2012年12月14日発行
第8巻第4号（通巻33号）
ISSN 1349-8622

SEC journal 31

巻頭言

森下 俊三　一般財団法人関西情報センター　会長
組込みシステム産業振興機構　副理事長

所長対談

浜口 友一　一般社団法人情報サービス産業協会（JISA）会長

岐路に立つ日本のソフトウェア開発、その方向性について考える

論文

事例紹介：OEMソフトウェア製品の検証プロセス

名倉 正剛　株式会社日立製作所／田村 清朗　株式会社日立製作所／川口 真司　奈良先端科学技術大学院大学
高田 眞吾　慶應義塾大学／飯田 元　奈良先端科学技術大学院大学

ソフトウェアプロジェクトデータにおける量的変数の予実差分析

古山 恒夫　東海大学理学部　教授

非ウォーターフォール型（アジャイル）開発の動向と課題

ソフトウェアの品質説明力強化に関する実験を実施

国際連携

CEA-LISTとの国際連携活動について

トピックス

ラーセン教授を迎えて

テストの技術力強化に向けたテスト技術のスキル標準

渡辺 登　株式会社アフレル ／ 小林 直子　アヴァシス株式会社

組織紹介

一般財団法人関西情報センターの紹介

森下 俊三　一般財団法人関西情報センター　会長

Column

周回遅れの電子政府

IPA 独立行政法人情報処理推進機構
Information-technology Promotion Agency, Japan

SEC journal No.31
Vol.8 No.4 Dec. 2012

巻頭言
145 **組込みシステム産業の一層の活性化に向けて**
森下 俊三　一般財団法人関西情報センター　会長
組込みシステム産業振興機構　副理事長

所長対談：浜口 友一　一般社団法人情報サービス産業協会（JISA）会長
146 **岐路に立つ日本のソフトウェア開発、その方向性について考える**

論文
150 **事例紹介：OEMソフトウェア製品の検証プロセス**
名倉 正剛　株式会社日立製作所
田村 清朗　株式会社日立製作所
川口 真司　奈良先端科学技術大学院大学　有人宇宙システム株式会社
高田 眞吾　慶應義塾大学
飯田 元　奈良先端科学技術大学院大学

SECエンタプライズ系プロジェクト解説
158 **ソフトウェアプロジェクトデータにおける量的変数の予実差分析**
古山 恒夫　東海大学理学部　教授
IPA/SEC専門委員
164 **非ウォーターフォール型（アジャイル）開発の動向と課題**
—IPA/SECにおける4年間の調査・検討から明らかになったこと—
山下 博之・柏木 雅之

SEC統合系プロジェクト解説
173 **ソフトウェアの品質説明力強化に関する実験を実施**
中谷 浩康・室 修治

国際連携
178 **CEA-LISTとの国際連携活動について**
—CEA-LISTとその研究成果を展開するパートナー企業との意見交換を実施—
杉浦 秀明・八嶋 俊介

トピックス
181 **ラーセン教授を迎えて**
—【SEC特別セミナー】形式手法の98導入事例の調査・分析から見る高信頼性ソフトウェア開発の現状（2012年10月23日開催）より—
新谷 勝利
184 **テストの技術力強化に向けたテスト技術のスキル標準**
—「SSFに基づくテスト技術スキルフレームワーク（Test.SSF）」策定の取り組み—
渡辺 登　株式会社アフレル　元IPA/SEC研究員
小林 直子　アヴァシス株式会社　元IPA/SEC研究員

組織紹介
188 **一般財団法人関西情報センターの紹介**
森下 俊三　一般財団法人関西情報センター　会長

Column
190 **周回遅れの電子政府**
松田 晃一　IPA顧問

191 BOOK REVIEW
192 編集後記
ITパスポート試験のご案内／ SEC journal論文募集／バックナンバーのご案内

巻頭言

組込みシステム産業の一層の活性化に向けて

一般財団法人関西情報センター　会長
組込みシステム産業振興機構　副理事長
森下　俊三

IOT（Internet of things）やM2M（Machine to Machine）という言葉に見られるように、スマートフォンやテレビはもちろん、冷蔵庫や掃除機といった白物家電から自動車や住宅まで、私たちの周りのあらゆる機器に制御システムが組み込まれ、インターネットに接続されて、様々なサービスと連携するIT融合型システムの構築が始まっています。

一方で、これからのビジネス展開においては、従来日本が得意とした高品質なものづくりだけではなく、融合新産業の構築を意識した新たな製品・サービス開発が重要となってきております。産業構造審議会が取り上げている具体的な重点分野としては、エネルギーとICTの連携を目指すスマートコミュニティ産業、医療のトータルソリューションや健康・介護分野との連携を目指すスマートヘルスケア産業、日常生活と社会インフラの連携が不可欠なロボットビジネス、情報端末化する自動車と交通システム産業、自然環境や市場との最適制御が求められるスマートアグリシステム産業、電子化されたコンテンツの流通を前提とするコンテンツ・クリエイティブビジネスなどがあります。

こうしたビジネス・産業の展開において、モノとサービスの連携に重要な役割を果たすのが「組込みシステム」です。高度なサービス連携を実現するためには、「リアルタイム性」、「高信頼性・安全性」、「コンパクト化」、「高品質化」が一層求められており、これらの要求に応えていくためには、我が国の組込みシステム産業が抱える諸課題の解決を図らなければなりません。具体的には「ソフトウェアの品質低下」、「高度マネージャの不足」、「ソフトウェア開発標準がないことによる効率性の低さ」、「機器製造業からのソフト、ハード一体開発への要望に応えられない」、「大企業の系列化傾向が強く、新たな取引などが行いにくい」などの課題であります。

そこで、関西経済連合会が中心となり2010年6月に産学官が連携して、人材育成、経営力強化、技術力向上、ビジネスマッチングのプラットフォームとして「組込みシステム産業振興機構」を設立し、「教育事業」、「開発支援事業」、「企画広報事業」を3本柱として組込みシステム産業の活性化に取り組んでおります。

SECにおいても組込みソフトウェア開発力の強化手段として、高品質な組込みソフトウェア開発の実現に向けた具体的な技術整備の研究や組込みソフトウェア開発者向けスキル標準の策定が行われていますが、こうした活動の成果としてのガイドラインや教育カリキュラムには大きな期待を寄せています。

また、全国各地域においても組込みシステム産業の競争力強化のために協議会やフォーラムの活動が活発化しています。一般財団法人関西情報センターでも「組込み産業活性化フォーラム in KANSAI」を開催しているほか、「組込みパワフル企業集 in 関西」という企業データベースの構築を行うなど、組込みシステム産業の活性化に取り組んでおります。

SECはもちろん日本全体の有機的連合体によって組込みシステム産業が一層活性化することを望みます。

岐路に立つ日本のソフトウェア開発、その方向性について考える

一般社団法人情報サービス産業協会（JISA）会長
浜口　友一 × SEC 所長 松本　隆明

クラウドコンピューティング時代に入り、IT活用のパラダイムは所有から利用へとシフトしている。大きな変化を迎える中、ソフトウェア開発事業者が目指すべき方向、顧客企業のあるべき姿、今後のソフトウェア・エンジニアリングの方向性について、一般社団法人情報サービス産業協会会長の浜口友一氏に伺った。

松本：IT 活用のパラダイムは、所有から利用へとシフトしています。ソフトウェアも、つくる時代から使う時代へとシフトしています。それに対して、ソフトウェア開発の手法はウォーターフォール型が主流であまり変わっていません。情報サービス産業としてパラダイムシフトにどう対応すべきかが急務になっていると考えています。本日は、ソフトウェア開発事業者が目指すべき方向、顧客企業のあるべき姿、今後のソフトウェア・エンジニアリングの方向性について考えたいと思っています。情報サービス産業協会（JISA）は、IT をめぐるパラダイムシフトを受けて5つの構造改革を進めていくべきだと提言されていますね。初めに提言の内容についてお話しいただきたいと思います。

浜口　友一（はまぐち　ゆういち）
1967年京都大学工学部電気工学科卒業。同年日本電信電話公社（現NTT）に入社し、システムエンジニア、プロジェクトマネージャとして、バンキングシステム等の開発に関わる。1988年NTTデータ通信（現NTTデータ）に移り、1995年取締役第一産業システム事業部長に就任。以降経営企画部長、公共システム事業本部長、代表取締役副社長を経て、2003年代表取締役社長、2007年相談役。2007年より一般社団法人情報サービス産業協会会長を務め、政府のIT政策関連の委員会にも数多く参加。

浜口：JISA は、今後、情報サービス産業が取り組むべきテーマとして、「受託開発型からサービス提供型」、「労働集約型から知識集約型」、「多重下請構造から水平分業」、「顧客従属型からパートナー型」、「国内競争から国際競争」という5つの構造改革を掲げています。受託開発型からサービス提供型へというテーマは、我々のビジネスをシフトしていこうということです。代表的な例としてクラウドサービスがあります。受託開発は顧客からスペックをいただいてシステムをつくることを仕事としてきました。これがサービス提供型の仕事になると、私どもベンダーが自分でスペックを考える能力が必要となります。スペックを考えたあとにソフトウェアを開発するわけですが、その部分は従来から行ってきた SI 事業と基本的には似ているものです。ただし求められる能力は、従来の受託開発型のものとは異なります。例えばサービス提供型のビジネスでは、サービスをリリースしてからもお客様の反応によりサービスの内容を変えていくことが求められます。また、開発に1年も2年もかけるわけにはいきません。様々な状況の変化に迅速に対応することが求められます。ソフトウェア開発においても、ウォーターフォール型にプラスして反復型あるいはアジャイルで開発する能力が求められると思います。

松本：受託開発はお客様のご指示通りにつくることを一番大切なことと考えてきましたが、サービス提供型では経営に近い上流の部分まで含めて考えることが、開発企業に求められますね。

浜口：サービス提供型にシフトするために大切なことは、お客様の身になってビジネスモデルをつくることです。ただし、お客様ごとにシステムを一からつくるのではなく、ある程度システムを共通化する必要があります。実際には、お客様ごとにニーズは異なるので、例えば、3割のお客様が利用出来る共通部分を用意しておき、残りの部分についてはお客様ご

とに開発する、といった対応が求められます。

松本：共通的な部分はクラウドに持っていくという流れも出てくるでしょう。受託開発型からサービス提供型へという流れの中で、プログラムのつくりかたも変わりますね。

浜口：そうです。従来のソフトウェア開発は、ビジネスモデルを考える人、それをコンピュータに載せられるようなスペックに落とす人、そのスペックに合わせてプログラムを書くコーダー、という役割分担がある形で進んできました。スペックに関しては間違ったシステムを開発しないよう、厳密なスペックを作成してきました。これがサービス提供型になると、ある程度、スペックが固まったらすぐにソースコードをつくるという方法が必要になるのではないかと思います。

松本：それは産業構造の変化にもかかわることだと思います。従来、情報サービス産業界は階層構造化されていて、大手企業が仕事を下請に出すという産業構造の仕組みがあります。今後、それが、一体化した形の産業構造になる可能性があるということだと思います。

浜口：請負型からサービス提供型へという流れの中でも、請負で開発するという仕事は依然として残ると思います。しかしその請負型もお客様のご指示通りにつくるだけではなく、ベンダーから提案をするようにしなければお客様からは評価されなくなるでしょう。ベンダーはお客様の経営課題に対して提案を行い、お客様の側も経営課題の解決に関する領域の仕事をベンダーに依頼するというような、パートナー関係を目指そうということです。それが先ほどご紹介した構造改革の中の「顧客従属型からパートナー型」というテーマです。このように、JISA が目指している構造改革の５つのテーマはつながっているのです。

上流に関する能力の重要性が高まる

松本：知識集約型へのシフトに関連して、ソフトウェア開発における上流工程の重要性が指摘されています。SEC でも、超上流の要件定義フェーズにきっちり取り組みましょうといってきたのですが、上流に関する能力の重要性が更に高まり、今後ソフトウェア開発企業にはエンジニア個人も上流から下流まで一体で提案出来ることが求められていくと思います。ソフトウェアを開発する能力に加えて、顧客の業務を理解するスキルが必要になります。

浜口：そこが非常に重要な点です。本当にそういう人材を育成出来るのかという議論もあります。現実には上流から下流までひと通りの知識・スキルを学んでいくと、エンジニアによって強いところ、弱いところが出てくると思います。ただ、エンジニアの能力の基本は、スペックを自分の中で構造化してコーディングまで持っていくことです。その基礎能力に、業務改善を提案する能力がプラスされると非常にいいでしょう。

松本：アジャイル開発になると、チームで開発するので専門のコーダーがいないため、コーディングが出来る人間が上流も含めて考えなければいけません。米国 IT 企業に見られるスモールチームとはどのようなものですか。

浜口：グーグルやマイクロソフトはスモールチーム型でソフトウェアを開発しています。以前、グーグルを訪問してエンジニアの採用方法について聞いたことがあります。彼らは５～６人のスモールチームで開発を行っていて、エンジニアを新しく採用する場合、面接して最後に、採用を決めるのはスモールチームのメンバーなんですね。５人のチームだとすると、５人全員が「この人ならいい」と言わないとチームに入れないと話していました。面接すれば、能力が分かり、自分のチームに入って一緒に出来るかどうかが分かるというのです。サービス提供型でソフトウェアをつくっていくとなると、最後はそういうところへいくのかなと思いますね。日本でも、ゲームソフト開発（の現場）はそうなっているようです。

松本：ゲームソフト会社の方は、どれだけコーディング出来るかがエンジニアの本当の能力だといいます。どれだけ品質がよくてきれいなソフトウェアが書けるかが（まず重要な）能力であり、設計やマネジメントの能力はあとで身に付ければいいと考えています。

浜口：加えて、エンジニアにとって大切な能力は発想する能力です。実現出来ないことを言われるのは困る。

松本：そうですね。まったくソフトウェアが書けないようなものを発想されても困りますからね。ゲームソフトをつくるなら、エンジニアの頭の中に例えば「スマホレベルで性能が出るか。

松本　隆明（まつもと　たかあき）
1978年東京工業大学大学院修士課程修了。同年日本電信電話公社（現NTT）に入社、オペレーティング・システムの研究開発、大規模公共システムへの導入SE、キャリア共通調達仕様の開発・標準化、情報セキュリティ技術の研究開発に従事。2002年に株式会社NTTデータに移り、2003年より技術開発本部本部長。2007年NTTデータ先端技術株式会社常務取締役。2012年7月より技術本部ソフトウェア・エンジニアリング・センター（SEC）所長。博士（工学）。

実現出来るか」ということがベースにあり、そのベース上で企画・提案が出来ないといけないでしょう。

浜口：ゲームソフト会社はそういう人には相当な報酬を払うわけでしょう？

松本：そうですね。

浜口：我々の業界では、それほど報酬の差を付けることは難しい現状があります。しかし一方、情報サービス産業のソフトウェア会社はそれぞれの会社なり、その中にあるプロジェクトチームなりが固有のスキルを持つようにしなければいけないと思います。例えば、あるアプリケーションに強みを持っているとか、データベースを扱うのに優れているといったことです。またそういうスキルを個々のエンジニアが持つように、会社は仕掛けをつくることが求められます。

松本：成果の評価方法も変えることが求められます。「人月」で評価すると、きれいなプログラミングでコンパクトにつくると報酬が小さくなるということがあります。

浜口：その通りです。

松本：契約の仕方も考えていかないといけません。上流も含めて、IT 化によって得られた成果をもとにして報酬をいただく形にしていかないと。とくにアジャイルになると、プログラムを何度もつくり直すので成果で評価することが求められます。

浜口：アジャイル開発はこれからどんどん増えていくと思います。自分たち自身で構想を練りスペックを決めたソフトウェアを開発する場合にアジャイルは向いています。でも、請負型 SI にアジャイル開発を使うことも有効ではないかと思います。ただその場合は、お客様のほうも対応出来る体制をつくる必要があります。なぜならアジャイル開発は、プログラムをつくってみて評価して直すということを繰り返していくので、お客様と開発チームが一体的に動かないと成功しません。

松本：IPA では、「超上流から攻める IT 化の原理原則 17 ヶ条」というパンフレットを作成しています。その中で、お客様と信頼関係をつくって一緒に目的に向かってものをつくることが大事だと謳っています。

浜口：超上流や上流から落としていってソフトウェアを開発するというとき、現在はドキュメントをたくさんつくっていますね。でも、米国の会社はドキュメントをつくらないそうです。

松本：そうですね。まずプログラムを先につくりますね。

浜口：上流の構想がある程度具体的になっている場合、そこをきめ細かく書いていくと、コードを書いていくことと等しくなる。だから彼らは、「ソースコードがあればいいじゃないか」というわけです。

松本：そのようなやり方の一つに、形式手法があります。形式手法で上流工程をきちっと書いておくと、論理的なミスがかなり減少するんですね。その分、下流のテスト工程の負荷を軽くすることが出来ます。

浜口：なるほど。

松本：そういうメリットがあるのできちんと上流からスペックを決めてコードを書いていこうという流れはあると思います。とくに品質が求められるソフトウェアの場合は。

浜口：確かに論理的な矛盾がないようにすることは非常に重要ですね。

松本：ただ、日本では、形式手法はなかなか広がらないんですね。超上流になると、お客様と一緒につくるので、お客様にも形式手法で書いたことを理解してもらわないといけない。

浜口：形式手法の場合もアジャイル開発と同じで、お客様と一体的な作業になるということですね。

松本：ですからお客様もある程度ソフトウェア開発に関する知識を持っていただけるとよいですね。

浜口：ヨーロッパなどでも聞くのですが、ウォーターフォール型開発を含めて、上流から下流まで問題なく開発が進むことはないといいます。必ず仕様変更が出るし、仕様の漏れもある。彼らに言わせると、それは万国共通、It's a universal problem だそうです。

松本：確かに世界共通の問題ですね。

浜口：そこを前提にしているから、彼らは「コードを書けばいい」というところへいく。コンピュータにかかわる SE のビジネスランゲージはコードだと思います。

松本：私は標準化の国際会議で外国の人と C 言語でディスカッションしたことがあります。実際、C 言語でコミュニケーション出来るんですね。

グローバルな開発方法の確立が求められる

浜口：JISA が掲げているテーマの 1 つにグローバル化があります。日本経済は今後大きく成長するとは必ずしも想定出来ません。ですから売り上げを伸ばす場合には、どうしても海外のマーケットに出ていかざるを得ません。ただ、ソフトウェアというものはポータビリティがあります。鉄やガラスと違って運ぶのが簡単。どこでつくってもいいわけです。

松本：ネットワーク回線で送れますからね。

浜口：ソフトウェア開発は、現地で生産するほうが生産コス

トが安価という製造業のモデルとは違うのではないかと思います。ソフトウェアの場合は、南米で使うソフトウェアをロシアで開発してもいいわけです。そのときに必要なことは、開発方法をグローバルにしておくことです。ただしそこで「日本はこの開発方法です」と押し付けるわけにはいきません。グローバル化を突きつめていくと、開発方法だけではなく、グローバルなサービスを提供し、グローバルなパッケージをつくるところへ行き着くと思います。ただアプリケーションは法制度や文化が国によって千差万別であり別物です。開発方法についていえば、日本はグローバルな開発方法に適合出来ると思います。その一方で、ドキュメントを緻密につくったり、非常に高い品質を追求するという日本流のやり方は、現地で通用するかどうか、見極める必要があると思います。

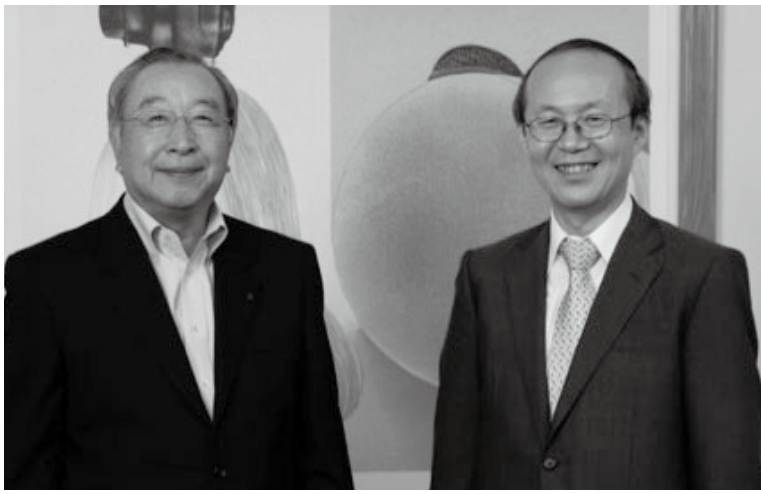
松本：そうですね。その一方で、海外では日本のやり方を新しい発見と感じるようです。先ほどの「超上流から攻めるIT化の原理原則17ヶ条」に書かれていますが、ユーザーと一緒に上流工程を考えましょうということや仕様はこの段階までに決めましょうという、ある意味で当たり前のことを外国に提案しているのです。彼ら自身、今言ったことを当たり前のようにやってきたのだけれど、「やってきたことがきっちり書いてある」ことが、外国人にとっては非常に新鮮に映るんですね。そういう意味で、日本として「世界に共通する開発フレームワーク」の方法論を出せるチャンスだと思います。

浜口：今、JISAのメンバーが進めている海外展開は中国や東南アジアが中心ですが、今後、米国やヨーロッパ、南米などを含めてもっとグローバルに展開していくとなると、1つのグローバルな開発方法をつくることが求められるでしょう。

クラウド時代のデータマネジメントとは

松本：開発プロセスも大事ですが、データ設計やばらばらなデータの形式を標準的なものにそろえるというデータマネジメントはソフトウェア・エンジニアリングに関する重要なテーマだと考えています。

浜口：まさにその通りです。企業のデータベースは、現状はそれほど整理統合されていません。データの鮮度や粒度もシステムの目的に合うようになっていないのではないでしょうか。そこで、データの更新サイクルをきちんと考えることが大切になります。また、不要なデータがデータベースに増えていくという問題もあります。そのため、あらゆる角度からデータの整理を行う必要があります。役割が終わったデータは捨てればいい。あとで利用する可能性があるかもしれないというデータはアーカイブにすればいい。このように、マスターデータはクリーニングをして整理しないといけない。そうしておかないと、サービス提供型のクラウドにデータを預けることも難しい。そこで、データの管理や利活用の仕組みづくりのために必要なガイドラインの提供や提言を行う組織として、一般社団法人日本データマネジメント・コンソーシアム（Japan Data Management Consortium、JDMC）が活動しています。



松本：データマネジメントはクラウドにかかわるテーマですね。

浜口：米国に視察に行った際、現地の先進企業のシステム部門の方がこう話していました。「クラウドにデータを預ける際に、どういうデータベースを使用し、どういう方式で預かるのか、その点を確認することが大切だ」と。クラウドにデータを預けても、いつかそのクラウドから他に移すときがくるかもしれない。そのときにトランスポート出来るデータ形式でクラウド側が管理しているかどうか、確認しておきなさいということです。

松本：何年か経って、クラウドからデータを引き出そうとしたときにデータが読めないということになっては困ります。

浜口：そういうことです。日本でもクラウドが注目されていますが、データを引き出すことまで含めて、きちっと戦略を考えるところまでいっているでしょうか。

松本：政府も貴重なデータをたくさん持っていますが、各省庁のデータ形式はばらばらです。

浜口：とくに公共データは、再利用しやすい形で公開するオープンデータ戦略を進めていくことが大切です。例えば、国土地理院が持っている地図データベースは非常に貴重な情報です。その情報をもっと活用すればいろいろな新しいサービスが実現出来ると思います。

松本：SECの活動にとっても、JISAやJDMCの活動は非常に参考になります。本日はありがとうございました。

文：小林 秀雄　写真：越 昭三朗

事例紹介:OEMソフトウェア製品の検証プロセス

名倉　正剛†，田村　清朗†，川口　真司‡*，高田　眞吾☆，飯田　元‡

他組織がOEMによって開発したソフトウェアを販売する場合,販売を行う企業は自社製品と同様に品質に責任を持つ必要がある.しかし資本関係のない他組織が開発しているため,組織間で同程度の要求を満たすように設計されていないことが多い.そのため他組織の開発したソフトウェアを利用しても期待するほど生産性が高くならない.本研究では,海外他社がOEMによって開発した企業業務に利用されるソフトウェアを自社製品に導入する際に行った検証プロジェクトの経過観察をした.その結果,このプロジェクトでは重要な影響を与える不具合が,検証プロセスの後半に発見されていた.発見された不具合の報告状況から,そのような状況を回避するために,発注先で実施した単体テストや統合テストに関する検証項目のリストの開示,要求定義に関する追加的な情報の開示の効果を考察した.そして,それらを開示した場合の検証作業プロセスに関する指針を述べた.

Case Study : Verification Process for an OEM Software Product

Masataka Nagura† ,Seiro Tamura† ,Shinji Kawaguchi‡* ,Shingo Takada☆, and Hajimu Iida ‡

When a company sells a product made by another company as an OEM product, that company is responsible for the quality of the OEM product in the same way as its own product. However, because the two companies do not have any ties in terms of capital, the OEM product is often not developed according to the shared requirements of the two companies. Accordingly, the productivity due to incorporating OEM is not as high as expected. In this paper, we describe the results of observing the verification process that was taken for an actual OEM software product. We especially found that critical bugs were found later in the verification process, which could be avoided if information such as the original requirements and test data were made available.

1 はじめに

ソフトウェア製品やその開発プロジェクトは，年々大規模になっている.そのため,ソフトウェア開発組織間でモジュールごとに受発注を行ったり，更には他社がOEM（Original Equipment Manufacturer）として開発したソフトウェアを自社製品として導入したりすることによって，開発コストを削減した上で大きな利益を得られるように，開発体制が変化している.

一般的に商用ベースのソフトウェア開発において，他組織が開発したソフトウェアを利用する場合，販売企業はその品質にも責任を持つ必要がある．しかし資本関係の無い他組織のソフトウェアを利用する場合，組織間で同程度の要求を満たすように設計されていないことが多い［BARBOSA2007］.このため，販売企業は他社開発のソフトウェア部品を利用する際に，再度検証を実施することになる．要求に関する認識が異なっていたり，要求自体が異なっていたりすると，この検証作業に多くのコストを要する．結果として他組織の開発したソフトウェアを利用しても，期待するほど生産性が高くならないというのが現状である.

本研究では，このような状況における検証の実例として，企業業務に利用される海外他社によってOEMによって開発されたソフトウェアを自社製品として導入する際に，実際に行われた検証プロジェクトの経過を観察した．そしてOEMソフトウェア製品を検証する際に，どのような問題が発生するか分析した．その上で，発注先と発注元企業でどのような情報をやり取りして，どのように検証プロセスを進めれば効率的に検証を実施出来るか，指針を示す.

† 株式会社日立製作所，Hitachi,Ltd.
‡ 奈良先端科学技術大学院大学，Nara Institute of Science and Technology
☆ 慶應義塾大学，Keio University
* 現在，有人宇宙システム株式会社

2 他組織の開発したソフトウェアを導入する際の課題

Robinson らは，複数の組織によってシステムを開発する際の発注元と発注先の関係を，ベンダ間の資本的関係と地理的関係によって，次のように分類している［ROBINSON2004］.

●資本的関係の無いベンダへ発注（Outsourcing）

① 国内ベンダに発注（Onshore Outsourcing）

② 国外ベンダに発注（Offshore Outsourcing）

●資本的関係のある同一組織内のベンダ（子会社，関連会社など）へ発注（Insourcing）

③ 国内ベンダに発注（Internal Domestic Supply）

④ 国外ベンダに発注（Offshore Insourcing）

このように，外部組織で開発したソフトウェア製品を利用してシステムを開発することを“Outsource”と定義し，関連会社などのような自組織内で開発したソフトウェア製品を利用してシステムを開発することを“Insource”と定義している．また，自国内での生産かどうかによって，“Onshore”及び“Offshore”に分類している．本論文における本節での記述では，Offshore Insourcing と Offshore Outsourcing を合わせて，「オフショア開発」と記載する．

ソフトウェアの大規模化により，近年では Outsource によって外部の組織が開発したソフトウェアを部品として利用してソフトウェアを開発する場面が多い．更には開発コストを削減した上で大きな利益を得られるように OEM によって開発された他社製ソフトウェアを自社製品に導入して販売することもある．このことは，主に開発期間の短縮や，人的／経済的な開発コストの削減を目的としており，新技術に追随するために OEM によって開発された海外製のソフトウェアを導入することもある．しかし実際には，Outsource 開発がコスト削減に必ずしも役立っていない場合も多い．

総務省がソフトウェア開発を行う日米の約 700 社へ実施したアンケート調査［MIC2007］において，オフショア開発を実施していると答えた 212 社の回答（日本 96 社，米国 106 社）を参照すると，オフショア開発導入時のコスト削減見込みに比べ，実際に得られた削減効果は 10% 近く低くなっている※1．なお，回答した企業には Insource 開発（［ROBINSON2004］における④に相当）を実施している企業も含まれていたが，多くは Outsource 開発（［ROBINSON2004］における②に相当）を実施していた※2．この結果から，海外の他組織で開発したソフトウェア製品を利用することが，期待するほどのコスト削減に結びついていないことが分かる．また，［ITO2010］では，それぞれの開発形態における生産性を比較している．これはソフトウェアに限らず日本の製造業の全業種を対象に生産性の推移を分析しているものであり，必ずしもソフトウェア開発の傾向を表しているとは言えないが，資本関係を持たない企業に対する Offshore Outsourcing（［ROBINSON2004］における②に相当）が生産性の向上に有意な効果を持たないことが報告されている．このように，［MIC2007］［ITO2010］のどちらからも，Outsource 開発がコスト削減に必ずしも役立っていないことが分かる．

Outsource によって外部の組織が開発したソフトウェアを部品として購入し利用する場合には，通常は次のような特徴が存在する［VIDGER1996］.

A） 発注元企業では，ソースコードにアクセス出来ない．また，内部文書が存在しない場合が多く，存在しても発注元企業からアクセス出来ない．

B） 技術情報（制限事項，パフォーマンス，リソース消費量等）は十分に記述されていない場合が多い．ユーザレベルの文書やトレーニング情報は，十分に記述されていることが多い．

発注元企業は，このような特徴を持ったソフトウェア部品を組み合わせてソフトウェアを開発する．Insource によって開発を実施する場合に比べ，開発プロジェクトを推進する際に参照出来る情報が大幅に少ない．なおこのような特徴は発注元と発注先との契約関係に依存するため，必ずしもすべてのプロジェクトに当てはまらない．しかし，OEM による Outsource 開発を実施する場合は，製品を納入することのみが契約になっていることが多く，このような特徴がとくに現れやすい．

他社開発のソフトウェア部品を導入する際には，発注元企業においても検証を実施する場合が多い．なぜなら，発注元企業では購入したソフトウェア部品に対して自社開発製品と同様の信頼性と品質を求めているものの，発注先においては発注元と同程度の品質を満たすような要求に基づいて設計されていない場合が多いからである［BARBOSA2007］※3．OEM によって開発されたソフトウェアを自社製品に導入する場合，既に販売されている製品が納入されるため，発注元では厳密な検証作業を省略出来るはずである．しかし導入したソフトウェアに対しても，発注元企業である自社がその品質に責任を持つ必要があるため，一般的には発注元企業の品質を満たすことを納入時に確認することが多い．このため，納入時に検証作業を省略出来ないことになる．このことが期待するほど生産性を高めることが出来ない一因になっていると考えられる．

我々はこのような状況を確認するため，実際のソフトウェア開発プロジェクトにおいて，導入時の検証作業としてどのようなことを実施しているかを観察した．本論文ではその一例として，海外ベンダが開発した OEM ソフトウェアを，自社製品群のラインナップに追加して販売する際に実施した検証プロジェクトの経過を紹介する．そして海外他社製品を導入する際の検証作業を推進する上で，どのような部分が問題になり検証コストが増加しているかを分析する．その上で問題を回避するために検証作業で留意すべき点について論じる．

脚注

※1 日本企業の場合，オフショア開発導入時にコスト削減効果は平均 34.7% と見込まれていたが，実績は平均 25.2% であったと報告されている．

※2 日本企業の場合，回答企業の 70.8% が，関連会社以外に開発を委託していた（［ROBINSON2004］における②に相当）．

※3 ［BARBOSA2007］では，入力に対して応答を返すタイミングや，ソフトウェア障害発生時のシステムに与えるインパクトに関する要求が挙げられている．

表1　検証対象ソフトウェアの規模

ソフトウェア全体		JAR ライブラリ		スクリプトファイル	
ファイル数	サイズ	ファイル数	サイズ	ファイル数	コード行数
20,917	1,442.9MB	932	732.1MB	385	92,733 行

表2　不具合報告のスケジュール

指摘回	経過期間	指摘内容に含める不具合についての初期方針	作業開始時の想定	
			重要度	報告数
第一回	約 12 週後	ユーザが必ず実施する手順に関連する不具合. （インストールや起動に関して発生した不具合や，当該ソフトウェアを利用する上で必ず行われる基本的な操作を行った際に発生する不具合など）	高	多
第二回	約 22 週後	場合によっては実施する可能性が高い手順に関連する不具合. （他製品と組み合わせる際の手順や，より専門的な作業を実施する場合に利用する当該ソフトウェアの詳細な利用手順に発生する不具合など）	中〜低	中
第三回	約 26 週後	上記に含まれない／上記において指摘漏れとして残った不具合で，リリースまでに緊急に修正する必要がある不具合. （リリース直前であるので，ドキュメントを添付することにより，ユーザに回避策を提示出来る場合は，この段階では不具合を報告しない）	高 （修正に多くの手間を要しないもの）	少

3 検証対象プロジェクト概要

本章では分析を行った検証プロジェクトの概要について述べる.

3.1 検証対象ソフトウェアの規模

検証対象ソフトウェアは主に Java で実装されており，一部についてはユーザのカスタマイズを可能にするために，スクリプト言語で実装されている．スクリプトを除いた部分の製品は，バイナリで納入されている．検証対象ソフトウェアの規模として，ソフトウェア全体のファイル数とサイズ，Java の JAR ライブラリファイル数とサイズ，スクリプトのファイル数とコード行数を表 1 に示す.

また，このソフトウェアにはユーザ向けのドキュメントが添付されていた．これは，2,221 ページから構成され，ほとんどの記載内容は，インストールと，UI 操作に関する説明であった. UI 操作に関する説明は，画面上の要素の操作（ボタンのクリック，テキスト入力など）に関してのみで，どのような場合にその操作を実行するのか，という利用シナリオに関する説明がほとんど記述されていなかった.

3.2 作業スケジュール

OEM によって開発された他社製品を自社製品として販売するにあたって，約半年間で検証作業を行った．社内製品を検証する場合とは異なり，検証した結果生じた不具合に関する指摘事項を随時報告出来る状況ではなかった．そのため，ある程度指摘事項を蓄積しておき，不具合による影響の重要度に応じ 3 回（約 12 週／ 22 週／ 26 週後）に分けて指摘を行うように，スケジュールを作成した（表 2）.

3.3 作業メンバの構成

検証作業は，合計 11 名のメンバによって行った．このうち 5 名は，10 年以上のソフトウェア開発経験を有しており，残りの 6 名は 1 年から 6 年であった．なお，メンバ全員が検証対象のソフトウェアの利用経験がない状態であった．検証作業の全期間に従事出来たメンバは 2 名のみで，他のメンバは途中から参加したり，途中で検証作業から抜けたりしており，全体では 40 人月の工数で検証を行った.

3.4 検証作業実施方針

今回の検証作業は他社が OEM によって開発したソフトウェアの自社製品への導入のための作業であり，ソフトウェアのどのモジュールがどのような目的に利用され，そのためにどのように操作すべきか，ということを把握出来ている開発者が社内にいない．そこで検証作業の初期段階では，動作の把握が急務であった

しかし 3.1 節で述べたように，ドキュメントのほとんどの部分は UI 操作に関する説明であり，ソフトウェアの動作を把握することは困難であった.

そのため，最初に次の作業の実施を計画した.

【フェーズ 1：UI の要素に基づく検証】ドキュメントに記載された各 UI の要素につき，チェックリストを作成した上で，動作の検証を行う．主に，UI の要素を操作したときの処理や画面遷移が，ドキュメントの記載通りに行われるかどうかを確認する．また，チェックリスト作成の際に UI の各要素を操作することにより，検証対象ソフトウェアの動作を把握する.

検証対象ソフトウェアの処理が実際に正しく動作しているかどうかについては，個々のUIの各要素の操作を確認するだけでは検証出来ない．品質を完全に保証するためには，あらゆる状況下における処理結果を確認しなければならない．しかし，検証作業の期間が限られており，すべての状況で処理を試行することは現実的ではない．ある程度利用場面を想定して検証作業を行う必要があり，次の作業の実施を計画した．

【フェーズ2：ユースケースに基づく検証】検証対象ソフトウェアを利用して行う業務のユースケースを作成する．そして作成したユースケースから，どのような状況下でどのような検証を行う必要があるかを明らかにし，ユースケースの一連の流れを明記したチェックリストを作成した上で，検証を行う．

このフェーズでは，まず製品導入を決定した企画部門が作成した製品開発ロードマップを参照し，検証対象ソフトウェアを発注元の製品ラインナップに加える目的を整理した．そしてその目的を実現するためのユースケースを導出した．なお，発注元の製品ラインナップに含まれることになるので，発注先が想定していない他の製品と一緒に利用するようなユースケースも導出した．そして，そのユースケースを満たす一連の操作手順に基づいて，チェックリストを作成した．

検証作業を実施するうちに，実行時の状況を変化させると，期待通りに実施出来ないユースケースが存在することが判明した．実際に検証したソフトウェアとは異なるが，例えば文書作成のための海外製のエディタを導入する目的で検証作業を実施していて，「ファイルを編集する」というユースケースが存在したとする．そこでこのユースケースに基づいて検証したところ，日本語文字列を含むファイルを編集出来るが，ファイル名に日本語文字列を含む場合にはファイルのオープンすら出来ない，という状況である※4．そこで，少なくとも基本的なユースケースについては，発注元で想定出来る範囲内で実行時の状況を変化させたときに，期待通りに実施出来ることを確認する必要が生じた．このため，フェーズ2と並行し次の作業を実施することを計画した．

【フェーズ3：基本ユースケースに関する重点検証】フェーズ2で作成したユースケースから，検証対象ソフトウェアを導入する際に最低限実行出来る必要がある基本的なものを抽出する．そして，導入対象ソフトウェアを利用する顧客に対して調査を実施する※5．その結果，想定出来る範囲で検証対象ソフトウェアが動作するいろいろな状況を用意し，抽出したユースケースを期待通り実施出来るかどうかを確認する．

なお，フェーズ1，フェーズ2のように，チェックリストを作成することで検証すべき項目を明確化出来るものに関しては，チェックリストを作成する担当者と，その作業を実施していく担当者に分け，チェックリスト作成作業に起因する思い込みがなるべく生じないように配慮した．

4 検証作業実施経過

4.1 （フェーズ1）UIの要素に基づく検証

3.1節で述べたように，検証対象のソフトウェアは規模が大きいため，ドキュメントに記載されているすべてのモジュールの動作を精査するのは困難であった．そこで最初に，自社製品への導入目的として利用されるモジュールを選別した．そのようなモジュールとして，全31モジュール中，13モジュールを選別した．

作業開始から6週目にかけては検証作業に取り掛かれるメンバが少なく，メンバのうちの2名によりユーザ向けドキュメントを参照しながらこの選別作業を実施した．ただし，どちらのメンバも他の作業と兼任であったため，本格的に検証作業を開始出来たのは，残りのメンバが加わった7週目以降であった．

フェーズ1では，選別したモジュールのUIの要素を検証した．この検証作業は，作業開始7週目から14週目にかけて実施した．

まずドキュメントを参照して，各UIの要素について，チェックリストを作成した．その際に，説明に間違いがあって，UIの要素の操作手順と一致しない場合には，ドキュメントの不良として不具合指摘を行った．また，ドキュメントの記載から漏れているUIの要素がいくつか存在した．その場合は，他のUIの要素と共通の部品であると考えることが出来，発注元で動作を想像出来るものについては，共通部品としてUIの要素のチェックリストを参照してチェックリストを作成した．発注元で動作を想像出来ないものについては，不具合報告のタイミングで発注先に質問を行い，回答後に追加でチェックリストを作成して検証作業を実施した．

ドキュメントの記述の例を図1に示す．これは，プロパティダイアログボックスを表示するための操作と，ダイアログボックス内に表示されたUIの各要素についての操作方法と，操作の結果として得られる出力を記述している．検証対象のソフトウェアは画面表示をしてユーザに操作させる手順が多く，たくさんの画面から構成されていたが，すべての画面表示について，ドキュメントには，このような操作方法に関する記述がされていた．

このドキュメントに基づいて作成したチェックリストの例を表3に示す．ドキュメントに記述されている操作を実施した際に，記載された通りの出力が得られるかどうかを確認出来るように，チェックリストを作成した．検証対象ソフトウェアには，ユーザ向けのドキュメントのみが添付

脚注

※4 この例で発注先は，日本への納入に向けてファイル編集のユースケースについて日本語での記述を考慮していたが，そのユースケースを実施する際の状況としてファイル名に日本語が含まれることを考慮していなかった．このことが，発注元の想定と異なっている．

※5 検証対象ソフトウェアを発注元の既存販売製品のラインナップに加えて販売する予定だったため，既存販売製品の顧客に検証対象ソフトウェアを販売することを想定していた．このため，販売パートナーから具体的な顧客環境をヒアリングした．

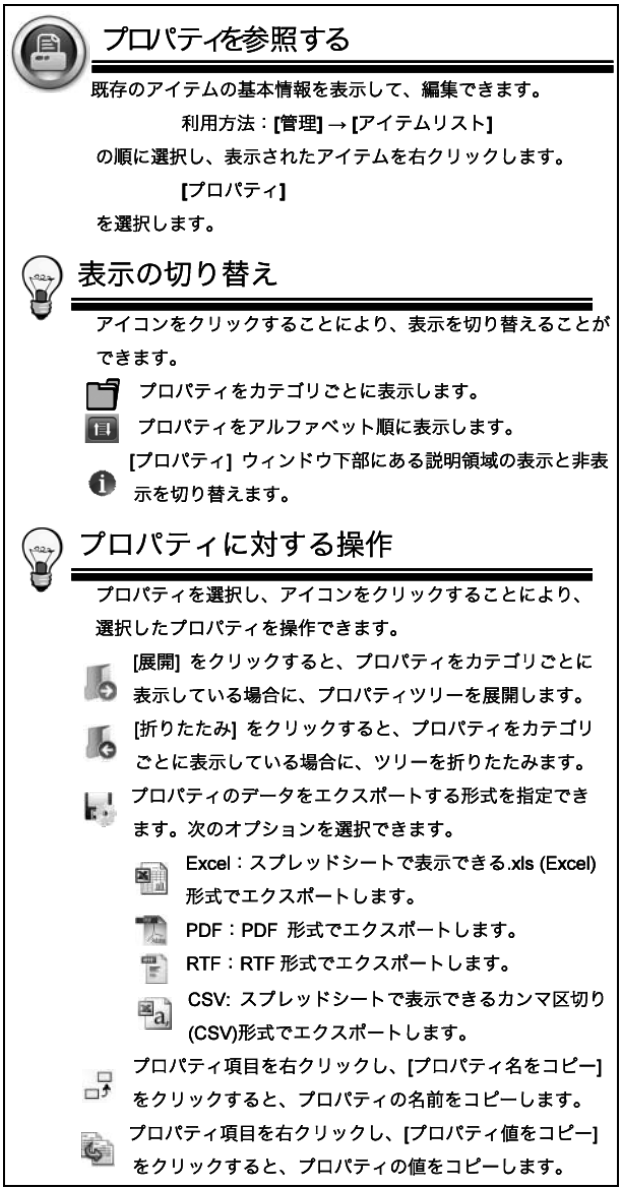


図1　UIに関するドキュメントの記述の例（抜粋）

表3　作成したチェックリストの例

#	関連する UI	チェックリスト内容のサマリ（以下の動作を確認）
1		ボタンをクリックすると，カテゴリ別のプロパティが表示される
2		ボタンをクリックすると，アルファベット順のプロパティが表示される
3		ボタンをクリックすると，プロパティウィンドウの下方の詳細表示エリアが表示または隠される
4		（カテゴリ別表示の状態で）ボタンをクリックすると，プロパティツリーが展開される
5		（カテゴリ別表示の状態で）ボタンをクリックすると，プロパティツリーが折りたたまれる
6		「エクスポート」を実行すると履歴がエクスポートされる（選択に応じて，Excel，PDF，RTF，CSV，XML の各形式で出力される）
7		プロパティの項目を右クリックし，「プロパティ名をコピー」によって，プロパティ名がコピー出来る
8		プロパティの項目を右クリックし，「プロパティ値をコピー」によって，プロパティ値がコピー出来る

表4　不具合報告件数
（カッコ内は各フェーズの報告件数合計に対する割合）

重要度	A	B	C	D	合計
フェーズ 1	0 (0%)	28 (18%)	76 (49%)	50 (33%)	154
フェーズ 2	0 (0%)	10 (10%)	54 (53%)	37 (37%)	101
フェーズ 3	2 (3%)	27 (44%)	31 (50%)	2 (3%)	62

されており，ソフトウェアの動作を理解するためにはそれを参照するしか手段が無かった．このためフェーズ 1 では，UI 操作を通して検証作業メンバがソフトウェアの動作を把握することも目的とした．

このような作業を実施し，755 件のチェックリストを作成した．そして，作業開始 9 週目から順次チェックリストに沿って検証を行った．なお後述の各フェーズも含め，報告者は不具合報告の際にその問題の重要度を，高い順に「A（導入の目的とする業務に致命的な影響を与えるもの）」，「B（致命的ではないが重大な影響を与えるもの）」，「C（与える影響は軽微だが修正する必要があるもの）」，「D（運用でカバー出来る場合など，修正する優先順位が低いもの）」の 4 段階で評価した．

表 4 に後述のフェーズも含めたフェーズごとの不具合報告件数を示す．フェーズ 1 ではドキュメントの不良に起因するドキュメントの記載を修正する必要がある不具合が多く発生したため，重要度 D よりも重要度 C の不具合が多くなっている．またフェーズ 1 では，UI に関するドキュメントのみを基に作業を実施した．複数の別画面で UI に対する別手順により実施される処理が同一データを利用する場合には，それらの処理が同じデータを基に実行されていることを検証結果から推測して把握する必要があり，状況の把握が困難であった．

4.2 （フェーズ 2）ユースケースに基づく検証

フェーズ 2 では，販売先として想定しているパートナー企業からの情報に基づき，具体的にこのソフトウェアを利用する部署を想定した上で，検証対象ソフトウェアを利用した業務に関するユースケースを作成した．そしてその際の操作に合わせて，ユースケースを満たす操作手順を，チェックリストとして挙げた．

例えば，「エディタによってファイルを編集する」という

ユースケースを作成したとする．このユースケースを満たすために必要な操作としては，単純に文字を入力したり挿入したりするような操作であったり，いろいろな文字列をコピーして貼りつける操作であったり，いくつもの操作が考えられる．それらの操作として考えられるものをすべて列挙し，チェックリストを作成した．

フェーズ 2 の作業は，作業開始 12 週目から 22 週目にかけて行われた．まず 13 個のユースケースを作成し，441 件の操作手順をチェックリストとして挙げた．そして，作業開始 14 週目から順次チェックリストに沿って検証を行った．

3.4 節で述べたように，フェーズ 1 ではある程度モジュールを選別してから検証を行った．フェーズ 2 で検証対象としたモジュールの一部には，フェーズ 1 で選別していないものが存在した．そのため，フェーズ 1 で検証を行わなかった UI を検証することがあり，フェーズ 1 と同様にモジュールの動作不良の不具合報告よりも，UI についてのドキュメント記述ミスに起因する不良という不具合報告が多くなった．このため，表 4 に示したように，不具合報告件数の重要度の割合はフェーズ 1 と同じような傾向になり，重要度 A や重要度 B の不具合の割合が小さくなっている．なお，検証対象ソフトウェアには UI に関するドキュメントしか存在せず，発注元の目的とするユースケースを実現するためにそのドキュメントを利用出来るかどうかは，UI の説明から推測する必要があった．

4.3 （フェーズ 3）基本ユースケースに関する重点検証

フェーズ 2 で洗い出したユースケースのうち，検証対象ソフトウェアを利用した業務の実施に最低限必要となるものを抽出した．その上で，発注元企業が想定した状況下で，抽出した個々のユースケースに含まれる操作を実施出来るか，フェーズ 2 と並行し検証した．

フェーズ 3 の作業は，作業開始 19 週目から 22 週目にかけて行われた．フェーズ 3 ではチェックリストを作成せず，検証対象ソフトウェアの基本的な処理が正常動作する状況を準備しながら動作確認を行った．

表 4 を参照すると，フェーズ 1 やフェーズ 2 に比べ，重要度が「B」,「C」の報告の割合が大きく，かつ「D」の報告の割合がとても小さくなっている．フェーズ 3 は，自社製品への導入に至る主目的を満たす処理の動作検証であり，不具合の存在が製品化に重大な影響を与えていることが分かる．フェーズ 3 では，発注元で満たすべきと考える品質に基づいて，検証工程末期にも関わらず単体テストの一部を実施したことになるため，発見された不具合の影響が比較的大きくなった．

4.4 検証プロセス全体での不具合報告の推移

フェーズ 1 及びフェーズ 2 では，最も多い不具合報告が，重要度「C」のもの，そして次が重要度「D」のものになっており，この 2 つで全報告の多くの割合を占めている．その一方で，フェーズ 3 のみが他と異なり，重要度「B」のものと「C」のものが同程度存在しており，この 2 つだけで全報告のほとんどを占めている．

検証対象のソフトウェアは，既に販売されている製品であり，発注先で一旦は検証しているはずである．フェーズ 1 及びフェーズ 2 では，発注先において動作すると想定していたことが見込まれる状況で検証を実施した．そのため報告された不具合は，日本導入にあたり作成されたドキュメントの内容の誤記載に起因するものが多く，フェーズ 1 とフェーズ 2 の全 255 件中 110 件がドキュメント内容の誤記載であった．

フェーズ 2 で作成したユースケースを実施する際に，発注先で想定されていない状況があることが推測出来たために，フェーズ 3 の検証作業が必要になった．フェーズ 3 の検証作業は基本的なユースケースに関するものであるにもかかわらず，フェーズ 2 の作業を実施するまでは作業の実施自体が計画されていなかったものである．このことにより，結果的に検証作業工程の後半になって重要度の高い不具合の報告が発生した．

全フェーズでの不具合報告状況推移として，期間全体を一週間ごとに区切った不具合報告件数の累積を図 2（a）に，フェーズごとの不具合報告件数の推移を図 2(b)に図示する．表 2 に示したように発注先への 3 回の指摘では，出来るだけ早い段階で重要な不具合を指摘出来，検証期間の後半では，不具合の発生状況が収束すると予測していた．しかし実際には，検証作業の遅い時期に重要度の高い不具合が多数発見された．とくに図 2（b）に示したように，フェーズ 3 については検証作業の終盤の短期間に実施したにもかかわらず，重要度 B の不具合が他フェーズよりも多く発見された．なおフェーズ 3 の検証作業は，当初の作業計画に対する追加の作業であったが，発見された不具合は修正しないと製品として出荷出来ない状態となり，結果として検証作業期間を約 1 カ月延長することになった．

5 OEM 製品検証作業プロセスの課題

5.1 対象にした検証プロジェクト事例の課題

2 章で述べたように，資本関係の無い企業間で Outsourcing を実施する際には，発注元と発注先の品質基準が異なるため，発注元において検証作業を実施することになる．本論文で例示した検証プロジェクトも，同様に発注元で実施した検証作業である．この作業の各フェーズにおいて，次の問題が発生した．

【フェーズ 1】 UI に関するドキュメントしか存在しないため，複数の別画面に対する手順により実施される別個の処理が同一データを利用する場合に，そのことを把握出来なかった．このため，ある UI について実施した検証と別の UI について実施した検証が同一データを利用して同一目的の処理を実施している場合に，状況の把握が困難になった．

【フェーズ 2】 UI に関するドキュメントしか存在せず，ユースケースを実現するためにどの UI を利用すべきかを UI の説明から推測する必要があった．

【フェーズ 3】 発注元で満たすべきと考える品質を発注先が満たさなかったため，単体テストの一部を実施する必要が生じた．

これらは，2 章で述べた次の状況に起因するものである．

状況 1）発注元が要求や設計に関わる情報を参照出来ない［VIDGER1996］.

状況 2）発注先において発注元と同程度の品質を満たすような要求に基づいて設計されていない場合が多い［BARBOSA2007］.

OEM によるソフトウェア供給関係では，発注によって開発が開始される通常の Outsourcing とは異なり，発注元の要求に基づいてソフトウェアが実装されるものではない．通常の Outsourcing では，自社の要求の確定後にソフトウェア開発が実施されるため，発注元の要求を満たしたソフトウェアが納入される．しかし OEM では，発注元の要求と納入されるソフトウェアの満たす要求が完全に一致しない場合がある．今回の事例では，発注元はある特定の新技術を提供するために OEM によって開発されたソフトウェアを自社製品に導入して販売したが，当該技術を利用するために必要な動作環境に対する想定が発注先と異なっていた．

更に OEM によって他社製品を導入する際には状況 1）に挙げたように，要求に関わる情報を発注元が参照出来ず，発注先がどのようなユースケースを想定しているのかを確認出来ない．今回の検証作業の初期段階では，発注元のユースケースにおいて想定する業務の実施に最低限必要になる基本的な処理が，発注先が想定するユースケースを実現する上でも必要な処理であり，十分に検証が行われているものと推測していた．しかし検証作業を進めていくにつれて，それらの処理について発注元の想定する状況に合致する検証が十分に実施されていないことが判明した．その結果，発注元のユースケースを実現するための基本的な処理に対する検証作業が作業工程の終了近く（フェーズ 3）に実施されて工程末期に重要な不具合が発見されることになった．発注元から発注先のユースケースを参照出来れば，発注元が想定するユースケースにおいて実行される基本的な処理を発注先のユースケースでどのように扱っているかを知ることが出来る．これにより，発注先が当該処理について，どのような想定で検証作業を実施したのかを正しく推測出来る．

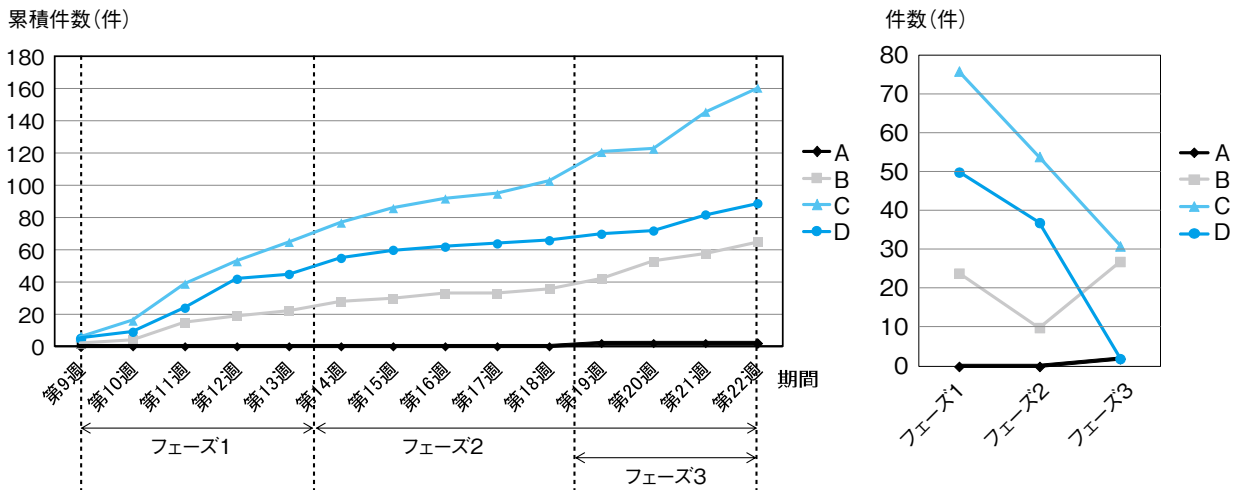
更に，状況 2）に記載したように，海外製ソフトウェアを導入する場合に，発注元と発注先でテストの品質基準が大きく異なる場合が多い．前述のエディタでのファイル編集の例では，日本への納入に向け日本語ファイル名の指定を想定して検証を実施することは発注元では当たり前の品質基準であったが，発注先ではそれが満たされていなかった．いくつかのモジュールの動作を確かめたところ，その当たり前の部分で品質が満たされていないことが判明し，結果的にフェーズ 3 で単体テストを追加的に実施した．このように今回の事例では，発注元の品質基準を発注先が満たしていなかった．すべての事例においてこのような状況にはならないが，発注元と発注先での品質基準の相違を，発注元で確認する必要がある．その際に，発注先でどのような基準に基づいて検証が実施されたかをあらかじめ知ることが出来れば，検証プロジェクトの早い段階でフェーズ 3 の単体テスト実施の要否を決定出来る．その結果，早い段階で重要な不具合を検出出来るようになる．

このように，OEM によって開発されたソフトウェアを自社製品へ導入する際の検証作業では，次の情報が発注先から発注元に開示されていれば円滑に作業を遂行出来る．

A） 要求定義や，想定したユースケースに関する詳細な情報（状況 1 に対応）

B） 単体テストの検証項目リスト（状況 2 に対応）

A）に関しては，通常は統合テストでの検証項目とほぼ一致するはずである．従って，単体テスト，統合テストに関する検証項目のリストと，それらが不十分な場合に備え追加的に要求定義に関する情報が開示されていれば，検証作業を円滑に実施出来たと考える．今回の検証作業ではフェーズ 3 の実施により約 1 カ月の作業遅延が発生した．フェーズ 3 では前述のように単体テストの一部を実施したと考えることが出来，他のフェーズとは独立して作業を実施可能であると考えられる．従って，早い段階でこの作業を実施出来れば，約 1 カ月の作業遅延の期間を抑えることが出来る．すなわち A）や B）の情報が開示されれば，フェーズ 3 で実施した検証作業追加の判断とそれに伴うスケジュール調



(a) 全期間での累積件数推移　　(b) フェーズごとの件数推移

図2　不具合報告件数推移

整を初期段階で実施出来，作業期間を１カ月近くは短縮出来たと考えることが出来る．

5.2　OEM ソフトウェア検証作業プロセス推進の指針

OEM によって開発されたソフトウェアを自社製品に導入する場合，今回事例として紹介した検証作業プロセスのようにプロセスの後半で重要度の高い不具合が発見されることを防ぐためには，前述情報を発注先から入手出来るように契約を行うべきである．それによって，次のような順序で発注先の想定していないユースケースから順に検証を実施することが出来る．

手順①　発注元のユースケースのうち，発注先が想定するユースケースと異なるものを抽出する．

手順②　手順①で抽出したユースケースを満たすために必要なモジュールの単体テストを実施する．

手順③　手順①で抽出しなかったユースケースを満たすために必要なモジュールの単体テストを実施する．

手順④　発注元の全ユースケースについて，結合テストを実施する．

手順⑤　発注先の想定するユースケースのうち，手順④で結合テストを実施していないものを満たすために必要なモジュールの単体テストを実施する．

手順⑥　発注先の想定するユースケースのうち，手順④で結合テストを実施していないユースケースについて，結合テストを実施する．

このような手順に基づいて検証プロセスを実施することにより，手順②において，発注元が想定しているユースケースのうち，発注先で想定していなかったユースケースに基づいて，モジュールの単体テストを実施出来る．今回対象にした検証プロジェクトでは，発注元が想定しているユースケースのうち，発注先で想定していなかったユースケースに関連する不具合の方が，結果的に重大な不具合になっており，これが検証プロセスの後半（フェーズ 3）で発見された．発注先で想定しているユースケースに基づいた単体テストは発注先で実施していると期待出来るため，手順②では発注先で想定していないユースケースに基づく単体テストを優先的に実施する．これにより，検証プロセスの初期段階で，発注先の想定していなかったユースケースに基づく重要な不具合を発見することが可能になる．

この手順を実施するためには，発注先から前述の A），B）の情報を入手する必要がある．パッケージソフトウェアなどの一般的に販売されるソフトウェアではなく，顧客の要求に合わせて開発されるソフトウェアでは，要求定義などに顧客の機密情報を含む場合があり，OEM 開発で供給される側の発注元では，これらの情報を入手出来ない可能性が高い．ただし，ある会社において顧客の要求に合わせて開発されたソフトウェアには顧客ノウハウが混入することが多いため，一般的にそのようなソフトウェアを別の会社が OEM で導入することは考えにくい．

仮にそのような状況がある場合でも，受発注者間の開発状況を可視化するための提案［MATSUMOTO2009］や，それを実現するための各種支援ツール群［INOUE2009］を利用することで，発注先がどのような想定で開発を実施したか，どのようなテストを実施していたかを，発注元で知ることが出来る．これらの提案は，「いつ，どこで，誰が，どのように」開発したものであるかというトレーサビリティ情報（ソフトウェアタグ）をソフトウェアに添付し，受発注者間で開発プロセスを確認する際の手段として利用出来るような社会基盤と，その実現を支援するツール群を提供している．Outsource 開発において，ソフトウェアタグにある情報を添付することが標準的になれば，要求に関する情報について匿名化した上で，受発注者間でやり取りされるようになる．従ってこのような基盤を整備すれば前述のような情報を発注先から入手することは実現可能であり，それによって発注元での導入作業を迅速に実施出来るようになる．

6　まとめ

本研究では，企業業務に利用される海外他社の OEM によって開発されたソフトウェアを自社製品に導入し，販売する際に実際に行われた検証プロジェクトの経過を観察した．その結果，導入の際の検証作業を効率的に推進するために，ソフトウェア製品とユーザ向けのドキュメントに加え，発注先で実施した単体テストや統合テストに関する検証項目のリストと，要求定義に関する追加的な情報の開示の効果を検討した．そして，それらを開示した場合の検証作業プロセスについて指針を述べた．

謝辞

本研究を推進するにあたって貴重なアドバイスをいただいた，日立製作所 飯塚大介氏，川田伸一氏，村上貴史氏に，深く謝意を表します．

参考文献

[BARBOSA2007] R. Barbosa, N. Silva, J. Duraes, H. Madeira : Verification and Validation of (Real Time) COTS Products using Fault Injection Techniques, 6th Int' lIEEE Conf. on Commercial-off- the- Shelf (COTS) - Based Software Systems (ICCBSS'07), pp.233-242, 2007

[ITO2010] B. Ito, E. Tomiura, R. Wakasugi : Does Firm Boundary Matter?: The Effect of Offshoring on Productivity of Japanese Firms, RIETID is cussion Paper Series 10-E-033, 2010

[INOUE2009] 井上克郎，楠本真二，飯田元：事故前提社会に向けたユーザ・ベンダ間での開発データ共有ーソフトウェアタグ規格とソフトウェアタグ支援ツールー，SEC journal, Vol.5, No.4, pp. 234-243, 2009

[MATSUMOTO2009] 松本健一：事故前提社会に向けたユーザ・ベンダ間での開発データ共有ー StagE プロジェクトとソフトウェアタグー，SEC journal, Vol. 5, No. 3, pp. 198-203, 2009

[MIC2007] 総務省情報通信政策局情報通信経済室：オフショアリングの進展とその影響に関する調査研究，総務省平成 19 年調査研究報告，2007.http://www.soumu.go.jp/johotsusintokei/linkdata/other017_200707_hokoku.pdf

[ROBINSON2004] M. Robinson, R. Kalakota : Offshore Outsourcing: Business Models,ROI and Best Practices, US: Mivar Press, 2004

[VIDGER1996] M. R. Vidger, M. W. Gentleman, J. Dean : COTS Software Integration: State of the Art, NRC-CNRC Report, National Research Council, Canada, 1996

ソフトウェアプロジェクトデータにおける量的変数の予実差分析

東海大学理学部　教授
IPA/SEC 専門委員
古山　恒夫

IPA/SECで収集したエンタプライズ系ソフトウェアプロジェクトデータから、開発種別ごとにFP規模、工数、工期の予実差を求めた。予実差は各変数の計画値と実績値を対数変換してから回帰分析により計算した。その結果、新規開発プロジェクトではFP規模、工数、工期すべてプロジェクトの大きさによらず実績値が計画値を上回る傾向にあること、改修・保守ではプロジェクトが大きくなると実績値が計画値を下回る傾向にあること、新規開発プロジェクトのFP規模以外はプロジェクトが大きくなるに従って予実比率（=実績値／計画値）が低下する傾向が見られること、工期の予実差の推定式からのばらつきは、FP規模や工数に比べて小さいことが分かった。

1　はじめに

規模・工数・工期の３つの代表的な量的変数に関する計画値と実績値の予実差分析は既にデータ白書で報告されている［IPA2010］。しかし、これらの３つの量的変数はいずれも対数正規分布に従うことから［古山2011］、予実差分析においても、収集したデータの値を対数変換することでモデル化が容易になり、より詳細な分析が出来ると考えられる。

本稿ではこれら３つの量的変数の対数変換後の予実差、すなわち対数変換前の予実比率（＝実績値／計画値）の分析結果を報告する。モデル化をすることにより、データ白書に比べて、プロジェクトの大きさの変化に伴う予実比率の変化の傾向を明らかにしたこと、及び開発種別ごとにFP規模・工数・工期の予実比率を横断的に俯瞰したことが主な特徴である。

2　分析対象プロジェクト

2.1　分析対象プロジェクト

分析は、2011年４月末までにIPA/SECで収集したエンタプライズ系ソフトウェアプロジェクトデータ2,847件を対象に行った。

2.2　目的変数と説明変数

FP規模、工数、工期の予実差分析における、目的変数はそれぞれの「実績値」を、説明変数はそれぞれの「計画値」をそれぞれ常用対数で対数変換したものとする。

①FP規模

・目的変数：　log（FP規模（実績値））
　　＝log（5001_FP実測値_調整前）
・説明変数：　log（FP規模（計画値））
　　＝log（5084_調整前FP計画値_基本設計後）

②工数

・目的変数：log（工数（実績値））
　　＝log（10050_実績工数（総計人時）_プロジェクト全体）
・説明変数：log（工数（計画値））
　　＝log（11015_プロジェクト開発工数計画値（基本設計開始時点））

③工期

工期はプロジェクト開始日から終了日までのカレンダー日数を用いる。

・目的変数：log（工期（実績値））
　　＝log（実績日数_開発５工程）

・説明変数：log（工期（計画値））
= log（計画日数 _ 開発 5 工程）

2.3　データ件数

一般に回帰分析を行うためには少なくとも 30 件程度のデータは必要であると言われているので、ここでも 30 件以上データの揃っている層を分析対象とする。

3　分析結果

3.1　見積り過不足のプロジェクト件数

新規開発プロジェクト 143 件の FP 規模の予実比率を対数変換して大きさの順に並べたものを図 1 に示す。このような図を描くことにより、予実比率が 1 と等しい（対数をとると 0 になる）プロジェクト、1 より大きいプロジェクト、小さいプロジェクトそれぞれの割合と誤差の分布状況が把握出来る。

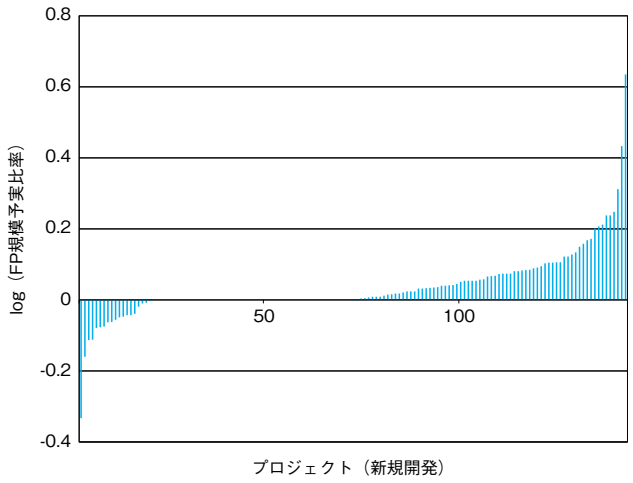


図1　新規開発プロジェクトのFP規模の予実比率

図 1 のような分布状況をもとに、FP 規模、工数、工期それぞれに関する、開発種別ごとの見積り過不足のプロジェクト件数を調べた結果を表 1 ～表 3 に示す。表 1 ～表 3 では、対数変換後の予実差が ± 0.0043 以内（対数変換前の予実比率が 99% から 101%）のものを「計画通り」としている。表 1 ～表 3 から次のことが分かる。

- FP 規模に関しては、新規開発では見積り不足（実績値が計画値の 101% を超える）のプロジェクトが半数近く（48%）あるのに対し、改修・保守では 7 割のプロジェクトが計画通りである（表 1）。
- 工数に関しては開発種別によらず半分以上（50% ～ 59%）のプロジェクトが見積り不足となっている（表 2）。
- 工期に関しては開発種別によらず半分以上のプロジェクトが計画通りとなっている（表 3）。

表1　FP規模見積りの過不足プロジェクト件数

開発種別	見積り状況			合計
	過多	計画通り	不足	
新規開発	18（13%）	56（39%）	69（48%）	143
改修・保守	7（12%）	41（71%）	10（17%）	58

表2　工数見積りの過不足プロジェクト件数

開発種別	見積り状況			合計
	過多	計画通り	不足	
新規開発	168（29%）	78（13%）	342（58%）	588
改修・保守	116（29%）	81（21%）	196（50%）	393
再開発	15（22%）	13（19%）	40（59%）	68
拡張	39（25%）	30（19%）	86（56%）	155

表3　工期見積りの過不足プロジェクト件数

開発種別	見積り状況			合計
	過多	計画通り	不足	
新規開発	75（15%）	254（51%）	168（34%）	497
改修・保守	49（17%）	185（64%）	56（19%）	290
再開発	8（16%）	31（61%）	12（23%）	51
拡張	25（14%）	96（56%）	52（30%）	173

3.2　対数変換後の回帰分析

FP 規模、工数、工期それぞれに関する対数変換後の回帰分析の結果を表 4 ～表 6 に示す。

表 4 から次のことが分かる。

- 従属変数（実績値）の変動を独立変数（計画値）の値でどれだけ説明出来るかを表す寄与率は、新規開

発プロジェクトでは 0.954 と高く、実績値はほぼ計画値から説明されることが分かる（実績値が計画値と等しいという意味ではないことに注意）。改修・保守の寄与率は 0.884 と新規開発ほど高くはなく、計画値に対する実績値にややばらつきがある。

- 新規開発の回帰直線の傾きは 1.028 であり、統計的に有意ではないが、FP 規模の計画値が大きくなるにつれて少しずつ予実比率が大きくなる傾向が見られる。一方、改修・保守の回帰直線の傾きは 0.909（有意水準 5% で傾きが 1 ではない）で、FP 規模の計画値が大きくなるにつれて予実比率が小さくなる。

表4　FP規模の計画値に対する実績値の回帰分析

開発種別	標本数	対数変換後の回帰分析			
		寄与率	傾き	切片	標準誤差
新規開発	143	0.954	1.028	-0.046	0.098
改修・保守	58	0.884	0.909	0.215	0.160

表 5 から次のことが分かる

- 寄与率は、新規開発が 0.906 と最も高く、再開発が 0.764 と低い。改修・保守と拡張はその中間である。
- 回帰直線の傾きは、開発種別によらず 1 より小さい。すなわち工数の計画値が大きくなるにつれて予実比率が減少する。このうち改修・保守と拡張は統計的に有意である（改修・保守が 5% 有意、拡張が 1% 有意）。再開発は他より標本数が少なく統計的に有意にはならない。新規開発の傾きは 0.990 とほぼ 1 に近く、工数の予実比率の変化はプロジェクトの大きさには依存していないと言ったほうが適切である。

表5　工数の計画値に対する実績値の回帰分析

開発種別	標本数	対数変換後の回帰分析			
		寄与率	傾き	切片	標準誤差
新規開発	588	0.906	0.990	0.089	0.200
改修・保守	393	0.892	0.961	0.180	0.221
再開発	68	0.764	0.907	0.449	0.261
拡張	155	0.874	0.901	0.410	0.197

表 6 から次のことが分かる。

- 寄与率はすべての開発種別で 0.9 を超えていて、実績値が計画値から説明される割合が高い。
- 傾きはすべての層で 1 より小さい。すなわち、工期の計画値が大きくなるに従って予実比率が小さくなる（改修･保守は 1% 有意、拡張は 5% 有意）。ただし、再開発の傾きは 0.990 とほぼ 1 に近く、工期の予実比率はプロジェクトの大きさに依存していないと言ったほうが適切である。

表6　工期の計画値に対する実績値の回帰分析

開発種別	標本数	対数変換後の回帰分析			
		寄与率	傾き	切片	標準誤差
新規開発	497	0.925	0.976	0.073	0.079
改修・保守	290	0.953	0.966	0.077	0.071
再開発	51	0.977	0.990	0.035	0.042
拡張	173	0.944	0.960	0.103	0.062

3.3　予実比率の推定値と見積り不足となる確率

対数変換後の回帰分析の結果は一般に必ずしも直観的ではない。そこで予実比率の傾向が理解しやすくなるように、3 つの代表的なプロジェクトとして「典型的なプロジェクト」、「大きなプロジェクト」、「小さなプロジェクト」を選んで、それぞれの予実差を対数逆変換することにより予実比率として示すことにする。「典型的なプロジェクト」とは、計画値の対数変換後の平均値 $\bar{X}$ に対応する大きさのプロジェクト、「大きなプロジェクト」と「小さなプロジェクト」はそれぞれ $\bar{X} \pm 2\,\sigma_X$ に対応する規模のプロジェクトと定義する。ただし、σ_X は対数変換後の計画値の標準偏差である。また、これらのプロジェクトの大きさを表す計画値は、直観的に把握しやすいように丸めたものを用いている。「大きなプロジェクト」と「小さなプロジェクト」は、それぞれ大きいほうから 2.3%、小さいほうから 2.3% に相当するものであり、95.4% のプロジェクトがこの範囲にあると考えて良い。なお、これら 3 つの代表的なプロジェクトは FP 規模、工数、工期ごとに異なる集合から得られたものであることに注意して欲しい。

表 7 ～表 9 に FP 規模、工数、工期それぞれに関する予実比率の推定値及び実績値が計画値よりも大きくな

る、言い換えると見積り不足となるプロジェクトの確率を示す。この値は、対数変換後の計画値 X に対する対数変換後の実績値 Y の推定値を $\hat{Y}$ としたとき、

$$t = \frac{\hat{Y} - (\hat{a}X + \hat{b})}{\sqrt{\left\{1 + \frac{1}{n} + \frac{(X - \bar{X})^2}{ns_{XX}}\right\} V_e}}$$

が自由度 $n - 2$ の t 分布に従うことから、この分布で $\hat{Y} \geq X$ となる確率から求めている。ただし、$\hat{a}$ と $\hat{b}$ は X に対する Y の回帰分析での回帰係数、n はデータ数、$\bar{X}$ は X の平均値、s_{XX} は X の標本分散、V_e は予測誤差の分散の不偏推定値である［古山 2012］。

以下に示す表 7 ～表 9 では、計画値の欄の上段に「大きなプロジェクト」の値、中段に「典型的なプロジェクト」の値、下段に「小さなプロジェクト」の値を示している。

表 7 から FP 規模の見積りに関して次のことが分かる。

- 新規開発における「典型的なプロジェクト」（870FP）での予実比率の推定値は 1.09 である。実績値が計画値を上回る確率は 0.64 で、おおむね 3 件に 2 件のプロジェクトは実績値が計画値より大きくなると推定される。
- 新規開発の「小さなプロジェクト」から「大きなプロジェクト」までの範囲にある FP 規模（120 ～ 6,400FP）では、実績値が計画値を上回る確率が 0.5 を超えていることから、［IPA2010］で既に指摘されているように新規開発プロジェクトでは常に計画値よりも実績値が大きくなる（過小見積りする）傾向にあると言える。
- 改修・保守の「典型的なプロジェクト」（320FP）の予実比率の推定値は 0.97、実績値が計画値を上回る確率は 0.47 であり、平均的には実績値はほぼ計画値通りと言える。
- 改修・保守では「典型的なプロジェクト」より大きい（320FP を超える）プロジェクトでは、実績値が計画値を上回るプロジェクトは半分以下であり（過大見積りの傾向が強く）、とくに 3,000FP 程度の「大きなプロジェクト」では 1/4 程度（27%）しかない。すなわち、「大きなプロジェクト」では 3/4 程度のプロジェクトが過大見積りをしている。逆に「小さなプロジェクト」では 2/3 を超える（69%）プロジェクトで実績値が計画値を上回る。

表7　FP規模の予実比率と見積り不足となる確率

開発種別	計画値（FP）	予実比率（推定値）	実績値＞計画値となる確率
新規開発	6,400	1.15	0.73
	870	1.09	0.64
	120	1.03	0.55
改修・保守	3,000	0.79	0.27
	320	0.97	0.47
	30	1.21	0.69

表 8 から工数の見積りに関して次のことが分かる。

- 「典型的なプロジェクト」の工数の予実比率の推定値は開発種別によらず 1 を超えている（過小見積りを行っている）。
- 「小さなプロジェクト」ではその傾向がより顕著であり、実績値が計画値を上回る確率は新規開発で 0.62（約 3/5）、改修・保守で 0.65（約 2/3）、再開発で 0.75（3/4）、拡張で 0.78（約 4/5）となっている。
- 「大きなプロジェクト」の予実比率の推定値は、「新規開発」以外は 1 より小さくなる。とくに拡張プロジェクトでは実績値が計画値を上回る確率は 0.35 となり、2/3 が過大見積りの傾向を取ることが分かる。
- 新規開発の「小さなプロジェクト」から「大きなプロジェクト」までの範囲にある工数（490 ～ 163,000 人時）

表8　工数の予実比率と見積り不足となる確率

開発種別	計画値（人時）	予実比率（推定値）	実績値＞計画値となる確率
新規開発	163,000	1.09	0.57
	9,000	1.12	0.60
	490	1.15	0.62
改修・保守	104,000	0.96	0.47
	5,000	1.08	0.56
	240	1.22	0.65
再開発	97,000	0.96	0.48
	8,900	1.20	0.62
	820	1.50	0.75
拡張	83,000	0.84	0.35
	5,900	1.09	0.58
	410	1.42	0.78

では、実績値が計画値を上回る確率が 0.5 を超えていることから、新規開発プロジェクトでは常に計画値よりも実績値が大きくなる（過小見積りする）傾向にあると言える。

表 9 から工期の見積りに関して次のことが分かる。

- 「典型的なプロジェクト」の予実比率の推定値は、開発種別によらず 1 を超えている（過小見積りを行っている）。しかし､改修･保守の「大きなプロジェクト」では実績値が計画値を上回る確率が 0.40 となり、過大見積りの傾向が強くなる。
- 新規開発の「小さなプロジェクト」から「大きなプロジェクト」までの範囲にある工期（49 ～ 656 日）で実績値が計画値を上回る確率が 0.5 を超えていることから、新規開発プロジェクトでは常に計画値よりも実績値が大きくなる（過小見積りする）傾向にあると言える。

表9　工期の予実比率と見積り不足となる確率

開発種別	計画値（日）	予実比率（推定値）	実績値＞計画値となる確率
新規開発	656	1.01	0.52
	179	1.04	0.59
	49	1.08	0.66
改修・保守	630	0.96	0.40
	138	1.01	0.53
	30	1.06	0.65
再開発	724	1.01	0.56
	204	1.03	0.61
	58	1.04	0.65
拡張	481	0.99	0.48
	140	1.04	0.61
	41	1.09	0.73

4 考察

4.1 予実比率の推定値と見積り不足となる確率

FP 規模、工数、工期と開発種別の予実差を横断的に俯瞰するために、表 7 ～表 9 の新規開発と改修・保守の数値をまとめて表 10 に再掲する。FP 規模、工数、工期それぞれの間では予実差計算のための集合が異なるが、大きな意味での傾向は読み取ることが出来る。例えば、表 10 の 1 行目については、6,400FP の規模のソフトウェアを 163,000 人時の工数で 656 日かけて開発する仮想的なプロジェクトの場合を示している。予実比率の大きさの順序と見積り不足となる確率の大きさの順序が必ずしも対応していないのは、変数によって標準誤差が異なるためである。

表 10 から次のことが読み取れる。

(1) 全般的傾向

- 新規開発の FP 規模以外はプロジェクトが大きくなるにつれて予実比率が小さくなる。

(2) 新規開発プロジェクト

- 予実比率の推定値はプロジェクトの大きさによらずすべて 1 より大きく、新規開発プロジェクトの計画立案の困難さが伺える。
- FP 規模はプロジェクトが大きくなるにつれて予実比率が大きくなる。新規開発ではプロジェクトが大きくなるに伴い FP 規模の見通しが立ちにくい、あるいは機能追加要求が多くなる傾向が伺える。
- 「大きなプロジェクト」では、工期は平均的にほぼ見積り通りであるが、工数、FP 規模の順に予実比率が大きくなる（実績値が計画値を上回る確率もこの順で大きくなる）。計画値に比べて FP 規模は増えても工数の増加はそこそこで抑え、工期はユーザとの約束通りに何とか守って仕上げるという傾向が伺える。
- 「小さなプロジェクト」では「大きなプロジェクト」と逆の傾向が見られる。すなわち、FP 規模は計画値に近いが工期が若干延び、工数の増加が大きくなり

表10　プロジェクト規模ごとの予実比率（表7～表9のまとめ）

開発種別	プロジェクトの大きさ	予実比率		
		FP 規模	工数	工期
新規開発	「大きい」	1.15 (0.73)	1.09 (0.57)	1.01 (0.52)
	「典型的」	1.09 (0.64)	1.12 (0.60)	1.04 (0.59)
	「小さい」	1.03 (0.55)	1.15 (0.62)	1.08 (0.66)
改修・保守	「大きい」	0.79 (0.27)	0.96 (0.47)	0.96 (0.40)
	「典型的」	0.97 (0.47)	1.08 (0.56)	1.01 (0.53)
	「小さい」	1.21 (0.69)	1.22 (0.65)	1.06 (0.65)

（注）括弧内は実績値が計画値を上回る確率

やすい傾向が伺える。

(3) 改修・保守プロジェクト

- FP 規模、工数、工期とも一般に「大きなプロジェクト」では見積り過多、「典型的なプロジェクト」では見積り通り、「小さなプロジェクト」では見積り不足の傾向となる。

4.2 予実差のばらつきと 50% 信頼区間

標準誤差は推定式から得られた実績値の推定値と実測値の平均的な違い（ばらつき）の程度を表すものである。予実差の推定値が実績値の推定値から計画値を引いたものであるので、実績値の標準誤差と予実差の標準誤差は等しくなる。

表 4 ～表 6 に示された各変数の標準誤差を見ると、開発種別によらず工数の標準誤差が一番大きく、次いで FP 規模の標準誤差、最も小さいのが工期の標準誤差であることが読み取れる。これらの標準誤差を二乗して等分散の検定を行ったところ、同一開発種別内のすべての変数間（例えば新規開発の FP 規模と工数、FP 規模と工期、工数と工期）で統計的に有意（1% 有意）であった。すなわち、推定式からのばらつきは 3 つの変数でそれぞれ異なり、工数、FP 規模、工期の順に小さくなっている。

ただし、計画値の値が、工数は基本設計開始時点、FP 規模は基本設計終了後のものであるため、予実差のばらつきは FP 規模の方が工数よりも小さいとは言い切れない。工期の計画値が基本設計開始時点であると推定されるので、工期の予実差の推定式からのばらつきが最も小さいことになる。

標準誤差の値に 0.6745 を掛けた値が 50% の信頼区間のよい近似となる［古山 2012］。この計算方法を用いて 50% の予実比率の信頼区間を計算した結果を表 11 に示す。表中の値は表 7 ～表 9 の予実比率に対する倍率を表す。例えば、新規開発の「典型的なプロジェクト」の FP 規模に関する 50% 信頼区間のおおよその下限値は 1.09 × 0.86 = 0.94、おおよその上限値は 1.09 × 1.16 = 1.26 となる。

表 11 から分かるように、各標準誤差同士の差に統計的な有意差があっても実際の 50% 信頼区間の大きさにそれほど差が見られるわけではない。

表11　予実比率の50%信頼区間（概算値）

開発種別	FP 規模	工数	工期
新規開発	0.86 ～ 1.16	0.73 ～ 1.37	0.89 ～ 1.13
改良・保守	0.78 ～ 1.28	0.71 ～ 1.41	0.90 ～ 1.12
再開発	－	0.67 ～ 1.50	0.94 ～ 1.07
拡張	－	0.74 ～ 1.36	0.91 ～ 1.10

（注）表中の値は、表 7 ～表 9 の予実比率に対する倍率を表す。

5 まとめ

予実差分析では、ソフトウェアプロジェクトデータの他の分析と同じように統計的に有意な結果は少なかったが、次のような大まかな傾向を明らかにすることが出来た。

- 新規開発における予実比率（＝実績値／計画値）は FP 規模、工数、工期ともプロジェクトの大きさによらずすべて 1 より大きい。とくに「大きなプロジェクト」では工期、工数、FP 規模の順に大きくなる。計画値に比べて FP 規模は増えても工数の増加はそこそこで抑え、工期は何とか計画通りに近いところで仕上げるという傾向が伺える。
- 改修・保守における予実比率は、「大きなプロジェクト」では見積り過多、「小さなプロジェクト」では見積り不足の傾向となる。
- 新規開発の FP 規模以外は、プロジェクトが大きくなるに従って予実比率が低下する。
- 工期の予実差の推定式からのばらつきは FP 規模や工数に比べて小さい。

計画値の策定にあたっては、予実差が大きくならないように過去の経験をフィードバックしているはずである。ここで述べた分析結果は、フィードバックが働いていても、なおかつこのような傾向が見られることを示しているとも言える。

参考文献

［IPA2010］IPA/SEC 監修：ソフトウェア開発データ白書 2010-2011, 2010

［古山 2011］古山恒夫：ソフトウェアプロジェクトデータの量的変数に関する分析の一指針と分析事例，SEC journal, Vol. 7, No. 3, pp. 105-111, 2011

［古山 2012］古山恒夫：エンタープライズ系ソフトウェアプロジェクトにおける層別生産性とその信頼区間，SEC journal, Vol. 8, No.1, pp. 17-24, 2012

非ウォーターフォール型（アジャイル）開発の動向と課題

―IPA/SECにおける4年間の調査・検討から明らかになったこと―

SEC エンタプライズ系プロジェクト
プロジェクトリーダー
山下　博之

SEC エンタプライズ系プロジェクト
研究員
柏木　雅之

市民生活や社会経済活動におけるITサービス及びそれを実現するITシステムへの依存度が高まってきている中で、私たちが安心してITサービスを使い続けられるためには、ITシステムに対し、環境の変化に俊敏に対応することが求められる。非ウォーターフォール型（アジャイル）開発手法は、環境変化への俊敏な対応を可能とするソフトウェア開発手法の一つとして注目されている。IPA/SECでは、約4年間にわたり、アジャイル開発に関する調査・検討を続けてきた。その内容は、経営層の理解促進、技術者に求められるスキルと人材育成、適用領域、契約形態など、多岐に及ぶ。本稿ではその結果について取りまとめる。

1 背景

市民生活や社会経済活動における IT サービス（及びそれを実現する IT システム）への依存度は、IT 導入の目的が、業務効率化によるコスト削減から生活やビジネスの中核を担う戦略的活用へと変遷するのに伴い、量・時間・質の各面において高まってきた。また、市民生活や社会経済活動を取り巻く環境※1 は変化するが、それらの変化も、量的・時間的・質的に拡大傾向にある。

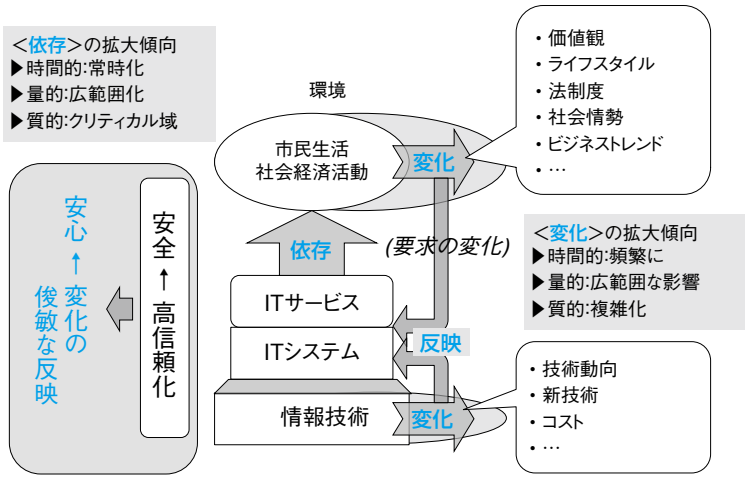


図1　ITサービスの安心のための環境変化対応

このような状況において、従来、IT システムの信頼性を高めて高信頼・安全なサービスを提供することが事業者にとって最も重要な使命であったのが、私たちが IT サービスを安心して継続的に使い続けられるために、環境の変化に対して IT サービスが俊敏に対応することも、信頼性・安全性に加えて重要なこととなってきている（図 1）。

環境の変化に対する IT サービスの俊敏な対応がビジネスや生活に影響を与えたと思われる事例としては、携帯電話各社における学割サービスの導入がある［古川 2011］。A 社は 2010 年 1 月下旬にサービスを発表し、約 20 日後の 2 月上旬に開始した。これに対して競合する B 社は、1 月末に対抗サービスを発表し、5 日後の 2 月初めに開始した。更に C 社は、2 月上旬に対抗サービスを発表したその翌日に開始した。この例では、B 社及び C 社が競合他社の動向という環境変化に俊敏に対応したことにより、一番初めに発表した A 社のサービス開始が最も遅くなってしまったのである。

別の事例としては、携帯電話からスマートフォンへのシフトという、モバイル端末利用動向の変化に対して適切な対応が行われなかったため

に、携帯電話ネットワーク装置の誤動作を誘発し、利用者に多大な影響を与えたトラブルが記憶に新しい。スマートフォンには無通話／操作状態でも一定量の通信を行うという特性があり、あるとき、全体の通信量が急増してネットワークの許容限界を超えてしまったのである。

また、私たちの生活に大きな影響を及ぼし得るケースとしては、諸手当の改定法案の国会審議の遅れにより確定時期が前年度末となり、IT システムの対応が間に合わず、新制度による手当の交付開始が新年度から夏頃にまでずれ込む、といったようなことも想定される。

経済産業省の報告書［METI 2011］によれば、環境の変化に適切に対応するためには、「柔軟化」が求められるという。そして、柔軟化に対する次の 4 つの要求特性を示している。

- 予測性…可変要素（外部／内部環境）が将来変化をする予兆を事前に捉えること。
- 拡張性…既存のリソース（人、モノ、カネ、情報等）に将来の環境変化を想定した余裕を持たせておくこと。
- 迅速性…起きた変化／起こすべき変化に対して、すぐに対応出来ること。
- 適用性…これまでと違った環境、シチュエーションに、うまく対応出来ること。

柔軟化の対象は、必ずしも IT システムだけとは限らず、組織や業務プロセスなど、様々にあり得る。ただし、IT（情報通信技術）が生活やビジネスの中核を担う今日では、IT システムに帰着することが多い。

環境の変化は、IT システムに対しては、要求の変化として現れる（技術の進展は、制約の変化として現れるが、ここでは要求に含めて取り扱う）。要求の変化を想定し、IT システムに対しては、システム自身の拡張性と俊敏な構築・運用体制が求められることになる。これに応え、IT システムが要求の変化に柔軟に対応出来るためには、柔軟なアーキテクチャの採用と共に、俊敏なシステム構築及び運用の観点からは、とくにソフトウェアについて、次の 3 種の方策が考えられる。

a. 非ウォーターフォール型開発（アジャイル開発）
b. クラウドコンピューティング
c. 自動コード生成／ビジネスルールマネジメントシステム（BRMS）

本稿では、上記のうち「a. 非ウォーターフォール型開発（アジャイル開発）」に焦点を当てる。

2 アジャイル開発とは

IPA/SEC では、ウォーターフォール型以外の開発手法を「非ウォーターフォール型開発」と呼んでいる。その代表として、「アジャイル開発」について調査・検討を行ってきた。一言で「アジャイル開発」といっても、スクラム※2 や XP※3 など、様々な作法がある。ここでは、特定の作法に限定せず、後述の特徴を有する開発手法の総称として「アジャイル開発」という呼称を用いる。

アジャイル開発では、顧客の要求に従って、優先度の高い機能から順に、要求・開発・テスト（・リリース）を短い期間で繰り返しながら、システム全体を構築していく。原則として、事前に開発の詳細な計画は作らず、1 ～ 4 週間という一定の短い周期※4 で要求・開発・テストを繰り返しながら、動作可能なソフトを作り上げる。プロセスの観点による、ウォーターフォール型開発との比較を図 2 に示す。

このようなアジャイル開発は、次の特徴を有する。

- 顧客（ユーザ）の参画の度合いが強い。
- 動くソフトウェアを成長させながら作る。
- 反復・漸進型である。

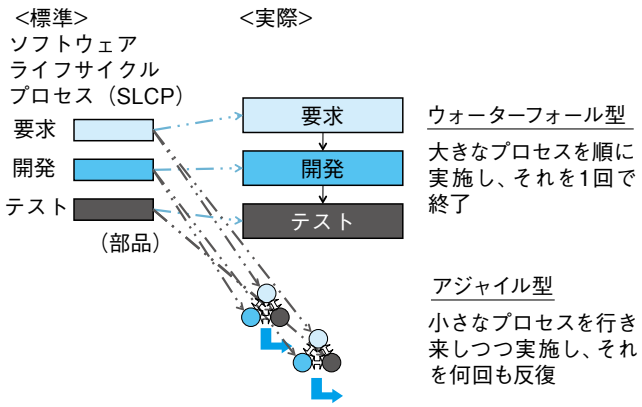


図2　開発プロセスの比較

脚注

※1　変化し得る環境には、ビジネス顧客や競合他社の動向、法制度や社会構造、技術等の外部環境と、事業再編等の内部環境がある。
※2　スクラム：現在では、アジャイル開発手法の中でも最も多く使われている手法。米国におけるある調査によれば、半分以上を占める。
※3　XP：Extreme Programming
※4　この周期を「反復（イテレーション）」と称する。

- 人と人のコミュニケーション、コラボレーションを重視する。
- 開発前の、要求の固定を前提としない。

IPA/SECでは、環境の変化に俊敏に対応可能なソフトウェア開発形態として注目度が増し、内外での成功例の報告も徐々に増えつつあったアジャイル開発について、平成21年度以降、調査・検討を通して整理を試みてきた。当時、アジャイル開発に注目した具体的な背景としては、次の3点が挙げられる。

① ビジネス・ニーズへの適切な対応

- 他社に先駆けた市場投入が必須で、それにより徐々に明確となるニーズを迅速に反映し改善していくことが必要な分野（Web系ビジネス等）の出現。
- 顧客ニーズは最初にすべては把握出来ず、またビジネス環境の激しい変化に伴いニーズも変化するが、この状況に迅速な対応が必要。

②（純粋な）ウォーターフォール型開発における問題点

- 初期には必ずしもすべての要求内容は確定しない。
- 誤要求や要求の誤解が総合テスト段階で判明すると、多大に影響。
- 開発途中で要求が変更されると、対応が非常に困難。

③ ソフトウェア産業構造（多重下請構造）上の課題

- 開発者（とくに若者）の参画意識・達成感が低い。

上記3点に加えて最近では、次の傾向が顕著である。

ⅰ）ビジネス環境の変化の俊敏さへの対応要求の増大
ⅱ）グローバル化の拡大
ⅲ）ウォーターフォール型開発に適合しにくいケースの増大

例えば、ある携帯電話会社の社長は最近、「まずは七分でよし。利用者のお叱りを受けながら100%に磨き上げていく。」というように、経営のスピード感重視を掲げている。また、「従来は100%にしてから世に出していた。現在はエンドユーザとの共同作業により、よいものにしていく。」といったβ版文化の普及が指摘されている。更には、ユーザ企業を対象とする最新の調査［JUAS2012］によれば、システム企画時に重視した事項として、品質が29%、コストが24%に対して、納期が47%という結果が得られている。

これらの背景に応え得るソフトウェア開発手法の一つが、アジャイル開発である。そして、いまやアジャイル開発は世界の主流となりつつあり、国内でもアジャイル開発に取り組む企業が増大傾向にある。しかしながら、あらゆるソフトウェア開発に対して、あるいはすべての組織において、アジャイル開発手法が単純に適用出来るとは限らない。

次章以降では、IPA/SECにおけるアジャイル開発に関する調査・検討の結果について、そのポイントを述べる。詳細については、公開している各報告書［報告書群］を参照願いたい。

3 日本におけるアジャイル開発の状況と課題

平成21年度に実施した調査において、我が国におけるアジャイル開発の事例22件を収集し、分析・整理した。その主な内容と明らかになった課題は、次の通りである。

［日本におけるアジャイル開発の現状］

- 開発チーム：約8割が8名以下。
- 開発期間：半数は2カ月～4カ月。
- アジャイル開発のプラクティス※5群の中から取捨選択して用いている。
- 内部開発が多い（自社内で利用するソフトを内製する、もしくは販売するためのパッケージを開発する）。
- 要求の変化への柔軟な対応、市場への投入の迅速化に効果が表れている。
- 大手SI業者でも、トライアルが行われ、社内標準への取り込みが始められている。

［日本におけるアジャイル開発の課題］

- 経営層の意識向上。
- スキルの明確化、人材の育成方法と適正配置。
- 日本におけるアジャイル開発に適した契約のあり方。
- 欧米の競争力（普及要因）の調査。
- 適用領域と適用事例の調査。

4 アジャイル開発普及上の課題の検討

前述の各課題について、アジャイル開発に関する経験の豊富な実務者及び有識者から成る非ウォーターフォール型開発WG（ワーキンググループ）を設置して検討すると共に、関連調査を行った。

4.1　アジャイル開発のプロセスモデル

平成 21 年度の調査により収集した国内におけるアジャイル開発の適用事例に基づき、まず、アジャイル開発のプロセスモデルを次の 3 種に整理した（図 3）。

モデル 1：企画終了後、要求・開発・テストの反復を繰り返しながら、適宜リリースする基本的な形態。

モデル 2：企画終了後、反復に入る前に、要求・アーキテクチャ設計・基盤開発をきちんと完了する、モデル 1 の拡張形態。基盤・共通部といくつかの機能部とから構成されるソフトウェアにおいて、まず、基盤・共通部をウォーターフォール型開発などにより完成させた後、機能部群についてアジャイル開発を行うもの。

モデル 3：何回かの反復の後、リリースする直前に重点的なテストを実施する、モデル 1 の拡張形態。顧客やビジネスの特徴から、とくに高い品質が求められたり、品質がクリティカルであったりする場合に、リリース前に一層の品質確保を行うもの。

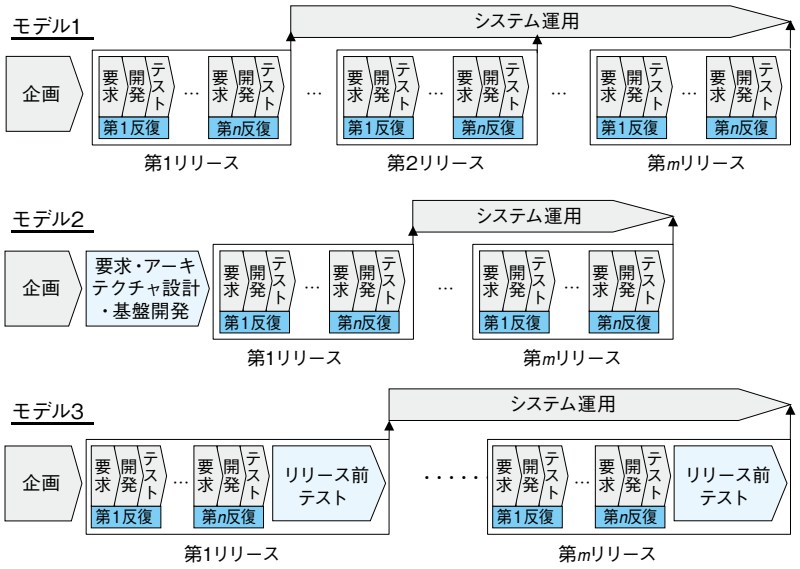


図3　アジャイル開発のプロセスモデル

4.2　アジャイル開発採用における経営層の理解

(1) 顧客（ユーザ）経営層

経営層は、ビジネス環境が激しく変化する現状において、IT システムに関し、従来のように情報システム部門に任せきりでは適切に対応出来ない。開発形態にも深く関与する必要がある。具体的には、次の責任を有する。

- 情報システムに関する理解の増進
- 迅速かつ適切な意思決定
- 関係部門との経営上の綿密な調整

また、アジャイル開発を進めるに際しては、経営層・情報システム部門共に、次の点に留意する必要がある。

- アジャイル開発の採用を決断した時点で、顧客がチームの一員として参画し、主体的に開発に関わらざるを得ないということに十分な理解と覚悟を持つ。
- アジャイル開発を誤解し、「最初にすべての要件を決めなくても良く、途中で適宜明確にしていけば良い」という安易な心構えで始めると、途中で破たんすることが多い。
- 開発を外注する場合には、ユーザ／ベンダ間の信頼感の醸成が、円滑に開発を進めるために重要。

(2) ベンダ経営層

俊敏な開発の実績を武器に受注を狙う海外勢などに対抗するには、自ら俊敏な開発を実施出来る体制作りに取り組むと共に、その結果を顧客に売り込む必要がある。

4.3　アジャイル開発に求められるスキル

アジャイル開発手法の代表であるスクラムにおいては、開発する製品の優先付けされた機能要求リスト「プロダクトバックログ」の中から、“プロダクトオーナー”が 1 つの反復で開発する分を取り出し、開発チームに指示する。開発チームは、“スクラムマスター”の管理・支援のもと、製品に要求機能を追加実装し、リリース可能な状態に作り上げる。その後、“プロダクトオーナー”

脚注

※5　アジャイル開発を実践する活動項目を“プラクティス”と呼ぶ。朝会、リファクタリング、継続的インテグレーション等、50 以上あるが、一つのプロジェクトですべてを使用するわけではない。

はリリース判定を行うと共に、次の周期の開発機能を指定し、このような反復を繰り返す。

従って、ユーザ参画の度合いが強いアジャイル開発において特徴的な、ユーザ側に求められるスキルは、投資対効果を最大とするためにコアとなる機能を見定め、定期的なサイクルで優先度を考慮しながら開発プロジェクトの運営を指揮していく“プロダクトオーナー”としての能力ということになる。

また、アジャイル開発の開発者側にとって重要なスキルは、次の通りである。

① “スクラムマスター”として、開発チームの自律的・協働的な作業を支援し、アジャイル開発の進め方を踏襲させるためのファシリテーションスキル。

② チームメンバとして、反復活動の中で、実際に動くものを作りながら、小規模に、かつトータルにプロジェクトのアウトプットを積み重ねていくスキル。

③ 設計、コーディング、テストを一貫して実施出来る、マルチタレントなスキル。

4.4　アジャイル開発のための人材育成と動機付け

(1) 人材育成

前節で示したスキルを身に付けるためにプラクティスを習得するわけであるが、そのための人材育成においては、次のようなアジャイル開発の実情を考慮する。

- 一つのプロジェクトですべてのプラクティスを使うわけではない。
- 各プラクティスに厳格な規範はない。
- 様々な方法論・数あるプラクティスから、プロジェクトや組織に適したものを取捨選択し、カスタマイズすることが必要。

すなわち、平時における一通りのプラクティスを理解するための施策と、プロジェクト参画時における使用プラクティスの深い習得のための施策とを使い分ける。

ただし、すべてのプラクティスを完全に身に付けるより、価値に従って行動する習慣を確実に身につけることが重要であり、その結果として適切なカスタマイズに結びつく。

なお、人材育成に際しては、次の点に留意する。

- 自社システムの開発から、アジャイル開発の OJT を行う。
- 導入初期の実プロジェクトでは、コンサルタント（アジャイル・コーチ等）の支援を得る。
- 個人プラクティスは普段の業務中に、チーム・プラクティスは組織的な研修などで、それぞれ習得する。

とはいえ、アジャイル開発に向いている技術者とウォーターフォール型開発に向いている技術者とが存在することは確かであり、人材の見極めと適切な配置が、個人と組織の双方にとって重要である。

(2) モチベーション

米国の著名なビジネスジャーナリスト・作家であるダニエル・ピンク［DANIEL］によれば、報酬のインセンティブは、視野を狭め、心を集中させることから、単純な仕事では効果があるが、（ソフトウェア開発のような）そうでない創造的な仕事では逆効果であるという。そして、成果を高めるのは、内的な動機付けに基づくアプローチ、すなわち、重要だからやる、好きだからやる、面白いからやる、何か重要なことの一部を担っているからやる、というものであるとのこと。更に、仕事において重要な要素は次の 3 つと述べている。

- **自主性**…自分の人生の方向は自分で決めたい
- **成　長**…何か大切なことについて上達したい
- **目　的**…自身よりも大きな何かのためにやりたい

これらをアジャイル開発に照らすと、「自主性→ある程度の裁量」、「成長→スキルアップになる」、「目的→製品の完成ではなく顧客の“価値”を高める」、ということになる。

このような動機付けが、アジャイル開発に携わる技術者にとって重要であるといわれている。

4.5　アジャイル開発に適した契約形態

日本におけるソフトウェア開発では、ユーザ／ベンダ間で受発注契約を締結することが多い。アジャイル開発には、ウォーターフォール型開発と比較し、次の特徴がある。

① ユーザとベンダの緊密な協力体制が必須。

- 相手方の問い合わせへの迅速な応答。
- 担当作業の迅速な実施。
- ユーザ／ベンダ間の責任分担が不明確になりがち。

② ユーザ要求の詳細が契約時点では未確定。

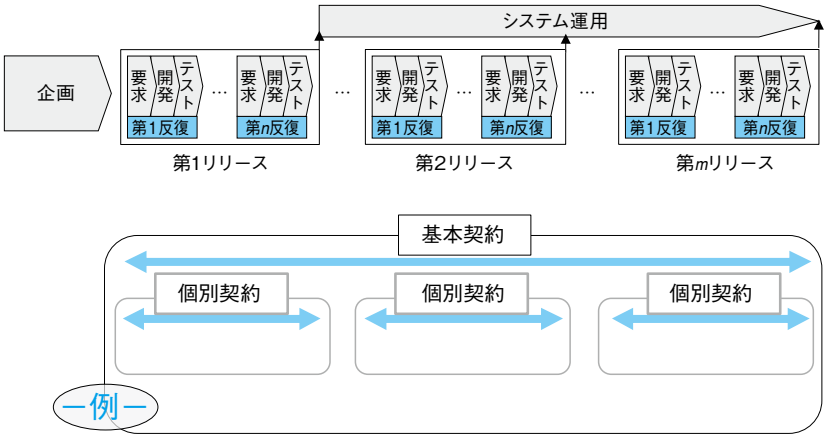


図4　基本／個別契約モデルのイメージ

- 何を作るか決まっていない（成果物未定）。
- 性能・品質などが不明確。
- 工数見積りが困難（コスト未定）。

③ 開発途中でのユーザ要求の変化を柔軟に受け入れる。

- 決定した事項も変更されることがある。

すなわち、そもそも「契約」とは、合意内容を固定して、当事者を法的に拘束するものであるのに対し、アジャイル開発は、変化に対応すべく、合意内容の変更を柔軟に認め、当事者をなるべく拘束しないという特徴を有することから、「契約」とは相容れないことになる。

従って、開発内容が決まっていない段階で、開発プロジェクト全体につき、一つの請負契約を結ぶのは適切ではない（何をいくらで完成させるか不明なため）。他方、開発プロジェクト全体を準委任契約にすることは、ベンダが完成義務を負わない点で、ユーザ側に不安がある（たとえ成果物が完成しなくても、ユーザは対価を支払う必要があるため）。また、アジャイル開発の特徴であるユーザとベンダの協働関係を、契約に取り入れる必要がある。

このような状況に対し、日本におけるアジャイル開発にふさわしい契約モデルとして、次の２つを提案した。

- 基本／個別契約モデル：プロジェクト全体に共通する事項につき、基本契約を締結する。また、小さな機能単位ごとに、開発対象と費用がある程度確定したタイミングで個別契約（請負／準委任）を順次締結する（図4）。
- 組合モデル：ユーザとベンダが共同でジョイント・ベンチャーとしての組合を組成し、協力してシステム開発を企画・実施する。開発された成果から得られた収益は、ベンダとユーザに分配される。

なお、ユーザ側での労働者派遣契約に基づく開発チーム構成には、契約上の問題はとくにない。従って、最近高まる傾向にある内製には容易に採用出来る。

4.6　アジャイル開発の普及に向けた課題と解決方法

ソフトウェア開発プロジェクト、IT人材の状況、IT人材育成の３点について、米国・英国・デンマーク・中国・ブラジルの５カ国と日本とを比較調査し、欧米の競争力の源となっているアジャイル開発の普及要因を、駆動要因と土壌に分けて分析した。その結果、次の事項が明らかとなった。

① 最も有名なスクラムに関する資格者（取得試験は英語）は、米国の取得者が75,000人以上と群を抜いて多く、ついで13,000人強の英国が多い。日本は500人未満と極めて少ない（図5）。

② 米国では、ソフトウェアに対する投資において、外注は約1/3を占めるに過ぎない。更に、他国に比べて多くのIT技術者がユーザ企業に所属している（図6）。これを反映し、米国のプロジェクトの形態の特徴としては、37%が内製である。

③ デンマークでは、政府が発注するソフトウェア開発

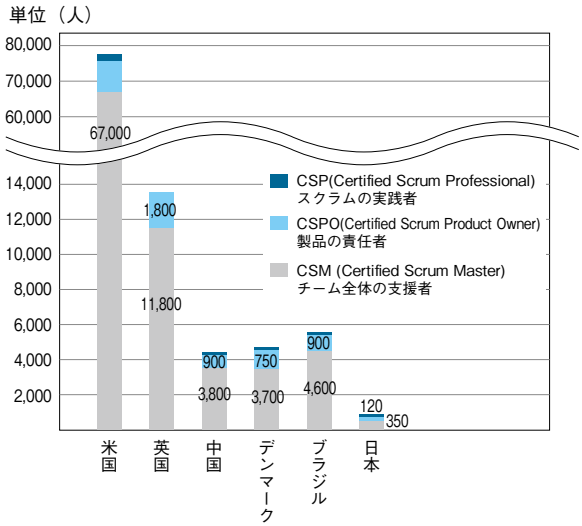


図5　各国のスクラムマスターなどの人数（2012年3月現在）

をアジャイル開発で実施することを推奨している。

④ ブラジルでは、実践した結果、顧客のビジネスの成功率が高いことが要因となり、アジャイル開発が普及している。また､時差が少なく､リアルタイムコミュニケーションが可能であることから、北米からの受託開発が伸びている。

⑤ 米国では、恒常的にIT関連職の給与が高く、人気も高い。中国、ブラジルでは、IT関連職の人材が不足しているため、IT関連職の給与が高く、人気も高い。これらの国々では、IT関連職の人気が高いため、優秀な人材が集まる。

以上の結果を受け、SECではいくつかの施策について現在検討中である。そのうち、プラクティス事例に関するリファレンスガイドの公開を、今年度末に予定している。

5 アジャイル開発の適用領域

すべてのソフトウェア開発にアジャイル開発手法を適用すべきという訳ではない。開発対象の特徴や組織プロジェクトの状況に応じて、適用すべきか適用すべきでないかを判断する必要がある。

(1) 当初の適用領域

アジャイル開発が得意とし、現在その適用により効果を挙げている領域は次の通りである。

① ビジネス要求が変化する領域

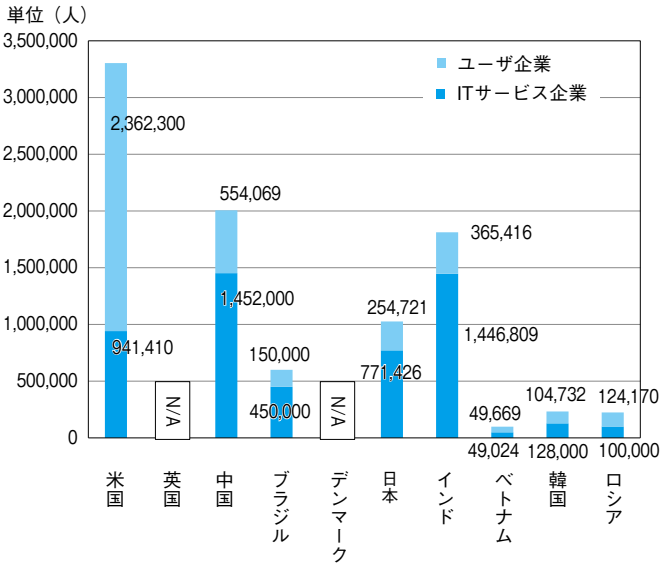


図6　IT技術者の所属先の国別比較

- 要求の変化が激しく、あらかじめ要求が固定出来ない領域。

② リスクの高い領域

- 不確実な市場を対象としたビジネス領域（市場リスク）。
- 技術的な難易度が高い開発領域（技術リスク）。

③ 市場競争領域

- 他社に先駆けた製品・サービスの市場投入が命題であり、TTM（Time to Market）の短縮が優先となる領域（Webサービス、パッケージ開発、新製品開発）。

(2) 当初の試行領域

一方、アジャイル開発による経験が十分には蓄積されておらず、現在、チャレンジと創意工夫が求められている領域もある。

① 大規模開発

コミュニケーション・コラボレーションを重視するアジャイル開発では、開発者が10人程度を超えると、システム分割、チーム分割が必要となる。その分割方法、及び分割されたチーム間のコミュニケーションが課題である。

② 分散拠点（オフショアを含む）開発

①と同様の理由により、開発拠点が分散し、更に時差によって分断される場合のコミュニケーション手法、また、それをサポートするツールが必要である。

③ 組織（会社）間をまたぐ開発チームによる開発

①②と同様の理由により、共通のビジネスゴールを持ったチームを組むことが難しい。

④ 組込みシステム開発

一般に、組込みシステムではリリース後のソフトウェア修正が極めて困難であり、頻繁なリリースを繰り返すアジャイル開発の採用には工夫が必要である。

(3) 中・大規模開発への適用事例

近年、国内での中・大規模開発へのアジャイル開発手法の適用事例が増えてきている。少人数での開発を前提としたアジャイル開発に対し、人数が増えた場合にどのような問題が発生し、その解決のためにどのような工夫がされているかについて、10件の調査事例をもとにまとめた（表1）。その主なものを以下に説明する。

●チーム間ローテーション

チーム間の知識伝播促進のため、メンバのローテーションを行う。一時的に開発速度は落ちるが、各チーム

の知識を効率よく伝播することが出来、技術者のマルチタレント化が高まる。

●段階的朝会

複数チーム間では朝会を段階的に実施する。

チーム→全体（→チーム）

●漸進的な展開

新しい施策は、一度にすべてのチームに広げるのではなく、まずは導入障壁の低いところ、最も必要なところから順次導入し、少しずつ展開する。振り返りで、次はどこに広げていけば良いかを考える。

●コミュニケーション・ツールの活用

TV 会議システム、雑記帳システム（SNS、Blog など）を活用し、コミュニケーションの円滑化を図る。

●アーキテクチャの重視

プロジェクト前半にアーキテクチャを構築する事例が多く、アーキテクチャ専門チームを編成して構築する。

●疎結合な機能分割

疎結合な機能でサブシステム分割を行う。7 割のチームが CI（継続的インテグレーション）を実施している。

●テスト専用フェーズ

プロジェクト後半で専用のテストフェーズを実施する。プログラム開発の反復を停止する事例と、テストのみの反復期間を設ける事例がある。

一方、次のような問題点も明らかになり、更なる工夫が必要である。

●全体計画の把握困難

要求の変化や開発状況に応じて着手する順番や範囲を決めるため、プロジェクト開始時にプロジェクト全体の把握が困難な事例がある。

●ビジネス企画側にボトルネック発生

スクラム導入の結果、ビジネス企画者の決定待ちなどのボトルネックが発生した事例が多い。中には開発者の信頼をなくした事例もある。

●反復リズムとの不適合状態の発生

セキュリティ監査や外部テスト業者、発注者の外部組織や関連組織との関係において、開発の反復リズムと適合せずに問題が発生している事例がある。

6 イノベーションに向けて

環境の変化に俊敏に対応出来、私たちが安心して使い続けられる IT システムの開発手法の一つとして注目されているアジャイル開発手法について、IPA/SEC にお

表1　中･大規模開発への適用時の工夫項目

中・大規模開発特有の工夫		小規模開発と同様だが とくに注意して実施する工夫
■組織体制 ●チーム間ローテーション ■コミュニケーション ●段階的朝会 ●チーム跨ぎの振り返り ■展開 ●漸進的な人数増加 ●漸進的な展開 ●社内勉強会 ■分散拠点開発 ●同一拠点から分散へ ●TV会議 ■アーキテクチャ ●組織の共通基盤アーキテクチャの利用 ●アーキテクチャについての教育 ■システム分割／インテグレーション ●同じリズム	■品質 ●第三者テスト ■部分適用 ●必要な部分のみ適用 ●疎結合なチーム ●工程の見える化 小規模開発とは逆のアプローチを取る工夫 ■アーキテクチャ ●最初のアーキテクチャ構築 ●アーキテクチャ専門チーム ●運用保守チーム ■品質 ●テスト･フェーズ	■コミュニケーション ●完全透明性 ■展開 ●パイロット導入 ●認定研修･コンサルタントの利用 ■分散拠点開発 ●チケットシステム ●リアルタイムチャット ■アーキテクチャ ●アーキテクチャの改善 ■システム分割／インテグレーション ●疎結合で分割 ●早期からのインテグレーション ●継続的インテグレーション ■品質 ●重視するビジネス価値 ●ビジネス価値の変化 ●タイムボックス優先の品質 ●自動単体テスト

ける約４年間の調査・検討の結果をまとめた。

普通のビルや橋を建設する場合には、作るものも使用する技術も明確である。しかし、ソフトウェアの中には、計画時にビジネス上やシステム上の課題が未解決で、開発開始後も仕様変更の可能性が大きいものが少なくない。このような場合には、最初から綿密な計画を立てるより、少し試して、その結果に基づいて次のステップを進めるというやり方のほうが、失敗のリスクが小さい。このような考え方が、アジャイル開発を推進する理由の一つとなっている。

アジャイル開発手法を導入する際のポイントは、次のようにまとめられる。

① 適切な開発手法の選択

開発対象の特徴や開発組織の置かれた環境などを加味しつつ、適切な開発手法を選択するか新たに考案する。

② プラクティスの活用

各プロジェクト・組織（企業）で、自らの開発に合った方法の採用に向け、プラクティスを選択あるいは参考にして利用する。必要に応じ、カスタマイズする。

③ 開発手法に対する正しい理解の促進

プラクティスの意図やプラクティスが提唱されている背景についても理解を深める。

ただし、「銀の弾丸は無い」ということを肝に銘じておく必要がある。すなわち、実践現場でのたゆまない問題解決の積み重ねを続けることが最も重要である。

ウォーターフォール型開発が「プロセス」重視の手法であるのに対し、アジャイル開発は「人」重視の手法であるといわれる［CONBOY2011］。別の観点からは、ウォーターフォール型は開発が失敗しないための手法であり、アジャイル開発はビジネスが成功するための手法であるといえなくもない。すなわち、両手法は全く文化が異なるといえる。比較的早くからアジャイル開発手法を導入してきた大手企業も、「多くの組織、チーム、個人にとって、アジャイル開発プロセスへの転換は“挑戦的”である。それは、ある種の文化的変革を必要とするからだ。」と述べている［IBM］。

しかしながら、この転換に挑戦する企業も現れている。例えば、ある大手の Web 系企業では、高品質を追求する日本的なソフトウェア開発手法と米国流の俊敏なアジャイル開発手法との『いいとこ取り』をし、高品質はそのままに開発速度を速める工夫を行っている［日経 SYSTEMS2012］。また、ウォーターフォール型開発とアジャイル開発とのハイブリッド形態の採用も増加傾向にある［VERSIONONE2011］。

各開発手法を得意とする技術者が協働し、両文化をうまく融合させることにより、ソフトウェア開発のイノベーションを期待したい。

最後に、非ウォーターフォール型開発 WG 委員をはじめとする、アジャイル開発に関する調査・検討にご協力いただいた方々に深く感謝する。

参考文献

［CONBOY2011］Kieran Conboy, Sharon Coyle, Xiaofeng Wang, Minna Pikkarainen：“People Over Process: Key People Challenges in Agile Development”, IEEE Software, July 2010
［DANIEL］ダニエル・ピンク：やる気に関する驚きの科学，2009,
http://www.ted.com/talks/lang/ja/dan_pink_on_motivation.html
［IBM］Agile transformation,
http://www.ibm.com/smarterplanet/us/en/business_analytics/article/agiledevelopment.html
［JUAS2012］一般社団法人 日本情報システム・ユーザー協会 (JUAS)：ソフトウェアメトリックス調査 2012，2012
［METI2011］社団法人日本情報システム・ユーザー協会：平成 22 年度経済産業省委託調査「IT 経営普及促進に向けた調査研究」報告書，p.76, 2011 年２月,
http://www.meti.go.jp/meti_lib/report/2011fy/0022948.pdf
［VERSIONONE2011］VERSIONONE：State of Agile Development Survey Results，2011
［日経 SYSTEMS2012］特集１　ここがヘンだよ　日本のシステム開発 - 日本と米国のいいとこ取り，日経 SYSTEMS，2012 年８月号，2012
［古川 2011］古川昌幸：「Gen-Y」世代が主力ユーザーとなる時の IT，野村総研，知的資産創造，2011 年３月号，pp.32-45，2011,
http://www.nri.co.jp/opinion/chitekishisan/2011/pdf/cs20110305.pdf
［報告書群］
・H21 年度版報告書,
http://sec.ipa.go.jp/reports/20100330a.html
・H22 年度版報告書,
http://sec.ipa.go.jp/reports/20110407.html
・H23 年度版報告書,
http://sec.ipa.go.jp/reports/20120326.html
・ウォーターフォール型開発の普及要因と適用領域の拡大に関する調査報告書（国内の中・大規模プロジェクト事例),
http://sec.ipa.go.jp/reports/20120328.html
・非ウォーターフォール型開発の普及要因と適用領域の拡大に関する調査報告書（非ウォーターフォール型開発の海外における普及要因編),
http://sec.ipa.go.jp/reports/20120611.html

ソフトウェアの品質説明力強化に関する実験を実施

SEC 統合系プロジェクト
研究員
中谷　浩康

SEC 統合系プロジェクト
研究員
室　修治

1 ソフトウェアの品質説明力とその強化のための SEC の取り組み

日本の製造業は世界に冠たる技術力・人材力によって提供する製品・サービスにおいて絶対的な品質、信頼性を誇ってきた。消費者は購入する製品について不具合があるなどとは思いもしなかったはずである。技術は更に発展し製品に載せることの出来る機能の集積度を上げることになる。結果として製品は、より便利で高機能なものとなってきている。この技術の発展を支えてきた大きな要素はコンピュータ技術である。

コンピュータに目的とする仕事をさせるためにはソフトウェアが必要となる。製品を構成する技術としてコンピュータ技術の占める割合が大きくなっている現在、消費者の製品に対する期待はソフトウェアに対する期待と言い換えても良いほどになっている。

ソフトウェア規模が小さければ内包する様々な問題も何とかクリアし、使用上問題のない品質とすることが出来た。また、コンピュータ技術が産業として定着して以来の不断のソフトウェア開発技術、生産性向上技術等々の進展があったのは言を待たない。しかし、複雑で高度な仕事をさせるためにソフトウェアの規模が巨大となった今、従来のような高品質を維持することがむずかしくなってきている。ソフトウェアの品質向上への取り組みは今後とも引き続き重要な課題となっている。

このような状況の中、2009 年に発生した北米における日本メーカ製自動車による死亡事故の発生と、その原因がソフトウェアではないかと問われた事例は、我々にソフトウェア搭載製品についての新たな課題を突きつけたものであった。製品開発における絶対的品質というものを高度に達成してきた日本企業は、成熟した機械系技術ではない比較的新しい技術であるコンピュータの技術でもその品質達成指標を徒に下げることなく様々な問題も克服し、難しい要素がある中でも製品化をしてきた。本件についてもメーカは、問題のないことを表明したが米議会を納得させるに至らず、消費者側や第三者の検証を受けることとなった。結局この第三者での検証においてもメーカが主張した通り不具合の証拠は発見されなかった。つまり品質に問題がなかったとしても起こり得るこのような事故に対して、製品が問題のない品質レベルにあるということの説明力すなわち相手側を納得させる技術が欠如していたということが強く印象付けられたのである。

ここまでを安心・安全なソフトウェアに向けての 2 つの視点として図 1 に整理する。

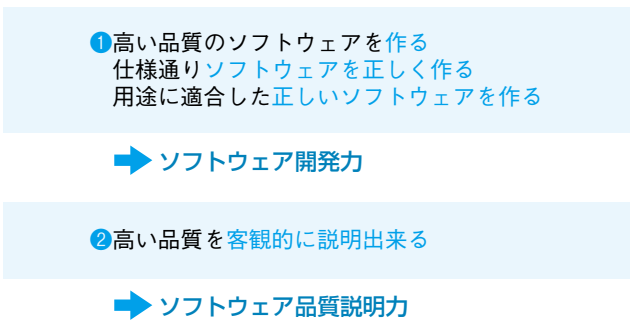


図1　安心･安全なソフトウェアに向けた2つの視点

このような背景の中、IPA/SEC ではソフトウェアの品質の説明力の重要性を認識し、その強化のための検討を 2010 年度より開始した。

品質説明力として必要とされる原則を図 2 に示す。

工業製品の場合その説明力とは図 2、図 3 にあるように明確であり、網羅的であり、根拠を持った論理的なものでなければならない。ソフトウェアの開発を経験した方であればよく理解出来るものと思われるが、曖昧で根拠のない主張では到底相手を納得させられるものではない。ソフトウェア及びソフトウェアの品質とは、ソフトウェア開発活動全般の成果

■品質目標が明確であること
■品質目標を達成するために何をしたらよいか(ロジック)が明確化されていること
■ロジックを実施した証拠(エビデンス)が存在すること
■品質目標、ロジック、エビデンスが相互にトレースできること

作った側が単に主張しても通じない
客観的に評価出来なければならない

図2　品質説明力の原則

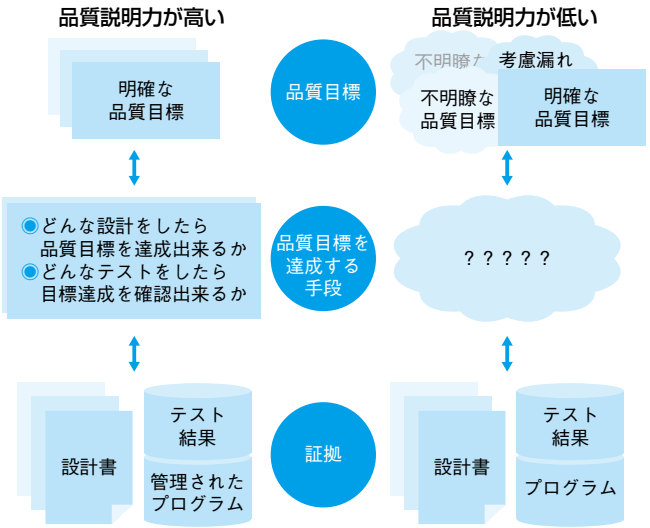


図3　品質説明力の高低

として成り立つものであり、それを説明するためには場合によってはソフトウェア開発活動全般について説明出来なければならないという事態も考えられる。説明には膨大かつ多岐にわたる情報が必要となるため、何らかの効果的な仕掛けも必要となる。また北米での件をみると、説明の客観性も重要である。このような要求に基づいて検討したものがソフトウェアの品質を説明するためのフレームワークである(図4)。

図5の項目について、開発側がこのような仕掛けを効率よく実施していくことで、ソフトウェアの品質説明力を強化する取り組みをより一層推進していく必要がある。

このような活動が実際に現場で機能するかどうかについて具体的なモデルを設定して実験を行った。以降にその内容と結果を報告する。

2　実験の内容

この実験はソフトウェアの品質説明力を強化する仕掛けとして検討したフレームワークを参照して実施した。実験では、実際のシステムや製品に対し、既存の品質監査や認証の仕組みを基に、監査で発生する負荷の推定（監査で発生する作業

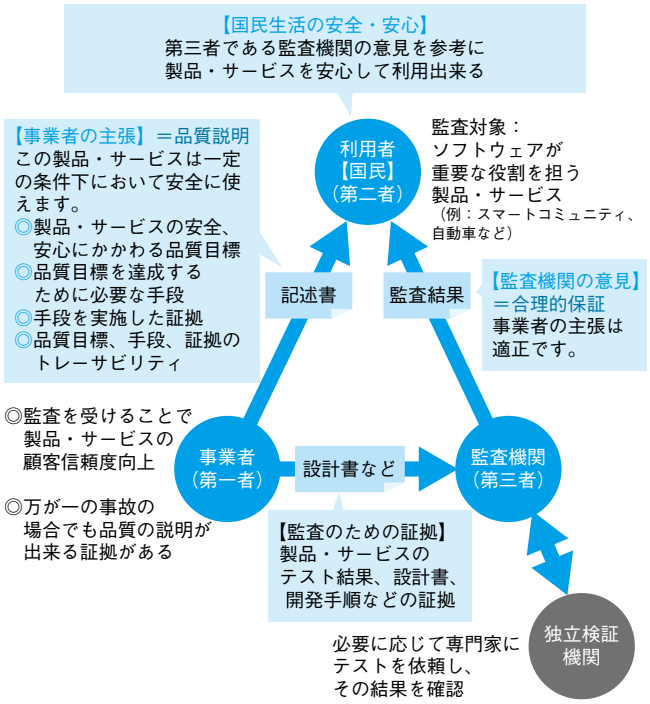


図4　ソフトウェア品質説明力強化のためのフレームワーク

開発プロセス
■開発工程・作業項目・手順の標準化
■開発手順の忠実な実施
■実施状況の証跡

第三者の理解しやすい開発
■形式的仕様記述、モデルベース開発など
■標準への準拠
■ツールの活用　など

図5　ソフトウェア品質説明力を強化するために開発者側が推進すべき事項

の洗い出し、洗い出した作業のコスト評価等）や、監査対象の品質保証の可能性（以下、フィージビリティ）の観点を重視した。実験の実施者は、実験の対象製品、目的、説明力を強化したい品質の内容等の観点で公募を行って決定した。そして12テーマの実験を実施することとした。

独自に制定した枠組みを利用して、ソフトウェアの品質説明を行うことを目的としたこれらの実験の中から本稿ではCSAJ※1で実施した1つの実験を選び、その結果について説明する。なおすべての実験結果は2013年初めにIPA/SECホームページ上に公開予定である。

3　パッケージソフトウェア品質認証度のフィージビリティ評価及び監査制度導入によるコスト評価

国産パッケージソフトウェアの品質は、現時点では業界内での統一的な評価基準はないものの、個別企業の不断の努力

によって保たれている。しかし、グローバル化が進む昨今では「見える化」が重視され、どの品質基準に従い、どのように製品化し、誰が基準を確認したのかを明らかにすることが求められている。製品の競争力強化の観点からも、規格に準じた標準化、品質の見える化、品質説明力強化が必須となってきている。

そこでCSAJでは、パッケージソフトウェアを対象に統一的な品質評価を行うための基準が必要であると考え、ソフトウェア品質に関する国際規格であるISO/IEC25000(SQuaRE※2)シリーズに着目し、シリーズ中においてパッケージソフトウェアを対象としたISO/IEC25051（JIS X 25051）を採用して、具体的な品質評価を行うための基準と手続きを策定した。

この基準に準拠して審査を行い、適合している製品には「パッケージソフトウェア品質認証制度（PSQ認証制度）」に合格した旨の認証を付与する。認証を付与することで利用者や市場に対する説明力が強化されることとなり、国際市場における正当な評価の確立や情報システムの本質的な向上に繋がることが期待される。

今回の実験では勤怠管理用業務ソフトとツールソフト（グループウェア）の2つのパッケージソフトウェアに対し、以下の2つを実施した。

① JIS X 25051（ISO/IEC25051）準拠レベルでのパッケージソフトウェア製品認定を行う認証制度のフィージビリティ評価

② ソフトウェア品質説明力強化のためのフレームワークを本認証制度に導入適用した場合のコスト評価

4 実験で説明力強化を行うソフトウェア品質

JIS X 25051（ISO/IEC25051）を根拠とした具体的な審査基準を策定し、パッケージソフトウェアの品質を担保する(図6)。具体的には、「製品説明」、「ユーザーズマニュアル」、「ソフトウェアの機能・性能」、「試験文書」の間に矛盾がないことを品質要求事項として整理する。

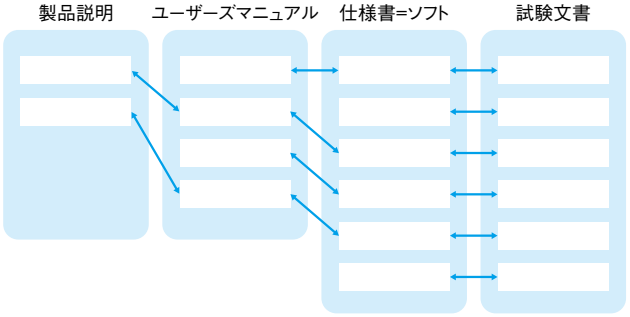


図6　JIS X 25051が示すパッケージソフトウェアの品質概念図

① 製品説明とユーザーズマニュアルの内容の一致

② ユーザーズマニュアルとソフトウェア機能が一致

③ ソフトウェア機能を示す試験文書をソフトウェア品質特性に応じて体系化

JIS X 0129-1が求めるソフトウェアの品質要求事項の中でもパッケージソフトウェアにも適用可能な内部品質、外部品質に共通する品質特性をJIS X 25051（ISO/IEC25051）の要求項目に合わせて具体化する。一部、利用時の品質も対象とするケースがある。

5 監査項目と監査方法の考え方

ソフトウェア品質説明力強化のためのフレームワークで検討されている監査レベルに対応した監査内容に基づき（図7)、「監査レベル1」、「監査レベル2」でのフィージビリティスタディを実施した。

■監査レベル1の定義

重要項目に対する抜取監査（サンプル監査）

PSQ認証制度におけるソフトウェア品質説明力強化のためのフレームワークで定義される「重要項目」を以下のように定義した。

■PSQ認証制度における重要項目

製品説明の記載事項とユーザーズマニュアルの記載事項に共通するソフト機能より抽出

■監査レベル2の定義

全項目に対する抜取監査（サンプル監査）

監査レベル1の重要項目を含む全項目の導出を行う。

監査レベル	監査する審査項目	監査方法	独立検証
4	全項目	網羅検査(全件監査)	必須
3	重要項目	網羅検査(全件監査)	必須
	その他の全項目	抜取監査(サンプル監査)	任意
2	全項目	抜取監査(サンプル監査)	任意
1	重要項目	抜取監査(サンプル監査)	任意
0	対象外	対象外	対象外

図7　監査レベルに対応した監査内容(出典:ソフトウェアの品質説明力強化のための制度フレームワークに関する提案(中間報告案))

脚注

※1　CSAJ：Computer Software Association of Japan，一般社団法人コンピュータソフトウェア協会

※2　SQuaRE：Software product Quality Requirements and Evalution

6 フィージビリティスタディ結果

A）業務ソフト

考察

＜重要項目の特定＞

審査対象となるソフトのカテゴリや機能規模によって、重要項目が異なることが明らかになった。たとえ同カテゴリのソフトであっても機能規模によっては重要項目が異なることは十分想定出来る。

PSQ 認証制度を運営するにあたり、審査工数を低減させるためには審査ノウハウの集積が必要となるだろう（図８）。

＜審査業務用マトリックス作成＞

マトリックスの作成は審査作業として有効であることが分かった。今後の審査作業でも利用することとし、ある一定の項目までは共有化出来る所までノウハウを蓄積させることで効率化が図れると考えられる（図９）。

B）ツールソフト（グループウェア）

考察

＜審査必須の重要項目＞

製品説明、利用者用文書、それぞれに、「重要項目として必須の項目」があると考えられる。例を以下に示す。

1） 製品説明自体の可用性（審査基準 JIS X 25051 該当 5.1.1）
2） 製品説明中の移植性（審査基準 JIS X 25051 該当 5.1.9）
3） 利用者用文書自体に適用される項目（審査基準 JIS X 25051 該当 5.2.1 ～ 5.2.6）

利用者用文書全体にわたるため、項目の粒度は大きいが、審査基準の遵守性を見る上で必須と考えられる。

審査必須の項目があらかじめ選定･合意されるのであれば、審査業務にかかる作業を効率化し、工数を低減することが可能となると考えられる。

こうした知見は審査実務により得られるであろう。

＜項目に対する審査方法＞

審査項目に対して、審査方法が自ずと決まる場合がある。具体的には審査基準 JIS X25051 該当 5.1 製品説明に対する要求事項で記されている通り、以下のいずれかとなる（現地での立ち入り検査は対象外としている）。

1） 製品説明の精査
2） 利用者用文書の精査
3） 製品説明・利用者用文書双方の精査（照合）
4） 試験結果の精査

審査項目・審査基準と審査方法との対応づけがあらかじめなされているのであれば、審査業務を効率化することが可能となると考えられる。＜審査必須の重要項目＞同様、こうした知見は審査実務を行うことによって得ることが出来る。

＜審査基準マトリックス＞

ツールソフト（グループウェア）を対象としたフィージビリティ評価では、以下に記述の通り、

1）製品説明や利用者用文書から、審査基準に照らして重要と考えられる項目を抽出し、

2）その上で改めて審査基準と照合するという手順をとった。実際に審査を行うためには、図８のような表をあらかじめ用意しておき、製品説明や利用者用文書から項目を抽出しながら、随時審査基準と照合するのが望ましい。

7 結果

パッケージソフトウェアを対象とした PSQ 認証制度において、ソフトウェア品質説明力強化のためのフレームワークの定義を適用した場合、監査レベル１が適当であると判断される。

(1) 監査レベル１【重要項目 - 抜取監査（サンプル監査）】が適当な理由

●重要項目の設定

対象ソフトの実務利用シーンを想定した機能を抽出し、重要項目を設定することで、抜取監査を実施するためのサンプリング精度がより向上する。

●認証制度としてのコストバランスが成立

審査工数が適切で、審査精度が高いサンプリングを行うことで認証制度に対して申請者に発生するコストと製品説明力のバランスが取れる。

(2) 監査レベル２【全項目ー抜取監査（サンプル監査）】が適当でない理由

●抜取監査のサンプリング精度が低下

対象ソフトの審査項目全項目をサンプリング対象とした場合、重要項目を設定するケースと比較して抜取監査のサンプリング精度が低下する。審査項目には不具合発生によるユーザーへの影響度等が考慮された（ソフトウェアカテゴリによって異なる）重要とされるソフト機能を抽出することが、重要な観点である。全項目を対象とした抜取監査を実施した場合、この観点から抽出されるべきソフト機能が選ばれない危険性がある。

●コストバランスが不成立

監査レベル１で重要項目を設定するケースより項目数で約 10 倍、審査工数では３倍以上の工数が必要となる。工数増加はコストの増大を招き、申請者負担増となる結果を導く。パッケージソフトウェア企業にとっての製品説明力とコストのバランスが取れた品質認証制度には及ばないと考えられる。

(3) 今後の PSQ 認証制度の課題

●重要項目の設定と抽出に関して、その明確な方法を更に検討することが必要である。

8 まとめ

本来、パッケージソフトウェア産業における品質は自社によるテストが緻密になされている以上、環境を前提に語られるべきである。既に保たれている品質レベルを第三者機関によって証明するために品質認証制度は構築されることが望ましいと考察出来る。

結果、一定の品質の保たれない製品が排除されていくことが期待出来る。

なお監査レベルごとの工数は図 10 の通りとなった。これからも製品にかけることの出来るコストを考慮すると監査レベル 1 が適当であることが判断出来る。

No.	大機能	中機能	中機能	(A)不具合発生可能性(1〜3)※1	(B)不具合発生時の利用者影響度(1〜3)※2	不具合発生リスク(A×B)	備考
1	設定	起動・終了	起動と終了の機能	1	3	3	
2		会社情報	基本設定	1	1	1	
3		部署設定	基本設定	1	1	1	
4		カレンダー設定	基本設定	1	1	1	
5		勤務パラメータ設定	就業規則登録	1	1	1	
6	有休管理	有休付与設定	有休付与に関する修行規則の登録⇒この設定を元に有休日自動算出	1	1	1	
7		社員管理	社員の登録	1	3	3	
8		実績レポート社員設定	オプションツール「実績レポート」と連携するための社員の基本情報データ作成機能	1	2	2	
9		有休付与日数設定	導入時の有休日数を設定⇒ここから自動設定	1	2	2	
10		有休消化日設定	導入時の有休消化日設定⇒ここから自動設定	1	2	2	
11		有休残日設定	導入時の有休残日数の設定⇒ここから自動設定	1	2	2	

図8　重要機能の決定表

監査レベル	PSQ認証制度フィージビリティ	
	監査レベル1	監査レベル2
	重要項目に対する抜取審査	全項目に対する抜取審査
a) 業務ソフト	工数:61.5時間	工数:182.5時間
b)ツールソフト（グループウェア）	工数:49.25時間	工数:48.25時間

図10　監査レベルごとの工数（本実験によるツールソフトの全審査項目数はレベル2においても≒抜取検査項目数となったため、レベル1における重要項目の選定作業分工数が増大するという現象がみられた）

機能		テスト要素		想定テスト項目数 機能性	使用性	完全性	正確性	一貫性	合計	サンプリング結果	試験文書確認定工数
勤務表入力	勤務表の表示	メニューからの起動		2	1	1	1	1	6	1	0.5
		起動時の年月自動表示		2	1	1	1	1	6	1	0.5
	勤務表の設定	西暦入力		2	1	1	1	1	6	1	0.5
		社員選択		2	1	1	1	1	6	1	0.5
	勤務表の機能	表示期間	締日	2	1	1	1	1	6	1	0.5
			支払月	2							
	労働時間数等の自動計算機能	自動での労働時間計算	休日の種類	6	1	1	1	1	10	1	0.5
			休憩時間	5							
			残業時休憩	5							
			深夜残業休憩	7							
			実働時間	4							
			時間外	7							
			休日勤務①	2							
			休日勤務②	2							
			休日勤務③	2							
			深夜勤務	2							
			深夜残業	2							
			不就労時間	3							
			届出事項	14							
		入力のみの項目	始業時刻	2	1	1	1	1	6	1	0.5
			就業時刻	2							
			外出時刻	2							
			外出戻り時刻	2							
			自動計算無効機能	2							
			自動計算を有効に戻す	2							
	月の時間・日数（回数）集計機能	時間集計項目算出ルール	休憩時間	2	1	1	1	1	6	1	0.5
			残業時休憩	2							
			深夜残業休憩	2							
			実働時間	2							
			時間外	6							
			休日勤務①	2							
			休日勤務②	2							
			休日勤務③	2							
			深夜勤務	2							
			深夜残業	2							
			不就労時間	6							
			60H 超過残業	2							
		月の回数集計項目算出ルール	労働日	2	1	1	1	1	6	1	0.5
			有休	2							
			欠勤	2							
			代休	2							
			半休	4							
			特休	2							
			休出	2							
			遅刻	2							
			早退	2							
			病欠	2							
	休日の種類の変更機能	社員に設定された休日設定の表示		6	1	1	1	1	10	2	1
		休日種類の変更		6	1	1	1	1	10	2	1
		休日種類変更時の自動再計算		6	1	1	1	1	10	2	1
	届出事項の設定機能	届出事項の変更（リストから）		4	1	1	1	1	8	1	0.5
		届出事項の変更（ダイアログから）		5	1	1	1	1	9	1	0.5
		届出事項の自動設定可能判定		18	1	1	1	1	22	3	1.5
	時刻の入力支援機能	ピリオドを使って入力する方法		4	1	1	1	1	8	1	0.5
		スペースを使って入力する方法		4	1	1	1	1	8	1	0.5
		数字のみで入力する方法		3	1	1	1	1	7	1	0.5
	勤務表の編集	過去に保存した勤務表データの呼び出し		35	1	1	1	1	39	5	2.5
		該当がない場合の処理		2	1	1	1	1	6	1	0.5
		エクセルからの貼り付け		2	1	1	1	1	6	1	0.5
	勤務表の新規作成	勤務表の新規作成		2	1	1	1	1	6	1	0.5
	勤務表の保存	勤務表の保存		2	1	1	1	1	6	1	0.5
	勤務表の印刷	プレビュー		2	1	1	1	1	6	1	0.5
		印刷		4	1	1	1	1	8	1	0.5
	勤務表の削除	勤務表の削除		2	1	1	1	1	6	1	0.5
	勤務表のタイムカード取り込み	該当する勤務表のタイムカード取り込み		5	1	1	1	1	9	1	0.5
		勤務表への反映		35	1	1	1	1	39	5	2.5
		労働時間などの自動算出		25	1	1	1	1	29	5	2.5
		既に入力してある日の勤務には反映されない		10	1	1	1	1	14	2	1
	タイムカード印刷	プレビュー		2	1	1	1	1	6	1	0.5
		印刷		4	1	1	1	1	8	1	0.5
			合計	327	32	32	32	32	455	60	30

図9　審査業務用マトリクス

CEA-LISTとの国際連携活動について

—CEA-LISTとその研究成果を展開するパートナー企業との意見交換を実施—

SEC
副所長
杉浦　秀明

SEC 統合系プロジェクト
主任
八嶋　俊介

システム統合技術応用研究所（CEA-LIST）は、フランス原子力・代替エネルギー庁（CEA）の1機関として、714名の研究者と技術者（PhD136名含む）を擁し、ソフトウェア・エンジニアリング、エネルギー、交通、スマート・ディジタルシステム等の各分野に関する研究開発を実施しており、産官連携の橋渡しを行うことで、開発した技術や研究の成果を社会に有効移転している。
IPAは、CEA-LISTとの間で2011年に相互協力協定を締結しているが、2012年10月にCEA-LISTを訪問し、今後の相互協力活動についてのディスカッションを行った。ここではCEA-LISTが重点分野としているモデルベースや形式手法などに関する研究の最新の取り組み内容や、民間企業との共同研究、技術移転の枠組みなどパートナー企業との連携状況について、パートナー企業6社でのCEA-LISTの研究成果の展開も含め報告する。

1 CEA-LIST フォントネーオーローズ研究所

パリ中心部から南西7～8kmの場所にあるCEA-LISTのフォントネーオーローズ研究所で、CEA-LISTが経済的・社会的ニーズが高いとして力を入れているAmbient Intelligence & Interactive Systems部門の研究内容について意見交換を行った。この部門の研究テーマは、バーチャルリアリティ、インタラクティブロボット工学、感覚インターフェースなどとなっている。研究対象となっているシステムの中には、実際に動作する試作機もあった。

アームを身体に装着して身体の動きを画面上に投影させるシステムでは、画面上のモデル化された仮想の障害物に当たると、現実にも何かに当たったかのようなリアルな感覚を再現することが出来、将来は遠隔での手術など、医療現場での利用を想定している。また、自由度の高いリンク機構を有するアームを腕に装着し、重量物をわずかな力で持ち上げることや、力を要する緻密な作業の安定性を高めることが出来るシステムでは、器具が身体の動きをサポートすることで、将来は身体の不自由な人でも多様な動きが出来るようになる。更に、人間の五感のうち、コンピュータに再現させることが難しいと言われている「触覚」を実現するシステムでは、等身大ロボットの頭を指で触れた場合、手のひらで触れた場合、指で掻いた場合の3種類の違いを明確に判別することを可能としている。

写真1　フォントネーオーローズ研究所の主要メンバーと

IT融合システム時代に向け、飛躍的な操作性向上技術の1つとなりうるユーザインタフェース技術や、新たな産業分野としてのロボティクス技術に関する取り組みが行われていた。

また同じサイト内のZOEという施設では、フランスで実現した初めての原子力発電の原子炉が当時のまま保存されている。小電力ではあったが、1948年12月15日に初めて発電した際には、フランス大統領だけでなく、世界中から多くの研究者が訪問し、CEA-LISTのソフトウェア・エンジニアリング研究の端緒となった。

2 CEA-LIST ナノ・イノブ・センタ

次にパリ中心部から南西約15km、サクレーに新しく設立されたCEA-LISTの拠点であるナノ・イノブ・センタを訪問した。

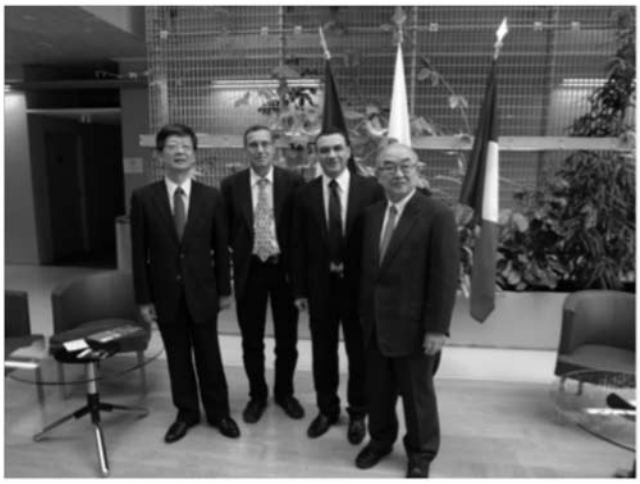


写真２　CEA-LIST の副所長と

ここでは、CEA-LIST の APOLINARSKI 副所長、IPA の藤江理事長など関係者を交えて、CEA-LIST と IPA の相互協力の具体的な内容についてディスカッションを行うと共に、CEA-LIST の形式手法やモデルベースをはじめとするソフトウェア信頼性向上技術に関する最近の研究内容について説明を受けた。

例えば、ソフトウェアのコードからリスクを検出し、安全性の分析を行う“Frama-C”は、CEA-LIST が開発に参画した C 言語の静的解析ツールである。実行の各時点でプログラムの変数の取り得る値のチェックを行い、プログラム上のデータの流れをトレースする機能を有する。導入事例の中には、エアバスの大型旅客機 A380 の組込みプログラムも含まれている。

“FLUCTUAT”は、制御系等における C 言語での浮動小数点演算の丸め誤差の伝播を解析するために開発された静的解析ツールである。ソフトウェアの数値誤差が引き起こした障害の例としては、1996 年のアリアン 5 ロケット爆発事故や、1991 年のパトリオットミサイル迎撃失敗事故などが挙げられるが、このようなソフトウェアの安全性を追求する研究は極めて重要である。

CEA-LIST では研究開発成果を積極的に外部に展開するため、共同研究の枠組みに積極的に参加すると共に、パートナー企業との連携も促進している。欧州における R&D 投資の特色として、民間 R&D 投資が低く、公的部門の比率が高くなっており、民間 R&D 投資の比率を増やすため ETP※1 という民間主導の枠組みが導入されている。CEA-LIST は“ARTEMIS”という組込みシステムのための ETP に参画している。その他にも、フランスの DIGITEO※2 や SYSTEMATIC※3 などと連携し、欧州レベルでは、EICOSE※4 など、多くの研究団体に加盟して有意義なイノベーションエコシステムのもとに、共同研究を実施している。

また、年間活動予算の内の 2/3 以上（2011 年は約 4,600 万ユーロ）を政府外の補助金で賄っている。例えば、パートナー企業からの研究費や欧州委員会から委託されたプロジェクトの研究費等、多くの企業や研究機関との連携形態により、様々なビジネスモデルを提供している。現時点でのパートナー企業の数は 100 社に及び、共同プログラムの件数も年々増え続けている。

3　CEA-LIST のビジネスパートナー企業視察

応用技術研究所である CEA-LIST は、産官連携の橋渡しを行うことで、開発した技術や研究の成果を社会に有効に移転している。

研究技術が実用化段階に入ると、IP（知財管理）を中心とする独自のビジネスモデルを提供している（図 1）。インテグレーターからの依頼に基づき、CEA-LIST はソフトウェア、システム、ツール等の開発を担い、その後のアップデートやトレーニングについては、ライセンスを譲渡したテクノプロバイダがインテグレーターに対し直接的なサポートやメンテナンス役務を提供する。該当するテクノプロバイダが存在しない場合は、必要に応じて、下述（2）のようなベンチャー企業を新たなテクノプロバイダとして立ち上げる補足的ビジネスモデルを採用している。

今回 CEA-LIST のパートナー企業 6 社を訪問し、事業内容や共同研究の取り組み、CEA-LIST との連携によるソフトウェア・エンジニアリングを適用した製品・ツール等の市場展開の状況・展望等に関して意見交換を行った。

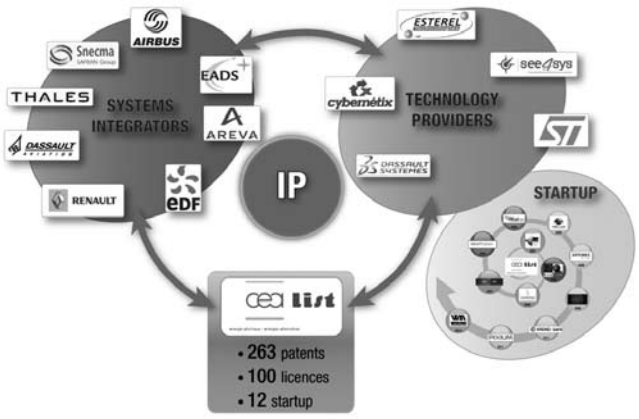


図 1　CEA-LIST 独自のビジネスモデル

脚注

※1　ETP：European Technology Platform
※2　DIGITEO：情報通信技術連合会
※3　SYSTEMATIC：複雑系産業クラスター
※4　EICOSE：European Institute for Complex Safety Critical Systems Engineering
※5　CESA：フランスにおける Automotive Electronics 会議
※6　SASHA：Safety check of Automotive Software & Hardware Architecture

(1) STmicroelectronics 社

世界有数でヨーロッパ最大の総合半導体メーカである、STmicroelectronics 社は、エネルギーマネジメント、セキュリティ、ヘルスケアスマートデバイスなど、幅広い半導体ソリューション事業を展開している。従業員は約 50,000 名、うち約 1/4 の 12,000 名が R & D または製品設計に従事しており、2011 年度の研究開発費は売り上げの 1/4 を占めている。創業以来、政府機関である CEA-LIST との密接な連携を継続しており、研究開発に対する積極的な取り組みを続けている。

最近の研究成果として、CESA※5 では SASHA ツール※6 が報告された。SASHA ツールは、機能安全の新しい規格である ISO 26262 に準拠した、自動車組込みソフトウェア開発用ツールである。ハードウェアを使うことなく、前もってデバイスの内部動作の詳細を確認することが出来、機能安全の新基準を適用する際の全体的なコストへの影響を軽減するのに役立つ。またこの他にも、最近ではスマートフォンと連動したシステムの開発や、そのセキュリティへの対策も進めている。

(2) Krono-safe 社、Sherpa-Engineering 社、See4Sys 社

Start-up 企業として、Krono-safe 社、Sherpa-Engineering 社、See4Sys 社を訪問した。これらの会社は CEA-LIST から投資と技術移転を受け、ルノー社や Delphi 社などと共にテクノプロバイダとして、自動車分野向けの製品やサービスを開発している。

Krono-safe 社は、従業員 12 名で、組込みソフトウェア用リアルタイム OS を開発している。自動車に搭載されている組込みソフトウェアの規模は、1980 年は 5 万行程度だったのに対し、2010 年には 2 千万行にも達しており、リアルタイム OS に求められる要件はますます厳しいものとなっている。CEA-LIST は一昨年、このようなテクノプロバイダを利用したビジネスモデルの成果として、より安全で効果が高い車載向けの新型リアルタイム OS「PharOS」を開発した。

Sherpa-Engineering 社とそのグループ会社である See4Sys 社は、CEA-LIST と Joint Lab を設立し、共同研究を行っている。Sherpa-Engineering 社の主要顧客はルノーをはじめとする自動車メーカで、従業員は 250 人、売り上げは 1,600 万ユーロを超える企業に成長しており、CEA-LIST から技術の使用ライセンスを受け、売り上げの一部をライセンスフィーとして支払うビジネスモデルを採用している。

(3) ルノー社

フランスの自動車メーカであるルノー社は CEA-LIST と、「Sustainable mobility（持続可能な移動手段）」、「電気自動車のためのリチウムイオン電池」、「INNOVIA」の 3 テーマで戦略的パートナー協定を結び、研究開発を行っている。INNOVIA は、CEA-LIST とルノー社がバーチャルリアリティの分野において共同で設立した研究所である。ここではパテ、溶接、塗装作業などの動作を検証する研究、複雑な動きをバーチャルリアリティ上で訓練する研究などが行われており、最終的にはこのような動作の遠隔操作を実現することを目指し、研究を行っている。

(4) Esterel 社

Esterel 社はソフトウェア開発のソリューションプロバイダとして、モデルベース開発手法によるセーフティクリティカルな組込み制御ソフトウェア開発支援ツール“SCADE”を開発している。SCADE は、DO-178B、Def Stan 00-56 をはじめとする、航空・軍事・鉄道・原子力・医療などの分野で、ソフトウェアの国際安全規格に多数認証されている。

CEA-LIST と Esterel 社が共同で設立した、通称“LISTEREL”研究所では、CEA-LIST が開発した UML 開発ツール Papyrus と SCADE を組み合わせた、オープンソースツールを開発している。他の Papyrus コンポーネントと操作方法が共通なことや拡張性が高いこと、また SCADE と同等の操作感覚を得られるなど、多くの利点がある。

Esterel 社が提供するモデリングツールは、厳密な数学的モデル定義、トレーサビリティの記述など形式手法によるモデルベースアプローチを利用したものであり、上述の国際標準に準拠している点と相まって、大規模システムのモデル生成をスピードアップ出来る点、マルチユーザとのコラボレーションを可能とする点などの強みを持っている。Esterel 社は、形式手法によるモデルベースアプローチがエンジニアリング及びデザインプロセスを変革していくと捉えている。

4 まとめ

CEA-LIST のフォントネーオーローズ研究所では、バーチャルリアリティ、ロボティクスなど、経済的・社会的ニーズが高い先進の研究内容について、意見交換を行った。

サクレーのナノ・イノブ・センタでは、相互協力協定に基づく共通のビジョンやこれからの相互協力活動について意見交換を行った。更に形式手法やモデルベースをはじめとする最近の研究取り組み内容など、CEA-LIST の先進的なソフトウェア・エンジニアリング研究の取り組み状況や産業界との連携による成果の展開状況についての議論を行い、SEC の活動に活用出来る有用な情報を入手することが出来た。

また、パートナー企業についても 6 社を訪問して各社の事業内容や CEA-LIST との共同研究の具体的な内容、連携によるソフトウェア・エンジニアリングを適用した製品・ツール等の市場展開の状況・展望等に関して意見交換を行い、有用な情報を入手することが出来た。

IPA は引き続き、CEA-LIST との連携・協力を行っていく予定である。

ラーセン教授を迎えて

ー【SEC特別セミナー】形式手法の98導入事例の調査・分析から見る高信頼性ソフトウェア開発の現状（2012年10月23日開催）よりー

SEC 調査役
新谷 勝利

1 はじめに

SEC では、開発の上流工程からの品質作り込みという課題に取り組んでいるが、上流品質技術部会の人材育成ワーキンググループ（WG）では形式手法の導入教材の検討・作成を行い、その成果を公開している。その人材育成 WG の準備会において、海外での動向について広く紹介していこうということを相談し、国際的な形式手法活用調査を実施しているデンマーク・オーフス大学のラーセン教授を招聘し講演していただく計画を立てた。

ラーセン教授はヨーロッパのみならず、日本を含む多くの国で形式手法の実践を支援しており、その活躍はデンマーク国内にとどまらず、EU 域内、日本を含むアジア諸国、ロシア、全米と広範囲に及んでいる。

多忙なラーセン教授の日程調整は、以前から VDM 及びそのツールの日本への導入で教授と協力関係にあった佐原委員にフォローをお願いした。その後、パリでの形式手法会議にて、荒木主査及び佐原委員とラーセン教授との間で、どのような講演内容が日本での形式手法の啓発に有効かを相談していただいた。詳細な内容と流れ、及び具体的な時期などについては新谷が担当し、何回かやりとりをした後、講演内容を以下のように決めた。

「社会の隅々までソフトウェア及びそれを中心とするシステムが構築されている現在、それらの構築を担当する私たちは、プロフェッショナルとして如何なる知見を持つのが望ましいのか。銀の弾丸は無いとされながらも、一つの可能性としての形式手法についての展開、実践、今後についての講演を行う、1 日の SEC 特別セミナーを実施する。このセミナーは一方通行の講演をするだけではなく、参加者との積極的な質疑応答も可能とするようにしたい。」

当日の資料は SEC ホームページからダウンロード可能となっている（http://sec.ipa.go.jp/seminar/2012/20121023.html）。

以降で、講演資料を引用しながら、ラーセン教授のメッセージを再現することとする。

図 1　ラーセン教授（デンマーク・オーフス大学）

2 セッション 1：形式手法の概要

今回の SEC 特別セミナーに参加していただきたい受講者のプロファイルを次のように設定した。

「実務者が大部分であり、必ずしもいままで形式手法について詳しく学習・調査した経験は無い。しかしながら、形式手法を机上の話ではなく実践するとどうなるか、どのように導入出来るのかを理解したいとの思いを持っている。」

SEC 人材育成 WG では、形式手法入門の教材を開発し、パイロット研修コースの実施を通じてその改良を進めてきた。この WG のテーマは、どうすれば形式手法へのある意味「食わず嫌い」を払拭出来るかということであった。この経験も踏まえて、ラーセン教授による最初のプレゼンテーションにあたっては、SEC 人材育成 WG の荒木主査及び WG の各委員から日本での導入にかかわる経験など追加のコメントをしていただいた。

「数学が形式手法の基礎となっている」ため、近寄り難いとい

う傾向があることが実際のようである。そこで、ラーセン教授には、形式手法は数学を基礎にしているが、多くの人が考えるように難しいものではない、という話から始めていただいた。

ただ「数学」というだけで苦手意識を持ってしまう人に対しては、複雑な記号や数式の意味を説明するのではなく、どのようにしたら数学が本質的に持っている抽象化能力を活用してもらえるであろうか。例えば、「ある対象の要素が集まって構成されている」と見えれば「集合論を活用すればよい」ということである。このときに高校で学習する集合論の基本的知識で十分ということである。また、対象を何らかの表現で説明するときに、厳密さの観点では「段階」があることに言及していただいた。なぜなら「形式手法を導入するということは、数学が基礎であるから証明をしなければ使ったことにならない」という誤解があるからである。そうではなく、まずは「背景となる数学の考え方で記述してみる」というのを“レベル 1”とし、次に「ツールの支援で抽象化記述が可能な言語、例えば VDM を使用する」ことを“レベル 2”とする。このアプローチは、実際に世界の多くの形式手法の導入例で実践されている。

例えば、日本における本格的な形式手法導入の例としてしばしば引用されるフェリカネットワークス株式会社においても、信頼性を確保することを意図した部分（モジュール）での信頼性獲得のために十分な成果を出している。私たちは自然言語で表現された仕様書を、コンピュータで実行出来るように C 言語を用いて翻訳し、HOW の記述をしている。このときに留意しなければならないのは、翻訳前の仕様書があいまいであったら、どんなにうまく翻訳をしても良い結果にはならないことである。形式手法導入のレベル 1 及びレベル 2 は、対象に数学の力を活用して抽象化、あるいはモデル化して、WHAT の仕様記述をすることである。形式手法をこの段階で導入することのメリットの一つは、仕様記述言語を利用出来ることであり、WHAT の段階で、従来だと HOW の段階でなければチェック出来なかったことをチェック出来ることである。HOW あるいは実装の段階の前に挙動をチェックすることは、ソフトウェアのような論理対象ではなく、物理対象ではモデル化してその挙動を解析することで実施されている。例えば、航空機の操縦ではシミュレータを作り、空中での挙動を論理的に表現して抽象化している。形式手法をソフトウェア開発に適用するということは、実装の前の段階で、対象に数学の力を活用し、抽象化あるいはモデル化して、シミュレーションするということに他ならない。

3 セッション２：形式手法の導入事例の分析

形式手法には多くの提案がなされているが、実践という観点では、ツールによる支援の豊富さもあり VDM[※1] が代表的なものとなっている。この 2 番目のセッションではまず、1994 年の ConForm プロジェクトから 2007 年の FelicaMobileChip プロジェクト（現在、三代目として進行中）までの 7 つの VDM プロジェクトが説明された。次いで、交通システムへ展開されている形式手法の例として B メソッドが示された。

(1) セッション2の概要

セッション 2 のハイライトは、2009 年に実施された 60 以上のプロジェクト調査からの報告と、2012 年に報告書としてまとめられた Deploy[※2] という EU の 4 年にわたるプロジェクトからの報告である。当報告書は、98 のプロジェクトが同じ様式で整理してまとめられている。

① 2009 年の調査報告

以下の項目がサマリーとしてまとめられている。

- 形式手法導入の将来は明るい。活用状況は報告されており、素晴らしい成功例が出ている
- 形式手法導入例の殆どが軽量の形式手法[※3]
- 形式手法導入にあたり、ツールが極めて重要な位置を占めているが使用容易性にまだ問題がある
- 形式手法導入の決定の助けになる証拠は不十分である
- 形式手法導入の理由はリスク回避のためである
- 形式手法が継続したプロジェクトに活用されたというデータは不十分である
- 形式手法導入に関わるスキルと心理的障壁は依然として高い
- 形式手法導入に対しては、訓練と教育が依然として必須である

② 2012 年の調査報告

以下の項目が定量的にまとめられている。

- 分野別：交通機関に関するものが 30 以上。金融関係が 10 以上。防衛、消費者用電気製品及びテレコムがそれぞれ約 10
- 適用タイプ:実時間システムが 30 以上。分散システムが 25 弱。大量にデータを扱うもの、制御、並列処理、及びハードウェアが 15 程度
- 適用技術内容：仕様記述及びモデリングに 80 以上。形式記述の実行に 50 以上。インスペクションに 40 弱。モデルチェッキングに 30 以上
- 事前経験：51% が以前の経験が大。37% が以前の経験が少。18% が未経験。これは、2009 年調査報告からは明らかな変化で、複数回の経験があることを明確に示している。これは結果として新しい訓練・教育を必要としない
- 品質：88% が改善。12% が以前と変わらず
- 経費：33% が改善。59% が以前と変わらず。8% が以前より悪い
- 期間：25% が改善。55% が以前と変わらず。20% が以前より悪い
- 抽象化に対して効果的：実際に形式手法を導入しないとしても、形式手法の考え方を取り入れるだけでも開発の助けになっている

・テストの自動化に効果的：数千の異なるパラメータを構成してソフトウェアが出来ており、テストケースをすべて手で書くのは不可能
・形式手法を導入したプロジェクトの 80% がツールは十分揃っているとし、7% がそう思っていない
・プロジェクトの 73%が形式手法を積極的に再使用するとし、２%が再利用しない

(2) 2012 年調査報告の分析によるラーセン教授の推奨事項

・「産」は新しい解決策によってシステム全体に対応する製品を早くマーケットに出さなければいけない。一方、「学」は研究及び論文になる問題に偏っているため、システム全体への適用というより小さな差分アプローチが主体であり、それではマーケットに対応出来ない。このような違いを認識した上で両者を結び付ける橋渡しの役割が必要になる。この役割を果たす支援者あるいは組織があってこそ「産学協働」は成立する
・形式手法導入の経験が無い場合、小プロジェクトで試行し成功するアプローチが適切。開発環境を最初から大きく変革しない
・形式手法で全体を記述することはせず従来型開発との並行利用をすることが重要
・B メソッドを交通機関に採用した方法、ロックウェル・コリンズ社がツールチェーンで実施した方法では、長期間のコミットメントが必要
・形式手法に知見が無い場合、まずは教育から始める
・形式手法の適用先として、クリティカルなシステムなどの従来手法では適切に問題解決出来ない分野がある
・形式手法は、どんな問題にも対応出来る万能薬とは考えない
・形式手法導入にあたっては、良い指導者を周りに持つ

4 セッション３：これからのソフトウェア開発と形式手法

今後形式手法の導入を進める分野は、従来手法では不十分で問題がある、と考えられる分野であり、それは計算（Computation）、通信（Communication）及び制御（Control）の 3 分野をカバーする Cyber-Physical Systems と考えられる。この予測から、ラーセン教授が関係するプロジェクトのみでも約 10 億円のファンドを EU から得て計画が進行中であることに留意する必要がある。日本においてもシステムを構成するソフトウェアが社会のインフラの基盤となっているという状況はますます拡大するであろう。Cyber-Physical Systems というものの具体化も進むであろう。既存システムの単なる追加拡張というものではなく、全体的なシステムとしてアーキテクチャ上の統一性が維持され、システムとしての安心・安全は担保されていなければならない。このために、前節でラーセン教授が指摘されているように、「産学協働」が必須であり、それを推進するために「官」の支援も必要になってくる。EU における Cyber-Physical Systems への取り組みは、以下の要素をもって「産官学」で推進されていることに留意する必要があるであろう。

・組込みシステムにおけるシミュレーション機能を取り込んだ DESTECS※4
・システム・オブ・システムズ（SoS、Systems of Systems）への統合ツール環境としての COMPASS※5

これからのシステムは規模も大きく、複数のシステムが相互に関係する SoS に向かうであろうし、SoS では 3 つの C（Computation, Communication, Control）を要素として持つ Cyber-Physical Systems になるであろう。これらは規模も大きく複雑で、相互に関係するが故に、従来の開発手法のみでは必要とされる高信頼性を担保するのは困難であろう。形式手法は万能薬ではないが、極めて重要な手法であることは明らかであると考えている。

5 おわりに

VDM が開発されたのは、1960 年代から 70 年代にかけてであり、決して新しいものではない。またその導入は、遅々としたものではあるが徐々に実績が出てきている。その習得のための教育に企業が投資することは十分に意味があることであると考えている。今後のシステム開発で想定しなければならない事項は、更に大きく複雑なものとなり、それに備える上でも形式手法の学習及び小さなプロジェクトでの形式手法の適用が有効であると考えられる。今回のラーセン教授のセミナーは、それらを教示していただいたものであり心より感謝申し上げる次第である。また、このセミナーの質疑応答において積極的に関与していただいた荒木主査を始め委員の皆様方にも感謝申し上げる。

脚注

※1 VDM：Vienna Development Method, オーストリアのウィーン（Vienna）になる IBM の研究所で 1960 年代から 70 年代にかけて開発された形式手法。その仕様記述言語である VDM-SL は 1996 年に ISO/IEC 13817 として ISO 化。
※2 Deploy：EU における形式手法の実践プロジェクトで、成果が Industrial deployment of system engineering methods providing high dependability and productivity, A. Romanovsky, M. Thomas（Eds）, Springer. November 2012. にて発表予定。
※3 軽量の形式手法：「2. ステップ 1：形式手法概要」に説明されている厳密さのレベル 1 に相当する仕様記述がされている状況を軽量と称している。
※4 DESTECS：Design Support and Tooling for Embedded Control Software というコンソーシアムで、詳しくは、http://www.destecs.org/ 参照
※5 COMPASS：Comprehensive Modelling for Advanced Systems of Systems というコンソーシアムで、詳しくは、http://www.compass-research.eu/index.html 参照

テストの技術力強化に向けたテスト技術のスキル標準

ー「SSFに基づくテスト技術スキルフレームワーク（Test.SSF）」策定の取り組みー

株式会社アフレル
元 IPA/SEC 研究員
渡辺 登

アヴァシス株式会社
元 IPA/SEC 研究員
小林 直子

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※1）と一般社団法人IT検証産業協会（IVIA※2）は、「SSFに基づくテスト技術スキルフレームワーク（Test.SSF）」を策定し、両団体のWebサイトで公開している。本稿では、スキル標準策定の背景と取り組みについて、また策定したスキル標準を紹介し、その活用方法について解説する。

1 はじめに

システム開発やソフトウェア開発において、テスト技術の重要性が増している。製品やサービスにおける品質の善し悪しが、競争力を左右しているためである。人命に影響を与えるシステムや、日々の生活を支えるシステムといった社会インフラシステム、システムの提供開始時期がビジネスに対して大きく影響を与えるシステムなど、製品やサービスに求められる品質は多岐にわたる。テスト技術は、これらに求められる品質を確認する重要な技術の一つであり、これからの日本の産業においても、重要な役割を担う技術であるといえる。

近年、システムやソフトウェアの規模は増加傾向にあり、また、その複雑度も高まっている。このようなシステムやソフトウェアの動作を確認するには、高度なテスト技術が求められる。また、生産性や品質を高めるために、OSS（オープンソースソフトウェア）、既存プラットフォームを活用した開発、派生開発やプロダクトライン開発などが行われている。これらにより開発量は抑えることが可能となっている一方、システムの動作確認量は増加し、テスト技術の重要性もより高まっている。

参考までに、組込みシステム開発における人材として、まず、ソフトウェアの開発技術者（ソフトウェアエンジニア）の割合が 37% と一番多いが、次いで多いのがテストエンジニアの 22% であり、高い人数比率になっている（図 1）。

このようにテスト技術の重要性が高い一方で、日本のソフトウェアテスト技術者の多くは、過去の経験に頼ったテストを実施していることが多く、自分の保有するテスト技術を明確に他人に説明することが出来ない状況にある。ただそのような中で、一部のソフトウェアテスト技術者は豊富な経験と鋭い洞察力を基にした質の高いテストを行っている。しかし技術の高さを説明出来ないままでは、テストを事業とする企業や組織は、顧客に価値を訴求出来ず収益が悪化し、有能な人財にも去られ、壊滅していくという最悪のシナリオも想定される。

そこで ASTER と IVIA では、日本のソフトウェア品質向上において重要な役割を担うテストの技術力強化を目的に、ソフトウェアテスト技術に関するスキル標準「SSFに基づくテスト技術スキルフレームワーク（Test.SSF）」を策定した。

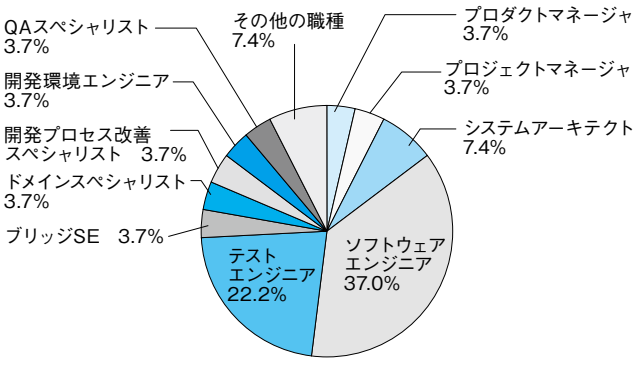


図1　2010年版組込みソフトウェア産業実態調査報告書（Q3-3-1プロジェクトの職種構成）

2 Test.SSF とは

Test.SSF ではソフトウェアテストに関する、最適な人材育成や人材を有効活用するための指標や仕組みを規定している。Test.SSF を活用することで、テストのスキルが高い組織は、スキルの高さを明示することが出来る。逆にテストのスキルが高いとはいえない組織は、目標を明確に設定し、その目標に向かいスキル向上の努力を始めることが出来る。また顧客組織は、より高いスキルを持つ組織を選定して仕事を依頼することが出来るようになる。

Test.SSF の策定においては、テスト技術を体系的に整理するために IPA の「組込みスキル標準 ETSS※3 のスキル標準フレームワーク（SSF※4）を活用しており、Test.SSF という名称はこの SSF を用いていることを意味している。また Test.SSF では組込み領域やエンタープライズ領域と呼ばれるドメインに依存しないようソフトウェアテスト技術者のスキルの抽象化を行い、広くソフトウェアテストにおいて利活用することを目指している。

Test.SSF は 3 つの基準で構成している。当該分野の技術を体系的に整理する「スキル基準」、職種を定義した「キャリア基準」、人材育成を実現する「育成ガイド」である。現時点で、スキル基準が正式版（Ver1.0）、キャリア基準がβ版として公開されており、現在育成ガイドの作成を検討中である。次節では、現在公開中のスキル基準とキャリア基準について解説する。

3 Test.SSF「スキル基準」

ソフトウェアテストに必要なスキルを明確にし、体系的に整理したものが、Test.SSF「スキル基準」である。ソフトウェアテスト技術者のスキルを可視化し、効果的で効率的な人材育成の推進を目的とする。Test.SSF「スキル基準」は 2011 年 8 月にβ版が公開され、現在は正式版（Ver1.0）が公開されている。

Test.SSF「スキル基準」は ETSS のスキル基準を基に策定している。ソフトウェアテスト実施時に使用する技術を開発技術、管理技術、技術要素の 3 つのスキルカテゴリに分類し、更にテストアクティビティに沿って技術とスキルを体系的に整理している。

整理の過程においては、ソフトウェアテスト技術者が現場で実施していることをベースとしたが、これを「普段やっていること」という意味にとどめず、ソフトウェアテスト技術の進化を促すために、先進的な現場で取り入れられている技術についても含めることとした。したがって、スキルの体系化においては、ISTQB※5 及び JSTQB※6 のテスト技術者資格認定や IVEC※7 のシラバス類も参考にしている。

ETSS のスキル基準と異なる Test.SSF「スキル基準」のポイント 2 点を次に紹介する。

[1] テストを開発する

近年のテストのライフサイクルの変化に伴い、Test.SSF ではテストの要求分析から評価までをテストライフサイクルと定義し（図 3）、テスト技術の体系化を行っている。

図3　テストライフサイクル

その過程で基となる考え方が、「テストを開発する」という基本概念である。これは、テスト対象ごとにテスト要求分析で定義したテスト目的と品質目標からテストアーキテクチャスタイルを選択し、それに基づくテストの設

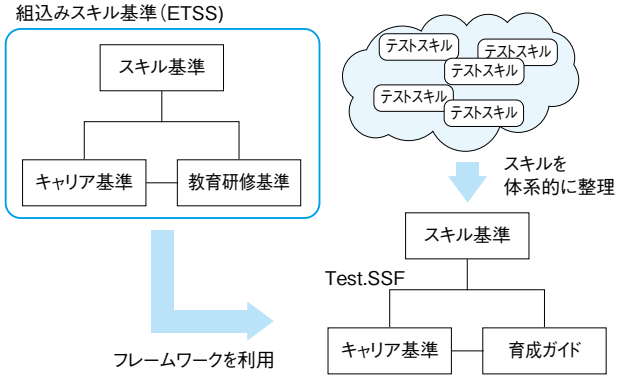


図2　Test.SSFの構成と組込みスキル標準

脚注

※1　ASTER：Association of Software Test Engineering, 特定非営利活動法人ソフトウェアテスト技術振興協会, http://aster.or.jp/
※2　IVIA：IT Verification Industry Association, 一般社団法人 IT 検証産業協会, http://www.ivia.or.jp/
※3　ETSS：Embedded Technology Skill Standards, 組込みスキル標準, http://sec.ipa.go.jp/ETSS/index.html
※4　SSF：Skill Standards Framework
※5　ISTQB：International Software Testing Qualifications Board, 国際ソフトウェアテスト資格認定委員会, http://istqb.org/
※6　JSTQB：Japan Software Testing Qualifications Board, 日本におけるソフトウェアテスト技術者資格認定運営組織, ISTQB 加盟, http://jstqb.jp/
※7　IVEC：IT 検証技術者認定試験 ,IT 検証産業協会（IVIA）が認定するテストエンジニアの資格試験, http://www.ivia.or.jp/item/43.html

計・実装とテスト実行、そして報告・評価を実施するという考え方である。つまり、テストを実行する行為だけをテストとして取り扱うのではなく、ソフトウェア開発と同様にソフトウェア開発のライフサイクルを通し、それぞれテスト対象に合わせたテストを開発していく、という考え方に基づくものである。

[2] 繰り返し構造

ソフトウェア開発工程とテストライフサイクルの関係を示す(図4)。ソフトウェア開発工程において、ソフトウェアテストのライフサイクルは、テストレベル（コンポーネントテスト、統合テスト、システムテスト、受け入れテスト）ごとに繰り返し存在する。

Test.SSF「スキル基準」では、テストレベルを第1階層として、第2階層以降でテストライフサイクルが繰り返し表現される構造を図5のように捉え、それをスキルカテゴリとして整理している（図6）。

以上の考え方を基に策定したTest.SSF「スキル基準」を用いることで得られる効果は、ソフトウェアテスト技術者のスキル（強みや弱み）が可視化され、目指すべき姿や伸ばすべきスキル、新たに獲得するスキルを明らかに出来ることにある。また、スキルアップの度合いが可視化されることで、成長へのモチベーションアップや維持にも役立つことが期待が出来る。

更に個人や組織（企業やチームなど）のスキルを可視化することで、経営戦略やプロジェクト計画の立案、人材の採用・調達、リスクマネジメントなどの人材活用面においても利用が可能となる。

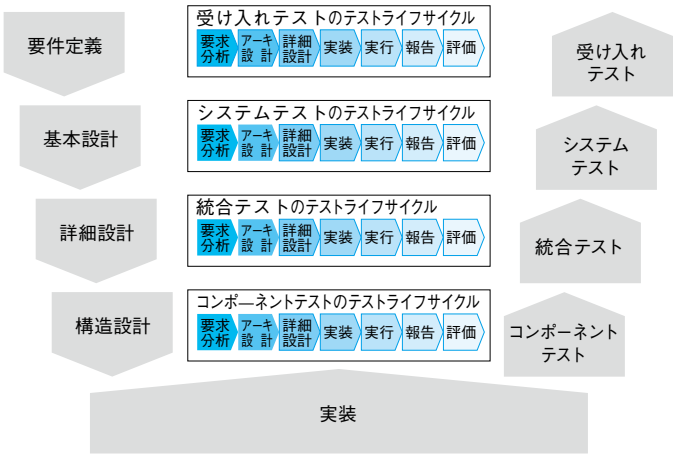


図4　ソフトウェア開発工程とテストライフサイクル

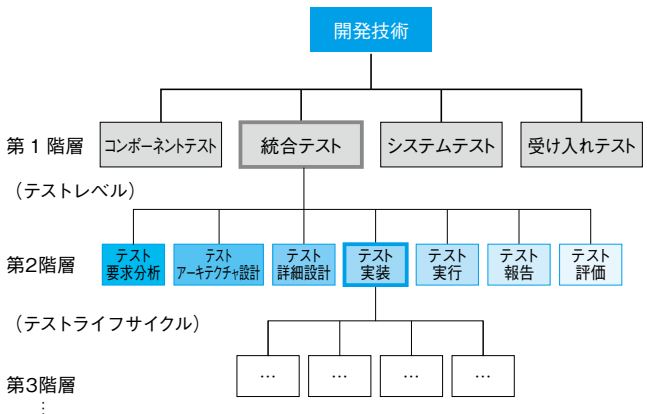


図5　テストライフサイクルとスキル基準

4 Test.SSF「キャリア基準」

Test.SSF「キャリア基準」は、テストプロフェッショナルの戦略的育成を実現するために策定している。ソフトウェアテストに関する職種の分化と深化を図り、スキルを高め、キャリアプランを設計出来るようにする。また、どんなレベルの人材が何人必要かといったプロジェクト体制の確立などで利用されることも想定している。現在はβ版が公開されており、今後有識者や利用者の意見も得ながら、正式版を策定する予定である。

ETSSのキャリア基準を基に策定したTest.SSF「キャリア基準」は、ETSSキャリア基準で定義されているテストエンジニアとの互換性を考慮しつつ、ソフトウェアテストを担う職種を定義している。

職種としては、テストエンジニア、テストマネージャ、テストアーキテクトの3職種がある。

ETSSでも定義されているテストエンジニアは、Test.SSF「スキル基準」を反映した形で、再定義している。そのため、人材育成での利用を目的として定義しているスキルの分布特性はETSSとは異なるが、両スキル標準間の互換性は確保している（図7）。

第1階層	第2階層	第3階層			スキル項目例
1 コンポーネントテスト	1 テスト要求分析	準備	1	テスト要求分析の準備	インタビュー技法、文献調査、影響度分析
2 統合テスト		獲得	2	テスト要求の獲得	インタビュー技法、文献調査
3 システムテスト		分析	3	テスト要求の分析	ゴール指向分析、メトリクス、GQM など
4 受入テスト		作成	4	テスト要求分析成果物の作成	文書作成
		検証	5	テスト要求分析成果物の検証	レビュー技法、トレーサビリティ
※共通	2 テストアーキテクチャ設計	準備	1	テスト要求分析成果物の確認と用意	トレーサビリティ
			2	テストベースを準備する	文献調査
		獲得	3	テストアーキテクチャスタイルに関する要求の獲得	
		分析	4	アーキテクチャスタイルに関する要求の分析	モデリング技法
			5	アーキテクチャスタイルの選択	品質特性、網羅型／検出型、テストタイプ など
			5	テスト全体の構造の設計	リスク識別、リスク分析
			6	テスト全体のバランスの調整	ワイドバンドデルファイ
			7	テスト環境の構築方針・方法の検討	
			8	テスト詳細設計の指針・原則の検討	

図6　テストライフサイクルとスキルカテゴリ（開発技術）

テストマネージャは、ソフトウェアテストにおけるプロジェクトのマネジメントを担う職種である。システムやソフトウェアの開発において、ソフトウェアテストとしてのサブプロジェクトのマネジメントや、評価を目的としたテスト単体のプロジェクトのマネジメントを担う。

テストアーキテクトは、テストの技術力向上を目的として戦略的に定義した職種である。現時点では人数が限られるが、今後のテスト技術向上を考えた場合、キャリアパスとして明示する必要性があり定義した。システム側の要求から、テストの戦略や構成、環境を具体化することが求められる。テストの効率や品質を大きく左右するため、これを担える人材は重要と考え、多くのテストアーキテクト職種の人材を育成することを目指す（図 8）。

テストオペレータやテストコンサルタントといった職種も検討候補ではあったが、ソフトウェアテスト技術に関する人材育成という観点を考慮して定義の対象から外した。

キャリア基準では、それぞれの職種の定義とキャリアレベルごとに求められるスキルのレベルを明示している。それをスキル分布特性として表現することで（図 7）、求められるスキルと、現状保有するスキルのギャップを明らかにすることが出来る。しかし、これはソフトウェアテスト技術者のキャリアレベルを判断するために利用するものではなく、人材育成において獲得すべきスキルを明示するために利用することを想定している。従って、キャリアレベルの判断には、キャリアレベルの定義を利用する。これは共通キャリア・スキルフレームワーク（CCSF※8）と同じキャリアレベル定義としているため、組織や業界における相場観の共有が可能となる。

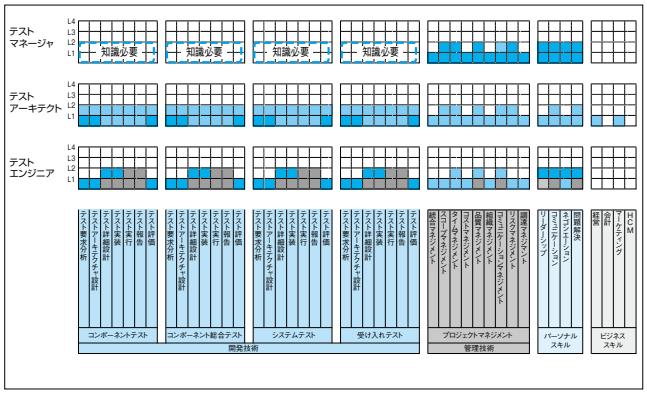


図7　人材育成のためのスキル分布特性

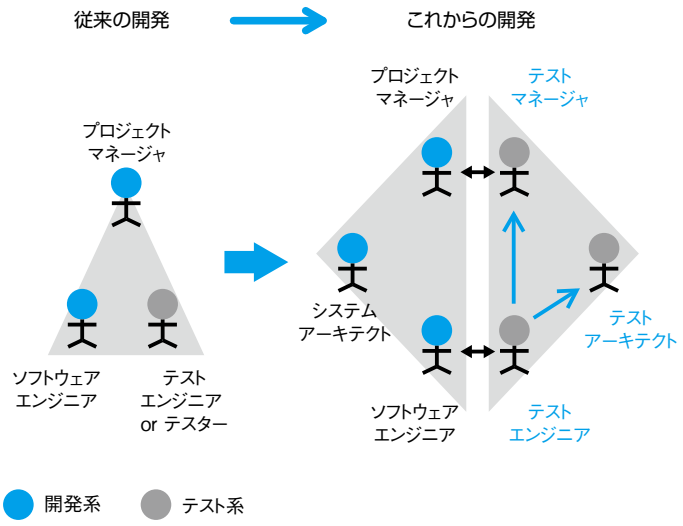


図8　テストの職種を分化･深化

5　おわりに

Test.SSF を用いて自分たちのテストスキルを明確に把握し、鍛える組織が増え、日本のソフトウェアテスト技術が世界に冠たる存在に成長して欲しいと考えている。

そして日本のソフトウェア開発力強化の実現、更に日本経済の発展には、ソフトウェアテストが重要な技術であることをアピールし、人材育成の活性化を進めたい。

Test.SSF は今後、育成ガイドとキャリア基準の正式版（Ver1.0）を策定すると共に、有識者及び活用者の声を取り入れながらブラッシュアップしていく予定である。ぜひ多くの組織、技術者にご活用いただき、テスト技術の向上に役立てていただきたい。

最後に Test.SSF 策定メンバを紹介する。現在、各団体･企業の有識者 11 名によるボランタリーな活動により、本取り組みが進められている。

参画団体・企業並びに策定メンバには感謝を申し上げたい。

鈴木 三紀夫　（ASTER，MRT コンサルティング）
辰巳 敬三　（ASTER，富士通株式会社）
吉澤 智美　（ASTER，日本電気株式会社）
石川 俊一　（IVIA，日本ナレッジ株式会社）
佐々木 方規・相馬 武　（IVIA，株式会社ベリサーブ）
前川 祐次郎・西家 英生（IVIA，株式会社ヴェス）
小林 直子　（元 IPA/SEC，アヴァシス株式会社）
渡辺 登　（元 IPA/SEC，株式会社アフレル）
藤原 由起子　（IPA/SEC）

［敬称略］

脚注

※8　CCSF：Common Career Skill Framework，共通キャリア・スキルフレームワーク（第一版･追補版を 2012 年 3 月 26 日に公開），http://www.ipa.go.jp/jinzai/itss/ccsf/download.html

一般財団法人関西情報センターの紹介

一般財団法人関西情報センター
会長
森下　俊三

1970年の大阪万博の年に関西における情報化の推進拠点として設立されたのが「財団法人関西情報センター」です。その後、ダウンサイジングやマルチメディア化、インターネットの普及などの情報化の変遷を乗り越え、その時代に合った様々な情報化支援事業・産業活性化事業を展開してきました。近年、情報化が進展し情報化社会が成熟してきたことにより当財団の役割が大きく変わっており、今後は「情報化の推進」から「情報化社会の進展に伴い顕在化する新たな課題への対応」に重点を移していくことが重要だと思います。

1 経緯

当財団は、1970 年に当時の通商産業省の認可を得て、大阪府、大阪市、関西財界企業が一体となって設立された、情報化の推進、システムの開発・普及を目的とする情報系シンクタンクです。

設立当初は、大阪万博で使用した汎用機を移設し、統計分析やシミュレーションなどの計算処理の他、欧米への経営情報システムの視察団の派遣、システムエンジニアの育成などの事業を実施していました。

インターネットの黎明期には、ネットワークの相互接続に関わった他、ネットワーク技術者の育成やショッピングモールの実証実験の事業を実施してまいりました。

2002 年には、財団法人関西産業活性化センターとの一部統合を経て、財団法人関西情報・産業活性化センターに名称変更し、産業クラスター計画関連事業など中小企業の振興などにも取り組んできました。

そして、2012 年 4 月に、法人制度改革に伴い一般財団法人関西情報センターにリニューアルしました。

2 事業紹介

（1）調査研究・普及啓発事業

① e-KANSAI レポート

関西地域における情報化の実態を様々な角度から調査し、経年的に実施してきた調査結果と合わせて、政策支援やビジネス展開に有効なレポートとして情報発信しています。

② IT シンポジウム　－ Infotech －

情報通信分野における最新の技術やビジネスにおける課題や制度の中から関心の高いテーマを選定し、シンポジウム「インフォテック」を開催しています。

③関西 CIO カンファレンス

企業が経営戦略と一体化した効果的な IT 投資を実施するために、CIO の設置に向けた組織改革、CIO による業務改革などの重要性について議論していただく円卓会議とシンポジウムを実施しています。

④ビジネスイノベーションセミナー

中小・中堅企業や自治体における IT 戦略の再構築や新たなビジネス展開に資するため、ICT 利活用によるビジネス・イノベーションに繋がる技術動向を提供するセミナーを実施しています。

（2）ビジネス・政策支援事業

中小企業における IT 経営の支援事業として「e- 相談所」、情報家電企業とベンチャー企業のビジネスマッチングを支援する「DCP」、製品マニュアルやライターの技術向上を支援する「テクニカルライターの会」などの事業を実施しています。

（3）情報化推進事業

地方自治体におけるスポーツ施設の予約システムを

ASP にて提供しています。また、ビジネスにおいて安心で安全な情報の交換を実現出来る「セキュアサポートサービス」事業も手掛けています。

（4）社会システム支援事業

健康保険組合における事務の効率化やコスト削減を目的として、システム開発、システム運用支援など総合的なシステムソリューションサービスを提供しています。

（5）情報化基盤整備事業

近畿地域におけるプライバシーマークの審査期間（指定機関）として、プライバシーマークの審査・認定業務を実施し、個人情報の保護に取り組んでいます。

設立当初から考えると、汎用機の時代からインターネットとスマートフォンの時代へと随分社会的環境も変わりましたが、より安全で安心の出来る情報化社会の実現へ向けて、果たさねばならない使命も徐々に進化していると考えています。

2012 年 4 月に一般財団法人へ移行しましたが、全国的にも特色のある情報系シンクタンクとして、今後も地域の発展と産業の活性化のために活動していく所存です。

一般財団法人関西情報センター
会　　長　森下　俊三
専務理事　田中　行男
〒 530-0001
大阪市北区梅田 1 丁目 3 番 1-800 号
大阪駅前第一ビル 8 階
TEL 06-6346-2441
http://www.kiis.or.jp/

写真1　インフォテック2012

Column

周回遅れの電子政府

IPA顧問
松田 晃一（まつだ こういち）

最近、年金の手続きが必要になった。まぎらわしい用語と難解な説明の解読に四苦八苦しながら、なんとか済ませた手続き。その後送られてきた手続き完了のお知らせには、「住民票コードが確認出来たので、今後年１回の現況届や住所変更届の提出が不要になる」との記載があった。住基ネットと年金システムが何らかの形で連動したおかげで、届けの手間が省けることになったのだろう。大変ささやかではあるが、行政システムの電子化の恩恵をわずかに感じた次第である。

日本は18位

しかし、電子政府計画が声高に喧伝されてきた割には、私たちの日常生活が非常に便利になったとか、行政の効率がよくなって予算が節減された、といったようなことはほとんど実感出来ない。国連による世界電子政府ランキング2012年版では、日本は18位（2010年の17位からダウン）である。第１位は韓国、続いてオランダ、英国、デンマーク、米国となっている。米国は、2010年の２位から３ランク落とし５位となったが、オバマ大統領が就任直後から強いリーダシップのもとでディジタル・ガバメントを推進している。

米国は「政府の透明性と市民参加」へ

とくに注目したいのは、政府内の業務プロセスの電子化・IT化による「業務の効率化」という段階から、「政府の透明性確保と市民参加・協業」という新たな段階へ進化させようとしている点である。いわゆるオープンガバメントである。

オバマ大統領が就任直後に発表した「透明性とオープンガバメント」の覚書では、そのねらいを３つの基本原則として明快に示している。すなわち、

(1) 透明性：政府機関の活動についての情報を市民に分かりやすく提供し、政府が何を行っているかを市民が理解出来るようにする。
(2) 市民参加：行政機関の政策決定に市民の知見を幅広く採り入れ、行政機関の効率性と意思決定の質を向上させる。
(3) 協業：市民参加から更に踏み込んで、市民、団体、企業などと政府とが協業を進める。

米国では、このような取り組みが着々と進んでいる。

日本は「業務の効率化」の途上

これに比べ日本は、「業務の効率化」という初期の段階すら実現に四苦八苦しており、周回遅れと言わざるを得ない。政府システムの調達失敗のニュースを歯がゆい思いで聞いているのは筆者だけではあるまい。

この現状に対して、日本のIT戦略本部は2012年８月に、これまでの電子行政の取り組みが十分な成果を挙げてこなかったことを認め、電子行政推進の司令塔としての政府CIO制度の導入を決定した。政府CIOの強力なリーダシップのもとで、まずはITの活用による「行政業務の効率化やサービス性の向上」を着実に実現し、周回遅れを立て直していくことが待ったなしの課題であろう。

行政の「業務プロセス改革」が鍵

企業システムの成功には業務プロセスの改善から着手することはもはや常識である。これは政府システムでも同様で、行政の制度・業務プロセスの改革が実行出来るか否かにIT化の成否はかかっており、ここに政府CIOの大きな役割がある。現在の官僚組織とそのメンタリティのもとでは、このような業務プロセス改革には極めて大きな困難が予想されるが、これを乗り越えなくてはこれまでの失敗を繰り返すことになる。そのためにも政府CIOにはこのような業務改革に対する権限と責任を法の裏付けのもとに与える必要がある。

企業は生き残りをかけてIT活用による業務プロセス改革や新事業・新サービスの開拓にしのぎを削っている。国家にとっても同様である。電子政府の実現は国家の将来を左右する一大事業である。IT推進の旗を振っているIPAも、国家の一大事業である電子政府の実現に対して、政府CIOを技術面、実務面で支えるなど、より直接的な役割を分担していくべきではないだろうか？

ソフトウェア工学のベストプラクティス

Capers Jones　著
富野　壽・岩尾　俊二　監修
水野　哲博・吉田　善亮　監訳

ISBN：978-4-320-09760-5
共立出版
菊判・672頁
定価9,135円（税込）
2012年7月刊

2049年頃におけるソフトウェア開発と保守の状況とその展望

まず、統計値や実践的なソフトウェア工学の研究に裏付けされた全9章・672ページに圧倒される。どの章からでも読み始めることができる。ベストプラクティスの全容を把握したければ第1章から、その一つひとつに関心があれば第2章から、ソフトウェア・ライフサイクルの各プロセスにおけるベストプラクティスに関心があれば、第4章以降からが良いであろう。

原本の発行は2010年であり、第3章の内容は著作時から40年後を展望した仮説である。時間が経過すれば可能になるというのではなく、展望する諸事項に関連する研究・試行・実践というものが今後もなされなければ成立し得ないものである。

- 開発が芸術から工学に基づくものに転換
- 要求獲得のために、ユーザはインタビュー以外の方法を受ける
- インテリジェントエージェントの実用化でエキスパートのアドバイスにより開発が進行
- 特許により保護された再利用のためのコンポーネント及び設計手法の活用により、アプリケーション機能の85%が再利用可能なコンポーネントからなる
- 再利用のために、既存のコードから設計情報等の抽出が可能
- 生産性は、2009年では10-15FP/月で、2049年には45-50FP/月に向上
- 欠陥除去率は、2009年では87%、2049年では98%に向上
- 開発に100%再利用可能コンポーネントが用いられた場合、生産性は100FP/月、欠陥除去率は99%に向上　（新谷 勝利）

リーン・スタートアップ

エリック・リース　著
井口　耕二　訳
伊藤　譲一　解説

ISBN：978-4-8222-4897-0
日経BP社刊
四六判・408頁
定価1,890円（税込）
2012年4月刊

日本企業が今すぐに取り組むべきプロセス

日本経済が元気のない状況において、本書で紹介する方式は非常に有効と考える。とくに日本企業が得意としてきた生産方式をベースとしている点で親近感と興味がわく。

リーンスタートアップは、リーン生産方式を参考に構築された方式である。ソフトウェア開発では、アジャイル開発として近年注目を集め、国内でも実践事例が紹介されるようになっている。起業だけでなく、製品やサービスの開発も対象になる。ベンチャ企業だけでなく、大企業での製品やサービス開発も対象になる。

具体的には、会議室の中で繰り広げられる調査・分析の議論に時間をかけるのではなく、現地・現物が基本となる。アイディアの製品化、顧客の反応を見て学び・判断する。これを小さく早く回すことが求められるのだ。

この本を読んでいて、10年ほど前に聞いた楽天株式会社の三木谷浩史氏の発言を思い出した。「楽天が成功できたのは、インターネット上で市場を展開していることであり、間違えていたらすぐに作り直して対応できた」。製品化と学び・判断を早くまわした結果と言えるだろう。

組込み製品を含め、全てのシステムがインターネット接続される時代。この恩恵を活用した開発が求められる。

また、日本企業が得意としてきた改善による品質の作り込み。これにより改善が繰り返され、ユーザが求める、もしくは求める以上の品質を実現してきた。これからは、品質の定義・範囲を広げ、利用者が求める、もしくは求める以上の製品やサービスを提供することを実現したい。

（渡辺　登）

編集後記

クールビズも終わり、紅葉の季節もあっという間に過ぎ、いつの間にか寒さが身にしみる季節となりました。SEC journal 31号が出来上がりましたので、お届け致します。

今号の所長対談のお相手は、JISAの浜口会長となっています。JISAが掲げた「顧客従属型からパートナー型」、「国内競争から国際競争」など、5つの構造改革テーマを推進する中で感じておられること、技術者が心がけるべきことなどを忌憚なく話していただきました。

その中から、「エンジニアの能力の基本は、スペックを自分の中で構造化してコーディングまで持っていくこと」、「エンジニアにとって大切な能力は発想する能力」、「コンピュータにかかわるSE のビジネスランゲージはコード」、「データマネジメントはソフトウェア・エンジニアリングに関する重要なテーマ」の言葉が心に残りました。

この所長対談でもアジャイル対応の重要性が語られていましたが、「アジャイル開発」はSECが約4年間調査・検討を続けてきたテーマでもあり今号では、その適用領域、契約形態、技術者に求められるスキルなどについて多くのページを割いて詳しく調査・検討の結果を紹介しています。 (h-tanaka)

編集部より

次世代のソフトウェア・エンジニアリング等に関して、忌憚のない意見をお待ちしております。下記のFAXまたはメールにてご連絡ください。

SEC journal編集部　FAX：03-5978-7517
e-mail：sec-journal_customer@ipa.go.jp

SEC journal 編集委員会

札幌・北海道大学にて　（撮影：h-tanaka）

SEC journal® 第8巻第4号（通巻33号）　2012年12月14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 技術本部 ソフトウェア・エンジニアリング・センター　所長　松本隆明
Tel.03-5978-7543　Fax.03-5978-7517
http://sec.ipa.go.jp/

IT パスポート試験® のご案内

― IT を安全に最大限活用できる能力を身につけ、スキルアップ！ ―

- IT パスポート試験は、全ての職業人に必要な情報技術の基礎知識を問う**国家試験**です。合格者には経済産業大臣から合格証書が交付されます。
- IT 技術だけでなく、ストラテジ（経営全般）、マネジメント（IT 管理）など幅広い分野から出題します。
- IT を活用し、イノベーションが叫ばれる今、IT 社会で働く全ての方に求められる IT の基礎知識を証明できます。
- 企業における採用時の参考資格や社員の IT リテラシー向上を目的とした社員教育、また、教育機関における評価ツールとして活用されています。
- 試験の実施は CBT（Computer Based Testing）方式を採用。試験終了後、結果がすぐに分かります。また、試験前日までインターネットでお申し込みが可能です。

CBT 方式の特徴

結果がすぐに分かる
試験会場でコンピュータの画面上に表示される問題に解答。試験結果はすぐに分かります。

土曜・日曜日、夜間でも受験出来る
土日・夜間を含めた時間帯が選べるので、社会人・学生問わず受験が可能です。

受験会場は約 130 カ所
全国約 130 カ所の会場から受験場所が選べます（2012 年 11 月 30 日現在）。

お問い合わせ・お申込み
独立行政法人情報処理推進機構 (IPA)
IT パスポート試験コールセンター
https://www3.jitec.ipa.go.jp/JitesCbt/html/reference/reference.html
TEL : 03-5220-6736　FAX : 03-3216-7553

SEC journal 論文募集

独立行政法人情報処理推進機構　技術本部　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ています。

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

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

応募について
投稿は随時受付けております。応募様式など詳しくは HP をご覧ください。
http://sec.ipa.go.jp/secjournal/papers.html

バックナンバーのご案内

SEC journal バックナンバーの PDF をダウンロードいただけます。
詳しくは HP をご覧ください。
http://sec.ipa.go.jp/secjournal/

ESxR 特集号

No.26

No.27

No.28

No.29

No.30

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

SEC journal No.31
第8巻第4号（通巻33号）
2012年12月14日発行

ISSN 1349-8622