技術の問題になります。投資許容額でシステムを作るということが重要になってきます。

西浦　そのときに、手作りなのか、パッケージを使うのか、あるいはまとめて委託するのか等、ユーザとしての選択肢につながっていくわけですね。

技術力とコミュニケーション能力の両方が問われる

西浦　これまでは、ユーザの要求に沿って、ソフトウェア会社は与えられた時間と与えられた費用の中できちんとやればよいという時代が長く続いてきました。そうこうしているうちに、いくつもの業界で、顧客側の産業構造の変化、業界再編等により、ソフトウェア会社が仕事を失ってしまうということが起きています。例えば、百貨店業界はこの5年間ぐらいですごく変わり、今4系列ぐらいに統合されようとしています。その中でITがどう使われるかは、過去の経験しかない人ではわかりません。これからユーザとのコミュニケーションで問われるのは、1つはソフトウェアの価値をどう説明していくか、もう1つは、ユーザが置かれている環境を理解し、ITに関する需要がどう変わっていくかを考えながら進めていくことです。下請けに徹するのであればそれはそれで残れるかも知れませんが、低賃金に甘んじる、あるいは海外の低コストのソフトウェア会社に代替されていくということになってしまうかもしれません。ソフトウェア産業の中できちんとした地位を固めていこうと思えば、顧客の産業動向の変化に敏感になっていかざるを得ません。

鶴保　ソフトウェア会社にとって、そこが難しい点です。対顧客折衝やマーケットを洞察する能力と、正確にソフトウェアを設計する、あるいはコーディングするという能力は、持つべき素質が違います。設計や製造の方は論理的な深さが要求されますが、対顧客折衝等は、幅広くいろいろなことを知って判断しなければならない。同一人ではおそらく不可能ですが、会社や事業部としては同時に兼ね備えることが必要となります。ITコーディネータ制度というのがありますが、いまひとつ機能していない。私が大阪で少しお手伝いした経験で言うと、保守的な現経営者と積極的にIT化を進める後継者というような対立があった場合、普通のITコーディネータは踏み込んでものが言えず、現経営陣にひっぱられてしまうことが多いのではないかと思います。それだと、十分なIT活用の図が描けないということになります。

西浦　ITコーディネータに関連して言えば、独立した立場、客観的な立場できちんとITの構築、運営に関して企業にアドバイスしていくという存在は必要です。ソフトウェア会社でも技術的な側面だけでなく、ユーザとのコミュニケーションをきちんとできるような人が経営の核として存在していれば、従来とは違うユーザとソフトウェア会社の関係ができるかもしれない。今は、ユーザの側にもソフトウェア会社の側にも、技術的な面とコミュニケーションの面、両方の役割を果たせる人が極めて少ない。したがって、ITコーディネータが両者の橋渡しをしてくれれば理想的ですね。

鶴保　ソフトウェア会社によって得手不得手はどうしてもあります。上流は強いがインプリメントは弱いとか、インプリメントは強いがビジネスモデルを語るのが苦手とか、そういうのはどうしてもありますが、これからは両方を兼ね備えることが必要になってくるわけですね。

共通の課題を議論できる場を

西浦　日本の百貨店の利益率はアメリカ等に比べて低いと言われています。ひとことで言うと、日本の百貨店は個店経営の集合体で、それぞれの各店が努力するという意味ではよいけれど、コスト構造では無駄もあります。今それが急速に変わろうとしています。地方の百貨店の系列化ということも含めて、オペレーションのプラットフォームを作っていき、地域の百貨店で自立し得ないようなところをその下にぶら下げていこうという動きが強まっている。それぞれの地方百貨店で言うと、看板は今まで通りでも、商品開発、物流、顧客管理等はプラットフォームとして共有化していこうということです。こうしたオペレーションプラットフォームの統合化は急速に進んでいくと思います。そういう中で、ソフトウェア会社の側から改革を仕掛けていく気概があってもよいと思います。石油の元売りの業界等では、競合社同士でも一部の業務の統合が進んでいます。最下流のところでは、顧客との接点はそれぞれの会社でやりますが、物流等の上流の部分は一緒にやっている例が多くあります。他の産業でも、競争力を左右しないところは一緒にやっていこうという例が増えています。

鶴保　それこそプラットフォームですね。ソフトウェアの世界でも、昔からネットワークについては、対象としている業界全体に対して売り込むということがありました。それはネットワークの特殊性をはじめから気が付いていたわけです。今度は業務アプリケーションの時代になってきて、プラットフォームを作った上で、競争するところはその上で競争する時代が来ています。しかし、なかなかそのプラットフォーム作りが進んでいない。

西浦　日本のソフトウェア産業が、本当の意味で競争力を回復していくためには、技術的な生産性向上のあり方と、経営のあり方の2つの観点からソフトウェア会社、ソフトウェア産業が見直されていかなくてはいけないのではないかという気がしています。経営とITという点で言えば、今特に変わらなければいけないのはソフトウェア会社の経営だと思います。

鶴保　おっしゃる通りです。今ソフトウェア会社も経営の端境期に来ている。ビジネス環境もそうだし、ソフトウェア産業の経営者を見ても、第一世代の創業以来40年くらい経っています。次の世代になって、従来のように独立して群雄割拠してやっていけるのかどうか疑問だと思っている経営者がかなりいます。

西浦　優秀な経営者は自分で考えていくと思いますが、個々のソフトウェア会社の経営努力に任せるだけでよいのかという気もします。中堅のソフトウェア会社だと、情報収集力や分析力は十分とはいえません。ソフトウェア会社の経営をどうしていったらよいのか、どういうビジネスモデルがよいのか、それも決して単一ではなく、わが社としてはどのビジネスモデルを目指していくのかといった、よりオープンな議論や検討があってもよいのではないかと思います。それはSECかどうかはわかりませんが、経済産業省あるいは国が音頭をとっていかないと、技術面ではよくなったが、経営破綻しましたということにならないのかという危惧があります。

鶴保　これからは、それぞれの現場ごとに効率化すれば全体がよくなるというわけにはいかなくなり、会社全体で考えていかなければなりません。官のほうも、個々の企業育成だけでなく、全体を最適にするという発想で業界を指導していくことが必要になってきます。プラットフォームの考え方はその点でも重要ですね。

西浦　国がどこまで民間の経営に口を出すのかというのは非常に難しい問題ですが、ソフトウェア産業、あるいはIT産業というのは日本の経済全体の命運を非常に大きく左右すると思うので、産業に共通に横たわる課題に関しては、議論の場をきちんと作っていくようなことをしていかないと、本当の意味での産業の競争力回復にはつながらないのではないでしょうか。

鶴保　官が正面きって個々の会社経営のあり方を議論するのは難しいですが、ソフトウェア業界共通の技術的な流れや課題については経営者に理解してもらえるよう、さまざまな機会を設けていきたいと思います。

文・松井利行　　写真・越昭三郎

論文 paper

プロジェクトデータ分析の指針と分析事例

A guide to analyzing software project data sets including relevant examples of data set analysis

東海大学理学部教授
工学博士
古山　恒夫

Tokai University,School of science,Professor,Dr.of engineering
Tsuneo Furuyama

ソフトウェアプロジェクトデータは，多岐にわたるデータ項目を実際の開発現場から多くの場合人手で収集しなければならない事から，（1）欠損データが多くかつ一般に個々のデータの精度が低い．また（2）多くの要因が規模と高い相関を持つ等の特徴を持つ．本稿では，これらの特徴を持つデータを分析する際の一般的な注意事項を述べる．また，2004年度ソフトウェア・エンジニアリング・センターが収集したエンタプライズ系ソフトウェアのプロジェクトデータに関する試行分析事例を通して，ソフトウェアプロジェクトのデータの特徴に着目したデータ分析方法およびデータ分析の際に便利な統計指標の目安を紹介する．

This paper describes the items to which data analysts must pay attention when analyzing a software project data set having three types of characteristics. (1) The data set generally includes a significant amount of loss data & precision of the data is sometimes lower than that of data collected in other engineering areas such as mechanical engineering. (2) Most project-related factors such as development efforts are strongly correlated to product size. This paper also introduces some typical methods that are applied to analyzing the enterprise software data set collected by the Software Engineering Center last fiscal year, as well as some statistical indices that are convenient for data analysis.

Key Words & Phrases： データ分析，ソフトウェアプロジェクトデータ，エンタプライズソフトウェア，生産性の分析，影響要因の分析
data analysis, software project data, enterprise software, productivity analysis, effective factor analysis

1　はじめに

ソフトウェア開発を科学的にかつ効率よく進めていくためには開発の様々な側面での定量化が重要である[1]．そのためには，ソフトウェア開発プロジェクトにおけるプロセスとプロダクトに関する様々な属性を測定し，得られた測定データを分析する必要がある．また，分析結果を有効に利用するためには，それらを経営者，プロジェクト管理者，共通部門の担当者などにフィードバックする必要がある．分析の結果は，(1) 計画作成時には，見積もり，プロジェクトチーム編成，プロジェクト環境の構築などへの有用な指針となり，(2) プロジェクト開始後は，効率的なプロジェクト運営に役立てる事ができる．また，(3) 複数の組織間でソフトウェア開発について議論する際の共通の土台を構築できるようになる．

ひとくちに測定と分析と言っても，測定すべきデータ項目や収集したデータの分析方法は，組織やプロジェクトの構成や目的によって様々なものが考えられる．例えばデータ分析方法についてみれば，年間の総開発規模をプロジェクト数で割って1プロジェクトの平均規模を求めるだけの簡単なものから，生産性や信頼性に影響を与えるプロジェクト要因を，統計的理論に基づいて誰もが納得する形で抽出し，その結果に基づいて工数の予測式を作成する事まで様々なレベルがある．ソフトウェアプロジェクトデータ（以下では単に「プロジェクトデータ」と呼ぶ）の精度は，物理現象を対象とする工学の分野で扱われるデータの精度に比べて一般に低いため，明確な分析結果が得られにくい事が多い．また，分析で使われている統計的手法も比較的簡単なものに止まっている事が少なくない．ときには，統計的理論を必ずしも十分理解せずに用いているケースも見受けられる．

プロジェクトデータ税金論

データの収集は開発現場にとって負担である．しかしそれは効率的なプロジェクト運営には必須であり，あたかも税金徴収のようなものである．税金が軽く，結果が現場にフィードバックされるなら，「納税者」も納得する．目的の明確なデータを収集し，結果を現場にフィードバックすることが大切である．最初はSmall startで始めるのがうまくいく秘訣ではないかと思われる．

プロジェクトデータの測定と分析は，一般に次のような手順で行われる[2]．

(1) 分析目的の明確化（何を知りたいのか）

(2) 分析対象のモデル化（定量化の前に，まず定性的なモデルを構築）

(3) 測定量の設計（どのようなデータ項目をどのように収集するのか）

このとき収集するデータの有効性とデータ収集の困難さのトレードオフを考慮する事も大切である

(4) データの精査（得られたデータの質は確かか）

データの質には偏りのなさと精度の2つの側面がある．前者はバイアスの小ささ，すなわち標本データの平均の期待値が母集団の平均にどれだけ近いかを意味する．後者は有効桁数の大きさ，すなわち標本データの分散がどれだけ小さいかを意味する．報告されたデータには転記ミスが含まれている可能性が少なくない．収集データを一覧表や散布図にして眺め，また各種の基本統計量を得る事から異常データを発見し，報告元に問い合わせて報告データに間違いがないかどうか確認する事が必須である．

(5) 適切な統計手法の選択とデータ分析手法の前提条件の確認

統計手法は分析対象の真の姿を浮き彫りにさせる科学的な手段であり，統計学に基づいたデータ分析は対象の真の姿を客観的に裏付けるためのものである．当然の事ながら「統計学をごまかしのために用いる」ような事はしない事が重要である．また，統計学的なデータ分析手法には与えられたデータにさまざまな前提条件があるのが普通である．それを忘れてしまうと，意味のない分析を行う事になってしまう．

(6) 分析結果の意味付け

当初の仮説通りではなかったという結果も含め，分析目的に適った結果が得られたのかどうかの確認をする必要がある．

(7) 分析方法の適切さ・データ（測定量とその値）の適切さの確認・修正

分析目的に適っていなかった場合は，測定量は適切であったか，データの精度は十分であったか，採用した統計手法に問題がなかったかなど，その原因を究明する必要がある．分析目的に適っていた場合でも，よりよい測定量，より適切な分析方法がないかを再度検討する事はその後のデータ分析にとってプラスとなる．

本稿では紙面の都合もあり，残念ながらこれらすべてにわたって述べる事はできない．(4)～(6) に関連する事項のみを紹介する．具体的には，昨年度SECが収集したエンタプライズ系プロジェクトのデータに関する試行分析事例を通して，プロジェクトデータの分析における統計的理論の用い方とその手順，データ分析の際に便利な統計学的指標の目安，データ分析上の注意事項をいくつか紹介する．

2　プロジェクトデータの問題点と特徴

プロジェクトデータは，電気工学や機械工学等，他の分野の測定データに比べて，一般に次のような問題や特徴を抱えている．

(1) データが収集しにくい

ソフトウェア開発が人間の頭脳労働によるものであるため，化学プラントのようにセンサを使ってデータを自動収集することは困難なものが少なくない．例えば，作業内容と作業時間に関する1次データは個々の担当者が手作業で入力するのが一般的である．

(2) 欠損データが多い

多くのデータ項目を完全に収集するには，かなりの工数を必要とする．そのため，データ収集のコストに見合った分析結果が得られる保証がないとデータが集まらな

尺度について

尺度とは，「連続的若しくは離散的な値の順序集合又は分類の集合で，それに属性を対応付けるもの」である[3]．SEC収集データの尺度は，名義尺度，順序尺度，比尺度に分類される．名義尺度の例としては，スタンドアロン，2層クライアントサーバ，3層クライアントサーバ等に分類される「アーキテクチャ」がある．これらの分類には順序関係は存在しない．順序尺度の例としては，要求仕様が非常に明確，明確，ややあいまい，非常にあいまいという4つの階級を持つ「要求仕様の明確さ」という尺度がある．これらの階級には順序関係が存在する．ただし，それぞれの階級間の差が同じであるということは保証されていない．比尺度の例としては，主たる測定量である規模，工数，工期等がある．これらにはある単位が存在し，属性にはその倍数となる値が割り当てられる．

中央値のススメ

測定データの「中心」を特徴付ける統計値として，「平均値」，「中央値」，「最頻値」がある．このうち「最頻値」が利用されることは少ない．「平均値」は外れ値に影響を受けやすいという問題点を抱えていることはよく知られている．とくに規模や工数のように非対称な分布をするデータでは「平均値」と「中央値」が大きく異なることになる．それに対して「中央値」は全データを大きさの順に並べ替えてその真ん中に位置するデータの値をとるものであり，外れ値の影響をほとんど受けない．日常的な感覚でいえば，最も典型的なプロジェクトのデータである，ということもできる．したがって，プロジェクトデータでは，「平均値」よりも「中央値」を代表値とすることをお勧めしたい．

い．また，データを収集すべき役割を持った担当者が高い「測定意識」を持たないと必要なデータを収集することは難しい．定義されたすべてのデータ項目を収集できるプロジェクトは稀である．

(3) データの精度が低い

収集データ項目の厳密な定義が難しいことが多い．例えば，ソースコード行数にコメントや空白行を加えるか否か，1行に2文書かれているようなプログラムと1行1文のプログラムの違いを区別するのかしないのか，という問題がある．作業工数の場合は，作業項目や工程区分の定義が組織間で異なるという問題や，プロジェクト管理工数の算出方法，ユーザ関与の作業工数の算出方法をどこまで明確に規定できるか，という問題がある．また，個々の担当者の意識が十分でないと1次データの精度が低くなる場合がある．恐らく多くのプロジェクトデータの相対誤差は1/100程度，どんなによくても1/1,000より小さな値になることはない思われる．

(4) 人間の主観に基づくデータが多い

多くのプロジェクト特性は，質問表の回答結果に基づいて定められる．回答すなわち収集データはレベル付けされた順序尺度であることが多いが，各レベルの境界は実際上必ずしも明確でない．例えば，「要員のスキル経験」を4段階に分けても，具体例を明示しないと評価結果が回答者ごとに異なる．また，もともと離散的な4段階のレベルに明確に分けるのは容易ではない．これはデータ精度の低下にもなり得る．

(5) データは非負の非対称な分布をなすことが多い

規模, 工数, 工期等多くの収集データは正またはゼロの値をとり, その分布は右裾が長く延びる非対称形をしている. そのため平均値と中央値が大きく異なることが多い．

(6) 規模が主要な説明変数である

工数, 欠陥数, 工期等主要な目的変数に最も影響を与える要因の最大のものは規模である（例えば4.4項を参照）．

(7) 影響要因が規模と相関を持つことが多い

例えば，大規模プロジェクトの管理者に未経験者を割り当てることはなく，プロジェクト管理者（PM）のスキルレベルという説明変数は開発プロジェクトの規模と無関係ではない．

3　データ分析における一般的留意事項

この章では，データ分析に関する留意事項のうち，多くの統計学の教科書では述べられていないか，十分な説明があるにもかかわらずソフトウェア工学の分野では浸透が不十分と思われる点について述べる．

(1) データの収集と分析の目的を明確に

思いつくままに多くの種別のデータを収集し，例えば順次データ項目間の相関係数を計算していくと何か特徴的な関係が得られるかもしれないという方法は，あまり効率的なデータ収集・分析の方法ではない．何のために，あるいは何を知りたいがためにデータを分析するのかを明確にしてからデータを収集し，分析するのがよい．そのためには，しかるべき書物（例えば参考文献[2],[4],[5]等）や各自の経験をもとに，データ収集・分析モデル，言い換えると各データ項目の相互関係を明らかにし，データ分析にあたっても仮説を立ててから行うとよい．

(2) 適切な層別を

層別は異なった属性を持つものを混在して分析することにより，分析結果があいまいになることを防ぐために行う．例えば，システムソフトを開発する場合とアプリケーションソフトを開発する場合では，開発条件が異なり，したがって生産性の傾向が異なると考えられる．生産性を分析する際には，それぞれ別々に分析した方が有用な結果が得られる．

(3) なるべく生のデータを分析対象に

収集データはなるべく加工せずに分析する方がよい．例えば，PMのスキルレベルと生産性の関係を分析する場合に，工数／規模で表した生産性を目的変数とし，PMのスキルレベルを説明変数とする2変数で分析すると，割算によって規模情報が失われ，その結果必ずしも正確な結論を導き出すことができなくなる恐れがある．このような場合は，例えば工数を目的変数とし，規模とPMのスキルレベルの2つの変数を説明変数として分析するとよい．

(4) まず，2変数間の分析を

ある目的変数に影響を与える要因を分析する代表的な方法に回帰分析や分散分析がある．最初からすべての説明変数を組み込んで，それらの分析を行う方法もあるが，目的変数に影響を与えない説明変数が多くなると，分析結果の精度が低下することが知られている．また，重回帰分析において大きな相関を持つ複数の説明変数を採用すると，結果が不安定になり，ときにはそのうちの1つの説明変数が負の回帰係数を持つことすらある．

これまでの筆者の経験によると，プロジェクトデータの分析では，すべての変数に対していきなり重回帰モデルや数量化I類等で分析しても，ある目的変数，例えば工数に影響を与える要因を統計的な基準に従って抽出することが難しい[6]．その理由としては，①多くの欠損データがあること，②規模の影響が他の要因に比較して格段に大きいこと，③影響要因の低いデータ項目が少なからず存在すること，④規模と相関の高い項目が存在すること等が挙げられる．

したがって，まず，着目する説明変数が規模と相関が低いことを確認した後，目的変数との相関を調べることにより，影響の高い要因を1つひとつを拾い出してから，それらの要因の相互関係を重回帰分析や分散分析で調べていく方法がよいのではないかと考えている．

(5) 分析の前にデータ群の特性の把握を

統計的分析の前に必ず分析対象とするデータ群を眺めておくのがよい．例えば，相関分析や回帰分析を行う前には必ず散布図を眺めることをお勧めする．それにより，分布の異常さをいち早く発見できる．例えば，1つのデータ群と思っていたのが，実は2つのグループに分かれている，というようなことも発見できる．

(6) 相関係数の信頼度に留意せよ

データの因果関係の分析で広く用いられているのは，2つの要因間の関係の強さを調べる相関分析であろう．関係の強さを表す相関係数rの二乗は，寄与率，すなわちある要因がどの程度目的変数の変動に寄与しているかを示す指標と等しくなる．例えば$r=0.8$であれば，寄与率は0.64となり，目的変数の64％がその説明変数の影響下にあるということを示す．

相関分析でよく見受けられる第一の誤りとして，「相関係数が高いから母集団における2つの変数間に相関がある」と断定してしまうことがある．標本から計算される相関係数は確率変数であり，上記の例では，母集団での相関の度合いは0.8が最も確からしいが，実は0.5かもしれないし，ゼロすなわち無相関かもしれないのである．したがって，相関係数は，その絶対値の大きさだけでなく，母集団では無相関である確率（P値）も常に意識しておく必要がある．P値が小さいほど母集団に相関がある確率が高くなる．P値が0.05より小さければ，危険率5%で母集団に相関があるといってよい．

相関分析に関する第二の誤りとして，「相関係数が低いから2つの変数間に相関がない」と断定してしまうことがある．rの絶対値が小さくてもP値が十分小さければ，「影響度は小さくても，確実にその説明変数は目的変数に影響を与えている」ということを理解すべきである．

P値は，標本数nと相関係数rで決まる．nが大きくなれば，危険率5％のrの限界値（の絶対値）は小さくなり，nが小さくなればrの限界値（の絶対値）は大きくなる．例えば，相関係数が0.8でも標本数が5しかなければP値は0.12になり，危険率5%では有意ではない．

(7) 疑わしきは経過観察

例えば，相関係数rのP値が0.1（10％）で0.05（5%）より大きい場合，2つの変数の間に相関はないと断定すべきかどうか？　一般に，「母集団のパラメータ（例えば平均値）の推定精度はnの平方根に比例して向上する」ので，もし2つの母集団同士で本当に相関がある場合は，標本数を増やすに従ってP値は漸次小さくなる．nが小さいためにP値が大きいと思われる場合は，それら2つの変数間の相関係数は経過観察すべきである．

(8) 検定／推定の前提条件を忘れずに

分析した結果が統計的に有意であるかどうか，あるいは推定値の信頼幅を議論するためには，検定や推定を統計学の理論に基づいて行う必要がある．これが有意性の検定で，よく利用されるものにt検定，F検定，χ^2検定がある．注意すべきは，これらの理論は（6）で述べた相関係数の有意性の検定を含めてすべて「母集団が正規分布である」ことを前提としていることである．したがって，検定や推定を「厳密に」行うためには，対象とするデータの母集団が正規分布であることを検定する必要がある．

ただし，毎回正規分布の検定をするのでは，それだけ

検定の目安

危険率5%で有意性の有無を判断する目安として次の2つの値は便利である．

(i) 相関係数：標本数が30の場合，相関係数が0.36　1／3以上あれば，母集団に相関があると言える．

(ii) t検定：2つの標本について，それぞれの平均がμ_1，μ_2，標準偏差がσ_1，σ_2，標本数がn_1，n_2であるとする．n_1，n_2がそれぞれ30以上の場合は

$$t\text{値}=|\mu_1-\mu_2|/\sqrt{\sigma_1^2/n_1+\sigma_2^2/n_2}>2.0$$

が成立すれば，2つの母集団の平均値に差があると言える．標本数が無限大の場合は1.96，10ずつでも2.1を超えれば有意であることから“2”という数字がおおまかな目安となる．

重回帰分析における要因の選択方法

(1) すべての組み合わせを調べる総あたり法，(2) 説明変数が1つも含まれない場合から始めて順次1つずつ説明変数を追加していく方法，(3) 説明変数の候補すべてを含めた場合から開始して不適切な説明変数を1つずつ減らしていく方法がある．プロジェクトデータの場合は，経験上(2) がよさそうに思える．

で分析工数が大きくなる．実際には，非常に重要な判断を下すため，あるいは共通の土台で評価結果を比較する等，分析結果の厳密性が要求される場合だけ検定を行えばよい．ヒストグラムを人間の目で見て，正規分布とみなせれば，概ね検定を行わなくても大丈夫であるし，多くの分散分析で行われるようにあらかじめ各水準の分布は正規分布と仮定することも少なくない．

(9) 必要なら変数変換を

工数も規模も変数変換をしない生のデータは正規分布をしていない．検定や推定を厳密に行うためには，もとのデータを変数変換して，分布が正規分布になるようにしなければならない．もちろん，生のデータのままで，それぞれの平均をとり，回帰分析を行うこともできる．それ自体は統計学的な誤りはないが，その結果は値の大きなデータに引きずられ，一般的な感覚，言い換えると欲しい情報と違ってくる危険性がある．とくに回帰分析の場合は，一般に規模が大きくなれば工数の回帰からの誤差も大きくなると考えられ，生データのままでは「各データの回帰からの誤差の分散は等しい」という回帰分析の仮定自体を満たしていないと考えた方が自然である．

多くのプロジェクトデータは，対数変換により正規分布に変換することができる．正規分布の平均値はもとのデータ群の中央値とほぼ等しくなり，結果は感覚と一致することが多い．

(10) 統計学的基準に振り回されない

統計学的基準には単なる約束に過ぎないものがある．代表的な例は有意水準である．「危険率5%で相関係数が有意である」という意味は，「2つの変数が独立で本当は相関がない場合でも，計算で求まったような相関係数が得られる可能性があるが，それが高々20回に1回（5%）と小さいから母集団に相関があるといってもよい」ということである．

それでは，P値が6%の場合，あるいは10%の場合，20%の場合はそれぞれどのように解釈すればよいのであろうか．危険率5%という基準に従えば，これら3つの場合はすべて「相関があるとはいえない」ことになる．共通の土台で何かを議論する場合は，議論を統一的・普遍的にするために，有意水準を5%や1%で線引きをするのは重要である．しかし，5%と6%で決定的な違いがあるのか，5%を境にして片方は「相関があると言って差し支えない」，それを1%でも超えると「相関があるとは言えない」というように分けてしまわないといけないのだろうか．

プロジェクト運営をする場合には，そのような厳密な線引きに必ずしも従う必要はないと思われる．P値が6%となった場合は，「相関があるとは言えない」のではなく「本当は相関がない可能性が6%＝1／16程度あるかもしれないが，かなりの高い確率で相関があると思われる」と解釈して利用すべきである．

(11) 分析結果は利用者の立場で

――数学的な有効桁数と読者の認知レベルのバランス

統計分析ツールを使うと多くの桁数を持つ結果が出力される．しかし，分析結果を報告するときは，数学的な有効な桁数と報告書の読者が認識・記憶できる桁数とを考慮して決めるべきである．計算結果の数学的な有効数字が4桁以上あることは稀である．例えば，精度が1/100のデータを100個集めて平均をとっても，その精度は1桁しか上らず，せいぜい3桁になるに過ぎない．例え有効数字が数学的に4桁あったとしても，4桁の数字を記憶できる経営者や管理者は多くない．最終結果の有効数字は2桁から3桁で十分である．

――有効数字は1桁減らして採用

分析対象のプロジェクト群とこれから運用しようとする個々のプロジェクトでは，いくら表示された条件が同じように見えても条件はどこかで異なるものである．したがって，分析結果を自分たちのプロジェクトで利用する場合は，数字（有効桁数）をそのまま使うのではなく，1桁程度少ない数値を用いるのがよい．例えば，生産性が3.62（単位は省略）で示された場合は，3.6までは信じてよい，ということである．生産性が3.6で示されている場合は4という数値を用いるのがよい．

4　分析手順の例（回帰分析）

経営者や管理者が最も大きな関心を持つと思われるものに生産性がある．生産性の分析では，工数と規模という，2つとも比尺度となるデータ項目を用いる．生産性は，工数／規模で定義する方法とその逆数である規模／工数で定義する方法があるが，ここでは規模あたりの欠陥数や規模あたりの工期と同様に，規模あたりの工数という考え方でSEC収集データのあるデータ群（標本数は225）について回帰分析を試みた例を紹介する．

4.1　ヒストグラムによる生データの分布の確認

ヒストグラムはExcelのグラフ機能を利用すれば簡単に得られる．今回分析対象としたデータは，規模，工数ともに右裾の長い非対称の典型的なプロジェクトデータの分布を示した．

4.2　変数変換

信頼幅の議論を厳密に行うために，工数と規模の分布が正規分布となるように対数変換する．対数変換後の規模と工数の基本統計量を表1に示す．平均値と中央値がほぼ等しいことがわかる．また，対数変換後の規模のヒストグラムを図1に示す．

対数変換後の規模（Log（FP））の平均値μは2.79（もとの規模に直すと616FP），標準偏差σは0.47である．95%の範囲にある（対数変換後の）規模の範囲は，μを平均，σを標準偏差とするとおおよそ[μ－1.96σ，μ＋1.96σ]=[1.87, 3.71]であり，元の規模に直すとおおよそ[70FP, 5500FP]となる．

4.3　正規分布の検定

図2からLog（FP）は正規分布になると思われる．χ^2検定を用いた正規分布の検定結果を表2に示す．

$$\sum_{i=1}^{n}(O_i-E_i)^2/E_i=0.43<5.99 \qquad (1)$$

であることから，log（FP）の母集団の分布が正規分布であるという仮説が有意水準5%で棄却されない．ただし，E_iとO_iはそれぞれ正規分布の各領域における理論的頻度と実測頻度であり，5.99という値は自由度2，有意水準0.05のχ2検定の棄却限界である．

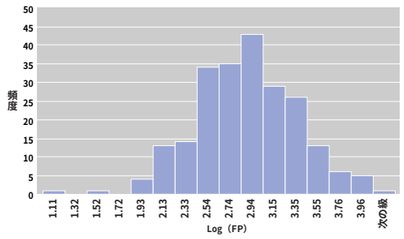


図1　Log（FP）のヒストグラム

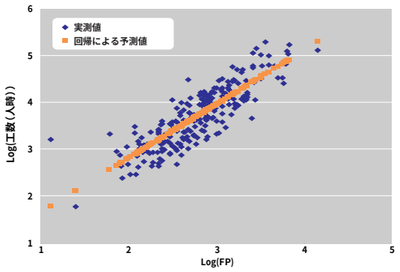


図2　規模と工数の散布図と回帰直線

ここでの「正規分布という仮説が有意水準5%で棄却されない」という意味は次のようなものである．「母集団が正規分布である集合からの標本（この場合225個のデータ）はいろいろな分布をするが，その可能性を95%まで広げると，与えられた標本の分布はその中に入っている」ということである．言い換えると，「log（FP）の母集団の分布を正規分布とみなしてよい」ということになる．

4.4　散布図の作成と回帰分析

回帰分析の結果はExcelの分析ツールを利用するだけですぐに得られる．散布図の例と回帰分析の結果を図2と表3に示す．

この結果から，

$$\text{Log（工数）}＝1.15\,\text{Log（FP）}＋0.513 \qquad (2)$$

すなわち，

$$\text{人時}＝10^{0.513}\times \text{FP}^{1.15}$$
$$＝3.26\times \text{FP}^{1.15} \qquad (3)$$

が得られる．この例では，Log（FP）とLog（工数）の相関係数は0.85であり，寄与率は0.72である．すなわち**工数の72%は規模要因で決定される**ことになる．

4.5　信頼幅の計算

規模に対する工数の回帰の95%信頼幅を計算する（詳細な計算式は標準的な統計学の教科書を参考にされたい）．ソフトウェア工学の世界では，一般に95%の信頼幅は広くなることが多い．そこで50%の信頼幅のグラフも加えて，95%の信頼幅は特異なプロジェクトの検出に，50%の信頼幅は平均的なプロジェクトであることの確認に用いるのがよいと思われる．

今回の例では，回帰係数1.147に対して95%の信頼幅は±0.094である．95%の下限値がプラスであることから危険率5%で（実際には表3のP値が示すようにもっと低い危険率で）回帰係数はプラス、すなわち工数と規模には正の相関があることがわかる．50%の信頼幅は95%の信頼

表1　規模と工数の基本統計量

	Log(FP)	Log(工数)
平均	2.79	3.71
中央値	2.79	3.69
標準偏差	0.47	0.63
標本数	225	225

幅の34.5％（約1／3）となるので，今回の例では1.147±0.032すなわち50％の上限値は1.179，下限値は1.115である．

4.6　変数逆変換と結果の表示

対数変換のままでは，報告書の読者にとって具体的なイメージがわかない．そのため，報告書には必ず対数逆変換後のグラフを示すのがよい．今回の例で得られた結果を図3に示す．

なお，95%の信頼幅は，統計ツールが出力する分散分析表の上限95％，下限95％の回帰係数と切片を使った次の式でもよい近似が得られる．

上限95％：人時＝$10^{0.777}\times FP^{1.24}$

＝$5.99\times FP^{1.24}$　　（4）

下限95％：人時＝$10^{0.248}\times FP^{1.05}$

＝$1.77\times FP^{1.05}$　　（5）

5　分析手順の例(影響要因の分析)

規模だけで工数の変動の約70％は説明できることを第4節で示した．残りの約30％はプロジェクトに依存している。規模以外で工数に影響を与える要因の分析例を以下に示す．分析例としては，「要求仕様の明確さ」を取り上げる．「要求仕様の明確さ」は次の4つの階級（レベル）に分けられている．

・aレベル：要求仕様が非常に明確
・bレベル：要求仕様がほぼ明確
・cレベル：要求仕様がややあいまい
・dレベル：要求仕様が非常にあいまい

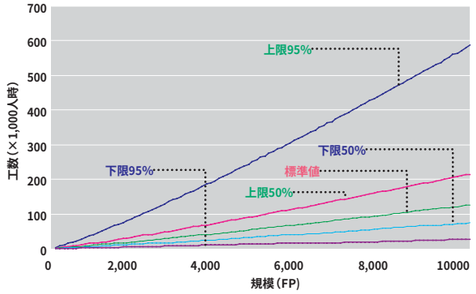


図3　規模あたりの工数の予測曲線と信頼幅

表2　Log（FP）に関する正規分布の検定結果

領域	発生頻度		
	理論値（*Ei*）	実測値（*Oi*）	誤差(*1)
(−∞, μ−1.5σ)	15.03	12	0.61
[μ−1.5σ, μ−0.5σ]	54.38	56	0.05
[μ−0.5σ, μ+ 0.5σ]	86.18	90	0.17
[μ+ 0.5σ, μ+ 1.5σ]	54.38	52	0.10
(μ+ 1.5σ, ∞)	15.03	15	0.00
合計	225	225	0.93
自由度(*2)=2, 危険率0.05のχ^2検定の棄却限界=5.99			
(*1) $(Oi-Ei)^2/Ei$, (*2) 領域数(5)−3			

表3　規模に対する工数の回帰分析結果

	係数	*P*値	下限95%	上限95%
切片	0.513	0.000177	0.248	0.777
回帰係数	1.147	4.4E-64	1.053	1.241

5.1　階級ごとの散布図の描画

「要求仕様の明確さ」のレベルごとに規模と工数の散布図を描いたものを図4に示す．規模と工数はなるべく正規分布の仮定を満たすように，いずれも対数変換した値を用いている．

5.2　階級間での規模の差の検定

最初に「要求仕様の明確さ」が規模に影響を受けていないかどうかを調べる．表4にLog（FP）の基本統計量を示す．「要求仕様の明確さ」のレベルが低下するに従ってLog（FP）の平均値及び中央値が大きくなっていることがわかる．言い換えると，規模が大きくなるに従って要求仕様の明確さが低下する傾向にあることがわかる．

このことが統計的に有意であるかどうかを，分散分析で検定する．検定すべき仮説は，「4つの階級間の平均規模はすべて等しい」ということである．4つの階級をまとめて分析した結果を表5に示す．この結果は，危険率5％で有意（P値が0.05より小さい），すなわち「4つの階級間の平均値がすべて等しいとはいえない」ということである．

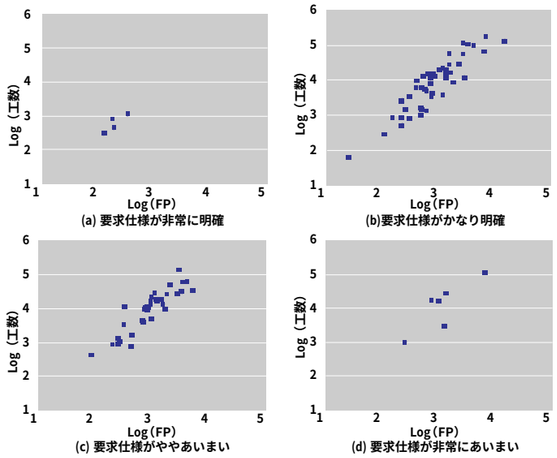


図4　「要求仕様の明確さ」に関するレベルごとの規模と工数の散布図

表4からaレベルが極端に小さいことがわかるので，次に，これを除いた3つの階級に絞って同じように分散分析する．結果を表6に示す．今度は「3つの階級間の平均値が等しいという確率は70％以上である」ということがわかる．実際，aレベルと他のレベルそれぞれとの間の平均値の差の検定（t検定）の結果では，すべて統計的に有意な差がみられ，他のレベル同士間では差がみられなかった．このことからaレベル（要求仕様が非常に明確）のプロジェクトは他のレベルより規模が小さいということが言える．

5.3　階級間で生産性の差の検定

次に，平均規模の等しい3つの階級間で平均工数に差があるかどうかを，Log（工数）を対象とする分散分析で検定する．結果は77％の確率で差がないことが示された（表7）．すなわち，b～dレベルでの平均工数の差はなく，平均規模に差がないことから生産性にも差がないと言える．

各レベルの平均生産性（工数／規模）の基本統計量を表8に示す．t検定の結果では，aレベルと他の3つのレベルに関してそれぞれ平均生産性に差があり，aレベルを除いた3つのレベル間では差がないことが明らかとなった．

5.4　分析結果の意味付け

データ分析結果の意味付けは，あくまでもソフトウェア工学的な知見に基づいて行われるべきものである．今回の例では，「要求仕様が「非常に明確」なプロジェクトは，一般に規模が小さい．そのようなプロジェクトは生産性も高いが，その理由が，要求仕様の明確さによるものか，小規模であることによるものかは，これだけでは断定できない」という結論を導き出すことができる．しかし，この結論が正しいとは限らず，分析者によっては異なる結論を導き出すかもしれない．

表4　Log（FP）の基本統計量

	aレベル	bレベル	cレベル	dレベル
平均	2.28	2.89	2.94	3.05
標準偏差	0.18	0.50	0.41	0.47
標本数	4	48	34	6

表5　Log（FP）に関する分散分析表

変動要因	変動	自由度	分散	分散比	*P*値	*F*境界値
グループ間	1.7405	3	0.580	2.76	0.047	2.71
グループ内	18.476	88	0.210	—	—	—
合計	20.217	91	—	—	—	—

表6　Log（FP）に関する分散分析表（aレベルを除く）

変動要因	変動	自由度	分散	分散比	*P*値	*F*境界値
グループ間	0.1500	2	0.075	0.35	0.708	3.10
グループ内	18.384	85	0.216	—	—	—
合計	18.534	87	—	—	—	—

表7　Log（工数）に関する分散分析表（aレベルを除く）

変動要因	変動	自由度	分散	分散比	*P*値	*F*境界値
グループ間	0.2493	2	0.125	0.26	0.774	3.10
グループ内	41.2773	85	0.486	—	—	—
合計	41.5266	87	—	—	—	—

表8　Log（工数/FP）の基本統計量

	aレベル	bレベル	cレベル	dレベル
平均	0.50	0.99	1.02	1.02
標準偏差	0.14	0.36	0.31	0.42
標本数	4	48	34	6

6　おわりに

本稿では，プロジェクトデータの特徴を意識しながら，データ分析の一般的注意事項と，統計的理論をベースにしたデータの分析事例を紹介した．統計的理論をうまく利用することによって，普遍的な結果を得ることができる．また，それに基づいてソフトウェア開発の現状に対する共通認識を得ることができ，科学的根拠に基づいた議論ができるようになる．

とはいえ，現実のデータは千変万化であり，定型的な分析手法でこと足りることはほとんどない．分析対象に合わせて統計的分析手法を臨機応変に修整し，分析の試行錯誤を繰り返した上でなければ，プロジェクトの真の姿を浮き彫りにさせることは難しい．本稿からデータの分析作業の大変さ，難しさの一端をソフトウェア開発に関係する方々にご理解して頂ければ幸いである．

参考文献

[1] Capers Jones著，鶴保征城，富野　壽監訳：ソフトウェア開発の定量化手法（第2版），共立出版，1998
[2] John McGarry他著，古山恒夫，富野　壽監訳：実践的ソフトウェア測定，共立出版，2004
[3] JIS X-0141：2004（ISO/IEC 15939:2002）ソフトウェア測定プロセス，平成16年6月20日制定
[4] Stephan H. Kan著，古山恒夫，富野　壽監訳：ソフトウェア品質工学の尺度とモデル，共立出版，2004
[5] ソフトウェア開発データ白書2005
[6] 古山恒夫，石橋雄一，岩尾俊二，花本佳一：ソフトウェアプロジェクトにおけるリスク要因の推定の一方法，情報処理学会ソフトウェア工学研究会 Vol.125-4，2000

論文 paper

品質マネジメントシステムの再構築：競争優位性の獲得に向けて

Gaining Competitive Advantage through Restructuring Quality Management System

本論文は，財団法人 日本科学技術連盟（日科技連）ソフトウェア品質管理研究会20周年記念行事「21世紀の日本のソフトウェアマネジメントと国際競争力を考える」（2005年6月17日開催）にて発表した論文に，日科技連の許可を得て，加筆修正の上で掲載したものである．

東洋大学経営学部
専任講師
野中　誠

Faculty of Business Administration, Toyo University
Makoto Nonaka

日本のソフトウェア製品の品質は，信頼性という面では世界的にも高い水準にある．しかし，近年は，その優位性も失われつつある．また，開発体制が脆弱であるばかりか，品質面での優位性を戦略的に推し進めていく姿勢に欠ける組織も多い．来るべき国際競争・協業の時代に向けて，日本のソフトウェア組織は，品質に競争優位性を見いだした戦略的な取り組みが求められる．本論文では，ソフトウェア組織が品質マネジメントシステムを通じて，どのようにして組織能力を強化し，競争優位性の獲得を目指していくべきなのかについて議論を展開する．

The quality of Japanese software products is comparatively excellent in the world market, in terms of reliability. However, this advantage is disappearing in recent years. Furthermore, there are many Japanese software organizations which have vulnerable development organizations and lack their strategies to promote their competitiveness about quality. In order to compete with global software organizations and to cooperate with them, Japanese organizations should have their strategy for gaining competitive advantage about quality. This paper discusses how Japanese organizations enforce their organizational capabilities and gain competitive advantages through restructuring their quality management systems.

Key Words & Phrases： ソフトウェア品質，品質マネジメントシステム，競争優位性
software quality, quality management system, competitve advantage

1　品質マネジメントシステム構築の課題

1.1　QMS構築の現状ー模倣によるQMS構築

多くのソフトウェア組織が，品質マネジメントシステム （QMS: Quality Management System）(1) の構築に取り組んでいる．その際に，次に挙げた国際規格や標準的なモデル （以下，標準モデルと総称） の模倣により，QMS構築を試みている場合が多い．すなわち，品質保証の国際規格であるISO 9000シリーズ，ソフトウェア組織の成熟度をプロセス領域という観点から段階的に表したCMM（Capability Maturity Model）[1]，その後継版でありプロセス領域別の成熟度表現 （連続表現） を含んだCMMI（CMM Integration）[2]，QMS要求事項を規定したISO 9001:2000等，これら標準モデルの模倣によるQMS構築である．

多くのソフトウェア組織が，これらの標準モデルに高い関心を示している．例えば，CMM/CMMIでは，国内情報サービス企業の売上高上位130社のうち，約30組織がCMMレベル2以上をすでに達成，その他の60組織が検討中である等，関心の高さが読み取れる[3]．もっとも，この傾向は国内に限った話ではなく，世界的に見ても同様の傾向が見られる．

これらの標準モデルに高い関心が寄せられる理由として，世界的なベストプラクティスを組織のQMSやソフトウェアプロセス標準に取り入れることで，ソフトウェア製品の品質向上や品質確保に結び付けたいという狙いがある．また，異なる観点として，標準モデルに対するソフトウェアプロセスの適合性を示すことが，顧客への訴求力となり得ることも理由の1つである．逆に，標準モデルへの適合性が顧客から要求される場合さえある．

QMS構築の動機は様々ではあるが，いずれにせよ，標準モデルの模倣によりQMSを構築した結果，品質確保において一定の効果を得た組織は多い．

(1) 品質マネジメントシステムとは，ISO 9000:2000では「品質に関して組織を指揮し、管理するためのマネジメントシステム」と定義している．本稿では，この定義が意味する一般的な語としてQMSを用い，ISO 9000:2000の要求事項の適合性は対象外としている．

1.2　模倣によるQMSの限界

しかしながら，標準モデルの模倣に基づいたQMS構築では，組織に競争均衡をもたらすことはあっても，本質的な競争優位性をもたらすことは期待できない．なぜなら，模倣によるQMS構築は組織間の同質化をもたらすのみであり，差別化や異質化には結び付かないからである．そればかりか，模倣が逆に作用して，問題の原因を分析して改善案を提案・実践するといった，改善活動の基本が実践できていない場合も多い．すなわち，模倣のために，品質向上のための組織的な学習の機会を逸してしまい，組織能力をかえって低下させている場合である．

また，投資対効果という観点から，模倣によるQMS構築は十分な成果があったといえるだろうか．1980年代までに品質の先進組織が築き上げた「品質中心の文化」が忘れ去られ，QMSはスタッフの仕事であって現場の仕事ではないと見なされる等，QMSが開発現場の改善意識をかえって薄めてしまっていないだろうか．さらには，QMSに関連する取り組みには戦略性があったといえるだろうか．このような疑問を考えると，多くのソフトウェア組織は，QMSの構築においてひたすら迷走を続けてきたかのようにさえ思える．

このような迷走の原因の1つには，1990年代以降の劇的な環境変化が挙げられる．オープン化，ダウンサイジング化の波に端を発し，オブジェクト指向技術，インターネット，webアプリケーションの普及，web Service等の新しいパラダイムに基づく技術の登場，組込みソフトウェアの規模及び利用分野の拡大，オープンソースソフトウェアの利用等，枚挙にいとまがない．また，顧客及びエンドユーザの品質ニーズが多様化し，「そこそこ品質（good enough quality）」のソフトウェア製品を短期間に出荷する方が，ビジネス上は成功を収めるといった事例も増えてきている．さらには，XP（eXtreme Programming）に代表されるアジャイル手法等，開発手法も多様化してきている．

このような変化が起きている中，多くの組織は，模倣によるQMS構築に邁進してきた．しかし，その結果をあらためて俯瞰してみると，紋切り型の，環境の変化に適応できていないQMSが，各組織に構築されてきたかのように思える．その先に待ちかまえているのは，QMSの形骸化であり，QMSの機能不全であろう．

1.3　QMS構築の進むべき方向

品質の向上・確保のためにQMSを構築することは，ソフトウェアビジネスを営む企業にとっては実施しなければならない当然の活動である．ただしこれは，目の前の品質問題を解決する目的といった，どちらかといえば近視眼的なアプローチである．

しかし，来るべき国際競争・協業の時代を迎えるにあたって，今後は，品質向上・確保といった当たり前の目的に加えて，組織の品質マネジメント能力を継続的に向上させること，組織に競争優位性をもたらし得る情報資源の獲得・蓄積に資することを追求していかなければならない．その手段としてQMSを活用するといった，より長期的な視点での取り組みが必要であろう．

2　日本のソフトウェア品質の競争優位性

2.1　品質水準の国際比較

QMS構築において迷走を続けてきたとはいえ，日本には，高い品質水準を達成できているソフトウェア組織が数多く存在する．図1は，Cusumanoらが実施した，ソフトウェア生産性及び品質に関する国際比較の調査結果である[4]．ソフトウェア製品の出荷後1年以内に発見された欠陥件数を，新規ソースコード1,000行あたりに換算した値 （欠

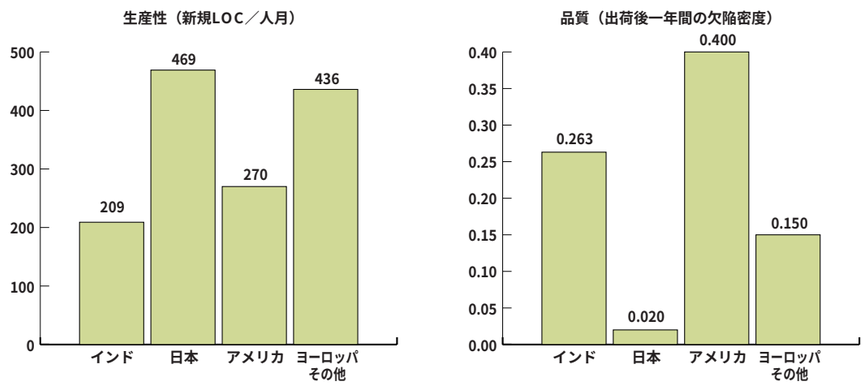


図1　生産性と品質の国際比較[4]

陥密度）で比較すると，日本の調査対象企業では中央値が0.020，米国では0.400，インドでは0.263であった[(2)]．直感的に解釈すると，5万行のソースコードが新規に開発されたソフトウェア製品の場合，日本製ではおよそ1件，米国製では20件，インド製では13件の欠陥が運用中に顕在化する，という状況である．

このような高いパフォーマンスを発揮できるソフトウェア組織が日本に存在している理由の1つとして，1970年代から1980年代にかけて見られた，品質に関する先進的な取り組み[5][6]における経験の蓄積が挙げられるだろう．これらの取り組みは，Humphreyによるソフトウェアプロセス改善モデル[7]及び後のCMM [1]にも影響を与えたと考えられている．また，日科技連ソフトウェア品質管理研究会を通じた，品質に関する先進的な取り組みを企業の垣根を越えて横断的に普及・啓蒙させる活動[8][9]があったことも，日本の組織が高いパフォーマンスを発揮できている1つの要因であろう．

さらに，定性的な議論ではあるものの，日本に独自な文化的風土も，高い品質水準の達成に貢献していると考えられる．すなわち，全員で力を合わせる，あうんの呼吸で仕事ができる，ねばり強い，職業的道徳観がある，柔軟性がある，等の要因[10]が，陰に陽に，ソフトウェア品質の向上に影響していると思われる．

2.2　揺らぐ競争優位性

しかしながら，このような品質水準の高さも，もはや圧倒的な優位性を持たなくなりつつある．表1は，TSP（Team Software Process）[11]を導入したプロジェクト20件のデータと，典型的なプロジェクト[(3)]を比較した結果である[12]．表1にある通り，TSP導入プロジェクトでは出荷後の欠陥密度は平均で0.06であり，先に示した日本の0.02に近い品質水準を達成している[(4)]．しかも，システムテストの工数比率が典型的なプロジェクトに比べて圧倒的に低いことが大きな特徴である．

表1　TSPプロジェクトと典型的プロジェクトとの比較 [12]

	TSP	典型的
システムテストでの欠陥密度	0.4 （0 - 0.9）	15
出荷後の欠陥密度	0.06 （0 - 0.2）	7.5
システムテスト工数比率 （%）	4 （2 - 7）	40
システムテスト期間比率 （%）	18 （8 - 25）	40

括弧内は範囲を表す

このように，日本の組織が現時点で有している高品質ソフトウェア開発能力は，もはや圧倒的な競争優位とはいえなくなりつつある．しかも，TSPプロジェクトにおけるテスト工数比率の少なさは，コスト予測が困難なテスト工程を管理可能な状態にするのに貢献している．その結果として，当初のプロジェクト計画からの遅延やコスト超過といったリスクを軽減するのに結びついている．

TSPの実践事例は米国でも限定された範囲であるとはいえ，米国の模倣の中から低コストを武器に競争力を増してきたインドを始めとする諸外国の存在を考えると，危機感を持って受けとめるべきデータであろう．

2.3　品質追求による競争優位性の獲得

では，日本のソフトウェア組織は，どこに活路を見いだしていけばよいのであろうか．世界第2位の巨大な情報サービス市場の中にいて，日本語の壁に守られているという理由だけでソフトウェア案件が受注できた時代は，すでに過去のものになりつつある．今後は，日本の情報サービス企業の売上高総額によって情報サービス市場を測定することが，意味をなさなくなるかもしれない．

周知の通り，国内のソフトウェア組織の人件費は，周辺アジア地域等に比べて圧倒的に高く，その状況が逆転することはあり得ない．ソフトウェア調達のグローバル化時代を迎えつつある今日，国内ソフトウェア組織の競争力の源泉となるのは，やはり，広義の「品質」の追求をおいて他にない．

ソフトウェア品質という概念定義には諸説あるが，一般には，Crosbyによる「要求に対する充足度」と，Juran & Grynaによる「利用に対する適合度」という2つのレベルで捉えられる[13]．ソフトウェアに関していえば，前者は，ソフトウェア製品が要求仕様を満たしている度合いを表している．一方，後者は，ソフトウェア製品が利用される場面において，ソフトウェア製品が利用者のニーズに合致している度合いを表している．その他，Weinbergは「誰かにとっての価値」という定義を与えている．

これら2つの品質概念の関係を，ソフトウェアのライフサイクルと関連づけて示したのが図2である．ここでは，品質に関する「利用者のニーズ」をソフトウェア要求仕様として仕様化したものを「要求品質」，さらに開発の観点から仕様化して設計仕様に落とし込んだものを「設計品質」とよんでいる[(5)]．これに対して，出来上がったソフトウェアが設計仕様通りに作成されている度合いを「適合品質」，要求品質に照らし合わせて評価される特性を「製品の品質」，ソフトウェア製品が利用されたときに利用者のニーズと照らし合わせて評価される特性を「利用時の品質」とよんでいる．ここで，利用者のニーズと利用時の品質との間での評価が，Juran & Grynaの「利用に対する適合度」に対応する．また，要求品質と製品の品質との間での評価が，Crosbyの「要求に対する充足度」に対応する．

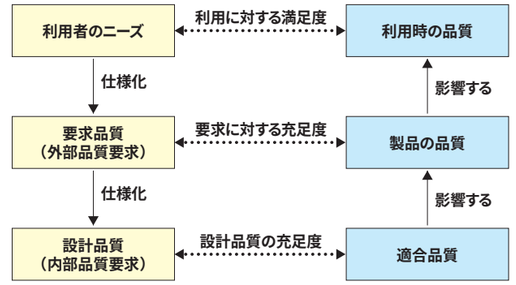


図2　ソフトウェア品質の概念とライフサイクルとの関係
（JIS X 0129-1:2003を元に，著者が修正した上で掲載）

表2　ISO/IEC 9126-1のソフトウェア品質特性

品質特性	品質副特性
機能性	合目的性，正確性，相互運用性，セキュリティ
信頼性	成熟性，障害許容性，回復性
使用性	理解性，習得性，運用性，魅力性
効率性	時間効率性，資源効率性
保守性	解析性，変更性，安定性，試験性
移植性	環境適応性，設置性，共存性，置換性

その他,各品質特性に「標準適合性」という副特性が含まれる

ここでの広義の「品質」とは，「利用に対する適合度」または「誰かにとっての価値」の意味で捉えている．すなわち，ソフトウェア製品が利用される場面で，ソフトウェア製品が利用者のニーズに合致していて，価値を提供している度合いの意味である．

したがって，欠陥密度が低いことや故障の少ないことといった信頼性の意味だけで品質を捉えることは，極めて狭い概念範囲ということになる．また，ISO/IEC 9126-1: 2001 （JIS X 0129-1:2003） では，ソフトウェア製品の品質特性として，機能性，信頼性，使用性，効率性，保守性，移植性といった６つの品質特性と，その下位概念である品質副特性を定義している（表２）．このように，ソフトウェア品質を構成している特性は多様であるということを認識しておかなければならない．

その上で，日本のソフトウェア組織は，どのレベルの「品質」に対して競争優位性を求めるべきであろうか．広義と狭義の意味があるが，目指すべきは「すべて」の意味での品質に対して競争優位性を求めるべきであろう．すなわち，要求される信頼性レベルを確実に達成し，要求仕様または計画したソフトウェア要求を確実に充足した製品を開発し，かつ，顧客及びエンドユーザのニーズを満たせるソフトウェア品質を創り出せる能力が求められる．

「品質」追求を目指したときに，QMSの役割が，信頼性という狭い意味での品質確保を確実にするためのシステムだけでは，あまりに力不足である．QMSの存在が，ソフトウェア組織を「品質」追求に向けてドライブし，高い「品質」を創出できる組織能力の獲得・強化に貢献し，組織に競争優位性をもたらすものであるべきであろう．模倣によるQMS構築の中からは，組織に競争優位性をもたらすことは期待できない．

3　競争戦略論の視座から

では，どのような点に着目して，競争力の強化に貢献し得るQMS構築に結び付けていけばよいのであろうか．明確な正解や道筋はないものの，ここでは，競争戦略論の視座からこの問題について議論する．

(2) ただし，同質の企業及びプロジェクトを比較していない，ユーザ数によって報告される欠陥数が異なる，回答企業のデータの精度に問題がある，サンプルサイズが小さい等，必ずしも妥当性の高い調査とは言えない．
(3) ここでの「典型的なプロジェクト」とは，Standish Groupによる "CHAOS'94:Charting the Seas of Information Technology ”(1994）で報告されたデータを用いている．
(4) 日本のデータとTSPプロジェクトのデータでは，プロジェクトの性質（例えば予算規模や開発期間の制約等）が同一とはいえないため，比較の際には注意が必要である．
(5) ISO/IEC 9126-1:2001 (JIS X 0129-1:2003) では，それぞれ「外部品質要求」「内部品質要求」とよんでいる．

3.1　ソフトウェアビジネスにおけるリスク

一般に，競争優位性を獲得するためには，戦略に裏付けられた差別化が求められる．しかし，企業が適合性の高い事業戦略を立案し，これを確実に遂行することは決して容易ではない．三品の調査では，類似の経営環境を共有している企業が逐次適応にベストを尽くすとき，その結果は横並びになることが多いことを示している[14]．三品はこれを「戦略の機能不全」とよんでいる．

では，ソフトウェアビジネスではどうであろうか．図3は，国内の大手独立系システムインテグレータの連結での売上高営業利益率を，過去3年の範囲で調べた結果である．3年という極めて短い範囲のデータではあるが，図に示されているように，ほぼ同質の経営環境を有しており，同質のソフトウェア開発能力を持っている企業であっても，売上高影響利益率のグラフには大きな差が生じている．すなわち，決して横並びという状態とはいえない．

しかし残念ながら，この差は，ソフトウェア企業の事業戦略の差異によって生じているものではない．各社の公表資料によると，2002年度に利益率の最も高かったD社では「専門知識の共有化，プロジェクト監査の徹底，生産性の向上」が奏功したと述べている．一方，売上高営業利益率がこの中ではもっとも低いE社では，利益率低下の最大の原因は赤字プロジェクトの影響であった．

冒頭に述べた「模倣によるQMS構築」等のように，ソフトウェア企業は同質間での競争に向かいつつある．しかし，プロジェクトマネジメントも含めた，広い意味でのソフトウェア開発能力の差異によって，その業績が異なってくる．ソフトウェアビジネスにおいて競争優位を獲得するためには，戦略の適合性もさることながら，ソフトウェアエンジニアリングのより一層の強化，組織能力の強化が求められる．

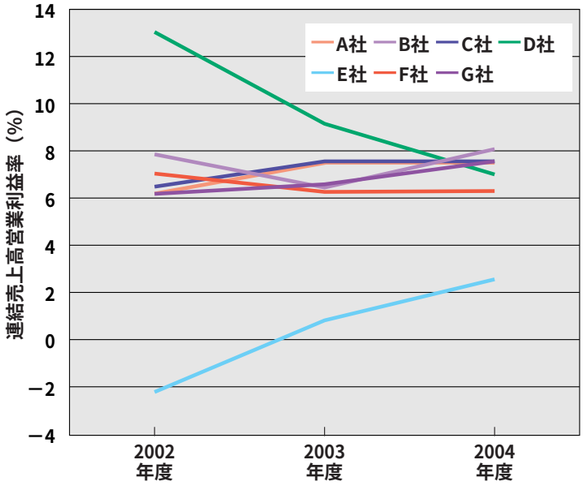


図3　大手システムインテグレータの営業利益率推移

V	R	I	O	競争優位の意味合い	経済的なパフォーマンス
No	—	—	No	競争劣位	標準を下回る
Yes	No	—	↓	競争均衡	標準
Yes	Yes	No	↓	一時的競争優位	標準を上回る
Yes	Yes	Yes	Yes	持続的競争優位	標準を上回る

図4　経営資源の特性と競争優位性の対応[19]

3.2　受託型中心の日本のソフトウェア産業

日本のソフトウェア企業は，パッケージ販売よりも受注ソフトウェア開発や保守運用サービス等を機軸とする企業が多数を占めている．一般論では，受注型ビジネスよりも，より利益率の高いパッケージ販売型ビジネスへの移行を目指すべきとなるのであろう[15]．しかしながら，かつてBrooksが指摘したように，パッケージ製品に耐えうるレベルにまでプログラムを洗練するには，通常のプログラム開発にかかるコストよりも多くのコストをかけなければならない[16]．

また，新しいニーズを喚起させるパッケージ製品を企画し，流通経路に乗せて広く販売し，そのビジネスを成功させることは決して容易ではない．たとえ一時的に成功したとしても，そのビジネスを長期にわたって継続させることも容易ではない．これは，日本の情報サービス産業には大小合わせて7,000を超える企業がありながら，10年スパンで眺めても大ヒットしたパッケージ製品が少ないことからも容易に推察できる．

製品開発や販売を機軸とするソフトウェアビジネスでは，販売不振になった場合のリスクが高いため，実際には，製品とサービスをバランスよく配置したビジネス展開が求められる[17]．さらには，組込みソフトウェア開発のように，そもそもパッケージ開発には適合性の低いソフトウェアビジネスも存在する(6)．

3.3　資源ベース観によるアプローチ

このような状況における競争戦略を考えた場合，Porterに代表されるポジショニング・アプローチによる競争戦略論[18]は，戦略を考察する上での適合性が低いと考えられる．むしろ，経営資源に競争優位性を求めるといった，資源ベース観に基づく競争戦略論[19]の方が，日本のソフトウェアビジネスには適合性が高いと思われる．

資源ベース観に基づく競争戦略論では「ある企業が他社よりも業績を上げているのは，他社よりも継続的に優れた経営資源や能力を持っているため」という考え方に基づいた理論である．一般に，コア・コンピタンスと呼ばれる経営資源に，競争優位の源泉を求めるという考え方である．

Barney[19]は，組織内部にある経営資源が持続的な競争優位を生み出すための必要条件として，次の4つの特性を示している(7)．それぞれの頭文字をとって，VRIOと呼ばれる．

・経済的価値があること（Valuable）
・稀少性があること（Rarity）
・完全には模倣できないこと（Imitability）
・組織的に活用されていること（Organization）

これらの条件すべてを満たした経営資源を有していれば，組織には持続的競争優位がもたらされる可能性が高い．しかし，そうでない経営資源しか有していない場合は，競争劣位，競争均衡，または一時的競争優位にしか到達ができない可能性が高い．このような考え方が，Barneyの資源ベース観に基づく競争戦略論である．VRIOと競争優位性との関係を図4に示す．

4　競争優位性をもたらすQMS構築に向けて

4.1　品質に関する情報資源の強化

BarneyのVRIOフレームワークに基づいて競争優位性の獲得を目指す場合に，ソフトウェア組織が実施すべきことは何か．まずは，自らの組織において，持続的競争優位をもたらすような，ソフトウェア品質に関連する情報資源が何であるのかを認識することが求められる．

そのような情報資源が何も存在しない場合は，その芽となる情報資源を発見し，ノウハウを積極的に蓄積していく仕組みを整えることが求められる．また，模倣の困難性を考えた場合，情報資源の蓄積過程が不明確であるほど模倣が困難になる．これは，一般には情報の経路依存性と呼ばれる特性であり，これに合致する情報資源を強化していくとよい．

4.2　VRIOを満たす情報資源の活用事例

ここでは，VRIOを満たした品質に関連する情報資源の活用事例を2件紹介する．

1件目の事例として，ある特定の組込み製品分野のソフトウェア開発において，長年にわたって蓄積されてきたテスト項目を体系的に整理し活用しているソフトウェア組織がある．この事例では，テスト項目に経済的価値があるのは当然の事として，テスト項目自体が特定の組込み製品分野に限られたノウハウの塊であるために，稀少性が高い．これを組織内で共有化し，テスト工程で効率的に実施できる仕組みや体制を整えれば，同じ製品分野における信頼性について競争優位性を発揮することが期待できる．この事例におけるテスト項目は，VRIOフレームワークの観点からも整合性のある，競争優位をもたらす情報資源であるといえよう．

別の事例として，株式会社CSKでは，IFPUG（International Function Point Users Group）法を独自に改良し，機能規模の定量化を積極的に推進している[20]．規模データを収集したプロジェクトはすでに1,000件に上っており，新規開発・一括請負・Web系開発等プロジェクトのタイプによってデータを分類し，それぞれの分類毎に回帰分析を行って見積りモデルを得ている．見積りモデルは，自社データの蓄積がない場合に他社データを流用して構築するよりも，自社データで構築した方が，やはり精度が高いことが研究により示されている[21]．すなわち，独自の工夫を通じて得られた見積りモデルという情報資源は，情報粘着性が高く，他の組織が活用するのは困難な情報資源であるといえよう．これも，やはりVRIOを満たす情報資源を獲得した事例といえる．

(6) ただし，携帯電話のブラウザソフト等のように，組込み製品プラットフォーム上で共通利用できる部分の多いパッケージ的な要素を持った組込みソフトウェアビジネスも登場しつつある．
(7) ただし，これらの要素は事後的に観測されたものであり，必要条件ではあっても十分条件であるとはいえない．

4.3 VRIOを満たさない情報資源の扱い

一方で，ソフトウェア開発にUML（Unified Modeling Language）を用いるといった取り組みは，うまく活用していれば経済的価値を生む有益な活動である．しかし，資源ベース観に基づく競争戦略論の観点から見ると，もはや稀少性はなく，模倣も容易な活動である．こうした取り組みは組織を競争均衡に引き上げるのには貢献するが，決して競争優位に結びつくものではない．

もちろん，競争均衡のための活動に何ら意味がないということではない．むしろ，ソフトウェア組織を競争均衡に引き上げるための基盤となる活動は確実に実施していかなければならない．例えば，ソフトウェアエンジニアリングの組織的な実践等は，ソフトウェアビジネスを営む上で当然のものとして実施すべき活動である．こうした当たり前の活動を確実に実施した上で，さらに広義の「品質」に関して競争優位に結びつく情報資源を探り，その組織的活用を目指していくべきである．

こうして獲得・蓄積された情報資源の組織的活用を促進するために，レビュープロセスやテストプロセスを改善することが求められる．改善後のプロセスが組織レベルで確実に実践されることをサポートするための仕組みとして，QMSが機能すべきであろう．

5 品質コストに基づくプロセス改善の評価

5.1 品質コスト評価の概要

こうしてプロセスが改善されると，一般には，チェックリストが肥大化し，レビュープロセスが重たくなりがちである．これは，プロセスを改善したばかりの時点では仕方のないことが，いずれは，設計手法の改善やツールによる自動化等，有効性及び効率性を高めていくことが求められる．このようなプロセス改善の効率性評価の道具として，品質コストの考え方が利用できる．

品質コストを適切に評価するためには，QMSの要素として，活動基準原価計算（ABC：Activity-Based Costing）及び活動基準管理（ABM：Activity-Based Management）に関する取り組みが求められる．すなわち，ソフトウェア開発において，品質コストを把握するためのプロセスメトリクスを収集し，ソフトウェア開発及び改善活動に関する品質コストを測定することが求められる．こうした取り組みを実施する際に，ソフトウェア技術者がPSP（Personal Software Process）[22]のコンセプトや手法を習得済みであると測定が実施しやすい．

品質コストは，一般にPAF法（prevention, appraisal, failure）に基づいて分類される[23]．すなわち，品質問題を防ぐための予防コスト，不良品を出荷しないための評価コスト，欠陥品の手直しや欠陥品の出荷による問題解決のために支出される失敗コストがある．失敗コストはさらに，内部失敗コストと外部失敗コストに分類される．これらの合計が総品質コストとなる．以下に，それぞれの概要と，ソフトウェアに関連した活動の例を示す[24]．

(1) 内部失敗コスト：出荷前に検出された品質問題に関するコスト．出荷前の欠陥管理，手戻り，再レビュー，再テスト等．
(2) 外部失敗コスト：出荷後に検出された品質問題に関するコスト．テクニカルサポート，苦情調査，欠陥公表，補修アップグレード，欠陥修正等．
(3) 評価コスト：製品状態の評価，品質水準の保証に関するコスト．テスト，ソフトウェア品質保証，インスペクション，レビュー，品質監査，出荷判定等．
(4) 予防コスト：品質基準の定義，プロジェクトへの介入に関するコスト．受け入れテスト時の出荷基準定義，関連する品質標準の定義，訓練，プロセス改善，メトリクス収集，分析等．

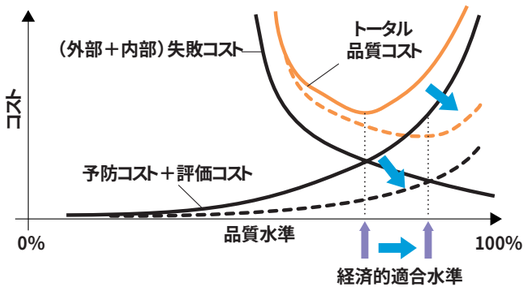


図5 ソフトウェア品質コストの経済モデル

5.2 品質コストの経済性

予防コストと評価コストは，一般に，失敗コストとトレードオフの関係にある．この関係を概念的に示したのが図5である．図5において，一般に，ソフトウェア製品の要求品質水準を高めると，失敗コストは低下するが，その品質確保に必要な予防コストと評価コストは高くなる．逆に，ソフトウェア製品の要求品質水準を下げると，予防コスト及び評価コストは減少するものの，失敗コストが上昇してしまうことになる．これらのコストを合成したコスト曲線を描いた場合，総品質コストを最小化する経済的適合水準が存在すると考えられる[23]．

しかし，経済的適合水準は，顧客の品質要求水準よりも低い場合が多いようである．そのため，ソフトウェア組織は，失敗コストを抑制するために，予防コストと評価コストに多くの工数を費やさざるを得なくなる．これは，品質コスト評価という観点からすると，最適な意思決定であるとはいえない．

ここで，プロセス改善の質の問題が問われてくる．もしも，評価コストに関連したソフトウェア活動の有効性及び効率性を高めることができれば，より高い品質水準達成のための活動を低コストで実施できるようになる．このとき，予防コストと評価コストの合成関数は，図5における実線から破線へと推移する．このような推移が実現できると，経済的適合水準を高められると同時に，総品質コストを低下させることができる．

模倣によるQMS構築では，標準モデルに示されたプロセス領域を1つずつ塗りつぶしていくのが精一杯で，品質コスト評価の観点から見てプロセス改善にムダがあるのかどうかが見えなくなってしまう．組織能力の強化に向かった活動が，効率よく実施されていることを把握するには，品質コスト評価が役立つであろう．

6 おわりに

本稿では，組織に競争優位性をもたらすためのQMS構築について議論を展開した．ソフトウェアエンジニアリングでは，一般に，人に起因する生産性や品質の低下を防ぐために，体系的で，規律化された，共通のアプローチの適用を目指す．しかし，競争優位性を獲得するためには，共通化の中から差別化・異質化を図れる情報資源の獲得・強化を目指すべきであろう．

また，産業界における情報共有を促進させることも必要である．共有される情報は，競争均衡のために実施した事例だけでなく，持続的競争優位性を獲得できた事例の共有が行えるのが望ましい．真に持続的競争優位性をもたらす情報資源であれば，それは模倣が困難であるため，提供者の利益も保護されることであろう．

本稿で述べた内容は，実績のある確立された方法論ではなく，ソフトウェアエンジニアリング及び競争戦略論に関する研究結果を元に論理的な推測に基づいて議論したものである．ここで述べた内容の幾つかは，今後，事例研究や検証を通じて，確固たる裏付けを取るべきものもある．しかしながら，本稿を通じて，模倣によるQMSを一歩高い視点から俯瞰し，組織の向かうべき方向性等について，議論の契機や考えるヒントを少しでも読み取っていただければ，著者としては望外の喜びである．

謝辞 本論文の掲載をご快諾いただいた日科技連 中島宣彦氏をはじめとする日科技連ソフトウェア品質管理研究会の関係各位に深く感謝する．

参考文献

[1] Paulk, M.C., et. al.: Capability Maturity Model for Software, Version 1.1, CMU/SEI-93-TR-24, 1993
[2] Chrissis, M. B., et. al.: CMMI: Guidelines for Process Integration and Product Improvement, Addison-Wesley, 2003
[3] コンピュートピア2004年9月号，コンピュータ・エージ社，2004
[4] Cusumano, M., MacCormack, A., Kemerer, C.F. and Crandall, B.: Software Development Worldwide: The State of the Practice, IEEE Software, vol. 20, no. 6, pp. 28-34, 2003
[5] 全社SWQC活動調整委員会編: ソフトウェアの総合的品質管理NECのSWQC活動，日科技連出版社，1990
[6] 富士通通信ソフトウェア開発部編: 富士通における「あゆみ」活動―高品質ソフトウェア開発への挑戦，日科技連出版社，1992
[7] Humphrey, W.S.: Managing the Software Process, Addison-Wesley, 1989（藤野喜一監訳: ソフトウェアプロセス成熟度の改善，日科技連，1991）
[8] 菅野文友・吉澤正監修: 21世紀へのソフトウェア品質保証技術，日科技連，1994
[9] 菅野文友監修: ソフトウェア品質管理事例集，日科技連，1990
[10] 平林良人: ISO 9001 有効活用のためのビジネス改善ツール （解説部），日本規格協会，2005
[11] Humphrey, W.S.: Introduction to the Team Software Process, Addison-Wesley, 1999
[12] Davis, N. and Mullaney, J.: TSP in Practice: A Summary of Recent Results, CMU/SEI-2003-TR- 14, 2003
[13] Kan, S. H.: Metrics and Models in Software Quality Engineering, 2nd ed., Pearson Education, 2003（古山恒夫・富野壽監訳: ソフトウェア品質工学の尺度とモデル，共立出版，2004）
[14] 三品和広：戦略不全の論理―慢性的な低収益の病からどう抜け出すか，東洋経済新報社，2004
[15] ソフトウェア産業研究会:ソフトウェアビジネスの競争力, 中央経済社, 2005
[16] Brooks, Jr., F.B.: The Mythical Mon-Month: essays on software engineering, anniversary edition, Addison-Wesley, 1995（滝沢徹他訳：人月の神話，アジソンウェスレイ，1996）
[17] Cusumano, M.A.: The Business of Software, Free Press, 2004（サイコム・インターナショナル訳：ソフトウェア企業の競争戦略，ダイヤモンド社，2004）
[18] Porter, M.E.: Competitive Advantage, The Free Press, 1985（土岐坤ほか訳，競争優位の戦略，ダイヤモンド社，1985）
[19] Barney, J.B.: Gaining and Sustaining Competitive Advantage, Prentice-Hall, 2nd Ed., 2001（岡田正大訳: 企業戦略論，ダイヤモンド社，2003）
[20] 本間周二: CSKに見るFP法の実践 ―プロジェクト定量化で品質と見積もり精度向上，日経ITプロフェッショナル2004年7月号，2004
[21] Mendes, E. and Kitchenham, B.: Further Comparison of Cross-company and Within-company Effort Estimation Models for Web Applications, Proc. 10th International Software Metrics Symposium, pp.348-357, 2004
[22] Humphrey, W.S.: A Discipline for Software Engineering, Addison-Wesley, 1995（松本正雄監訳: パーソナルソフトウェアプロセス技法，共立出版，1999）
[23] 伊藤嘉博: 環境を重視する品質コストマネジメント，中央経済社，2001
[24] Krasner, H.: Using the Cost of Quality Approach for Software, CrossTalk, Nov., pp.6-11, 1998

論文 paper

ものづくり戦略論とアーキテクチャ
—ソフトウェア・アーキテクチャの測定と分析—

Architecture based strategy and its metrics method

東京大学
ものづくり経営研究センター　COE特任助手
立本博文

Manufacturing Management Research Center,University of Tokyo.
Specially Appointed Research Assistant
Hirofumi Tasumoto

一般に日本のメーカの技術力は，世界平均に比べ高いと言われているが，収益力が高いとはいえない．このギャップを埋め，技術力と収益力をつなげるものが，ものづくり戦略論である．経営学の一分野であるものづくり戦略論では「アーキテクチャ」がキーコンセプトとなる．しかし，工学的な文脈での「アーキテクチャ」とは，多少重点を置くポイントが異なる．

例えば，ソフトウェア・エンジニアリング的な意味でのアーキテクチャというと，MPUのインストラクションセットのことであったり，大規模システムでの３階層クライアント／サーバであったりする．複雑な問題に対して，それを制御するための工学的な解がアーキテクチャである．

ものづくり戦略論的な意味でのアーキテクチャは「どのアーキテクチャを選択するかは，ビジネスの意志決定の問題であり，競争上もっとも有利になるように選択するもの」である．アーキテクチャの問題を，企業の競争上の意志決定の問題と捉える訳である．

ソフトウェアの世界では，近年アーキテクチャが重視されているが，現実の企業活動では，上記のように２つの視点でアーキテクチャを捉えることが重要である．本稿では，後者のものづくり戦略論的な「製品アーキテクチャ」を取り上げ，WebサーバApacheを例にソフトウェアアーキテクチャの測定と分析を行う．

Product Architecture(PA) is selected by a firm according to its business strategy like IBM's computer business in 1950-1980's. PA has been widely recognized as a key factor of strategy in management science area.

In this paper, I measured PA using the method of social network analysis(SNA) and evaluated its validity. In addition, applying its method to the software product(Apache HTTP Server), the dynamic shift of PA is observed in time series analysis of the architectural index.

Key Words & Phrases：アーキテクチャ測定，アーキテクチャシフト，機能構造行列，ネットワーク分析，応用 アーキテクチャ指数
metrics of product architecture, architectural shift, function-structure, matrix, application of network analysis, architectural index

1　なぜアーキテクチャなのか？

ものづくり戦略論的な「アーキテクチャ」では，広義にはビジネス・アーキテクチャを考え，狭義には製品アーキテクチャを考える．ビジネス・アーキテクチャとは，製品アーキテクチャを考えた上で，ビジネスの諸要素の結び付け方を考える事である．製品アーキテクチャを考える際に，競争上意味のある軸として，2つの軸が考えられる．1つが「インテグラル／モジュラー」の軸であり，もう1つが「オープン／クローズ」の軸である（図1）．この2軸により，今置かれている製品のポジションを考え，次にどのように動けば（または動かなければ）よいかを考えるのが，アーキテクチャ戦略である[1], [2].

例えば，コンピュータ関連製品では，産業自体が成立

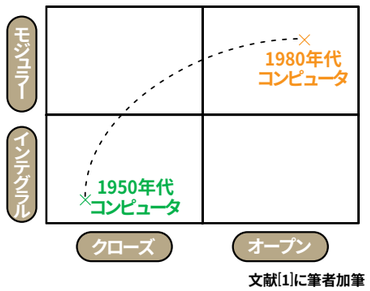


図１ 製品アーキテクチャの分類

した1950年代には，クローズ・インテグラルのポジションであったが，その後50年で急速にオープン・モジュラーに変わっていった．このような製品アーキテクチャの変化を所与であるとみれば，企業が適応しなくてはいけない環境要因であるが，実際には，企業が戦略的に決定していくことが多い．先のコンピュータの例では，IBM社のとったモジュール化戦略，オープン・アーキテクチャ戦略の影響が大きい[3]．IBM社では，汎用機ビジネスで，1950年からモジュール化戦略をとり，莫大な利益を上げた．その後1981年から参入したPC事業ではオープン・アーキテクチャ戦略をとり，当初は大きな成功を収めた．アーキテクチャは，戦略的に選択するものである．

2　アーキテクチャの測定方法

2.1　測定の問題

コンセプトとして，アーキテクチャが重要であることは誰もが認めるところであるが，実証研究の立場から言えば測定できる物となっていなくてはいけない．

しかしながら，製品アーキテクチャの測定は非常に難しい．1つには，「アーキテクチャ」という言葉が持つ曖昧性が挙げられる．「アーキテクチャ」とは，基本的な設計思想であり，大きなコンセプトを表す．よって，定量的な分析をしようとする際には，「アーキテクチャ」という概念を測定できるレベルまで具体的にし，操作化できるようにしなくてはならない．

そこで考えられるのが，システムの機能と構造の対応関係をアーキテクチャとして測定するというアイデアである[4]．機能と構造を各々リストアップし，機能要素と構造要素の対応関係図を書いてみる（図3）．要素間の連結度合いを見ることによって，アーキテクチャを分析することができる．

A＝「機能×構造」行列
${}^tA$＝「構造×機能」行列（${}^tA$はAの配置行列）

$$A \times {}^tA = \begin{pmatrix} 1&1&1&1 \\ 1&1&0&0 \\ 1&0&1&1 \end{pmatrix} \times \begin{pmatrix} 1&1&1 \\ 1&1&0 \\ 1&0&1 \\ 1&0&1 \end{pmatrix} = \begin{pmatrix} 4&2&3 \\ 2&2&1 \\ 3&1&3 \end{pmatrix} = B$$

B＝「構造×構造」行列
行列Bの各セルは、紐帯数となる

図2　「機能×構造」行列と「構造×構造」行列の関係

機能×構造の関係図では，機能と構造の連結線が一対一に対応している製品設計は，モジュラー的である．逆に一対多に対応している場合は，インテグラル的である．

本稿では，「オープン／クローズ」の軸は置いておき，「インテグラル／モジュラー」の軸を考える．

2.2　DSMと構造行列

「機能×構造」行列をアーキテクチャ測定の基本形と考えたときに，そこから2つの派生的な測定行列を考えることができる．1つは，「機能×機能」行列（DSM：Design Structure Matrix）である．これは，「機能×構造」行列に対して，その転置行列を掛けることで作ることができる．行列要素には，ある機能と別の機能の関係の数が入る（図2）．

DSMの文脈では，機能は設計パラメタとよばれ，設計パラメタ間の依存関係が主な測定目標となる[3]．「機能×機能」行列が測定対象になる一方で，「構造×構造」行列を測定目標にするアプローチも当然考えられる．「構造×構造」行列を構造行列とよぶ(1)．

これら3つの行列は，結局は問題解決が諸要素間でど

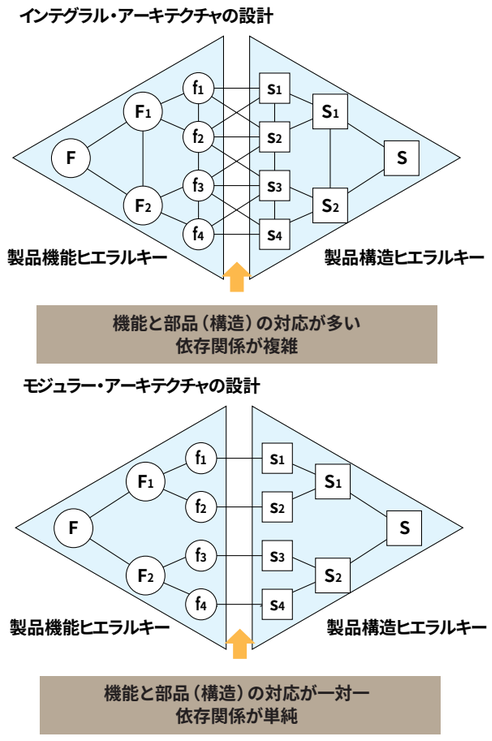


図3　インテグラル・アーキテクチャとモジュラー・アーキテクチャ[11]

(1) 一般的なコンセンサスがあるわけではないが，本研究では以後「構造×構造」行列を構造行列とよぶことにする．

のように依存しているかを表している．本論文では，構造行列を測定の目標としている．

2.3 インテグラル／モジュラーアーキテクチャ

インテグラル／モジュラーの例を行列表現をしてみよう（図4）．モジュラーアーキテクチャの場合，言い換えれば，機能と構造が一対一関係にある場合，「機能×構造」行列は，対角成分のみ≠0，その他の成分=0となる．インテグラルアーキテクチャの場合は，対角成分とその他の成分≠0となる．DSM，構造行列においても同様のことがいえる．モジュラー化とは，DSM，構造行列が対角化されることと等しい（図5）．

2.4 行列の縮約化

構造行列を測定の目標とすると，分類基準が問題となる．例えば自動車は，分類基準によっては数百点の部品でできているとも，また数万点の部品からできているともいえる．分類基準の問題は，測定にはつきものであり，避けて通ることはできない．

アーキテクチャの測定という面からは，細かい分割には意味がない．アーキテクチャは基本的な設計思想であるのだから，小さな変化を見るのではなく，大きい意味での設計構造を捉えるのである．そのため，ある程度の大きさの行列にオリジナルの構造行列を捉え直す必要がある．この問題を解決するのが行列の縮約化である．

行列の縮約化[2]には大きく2つの方法がある．1つは，演繹的な方法で，「既に要素間にある程度の関係が想定される場合，それを元に要素をまとめる」という方法である．例えば，「自動車の部品は数万点あるが，意味のあるようなまとまりは数百点レベル」であることが経験的にわかっている．そういう場合であれば，数百点レベルの構造行列にするべきである．

もう1つは，そのような経験的な基準が未だ不明確である場合に，データとある一定のアルゴリズムから事後的に縮約を行う方法がある．縮約化のアルゴリズムには，要素の関係性の類似を計るのにユークリッド距離を用いる「STRUCTUREアルゴリズム」[7]やピアソンの積率相関係数を用いる「CONCORアルゴリズム」[8]等がある．行列の縮約化とは，簡単に言えば，要素間の関係性をなるべく維持するように，大雑把に行列を捉え直し，サイズの小さい行列に直すことである．今回の測定にはCONCORアルゴリズムを用いた．

図6では，7×7行列を3×3行列に縮約した例を示している．縮約化して作成された行列のことをイメージ行列とよぶ．

【モジュラー型】

	1	2	3
1	1	0	0
2	0	1	0
3	0	0	1

■要素1は、要素1にしか依存していない（自ブロック内部で完結している）要素2、要素3についても同様

【インテグラル型】

	1	2	3
1	1	1	1
2	0	1	1
3	0	0	1

■要素1は、要素1とともに、要素2要素3にも依存している。

図4　イメージ行列の見方

	製品アーキテクチャ	
	モジュラー	インテグラル
機能と構造の対応関係	一対一	一対多
機能×構造行列	対角化されている	非対角成分が≠0である

図5　製品アーキテクチャと「インテグラル/モジュラー」の整理

図6　行列の縮約過程

(2) 縮約化の手法自体は，社会ネットワーク分析では非常にポピュラーな手法である（[5]，[6]）．

3 ソフトウェア製品へのアーキテクチャ測定の適用

今回のアーキテクチャ測定の基本的な戦略は，構造行列を測定の目標とし，CONCORを用いた縮約化を行うこととした．これをソフトウェア製品に適用するためには，どうすればよいのだろうか．

ソフトウェア製品の設計図であるソースコードには，いくつもの機能を実現するために大量の関数が定義されている．「構造×構造」行列を測定するために，ソフトウェアの部品である関数同士がどのような構造で定義されているかを分析すればよい．関数同士がどのような呼ばれ方をしているのかを，コール構造と呼び，グラフで示した物がコールグラフである．

コールグラフから，構造行列を作成し，分析対象とする（図7）．

3.1 測定手順

測定手順をまとめると，次のようになる．

①ソースコードより，関数の呼び合い構造（コール構造）を抽出する
②関数の呼び合い構造から構造行列を作成する
③構造行列にCONCORアルゴリズムを用いて縮約化しイメージ行列を作成する
④作成したイメージ行列の製品アーキテクチャを分析する

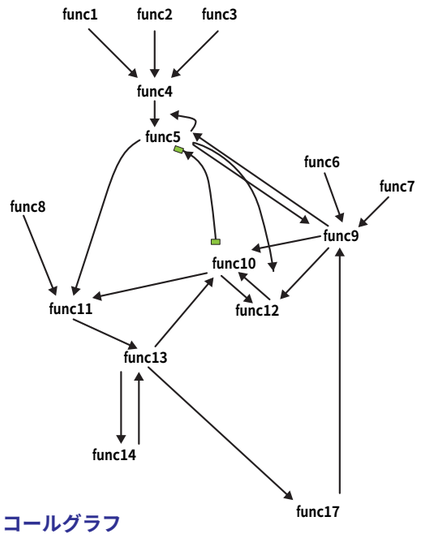


図7　コールグラフ

3.2 測定対象プロジェクト

測定対象となる関数のコール関係を反映した構造行列を作成するためには，ソースコードにアクセスする必要がある．このため，オープンソースプロジェクトApacheを今回の測定の対象とした．オープンソース製品であれば，ソースコードの解析を行うことができるからである．Apacheは，ホームページを公開するためのWebサーバ・ソフトウェアである．

Apacheは，1997年以降Webサーバのトップシェアを保っている[3]．Apacheは，いくつかの系列に分かれて開発されており，今回の分析対象は1.3系列とした．

4 適用結果・考察

(1) 縮約化

Apache 1.3系列の中でも，最も初期のリリースであるリリース1.3.0を用いて縮約化を行った．関数同士の構造行列の大きさは984×984行列であり，これを4×4行列に縮約[4]した．その結果が図8である．

きちんと縮約できていれば，各行要素間の関係が保存される．確認の意味からもこの4つの行要素の関係を見てみよう．なお，確認の際に，各行要素に含まれる関数名とその関数が属するファイル名およびソースコード内のコメントを参考に，各行に名称付けを行った（図9）．

イメージ行列を確認すると，4つの特徴が見える．①(3,1）の値が0.007と高い．②（2,2）の値が0.008と高い．③（3,2）の値が0.007と高い．④（4,4）＝0.001の行列要素の値が小さい．この4つの特徴が，Apacheのアーキテクチャの従来の説明と整合性があるかあるかどうかを考察していく．

(2) 従来のApacheの構造図との比較

従来Apacheのアーキテクチャの説明に使われている構成図との比較をした．名前を付けた要素との対応を図10に示す．

1点目に，①部分のプロセス制御エンジン内の関係性が強い．プロセス制御エンジンは，httpリクエストの処理サイクルを制御するもので，最も処理効率に関係する部分である．効率を高めているという意味で，この部分が内部依存度を高めていると解釈できる．

(3) Netcraft社調査発表　http://news.netcraft.com/archives/web_server_survey.html
(4) 縮約には，UCINET 6 [8]のCONCORプロシージャを用いた．プロシージャのパラメタはすべてデフォルト値を使用した．

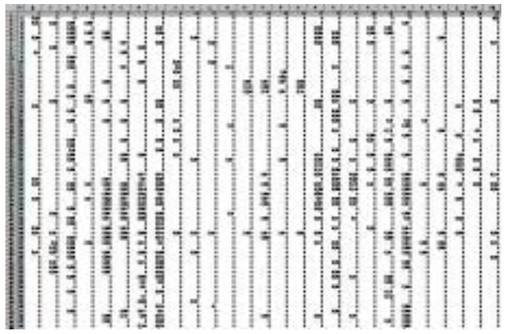
2点目に，②部分で，リクエスト処理エンジンとプロセス制御エンジンの関係性が高い．プロセス制御エンジン内では，コンテンツ毎にいろいろな処理を行う必要性があり，この部分をリクエスト処理エンジンが行っている．そのため，この部分の値が高いことは整合的である．

3点目に，③部分で，プロトコル解析関数とリクエスト処理エンジンの関係性が強い．これは，httpリクエストを処理するのに，各プロトコル解析をする必要があるためで，この部分の関係が強いことは整合的である．

4点目に，④の共通関数部分は，自分自身への依存度が高くない（(4,4)＝0.001）．共通関数という性質のため，他のモジュールから呼ばれたり，逆に呼んだりすることの方が多く，共通関数内部での呼び合いは少ないことを意味しており，これも整合的である．

この図を念頭に，再度要素毎の関係を見たのが図11である．

984×984の構造行



縮約化にCONCOR
アルゴリズムを使う

4×4のイメージ行　　呼ばれる側

呼出側	1	2	3	4
1	0.002	0.001	0.001	0.002
2	0.001	0.008	0.001	0.000
3	0.007	0.007	0.004	0.003
4	0.004	0.002	0.003	0.001

図8　Apache 1.3.0の場合の縮約化

行	名称	系列
1	プロトコル解析関数群	httpプロトコルの解析およびそれに関係のある文字列操作
2	プロセス制御エンジン	httpリクエストのサイクルの制御（プロセス制御）およびその記録
3	リクエスト処理エンジン	httpリクエストと関係する処理モジュール（コンテンツ種類の判別や権限制御）のネゴシエーション
4	共通関数群	設定ファイルやサーバの状態から変数を設定する

図9　イメージ行列の各要素の名称と内容

(3) アーキテクチャ測定への利用の可否

以上4点の解釈を通して，縮約化によって要素間の大まかな関係を維持したまま，構造行列がサイズの小さな行列（=イメージ行列）になったことがわかった．構造行列がわかっているときには，縮約化することで，アーキテクチャ測定に利用できることが確認できた．

5　時系列分析

5.1　時系列分析と製品アーキテクチャの変遷

前項では，構造行列の縮約化という手法を用いて，Apacheの構造行列のイメージ行列を作成し，製品アーキテクチャの観察を行った．これを用いて製品アーキテクチャの時系列分析を行う．製品アーキテクチャは，ある時点の状態が問題なのではなく，どのように変化するかという点が，実務面及び理論面からも重要である．

前出のApache 1.3系列には，いくつかのリリースバージョンがあると紹介したが，分析の時点では，19種類の情報を収集することができた（図12）．

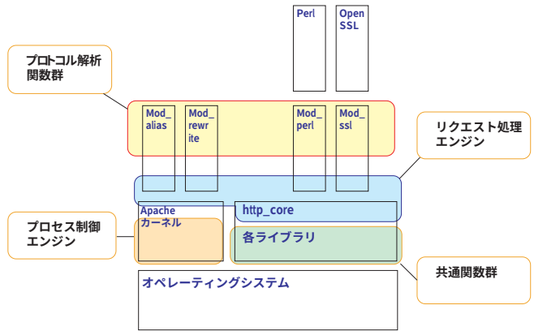


図10　Apache1.3.0の構造図とイメージ行列の関係

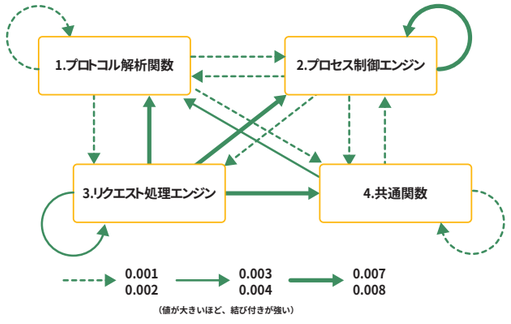


図11　縮約して得られたApache1.3.0のコールグラフ

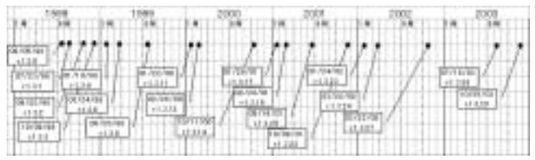
注意が必要なのは，図13にもあるように，1.3系列以外にも1.2系列や2.0系列が存在するということである．各系列は並行開発期間もあるが，おおまかには1.2系列，1.3系列，2.0系列の順にプロジェクトが進められている．

1.3系列の各リリースの期間を見ると，98～99年にかけては頻繁にリリースされ，00年以降ではリリースの期間が長くなっている．1.3系列は，03年以降もリリースされているがリリースの頻度は少なくなっている．

またソースコードのサイズをライン数[5]で見ると，最初のリリースでは約39,000行，19回のリリースの後に，約51,000行である．約5年間のことなので，サイズの増大は緩やかであるといえる．

5.2　アーキテクチャ指数

バージョン間のアーキテクチャの変化をインテグラル／モジュラーという視点から比較するために，アーキテクチャ指数を定義する．定義自体は簡単であり，モジュールが成立している行列成分の密度をとっている．アルゴリズムによる縮約の場合，ケースによっては対角成分≠0にならず，モジュールが成立しないことがある．しかしながら，今回の例では，約1,000×1,000行列を4×4行列まで縮約した結果，すべてのイメージ行列で対角成分≠0であった（図14）．イメージ行列の各成分に対して0を閾地として二値化を行った上で，アーキテクチャ指数を計算した。



■1.3系列は，1998年～現在まで，リリースが続いている
■対象のリリースは，version 1.3.0～1.3.29までの19リリース（約5年間）
■基本的な機能は，1.3系列で同じだが，リリースごとに機能追加や機能削除がある

図12　Apache1.3系列のリリース変遷

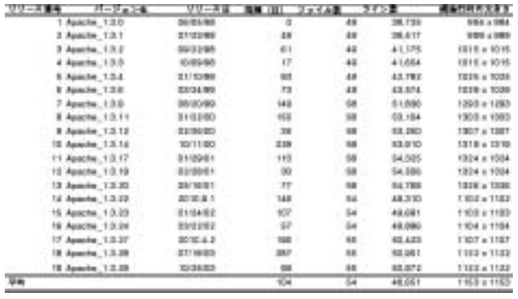


図13　Apache1.3系列のリリース詳細

$g \times g$ 行列の場合

$$\text{アーキテクチャ指数} = \frac{\sum_{i=1}^{g} \sum_{g=1}^{g} X_{ij}}{g\,(g-1)}$$

ただし　$i=j$：$X_{ij}=C$

$i \neq j$：

X_{ij} { 1 if $X_{ij}>0$
　　0 otherwise

図14　アーキテクチャ指数

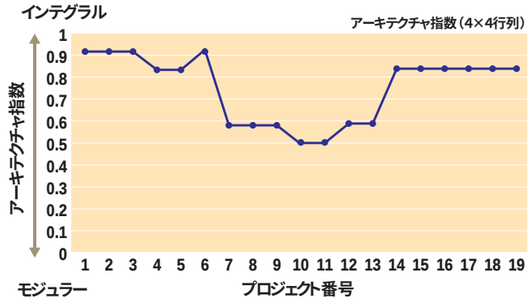


図15　アーキテクチャ指数の時系列変化

19バージョン間における構造行列のサイズを比較すると，最初の構造行列のサイズは，984×984行列であり，19回のリリース後でも1,122×1,122行列程度であった．サイズ的な違いはそれほど大きくない．図15は，各構造行列を4×4のイメージ行列に縮約し，上記で定義したアーキテクチャ指数をプロットしたものである．

バージョン毎のアーキテクチャ指数を観察すると，リリース6から7の段階で大きくモジュール型に落ち込み，その後リリース13まで同じような値で推移する．リリース13からリリース14にかけて，インテグラル型に振れ，現在に至っている．ちょうど前半部分はインテグラル型からモジュラー型にアーキテクチャ指数が振れ，その後モジュラー型からインテグラル型に移行している．

この理由はいくつもあると思われるが，リリースに付属している開発ログ[6]を確かめると，リリース7の時点で，システムに新しいモジュールを追加した旨が記載されている．「モジュールを追加できるような構造にしたこ

(5) ライン数でソフトウェアのサイズを測る事は，見積もりの観点からは問題があるとされているが，今回は，バージョン間でのソフトウェアのサイズの比較であるので，問題はないと思われる．各リリースには，いろいろなオプションが含まれていたが，すべてデフォルトでビルドしたときのソースコードの大きさを測定した．そのため，デフォルト設定に含まれないオプションとして提供されるモジュールのソースコードは，測定の対象となっていない．

(6) すべてのApacheのリリースには，CHANGESという名称の開発ログファイルが添付されている．

と」と「新モジュール追加」によりApacheのシステム全体のモジュール度が上がったのだと思われる．一方，リリース14の開発ログには，セキュリティの問題が高まったため，変更を行った旨が記載されている．おそらくセキュリティ問題は，その性質上，1つの部分で解決できるものではなく，該当する部分すべてに変更が必要であり，その結果，コードの見直しの際に，関数の呼び合い構造も変化したのではないかと思われる．

6　インプリケーションと課題

6.1　今回の研究のまとめと考察

今回の研究では，構造行列の縮約化という手法を用いて，アーキテクチャの測定を試みた．構造行列を縮約したイメージ行列と従来概念的に説明されていたアーキテクチャ図とを比較した場合に，十分整合的な説明ができることを確認した．

次に，測定の評価軸としてインテグラル／モジュラーの軸を念頭に，アーキテクチャ指数を定義した．このアーキテクチャ指数を用いて，時系列的にソフトウェアのアーキテクチャがどのように変化するかを検証した．その結果，当初は，インテグラル型であった構造が，リリースバージョンを経る毎にモジュラー型に変化する様子が観察された．さらに，その後，モジュラー型であったアーキテクチャがセキュリティの問題からコードの見直しが行われ，再度，インテグラル型に揺り戻しがあったことが確認された．

①製品アーキテクチャの定量

本研究の第一の意義としては，製品アーキテクチャを定量的に捉えることができた点である．従来の議論では，インテグラル／モジュラーという評価軸の定義が曖昧で，同義反復的に使われている場合もあった．この点を整理し，現実のソフトウェアに適応できたことに意義がある．

②製品アーキテクチャの移行（IMシフト）

第二の意義としては，アーキテクチャ指数を時系列的に測定することによって，インテグラル型アーキテクチャがモジュラー型へ移行することが確認された．

新しい製品分野の場合，初期段階では製品アーキテクチャがインテグラルであることは普通である．この段階では，製品システムをどう構成するかの情報を明確に定義することができないし，構成要素間がどのように相互関係しているかもわからない．このような問題には，「風通しがよい組織」「チームプレーの組織」といった特徴を持つ統合的組織が有効であるといわれている．統合的組織は，日本企業が戦後鍛えてきた能力にオーバーラップする部分が大きい．インテグラル・アーキタクチャの持つ複雑性の問題に対処できる組織能力を持った企業は，初期より新製品分野に参入することができる．

純粋に問題解決の困難さだけを考えれば，インテグラルアーキテクチャは，複雑性の問題が大きくなりがちである．この問題に対処するために，企業はインテグラルアーキテクチャをモジュラー型へ移行していく．モジュラー型アーキテクチャでは，サブコンポーネント間で標準が確立し，相互依存・相互作用が規定化される．

今回のApacheの測定では，モジュール化は，最初のリリース後，7つ目のリリースで最もモジュール化度が高くなった．リリース当初はインテグラル型アーキテクチャであり，その後モジュラー型に移行していくプロセスが今回観察されたわけである．

この現象に関して，小川[9]では，光ディスク装置産業を例にとり，CD-ROM，CD／RW，DVD-ROMといった各世代で，インテグラル→モジュラーへの周期的なアーキテクチャ・シフト（以下IMシフト）が起きたとしている（図16）．

そもそも，Apacheがリリースされ始めた95年から00年は，インターネットの爆発的普及があり，Apacheへの要求も大幅に増えていった．このニーズ変化に対応するため，Apacheの各系列では，それぞれの最初のリリースがインテグラルアーキテクチャになったのだと思われる．ある系列での問題解決をあきらめ，次の系列で問題解決したわけである．このような開発戦略の下で，Apacheの開発チームは，Webサーバへ要求の増大に答えることができ，世界でトップシェアとなったと考えられる．

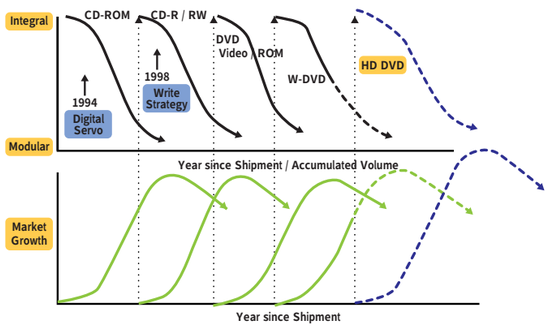


図16　光ディスク装置産業の周期的なアーキテクチャ・シフト[9]

しかし，小川[9]の議論は，さらに示唆的である．CD-ROMに始まる光ディスク装置の周期的なアーキテクチャ・シフトが起こったときに，日本企業は，このアーキテクチャ・シフトについていくことができなかった．新しい世代の初期には，日本のメーカが善戦するのに対して，製品アーキテクチャがモジュール化するに従い，台湾・中国メーカに苦戦を強いられているという．そして，この状況はCD-ROM，CD/RW，DVD-ROMといった光ディスク装置の各世代で起きているという．

アーキテクチャ・シフトに対応する企業戦略は，実証の分野では，とくに研究が盛んな分野であり，今後，研究成果が期待される．

③アーキテクチャの揺り戻し（MIシフト）

第三の意義としては，アーキテクチャの揺り戻しが確認された事である．アーキテクチャの揺り戻しに関してHDD装置のケースが報告されており，モジュラリティの罠として紹介されている[10]．

一度インテグラル型からモジュラー型へシフトしたアーキテクチャが，再度モジュラー型からインテグラル型へシフト（以下MIシフト）した．一般的に，インテグラル型アーキテクチャをモジュラー型に移行することは，自然な流れである．要素間の相互依存関係が大きいという問題を相互依存性が少ないように，問題の切り分けを行う訳である．ところが，これとは逆にモジュラー型アーキテクチャがインテグラル型に変化する事をどのように考えればよいだろうか．今回の事例では，「セキュリティの問題を解決する」といった製品へのニーズ変化・新ニーズの追加のためこのような変化が起こった．ある種のニーズ変化に対応する場合は，製品アーキテクチャをモジュラー型からインテグラル型に振らなければならない．MIシフトの問題は，開発組織にとって，IMシフトの問題解決のやり方とは，異なるスタイルが必要だと思われる．MIシフトの問題は，それまで，せっかく相互依存性を少なくし，きれいな設計にしたものを，ある意味では汚い設計にするわけである．複雑性の問題の再発を起し，元の木阿弥に戻ってしまう可能性もある．MIシフトの局面では，「筋のいいインテグラル化」が必要になると思われる．現実の組織として，「筋のいいインテグラル化」をする際に，どのような取組みが必要なのかは，今後研究が必要な分野である．

7　おわりに

今後の研究として，製品アーキテクチャ変化が，組織デザインやサプライヤとの関係に，どのような影響を与えるのか，開発パフォーマンスにどのような影響を与えるのか，最終的な収益パフォーマンスにどう影響するのかを明らかにすることが重要であると考えられる．

日本のソフトウェア産業を考える際に，エンジニアリング的な意味でのアーキテクチャは，一面にすぎない．経営戦略的な意味でのアーキテクチャとエンジニアリング的なアーキテクチャは，両輪であり，2つを同時に考えることが日本のソフトウェア産業の競争力向上の面で重要であり，必須であると思われる．

謝辞　この研究は，「市場とボランタリの協働としてのリナックス・モデル」（事業番号S-202-22）として特定非営利活動法人グローバルビジネスリサーチセンター（GBRC）が笹川平和財団から研究助成を受けた調査研究の一部である．ここに記して謝意を表したい．

参考文献

[1] 藤本隆弘，武石彰，青島矢一(編)：『ビジネス・アーキテクチャ』有斐閣，2001
[2] 柴田友厚，玄場公規，児玉文雄：『製品アーキテクチャの進化論：システム複雑性と分断による学習』白桃書房，2002
[3] Baldwin，K. Y. and Clark，K. B.：Design rules: The power of modularity. MIT Press.2000（邦訳，カーリス・Y・ボールドウイン，キム・B・クラーク『デザイン・ルール』安藤晴彦訳. 東洋経済新報社，2004）
[4] Ulrich，K. T.： Product architecture in the manufacturing firm. Research Policy. 24, 419-440. 1995
[5] 安田雪：実践ネットワーク分析: 関係を解く理論と技法. 東京新曜社，2001
[6] 金光淳： 社会ネットワーク分析の基礎: 社会的関係資本論にむけて，東京: 勁草書房，2003
[7] Burt，R. S.： Positions in networks. Social Forces. 55，91-122，1976
[8] Schwartz J.E.： An examination of CONCOR and related method for blocking sociometric data. In Heise，D.R.(ed.)，Sociological Methodology，p.255-282. San Francisco，Jossey-Bass，1977
[9] 小川紘一：「光ディスク産業の興隆と発展―日本企業の新たな勝ちパターンを求めて―」東京大学ものづくり経営研究センター（MMRC）ディスカッション・ペーパー，2005年3月，http://www.ut-mmrc.jp/
[10]楠木健，ヘンリー・W･チェスブロウ「製品アーキテクチャのダイナミック･シフト」（文献[1]所収）
[11]MMRC Discussion Paper Series No.1，藤本隆宏「日本型プロセス産業」の可能性に関する試論―そのアーキテクチャと競争力

ソフトウェア・アーキテクチャに関する技術動向

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　総括グループ　研究員
吉田　尚志

本稿では、複雑化、大規模化するソフトウェア・システムの開発において、適切な抽象化を推進し、効率的な開発を可能とする技術として注目されるソフトウェア・アーキテクチャに関する技術を整理して紹介する。

1. 抽象化の進化と残された課題

これまでソフトウェアの開発は、構成要素とそれら構成要素間の関係を記す抽象化の方法を変えながら、増えていく機能やそれに伴う複雑さに対応してきた。初期のソフトウェア開発ではプログラマが、サブルーチンと基本的なブランチ構造を用いてプログラムコードを記述した。その後、データフロー分析、ER図によるデータ・モデリング、情報隠蔽等の技術を組み合わせ、より大きな単位の機能の集合体であるモジュールを構成要素とするようになった。

モジュール化によって単純な機能の正しさの提供にとどまらず、開発の効率への配慮等、より高レベルでの抽象化が達成された。さらに、情報隠蔽や抽象データ型を発展させたオブジェクト指向技術の出現によって、抽象化の中心となる構成要素はコンポーネントやオブジェクトへと変化した。これにより構成要素の間には、動的束縛のようにプログラミングの時点ではなく、コードが動作する時点で初めて呼び出し先が特定される関係を実現する仕組みが実現された。その結果最近では、画面コンポーネントやトランザクション処理をはじめ、よく使われるオブジェクトはフレームワークに取り込まれたり、コンポーネントとして整理され、開発に容易に利用できるようになった。

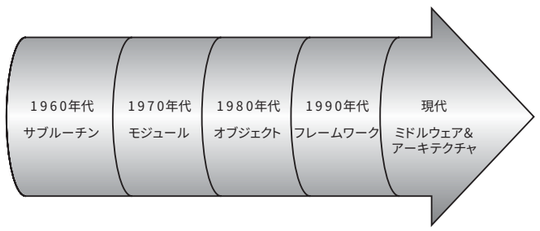


図1　ソフトウェア開発における抽象化レベルと構成要素

このような技術上の進化があったものの、ユーザ要求を特定のシステムに具現化していく過程で、

・優秀な設計者が頭の中だけで、時にはアドホックに、ソフトウェアの構成要素やその関係を決める
・まず特定の製品やシステム方式ありきで開発が進み、なぜそのような選択を行ったかという理由がないか、あるいは総合的には検討されていない

等、突然結果が決められており、入力（要求や制約）と出力（システムやソフトウェア）の間に”ギャップ”が存在する。そこで、ソフトウェアのシステムレベルでの構成を大きな粒度の部分とその組成を抽象化して示し、このギャップ部分の効率的な開発を可能とする技術として「ソフトウェア・アーキテクチャ」に関する技術が重要視されている。

2. ソフトウェア・アーキテクチャとは

単純にソフトウェアのアーキテクチャとは言っても、統一的に受け入れられている定義はなく、様々な定義が混在している。例えば、Software Engineering Institute（SEI）のWebサイト[1]にはソフトウェア・アーキテクチャの定義が新旧合わせて百もある。一般にソフトウェア・アーキテクチャは、Len Bassらの定義[2]やANSI/IEEE Std 1471-2000のもの[3]がイメージされている場合が多い。Len Bassらの定義は、以下のようなものである。

> **プログラムやコンピュータ・システムのソフトウェア・アーキテクチャとは、ソフトウェア構成要素、これら構成要素の外から見える特性、そしてそれら構成要素間の関係から成るシステムの構造のこと**

また、アーキテクチャという語から建築のそれと対比される場合も多い。建築の場合、例えば建築物の物理的な要素とそれらの関係を記すことで、建築家が家の快適さや明るさ、さらには美しさや雰囲気といった特性を提供する構造＝アーキテクチャを決める。これをソフトウェアの場合に置き換えると、ソフトウェア・アーキテクチャとは、ソフトウェアの構成要素とその関係を定義し、システムに要求される特性（性能、可用性、再利用性、保守性、信頼性等）を実現するためのシステムレベルでの設計とその思想であるといえる。

3. ソフトウェア・アーキテクチャの果たす役割と対応する技術

システムの目的に合致したソフトウェア・アーキテクチャを構築していくことは、以下のような観点から重要であり、ソフトウェア開発の効率や結果としてできるソフトウェアの品質に大きな影響を持つため、様々な技術の研究が盛んである。

①システムの利害関係者の間での意思伝達に利用するため

エンドユーザ、顧客、プログラマ、運用者、プロジェクト管理者、受発注それぞれの企業経営者等様々な利害関係者がソフトウェア・システムの開発に携わっており、ソフトウェア・アーキテクチャはこれらの関係者が共通で見ることができる抽象化されたシステムを示している。

②初期の設計判断を示し、それ以降の開発工程や運用で従う設計思想となるため

システムのライフサイクルで最も初期の、システムに関する設計判断であり、後に続くどの過程にも大きな影響を及ぼすとともに、最初にシステムを分析・評価することができる時点の構造である。

③システム間での再利用等大規模な開発効率化を可能にするため

プロダクト・ラインをはじめとして、似通った品質特性や要求を持つシステムの間で大規模な再利用を可能とすることや、特定のパッケージ製品等の市販されているソフトウェア部品の利用を検討するための基盤となる。

④システム保守の段階でシステム理解を容易にするため

開発者が保守を行う上で最も時間を割く、システム理解の作業軽減のためにソフトウェアの構成とそれを決定したときの設計意図や設計判断を明示するとともに、追加・修正を行う際にシステムレベルでの評価・分析を可能とする。

⑤開発を行う上で分割統治（Divide & Conqueror）し、開発効率を上げるため

適切な構造に分けて、設計思想を明示することで、複数の開発者がそれぞれの部分に対して効率的に開発作業を行っていくことができ、全体としての開発効率を上げることができる。

ソフトウェア開発時にアーキテクトは、上記のような用途を前提として、次のことを行う。

利害関係者から必要な要求を獲得する
アーキテクチャ構築のための知識や道具を用いる
案の作成・分析をしたり、代替案の比較・評価をしたりしながら、アーキテクチャを構築する
設計意図を利害関係者に伝えるとともにフィードバックを得て、再構築することを繰り返す
具体的なシステムへと実装されるアーキテクチャを作り上げる

このような開発において、ソフトウェア・アーキテクチャの果たす役割を図示し、関連する技術・研究テーマを整理して当てはめていくと図2のようになる。

この図の上部にはソフトウェア・アーキテクチャを構築していく上でのインプットとなる要求に関する技術・研究（[R]）と意思伝達への利用に関する技術・研究（[C]）のテーマが挙げられる。図2の下部では、アーキテクトがノウハウとして持っておくべき道具を示すメタモデルに関する技術・研究（[M]）、そしてそれらのノウハウを利用して個々のアーキテクチャ構築を行っていく分析手法や方法論についての技術・研究（[A]）、さらにはその

アーキテクチャを元に特定のシステムを作り込むことや、逆にシステム変更等にアーキテクチャを対応させるという同期化に関する技術・研究（[S]）のテーマが整理されている。

さらには、上記のように個々の技術としてソフトウェア開発に果たす役割に注目するのではなく、ソフトウェア・アーキテクチャ自体をドメインや製品に特化して開発全体の効率化を図る技術もある。例えば、製品ラインを意識してシステム間での再利用を試みるプロダクト・ラインの技術では、ソフトウェア・アーキテクチャのレベルでの再利用によって大規模な効率化を狙っている。また、ドメイン特化型のソフトウェア・アーキテクチャを定義していくような例（保険業界の例 [5]）も、同様に要求やアーキテクチャのレベルでのノウハウを共通に用いることで大幅な効率化を狙っている。

また、アーキテクトをサポートするツールとして開発の効率化を図る技術もある。例えば、ソフトウェア・アーキテクチャを記述するためのADL（Architecture Description Language：アーキテクチャ記述言語）は、通常その言語をベースとしたツールを伴う。ADLは、ソフトウェア・アーキテクチャをコンポーネントとコネクタ、及びその属性や役割といった要素でモデル化するように設計されており、図２の[M][A][S]の部分でのアーキテクチャ分析や具象システムへ実装（あるいは逆にリバース・エンジニアリング）するのを支援するような技術を提供している[(1)]。

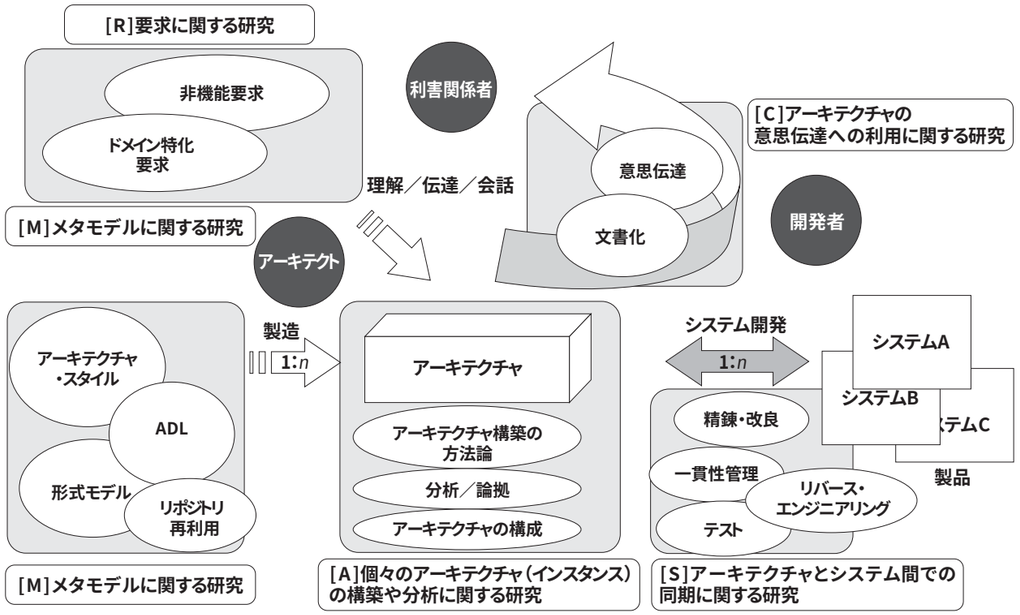


図2　ソフトウェア・アーキテクチャの果たす役割に注目し研究テーマを分類 ※文献[4]より

4. ソフトウェア・アーキテクチャとその要求

ソフトウェア・アーキテクチャは、ソフトウェア・システムの利害関係者の要望をはじめ、開発組織や開発者のおかれる状況・立場、経験・経歴、技術や知識に影響を受けながら、アーキテクトの判断で決められていくものであり、単にエンドユーザの利用する機能を示しているわけではない。例えば、エンドユーザにとって同じ「リクエストをしたら、結果を表示する」機能を持ったソフトウェアだったとしても、別のシステムでの再利用を考慮して作られたものとそうでないものでは異なる判断をベースとしており、異なったソフトウェア・アーキテクチャとなる。

ここで重要なことは、そもそも単純に「機能」が求められる場合、ソフトウェアが複数の構成要素を持つ構造になる必要がなく、１つの塊でよいことである。つまり、構成要素に分ける理由は実は「特定の入力に対して正しい出力を返す」ような機能的な理由ではなく、機能以外の要望や開発者の置かれる環境等の制約が原因に他ならない。こうした理由から、ソフトウェア・アーキテクチャを構築する場合、要求のうちでもとくに非機能要求として示されるものに注目する。

とくに非機能要求の獲得を目指した手法として、QAW（品質特性ワークショップ： Quality Attribute Workshop）がある。この手法では、非機能要求を具体的な品質特性を示すシナリオに落としていく単純な手順が示され、グループでのワークショップの行い方が定義されている。その手順は、

品質特性を1つ選ぶ
開発中のシステムにおいてその品質特性の主要な関心事を挙げる
主要な関心事が具体的にはどういうことか、シナリオとして記述する
ステップ1から、全ての品質特性を網羅するまで繰り返す

というものである。まず、品質特性を一通り網羅するべき品質特性のリスト（例えばISO/IEC 9126や文献[2]の6項目）から1つを選び出す。例えば、性能という品質特性の項目を選択する。次にその品質に対する主要な関心事をワークショップ参加者でリストアップする。例えば、先の性能という特性に対してはスループット、レスポンスタイム等、実際に性能として気にしていることを具体化する。最後に、それらの関心事を具体的なシナリオにしていく。そして、このステップを品質特性のリストを一通り網羅するまで繰り返す。

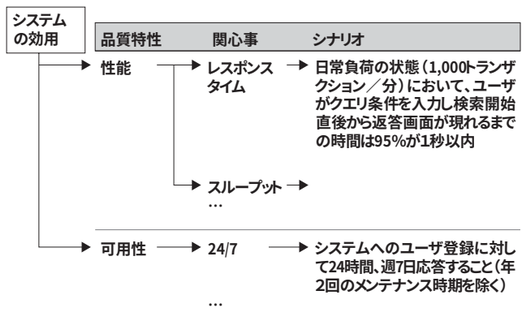


図3　ユーティリティ・ツリー例

シナリオは、できる限り詳しく書いておくことが望ましい。とくにテストや評価ができるよう推奨される形があり、入力、環境条件、期待される結果等を示すように指示されている。例えば、単に「レスポンスタイムは1秒」というのではなく、「日常負荷の状態（1,000トランザクション/分）において、ユーザがクエリ条件を入力し検索開始直後から返答画面が現れるまでの時間は95%が1秒以内とする」のように具体化して書いていく。このワークショップの結果として、アーキテクトが考慮しなくてはならない要件が定義され、システムの効用を示すユーティリティ・ツリー（図3）が完成する。

5. ソフトウェア・アーキテクチャの設計と記述

非機能要求を獲得した後、アーキテクトはその情報に優先順位を付けてアーキテクチャを設計していく。ユーティリティ・ツリーで記述された特性をどのように実現していくかを検討するときに、アーキテクトはそれぞれの品質特性に合った一般的な戦術にどのようなものがあるかを知っている必要がある。それは、例えば可用性を担保するために故障検出を行うハートビートを使う方法であったり、変更容易性を向上するための動的バインディングであったり、セキュリティ向上策として侵入検知であったりする。

これら戦術の実現方法は様々で、自前で開発をしなければならない場合もあれば、製品として提供されているものを利用できる場合もある。利用できる解を選択していく段階でよく用いられるのは、パターンやスタイルといった技術で、とくに著名なものとしては、文献[6]や文献[7]がある。さらには、これらのパターンやスタイルを特性と結び付けて整理を試みているABAS（文献[8]）や、品質特性をもとにアーキテクチャ設計を行っていく手法としてADD（文献[9]）も提案されている。実際、このようなシステム開発の「戦略」と言える部分を練っていく工程は現状ではアーキテクトの力量に依存する部分は多

(1) SECでは、ソフトウェア・アーキテクチャに関する技術研究の動向調査を行い、報告書（文献[4]）で2005年8月公開予定。ここに示したようなソフトウェア・アーキテクチャに関する技術の動向に関心のある方はご覧いただきたい。

いが、これらのノウハウや手法がシステムの効用を考慮した設計を効率的に行うのに貢献する。また、後の節で紹介するような分析・評価を行う手段があり、適切に行うことでソフトウェア開発の大幅な効率化が狙える。

アーキテクチャ設計時の重要な点の１つは既に述べたシステムを構築していく上での戦略を作っていく部分であるが、もう１つ重要なのは、作り上げたアーキテクチャ設計にしたがって開発が進んでいくことである。最終的にソフトウェア・システムの効用を最大化させるためには、この２つがうまくいかねばならない。後者の点で重要になるのが、文書化等記述の技術となる。

アーキテクチャを記述した文書には下記の３つの主な利用用途がある。

- **教育**： 新たにプロジェクトに入った人材等へシステムを紹介する
- **意思疎通**： 利害関係者が要求が満たされているか確認したり、要求を伝えたりする
- **分析**： 品質特性が満たされているかどうかを確認するとくに、開発を進めていく上では、どのような利害関係者がいて、どのような情報が必要かということがどのような文書を書くかに影響する。言い換えると、ソフトウェア・アーキテクチャの文書とは、様々な視点で適切に書かれた文書とそれらの視点間をつなぐ情報を文書化したものである。とはいえ、どのような視点で書いていけばよいのかについての大まかな指針としては、
- **モジュール的な視点**： コードの単位としてどのように構成されるか
- **実行時の視点**： 実行時に要素はどのように振る舞い、相互作用する構成となるか
- **配置の視点**： 特定の環境の構造にソフトウェア要素はどのように配置されるのか

が挙げられる。それぞれの視点で代表的なアーキテクチャのスタイルが存在し、それらも記述を行う上で参考になる。例えば、モジュール的な視点ではレイヤースタイルのアーキテクチャが存在し、これはそれぞれ１つ分上下のレイヤとだけやり取りをすることで特定のレイヤの置き換えを可能にする等の利点があるスタイルとなる。実行時の視点では、クライアント／サーバ型やイベント・キュー型等のスタイルが挙げられる。文書化というとUML等の特定の言語をイメージしがちであるが、アーキテクトは適切な視点でアーキテクチャ要素を記述し意思疎通を図るためにどのような情報を盛り込むことが必要かを考慮しておくことが肝要である。このような文書化の考え方については、文献[10]が詳しい。

6. ソフトウェア・アーキテクチャの分析・評価

アーキテクトは、ソフトウェア・アーキテクチャの設計を行い、適切に記述しつつ、それが実際にソフトウェア・システムとしての要件を満たすか、これから実装していくに当たってどのようなリスクや検討事項が残っているかといったことを分析・評価する。特定の品質特性、特定のスタイルが選択された設計をした場合には、この時点である程度定量的な分析を行うことも可能である。これには、リアルタイム・システムの振る舞いを分析するRMA（Rate Monotonic Analysis）や、キューを利用した仕組みを分析する待ち行列の理論等が活用できる。

一方、定性的な分析・評価も、ソフトウェア開発を効率的に進める上で重要である。この場合、有効な手法には、ATAM（Architecture Tradeoff Analysis Method）という手法[11]がある。この手法は、下記３つのポイントをあぶりだし、利害関係者間での調整を行うセッションを定義しており、

- **リスク**： 設計判断のうち問題が起こり得るもの
- **感度の高い点**： システムに要求される特定の品質特性を満たすために必須な設計判断
- **トレードオフ**： 設計判断のうち、ある品質特性を満たすことが他の品質特性を欠けさせるようなトレードオフになるもの

ATAMは、リスク軽減のための施策を早く打つことを可能とし、システムに必須な設計判断の論拠とトレードオフをアーキテクチャ文書に記述するきっかけを与えてくれる。ATAMは、以下の３つの参加者グループから構成され、４つのフェーズ（図４）とフェーズごとに更に細かいステップに分けられて実施方法が定義されている。

- **評価者**： 評価されるアーキテクチャ設計に関っていない人で、中立的な立場から進行等の役割を持つ
- **プロジェクトの意思決定者**： アーキテクトを含む、アーキテクチャ設計に対して判断や条件提示を行う人

プロジェクトマネージャや顧客も含まれる場合が多い

- **他の利害関係者**： 開発者、テスター、運用者等が含まれる

ATAMの肝となるのはフェーズ１、２の評価セッションである。セッションに向けてアーキテクトはたたき台となるアーキテクチャや代替案を検討する。セッションでは、ユーティリティ・ツリーをベースに重要な品質特性から順に実現方法をアーキテクトが提案して、参加メンバはそれの評価をしていき、リスク、感度の高い点、トレードオフをリストアップしていく。定性的にアーキテクチャの要点を挙げていく手法であるため、ATAMがとくに有効な場面は、実装が行われていないか（あるいは、まだ少ない）段階や、代替案を検討しトレードオフなどの調整を行う段階である。

アーキテクチャは一旦出来上がるとそれに従った組織構造を元に作業の分担や実装が始まってしまうため、変更は容易でない。そのためATAMのように、早めに要点を把握する手法や計画されたインクリメンタル型のプロセスを通じてリスクを回避しながら具体化していくことが重要である。

7. おわりに

ソフトウェアの範囲だけでもアーキテクチャを決めていくのは大変な作業である。しかし、ソフトウェア・システムの効用を最大化していくには、さらに視野をソフトウェアが搭載されるハードウェアや利用されるビジネス・シーンにまで広げて考えなければならない。例えば、組込みソフトウェアの場合は製品としての優劣を、そしてエンタプライズ系ソフトウェアの場合ではビジネスプロセスとの適合性等、それぞれアーキテクチャを決めていく段階で検討に入ってくるべきである。また、それらを適切なタイミングで投入できるようにするためのプロジェクト管理への理解も行えることが重要であろう。そのため、アーキテクトは、個々の分野の専門家たちとのコミュニケーションがとれることや広範囲の利害関係者の要望の調整を行っていける幅広い能力と知識も求められる。

今後は、日本においても既にスキルや経験を保持している技術者のアーキテクトとしての認知度アップと必要レベルに満たない技術者の今後のアーキテクトとしてのレベルアップが期待される。

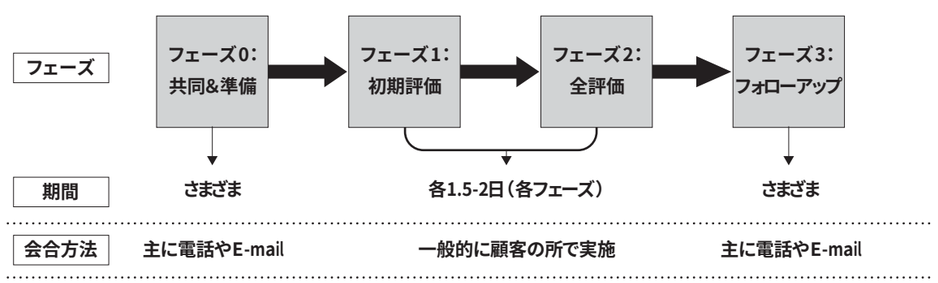


図4　ATAMの４フェーズ

参考文献

[1]How Do You Define Software Architecture?：http://www.sei.cmu.edu/architecture/definitions.html
[2] Len Bass，Paul Clements，Rick Kazman：Software Architecture in Practice 2nd Edition，Addison-Wesley，2003
[3]Recommended Practice for Architectural Description of Software-Intensive Systems，ANSI/IEEE Std 1471-2000
[4] ソフトウェアアーキテクチャ研究動向調査，経済産業省，2005，（http://www.ipa.go.jp/software/sec/index.php　2005年8月公開予定）
[5] IBM Insurance Application Architecture (IAA) http://www.ibm.com/industries/financialservices/iaa/
[6] F.Buschmann他：Pattern-Oriented Software Architecture 〜A System of Patterns〜，John Wiley & Sons，1996.（邦訳“ソフトウェア･アーキテクチャ 〜ソフトウェア開発のためのパターン体系”，トッパン，1999）
[7] M.Shaw，D.Garlan：Software Architecture-Perspectives on an Emerging Discipline，Prentice Hall，1996
[8] Mark Klein，Rick Kazman：Attribute-Based Architectural Styles，CMU/SEI-99-TR-022
[9]Attribute-Driven Design Method，http://www.sei.cmu.edu/productlines/products_services/add_method.html
[10] P.Clements 他：Documenting Software Architectures-Views and Beyond，Addison-Wesley，2003
[11]Architecture Tradeoff Analysis Method，http://www.sei.cmu.edu/architecture/ata_method.html

エンタプライズ系ソフトウェア開発における要求品質の確保

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
小林　陽二郎
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
室谷　隆

企業の情報化にあたり、現在起こっている問題の多くは、事業や業務検討の段階から要求定義までの「超上流」(1)にある。本稿では、この「超上流」において誰が、いつ、何をしなければならないかを関係者（経営者、利用（業務）部門、情報システム部門、ベンダ等）が共通に認識し、それぞれが果たす役割と責任を明確にして多くの問題を解決するために、要求品質(2)を確保すべきであるということを述べる。

1. 情報システムの現状

20世紀の後半から始まったコンピュータの高度活用は、ビジネスの仕組みを根底から変えた。業務の情報システムは、企業内の業務の効率化を目的として始まったバッチ中心のシステムから、企業の枠を超えて、ネットワークを中心に相互に接続され、巨大な社会インフラとなった。

このため、情報システムの優劣はビジネスや社会に影響を及ぼす時代になり、システムの停止は、ビジネスを停止させ、「システムリスクは経営リスク」となり、システム品質の向上はITガバナンスの重要課題となった。それゆえ経営課題として、「経営層がシステムに大きく関与していく時代」になってきた。

2. 情報システムの問題とその原因

情報システムは社会インフラとなったが、そのシステム開発の現場は、人間の手作業によるところが多いのが実態である。

(1) システム開発は人間技

現在の、ネットワーク化されたシステムに支えられた社会活動の安全性、生産性を保証していくためには、システム開発は人間技による、という事実を踏まえたうえでシステムの品質の確保を追求することが必要になる。現状では、人間技であることからシステム開発コストの増加、納期遅延、使われないシステム、開発担当者の疲弊等が言われている。

(2) 関係者・関係システムの増加

企業の内外においてシステムが相互に密接な関係を持つようになった結果、１つの施策を実現するためには、内部・外部の多くのステークホルダと調整し、それに対応する広範囲で大量のシステムに手を入れなくてはならなくなった。

(3) オープン化

近年、システム構成要素のオープン化により、メインフレーム時代のようなアプリケーション開発に集中できた時代から大きく変化した。

オープン化では、システム構成要素の選択肢が増した結果ソフトウェア開発の利用技術に、ハードウェアやネットワーク等インフラの活用において、業務にベストフィットさせる必要がある。そのため、パフォーマンス、キャパシティ、セキュリティ、障害対応、稼働時間、運用監視等、必要な非機能要件を踏まえた検討が、都度要求されるようになった。また、業務にベストフィットしたシステムでも必ずしもシステム全体の最適化にはならず、更なる対応を迫られることも少なくない。

しかし、多くの企業は、その進歩、変化の早さ、広がりの大きさ、多様さから、業務を、システム全体として、事業としてまとめることのできる要員を十分に確保することは簡単ではない。そのため、巨大な社会インフラとなったシステムに対応できる体制の充実が急務になっている。

(4) 運用テストで問題が顕在化

図１は要求／要件定義・仕様作成とテスト／評価の関係を示している。「評価」では、システムの方向性やシステム化計画を練ったものが、事業にとってどのような価値を生み出したかを評価する。同様に、「業務システム」では、利用者や運用者が運用テストによりシステムの正しさを確認する。ここで注目したいのは、「ITシステム」や「ソフトウェア」の確認は、開発担当によって行われるのに対して、「業務システム」の運用確認は、開発者ではなく、いわゆるエンドユーザ等の利用者が主担当になることである。

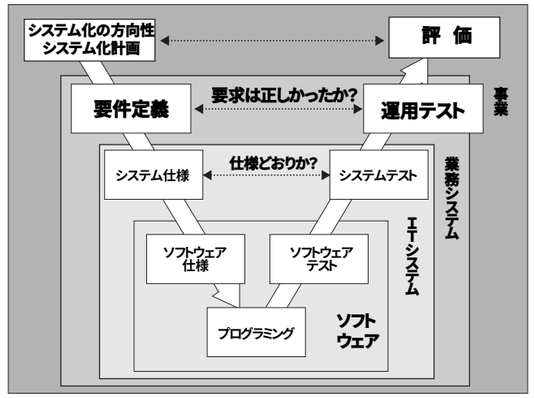


図1　要件定義・仕様とテストの関係

ITシステムの設計・開発・テストに入る前に、利用者のニーズが反映され業務仕様がきちんと定義されていないと、開発者が要求仕様通りに設計・開発・テストしても、運用テストで利用者が違いを見つけて要件定義工程からのやり直しになりかねない。

(5) 要求仕様書は書けなくなった？

ITシステム開発の現場で、開発プロセス以前の実態はどうか。JUASの調査[1]では、企業規模による若干の差はあるが、平均では、要求仕様書をユーザが作成する割合は16％、ベースはユーザで作成し細部はベンダが作成する場合の割合は33％という調査結果が出ている。また、発注者の反省点として、「システム仕様の定義が不十分のまま発注した」、「発注先に要求仕様条件（RFP）を明確に提示しなかった」という事項が第１位、第２位を占めていることも明らかになった。

ユーザ企業が、要求仕様書を書けない、書かない状況にあり、当事者も認識していることがわかる。しかし、開発に入る前の要求品質が確保されていない状況では、出来上るシステムが満足できるものにならないのは必然であり、開発担当者の努力だけでは開発は成功しないことを示している。

では、なぜユーザ企業が要求仕様書を書けなくなったのか。以前は、システム化とは、現実の業務をコンピュータに写像することであり、要件は非常に明確であった。現在では、ITシステムの広がりに伴い、未知、未経験の業務等、明確にできない業務要求を元に要求仕様書を作成することもある。また、前提、制約となるインフラ構成も非常に複雑になっており、開発着手時にすべが明確になっていないことも原因といえる。

(6) “ ITシステムを求める人 ” と “ ものづくりの人 ”

経営者やエンドユーザ等 “ ITシステムを求める人 ” は、企業利益や優位性の確保の視点から経営に役立つ機能、スピード、費用、品質等を求めている。これに対して、情報システム部門やベンダ等 “ ものづくりの人 ” は、システム開発の視点から仕様の固まり具合、ステークホルダの合意、IT技術、技術者のスキル等、万全の準備を求めている。両者の間には大きなギャップがある。

3. 超上流と経営者の参画の重要性

3.1 超上流

ここでいう超上流とは、以下に示す３つの工程と実施内容をまとめて強調した表現としている。

(1) システム化の方向性工程

この工程では、経営の方針や業務部門からの業務上のニーズ、あるいはシステムの課題等の要求をまとめる。各ステークホルダ間での利害も異なるため、十分な検討を繰り返しシステム化の方向性をまとめる。

(1)「超上流」とは、事業や業務検討の始まりから要件定義までの「工程」をまとめて強調した言葉である。
(2) 本稿で述べる「要求品質」とは、『要求の具体化の程度やステークホルダ間での要求の理解の程度』を指している。

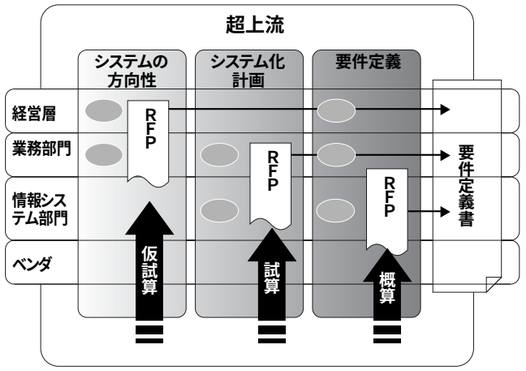


図2　役割分担

(2) システム化計画工程

この工程では、システム化の方向性を具体的にするために要求分析を実施し、システム化のための体制、スケジュールを含む計画書を作成する。

(3) 要件定義工程

この工程では、前工程で得られた要求の結果を分析して、全体の要件をビジネス要件、業務要件、システム要件、非機能要件として定義する。図２は、超上流としてまとめた３つの工程の役割分担を表している。

3.2　経営層の役割（ITガバナンスの実現）

(1) ITシステム部門への投資

すべての業務がシステムの上で動くようになった今日の組織では、企業が事業において価値を創出するために、経営者はシステムを中心において戦略を描く必要がある。ビジネス戦略は、システム開発力なくしては実現しない。ITシステム部門に重点投資するとともに、価値を創出する有能なベンダを選択し、さらに、業務部門の担当者を増やし、開発･発注キャパシティを拡大する事で多様なマーケットニーズに応えていくことが競争優位実現の鍵となる。

しかし、開発・発注キャパシティは無限に増やす事ができるものではなく、限りある資源を最大限に生かすためには、開発対象の選択と集中が不可欠であり、経営の観点から、経営の責任でこれを行う事がITガバナンスの要として求められる。

(2) 開発品質向上と障害管理

システムの利用は企業の外へと大きく広がり、影響範囲が大きくなっている。システムが止まると業務が停滞し顧客にも影響を及ぼす。例えば、ＡＴＭや交通機関の予約等のシステムがダウンした場合の社会的影響は計りしれない。したがって、ITガバナンスのうえでも開発品質向上と障害管理は重要である。システム障害リスクは経営リスクである。開発品質向上を実現し、システム障害リスクマネジメントを実現する経営者こそが、社会的責任を果たせる存在であるといえる。

(3) 投資効果（ＲＯＩ）の視点

経営者は、プログラムの塊を手に入れることを望んでいるのではない。プログラムによって実現されるビジネスの成果を手に入れたいのである。システム開発は、そのコストをビジネスの成果との関係でマネジメントされるべきである。ビジネスのサイズにあった、適切なサイズのシステム実現への一歩は、システム開発における投資効果分析および評価にある。

3.3　業務部門の役割

(1) 供給責任

マーケットでのシステムの利用者が増加しており、業務の企画、設計を行う業務部門はシステムの提供者として、業務の仕組みをシステムにして供給する役割と責任を担う。業務部門は、アプリケーションの責任者として、また個々のプロジェクトの責任者として、情報システム部門、ベンダとともに要件を確定し、システムを検証し、利用者に供給する責任がある。

(2) ステークホルダ間の合意形成責任

システム開発に関わるステークホルダが増えてきている。業務部門は、これらステークホルダの方針、意見、課題等について漏れなく正確に把握し、できることとできないことを情報システム部門、ベンダとともに切り分け、業務要件として取りまとめていく責任を果たす必要がある。ステークホルダもまた、システムの供給側に立った場合には積極的にシステム開発要件の策定に参加し、利用者ニーズを確実に把握して、効果的にシステム機能に反映していくことが必要である。

(3) 要件確定責任と説明責任

システム開発を受託側から見ると原則は「決めたとおりに作る」ことである。しかし、決め事も人間技である限り、見込み違い、思い込み、決め付け、聞き違い、聞き漏れといったことからくる「誤り、漏れ」は免れない。システム開発はそのような「誤り、漏れ」を解消していく過程ともいえる。システム開発における万全な準備とは、次工程に向けての「正確な要件」という情報の伝達であり、それを実現するためには、自分が次工程に伝える必要のある情報について、要件確定責任と説明責任を負う必要がある。発注側、受注側双方の説明責任がシステム開発において品質を確保する重要なポイントとなる。

3.4　情報システム部門の役割

(1) 全体最適の実現

システムは作り上げるまでは可能性の塊であるが、出来上がった瞬間に制約と条件の塊となる。アプリケーションを稼動させるシステムインフラも同様である。レガシーシステムを抱えた企業は、一方で新たなビジネスを実現するために戦略的なシステム開発を進めることもある。情報システム部門は企業の置かれたこのような状況を踏まえ、全体最適の観点からアプリケーションや、インフラを見直さなくてはならない。保有システムを可視化し、共通的で一貫した技法、構造を定義、活用できるようにすることが情報システム部門に課せられたミッションといえる。

(2) 非機能要件の取り込み

情報システム部門は図３のように、機能要件に加えて、多種多様な非機能要件を定義し、開発に取り込んでいかなければならない。そのため情報システム部門は、業務部門、ベンダと協力し徹底的に非機能要件を抽出して管理し、システム開発に反映させる必要がある。

(3) 開発品質向上の実現

情報システム部門は、システム開発のプロとして開発品質を高めることにより経営リスクの極小化を実現することが大切である。加えて、上記の「非機能要件の取り込み」で指摘したとおり、現在のシステム開発はプログラムに実装される機能要件の充足だけでは完了しない。さらにプロジェクトに参加するシステム供給側のステークホルダに関わる役割を明確にし、その作業進捗を管理していくことはIT担当者の重要なミッションである。進捗管理、プロジェクトレビューを活用することにより、プロジェクトを可視化し担当者に情報を供給し、タイムリに必要なアクションがとれるようにすることが開発品質を上げることになる。

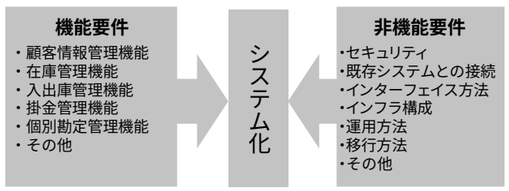


図3　機能要件と非機能要件

3.5　ベンダの役割

ベンダは顧客企業のビジネス戦略実現を担う存在である。ベンダはシステム開発のプロとしてその技術力、構築力を用いて、顧客企業の戦略実現を支える役割を担っている。

(1) バリューパートナ

ベンダは企業のビジネスにおける戦略目標を共有化し、システム技術を活用して、ビジネスの価値を創出する担い手でなければならない。そのためにはビジネスの性格、規模に応じた柔軟で適正、合理的なシステム提案をすることが肝要で、開発およびプロジェクトマネジメントについても可視化に努め、高品質を実現しなければならない。顧客企業のビジネスを通じて、顧客企業とともに価値を創出するバリューパートナとなることが、ベンダには求められる。

(2) ビジネス構想の段階から戦略パートナとして参加

現在のシステム開発では既存システムへの影響、社外のステークホルダの存在、新技術の活用、といったことも考慮しなければならない。このため、システム化計画のための周到な調査分析と実現システムのシステム設計が必要となる。企画を実現するには、仕組み、要員、予算、時間、稼動後コスト等を正しく出していかないと、そのビジネス企画自体が実現しない。過去事例実績、他事例実績、新技術提案、開発ノウハウを踏まえたベンダの提案がおおいに必要とされている。

(3) プロジェクトマネジメント

新規の開発にしても保守にしても巨大化・複雑化したシステム開発を管理するには、プロジェクトマネジメント能力が重要になる。

例えば、PMBOK等の先進のプロジェクト管理技法をシステムの開発に積極的に導入し活用することを意味す

る。

能力あるベンダによる高いレベルのプロジェクトマネジメントの実現が、高品質なシステム開発を実現し、企業が望むビジネスを実現する。

4. 超上流での作業内容と成果物

4.1 システム化の方向性工程とシステム化計画工程

超上流としてまとめた３工程のうち、はじめの２工程は、経営層、業務部門、情報システム部門というステークホルダが、合意形成を図る場である。それぞれの視点から「何をすべきか」「どのようにすべきか」を考え、合意の結果結論を導き出すことが必要である。さらに、おのおのの役割分担を行い、相互に連携を取り合いながら、自分たちが実現したいビジネスモデルと実現方法について十分に議論検討するべきである。そして経営層が明確な意思決定を行い、責任を負うことが必要である。

(1) はじめの２工程を進める上でのポイント

業務部門、情報システム部門を中心に行うはじめの２工程を進めるにあたって重要な７つのポイントを紹介する。

①最低限やりたいことを考える

プロジェクトの目的に応じてシステム化の範囲を決定する。欲張ることなく、真にやりたいことが何であるのかを考える。

②システム化ありきで進めない

プロジェクトの目的達成に向けて、課題を明文化し、その解決策はシステム化だけではなく、場合によっては手作業で行った方が効率の良いものもある。システム化ありきで進めず、必要性を見極めることが必要である。

③優先順位を決めてとりかかる

要件が膨らみ、コストやリソースの限界にきたプロジェクトは何かをあきらめなくてはならない。そのような場合を想定し、あらかじめ優先順位を付けておくことが必要である。システム化する機能も「目的達成のために欠かせない機能」と「できれば欲しい機能」に分けておく。

④シンプルな業務設計を心がける

手順が複雑な業務であっても、現場の担当者には違和感や課題認識が全くないということがある。また業務内容の経年変化により業務のための業務が存在していることもある。このような複雑な状態のまま新システムに置き換えようとすることは避けるべきである。システム化は現業務を見直す機会であり、無駄な業務や非効率な手順を客観的に評価し、新業務を一から設計することが大切である。

⑤死守すべきQCDを決める

プロジェクトは品質（Q）、費用（C）、納期（D）のうち、死守すべきものはどれなのかを明確にし、ステークホルダ間で合意しておく。

⑥プロジェクト体制の穴を見る

プロジェクトの推進体制のチェック対象は、体制上にすべてのステークホルダが現れているかどうか、体制的に弱い面はないか等である。また各担当者の役割、責任範囲、意思決定者も明確にする。プロジェクトメンバの割り当ては、必ず当該メンバの業務状況を確認し、本人にも確実に認識させるようにする。

⑦責任の所在を明確にしておく

「どの段階で誰が、何を、どの範囲で承認するのか」を決めておく。システム化計画工程での検討終了時、経営層を交えて検討結果を共有し、要件定義工程開始の承認を得る。また同時に、費用対効果やKPI（Key Performance Indicator：重要業績評価指標）等評価項目の確認と、評価のタイミング、その方法、及び評価結果に対する責任者を明確にする。プロジェクトを承認した経営層も責任の一端を担う。

(2) 役割分担と成果物

役割分担と成果物の例（一部）を表１に示す。

各工程で確認したこと、決めたことは成果物として必ず文書化する。

4.2 要件定義工程

要件定義工程は、設計・開発に入る前にシステム化のための要件をすべて洗い出し、確定する工程である。

(1) 要件定義と前後工程の関連

要件定義の前工程では、システムに対する大枠の要求事項がステークホルダ間で合意されていると想定する。一方、要件定義の後工程となるシステム設計には、要件定義からシステム化すべき機能について、より具体的な情報をインプットとして渡す必要がある。さらに、業務的な機能だけではなく、既存システム環境、マシンセンタ運用条件等の制約条件も明示しておく必要がある。

(2) 役割分担と成果物例

要件定義での成果物の作成は以下のように整理できる。

- When ：処理契機・処理間隔・頻度
- Where：処理所属・入出力場所
- Who ：処理主体・権限
- What ：処理データ・インタフェース
- Why ：処理根拠・法令・規約
- How ：処理方法・ロジック・操作

これらをすべて表現できれば、ドキュメントの種類に関して特にこだわる事はない。数々の事例から、上記を表現できるドキュメントを「機能要件」と「非機能要件」に大別し、さらに体系的に分類・整理したのが表２（一部）である。

表１　システム化の方向性工程とシステム化計画工程での役割分担と成果物例（抜粋）

名称（大項目） 名称（小項目）	目　的	内　容	主体（◎は主担当）
システム化方針の確認			
事業戦略／事業計画	事業戦略（商品戦略）や事業運営計画などを明確にする	事業戦略（商品戦略）、事業方針、売上計画、利益計画、要員計画などを明確にする	業務部門長（◎） 担当役員
中長期システム化戦略	事業戦略、既存システムのライフサイクル、システム技術動向などの視点でシステム化戦略を立案する	事業やシステムの課題、IT動向などを元にシステム施策を立案し、３年程度のレンジでスケジュールを立案する	システム推進担当

表２　要件定義工程での役割分担と成果物例（抜粋）

名称（大項目） 名称（小項目）	目　的	内　容	主体（◎は主担当）
機能要件（インタフェース）			
システム間関連図	検討対象のシステムと、既存システムまたは周辺システムとのデータの流れを明確にする	各システム間で受け渡されるすべてのデータの流れをフロー図で表現し明確にする	システム推進担当（◎） システム開発担当
システム間 インタフェース定義書	検討対象のシステムと既存システムまたは周辺システムとのデータのやり取りを明確にする	各システム間で受け渡されるデータ毎に以下を明確にする ・主なデータ項目、データ量、媒体等	システム推進担当（◎） システム開発担当

5. ま と め

システムの要件は、利用（業務）部門、ITシステム部門、その他の関係者がそれぞれの意志、思惑、目論見のなかで、同じゴールを目指さなければならない。そこで必要なことは、発信者が意図を明確に伝えること、伝えた内容を明文化すること、関係者間で共有することである。

そしてこのシステムの要件が「事業の要求」と一致していなければ企業にとって意味がない。経営層がシステム開発と深く関わることにより、「事業の要求」を明確にし、業務およびITシステムに反映させることで、効果的な経営を達成できる。

「経営層はシステム開発責任の当事者」と認識することが「要求品質の確保」につながる。

参考文献

[1] 日本情報システムユーザー協会（JUAS）：「要求仕様書（RFP）の役割分担」ユーザ企業IT動向調査 調査概要報告書，平成15年12月
[2] 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：SEC BOOKS 経営者が参画する要求品質の確保 ～超上流から攻めるIT化の勘どころ～，オーム社，2005

組込みスキル標準解説

東海大学情報理工学部ソフトウェア開発工学科教授　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

本稿では、組込みスキル標準2005年版について解説する。2005年版は、スキル基準v1.0、キャリア基準（ドラフト）および教育カリキュラム（ドラフト）の３部からなっている。スキル基準では、企業等の利用者がノウハウの流出を防ぎつつ、その都合に応じてスキル項目を設定でき、しかも利用者間での標準化を保障できるスキルフレームワークという仕組みを策定した。キャリア基準においては、ITスキル基準の構造から大きく逸脱しないように注意し、組込みソフトウェアに関する９つの職種を策定した。教育カリキュラムでは組込みソフトウェア開発に関する経験が無い人材を教育する効果的なカリキュラムと実施方法を策定した。

1. はじめに

産業機器、通信機器、運輸機器、家電機器、コンピュータ周辺機器等の機能を、マイクロコンピュータといったマイクロチップとソフトウェアを用いたシステムによって実現するとき、このシステムを組込みシステムとよび、そのソフトウェアを組込みソフトウェアとよぶ。2005年に発表された経済産業省の組込みソフトウェア産業実態調査によると、開発される機械機器のほぼ94％が組込みシステムを用いている。さらに、開発コストに占める組込みソフトウェアのコストが30％を超えている機械機器はほぼ半数にのぼっている。このことは、組込みシステムが産業横断的に使用される技術であり、しかもコスト的にも機械機器の機能を実現する主要な要素となっていることを示している。

開発される組込みソフトウェアは大きなものになると1,000万行を超えるものがあり、しかも組込みソフトウェアの開発期間は、約20％が6ヶ月未満であり、1年未満まで拡張すると約60％を超える。このことからも分かるように、組込みソフトウェアは肥大化の一方であるが、開発期間は非常に短い。このような厳しい状況になると、とても１社の手に負えるものではなくなってくるもので、事実80％を超える企業が外部委託を行っている。

このように機械機器開発の重要なポジションを占める一方で、開発量と開発期間の関係が非常に厳しい組込みソフトウェア開発に対して、高品質を求める要求が極めて高くなっている。その理由は、機械機器に求められる品質がそのまま組込みソフトウェアにも要求されるからに他ならない。ある意味で、不特定多数のユーザが世界各地で使用する機械機器においては組込みソフトウェアの不具合を発生させることは許されないことである。しかし、既に述べたように外部委託が多いことから、品質を確保するには技術的に体系化された開発工程とその管理技術、開発工程にしたがって開発できるスキルをもった人材の育成・確保が重要なポイントとなる。

以上の観点から考えると、企業間での開発・管理技術、人材のスキル評価基準、さらに職種の定義等を標準化することは、産業界全体の組込みソフトウェア開発力の向上に有効であることが理解できよう。

2003年10月に、経済産業省は「組込みソフトウェア開発力強化推進委員会準備会」を立ち上げた。この委員会には、産学官から約20名の委員が参加し、組込みソフトウェアに関するエンジニアリングと組込みスキル標準（Embedded Technology Skill Standards：以下ETSSと略記）について検討を行い、その検討結果を2004年5月に公開した。組込みソフトウェア分野の人材不足を解消するETSSの基本的な構造は図１に示す通りである。

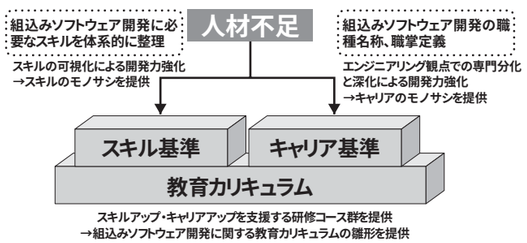


図１　ETSSの狙い

第１は、スキルの可視化による開発力強化を目指すもので、人材のスキルを評価する「モノサシ」を提供しようとするものである。

第２は、仕事を行う上での職種の専門性と深化による開発力強化を目指すもので、キャリアの「モノサシ」を提供しようとするものである。

第３は、人材のスキルアップやキャリアアップを支援する研修コース群の提供を目指すもので、組込みソフトウェア開発に関する教育カリキュラムの「雛形」を提供しようとするものである。

2004年7月には、「組込みソフトウェア開発力強化推進委員会」となり、委員を80名に増員して標準化の検討作業を継続させた。ETSSはこの委員会の中のスキル領域で検討された。スキル領域は、スキル基準部会、キャリア開発部会、教育部会の３部会構成となっている。さらに、2004年10月には、この委員会の活動支援及び検討内容の普及・啓蒙を行う組織として、SECが発足した。

本稿では2005年6月に公開したETSS 2005年版の背景と解説を行う。

2. プロジェクトマネジメントプロセス

2.1　技術の基本的な捉え方

組込みソフトウェア技術者に求められるスキルとはどのようなものであるかを把握しておく必要がある。そしてそのためには、技術とスキル（技能）の基本的な違いを明確にしておく必要がある。ここでは、ETSSを検討する上で共通認識とした技術とスキルについて述べる。

(1) 技術と工程

技術とは、要求に対する結果を経済原則に基づいて出力できることを保障する工程（プロセス）のことである。ただし、ここでいう工程とは、ハードウェアを含めたすべてのシステムの開発工程、製造工程、さらにはシステム内部におけるデータの入力、処理、出力といった一連のデータ処理の流れ等をさす。また、ここでいう経済原則の基準は要求によって束縛されるものであり、要求そのものの経済的価値から無縁のものではない。技術競争とは、的確な要求、要求を十分に満足する結果、要求から結果に至るまでの経済性を満足できる工程を開発することにあるといえよう。ただし、新たな工程の可能性等を原理原則から探る場合は、経済原則といった縛りは緩くなると考えてよい。この場合は、技術開発というよりは研究という色彩が強まり、将来への投資という性格が強くなり、経済原則的にはどのようなリスクをどれだけとれるかという観点を含めた競争となる。

(2) サブ工程

工程そのものはさらに細分化される。細分化された工程（サブ工程）は、法則、機械機器、人等といった資源から構成される。サブ工程の資源の違いがサブ工程間の特徴を明確にすることになる。完成された技術は、サブ工程の中の人的資源を除けば、すべて知識として伝達可能なものとなる。

さらに、技術を考えるとき、その対象は機械機器といった工学的な世界に限定する必要はない。経営、教育あるいはグラフィックデザイン等あらゆる対象の中に技術の種は存在していると見るべきなのである。対象としては、例えば機械機器を作り出す設計･開発･製造といった工程はもちろんのこと、機械機器の内部での入力から出力に至るデータや信号の処理工程も含んでいる。従って、先に述べた技術競争の世界は、社会のあらゆる分野にまたがっており、なおかつサブ工程を構成する個々の資源まで含めた細かい次元も含めて優劣を競うことになるのである。こうした技術の特性とソフトウェアの特性から考えるならばソフトウェアがあらゆる工程の中に組込まれていくことは自然なことと考えられる。

2.2　組込みソフトウェア技術

コンピュータやコンピュータプログラムは汎用的という性格から、社会のあらゆる分野で利用され、さらにサブ工程の資源として利活用しやすいという特性を持っている。とくに、機械機器内部のデータや信号処理工程にコンピュータ技術やコンピュータプログラムの技術等を利活用する組込み技術を用いた方が有利であることは、説明するまでもないであろう。こうした技術的側面から組込み技術をみるならば、機械機器内部のデータ処理のサブ工程の資源として組込み技術の結果を組込んだとき、それらを「組込みシステム」とよび、組込みシステムの

中のソフトウェアを作り出す工程を「組込みソフトウェア技術」とよぶことになる。

2.3　組込みソフトウェア技術者のミッション

これまで、技術の中での組込みソフトウェア技術を明確にしてきたが、その中での技術者の役割を明確にしておく必要がある。技術者の役割は技術競争力を確保することにあるが、その確保の仕方がポイントである。技術者が取り組む対象は技術としての工程であり、従ってサブ工程の資源を含めた工程の改善、改革等になる。組込みソフトウェア技術者にとっては、技術競争力を組込みソフトウェア技術によって強化することが最大のミッションである。そのミッションは次のように分類できる。

①組込みソフトウェア内部のデータ処理工程の設計
②組込みソフトウェアの開発工程の設計
③組込みソフトウェアの開発工程のサブ工程を構成する資源の確保
④組込みソフトウェアの開発工程の機能確保とスケジューリング確保
⑤経済性の確保
⑥後進の指導

これらを体系化し、さらに実現する方法、手段は限定されるものではない。すなわち、サブ工程を構成する資源をまったく別な資源に置き換えようと、複数のサブ工程を１つのサブ工程に置き換えようと、あるいは工程全体を大きく変えようと、すべては組込みソフトウェア技術者の責任の下に判断されるべき事である。

一般的に、工程を根底から覆し、それまでの工程を全て凌駕するような新たな工程が出現した場合、我々は技術革命とよぶ。技術競争の世界にあって、技術者に要求されることは、技術を革新することである。見方を変えると、技術革新あるいは技術の進歩は人材に依存するのである。技術によって生み出された機械機器などを技術そのものであるとした途端に、技術革新はストップし技術競争力を失うことを認識しておくべきである。

3. ETSSの考え方

3.1　組込みソフトウェア技術とスキルの関係

(1) 人的資源とスキル

組込みソフトウェア技術者が技術競争力を強化するミッションを遂行するには何が必要であろうか。組込みソフトウェア技術者が工程を改善あるいは革新できるためには、現行の工程、サブ工程あるいはサブ工程を構成する資源に関する知識や具体的な作業内容を十分に把握している必要がある。工程の中でとくに重要な点は、人的資源の能力にある。人的資源の能力は、知識の形だけでは伝達できないという特性を持っている。そのため、一定の能力を持った人的資源を確保できなければ、如何にすぐれた環境を揃えていても、工程を機能させることはできなくなる。工程の中での人的資源の価値は、その工程に与えられる要求や使用するツール等を使いこなすスキル（技能）という能力にある。すなわち、工程の中での人的資源には、経験を積むだけではなく、その経験の幅、量、質をスキルとして昇華させ、臨機応変にそのスキルを発揮できることが求められる。このようにスキルは個人に依存するものであり、個人の特性に応じてスキルの発揮させ方は異なってくる。当然、スキルに依存する割合が大きい工程ほど、その工程から得られる結果はスキルを提供する人的資源に依存することになる。スキルを提供する人的資源からみるならば、工程に依存して獲得したスキルが自身の価値を高めることにつながるが、技術革新によって工程そのものが消滅したり、あるいは工程で使用するツールなどが変わってしまうと、それまでに獲得したスキルは通用しないものになってしまう。

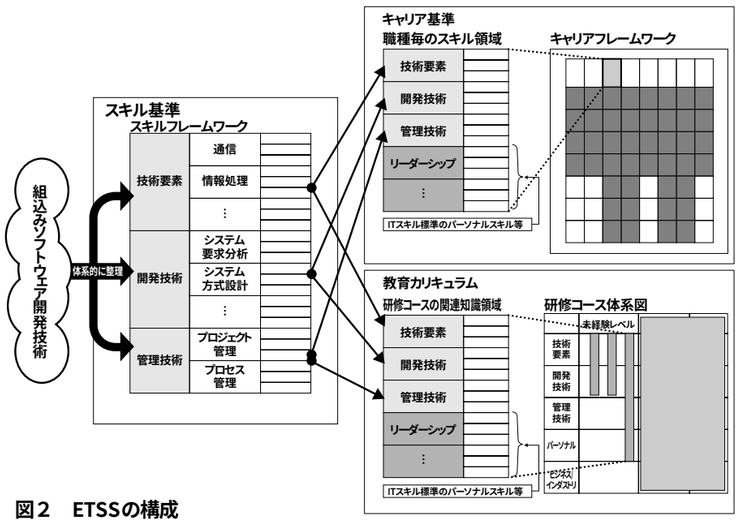


図２　ETSSの構成

(2) 組込みソフトウェア技術者に求められるもの

例えばあるサブ工程において、コンパイラといった環境を使いこなす能力、ある特定のプログラミング言語を用いてコードを作るプログラミング能力等はスキルとみることができる。学校の数学で微積分等の式を、修得済みの解法に従って解く能力等は完全にスキルとみることができる。この場合の解法が工程である。

スキルは重要であるが、スキルだけに偏った人材育成は、非常に危険である。スキルは工程に依存するため、その工程が新たな工程によって凌駕されると、スキルそのものが不要となり、人材の価値が低下してしまうからである。

こうした考察から、組込みソフトウェア技術者に求められるスキルとは、組込みソフトウェア開発工程の改善につながる手法や環境を使いこなす能力であるということになる。ただし、この能力を発揮できるようになるには、技術者自身が持っているスキルを否定できる潔さが求められる。

3.2　ETSSの構造

これまで述べてきたように、組込みソフトウェア技術者が持つべきスキルは、技術開発につながっていく性格を持つ必要がある。各企業の特性を超えて組込みソフトウェア技術者を育成し、人的側面から開発力を強化していくためには、こうした技術のスキルや考え方を統一化し広く標準的に使用できるようにするものさしが必要である。しかしながら、各企業が必要とする技術やスキルをすべて並べても、個々の企業にとっては多くの余分な項目が羅列にすぎないことになる。また、企業によっては技術の流出につながるため、その企業で特別に必要となるスキル項目を公開することはできないという事情もある。すなわち、総論賛成各論反対ということである。こうした条件をクリアさせた解が図２の構造である。

基本的な考え方は、ものさしをレベルの定義と技術項目の粒度に関する構造として捉えることである。原則的に個々のスキル項目は各企業あるいは業界団体等で定義するものである。いわば建物の構造は標準化するので、その間取りや内装は各企業あるいは業界団体等の利用者側において行うと考えればよい。ただし、ごく一般的に流通している項目はあらかじめ提示する。

4. スキル基準

4.1　スキルカテゴリと階層

スキル基準の基本構造は図３に示すスキルフレームワークであり、技術要素、開発技術、管理技術の３つのスキルカテゴリになっており、スキルカテゴリごとにスキル項目を記入する。その際、スキル項目は階層をたどるようにさらに詳細な項目に分解していく構造である。概ね1つの項目から階層を1つ深くするごとに最大10項目程度の粒度に分解する。階層の深さは最大で４階層程度までを目安とする。図中のスキルレベルについては後述する。

スキル項目をどの程度の粒度に分解し、階層の深さをどうするかは、どの程度のスキル評価をしたいかによってETSSの利用者側において適宜決めればよいのである。

(1) スキルカテゴリ	(2) スキル粒度			(3) スキルレベル			
	第1階層	第2階層	第3階層	初級	中級	上級	最上級
技術要素							
開発技術							
管理技術							

図３　スキルフレームワーク

図４はスキルフレームワークの基本的な構造を示したものである。スキル標準を使用するには、必要なスキル項目をスキルノードに割り当てることになる。また、ある階層のノードに割り当てたスキル項目をさらに詳細化していきたい場合は、必要に応じてそのノードから次の階層へ複数の枝を伸ばして各枝ごとにノードを設け、スキル項目を対応させていけばよいのである。図４では、スキル項目の粒度を高めるために、後からノードを追加しスキル項目を割り当てたり、あるいは黒丸で示すように追加したノードの粒度を高めるためにさらにそこからノードを追加できることを示している。こうした、ノードの追加あるいは削除、そして各ノード割り当てるスキル項目の内容は、ETSSの利用者が自由に行ってよい。また、そうして作成したスキルフレームワークの内容は、利用者のノウハウを守る立場から非公開のまま使用して構わないのである。

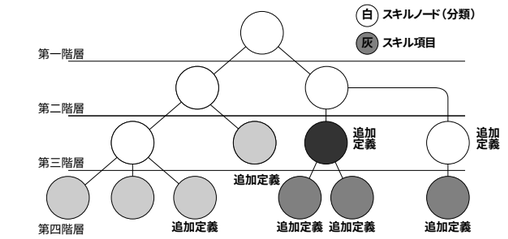


図４　スキルフレームワークの構造と使い方

第1階層		第2階層		内　容
1	通信	1	有線	WAN,LAN等有線通信技術
		2	無線	電気通信事業用無線,一般業務用無線等無線通信技術
		3	放送	デジタル方法,アナログ放送等放送技術
		4	インターネット	透過的データ転送,アプリケーション等インターネット通信技術
2	情報処理	1	情報入力	データ入力,音声入力等情報入力技術
		2	セキュリティ	暗号,著作権保護等セキュリティ技術
		3	データ処理	圧縮,データベース等データ処理技術
		4	情報出力	マークアップランゲージや文書ビューア等情報出力技術
3	マルチメディア	1	音声	データ処理,圧縮・伸張等音声処理技術
		2	静止画	データ処理,圧縮・伸張等静止画処理技術
		3	動画	データ処理,圧縮・伸張等動画処理技術
		4	統合	音声・画像等の統合処理技術
4	ユーザインタフェース	1	人間系入力	ボタン,座標等人間系入力デバイス制御技術
		2	人間系出力	表示,音声等人間系出力デバイス制御技術
5	ストレージ	1	メディア	リムーバブル,メモリ等ストレージメディア技術
		2	インタフェース	リムーバブル,常時接続型等ストレージインタフェース技術
		3	ファイルシステム	ISOやOSネイティブ等ファイルシステム技術
6	計測・制御	1	理化学系入力	電気,圧力,光等理化学系入力技術
		2	計測・制御処理	座標・運動,信号処理等の計測・制御技術
		3	理化学系出力	アクチュエータ,光,熱等の理化学系出力技術
7	プラットフォーム	1	プロセッサ	CPU,GPU等プロセッサ技術
		2	基本ソフトウェア	カーネル,ブート等基本ソフトウェア技術
		3	支援機能	情報記録,情報収集等支援機能技術

図５　技術要素のカテゴリ

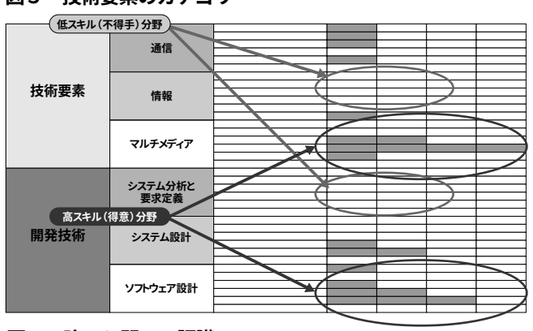


図６　強みと弱みの認識

4.2　スキルレベル

図３に示したように、スキル項目のレベルは４レベルで評価する。レベル評価は次の基準で行う。

初　級：上位者の指導のもとに実施できる
中　級：上位者の指導が無くとも自律的に実施できる
上　級：下位の技術者の指導が出来る
最上級：経験を体系化し先進的なやり方を工夫・開発できる

この最上級が組込みソフトウェア技術者に求められる組込みソフトウェアの工程の改善、革新を可能にすることを意味している。したがって、最上級が本来の技術者に求められるレベルであり、初級から上級の技術者はこの最上級を目指すよう心がけていくことである。最上級の技術者は、技術革新を目指すことはもちろんのこと、下位の技術者にスキルを修得させるとともに、少しでも改善・改革の方向を目指す技術的な思想を指導していくことを心がけるべきである。

4.3　スキル基準の使い方

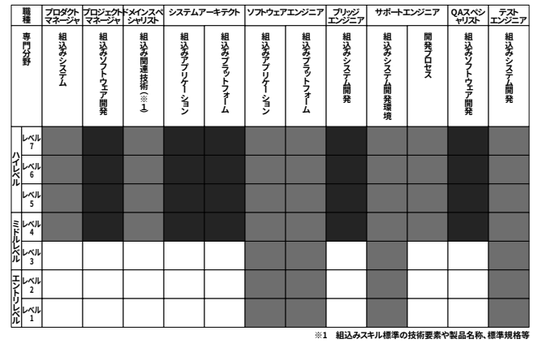
スキル基準は個人のスキル特性を評価することができるので、人材育成戦略に役立てることができる。図６はそのような場合を想定したものである。この図はある技術者個人のスキル特性を評価した結果である。ただし、スキル特性を評価する場合、すべてのスキル項目を対象とする必要はなく、利用者側あるいは技術者に関連するスキル項目に限定して評価すればよいのである。技術要素の中のマルチメディアのある項目に関しては、技術的リーダとしての力を備えていることがわかる。さらに開発技術においても、ソフトウェア設計では指導できるレベルにある。しかし、情報処理とシステム分析・要求定義に関するスキルは皆無であることから、マルチメディアという特定のプロジェクトであるならば技術の推進役として活用できることがわかる。また、この人材の仕事の範囲を広げるにはどうするかは、このスキル特性と今後の人材戦略に基づいて育成カリキュラムを設計することになる。

5.　キャリア基準

5.1　キャリアフレームワーク

基準のスキル基準が技術者個人の持つ技術スキルを評価する枠組みであるのに対し、キャリア基準は組込みソフトウェア開発を推進する上での専門性、経済性、責任性を評価するものである。言い換えると、組込みソフトウェア開発の中で、技術者として如何なるミッションを遂行するのかをわかりやすくカテゴライズしたものと考えればよい。今回の公開したキャリア基準はドラフト版である。

キャリアフレームは図７に示す通りであり、９職種からなっている。この９職種によって、組込みソフトウェア技術の開発力を推進することが出来ると考えている。ITSSと比較するとQAスペシャリスト、テストエンジニア等組込み分野特有の職種が上がっている。これは、機械機器等を大量に出荷した後で、ソフトウェアの不具合



※1　組込みスキル標準の技術要素や製品名称、標準規格等

図７　キャリアフレームワーク

職種と活動局面	専門分野	企　画	システム要求分析	システム方式設計	ソフトウェア要求分析	ソフトウェア方式設計	ソフトウェア詳細設計	ソフトウェアコード作成及びテスト	ソフトウェア結合	システム結合テスト	ソフトウェア導入	保　守
プロダクトマネージャ	組込みシステム	プロダクツ調査・分析 システム戦略策定	要求分析 要求仕様策定	要求仕様変更の検討および決定								保守変更機能策定
プロジェクトマネージャ	組込みソフトウェア開発		プロジェクト計画立案	プロジェクトマネジメント								
ドメインスペシャリスト	組込み関連技術 ※組込みスキル標準の技術要素や製品名称、標準規格等		要求仕様策定	システム方式設計	ソフトウェア要求分析／レビュアー	レビュアー			レビュアー	レビュアー		保守変更機能設計／レビュアー
システムアーキテクト	組込みアプリケーション		要求仕様策定	システム方式設計	ソフトウェア要求分析	ソフトウェア方針設計／レビュアー	ソフトウェア詳細設計／レビュアー		レビュアー	レビュアー		保守変更機能設計／レビュアー
	組込みプラットフォーム		要求仕様策定	システム方式設計	ソフトウェア要求分析	ソフトウェア方針設計／レビュアー	ソフトウェア詳細設計／レビュアー		レビュアー	レビュアー		保守変更機能設計／レビュアー
ソフトウェアエンジニア	組込みアプリケーション				ソフトウェア要求定義策定	ソフトウェア方針設計	ソフトウェア詳細設計	ソフトウェアコード作成／モジュールテスト	ソフトウェア結合テスト	システム結合テスト		保守変更機能実装／モジュールテスト
	組込みプラットフォーム				ソフトウェア要求定義策定	ソフトウェア方針設計	ソフトウェア詳細設計	ソフトウェアコード作成／モジュールテスト	ソフトウェア結合テスト	システム結合テスト		保守変更機能実装／モジュールテスト
ブリッジエンジニア	組込みシステム開発			コミュニケーション計画立案	拠点間のコミュニケーション管理						納入物評価／支援	保守変更機能実装／テスト
				リモートプロジェクト管理計画立案	リモートプロジェクトのマネジメント							
サポートエンジニア	組込みシステム開発環境			システム開発環境設計／開発環境調整計画立案	システム開発環境保守／運用							システム開発環境保守／運用
	開発プロセス	開発プロセス設計／開発プロセステーラリング／開発プロセス運用支援／開発プロジェクトモニタリング										
QAスペシャリスト	組込みソフトウェア開発			品質監査計画立案	設計品質監査			ソフトウェアコード品質監査	ソフトウェア結合テスト	システム結合テスト		
テストエンジニア	組込みシステム開発						システム総合テスト・設計	ソフトウェア結合テスト設計	ソフトウェア結合テスト	システム結合テスト	ソフトウェア導入試験	保守変更機能確認試験

主たる活動局面　　従たる活動局面

図８　職種と活動局面

によって重大な事故が多発することを防ぐ技術を追求するためである。

5.2 キャリア基準におけるレベル

図７の専門分野に対して、７レベルの評価基準を設定している。さらにレベルを把握しやすいように、エントリレベル、ミドルレベル、ハイレベルの３つの大きなレベルにくくってある。基本的には、これら３つのレベルの中を上位、下位に分けた結果がレベル１からレベル６である。レベル７はこの中でも特別なものとして位置づけている。キャリアのレベルをこのように７つのレベルで評価する構造にした理由は、ITスキル標準のレベルでの整合性を確保するためである。各レベルの評価内容は次の通りである。

エントリレベル ： 上位レベルの指導の下に、業務上の課題の発見・解決ができる

ミドルレベル ： 業務上の課題の発見・解決をリードする

ハイレベル ： テクノロジやメソドロジ、ビジネスをリードする

ハイレベル（レベル７） ： 新技術開発や標準化等により社内・社外をリードする

すでに述べたように、キャリアは経済性も含まれることから、例えばエントリレベルにおいては、上位レベルの指導がつくため経済性は極めて低い状態である。しかしミドルからハイレベルに移るに従い、社会に対する経済的および技術的責任の度合いは増していくことになる。

5.3 キャリアの活動局面

ここで定義したキャリアの技術上の仕事の内容を示したものが図８である。職種及び専門分野の責任をどこで果たすかという場面は、組込みソフトウェアの開発工程に対応させて示してある。濃く塗りつぶした箇所は主たる活動局面であり、薄く塗ってある箇所は従属的な活動局面であることを示している。塗っていない箇所は責任が無いことを示す。

なお、これまで述べたキャリアフレーム等は、企業の組織をこのようにしなければならないという意味ではなく、あくまでも参考として各企業の実情に合わせて利用していただくものである。

6. 教育カリキュラム

6.1 教育カリキュラムの構成

技術者個人の立場からは、自己のスキル特性をさらにどのように向上したいか、あるいはキャリアをどのようにデザインしたいかという希望が生じるであろう。また、企業という組織側からは、スキル調達あるいはキャリア全体を示すマップによる人材戦略に基づいて人材の育成あるいは調達を進めることになる。

このようなミッションにおいては、人材育成の仕組みを持っているか否かは、企業の技術力推進の上で極めて重要である。人材を育成する上での基本的な目標は、スキル項目や技術の意味を習得させることである。そのためには、項目を教える順序や習得内容を決めたカリキュラムが必要がある。図９は今回公開した教育カリキュラム（ドラフト）である。ここで示した教育カリキュラムは、スキルの初級レベルの前段階に位置する人材向けの内容であり、そのため新たなレベルとして未経験レベルを定義した。このレベルは、まだ組込みソフトウェア技術のスキルを獲得していない新人や他の分野の技術者あるいは学生のレベルを指していると考えればよい。

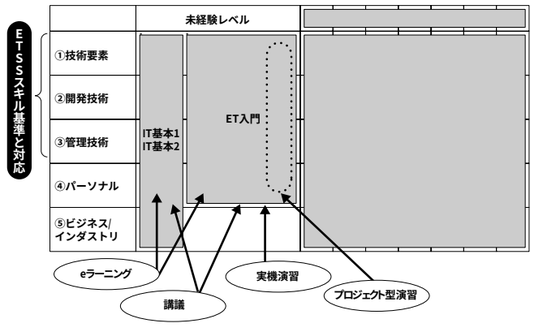


図９　教育カリキュラム

また、一方でITスキル標準の研修ロードマップがすでに定義されているため、ETSSにおいてはこの中のIT基本1とIT基本2の修得を前提として、その上で組込みシステム技術、組込みプログラミング演習、組込みソフトウェア開発プロジェクト型演習を入門コースとして設置した。

研修コース名	概　要	関連知識項目
組込みシステム技術	組込みソフトウェア技術者として必要な組込み基礎技術を修得する。	組込みシステムの歴史、組込みシステムの特徴、組込みシステムの現状、I/O制御、スタートアッププログラム、メモリ管理、割り込み処理、ハードウェア監視、排他制御、トレードオフ処理、ハードウェアアーキテックチャ、MPU周辺技術、基本I/O、外部周辺機器、実装技術、高信頼性設計、安全性設計、システムLSI、組込みソフトウェアの特徴、リアルタイムカーネル、デバイスドライバとミドルウェア、マルチタスクプログラミング、実行管理、開発環境、組込みソフトウェア開発技術……等。
組込みプログラミング演習	組込みソフトウェア技術者として必要なC言語を中心とする技術を修得する。	メモリ配置、スタックサイズ、スタートアッププログラム、割り込み処理、I/Oアクセス、コーディング作法、最適化、開発支援ツール（総合開発環境、コンパイラ、デバッガ、…）、アセンブリ言語、要求定義、ソフトウェア設計、プログラム実装、テストとデバッグ、……等。
組込みシステム開発プロジェクト型演習	組込みシステム開発未経験者向け教育カリキュラムの総まとめとの位置づけとして、組込みソフトウェア開発に従事するために必要な技術や知識をプロジェクト型演習にて体験の上、修得する。	※本研修コースの履修条件である、「ET入門コース」カリキュラムにおける「組込みシステム技術」、「組込みプログラミング演習」の関連する基礎知識、及びこれらの履修条件となっている、ITスキル標準の教育ロードマップにおける「IT基本1」、「IT基本2」の研修コース群の関連知識項目をプロジェクト型演習で実際に活用し、より実践的な知識や技術の修得を行う。

図10　研修コースの概要

6.2 研修コース

図10は具体的な研修コースの内容である。このコース内容は大学等で学生教育のためのカリキュラムとして導入してもよいし、科目として導入してもよい。また、情報系の技術者を組込み系へキャリアチェンジする場合の研修コースとしても利用することを想定して体系化してある。関連知識項目を参考にして、各企業もしくは各個人にあったカリキュラムを策定することができるであろう。

6.3 研修方法

図11は教育効果を引き出す方法としてプロジェクト型の研修方法を提案したものである。基本的に、研修としては講義、演習、ｅラーニング等があるが、スキルと技術をセットで獲得する最善の方法はこのプロジェクト型研修であると考えている。プロジェクト型演習では、それまでに履修した内容を実際のプロジェクトと同様に、要件定義からテストまでの一連のソフトウェア開発ライフサイクルを体験する事で習熟度を高めていくものである。

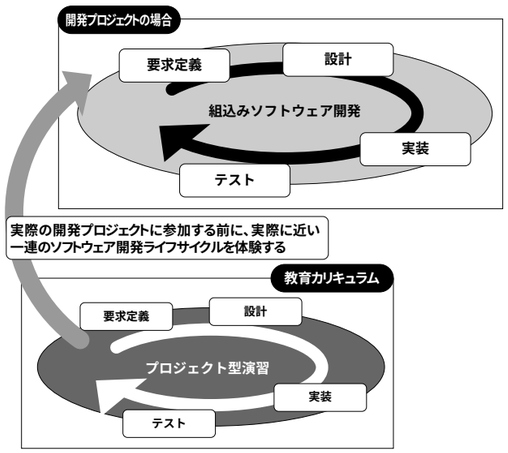


図11　プロジェクト演習

本論の最初で述べたように、特に組込みソフトウェアにおいては、技術とスキルを峻別する必要があると同時に、スキルに関しては経験を積み、技術に関しては工程を改善するという特徴を正しく修得する必要がある。この観点から、本ETSSで提案するプロジェクト型研修はさらにOJTとして展開させることも可能であり、ある意味で画期的な仕組みである。

7. おわりに

本稿では、技術とスキルに対する考え方を説明し、その延長線上にあるETSS（2005年版）の仕組みについて解説した。2005年版は組込みスキル基準のみがv1.0であり、キャリア基準と教育カリキュラムはドラフトとなっている。今年度は、キャリア基準についてはv1.0を目指し、教育カリキュラムは実際に現場の技術者育成に役立つ仕組みを検討する。また、スキル基準については、実際に企業へ展開するための方法等について実証実験等を行ない、現実的な解を求めていく方針である。

最後になるが、ETSSを策定し、さらに本稿をまとめるためにご協力いただいたSEC研究員の皆様、組込みソフトウェア開発スキル領域の各委員の皆様、貴重なご意見をいただいた産官学の皆様にこの場をお借りして感謝する次第である。

参考文献

[1] 独立行政法人 情報処理推進機構　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
[2] 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：2005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
[3]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：SEC Books 組込みスキル標準ETSS概説書（2005年版），翔泳社，2005

組込みターゲットアーキテクチャ

慶應義塾大学理工学部　教授　工学博士
天野　英晴

組込み用のマイクロアーキテクチャとしては、低コストで高い性能を実現できるRISC（Reduced Instruction Set Computer）型のマイクロプロセッサが主に用いられてきた。しかし、近年、IT機器の多様化、高機能化に伴い、強力な演算性能が要求されるようになり、ヘテロジーニアスなマルチプロセッサ、ホモジーニアスなマルチプロセッサ、プロセッサアレイ、コンフィギャラブルプロセッサ、リコンフィギャラブルプロセッサ等様々なマイクロアーキテクチャが登場し、発展を遂げている。本稿では、アーキテクチャの立場からこれらを整理しつつ紹介する。

1. はじめに

組込み用のマイクロアーキテクチャとしては、低コストで高い性能を実現できるRISC（Reduced Instruction Set Computer）型のマイクロプロセッサが主に用いられてきた。ARM、MIPS、SH等、それぞれ効率的な命令セットを持つプロセッサが独自の発展を遂げ、これらを対象として様々な組込み用OS、アプリケーションソフトウェアが開発された。しかし、近年、IT機器の多様化、高機能化に伴い、画像処理、音声処理等のメディア処理、無線通信を含むネットワークへの接続、暗号化、復号化、個人認証等、強力な演算性能が要求されるようになった。これに対応し、ヘテロジーニアスなマルチプロセッサ、ホモジーニアスなマルチプロセッサ、プロセッサアレイ、コンフィギャラブルプロセッサ、リコンフィギャラブルプロセッサ等様々なマイクロアーキテクチャが登場し、発展を遂げている。これらのマイクロアーキテクチャは数多くの雑誌の記事、特集で紹介されているが、これらの紹介記事のほとんどは、SoC（System on a Chip）、協調設計、プラットフォーム設計等チップ設計の視点に立っており、設計手法と出来上がったチップの性能、省電力技術の紹介等に重点が置かれていた。

本稿は、アーキテクチャの立場からこれらを整理しつつ紹介し、これらのアーキテクチャの上での組込みソフトウェアの開発における問題点について述べる。

2. 組込み用CPU＋Xのアプローチ

組込み用マイクロアーキテクチャが、PC等に用いられる汎用アーキテクチャと最も異なるのは、ある特殊な目的にのみ必要な性能が実現できればよく、あとはコストが小さく、消費電力が少ない方が有利な点である。そこで、組込み用マイクロアーキテクチャとして最も単純かつ効果的なのは、図1に示す組込み用CPUに目的別アクセラレータを接続した構成である。目的別に特化されているとはいえ、多くのIT機器でユーザインタフェースは必要であり、I/Oを管理し、システム開発を容易にするため、組込み用OSを走らせる必要がある。これには今までの実績のある組込み用CPUを用いるのがよい。しかし、例えばこの機器が画像を圧縮、伸張を行うため、MPEG2のエンコード・デコード機能を必要とする場合、従来の組込み用CPUの演算能力では必要性能を満足することはできない。そこで、エンコード、デコードの処理に関して必要な性能を満足するためのアクセラレータを組み合わせることになる。さて、それではこのアクセラレータXをどのような方式で実現するか、代表的なアプローチを紹介する。

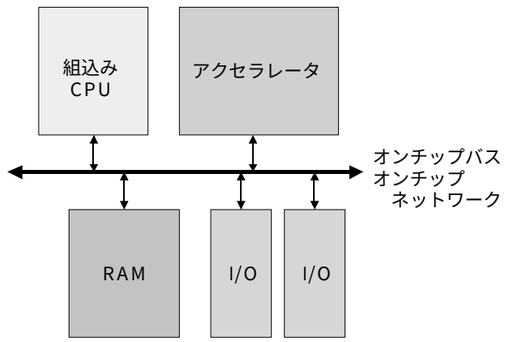


図1　組込みCPU＋アクセラレータの構成

2.1 組込み用CPU＋専用ハードウェア

まず、アクセラレータ専用のハードウェアを用いる方法が考えられる。この方法は最も面積が少なく、消費電力の点でも最適化が可能である。しかし、もちろん、対象毎に専用ハードウェアを設計しなければならないのが問題である。この構成のシステムを対象に、SoC（System on a Chip）設計技術あるいは、ハードウェア、ソフトウェア協調設計技術が発達した。このアプローチでは、まず、ソフトウェアを含むシステム自体をSystemC、SpecC等のシステム記述言語[1]で記述する。そして、このレベルで、シミュレーションにより専用ハードウェア化する部分に関してコストと性能を十分検討し、組込みCPUのソフトウェアで実行する部分と専用ハードウェアで実行する部分に切り分ける。切り分けが決まった後は、高位合成技術によって、ハードウェア構成と組込みCPUで実行されるプログラムを自動的に生成する。図2にこの手法の概念図を示す。もちろん、実際には、全てを自動的に生成することは難しく、人手による指示、RTLシミュレーションとVerilog HDL, VHDL等ハードウェア記述言語のレベルでの設計変更、試行錯誤が必要となる。このアプローチでは、専用ハードウェア部をいかに短期間に効率よく設計するかが重要であるが、一度設計したハードウェア資産をIP（Intellectual Property）化して再利用する技術も重要である。

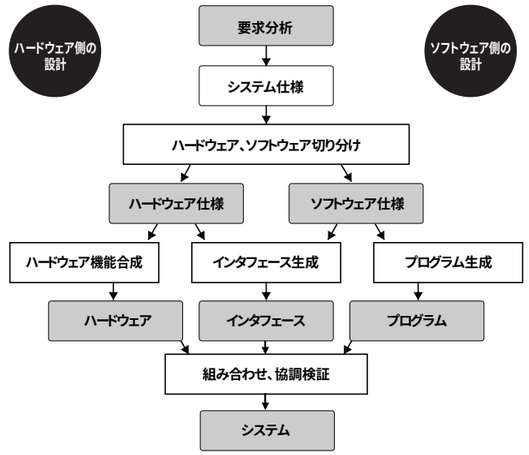


図2　協調設計の流れ

2.2 組込み用CPU＋専用プロセッサ

専用ハードウェアは、面積、消費電力を最適化することが可能だが、柔軟性に乏しく、特定用途にしか用いることができないという欠点がある。また、バグの修正、機能追加、変更も困難である。さらに、協調設計技術、高位合成技術の発達により、設計の上流は簡単化が進んでいるが、最新のプロセスを利用する場合は、配線遅延の問題により、むしろ配置配線等下流設計に時間と設計コストの多くが費やされる。このため、専用ハードウェアを設計するためのコストは依然として小さくならない傾向にある。そこで、専用ハードウェアに代わって特殊な目的にのみ特化して高い性能を持つ専用プロセッサを利用する手法が考えられる。これには以下の種類がある。

(1) DSP（Digital Signal Processor）

特定長の積和演算が特に高速な専用プロセッサで、長い歴史を持つ。VLIW（Very Long Instruction Word）型命令やSIMD（SingleInstruction stream, Multiple Data Streams）型命令の導入により並列性を生かした高速化が可能である。動作周波数も向上しており、TI社の最新のDSPは1GHzの動作周波数を実現している[2]。一方で、省電力技術を細部まで用いることで、電力消費は汎用プロセッサに比べてはるかに低く抑えている。

(2) Stream Processor

DSPよりもさらにメディア処理用に高い性能を実現する専用プロセッサ。主としてSIMD的に動作する多数の演算装置と、ストリームデータの高速入出力用機構を持つ[3]。

(3) グラフィック専用プロセッサ

グラフィック専用のGPUは、頂点演算用のVertexプロセッサ、色値演算用のFragmentプロセッサを組み合わせた構成で、全体として強力

な浮動小数演算機能を実現する。画像データ格納用メモリを同時にアクセスする機能を持つ[4]。

(4) ネットワークプロセッサ

高速ネットワーク処理用プロセッサで、パケット格納用のFIFO、IP checksumの計算、暗号、復号処理用のハードウェアを持つ[5]。

専用プロセッサのプログラム用には、C言語からのコンパイラ、デバッガが用意されている。とくにDSPは、様々なオプションを組み合わせることで、各種並列化を試みることができる。しかし、ぎりぎりのチューニングにはアセンブラのプログラムが用いられる場合もある。

2.3 組込みCPU＋FPGA (Field Programmable Gate Array)

FPGAは、LUT（Look Up Table）という小規模なメモリとフリップフロップから構成する論理ブロックを静的なスイッチで接続した構造を持つプログラマブルな論理素子で、90年代から急速な勢いで発展を遂げている。このうち、LUTの内容やスイッチの接続等を示す構成情報をSRAMに格納するSRAM型FPGAは、100万ゲートを越える大規模なものから、専用ハードウェアに迫る低コストのものまで様々な製品が用いられている。このFPGAを組込みCPUと組み合わせることにより、プログラマブルなアクセラレータを実現することができる。

Xilinx社のVirtex II Pro、Virtex IVはPowerPCをFPGA上に混載しており、Altera社ExculibourはARMとFPGAを混載している。図3にVirtex II Proの構成を示す。このチップは、中心部にIBM Power PCを持ち、一定の間隔でRAM、乗算器を配置し、周辺には高速リンク用のネットワークインタフェースを持っている。ただし、現在のところ、これらの商用チップは組込みシステムに用いるには余りにも高価であり、多くの場合、開発時のプロトタイピングに用いられる。FPGA部の設計はVerilog HDL, VHDL等のRTL記述を用いる場合が多いが、Handel-C等Cレベル設計も普及しはじめている。

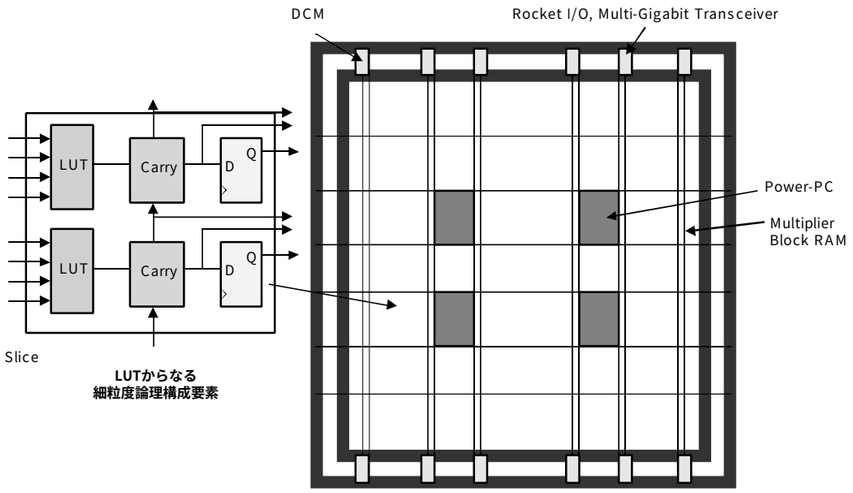


図3 組込みCPUとFPGAの混載例(Xilinx Virtex II Pro)

2.4 組込みCPU＋動的リコンフィギャラブルプロセッサ

FPGAは、細粒度の論理ブロックを用いることで、任意の機能を実現するのには有利だが、演算回路を実現するためには大きな面積を必要とする。そこで、図4に示すように演算器、シフタ、レジスタファイル等を組とした一種のプロセッサを構成要素とし、これらをFPGA同様の静的なスイッチで接続したのが動的リコンフィギャラブルプロセッサである。これらの構成要素はプロセッシングエレメントと呼ばれるが、実際は単純なプロセッサのデータパスに相当し、メディア処理等多数の演算を並列に行う場合、FPGAよりも効率がよい。このようなリコンフィギャラブルプロセッサは、構成情報を動作中に切り替える機構を設けることで、同一の半導体面積を複数の目的に用いて、面積効率をさらに改善する機構を持つ。IPFlex社のDAPDNA-2、NECエレクトロニクスのDRP-1は、ハードウェア構成を1クロックで切り替える機能を持つ。PACT社のXpp、Elixent社DFA1000は、1クロックでは切り替えられないが、多数の構成情報をチップ上に持たせることが可能である。Stretch社のS5000は、組込みCPUに密結合して、ユーザ定義可能な命令の形で利用できるようになっている。SONYのVME等、実際に携帯用のゲーム機に用いられる例も登場している。問題の記述はC言語で行い、自動的に構成情報の動的切り替えを制御することができるツールが開発されている（2.3項と2.4項の詳細は文献[6]を参照）。

2.5 組込みCPU＋プロセッサアレイ

動的リコンフィギャラブルプロセッサは、単一のコントローラにより、一定範囲のプロセッシングエレメントの構成と接続を、一斉に変更する。このため、プロセッシングエレメントのアレイ構造は、全体として並列演算用のデータパスを構成する。これに対して、それぞれのプロセッシングエレメントが個々に命令を保持、実行して自律的に動作可能なものをプロセッサアレイと呼ぶ。Quicksilver 社のACM[7]、PicoChip社のPC101、あるいはMITのRAW[8]等がこの例である。ACMでは、個々のプロセッサはDSP、汎用RISC、アプリケーション用に特化したプロセッサ等から構成され、全体としてヘテロジーニアスな構造となる。図5にACMの構成を示す。ストリーム処理におけるタスクを各ノードが分担して実行することを想定しており、各プロセッサはオンチップネットワークで接続され、ネットワークを介してデータストリームをやりとりするためのFIFOを装備する。データストリームの到着によってプロセッサの処理が起動される仕組みになっている。これに対して、PiCoChipやRAWでは、同一構造のプロセッシングエレメントのアレイを用いる。それぞれがSPMD（Single Program Multiple Datastreams）的に動作することも可能であり、いくつかのグループ単位にストリーム処理のタスクを割り付けることも可能である。これらのプロセッサアレイの多くでは、それぞれにプロセッサに対するプログラムを別々に記述する必要があり、並列ソフトウェアの記述とデバッグ環境は一般プログラマにとってやや困難を伴う。プロセッサアレイにおいては、OSやユーザインタフェースを処理するプロセッサを別に持たせる場合もあるが、ACM等ではアレイ中のプロセッサの1つがこれを受け持っている。このように、制御用CPUを中に取り込んでしまった場合、プロセッサアレイは、４節で紹介するマルチプロセッサと区別が付きにくくなる。

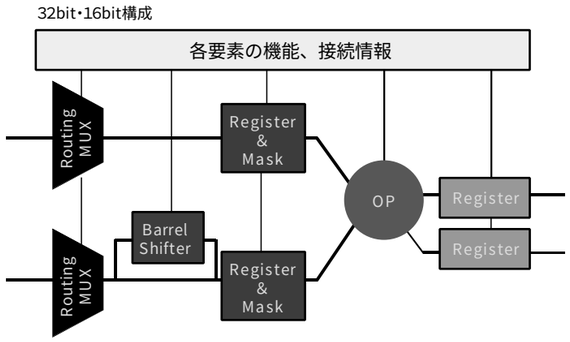


図4 動的リコンフィギャラブルプロセッサの構成要素の例 (Chameleon CS2112)

3. コンフィギャラブルプロセッサ、プラットフォーム設計

組込みCPU＋Xのアプローチは、それぞれ独立したモジュールとなり、バスを介してメモリ上でデータ転送を行う等、単純な組み合わせ方となる。これに対してあらかじめ元となるプロセッサの大枠を決めておき、目的に合った専用命令を定義する等の形で、専用ハードウェアを様々なレベルで組み合わせることを可能とするのがTensillica社のXtensa[9]や東芝MeP[10]等のコンフィギャラブルプロセッサのアプローチである。このアプローチでは、専用ハードウェアをCPUに組み合わせるというよりは、専用目的化したプロセッサ自体を生成するという考え方をとる。基盤となるアーキテクチャを決めておき、これらを様々な形で専用化するプラットフォームアーキテクチャもこれと似た考え方といえる。図6に東芝のMeP[10]を示す。MePでは、MePコアと呼ばれる拡張可能

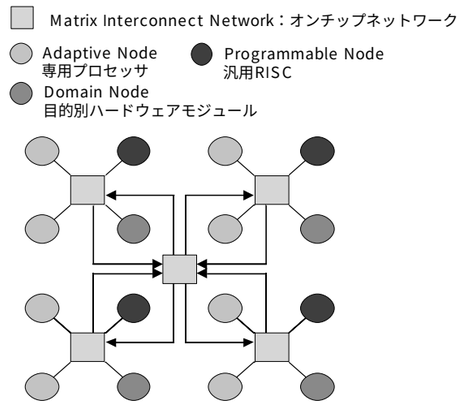


図5 Quicksilver ACMの構成

な32bitプロセッサコアをベースとし、専用ハードウェアやDSPや浮動小数演算器等の拡張部と組み合わせてMePモジュールを構成し、さらにこれらを複数接続してマルチプロセッサ構成をとるレベルまでを含む枠組みになっている。

4. マルチプロセッサ化

4.1 ヘテロジーニアスなマルチプロセッサ

CPU＋Xというアプローチで多用途のシステムを実現する場合、さまざまなXを組み合わせる必要が生じる。例えば、CPU＋DSP＋グラフィック用プロセッサ＋専用ハードウェア等、様々なプロセッサの複合体となる。このように、CPU＋Xのアプローチを押し進めていくと、ヘテロジーニアスなチップマルチプロセッサとなる。これらのシステムは、それぞれのプロセッサの役割が決まっており、特定のタスクのみを実行する。多くの場合、複数のプロセッサは、共有バスによって接続され、データの交換は、メモリを介してDMA転送等で行う。しかし、システムの規模が大きくなると、バスのボトルネックを避けるために、オンチップネットワークや、スイッチを用いる必要が生じ、これらに関する研究が盛んになっている。ヘテロジーニアスなシステムでは、プロセッサの役割は決まっており、プログラミングは、プロセッサ毎に専用の環境と言語で行い、全体の制御は組込みCPU上のOSによって行われる。しかし、様々な性質を持つプロセッサ間のデータ転送、同期等を完全に管理することが難しく、ソフトウェア開発は困難になりつつある。

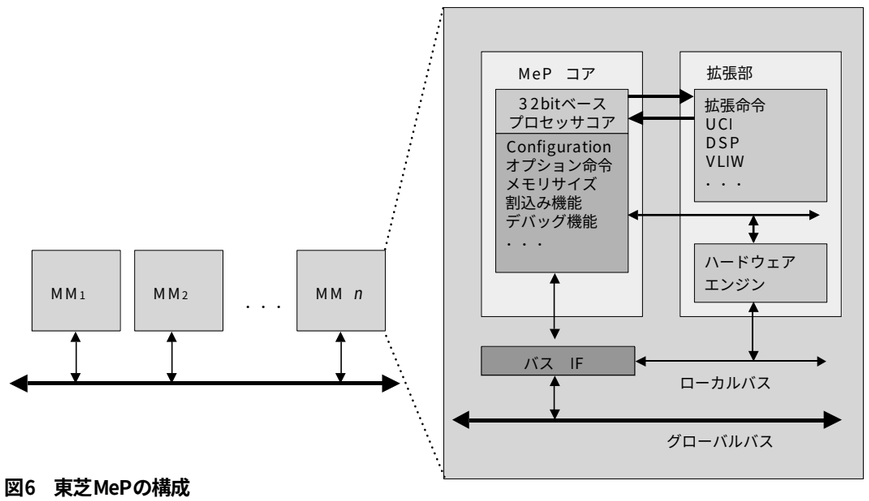


図6 東芝MePの構成

4.2 ホモジーニアスなチップマルチプロセッサ

ヘテロジーニアスなマルチプロセッサは、組込みCPU＋Xタイプのアーキテクチャの自然な発展形であり、コスト当りの目的別性能を最適化することができるが、システム全体を管理するのが極めて難しくなる。そこで、専用プロセッサによる目的別特化を行わず、強力な演算機能を持つ比較的少数のプロセッサによる並列処理により目的の性能を満足させる方法が考えられる。ここでは、この手法をチップマルチプロセッサと呼ぶ。チップマルチプロセッサは、ハイエンドのサーバ用に開発が進んでいる方法を組込み用にシフトしたものと考えられ、汎用システムで培った並列化技術、プログラム開発環境を利用することができる。

組込み用では、性能をさほど必要としない場合、周波数を下げると共に、それに対応して電圧を下げることで、消費電力を大幅に削減することができる。必要のないプロセッサの電源自体を切っておけば、スタティック電力の削減にもつながる。プロセッサアレイが簡単なプロセッサを多数用いて特定の処理の性能向上を狙うのに対して、少数の強力なプロセッサ上で、並列OSを走らせることができ、プログラムの自動並列化やタスクの割付、マルチジョブの制御をエレガントに行うことができる。単一のタスクを並列処理することも可能であるが、個々のプロセッサで別々のタスクを実行することも可能である。この方式はプロセッサが専用化されていない点、単一タスクの並列処理を行う場合、同期操作やデータ交換のロスが大きい点で、専用目的に対するコスト当たりの性能が低くなる。しかし、プログラム作成、デバッグ、管理が容易となる点でソフトウェアの開発コストを下げることができ、将来の発展が期待される[11]。

5. おわりに

本稿で紹介した様々な組込み用ターゲットアーキテクチャの位置づけを図7に示す。この図は、特定用途に限って高い性能を示すものを上方、広い用途に平均的な性能を示すものを下方に配置している。これらのアーキテクチャは互いに様々に関連している点に注意されたい。例えば、コンフィギャラブルプロセッサのアプローチに基づき、構成要素として動的リコンフィギャラブルシステムとDSPを取り込み、結果としてヘテロジーニアスなマルチプロセッサを生成するという場合もあるわけで、それぞれの要素技術が整然と分類されるわけではない。また、汎用システムと異なり、組込み用はターゲットによって様々な構成が最適となることから、紹介したターゲットアーキテクチャのどれかが全体の市場を制覇することは考えにくい。しかし、現在の組込み用ソフトウェア資産は、RISC型の組込みCPUをターゲットとして発展し、メモリ要求量等コストを最適化することに重点に置いているため、今回紹介したターゲットアーキテクチャ上のどれに対しても向いているとは言い難く、効率良く動作させることは難しい。

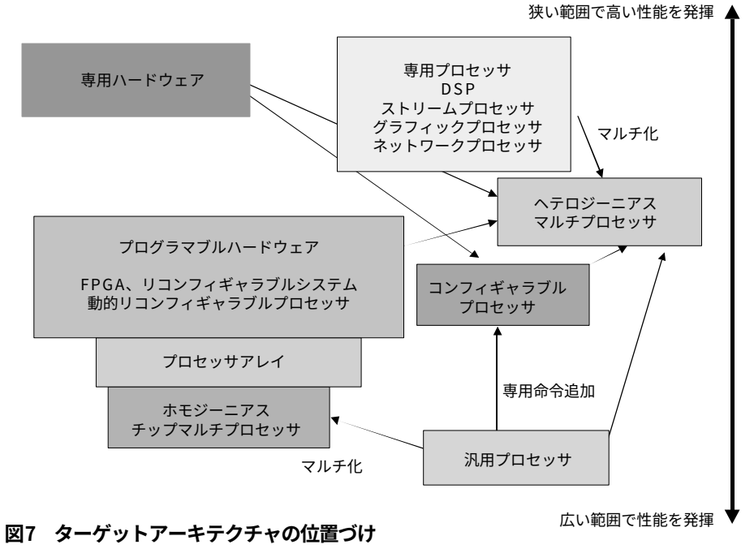


図7 ターゲットアーキテクチャの位置づけ

とくにヘテロジーニアスなマルチプロセッサ上でそれぞれのプロセッサのソフトウェアを別々に開発して、システム全体でバグなく動作させるのにはきわめて多くの人員と開発時間を必要とする可能性がある。このように現在の組込みターゲットアーキテクチャの乱立とヘテロジーニアスなマルチプロセッサの無秩序な発展は「第2のソフトウェア危機」を呼ぶ可能性がある。ホモジーニアスなチップマルチプロセッサ化により、汎用プロセッサで培った並列化技術、プログラム開発技術、OS技術を導入することは、解決の１つの手段ではあるが、専用プロセッサを利用するのに比べて本質的にコスト面のロスが大きいため、組込みシステム全体に広く普及するかは疑問である。専用プロセッサ、リコンフィギャラブルシステム、動的リコンフィギャラブルシステムを要素として構成したヘテロジーニアスな環境においても、汎用プロセッサで培ったOS技術、ソフトウェア開発技術を導入していくことが必要になろう。CAD技術、チップ開発技術に加えて、アーキテクチャ技術、システムソフトウェア技術の出番が来ているのだ。

参考文献

[1] D.D.Gajeski, Specification and Design of Embedded Systems, Prentice Hall
[2] http://www.ti.com/
[3] U.J.Kapasi, et.al.: Programmable Stream Processors, IEEE Computer, pp.54-62, Aug, 2003.
[4] http://www.nvidia.com/
[5] http://www.myri.com/
[6] 末吉, 天野: リコンフィギャラブルシステム, オーム社, 2005
[7] F.Furtek, E.Hogenauer, and J.Scheuermann: Interconnecting Heterogeneous Nodes in an Adaptive Computing Machine, in Proc. FPL, pp.125--134, 2004
[8] M.B.Taylor, et al.: The RAW Microprocessor: A Computational fabric for Software Circuits and General Purpose Programs, IEEE Micro, pp.25-35, March/April, 2002
[9] http://www.tensillica.co.jp/
[10] 宮森：コンフィギャラブルプロセッサMePとその開発事例, 信学報VLD2004-115, 2005
[11] M.Edahiro: Chip Multiprocessors for Embedded Systems, Cool Chips VII, tutorial, pp.97-120, 2004

情報処理学会 組込みシステム研究グループ

http://www.ertl.jp/SIGEMB/

名古屋大学 大学院情報科学研究科　教授　博士（理学）
高田　広章

組込みシステムおよび組込みソフトウェア技術の研究の活性化を目的として、2005年4月に、情報処理学会に組込みシステム研究グループを設立した。研究グループでは、単独での研究発表会の開催に加えて、組込みシステムに関連する既存のシンポジウム等に共催・協力していく計画である。2006年4月には、研究会の設立を目指しており、多くの方の参加をお願いしたい。

1　情報処理学会と研究グループの位置付け

情報処理学会は、情報技術分野における我が国の中心的な学会で、2万人強の会員が参加している。情報処理学会における調査研究活動は、研究テーマ毎に設けられた研究会と呼ばれる組織を中心に進められている。現在、3つの領域の元に、35の研究会が設けられており、研究会毎に、シンポジウムや定期的な研究発表会（こちらも研究会と呼ばれることが多いので、組織としての研究会と混同しないように注意）等が開催されている。

研究グループは、新しい研究分野となり得る萌芽的研究の促進等を目的とした組織で、研究会を設立する前段階に位置付けることができる。

組込みシステム研究グループは、本年の2月に筆者が発起人代表となって設立申請し、多くの方のご支援の後押しを得て、4月からの設立が認められた。なお、研究グループの活動期間は通常は2年であるが、2006年度からの研究会設立を申請することにしており、2005年度限りで設立している。

2　組込みシステム研究グループ設立の背景と目的

今さら繰り返すまでもなく、組込みシステム技術は、我が国が競争力を持つ産業の多くを支える重要な技術となっており、活発な研究・開発が望まれている。近年、産業界においては組込みシステム技術の重要性が認識され、企業における研究開発も活発化しつつある。ところが、大学や公的研究機関で組込みシステム技術に取り組んでいる研究者は、徐々に増えつつあるとはいえ、期待される程には増えていないと捉えている。

一方海外においては、ACM (1)（情報技術分野の国際的な学会）が組込みシステムに特化した論文誌であるTransactions on Embedded Computing Systemsを創刊したことに加えて、組込みシステムを扱う研究組織であるSpecial Interest Group on Embedded Systems（SIGBED）を設立する等、組込みシステム技術に対する取り組みを急速に進めている。組込みシステム技術は、我が国が比較的競争力があるといわれている分野であるが、このまま大学等での取組みが活発化しない状況が続くと、国際競争力低下が避けられないと危惧される。

組込みシステム研究グループは、このような問題に対して、学会の立場から取り組むことを趣旨としている。具体的には、組込みシステム技術に関する研究発表の場や技術交流の場を設けることで、大学等での研究の活性化を図ることができる。

それに加えて、産業界における要求事項や課題を大学等に知らせることや、大学等での研究成果を産業界に紹介することを通じて、産学連携の推進も図る。とくに組込みシステム分野においては、産業界における開発の現状や課題が大学等の研究者からは見えにくい状況にあり、研究活性化のために産学連携の推進が欠かせないと考えられる。

3　他の研究会との関係

情報処理学会においては、これまでも、ソフトウェア工学研究会（SE研究会）、システムソフトウェアとオペレーティングシステム研究会（OS研究会）、システムLSI設計技術研究会（SLDM研究会）等において、組込みシステムに関する研究発表が行なわれている。また、SE研究会は、組込みソフトウェアにテーマを絞った組込みソフトウェアシンポジウム（ESS）を毎年開催している。

しかし現状では、これらの研究会活動の間に連携がなく、数少ない研究者がそれぞれに分散している。組込みシステムにおいては、アプリケーション毎に要求に応じてコンピュータシステム全体が最適に設計されるために、システムを構成する様々な要素技術が密接に関係する傾向にある。そのために、異なる技術分野間の連携が強く望まれている。一例を挙げると、ハードウェア設計とソフトウェア設計が密接に関係しているために、ハードウェア／ソフトウェア協調設計技術が必要とされている。このような技術の研究を推進するには、ハードウェア設計技術を扱う研究コミュニティとソフトウェア設計技術を扱う研究コミュニティの連携が必要である。

組込みシステム研究グループは、それらの間の連携を図ることで、組込みシステム技術に関して幅広くかつ統合的に扱うことを目的としている。

(1) ACM：Association for Computer Machinery

4　2005年度の活動計画

組込みシステム研究グループでは、2006年度には研究会を設立することを前提に、2005年度はそのための準備活動と位置付けている（表1にイベントスケジュールを示す）。

組込みシステム分野の研究者がいくつかの研究コミュニティに分散している現状から、組込みシステム研究グループで新たなイベントを開催すると、研究者がさらに分散してしまうので、既存のイベントに共催・協賛することにより、研究コミュニティ間の連携と統合を図ることを基本方針としている。

研究グループ単独では、2005年度内に2回の研究発表会を開催する。初回の研究発表会は、発足記念のシンポジウムの位置付けとし、7月27日に東京で開催した。ここでは、SEC鶴保征城所長と北陸先端大学院大学片山卓也教授による招待講演等をお願いした。

また、2回目の研究発表会を1月頃に開催する計画である。その他に、3月に開催予定の組込み技術とネットワークに関するワークショップ（2003年度までは実時間処理に関するワークショップ）に共催を申し入れている。

表1　2005年度 組込みシステム研究グループ イベントスケジュール

2005年	
7月27日	研究発表会（主催,発足記念シンポジウムを兼ねる）
8月22日～24日	組込みシステム技術に関するサマースクール（共催）
8月25日～26日	組込みシステム技術に関するサマーワークショップ（共催）
8月24日～26日	DAシンポジウム（協賛）
10月17日～19日	組込みソフトウェアシンポジウム（協賛）
11月29日～30日	コンピュータシステムシンポジウム（協賛）
2006年	
1月頃	研究発表会（主催）
3月頃	研究発表会（共催予定）

また、8月に開催予定の第7回組込みシステム技術に関するサマーワークショップ（SWEST7）と第1回組込みシステム技術に関するサマーワークスクール（SSEST1）を共催する。SSEST1は、学生の企画・運営により、組込みシステム開発の基礎を幅広く学ぶためのものである。この2つのイベントについては、2006年度には主催をする方向で調整している。

シンポジウムについては、研究グループで主催するのではなく、情報処理学会主催で組込みシステム関連のテーマを取り上げている以下のシンポジウムに協賛する。

- DAシンポジウム（SLDM研究会主催）
- 組込みソフトウェアシンポジウム（SE研究会主催）
- コンピュータシステムシンポジウム（OS研究会主催）

2006年度には、この中の組込みソフトウェアシンポジウムを、SE研究会との共催とする方向で協議を開始した。

5　参加のお誘い

組込みシステム分野における我が国の競争力を高めるためには、大学等における組込みシステム研究の活性化が極めて重要であり、そのために学会が果たすべき役割は大きい。また、産学連携を促進する意味でも、組込みシステム研究グループの活動に対して、産業界からも積極的に参加いただけると幸いである。そのためにはまず、情報処理学会に入会いただき、2006年度に組込みシステム研究会が設立された暁には、研究会にも参加登録いただけると幸いである。また本活動の成果はSECを通じて発表することも計画している。ご期待いただきたい。

社団法人 日本情報システム・ユーザー協会 JUAS

http://www.juas.or.jp/

社団法人 日本情報システム・ユーザー協会　広報・調査担当
佐藤　亘

JUASはユーザの立場からの産業情報化の推進を目的とし、大手ユーザ企業を中心に、約250社の会員を擁し、経営とITに関する様々なテーマや、立場に応じた40以上の委員会、研究会、研究プロジェクトを実施し、毎年、各種調査・研究報告書の刊行や、提言を行っている。1962年に日本データ・プロセシング協会として創立、1992年には社団法人 日本情報システム・ユーザー協会として、全面的に拡充改組した。

1　JUASの活動

環境・ニーズの変化の中、企業におけるIT活用について、従来の思考方法で行うだけでは経営をサポートすることは困難であり、新しい視点での取り組みの必要性を多くの企業が感じている。

このような中、社団法人 日本情報システム・ユーザー協会（JUAS（ジュアス））は、日本唯一の中立的なITユーザ団体として、IT活用における様々な課題の解決に寄与するために積極的な活動を展開している。

会員企業を中心に実施している委員会・研究会・研究プロジェクト等の40を超える研究活動、その時々のタイムリーな内容を扱っている研修事業、平成13年より毎年開催している「ITガバナンス」をテーマとしたユーザカンファレンス等、活動は多岐にわたり、今後は「戦略・企画」に関する活動に注力していく方針である（「図1, 図2)。

ここでは、そのうちの「システム・リファレンス・マニュアル」「企業IT動向調査」「ユーザ向けソフトウェアメトリクス調査」について、簡単に紹介する

2　システム・リファレンス・マニュアル（SRM）の概要

企業において現実に活用されているシステムは広範囲、多様であるがゆえに、その活用コンセプトや定量データはまとめにくく、まず考え方の整理が必要となり、その上に目標とそれを元にした結果の実績を積み上げられなければならない。ソフトウェアの世界が「様々な要因から活用方針、標準値等がまとまらない」ものとして扱われている所以である。

このような現状を打開するため、JUASでは、ITユーザの立場から、IT化戦略立案・企画、システムの導入、運用、評価を行う各フェーズにおける広範な情報を整理し、担当者を支援するための、「システム・リファレンス・マニュアル（SRM)」の作成を進めている（2005年7月刊行予定)。

図1　JUAS活動コンセプト

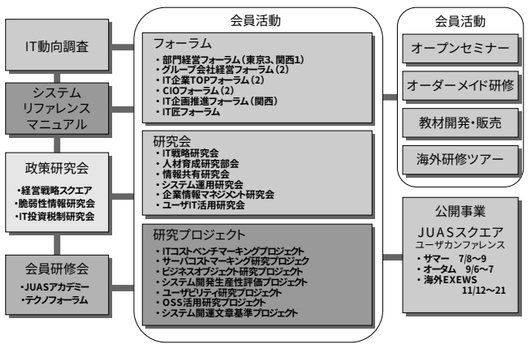


図2　JUAS 2005年度活動関係図

3　企業IT動向調査の概要

JUASはITユーザ企業のIT動向を把握するための「企業IT動向調査」を1994年より毎年実施しており、2004年度の調査が11年目にあたる。アンケートで1,000社近くのデータを収集すると共に、毎年40～50社へのインタビュー調査も実施している。またIT投資・IT利用・推進体制について経年的変化の分析を行うと共に、2004年度は重点的テーマとして「IT人材育成」「プロジェクトマネジメント」の２つを取り上げ、各種提言を取り入れた報告をまとめた。

本調査の報告は、毎年新聞・雑誌・TV等のメディアに取り上げられたり、様々な資料に引用されたりする等、各方面で中立、最新の貴重なデータとして、重要な役割を担っている（2005年4月報告書発刊)。

4　システム開発生産性評価プロジェクト、ユーザ向けソフトウェアメトリクス調査の概要

「システム構成の複雑化に加え、情報技術の多様化により、システム開発の生産性、品質の比較が難しくなってきている。一方、IT部門担当者へは、経営トップからは「当社の情報システム費用は適切か」との問いがなされている。ところが他社と比較し、適切さの判断ができるだけのデータや指標がほとんどないのが実情である。

このような背景を受けて、平成16年度の会員研究活動の１つとして、「システム開発の生産性評価指標の設定」「指標を設定する際の考慮すべき要素（業務内容、RFPの品質、役割分担、見積方式、開発方法等)」を検討する「システム開発生産性評価プロジェクト」がスタートした。

全10回の活動の中で、会員企業各社の生産性、品質を評価するデータを持ち寄り、また世の中で公表されているデータの検討を行った。

一方、SECの「定量データ分析部会」において、エンタプライズ系ソフトウェア開発力強化を目的とした定量データ収集が実施され、この中のユーザ側の情報を調査する「ユーザ企業向けソフトウェアメトリクス調査」をJUASが担当した。

また、JUASでは40社、計133プロジェクトの実績データを収集し、ソフトウェア開発生産性を計る上での、品質、工期、価格についてのユーザ企業の現状と標準値を調査し、企画、開発計画、進捗状況、開発結果それぞれにおける評価指標となる指標を検討した。そして、「システム開発生産性評価プロジェクト」と連携し、その知見を合わせて分析・検討することで、多くの興味深い報告を行った（2005年4月報告書発刊)。

5　SECとの連携

SECと連携するプロジェクトとしては、上記のシステム開発生産性評価プロジェクト、ユーザ向けソフトウェアメトリクス調査以外に、SECの「開発プロセス共有化部会」やJUASにおいてシステム開発における利用者、IT部門、ベンダの役割を検討した「ビジネスシステム定義研究プロジェクト（平成15年～16年度実施)」がある（図１)。

今後SECとの連携をますます強め、ユーザ、ベンダ間の認識のギャップを埋めることに寄与できれば幸いである。

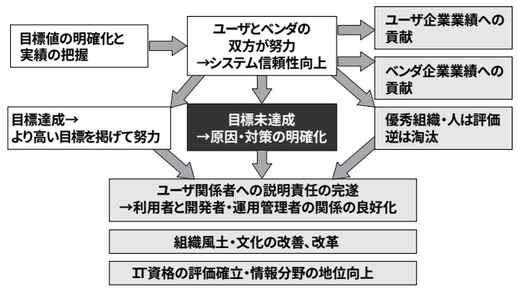


図3　ソフトウェア開発運用で指標を持つことの意義

名古屋大学 組込みソフトウェア技術者人材養成プログラム NEXCESS

http://www.nexcess.itc.nagoya-u.ac.jp/

名古屋大学　情報連携基盤センター　産学官連携研究員
山本　雅基

急速な組込みソフトウェア産業の拡大により、組込みソフトウェア技術者の人材養成の必要性が高まっている。こうした中、名古屋大学では、文部科学省の科学技術振興調整費における新興分野人材養成プログラムの一環として、平成16年度から「組込みソフトウェア技術者人材養成プログラム」を実施している。

1　人材養成の必要性

日本の「ものづくり産業」において重要な位置を占める自動車産業や情報家電産業をはじめとして、多くの産業分野における多数の製品に組込みシステムが使われるようになっており、組込みソフトウェア開発の需要は急増している。これに伴い、企業の開発現場においては、高度な技術を持った組込みソフトウェア技術者の絶対数が不足しているといわれている。

組込みソフトウェア技術者の人材養成では、一般的なソフトウェア技術教育に加え、リアルタイム性・高信頼性・ハードウェア操作等の専門性が高い技術教育カリキュラムが必要である。しかし、現在、大学における組込みソフトウェア技術に関する教育カリキュラムや教材が不足しており、十分な教育を受けないまま卒業し、企業で組込みソフトウェア開発の業務をしている実態がある。

一方で、企業においても、体系だった教育カリキュラムや教材が無く、無計画な現場主義が横行しており、企業内での教育をする体制や要員が不十分であるという意見も多数あり、大学等の教育機関に対する大きな期待がある。

2　NEXCESSの取り組み体制

このような社会の要求に応えるため、平成16年度から文部科学省／科学技術振興調整費により、組込みシステム技術の研究拠点である名古屋大学において社会人向けの組込み技術者人材養成を行う「組込みソフトウェア技術者人材養成プログラム」を開始している。

本プログラムの実行にあたり、名古屋大学情報連携基盤センターは名古屋大学大学院情報科学研究科の協力を得てNEXCESS（ネクセス）を設立した。その上で、社会人向け教育を効果的に実施するため、組込みソフトウェア開発企業において技術開発および教育経験が豊富な者を企業から雇用しプログラムマネージャに据えた。

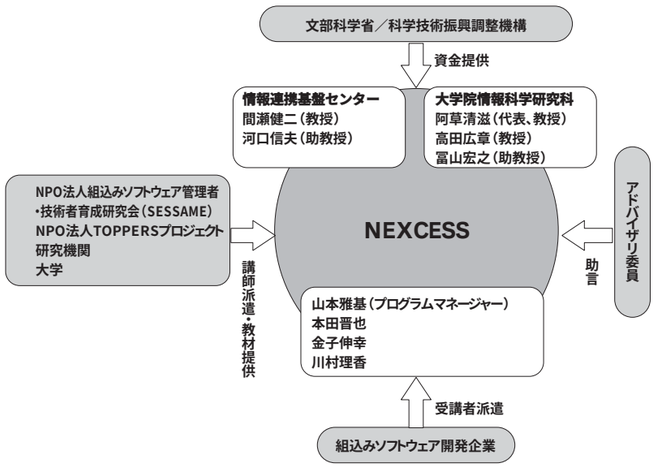


図1　NEXCESSの取り組み体制

NEXCESSでは、様々な組込み開発の現場知識を背景とした教育により教育効果を高めるため、教材および講師にはNPO法人組込みソフトウェア管理者・技術者育成研究会SESSAME（セサミ）とNPO法人TOPPERS（トッパーズ）プロジェクトの協力を得、また高度な専門教育を行うため、他大学や企業の研究者からも協力を得る仕組みを構築している。加えて、社会人向けの人材養成の実施の的確さを企業の立場から評価しフィードバックをかけるため、企業において組込みソフトウェア開発管理や人材養成等を行っている方々にアドバイザリ委員になって頂き、適切な助言を得る仕組みとしている（図１）。

以上のように、NEXCESSは効果的な人材養成を実施する体制を産学官の連携により整えた。

3　NEXCESSの活動

本プログラムは、平成16年度から平成20年度までの5年間実施され、のべ計590名の人材養成を行い、講義素材からコースウェアの製作・配布を目的としている。また、プログラムの終了後、開発したコースウェアを引き続き用いて、大学での技術者養成講座の開講や企業での社内教育での利用促進ができることを目指している。

また、社会人教育の特徴として、会社におけるキャリアパスを意識した教育コースの設計を行う。企業の技術者を初級・中級・上級の３クラスに分類している（表１）。多くの企業においては、上級クラスには専門職と管理職の2つのキャリアパスが用意されており、専門職の内容等に応じ、より細分化したキャリアパスがある場合もある。NEXCESSでは、初級･中級･上級（専門職・管理職）の各クラスを設け、それらのクラスの技術者を教育する教育コースを用意する。

なお、中級と上級においては、選択可能な複数の教育コースを用意し、それぞれのキャリアにおける多様化するニーズに対応する。また、各コースは、社会人の受講が容易になるように2日から4日程度の短期集中コースを基本とする。

表１　会社におけるキャリアパスを意識したクラス分けとクラスの能力

クラス	経験年数の目安	クラスの能力
初級	入社5年目	上司の指示に従い決められた業務を遂行する
中級	入社10年目まで	自ら業務を遂行し業務範囲内をリードする
上級	入社10年以上	会社をリードする

写真１　講義の様子

平成16年度に初級・中級・上級の各クラスにおいて合計9つの教育コースを開講した。369名の受講申し込みがあり、227名を選考し講義を開催し、214名が各コースの全日程を修了した。受講者に対して帰社後もWebでの入力可能なアンケートを実施し、多くの意見を得ることができた。受講生の総合評価では、5点満点における4.4点であり、教育コースとしては高い評価を得ている[1]。

平成17年度は、指導者養成コースを新設すると共に、受講者を会員としたNEXCESS倶楽部を設立し受講知識を現場における発揮能力に転化する等の継続的な取り組みを行い、社会人教育の拡充に取り組む。また、受講者上司へのアンケート等を通じて、管理者層からの人材養成に対する要求を掘り起こす。これらの取り組みにより、社会人の人材養成として真に有効なコース構築を行っていく。

4　SECとの連携

SECは、組込みソフトウェア開発力強化推進委員会の教育部会において教育カリキュラムの具体化を進めている。NEXCESSは教育部会に委員を出し、NEXCESSの活動を教育カリキュラム作成へ生かすと共に、教育部会における検討内容をNEXCESSの活動へ反映させる。今後とも相互に協力し、組込みソフトウェア人材養成をより高い次元へ引き上げ、わが国のソフトウェア産業の発展に尽していく。

参考文献

[1] 名古屋大学組込みソフトウェア技術者人材育成プログラム平成16年度活動報告書，2004

北陸先端科学技術大学院大学との共同研究「形式的手法の組込みソフトウェア開発への適用性の検討」

北陸先端科学技術大学院大学　情報科学研究科
教授　工学博士
片山　卓也
北陸先端科学技術大学院大学　情報科学研究科
産学連携客員教授　博士(情報科学)
岸　知二
北陸先端科学技術大学院大学　情報科学研究科
助手　博士(情報科学)
青木　利晃

形式的手法は数学的、論理学的基礎を用いて、ソフトウェアを厳密に記述し、その正しさを保証する手法であり、大規模なシステムや高い安全性の求められるシステム等での適用事例が報告されてきた。近年ツールや手法が整備され、一般のソフトウェア開発への適用検討が急速に加速しており、組込みソフトウェア開発への適用も期待されている。SECと北陸先端科学技術大学院大学（JAIST）では、組込みソフトウェア開発への形式的手法の適用性に関する共同研究をスタートさせた。本稿ではその概要について紹介する。

1　はじめに

組込みシステムとは、機器を制御するために組み込まれた計算機システムのことである。従来、組込みシステムは、一部の家電製品に組み込まれる小規模のシステムであった。現在は、プラント制御システム、自動車エンジン・機器制御、携帯電話、オーディオ機器等極めて多種多様な製品や機器に組み込まれ、それらの価値を決める主要な要素になっている。その一方で、ニュースや新聞で、組込みシステムのバグや、それにより引き起こされた混乱について聞くことが多くなってきており、組込みシステムの信頼性確保が急務となっている。組込みシステムの検証には、現在は、テスト中心の手法がとられているが、組込みシステムの複雑化と大規模化に対応して、様々な検証手法が提案されている。その中でも、形式的手法・検証と呼ばれる手法が注目を集めつつある。

2　形式的手法とは

形式的手法・検証とは、数学的・論理的基礎を用いてソフトウェアを厳密に記述し、その正しさを保証する手法のことである。その研究の歴史は比較的長い。1960年代ごろからプログラムやフローチャートの検証に関する研究が盛んに行われるようになり、Floyd法やホーア論理といった検証手法が提案された。その後、プログラムだけではなく仕様や設計を厳密に記述するZやVDMといった形式的仕様記述言語、および、その検証法が提案されてきた。また、それらを実際のシステム開発に応用する試みもされてきた。有名な事例としては、1980年代、Oxford大学とIBM Hursley研究所が共同で顧客情報管理システムの一部をZで記述したものがある。これにより、9%の開発コストが削減され、さらに、製品の品質を飛躍的に向上されることに成功したという報告がある。さらに、1990年代前半には、当時のCalifornia大学IrvineのSafety-Critical Systems Research Groupが、航空機衝突回避システムTCAS（Traffic Collision Avoidance System）Ⅱの形式的仕様を作成し検証を行った。当初このシステムの仕様は産業界主導により自然言語で記述されていた。しかしこのグループの手法が認められ、それにとってかわり正式な仕様となった。その他にも、パリの地下鉄列車制御システム、Darlington原子力発電所緊急炉停止システム、IEEE Futurebus+仕様等、数多くの事例が見受けられる[1][2]。

現在では、ツールや理論が整備され、より身近な技術となり、適用事例の数も増えてきた。ソフトウェアの検証のための手法は、大きく分けて演繹手法とモデル検査手法がある（表１）。これらの手法やツールは、多くの研究機関が長年研究を行い提案された最先端技術であり、組込みシステムの信頼性を飛躍的に改善する潜在能力を持っており、近年の組込みシステムの社会的重要性を考えると、この分野への適用が大いに期待されるところである。

一方で、先に挙げた事例は、先端的開発事例であり、大学や政府の協力や支援が大きく関わっている。開発期間、必要とするスキル等を考えると、原子力発電所と同様の手法をDVDプレーヤの開発に用いることはできない。さらに、要求される信頼性、規模、システムの特性も異なる。このような最先端の科学技術が、現在の民需の組込みシステム開発の問題をどこまで解決できるのか、その可能性や適用性を探り、その利用方法を明らかにすることが重要である。

表１　ソフトウェアの検証のための手法

手　法	特　徴	代表的な支援ツール
演繹手法	定理証明システムを用いて、対話的に性質の正しさを証明する。この手法では、記述能力が高く様々な性質を取り扱うことができる反面、検証コストが高いという問題点がある。	PVS、HOL、Isabelle等
モデル検査手法	記述できる内容は有限状態で特徴づけられるものに限定されるが、デッドロックや飢餓状態などの望ましくない状況を自動的に検出できる。また、検出された場合、そのような状況になる振る舞い方（反例と呼ぶ）を出力する。	Spin、NuSMV、LTSA

3　産業界への技術展開に向けて

SECと北陸先端科学技術大学院大学では、組込みソフトウェア開発に対する形式的手法の適用に関して共同研究をスタートさせた。本共同研究においては、以下の活動を進める予定である。

(1) 形式的手法の理解のためのサンプルの作成

形式的手法の適用性を検討するためには、形式的手法の専門家が集まるだけでは不十分であり、組込みソフトウェア開発の問題を理解している人との議論が不可欠である。そのためには、わかりやすいシンプルな組込みソフトウェアを題材とし、典型的な形式的手法をそれに対して適用したサンプルを作成して、議論の具体的なよりどころとすることが有益であると考えている。本共同研究ではそうしたサンプルを作成しながら、組込みソフトウェア技術者と形式的手法の研究者との議論の共通基盤の構築を行う。

(2) 形式的手法の議論を行うための適用マップの作成

前述した技術的な検討と同時に、組込みソフトウェアを扱う産業界の問題、ニーズ、開発の実態を踏まえたときに、形式的手法をどのような分野や対象に対してどのように適用することが効果的なのかを示す適用マップを作ることが重要であると考える。これは個別の技術の詳細や適用のテクニックを議論するためのものではなく、問題や技術のカテゴリや関係を示し、全体像を俯瞰するために使うことを意図している。

こうした適用マップを踏まえて議論や検討を進めることによって、アドホックな技術検討を行うのではなく、産業界全体にとって形式的手法をどう使うことが望ましいのかを意識しながら検討を進められるとともに、マネジメントが形式的手法を理解し、その導入を検討するための一助となることを期待している。

(3) 形式的手法の有効性検討のための事例研究の実施

新しい技術を導入すべきかどうかを考える際に、最も参考になる情報の１つは適切な実例であろう。能書きばかりをいくら並べられても、あるいは全く異なった分野の事例を見せられても、本当の意味の参考にはならない。そうした観点から、現実規模の問題に対する事例研究を実施し、その成果を公開することが有益であると考えている。もちろん組込みソフトウェアは多様であり、すべての人が納得する適切な事例は存在しないだろう。しかしながら、前項で紹介した適用マップを踏まえたトップダウンな適用性検討を行い、適用効果が期待される典型的な対象領域や適用技術を見極めながら、産業界への貢献が大きな対象を選定することを考えている。

こうした産業界にとってリアルで身近な事例を構築することは、SECとの共同研究ならではのテーマの１つと考えている。なお本テーマは先の（１）や（２）よりもやや時期を遅らせて開始する予定である。

4　まとめ

形式的手法が現実規模の問題に本当に有効なのかどうかを疑問視する人は多い。また極めて使いこなしの難しい技術であり、民需開発には不向きな技術ではないかと危惧している人もいる。もちろん形式的手法は万能薬ではなく、またエディタやデバッガのようにすべての技術者が使いこなせなければならない技術でもないだろう。それだけにこの強力な技術を、どういう対象に、誰が、どのように使っていくかを検討することは重要であり、形式的手法の応用研究の世界的状況から考えて、日本の組込み業界にとっても、今着手すべき重要な問題である。それに対して貢献ができるよう、鋭意共同研究を進めていく所存である。

参考文献

[1] E. M. Clarke and J. M. Wing: Formal Methods： State of the Art and Future Directions, ACM Computing Surveys, Vol. 28, No. 4, December 1996
[2] D. Craigen, G. Gerhart and T. Ralston：An International Survey of Industrial Applications of Formal Methods Volume 2 Case Studies, 1993

ドイツ フラウンホーファ協会IESEとの共同研究「見積り手法のフィージビリティに関する研究」

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サブリーダ
石谷　靖

SECは昨年11月からIESE(1)と先進的な見積り手法について日本国内での適用可能性を実証することを目的に共同研究を実施している。具体的には、IESEで開発されたCoBRA法とOSR法について、取り上げている。現場でエンジニアリングの実践を促進するためには、単に概念・考え方だけではなく、具体的な方法・手段をソフトウェア開発の現場に提供していくべきと考えるSECの活動の一環である。ここでは、共同研究の概要と各手法について紹介を行う。

1　共同研究の目的

IESEについてはSEC Journal等で何度も紹介してきたが[1]、定量的に実証する文化が深く根付いた組織である。IESEとの共同研究は、その知識と経験を活用して、具体的な国内の課題について定量的な解決方法の実証を行い、企業への具体的な処方の提供を目的とする。

現在、見積り手法の国内企業での適用・実証を現在の共同研究テーマとしている。背景には、見積り活動がプロジェクトの最初の要（かなめ）でありながら、検討されないこと、また一方でデータに基づいた「エンジニアリング的な」方法が最も適用可能な点がある。

2　共同研究の概要

IESEで開発されたCoBRA法[2]とOSR法[3]と呼ばれる手法を対象として取り上げてSEC研究員数名への技術トランスファーを行っている。

CoBRA法の特徴は、経験豊富なPMの知識を使って見積りモデルを作るところにある。経験豊富な人間の直感が正確であるとの経験則に基づく。OSR法の特徴は、大量のプロジェクトデータからデータマイニング手法を用いて見積りモデルを作成するものである。

昨今、ソフトウェア開発組織では、プロジェクトデータが集められていない例が多く、また、データを集めている場合でも有効な活用で苦労している例も多い。CoBRA法は前者に、OSR法は後者に解決策を与えると期待され、両者で広範囲の企業の要望をカバーできると考えている。

共同研究は、大きく２つの段階に分けられる。最初の段階で国内企業に手法を適用して見積りモデルを作成し、次の段階でそのモデルをリファインする。これは、国内の開発状況や文化に手法がどの程度マッチするかを探るのと併せて、見積りモデルが、一度作るだけでは十分ではなく、精度向上には常にモデル改善のプロセスを踏む必要があることを踏まえている[4]。

3　見積り手法の概要

(1) CoBRA法

例えば工数（Effort）を見積もる場合、CoBRA法は、

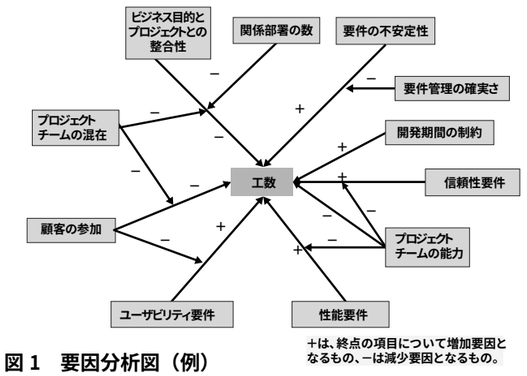


図１　要因分析図（例）

「理想」のプロジェクトでは工数は規模（Size）に正比例するが、「現実」ではさまざまな工数増加要因により、「理想」に比べ大幅な工数増加（オーバーヘッド: CO）が発生すると考える。式では次のとおりである。

$$Effort = \alpha \times Size \times (1 + CO)$$

CoBRA法の特徴は、経験豊富なプロジェクトマネージャなどの専門家の知識を利用し、COの部分を分析・定量化することにある。

最初に工数増加の要因となるものをブレーンストーミング形式で決定する。複数名の専門家が集まり、何が重要な工数要因になるかを議論して決定する（図1）。通常、10～20程度の要因が決定される。

続いて、各要因がどの程度の影響を及ぼすのかを、専門家1人ひとりに、通常考えられる影響度合いと、最善と最悪の３つの場合をインタビューにより確認する。３点と幅を持たせて聞くのは直感や結果自体にぶれがあることを反映している。例えば工数増加要因として「要求の安定性」があったとき、専門家は、不安定な場合の工数増加を通常30％増し、最善20％増し、最悪では70％増し等と回答する（図 2）。複数の専門家（10名程度）のおのおのに要因ごとの影響度を確認し、三角分布を得て、続いてモンテカルロ法により、工数の増加分の確率分布を求める（図 3）。この方法により見積り工数とリスクを同時に評価できることがわかる。

(2) OSR法

OSR法はデータマイニング法の一種で、過去のプロジェクトデータの中から予測精度が高くなる最適なデータセットを自動的に選択する手法である。条件の組み合わせを増やしデータを絞り込むが、条件ごとの絞り込み前後のデータセット間の見積り精度に関する統計的な差を比較しながら、最適な組み合わせを探索するところに特徴がある（図 4）。

4　結果と今後の展開

CoBRA法は１社で試行している。10名の専門家と２回ほど半日使って要因抽出とその関係を検討し、要因の影響の定量化は各専門家ごとに１時間程度かけた。最初の結果は平均見積誤差が30％であり、準備の容易さ等も考え合わせ、比較的良好な結果と評価している。現在第２段階として要因モデルの見直しを終え、これからさらに精度向上を目指し改善モデルを作成・評価する。

OSR法は、１回目の試行で２社のモデル作成を行い、結果は平均見積誤差が40％弱～70％弱であった。まだ改善の余地が大きいと考えており、データ分類の整理や、専門家の知識などを利用したデータ項目の絞り込みで現在精度向上方法を探っている。これらの結果は、すべての試行を終えた後、共同研究成果として発表する。

なお、今回をエンジニアリングアプローチの有効性を確認する最初の例として、今後他のテーマでもSECはIESEと共同研究を進める予定である。

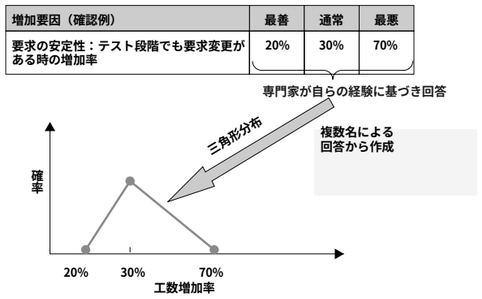


図２　要因の影響度の確認例

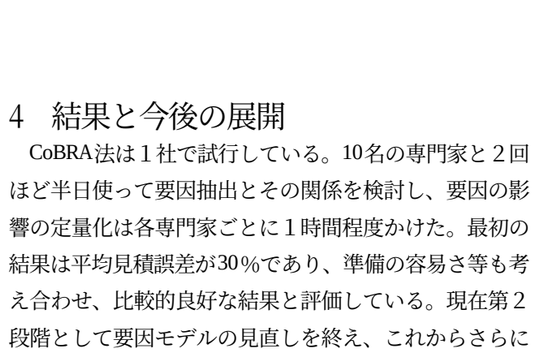


図３　見積工数の分布と見積シナリオ

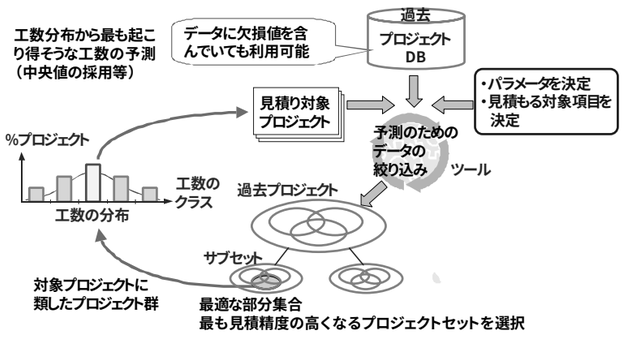


図４　OSR法による見積り手順

参考文献
[1] 石谷 靖：ドイツフラウンホーファIESE，SEC Journal No.1，PP38-39，2005
[2] Lionel C.Briand，Victor R.Basili，et.al: Pattern Recognition Approach for Software Engineering Data，IEEE trans. on SE Vol18，No.11
[3] Lionel C Briand，et.al: COBRA: A Hybrid Method for Cost Estimation，Benchmarking，and Risk Assessment，ISERN-97-24
[4] 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：SEC BOOKS ITユーザとベンダのための定量的見積りの勧め，オーム社，2005

(1) Institute fur Experimentelles Software Engineering：実験的ソフトウェア工学研究所。http://www.iese.fhg.de/

BOOK REVIEW

ソフトウェア工学・システム工学ハンドブック–エンピリカルアプローチによる法則とその理論

Albert Endres, Dieter Rombach共著　吉舗紀子訳　EASEプロジェクト監修

ISBN：4-87566-314-5　コンピュータ・エージ社刊
A5判・472頁・定価5,250円（税込）　2005年8月刊

素直に読み、そして考えたい必読書

本書は実験的・実証的（エンピリカル）ソフトウェア工学に関する知見を集大成したものである。過去30年以上に亘って発表された著名な論文や刊行物をベースとし、特定の理論や方法論ではなく、またソフトウェア工学に関する体系的な教科書でもないが、ソフトウェア産業に身をおく者にとって必読の書となっている。

「遅れたプロジェクトにマンパワーを追加してもさらに遅らせるだけだ」（Brooks）、「エラーは要件および設計作業中に最もよく起き、あとでそのエラーを取り除くのは最も費用がかかる」（Boehm）等、よく知られた知見を含む50の法則、25の仮説、12の推測がリストアップされ、適用可能性、根拠および理論が整然と述べられている。

長年に亘るソフトウェア工学分野の赫々たる成果が、極めてわかり易く凝縮されている。

知見の多くは大学の研究者によって創出されたものと思われるが、産学連携がベースになっていることは明白で、その意味で本書は産学連携の有り方にも大きな示唆を与える。

翻訳家吉舗紀子氏の訳を、奈良先端科学技術大学院大学の松本健一教授らのEASEプロジェクトで監修し、読みやすいものとしている。

（神谷　芳樹）

開発者列伝

日経エレクトロニクス編

ISBN：4-8222-0235-6　日経BP社刊
四六判・292頁・定価1,470円（税込）　2001年4月刊

父さん、すごいの作ったね

ヒット商品となった組込み製品の開発ストーリを紹介した書籍。「数十人のソフトウェア技術者が昼夜関係なく開発にあたった」、「品質保証部の検証で600個の不具合が発生」、「デグレード、またやっちゃいました」等々、身に覚えのある現場が生々しく描写されている。

ヒット商品は製品企画、マーケティング等によるものが大きいが、本書からは技術者達の英知と努力が大きく関与していることが読み取れる。例えば『AIBO』では成長をも表現する動き（モーション）の実現。『T（タウ）』ではEPG（電子番組ガイド）やリアルな画像を実現する画像処理等、付加価値機能の実装へのこだわり。『FinePix』のソフトウェア処理によって部品コストや実装面積を抑える戦略。ソフトウェア技術者の英知と努力が、魅力的な製品へと繋がることを実感できる。

本書は5年以上前の現場が著されているが、どの開発現場でも昔から非常に忙しいことがわかる。SECの調査でも、日本の技術者の労働時間は、欧米諸国よりも長いことがわかっている。長時間労働が魅力的な製品づくりを支えているとすると少し寂しい気がしてしまう。

組込みソフトウェア開発の面白さ、それは動きや音、表示を伴うリアルな製品として市場に提供されることかもしれない。「父さんは、またすごいの作ってるんだぞ‼」と会話しながら、19時頃には家族と食卓を囲みたいと願う。

（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	URL
7月	27日（水）	情報処理学会　組込みシステム研究グループ設立記念シンポジウム（第1回研究会）	情報処理学会	慶應大学 三田キャンパス 北館ホール	http://www.ertl.jp/SIGEMB/
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	静岡県浜松市・遠鉄ホテルエンパイヤ	http://www.ertl.jp/SWEST/
9月	6日（火）	組込みプロジェクト・マネジメント・フォーラム2005	翔泳社	東京都中央区・秋葉原コンベンションホール	http://www.shoeisha.co.jp/
	7日（水）～9日（金）	FIT2005 第4回情報科学技術フォーラム	社団法人 情報処理学会	東京都文京区・中央大学後楽園キャンパス	http://www.jasa.or.jp/
	8日(木)～9日(金)	第24回ソフトウェア品質シンポジウム	財団法人 日本科学技術連盟	東京都新宿区・新宿NSビル	http://www.juse.or.jp/
	14日（水）	情報処理学会連続セミナ2005（第3回）「組み込み用LSI」	社団法人 情報処理学会	東京都千代田区・東京電機大学神田キャンパス7号館1F 丹羽ホール	http://www.ipsj.or.jp/
10月	3日（月）	情報化月間	経済産業省	東京都港区・全日空ホテル	http://www.ipa.go.jp/
	7日（金）	JASA組込みソフトウェアフォーラム in大阪	社団法人 日本システムハウス協会（JASA）	大阪府大阪市・マイドームおおさか	http://www.jasa.or.jp/
	7日（金）	情報処理学会連続セミナ2005（第4回）「組み込みソフト開発手法・検証・ツール」	社団法人 情報処理学会	東京電機大学 神田キャンパス7号館1F 丹羽ホール	http://www.ipsj.or.jp/
	12日（水）～14日（金）	SEPG japan 2005	日本SPIコンソーシアム(JASPIC)	東京都港区・東京コンファレンスセンター品川	http://www.jaspic.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	東京都港区・日本科学未来館	http://www.ipsj.or.jp/
	21日（金）～22日（土）	ITC Conference	特定非営利活動法人 ITコーディネータ協会（ITCA）	東京都大田区・大田区産業プラザ	http://www.itc.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（仮称）	IPA/SEC	東京都千代田区・経団連会館11階　国際会議場	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）（協賛予定：IPA）	神奈川県横浜市・パシフィコ横浜	http://www.jasa.or.jp/
	16日（水）	ETソフトウェアロボットコンテスト・チャンピオンシップ	社団法人 日本システムハウス協会（JASA）（特別協力：IPA/SEC、SESSAME）	神奈川県横浜市・パシフィコ横浜	http://www.etrobo.jp/
	25日（金）	情報処理学会連続セミナ2005（第5回）「組み込みソフト開発事例（組込みOS系）」	社団法人 情報処理学会	東京都千代田区・東京電機大学神田キャンパス7号館1F 丹羽ホール	http://www.ipsj.or.jp/
	28日（月）	情報処理学会連続セミナ2005（第6回）「組み込みソフト開発事例（ユビキタス系）」	社団法人 情報処理学会	東京都千代田区・東京電機大学神田キャンパス7号館1F 丹羽ホール	http://www.ipsj.or.jp/
2006年2月	9日（木）～10日（金）	デベロッパーズサミット2006	翔泳社	東京都目黒区・目黒雅叙園	http://www.seshop.com/event/dev/
3月	7日（火）	日本のコンピュータ生誕50周年記念シンポジウム	情報処理学会	東京都新宿区・工学院大学　新宿キャンパス	http://www.ipsj.or.jp/
	7日（火）～10日（金）	第68回全国大会（学会創立45周年記念大会）	情報処理学会	東京都新宿区・工学院大学　新宿キャンパス	http://www.ipsj.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。

編集後記

IPAX 2005（IPA主催イベント5月18日～20日）、SEC Forum 2005（SEC主催イベント6月20日～21日）の2大イベントが、SEC journal 3号の編集制作期間中に開催されました。大きなイベントの準備に追われ、定期刊行物のハンドリングの難しさを痛感いたしました。今回も無事に発行することができ、ご執筆をいただいた方々にはこの場をお借りして感謝申し上げます。また、「SEC Forum 2005」は、2004年10月にSECが設立されてから初めての“ エンタプライズ系プロジェクト ”（6月20日）と“ 組込み系プロジェクト ”（6月21日）の合同の報告会であり、無事に開催できましたことをこの場でご報告いたします。しかし「SEC Forum 2005」への参加お申し込みにつきましては、なんと、Webでの申し込み受付開始後2日間でエンタプライズ/組込みともに450席が満席となり、多くの方々からキャンセル待ちのご連絡を頂く状況を招いてしまいましたことにつきまして、この場をお借りしてお詫びを申し上げます。SECの成果報告会では本当に450名の席が埋まるかどうか、非常に不安な状況での開催でしたがソフトウェア開発に係わる方々の「ソフトウェア・エンジニアリング」に関しての意識の高さに驚かされ、SECの成果を今以上に発表する機会の必要性の高さを実感いたしました。次回成果発表会は、「SECソフトウェア・エンジニアリング・コンファレンス（仮称）」（10月24日予定）を計画し、ここでは先般募集をさせていただいたソフトウェア・エンジニアリングに関する論文の優秀賞発表と最優秀賞審査を行います。開発者の方々からは、論文というと別世界のように感じる方も多いかと思いますが、現場で役に立つことが裏づけされた手法等を中心とした発表となる予定です。ソフトウェア・エンジニアリング活動を行っている開発者の皆様には是非ご来場頂き、ご活動の役に立つ手法の発掘をして頂けるとSEC冥利につきます。SEC journalに掲載しております論文を、今後投稿論文から選出し掲載を予定しております。論文投稿受付の準備を始めましたので、今後SECのWebにご注目ください。

本Journalに対してのご意見もお待ちしております。＜ご意見用メールアドレス：sec-journal@ipa.go.jp＞　（ヒゲ）

SEC journal　第1巻第3号（通巻3号）　2005年8月5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

お知らせ

SEC journal 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

論文の評価基準

a. 実用性（実フィールドでの実用性）
b. 可読性（記述の読みやすさ）
c. 有効性（適用した際の効果）
d. 信頼性（実データに基づく評価・考察の適切さ）
e. 利用性（適用技術が一般化されており参考になるか）
f. 募集テーマとの関係

応募要項

スケジュール

A募集　2005年10月末必着
B募集　2006年4月末必着
両募集とも、採録の場合にはSEC journalへの掲載およびIPA SECのWebやイベント等での発表を行います。

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
eメール：sec-journal@ipa.go.jp

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

応募様式は、下記のURLに2005年8月下旬に公開予定です。

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

SEC journal バックナンバーのご案内

http://www.ipa.go.jp/software/sec/journal.phpより
ご注文いただけます