2. アウトサイドイン指向ソフトウェア工学に向けて

山本修一郎　（株）NTT データ 技術開発本部　システム科学研究所

今後のソフトウェア工学の展望としてアウトサイドイン指向ソフトウェア工学について述べる．アウトサイドイン指向とは，コミュニティの外部にいる顧客への価値提供を目的として製品やサービスを開発する活動のことである．特定の開発手法やツールを指すのではなく，研究開発を進める上での基本的な考え方や姿勢を指す．ソフトウェア工学の研究コミュニティの中で生まれた成果をソフトウェア開発現場に導入するというインサイドアウト指向の技術移転では一方通行のコミュニケーションになる．アウトサイドイン指向ソフトウェア工学では，現場が必要とするソフトウェア開発技術をダイナミックに再構成して供給する．

ソフトウェア工学の顧客はだれか

ソフトウェア工学が約 40 年前に必要に迫られて生まれた時代には，手探りでソフトウェア開発作業を進めながら，現場の経験を積み重ねることで，仕様化，設計，製造，試験などのソフトウェア開発のプロセスが定義された．この過程で，構造化手法，オブジェクト指向手法などの方法論やソフトウェア成熟度モデル（CMM, Capability Maturity Model）などのように制度化され，ソフトウェア産業界に大きな影響を与えたソフトウェア工学の成果も生まれた．

筆者は 1979 年に電信電話公社の横須賀電気通信研究所のプログラム言語処理研究室に入所し，日本のソフトウェア工学研究の黎明期に先導的な役割を果たされた故花田収悦室長の下で DIPS（Denden-kosha Information Processing System, 電電公社仕様のメインフレーム・コンピュータ）上のプログラム言語処理系の開発を経験した．DIPS ソフトウェア作業標準と呼ばれる標準的な作業工程や問題管理票などのドキュメンテーションの体系を学んだ．これが実用的なソフトウェア開発に参加した最初の経験だった．また当時のプログラム言語処理研究室では，テスト網羅率測定ツールや表明チェッカをもつコンパイラを開発していた．このような現場活動の成果の 1 つが花田収悦編集責任による「ソフトウェアの仕様化と設計」[1] に代表される日科技連ソフトウェア品質管理シリーズだろう．約 20 年ほど前はソフトウェア工学の先端研究と開発現場がまだ一体化していた．

今ではソフトウェア工学も学問として確立し，教科書もたくさん出版され，ソフトウェアの開発現場にまで行かなくてもソフトウェア工学の研究が生産されるようになった．この間にソフトウェア開発現場では，これまでの開発経験の蓄積があり，現場に応じた方法論，ツールや開発標準が整備されてきた．ソフトウェア工学の研究が成功し現場で活用が進んだことの必然的な結果であるともいえる．

一方で，ソフトウェア工学が学問として確立されたことでそれを学生が大学でも学ぶことができるようになった．

ここでソフトウェア工学の顧客を整理しておくと，ソフトウェア工学を活用してソフトウェアの製造にかかわる関係者，新たなソフトウェア工学の知見を生産する研究者，そしてソフトウェア工学の学習者に分類できる．ソフトウェア工学の学生は潜在的な関係者や研究者でもある．

インドや中国，韓国などの大学では国策もありソフトウェア工学教育を充実させている．このようにインドや中国などの従来は発展途上国だと思われた国で IT 教育

	臨床の知 [4]	モード論 [5]	マネジメント論 [6]	アウトサイドイン指向ソフトウェア開発手法 [7]
提唱者	中村雄二郎	Michel Gibbons	Peter Drucker	Carl Kessler, John Sweitzer
概要	人間存在の多面的な現実に着目した学問の方法	問題設定がディシプリンの枠内にある（モード 1）とアプリケーションのコンテクスト内にある（モード 2）との知識生産の方法を比較	顧客を明確に識別することで，顧客価値向上を可能にするイノベーションの方法	関係者とそのコンテクストに着目したソフトウェア開発手法
問題認識	科学の知が普及し信頼されすぎたことにより，それに適合しない領域が顕在化した	科学技術，社会科学，人文科学における知識生産の方法の変化を明らかにする	あらゆる組織が内部のことに忙しく，顧客のいる外部の世界に焦点を合わせることができないでいる	ソフトウェアが埋め込まれる製品やビジネスプロセスが使用されて初めて価値を生む．ソフトウェアを開発しただけでは成功とはいえない
主要概念	個々の場合や場所を重視して深層の現実にかかわり，世界や他者が示す隠された意味を相互行為のうちに読み取り捉える	社会や産業応用を目的として問題を設定し，多様な既存のディシプリンの参加とそれらを超越したトランスディシプリンな枠組みを設定することでダイナミックに問題を解決する	企業の目的は顧客とともに企業の外にある．企業が何であり，何を生み出すかを規定し，企業が成功するか否かを左右するものは顧客である	関係者とそのコンテクストを理解し，関係者が導入容易な（consumable）ソフトウェアを作成し，関係者のゴールと整合させる

表 -1　科学技術と社会の相互関係の方法

が充実し，これらの国が IT 分野で国際競争力を持つようになったのは，まさにソフトウェア工学という学問がグローバルに制度化された成功例でもある．

IT 教育を受講した学生を採用する海外企業との競争という点から，日本でも産学連携による実際的なソフトウェア工学教育が必要とされるようになった．文部科学省ではこのような要請を受けて国内で先導的 IT スペシャリスト育成推進プログラムを開始した．たとえば名古屋大学では「我が国における IT 教育の問題点は，基礎技術の教授のみにとどまり，要素技術の統合や実践教育が企業にゆだねられていることだ」と指摘し OJL（On the Job Learning）による IT 人材育成拠点を形成している [2]．ベトナムにも日本政府の援助でハノイ工科大学に日本語が話せる IT 技術者養成プログラムが設置された．

このように 20 世紀のソフトウェア工学は産業界に着実に影響を与える一方で，ソフトウェアは現実社会に大きな影響を与えるようになった．日科技連ソフトウェア品質管理シリーズの刊行のことばの冒頭には「1980 年代は情報化社会の時代，特にその中心となるソフトウェアの時代だ」と述べられている．Boehm は最近の IEEE Computer 誌上の記事で 21 世紀をソフトウェアの世紀だと指摘した [3]．ソフトウェアは情報化社会の中心に限定されるのではなく，社会全体にまで浸透したといえよう．21 世紀のソフトウェア工学はソフトウェア工学の内部にとどまるのではなく，ソフトウェア社会とソフトウェア産業の現実に着目した問題を設定し研究していく必要があると思われる．

アウトサイドイン指向ソフトウェア工学とは

社会や産業の現実に着目した問題を設定し研究していくという考え方は新しいものではない．**表 -1** に示したように，哲学における「臨床の知」，知識生産様式における「モード論」，経営学における「マネジメント論」，ソフトウェア開発における「アウトサイドイン指向ソフトウェア開発手法」などでも同じように指摘されている．以下ではこれらの考え方とアウトサイドイン指向ソフトウェア工学の関係について考えてみる．

■ 臨床の知

中村雄二郎は人間存在の多面的な現実に着目した学問の方法として従来の科学の知に対して臨床の知を提唱した [4]．この理由は，科学の知が普及し信頼されすぎたことにより，それに適合しない領域が顕在化したためである．臨床の知では，「個々の場合や場所を重視して深層の現実にかかわり，世界や他者が示す隠された意味を相互行為のうちに読み取り捉える」ことで，新たな知を創造するのである．

既成の理論や学問と現実とのずれが大きくなっている原因は，近代科学が前提とした，普遍性，論理性，客観性という 3 つの原理にある [4]．近代科学はこの 3 原理にそぐわない現実の側面を無視し排除したことから，学問と現実のずれが生じたというのである．

近代科学が無視し排除した現実の側面を捉えなおすためには，普遍性，論理性，客観性に対して個別性，多義性，身体性という新たな 3 原理を示し，これらを統合した「臨床の知」が必要になる．

- 個別性：他とは異なる固有の存在としての個々の場所や時間に着目する
- 多義性：一義的な因果関係に基づく論理性ではなく，事物の多義的な関係を考慮する
- 身体性：事象を主観と切り離すのではなく，行為を通じた相互作用として事象を捉える

つまり，最初から普遍的で論理的に説明できる客観的な理論や学問の存在を仮定するのではなく，まず現実を観察してそこにある問題を問うということだと思われる．

またPlatonは次のようなパラドックスを紹介している[5]．

> すべての研究は問題から出発しなければならない．しかし，問題に対して解答を求めることは不合理である．

この理由は，もし問題が見えていなければ，探しているものが何かを知らないのだから解答を求めることはできない．逆に解答が見えていれば，それを改めて研究する必要はない．したがって研究すべき問題は存在しない．

このパラドックスに対するPlatonの解決は，発見とは過去の経験を想い出すことだという．つまり実践的な経験が新たな発見の根拠を与えているということだ．

「臨床の知とは何か」の冒頭でこういうチェーホフの小説が紹介されている．

> ある控えめな人のための祝賀会が催された．参加者たちはこのときとばかりに自慢話や互いの賞賛に花が咲き，時の経つのを忘れた．ところが祝賀会が終わる頃になって参加者たちが気づいてみると，当の主人公を招待することを忘れていた．

「ソフトウェア工学の成果」がこの祝賀会の参加者で，参加者から招待されるのを忘れられた控えめな主人公が「ソフトウェア産業・社会の現実」ではないと果たして言いきれるだろうか．ソフトウェア産業・社会の現実に即した「臨床の知」が求められている．

■ モード論

Michel Gibbonsらは問題設定がディシプリンの枠内にあるモード1とアプリケーションのコンテクスト内にあるモード2という知識生産の方法を比較している[6]．この背景には，科学技術，社会科学，人文科学における知識生産の方法が新たな変化に直面しておりそれを明らかにすることが現代社会における知の創造に求められているという姿勢がある．

モード2の知識生産では，社会や産業応用を目的として問題を設定し，多様な既存のディシプリンの参加とそれらを超越したトランスディシプリンな枠組みを設定することでダイナミックに問題を解決するのである．

モード1ではディシプリンの内部で生産された成果は技術移転のリニア･モデルで需要側に伝達される．モード2ではディシプリン内部だけでなく産業界も含むメンバが構成する技術インターチェンジの場でメンバ全員が協調することにより技術の生産と交換がダイナミックに発生する．

ソフトウェア工学でもリニアなウォーターフォール・モデルに対して，ダイナミックなスパイラル・モデルやアジャイル・モデルが提唱された．ソフトウェア工学の成果を産業界や社会に伝達することを考えれば，インサイドアウト指向のリニア・モデルではなく，ダイナミック･モデルとしてのアウトサイドイン指向が必要である．

■ マネジメント論

Peter Druckerは，顧客を明確に識別することで，顧客価値向上を可能にするイノベーションの方法を示した．彼は「あらゆる組織が内部のことに忙しく，顧客のいる外部の世界に焦点を合わせることができないでいる．企業の目的は顧客とともに企業の外にある」からアウトサイドインの考え方が組織には必要だとした[7]．「企業が何であり，何を生み出すかを規定し，企業が成功するか否かを左右するものは顧客である」．ここで企業を「ソフトウェア工学」に置き換えることができることに気づくだろう．アウトサイドイン指向ソフトウェア工学とは，Drucker流に言えば「ソフトウェア工学が成功するかどうかをソフトウェア工学の顧客の立場で評価し，ソフトウェア工学が何であるか，そして何を生み出すかを規定する取り組みである」ということになる．

このようにアウトサイドイン指向ソフトウェア工学では，ソフトウェア工学を活用する顧客を明確にして顧客の立場で必要な技術を開発する．そのために顧客にとって価値を生むソフトウェア工学上の問題を収集し解決することがアウトサイドイン指向ソフトウェア工学では重要になる．

アウトサイドイン指向ソフトウェア工学ではDruckerに従えば最も重要なものは顧客にとっての成果である．顧客のいる外側から内側を見るアウトサイドインの姿勢を可能にするためには顧客のいる外部の情報を収集し解釈する能力が必要になる．

■ アウトサイドイン指向ソフトウェア開発手法

Carl KesslerとJohn Sweitzerは関係者（Stakeholder）とそのコンテクストに着目したアウトサイドイン・ソフトウェア開発手法（OID, Outside-In software Development）を提案している[8]．

OIDの背景には，ソフトウェアが埋め込まれる製品やビジネスプロセスが使用されて初めて価値を生む．ソ

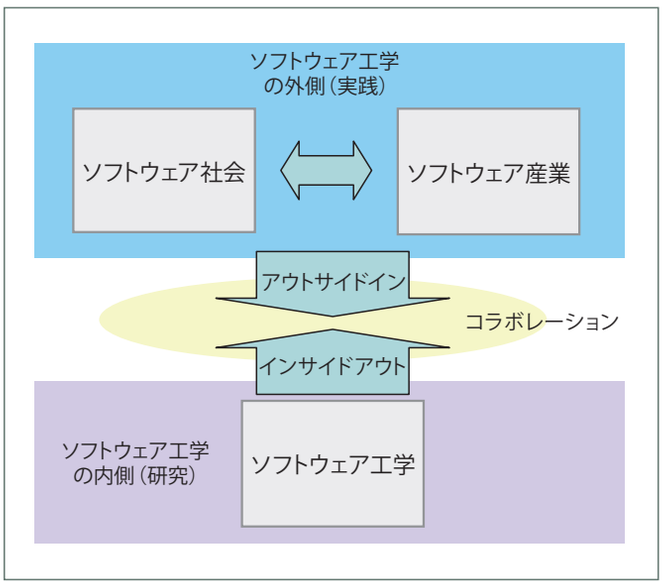

図 -1　ソフトウェア工学とソフトウェア社会，ソフトウェア産業の関係

フトウェアを開発しただけでは成功とはいえないというDruckerのアウトサイドインの考え方と共通する姿勢がある．

OIDでは関係者とそのコンテクストを理解し，関係者が導入容易な（consumable）ソフトウェアを作成し，関係者のゴールと整合させることで成功するソフトウェア開発を目指すのである．

◆ 顧客（Stakeholder）の理解

エンドユーザ，購買責任者，などソフトウェア製品販売を成功させるために満足させるべき顧客に焦点を当てた開発が必要である．

◆ 組織コンテクストの理解

顧客がソフトウェアによって変革したい組織コンテクスト，効率化，柔軟性，競争優位性を理解し，ソフトウェアがこれらのコンテクストに合致することを確認する必要がある．

◆ 導入容易性（consumability）

ソフトウェア製品を顧客の環境に導入するために必要となるソフトウェアを取り巻く作業を実施する必要がある．このような購買，インストール，展開，運用への統合，問題管理，バージョンアップなど，困難，時間がかかり，退屈，非効率，煩雑な作業をメタタスクと呼ぶ．ソフトウェア開発だけではなく顧客の運用環境への導入作業を軽減することを最初から考慮すべきであると指摘する．メタタスクを用いて顧客の導入障壁を最小化する．

◆ 関係者ゴール整合性

顧客の目標は外部要因によって常に変化するからそれを監視してソフトウェアを顧客のビジネスゴールに整合性を持つように変更していく必要がある．

◆ 関係者による成功の確認

アウトサイドイン開発手法は顧客指向であるから，顧客の言葉で成功条件としての評価尺度と客観的な数値を定義しシステム移行を準備するとともに事後分析を実施する．

以上述べたことをまとめると次のようになる．

◆ アウトサイドイン指向ソフトウェア工学では，顧客としてのソフトウェア産業・社会の現実を観察する必要がある

◆ アウトサイドイン指向ソフトウェア工学の顧客の立場で技術を開発する必要がある

◆ アウトサイドイン指向ソフトウェア工学の成果を顧客の現実に適応する形で提供する必要がある

◆ アウトサイドイン指向ソフトウェア工学の顧客と研究者の技術インターチェンジを介したダイナミックな双方向の交流が必要である

アウトサイドイン指向ソフトウェア工学の可能性

ソフトウェア工学の顧客の例を考えたが，ソフトウェア産業だけではなく，ソフトウェアが浸透した社会（ソフトウェア社会と呼ぶ）では，政府や企業もソフトウェアを調達するという点では，ソフトウェア開発に参加することになる．このようにソフトウェア開発の境界が曖昧になっている．たとえば組込みシステムを考えれば分かるようにすでに多くの製造業が自前のソフトウェア開発部門を持っていることに気づく．将来のソフトウェア産業の中心が製造業になっている可能性もある．

このようなソフトウェア社会，ソフトウェア産業，ソフトウェア工学の関係を**図 -1** に示した．ソフトウェア工学にはすでに示したようにソフトウェア工学に内在する問題を解決しようとするインサイドアウト指向とソフトウェア社会・産業の現実にある問題を解決しようとするアウトサイドイン指向の研究がある．この図で示すようにこの2つのソフトウェア工学研究はどちらかがあればよいというものではなく，両者のコラボレーションが必要である．すでに述べた臨床の知，モード論，マネジメント論，OIDでもこの点は同じである．学問が発展することで知識生産の場が拡散するのである．そういう意味でソフトウェア工学も40年を経て成熟期に入ったといえよう．

Boehmは**表 -2** に示すようなソフトウェア工学の5課題を示している[3]．21世紀のソフトウェア社会ではこれまで以上に多くの関係者がソフトウェア開発に参画することになる．いままでのようにはソフトウェア工学の成果をこれらの拡散した関係者に提供することは容易ではなくなると予想される．また関係者が多様化する

課題	
急速な変化	CMMI レベル 5 継続的最適化では過剰適応のリスクがあるため，発注仕様に基づく開発契約ではなく，運命共同体型契約が米国では受容され始めた．どのように知識を習得するかを学習することが重要になる．
不確実性と新技術の出現	GPS や WWW など予測不可能な技術の出現に対応する必要がある．リスク低減のための適応性が組織，製品，プロセスの構造に求められる．
ディペンダビリティ	ユーザ，発注者，開発者，保守者の間でのディペンダビリティに対する要求の矛盾が，システムの欠陥原因である．このため，すべての関係者間で意図の矛盾を解消するためには，目的やゴールという利用者の用語を用いた合意形成と，開発者も利用者側の背景知識を持つ必要がある．
多様性	組織文化，ユーザインタフェースなど普遍性が通用しない文化的差異が各国にある．
相互依存性	独立なシステムはどこにもなく，すべてのシステムが一体となって ECOSYSTEM を構成する．人間工学，ハードウェア工学，ソフトウェア工学の統合が必要である．

表 -2　ソフトウェアの世紀の課題

のと同じようにソフトウェアの利用環境や開発環境も多様化する．最初からこれらの関係者や利用環境，開発環境の変化と多様化に対して適応可能なソフトウェア工学とは何かを考えることが大切になる．これらの課題に挑戦するためには以下に示すようにソフトウェアの利用環境や開発環境を含めたより広い問題に対する研究が必要であり，それはアプリケーションのコンテクストに応じた問題解決というアウトサイドイン指向ソフトウェア工学を実践することでもある．

- ◆ 社会が急速に変化するため，開発対象ソフトウェアの仕様を定義してから発注していては変化に対応できなくなるので，運命共同体型契約に基づくソフトウェア開発プロセスが求められるようになる．発注者と開発者がソフトウェアを長期的に協調開発するためには開発者側が発注者側の知識を獲得していく必要がある．
- ◆ 予測不可能な技術の出現に対応するための適応性がソフトウェア開発組織，ソフトウェア製品，ソフトウェア開発プロセスの構造に求められる．
- ◆ システムのディペンダビリティに関する欠陥原因は，ユーザ，発注者，開発者，保守者の間でのディペンダビリティ要求の矛盾にある．このため，すべての関係者間でディペンダビリティ要求の矛盾解消のための合意形成と開発者による利用者側の背景知識の理解が必要になる．
- ◆ ソフトウェア開発がグローバル化するということは，組織文化，ユーザインタフェースなど各国にある普遍性が通用しない文化的差異に応じた個別的な取り組みとその統合が必要となる．
- ◆ ユビキタスサービスに代表されるようにコンピュータが社会に浸透することにより，独立なシステムはどこにもなく，すべてのシステムが一体となって，自然界で多数の生物が相互作用するのと同じようにシステムの生態系（ECOSYSTEM）を構成するようになる．

このため人間工学，ハードウェア工学，ソフトウェア工学など異なるディシプリンの統合も必要になる．

それではどのようなコラボレーションの可能性があるだろうか？　つまりソフトウェア社会や産業から，現実のソフトウェア問題をソフトウェア工学に持ち込むにはどうしたらいいか？

■ 産学連携の共同研究

これまでも個別的な共同研究は大学と企業でやっているが，よりオープンな産学連携のためには知財問題を解決する必要がある．企業や大学ごとに知財契約の考え方に差異があり，知財契約手続きの煩雑さからコラボレーションが進まない可能性もある．少なくとも論点整理の段階ではオープンで効率的な議論ができるような枠組みが欲しい．

■ 人材交流によるコラボレーション

優秀な人材が大学に移動することで，現場感覚のある問題指向研究が成功する可能性が高い．たとえば Hoare は大学卒業後しばらく産業界で働いており，表明式（assertion）とプログラム証明におけるその役割がそのときの経験に触発されたものであると述べている [9]．また当初はプログラムの正当性の証明を意図して研究開発した表明式が，実際にはプログラムの挙動を診断するための手法として産業界で使われることになった．これは問題と解答は最初から明確には対応していない例である．

このように開発者が研究職を得ることで，問題と解決策が産業界から研究所に持ち込まれることが多い [9]．現在 Hoare はマイクロソフト社の研究所に所属し，プログラム開発における表明式の現代的な役割を再び産業界で研究している．優秀な研究者が産業界と大学を渡り歩くことで，産業界の実践的な問題が大学に持ち込まれるだけでなく，大学で研究された成果が産業界にもたらされ，継続的な研究交流が進むのであろう．Hoare はレガシーコードを扱う開発者も含めて現場のソフトウェア開発者に役立つプログラミング理論に研究成果を提供することを期待している．きわめて現実的な姿勢である．

最新の理論的な成果を直接，開発現場に持ち込むのは容易ではない．すぐに思いつくのはレガシーコードを捨ててすべて作り直すという考え方である．これはインサイドアウトの考え方である．現状を否定するのではなく観察してその上で必要な技術は何かを問うアウトサイドインの姿勢が必要である．また博士課程の学生や社会人ドクターなどの場合も大学の研究と企業の現場の問題を切り離すのではなく，知財権問題を上手に扱うことでアウトサイドイン指向の研究ができる可能性が高い．逆に大学の研究者が企業に移動することによって最新のソフトウェア工学の成果をソフトウェア産業に持ち込んで現実の問題を解決できる．

また国立情報学研究所のトップエスイープロジェクト（http://www.topse.jp/）では，企業の人材に最新のソフトウェア工学の成果を教育している．この教育を受けた成果を用いて，現実のソフトウェア開発課題を解決することで，新しいソフトウェア工学の知が現場で生まれる可能性がある．

■ 官の支援による産学コラボレーション

米国では，国防総省（DOD, Department of Defense）が出資したカーネギーメロン大学のSEI（Software Engineering Institute）がある．SEIではCMMI（Capability Maturity Model Integration）を事業化して成功した．SEIにおける研究はDODを顧客とするミッション指向であり，アウトサイドイン指向ソフトウェア工学を実践しているといえよう．

日本では情報処理推進機構（IPA）のソフトウェア・エンジニアリング・センター（SEC, Software Engineering Center）の例がある．SECでは数多くの部会活動があり，産学から有識者が集まり，オープンなコミュニティで一緒になって現実の問題についての検討を進めている．

■ 大学における教育プログラム

先に紹介した文部科学省の先導的ITスペシャリスト育成推進プログラムでは，産業側からの実践的な教材の提供，実務経験豊富な講師の派遣，インターンシップ制度による企業での学生によるソフトウェア開発の実務体験などが進められている．実践的なソフトウェア工学を学んだ学生が企業や研究機関でソフトウェア社会・産業の問題解決に工学的に取り組むことが期待される．

以上述べたようにアウトサイドイン指向ソフトウェア工学に取り組む環境が徐々に整備されつつある．

次にソフトウェア開発現場を対象とするアウトサイドイン指向ソフトウェア工学の取り組みを紹介しよう．

開発現場を顧客とする取り組み

以下では，(1) これまでのソフトウェア工学でも認識されていたが現在も開発現場で苦労している要件レビュー，(2) ソフトウェア工学コミュニティと産業をつなぐ技術インターチェンジ，(3) これまでのソフトウェア工学では考慮されていなかったソフトウェア産業における労働ストレス解消，についてのアウトサイドイン指向ソフトウェア工学の取り組みを紹介する．

■ 要件レビュー

花田収悦は「要求分析の終了時点は，なかなか定めにくい」と文献1）のあとがきで述べている．たとえばIEEE std-830などでも「要求が正しいことを確認する」という程度しかガイドラインとして提示されていない．要求が正しいかどうかの判断は要件レビュー参加者に依存するのだ．最近もあるところで「要件レビューを実施して要件変更管理表を作成したので，要件定義工程の完了判断をしたいのだがどうすればいいか」と聞かれた．

要件レビュー能力と要件管理の精度／品質を高めるためにレビューでの指摘事項や，要求変更／修正を，要件定義の構成要素として捉え，各構成要素のどこに問題があったかで分類し，要件定義に関する誤りを構成要素ごとに確認する考え方を整理する必要がある．

たとえば要件定義①依頼元アクタ②依頼元アクタ状況③イベント④入力⑤処理⑥出力⑦結果状況⑧結果アクタという8つの構成要素からなると考える．これまでは，このうち④入力⑤処理⑥出力に注目して機能仕様を作っていた．要件レビューでは，その周りにある「誰のための機能か」「どういう状況でその機能を使うのか」「その状況でどんなイベントがあるのか」ということまで押さえる必要がある．

あるプロジェクトにおける要件変更票をこの分類で分析してみたところ，8つの構成要素で要件変更理由の約90%を説明できた．残りはドキュメント誤り（約7%）と記述不足のため理由を特定できなかった項目（約3%）であった．記述不足項目については，内容が明確化できれば，8つの構成要素に分類できたと思われる．したがって，もし新たな要件変更があるとすればこの8つの要素のどれかである可能性が高い．逆にいえば，この8要素のすべての組合せを網羅して必要な組合せを抽出できれば要件定義工程を完了できることになる．これは開発現場での問題解決の例である．今後要件レビュー手法として洗練していく必要がある．

■ 高信頼性システム開発手法

2007年から始まったIPA-SECの高信頼性システム開

発手法調査検討会は社会的重要インフラシステムの高信頼化を図るための開発手法を具体化することを目的としてユーザ企業を含む産学の有識者から構成されている．この検討会は図 -1 のコラボレーションを支える「技術インターチェンジ」として機能していくことが期待される．形式的仕様記述言語やモデル検査ツールなどのこれまでのソフトウェア工学の成果と産業界からの高信頼性システム開発の課題とをダイナミックに結合させていく必要がある．

これまでに実施した検討会での議論に基づいて**表 -3** に示すような高信頼システム開発フレームワークの整備が必要であることが明らかになった．この理由は現場への形式手法などの導入障壁を下げ，普及を促進させるためには，現行の開発プロセスの中に形式手法・ツールをシームレスに導入でき，自動化の恩恵が得られることが重要である．そのためには，まず要求，アーキテクチャ，仕様の各層でどのような関係をゴール指向モデリング言語，アーキテクチャ記述言語，形式的仕様記述言語で表現すべきかを明らかにし，次いで各層間の追跡性を定義することが必要である．これにより各層を記述する具体的な言語やツールの相互連携を容易化できると思われる．

形式手法にも多様な個別技術が存在する．しかし銀の弾丸はない．したがってそれらを活用するためには相互連携の仕組みとしてトランスディシプリナリな表 -3 のようなフレームワークが必要になる．相互連携を前提にした技術が望まれている．

対象	関係例	言語例
要求	アクタ間の依存関係	ゴール指向モデリング言語
アーキテクチャ	コンポーネント間の接続関係	アーキテクチャ記述言語
仕様	コンポーネントの状態関係	形式的仕様記述言語
コード	オブジェクトの状態関係	プログラミング言語

表 -3　形式手法を含めた高信頼システム開発フレームワークの例

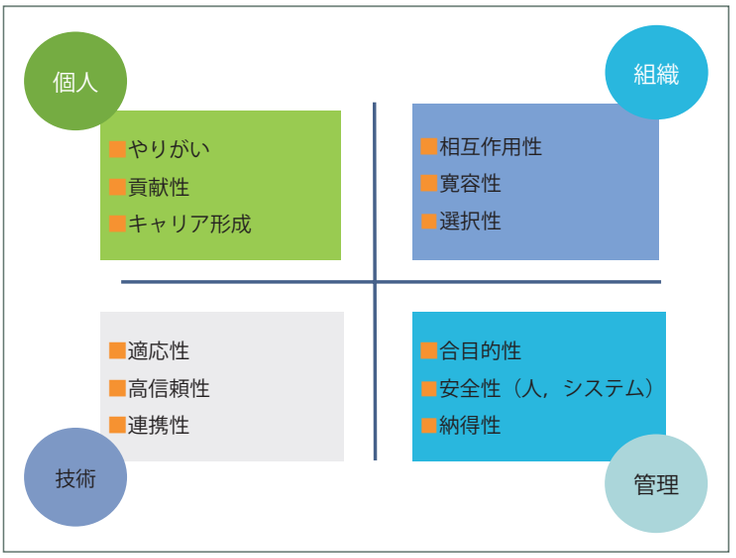

図 -2　ゼロストレス開発の留意点

■ ゼロストレス開発手法

栗田太郎によれば，「従来の開発手法では適当に書かれた仕様に基づいて適当にレビューして，開発して適当に試験していたためにいくらレビューしても，テストしても正しいのか正しくないのかよく分からない曖昧な状況だった．このため現場の担当者のストレスは大変なものがあった」[10] そうである．

これに対して形式的仕様記述言語を用いたことで，「明確な仕様記述に基づいて，論理的なレビューを実施でき，仕様記述に基づいて明確なテスト仕様を作成し，テストを実施することができたために従来のような曖昧な状況が生じなくなることで，正しいのか正しくないのかが明確に判断できるようになり，客観的に仕様の問題点を摘出しやすくなるだけでなく，現場のストレスも減少した」[10] ということである．栗田はこの経験に基づいて「ストレスフリー指向開発」を提唱している [10]．

現場の開発者のことを考えてストレスのないシステム開発環境を具体化することはアウトサイドイン指向ソフトウェア工学である．もし現場の開発者がストレスに悩んだりそのために休職することがなくなればソフトウェア開発の生産性は間違いなく向上する．従来のソフトウェア工学は自動化の追及とその習得を現場に要請してきたが，研究から現場への一方通行の成果流通経路ではなかったか．この反省に立って，筆者も栗田と同じように現場の開発者のストレスをゼロ化することで生産性を向上させる「ゼロストレス開発手法」を目指すべきだと考えている．このため，個人，組織，技術，管理の各側面で**図 -2** のような留意点を整理しているところである．紙面の都合で詳細は省略するが，開発者が「やりがい」をもってシステムやチームのメンバに「具体的に何を貢献するのか」，「それによって個人のキャリアがどのように形成できるのか」を明確にできるようなソフトウェア開発であってほしい．それによってソフトウェア開発に参加するすべてのメンバが喜びを分かち合えるようなソフトウェア社会が来ることを望む．

今後の展望

述べたかったことは単純である．「誰のためにどのような場面で利用されるソフトウェア工学なのかを考えて研究開発に取り組むことが求められる世紀になった」ということである．そういう意味ではソフトウェア開発に多様な関係者が参画することや社会的に受容可能なソフトウェアとはなにかについて社会参加型の合意形成が不可欠となる．たとえば社会生活者としての住民でも参加し貢献できるソフトウェア工学が必要になるだろう．このような幅広い層の関係者が受容可能で相互理解できるようなソフトウェア工学とは何かを考えることがソフトウェアの世紀には求められるのではないだろうか．

アウトサイドイン指向ソフトウェア工学の取り組みは特定の企業や研究コミュニティだけでできるものではない．産業界や社会も含めて取り組むべきである．21 世紀はソフトウェア社会なのだから．

謝辞　九州大学荒木啓二郎教授，IPA SEC の新谷勝利さん，塚本英昭さんをはじめ「高信頼性システム開発手法調査検討会」のみなさん，「ストレスフリー指向開発」をご教示いただいたフェリカネットワークスの栗田太郎さんに感謝いたします．

参考文献

1）花田收悦編集責任：ソフトウェアの仕様化と設計，日科技連品質管理シリーズ第 2 巻，日科技連（1986）．
2）META TECHNOLOGY, OJL による最先端技術適応能力を持つ IT 人材育成拠点の形成：http://www.agusa.i.is.nagoya-u.ac.jp/ocean/#top
3）Boehm, B. : Making a Difference in the Software Centry, IEEE Computer, Vol.41, No.3, pp.32-38 (Mar. 2008).
4）中村雄二郎：臨床の知とは何か，岩波新書（1992）．
5）プラトン著／藤沢令夫訳：メノン，岩波書店（1994）．
6）マイケル・ギボンズ編著／小林信一監訳：現代社会と知の創造 ―モード論とは何か，丸善ライブラリー（1997）．
7）Edelsheim, E. : THE DEFINITIVE DRUCKER, McGraw-Hill (2007)（上田惇生訳：P.F. ドラッカー ―理想企業を求めて，ダイヤモンド社（2007））．
8）Kessler, C. and Sweitzer, J. : Outside-in Software Development A Practical Approach to Building Successful Stakeholder-based Products, IBM Press (2008).
9）Hoare, T. : Assertions : A Personal Perspective, IEEE Annals of the History of Computing, Vol.25, No.2, pp.14-25 (Apr-Jun 2003).
10）栗田太郎：品質は人がつくる「ストレスフリー指向開発」のすすめ，EngineerMind, Vol.9, pp.50-58 (2008).

（平成 20 年 4 月 30 日受付）

山本修一郎（正会員）
yamamotosui@nttdata.co.jp

1977 年名古屋工業大学情報工学科卒業．1979 年名古屋大学大学院工学研究科情報工学専攻修了．同年日本電信電話公社入社．2002 年（株）NTT データ 技術開発本部 副本部長．2007 年同社初代フェロー，システム科学研究所 所長．ソフトウェア工学，ユビキタスコンピューティング，知識創造デザインの研究に従事．本会業績賞，通信協会前島賞など受賞．博士（工学）．著書に，「要求定義・要求仕様書の作り方」（ソフト・リサーチ・センター，2006），「～ゴール指向による～システム要求管理」（ソフト・リサーチ・センター，2007）などがある．人工知能学会知識流通ネットワーク研究会主査（2007 ～）．電子情報通信学会，日本ソフトウェア科学会，人工知能学会，ACM，IEEE 各会員．