序　文

本書を執筆し始めてから書き終わるまでの間に，世界的な経済不況が起きた。そのことに影響されたこともあり，本書はある意味でソフトウェア工学の今までの本とはやや趣が異なるものとなった。

すなわち，経済不況に関連して，レイオフやダウンサイジング，米国と他国との経済的なバランスの変化，経済不況時におけるソフトウェアの経済性などの話題を含めている。

ソフトウェア工学は，2008 年当時に財務的な混乱や経済不況の影響を直ちに受けたわけではない。しかし，時間が経つにつれ，ベンチャーキャピタルは次第に枯渇し始め，ソフトウェア企業に対する他の形での財務支援は困難になり，2009 年の半ばにはソフトウェア要員のレイオフが起き始めた。品質保証やテクニカルライタなどの特殊技能の職種は，経済状況が悪いときに最初に影響を受ける傾向があるために，さらに厳しい影響を被った。

不況はソフトウェア技術者の報酬の低下と流動性といった副産物を生み，米国をアウトソース国の候補にした。

2009 年時点では，米国，インド，中国のコスト差は減少し，国内契約と国際契約のいずれも選択できるということが，米国のソフトウェア発注者にとって有利な方向に動いている。

不況が長引く中で，ソフトウェアの高コスト体質は過去よりも厳しく精査されている。不況はまた，ソフトウェアが必ずしも安全ではないことを明らかにした。不況によって，大切な情報の窃盗，アイデンティティの盗用などのサイバー犯罪が従来よりも急速に増大している。また，「フィッシング詐欺」や他人の銀行口座にアクセスするために偽のメールを用いようとする試みが急増している。

筆者の見方では，不況はソフトウェア工学が確かな工学分野になるために必要な 4 つの重要な改善点を明らかにしている。

1. ソフトウェアのセキュリティは，アプリケーションの免疫レベルを高めることと，プログラミング言語そのものに，より優れたセキュリティ機能を含めることで総合的に改善していく必要がある。アクセスの管理と承認はソフトウェア工学の弱い部分である。
2. ソフトウェア品質は，欠陥予防手法および欠陥除去手法の両面から改善する必要がある。過去 50 年間，不十分な品質によりソフトウェアの経済性は損なわれてきたが，この状況を続けることはできない。あらゆる主要なアプリケーションは，インスペクション，静的解析，およびテストを効果的に組み合わせて用いなければならない。当然ながら，テストだけでは高品質レベルは達成できない。
3. ソフトウェア測定は，ソフトウェア開発と保守の真の経済的な全体像をよりよく把握するために改善する必要がある。そのためにはアクティビティベースから見たコストの把握に変えていかなければならない。より優れた測定とは，標準的な経済原則に合わない「欠陥あたりのコスト」や「コード行数」尺度などの従来的尺度の欠点を分析することを意味している。
4. 経済不況により新しい開発は減少しているので，ソフトウェア保守およびリノベーションの経済性をよく理解する必要がある。老朽化アプリケーションを継続利用し，リノベーションを行う手法は，「失われた要求」やビジネスルールを発掘するために古いアプリケーションを利用する手法と同じくますます重要になってきている。

2009 年時点では，ほとんどの企業やソフトウェア技術者は生産性についても品質についても効果的な測定ができていない。このことは真の工学分野にとっては大きな問題である。測定の不足と問題のある尺度の利用は，アジャイルや Rational Unified Process（RUP），およびチームソフトウェアプロセス（TSP）などの手法の有効性の評価が必要以上に難しくしている。

測定の不足とソフトウェア工学手法やプラクティスの有効性を判断する能力の不足は，経済が成長しつつあるときには単に不都合なことですんだが，経済不況時には重大な問題である。ほとんどのベストプラクティスの主張は有効な尺度を用いた確かな測定に基づくものではないために，「ベストプラクティス」といっ

た表現は貧弱な測定によってその名に値しないものになりかねない。

本書では，有効性を明らかにすることができ，ソフトウェア工学を健全な経済的基盤の上におくことができる尺度と測定の組合せを示そうとしている。ここに示す「ベストプラクティス」は，少なくとも暫定的に有効性を判断できる定量的データの得られているものである。

本書は以下の 9 章からなり，それぞれに一連の技術的および社会的な問題を扱っている。

第 1 章：序文およびソフトウェアベストプラクティスの定義

多くのソフトウェアアプリケーションは，最初にそれがユーザに引き渡されてから 25 年かそれ以上も長く生き続ける。ソフトウェア工学は開発だけでなく，ユーザが使い始めた後の永年にわたる保守と機能拡張を考慮する必要がある。ソフトウェア工学はまた，失われたビジネスルールや要求を復旧するために老朽化したアプリケーションを利用するための効果的な手法を含む必要がある。

現在，開発よりも保守の仕事により大勢のソフトウェア技術者が関わっている。多くのソフトウェア保守技術者は，自分が開発したのではないアプリケーションの保守に携わっており，それらはすでに使われなくなった言語でコーディングされ，そこでは仕様書もなければ有効なコメントもない可能性がある。

ソフトウェア工学の「ベストプラクティス」は，1 つのことがすべての場合にあてはまるような技術ではない。ベストプラクティスの評価には，開発アプリケーションの規模に則したプラクティスが必要である。

さらに，Web アプリケーションや情報技術アプリケーションに効果的なベストプラクティスは，必ずしも組込みアプリケーション，システムソフトウェア，および兵器システムに有効なプラクティスと同一ではない。

したがって，本書ではこれら 2 つの要因を念頭におき，ベストプラクティスをアプリケーションの規模とタイプの面から考える。

第 2 章：50 のソフトウェアベストプラクティスの概観

第 2 章では，50 のソフトウェア工学ベストプラクティスについて論ずる。すべてが純粋に技術的なプラクティスではない。例えば，経済不況時に企業は，何年にもわたって組織運営に大きな影響を与えるレイオフを行う必要に迫られる。

本章では，開発ベストプラクティス，保守ベストプラクティス，マネジメント

ベストプラクティス，およびレイオフの扱いなどの社会的な問題に関するベストプラクティスについて考える。レイオフは不適切に扱われることが多く，管理者や経営者よりソフトウェア技術者やスペシャリストが不当に多くレイオフされる状況がある。

レイオフ以外の労働時間の短縮や従業員の報酬の減額などは，通常，実際の要員の削減よりも好ましい。

他のベストプラクティス分野には，セキュリティ管理，品質管理，リスク分析，統治，開発，保守，および老朽化アプリケーションのリノベーションなどがある。

本章の一部分は，ソフトウェア工学のベストプラクティスを守れなかったことが一因であるとして起こされたソフトウェア訴訟の際に，原告の訴えの一部として用いられた。

第 3 章：2049 年頃におけるソフトウェア開発と保守の状況についての展望

ソフトウェア工学の 2009 年時点のプラクティスをそのまま詳しく見てみるならば，変化や改善を予見することは難しい。第 3 章は，ずっと先の 2049 年時点を展望し，我々すべてが社会的なネットワークに結ばれており，ソフトウェア工学の業務は異なる国に住むソフトウェア技術者によって容易に分担されるような世界で，ソフトウェア工学はどのようになっているかを考えてみる。

本章ではまた，要求収集におけるデータマイニングの役割や品質が保証された再利用可能な膨大なライブラリの利用の可能性などの特定の技術的な話題について展望する。また，重要な話題に関して情報を収集し分析するインテリジェントエージェントやサーチボッドの可能性についても触れる。

技術的進歩の早さを考えれば，2049 年には計算機器，ネットワーク，およびコミュニケーションチャネルは著しく洗練されたものになっていることが予想される。しかし，ソフトウェア工学はハードウェアより遅れがちであり，ハードウェアとネットワークの進化に追随するためのセキュリティ，品質，および再利用性についての著しい改善が必要である。

第 4 章：ソフトウェア要員はどのようにして新しいスキルを身につけるか？

一部には経済不況の影響もあって，書籍やソフトウェア雑誌の出版社は厳しい経済的状況にあり，多くは従業員を減らしている。一方，Amazon，Kindle，ソニー PR-505 などのオンライン出版や電子書籍は急速に拡大している。さらに，

Web アプリケーション，ブログ，ツィッターも，プロバイダ，ユーザともに増加している。

第 4 章では，新しいソフトウェア工学の情報の伝達と学習に利用できる 17 のチャネルを評価する。各チャネルを学習の有効性と費用対効果の面から格付けする。また，各チャネルの将来についても長期的な予測を示す。

第 4 章で扱う学習チャネルには，従来的な紙書籍，電子書籍，ソフトウェア雑誌，電子雑誌やブログ，Wiki サイト，商業教育チャネル，社内教育，学術教育，カンファレンス，Web 上で開催されるセミナーもしくはオンラインカンファレンスなどがある。電子メディアは，費用対効果の面ですでに印刷メディアを越えており，学習の有効性の面で古いメディアに挑戦しつつある。

第 4 章ではまた，ソフトウェア技術者，ソフトウェア品質保証要員，ソフトウェアテスト要員，ソフトウェアプロジェクトオフィス要員および管理者の教育訓練カリキュラムについても提言している。今日の大学のカリキュラムは，ソフトウェア工学の主流については十分であっても，規模予測，見積り，計画策定，セキュリティ，品質管理，保守，リノベーション，およびソフトウェア工学の経済性分析などの確かな教育が欠けている。

ソフトウェア尺度についての学術界からの支援は十分ではなく，欠陥あたりのコストやコード行数などの一般的尺度の欠点についての批判的な分析は非常に少ない。

機能的尺度は多くの大学において教えられているが，製造業における経済性原則に反する古い尺度の重要な経済的分析における有り様はほとんど教えられていない。特に生産性に関する固定費の影響の扱いは不備であり，そのためにコード行数も欠陥あたりのコストも経済的な分析には妥当でないことが理解されていない。

第 5 章：ソフトウェアチームの組織と特化

ソフトウェア工学に携わる組織は，小さなアプリケーションを開発する個人の会社から従業員数が 50,000 人にも及ぶ大企業のソフトウェア開発組織に至るまでさまざまである。

大規模なソフトウェア工学組織は，一般のソフトウェア技術者に加えて品質保証，技術文書作成，データベース管理，セキュリティ等々の 90 種類にも及ぶスペシャリストを抱えている。

第5章では，ペアプログラミング，小規模アジャイルチーム，階層組織，マトリクス組織，地域的に分散した仮想組織などの多くの異なった組織構造の結果について検討する。また，ソフトウェア品質保証，テスト，技術文書の作成，プロジェクトオフィスなど，スペシャリストの組織化に最も効果的な方法についても示す。

例えば，大規模企業の大規模プロジェクトでは一般的に，開発チーム自身で保守およびテストを行うよりも保守チームやテストグループを分離するほうが効果的であるとされている。スペシャリストとゼネラリストは協働しなければならないが，組織構造は全体的な成果に強い影響を及ぼす。

第6章：プロジェクト管理とソフトウェア工学

多くのソフトウェアプロジェクトは規模が誤って予測され，見積りは不正確で，開発チームの能力に余る短いスケジュールがコミットされていることは周知の事実である。これらのプロジェクト管理における過失は，ソフトウェアプロジェクトの完全な失敗，重大なコスト超過やスケジュール遅延を引き起こす。

第6章は，規模予測，計画策定，見積り，進捗管理，資源管理，ベンチマーク，変更管理など，もしうまく対処できなければソフトウェア工学の失敗を招く重要な管理機能について述べる。

第6章は，一定規模以上のあらゆるソフトウェアプロジェクトがベースラインとベンチマークに用いることのできる品質と生産性データを集めるべきことを提言している。

用いられる手法とその結果の入念な測定はプロの証しである。測定の失敗は「ソフトウェア工学」がまだ真の工学ではないことを示す兆候である。

第7章：要求定義，ビジネス分析，アーキテクチャ，エンタプライズアーキテクチャおよび設計

プログラムコードが書かれるはるか以前に，まずユーザ要求を理解する必要がある。それらの要求は効果的なソフトウェアアーキテクチャに展開される必要があり，さらに詳細設計に変換される。加えて，新しいアプリケーションは「アプリケーション・ポートフォリオ」の文脈の中に位置づけられる必要がある。20以上もの要求定義の手法と40もの設計手法がある中で，ソフトウェア技術者には非常に多くの選択肢があるのである。

本章では，要求と設計問題についての最も幅広く用いられている手法について議論し，それらが最も適しているアプリケーションの種類やタイプについて示す。アジャイル手法，Unified Modeling Language（UML），およびその他多くの技法について議論する。

2009 年時点における大規模企業の全体ポートフォリオは，1,000 万 FP を超える 5,000 種以上のアプリケーションからなっている。ポートフォリオには社内開発アプリケーション，アウトソースしたアプリケーション，市販アプリケーションおよびオープンソースのアプリケーションが含まれる。

また，ポートフォリオは，Web アプリケーション，情報技術アプリケーション，組込みアプリケーションおよびシステムソフトウェアを含んでいる。経済不況により，企業の経営者にとって自社のポートフォリオの規模，内容，価値，セキュリティレベルおよび品質レベルを知ることはますます重要である。

ポートフォリオ分析は，さまざまなアプリケーションの規模の定量化の難しさによって阻まれてきた。社内アプリケーションと同じように，市販アプリケーションにもオープンソースアプリケーションにも使える高速の新しい規模測定手法が現れて歴史的な問題を解決し始めている。今や，何千ものアプリケーションの規模を数日あるいは数週間で測ることが可能である。

第 8 章：プログラミングとコードディング

本章では，通常とはやや異なる視点でプログラミングとコード開発について論ずる。2009 年時点では，約 2,500 のプログラミング言語と派生言語があるが，なぜそれほど多くの言語があるかについて考える。

また，第 8 章では大量の言語がソフトウェア職業人にもたらす価値について疑問を呈する。さらに，多くのアプリケーションが 2 ～ 15 の異なる言語を用いている理由についても言及する。一般的な結論として，アプリケーション言語の中にはソフトウェア開発に役立つものもあるが，それほど多くの言語の存在は保守にとっては悪夢である。

本章では，あらゆる既知の言語のシンタックスを記録し，廃れた言語で書かれたアプリケーションを近代的な言語に変換することを助ける国家的なプログラミング移行センターの必要性を提案している。

本章ではまた，ソースコードで発見される種類のバグについての情報およびソフトウェア技術者が機能テストやリグレッションテストなどの活動に先立って行

う欠陥予防や欠陥除去に最も効果的な「個人レベル」での手法についても示している。

この章では，プログラミング生産性と品質レベルの測定手法についても議論し，ソースコード行数（LOC 尺度）の経済的問題についても議論する。

2009 年時点ではすでに，LOC 尺度では要求設計，画面，文書などを扱うことはできない。しかし，これらの非コードアクティビティは合わせれば全体開発費の 60 %以上も占めるのである。

代わりは機能的尺度であり，すべての既知のソフトウェア工学アクティビティを扱うことができる。しかし，機能的尺度は測定に時間がかかり，高価である。2009 年時点では，普及が見込まれる新しい高速の機能的尺度測定法が現れ始めている。

第 9 章：ソフトウェア品質：ソフトウェア工学の成功の鍵

品質は，長い間ソフトウェア工学に関わる技術の最も弱い部分の 1 つとされてきた。本章では，欠陥予防手法も欠陥除去手法も含め，ソフトウェア品質に影響を与えるすべての主要要素について触れる。

本章では，正規のインスペクション，静的解析，および 17 の異なる種類のテストの強みと弱みについて論ずる。さらに，ソフトウェア品質についての理解をゆがめかねないさまざまで厄介な尺度についても検討する。例えば，一般的な「欠陥あたりのコスト」尺度は，実際には高品質にとって不利に働き，一種のパラドックスを生む。さらに，品質は単なる欠陥除去のコストをはるかに超える経済的価値があるが，この値は欠陥あたりのコスト尺度を用いては明らかにすることができない。

本章の主たるテーマは，品質はソフトウェアのコスト，スケジュールおよび成功について他の何よりも重大な影響要因であることである。しかし，貧弱な測定のプラクティスは正当なソフトウェア工学の経済的な研究を行うことを困難にしてきた。

本章は，2 つの一般的な品質定義にチャレンジしているので議論のあるところである。品質とは，「要求の充足度である」とする定義については，多くの要求には問題がないとはいえず，「害」をもたらす可能性のあるものは実装すべきではないとの立場から疑問を呈する。一方，品質を「可搬性」などの「○○性」的な表現で規定された要求の充足であるとする定義も，それらの用語のあるものは

予測も測定もできないとの立場から疑問を呈する。品質については，アプリケーションの開発以前に予測ができ，完了時に測定できる定義が必要である。

品質はソフトウェア工学の成功の鍵を握る。しかし，鍵が真のプロの職業の扉を開く以前に，ソフトウェア品質とソフトウェアの経済性の測り方を知る必要がある。本章では，自らの結果を測定できない活動は真の工学的技術ではないと結論づける。今や，ソフトウェア工学にとって潜在欠陥や欠陥除去率レベルなどの重要な話題を研究するときである。

2009 年時点では，ほとんどのプロジェクトはバグや欠陥が非常に多く米国における引渡し前の欠陥除去率は 85 %未満でしかない。あらゆるソフトウェア技術者とソフトウェアプロジェクト管理者は，99 %に近い欠陥除去率レベルを達成するためにはインスペクション，静的解析，テスト段階のどのような組合せが必要かを知らなければならない。測定に基づくそのような知識なしには，ソフトウェア工学はその名に値せず，ソフトウェア開発は単なる工芸であって真の職業ではないのである。

ソフトウェア工学のベストプラクティスの全体としての目標

本書についての当初の着想の 1 つは，1982 年にピューリツァ賞を受賞した Paul Starr の『*The Social Transformation of American Medicine*』に根ざしていた。

同書を読む前は，150 年前，医学の学位はいかなるインターンシップや研修医としての経験も必要なしに，2 年間の学習で与えられていたとは思ってもいなかった。事実，ほとんどの実習医は医学校で学んでいる間，病院を訪れることはなかったとされる。さらに意外なことは，医学校に入るには大学の学位も高校卒の資格も必要なかったことであった。米国の医者の 50 %以上は大学に行ったことがなかった。

Paul Starr の本は，医者の学術的な訓練の改善，専門的な医療過誤基準の確立，および医者の職業人としてのステータス向上を目指す米国の医学協会の試みを詳細に示している。そこにはソフトウェア工学にとって価値のある多くの教訓がある。

ソフトウェア工学のベストプラクティスについての本書の主たる目標は，ソフトウェア工学を正確な品質および生産性の測定から得られる確かな事実基盤のうえに置くためのインセンティブ（刺激・誘因）を提供するところにある。

経済不況が続く中で，ソフトウェアの失敗の最小化，ソフトウェアの提供のスピードアップおよびソフトウェアの保守費用の軽減のニーズはますます高まっている。これらのニーズは，ツール，手法，言語およびソフトウェア組織構造の有効性についての注意深い測定なしには満たすことができない。

正確な測定は，より良いソフトウェア品質とセキュリティへの扉を開く鍵である。より良いソフトウェア品質とセキュリティは，ソフトウェア工学が成功を収めている古くからの工学分野と同じく真のプロフェッショナルな職業になることを可能にする鍵である。

ソフトウェア工学の成果の測定は，より多くのより良いベンチマークにつながり，効果的であることが実証されたソフトウェア工学手法の確かな証拠を提供するであろう。本書の全体的な主題は，ソフトウェア工学の成功に先立つ，より良い測定，より良いベンチマーク，優れた品質管理，そしてより良いセキュリティ対策の必要性である。