2008年1月15日発行
第3巻第4号（通巻12号）
ISSN 1349-8622

SEC journal 12

Software Engineering Center

SEC journal 論文賞受賞論文

組込みソフトウェア設計検証へのモデル検査技術の適用と考察

岸 知二，金井 勇人

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.12目次

SEC第1期の総括と次期フェーズにおけるゴール明確化

経済産業省 商務情報政策局
情報処理振興課長
八尋　俊英

SEC第1期の総括

ソフトウェア・エンジニアリング・センター（SEC）は、創設以降3年が経過し、これまでソフトウェアエンジニアリングの世界において、EPMやSLCP、ETSSをはじめとしてエンタープライズ系、組込み系分野において多くの成果を挙げてこられました。その過程で、産学官から英知を結集し、民間企業各社の非公開情報から共有可能なものを抽出してまとめることにより、ガイドラインや標準などの公共財を作り上げたという意味で、民間企業ではなし得ない独立行政法人ならではの貴重な役割を担われてきたのではないでしょうか。さらに、こうした成果を通して、産学連携の場として、産業界、学会等から認知され、国内外の代表的な拠点との関係を確立し、我が国ソフトウェアエンジニアリングの代表的な機関としての地位を獲得されたと思います。

立ち上げの素晴しい活躍は高い評価を受けて然るべきですが、第2期は目指すべきゴールを地球視野でどう設定するのか、難しい時期にきております。

システム開発における最近の課題

最近の情報システム・組込みシステムにおけるトラブル多発を背景に、その安全・安心確保への要請はますます増大し、これに伴い、システム開発の各段階において安全・安心を確保し、システムトラブル等を最低限に抑えることができるソフトウェアエンジニアリング手法の確立・普及が急務となっています。

また、サービス業等のエンドユーザや自動車、携帯電話、情報家電等の製造業の産業競争力は、ますますソフトウェアに依存してきているにもかかわらず、品質・コスト・納期に関し計画通りのものは半分程度といわれております。このため、ソフトウェア開発の生産性を向上させることができるソフトウェアエンジニアリング手法の確立・普及も急務となっています。

SEC次期フェーズ

こうした課題を解決するため、SECには、我が国ソフトウェアエンジニアリングの代表的な機関として、まず、世界最高水準のシステムの安全・安心を確保するため、信頼性ガイドライン、信頼性評価指標、機能安全に関する国際標準に基づいたシステム開発手法等の確立・普及や、SLCP等これまでの開発成果の国際標準化に向けた取り組みを期待しております。

一方で、地球市場においては、よりグローバルに活動する企業が国境を越えて活動し、一国の国民も、自国の開発手法で提供されるべきかどうかよりも、より安心できる世界スタンダードを求めていることを厳しく見つめる必要があります。

また、我が国のソフトウェア開発の生産性を向上させるため、モジュール化やSOA等、新しい技術の活用を通した高生産性手法の確立・普及や、ESxR、ETSS等これまでのSECにおける成果の民間開発現場への適用を期待しております。

今後は、SECの成果について、普及状況及びシステム開発への適用効果を可能な限り定量的に把握・分析することにより、民間企業の経営層や開発現場へのさらなる普及に努めていただきたいと思いますが、限られた時間・リソースの中でプライオリティをつけることも大切です。

ソフトウェアエンジニアリングの中核的機関として、我が国経済社会の基盤を担うシステム全体の信頼性・生産性の向上につながるだけでなく、世界規模で自らのポジションを捉え直す時期にきているSECの更なる飛躍を心から期待しております。

所長対談　繁野 高仁　株式会社情報システム総研　代表取締役社長

情報の源泉を押さえ、情報体系を管理することがIT部門のミッション

企業が事業を展開する上で、情報システムの重要性が認識されているが、必ずしもその活用が成功を収めているとはいえない。日本企業のCIOは、どこから、そしてどのような情報システム活用にまつわる問題を解決すべきか。KDDIでCIOを務め、現在は企業の情報システムの構築と活用に関するコンサルティングを行っている繁野高仁・情報システム総研社長とIPA/SECの鶴保征城所長が語り合った。

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　繁野さんは、KDDIのCIOとして腕を振るわれ、現在は企業のIT活用に関するコンサルティングをされていますね。企業のIT部門のミッションにはどうようなものがあると考えられていますか。

繁野　私は、IT部門として実行すべき主要なミッションは3つあると考えています。第1は情報体系を管理することです。これが、企業のIT活用のベースとして欠かせません。第2は、情報基盤の整備です。第3は、情報とITを使って社内の問題を解決することです。

鶴保　ミッションの１番目に挙げられた、企業における情報体系の管理とはどのようなことを指すのですか。

繁野　最も大事なことは、情報の源泉をきちんと押さえることです。企業には、情報があふれかえっていますが、その大元をたどると結構シンプルなのです。しかし、多くの企業では情報の源泉がきちんと押さえられていない、管理されていないことに問題があるのです。情報の源泉とは、ビジネス現場の現物、現実です。製造業の例で言えば、工場の設備や製品の仕様、製品を製造するための方法などが該当するでしょう。ほとんどの企業でそうした情報が部門毎あるいは属人的になっていて、体系的に管理されていません。現場の情報を的確に把握できていないと、製造業では、工場に対して生産可能なオーダーを出すことができないため、需要の変動に対応することが難しくなります。工場を多く持つ大企業ほど問題は深刻です。

鶴保　大企業でもそのような状況なのですか。工場の実態が本社では見えていないのですね。

繁野　そうなのです。工場のKPI（評価指標）は従来からの製品を大量に生産することで評価されるということが続いていて、多品種少量生産という時代に、まだまだ合致していないのです。そのために、欠品や在庫過剰が日常的に起きているという状況があります。コンピュータというのは、実世界のシミュレータです。地球シミュレータは、コンピュータ上に自然のデータを取り込み、自然を再現しようとしています。ビジネスも同じで、ビジネスに関する情報をデータとしてコンピュータに取り込み、ビジネスの事実が見えるようにします。そのためには、ビジネスの基本的な要素（要のもの・こと）を捉えなければいけません。

鶴保　そのあたりのことを舵取りすることがCIOやIT部門の役割ですね。

繁野　そうです。CIOはまず情報体系の管理を行い、ビジネスモデルや業務プロセスを情報という観点からおさえていくことがCIOの役割です。米国のCIOは、初めにインフラとしてのITの管理に取り組み、その次に業務プロセスの管理へと進みました。しかし、それだけでは限界があると認識され、情報の源泉をきちんと押さえようという流れになっています。多くの日本企業では、情報の源泉が押さえられておらず、また、ビジネスモデルがしっかり構築されていません。それなのに、現場に業務プロセスばかり議論させているというのが実態です。しかし、

考え方を変えれば、業務プロセスがきちんとできていない日本企業においても、今から情報体系の管理に真剣に取り組めば、欧米企業の上にいける可能性があります。それから、業務プロセスの改革を進めればいいのです。

鶴保　情報体系を整理することからはじめて、ビジネスモデルと業務プロセスを組み立てていくと日本の企業は元気になりますか。

繁野　絶対によくなると思います。逆に、そうしないと無秩序に産出したデータに絡め取られて身動きできなくなるのではという心配をしています。

鶴保　ビジネスはうまく回っているけれど、情報システムがついていけないという会社も多く見受けられます。

繁野　そうなのです。それで、欧米のパッケージソフトを入れてさらに傷口を広げてしまっているケースが結構あります。パッケージソフトを利用するとき、本来は、そのパッケージソフトが対象としているビジネスの基本的な要素（要のもの・こと）を見た上で、自社に合うかどうかを判断しなければいけないと思うのですが、そういうレベルで判断できるパッケージソフトはありません。逆にいえば、そういうソフトウェアを日本で作り出せれば、日本は世界に冠たるソフトウェア立国になれる可能性があると思っています。

工場の現場のシミュレーションを行うソフトウェアを製品化したい

繁野　顧客ごとの要求にきめ細かく対応できることが日本の製造業の強さです。ところが、大量生産を基本とした海外製ERPパッケージを無理矢理、製造現場に押しつけるという悲劇があちこちに起こっています。そうではなく、個別対応ができる柔軟な工場の能力を失うことなく、大量生産を効率よくできる仕組みをめざしたいではないですか。そのためには、複雑な工場のシミュレーションができるソフトウェアがいるし、実世界の情報構造をモデリングして情報体系として管理する必要があるのです。

鶴保　海外の企業はトップダウンでパッケージソフトの導入を進めていますね。

繁野　多くの日本企業が海外に工場を持っていますが、面白いのは、工場と本社の関係が日本と米国では正反対なのです。日本は、本社から的確な指示は出せないけれども、工場の現場で

繁野　高仁（しげの　たかひと）
北海道大学工学部電気工学科卒業後8年間、日本NCRのSEとして大手百貨店向け全店POSシステムの開発とインストレーションに従事。1985年、DDI（現KDDI）の創業に参画し、DDI、DDIポケット（現ウィルコム）、KDDIにおいて情報システム部門を立ち上げ統括する。KDDIの合併対応およびシステム構造改革では、日経コンピュータ誌「情報システム大賞」グランプリを受賞。一昨年、KDDI執行役員情報システム本部長を退任して株式会社情報システム総研を設立。現在、通信、製造、金融など幅広く大手企業の情報システム部門を支援している。情報処理学会、経営情報学会会員。経済産業省「CIO戦略フォーラム」委員。

調整して生産をうまくやっています。米国では、日本よりも的確な指示を本社から出しますが、現場がその通り生産できないのです。

鶴保　大きな計画は本社が指示を出し、細かいところは現場の創意工夫を生かす。それがいちばんいいですよね。

繁野　そうです。全社的に押さえるべきところと、現場で自由にできるところが両立するような柔軟なシステム構造にしなければいけないと考えています。実は我々は、今、製造業のアドバンスト・プランニング・アンド・スケジューリング（APS）という分野の先進的なソフトウェアを日本発で開発しようと取り組んでいます。そのソフトウェアが行うのは工場の現場のシミュレーションです。工場の設備や製品の仕様、生産方法を詳細に納めたデータベースを構築し、生産要求をオーダーネットワークに展開して工場の生産活動をシミュレーションすると、生産可能なオーダーかどうかが検証できます。工場で行うことをシミュレーションできるようになると、本社側から適切な指示が行えるようになります。

鶴保　そのソフトウェアができると、トヨタがしているような生産管理が他の企業でもできるようになるのですか。

繁野　どこでもというと言い過ぎかもしれませんが、中小企業でも、トヨタのようなことができるレベルのものを提供したいと考えています。最近のソフトウェア産業は若い人に人気が無いようですが、我々が情報構造を捉える新しいアプローチで先進的なソフトウェアをつくり、実績を上げることによって、こうするともっといいシステムできるとか、火だるまになるようなプロジェクトが少なくなる、若い人たちが夢を持てる職場に

なるといったことを現実に見える形で示したいと思っています。

鶴保　それは、ものづくりをしている他のITベンダにとっても価値あることですね。

繁野　中長期でソフトウェア産業の将来を考えている人にとってはきわめて意味ある方向だと思います。

SECは2008年度からの2期事業で情報システムの設計を対象に実証実験を行う

鶴保　システムの設計に関しては、機能中心設計とデータ中心設計とがありますね。情報の源泉を押さえるということは、システム設計においては、データ中心設計の方がいいという意味ですか。

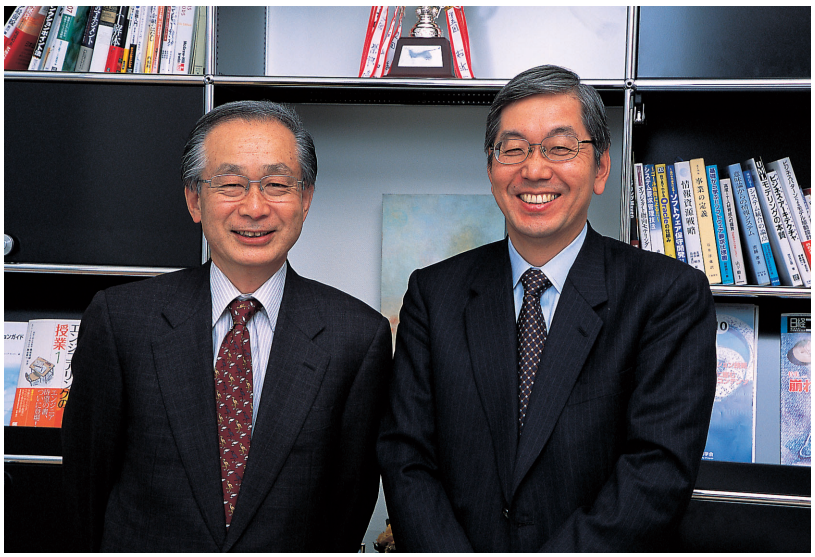
繁野　もちろんそうです。ただし、データはコミュニケーションの媒体にすぎません。データにのっている情報の意味を捉えること、中でもビジネスの基本的な要素（要のもの・こと）を捉えることが大切だと考えています。私がコンサルティングをしている企業では、要の「もの」や「こと」を捉えることから議論し、そこで描いたモデルからデータベースとトランザクションの設計ができるという言い方をしています。そういうアプローチをとると安定した構造のシステムを構築することができますし、同時にビジネス改革や業務改革を議論できるようになるのです。

鶴保　企業は既に部門単位等で情報システムを持っていますね。企業が全体として情報が管理できるようになるためには、そこに横串を通さないといけないのですが、そのときに、既存のシステムを生かして横串で結ぶのがいいものか、それとも抜本的に再構築したほうがいいものか。その点については、どう考えられますか。

繁野　大手企業の情報システムを一度に作り変えることは不可能です。ですから、きちんと情報体系を整理するとともに、情報基盤の青写真を描きながらシステム移行のシナリオを作って何年もかけて整備していくしかないと考えています。

鶴保　今、産業界ではM&Aが盛んに行われています。米国では、通信会社の合従連衡も進んでいますが、それぞれ各社が持っていた情報システムを統合することは容易ではないでしょう。米国でM&Aや合併がうまくいっているのは、情報システムに対する何が、日本と違うのでしょうか。

繁野　米企業は、情報システム部門がトップダウンで全体のアーキテクチャをしっかり議論し、2〜3年の計画を立てて情報システムの統合を進めています。米企業にはそれを可能とする



人材がいるのです。日本には、トップダウンでアーキテクチャをデザインできる人はほとんどいません。経営者と話ができるレベルでシステムのことがわかっている人材がいないので、そこは真空地帯になっています。それができる人材はITベンダの人材とは違います。

鶴保　情報システムのユーザである企業がしっかりしないといけないわけですが、日本の場合は、そのユーザがしっかりしていないという状況があります。

繁野　私は、数少ない情報システム部門の優秀な人材を流動化して、会社の垣根を越えて適材適所に供給される状況を作りたいと思っています。

鶴保　ビジネスがうまくいっている会社ほどITに投資をし、情報システムは肥大化するものです。そのため、CIOには投資や効果のバランス感覚が求められると思います。海外企業のCIOはしっかりしているのですか。

繁野　しっかりしています。CIOは情報システムのプロフェッショナルだという意識があるからだと思います。

鶴保　日本では、情報システムを作ること自体が評価されるという感覚があって、開発した情報システムがどのようにリターンに結びついたのかという意識が薄いように感じられます。

繁野　きちんとビジネスモデルを考えて、そのビジネスをするためにシステムを役立てるという議論をしなければいけないのに、ビジネスモデルがほとんど語られていないためでしょう。また、IT部門はシステムを構築するときに、社内のユーザから要求を聞くことから始めようとするのですが、細かい要求を聞くのは後回しにして、基本的なビジネス構造を示す要の「もの」や「こと」、要するに使われる情報の意味構造、意味の体系を整理することから入りましょうと我々は言っています。しかし、それを行うには高度なスキルが必要で、社内のユーザからきちんと話を聞き出せる触媒のような人が必要です。

鶴保　そういうことは社内のIT部門の人材でできるのですか。

繁野　社内の人材では難しいですね。一方で、ビジネスコンサルティング会社が行うのは業務プロセスに関するコンサルティングが中心です。ですから、私が述べたようなアプローチはほとんどとられていないというのが実態です。

鶴保　データ中心やオブジェクト指向という考え方は大学で教えれば身に付くと考えられますか。

繁野　教えないといけないでしょうね。オブジェクト指向といっても、私が話しているビジネスを捉えるというレベルや、システムに実装するレベルなどいろいろあります。大学で教えて欲しいのは、情報システムとは何かということです。「コンピュータは実世界のシミュレータである」というような基本的な原理・原則をしっかりたたき込んでほしいのです。

鶴保　コンピュータサイエンスやソフトウェアエンジニアリングの分野の先生はいるのですが、情報システムの分野を教えられる先生は少ないのが実態ですね。

繁野　SECには、いろいろな業種、業態を対象として、企業のビジネスアーキテクチャをとらえる概念データモデルを研究対象にして、その方法論を作ってほしいですね。

鶴保　SECの1期事業は2007年度で完了しますが、これまではソフトウェア開発のマネジメントに関する要素技術を中心に取り組んできました。例えば、見積りやプロジェクトマネジメントをデータに基づいて行うなど、実践的なエンジニアリング手法を用いてソフトウェア開発の科学的・定量的マネジメントができるよう研究を進め、成果を生み出しました。2008年度からの2期では次のフェーズとして設計論を取り上げようと思っています。

繁野　データ・モデリングとアーキテクチャは車の両輪ですからアーキテクチャを作るところも大切です。また、できれば、企業を対象にしてデータモデルを書いてみるといった、教育のためのモデルケースを作ってもらえるとありがたいですね。

鶴保　2期では、経済産業省と連携してワーキンググループにベストプラクティスを集め、ツールにできるものはツールにして実証実験を行おうと考えています。まず、概念データベースやデータ中心のワーキンググループを設置する計画です。

繁野　個々の企業が行うデータモデリングは企業の本質的なところなので、公開しにくいのが実情です。ですから、SECに取り組んでほしいのです。企業に働きかけてデータモデリングを行い、その成果を差し支えない範囲で公開してほしい。あと、データモデリングができる人材の養成ですね。そういうインキュベータのようなものができるといいなと思っています。

鶴保　実証実験のようなものがぴったり合うのではないでしょうか。今後も、データモデリングに関して一緒に取り組んでいきましょう。

文：小林秀雄　写真：越昭三朗

「SEC journal」論文賞 受賞論文発表

SECは、我が国のソフトウェア産業発展のための様々な取り組みを実施しておりますが、その取り組みの一環としてソフトウェアエンジニアリングに関する論文に賞を設け、表彰を行っております。

今年の「SEC journal」論文賞は、2006年8月から2007年7月に投稿された合計14編の論文を候補とし、査読委員による厳正な審査の結果3編を優秀賞とし、また審査委員により、その中の1編を最優秀賞に選定いたしました。論文をご投稿いただいた皆様の熱意と努力に敬意を表します。

最優秀賞の審査、各賞の発表と表彰は、2007年10月30日に実施したIPA フォーラム 2007「SECコンファレンス～「SEC journal」論文発表会～」において行いました。審査報告と論文の講評は、本号20頁からご覧ください。

また、SECコンファレンスでは、「ISO/IEC 15504に準拠するプロセス改善」という表題で山形薫氏（財団法人日本規格協会）による「ISOとプロセス改善標準化」、堀田勝美氏（株式会社コンピータジャパン）による「SPEAK IPA版の概要と留意点」、新谷勝利（SEC）による「オートモーティブスパイス概要」という3本の講演を行いました。当日の論文発表及び講演の資料は、SEC-Webサイト（http://sec.ipa.go.jp/）よりダウンロードが可能ですので、こちらもご覧ください。

（最優秀賞受賞論文『組込みソフトウェア設計検証へのモデル検査技術の適用と考察』は本号に掲載、優秀賞受賞論文『WBSに基づくプロジェクト管理システムの実現』は「SEC journal」No.9に掲載、優秀賞受賞論文『メモリリーク検出システム λ trace』は「SEC journal」No.11に掲載）

「SEC journal」論文賞審査委員会

委員長	相磯　秀夫	東京工科大学　学長
委員（50音順）		
	有賀　貞一	株式会社CSKホールディングス　取締役
	井上　克郎	大阪大学大学院情報科学研究科　コンピュータサイエンス専攻　教授
	大原　茂之	東海大学　専門職大学院　教授　組込み技術研究科　研究科長
	櫛木　好明	松下電器産業株式会社　シニアフェロー
	重松　　崇	トヨタ自動車株式会社　常務役員
	冨永　　章	日本IBM株式会社　技術顧問　兼　東京大学大学院 工学系研究科 特任教授
	藤本　隆宏	東京大学COEものづくり経営研究センター　センター長
	横田　英史	日経BP社　制作室長
	鶴保　征城	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

「SEC journal」論文賞査読委員会

委員長	井上　克郎	大阪大学大学院情報科学研究科　コンピュータサイエンス専攻　教授
委員（50音順）		
	天嵜　聡介	鳥取環境大学大学院　情報システム領域　助教
	片岡　欣夫	株式会社東芝　研究開発センター　システム技術ラボラトリー　主任研究員
	古賀　惠子	株式会社日立製作所　ソフトウェア事業部　品質保証部　技師
	平山　雅之	株式会社東芝　ソフトウェア技術センター　企画担当　参事
	二上　貴夫	株式会社東陽テクニカ　ソフトウェア・ソリューション　チーフコンサルタント
	渡辺　雅人	株式会社CSKシステムズ　製造グループ　第三開発部　主査

IPAフォーラム2007

SECコンファレンス～「SEC journal」論文発表会～

2007年10月30日（火）　主催：独立行政法人 情報処理推進機構　後援：経済産業省

最優秀賞

賞状、記念品　副賞：金1,000,000円

組込みソフトウェア設計検証へのモデル検査技術の適用と考察

岸 知二，金井 勇人

優秀賞

賞状、記念品　副賞：金500,000円

WBSに基づくプロジェクト管理システムの実現*1

原田 晃，粟根 達志，伊野谷 祐二，大里 立夫，大野 治，松下 誠，楠本 真二，井上 克郎

メモリリーク検出システム λ trace*2

青島 武伸，伊藤 智祥，春名 修介

上段左より、堀田 勝美・山形 薫・新谷 勝利・渡辺 雅人・古賀 惠子・二上 貴夫・片岡 欣夫
有賀 貞一・櫛木 好明・伊野谷 祐二・楠本 真二・金井 勇人・井上 克郎・大原 茂之・重松 崇・横田 英史
鶴保 征城・大野 治・原田 晃・岸 知二・青島 武伸・春名 修介・相磯 秀夫・藤原 武平太

＊1 SEC journal No.9 PP.10-17,2007 ，＊2 SEC journal No.11 PP.6-15 ,2007にそれぞれ掲載

（敬称略）

WBSに基づくプロジェクト管理システムの実現

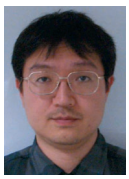
原田 晃(日本電子計算株式会社サービス統括本部，本部長，博士（情報科学））
粟根 達志(株式会社日立製作所情報・通信グループ プロジェクトマネジメント統括推進本部，主任技師）
伊野谷 祐二(株式会社日立製作所情報・通信グループ プロジェクトマネジメント統括推進本部，主任技師）
大里 立夫(株式会社日立製作所情報・通信グループ プロジェクトマネジメント統括推進本部，技師）
大野 治(株式会社日立製作所情報システム事業部，事業部長，工学博士）
松下 誠(大阪大学大学院情報科学研究科，准教授，博士（工学））
楠本 真二(大阪大学大学院情報科学研究科，教授，博士（工学））
井上 克郎(大阪大学大学院情報科学研究科，教授，工学博士）


原田 晃

粟根 達志

伊野谷 祐二

大里 立夫

大野 治

松下 誠

楠本 真二

井上 克郎

エンタープライズ系ソフトウェアの開発プロジェクトは、メンバが多く、作業や成果物が膨大であり、また作業にあたって参照する資料が膨大なために、効率的なプロジェクト管理を支援するシステムが必要となります。私たちは、作業や成果物に関連する多くの情報を相互に関連付けして一元管理するためのWBSモデルの作成と、そのWBSモデルを利用したプロジェクト管理システム「プロナビ」を開発し、現場への適用を進めてきました。

WBSはプロジェクトで遂行する作業を、成果物に着目して管理しやすいレベルまで細分化した階層図ですが、WBSの構成要素である各作業に対し、担当者、成果物、スケジュールと実績、作業遂行に際して参考にする資料等の情報を付加したものをプロナビWBSと呼びます。プロナビでは、プロナビWBSの管理及び成果物の構成管理や参考資料の文書管理をサポートしており、プロジェクトの管理者や開発者は、PC上のWeb環境からほとんどの作業を遂行できるので、効率よいプロジェクト管理を実現できます。

プロナビは2000年3月から現場での適用を開始しており、累計で約3,000、常時500を超えるプロジェクトが適用しています。プロナビを利用したプロジェクト管理者にアンケートやヒアリングを実施しましたが、ほぼ全員が満足しており、また成果物等の情報の一元管理を高く評価している人が約半数いることを確認できました。

組込みソフトウェア設計検証へのモデル検査技術の適用と考察

岸 知二（北陸先端科学技術大学院大学情報科学研究科，特任教授，博士（情報科学））
金井 勇人（北陸先端科学技術大学院大学情報科学研究科，博士後期課程，修士（情報科学））


岸 知二

金井 勇人

北陸先端科学技術大学院大学で進めている高信頼組込みソフトウェア設計に関するプロジェクトでは、形式検証技術による組込みソフトウェアの設計検証、具体的にはモデル検査技術によるUML設計検証、及びその実用化に企業とともに取り組んでいます。

UML設計検証とは、平たく言えばUMLで書かれた設計書が正しいかどうかを確認することです。モデル検査技術により、例えば割込みやタスクのスイッチングがどのタイミングで起こっても正しく動作するか否かを網羅検証することができます。しかしながら、どのようなUMLを記述すべきか、状態爆発（網羅検証する状態が大きく検証ができない状況）にどう対応するか等、その適用には難しさが伴います。

本論文では、企業から提供された組込みソフトウェアに対してアーキテクチャレベル、タスク設計レベル、ソースコードレベルの3種類の異なったUML設計を行い、それぞれで検証できる性質や検証コスト（時間やメモリ）を比較・実測しました。この試行を通じ、検証目的に応じた記述レベルの設定が重要なこと、状態爆発に対しては、シナリオの限定やパターン化などソフトウェア技術者が用いている設計上の技術が一定の有効性を持つこと、ひいてはよい設計は検証しやすい設計につながるという知見・観測が得られました。

今後、より多様な試行や実証を重ねながら、形式検証技術の実用化に向けたソフトウェア工学面からの研究を深めていきたいと考えています。

メモリリーク検出システム λ trace

青島 武伸（松下電器産業株式会社システムエンジニアリングセンター，主任技師，博士（工学））
伊藤 智祥（松下電器産業株式会社eネット事業本部，主事，工学修士）
春名 修介（松下電器産業株式会社システムエンジニアリングセンター，主幹技師，博士（工学））


青島 武伸


伊藤 智祥


春名 修介

近年、家電製品に搭載されるソフトウェアはますます巨大化し、その品質を保ちながら、開発工数の増大の度合いを低く抑えることが大きな課題となっております。こうした背景のもと、特に私たちは、メモリリークの問題に着目しました。メモリリークは、その発生直後ではシステムの機能は失われず、実行が継続するため、テストでの発見に多大な工数が必要になるという特徴があります。また、度重なるリークの後には、システムの停止等、重大な障害を引き起こす場合があり、リークが発生する可能性は、漏らすことなく取り去る必要があります。

本論文は、C言語で記述されたソースコードを入力とし、メモリリークが発生する箇所を検出するシステムについて、その解析手法のほか、検査ツールの実行環境、検査結果の精査環境をまとめたものです。本システムにより、テスト工程の前に効率よくメモリリークに関する検査を行うことができ、テスト工数の削減に大きく寄与します。

解析手法の設計においては、コードの一部に未完成の箇所があったとしても、スタブ等を手動で用意することなく、十分な精度をもつ検査になることを念頭におきました。コードが完成した開発終盤に検査を行って、この時点でコードを修正する場合、すでに行われているテストのやり直し等、手戻りのコストが増大するため、これを避ける必要があると判断しました。

また実行環境では、検査対象の関数の大小に依らず検査をスケジュール通り完了させること、検査結果の精査環境では精査の工数を削減すること等、実際のプロジェクトに寄与するために必要な要素を強く意識し、設計、及び実装を行いました。

これらはメモリリークの検査以外にも応用できるものと考えております。本論文を他の検査系を設計される際の一助として頂けましたら幸いです。

組込みソフトウェア設計検証へのモデル検査技術の適用と考察


岸 知二†　金井 勇人†

組込みソフトウェア開発において設計品質の向上は重要課題であり，設計検証に対する形式検証技術の適用が期待されている．一方，多くのソフトウェア技術者にとって形式検証技術は新しい技術であり，わかりやすい適用方法の整備が求められている．本稿ではモデル検査技術を利用したカーオーディオのソフトウェアの設計検証を題材に，ソフトウェア設計手法の分野での既存のモデル化技術を活用した設計検証の事例を紹介するとともに，設計検証のアプローチについて考察する．

Design Verification of Embedded Software based on Model Checking Techniques - Application and Discussions -

Tomoji KISHI and Hayato KANAI

It is claimed that design quality is crucial for embedded software, and it is expected to apply formal methods to improve the design quality. As actual use of formal methods is a new challenge for ordinary software developers, it is expected to set up handy guidelines for the application. In this paper, we introduce a case of applying model checking techniques to design verification of car audio software, in which we utilize existing modeling techniques proposed in software design field. Based on that, we discuss design verification approaches.

1　はじめに

組込みソフトウェアの設計品質は，その生産性や品質に大きな影響を持っており，設計品質の改善は重要な関心事となっている．産業界においても，新たな検証技術として形式検証技術，特にモデル検査技術への期待が高まっている．そうした背景の中，我々はモデル検査技術[CLARKE1999][BERARD2001]を用いてUMLで記述された組込みソフトウェア設計の検証を行うツール（以下UML設計検証ツールと呼ぶ）を開発し，事例への適用を行いながら，その適用方法について検討を進めている[KISHI2004][KISHI2006].

一般に組込みソフトウェアは複数のタスクや割込みを活用した複雑な内部構造を持ち，またその利用特性上，様々なイベントの順序や組合せに対して正しく振る舞う必要がある．モデル検査技術は多様な状態やイベントの組合せに対して網羅的な検証を行うことができるため，組込みソフトウェアのこうした側面の検証に適していると期待されている．

一方，モデル検査技術は例えばテスト等とは異なった特性を持っているため，検証モデルの作成，状態爆発への対処，検証結果の解釈等に関し，特有の技術やノウハウの蓄積が必要となる．さらにそれらに基づき，よりマ

†　北陸先端科学技術大学院大学 情報科学研究科，　Japan Advanced Institute of Science and Technology

クロな検証の組立てや戦略についても，わかりやすい指針が求められている．

本稿では企業より提供されたカーオーディオのソフトウェアの事例に対して，モデル検査技術による設計検証を行った結果について報告する．適用においては検証モデルの作成方法（抽象化の方法）に注目し，どのようなモデルで，どのような検証ができ，どの程度のコストがかかり，どのようにその結果を活用することができるかを検討した．

特に本事例の検証モデルの作成においては，モデル検査技術のための特殊な抽象化を極力避け，ソフトウェア設計手法の分野で知られている既存のモデル化技術を用いた．その結果，ソフトウェア設計手法の分野での既存技術に基づく検証モデルが，目的に応じては一定の有効性と優位性を持つことが確認された．本稿ではその事例を報告するとともに，それに基づき，設計検証へのモデル検査技術適用に関するアプローチについて考察する．

第2章では本稿で議論するモデル検査技術上の課題について述べる．第3章ではソフトウェア設計の分野でのモデル化技術について述べる．第4章ではカーオーディオソフトウェアへの適用事例について述べる．第5章では本結果に基づきモデル検査手法に関する考察を行う．

2　モデル検査技術での設計検証の課題

2.1　モデル検査技術とUML設計検証ツール

モデル検査技術は，有限状態モデルと論理的な性質が与えられると，そのモデル上でその性質が成立するかどうかを自動的に検査する技術である．ソフトウェアの設計をモデル検査技術で検証する際には，対象とするソフトウェア設計を有限状態モデルで表現すると同時に，その設計上で確認したい性質を有限状態モデル上の論理的な性質として表現し，モデル検査を行うエンジン（以下検証エンジンと呼ぶ）に入力として与える．検証エンジンは，与えられた性質が成立するかどうか，起こり得るシステムの全状態を網羅検査し，成立するかしないかを報告する．また成立しない場合には反例（性質が成立しない状況に至る動作例）を示す．

組込みソフトウェアが，外界で生起する様々な事象のどのような順列組合せに対しても正しく動作するかどうかを確認するのは通常のテストやレビューでは難しい．同様にタスクのスイッチングや割込み発生の順序関係に関する確認も困難である．モデル検査技術はこうしたタイミングに関わる性質を網羅的に確認することに適している．一方，何秒以内に反応するといった実時間に関わる性質の確認には不向きである．

様々な検証エンジンが利用可能だが，例えばSPINという検証エンジンの場合，有限状態モデルを直接記述するのではなく，Promelaと呼ばれる専用の言語を用いて対象を記述することにより，内部的に有限状態マシンに変換してくれるため，特に複数の並行動作単位が同期・非同期の通信を行いながら協調動作するシステムの記述が容易となっている．また性質の記述にはLTL（Linear Temporal Logic）と呼ばれる時相論理のクラスが利用される[HOLZMANN2004].

我々はこのSPINを利用し，UMLを用いてソフトウェアの設計検証を行うことのできるUML設計検証ツールを開発した[KISHI2004][KISHI2006].

図 1は本ツールの機能を模式的に示したものである．利用者はモデル定義機能を用いて検証対象となる設計をUMLのクラス図とステート図からなる検証モデルとして表現する．一方，検証項目定義機能を用いて，確認したい設計上の性質をUML検証モデル上の性質記述として与える．変換機能はそれぞれを内部的にPromelaでの検証モデル，LTLでの検証性質記述に変換し，検証管理機能

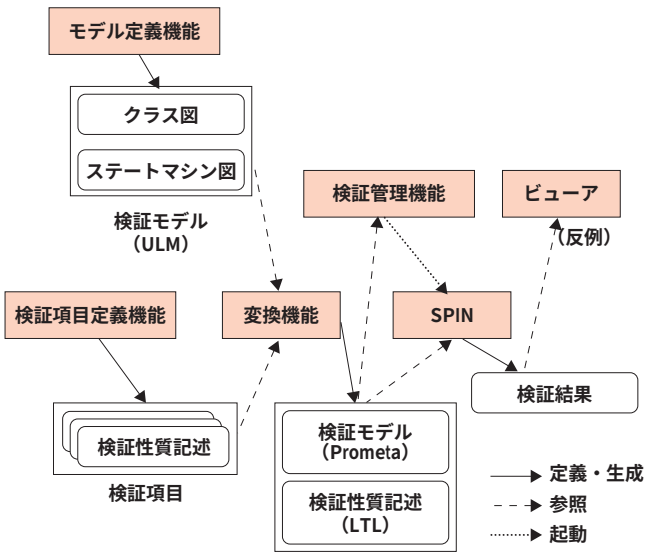


図1　UML設計検証ツールの機能

はこれらを入力としてSPINを起動し，性質が成立するかどうかを検査する．反例がある場合には，ビューアを利用することで，それをUMLのシーケンス図として確認することができる．

UML設計検証ツールにより，Promela等の検証エンジン独自の表現を用いて対象ソフトウェアをモデル化することなく，ソフトウェア設計に利用されているのと同じUMLを用いて検証モデルを表現することができる．もちろんモデル検査のためには，明確に実行意味を意識し，目的に応じた厳密なモデル化をすることが必要となるが，ソフトウェア技術者にとっての親和性が向上するものと期待される．

本稿で紹介する事例では，このUML設計検証ツールを利用した．なお以下特に断らない限り，検証モデルとは，UMLで記述された検証モデルを意味するものとする．

2.2 課　題

UML設計検証ツールの利用はソフトウェア技術者によるモデル検査技術の適用を支援するが，その適用には依然として以下のような課題が存在する．

・検証モデルの作成：モデル化は特定の目的のための対象の抽象化であり，検証モデルの作成においては検証する性質に応じてモデル化の視点，範囲，詳細度を設定する必要がある．検証モデルが必要な情報を含んでいなければ十分な検証結果が得られないし，逆に過度の情報を含む場合には状態爆発（検査する状態数が多くなりすぎて現実の時間やメモリの中でモデル検査をすることができなくなること）を引き起こす．適切な検証モデルの作成は困難な問題である．

・状態爆発への対応：状態爆発を起こす際には，なんらかの手法で検証モデルが内包する状態数を減らす必要がある．モデル検査の技術的特性を踏まえて，状態を多くする部分を抽象化したり，検証エンジンの探索の深さを制限したりする等の方法がとられるが，モデル検査技術に精通していないソフトウェア技術者にとっては，その対応は必ずしも容易ではない．

・結果の解釈：状態爆発によって検証エンジンが途中で検証を中断したり，あるいはその対応として検証エンジンで探索の深さを制限して検証を行ったりした際に，その結果から，対象とする設計の性質に関してどのような知識を得ることができるのか，自明ではない．例えば検証エンジンが状態を深さ50まで検索した範囲では性質が成立する，という結果が得られたときに，それが元となるソフトウェア設計上でどういう意味を持つのか理解するのは容易ではない．

モデル検査技術によるソフトウェアの設計検証を広く使えるようにするためには，こうした課題について一定の指針を与えることが重要である．特に検証モデルの作成は，検証目的，検証における状態数，また結果の解釈に関係する本質的な部分であると考えられる．

一方，ソフトウェア設計手法の世界では多くのモデル化技術が提案され使われてきた．それらの技術は，モデル化の目的に応じた視点を設定したり，複雑さを制御したりすることに有効である．本稿での素朴な問題意識は，こうした既存のモデル化技術が，モデル検査技術による設計検証における検証モデルの構築に対してどの程度活用できるか，ということである．例えば状態爆発に対して，モデル検査技術に依存した抽象化を行うのではなく，ソフトウェア技術者が使い慣れた手法で抽象化を行うことで対応できるのであれば，その適用も結果の解釈も容易になることが期待される．

もちろん，モデル検査技術を活用するには，モデル検査技術に対する一定の理解が必要なことは明らかである．しかしながら，本来ソフトウェア技術者が得意とするモデル化技術が必ずしも十分に活用されていないことも事実である．本稿は，そうした認識から，ソフトウェア設計手法の世界でのモデル化技術の適用可能性について，具体的な事例に基づいて実証的に確認し，議論することを目的とする．

3　設計手法におけるモデル化技術

ソフトウェア設計手法の分野では，従来から多くのモデル化技術が提案されてきた．ここではそれらの中で，本稿に関わる部分について説明する．

ソフトウェア設計には様々な意義があるが，典型的には以下のような目的があり，そのために必要なモデル化技術が検討されてきた．

・ビューの明確化：設計は基本的には，実際に実装する

前に確認すべきことや実装するために必要なことを確認・決定するために行う．例えば方式の妥当性を確認したり，作業分担のためのインタフェースを決定したりする．こうした目的に応じてモデル化のビューが異なる．ここでビューとは，対象とするソフトウェアのどの局面の姿を捉えるかという視点をいい，例えばソフトウェアが動作しているときの姿を捉えるビューを実行時ビュー，ソフトウェアを開発しているときの姿を捉えるビューを開発時ビュー等と呼ぶ．このように，モデル化の基本は，ビューの決定であるといえる．

・目的に応じた抽象：モデル化は目的に応じて対象を抽象化することであるから，目的が異なればそのモデル化が異なる．ビューの明確化も抽象化の重要な要因である．また同じビューであっても，目的に応じてモデル化の範囲や詳細度は異なる．さらにモデル化は特定の形式性の上でなされるため，その記述の基本要素や実行意味を決定しなければならない．例えば具体的なOSや言語の提供するタスク，スレッドといった基本要素や実行意味を用いてモデルを記述することもできるし，より概念的な基本単位や実行意味に基づいてモデル化することもできる．これらは，どういう目的でモデルを作成するかに依存して決定される．

・複雑さの制御：大規模で複雑なシステムをいきなり実装することは困難である．設計には，こうした複雑さを制御しながら実装可能な形へと解きほぐすという役割がある．そのための手法として，階層化や段階的詳細化，データ抽象化やカプセル化，観点の分離等の様々な技術が使われてきた．パターン化も，複数のものに表れる共通性を捉えたり，逆に複数のものを共通性に基づいて類型的に実現したりする技術であり，複雑さの軽減に役立っている．

このように，同一の対象に対して，様々なモデル化の方法が存在し，異なったモデルが存在する．モデルが異なれば，それによって捉えることのできる対象の特性も異なるし，モデルの複雑度も異なる． 以下，モデル検査技術による設計検証において，こうしたモデル化の違いがどのような結果の違いを引き起こすかを確認するとともに，それをどのように活用することができるかについて検討する．なおモデル検査技術は典型的には対象システムの動的な側面を捉えるものであるから，本稿では，すべて実行時ビューのモデルを扱う．

4　カーオーディオシステムの設計検証

4.1　目的と概要

本検証は，モデル検査技術を利用した設計検証において，ソフトウェア設計手法の分野で知られている既存のモデル化技術を用いて検証モデルを構築することの意義や課題を実証的に確認することにある．具体的にはソフトウェア設計において典型的と思われる複数の抽象化視点を設定し，それらに対応した検証モデルを，UML設計検証ツールを用いて構築し，それぞれに対して以下を確認，検討する．

・どのような性質が検証できるか．
・検証のコストはどの程度か．
・状態爆発への対応や結果の解釈．

検証は以下の手順で行った．

① モデル化の視点を設定：どのようなモデル化の視点で検証モデルを作成するかを決定．
② 提供された事例に対して，上記で定めた視点に基づいて検証モデルを作成．
③ 設計検証の実施：構築したモデルに対してモデル検査を実施．検証コストの測定．
④ 結果の検討と考察：検証結果に基づき，意義や課題を検討．

このように，本事例では，提供されたソースコードや設計書を元に，設定したモデル化の視点に対応した検証モデルを新規に作成し，それに対してモデル検査技術による設計検証を行った．検証モデルの作成にはUML設計検証ツールを利用し，検証エンジンとしてはSPINを利用した．

4.2　カーオーディオシステム

今回対象とした事例は，企業より提供されたカーオーディオシステムのソフトウェアである．本システムはリファレンスセットとして開発されており，実装にはC言語を用い，32bitCPUの上でμITRONを用いて開発されている．CD，チューナ，交通情報等を聞くことができるが，本オーディオシステムではCDやチューナ等をソース，ソ

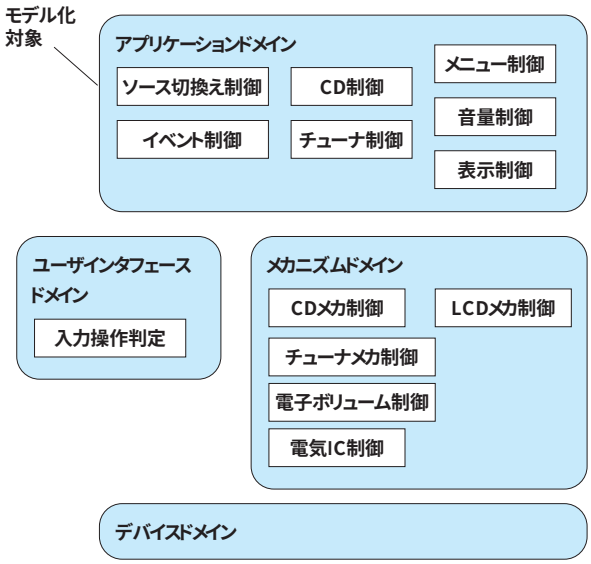


図2　カーオーディオのドメイン構成

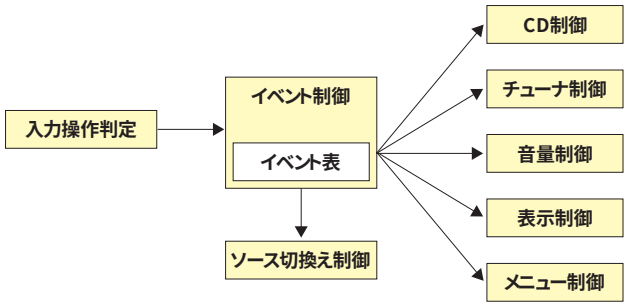


図3　カーオーディオの処理の流れ

ースを切り換える機能をソース切換え機能と呼ぶ．またLCDへの表示，音量の制御，メニュー制御等の機能を備えている．

図 2はソフトウェア構成の概要を示したものである．全体は大きくは，低レベルのデバイスインタフェースに関わるデバイスドメイン，ユーザのボタン操作を判断するユーザインタフェースドメイン，CD等のメカを制御するメカニズムドメイン，及びユーザインタフェースからの指示に従い，メカニズムを制御するアプリケーションドメインから構成される．

図3は基本的な処理の流れを示している．本システムでは，ボタンの押下等のユーザの操作は，そのときのソースによって異なった意味を持つ．例えば，あるボタンはソースがCDのときには再生・停止指示，ソースがチューナの時には選局の意味を持つ等する．ユーザインタフェースからはユーザ操作の種類がイベント制御に伝えられ，イベント制御はソース切換え制御に現在ソースを問い合わせ，そのときに必要な制御を内部に持つイベント表を引いて判断し，必要な指示を各種制御に送る．

表1はソフトウェアの規模である．なお，実行数とはコメント行や空白行を除いた行数である．

表 1　ソースコードの規模

ドメイン	実行数	関数の数
アプリケーション	12,349	540
ユーザインタフェース	2,365	93
メカニズム	3,154	143
デバイス	2,369	233
共　通	363	22

今回は，これらのうちアプリケーションドメインをモデル化の対象とした．ただし，チューナ制御（2,806行）は除外した．

4.3　モデルの作成

今回はソフトウェア設計において作られる典型的なモデルとして，以下のモデルを作成した．

・実現構造モデル：実際のソフトウェアの実現構造を表現したモデル．ソースコードに近い設計を表すモデル．
・アプリケーションモデル：リアルタイムOSのタスク構造と各タスクの機能に注目したモデル．実現構造の具体的詳細は省略される．
・方式モデル：処理方式等，基本的なアーキテクチャに注目したモデル．実現構造の詳細も機能の詳細も抽象化される．

以下，カーオーディオの事例での，これらのモデルについて説明する．

（1）実現構造モデル

実際のソフトウェアの実現構造を，リアルタイムOSの機能も含めて表現したモデル．以下の方針でモデル化を行った．

・タスクはPromelaのprocessとして実現． μITRONをPromela上で模擬するライブラリ[AOKI2005]を利用し

て，優先度やスケジューリング方法も実際のソフトウェアの方法に準じてモデル化．タスクの基本的な振る舞いは状態モデルとして定義．

・意味的にグループ化できる関数群と変数群をオブジェクトとして捉え，その単位でPromelaのprocessとする．オブジェクトの基本的な振る舞いを状態モデルとして定義．

・タスク間通信（メッセージキュー）は，ライブラリの提供する機能を利用．

・オブジェクト中の関数への呼び出しは双方向の同期チャネル上での，呼び出しとリターン値のやりとりで模擬．

・アプリケーションロジックは制御面に関してはほぼ忠実に実装．ただしパラメータとして渡されるデータ値やその計算等は制御にかかわらない限り省略．

・サービスコールのリターン値に対する対処（メッセージキューへの送信サービスコールが失敗した場合の再送処理等）もほぼ実装．

・メカニズムとのやりとりは省略．

なお詳細な処理部分については，UML設計検証ツールを用いず，直接Promelaを記述した．モデルはPromelaにして約3,700行である．またμITRONライブラリは約500行（今回利用した部分）である．

図4はイベント制御機能におけるメッセージキューの再送処理部分のステート図である．本モデルは，リアルタイムOSのサービスコールの利用方法までモデル化しているため，タスクの並行性やタスク間通信に関わるタイミング上の性質を，例えばサービスコールのリターン値の参照タイミング等の詳細なレベルまで立ち入って議論することができるモデルとなっている．

（2）アプリケーションモデル

実現メカニズムや機能の詳細は省略し，実現する基本的な機能に注目したモデル．以下の方針でモデル化を行った．

・タスクはPromelaのprocessとして実現．ただしリアルタイムOSの振る舞いは簡略化し，ブロックしていないprocessには公平にスイッチングの可能性があるという，実際よりも一般的な実行意味でモデル化．

・タスク間通信（メッセージキュー）は非同期チャネルで模擬．ただしサービスコールの詳細は模擬せず，より単純な振る舞いとして実現．

・意味的にグループ化できる部分をprocess化し，関数コールを疑似する部分は実現構造モデルと同様．

・実現構造モデルにおいて記述されたステート図の振る舞いはほぼ含む．ただしリアルタイムOSとのやりとりにかかわる部分や，直接Promelaとして記述された処理の詳細部については省略．

図 4　実現構造モデルの一部

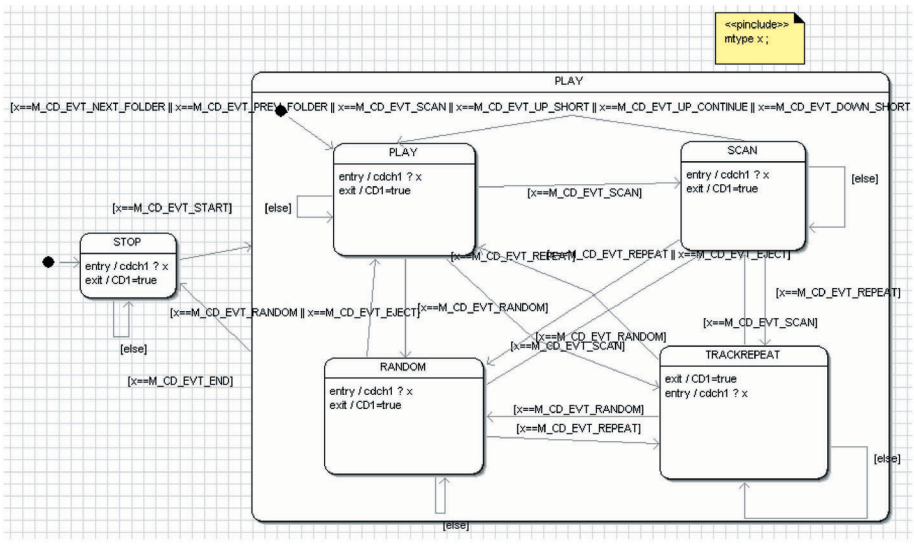


図 5　アプリケーションモデルの一部

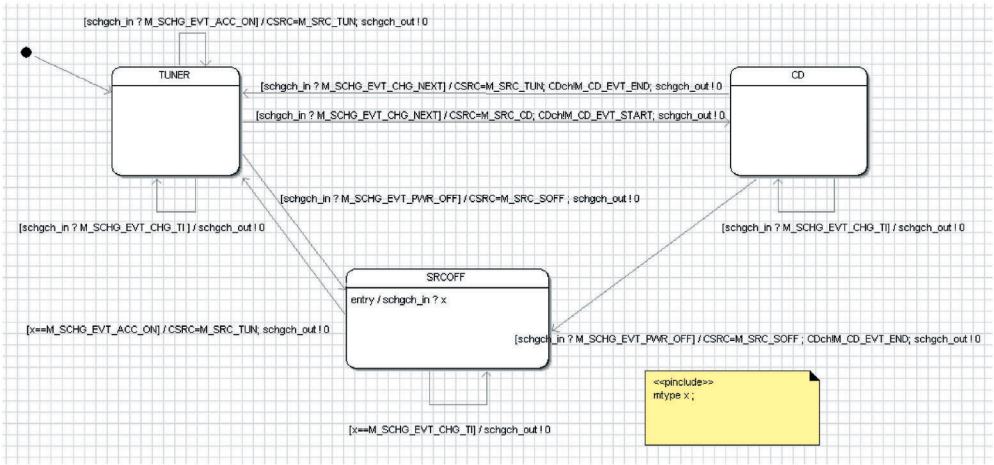


図 6　方式モデルの一部

モデルはPromelaに変換して約750行である．

図 5はCD制御機能のステート図である．本モデルではサービスコールの利用やアルゴリズムレベルの詳細はモデル化されていないため，タスクの並行性やタスク間通信に関わるタイミング上の性質を，基本的なタスクの状態（再生中，停止中等）のレベルで議論することができるモデルとなっている．

（3）方式モデル

ユーザ操作がイベント制御とソース切換え機能とを介して各種制御に伝達される基本的なイベント処理の方式を確認するためのモデル．実現メカニズムや機能の詳細は省略される．以下の方針でモデル化を行った．

- 実際にどう実装されているかを考えず，概念的な機能（イベント制御機能等）をprocessとする．
- process間に方式上の意味に従って同期・非同期のチャネルを定義．
- データフロー的実行意味（イベントが存在すればそれを読み，なければブロックする）で実行．
- 各種機能単位の具体的な処理内容は一切記述せず，イベントのフローのみを捉える．

モデルはPromelaに変換して約650行である．

図 6は，ソース切換え機能のステート図である．本モデルでは機能間のイベントのやりとりという側面のみを

モデル化しているため，各ソースにおける並行動作単位と通信に関わるタイミングの性質のみを議論することができるモデルとなっている．

4.4 モデルの検証

（1）検証の方法

今回の適用では，モデルの違いに応じた検証コストの比較をすることが必要であるため，どのモデルでも行え，かつ内包する状態数が検証コストに反映される検証として，不正な終了状態の検証（すべての動作系列を調べ，予期していない状態で終了することがないかどうかを網羅検査する検証）を行った．

検証はUML設計検証ツールから生成されたPromelaを用いて行ったが，動作させるためには外部のユーザインタフェースドメインからの操作指示をイベント系列として与える必要があり，また後述するように複数の異なったイベント系列パターンを与えて検証を行ったため，この部分は直接Promelaを記述し，生成されたPromelaとマージした．実行環境は以下のとおりである．

・検証エンジン：SPIN（Version 4，2.9）
・検証マシン：SGI Altix3700（本マシンは大量のメモリを使えるために利用した．実行速度に関しては通常のデスクトップ等と類似）．

検証コストの測定はspin自身が生成する実行ログと，UNIXのtimeコマンドを利用した．なお実行時間はマシンによって大きく異なること，検証項目によっては一定時間経過して終了しない場合は検証を中止する等したこと等から，本稿ではあくまで参考値として示す．

以下，検証ができたとはすべての状態探索を終えることができたことをいう．また検証が終わらないとは，ほぼ1日程度かけても検証が終了していないため検証を中止した状況をいう．もしもそのまま検証を続ければ検証が終わったかもしれないし，その後状態爆発等を起こして中断したかもしれないが，それは未確認である．なお上記時間内に探索の深さが上限に達した場合は，上限を引き上げて再検証した．今回の検証においては，上記時間内にメモリが足りないという状況にはならなかった．

（2）実現構造モデル

検証ではユーザ操作の系列を与えて，不正な終了状態の検証を行った．ただし最初は電源を入れなければ動かないこと，チューナを実装していないのでソースをCDにすることが妥当であることから，他のモデルを含めすべての検証において，電源を入れ，ソースをCDに設定するという操作を行った後に，操作シーケンスを与えた．

起こりうるすべてのユーザ操作（26種類）をランダムに送って検証を行った．今回は長さ2の操作シーケンスまでは検証できたが，長さ3以上は検証が終わらなかった．表2は検証結果の概要である．ここでLは操作シーケンスの長さ，深さは検証エンジン内部での状態探索の深さ，状態数は検証エンジン内部での状態数，使用メモリは実際に使ったメモリ（単位Mバイト），時間はtimeコマンドの示す所要時間である（単位分・秒）．

表 2　実現構造モデル（全操作）

L	深さ	状態数	使用メモリ	時間
1	316	178,401	126.887	0m26s
2	2,891	1,356,700	946.786	3m30s

特定の部分に対する操作指示に限定することで検証が可能になるかどうかを検討した．

ソース切換えにかかわる操作（5種類）に限定したところ長さ3までの検証が可能となったが，4では検証が終わらなかった．表3はLが3のときの結果である．

表 3　実現構造モデルの検証結果（ソース切換え）

L	深さ	状態数	使用メモリ	時間
3	6,857	18,023,900	12,543.557	1h2m8s

CDに対する一部の制御（21種類）に限定したが，長さ3では検証が終わらなかった．一方，基本的な操作（6種類）に限定することで，長さ3までの検証が可能となったが，4以上は検証が終わらなかった．表 4はLが3のときの結果である．

表 4　実現構造（CD制御）

L	深さ	状態数	使用メモリ	時間
3	6,857	131,693,000	91,632.728	25h6m10s

このように実現構造モデルでは，ある程度の限定を行っても，十分な検証ができなかった．

（3）アプリケーションモデル

アプリケーションモデルに対して実現構造モデルと同様の検証を行った結果を示す．

全操作に対しては，長さ5までの検証が確認できた．表5はLが1から5までの時の結果である．

表 5　アプリケーションモデル（全操作）

L	深さ	状態数	使用メモリ	時間
1	73	9,891	0.651	0m3s
2	94	197,243	8.527	0m7s
3	110	2,915,070	118.601	1m5s
4	127	24,794,600	995.555	21m56s
5	143	112,792,000	4517	4h12m39s

ソース切換え関係だけであれば，長さを限定しない一般的な系列に対しての検証を行うことができた．表6は長さを限定せずに行った検証の結果である（Lが＊とは，長さを限定しないことを示す）．

表 6　アプリケーションオンモデル（ソース切換え）

L	深さ	状態数	使用メモリ	時間
＊	1,139	19,797	1.068	0m1s

同様に，6種類に限定したCD操作に対しても，長さを限定せずに検証を行うことができた．表 7は検証の結果である．

表 7　アプリケーションモデル（CD制御）

L	深さ	状態数	使用メモリ	時間
＊	42,489	1,181,650	21.548	0m18s

このようにアプリケーションモデルでは，ある程度の限定を行うことで，一定の検証が可能であった．

（4）方式モデル

本モデルに対しては，全操作に対して長さを制限せずに検証を行うことができた．表8はその結果である．

表 8　方式モデル（全操作）

L	深さ	状態数	使用メモリ	時間
＊	227,617	4,829,040	196,981	3m41s

このように方式モデルは十分な検証ができた．

5　考　察

以下，本適用結果に基づく考察と議論を行う．

5.1　どのような検証ができるか

前述したようにモデル検査技術はタイミングに関わる性質の検証に適しており，第4.3節で述べたように各モデルはそれぞれの詳細度に応じたタイミング上の性質を議論できる．しかしながら今回の事例からわかるように，最も多くの情報を含んでいるモデルが最も検証に適しているわけではない．例えば実現構造モデルはアプリケーションモデルの持つ機能的な情報を含んでいるが，機能確認の観点からは実現構造の詳細がむしろ阻害要因となる．例えば機能確認という目的のためには，再送失敗やタイムアウト等のエラー時の振る舞いが詳細に記述される実現構造モデルよりも，アプリケーションモデルの方が合目的である．また状態爆発の問題も生じた．すなわち，できるだけ観点を分離した検証を行うことが重要であり，詳細なモデルであれば，より詳細度の低いモデルの代替になるということには必ずしもならない．

5.2　検証のコストはどの程度か

前章でデータを示したように，モデルが詳細になると検証コストは非常に大きくなる．特に実現構造モデルは単に検証時間だけでなく，モデル作成，モデルの妥当性確認が大変であり，このレベルのモデルを使うためには，生成の自動化等の支援が必要になると考える．

5.3　状態爆発への対応や結果の解釈

今回の事例では状態爆発に対して，設計手法におけるモデル化技術の観点から，以下の3つの方法を試行し，一定の有効性を確認した．

・モデルの観点を絞り込む：アプリケーションモデルや方式モデルは目的や表現内容が絞り込まれているため，潜在的に検証できる性質は限定されるが，現実には検証が容易となり，またモデルも状態数もコンパクトになる．検証観点を分離し，検証内容やモデルを絞り込むことが有効と考えられる．なお方式モデルでは，異なったユーザ操作でも類似のイベントフローを持て

ば同等に処理される．これはパターン化とも捉えられる．

・モデルの範囲を絞り込む：CD制御やソース切換え制御等，検証対象を絞り込むことでよりコンパクトな検証となることが期待される．今回の試行のように，特定部分への操作シーケンスに限定する等の工夫等による手段が考えられる．

・検証の深さを絞り込む：操作シーケンスの長さを制約することで，状態数を削減する．長さを限定しない場合には検証が終了しない場合でも，この手法により検証可能な場合がある．

検証エンジンの探索の深さ等を制約するのではなく，ソフトウェア設計の観点からモデルや網羅性を制約することにより，限定された検証の結果が持つ意味がより理解しやすくなることが期待できる．もちろん，特定の対象に関わる制御シーケンスに限定して検証ができたという結果からいえることは，このシーケンス下において，不正な終了状態になることはない，ということであり，それ以外のシーケンスについては何も検証できない．こうした意味解釈には留意が必要である．

6　議　論

本稿では，ソフトウェア設計手法の分野で使われる典型的なモデル化技法を用いた検証モデルに対して，モデル検査技術による設計検証を適用した事例について報告した．この例で示すようにソフトウェア設計技術の観点からの抽象度の設定や，対象やシナリオの限定による状態爆発への対応等は一定の有効性や意義があると考えられる．

モデル検査技術による設計検証を行う際には当然モデル検査技術に関する知識や技術が不可欠であるが，ソフトウェア設計技術の分野で長らく議論されてきたモデル化技術の意味を再度見直し，それを適用することが設計検証において有用であると考え，本試行を行った．コード検証等の世界では自動的な抽象化が重要となるが，設計検証においては設計意図の明確化が不可欠であり，そのためにはソフトウェア設計手法を踏まえたモデル化が重要であると考えている．

なお，どのような手法を使っても，作り散らかし，複雑にからみあった設計を適切に検証することは困難であり，本質は検証容易な設計を行うことである．一方，設計検証を行うためには目的を明確にし，そのために必要十分な情報を備えたモデルを作ることが必須である．本稿で議論したようにモデル検査のための抽象化観点と，既存のソフトウェア設計手法で使われてきた抽象化手法とに一定の整合性があるとすれば，モデル検査技術による設計検証を活用することが，より適切な設計プロセス実現の一助になるとも期待される．

7　おわりに

本稿では，カーオーディオの事例に対して，ソフトウェア設計手法の分野で使われてきたモデル化技術を用いた検証モデルを作成し，モデル検査技術による設計検証を行った事例を報告した．今後こうした試行を積み重ね，より適切な設計及び設計検証の在り方を探っていきたい．

謝辞　本事例を提供いただいたNECエレクトロニクス 水瀬晴美氏，丸野剛治氏，及び設計検証に関する有益な議論を頂いたNEC 野田夏子氏に謝意を表します．また事例の理解やモデル作成に協力してくれた研究室一同に感謝します．

参考文献

[AOKI2005] 青木利晃，片山卓也：RTOSに基づいたソフトウェアのためのモデル検査ライブラリ，組込みソフトウェアシンポジウム2005, pp.56-63, 2005
[BERARD2001] Berard, B., et. Al.: Systems and Software Verification: Model-Checking Techniques and Tools, Springer, 2001
[CRARKE1999] Clarke, E., Grumberg, O., Peled, D. : Model Checking: MIT, 1999
[HOLZMANN2004] Holzmann, G.J. : The SPIN Model Checker - Primer and Reference Manual, Addison-Wesley, 2004
[KISHI2004] 岸知二，青木利晃，中島震，野田夏子，片山卓也：プロジェクト紹介：高信頼組込み用オブジェクト指向設計技術，情報処理学会ソフトウェア工学研究会，SE146-7, pp41-46, 2004
[KISHI2006] 岸知二，野田夏子: 組込みソフトウェアのためのUML設計検証支援環境，情報処理学会，組込みソフトウェアシンポジウム2006, pp50-57, 2006

「SEC journal」論文賞審査委員会審査報告

「SEC journal」
論文賞審査委員会委員長
東京工科大学　学長
相磯　秀夫

第３回 SEC journal 論文発表会において、最優秀論文賞ならびに優秀論文賞を受賞された皆様に心からお祝いを申し上げます。

今年の論文審査は、既に「SEC journal」に採録された論文と、今回の論文賞コンテストのために応募された論文、計14件を対象に行いました。審査にあたっては、最初に大阪大学大学院情報科学研究科教授 井上克郎 委員長ら８名による論文査読委員会が厳正かつ公平な査読を行い、３件の優秀論文賞候補を選出しました。その後、私が委員長を務める論文審査委員会が、臨時委員として２名の特別参加を求め計９名の出席委員により、優秀論文賞候補の選考結果を追認し、その中から最優秀論文を選出いたしました。特に最優秀論文の選考にあたっては、研究成果の実用性・有効性・信頼性・利用性、ならびに論文としての可読性の観点から厳しく比較評価し、総合評価点が最も高い論文を最優秀論文に決定いたしました。

今回の審査対象論文は、過去９回の論文と多少異なる性格を持っています。過去の論文は比較的にソフトウェア開発のプロジェクト管理やプロセス改善に関する課題を論じたものが多かったのですが、今回の論文はソフトウェア開発の効率や質的改善に関するより実用的で、やや個別技術的な課題を取り扱ったものが多く、産学連携による研究が目立っていました。このことはソフトウェア工学の重要性が開発の現場に浸透し、エンピリカルソフトウェア工学の考え方が定着してきたと見ることができ、SEC の活動方針にも一致する望ましい傾向といえます。

受賞論文を概観すると、原田氏らの論文「WBSに基づくプロジェクト管理システムの実現」では、業務用ソフトウェア開発における大規模なプロジェクト管理を効率的に支援するシステムをWBSの考え方を活用して開発し、その有用性を示しています。この論文が評価された点は、実用的なプロジェクト管理システムを開発したこと、多種多様なプロジェクト管理を相互に関連付けし一元管理していること、実際に数千のプロジェクトに適用した結果を評価し、その有用性を示していること、また多くの利用者の意見ならびに要望を収集・分析し、より有用なプロジェクト管理システムの開発に向けて改善に努めていること等です。岸氏らの論文「組込みソフトウェア設計検証へのモデル検査技術の適用と考察」では、高信頼組込みソフトウェア設計の品質向上を目指して、設計検証に対し形式検証技術を適用する方法を提案し、その有用性を考察しています。この論文の特色は、企業から提供された実例を題材にし、提案する設計検証の実用化と解決すべき課題ならびにその解決策について有用な知見と貴重な教訓を得ていること、提案する手法は一般のソフトウェア設計にも適用できること、今後更に発展し、実用化が進む可能性が期待できること等です。青島氏らの論文「メモリリーク検出システムλtrace」では、大規模ソフトウェア開発において問題となるメモリリークを自動的に検出するシステムの開発ならびにメモリリークの検出手法について述べています。この論文の特徴は、C言語ソースコードにおけるメモリリークの検出に特化したシステムを開発し、その実用性を示したこと、またそれに関連した種々の問題点を理論的に考察し、具体的な解決策の実装と効率的な実行環境を提案・実現していること、約500万行の大規模コードの検査を通して、試作システムの評価を行い貴重な知見を得ていること、この種の研究は高品質ソフトウェアの開発において重要な貢献をすることが期待されること等です。いずれの論文も分かりやすく書かれている点も高く評価されました。

以上の受賞論文は、いずれも研究成果の実用化を目標とした有用な論文ですが、解決すべき課題も残されています。そのような意味でも３つの論文の評価は甲乙付け難いものでした。その中で、岸氏らの論文は、わが国が得意とする組込みソフトウェアに関する先端的な研究成果を示したもので、今後の発展が期待されることが評価され、最優秀論文賞に輝きました。優秀論文賞に残念ながら届かなかった論文の中にも、示唆に富んだ有望なものがありました。今後の研鑽に期待しています。

わが国の経済成長ならびに社会発展を維持するためにもソフトウェア工学はますます重要な技術になると思います。この分野で活躍されている皆様方の一層のご努力を願う次第です。

論文講評とSECへの期待

優秀賞受賞論文について、評価できる点、不足している点を審査委員の皆様に解説していただきました。また、同時に、SECに期待することをお伺いしました。

「SEC journal」編集部

株式会社CSKホールディングス
取締役
有賀　貞一

いずれも力作の3編が最終審査に残った。昨年本稿で指摘した「もう少し『大きさ』を感じさせるものがほしい」という点にも応えており、まとまりもよい。ITの活用範囲が拡大し、社会インフラとなる中、システムをいかに効率よく構築するか、いかに品質を上げるか、といった点は従来にも増して喫緊の社会的課題となっている。このような状況に焦点を当てた3論文は、時宜を得たものといえよう。

「WBSに基づくプロジェクト管理システムの実現」は、筆者が転職前の会社における経験をまとめた論文である。「当たり前を当たり前に」がなかなか実現できない当業界において、それを会社全体で活用する管理システムにまで昇華させた点は高く評価できる。転職後の職場においても同様に実践し、その差の分析等も付加してもらえたら、より高い評価が与えられたのではないかと考える。

「組込みソフトウェア設計検証へのモデル検査技術の適用と考察」は、年々急拡大する組込みソフトの品質をいかに確保するかに焦点を当てた意欲的論文である。組込みソフトの規模が百万ライン単位になり、製品開発速度が上がる中で、担当者の努力や単純なテストツールに依存するのみでは、もはや品質維持は不可能になってきた。ここに新しいモデル化技術を活用する意義は大きい。最優秀賞にふさわしいものと考える。

SECの活動は年々厚みを増し、日本のソフトウェア・エンジニアリング活動に不可欠なものとなってきた。「SEC journal」による論文もまた年々洗練されたものとなっている。今後この活動体制を維持し、IPA内の諸機能や外部関連機関との連動によって、ソフトウェア・エンジニアリングの普及促進に更なる発展を期待するものである。

大阪大学大学院情報科学研究科
コンピュータサイエンス専攻　教授
井上　克郎

今回で3度目になる論文賞であるが、以前にも増して本年度の各論文は、それぞれ強力な産学連携の成果に基づいているといえよう。

「WBSに基づくプロジェクト管理システムの実現」は、日立製作所での全社的なプロジェクト管理の仕組みを実現した話であるが、その概念の整理や評価方法に関して大学との連携を行っている。

「組込みソフトウェア設計検証へのモデル検査技術の適用と考察」は、産業界での豊富な知見を有する著者が、大学発の技術であるモデル検査技術を何とか現場に適用させようとする試みを述べている。

「メモリリーク検出システムλ trace」は、大学で在学中に学んだ知見を、会社に就職後、会社の現場で、大規模に適用を試みた例である。

いずれも、産業界の現場での知見だけでは、このような論文の形にするのは、非常に困難だったと思われ、知見は、単なる会社内のノウハウに終わり、普遍的な情報としては広がらなかったと思われる。また、大学の中だけでは、単なる理論や簡単な例題を展開するだけとか、それを論文として発表するだけで、実践的な適用は行われず、産業界には見向きもされずに終わってしまっていたと思われる。

SECの役割として、産業界の中での情報流通と共に、産学連携の橋渡しや育成も重要なことである。大学からの技術シーズを吸い上げると共に、産業界のニーズを確かめ、産学連携の橋渡しのコーディネートを積極的に行ってもらいたい。

今後、ここで発表された論文のように、産学連携が着実に増え、SEC journalがますます賑やかになることを期待する。

（50音順、敬称略）

東海大学 専門職大学院　教授
組込み技術研究科　研究科長
大原　茂之

発表された論文3件は、いずれも現実的な問題解決に向けた意欲的な試みであり興味深いものであった。

原田氏らの論文「WBSに基づくプロジェクト管理システムの実現」は、プロジェクト管理に関するものである。プロジェクト管理分野の論文は少なくないが、実際の開発現場でのデータ収集からシステムの評価に至るまでの、根気強く意欲的な取り組みは、十分評価できる。

岸氏らの論文「組込みソフトウェア設計検証へのモデル検査技術の適用と考察」は、検証に関するものであり、極めて意欲的な取り組みである。特に、ソフトウェアの検証を客観的に行う技術に挑戦するものであり、今後は実用化に向け、カバレッジなどの明確化が期待される。

青島氏らの論文「メモリリーク検出システムλ trace」は、C言語を用いた場合のメモリリークの検出に関するものである。ツールの評価や、プログラムの質を高める技術としての工夫があり、高く評価できる内容である。

以上が発表論文に関する全体的な講評である。次に、組込みソフトウェアを対象にSECへの期待等について述べたい。組込みソフトウェアは、ハードウェアモジュールと異なり、人間の知的活動の成果物がそのまま社会インフラやエンドユーザの日常的環境を構成する資源として使用される。

高品質な社会を築くには、企業単位での組込みソフトウェアの機能という観点を捨象し、社会的安全性、信頼性といった共通課題を解決することである。こうした共通課題を構成する要素は何かを洗い出し、そうした要素をクリアするにはどうすればよいかといった課題を解決していく必要がある。しかし、これらの課題解決は具体的成果を見える形にするのは困難であり、一般の学会論文には採録されにくい面もある。SECには、ぜひともこうした面で、社会に寄与する共通の議論を明示し、解決の場を提供していくことを期待する。

松下電器産業株式会社
シニアフェロー
櫛木　好明

日本のソフトウェア産業の規模が17兆円、人口84万人、そのうち組込みソフトウェアの規模が2.4兆円、人口23.5万人と聞いている。日本にとって重要な製造業においては、デジタル化の進化により組込みソフトウェアが大変重要な位置を占めるようになってきた。組込みソフトウェアは急速に複雑かつ巨大化しているので、見えない、理解できないという課題に加えて、開発側での負荷が尋常ではない。

SECの役割は、産官学をつなぐ要の位置にあって、経営者からも現場からも学問的にもソフトウェア経営課題が理解できるようにすること、また、合理的で効率的な開発が実現できるよう推進することである。ソフトウェア開発の課題を着実に解決していく活動が論文で公表されることは、技術力の底上げに大いに役に立つ。

今年の優秀論文は3件とも、ソフトウェア開発の本質的な課題に正面から取り組み、効果または展望を明示している、まずはそのことに敬意を表したい。

各論文にコメントを付記し今後に期待したい。「WBSに基づくプロジェクト管理システムの実現」については、プロジェクト管理を更に極める工夫を継続して改善されることを期待する。「組込みソフトウェア設計検証へのモデル検査技術の適用と考察」については、今後、検証実績を積み重ね、実用に供せられるよう、その進展に期待したい。「メモリリーク検出システムλ trace」については、多くの効果実績の更なる深彫りに期待する。

今回審査対象となった他論文も充実してきたと聞いている。地道な活動が真の進化につながり、業界全体を着実に底上げするものと確信して、参加者全員に敬意を表したい。

トヨタ自動車株式会社
常務取締役
重松　崇

第３回目となる今回の審査対象論文も、これまでと同様、実務目線のテーマが多く掲載されており、読者にとって利用価値の高いものとなっている。また、今回は研究レベルの提案から、現場で実証された手法の紹介まで幅広く取り上げられており、SEC活動の広がりを感じさせられる。

最優秀賞に選ばれた「組込みソフトウェア設計検証へのモデル検査技術の適用と考察」は高い市場品質と多数のバリエーション開発を要求される、我々自動車用組込みソフトに不可欠な検査手法となりうる新たな提案である。その実現の可能性追求とそれに伴う実証研究の進展を大いに期待したい。

また、他の２編の論文は開発現場でその有用性が実証されたものであり、波及効果の高い内容となっている。実務レベルで上手く活用できた事例を公開し、広く普及を図ることも「SEC journal」誌の大きな使命であり、今後もその活動が拡大していくことを支援したい。

SECはこの「SEC journal」をはじめ多くの活動成果を冊子や書籍として「見える化」している。この３年間の活動は出版物の内容に沿ってレベルアップし、現在はソフトウェア技術分野のかなり広い部分をカバーしている。関係者の方々の高い志がこれらの足跡を残してきたものと頭が下がる思いである。

一方で、自らを振り返って自動車産業の中でのソフトウェア技術の進展を見ると、未だ、現場に根ざした基盤作りすら充分にできているとは言い難い。今後一層、電子化が加速される自動車を一段高い商品に成長させるためにも、SECの成果を如何に現場に取り込むかが大きな課題となってくる。SECの活動への協力と共に、日本の自動車産業の核となるソフトウェア人材の育成にも努力したい。

日本IBM株式会社　技術顧問　兼
東京大学大学院工学系研究科　特任教授
冨永　章

膨れ続ける組込みソフトウェアに対し、Functional Safetyへの要請は高まる一方で、とりわけ上流での検証は強く望まれる。「組込みソフトウェア設計検証へのモデル検査技術の適用と考察」は、検証エンジン専用言語でなく、一般的なUMLのモデル記述を検証する素晴らしいチャレンジである。実用へ向けた更なる前進を大いに期待したい。

「メモリリーク検出システムλ trace」は、利用が広まっているCBMCの適用例を提供している。メモリリークに特化した検出を容易にする各種の工夫をしただけでなく、成果や方法の共有へ向けてわかりやすく述べた点が有益である。

「WBSに基づくプロジェクト管理システムの実現」は、WBSを用いWebで共有するPMS（プロジェクトマネジメント・システム）を開発し、多数プロジェクトに適用した。市場に多々あるPMSに比し当システムが特別とはいえないが、日本の適用業務プロジェクトで意外に普及が遅いPMSを敢えて実践し、改善努力をした点は評価できる。

さて、ソフトウェア業は人気衰退とか新3Kとすらいわれることがある。しかし、世界的には職種専門化や層別が進み、日本でも関係者の努力によるITSSの普及や新カリキュラムJ07の作成等、よい兆候は多々ある。現に米国の大卒初任給では、理系、特にIT各専攻は他を圧する地位となっている。コンピュータエンジニアリングをはじめ、コンピュータ科学、情報科学、情報システムのどれもがトップグループに位置し、文系専攻の実に1.5倍である。グローバルを俯瞰した適材適所、プロフェッショナル化、選択・集中は、当分野の価値を高める鍵であろう。とりわけ今回の論文テーマが属するような重要分野への更なる選択・集中の必然性と、SECの成果への期待は、今後ともますます高まるだろう。

株式会社日経BP
制作室長
横田　英史

日本のソフトウェア産業は大丈夫なのか。技術が錆びつき、産業構造が劣化しているのではないか。

技術ジャーナリストという職業に就いて20年になるが、こうした思いがどんどん強くなっている。ソフトウェア産業の衰退という話になると、エンタプライズ系に注目が集まることが多い。しかし、鉄道など社会インフラで起こった最近のシステム障害をみると、組込みソフトウェア産業も対岸の火事だと眺めているわけにはいかない。間違いなく足元に火がつき始めている。

産業構造の劣化は生活習慣病のようなものである。長年染みついた習慣が、産業の活力を減退させ、存続を脅かす。産学を挙げての地道で時間をかけた体質改善活動が求められる。このとき問題となるのが、産と学のギャップである。ギャップがありすぎ、「どこから手をつけていいかわからない」といったことになりかねない。

こうした観点で審査対象になった論文を読むと、１つの傾向に気づく。実証と実践にかなりの力点を置いているところだ。例えば「WBSに基づくプロジェクト管理システムの実現」には3,000件の適用事例があるし、「メモリリーク検出システムλtrace」では5,000万行のコードで検証を行い、いずれも貴重な知見を得ている。「組込みソフトウェア設計検証へのモデル検査技術の適用と考察」も、「形式検証」と「UML設計」という現場で関心の高いタイムリなテーマを扱っている。３つの論文からは、産と学のギャップを埋めようとする努力がくみ取れる。

論文を審査して強く感じたのが、エンタプライズ系と組込み系の技術の両方に触れられることの重要さである。組込み系では現在、情報処理の比重がどんどん高まっている。これから、エンタプライズ系の知見が役立つ場面が間違いなく多くなる。産学官の連携の“触媒”の役割を果たすSECの“場”をうまく利用し、日本のソフトウェア開発の国際競争力を高めていただきたい。

原田　晃氏

岸　知二氏

青島　武伸氏

『SEC journal』既掲載論文一覧

SECは、2007年10月に三周年を迎えました。ここにこれまでの掲載論文をまとめます。

No.1

「SEC journal」創刊記念招待論文

●ソフトウェア開発負荷見積り式の汎用化の提案

冨永　章

ソフトウェア開発負荷見積りのための「アルゴリズム的見積りモデル」は，1970年代後半から多くが提案され，各種が実用化されてきた．過去のモデルは負荷がサイズの累乗に比例する形式であるが，用いる指数（累乗，Exponent）の値または範囲が，各モデルにより異なる．

このためサイズの領域によっては，各モデルの間で見積り結果に大きな差が生じる．この乖離は，サイズの累乗に比例する式を用いる限り，どのようにしても避けられない．そのため，広いサイズ領域を取扱う実ビジネスでの見積りにおいては，個別に色々な工夫をしないと過小見積りや過大見積りの原因となる．

筆者はこの不都合を回避するために，どのような形の式が適切かを研究・考案し，企業内で多数の顧客向けプロジェクトで実際に使用してきた．本稿では，考案した「見積り汎用式」を導出した過程と結果を示す．この式は，過去12年間に少なくとも1万件を超える大小の開発負荷見積りに用いられ，適合してきた．一方この間に開発環境が多様化してきているから，環境の反映方法に関しては，今後さらなる研究と改善が望まれる状況である．

今般，独立行政法人 情報処理推進機構にソフトウェア・エンジニアリング・センターが設立されたのを機会に，この汎用式を一般の使用に供すべく，本誌面をお借りして初めて広く提案するものである．

「SEC journal」創刊記念招待論文

●最新ソフトウェア技術による高信頼組込みソフトウェアの開発

片山　卓也

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

No.2

●エンピリカルソフトウェア工学の現状と展望：SELが遺した13の教訓

松本　健一

ソフトウェアやその開発過程から得られる定量的データに基づいてソフトウェアの生産性や品質の向上を目指す実証的アプローチ（エンピリカルアプローチ）が注目されている．本稿では，米国SEL（Software Engineering Laboratory）の研究者たちが遺した「プロセス改善におけるエンピリカルアプローチに関する13の教訓」を紹介すると共に，それら教訓と対比させる形でEASE（Empirical Approach to Software Engineering）プロジェクトの現状とSECへの期待について述べる．

●ソフトウェア開発の生産性管理に基づく見積りモデル

太田　忠雄

株式会社ジャステックは，創業以来，役務提供を前提とした人月契約から脱却し，一括請負契約を拡大すること，および生産性原理に立脚した能力主義を実践するために，独自の生産管理システムを構築した．

生産管理システムの中心となるのがソフトウェア開発の見積りモデルである．見積りモデルは，開発計画の精度を高め，開発計画達成へのコントロールを可能にし，さらには開発プロセス改善の機会を創出するために必要である．

そのような要求を満たす見積りモデルとして，ジャステックは生産物量と生産性を変数とした基本アルゴリズムを考案した．生産物量とは，ソフトウェア開発の各工程で作成する生産物の量であり，生産性とは生産物単位のスピード，あるいはコストである．基本アルゴリズムは，生産物量見積り方式および生産性見積り方式から成り立っている．また，それぞれの方式において，開発環境の違いや品質要求の多寡による変動を吸収する「環境変数」と呼ぶパラメータを導入している．さらに，基本アルゴリズムを拡張し，改造型開発の見積りモデルおよび仕様変更の見積りモデルを構築した．

本稿がソフトウェア開発産業発展の一助となれば幸いである．

●組込みシステムとユーザビリティ工学

平沢　尚毅

ユーザビリティ工学が目指す品質概念として，利用品質という考え方を説明した．利用品質は，組込みシステム市場の今後の発展を考える上で，欠かせないものである．その上で，高い利用品質の水準を達成するために，組込みシステム開発が直面する課題に対して，ユーザビリティ工学が貢献する可能性が大きいことを示した．組込みシステム開発プロセスに，ユーザビリティ工学を取込むことによって，国際的に競合できる新しいものづくりを確立することができる．そのためには，人材育成に取組み，そのための組織的な基盤作りを緊急にしなければならない．

No.3

●プロジェクトデータ分析の指針と分析事例

古山　恒夫

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．

●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

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．

●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

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．

No.4

「SEC journal」創刊記念論文 最優秀賞受賞論文

●開発現場の実態に基づいた
ピアレビュー手法改善と改善効果の定量的分析

小室　睦，男澤　康，木村　好秀

最初に現状分析を実施し，ピアレビューの品質向上・原価低減への効果及びレビュー手法によるパフォーマンスの差異を現場の実データにより明確にした．この知見に基づき効果的なレビュー手法と統計的分析手法の教育を展開した．さらに，プロジェクト自ら分析を実施できる分析ツールの提供を行った．

これらの施策の結果，欠陥の捕捉率が高まり，レビュー効率も4～5倍に向上する等の成果を上げることができた．

「SEC journal」創刊記念論文 優秀賞受賞論文

●実践型ＥＶＭを活用したプロジェクト管理の適用研究

竹本　昇司

EVMは，タイムマネジメント，コミュニケーションマネジメントの手法としてPMBOKで取り上げられている．しかしながら，日本国内でのソフトウェア開発プロジェクトでは価値の算出の難しさからEVM利用の普及が進んでいないのが現状である．

本研究では予定価値，実績価値の算出を効率的に行うために日本国内でのソフトウェア開発プロジェクトに則したEVM手法を考案した．これを実践型EVMと呼び，その効果について報告する．

「SEC journal」創刊記念論文 優秀賞受賞論文

●プロジェクト混乱予測システムの
ベイズ識別器を利用した開発
―ソフトウェア開発現場への本格導入を目指して―

水野　修，安部　誠也，菊野　亨

我々は，これまでロジスティック回帰分析に基づくプロジェクト混乱予測を行ってきた．本研究ではソフトウェア開発現場への本格導入を目指して，ロジスティック回帰分析に残っていたいくつかの問題を解決した，ベイズ識別器を利用する混乱予測手法を提案する．まずベイズ識別器に基づく6つのデータマイニング手法に対して精度比較実験を行い，最も精度の高い手法を選択した．次に，ベイズ識別器を利用した提案法の有効性を実際の開発現場から得られたデータに適用することで確認した．具体的には，ロジスティック回帰分析では混乱予測を誤ったプロジェクトのすべてについて提案法では正しい予測結果を得ることができた．

2006年優秀賞受賞論文

●大学における社会人向け組込みソフトウェア
技術者人材養成の実施と分析

山本　雅基，阿草　清滋，間瀬　健二，高田　広章，河口　信夫，
冨山　宏之，本田　晋也，金子　伸幸

近年，企業において組込みソフトウェア開発が増大している．これに伴い，企業における組込みソフトウェア技術者人材養成の必要性が高まっており，大学に対する期待も大きい．我々は，大学において社会人向けの組込みソフトウェア技術者人材養成を，職種と技術レベルに対応した短期集中型として実施している．初級，中級，上級の各技術レベルに対応した8種類のコースを実施したところ，初級技術者教育の必要性が高いこと，実務能力を育てるためには体験型学習が有効であること，教育における上司の役割が大きいこと等がわかった．また，産学官のそれぞれの立場における社会人教育に対する課題を整理し，学としては初級技術者の育成強化と，実務能力を育成する教育の提供が必要であること等がわかった．

No.5

●第４世代のテスト・プロセス

山浦　恒央

ソフトウェア開発で最新のテスト・プロセスが第3世代である．これは，開発部門とは独立のQA部門がテストをする方式で，高レベルの品質保証が可能となる．ソフトウェア開発は，世界的に短納期傾向にあり，高品質を維持しつつ，テスト工数，開発工数，納期を短縮する方法として，第4世代のテスト・プロセスを提案する．これは，！品質開発マネージャを軸にしたテスト・プロセスの導入，"開発側のデバッグと，QA側のテストの一体化による工期と工数の削減を骨子とするものである．

2006年最優秀賞受賞論文

●企業横断的収集データに基づく
ソフトウェア開発プロジェクトの工数見積り

大杉　直樹，角田　雅照，門田　暁人，松村　知子，松本　健一，
菊地　奈穂美

プロジェクトの工数見積のための基礎データとして，複数の異なる組織で収集されたソフトウェア開発プロジェクトのデータを利用できれば，蓄積データの少ない組織においても精度の高い見積りができると期待される．

本論文では，1つのケーススタディとして，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・センターが収集した多数のソフトウェア開発プロジェクトのデータを用いて工数見積を行い，その精度を評価した．見積手法として，重回帰分析，対数重回帰分析，ニューラルネット，協調フィルタリングを用いた．評価の結果，協調フィルタリングは他の手法よりも高い精度を示し，実績工数に対する見積値の相対誤差の平均値が0.642，Pred25（相対誤差が0.25以下のプロジェクトの割合）が30.1%であった．

●通信ソフトウェア開発におけるプロセス改善のための
フィールド品質に注目した主要な活動要因の抽出

菊地　奈穂美，安藤　津芳，水野　修，菊野　亨

本論文では，高信頼性が要求される通信ソフトウェアの開発プロジェクトに対し，プロダクトのフィールド品質を改善するのに有用な開発活動に関する要因を特定する試みについて述べる．ここでは，プロダクトのフィールド品質をそのプロダクトをリリース後の6ケ月間にユーザから報告された問題（クレーム）の数で判断する．提案するアプローチでは，まずフィールド品質の良しあしでプロジェクトをGoodとFairの2つのグループに分類し，それぞれのグループの代表的なプロジェクトへのインタビューを行った．その結果明らかになったフィールド品質と関わりのある4つの問題について，それを計測するメトリクスを決定した．引き続いて，メトリクスデータを収集し，フィールド品質との関連をロジスティック回帰モデルで記述した．次に，24個のプロジェクトを対象とし

て評価実験を行い，品質に強い影響を与えると思われる主要な要因として，レビュー活動，母体の品質の改善，及び仕様に基づく活動の3点を確認した．さらに，モデルによるフィールド品質予測の精度を評価する実験を行ったところ，精度が約96％になることを確認した．

No.6

論文の掲載はありません

No.7

●ハイブリッドなコスト見積りモデルの反復的な構築方法について

Adam Trendowicz, Jens Heidrich, Jürgen Münch

石谷　靖，横山　健次，菊地　奈穂美（翻訳：石谷　靖）

コスト見積りは，ソフトウェア開発企業にとって非常に重要な活動である．そして，見積り技術の導入に際しては，見積り精度が重要な判断材料となる．しかしながら，精度の評価はただ一度の試行で決定されることが多く，当該見積りモデルの改良の可能性は，採用の判断に当たって考慮されないことが多い．さらに，見積り方法の多くは，反復的な構築による見積りモデルの改良を明確には意識していない．これは，重要なコスト要因またはデータ間の矛盾を見落とす危険性を増加させる原因となる．本論文は，CoBRA 法に基づくコスト見積りモデル構築に対して，反復的な分析，フィードバックサイクル及びその評価を系統的に導入し，拡張したプロセスを提示するものである．沖電気工業株式会社のソフトウェア開発部門でモデル改良サイクルを実施し，見積り精度は，当初のモデルで120%の誤差であったが、最終的なモデルで14%に向上した．本論文では，反復的なモデル構築アプローチによって得られた知見を示す．

No.8

2006年優秀賞受賞論文

●コードレビューの密度と効率がコード品質に与える影響の分析

中野　裕也，水野　修，菊野　亨，阿南　佳之，田中　又治

ソフトウェアプロジェクトにおいてレビューを行うことは品質確保の面から重要な作業とされている．一方でレビューには相応の工数がかかるため，コストの面からはあまり長い期間レビューをすることは好ましくない．しかし，レビューを行う効率を高くしすぎるとレビューが乱雑になり，最終的な品質の向上があまり期待できない．そのため，レビューの効率とレビューによる指摘密度の関係について分析し，これらが後工程での品質にどう影響を及ぼしているかについて調べることで，適切なレビュー効率でレビューが行われているかを判定する条件を発見できると考えられる．

まずSEPGが関連性があると感じていた2つのメトリクス（レビュー効率とレビュー指摘密度）に注目した．そして，これらのメトリクスに関連した仮説「レビュー指摘密度がある条件を満たさないプロジェクトは，後工程での品質が悪くなる傾向がある」を示すことを本研究の目的とする．

この仮説を検証するために，実際の企業の開発現場から収集された500件のプロジェクトデータを使用して分析を行った．その結果は，レビュー効率に対してレビュー指摘密度が大きいプロジェクトでは後工程での残存不具合数は多くなる，というものとなった．よって，この仮説の正しさを確認できた．また，実際の開発現場への適用法も提案した．

2006年優秀賞受賞論文

●Java開発者のオンデマンド・ラーニングを支援するソシオテクニカル環境

葉　雲文，山本　恭裕

Java 開発において大規模なクラスライブラリの利用は不可欠である．開発者がそれらのライブラリをいかにして習得するかは大きな課題である．本稿では，Java 開発者がクラスライブラリをオンデマンド的に習得するためのソシオテクニカルな支援環境を提供することを目的として，開発者同士による知識交換の場としてのダイナミックコミュニティの形成を，エキスパート同定および選定のプロセスを通して行う仕組みを提案し，その仕組みを実装したSTeP_INシステムを報告する．

No.9

2007年優秀賞受賞論文

●WBSに基づくプロジェクト管理システムの実現

原田　晃，粟根　達志，伊野谷　祐二，大里　立夫，大野　治，松下　誠，楠本　真二，井上　克郎

業務ソフトウェアの開発プロジェクトにおいては，プロジェクト管理が重要になってきている．一方，そのようなプロジェクトは，規模が大きい場合が多く，プロジェクト管理は非常に負荷の大きなものとなっており，効率的なプロジェクト管理を支援するシステムが必要となる．本研究では，業務ソフトウェアの開発プロジェクトにおいて，工程，作業，成果物，参考資料等を相互に関連付けし，一元管理するためのWBS（Work Breakdown Structure）モデルの作成と，そのWBSモデルを利用したプロジェクト管理システム「プロナビ」を開発した．プロナビを多数のプロジェクトに適用し評価した結果，プロジェクトを効率よく進める上で有効であることを確認した．

No.10

論文の掲載はありません

No.11

2007年優秀賞受賞論文

●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．

No.12

2007年最優秀賞受賞論文

●組込みソフトウェア設計検証へのモデル検査技術の適用と考察

岸　知二，金井　勇人

組込みソフトウェア開発において設計品質の向上は重要課題であり，設計検証に対する形式検証技術の適用が期待されている．一方，多くのソフトウェア技術者にとって形式検証技術は新しい技術であり，わかりやすい適用方法の整備が求められている．本稿ではモデル検査技術を利用したカーオーディオのソフトウェアの設計検証を題材に，ソフトウェア設計手法の分野での既存のモデル化技術を活用した設計検証の事例を紹介するとともに，設計検証のアプローチについて考察する．

非機能要求記述への取組みについて

SEC企画グループ
研究員
塚本　英昭

昨今の大規模化・複雑化が進むシステムの開発では、上流工程の重要性がますます増している。例えば、開発プロセス共有化部会の成果をまとめた書籍「経営者が参画する要求品質の確保」[SEC2005]は、IT書籍の中では突出した部数を印刷しており、同じく同部会の成果として10月に刊行された書籍「共通フレーム2007」[SEC2007]でも、要求品質を向上させるために、ステークホルダ間で明示的に要求を決定・合意・承認するための要件定義プロセスを新たに追加している。

また、今までは要求というとシステムの機能要求を中心に検討がなされてきたが、それだけではユーザが十分満足するようなシステムは完成しない。ユーザが十分満足するシステムを構築するためにはシステムの機能以外の要求、すなわち、非機能要求も重要な要因となるからである。経済産業省が2006年6月に公表した「情報システムの信頼性向上に関するガイドライン」[METI2006]でも、ユーザとベンダ間で非機能要求の実現に向けた合意をとることが重要であると述べている。

SECの要求工学・設計開発技術研究部会 非機能要求とアーキテクチャWGでは、このような背景から非機能要求に対象を絞って検討を進めてきた。現在も引き続き検討は進めているが、2007年９月に2006年度の活動を中心に進捗をまとめた『非機能要求とアーキテクチャWG 2006年度活動報告書』をSEC-WEBサイトに公開した。（https://sec.ipa.go.jp/download/files/report/200708/NFR-Arch_report_2006.pdf）

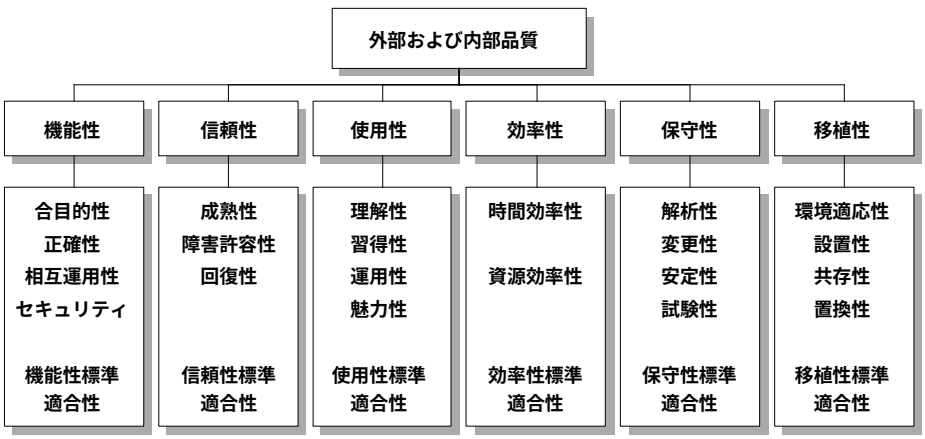
今回は、この活動報告書の概要について説明する。

1. 非機能要求の分類

非機能要求というと、すぐに思いつくのは性能や信頼性、セキュリティ、使用性（ユーザビリティ）等の品質を向上させる要求ではないだろうか。しかし、システム化の目的が業務効率を向上させることからビジネス価値を向上させるという一段上のレベルで捉えられることが多くなってきたため、ビジネスというさらに広い視点で非機能要求を捉えると、例えば以下のような要求が非機能要求として分類される[KARL2003]。

①業務ルール

会社方針、政府規制、業界標準、会計慣行、計算アルゴリズム



出典：JIS X 0129-1: 2003（ISO/IEC 9126-1: 2001）

図１　ISO/IEC 9126（JIS X 0129）品質特性の構造

②品質特性

例えば、使用性、移植性、完全性、効率性、堅牢性

③外部インタフェース

システムと外界との間の取り決め

④制約

設計と実装のために開発者が利用できる選択肢に制限を課すもの

また、品質特性に絞った非機能要求の分類では、ISO/IEC 9126（JIS X 0129）が有名である。ISO/IEC 9126（JIS X 0129）は6つの品質特性、さらにそれらを詳細化した品質副特性を定義している（図 1）。セキュリティが品

表 1　品質特性とメトリクス/評価指標に対するWGの所見（活動報告書の一部を抜粋）

品質特性		ソフトウェア品質評価ガイドブックの測定法/メトリクス例	WGの所見	
			メトリクス/評価指標	アーキテクチャによる解決
主特性	副特性		要求記述の書き方	その他の方法による解決
信頼性	成熟性	平均故障発生間隔（MTBF）（注1） 障害収束率 障害密度（注2）	同左	高凝集性・疎結合 コンポーネント設計 オートノミック設計
			なし	ソフトウェア開発プロセス
	障害許容性（注3）	ダウン発生率 誤入力・誤操作検出率	障害時許容限界 サービスレベル	フォールトトレラント設計 フェイルセーフ設計
			フェイルケース ミスユースケース セーフケース	システム運用プロセス
	回復性	稼働率 平均ダウン時間 平均回復時間	同左	バックアップ設計 リカバリー設計
			フェイルケース	システム運用プロセス
効率性	時間効率性	処理時間（TAT） CPU実行時間 スループット トランザクション処理件数	応答時間 処理時間（TAT） バッチウインドウ 伝播遅延時間 スループット トランザクション処理件数（TPS）	ハイパフォーマンス設計 スケーラブル設計
			オンピークケース オフピークケース チェンジケース フェイルケース	
	資源効率性	主記憶使用量 主記憶使用率 ファイル使用率	資源使用量 資源使用率 同時利用数	バーチャライゼーション IT標準遵守
			オンピークケース オフピークケース チェンジケース フェイルケース	ITガバナンスプロセス

注意：

(1) ソフトウェアは、ハードウェアのように部品が磨耗や疲労して故障（Fault）することがない。ソフトウェアの故障は、要求仕様にない想定外の使い方をした場合、および、要求、設計、開発、テストの過程で見過ごされた不具合（バグ）が顕在化した場合である。

(2) 故障（fault）が起因となってシステムが誤った動作をすること、システム全体、または、一部の機能を失うこと、システムの品質が低下することを、一般的に故障（Fault）と区別して障害（failure）と呼んでいる。

(3) 障害許容性（fault tolerance）とは、システムの一部が故障しても完全に機能を保ったまま処理を続行できる、または、許容限界レベルに性能を低下させても続行できることである。ソフトウェアの場合、ハードウェアのように同じ部品の冗長化ではなく、同じ要求仕様に対して異なる設計・実装による部品の多様性が求められる。また、故障や想定外の誤操作により障害が発生した場合でも、必ず安全側にシステムを制御するフェイルセーフ（fail safe）の要求も含んでいる。

要求記述の書き方は、次のようにまとめられる。

- ユースケース……想定内の利用シナリオ
- ミスユースケース……想定外、誤操作、故意・悪意の利用シナリオ
- チェンジケース……将来の変更シナリオ
- テストケース……テストのシナリオ
- フェイルケース……障害時のシナリオ
- セーフケース……フェイルセーフのシナリオ
- ポーティングケース……移植、再利用のシナリオ
- オンピークケース……繁忙期のシナリオ
- オフピークケース……閑散期のシナリオ

質特性「機能性」の副特性になっていることは興味深いが、我々は階層構造については気にせず、このような分類を品質特性の網羅性を検討するときに利用できれば有効であると考える。

非機能要求とアーキテクチャWGでは、このような広範囲な非機能要求のうち、具体化の過程を経て個々の実現手段やその組み合わせを選択した結果として、後のアーキテクチャ設計へ密接につながる品質特性に対象を絞って検討を進めた。今回、ISO/IEC9126（JIS X 0129）より測定可能なメトリクス/評価指標を整理し、要求記述の書き方・アーキテクチャによる解決法、その他の方法による解決法をまとめた。表 1は品質特性の一部のみであるが、それを整理したものである。すべての品質特性についてまとめたものをご覧になりたい方は活動報告書を参照していただきたい。

2. 非機能要求記述における現在の技術

非機能要求からアーキテクチャ設計へと進めるためには、例えばどのように非機能要求を抽出するかという非機能要求獲得技術、非機能要求をどのようにアーキテクチャ設計につなげるかという方法論、あるいはトレーサビリティの技術、そのほかにも様々な技術が必要である。そこで、非機能要求とアーキテクチャWGでは、ユーザ、ベンダを含む利害関係者が意思疎通を図る上で基礎的な要素である非機能要求の記述を最初の中心テーマとして議論を進めた。

非機能要求記述の主要な現状技術として、調査したものは以下の3つである。

表2　主要な現状技術

主要な現状技術	概　要	非機能要求記述方法
Planguage	Tom Gilb氏が提唱する工学手法を説明するための体系。 仕様言語とプロセス記述・手法から構成される。	非機能要求はパフォーマンス要求、資源要求として扱う。 汎用パラメータ（用語のタイプ、ステークホルダ等）やスカラー特性（測定単位、目標値等）の要求記述項目が定義されている。
SEIのアーキテクチャ中心方法論	カーネギーメロン大学ソフトウェアエンジニアリング研究所（CMU/SEI）で研究開発されたアーキテクチャを設計するための方法論。 主な品質特性要求を発見・洗練・具体化するQAW（Quality Attribute Workshop），アーキテクチャの初期設計を行うADD（Attribute Driven Design），アーキテクチャ設計のレビューを行い、設計を洗練させるARID（Active Review for Intermediate Designs），アーキテクチャ設計の妥当性を確認するATAM（Architecture Tradeoff Analysis Method），費用・便益の視点でアーキテクチャ設計における費用対効果を分析するCBAM（Cost Benefit Analysis Method）から構成される。	アーキテクチャ設計を評価可能な形で、具体的な場面や機能を特定して品質特性要求を記述する要求シナリオを定義している。 記述項目は、 1. 刺激の発生源（Source of Stimulus）：刺激を発したエンティティ（人、システム等） 2. 刺激（Stimulus）：システムへ影響のある条件 3. 成果物（Artifact）：影響を受けるシステムの部分 4. 環境（Environment）：刺激が到着した時や応答しようとするときに成果物の置かれている条件・状態 5. 応答（Response）：影響を受けた結果としての対処 6. 応答測定（Response Measure）：応答した結果、測定可能な計測値の提示 の6つの部位から構成される。
NFRフレームワーク	カナダトロント大学のLawrence Chungが提唱する非機能要求を分析・表現するためのフレームワーク。 ゴール指向分析の概念を利用して非機能要求とそれに対する実現方法の間の関係を分析し、体系的に記述する方法を提案している。	境界の定義や満足・不満足の判断基準が不明確な「ソフトゴール」という概念を導入し、それを構成要素とした相互関係をSIG（Softgoal Interdependency Graph）という図に表現する。

・Planguage
・SEIのアーキテクチャ中心方法論
・NFRフレームワーク

表 2は、非機能要求記述の観点でこれらの現状技術をまとめたものである。

2.1 Planguageの記述例

図 2は、汎用パラメータやスカラー特性を使用した携帯電話機能のサービス性を詳細化した例（一部を抜粋）である。

携帯電話の機能：
サービス性：
修理：
目標（Ambition）：欠陥に対して利用者が十分満足するような修理のスピードに改善する。
尺度（Scale）：欠陥が発生してから利用者が満足して利用できるように修理するあるいは交換する時間。
実地測定（Master）【製品受取】：正式な実地試験を少なくとも20ケース実施、【実地の監査】：予告なしにランダムに実施試験実施。
現状レベルのベンチマーク（Past）【製品＝電話XYZ、国内市場、正規ディーラー店】：正規ディーラー店で0.１時間＋正規ディーラー店から搬送するのに0.9時間。
最新技術レベルのベンチマーク（Record）【競合製品XX】：平均0.5時間。利用者の会社に代替品を送るため。

図 2　Planguageの記述例

表 3　SEIアーキテクチャ中心方法論の記述例

シナリオの部位	シナリオにあてはまる可能性のある値
刺激の生成元（Source of Stimulus）	システムの内部、システムの外部
刺激（Stimulus）	障害：欠落、クラッシュ、タイミング、反応・応答
成果物（Artifact）	システムのプロセッサ、通信チャネル、永続ストレージ、プロセス
環境（Environment）	通常運転中、デグレードモード運転中（例：少ない機能、縮退運転）
反応（Response）	システムはイベントを検出し、下記の1つ以上のことを行う： ・記録する。 ・適切な相手へ通知する。ユーザや他のシステムを含む。 ・イベントの原因を定義されたルールに従って無効化する。 ・事前に明記した期間、使用不可の状態になる。期間の長さはシステムの重要度によって異なる。 ・通常モード、あるいは、デグレードモードで継続運転する。
反応の計測値（Response Measure）	システム復旧までの時間間隔（MTTR） 稼動時間 システムがデグレードモードで運転していることができる時間間隔 修復時間

2.2 SEIのアーキテクチャ中心方法論の記述例

表3はSEIのアーキテクチャ中心方法論で定義している品質特性「可用性（Availability）」を本方法論の要求シナリオを元に汎用シナリオとして記述した例である。

2.3 NFRフレームワークの例

図3は、品質副特性のセキュリティに対して、NFRフレームワークで定義しているSIG（Softgoal Interdependency Graph）で表現した例である。

また、NFRフレームワークでは、NFRソフトゴール（ISO/IEC 9126（JIS X 0129）の品質特性に対応）とそれを実現するための手段との間の相互依存関係をカタログという概念で整理している。表4は6つのNFRソフトゴールと5つの実現手段との間の相互依存関係を整理した例である。

3. 非機能要求からアーキテクチャ設計へ

現在、非機能要求とアーキテクチャWGでは、これまで調査した現状技術の優れた部分を利用、融合させて2007年度の成果として公開できるように非機能要求記述ガイドを作成している。主な変更点は以下のとおりである。

・Planguageの汎用パラメータやスカラー特性のようなタグを、WGで議論したタグに置き換えて、個々の品質

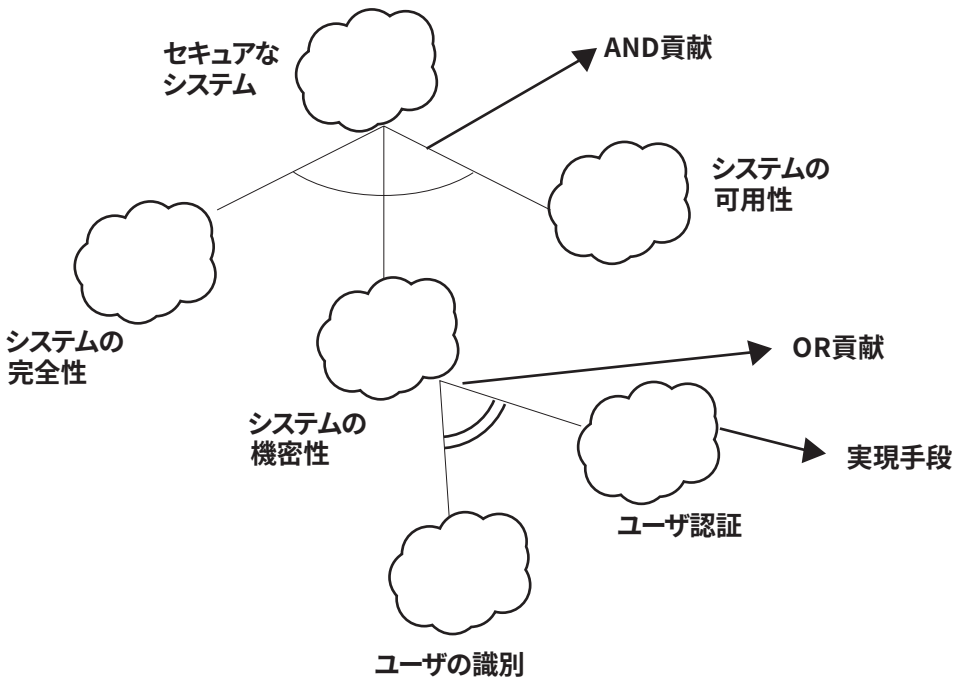


図3 NFRフレームワークのSIGの例

表4 NFRフレームワークの相互依存関係カタログの例

	NFRソフトゴール				
手段	正確性	機密性	応答性	資源効率性	使用性
妥当性評価	＋	＋	−		
圧縮			−	＋	
索引			＋		
認証		＋			
追加ID		＋			

特性の記述として定義する。

・SEIのアーキテクチャ中心方法論のシナリオの考えを非機能要求の定義や、そこからアーキテクチャ設計へとつなげる方法論に取り入れる。

・NFRフレームワークのSIGをユーザがより分かりやすいUMLクラス図として再定義し、さらに品質特性と実現手段間の相互依存関係の表の各項目やトレードオフの関係を洗練させる。

図4はこのような変更点を加味して、現在検討中の非機能要求からアーキテクチャ設計に進むためのプロセスの一部である。

このガイドラインにより、要件定義での非機能要求の獲得や記述、及びアーキテクチャ設計への取り組みに少しでも有効に働くことを願うものである。

参考文献

[SEC2005] IPA SEC：経営者が参画する要求品質の確保, オーム社, 2005

[SEC2007] IPA SEC：共通フレーム2007, オーム社, 2007

[METI2006] 経済産業省：情報システムの信頼性向上に関するガイドライン, 2006, http://www.meti.go.jp/press/20060404002/guideline-hontai-set.pdf

[KARL2003] Karl Wiegers：Software Requirements: Practical Techniques for Gathering and Managing Requirements Throughout the Product Development Cycle 2nd Edition, Microsoft Press, 2003
（渡部洋子訳：ソフトウェア要求（第1版訳）, 日経BP社, 2003）

[GILB2005] Tom Gilb：Competitive Engineering: A Handbook For Systems Engineering, Requirements Engineering, And Software Engineering Management Using Planguage： Butterworth-Heinemann, 2005

[SEI] Software Engineering Institute：
http://www.sei.cmu.edu/architecture/ata_method.html

[CHUNG] Lawrence Chung, Brian A. Nixon, Eric Yu, John Mylopoulos：Non-Functional Requirements in Software Engineering, Kluwer Academic Publishers (Springer), 1999

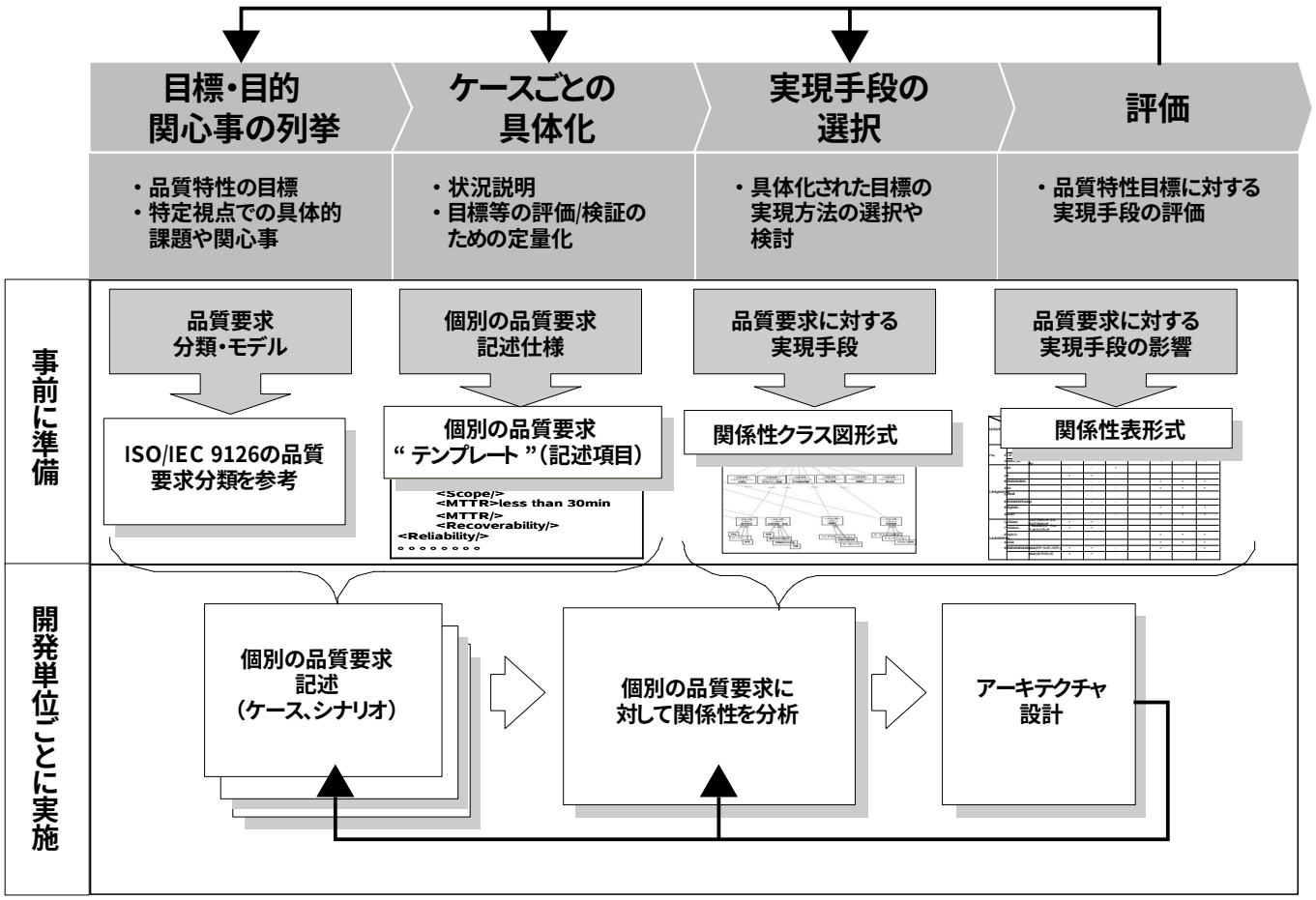


図4　非機能要求からアーキテクチャ設計へのプロセス例

見積り法COCOMO II概説

沖電気工業株式会社
菊地 奈穂美

株式会社 一（いち）
飯泉 純子

NTTソフトウェア株式会社
亀田 康雄

日本アイ・ビー・エム株式会社
細川 宣啓

日本アイ・ビー・エム株式会社
渡辺 千恵子

株式会社 一（いち）
大槻 繁

ソフトウェア開発の見積り予測やマネジメントの場面において、開発対象、プロセス、プロジェクトや人的特性に至る広範な分析が必要であり、COCOMO IIはそれに関する理論的基盤を与えている。この基本的な考え方とコストモデルを、原典に基づいて精確に解説する。

1. はじめに

1981年にBarry Boehm（現在南カリフォルニア大学（USC）教授）が“Software Engineering Economics”[BOEHM1981]を著し、その中でCOCOMO（COnstructive COst MOdel）が提唱された。以来、何世代も開発言語や環境、方法論や開発プロセスの変遷があるにも関わらず、基本的なコストモデルや考え方は普遍的である。近年、システムの経済合理性や、調達の公平性・透明性、科学的プロジェクトマネジメントが求められてきている中で、本質的かつ基本的コストモデルとしてのCOCOMOを基底とするのは有用である。多くのコストモデルや見積り手法が提唱されているが、主要な概念や方法を把握する上でも知識体系としてのCOCOMOを習得する意義は大きい。

著者らは、COCOMOの研究、実践的適用、見積り分析・評価他を専門とし、COCOMOを理解し有用性を認識して啓蒙的な活動等にも携わっている。国内でCOCOMOを主題にした日本語の書籍や解説が未だに少ないことから、多くの方々がCOCOMOを活用できるようにと考え、解説を試みた。とくに、本稿第4節の各種概念や要因等の訳語は、著者らによって議論を重ね調整した結果である。

本技術解説は、1981年提唱のCOCOMO 81[BOEHM1981]から、2000年発行のCOCOMO II.2000 [BOEHM2000]に至る約20年の研究成果を踏まえ、その概要、基本的な考え方、モデルの適用方法について概説する。第2節ではCOCOMOの概要と系譜を紹介する。第3節ではCOCOMOモデルの基底にある普遍的な基本モデル、予測手法について述べる。第4節は、より具体的でかつ現段階で最新のCOCOMO II.2000の基本式と予測手法、各種規模要因とコストドライバについて詳説する※1。また、モデル化手法も説明する。第5節は維持保守や再利用の場面や段階的拡充開発やコンポーネントベースの状況でのCOCOMOの適用や拡張について簡単に紹介する。

2. COCOMOの概要

2.1 COCOMOとは

COCOMOは、工数見積り及び開発期間の見積りモデルとして知られている。具体的には、ソフトウェアの規模（コード行数等）とコスト（開発工数等）、工数と開発期間との関係をモデル化した数式が提供され、また、プロジェクト実績データによってモデルを精密にしていくモデル化手法も提供されている。モデル式の作成方法は、式自体もすべて開示されており、モデル自体をCOCOMOユーザが自らの組織に応じて較正または拡張可能になっている。

モデル式には、システムやプロジェクトの特性による工数変動を規模要因、コストドライバとして盛り込んである。各種のヒューマンファクターも考慮できる。また、USCで収集した実際のプロジェクトデータによって求められたモデル式の係数値が公表されている。

このような特徴から、COCOMOは、見積りだけでなく、ソフトウェアに関して工数を要するような投資、コス

※1 [BOEHM2000]の表紙裏のまとめの式と評価値表に誤植があるので注意されたい。本文中の式や表が正しいものである。本稿では、正しい式と表を掲載する。

ト・開発期間・機能性・性能・品質要因等のトレードオフ検討、コスト・開発期間のリスク管理での決定、開発・購入・再利用等の意思決定、組織の改善等にも活用でき、応用範囲が広い。

2.2 COCOMOの系譜

Boehm教授は1970年代にTRW社（主に軍需、精密機器のソフトウェア開発企業）在籍時にプロジェクト実績データを基に作成したのが始まりで、Boehm教授が南カリフォルニア大学（USC）へ移籍後も継続し、1981年にCOCOMO 81が発表された。同じく軍需産業関連から発展した「Doty法（1977年発表）」とともに、見積り結果に対するアカウンタビリティを求められた軍需産業分野の文化が伺える。

COCOMO Ⅱは、COCOMO 81をベースとし、繰り返し型開発、オブジェクト指向開発、1980～1990年代のツールや開発環境変化等に対応し進化した。1995年に発表後、1998年、1999年、2000年とモデルが更新された。2000年に161件のプロジェクトデータに基づくUSC COCOMO Ⅱが発表された（「COCOMO II.2000」と呼ばれる）。COCOMO Ⅱは、近年のソフトウェアエンジニアリング技術や開発プロセスに対応して、自動生成や再利用等にも対応するようになった。COCOMO 81に存在したコストドライバのいくつかは古いため削除または定義の見直しがされたものもある。また、COCOMO Ⅱで新たに追加された要因もある（附表A）。さらに、5.2節に示すように、COTS （Commercial Off The Shelf）ソフトウェア等、最新のシステム構築の形態にも対応して拡張が進んでいる。

3. COCOMOの考え方

規模、工数、開発期間の基本的な関係は次の式で表される。

$$工数 = A \times (規模)^{B}$$
$$開発期間 = C \times (工数)^{D}$$

工数（人月）と規模（KSLOC： Kilo Source Lines Of Code；ソースコード行数）の関係は単純な線形の比例関係とは異なると仮定している。規模の指数であるBの値によって、工数と規模の関係は変化する。B<1であれば、規模が大きくなるにつれて生産性が向上する（規模が2倍になったとき、工数は2倍より少ない）。B=1であれば、規模が変わっても生産性は変わらない（規模と工数が比例する線形モデルとなる）。B>1であれば、規模が大きくなるにつれて生産性が低下する（規模が2倍になったとき、工数は2倍より大きい）。

COCOMO 81では、開発対象やプロジェクトの特性が工数に与える影響を表すためにコストドライバを、COCOMO Ⅱではさらに、規模の指数Bを可変に調整する規模要因を導入している。

開発期間は工数と関係があり、工数に応じて最適な開発期間が存在することを仮定している。したがって、COCOMOでは開発期間に関するコストドライバを導入して、最適な期間よりも短期間にする場合には工数が増加するモデルとなっている。

COCOMOを利用してプロジェクトを予測するときには、まず、プロジェクトの性質、予測時点での工程によってモデルの種類を選ぶ。次に、規模を予測する。また、規模要因、コストドライバを評価する。予測した規模、評価した規模要因、コストドライバをモデル式に代入し、工数、開発期間を得る。

4. COCOMO Ⅱの概要

4.1 基本式

COCOMO Ⅱには、Early DesignモデルとPost-Architectureモデルがある。Early Design、Post-Architectureで、工数の式のコストドライバの数は異なるが、工数及び開発期間の式は共通である。

① 工数

工数PMは次式で表される（単位：人月）。

$$PM_{NS}=A\times Size^{E}\times\prod_{i=1}^{n}EM_i$$

$$PM=A\times Size^{E}\times\prod_{i=1}^{n}EM_i+PM_{AUTO}$$

$$E=B+0.01\times\sum_{j=1}^{5}SF_j$$

$$PM_{AUTO}=\frac{Adapted\ KSLOC\times(AT/100)}{ATPROD}$$

A ：開発規模が開発工数に与える影響の大きさを表す定数。較正（calibration）によって求める。

AT ：自動変換されるコードの割合（%）。
再利用・適応する既存ソフトウェアを自動変換することによって適応先のソフトウェアに適合させることができるコードの割合（%）。

$ATPROD$ ：自動変換による作業のKSLOCでの生産性。

B ：工数予測式の規模指数の基底値。較正によって求める。

E ：工数予測式の規模指数。

EM ：コストドライバ（工数調整係数）を表す変数（Early Designは7個、Post-Architectureは17個）。

PM_{NS} ：標準工数（人月）。コストドライバのSCED無し（即ち、標準的な期間の場合）で、かつPM_{AUTO}無しで見積もった工数。

PM ：新規及び再利用コードの開発の工数（人月）。

PM_{AUTO} ：自動変換の活動の工数（人月）。

SF ：規模要因（5個）。

$Size$ ：規模　　※詳細は後述。

$Adapted\ KSLOC$ ：適応（改造）ソースコード規模（単位：KSLOC）。ホワイトボックスとして扱われ利用するために修正されうる既存コード。

② 開発期間

開発期間$TDEV$は次の式で表される。プロジェクトの要求確定後から検証終了までである（単位：月）。

$$TDEV_{NS}=C\times(PM_{NS})^{F}$$

$$TDEV=\{C\times(PM_{NS})^{F}\}\times\frac{SCED\%}{100}$$

$$F=D+0.2\times(E-B)$$

C ：工数が開発期間に与える影響の大きさを表す定数。較正によって求める。

D ：開発期間予測式の工数指数の基底値。較正によって求める。

F ：開発期間予測式の工数指数。

$SCED\%$ ：標準開発期間に対する割合（75%以上）。

$TDEV_{NS}$ ：標準開発期間（月数）。

$TDEV$ ：開発期間（月数）。

③ 係数

次のA,B,C,Dの値はCOCOMO Ⅱ.2000のものである。

$$A=2.94,\ B=0.91,\ C=3.67,\ D=0.28$$

COCOMO Ⅱ.2000版では、工数はソフトウェア開発プロジェクトに関わる活動に費やす時間（休日や休暇は含まない）で、1月=152時間を前提としている。

④ 規模

規模$Size$は次式で定義されている。式からわかるように規模は、新規に作成する量や、再利用や改造する量、自動生成する量、自動変換する量等を組み合わせて算出できる。新規のみの場合は、新規作成以外の規模はゼロであることから$Equivalent\ KSLOC$=0であり、$Size$は単純な式となる。

$$Size=\left(1+\frac{REVL}{100}\right)\times(New\ KSLOC+Equivalent\ KSLOC)$$

$$Equivalent\ KSLOC=Adapted\ KSLOC\times AAM\times\left(1-\frac{AT}{100}\right)$$

$AAF\leq 50$のとき、

$$AAM=\frac{AA+AAF\times(1+[0.02\times SU\times UNFM])}{100}$$

$AAF>50$のとき、

$$AAM=\frac{AA+AAF+(SU\times UNFM)}{100}$$

$$AAF=(0.4\times DM)+(0.3\times CM)+(0.3\times IM)$$

$Size$ ：規模。

$New\ KSLOC$ ：新規開発規模（単位：KSLOC）。

Equivalent KSLOC： 等価コード行数（単位：KSLOC)。
既存のソフトウェアコード（対象システムに統合されていない）を再利用又は修正して利用する場合に必要になる。

Adapted KSLOC ：適応（改造）ソースコード規模 （単位：KSLOC)。

AA ：改修有効性調査の割合（%)。
再利用ソフトウェアの評価等のための増加度（%）（0～8%）

AAF ：適応調整要因。
対象システムに結合する、既存ソフトウェアの変更による工数への影響を反映するために*DM*,*CM*,*IM*の3つの要因を予測した結果で表す。

AAM ：再利用ソフトウェアの修正を表す変数。

AT ：自動変換されるコードの割合（%)。

DM ：再利用・適応されるソフトウェアで、変更が必要な設計の比率（%)。

CM ：再利用・適応されるソフトウェアで、変更が必要なコードの比率（%)。

IM ：適応されたソフトウェアをプロダクト全体に対して結合するため、及び結合結果のソフトウェアをテストするための工数を同様のサイズのソフトウェアに対する結合とテストに必要な平均的な工数と比較しての比率（%)。

KSLOC ：キロ単位で表すコード行数。

REVL ：要件の変動性。
見積り以降の要件の追加や変更により廃棄されるコードの割合で、開発後稼働するソフトウェアのコード行数に対する比率（%)。

SU ：ソフトウェア理解度。再利用ソフトウェア理解のための増加度（%)。ソフトウェアの構造、アプリケーションの明解さ、コードと説明の自明さの3カテゴリの平均で決める。
VeryLow（50%）～VeryHigh（10%）の5段階。

UNFM ：プログラマ非習熟度。
プログラマが再利用ソフトウェアに精通していないことによる増加乗数 （0～1の間で評価)。 完全に精通（0.0）～全く馴染み無し（1.0）の６段階。

基本式で*Size*の単位はコード行数であるが、他にファンクションポイント等も変換式を用いて扱うことができる。コード行数の計測基準は[PARK1992]による。

4.2 予測手法

COCOMOでは、規模、コストドライバ（Cost Driver）と規模要因（Scale Factor）によって予測を行う。規模要因は、プロジェクトで扱う規模が大きいほど、より大きなオーバヘッドが必要になり、伝統的な開発手法では規模と工数に累乗の関係が認められるため、モデル式の曲線の形状を決めるものである。コストドライバは、対象のソフトウェア開発プロジェクトの特徴を表し、各プロジェクトの工数に比率として影響する。

COCOMO IIでは、Early Designモデルと Post-Architectureモデルがある。見積もる時期によって使用するモデルを選択することができる。

① Early Designモデル

このモデルは、プロジェクトの初期段階、アーキテクチャを検討する段階で使用する。規模に、KSLOC（コード行）または未調整ファンクションポイント（UFP）を使用する。UFPは等価なKSLOCに変換して用いる。規模、規模要因（5個)、コストドライバ（7個）に基づくモデルである。

② Post-Architectureモデル

このモデルは、アーキテクチャ設計以降、すなわちソフトウェアプロダクト仕様化のためのスパイラルフェーズが終了し、ソフトウェアの仕様と開発プロセスが確定した段階以降で使用する。規模、規模要因（5個)、コストドライバ（17個）に基づく詳細なモデルである。

4.3 規模要因とコストドライバ

この節では規模要因とコストドライバについて詳説する。規模要因とコストドライバとも、評価は段階定義がされており、評価結果の精度を高めるようになっている。

① 規模要因（Scale Factor）

5つの規模要因の概要とその評価基準を表１に示す。Early Designと Post-Architectureで共通である。

② コストドライバ（Cost Driver）

Post-Architectureのコストドライバには、プロダクト、

プラットフォーム、人的、プロジェクトの4つの分類がある。コストドライバとその評価基準を表2、表3、表4、表5に示す。 Early Designのコストドライバと評価基準を表6に示す。ここで、Post-Architectureのコストドライバと評価値から、対応するEarly Designのコストドライバにマッピングする方法を説明する。VLを1、Lを2、Nを3、Hを4、VHを5、XHを6点とする。複数個のドライバの合成となるドライバは、複数のドライバの評価値を合計する。そして拡張したEarly Designのコストドライバの評価値のXLからXHへと割り付ける。Early DesignとPost-Architectureのコストドライバは表７のように対応付けがされている。

③ 規模要因とコストドライバの評価値

COCOMO Ⅱ.2000で与えられている規模要因の評価値を表8に、コストドライバの評価値を表9と表10に示す。

4.4 モデル化手法

基本式の係数と規模要因やコストドライバの値を妥当なものに決定することは重要である。そのためにはモデル化の方法が鍵となる。COCOMOでは、モデル式を適用した推定結果の確度を上げ、また、労力をかけて行うデータ収集が効果的となるべく、専門家の判断を取り込む７ステップから構成されるモデル化手法が開発された（図１）。この手法は、専門家によるEモデルと、収集されたプロジェクトデータによるDモデルとをベイズアプ

表1　規模要因と評価基準

略号	要因	概要	非常に低い VL	低い L	中位 N	高い H	非常に高い VH	極めて高い XH
PREC	類似システム分野の経験度	開発組織が開発対象システムと同様のシステムを開発した経験及び知識の量	全く先例無し	ほとんど先例無し	いくらか先例あり	だいたい馴染みあり	大方馴染みあり	完全に馴染みあり
FLEX	開発柔軟性	ソフトウェアに対する要件、外部とのインタフェース仕様等に対する制約からの自由度	厳格に準拠	場合により緩和	いくらか緩和	一般的な準拠	いくらか準拠	一般的な目標
RESL	リスク管理度	リスク管理のためのさまざまな活動の実行頻度	ほとんど行われていない（20%）	いくらか行われている（40%）	しばしば行われている（60%）	だいたい行われている（75%）	ほとんど行われている（90%）	完全に行われている（100%）
TEAM	チーム結束度	利害関係者（ユーザ、開発者、発注者、保守担当者等）間の協力と開発ビジョンの共有の程度	非常に困難な相互作用	いくらか困難な相互作用	基本的に協力的相互作用	大方協力的	非常に協力的	シームレスな相互作用
PMAT	プロセス成熟度	SEIの成熟度モデル（CMM）でのレベル	レベル1下位	レベル1上位	レベル2	レベル3	レベル4	レベル5
			または推定したプロセス成熟度					

表2　Post-Architecture：コストドライバ（プロダクト）と評価基準

略号	ドライバ	概要	非常に低い VL	低い L	中位 N	高い H	非常に高い VH	極めて高い XH
RELY	要求される信頼性	一定期間に渡って意図した機能を実行し続けることに対する要求度合い。失敗が起きたときの影響度合い	若干の不便	簡単に回復可能な損失	回復可能な中くらいの損失	財政上の大きな損失	人命に影響を及ぼす	
DATA	データベースの大きさ	データベースの大きさ。プログラムサイズに対する比率 *D*/*P*=テストに必要なデータベースの規模（バイト）÷プログラム規模（SLOC）		*D*/*P* ＜10	10≦ *D*/*P* ＜100	100≦ *D*/*P* ＜1000	*D*/*P* ≧1000	
CPLX	プロダクトの複雑性	制御、数値計算、機器依存性、データ管理、ユーザインタフェース等への要求の複雑さ	単純	いくらか複雑	適度に複雑	複雑	かなり複雑	極めて複雑
RUSE	要求される再利用性	プロジェクトで開発するソフトウェアがプロジェクト以降に再利用できる（再利用される見込みの）範囲・レベル		再利用を意図しない	プロジェクト内で再利用	複数プロジェクト間で再利用	プロダクトラインで再利用	複数プロダクトライン間で再利用
DOCU	要求される文書化の度合い	ライフサイクル上で要求される文書作成量	要求されない	部分的な要求	適度	過度	非常に過度	

VL：Very Low　L：Low　N：Nominal　H：High　VH：Very High　XH：Extra High

表3　Post-Architecture：コストドライバ（プラットフォーム）と評価基準

略号	ドライバ	概要	非常に低い VL	低い L	中位 N	高い H	非常に高い VH	極めて高い XH
TIME	実行時間の制約	システムが使用可能な実行時間のうち、予想される使用時間 （予想する実使用時間÷使用可能な実行時間）×100			50%以下	70%	85%	95%
STOR	メモリの制約	メモリ全体に対するソフトウェアのメモリ使用量の割合 （使用する主記憶量÷利用可能な主記憶量）×100			50%以下	70%	85%	95%
PVOL	プラットフォームの安定性	開発するソフトウェアに対するプラットフォーム（ハードウェア、OS、データベース等）の変更の頻度		大きな変更：12ヵ月ごと 小さな変更：1ヵ月ごと	大きな変更：6ヵ月ごと 小さな変更：2週間ごと	大きな変更：2ヵ月ごと 小さな変更：1週間ごと	大きな変更：2週間ごと 小さな変更：2日ごと	

表4　Post-Architecture：コストドライバ（人的）と評価基準

略号	ドライバ	概要	非常に低い VL	低い L	中位 N	高い H	非常に高い VH	極めて高い XH
ACAP	分析者の能力	分析者の分析・設計能力、コミュニケーション能力、協調性　（備考）	下位より15%	下位より35%	下位より55%	下位より75%	下位より90%	
PCAP	プログラマの能力	プログラマの能力、コミュニケーション能力、協調性（備考）	下位より15%	下位より35%	下位より55%	下位より75%	下位より90%	
PCON	要員の継続性	プロジェクトの参加者が継続して従事している度合い	48%の要員補充（年間で）	24%（年間で）	12%（年間で）	6%（年間で）	3%（年間で）	
APEX	アプリケーションの経験	開発対象ソフトウェアシステムと類似のアプリケーションの経験年数	2ヵ月以下	6ヵ月	1年	3年	6年	
PLEX	プラットフォームの経験	プラットフォーム（ユーザインタフェース、データベース、ネットワーク等を含む）に関する経験年数	2ヵ月以下	6ヵ月	1年	3年	6年	
LTEX	言語とツールの経験	プログラミング言語とソフトウェアツール（要件分析、設計作図、構成管理、ライブラリ管理、一貫性チェック等を行う）の経験年数	2ヵ月以下	6ヵ月	1年	3年	6年	

（備考）「分析者の能力」（ACAP）、「プログラマの能力」（PCAP）の基準は、分析者（またはプログラマ）が、同職種の一般的多数の人達（社外も含む）を対象として統計的分布で見た場合に、小さいほうから数えて何%の位置という見方をする。これは、「パーセンタイル」という統計的表示法の見方である。

表5　Post-Architecture：コストドライバ（プロジェクト）と評価基準

略号	ドライバ	概要	非常に低い VL	低い L	中位 N	高い H	非常に高い VH	極めて高い XH
TOOL	ツールの活用	利用するソフトウェアツールがサポートする機能の充実度と、複数ツールの統合の度合い	単体ツール（エディット、コーディング、デバッグ）	少し統合された単純なCASEツール	ある程度統合された基本的ライフサイクルツール	統合された強力なライフサイクルツール	よく統合された強力で成熟したライフサイクルツール	
SITE	複数拠点の連絡	開発拠点の地理的な分散の度合い	国際間	複数都市及び複数企業	複数都市または複数企業	同一都市または都市部	同一ビルまたは共同ビル	まったく同じ場所
		開発拠点間のコミュニケーション手段のレベル	電話（共有）、郵便	電話（個人毎）、FAX	電子メール	広帯域電子的コミュニケーション	広帯域電子的コミュニケーション、ビデオ会議	双方向通信のマルチメディア
SCED	スケジュール要求	標準的な開発期間に対する、求められる開発期間の割合	標準の75%	標準の85%	標準の100%	標準の130%	標準の160%	

表6　Early Design：コストドライバと評価基準
※[BOEHM2000]のTable2.36、2.37、2.38、2.39、2.40をまとめたもの

略号	ドライバ	概要	極めて低い XL	非常に低い VL	低い L	中位 N	高い H	非常に高い VH	極めて高い XH
RCPX	プロダクトの信頼性と複雑性	RELY、DATA、CPLX、DOCU評価値の合計	5、6	7、8	9〜11	12	13〜15	16〜18	19〜21
		要求される信頼性や文書化の度合い	僅か	少ない	部分的	基本的	強い	非常に強い	極めて強い
		プロダクトの複雑度	非常に単純	単純	部分的	適度	複雑	非常に複雑	極めて複雑
		データベースの大きさ	小	小	小	中	大	非常に大	非常に大
RUSE	要求される再利用性	Post ArchitectureのRUSEに同じ			再利用を意図しない	プロジェクト内で再利用	複数プロジェクト間で	プロダクトラインで	複数プロダクトライン間で
PDIF	プラットフォームの難易度	TIME、STOR、PVOL評価値の合計			8	9	10〜12	13〜15	16、17
		実行時間とメモリの制約			50%以下	50%以下	65%	80%	90%
		プラットフォームの安定性			非常に安定	安定	いくらか不安定	不安定	非常に不安定
PERS	要員の能力	ACAP、PCAP、PCON評価値の合計	3、4	5、6	7、8	9	10、11	12、13	14、15
		ACAPとPCAP（下位より）	20%	35%	45%	55%	65%	75%	85%
		プロジェクトの参加者が、継続して従事している度合い（年間で交代する比率で表す）	45%	30%	20%	12%	9%	6%	4%
PREX	要員の経験	APEX、PLEX、LTEX評価値の合計	3、4	5、6	7、8	9	10、11	12、13	14、15
		アプリケーション、プラットフォーム、言語及びツールの経験	3ヶ月以下	5ヶ月	9ヶ月	1年	2年	4年	6年
FCIL	ファシリティ	TOOL、SITE評価値の合計	2	3	4、5	6	7、8	9、10	11
		ツールの活用（表5のTOOLを参照）	最小	いくらか	単純	基本的	やや統合	よく統合	強力
		複数拠点の連絡（表5のSITEを参照）	弱い	やや弱い	いくらか	基本的	やや強い	強い	大変強い
SCED	スケジュール要求	Post ArchitectureのSCEDに同じ	標準の75%	標準の85%	標準の100%	標準の130%	標準の160%		

XL：Extra Low　VL：Very Low　L：Low　N：Nominal　H：High　VH：Very High　XH：Extra High

表7　Early DesignとPost-Architectureのコストドライバの対応
※[BOEHM2000]のTable2.35から引用

Early Designのコストドライバ	対応するPost Architectureのコストドライバの組合せ
RCPX	RELY、DATA、CPLX、DOCU
RUSE	RUSE
PDIF	TIME、STOR、PVOL
PERS	ACAP、PCAP、PCON
PREX	APEX、PLEX、LTEX
FCIL	TOOL、SITE
SCED	SCED

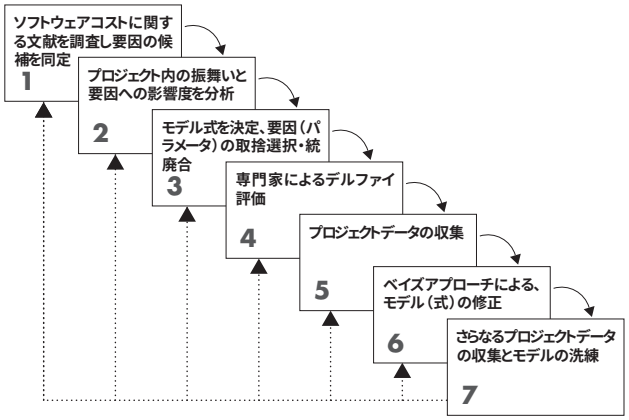


図1　COCOMO Ⅱ モデル化手法

表8　規模要因の値 （COCOMO II.2000）

	規模要因	非常に低い VL	低い L	中位 N	高い H	非常に高い VH	極めて高い XH
PREC	類似システム分野の経験度	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	開発柔軟性	5.07	4.05	3.04	2.03	1.01	0.00
RESL	リスク管理度	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	チーム結束度	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	プロセス成熟度	7.80	6.24	4.68	3.12	1.56	0.00

表9　Post-Architecture：コストドライバの値 （COCOMO II.2000）

	コストドライバ	非常に低い VL	低い L	中位 N	高い H	非常に高い VH	極めて高い XH
RELY	要求される信頼性	0.82	0.92	1.00	1.10	1.26	
DATA	データベースの大きさ		0.90	1.00	1.14	1.28	
CPLX	プロダクトの複雑性	0.73	0.87	1.00	1.17	1.34	1.74
RUSE	要求される再利用性		0.95	1.00	1.07	1.15	1.24
DOCU	要求される文書化の度合い	0.81	0.91	1.00	1.11	1.23	
TIME	実行時間の制約			1.00	1.11	1.29	1.63
STOR	メモリの制約			1.00	1.05	1.17	1.46
PVOL	プラットフォームの安定性		0.87	1.00	1.15	1.30	
ACAP	分析者の能力	1.42	1.19	1.00	0.85	0.71	
PCAP	プログラマの能力	1.34	1.15	1.00	0.88	0.76	
PCON	要員の継続性	1.29	1.12	1.00	0.90	0.81	
APEX	アプリケーションの経験	1.22	1.10	1.00	0.88	0.81	
PLEX	プラットフォームの経験	1.19	1.09	1.00	0.91	0.85	
LTEX	言語とツールの経験	1.20	1.09	1.00	0.91	0.84	
TOOL	ツールの活用	1.17	1.09	1.00	0.90	0.78	
SITE	複数拠点の連絡	1.22	1.09	1.00	0.93	0.86	0.80
SCED	スケジュール要求	1.43	1.14	1.00	1.00	1.00	

表10　Early Design：コストドライバの値 （COCOMO II.2000）

	コストドライバ	極めて低い XL	非常に低い VL	低い L	中位 N	高い H	非常に高い VH	極めて高い XH
RCPX	プロダクトの信頼性と複雑性	0.49	0.60	0.83	1.00	1.33	1.91	2.72
RUSE	要求される再利用性			0.95	1.00	1.07	1.15	1.24
PDIF	プラットフォームの難易度			0.87	1.00	1.29	1.81	2.61
PERS	要員の能力	2.12	1.62	1.26	1.00	0.83	0.63	0.50
PREX	要員の経験	1.59	1.33	1.12	1.00	0.87	0.74	0.62
FCIL	ファシリティ	1.43	1.30	1.10	1.00	0.87	0.73	0.62
SCED	スケジュール要求		1.43	1.14	1.00	1.00	1.00	

表11　RELY 保守：コストドライバの値 （COCOMO II.2000）

	規模要因	非常に低い VL	低い L	中位 N	高い H	非常に高い VH	極めて高い XH
RELY	要求される信頼性	1.23	1.10	1.00	0.99	1.07	

ローチ[GELMAN1995]によって統合して、B（Balanced）モデルを構築する点が特徴といえる。このモデル化手法は、COCOMO Ⅱから適用されている。

COCOMO Ⅱのデータ収集アプローチ

COCOMO Ⅱのモデルは、前述のモデル化手法に従い信頼性の高いものが構築されている。USCで収集された2,000以上の候補プロジェクトのデータから確認された161件のプロジェクトのデータから精査され作成されている。[BOEHM2000]の4章に記されているが、これら161プロジェクトすべてに対して、フォローアップインタビューとサイト訪問を実施してプロジェクト関係者にデータの正確性と定義を確認している。このようにして得られた信頼性の高い収集データがCOCOMOのモデル式と係数値の妥当性の論拠を与えている。

5. COCOMOの拡張

5.1 維持的保守と再利用の場面

① 保守

「ソフトウェア保守」とは、[BOEHM1981]によれば、既存ソフトウェアについて主機能を変更せずに既存ソフトウェアを修正するプロセスである。保守工数を見積もる式は、COCOMO Ⅱ のPost-Architectureモデルと同じである。

$$PM_M = A \times (Size_M)^E \times \prod_{i=1}^{15} EM_i$$

$Size_M$ ＝ベースコード規模×MCF×MAF
＝（追加規模＋修正規模）×MAF

MCF ＝（追加規模＋修正規模）÷ベースコード規模

$$MAF = 1 + \left(\frac{SU}{100} \times UNFM\right)$$

規模は保守規模$Size_M$を使用する（単位：KSLOC）。ベースコード規模は保守対象の既存ソフトウェアの規模である。

Eは、変更した規模（追加と修正の部分。削除部分は対象外）について適用する。

コストドライバでは、SCEDとRUSEは使用しない。RELYは異なる評価値を持つ。表11にRELYの評価値を示す。SUと$UNFM$は第4節と同じものを使用する。

② 再利用

再利用モデルは、既存ソフトウェアをそのまま、または修正して対象システムに結合する場合に利用する。モデルは第4節に示した基本式でカバーされている。

5.2 開発プロセスへの考慮

COCOMO Ⅱの拡張モデルには、次のものがある。

① アプリケーション組み立てモデル

アプリケーションの既存部品再利用に関する工数見積りと調整を可能にしたモデル。既存部品の再利用とその修正利用工数に関する「オブジェクトポイント」という概念を導入した点が特徴的である。

② COCOTS

COTS（Commercial-Off-The-Shelf, 市販部品）インテグレーションのコストモデル。COCOMO Ⅱでは工数や期間見積りの際、COTSをブラックボックス部品（ソースコード改変を伴わない部品再利用）として扱う前提で言及していたが、ソースコード改変・再設計を伴う形での見積り手法としてCOCOMO Ⅱとは区別し、独立したコストモデルとして定義している。

③ COPSEMO

開発期間と工数のフェーズごとの分布。COCOMO当時のWaterfallモデル以外で、RUP等の最近のライフサイクルモデルに対応した修正変更の調整方法である。

④ CORADMO

ラピッド・アプリケーション開発（RAD）の工数と開発期間の調整。新要因として、(a) 再利用＋高レベル言語；(b) 開発プロセスのリエンジニアリング度合い；(c) チーム協業体制の充足度 等々を追加している。

⑤ COQUALMO

欠陥密度と結果としての品質予測。Capers Jonesのソフトウェア欠陥混入・除去モデル（1975年）の概念に似た品質予測モデル。信頼性要求の考慮等Post-Architectureモデルを中心とした因子を列挙した上で、欠陥混入モデルや欠陥除去モデル等を紹介している。

⑥ COPROMO

ソフトウェア生産性向上のための戦略適用効果の見積

り。適用技術やプロセス改善の投資効果を計算する戦略計画ツールとして位置付けられている。とくに業務観点からの変動要素がプロジェクトに与える影響を考慮している点等、具体例を含めて解説している。

⑦COSYSMO

システムエンジニアリング・コストモデル。ソフトウェアとハードウェアとを含めたシステム全体を対象としたコストモデルである。

それぞれについて詳しくは[BOEHM2000]の第5章、最新情報はUSC主催のCOCOMO Forum [USC URL]等で発表されている。

6. まとめ

本解説で紹介したCOCOMOは、技術の進歩に合わせてモデルも進化している。COCOMOが優れている点は、南カリフォルニア大学を中心とした一大研究コミュニティを形成し、組織や世代を超えて技術伝承が図られていることである。

多くのコスト分析手法がブラックボックス的で詳細が見えないのに対し、COCOMOは、数式も予測方法も各種調整式の解釈もすべて公開されている。これは、システムに関わるすべてのステークホルダの共有情報として活用できることを意味しており、ビジネスの場面でも透明性、公平性を確保する上でも価値のあるものである。

ソフトウェアエンジニアリング分野で、長期に亘り各方面にインパクトを与え続けている方法は少ない。COCOMOはソフトウェア開発が、人間の記述活動であるという本質を捉え、これを規模と工数との関係として定式化している。今後も、新しいソリューションや産業構造に移行したとしても、基底になるものであり続けることは間違いないであろう。

参考文献

[BOEHM1981] Barry W. Boehm : Software Engineering Economics, Prentice Hall, 1981

[BOEHM2000] Barry W. Boehm and et al : Software Cost Estimation with Cocomo II, Prentice Hall, 2000

[USC URL] 南カリフォルニア大学 Center for Systems and Software Engineering ホームページ, http://sunset.usc.edu/csse/index.html

[PARK1992] R. Park : Software Size Measurement: A Framework for Counting Source Statement, CMU/SEI-92-TR-20, Software Engineering Institute, Pittsburgh, PA, 1992

[GELMAN1995] A. Gelman, J. Garlin, H. Stern, and D. Rubin : Bayesian Data Analysis, Chapman Hall, 1995

附表A. COCOMO 81とCOCOMO IIの比較
※[BOEHM2000]のTable 4.1を基に改良

	COCOMO 81	COCOMO II
モデル構造	ソフトウェアに割り振られた要求事項から開始する単一のモデル	要求事項を確定してアーキテクチャを固め、リスクを削減する、スパイラル型開発を通して進めることを想定している、3種類のモデル
工数の式	工数=$A\times\Pi(ci)\times$（規模）^ 指数	工数=$A\times\Pi(ci)\times$（規模）^ 指数
指数（Exponent）	モード（次の3種類）の関数による定数 Organic = 1.05 , Semidetached = 1.12 Embedded = 1.20	5つの規模要因の評定に基づいて定まる変数 PREC, FLEX, RESL, TEAM, PMAT
規模（*Size*）	ソースコード行 SLOC、配布ソース命令数 DSI（Deliverable Source Instructions）	アプリケーションポイント、ファンクションポイント、またはソースコード行 SLOC（注:DSIとは異なる）
コストドライバ（*ci*） 斜体のドライバは、モデル間で異なる部分	15個のドライバ	17個のドライバ （注1）
	RELY, DATA, CPLX	RELY, DATA , CPLX, *RUSE, DOCU*
	TIME, STOR, *VIRT*（仮想計算機の変動） , , *TURN*（ターンラウンド時間）	TIME, STOR, *PVOL*
	ACAP, PCAP, AEXP, *VEXP*（仮想計算機の経験） *LEXP*（言語の経験）	ACAP, PCAP, *PCON*, APEX（COCOMO 81のAEXPを改名）, *PLEX, LTEX*
	TOOL, *MODP*（モダンプログラム技法の利用）	TOOL, *SITE, SCED*
他の差異	モデルは以下に基づく -線型の再利用算定式 -比較的安定した要求事項が前提	以下のものなど多くの拡張がある -非線形の再利用算定式 -理解と適応に必要な工数を調査する再利用モデル -要求事項の不安定性を扱うのに用いる評定

（注1）COCOMO 81と同じドライバでも定義は最新のものに見直されたものがある

BOOK REVIEW

初めて学ぶソフトウエアメトリクス プロジェクト見積もりのためのデータの導き方

ローレンス・H・パトナム、ウエア・マイヤーズ　著　山浦 恒央　訳

ISBN：978-4-82228-242-4　日経BP社刊
A5判・360頁・定価2,940円（税込）・2005年10月刊
原書名：Lawrence H.Putnam and Ware Myers: Five Core Metrics. The Intelligence Behind Successful Software Management, 2003

読者にプロジェクト計測への行動開始を促す課題提起の書

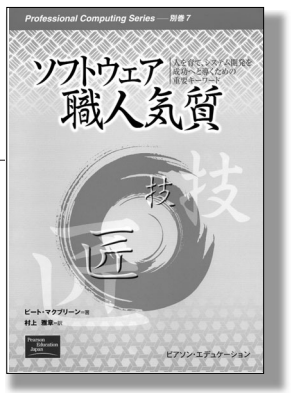
ソフトウェアプロジェクトの開発管理を成功させるための5つの核となる計測項目について解説している。和訳タイトルが示すような入門書ではなく、また、見積りは話題の1つではあるが、記述の中心は計測を基盤とした進行中のプロジェクト管理の進め方、つまりインプロセス計測に真正面から取り組んでいる。5つの核とは、機能量、生産性、時間、工数、信頼性であるが、これらには相互関連があり独立変数でないところがややこしい。著者の永年の米国陸軍および独立コンサルタントとしての活動の中で得た数1,000プロジェクトのデータ蓄積を背景に、蓄積データから逆算して得た関係式と呼ぶ数式が多数提示されている。またプロジェクトの進捗状況を示す横軸を時間軸とする夥しいグラフが示され、これらの解説と合わせて、計測し分析したデータの活用法をソフトウェア開発の14の局面に合わせて具体的に示している。示された考え方は支持できるが、関係式と提言の定量的裏づけは丸呑みにできず、いわゆる追試が必要である。そこに本領域の本質的な課題がある。読むだけでなく、追試によって自己に必要な定量データを得てはじめて本書の示す知恵が活きてくる。そういう意味で、読者の行動を促す課題提起の良書といえる。

（神谷芳樹）

ソフトウェア職人気質―人を育て、システム開発を成功へと導くための重要キーワード

ピート・マクブリーン　著　村上 雅章　訳

ISBN：978-4-89471-441-0　ピアソン・エデュケーション刊
A5判・208頁・定価2,415円（税込）・2002年3月刊



ソフトウェア工学に通じる職人気質

本書で述べられているソフトウェア職人気質とは、原文では"Software Craftsmanship"と表現されているが、思考だけでなく具体的な方法論である。SECが普及推進するソフトウェア工学に対するカウンターカルチャ的な方法論であるが、実践的で有効な人材育成を実現する思考や方法論といえる。

本書では、ソフトウェア工学は、ミッションクリティカルで大規模な開発向けには有効であると説く。そして、多くの開発チームは職人気質に基づく方法論の適用が有効であると説く。教育訓練実績や資格等に基づく育成やチーム編成でなく、過去の開発実績に対する評判や推薦に基づくチーム編成や、弟子入りという育成システムを提案している。ソフトウェア開発という知識でなくスキルが重要な世界において、非常に説得力のある方法論である。

しかし、ソフトウェア工学によって解決できる問題も多いことを忘れてはいけない。モデリング、プロセス、定量化等の方法論は、中小規模のソフトウェア開発でも有効に機能する。

これからは、ソフトウェア工学と職人気質の融合が必要である。これまでは職人と工学の融合を議論する際、両方とも概念的なイメージで議論していた。これからは、SWEBOK等で定義されるソフトウェア工学と、本書の職人気質を対象に具体的融合を検討すればいい。すでにワークプレイスラーニングという教育方法論は、この融合を具体化しているともいえる。

（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
1月	18（金）	Open SESSAME Workshop 2008	組込みソフトウェア管理者・技術者研究会（SESSAME）	東京都中央区・東実年金会館4階会議室	http://www.sessame.jp/
	23（水）	【SEC主催セミナー】要求シンポジウム	IPA/SEC　NTTデータ　共催	東京都千代田区・霞ヶ関プラザホール	http://sec.ipa.go.jp/
	25（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/
	28（月）	【SEC主催セミナー】ITプロジェクトの『見える化』（札幌）	IPA/SEC　北海道情報システム産業協会（HISA）　共催	北海道札幌市・ホテルオークラ札幌	http://sec.ipa.go.jp/
	29（火）	Embedded Forum　in Greater Nagoya	東海ものづくり創生協議会、社団法人組込みシステム技術協会、社団法人愛知県情報サービス産業協会	愛知県名古屋市・ミッドランドスクエア	http://www.chubu.meti.go.jp/jyoho/download/20071210_gni_forum.pdf
	29（火）	ソフトウェア ジャパン2008	社団法人 情報処理学会	東京都千代田区・東京ステーションコンファレンス	http://www.ipsj.or.jp/
	30（水）～31（木）	ソフトウェアテストシンポジウム2008東京	特定非営利活動法人 ソフトウェアテスト技術振興協会（ASTER）	東京都目黒区・目黒雅叙園	http://www.jasst.jp/
2月	7（木）	【SEC主催セミナー】ITプロジェクトの『見える化』（秋田）	IPA/SEC	秋田県秋田市・ルポールみずほ	http://sec.ipa.go.jp/
	8（金）	【SEC主催セミナー】ITプロジェクトの『見える化』（広島）	IPA/SEC	広島県広島市・日立中国ソリューションズ会議室	http://sec.ipa.go.jp/
	13（水）～14（木）	Developers Summit 2008（デブサミ2008）	翔泳社	東京都目黒区・目黒雅叙園	http://codezine.jp/devsumi/2008/gaiyo/
	13（水）	【SEC主催セミナー】共通フレーム2007（福岡）	IPA/SEC	福岡県福岡市・富士通九州R&Dセンター	http://sec.ipa.go.jp/
	13（水）	【SEC主催セミナー】プロセス改善（福岡）	IPA/SEC	福岡県福岡市・富士通九州R&Dセンター	http://sec.ipa.go.jp/
	14（木）	【SEC主催セミナー】定量データ分析（福岡）	IPA/SEC	福岡県福岡市・富士通九州R&Dセンター	http://sec.ipa.go.jp/
	14（木）	【SEC主催セミナー】ITプロジェクトの『見える化』（福岡）	IPA/SEC	福岡県福岡市・富士通九州R&Dセンター	http://sec.ipa.go.jp/
	14（木）	【SEC主催セミナー】共通フレーム2007（大阪）	IPA/SEC	大阪府大阪市・大阪科学技術センタービル OSTEC	http://sec.ipa.go.jp/
	14（木）	【SEC主催セミナー】プロセス改善（大阪）	IPA/SEC	大阪府大阪市・大阪科学技術センタービル OSTEC	http://sec.ipa.go.jp/
	15（金）	【SEC主催セミナー】共通フレーム2007（名古屋）	IPA/SEC	愛知県名古屋市・名古屋工業技術研究所	http://sec.ipa.go.jp/
	15（金）	【SEC主催セミナー】プロセス改善（名古屋）	IPA/SEC	愛知県名古屋市・名古屋工業技術研究所	http://sec.ipa.go.jp/
	15（金）	【SEC主催セミナー】ESMR	IPA/SEC	東京都文京区・文京グリーンコート	http://sec.ipa.go.jp/
	22日（金）	【SEC主催セミナー】ITプロジェクトの『見える化』（大阪）	IPA/SEC　財団法人 関西情報・産業活性化センター（KIIS）共催	大阪府大阪市・日立製作所関西支社	http://sec.ipa.go.jp/
	26日（火）	【SEC主催セミナー】ITプロジェクトの『見える化』（名古屋）	IPA/SEC	愛知県名古屋市・名古屋国際会議場	http://sec.ipa.go.jp/
	27日（水）	第1回　大阪大学・京都大学・神戸大学連携シンポジウム	第1回三大学連携シンポジウム実行委員会	大阪府大阪市・大阪国際会議場	http://www.3univ.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

2007年10月30日のSECコンファレンスにて、最優秀論文賞は「組込みソフトウェア設計検証へのモデル検査技術の適用と考察」（発表者：岸知二氏（北陸先端科学技術大学院大学））に決まり本号に掲載しました。

「SEC journal」に掲載される論文は数が少ないですが、高い評価を得ています。しかしながら、査読委員の方からは「企業の方は論文執筆に慣れていないのでは」というご指摘もあり、論文の書き方について掲載する予定でいます。今後、より多くの方から「SEC journal」に論文の投稿をいただくことを期待しています。

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。

（奥）

SEC journal® 第3巻第4号（通巻12号） 2008年1月15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2007年10月30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

・14号（2008年4月発行）
　応募締切　2008年2月22日
　投稿締切　2008年2月29日
　採否通知　2008年3月21日

・15号（2008年7月発行）
　応募締切　2008年5月23日
　投稿締切　2008年5月30日
　採否通知　2008年6月20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　No.11

SEC journal No.12
第3巻第4号（通巻12号）
2008年1月15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