第8回　ソフトウェア技術者のグローバルスタンダード化

青山 幹雄 新潟工科大学 情報電子工学科

2000年問題に象徴されるように，ソフトウェアの社会的影響が大きくなるにつれて，ソフトウェア開発のあり方，さらには，技術者の「質」が問われるようになってきた．このため，ソフトウェアプロセスやプロジェクト管理プロセスの標準化，ソフトウェア技術者の認定などの新しい仕組み作りが進められている．ISOでも1997年12月にISO10006としてプロジェクト管理プロセスの標準化が発行された．さらに，米国では，ソフトウェア技術者の認定制度の施行がテキサス州で決定された．ソフトウェア技術者にもグローバルスタンダードの波が押し寄せている．

プロジェクト管理の体系化と標準化が始まった

ソフトウェア開発はさまざまな技術を必要とする．その中でも，プロジェクト管理は最も抽象的でかつ開発の成否に大きく影響する．このため，プロジェクト管理技術を整理体系化し，国際標準化を図る動きが出てきた．

米国のPMI (Project Management Institute) が作成したプロジェクト管理の知識体系PMBOK (Project Management Body of Knowledge)[3] は，プロジェクト管理を9つの分野に分けて整理体系化した．さらに，1997年12月に発行されたISO10006プロジェクト管理における品質の指針 (Guidelines to Quality in Project Management)[7] はPMBOKとほぼ同じ枠組みでプロジェクト管理プロセスの国際標準化を提案している．この2つの体系の枠組みを，**表-1**に示す☆．

これらの体系が対象とする分野はソフトウェア開発に限らない．しかし，ISO9000と同様，内容の多くはソフトウェア開発にも適用されるだろう．

PMIは，PMP(Project Management Professional) と呼ぶ資格を制定し，その認定活動も行っている．1998年7月時点で，全世界で約7,500人の資格認定者がいる．我が国では，エンジニアリングを中心に10人程度の有資格者しかいないといわれている．このため，我が国でもプロジェクト管理に関する業界横断的な組織化が企画されている．

ISO10006はISO9000の発展であるので，ISO9000と同様，その認定についても関心が高まるだろう．だが，ここには，ISO9000と同様，実質的な改善より資格認定が第一目標になるという形式化の罠に陥る危険もある．

知識体系化の狙いは人材育成にある．プロジェクト管理では経験がものをいう．しかし，人材育成を個人の経験だけに任せておくことができなくなった．プロジェクト管理を個人のノウハウから技術として体系化し，かつ体系的に人材育成を図る段階に入ったといえる．

☆ ISO10006，PMBOKと後述のISO12207とを統合したGuide to Software Project ManagementがISO/IECで検討されている．

プロセスが組織の開発力を向上する

■ソフトウェアプロセスとその進化

ソフトウェア開発において個人差は常につきまとう問題である．しかし，個人差の背後にあるプロジェクトや組織全体の開発の仕組みが生産性や品質などを改善する鍵であるとの認識が高まった．このため，1987年にOsterweilは，ソフトウェア開発の手順をコンピュータプログラムのように記述するプロセスプログラミングを提案した．これを契機に，人間の創造的過程であるといわれてきたソフトウェア開発過程を工学的研究の対象として目に見えるようにモデル化し，分析，設計，改善する方法の研究，開発が行われるようになった．この結果，ソフトウェア開発過程をソフトウェアプロセスあるいは開発プロセスと呼び，**図-1**に示す多様な技術が開発された．したがって，ソフトウェア開発は，プロセスとプロダクトの2面から捉えることができる．

一方，プロセスプログラミング以前から，企業では，ソフトウェアのライフサイクル全体にわたるソフトウェアプロセスをウォーターフォール型プロセスで定義し，標準化を図ってきた．これを一般化し，調達者と開発者間の2者間取り引きモデルとして，ソフトウェアライフサイクルプロセスSLCP (Software Life Cycle Processes) (ISO/IEC 12207) が国際標準として1995

表-1　PMBOKとISO10006の体系

技術項目	PMBOK	ISO10006
プロジェクト管理の計画，変更	統合管理 (Project Integration Management)	戦略 (Strategic Process)
		相互依存性の管理 (Inter-dependencies Management)
範囲 (Scope) [開発対象や作業の明確化と管理]		
時間 (Time) [日程の計画と管理]		
費用 (Cost) [費用見積りと管理]		
品質 (Quality)	品質 (Quality) [品質管理の計画と実行]	なし (ISO-9000を適用)
資源 (Resource)	要員 (Human Resource)	経営資源 (Resource) [コンピュータや施設などの資源利用の計画と管理]
		要員 (Personnel) [組織構成，人員割当て，人材育成]
コミュニケーション (Communication) [会議や文書などの計画，情報管理]		
リスク (Risk) [リスクの特定，アセスメント，管理]		
外部調達	調達 (Procurement) [外注の計画，選択，契約，管理]	購買 (Purchasing) [購買の計画と管理，外注の評価，委託開発の実行と管理など]

年に制定され，我が国では1996年にJIS X0160として制定された．

ウォーターフォール型プロセスでは最初に要求が確定することを前提とすることや要求仕様などのシステム全体に関する問題の発見がソフトウェアプロセスの最後であるシステム試験まで持ち越される欠点が指摘されてきた．また，1980年代半ばから，GUIなどの新しいソフトウェアやオブジェクト指向などの新しい開発技術の発展により，ソフトウェアプロセスの構造も見直す必要が生まれた．この結果，要求仕様をプロトタイプによって目に見えるようにするプロトタイピングを導入したプロセスやリスクを考慮して段階的に開発を進めるスパイラルプロセスが提案された．プロトタイピングの発展形として，クライアント／サーバ・システムではビジュアル開発ツールを用いて迅速にソフトウェアを開発するRAD（Rapid Application Development）［ラッドと呼ぶ］が提案され，適用されている．

スパイラルプロセスの概念を発展させて，ソフトウェア開発をひとまとまりの開発プロジェクトから複数の段階的開発に分けて開発するインクリメンタル（Incremental）ソフトウェアプロセスが提案された．特に，オブジェクト指向開発ではカプセル化によって変更範囲が局所化できるので，インクリメンタル開発プロセスが適しているといわれる．たとえば，Rationalの提唱するObjectory Processがある．

さらに，インクリメンタルソフトウェアプロセスを発展させ，一定のサイクルタイムでインクリメンタルソフトウェアプロセスを繰り返し，サイクルタイム内では複数のチームが並行開発を行う並行ソフトウェアプロセス（Concurrent Software-Process）[1]もある．このような，一定期間で開発する形態はタイムボックス（Time Box）アプローチと呼ばれる．

インクリメンタルソフトウェアプロセスなどの段階的開発プロセスは大規模開発のプロセス全体を管理，予測できる小規模な単位に分割統治する方法を提供する．これによって，開発の見通しを良くし，開発リスクを減らせる．短期間でプロダクトの変更などのフィードバックができるなどの効果もある．

また，ソフトウェア開発は，単発プロジェクトとして一度開発すれば終わりとは考えられなくなってきた．むしろ，社会システムや要求仕様の変化によって開発途中の機能追加，変更や出荷後のソフトウェアの進化がこそが本質的であるとの認識が広がってきた．インクリメンタルソフトウェアプロセスは，このような機能変更や進化をインクリメンタルの中に取り込める点でも優れている．

■ケーパビリティの概念とCMMによるプロセス改善

ソフトウェアプロセスの概念は，プロセスを評価し，改善する技術を生んだ．この基礎となる概念がケーパビリティ（Capability）である．ケーパビリティは人の場合は能力を意味する．同様に，ソフトウェア開発では，開発能力を意味する．元来，品質管理において開発量（Capacity）に対し開発組織やプロセスの持つ開発能力を意味する概念として提案された．ケーパビリティの概念を用いて，Crosbyは品質管理の成熟度を5段階で評価するモデルを提案した．これを応用して，カーネギーメロン大学のSEI（Software Engineering Institute）は，図-2に示すような，ソフトウェア開発プロセスの成熟度を5段階で評価するCMM（Capability Maturity Model）を開発した[10]．レベル2以上の各成熟度に到達するために必要な技術をKPA（Key Process Area）と定め，プロセス改善の指針とした．たとえば，レベル2に到達するには，要求仕様の管理，プロジェクト計画，プロジェクト管理，ソフトウェアの外部調達管理，品質保証，構成管理が必要とされる．

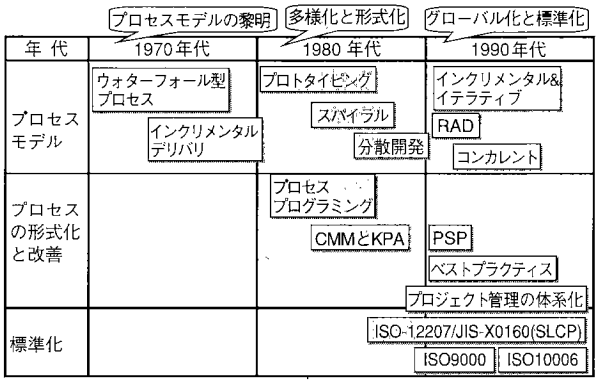

図-1　ソフトウェアプロセス技術の進化

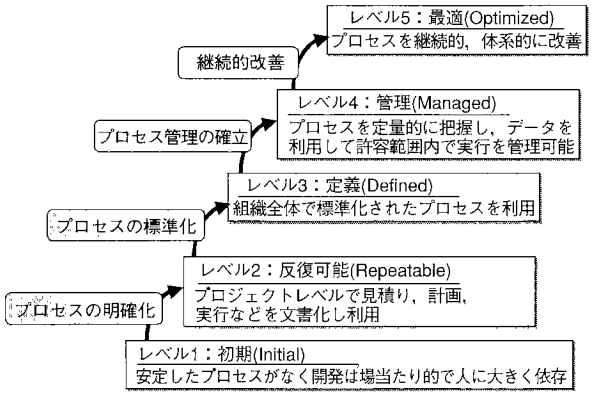

図-2　CMMの5段階

米国国防総省などの調達条件として，CMMが一定の水準に達していることが要求されるようになり，関連企業で適用が広まった．ISO9000に比べCMMでは改善の道筋を示している点で優れている．

以前筆者が従事していた部門でも「あゆみ」と呼ぶ外注ソフトウェア開発の改善プログラムを行ってきた．ここでも，組織やプロジェクト全体でのソフトウェア開発のマクロな指標を5段階で評価し，改善する指針を示した．

いずれも，重要な点はいかに改善するかの道筋を示し，ソフトウェア開発のあり方を実効的に改善することである．

■ベストプラクティスとPSP：プロセス改善の新たな方法

ISO9000やCMMは組織の評価システムであるので，

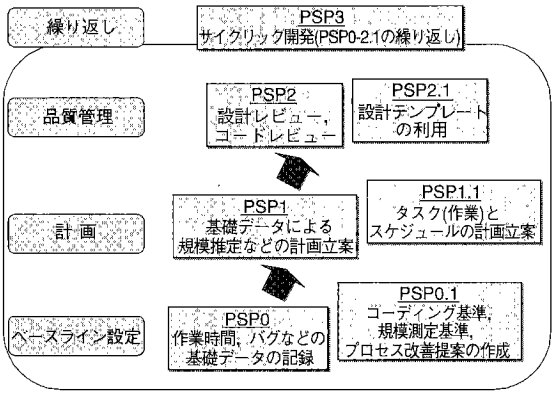

図-3 PSPの成熟度モデル

DoDの主要ベストプラクティス

1) 組織的リスク管理 (Formal Risk Management)
2) インタフェース仕様の確定 (Agreement on Interface)
3) 構造化ピアレビュー (Structured Peer Review)
4) メトリクスに基づくスケジューリングと管理 (Metrics-Based Scheduling and Management)
5) 詳細プロセスの2値品質ゲート (Binary Quality Gates at the Inch-Pebble Level)
6) プロジェクト全体の進捗予実管理の視覚化 (Program-Wide Visibility of Progress vs. Plan)
7) 品質目標の設定とバグ管理 (Defect Tracking against Quality Targets)
8) 構成管理 (Configuration Management)
9) モラル向上 (People-Aware Management Accountability)

調達条件として利用されるようになると，認定取得が第一義となり，本来のプロセス改善が形骸化し，実際に改善できていないという弊害も生んだ．この反省から，ボトムアップによる実践的なプロセス改善の仕組みとしてベストプラクティスという考えが提案された[8]．

たとえば，米国国防総省ではSoftware Acquisition Best Practice Initiativeとしてソフトウェア工学分野における産学のリーダら190名の協力を得て，ベストプラクティスの収集と整理，体系化を行い，Webに公開している[2]．この中には，次の9つの主要ベストプラクティス(Principal Best Practices)とそれを展開した数々のベストプラクティスがまとめられている．

この中で，詳細プロセスの2値品質ゲートでは，プロセスをより詳細に分け，各詳細プロセスごとに進捗や品質をYes/Noの2値で判断することを勧めている．現行の多くのプロジェクトでは比較的大きな単位のプロセスに分け，進捗度といった進捗基準を設けている．しかし，この方法では進捗度などに主観的な判断が入りやすい．これに対し，プロセスをより詳細な単位に分け，かつ，作業の完了か否かの主観的な判断の入らない基準で進捗を管理する方法を勧めている．筆者の従事したプロジェクトでも，プロセスの詳細な分析を行い，詳細プロセスの完了によって進捗を上げるという，同様のプロセス管理方法を用いて，異なる組織が参画する共同開発の進捗管理で効果を得た．このように，実践で実証された具体的な方法を提示している．

もう1つのアプローチが，PSP (Personal Software Process) である[6]．CMMの概念を個人の開発ケーパビリティの改善に応用し，1人1人の開発のあり方を改善するボトムアップの改善方法である．PSPでは，図-3に示すように，PSP0から2.1に至る5段階でプロセスを改善する．データを収集し，そのデータに基づいて見積もる方法なども具体的に示している．PSPの提案者らは，CMMとPSPは相互補完するものであり，併用を勧めている．

我が国の産業界ではISO9000の取得のみが重視されているようであるが，どれだけソフトウェアプロセスが改善されているのであろうか？　ソフトウェアプロセスの改善も多様な技術を必要とすることを認識する必要がある．ベストプラクティスやPSPなど最近のプロセス改善技術の広がりにも目を向ける必要があるのではないか？

ソフトウェア技術者の認定制度が始まる

テキサス州は，IEEE Computer Society，ACMの支援を得て，ソフトウェア技術者の認定制度を導入することを1998年7月に決定した[4]．これまで，米国の法律では，ソフトウェア技術者は，専門技術者（PE: Professional Engineer）として認められていなかった．厳密にいえば，今でも，ソフトウェア技術者と名乗ることはできない．しかし，消費者保護の観点から，ソフトウェア技術者の質の確保が重要であるとの認識に立ち，テキサス州では，早ければ1999年から，ソフトウェア技術者認定制度を開始することになった．認定の条件は，教育と実務経験である．認定を受けるための典型的な条件は，EAC/ABET（Engineering Accreditation Commission of the Accreditation Board for Engineering）が認定した工学系大学を卒業して12年間の実務経験があることとなっている．ここで，ABET［エイベットと呼ぶ］は工学教育の品質を保証するために設立された認定団体である(http://www.abet.org)．米国の主要大学は認定を得ている．なお，情報系学科の認定団体としてCSAB（Computer Science Accreditation Board）があり，EAC/ABETと相互認定(Joint Accreditation)を行う．

ソフトウェア技術者の場合，今のところ認定が意味を持つのは，自治体などの入札とコンサルティングなどの専門職であるといわれている．また，認定制度についても，まだ，賛否両論がある．しかし，続いて，カルフォルニア州でも導入の動きがあるので，今後，他の州へも導入が広がる可能性が高い．他の分野では，認定を持つ専門技術者がいないと入札に参加できないことがある．

ABETは国際的な相互認定制度を進めており，オーストラリア，カナダ，アイルランド，メキシコ，ニュージーランド，英国と相互認定を行っている．

この動きは，2つの意味を持つ．1つは，ソフトウェア技術者のグローバルスタンダード化である．米国標準には異論もあろうが，影響は見逃せない．今後，ソフトウェア技術者の質を目に見えるようにすることが求められるだろう．

もう1つは，これが，工学系大学教育のグローバル

スタンダードとその認定制度に関係することである．我が国の情報分野の大学教育は，質の点で問題が多いことがたびたび指摘されてきた．その結果，情報処理学会でも認定制度について検討が始まっている．しかし，ABETはすでに数カ国と相互認定を行っているが，我が国は対象となっていない．我が国の大学における情報関係教育の意義が問われることになる．

ソフトウェア開発における倫理が問われる

委託開発プロジェクトの規模が巨大となって，その破綻による経済的損害を許容できなくなる場合が出てきた．その結果，委託者が開発者を提訴する事例が起きている[9]．このCONFIRMプロジェクトの事例では，開発者が線表の改ざんをし，虚偽の進捗報告などの倫理性が裁判の争点になったといわれている．

この後，1994年からIEEEとACMの共同で倫理規範の策定が行われ，1997年にソフトウェア・エンジニアのための倫理規範として公開された[5]．

このような事例は米国社会の特質を反映しているかもしれない．しかし，恒常的な予算超過，納期遅れなどの，いわばソフトウェア産業の慣行や体質への警鐘と見るべきであろう．

CTOとチーフアーキテクトの出現

技術開発の競争が激化し，変化が恒常化している．1社ではすべての技術を開発できない．技術の方向を読み，組織全体の視点から技術開発の戦略的意思決定し，システム全体の視点から技術的決定を下すことが，重要である．

このため，米国を中心にCTO (Chief Technical Officer) やチーフアーキテクトといった肩書きを持つ人が増えている．CTOは経営トップとして技術を統括するポジションであり，チーフアーキテクトはソフトウェアシステムのアーキテクチャ設計を統括する．

これは，他産業では以前から行われてきた，いわばベストプラクティスの1つである．たとえば，自動車産業では，新車開発プロジェクトにおいて，重量級プロジェクト管理者と呼ばれる強力な権限を持つプロジェクト管理者を置くことが有効であると広く考えられている．

適切な決定には，プロジェクト管理と同様，専門家として，情報収集力，技術に対する洞察力，優れたセンスが必要である．我が国のソフトウェア産業でも，一部の企業でCTOが創設されている．プロの技術者として，あるいは，プロのプロジェクト管理者として，このような海外に比肩できるポジションの創設とそれにふさわしい人材の育成が必要である．

「プログラマ」の死：あなたはソフトウェア技術者として生き残れるか

ソフトウェア開発技術の進化，多様化は，ソフトウェア開発企業，ソフトウェア技術者の開発力や技術力の格差を増幅する．「技術を持たないプログラマ」，「他社以上の得意技術を持たないソフトウェアハウス」は差別化できない．企業レベルでは，自社の強みとなるドメインを持つことや開発力が問われる．米国のSIベンダのWebページには流通，金融，製造など自社の得意なドメインが強調されている．これは，個人でも同様である．特定ドメインのアナリスト，アーキテクト，プロジェクト管理，設計方法など「プロ」のソフトウェア技術者としての専門性が問われている．

さらに，ソフトウェアの社会的影響力が増すにつれて，安全性や経済的影響などの社会的な側面から，ソフトウェア開発のあり方が問われるようになってきた．この点で，ソフトウェア産業が他産業と異なる特別な産業であるとはいえない．従来のソフトウェア産業固有の慣行を見直す必要もある．

CONFIRMプロジェクトの事例（文献9）より抜粋）

1988年9月：Hilton Hotels Corp, Marriott Corpなどからなるコンソーシアムが，1年間の事前調査の後，航空機，ホテル，レンタカーなどの予約を統合したシステムCONFIRMの開発をAMR Information Service, Inc.（以後A社）に委託．開発費550万ドル．
1988年12月：A社基本設計を提示したが，要求仕様に対し不十分とクレームがつく．
1989年9月：A社は設計完了とし，開発計画を提示．開発費を5,570万ドルから7,260万ドルに増額．
コンソーシアムが開発計画承認．納期は1992年7月．
1990年2月：A社13週以上の進捗遅れを認めるが回復可能であると回答．線表改ざん．
1991年2月：A社縮小した見直し提案．
1992年4月：A社ベータ版提供するが機能不備．A社社長，15～18カ月の進捗遅れを認める．
A社8人のトップエグゼクティブを解雇．
1992年6月：1.25億ドルを費やしてプロジェクトキャンセル．この後，両者提訴．
1994年1月：示談成立．情報によるとA社は裁判による5億ドルの損失とコンソーシアムへ1.6億ドルの賠償金支払．

参考文献
1) Aoyama, M.: Agile Software Process and Its Experience, Proc. 20th ICSE, Kyoto, pp.3-12 (Apr. 1998).
2) DoD Software Acquisition Best Practice Initiative: The Program Manager's Guide to Software Acquisition Best Practices, Version 2.1, Software Program Managers Network (July 1998). http://www.spmn.com
3) Duncan, W.R.: A Guide to the Project Management Body of Knowledge, Project Management Institute (1996). http: //www.pmi.org
4) Charles, J.: A License to Code, IEEE Software, Vol.15, No.5, pp. 119-121 (Sep./Oct. 1998).
5) Gotterbarn, D. et al.: Software Engineering Code of Ethics, CACM, Vol.40, No.11 (Nov. 1997). [安藤　進（訳）：ソフトウェア・エンジニアリングのための倫理規範，情報処理，Vol.39, No.5, pp.456-460 (May 1998)]
6) Humphrey, W.: Introduction to Personal Software Process, Addison Wesley (1996).
7) ISO: Quality Management – Guidelines to Quality in Project Management, ISO10006 (Dec. 1997).
8) 松原友夫: ベストプラクティスの思想と実践，情報処理，Vol.39, No.1, pp. 43-48 (Jan. 1998).
9) Oz, E.: When Professional Standards are Lax: The CONFIRM Failure and its Lessons, CACM, Vol.37, No.10, pp.29-36 (Oct. 1994). [本記事と関連文献がR. L. Glass: Software Runaways, Prentice Hall PTR, 1998にある]
10) Paulk, M. et al.: The Capability Maturity Model, Addison-Wesley (1994). [アンダーセンコンサルティング（監訳）：成功するソフトウェア開発，オーム社 (1998)]

（平成10年10月12日受付）