2007年9月28日発行
第3巻第3号（通巻11号）
ISSN 1349-8622

SEC journal

Software Engineering Center

11

メモリリーク検出システムλtrace

青島 武伸,伊藤 智祥,春名 修介

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.11目次

巻　頭　言

電子情報産業における ソフトウェアエンジニアリングへの期待

社団法人 電子情報技術産業協会
常務理事
長谷川　英一

ソフトウェアは、コンピュータのみならず、家電、通信、製造装置、自動車等様々な分野においてわが国の得意とする製品の高機能化・高付加価値化を実現する手段として重要な役割を担うようになってきている。我々社団法人 電子情報技術産業協会（以下JEITA）としても、産業競争力の強化等のために、ソフトウェア事業の活性化のための人材育成、税制改革、インフラ整備、政府調達等に関する事柄についていくつもの要望を政府に上げている。

JEITAの統計によれば、2006年度におけるソフトウェア・ソリューションサービスの国内市場規模は5.4兆円に達し、コンピュータ本体の国内市場規模である2.4兆円の２倍を越えており、この事からもソフトウェア産業の重要性がうかがえる。しかしながら、ソフトウェア産業のさらなる発展にはまだ多くの解決すべき課題がある。

人材育成

産業競争力向上の重要なファクターとなるものに人材がある。即戦力となる人材育成のための産学協同の活動として、JEITAでは、2002年度よりJEITA講座（IT最前線）を開催し、東京大学、東京農工大学、電気通信大学、慶応義塾大学、立命館大学等と連携、企業の第一線の研究者、スペシャリストを派遣し、若手人材の育成に力を注いでいる。IPA/SECにおいても情報系人材の育成は大きなテーマとなっており、情報交換等の協力関係に期待をしたい。

海外研究機関との連携

ソフトウェアエンジニアリング分野については、IPA/SECにおかれても共同研究体制にあるドイツFraunhofer IESE 研究所（実験的ソフトウェア工学研究所）との交流を平成18年より持ち、各々の課題を確認した。また家電等で重要性を日々増している組込みソフトウェアについては、先進開発事例等の収集、及びそれらをベースとした改善への方向性に関して、IESEやIPA/SECと継続的な研究・検討を行い、活動成果を協会内外へ発信していきたい。

ITセキュリティセンター（ITSC）

JEITAでは、平成13年７月にITSCを設置し、ISO/IEC15408の評価機関として活動しているが、今後急増が予想される評価案件に対応すべく、体制強化を図っていきたい。

実務を支えるソフトウェア技術への取り組み

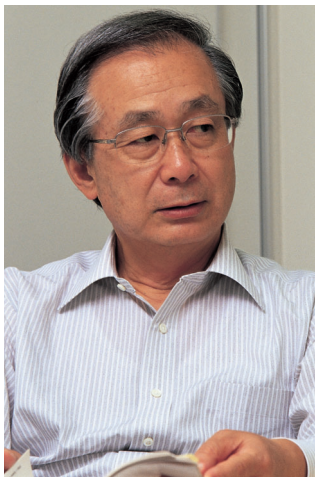
技術面での活動としては、実務を真に支えることのできる先進的なソフトウェア技術とは何かという問題意識をもとに、最新技術の調査、研究を行っている。一例としては、「問題解決手法に則った要求定義フェーズにおける取り組み方」、「要求工学の現場での取り組み」があげられる。

IPA/SECは、ソフトウェアエンジニアリングについて産学官の連携のもと、様々な研究を実施しており、JEITAのテーマとも深く関わりがある。また経済産業省及びIPA/SECにおいては10年ぶりに共通フレームを改定し、共通フレーム2007を策定したことは、我が国のベンダーとユーザー共通のものさしとして、高い機能を持つことを期待している。このようなIPA/SECとJEITAとの情報共有、共同研究は、わが国の産業競争力の強化をもたらすものとして、期待をしたい。

所長対談 片山 卓也　北陸先端科学技術大学院大学（JAIST） 情報科学研究科　教授

ソフトウェア開発における形式手法の適用と普及の方策を考える

数学的な手法を用いて正しいソフトウェアを開発する形式手法（フォーマルメソッド）がソフトウェア産業から注目されつつある。特に、ICカードや組込みソフトウェアの領域では形式手法を用いた開発実績が出現している。そうした背景を踏まえ、形式手法の研究に取り組んできた北陸先端科学技術大学院大学の片山卓也教授とIPA/SECの鶴保征城所長が産業界における形式手法の適用および普及の方策について語り合った。



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、日本電信電話公社（現NTT）入社。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10月独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センター所長に就任。

- 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～）
- 高知工科大学工学部情報システム工学科 教授（2003年～）
- 日本BPM協会 副会長（2006年～）
- 実践的ソフトウェア教育コンソーシアム 会長（2006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　初めに、片山先生のソフトウェアエンジニアリングに関する問題意識とこれまでの研究について簡単にお話しいただけませんか。

片山　ソフトウェア業界ほど、理論と現実が離れている世界はないでしょう。そこで何とか理論に近い方法できちんとしたソフトウェアが作れないだろうかという思いで、エンジニアリングの手法に則ってソフトウェア開発を行う方法をテーマとして研究し、今で言う形式手法に近いことを研究してきました。

そういう取り組みを行う中で、文部科学省のe-Societyプロジェクトにも参画しています。私の研究室では、e-Societyのうち、組込みソフトウェア開発への形式手法を用いたソフトウェアの開発手法の適用に取り組んでいます。e-Societyは、学の知識を生かして産業界に役立つものを作ろうというプロジェクトですが、学と産の連携は大変うまく行われています。従来、学と産業界の交流は十分ではなかったように思いますが、e-Societyによって１つの成功例が作られたと考えます。そうした状況の中、形式手法に対する産業界の注目が高まり始めており、ソフトウェア業界が学の成果を取り入れる契機になればと期待しています。

鶴保　SECは、発足してから3年が経過しようとしています。これまで、SECは形式手法を含んだ新しい設計方法論にはあまり積極的に取り組んできませんでした。ハードウェアのプロセス改善においては人の要素が少ないのですが、ソフトウェア開発には人間的要素が重要的な役割を果たすということが理由のひとつです。新しい方法論に対する重要性は認識していますが、ソフトウェア開発に関する方法論は、人の問題を含めて考えなければなりません。そうした問題意識のもと、SECでは今後、人材育成の問題を含めて、要求工学や形式手法という新しい方法論についての取り組みを進めていこうと考えています。まず、形式手法の概要についてお話しいただけませんか。

片山　形式手法というのは、1970年代から、主にヨーロッパにおいて、主に数学に興味を持つコンピュータ関連の人たちが取り組んできました。ですから、基礎にあるのは、論理や集合、関数等の数学なのです。彼らはソフトウェアの仕様記述に数学の言葉を使うと、わかりやすいものになると考えたのです。その記述方法が形式手法と呼ばれています。形式手法の仕様記述言語にはZ記法やVDM等がありますが、当初は仕様を記述するだけでよいと考えられていました。現在の形式手法は、それに加えて、仕様やプログラムの正しさを数学的に証明することにも力点が置かれていて、産業界にも注目されているのです。

鶴保　産業界にとっての仕様とは要求を意味しますが、形式手法と要求工学の関係はどのように理解するとよいでしょうか。

片山　実際のところ、要求工学と形式手法は関係がないとの主張と、要求工学も形式手法で行ったほうがよいとの主張の２つがあります。要求を出すときには、詳細なことが決まっていないことが多いですね。要求を形式手法で表すには、対象領域に関する知識体系を持っていないと難しいのです。そのため、２

つの意見があるのです。また、機能であれば動作に直せるので形式手法で記述できるのですが、産業界で言われている非機能要求とは、いうなれば目標や目的のようなものを指しているので、形式手法で表すのは非常に難しいのです。

形式手法とドメイン

鶴保　要求をフォーマルに仕様として記述するということはどのようなことを指すのですか。読者にわかるようにご説明いただけないでしょうか。

片山　だれが見ても理解に違いが出ないように、なおかつ機械が解析できるように記述を作り出すことです。

鶴保　C言語でもそのように記述することは可能でしょうか？

片山　C言語であっても、機械語であっても形式的記述ができるかといえばできます。しかし、ソフトウェアの機能がきちんと記述されているかどうかについては、プログラムコードの記述からでは、おそらく機械では分析できないでしょう。例えばC言語は、C言語特有のデータ構造やメモリという世界を書いています。一方で、形式手法で記述しようとしているアプリケーションは、例えば銀行の業務です。両者で記述する内容のレベルは違うのです。形式手法で仕様記述をするといった場合、その仕様が人間で読めるように記述するべきなのです。C言語コードから仕様を読み取るのは、かなり難しいと思います。また、C言語やその他一般的なプログラム言語では仕様を実現するためのコードの記述量が大きくなる傾向がありますが、大きすぎると、機械での推論ができません。機械のパワーにあった記述をすることも必要なことです。機械に与えることのできる記述を作成し、解析させることが形式手法の要です。

鶴保　ところで、形式手法とドメインの関係についてどのように考えられますか。

片山　形式手法というのは仕様を記述するためのものなので、形式手法を用いてドメインを記述することはできるでしょう。ドメインの記述対象となるのは、例えばSECに関連していうと、SECの「ような」組織全体について書くことになり、SEC内部の情報システムではありません。組織には責任者がいてセクションがあり、そのセクションとセクションの関係や責任体制がどうなっているかということを全体として記述する。それが、ドメインという考え方にあたります。ドメインは抽象的なものです。それは、会社法でいうところの会社のようなものです。会社はいろいろな規模や業種があるわけですが、ドメインというときに会社を一般的なものと捉えて記述したものが、ドメインの記述に相当するのです。

片山　卓也（かたやま　たくや）
東京工業大学電気工学科卒業（1962）、同情報工学科助教授（1974）、教授（1985）、北陸先端科学技術大学院大学情報科学研究科教授（1991～）、
情報処理学会理事（1985～1986）、日本ソフトウェア科学会理事長（1991～1995）、文科省e-Societyプロジェクト「高信頼組込みソフトウェア構築技術」研究代表者（2003～）
21世紀ＣＯＥプログラム「検証進化可能電子社会」拠点リーダ（2004～）
専門：形式的ソフトウェア開発方法論、高信頼組込みシステム、ソフトウェア発展・進化機構

鶴保　つまり、その記述をベースにして、特定組織のためのアプリケーションを作ることが可能なのですね。例えば、病院情報システムのアプリケーションであれば、病院の一般的なドメインに関する記述に加えて、個々の病院が持つ独自性を加えることによって作り出せるわけですね。しかし、そういうことには、まだ十分に取り組まれていないのが現状なのでしょうか。

片山　はい、まだそこまでは進んではいないという状況です。

鶴保　形式手法の産業界への普及をどのようにとらえていますか。

片山　形式手法はまだ産業界に広く普及している段階ではありません。その理由は、私見ではありますが、仕様を記述してもそこからコードの構造が得られないことにあると思います。産業界は、ソフトウェアを作りたいわけです。ですから、仕様書を書いたときに、仕様からソフトウェアの構造が見えてくるようになっていないと産業界の期待と合わないのですね。

鶴保　FeliCaのソフトウェアの開発ではVDMを用いた形式手法が活用されましたね。

片山　VDMで記述された仕様には2つの意味があります。1つは論理式による対象の性質の記述。もう1つは、動作の記述。形式手法でプロトタイプを作り、プロトタイプで確認をし、そのあとコーディングです。FeliCaが成功した理由は、コーディングの前に徹底的に仕様をデバッグし、その後でコーディングを行ったことだと思います。また、コーディング時に発見された誤りや変更をVDMに戻すというサイクルを実行したからだと聞いています。通常は、コーディングの段階以降は、ソースコードのみがメンテナンスされます。そのために、困ったことに、VDMで書いた仕様書と、現実の動作が合わなくなってしまうのです。しかし、FeliCaの場合は、コーディング段階での変更を、すべて仕様書に戻していたことが、成功の要因でしょう。

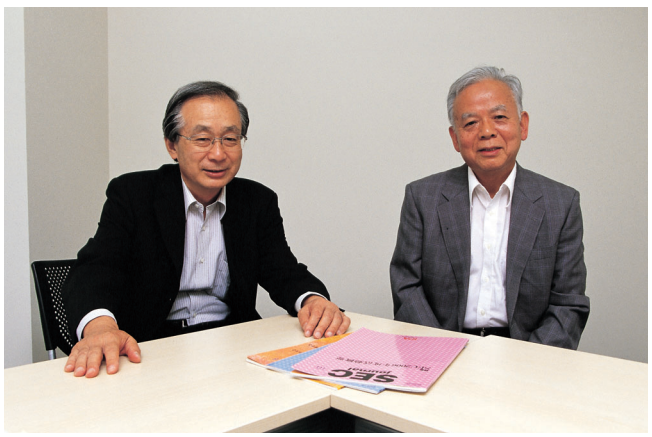
鶴保　なかなか徹底できない作業を地道に実行したわけですね。しかし、VDMを見てC言語でコーディングをする、また結

果の修正をVDMに反映しなおすというのは、人的要素や、その時々の技術上の問題もあり、エラーにつながることもあるでしょう。そのためにも、VDMからC言語等のコードが生成されると非常に便利だと思うのですが、そういう機能は今後生まれてくるものでしょうか。

片山　ＶＤＭにもそのような変換ツールは存在するようです。FeliCaの開発ではこの機能は使われなかったようですが、その理由は知りません。一般には、コーディングには仕様に含まれない詳細が必要ですので、仕様のレベルが高くなるにつれ直接変換することが難しくなると考えます。また、変換されたコードが十分に効率的なものであることも必要ですが、もしかすると、そのような実用なツールにはまだ仕上がっていないのかも知れません。

形式手法は難しくない

片山　先日、「日本の産業は現場力でもっていた」という新聞記事がありました。その記事は、現場がいろいろな知恵を持ち寄って問題を解決していった。しかし、現場で解決できない不具合の解決には設計段階に戻ってきちんと対処しておかなければいけないと述べていました。ソフトウェアの設計は、鶴保所長のご指摘のように、人的な問題や新しいテクノロジーを使わなければいけない、あるいは既存のシステムと連携を取らないといけないなど、いろいろな条件のもとで行わなければなりません。そういうことを解決できないと、形式手法できれいに書かれた仕様書があるというだけでは十分だとは認識されないと思うのです。仕様書はないけれど、ある程度ものを作らないといけないという場合もあります。そういうことを含めて形式手法が役に立つことが重要だと考えています。



鶴保　形式手法を用いた成功例も出てきていますが、ツールというものは万能ではありません。使う人の能力によっても結果が左右されるという宿命があります。ところで、形式手法は数学的素養がいるのでマスターするのが難しいという意見もあります。その点はどうなのでしょうか。

片山　それは、イエスでもありノーでもあります。少し前までの形式手法では産業界に浸透しないと思っています。形式手法のほとんどのアプリケーションは仕様を正しく記述することです。「正しさ」の保証をどこに求めるかというと、定理証明系などの数学的方法を使うという話になります。電子マネーカードSmart CardのJavaアプレットの検証に、定理証明系の形式手法を使ったオランダの大学、フランスの研究所の成果があります。形式手法を用いるとそういうことができるというインパクトはあったのですが、技術が難しく産業界には普及はしていないのが現状です。

一方、FeliCa等では、VDMで仕様を記述するのですが、実行可能な仕様として記述されていて、定理証明には使っていないのです。実行させて確認するということでは普通のテストと同じですが、コードよりレベルの高い仕様の問題が発見できるメリットがあります。また、仕様が正しく作られたなら、それをもとにテストデータを作ったり、コーディングすればいい。それだけでもかなりバグがなくなるので、FeliCaの場合は出荷後のエラーがほとんどないと聞いています。

鶴保　そうすると、テストする人の技術によって取れるバグの量が左右されないことになるのですか？

片山　仕様の誤りによるバグについては、そうなります。また、必要なテストケースを、形式的に記述された仕様から自動的に作り出すことも可能であると考えられるので、これを利用するとコーディング後にテストデータを作るよりも、問題のロジックに近いレベルでテストデータが作れます。

記述された仕様が正しいかをどうかを証明することは我々研究者にとっては大変面白いことなのですが、産業界では特別な場合を除いては要求されないのではないでしょうか。産業界では、書いたものを動かしてみて、それが技術者間での共通の理解となるための記述であるかどうかが重要なことでしょう。そう考えると、形式手法とはいっても、そんなに難しい勉強をする必要はありません。抽象的なレベルでシステムの振る舞いや機能を記述し、また、それらを理解する能力があれば十分であると思います。それはちょっとした高級言語を使うことと同じ

感覚です。

鶴保　そうすると、VDMはFortranやAlgolと同じように考えていいのでしょうか。

片山　私はそう捉えたほうがよいと考えています。

形式手法もオブジェクト指向であるべき

鶴保　オブジェクト指向が、ようやく普及してきたといえる状況になってきましたが、オブジェクト指向と形式手法との関係はどのように理解すればよいでしょうか。

片山　私は、形式手法もオブジェクト指向でなくてはならないと思っています。フォーマルに書いた仕様書をオブジェクト指向言語に書き換えるときに、その書き方が違いすぎると非常に難しい作業になります。オブジェクト指向は、世の中のシステムはオブジェクトでできているという考え方ですが、ソフトウェアでいえば、対象のオブジェクトが変化した場合には、それに合わせて仕様書を変更することになるので、仕様書自身もオブジェクト構造を取ることが望ましく、形式手法もオブジェクト指向であるべきと考えています。オブジェクト指向技術は、このソフトウェア進化に強い技術ですが、形式手法はソフトウェア進化にも貢献できる方法論であると思います。ソフトウェア進化では、問題や仕様の変更に応じて設計やコードを変更する必要があります。この変更作業は大変手間のかかる作業で現在でも十分な解決がなされていませんが、形式手法はこの問題を系統的に解決する上で必須な技術だと考えています。

鶴保　形式手法で仕様を書き、それを実行させることによって高レベルのバグを発見できるというお話でしたが、その話と形式検証とは関係があるのでしょうか。

片山　それはある意味で関係がありません。書いた仕様が正しいことを確認する方法が、1つはテストであり、1つは形式検証だということです。書いた仕様にバグがないことを数学的に証明することが形式検証です。

鶴保　そうすると、形式手法を高級言語ライクに産業界に普及させていこうとすると、当面は形式検証ということは前面に出さずに、テストの実施方法を工夫して、普及に取り組んだ方がいいということでしょうか。

片山　そうだと思います。形式手法に関わっている人たちも、定理証明などによる形式検証というのは現実に適用するのは大変だと思っています。ですから、まずテストの方で動かしてみて、良いか悪いかを見ることがよいと思います。形式検証は、その後で本当に必要な場合に行うのがよいと考えます。

鶴保　組込み系の場合でも、仕様を形式的に書くことは難しいと考え、形式検証から取り組んでみるというアプローチはどうでしょうか。

片山　今、組込み系で一部注目されている形式検証の方法にモデル検査という方法があります。それは、状態遷移図をくまなく舐め尽くす方法です。それによって問題点があるかどうかを調べるのです。その方法はもともとハードウェアのための技術だったのですが、組込み系のソフトウェアに使えるのではないかと研究が進められています。

鶴保　状態遷移図をそのまま実行するというイメージですか？

片山　実行というと限られた範囲のことですが、検証という場合は可能なすべてのパスをくまなく調べるのです。それを調べる方法論があり、状態遷移図が有限である、あるいは調べ尽くせる範囲の大きさであるという仮説のもとで、モデル検査が盛んに研究されています。SECとの我々の共同研究のテーマもそれです。

鶴保　モデル検査を成立させるためには、仕様や状態遷移図を形式仕様で記述する必要があるのですか。

片山　仕様が状態遷移図として書かれていればモデル検査ができます。モデル検査の利点は、検証が全自動で行われることですが、このほかにも、定理証明系でも自動証明をしようという取り組みがあります。OSのカーネル等のシステムプログラムにおかしなところがないかどうかを確認することに定理証明を用いる研究が行われています。以前は日本の産業界にもそういうことに取り組んでいた人がいたのですが、今は大変少なくなっていますね。

鶴保　日本の産業界にぜひとも取り組んでほしいですね。

片山　形式検証はまだまだ産業界にとって十分に使える形にはなっていないと考えています。学の側もまだまだやらなくてはいけないことがたくさんあると思っています。

鶴保　産業界が形式手法に取り組む場合、まず、形式記述から入り、形式検証は少し時間をかけて研究を進めていくことが望ましいということですね。SECは官の立場から産学連携の推進を進めています。学の成果を産に伝え、また産からの要望を学に伝えながら、ソフトウェア産業の発展に今後も尽くしていきたいと思います。

文：小林秀雄　写真：福永一興

論文　Paper

メモリリーク検出システム λ trace


青島 武伸†


伊藤 智祥†


春名 修介†

C言語のソースコードを入力とし，メモリリークを引き起こす箇所を検出するシステムλtraceの実装について述べる．Bounded Model Checkingを応用して高精度の検査を実現しているほか，数百万行の入力を処理するための実行環境，効率的な検査結果の精査環境を有する．本稿では，多くの製品開発において同システムを適用し，検討を重ねるなかで得られた知見について詳述する．

A Memory Leak Detection System: Lambda trace

Takenobu Aoshima †, Tomoaki Itoh †, Shusuke Haruna †

In this paper, we describe a memory leak detection system Lambda trace. The system automatically detects positions that may cause memory leakage in C language source code. The system has features to contribute to real projects such as highly accurate path extraction mechanism, execution environment on several PCs, and result checking environment.

1　はじめに

年々，家電ソフトウェアの開発規模と複雑さは増大の一途をたどり，出荷後に不具合が発見される事例が後を絶たない．特に，発見が困難で，かつ品質に与える影響が重大になる不具合の１つに，メモリリークに代表されるリソース管理の不具合によるエラーがある．

メモリリークとは，プログラム実行中，特定の処理を行うために確保したメモリを，その処理が終了しても解放せずに放置することによって起こる．図１に示すように，正常にメモリの解放が行われていれば，全体のメモリ使用量は一定の範囲内に収まり，正常な動作を続けられる場合でも，メモリリークによって使用メモリが増加していった場合，正常な動作の続行が困難になる．このとき，例えばプログラムが停止し，操作不能の状態に陥る等重大な品質問題が引き起こされる．

メモリリークの発見が困難となる主な理由として，リーク発生直後ではシステムは正常な動作を続ける，という点が挙げられる．一度のメモリリークが発生した直後

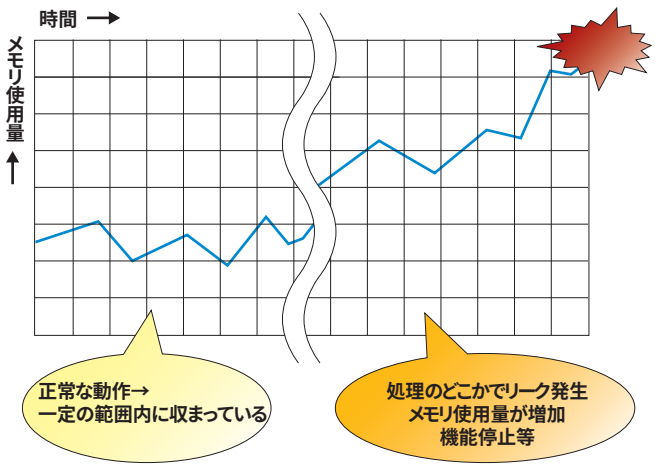


図1　メモリリーク

†　松下電器産業株式会社，Matsushita Electric Industrial Co., Ltd.

では，まだメモリに余裕がある場合が多く，不具合を引き起こすことは少ない．このため，リークの原因箇所と不具合発見時の実行箇所が異なることが多く，テストによる発見が困難になる．

この問題を解決するため，我々はメモリリークを発生させる箇所を，ソースコードを検査することにより自動的に検出するシステムλtrace（ラムダトレース）を開発した．

λtraceを開発する際に力点をおいたのは，特に次に挙げる三点である．

（1）未定義な部分を含むコードの検査

ソースコードが一旦完成してしまった段階での修正は，テスト工程の大幅な手戻り等が発生する場合が多い．このため未実装部分が残る開発途上の段階での検査の実施が重要になる．また特に組込みシステムの開発においては，アセンブラのコードが混ざる場合が多いため，これに対処する必要がある．

（2）擬陽性の排除

検査の結果，メモリリークであると判定された部分が，実際はメモリリークを起こさないという誤った検査結果を擬陽性と呼ぶ．例えばSPLint[SPLINT2002]等，プログラムに含まれる演算結果や関数呼び出し等を精度よく検査に反映しないツールを単純に適用すれば，擬陽性の検査結果が大量に出力されるため，実用的な工数での検査を行うことが困難な場合が多い．

（3）数百万行オーダーの大規模コードの検査

家電ソフトウェアで最大規模のソフトウェアは，携帯電話やデジタルテレビ向けのものであり，その規模は数百万行に上る．また大規模なソフトウェアには，しばしば千行を超える規模の関数が含まれる．こうしたソフトウェアを検査するためには，全体の規模，及び局所的な規模の大きさに耐えうる検査方式，検査を実行する環境，及び検査結果を効率的に精査し，整理する環境の整備が必要となる．

なお，本稿では，プログラム動作中にリソースが確保され，これが解放されずに残る問題を代表してメモリリークと呼ぶ．メモリ確保関数mallocとメモリ解放関数freeの対応のほか，ファイル操作関数のfopenとfcloseの対応等も同様の手続きで検査を行っている．

2　λtrace動作の概要

λtraceの入力はC言語で記述されたソースコードであり，出力はメモリリークの有無を検査した結果をファイル毎にまとめたWebページ群である．

メモリリークの検査は，

① 入力のソースコードのプリプロセス．
② メモリ確保関数を呼び出す関数のピックアップ．
③ ピックアップした関数のそれぞれについて，その関数から一定の深さまでで呼び出される関数を検査の対象範囲として切り出す．
④ ピックアップした関数を開始点として，検査の対象範囲において実行可能なパスを抽出．
⑤ 抽出されたパスについて検査を行う．

という手順で行う．

①のプリプロセスにより，入力に含まれるヘッダファイル，マクロを展開する．②では，例えば関数func1の定義にメモリ確保関数mallocの呼び出しが含まれる場合，検査の対象にfunc1が追加されることになる．③では，例えば②で選択されたfunc1の定義でfunc2が呼び出され，func2からfunc3が呼び出され，func3からfunc4が呼び出

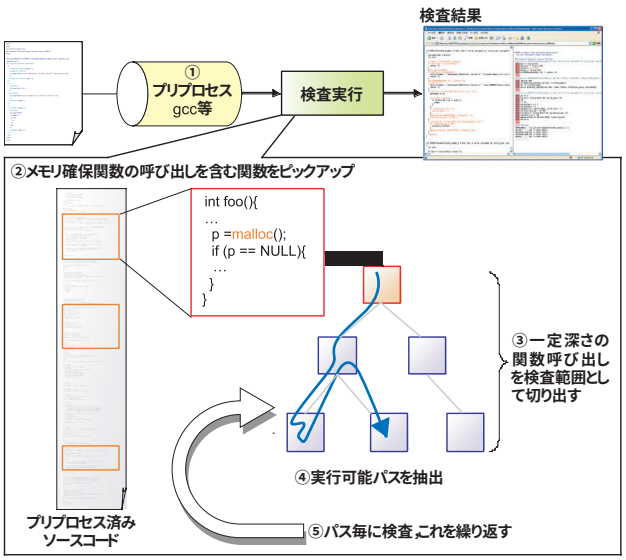


図2　λtrace動作の概要

される状況があり，検査の対象となる関数呼び出しの深さが3に設定されている場合，func3までの関数本体の定義が検査中考慮される．ここでfunc4以降の深さの呼び出しの関数の定義は考慮しないが，パスの実行可能性を判定する際に制約なしという扱いにすることで，最悪のケースを想定した検査となる．この点の扱いやすさが，第3節で詳述する実行可能パス抽出の方式のポイントといえる．また，メモリ確保関数の呼び出しを定義に含む関数は全て，②で選ばれる検査の基点となるため，一定深さ以降の関数定義を考慮しないことが，検出漏れにつながるということはない．④及び⑤については，それぞれ第3節，第4節にて詳述する．

```
 1: void foo(int   x){
 2:      int  *p, *q;
 3:    if (x > 0) {
 4:       p =    malloc(sizeof(int ));
 5:          (地点 1)
 6:    }
 7:    if (x < 0){
 8:       q = p;
 9:          (地点 2)
10:    }else{
11:       q = 0;
12:          (地点 3)
13:    }
14:      free(q);
15: }
```

図3　実行可能パス説明図

3　実行可能パスの抽出

3.1　実行可能パス

実行可能パスとは，一回の検査において，検査の対象となる範囲内で実行される可能性が論理的組合せとして存在するパスをさす．例として，図3に示すソースコードが，検査の対象である場合を説明する．まず，地点1，及び地点2を同時に通過するパスが実行可能であるかを考える．関数fooの引数xに関する制約は検査の範囲内に存在しないので，地点1を通過する条件である“x > 0”を満たすことは可能であり，地点2を通過する条件は“x < 0”であるため，これらの条件を同時に満たすことはできない．従って，地点1及び2を通過するパスは実行可能ではない．地点1と地点3を通過するパスは，通過の条件に矛盾がないため，実行可能パスである．また同様に地点1を通過せず，すなわち最初のif文の条件が偽になり，地点2を通過するパスは実行可能である．

実行可能ではないパスを検査の対象から排除することで，擬陽性の検出を防ぐことができる．

3.2　Bounded Model Checkingによるコード解析

λ traceでは，Bounded Model Checking（以下，BMC）[BMC1999]を用いてパスの実行可能性を判定する．BMCとは

① 状態マシンの状態遷移を1ステップ毎に展開してブール式として表し，

② 仕様の否定を表すブール式との論理積で1つのブール式を作り，

③ そのブール式の充足可能性を判定することで状態マシンが仕様を満たすかどうかを検査する，

モデル検査の一手法である．

①において，1つの状態は命題変数の真理値の集合，通常の状態遷移は論理積，分岐は論理和として表すことで状態遷移を表現する．BMCの特徴として，無限長の状態列を扱うことはできないが，過度の抽象化を行うことなく現実的な規模のシステムをそのまま扱うことができるということが挙げられる．またここで用いる利点として，未実装の部分やソースが手に入らない部分等検査のための情報が欠落する部分について，制約を単に省くことで「最悪のケースを想定したスタブ」が入っている状況を自動的に作り出せることが挙げられる．不適切なスタブによって，実行の可能性を見落とす場合を排除することができる．アセンブラで記述されている部分について，現在の実装では前述の未実装部分と同様の扱いをすることで検査を行っているが，アセンブラについても以下に述べるC言語部分と同様の変換を行うことで，本稿に述べる検査の枠組みの中で同列に扱うことができ，検査の精度を更に向上させることができる．

λ traceでは，実装の工数を削減する狙いから，内部で呼び出すツールとしてCBMC[CBMC2004]を利用する．ここでCBMCの動作例を通じ，プログラミング言語を対象

とするBMCについて簡単に述べる．

CBMCはまず，入力のC言語プログラムに含まれるループ，再帰呼び出し，後方へのgotoジャンプを一定回数の繰り返し文に，値を返す関数呼び出しを代入文に変形する．これにより，入力のプログラムはif文，代入，前方へのgoto，assert文のみが含まれている形になる（図4の上側）．次にこれを静的単一代入形式に変換する．これにより各変数の定義がプログラム上でそれぞれ唯一になり，論理積で連結した場合にも，実行順序を制約として表現することができる．図4の下側のうち，Cが上側のプログラムに対応する制約を表す式，Pがassert文に対応する式である．ここで，C∧¬PをCNF形式に変換し，zChaff[zChaff2001]等の充足可能性判定器に入力する．もし充足可能とするモデルが存在すれば，それがassert文に関する反例であり，充足不能であればassert文は常に満たされることを意味する．

3.3　実行可能パスの抽出

実行可能パスの抽出は，次の手順で入力となるC言語プログラムに変更を加えた後，CBMCに入力することで行う．

① 通過すべき地点に，通過した場合に真とするフラグ変数を挿入．

②フラグ変数が真かを検査するassert文の挿入．

③ CBMCのエラー回避，検査の効率化，行番号抽出のための変形（これについては3.4節に詳述）．

```
if(x != 4){
   x+=y+2;
   y=x-2;
} else {
   x++;
}

assert(x<=y);
```

```
C:=(x1=x0+y0+2) ∧
   (y1=x1-2) ∧
   (x2=x0+1) ∧
   (x3=(x0!=4)?x1:x2) ∧
   (y2=(x0!=4)?y1:y0)

P:=(x3<=y2)
```

図4　CBMCによる変換の概略

図5に挙げたプログラムは，①及び②を適用した例である．6行目のif文の条件が真，かつ11行目の条件が偽になるパスの実行可能性を調べるため，フラグ変数flag0及びflag1を挿入している．

assert文は，全てのフラグ変数が真となるとき，偽となるように挿入する．これにより，フラグ変数が全て真になる場合にCBMCで反例が出力される．この反例から，フラグ変数を真にした箇所を全て通るパスの実行例として，実行される行が取得できる．

なお，図5で挙げた例は，実行可能となる．6行目のif文条件の“x > 0 && x*x+y*y < 0”はx, yが整数であれば真になりえないが，オーバーフローも考慮するため，真になる組合せが存在する．実際，反例として出力されるx=452984830，y=33554430の値を設定してプログラムを実行した場合，同様のパスを通ることが確認できる．

フラグ挿入の組合せの集合として，λ traceは以下の2様式を用意している．

① ブランチカバレッジが100%になる組合せの集合．

② malloc等メモリ確保を行う関数を通り，free等メモリの解放を行う関数を通らない組合せの集合．

2様式のうち，①は網羅率が高いが，フラグ変数の挿入箇所の組合せが多く，検査に時間がかかる．②は，メモリリークを検査する上では効率がよい戦略といえ，検査する時間に余裕がない場合は，②を選択する．また特に②の場合，if文等の分岐が大量にあっても，パスの組み合わせの増大に左右されることなく検査すべきパスを

```
 1: int foo(){
 2:   int flag0 = 0;
 3:   int flag1 = 0;
 4:   int x, y;
 5    int *p;
 6:   if (x > 0 && x*x+y*y < 0) {
 7:     flag0 = 1;
 8:     p = malloc(sizeof(int));
 9:   }
10:
11:   if (x*y-y*1000 > 1){
12:     free(p);
13:   }else{
14:     flag1 = 1;
15:   }
16:   assert(flag0 == 0 || flag1 == 0);
17:}
```

図5　フラグ変数・assert文挿入例

抽出できる．この点が本方式のポイントの１つである．

ここで，重複するパスを検査しないため，実行可能と判定された場合の実行パスを全て蓄積していき，新たなフラグ位置の組合せが改めて実行可能かどうかを判定する必要がある場合にのみ，そのフラグ位置において実行可能性を判定する．例えば①の場合，新たに挿入するフラグ変数の箇所が蓄積済みの１つのパス上に全て存在する場合，改めて実行可能性を判定する必要はないため，このフラグ変数の箇所での判定を行わない．

3.4 CBMCの利用

本節では，CBMCを利用して実行可能パスを抽出する際の入力ファイルの変形について述べる．ここでの変換は大きく分けて2種類ある．１つは部分的に抜き出したプログラムを検査するにあたって必要なメモリ確保関数の補完である．これはCBMCに限らず同様の枠組みでツールを利用する場合は必要になる変形である．もう１つはCBMCを利用するために特有の変形である．

まずメモリ確保関数の補完について述べる．図6に挙げる関数が検査の対象として切り出された場合を考える．この例のように，プログラムを部分的に切り出した場合，ポインタ型の引数やグローバル変数の指すアドレスにはメモリが確保されていることが前提である場合がほとんどである．図6では，この範囲でpの値が設定されていないため，そのままの形で実行可能性を判定しようとする場合，「無効なポインタの参照」を検出して判定が終了してしまう．検査の範囲内が原因の無効なポインタの参照の場合は，その検出をもって終了としてかまわないが，コードの一部を切り出したことが原因のものは排除する必要がある．

このため，検査の開始点となる関数の引数，グローバル変数，静的変数に関しては図7の上側2行目のように，予めメモリを確保する関数を挿入する変形を行う．構造体型のポインタがネストしている場合等では，何段階の補完を行えばよいか，検査前に確定できない場合があるため，実行可能性の検査が無効なポインタの参照を検出して終了するたびに，1段階ずつ補完を増やしていく(図7下側3行目)．

また，第2節の説明で述べた関数func4のような，検査の範囲外になる最初の呼び出し深さに位置する関数の返り値の型がポインタ型である場合や，引数の型がポインタへのポインタ型である場合では，図7と同様のメモリ確保を，返り値や引数のポインタ変数に対して行う定義に置き換える．

```
 1: typedef struct _wz {
 2:     int a;
 3:     int b;
 4:     struct _wz *next;
 5: }wz;
 6:
 7: int foo(wz* p){
 8:     *p->a = 1;
 9:     *p->b = 1;
10:     *p->next->a = 2;
11:     return 1;
12: }
```

図6　メモリ確保関数の補完が必要な例

```
1: int foo(wz* p){
2:     p = malloc(sizeof(wz));
3:     {
4:         *p->a = 1;
5:         *p->b = 1;
6:         *p->next->a = 2;
7:         return 1;
8:     }
9: }

                    ↓

1: int foo(wz* p){
2:     p = malloc(sizeof(wz));
3:     p->next = malloc(sizeof(wz));
4:     {
5:         *p->a = 1;
6:         *p->b = 1;
                    …
```

図7　メモリ確保関数の補完の例

次に，CBMCを内部ツールとして利用するにあたって必要になる変形について述べる．これに関する変形は，数千万行に及ぶソースコードでテストを繰り返す中で必要となるものを抽出していき，100を超える変形がλtraceに実装されている．ここでは紙面の都合上，代表的な例を取り上げる．

（1）型が異なる代入文について，キャストを挿入

型が異なる代入で出力されるエラーを回避するため，明示的なキャストを挿入して回避する．これに類似するものとして，例えばヌルポインタ判定による条件分岐のため等で，if文等の条件節にポインタ型の変数が置かれ

る場合では，CBMCがエラーを出力して検査を止めるため，条件節の式をint型にキャストする．

（2） 検査対象の関数に関連しない宣言の除去

第2節で述べた，検査の対象範囲として切り出された関数に関連しない宣言はすべて除去する．ここで関数 f に関連する宣言とは，fの定義に含まれる全ての関数呼び出しや変数について，これらに関する型や関数の宣言であり，また関連する型の定義の中で使われる型の定義を再帰的に集めたものである．関連しない宣言を除去することで，CBMCの性能を大幅に改善する効果がある．

ここでは実装の仕方に多少工夫が必要となる．1ファイルがときに十数万行にも及ぶプリプロセス済みファイルでは，型宣言文の依存関連を示すグラフが巨大になるため，効率的に処理する必要がある．λ traceの実装では，型宣言文の依存関係グラフを強連結成分に分解し，１つの強連結成分をノードとするグラフを生成し，このグラフ上で探索を行うことで，効率化を実現している（図8）．

（3） 関数freeの無効化

CBMC自身にmallocとfreeの対応を検査する機能があるが，多くの場合に正しい結果が得られない．またこの機能が原因でCBMCがエラーを出力して検査を中断する場合も多いため，freeは別名の何もしない関数に置き換える．これにより，純粋に実行可能パス抽出のためのBMCツールとして機能させる．3.2節の図4で紹介したような変換はassert文の挿入ではよく機能するが，次節で述べる参照カウント等を用いる検査は，変数の動的属性等複雑な要素を考慮に入れる必要があり，BMCとして織り込むにはなじまない．このためλ traceではBMCによるパス抽出と検査部分を分離している．

以上は比較的単純な変形だが，なかには大掛かりな変形もある．次に挙げるのはその１つである．

（4）return文が複数の場合，１つに集約

関数定義において値を返却する箇所が複数ある場合，BMCではこれを適切に扱えない場合が多い．そのため，goto文と代入文を適切に挿入することによってreturn文を１つに集約する変形を行う．このとき，複数のgoto文から同一のラベルにジャンプするプログラムの処理にも不具合があるため，これも同時に回避する形にする（図9）．

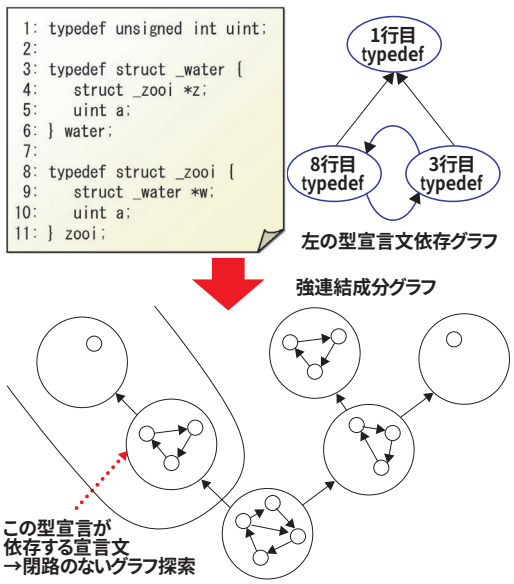


図8　強連結成分グラフの利用

4　パスの検査

本節では，第3節で述べた手続きによって，実行可能と判定されたパスについて，メモリリークを引き起こす

```
1: int foo(int x){
2:     if(x == 1) return 2;
3:     if(x == 2) return 4;
4:     if(x == 3) return 8;
5:     return 16;
6: }
```

↓

```
1: int foo(int x){
2:     int ret;
3:     {
4:         if(x == 1){ ret=2; goto RET1; }
5:         if(x == 2){ ret=4; RET1: goto RET2; }
6:         if(x == 3){ ret=8; RET2: goto RET3; }
7:         { ret=16; RET3: goto RET4; }
8:     }
9:     RET4: return ret;
10: }
```

図9　return文集約のための変形

かどうかを検査する方法について述べる．

検査を実行する手順は，以下の通りである．

① 実行可能パスとされた命令列を一列に並べたCプログラムを生成する．

② 検査のためのコードを挿入する．

③ 生成されたコードをコンパイルし，実行することで検査を行う．

①では，分岐のないC言語プログラムが生成される．基本的にこれを実行することで，検査対象となっているパスをそのままトレースすることができる．実行可能なプログラムとするため，第3節で述べたメモリ確保関数の補完や関連しない宣言の排除等を同様に行う．同一関数が2度以上呼び出される場合は，それぞれの呼び出しで実行パスが異なる場合があるので，呼び出し回数分，別の関数として定義する．例えば，図10の関数mainに対応するパスの6行目と8行目に関数fooの呼び出しがあった場合，foo1とfoo2とに定義を分け，1回目と2回目の呼び出し時のパスに対応する命令列をそれぞれの定義とする．

②では，確保されたメモリのアドレスを参照する変数の数を数えるコードを挿入する．新たな変数にそのアドレスを代入することで増え，変数のスコープ外に実行が移った場合や，上書きで別の数が代入された場合に減る．参照の数が0になったときにメモリが解放済みではなかった場合，リークと判定する．

検査の開始点となっている関数の返却値として，あるいは参照渡しの引数を通じて，メモリを参照する値が返

```
 1: int main(){
 2:     char * buffer;
 3:     int ret;
 4:     buffer = malloc(2048);
 5:     buffer == ((void*)0);
 6:     ret = foo1(buf);
 7:     ret != 1;
 8:     ret = foo2(buf++);
 9:     return ret;
10: }
11:
12: int foo1(char* p){
13:     …
14: }
15:
16: int foo2(char* p){
17:     …
18: }
```

図10　抽出したパスをトレースするコード例

される場合，この検査範囲ではリークと判定することはできないが，λtraceではメモリを参照する値が返却値，あるいは引数を通じて返されることを結果に明示する．これを次節で述べるユーザ定義関数の検査と組み合わせることで，より範囲の広い，漏れの少ない探索を行うことが可能となる．

また，グローバル変数にメモリを参照する値が渡される場合も，検査結果に明示する．二度と使われないメモリかどうかを自動的に判定することは困難な場合が多いが，開発者はこの情報を参考にすることで，適切に解放が行われているかのチェックを効率的に行うことが可能となる．

5　ユーザ定義関数の検査

λtraceではmalloc等標準APIでのメモリリーク検査のほか，ユーザの追加するリソース確保，及び解放関数に関する検査も行う．これらの関数は，下記項目を設定することで追加することができる．

・関数名，リソース確保関数か，解放関数か

・確保，あるいは解放の対象となるポイント

・成功した場合，及び失敗した場合の返り値

・実行パス抽出時に本来の定義を使用するか

「確保，あるいは解放の対象となるポイント」とは，例えば関数mallocのように，返り値にリソースが確保される場合は0，第1引数にポインタ型の変数を参照渡しして，そこにリソースを確保する関数の場合は1，以降第2引数ならば2等とする．リソースを指す変数はポインタ型に限らず，リソースの固有値を表すint型等であっても対応する．

「成功，失敗時の返り値」の設定は確保，及び解放の成否判定の条件分岐に対応するために用いる．数値に限らず，enum型の文字列にも対応する．

「本来の定義を使用しない」とは，元々の関数定義は無視し，返り値の型に対応したメモリ確保関数の補完コードが実行可能パス抽出用のコードとして挿入されることをさす．複雑な実装を隠蔽し，検査を効率化する．

図11は，これらの設定を行うための環境である．

各項目の設定からXMLファイルを生成するほか，ソー

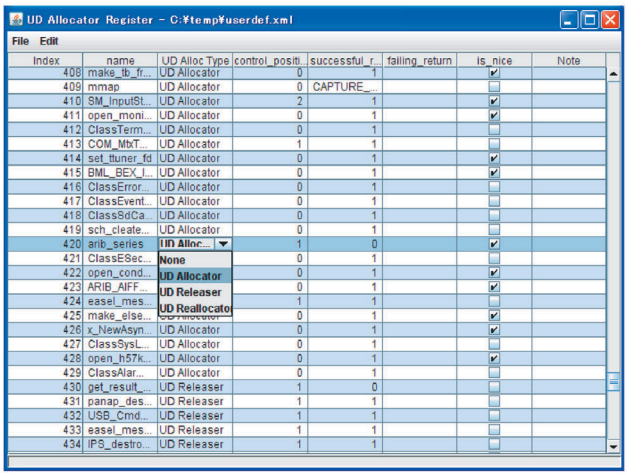


図11　ユーザ定義関数の設定環境

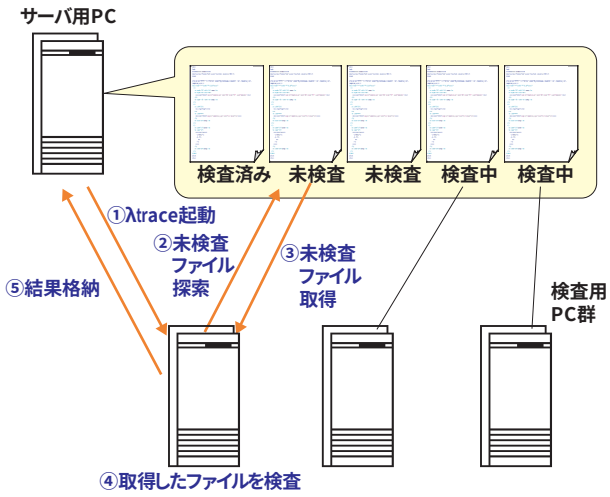


図12　分散実行環境

スコードからの関数一覧の抽出，第4節で述べた検査と連動し，引数にリソースへのアドレスが渡される場合に，これを表示する等の機能を有する．

6　分散実行環境

本節では，λtraceを複数のPCで同時に実行させる分散実行環境の実装について詳述する．

分散実行環境は，図12に示すとおり，1台のサーバ用PCと複数台の検査用PCとからなる．サーバ用PCは，入力ファイル群と，これらファイルが未検査か，検査中か，あるいは検査済みであるかという状態を管理する．

検査の実行は，以下の手順で行う．

①サーバ用PCがλtrace起動コマンドを各検査PCに発行．

② 各検査PCは起動コマンドによりλtraceを起動．

③ サーバ用PCにある未検査の入力ファイルを取得．

④ 取得したファイルの状態を検査中に変更した後，検査を開始．

⑤ 検査結果をサーバ用PCの検査結果フォルダに転送，そのファイルの検査状態を検査済みに変更し終了．

⑥ 全てのファイルが検査完了になるまでこれを繰り返す．

この環境により，極端に検査に時間を要するファイル等が原因で，週末に実行した検査がほとんど完了していない，という事態を回避することができる．

7　検査結果の出力

検査結果の出力は，左右に分割された2つのページで構成される（図13）．

左のページは入力のソースコードを表示し，リークを検出したパスを赤く強調している．右のページはリークを起こすまでの命令列であり，関数呼び出しやループ等は展開し，1つの列として表現する．

この表示で，検査結果が実際にメモリリークを起こすかどうか，修正の必要があるかどうかをユーザが判断する．このとき，実行パスは数十行に及ぶ場合が多く，精査に時間を要する場合がある．そこでλtraceでは，

①ソース中の変数をクリックした場合に，同一の変数を全てハイライトする．

② マウスカーソルをソース中の変数に重ねた場合に，対応する宣言，ローカル変数かどうか等をポップアップで表示する．

という方式を導入している（図14）．

結果の精査では，基本的にメモリを参照している変数のゆくえ，値の受け渡しの様子をたどっていけばよいため，上記①，②の仕組みが工数削減に非常によく寄与する．我々が検査を行った範囲での実績では，リークが検出された関数の平均行数が61.8行，ここでリークへのパスとして指摘された行数（図13左の赤い行数）の平均が

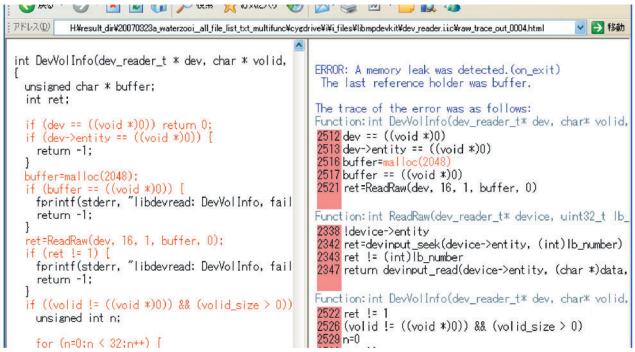


図13　検査結果の出力

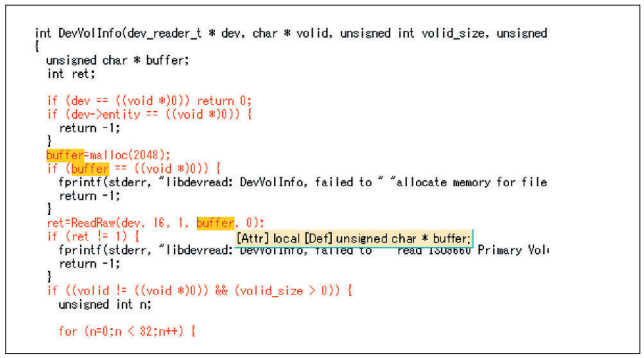


図14　特定変数のハイライトの例

17.3行，これに対し，精査時にハイライトされた平均行数は3.5行であった．すなわち，見る必要のある行数が79.8%減り，大幅な工数削減を実現している．

8　評価とまとめ

λtrace開発の傍ら，数多くのプロジェクトの協力を得て，開発中のソースコードに適用を重ねた．コードの行数の総合計は約5,000万行であり，大規模なコードへの適用において，有効性を確認した．

また，他のツールとの比較として，商用の一般的な静的解析ツール群との比較を行った．簡単のため，以下では比較対象のツール群をAと呼ぶ．ここでは，検出すべきリークを検出できなかった「検出漏れ」と，検出した結果が誤っている，すなわち第1節で述べた「擬陽性」となる場合について分けて述べる．

8.1　検出漏れ

まず，検査漏れとなる場合について述べる．λtraceが検出し，Aで検出漏れとなるリーク箇所の主な特徴は以下の通りであった．

（1）大量のif分岐を含む関数

Aでは，基本的にパスの組合せを展開し，抽象実行を行うことで検査を行う．このとき一般的に特定の値（5,000～20,000程度で設定による）のパス数を上限として検査を打ち切る．ある43個のif文を含む，総行数508行の関数では，Aでは検査が打ち切られたが，λtraceでは12通りのパスをピックアップして検査し，リークを検出した．このように，λtraceでは，一度の検査範囲をいくつかの関数に限定した場合にも，その検査範囲が大規模になるソフトウェアの検査に対応することができている．

（2）関数ポインタ，未定義部分を含む関数

Aでは，関数ポインタを含むパスについて，検査が中断される等する場合が多い．λtraceでは，実体の関数を返り値と引数の型から推定する機能もあるが，推定が不可能な場合でも「最悪のケース」を想定して検査を続行するため，問題にならない．

逆にλtraceで検出漏れとなるケースでの原因は，CBMCがうまく動作しない場合が殆どで，3.4節で述べたようなCBMC向けの変形を追加する対応を行う必要があった．例えばCBMCではGNUによるC言語の拡張表記に対応していない．このためλtraceでは「ネストした関数定義」や「caseの範囲指定」等，λtraceを適用したプロジェクトの中で頻繁に使われている拡張について，CBMCで解釈可能なANSI Cに準拠する形に変換を行うことで対応している．対応していない表記法が検査の範囲に含まれる場合は検査が中断され，ここにリーク箇所が存在する場合には検出漏れとなってしまうが，検査が中断された関数についてはその名前を結果出力において列挙し，他の方法での検査，あるいはレビューを促すことで対処している．

8.2　擬陽性

次に，擬陽性を引き起こす要因について述べる．Aの多くでは関数呼び出し間の変数値の整合性が無視され，実行しえないパスをリークとして検出する場合がある．数値の整合性に関する検査結果の精査は非常に労力を要

するが，λ traceでは数値の整合性を考慮したパスの抽出を行った上での検査を行っているため，そうした問題はない．

ツール群Aでも同様の問題が起こるが，λ traceで擬陽性となる原因のうち特に多いものとして，第2節で述べた検査時に考慮する関数呼び出しの深さの限定が挙げられる．例えば，グローバルなポインタ変数から連なる線形リストにデータを登録する関数を呼び出し，生成したデータを保存する場合を考える．ここで線形リストに登録される部分が検査の範囲外となったとき，データが別の箇所に保存されることを検知しないためリークと判定する．こうしたことは，プロジェクトにもよるが検査結果の約半数に及ぶケースもあった．λ traceでは初回の検査で前述の例のような関数を発見した場合，第5節に述べたユーザ定義によってリソース解放関数と指定することにより，以降の検査での同様の擬陽性検出をなくすことができる．

8.3 検査時間

ツールの実行時間について述べる．約500万行のコードの検査において，Aが数時間から数十時間を要したのに対して，λ traceはPC10台の分散実行環境上で5日を要した．この規模の検査では，1日でツールの実行を終えたとしても結果の精査に数日を要し，逆にλ traceのように，ツール実行に数日を要しても，その実行と順次出力される結果の精査は並行に行える．このため実行時間の違いは許容できる範囲であり，λ traceは全体の規模が大きい検査にも十分対応できたと考えている．ただし実行の高速化を実現すれば，検査範囲等の設定を適切に行うための試験実行を効率よく行うことができるため，今後の主な課題として挙げられる．例えば，本稿で述べた方式を全体に適用するのではなく，検査精度が低く，擬陽性となる場合が多くとも，検査漏れがなく高速に動作する検査方式を第一段階のスクリーニングとして適用し，ここで絞り込まれた部分について，本稿の方式を適用し，検査精度を高める方法などが有効である．

8.4 まとめ

本稿では，メモリリークの原因箇所を検出するシステムλ traceの実装について詳述した．ここに述べたシステムの枠組みは，メモリリーク検出以外にも幅広く応用できるものと考えている．

まとめとして，λ traceを適用してきた経験をふまえ，このような検査ツールが現実のプロジェクトに寄与するために重要と考える点を以下に述べる．

（1）検査アルゴリズム

検査アルゴリズムで重要な点は特に，ツールの振る舞いの分かりやすさと考えている．関数の戻り値に関する整合性が無視されるなどの原因による，検査アルゴリズムに精通していない開発者にとって不可解な擬陽性の検査結果が大量にある場合，そのツールは継続して利用されない傾向がある．実感として擬陽性の結果が全体の5割を超えると，精査を続けるモチベーションを維持する苦痛は急速に増加する．

強く擬陽性を減らすことを目指せば，検出漏れは増加する場合が多いが，これについては検出漏れが起こりうるケースを明確にした上で，他のツール，あるいは検査アルゴリズムを複合して用いるということで対処するのが現実的と考えている．

（2）ツール実行環境

検査アルゴリズムも重要だが，実行環境，結果の精査環境を，開発プロセスに適合する形で提供することが重要であることを最後に強調しておきたい．特に検査対象が大規模である場合，検査の実行や結果のまとめ自体が困難である場合が多く，この点への配慮なくして現実のプロジェクトへの寄与はなしえない．

参考文献

[BMC1999] A.Biere, et al., Symbolic model checking without BDDs, LNCS1579, pp.193-207, 1999
[CBMC2004] E.Clarke et al., A Tool for Checking ANSI-C Programs, LNCS2988, pp.168-176, 2004
[SPLINT2002] D.Evans, et al., Splint-Secure Programming Lint, http://www.splint.org, 2002
[zChaff2001] L.Zhang, et al., Efficient conflict driven learning in a boolean satisfiability solver, Proc. of ICCAD2001, pp.279-285, 2001

Overseas Report

SPICE Days 2007に参加して

SEC 企画統括グループ 研究員　新谷　勝利

①SPICEとは？

SPICE はSoftware Process Improvement and Capability dEterminationの省略形で、ISO/IEC 15504 Software Process Assessment（以降、ISO/IEC 15504と略称する）のプロジェクト名であり、またISO/IEC 15504に基づくプロセス診断・改善を実施するための国際組織の名称でもある。したがって、ISO/IEC 15504によるプロセス改善活動全体の呼称としても用いられている。

今回著者は、6月18～20日にフランクフルトで開かれたSPICE Days 2007※1に参加した。出席の目的は、現在実施中の経済産業省タスクフォース「プロセス改善研究部会」及びその成果の発表と、ISO/IEC 15504に準拠した日本のプロセス改善へのアプローチを紹介すると共に、プロセス診断・改善に関連する欧州の最新動向を探ることであった。

ここでは、SPICE Days 2007に関連する事項を報告すると共に、プロセス診断・改善に関連するトピックスについて報告する。

②ISO/IEC 15504とは？

ISO/IEC 15504は、1993年からISOで議論が進められ、1998年にまず「ISO/IEC技術報告書」としてまとめられ、実開発環境における有効性を確認する試行を経て、2004年にISO/IEC 15504 Software Process Assessmentとして発行された。日本では、2007年度中には翻訳され、JIS（X 0145予定）として制定・発行すべく作業が進められている。

ISO/IEC 15504の特色は、ソフトウェア開発ライフサイクルの各プロセスを診断の対象としていることである。各プロセスは明確な「達成目的」を持っている、という認識のもとに、その定義にあたっては、「目的」、「成果」と成果に至るまでの「活動のリスト」をセットとして記述する。プロセスを診断するということは、この記述を、予め決めた視点を通して数値化できる方法で確認する、というものである。

プロセス改善にあたっては、対象とするプロセスに関して、広くソフトウェア開発に関係する者が共通目的を持つ必要がある。その下で、各プロセスは合目的的に開発現場で実施される。1995年に国際的に合意された開発プロセスを定義する「ISO/IEC 12207 Software Life Cycle Processes」は、この共通認識により、2002年に追補版が発行され、プロセス定義の様式が上記の目的、成果、活動のリストに統一された。2007年度にSECより発行された「SEC BOOKS 共通フレーム2007～経営者、業務部門が参画するシステム開発および取引のために～」はこの様式でプロセスを定義している。

③プロセス改善関連における標準化の動き

（1）米国の動き

元々米国政府国防総省に納入されるソフトウェアに

筆者のプレゼンテーション

※1　SPICE Days 2007は、ドイツのNPOであるiNTACS（International Assessor Certification Scheme）が主催、今回の会議にはドイツ、欧州を中心とする約200名の参加があった。

は、品質の悪化、予定以上の経費、納期の遅れといった問題を抱えていた。この問題の解決を目的として、1980年代の終りから米連邦政府予算によりソフトウェア工学研究所（SEI：Software Engineering Institute）がカーネギー・メロン大学に開設された。この研究所にてWatts Humphreyを中心として研究開発されたモデルは、ソフトウェア開発現場にソフトウェア工学の導入を推進するものであり、その方法として、ソフトウェア工学の導入の程度を評価し、導入を動機付けようとするものであった。多くの企業における試行を経て、SW-CMM （Software Capability and Maturity Model）となり、現在は対象をSW開発に限定することなくCMMI 1.2（IはIntegration）になっている。CMMI 1.1においては、ISO/IEC 15504への準拠を明記していたが、CMMI 1.2では必要に応じて参照をしている。

（2）国際的な動き

CMMIが米国政府の管理下にあることもあり、ISOでは、パブリックドメインで活用可能で、ソフトウェア工学の適用をベースとする、開発プロジェクトのプロセス能力の診断・改善モデルの必要性が議論され、開発されたのが前項②で説明したISO/IEC 15504である。これは、前述のSEIが開発したモデルに加え、以下の点も取り込んだ国際標準になっている。

- TRILLIUM：カナダの通信関係ソフトウェア開発能力評価モデル
- スコットランドの企業が開発したPPAというソフトウェア技術診断ツール
- ヨーロッパESPRITプロジェクトの一環として開発されたBOOTSTRAP

このISO活動の成果について、前項②の経過を経て、

キーノートの風景

2004年に5分冊からなるIS（国際標準）となった。5分冊目であるパート5は、アセスメントモデルの例を示している。欧州を中心として、ドメインに応じたモデルとしてAutomotive SPICEが自動車業界用として、SPICE4SPACEが航空宇宙用として活用されている。その他、金融と医療用が活用され始めている。現在、ISOでの標準開発作業は、対象開発プロセスをソフトウェアからシステムに拡大（参照プロセスモデルを主としてISO/IEC 12207からISO/IEC 15288に）することと、評価対象をプロジェクトから組織に拡大（プロセス単位の「能力度」診断・改善から複数のプロセスの集合からなる組織の「成熟度」に）する作業を行っている。

（3）日本の動き

日本では、前述の国際的な動きに呼応し、1993年5月にSC7/WG10として、IS開発に参画し、ISO化することに貢献してきた（エディタとして3名参加、積極的なコメントの提出、国際会議での発言等）。現在JIS化作業が進められており、2007年度末にはパート5を除き揃う予定である。

パート5に相当するものとして（ただし、オリジナルのパート5が一般的例示でしかないものを現場に近

会場の入口

いプロセス・アセスメント・モデルの例として)、新日鉄ソリューションズ株式会社は、モデル及び手法等をパッケージにしたSPEAK（Software Process Evaluation & Assessment Kit）を自社及び新日鉄グループ用に開発した。現在、これをパブリックドメインで使用可能なようにSECプロセス改善研究部会で改定し、「SPEAK IPA版」としてSEC-Webサイトにて公開した。

なお、今回リリースされた「SPEAK IPA版」には、社団法人 情報サービス産業協会（JISA）にて開発されたSPINACHも軽量モデル、簡易アセスメントの方法論として統合されている。

④プロセス評価・改善の実践

（1）米国を中心とした動き

SEIでは、登録アプレーザ（現在約800人が世界中におり、うち日本には約5%の約40人）がSCAMPI（Standard CMMI Appraisal Method for Process Improvement）という手法を用い、以下の評定をSEIに登録する仕組みを持っている。

- 連続表現：レベル0～5の能力度
- 段階表現：レベル1～5の成熟度

（2）欧州を中心とした動き

iNTACS（International Assessor Certification Scheme）というドイツにて登録されているNPOがISO/IEC 15504のアセッサを登録する仕組みを作っており、現在約300人が登録されている。SEIのような、アセッサが診断した結果を登録し、ホームページで公開する仕組みはない。ただし、ドイツを中心とした自動車及び部品メーカはAutomotive SPICEモデルを使用し、iNTACSに登録されたアセッサによる診断結果を発注要件にしており、これを適用する企業が増加している。INTACSでは診断結果を登録する仕組みは作られているが、まだSEIのようには広く活用されてはいない。

（3）日本では

日本は、プロセス診断・改善に関わるISO活動に参加した当初から「レベル数値の弊害」について懸念を持ち、セルフアセスメントの推進を議論の中心にしてきた。したがって、アセッサについては、自社内でプロセス診断ができるレベルの人材を（新日鉄ソリューションズ社内（計30人程度)、JISA及びJSA主催（計40名程度））それぞれアセッサ教育コースで育成してきた。アセッサ登録制度はない。また、新日鉄ソリューションズ社以外では、アセッサの教育訓練をここ数年どこも実施していない。ただし、自動車部品メーカが欧州への輸出時にAutomotive SPICEの評価を受けることが発注仕様書に明記されていることを踏まえ、名古屋市工業研究所、ビジネスキューブ・アンド・パートナーズ、コンピータジャパンがAutomotive SPICEのアセッサ教育及び訓練をしている。また、宇宙航空研究開発機構（JAXA）はフラウンフォーファー協会のIESE（Fraunhofer Institute Experimentelles Software Engineering）等で訓練を受けている。

⑤SPICE Days 2007参加からの所見

（1）日本におけるプロセス改善スキームの遅れ

遅れの主たる理由は、前述の「レベル数値の弊害」についての懸念から、ISO/IEC 15504及びそのJIS版を開発することに主たるエネルギーを注いだため、一部を除きプロセス診断・改善実践スキームを開発、展開することに疎かったためである。アセッサ登録制度が日本にないことから、単純比較は適切ではないかもしれないが、ドイツは227人であるのに対し、日本は10人と少ない。なお、日本のアネッサはビジネスキューブによるAutomotive SPICEの教育を受けた者である。

今回SPICE Days 2007の参加にあたり、iNTACS会長との会議において、プロセス診断・改善の実践のためにはアセッサが必要であり、その登録はISO/IEC 15504

による定義に基づくことが重要であると共通認識した。現在は、前述③の(2)で示すISO化以前の様々なモデル及び手法からの移行期であり、日本におけるISO開発初期からの努力を認め、プロセス診断・改善のためのアセッサ資格者として、日本のWG10メンバによるISO/IEC 15504教育を高く評価し、iNTACSに登録できることを確認した。

今後日本は、アセッサ教育とその認証を実施し、登録し、アセスメントが国際的適合性に適っていると宣言できるような仕組みを作る必要がある。どのようにこれが可能か、2007年度後半のプロセス改善研究部会の検討課題としたい。

(2) ドイツを中心とした自動車業界の動きは ストラテジック

欧州におけるこれからの自動車開発においては、サプライヤのグローバルな位置づけの高まりを想定し、また最終顧客ニーズへの対応にはソフトウェアの果たす役割の高いことを認識し、サプライチェーンの全ての関与者がソフトウェア開発プロセスへの国際標準(ISO/IEC 12207)準拠を導入し、その実装の管理においても国際標準の導入(ISO/IEC 15504)準拠により、まさしくプロセス指向のグローバルなサプライチェーンを構築すべく活動していることを実感した。共通のプロセス診断・改善をAutomotive SPICEのもとに推進するべく、欧州の自動車関連企業がAutomotive SIGを作り、体系化・維持しており、その推進役はHIS(ドイツの完成車メーカの会議体)及びVDA(ドイツ自動車工業会)である。

(3) グローバルなサプライチェーンへの プロセス評価・改善への動き

ボッシュとTRW Automotive社から筆者が参加した1セッションに参加があり、これらはCMMI導入社として知られているところだが、CMMIからISO/IEC 15504への移行を進めていることを明言していた。同様なことを、基調講演の中で米国から参加したデルフィの品質管理マネージャが述べていた。理由として、サプライチェーンは一番弱いところからブレークしてしまう可能性があり、関係する全てのサプライヤが同様の導入を期待されている。よって、対経費においてCMMIよりISO/IEC 15504が有利という判断があるように思われる。また、CMMIは基本的に組織を対象(段階表現の場合の成熟度の考え方)としているのに対し、ISO/IEC 15504はプロジェクトを対象にしているので導入しやすいところが大きな動機になっている模様である。しかしながら、CMMIの導入が減少しているということではない。プロセス改善の必要性が認識され、可能なところから手をつけているということであろう。

⑥プロセス改善研究部会における今後の活動

日本においても、プロセスデータに基づく、よりエンジニアリング的なソフトウェア開発を進めることが必要と考える。即ち、プロセス診断に終わらず、診断結果からプロセス改善を実施し、QCD(Quality, Cost and Delivery)の向上を進めることが必要である。そのためには、レベル数値の弊害に留意しながら、アセッサ登録及び診断結果をベンチマークとして使用できるような仕組みが必要ではないか。そのためには、現在進めているプロセス改善研究部会は、

- 典型的なアセスメントモデルの提供
- ドメインに応じたアセスメントモデルの作成法の提供及び事例の提供
- 診断手法の提供
- アセスメント方法論の教育
- アセッサの育成と登録
- ベンチマークの仕組み提供
- プロセス改善を推進する仲間の増強、相互の助け合い

等々を整備していく必要があると考えている。

全体会議場のようす

先進ソフトウェア開発プロジェクト Part III

大きな足跡を残して2年余の計画を終了

SECエンタプライズ系プロジェクト　研究員
神谷　芳樹

SEC企画グループ　研究員
樋口　登

SECでは2005年春から約2年間にわたり、「先進ソフトウェア開発プロジェクト」（先進プロジェクト）を推進してきた。具体的には、実際に進行中のソフトウェア開発プロジェクトを総合的に計測し、計測結果及び計測データを分析し、その結果を可視化してプロジェクトマネジメントにフィードバックする実験を行った。計測対象のプロジェクトは「プローブ情報プラットフォームソフトウェア」を開発するもので、政府が発注し、ユーザとベンダで構成する技術研究組合が受注したマルチベンダ広域分散開発形態の中規模プロジェクトである。関係各方面の熱意ある支援、協力によりソフトウェア工学の分野でこれまでに例を見ないプロジェクト計測が実現した。対象プロジェクトは成功裏に完遂され、この領域に多くの足跡を残した。SECではこのプロジェクトで実現した環境を広く再現・普及できるようにEPM※1ツールキットを整備し、検証にご協力いただける企業に提供を開始した。この先進プロジェクトの2年間を概観し、今後の展開について報告する。

①「公開デモ」による成果の披露

SEC先進プロジェクトについては、これまで機会がある毎に報告してきた[HIGUCHI2005][MITANI2005][MATSUURA, MITANI, HIGUCHI2006][MITANI2006]。対象となるプローブ情報プラットフォームソフトウェアは、主にタクシー、バス、物流車両を、車両位置等の情報を発信するプローブカーとして扱い、収集された情報を分析し、交通情報などを集積して社会生活上有意な情報を作り出す情報システムの基盤ソフトウェアである。今回のシステムは、情報収集に既存の情報システムをそのまま利用したところが特徴である。2004年後半に準備が開始され、2005年春から2年間にわたり2フェーズのソフト

公開デモの会場となった青山テピア

会場を出発するデモ車両

デモ車両の中で道路状況を表示

最新の情報を更新中であることを表示

IPA/SEC編：ソフトウェアエンジニアリングの実践－先進ソフトウェア開発プロジェクトの記録，翔泳社，2007より

図1　バス乗車によるデモンストレーション

※1　EPM：Empirical Project Monitor

ウェア開発、そして最後に大規模な「公開デモ」を実施し、その成果を広く一般に披露することができた。

「公開デモ」（図1）は、2007年2月22日、23日、東京青山のテピアで開催された。井上克郎大阪大学教授、桑原雅夫東京大学教授の基調講演のほか、各種講演、パネルディスカッションと展示により開発対象のプローブ情報プラットフォームソフトウェア、その開発に適用したSECの提唱するソフトウェアエンジニアリングの成果が紹介された。また、プローブ情報プラットフォームソフトウェアの出力結果を表示する機能を装備したバスを用意した。そのバスに来場者に乗車していただき、会場周辺コースを巡回しながら完成したシステムの機能を走行しながら実体験していただいた。このデモンストレーションには、甘利明経済産業大臣、渡辺博通経済産業副大臣（当時）も参加され、体験した多くの方々に最新技術の深い印象を残した。システムはこの期間、都内約8,500台の対象車両のうち、そのとき稼動していた車両からの位置情報を収集し、旅行時間等の有意な情報を提供し完動した。

② プロジェクト計測の特徴

プローブ情報プラットフォームソフトウェアを開発したソフトウェアエンジニアリング技術研究組合（COSE：Consortium for Software Engineering、理事長：山下徹NTTデータ代表取締役副社長（当時））は7社（トヨタ自動車、デンソー、NTTデータ、日本電気、日立製作所、富士通、松下電器産業）で構成され、ソフトウェア開発は、広域に分散した各社の拠点で行われ、協調領域と競争領域を峻別して進められた。協調領域の情報は皆で共有し、競争領域の情報は社間では共有されなかった。この方式は複数企業による研究開発において、公開が難しい各社の最新技術を持ち寄り、成果として最先端のシステムを実現しようという新しい試みである。社間で共有されない情報としては、ソースコード、プログラム設計書及び詳細設計書の大部分、ソフトウェア生産性に関わる情報等が含まれる。この原則は全体のプロジェクトマネージャ（PM）を努める企業にも適用され、ソフトウェアエンジニアリングの視点からは透明度の低い難しいプロジェクトマネジメントが強いられることとなった。開発対象はLinuxサーバ上でRDBを使用するC++アプリケーション・ソフトウェアと、情報表示用のPC上のソフトウェアである。ソフトウェア開発はコンポーネントに分解され、各研究組合企業が分担して開発、社間結合試験以降は、各社が開発したソフトウェアを1つの環境に持ち寄り全体を組み立てた。公開デモに提供された情報もこの環境によって作成されたものである。

こうしたプロジェクトに対し、SECは連携するEASEプロジェクト※2と共同で、いくつかのプロジェクト計測・分析グループを構成し、次のような総合的な計測を実現した。

（1）EPMによる計測と分析

EPMを用いて、開発プロセスとプロダクトの基本情報を取得した。EPMは開発支援環境の構成管理システムや障害追跡システム、メール管理システムからデータを自動収集し、例えばプログラム行数の推移、累積障害件数の推移、未解決障害件数の推移、平均障害滞留時間の推移、社間メール件数の推移、これらとソースコードの更新タイミング等の情報をグラフにより可視化して提供することができた。

（2）レビュー記録の収集と拡張EPM（EPM-Pro＊（スター）※3）による測定と分析

専用の電子フォームを用いて、基本設計と詳細設計のレビュー記録情報を収集した。またEPMで収集蓄積した情報を活用して、レビュー記録の分析、ファイル更新に関する様々な分析等、構成管理システムに蓄積された情報に関する高度な分析や障害分析を試みた。

（3）CCFinder／Geminiによるコードクローン検出と分析

コードクローンとは、ソースコード中の類似するコード片のことである。このコードクローンの分布状況、含有率等からプロダクトの性質を推し量ることができる。

※2　EASEプロジェクト：文部科学省のリーディングプロジェクトe-Society基盤ソフトウェアの総合開発の一環として奈良先端科学技術大学院大学、大阪大学を核に産業界からNTTソフトウェア、日立製作所、日立システムアンドサービス、日立公共システムエンジニアリング、SRA先端技術研究所が参加して進められた産学連携のソフトウェア工学の研究・開発プロジェクト（代表：奈良先端科学技術大学院大学鳥居宏次学長（発足時））。定量データに基づくソフトウェア工学、エンピリカルソフトウェア工学の発展を目指し、Empirical Approach to Software Engineering：EASEと名づけられた。2003年からの5年計画で2007年度が最終年度となる

※3　EPM-Pro＊：先進プロジェクトでは、EPMによる分析に加えて、EPMで収集したデータを基にした様々な分析が試みられた。EASEプロジェクトでは、これらの分析を効率よく行うためのツールとしてEPM-Pro＊を開発した。

先進プロジェクトでは、ソースコードの提供を受け、CCFinder及びGeminiツールを用いて、ソースコードからコードクローンの含有状況、含有率の分析を実施した。ソースコードへのコードクローンの含有状況は、各種のクローン関係メトリクスと合わせて散布図と呼ばれる鳥瞰図で可視化された。

(4) ベンチマーク・データベースと協調フィルタリングツールの応用による分析

SECが収集している約400項目のプロジェクトデータ（プロジェクトの属性や工程毎の工数等のデータ）と同じ項目のデータを収集した。この項目のデータの集計結果は、ソフトウェア開発データ白書としてSECより毎年出版されている[SEC2007-2]。今回のプロジェクトでは、基本設計終了時に、プロジェクト全体の計画値とそこまでの実績値を集め、さらにプロジェクト終了時に全体の実績値を収集した。

EASEプロジェクトで開発した、欠損のあるデータセットから類似のデータを抽出し、抽出されたデータを使って見積もりを行う協調フィルタリング技術を応用したツールMagiを用いて、基本設計終了時点のプロジェクトデータから、SECが収集した約1,000プロジェクトの過去データを参照し、類似のプロジェクトデータを探し出し、そのデータを使って全体工数を予測する試みを行った。

(5) チェックシートを用いたリーダへのヒアリング調査

経済産業省とSECでは、プロジェクトの進捗や潜在するリスクの見えにくいソフトウェアプロジェクトの可視化を図るために「プロジェクト見える化」部会を組織して研究を進めてきた。その成果として、要求定義や基本設計等の上流工程と、製造工程を中心とした下流工程のそれぞれに、プロジェクトのリスク、進行中の状況を浮き彫りにするためのチェックシートを作成した。約30項目からなる自己評価シートと、約80項目からなる第三者の専門家による2時間程度のヒアリングを想定したヒアリングシートである[SEC2006][SEC2007-2]。今回COSE各社のリーダ、サブリーダを対象にこのチェックシートによる自己評価とヒアリングを実施した。この調査を通して、プロジェクトに潜むリスクや管理状況など、EPMのような自動収集ツールでは収集できない様々な情報を得ることができた。

(6) スキルデータの収集と分析

ITスキル標準ITSSに沿ってソフトウェア開発関係者60人のスキル診断を行い、開発チームごとに集計してスキル分布を分析した。

(7) プロジェクト会議への継続参加による情報の収集と分析への反映

プロジェクトの進捗に関する各種の情報を得るために、SECの研究員が各種のプロジェクト会議に参加し、自動収集では得られない情報を取得した。また、必要に応じて工程の区切りで開催された品質評価会議にEASEプロジェクトの大学側の研究員も参加した。

(8) 分析結果のリアルタイム・フィードバック

EPMへ入力するデータは、週次で情報の発生元にて収集され、直接SECへ集められた。SECではEASEプロジェクトと連携してデータを分析、グラフ等に可視化し、COSE各社と全体PMにフィードバックした。ただし、フィードバックには社間で共有されていない情報が入らないように配慮した。フィードバックは面談を伴う場合と資料送付のみの場合があった。

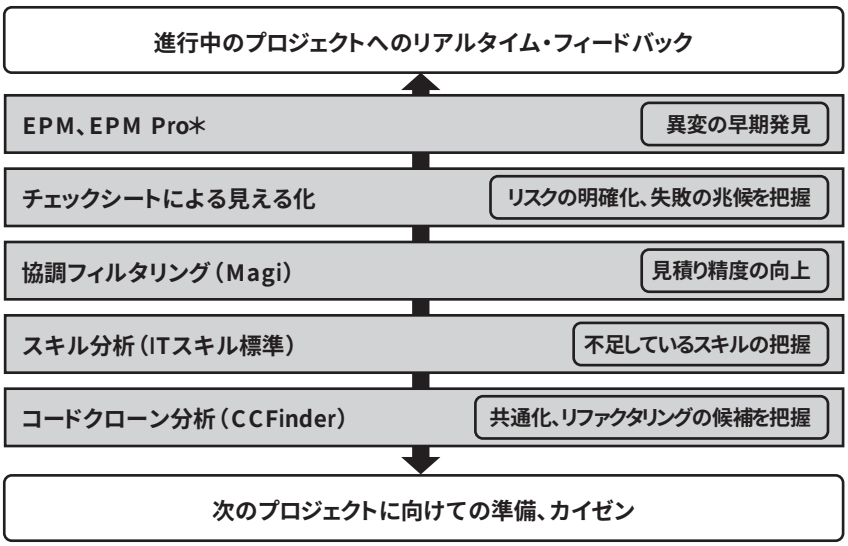


図2　分析結果のフィードバック

2005年8月から翌年3月末までフェーズ1の9ヵ月間に主なフィードバックイベントが25回あり、分析グループからCOSE各社に提供されたグラフ等の分析シートは合計1,000シート（1シート1グラフ）を超えた。2年目のフェーズ2では、1年目の成果を反映してデータ収集とフィードバックのサイクルを週次とし、フィードバックは2006年7月から翌年の2月までの8ヵ月間で、提供された分析シートが2,400シートに達した。このほかに数百枚の質問票や指摘事項一覧、これらへの回答がやり取りされた（PMには各社分がほぼ全量が提供されたので、フィードバックドキュメントの量は物理的にはこの2倍となった）。

③ プロジェクト計測の効果

SECではこうした大規模なプロジェクト計測に対して、いわばプロジェクトに密着した形でその効果を観察した。ここでは具体的に次の4つの視点からの観察結果について述べる。

① プロジェクト全体の進行を俯瞰した観察と評価
② リーダ，サブリーダへのアンケート
③ プロジェクト参加者のプロジェクト前後の振る舞い
④ プロジェクト終了後、キーマンからの本音の聴取

3.1 プロジェクト全体の観察

全期間を通してのプロジェクト観察から、例えば次のようなことが認められた。

① ソースコード行数の推移、障害件数の推移を追うことで、各社の開発の進捗状況、規模感を手にとるように俯瞰することができた。
② ソースコードのコードクローンを分析することで、ソースコードの素性、開発グループの特性を推し量ることができた。スクラッチで開発されたソースコード、大規模な流用部分を持つソースコード、経験の浅い要員によるソースコード、ソースコード全体の保守性に関する懸念事項、リファクタリングの状況等、コードクローン含有という視点からソースコードの素性を知ることができた。
③ 構成管理システムの情報からファイル更新状況を様々な角度で分析することで、開発の推移を推し量ることができた。順調なウォータフォール型の作業による開発、カットアンドトライのある開発、ファイル更新の安定度、製造工程以降での設計変更や障害検出等のインパクト等手にとるように見ることができた。
④ 障害分析により、障害の要因や計測した各種の要因と障害の関係を分析できた。特に、障害の混入工程等の分析は工程品質の評価に役立った。
⑤ レビュー記録の分析から、レビューに取り組む各社の姿勢の粗密が見えた。ウォータフォール型工程を組みレビューを重視している企業と、レビューよりも試験を重視している企業の差が見えた。
⑥ チェックシートを用いた各企業のリーダへのヒアリングにより、各企業内の開発体制に関する情報を得ることができた。これらの情報はプロジェクトのコンテキスト情報として、ツールで自動計測される情報の分析結果を解釈するときに非常に役立った。

これらの情報をPMに示すことで、PMは各企業の進捗状況など、プロジェクトマネジメントの基礎的な情報を得ることができた。

3.2 アンケート調査

開発担当各社は、1社1～2グループで開発を分担した。計測とフィードバックの効果を直接開発グループに問うアンケートがフェーズ1の開発終了後に、全体のPMと各グループのリーダ、サブリーダ等5名程度、合計26名に行われた。この回答では圧倒的多数が、フィードバックの効果を5段階評価で「有効」、「少しは有効」と評価した。しかしながら、この調査はプロジェクト途中の公式的なものだったので評価が甘くなっている可能性も考えられる。

3.3 関係者の振る舞い

計測とフィードバックに対して、実際にソフトウェアを開発する各利害関係者がどのような振る舞いをしたかを観察することは、計測とフィードバックの有用性を判断するバロメータとなる。

当初開発を担当する各グループは、プロジェクトへの擾乱が無ければ計画立案者の意向に沿って学術的な研究に協力するという姿勢であり、計測とフィードバックへの期待は持っていなかった。しかしながらプロジェクトの進行にともなって、下記のように変わっていった.

① 1年目、フェーズ1での計測とフィードバックは、それなしでもプロジェクトマネジメントが進行する体制の中で、プロジェクトマネジメントに擾乱を与えないモ

ニタ型の方式で行われた。モニタしたデータの分析結果は、工程区切りの品質評価会議で開発グループに参考情報として提示された。

② 2年目、フェーズ2の開発では、フェーズ1の成果を積極的に評価し、計測とフィードバックをプロジェクトマネジメントに組み込んだ（各開発グループにとって、計測とフィードバックは当初の想定以上に効果的との印象を与えた）。具体的には計測とフィードバックの周期を週次とし、毎週の進捗会議で分析結果が確認できるようにした。工程区切りの品質評価会議では週次の計測・分析データを共有しながら、全体PMの意向に沿ったさらに詳細な分析結果を参照しながら工程評価について協議する方式となった。

③ 2年目、フェーズ2の後の公開デモ向け開発では、短期開発という事情もあり、自己申告レベルの進捗報告を廃止し、計測とフィードバックを一歩進め、分析結果を進捗報告として代用し、プロジェクトマネジメントが進められた。

プロジェクトのこの推移は、開発グループから計測とフィードバックへの一定の信頼が得られたことを意味する。

次に、プロジェクト終了後のCOSE各社の振る舞いに着目すると、SECがこのプロジェクトに引き続いて企画したEPMツール検証プロジェクトへの参加を表明した企業、手法の一部を社内管理システムに組み込んだ企業、大学とコードクローン分析に関する共同研究を実施したところもあった。

3.4 本音の聴取

筆者らはプロジェクト終了後、公式的な制約の少ない環境下でプロジェクトのキーとなる参加者から本音の評価を聞く機会を持った。

プロジェクト参加者には大別して、発注者、ターゲットシステム（この場合、プローブ情報プラットフォーム）実現の責任者、サブシステム担当企業（この場合、組合企業）のターゲットシステム実現の責任者がいる。そしてソフトウェアの開発には、全体のPMとサブシステム担当企業の開発リーダがいる。サブシステム担当の開発リーダには、本来プロジェクトマネジメントを行うことがミッションであるが、下部組織にまかせて現場を見ていないリーダ、現場を見たくても物理的に見ることのできないリーダ、現場に接し現場を把握しているリーダの3タイプがある。そしてプロジェクト参加者として実際にソフトウェアを開発する担当者がいる。今回試みた計測とフィードバック施策はこのようなプロジェクト参加者に、それぞれの役割に沿って効果があったことが認められた。ターゲットシステムに責任を持つ人々には、ソフトウェア開発に煩わされることなく、ターゲットシステムの実現に専念でき、計測結果のフィードバックについては下部組織から肯定的な評価が上がったことが歓迎された。PMをはじめ、現場を見たくても見ることのできないリーダにはEPMツールのようなプロジェクト管理系のツールが高く評価された。一方、現場に接しているリーダには個別分析ツールのいくつかが高く評価された。そして全体PMからは、フィードバック情報は均一では

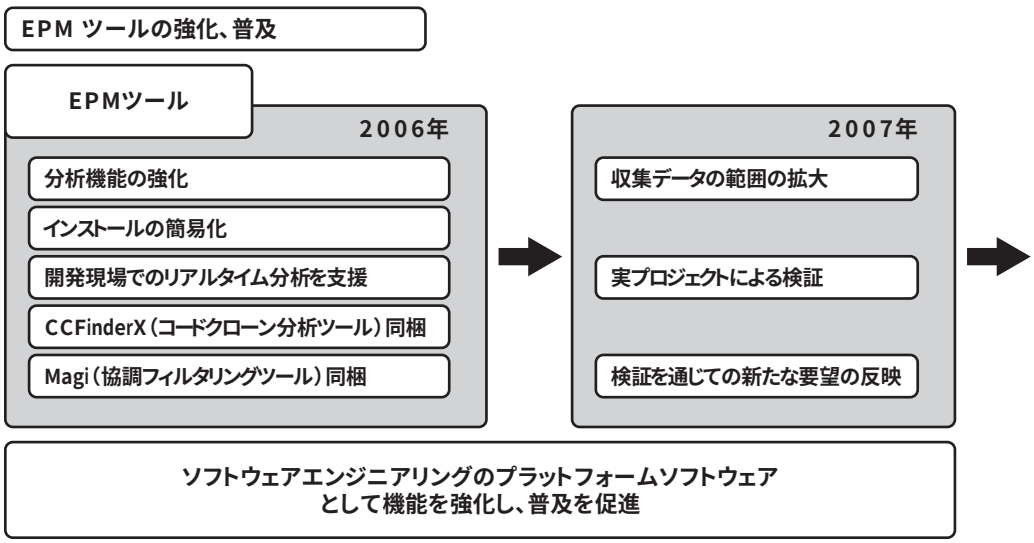


図3　IPA/SECによるEPMツールの展開

なく、情報過多にならないように、それぞれの役割に沿った選択的な提供がもとめられるという提言を得た。

④ 実験の再現性確保、ツール・手法の普及へ向けて

筆者らをはじめ本プロジェクトの計画者は、上記の経緯でこのような進行中のプロジェクト計測とフィードバックの有用性を確信した。そこで、ここで試みられた手法の再現性の確保、手法の普及を目指してツールの配布キットを作成し、その配布とこれを用いたツール検証プロジェクトを開始した（図3)。

EPMツールと名づけたツール配布キットには、次のものが含まれている。

① EPM： EASEプロジェクトがオープンソースで提供している製品に対してIPA/SECが商用レベルの品質確保をはかり、利便性を考えたいくつかの機能追加を実施したもの。
② コードクローン分析ツールCCFinderX（神谷年洋氏（産業技術総合研究所）提供）
③ 協調フィルタリングツールMagi試供版（奈良先端科学技術大学院大学提供）

SECではこれらのツール群と、SECからの書籍「ITプロジェクトの『見える化』」で提案しているソフトウェアプロジェクトのチェックシート、ITスキル標準ITSSに沿ったスキル診断を合わせて適用する「検証プロジェクト」を企画し、参加企業の募集を開始した。本論執筆時までに5回の説明会を実施し、30企業からの参加を得ている。SECでは、検証プロジェクトを今後も継続して実施し、EPMツールに対しても機能強化を行う予定である。

⑤ 総括――次のプロジェクトへ

2年余にわたる先進プロジェクトは、計測とデータを基盤とする実証的なソフトウェアエンジニアリングの分野に大きな足跡を残した。COSEは活動を終え解散したが、参加した各企業はここでの体験をもとにそれぞれ自らの資源を投入して新たなソフトウェアエンジニアリング施策への歩みを進めた。共同でデータ分析を行ったEASEプロジェクトの参加企業でありCOSEのデータ収集環境構築にも貢献したNTTソフトウェアでは、2006年春からEPMの全社導入を開始した[NTTSOFTWARE2007]。先進プロジェクトで用いられたツールと手法は、SECの手でブラッシュアップされ、普及へ向けて「EPMツール検証プロジェクト」として次の段階へ進められた。EASEプロジェクトやSECの研究員からは多数の学術論文、テクニカルレポートが発表された。中には国際会議でBest Paper Awardを受賞したものもある。収集、蓄積されたプロジェクトデータは保存されソフトウェア工学の学術研究に供される。そして、先進プロジェクトの経緯については書籍として出版される[SEC2007-3]。開発されたプローブ情報システムについては、次の事業化計画が検討されている。また経済産業省とSECでは、組込みソフトウェア領域を対象として、今回のSEC先進プロジェクトに続く実証プロジェクトを開始した。EPMをはじめとするプロジェクト計測の仕組みもこの計画に組み込まれている。

謝辞

本プロジェクトの推進に絶大なご支援、ご指導を頂いた、ソフトウェアエンジニアリング技術研究組合及び組合員各企業、EASEプロジェクトの奈良先端科学技術大学院大学、大阪大学大学院、参加各企業、そして経済産業省の各位に心より感謝の意を表します。

参考文献

[HIGUCHI2005] 樋口　登：Project Report 先進ソフトウェア開発プロジェクト，SEC journal No.2，pp.56-57，2005
[HIGUCHI2007] 樋口　登：エンピリカルソフトウェアエンジニアリングの実践、SEC Forum 2007（東京ドームシティ、プリズムホール），2007※
[MITANI2005] 神谷芳樹：産学官連携プロジェクトにおける定量データ収集・分析環境導入の実現へ「実証プロジェクト＋可視化＋定量データ　報告」，情報化月間特別行事，SECセッション（東京，全日空ホテル），2005※
[MITANI2006] 神谷芳樹：先進プロジェクト報告，情報化月間特別行事，SECセッション（東京，全日空ホテル），2006※
[MATSUURA, MITANI, HIGUCHI2006] 松浦　清，神谷芳樹，樋口　登： Project Report 先進ソフトウェア開発プロジェクトPart II，SEC journal No.5，pp.44-49，2006
[MATSUURA2006] 松浦　清：SEC先進プロジェクト，SEC Forum 2006（サンケイプラザ），（予稿）pp.84-96，2006※
[NTTSOFTWARE2007] NTTソフトウェア：NTTソフトウェア（株）におけるEPMの適用について，エンピリカルソフトウェア工学研究会，2007
http://sec.ipa.go.jp/tool/EPMcase_nttsoftware.pdf
[SEC2006] IPA SEC：ITプロジェクトの「見える化」（下流工程編），日経BP社，2006
[SEC2007-1] IPA SEC：ITプロジェクトの「見える化」（上流工程編），日経BP社，2007
[SEC2007-2] IPA SEC：ソフトウェア開発データ白書2007，日経BP社，2007
[SEC2007-3] IPA SEC：ソフトウェアエンジニアリングの実践－先進ソフトウェア開発プロジェクトの記録，翔泳社，2007

※　講演スライドは下記SEC -Webサイトからダウンロード可能です。
https://sec.ipa.go.jp/download/index.php

ソフトウェア開発見積り評価モデル CoBRA

SEC エンタプライズ系プロジェクト 研究員
高橋　茂

欧州ではすでに導入の実績があり、近年日本でも注目を集めているCoBRA※1は、ドイツのフラウンホーファー協会IESE※2において開発された見積りの評価モデルであり、COCOMO※3（Barry Boehmらによる）のように工数に影響を与える要因（コストドライバ、工数変動要因）に対して工数の見積り評価を可能とする。ただし、ソフトウェア開発の工数を単純に見積もるものではない。“Risk Assesment”ともあるように、ソフトウェア開発のリスクアセスメントの手法としても有効である。

1. はじめに

リスク分析モデルの構築では、熟練者の意見に基づいて不確実性の分布を導き適用する手法が採用されることがある。特に、得られるデータが少ない、またはデータの収集に膨大なコストがかかる等の場合、データの説明力不足を補完するために熟練者の意見を利用する。

ソフトウェア開発では、開発工数を変動させる要因の不確実性を明確にすることが難しい。CoBRAは、このようなリスク分析モデルの構築手法を取り入れ、要因の不確実性を熟練者（ここでは、ソフトウェア開発のプロジェクトマネージャ等）の意見により定義し、ソフトウェア開発工数を評価する。

2. CoBRA見積り評価モデル構築の手順

CoBRA見積り評価モデルの構築にあたっては、経験豊富な複数のプロジェクトマネージャの協力が不可欠である。CoBRAモデルは、プロジェクトマネージャとの合議（ブレーンストーミング）によって構築される。CoBRAモデルの構築には、検討すべきソフトウェア開発プロジェクトにかかわった複数（5〜6名程度）のプロジェクトマネージャと過去のプロジェクト実績データ、ソフトウェアの開発規模（SLOC、FP、画面数等）とその開発工数のデータが10件程度あれば十分である。以下ではCoBRAのモデル構築及びシミュレーションの方法について紹介する。

2.1　工数変動要因の洗い出し

CoBRAモデルにおける工数の算定は次式に従う。

$$（工数）＝\alpha \times（規模）\times（1＋\sum_i CO_i）\qquad（式1）$$

この式は、基本的に工数はソフトウェアの開発規模に比例するが、それに加えて複数の工数変動要因の増加率（ここではCO_i）により、工数が増加するということを示している。

CoBRAモデルの構築にあたっては、まずどのような工数変動要因があるのかを検討する必要がある。ここで過去のプロジェクトの実績をもとに、工数を増加させる（または減少させる）と考えられる要因について、プロジェクトマネージャからの意見を収集する。そして意見を

※1　CoBRA：Cost Estimation, Benchmarking, and Risk Assessment
※2　IESE：Institute for Experimental Software Engineering、実験的ソフトウェア研究所
※3　COCOMO：COnstructive COst MOdel

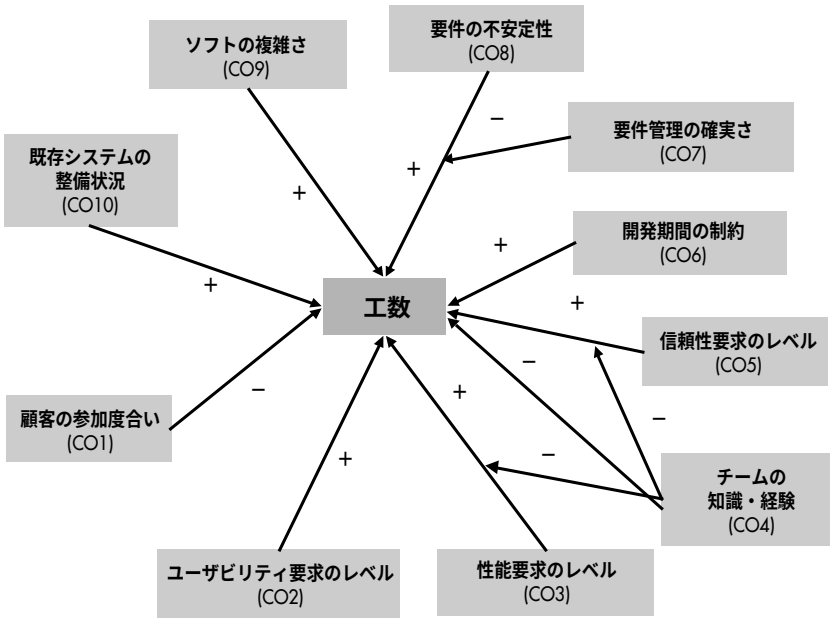


図１　工数変動要因の例

もとに、図１のような工数変動要因の関係図を作成することが望ましい。

工数変動要因の検討に際しては、互いに類似した要因は1つにまとめる必要がある。ただし要因間の相互作用のモデル化は可能である。例えば、図１の“要件の不安定性”と“要件管理の確実さ”の２つの要因のようにモデル化すればよい。

工数変動要因は、開発工数を増加させる場合または減少させる場合がある。図１においては、各要因が工数を増大させる場合は（＋）、減少させる場合は（－）で表している。CoBRAでは、工数の減少要因の扱いは注意を要する。詳細は次節で述べる。

2.2　工数変動要因の影響度合いの定量化

次に、2.1項において挙げられた工数変動要因に対して、具体的に意味するところを明らかにする（工数変動要因の定義）。

CoBRAでは、各工数変動要因に対して４段階のレベルを定義している。４段階は等差になるように定義する必要がある。レベル3とは、工数変動要因が最悪であった場合、レベル0は工数変動要因が最良であった場合を示している。

工数変動要因として「チームの知識・経験」があった場合、それぞれのレベルを例えば次のように定義すればよい。

・レベル3　全員がはじめての業務。
・レベル2　リーダに知識・経験がない、メンバに知識・経験がある。
・レベル1　リーダのみ知識経験がある。
・レベル0　リーダ及びメンバに知識・経験がある。

工数変動要因がレベル0であるとは、工数の増加はない（CO＝0％）ということである。工数変動要因がレベル3（つまり最悪）の場合のCOを三角分布として与える。三角分布の形状は、3点（最大値、最頻値、最小値）により表現することができる。この場合、プロジェクトマネージャは、工数変動要因に対しての工数増加割合（％）を最大値（工数の増加割合の最大と考えられる値）、最頻値（工数の増加割合の最もありそうと考えられる値）、最小値（工数の増加割合の最小と考えられる値）として示す必要がある。

ここで工数を減少させる要因の扱いであるが、減少要因はすべて増加要因に変換しなければならない。工数を減少させる最良の場合を最悪の場合として定義しなおす（おそらく、この場合は最良の反対の意味の定義となる）必要がある。そのため、工数はベースラインであるα×（規模）を下回ることはないことに注意しなければならない。このようにCoBRAでは、工数変動要因はすべて増加要因として扱う。これは工数が負になるというような、

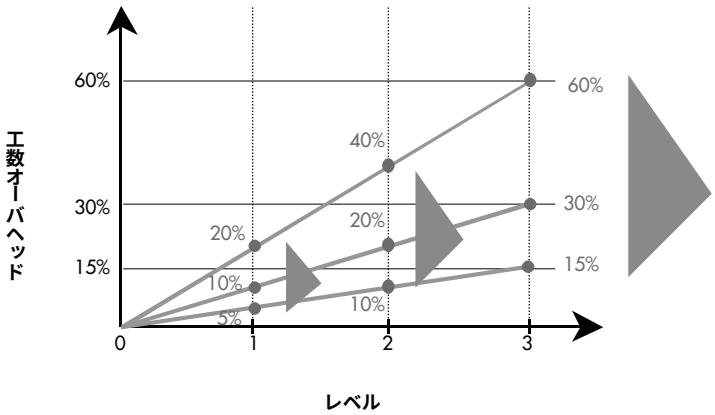


図2　三角分布の決め方

意味をなさない結果が導出されるのを避けるための制限である。

CoBRAで利用する三角分布はノンパラメトリック分布であり、熟練者の意見を容易に確率分布として示すことができる。熟練者の意見を反映する確率分布としては、三角分布の他、パラメトリック分布であるPERT分布（ベータ分布の一形態）等が利用されることもある。ただし、CoBRAでは、PERT分布の利用は特に推奨されてはいない。

三角分布はレベル3（最悪の場合）のみの定義をすれば十分で、レベル2～レベル0までは、

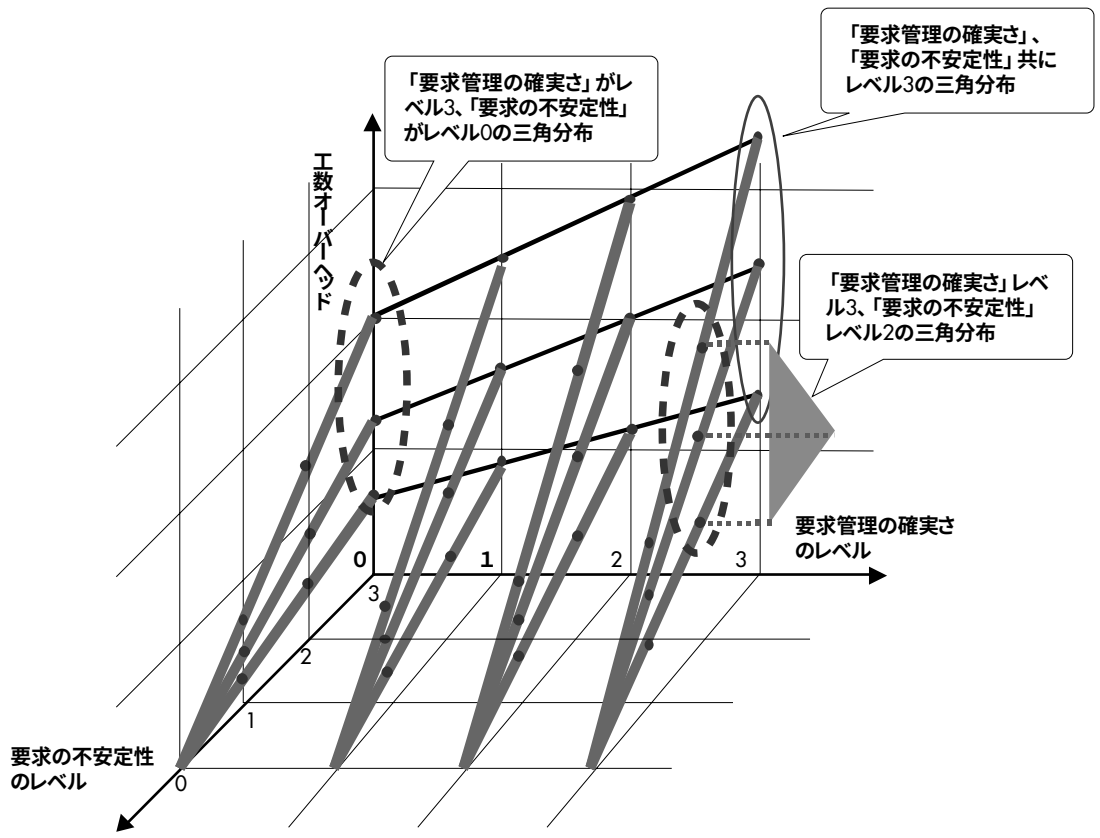


図3　三角分布の決め方（工数変動要因にインタラクションがある場合）

図２のような関係から、自動的に三角分布の形状は決定される。レベル0は、工数を変動させないので、図2における原点となる。レベル1、2の三角分布の3つの点は、レベル3の三角分布の3つの値をそれぞれ1/3、2/3とすることで決まる。

工数変動要因間にインタラクションがある場合（例えば図1の「要求の不安定性（CO8）」と「要求管理の確実さ（CO7）」の2つの工数変動要因の関係）の三角分布の決め方は、図3に示すとおりである。プロジェクトマネージャに確認すべき三角分布は、“「要求管理の確実さ」、「要求の不安定性」が共にレベル3 ”の場合と“「要求管理の確実さ」がレベル3、「要求の不安定性」がレベル0 ”の場合のみとなる。その他の場合の三角分布は、「要求管理の確実さ」及び「要求の不安定性」のレベルごとに、図3のように与えられる。

2.3　シミュレーションによる生産性ベースラインの算出

収集したプロジェクトのデータ（開発工数と開発規模）から、2.2項で与えられた三角分布を利用して、モンテカルロシミュレーションにより生産性ベースライン、すなわちαを求める。

収集したプロジェクトごとに各工数変動要因のレベルをプロジェクトマネージャ及びプロジェクト関係者へアンケートを実施することで明らかにする。レベルごとに、

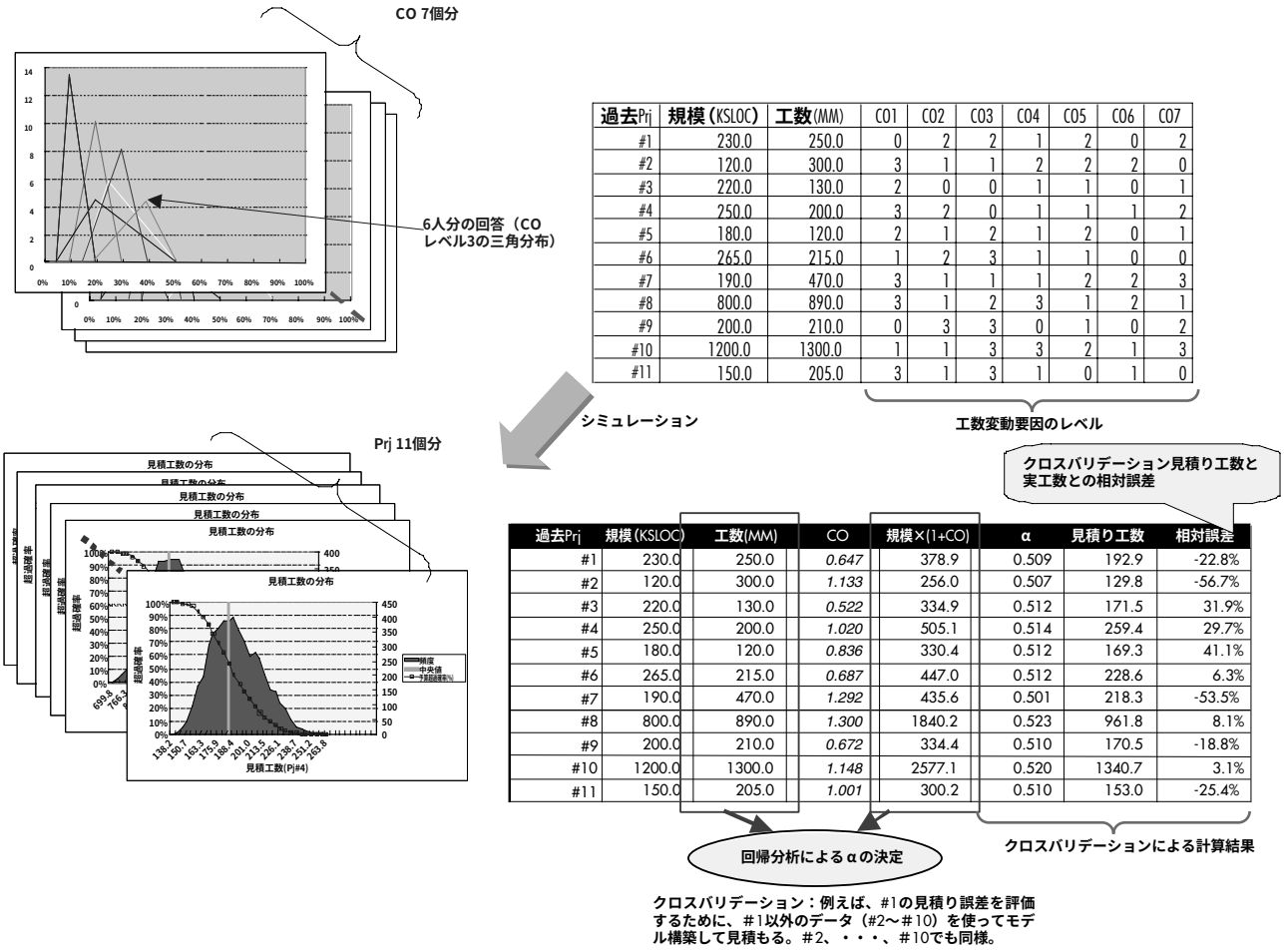


図４　モンテカルロシミュレーション

2.2により与えられた複数の三角分布を使って、式１のモンテカルロシミュレーションを行う。乱数により、各工数変動要因の三角分布に従うCO_iを発生させる。

2.2項で検討した三角分布は、複数のプロジェクトマネージャにより与えられているので、複数存在することになる。したがって、実際にモンテカルロシミュレーションをする場合には、複数ある三角分布に対して、次のような３つの処理方法があり得るだろう。

・すべての三角分布の3つの点の平均をとった、平均の三角分布を利用する（シミュレーションで利用する三角分布は１つ）。
・クラスタリング等を活用して、はずれた三角分布を排除し、残りの三角分布の平均をとる（シミュレーションで利用する三角分布は１つ）。
・すべての三角分布を活用する（シミュレーションに際して、三角分布の選択も乱数で実施する。すべての三角分布を利用してシミュレーションを実施する）。

次に、モンテカルロシミュレーションにより得られた$1+\sum_i CO_i$の分布の平均値（または最頻値）と規模を掛け合わせた値と、実際のプロジェクトの工数から回帰分析により生産性ベースラインαを求める（図4）。αを決定するに際して、規模と工数のデータの有効性を確認するためには、クロスバリデーション（交差検定法）を実施するとよい。クロスバリデーションは、N組のデータが与えられた場合、$N-1$組のデータでモデルを構築して残りの1組のデータを推計する方法である。そして推計によるデータと残りの1組のデータの誤差を評価する。

クロスバリデーションによるシミュレーションデータと実データの誤差が大きい場合には全体モデルについて再検討が必要とされる。この場合以下のような点に留意する。

・工数、規模データの再考。
・工数変動要因の定義は正しいか。曖昧な定義となっていないか。
・抜けている工数変動要因はないか。または余分な工数変動要因はないか。
・工数変動要因の定義は正しいか。
・三角分布の見直し。
・特異的なプロジェクトは存在していないか。場合によっては特異なプロジェクトはモデル化対象外としなければばならない。

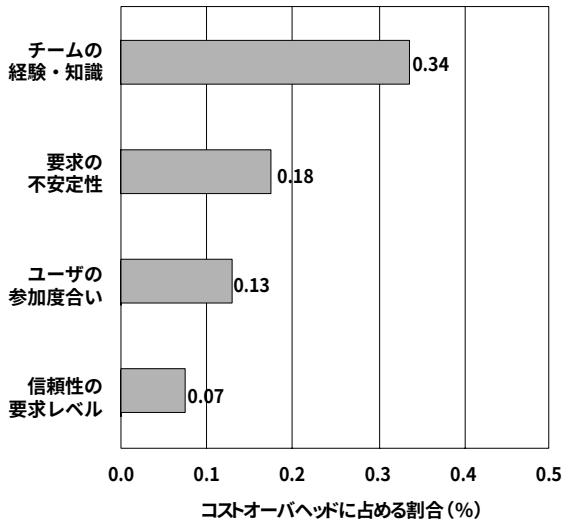


図５　トルネードチャート

2.4　工数見積り評価とリスクアセスメント

2.3項までで過去プロジェクト実績からのCoBRAモデルの構築は完了となる。

構築されたCoBRAモデルにより、新規プロジェクトの見積り評価を実施する場合には、まず新規プロジェクトに対する工数変動要因のレベルを定義する。その後は、2.3項と同様のモンテカルロシミュレーションを実施することで想定される工数の確率分布が得られるであろう。図5のようなトルネードチャートを使えば、シミュレーションにより得られる確率分布に対して影響を与える工

※４　@RISK：リスク分析ソフトウェア
※５　Crystal Ball：リスク分析、モンテカルロシミュレーションソフトウェア

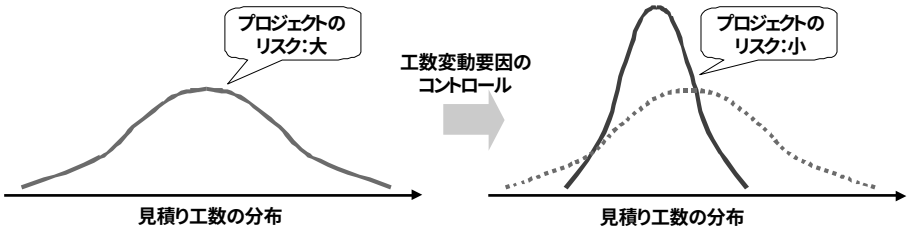


図6　リスクコントロール

数変動要因を確認することができる。分析者は、このようなチャートを活用することで、当該プロジェクトのリスクアセスメントを実施することが可能となる。トルネードチャートは@RISK※4やCrystal Ball※5のようなシミュレーションツールでは、標準的な機能として利用することができる。シミュレーションツールでは、定義された確率分布の乱数を発生させるための関数も実装されているため、ツールを用いれば、CoBRAのシミュレーションは容易である。

3. CoBRA見積り評価モデルの活用

CoBRA見積り評価モデルの利用にあたって注意しなければならないことは、本モデルにより計算された結果は、あくまでモンテカルロシミュレーションによるものであるため、工数見積り値そのものを示すものではないということである。

シミュレーションにより、見積り工数の分布が得られた場合、その分布の形状からプロジェクトに対するリスクの評価が可能になる。分布の裾野が拡がっている（分散が大きい）ときには、そのプロジェクトのリスクは高いといえる。トルネードチャート等から判断された影響の大きな工数変動要因をコントロールすることで、プロジェクトのリスクを軽減できる可能性がある（図6）。

このように、CoBRA見積り評価モデルの利用者は、シミュレーション結果から次のような判断が可能となるので、ぜひ取り組んでいただきたい。

・プロジェクト工数に対するリスクが評価可能となる。したがって、見積り価格の再考、もしくは、プロジェクトの実施判断の意思決定ができる。
・プロジェクトに大きく影響する工数変動要因を明らかにできる。工数変動要因がコントロール可能であればプロジェクトのリスクを軽減できる可能性がある。

先に述べたようにCoBRAで利用するプロジェクトのデータは、多くは必要とされない。プロジェクトマネージャ等の熟練者による経験によりモデル化が実施される。したがって、プロジェクトの工数見積りにこれまであまり積極的に取り組んでこなかった組織でも、少数のデータさえあればモデル構築が可能である。

すでにCOCOMO等の具体的な見積りモデルを採用している組織では、本手法と比較してみることで新しい知見が得られるかもしれない。CoBRAの強みはモデル構築の自由度が大きいことであり、そのため精度を犠牲にしている側面もある。他の見積り手法との比較は、構築されたCoBRA見積り評価モデルの説明力を強化する助けとなるであろう。

また、工数変動要因のモデル化（図1）はブレインストーミングで行われるので、組織における強みまたは弱みを再確認するための場を提供するであろう。このような場で得られた情報を組織内で共有することで、ソフトウェア開発プロジェクトの改善活動へとつなげるという効果も期待される。

参考文献

[BRIAND1998]Lionel C. Briand,etc:COBRA:A Hybrid Method for Software Cost Estimation,Bechmarking,and Risk Assessment,ICSE'98,1998

長野県での組込みシステムへの取り組み
～産業活性化と技術者育成～

http://www.asatech.or.jp/

財団法人 長野県テクノ財団　浅間テクノポリス地域センター　テクノコーディネーター
清水　信孟

組込みシステムの技術分野では、デジタル機器製品がますます発展し、ソフトウェアの複雑化、大規模化により技術者の需要が急増、世界的に人材不足が深刻となっている。また、生産性の向上、信頼性、安全性等も緊迫の課題となっている。

本稿では、長野県内における各種団体、大学等の組込みシステム技術に関する「産業活性化と人材の育成対応」の取り組み、及び当地域センターの「長野県組込みシステムコンソーシアム」の概要について紹介する。

1 県内情報サービス業の状況

長野県「平成17年特定サービス産業実態調査」によると、事業所数129、就業者数4,676人、年間売上高714億2,200万円となっており、事業所数前年比△6.5％、就業者数△10.0％、年間売上高△4.0％で、いずれも減少している。

年間売上高を業務種類別にみると「受注ソフトウェア開発」が最も多く45.7％、続いて情報処理サービス29.0％となっている。なお、長野県「平成17年工業統計調査結果報告書」によれば、長野県内の総数は事業所数11,585、従業員221,692人、製造品出荷額等6兆3,177億6,174万円となっている。

また、「社団法人 長野県情報サービス振興協会（NISA）」は昭和60年に設立され、情報サービス産業と他産業との連携強化、人材育成、調査研究等を通じ、県内情報サービス産業の健全な発展を図り、産業の情報化を推進し、地域社会の活性化を進めている。

なお、特に人材育成には力を入れており、長野県産業大学校講座の一環として、NISA学園を開催し、平成19年度は、システム開発管理研修、システム開発研修を実施する。さらに県内の一層の情報化を推進するため、他産業、大学、県等との交流及び連携を図り、県内外の情報関連団体、県内業界団体、県等で本会の趣旨に合致する諸事業に対し、共催及び後援等を行っている。

2 主な産業振興施策と組織

(1) 長野県産業振興戦略プラン
～メイド・インNAGANOを世界へ～

この産業振興戦略プランは、長野県経済の回復の遅れや経済・社会情勢の変化を踏まえ、長野県経済の再生と持続的発展、県民の豊かな生活の実現を目指し、全国に誇る加工組立型産業の集積や、豊富な地域資源等、従来から備え持つ潜在力を最大限に発揮させ、力強い長野県経済を構築するための戦略プランである。平成19年3月に策定、平成23年度までの5年間活動を行う。

また、基本戦略として以下の４点を策定し、展開している。

・産業集積戦略
　産学連携による信州型スーパークラスターの形成
　地域資源活用型産業の創出
・マーケティング戦略
・サポート戦略
・人材育成戦略

加えて、下記8テーマを重点プロジェクトとしてこれに取り組み、プロジェクトを実施する「長野県産業振興戦略会議」（仮称）を平成19年度から設置する。これらの施策により、「世界市場へ飛躍する長野県産業の構築」を目指している。

・学官連携とナノテク・材料活用支援センター（仮称）

・地域資源活用製品開発支援センター（仮称）と基金組成
・マーケティング支援センター（仮称）
・工業技術総合センター設備の拡充強化
・企業誘致強化プログラム
・投資型ファンドの展開
・中核企業の育成と産産連携
・産業人材育成強化プログラム

(2) 長野県工業技術総合センター

長野県工業技術総合センターは、長野県商工部の現地機関であり、県内製造業の発展に寄与するための試験研究機関の役を担っている。材料技術部門、精密・電子技術部門、情報技術部門、食品技術部門、技術連携部門、総務部門で組織されている。

情報技術部門の活動について補足すると、コンピュータ利用技術、デザイン技術、繊維・木工技術等に関する研究開発、技術相談、人材育成、依頼試験、施設利用を通じ、中小企業の技術の高度化、情報化の推進を図っている。また、サポートする技術分野としては、情報システム部、通信基盤部、人間生活科学部があり、ネットワーク技術、セキュリティ技術、ソフトウェア開発技術、超高周波通信技術、通信回路設計技術、ファームウェア開発技術、組込みシステム開発技術、デザイン技術、福祉工学技術、繊維技術、木製品技術等の研究開発を行っている。

長野県の紹介

長野県は本州の中央部に位置し、東西約120km、南北約212kmと南北に長く、面積は約13,600平方km、人口は約220万人である。

●工業製品全国一の例

①抵抗器の出荷額

精密機械から小型電子部品製造へシフトしていくなかで発展し、現在では様々な形状、機能、構造の抵抗器を製造。平成16年の抵抗器の出荷額は327億7,700万円で全国一。

②小型モータ（3W未満）の出荷額

精密技術を生かし発展し、携帯電話のバイブレーター、デジタルカメラの駆動部分等に使用されている。平成16年の小型モータ（3W未満）の出荷額は698億7,000万円。

③顕微鏡・拡大鏡の出荷額

製糸工業を基盤とし、豊かな水を利用した精密機械工業として発展。現在ではデジタル技術、レーザーを使用した顕微鏡も開発。平成16年の顕微鏡・拡大鏡の出荷額は323億9,900万円で全国一。

④眼鏡レンズ（コンタクトレンズ含む）の出荷額

国産初のプラスチック眼鏡レンズを開発する等、光学、精密加工技術を生かし発展した。平成16年における眼鏡レンズの出荷額は139億8,000万円で全国一。

⑤ギター（電気ギター含む）の出荷額

日照時間が長く、湿度が低い気候を利用し、ギター生産が盛んであり、近年ではエレキギターの生産も多い。平成16年のギターの出荷額は31億5,100万円で全国一。

⑥縫針・ミシン針の出荷額

戦時下、長野県への工場疎開を契機に、ミシン、縫製技術の発展により、多様な縫針・ミシン針の生産が行われている。平成16年の縫針・ミシン針の出荷額は52億8,300万円で全国一。

（http://www.pref.nagano.jp/より）

また、県内産業の中核的技術支援拠点として、中小製造業等の技術的ニーズに対応し、地域経済の活性化、豊かな地域社会の形成への貢献が期待されている。

3 組込みシステム（ソフトウェア）への取り組み

(1) 信州大学の取り組み―組込みシステム技術者育成コースの開設

信州大学工学部は、大学院に社会人向けの「高度ものづくり専門職コース」の開設を平成17年に発表し、超加工精密、組込みシステム技術、精密機器制御の分野を諏訪市、塩尻市、飯田市で開講することで産学連携、地域連携で準備を進めてきた。

準備が整い、平成19年4月より、大学院工学系研究科情報工学専攻内に新たに組込みシステム技術者育成コースが開設された。このコースは、長野県塩尻市が設置した塩尻インキュベーションプラザ（SIP）施設内において、同施設に入る多くの組込みシステム関連企業、塩尻市、経団連等の多くの協力のもとに実施している。最先端の組込みシステム設計開発技術者を育成することを目的とし、徹底した基礎技術の習得と、多くの組込みシステム関連企業と連携し、これら企業への長期インターンシップ教育が特徴である（図１）。

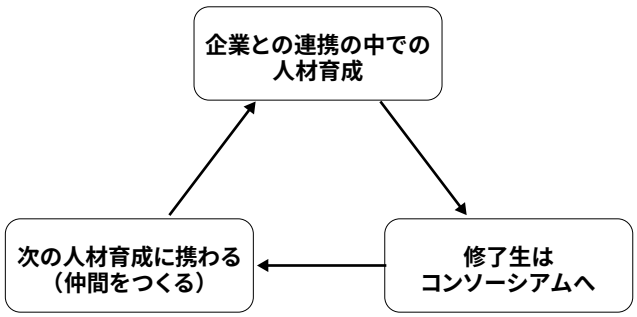


図１　人材育成連携サイクル

(2) 塩尻インキュベーションプラザ（SIP）の取り組み

塩尻インキュベーションプラザ（SIP）は、全国的、恒常的に人材不足とされている組込みソフトウェア技術者の育成を図り、その人材輩出とともに起業を支援し、そ

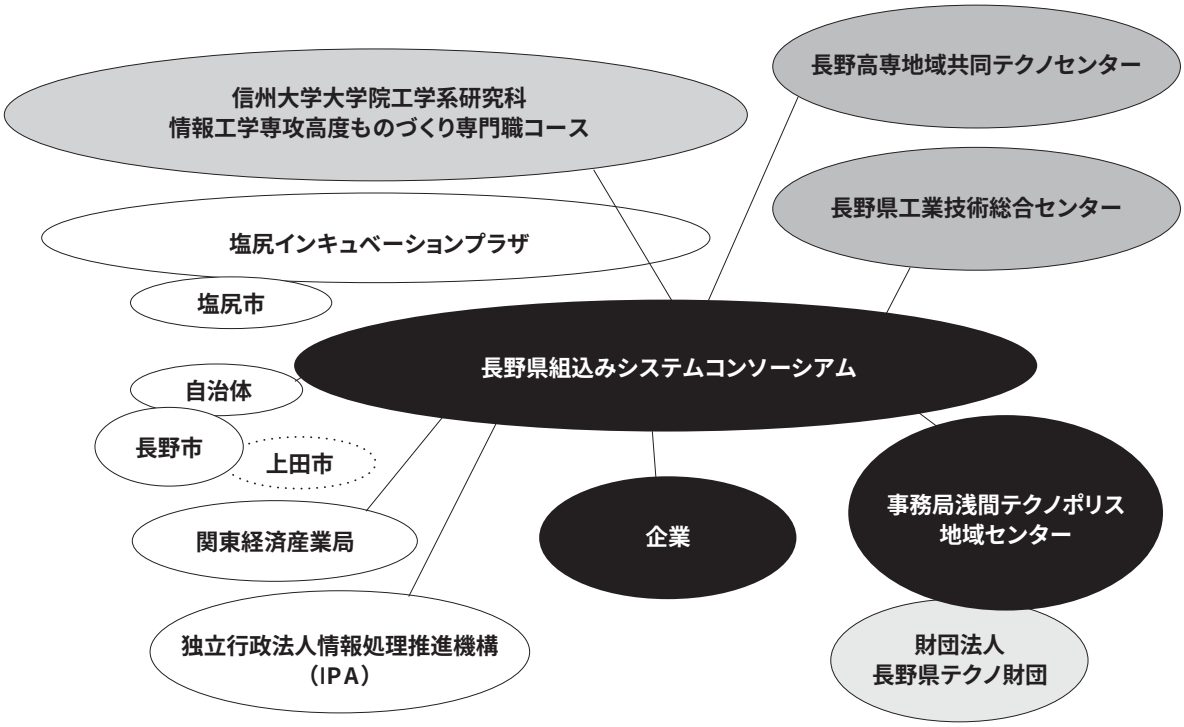


図2　連 携 体 制

の集積による地域の活性化に寄与することを目的とした活動を推進するものである（詳細は組織紹介を参照）。

(3) 財団法人 長野県テクノ財団の取り組み

財団法人長野県テクノハイランド開発機構と財団法人浅間テクノポリス開発機構の二つの財団を母体に、研究開発事業を核として、産学官交流や人材育成等の支援事業を一貫して行う機関として、2001年4月1日設立に設立された。

長野県内における地域産業資源を活用しつつ、技術革新による地域産業の高度化と産業創出を促進し、地域経済の活性化と自立化に資すること目的としている。

また、財団の地域センターとして、善光寺バレー地域センター（長野市）、浅間テクノポリス地域センター（上田市）、アルプスハイランド地域センター（松本市）、諏訪テクノレイクサイド地域センター（諏訪市）、伊那テクノバレー地域センター（伊那市）を設置した。

(4) 浅間テクノポリス地域センターの取り組み

浅間テクノポリス地域における地域産業資源を活用しつつ、技術革新による地域産業の高度化と産業創出を促進し、もって地域経済の活性化と自立化に資することを目的として、平成13年4月1日に設立された。

(5) 長野県組込みシステムコンソーシアムの取り組み

組込みシステム技術者の育成、技術力の向上、産業の育成を図るため、図2のような連携体制により平成19年1月31日に設立された（図3）。

事業内容は、

- 信州大学大学院と連携した組込みシステム技術者の創出への協力（将来の技術者の育成）
- 信州大学大学院、長野工業高等専門学校等を活用した組込み技術者のスキルアップ（現役技術者のスキルアップ）
- 最新組込みシステム技術・管理セミナーの開催
- 経済産業省、IPA等との連携支援

図3　セミナー風景（設立記念講演会）

- 個々の企業だけでは受注しにくい案件獲得について、共同研究や業務獲得支援の実施
- テーマを決めてWG（分科会）の開設

　本年度は「機能安全分科会」、「開発技法分科会」、「組込み開発のマネジメント」を開催し、セミナー研修、会員相互での研究等でレベルアップを図る計画で進めている。

- 展示会への出展

　ET 2007（組込み総合技術展）への出展

現在の会員数は企業会員47社80名、研究会員（大学の先生等）５名、賛助会員２自治体となっている。

今後は県内外の同種団体、大学等との交流や組込みシステム技術のセミナー、イベント等を企画し、地域産業の活性化の期待に応えていきたい。

参考文献

- 長野県の統計情報　平成17年特定サービス実態調査：http://www3.pref.nagano.jp/
- 長野県情報サービス振興協会：http://www.nisa.or.jp/
- 長野県産業振興戦略プラン：http://www.pref.nagano.jp/syoukou/sinkou/plantop.htm
- 長野県工業技術総合センター：http://www.nagano-it.go.jp/
- 信州大学工学部組込みシステム技術者育成コース：http://www.sugcess.jp/xoops/
- 塩尻インキュベーションプラザ：http://www.s-sip.jp/
- 財団法人長野県テクノ財団：http://www.tech.or.jp/
- 浅間テクノポリス地域センター：http://www.asatech.or.jp/index.html
- 長野県のプロフィール：http://www.pref.nagano.jp/profile/profile.htm

塩尻インキュベーションプラザ　S·I·P

組込みソフトウェア産業の集積地を目指して

http://www.s-sip.jp/

塩尻市　経済事業部商工課　主事
太田　幸一

塩尻インキュベーションプラザ（SIP）は、組込みシステム産業に特化したベンチャー企業のインキュベーション施設として平成19年1月より本格的稼働を開始した。組込みシステム技術の「知の集積」拠点を目指して人材育成やアライアンスの促進等を実施している。

1　塩尻市の組込みシステム産業振興施策

塩尻市では、中長期産業振興計画として定めた「塩尻市産業振興ビジョン」※1に基づき、情報関連産業分野に焦点を当てた施策を実施している。なかでも、基盤技術として重要度が極めて高く、ビジネスチャンスの拡大が見込まれる“組込みシステム産業”を、今後の塩尻市における中核産業として育てたい考えである。

特に、深刻な課題となっている“人材不足”に着目し、高度組込みシステムエンジニアを育成すべく、市と包括連携協定を締結している信州大学及び長野工業高等専門学校。それぞれ実施する人材育成プログラムを中心とした事業を展開し、業界が抱える課題の解決と、地域に根ざした産業発展を目指している。

塩尻市はまた、経済産業省が推進する産業クラスター計画の1つ「地域活性化プロジェクト」（中央自動車道沿線地域ネットワーク活動）における拠点地域にも位置づけられており、当該産業クラスターにおけるものづくり産業の機能補完を担うべく、基盤技術である組込みシステム産業を中心に据えたネットワークの形成を図っている。

2　SIPにおける産学官連携

SIPの設置にあたっては、企画の当初よりエプソンアヴァシス株式会社、信州大学、そして塩尻市による強力な産学官連携体制のもとで事業を推進してきた。平成18年度経済産業省新事業支援施設整備費補助金に採択され、同年12月に施設が完成、翌19年1月より稼働を開始した。SIPは、民間企業のオフィス、信州大学による大学院、市が運営するビジネスインキュベータといった異なる三者が共存する特徴的なビジネスインキュベーション施設となっている。

現在SIPのオフィスには、1都4県より集まった13の組込みシステム開発企業及び同分野での起業を目指すアントレプレナーが入居している。入居対象業種を組込みシ

図1　塩尻インキュベーションプラザ外観

図2　信州大学大学院の授業風景

※1　塩尻市産業振興ビジョン：平成9年「第一次産業振興ビジョン」を策定。現在は、平成17年の「第二次塩尻市産業振興ビジョン」による産業振興を実施中。詳しくは塩尻市のホームページhttp://www.city.shiojiri.nagano.jp/ctg/440035/440035.htmlを参照。

ステム産業に限定しており、入居者間のアライアンスやコラボレーションが期待されている。業界最先端での経験と、専門的な知識を有するインキュベーションマネージャのサポートにより、「組込みソフトウェア産業の集積地」を目指している。

3　SIPにおける人材育成

平成19年4月より、信州大学大学院工学系研究科情報工学専攻による組込みシステム技術者育成コースがSIPに開設された。同コースは、主に社会人を対象としたリカレント教育の場として、ソフトウェア・ハードウェア双方の知識と技術の習得できるカリキュラムとなっており、卒業者には修士号が与えられる（図2)。

同コースの特徴の1つは、技術者に不足しがちな「ドキュメンテーション能力」「コミュニケーション能力」「プレゼンテーション能力」といった“人間力”の強化に力点を置いていることである。その上で、より実践的な教育を実現するため、以下のような点にも配慮がなされている。

・チーム演習や長期インターンシップ等を取り入れた実践的カリキュラム
・SIP入居企業との連携
・企業における実際プロジェクトへの参加等

もう1つの特徴が、人材育成を通じた地域活性化を促進するための仕組みが構築されている点である。同コースによって育成された組込みシステム技術者が地域企業への就職（または地域での起業)、地域で活躍し、横断的なネットワークないしコミュニティを形成し、次世代技術者育成に参加還元するという中長期的な人材育成・交流のポジティブ・スパイラルを構築することによって、継続的な地域経済の発展につなげていきたいと考える。

さらに、平成19年第3四半期には長野工業高等専門学校によるエントリレベルの技術者を対象とした組込みエンジニア育成コースの実施が決定したことにより、上記信州大学大学院と合わせて、初等・中等段階から社会人まで継続的で、しかも各スキル水準に対応した組込みエンジニア育成プラットフォームが完成しつつあることもSIPの特徴である。

4　今後の事業計画

SIPを拠点とした産業振興を実施することにより、「組込みのことは塩尻に聞け」「塩尻に行けば組込みで何かがあるだろう」といわれるような組込みシステム産業の集積地となることがSIPの最終目標である（図3)。

組込みシステム産業による集積地を創り出すことによって、人や企業の集積と交流を促進し、地方自治体の重要な命題である“地域の活性化”に繋げていきたいと考えている。

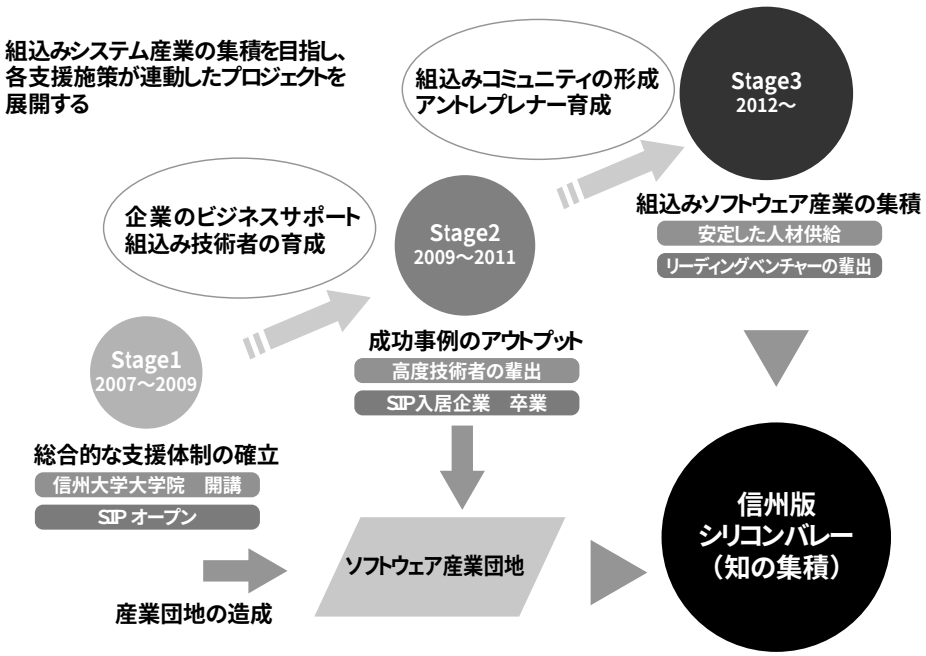


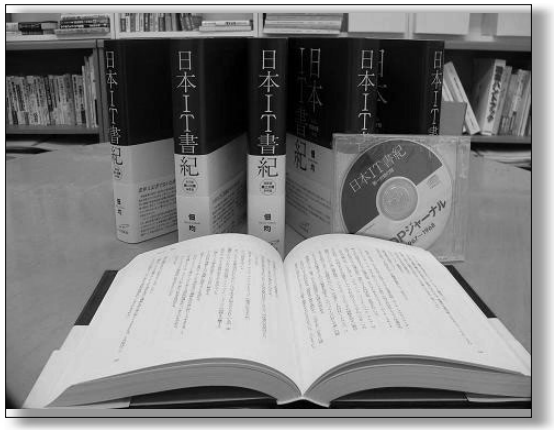
図3　SIP中期ビジョン

BOOK REVIEW

日本IT書紀（第五分冊）

佃 均　著

ナレイ株式会社出版局　刊
A5判・5冊組・3,016頁・定価31,500円（税込）・2005年3月刊



世代を超えて共有したい歴史と感動

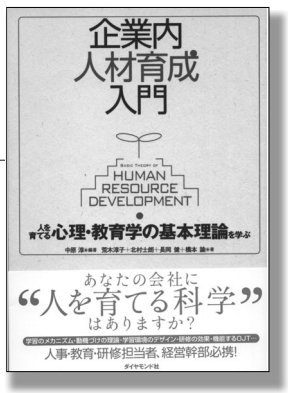
全巻3,016頁、凄い本である。とてもはじめからは読めないので第五分冊から始めた。のっけから、日比谷公会堂での全共闘、大菩薩峠、浅間山荘、そして市ヶ谷台での三島由紀夫と息もつかせぬ、そして極めて重い序章を経てようやくIPA誕生に至る。通産省情振課発足も情報処理技術者試験開始もそのあとだ。まさに1970年代の現代史、その前史から始まる。為替レートは1ドル360円、のちにこれが崩れる。本当にこんな時代があったのか、と信じられないような状況である。同時代を生きなかった若者たちには到底理解も想像もできないだろう。若い人に読んでほしい、読むだけでなくここに伝えられていることを語り合ってほしい。若者たちだけでは無理だから語れる世代と一緒に話し合ってほしいものだ。現代社会はこの時代に起きたことの痕跡、起こしたことの帰結で満たされているのだから。この時代を走った筆者には読んでいて涙の場面が2つあった。Amdahl機とMシリーズ機の誰もが予想もしなかった栄光の場面、そしてDIPS計画の到達点で締めくくられる感動の終章である。

（神谷芳樹）

企業内人材育成入門

中原 淳　編著　荒木 淵子・北村 士朗・長岡 健・橋本 諭　共著

ISBN：978-4-47844055-1　ダイヤモンド社刊
A5判・369頁・定価2,940円（税込）・2006年10月刊



企業内人材育成

ソフトウェア技術者が、人材育成担当者になることが多く見受けられる。ソフトウェア開発の人材育成には、現場感・勘どころがわかるソフトウェア開発経験者は最適な人材である。しかし、企業の人材育成施策に参加した経験を持ったとしても、人材育成に関してのプロフェッショナルではない場合が多い。

そんな人材育成担当者は、既存の人材育成施策を引き継ぐか、自分の経験に基づく人材育成を企画・提供してしまってはいないだろうか。

この書籍は、企業における人材育成の担当者に対して、最低限の心理学や教育学のセオリーを基礎から提供してくれる良書である。ソフトウェア開発の専門的な知識に加え、心理学と教育学の知識も備えることで、より効果的なソフトウェア開発に関する人材育成が提供可能となるだろう。

学習のメカニズムやモデルから始まり、OJTや研修デザイン、動機付けや教育の政治力学まで紹介されている。企業の人材育成に関する書籍というと、市場で活躍する企業の企業文化に基づく取り組みや、複数の成功事例紹介といったものが多く見受けられる。

ソフトウェア工学もセオリーを理解し、その上で工夫や改善をすることが開発力強化の近道である。ソフトウェア開発に関する人材育成も同じく、人材育成のセオリーを理解することが必要不可欠である。

（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

開催年月	開催日	イベント名	主　催	開催場所	URL
10月	1（月）	情報化月間記念式典特別行事	IPA	東京都港区・ANAインターコンチネンタルホテル東京	http://www.ipa.go.jp/
	10（水）～12（金）	九州・国際テクノフェア（ICTコンバージェンス2007）	九州・国際テクノフェア実行委員会	福岡県北九州市・西日本総合展示場新館	http://www.it-kyushu.net/
	18（木）～20（土）	ESS2007（組込みシステムシンポジウム2007）	社団法人 情報処理学会 組込みシステム研究会	東京都江東区・日本科学未来館	http://www.ertl.jp/ESS2007/
	26（金）	【SEC主催セミナー】ESCR	IPA/SEC	東京都文京区・文京グリーンコート	http://sec.ipa.go.jp/
	30（火）	IPA Forum 2007／SEC コンファレンス（SEC journal論文発表会）	IPA/SEC	東京都港区・明治記念館	http://www.ipa.go.jp/
	31（水）～11月2（金）	SPI Japan 2007（ソフトウェアプロセス改善カンファレンス）	日本SPIコンソーシアム	富山県富山市・富山国際会議場	http://www.jaspic.jp/
11月	14（水）～16（金）	Embedded Technology 2007／組込み総合技術展	社団法人 組込みシステム技術協会（JASA）	神奈川県横浜市・パシフィコ横浜	http://www.jasa.or.jp/et/
	20（火）	【SEC主催セミナー】プロセス改善	IPA/SEC	東京都渋谷区・伊藤塾5号館　法学館ビル	http://sec.ipa.go.jp/
	20（火）	【SEC主催セミナー】共通フレーム2007　SLCP	IPA/SEC	東京都渋谷区・伊藤塾5号館　法学館ビル	http://sec.ipa.go.jp/
	22（木）	【SEC主催セミナー】ETSS	IPA/SEC	東京都文京区・文京グリーンコート	http://sec.ipa.go.jp/
	28（水）	SEC設立三周年記念成果報告会	IPA/SEC	東京都目黒区・ウェスティンホテル東京	http://sec.ipa.go.jp/
12月	12（水）～14（金）	TRONSHOW 2008	T-Engineフォーラム、社団法人トロン協会	東京都千代田区・東京国際フォーラム	http://www.tronshow.org/
	14（金）	【SEC主催セミナー】ESPR	IPA/SEC	東京都文京区・文京グリーンコート	http://sec.ipa.go.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。

イベントのご報告

【展示会】

●「第10回組込みシステム開発技術展（ESEC）」

開催日：5月16日（水）～18日（金）
会　場：東京ビッグサイト

IPAは31小間という大きなブースを設け、SECはIPAの「未踏ソフトウェア創造事業」や「中小ITベンチャー支援事業」で支援を受けて起業した開発者23社と共に出展いたしました。ブース内セミナー及びパネル展示とデモを行い、3日間合計で4,000名以上の方にブースにご来場いただきました。

●「Embedded Technology West 2007／組込み総合技術展 関西」

開催日：6月6日（水）～7日（木）
会　場：マイドームおおさか

SECセッションとして「SEC2007年度成果プレビュー」及び「地域活用セミナー」を行い、それぞれ76名、67名の方にご参加いただきました。ブースにはPCを設置してSEC-Webサイトの利用者登録をお勧めし、初日72名、2日目188名の方にご登録いただきました。また、パネル展示とEPMツールのデモを行い、SECの活動について広報いたしました。

●IPAX「SEC Forum 2007 ～SECの成果の普及・実証に向けて～」

開催日：6月28日（木）～29日（金）
会　場：東京ドームシティ　プリズムホール

初日に組込み系活動報告、2日目にエンタプライズ系活動報告を実施し、それぞれ211名、460名にご参加いただきました。2日目には特別講演として東京証券取引所代表取締役会長の西室泰三様にご講演いただきました。

【SEC主催セミナー】

◆「プロセス改善セミナー（大阪）」

開催日　：5月22日（火）
会　場　：グランキューブ大阪（大阪国際会議場）
参加者数：48名

◆「プロセス改善セミナー（名古屋）」

開催日　：6月5日（火）
会　場　：名古屋市工業研究所　ホール
参加者数：143名

◆「第1回 ソフトウェア開発データ白書の読み方と活用」
◆「第2回 ソフトウェア開発データ白書の読み方と活用」

開催日　：9月4日（火）
会　場　：文京グリーンコート センターオフィス
参加者数：第1回99名、第2回93名

◆SPLC（第11回ソフトウェアプロダクトライン国際会議）併設「SECワークショップ」

開催日　：9月13日（木）
会　場　：京都リサーチパーク
参加者数：30名

・イベント時の配布資料、講演資料等、詳しいことはSEC-Webサイト（http://sec.ipa.go.jp/）をご覧ください。
・SEC-Webサイトにて「利用者登録」し、「SECからのお知らせを受け取る」を選択していただきますと、ソフトウェア・エンジニアリング関連のイベント情報をメールでお届けいたします。どうぞご利用ください。

編集後記

SEC journal 10号において掲載を見送らせていただいた「SEC先進ソフトウェア開発」の成果を本号で掲載することができてほっとしています。先進ソフトウェア開発の担当研究員は2名しかいないのですが、経済産業省への成果報告書作成だけでなく、大掛かりな実証実験の結果を開発者の方々が参考にできるように書籍執筆も並行して行ってまいりました。成果は、書籍「ソフトウェアエンジニアリングの実践～先進ソフトウェア開発プロジェクトの記録～」と「エンピリカルソフトウェアエンジニアリングの勧め」にまとめました。

また、前号にて、“ 企業の開発現場の方が投稿しやすい形として、「ソフトウェアエンジニアリング活用事例レポート」（仮称）の募集を検討開始いたしました。 ”とお知らせいたしましたが、「ソフトウェアエンジニアリング　ベストプラクティス賞」として募集を行わせていただきました。受賞者の発表は、SEC設立3周年記念成果報告会（11月28日）にて行う予定です。参考になる事例紹介ができる予定ですので、是非ご参加頂きたくお願いいたします。

なお、SEC journalは『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とした実証論文』の掲載を主目的として発行し、2号から編集長の大役を任された私は、約3年も継続いたしました（我ながらよく続いたと思います）。12号から私より数段知的でダンディな新編集長にバトンをタッチし、新たに事例等も掲載するNEW SEC journalがお目見えいたします。ご期待ください。

本journalに対してのご意見とSECへの情報提供・問題提起等には、SEC-Webサイト（http://sec.ipa.go.jp/）内の「SECへのお問い合わせ」をご利用ください。

（ヒゲ）

SEC journal 編集委員会

SEC journal® 第3巻第3号（通巻11号） 2007年9月28日発行

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

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、
下記の内容で論文を募集します。

応募様式は、下記のURLをご覧ください。
http://sec.ipa.go.jp/secjournal/oubo.php

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

論文賞

「SEC journal」では、毎年「SEC journal」論文賞を発表しております（前回は2006年10月24日SECコンファレンス）。受賞対象は、「SEC journal」掲載論文他投稿をいただいた論文です（論文賞は最優秀賞、優秀賞、SEC所長賞からなり、それぞれ副賞賞金100万円、50万円、20万円）。

応募要項

スケジュール

・13号（2008年1月発行）
　応募締切　2007年10月22日
　投稿締切　2007年10月29日
　採否通知　2007年11月12日

・14号（2008年4月発行）
　応募締切　2008年1月22日
　投稿締切　2008年1月29日
　採否通知　2008年2月12日

採録決定後、1週間程度のカメラレディ期間があります。
詳細は別途通知されます。
採録の場合には「SEC journal」への掲載およびSEC-Webサイトやイベント等で発表します。
詳しくはSEC-Webサイトよりお問い合わせください。

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-ronbun@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。

SEC journalバックナンバーのご案内

詳しくはhttp://sec.ipa.go.jp/secjournal/をご覧ください。

No.1　No.2　No.3　No.4　No.5　No.6　No.7　No.8　No.9　No.10

SEC journal No.11
第3巻第3号（通巻11号）
2007年9月28日発行

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

IPA®
独立行政法人 情報処理推進機構

ISSN 1349-8622