招待論文

ソフトウェアエンジニアリングの課題と展望

独立行政法人 情報処理推進機構（IPA）
ソフトウェア・エンジニアリング・センター
所 長 松田 晃一

1 ソフトウェア工学への実践的アプローチ

ソフトウェア工学は果たして工学として成立するのだろうか？言いかえれば，ソフトウェア工学がソフトウェア開発の現場を十分支えるだけの確固とした基盤を提供しているのだろうか？答えは残念ながら十分とは言えない。ソフトウェア工学の成果の中には，実際の現場において活用を敬遠されるものも多いことは，これまでに数多く経験してきたことである。

物理現象のように一定の原理・法則に支配される対象物であれば，その前提条件さえ整えば必ず同じ結果が得られ，つまり再現性がある。具体的な対象が別のものに変わっても，その原理・法則が成立する範囲であれば，意図した結果を再現させることができる。「再現性」こそ工学としての重要な要素の1つである。再現性があるからこそ，さまざまな実用的な問題に対して適用でき，社会に役立つ実学として成り立つ。一方，ソフトウェアは人間の知的活動によって生み出される人工物であるため，自然現象のような普遍の原理・原則の上に成り立っているものではない。では，ソフトウェア工学が工学として成り立つために「再現性」はどうやって担保できるのであろうか？

自然科学では仮説を客観的に検証するために実験という手段が使われる。ソフトウェアの場合は，さまざまな環境条件を制御して実験を繰り返し，データを取るということはほとんど現実的ではない。たとえできたとしても，きわめて小規模なトイモデルを対象とするのがせいぜいで，現実にある複雑で大規模なソフトウェア開発に説得力をもった結果として提示することはとても困難である。では，これを解決するには，どうすればよいか。現実のソフトウェア開発における実践例をできるだけ多数集めて，その中で比較的良い結果が得られた事例の中から，確からしい手法を類型化し，広く使える方法として一般化すること，逆に失敗の中から，その原因を掘り下げ，やってはいけない反面教師の事例として抽出すること，このような活動を積み上げることによって，多くのプロジェクトに効果が得られそうな，すなわち再現性の高そうな手法を体系的に集積していくことしかない。こうして得られた結果とそれを得るための過程がソフトウェア工学であろう。

筆者の属するソフトウェアエンジニアリングセンター（SEC）は，このようなアプローチを実践するために，2004年 10月 に IPA（Information-technology Promotion Agency）内に創立された。そして，40名弱の所員が，センター外の産学官の技術者や研究者と協力して，情報システムや組込みシステム分野におけるソフトウェア工学に関する活動を精力的に行っている。開発現場の実践例を持ち寄り，議論を通して多方面の角度からの分析によって，一般的，共通的な知見を得て，産業界の現場へフィードバックする活動である。ここでの活動は，正に上で述べたソフトウェア工学のアプローチの実践である。

本稿では，ソフトウェアの信頼性に焦点をあてて，SECにおける実践的な取り組みを中心に，ソフトウェアエンジニアリングの現状と課題について述べる。なお，筆者の力不足で，本来取り上げるべき重要な課題であっても必ずしもカバーできていない点については，あらかじめお断りしておく。ソフトウェア工学としての体系的な報告については，たとえばソフトウェア工学40年を記念した情報処理学会誌の小特集において，40年の活動をコンパクトにサーベイした優れた解説[1)]があるのでそれらを参照されたい。

2 ソフトウェアの信頼性

1968年にドイツにおいて「ソフトウェア工学」をテーマとする初の会議が開かれた。当時の問題意識として，「現代社会の中心的な活動にますます強く結びつきつつある情報システムの信頼性をどう確保するか，大規模ソフトウェア開発プロジェクトの納期を守り仕様を満たすことが難しい状況にどう対処したらよいか，ソフトウェア技術者の教育をどうすべきか」ということが背景にあったとのことである。これらは現在でもそのまま通用する問題であり，特に最初に挙げられている信頼性については，問題の大きさや深刻さはむしろいっそう拡大してい

るといってよい。情報システムは，企業活動は言うに及ばず，一般市民生活においても無くてはならない社会基盤となっており，その事故の影響は格段に大きくなっている。

また，これまでハードウェアで実現されていた機能がソフトウェアに置き換わり，いわゆる組込みソフトウェアの占める割合が飛躍的に増加し，一般消費者が日常的に利用する装置に広がってきている。このような状況の中で，もしそのソフトウェアに事故が起こると，多数の一般消費者に直接的に大きな影響をもたらすことになる。最近の自動車における電子制御ソフトウェアの不具合やパソコンや携帯電話などの電源制御ソフトウェアの不具合による発熱・発火事故などのように，ソフトウェアの事故が人身へ被害を与えるなど，事故の質的な変化をもたらしてきている。国民生活の安心・安全という観点から，ソフトウェアの信頼性・安全性の課題は，近年特に大きくクローズアップされてきている。

IT分野で常に先進的な米国の事情はどうであろうか？米国でもソフトウェアの信頼性向上は，近年特に関心が高まっており，なかでも米国が強みを有する軍事システムや宇宙航空分野，医療機器分野などにかかわる組込みシステムを，サイバーフィジカル・システム（Cyber-Physical System：CPS）として広くとらえ直し，これらの高信頼化に向けた省庁間連携の研究開発が進められている。また，重要インフラシステムに対しては，セキュリティを含む広義の信頼性確保の必要性が認識されており，その保護対策に取り組んでいる。

これらのソフトウェア高信頼化に向けた研究開発は，1998年に発足した高信頼ソフトウェアおよびシステム（High Confidence Software and System：HCSS）[13]と呼ばれる米国連邦政府の省庁連携ワーキング・グループに参加する政府機関を中心に実施されている。

このHCSSは，ソフトウェアの信頼性，安全性を向上させ，ソフトウェアの長期利用やシステム修復を可能にする技術の研究開発を行うことによって，ミッション・クリティカルなシステムの品質を向上させることを目的としている。そして，最終的には「国民を守り，消費者を守り，政府のサービスを向上させる」ことを目指している。

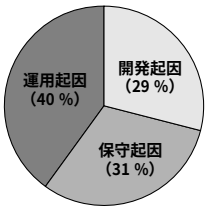
最近のHCSSの重点分野は，①サイバーフィジカル・システム（CPS）②リアルタイムシステム ③検証と妥当性確認（Verification and Validation：V & V）およびアシュアランス技術 ④複雑システムの4分野である。その中でCPSは，「コンピューテーショナル・リソースと物理的リソースの間にある密接な結びつきや連携」を意味すると広く定義され[14]，多くの項目に関連するものとして研究の重点対象となっているようである。

2.1 事故の発生状況と原因

事故対策の検討の第一歩は，事故の原因分析である。しかし，情報システムの事故の発生状況を正確に把握する仕組みが無いため，全体像をつかむことは困難であるが，報道を手がかりに原因の推定を試みると次のような点が観察される。

第1図は，2006年12月から2008年10月の間に報道された85の障害事例を「重要インフラ情報システム信頼性研究会」において分析した結果を示している[3]。開発段階に起因する原因で起きた障害が約30 %であるのに対して，運用や保守作業に起因して発生した障害がそれぞれほぼ同じ比率である。また，日経コンピュータによる2000年1月から2009年8月までの291事故事例の分析によれば，ソフトウェアの不具合が30.2 %に対し，うっかりミスが31.6 %，性能・容量不足が11.7 %，ハード故障などが16.5 %となっており[4]，開発に起因する原因よりも，保守や運用における要因が多くを占めていることを同様に示している。

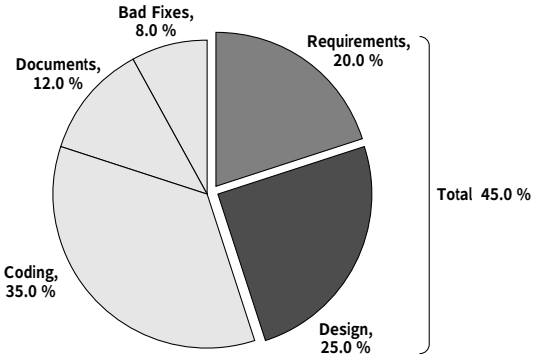
一方，ソフトウェア開発の問題に限定して分析した例を，**第2図**に示す[5]。ソフトウェア開発工程の中で，要件定義や設計段階での問題の方が実装段階よりも大きな比重があることを示している。上流工程での品質確保，特にユーザー要求を正しく認識・把握し，要件を定義して



原因の工程	原因例
運用	・ハード障害の切り替えミス ・データ容量上限値の設定誤り ・待機系の訓練で本番データ利用
保守	・確認テストで使用した環境の復元誤り ・ハード増設時のソフト組込みミス ・性能改善のためのファイル再配置ミス ・業務プログラム更新時の設定ミス

（重要インフラ情報システム信頼性研究会 報告書）

第1図 障害事例（06年12月～08年10月）85例の原因分析[3]



第2図 ソフトウェア故障の原因[5]

いく工程はソフトウェアの信頼性向上に重要なかぎとなるものである。

このようにソフトウェアの信頼性の観点からは，開発に先立つシステムの企画や要件定義の段階（超上流工程）および開発終了後の運用，保守の段階での取り組みが，（狭義の）開発段階に比べ相対的に重要になってきている。ソフトウェアの全ライフサイクルにわたって，技術的側面のみならず管理面，経営面，人的側面など総合的な対応が必要であることをあらためて認識しておく必要があろう。

2.2 ステークホルダーの責任分担と信頼性

前節で述べたとおり，ソフトウェアの信頼性の問題は全ライフサイクルにかかわる問題であるため，開発者のみならず，システムオーナーやユーザー，運用者など多くの関係者が役割を認識し，それぞれが責任を果たしつつ，協力してシステムの構築・運営に取り組むことが必要である。これまでのわが国のシステム開発は，一括請負契約によってベンダーに対して開発の一切を請け負わせる形が多く，システムのオーナー（発注者）として本来果たすべき役割をもベンダーに任せ切りにする傾向にあった。ユーザーはベンダーに任せたがり，ベンダーはユーザーが決めてくれないと文句を言いつつ仕方なく要求を想定して結果的にユーザーの期待とは異なったシステムを構築してしまうという状況も少なくなかった。欧米においては，ユーザーがCIO（Chief Information Officer）のもとPMO（Project Management Office）などの自組織で自らの責任において開発・調達を行うことが一般的であるため，このような問題はそれほど顕在化しないようであるが，わが国においては，ステークホルダーのこのような姿勢を正し，ユーザー・ベンダーのそれぞれの役割は何かという点について，共通の理解を進めることが必要であろう。

SECとしては，このような考え方について広く理解を得ることを目的に，「経営者が参画する要求品質の確保～超上流から攻めるIT化の勘どころ」[8]を取りまとめた。この中では，ステークホルダーがどのような役割と責任を果たすべきであるか，特にシステム開発の企画段階や要件定義段階において，どのような考え方で臨むべきかの原則を取りまとめている。また，その考え方を実行に移すに当たって，具体的に実施すべき作業項目や内容を定義したものとして「共通フレーム2007 第2版」[9]がある。これを参照することによって，システム開発に関係するステークホルダーの各工程における作業分担や成果物が明らかになる。作業を外部へ委託する場合に必要な契約にあたって，発注者と受注者のそれぞれが分担すべき作業項目を明確に規定するためにも活用されている。また，ソフトウェア開発にかかわる紛争において解決のための拠り所（よりどころ）としても利用されている。

2.3 要件定義と信頼性

要件定義は，ユーザーサイドと開発サイドの接点となる非常に重要な作業である。ユーザーからの要求が開発側に正しく伝わるか否かは，この要件定義の品質に大きく依存する。ユーザーの思いが正しく伝わらない限り，それ以降の開発の工程は無駄になり，プロジェクトの成功は期待すべくもない。要件定義におけるエラーが後のテスト工程や運用時まで持ち越され，その結果コスト増や納期遅延が発生することがウォータフォールモデルの抱える大きな問題であることはよく指摘される。たとえば，NIST（National Institute of Standards and Technology）の2002年のレポート[6]によれば，エラー全体の70 %は要件定義や設計段階に混入する。そして，その誤りの80 %の発見は統合テスト以降に持ち越されることを示している（**第1表**）。このように誤りの発見が下流工程に持ち越されると修正コストが急激に増大する。このNISTのレポートでは，設計段階でエラーを発見し修正する場合の労力を1としたときに，コーディング段階に持ち越した場合には5倍に，結合テスト段階では10倍，出荷後にまで持ち越して修正する場合には30倍もの労力がかかるとしている。このように，要件定義の品質は，開発プロジェクト全体の品質と生産性を大きく左右するものである。要件定義の誤りをいかに少なく高品質に行うか，もし誤りがあった場合にいかに早い段階で検出し修正するかが大きなかぎとなる。

第1表 エラーの発見と混入時期のモデル[6]

エラーの混入工程	エラーの発見工程					合　計
	要求収集・分析 アーキテクチャ設計	コーディング 単体テスト	統合 システムテスト	運用テスト ベータテスト	運　用	
要求収集・分析 アーキテクチャ設計	3.5 %	10.5 %	35 %	6 %	15 %	70 %
コーディング単体テスト		6 %	9 %	2 %	3 %	20 %
統合システムテスト			6.5 %	1 %	2.5 %	10 %
合　計	3.5 %	16.5 %	50.5 %	9 %	20.5 %	100 %

〔1〕機能要件と非機能要件

要件定義は機能要件と非機能要件に大別される。このうち，機能要件はシステムが果たすべき業務内容を示すものであるためユーザー自身が自分の言葉で示すことができる点で，比較的容易ではある。しかし，必要十分な要件を抜け落ちなく，あいまいさ無く定義できているか，その内容は開発者に誤解なく伝わり正しく設計に反映されているか，といった点は重要である。SECによる「機能要件の合意形成ガイドライン」[10]は，このような観点に着目し，機能要件に関して受発注者の間でスムーズに情報を共有し，機能要件を正しく反映した外部設計であることを双方で確認するための要点をとりまとめている。

一方，機能要件をシステム化する上での条件を規定する非機能要件は，業務内容に直結するものではなく，むしろ技術的内容を背景にするものが多いためユーザーにとって定義することが難しい。また，非機能要件の実現性には技術的な裏付けが必要であり，往々にして開発工程が詳細に進むまで先送りされる傾向が強く，そのために手戻りが生じる可能性も高い。これらを改善するために非機能要件の定義を手順化するための試みも行われている。ベンダー9社を中心とした検討会でまとめられた「非機能要求グレード」[15]はその一例である。

〔2〕要件定義の成功例

ここで，実際の開発プロジェクトにおける事例を紹介することによって，要件定義の重要性について具体的に示す。

それは，2010年当初にサービスを開始した東京証券取引所の新しい株式取引システム，アローヘッドの開発事例である。要件定義のエラーのうち，83 %がプログラミング前に発見されており，さらに開発途上で要件の実現が困難なために要件変更が必要となったもののうち，73 %が，同様にプログラミング前に発見された[16]，と報告されている。このデータは先に示したNISTのモデルから見ると，要件定義のエラー発見の工程が大幅に前倒しされている。つまり，上流工程の品質が大きく改善されたことが，プロジェクトの成功に大きく貢献したことが推定される。

このような要件定義の成功に向けてどのような工夫が行われたかはここでは触れないが，「要件定義は発注者の責任で行うべき」ことを貫き，4000ページからなる要件定義書の作成，およびその確認のための総合テストを自らの責任で実施したこと[11]，つまり発注者としての本来の役割を果たしたことがプロジェクトの成功の大きな要因であったことは明らかである。上流工程での品質の作り込みにおいて発注者の果たすべき役割と責任は大きい。

2.4 形式手法と信頼性

形式手法によるソフトウェア開発は，対象システムを数理的手法で厳密に記述することによって高品質のソフトウェアを効率よく開発することを目的としたものである。30年以上にわたってヨーロッパを中心に研究実用化が進められ，実用システムの開発に適用した事例も多くあるが，一般的な開発技術として開発現場に導入されるという状況にはまだない。しかし，今後は高信頼システムの開発に有力な方法の1つとして，また高信頼システムに要求される国際的な基準への対応のためにも，形式手法の開発現場への導入を促進していく必要があろう。

〔1〕信頼性の国際基準と形式手法

ISOやIECなどの国際標準化規格の中においては，安全性あるいは信頼性に関する基準として形式手法の利用が盛り込まれている。たとえば，情報セキュリティ国際評価基準を規定しているISO/IEC 15408では，軍や政府機関用の最高度のセキュリティレベル（EAL7）を要求されるシステムでは，形式的記述言語を用いた設計と分析・テストが求められている。また，その次に高いセキュリティレベルEAL5〜6においては，準形式的表現を用いてモジュールレベルまでの設計を行うことが求められている。また，電子機器の機能安全に関するIEC 61508やその自動車分野向けの規格であるISO 26262などにおいても，最高レベルの安全性を求めるシステムでは，形式手法の利用が強く推奨されている。その他，ヨーロッパでは鉄道信号システムなどに関する安全性技術指針，米国では航空機搭載システムの開発ガイドラインDO-178Bなどが制定され，その中でも形式手法の利用に言及あるいは必須とされている例もある。

このような動きは，今後その他の工業製品やシステムにも拡大することが予想され，日本の製品の国際競争力を高めていくためにも，このような基準への対応に向けて形式手法のような高信頼設計技術への対応を急ぐ必要がある。

〔2〕形式手法の導入シナリオ

形式手法は，その名前から受ける印象や手法に対する誤解などによって，現場からは敬遠されてきたきらいがある。現場への導入に際しては，理想的な形式手法による開発方法の適用を最初からねらうのではなく，たとえば，高い信頼性を保証する部分に限定して適用し，他の部分は従来の開発法を用いるというような組合せや設計書の記述など，特定の開発プロセスに部分的に適用するなどの方法が現実的である。

Bowen[7]らは，形式手法の利用に関してレベル0から2の三段階を示している。レベル0は，数学的な記法を用いて厳密に仕様を記述するレベルであり，証明や分析まで

は行わない。レベル1では，形式的開発および検証までを行い，さらに機械支援によるプログラムの性質の証明までを実施するのがレベル2である。

この中でレベル0の利用であっても，対象システムを厳密に記述することによって，わかりやすいシステム記述が得られ，仕様記述の不十分さや曖昧（あいまい）さに起因する多くの問題が開発の早い段階で明らかとなり，システムの品質と開発効率を向上させることに寄与する。わが国におけるモバイルFeliCa®（注） ICチップの開発プロジェクト[17]は，このレベルの例である。

形式手法において主要な役割を果たすのは形式的な仕様記述である。仕様が曖昧さや不整合がなく厳密に記述されていれば，それだけで，効率的で高品質のシステム開発を行うことに役立つ。開発者が普段行っている仕様書の作成やプログラミング，テストなどにおいて厳密な記述や網羅的な手法を導入して，形式手法の考え方や方法論に慣れていった上で，より高度なレベルの形式手法の適用へ進むというのは1つの現実的なシナリオであると言える。

2.5 コストと信頼性

わが国の製品は，高品質，高信頼であることが競争力の源泉になってきた。この状況は情報システムや組込みシステムについても同様であり，たとえば情報システムの信頼性についての他国との比較（**第2表**）において，日本のシステムは高い信頼性を示している[12]。これは，SECが発行しているソフトウェア開発データ白書[2]の信頼性の分析データ（**第3表**）を見ても，同様に高い信頼性を達成していることがわかる。

第2表　ソフトウェア開発データ[12]

	インド	日本	米国	欧州他	計
プロジェクト数	24	27	31	22	104
生産性*1	209	469	270	436	374
不具合密度*2	.263	.020	.400	.225	.150

*1．新規ソースコード行数／人月
*2．出荷後12箇月間に顧客から報告された不具合数／総ソースコード行数

第3表　開発言語別発生不具合密度[2]

開発言語	プロジェクト数	発生不具合密度*1（件／KSLOC）	
		中央値	平均値
COBAL	48	0.034	0.071
C	35	0.012	0.054
VB	42	0.026	0.105
Java	113	0.011	0.108

*1．システム稼働後6箇月間に発生した不具合数／総コード行数（KSLOC）

（注）ソニー（株）の登録商標

一方，わが国においては一般ユーザーや消費者の製品の品質に対する要求は非常に高く，情報システムの障害によるサービスの停止に対する反応は概して厳しい。このような高い要求に応えるため，必要以上にコストをかけて品質を確保するなど，過剰品質・過剰投資による社会的ロスも増大していることも事実である。

情報システムの信頼性を向上させるためには，相当の開発コストや対策コストが発生するため，限られた経営資源において，信頼性の確保とコスト低減はトレードオフの関係にある。今後は両者の間で適切なバランスを保つことが重要であることについて，社会の共通認識を形成することが必要である。まず，どのような情報システムがどの水準の信頼性を具備するべきか，その信頼性対策とコストとの適切なバランスはどうあるべきかなど，適切な信頼性の在り方について社会的なコンセンサス作りが重要である。

このような課題についての議論の端緒として，事故が発生したときの影響度に応じて情報システムを4つのタイプに区分し，それぞれの区分に応じて，企画から開発，運用，保守のそれぞれの段階で整合のとれた信頼性対策を行う枠組みが提案されている[3]。このような枠組みの実用性や各区分に対応した対策の妥当性，適切な定量的な品質目標値の設定など検討すべきことは多いが，信頼性とコストに関する社会的コンセンサス作りを目指して，この枠組みの具体化を進める必要がある。

3 むすび

本稿では高信頼化の話題を中心に，いまのソフトウェア開発現場におけるソフトウェア工学の課題について紹介した。この他に，組込みソフトウェアにかかわるさまざまな問題やクラウドを活用した開発，アジャイル開発などの新しいソフトウェア開発トレンドへの対応，人材育成など，本稿では触れることができなった多くの項目もこれからの重要な課題である。

ソフトウェア工学という言葉が生まれて40数年。現場での課題はいっそう大きく立ちはだかり，ますます複雑で困難な問題が山積している。「銀の弾丸はない」ことを肝に銘じて，現場の問題の一つ一つに真摯（しんし）に取り組んでいくことがわれわれの進むべき道であろう。

参考文献

1）玉井哲雄：ソフトウェア工学の40年　情報処理 49, No.7, pp.777-784 (2008).
2）IPA/SEC : ソフトウェア開発データ白書2009 (日経BP社).
3）経済産業省，独立行政法人 情報処理推進機構，（社）日本情報システム・ユーザ協会 : 重要インフラ情報システム信頼性研究会報告書 (2009.3). http://sec.ipa.go.jp/reports/20100427.html (参照2010.9.5).
4）日経コンピュータ 2009年8月19日号.
5）SOFTWARE QUALITY IN 2008 : A survey of the state of the art. Presentation by Software Productivity Research LLC at JaSST Japan.
6）The economic impacts of inadequate infrastructure for software testing. NIST Planning Report 02-3 (2002.5).
7）J. P. Bowen, et al. : Ten commandments of formal methods. IEEE Computer 28, No.4, pp.56-63 (1995).
8）IPA/SEC : SEC BOOKS 経営者が参画する要求品質の確保 ～超上流から攻めるIT化の勘どころ～ 第2版 （オーム社）(2006).
9）IPA/SEC：SEC BOOKS 共通フレーム2007 第2版 ～経営者，業務部門が参画するシステム開発および取引のために～ （オーム社）(2009).
10）http://sec.ipa.go.jp/reports/20100331.html (参照2010.9.5).
11）田倉聡史：上流工程での品質確保のための発注者責任　SEC journal 21（独立行政法人 情報処理推進機構） pp.94-99 (2010).
12）Cusumano, M., et al. : A quantitative analysis of U.S. and Japanese practice and performance in software development. IEEE Software Nov/Dec, pp.28-34 (2003).
13）http://www.nitrd.gov/Subcommittee/hcss.aspx (参照2010.9.5).
14）http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=50328 (参照2010.9.5).
15）http://sec.ipa.go.jp/reports/20100416.html (参照2010.9.5).
16）鈴木義伯：東証Arrowhead （次世代システム）の開発と要求実現プロセス　第4回要求シンポジウム講演資料 (2010.3).
17）栗田太郎 : 携帯電話組込み用モバイルFeliCa ICチップ開発における形式仕様記述手法の適用　情報処理 49, No.5, pp.506-513 (2008).

《プロフィール》

松田 晃一（まつだ こういち）

1968	京都大学 工学部電気工学科卒業
1970	京都大学大学院 工学研究科電子工学専攻 修士課程修了
1970	電電公社（現日本電信電話（株））入社
1995-1998	NTT コミュニケーション科学研究所 所長
1998-2000	NTT 先端技術総合研究所 所長
2000	大阪大学 博士（工学）
2000-2006	NTTアドバンステクノロジ（株）常務取締役
2006-2008	NTT-AT IPシェアリング（株）代表取締役社長
2006-2008	電気通信大学 理事
2006-2008	東京大学 客員教授 （国際・産学共同研究センター）
2008-2009	情報処理推進機構 IT人材育成本部長
2009-現在	情報処理推進機構ソフトウェア・エンジニアリング・センター 所長

日本学術会議連携会員，情報処理学会フェロー，
電子情報通信学会フェロー

専門技術分野：
ソフトウェア・エンジニアリング，オペレーティングシステムの研究開発，システムの性能評価の研究

主な著書：
情報流通を支えるコミュニケーション科学（共著）（電気通信協会，2001）
ITが地球環境を救う―「情流」がもたらす環境革命（監修）（ダイヤモンド社，2003）
eセキュリティ ― IT時代のリスクマネジメント（監修）（ダイヤモンド社，2001）