ソフトウェアベンチマーク・ベストプラクティス

(Caper Jones, “Software Assessments, Benchmarks, and Best Practices”,ADDISON-WESLEY, 2000)

用語

アセスメント（評価）ソフトウェアプロジェクトが設置され維持される方法を検討するための、公式的かつ構造化された手段。定性的データに基づくことが多い。ベンチマークやベースラインと相補的に利用する。定性的データによって原因を理解し、定量的データでそれを正当化できる

ソフトウェアベンチマーク　ある会社の定量的データを業界標準と比較する

ベースライン　ある会社の定量的データを、その会社の過去数年の記録と比較する

動機

（１）我々の生産性や信頼性は競争相手に比べて良いのか悪いのか？

（２）ある特定のキーエリアにおいて、競争相手に遅れをとっている場合、それを改善するのに何ができるのか？

（３）改良のために投資をする場合、どの程度の投資が必要になるのか？

（４）改良のために投資をする場合、何年ぐらい資金投下する必要があるのか？

（５）改良のために投資をする場合、ROI（投資収益率）はどの程度か？

（６）改良プログラムを実施後の生産性や信頼性はどの程度か？

ベンチマーク基礎データの様式

（１） 国コード

（２） 州コード

（３） 市コード

（４） SIC コード　企業の業種の分類コード

（５） 特性(Nature)コード

新規開発、新規機能追加、必須変更、保守、性能改善、新規プラットフォームへの変換・適応、国化、リエンジニアリング、大量の更新、ハイブリッド

（６） スコープコード

プログラムのサブ要素やサブルーチン、プログラムモジュール、再利用可能なモジュールやオブジェクト、廃棄型プロトタイプ、進化型プロトタイプ、スタンドアローンプログラム、システムのコンポーネント、システムのリリース、新しいシステム（または、アプリケーション)、複合システム

（７） クラスコード

個人的利用を目的としたパーソナルアプリ（個人利用、他人も利用)、大学で作られた大学用プログラム、内部アプリ（一箇所で設置、イントラネットや TSS でアクセス、多サイト設置、個人契約で開発、軍標準を利用した開発、フリーウェアまたはシェアウェア）外部アプリ（WWW に設置、利用者への貸出、組

込み、ハードウェアと結合、商品、アウトソース契約、政府契約、軍契約）

（８） 型コード

非手続的、ウェブアプレット、バッチアプリ、インタラクティブ、バッチデータベース、インタラクティブデータベース、ペンベースド、クライアントサーバ（2 層）、クライアントサーバ（3 層）、ERP(Enterprise resource planning)、科学技術計算、システムまたはハードウェア制御アプリ、通信アプリ、プロセス制御、組込みまたはリアルタイム、厳密なセキュリティをもつ信頼性システム、グラフィックス・アニメーション・画像処理、ロボティックス、エキスパートシステム、人工知能、ニューラルネットワーク、ハイブリッド

スケジュール、コスト、利用者満足度に影響を与える３６個のキーファクター

ソフトウェア分類ファクター

１．システムソフトウェア（物理装置を制御）
２．通信ソフトウェア（外部顧客にリースまたは販売）
３．情報システム（ビジネス情報システム）
４．アウトソースソフトウェア（契約に基づいて開発）
５．軍用ソフトウェア（DoD 標準を遵守）
６．エンドユーザソフトウェア（個人用）

プロジェクト依存ファクター

１．アプリケーションのサイズ（ファンクションポイント、LOC、Class/Method）
２．アプリケーションの複雑さ（サイクロマティックなど）
３．アプリケーションの制約（性能、スケジュール、セキュリティなど）
４．プロジェクトの特性（拡張、新規、保守）
５．ソフトウェアのクラスとタイプ（内部、外部、民間、軍用）
６．アプリケーションのスコープ（プログラム、システム、主システムなど）

技術ファクター

１．形式的手法（Merise, Jackson, Booch, Yourdon など）
２．プロジェクト管理、開発支援ツール
３．欠陥防止アプローチ（JAD, クリーンルーム、改善、QFD, TQM）
４．欠陥除去操作とツール（設計、コードインスペクション、テストなど）
５．プログラミング言語
６．再利用可能資料

社会学的ファクター

１．開発チームの経験レベル
２．プロジェクト管理者の経験レベル
３．顧客の経験レベル

北陸先端科学技術大学院大学　落水浩一郎

４．組織構造とプロジェクトで利用可能な専門家
５．プロジェクトチームのモラル（時間超過、ストレス、不合理な決定など）
６．開発組織の SEI または SPR ケーパビリティレベル

エルゴノミックファクター

１．開発チームメンバーが利用できるプライベートオフィススペースの大きさ
２．オフィススペースの妨害物と邪魔者のレベル
３．インスペクションのようなグループ活動に利用できるミーティングスペース
４．仮想チームに対するネットワークと遠隔通信
５．在宅勤務における遠隔通信支援
６．顧客とのネットワークやビデオ会議の設備

国際ファクター

１．国際ソフトウェアプロジェクトに影響を与える国内法律や条令
２．異なる国間における、クライアントと開発者間のコミュニケーションチャネル
３．プロジェクトに含まれる国に対する補償レベルの差異（variation）
４．異なる国間での、公共休日やバケーション期間の差異
５．異なる国間でのスタッフ補償レベルの差異
６．それぞれの国における仕事に関する慣習の違い

生産性に対するポジティブファクター

信頼度の高い引渡し物の再利用
高度なプロジェクト管理経験
高度なスタッフ経験

生産性に対するネガティブファクター

信頼度の低い引渡し物の再利用
プロジェクト管理経験の不足
スタッフ経験の不足

保守の生産性に対するポジティブファクター

保守の専門家
高度なスタッフ経験
表駆動変数とデータ

保守の生産性に対するネガティブファクター

エラープローンモジュール
埋めこまれた変数やデータ
スタッフ経験の不足

一般的な失敗の原因

不十分な見積り、不注意な計画、不十分なトラッキング

北陸先端科学技術大学院大学　落水浩一郎

Caper Jones, “Software Assessments, Benchmarks, and Best Practices”, Addison-Wesley Information Technology Series, 2000.の要約

（組込みシステムを含む）システムソフトウェアに対するベンチマークとベストプラクティス

システムソフトウェア分野では以下の品質管理アプローチは洗練されたものである

１．　ソフトウェア品質のリサーチプログラムに対して資金を出す
２．　形式的設計やコードインスペクションの利用
３．　品質見積りツールの利用
４．　キープロジェクトに対する品質と欠陥除去目標の利用
５．　QFD の利用
６．　Six-sigma 品質目標の利用
７．　IBM によって開発されたクリーンルーム技術の利用
８．　IBM によって開発された直交欠陥分類の利用
９．　複雑性解析ツールの利用
１０．　自動化された欠陥追跡システムの利用
１１．　テストライブラリの自動化されたサポート
１２．　自動化された変更管理ツールの利用
１３．　訓練されたテスト専門家の利用
１４．　公式的な回帰テストの利用
１５．　ライフサイクル全体にわたる品質尺度

システムソフトウェアの主要なコストドライバー

- 欠陥除去
- 紙文書の作成
- プロトタイピングやプロダクトプログラミングにおけるコード関連作業

システムソフトウェアの成功要因（1985 年から 2000 年にわたって実施された、数百個の SPR 評価とベンチマーク研究より得られた結論）

１．　システムソフトウェア分野は最善の品質尺度アプローチを持つ
２．　システムソフトウェア分野は最善の品質管理アプローチを持つ
３．　システムソフトウェア分野は最善の品質予測アプローチを持つ
４．　システムソフトウェア分野は SEI-CMM の高いレベルにある会社が多い
５．　システムソフトウェア分野は ISO 9000-900４が保証された会社が多い
６．　システムソフトウェア分野は最も完全な開発ツールスーツを持つ
７．　システムソフトウェア分野は最善のマイルストーン追跡システムを持つ
８．　システムソフトウェア分野は最善のコスト追跡システムを持つ
９．　システムソフトウェア分野は他の分野に比べより多くのプロジェクトオフィスを持つ
１０．　システムソフトウェア分野は複雑さの解析と削減においてリードしている
１１．　システムソフトウェア分野は性能モデリングやシミュレーション技術においてリードしている
１２．　システムソフトウェア分野は公式インスペクションを最も普及させており活用している
１３．　システムソフトウェア分野は最善プロセス改善戦略を持つ
１４．　システムソフトウェア分野は最も活発な再利用プログラムを持つ
１５．　システムソフトウェア分野は最善の構成管理アプローチを持つ
１６．　システムソフトウェア分野は最善の複雑性解析アプローチを持つ
１７．　システムソフトウェア分野はオブジェクト指向アプローチの採用においてリード

北陸先端科学技術大学院大学　落水浩一郎

している
１８． システムソフトウェア分野は 10,000 ファンクションポイント以上の規模のシステムに対する最善のトラック記録を持つ
１９． システムソフトウェア分野は最大でかつもっとも洗練された研究所を持つ
２０． システムソフトウェア分野は最善の社内教育能力を持つ
２１． システムソフトウェア分野は最善の図書館と情報収集機能を持つ
２２． システムソフトウェア分野は最も完全な専門家集団を持つ
２３． システムソフトウェア分野はスタッフに対して最善の利益と保証計画を持つ

システムソフトウェアの失敗要因

１． システムソフトウェア分野は機能的尺度の採用について遅れている。ファンクションポイントが適用できないと思っている。1986 年から他の分野ではフィーチャポイント尺度が採用されつつある
２． システムソフトウェア分野は問題を理解することなしに、LOC 尺度を利用している
３． システムソフトウェア分野は自動化されたコスト見積りツールの利用において遅れている
４． システムソフトウェア分野は生産性尺度の利用において遅れている
５． システムソフトウェア分野は巨大な量のペーパワークにおいて軍分野に次ぐ
６． システムソフトウェア分野は解雇やダウンサイジングによってときどき混乱する

システムソフトウェアに対する技術的ベストプラクティス

大規模なプロジェクトを制御するためには、スケジュール計画、コスト見積り、マイルストーン追跡などによるプロジェクト管理者を支援する機能を提供するプロジェクトオフィスの利用が普通である。

ベストプロジェクト管理プラクティス

１． **プロセス改善プログラム：** 評価とベンチマークに従ってプロセスを改善する
２． **年期の評価とベンチマーク：** 毎年のプロセス評価。2 年に一回の外部ベンチマーク調査。その間に生産性と品質を測定する。品質データは毎月まとめられ出版されるべきである。生産性に関するデータは年に一度報告される必要がある。
３． **プロジェクトオフィス：** 10,000 ファンクションポイント以上のシステムでは公式のプロジェクトオフィスを設けるべきである。スケジュール計画ツール、コスト見積りツール、マイルストーン追跡ツールが必要。大規模プロジェクトの管理経験
４． **評価の専門家：** 10,000 ファクションポイント以上アプリケーションに対しては、訓練されたコスト評価の専門家が必要である。
５． **プロジェクト管理の訓練：** プロジェクト管理者はソフトウェアの規模見積り、コスト見積り、品質見積り、マイルストーン追跡、コスト追跡について訓練される必要がある。年間に５日間から１０日間訓練する。
６． **プロジェクト管理のためのツールの集合：** プロセス改善、リスク解析、価値解析など
７． **人間管理ツール集合：** 報奨、スキル発見、業務計画
８． **アクティビティベースのコスト評価：** 1000 ファンクションポイント以上のシステムではフェーズ・レベルの評価では不十分である。
９． **ファンクションポイント解析：** すべての生産性、品質、ベンチマークはファンクションポイント解析を含むべきである。LOC 尺度は複数の言語間での比較を行う場合には有害である
１０． **経験的データ：** プロジェクト管理者は、彼等が責任を持つすべてのソフトウェ

アに対する経験データから計算できる品質や生産性のレベルを知っているべきである。とくに欠陥除去の効率レベルと生産性については知っているべきである。そうでないと、顧客やシニアエグゼクティブによる任意の過重見積りにたいして正確な見積りを防御することが困難である。

１１． **成熟度モデル：** ＣMMのベストプラクティスは利用すべきである。

１２． **マイルストーン追跡：** 100 ファンクションポイント以上のシステムでは公式のマイルストーン追跡を利用すべきである。これにより、スケジュールの遅延を最小化し、少なくとも早い段階で警告を与えることができる。主要なマイルストーンは以下の通りである。要求、外部仕様、内部仕様、インスペクションプラン、設計インスペクション、コードインスペクション、テスト計画テスティング、リスク解析プラン、性能解析プラン、文書化、設置などの完了

１３． **アクティビティコスト追跡：** 100 ファンクションポイント以上のシステムでは公式のコスト追跡をアクティビティレベルで利用すべきである。 一ファンクションポイントあたりのコストのような正規化をすべきである。

１４． **プロジェクト死体解剖：** プロジェクト終了後、プロジェクトを解剖してみることも重要である。公式評価をベンチマークをおこない長所と問題点を明らかにする。

ベスト要求獲得・解析プラクティス

１． **形式要求**: 一ファンクションポイント当り 0.5 ページの公式文書が必要である。

２． **要求インスペクション：** インスペクションには、ハードウェア技術者、マーケッティングと販売担当、顧客代表、プロジェクトの技術者をふくめるべきである。

３． **要求追跡：** 設計やコードの結果を要求にトレースバックできるようにする。

４． **性能要求** 性能目標と性能レベルを判断するためのプロトタイプ

５． **品質と信頼性の要求** 要求仕様の中に品質と信頼性に対する要求を含める。品質については、配置のあとに発生する欠陥と欠陥除去の効率を記す。信頼性については平均故障時間と平均故障間時間を記す。

６． **要求分割：** ハードウェア、マイクロコード、ソフトウェアに埋めこまれるべきフィーチャの公式に分割する

７． **ファンクションポイントサイズ：** 要求から計算できるファンクションポイント数を書く。

８． **要求欠陥：** 要求自体に含まれる欠陥やエラーの数と重大さを書く。エラーの 15 パーセントは要求に起因する。

ベスト設計・仕様化プラクティス

１． **アーキテクチャ仕様：** 10,000 ファンクションポイント以上のシステムに対しては公式のアーキテクチャ仕様を作成する。 アーキテクチャ仕様はフィーチャがハードウェア、マイクロコード、ソフトウェアにわたって、どのように割当てられるかを示す。

２． **仕様分割：** 仕様書の量は大量になるので、各巻が一つの特別な話題を述べるように分割すべきである。例えば、要求仕様、アーキテクチャ仕様、外部設計仕様、内部（論理）設計仕様、データベース仕様などがある。変更と繰り返しに耐えるように作成する必要がある。

３． **プロトタイプ：** キーフィーチャをアルゴリズムにたいしてはプロトタイプを作成するのがよい。投げ捨て型とタイムボックス型の二つがある。後者は周期的制約をもつ。たとえば一月以内に完成しなければならない。進化型は、高品質かつ高信頼なアプリケーションに適合する構造を欠くので、ベストプラクティスではない。10%程度の機能をプロトタイプするのがよい。

北陸先端科学技術大学院大学　落水浩一郎

４．　**設計インスペクション：**　すべての仕様文書に公式インスペクションをかけるのがよい。その理由はソフトウェア仕様書は、一ファンクションポイント当り約 1.5 個の欠陥をもつからだ。公式インスペクションは、欠陥除去に効果があり、65 パーセント以上の設計誤りを発見・除去できる。85%になった例もある。
５．　**性能解析：**　性能解析の専門家を雇うべきである。性能モデリング技術が必要である。また、キートランザクションに対してはプロトタイプは必要である。
６．　**再利用可能仕様：**　再利用可能仕様の利用を推進すべきである。ＵＭＬの利用やオブジェクト指向アプローチの採用は一つの手段である。

ベストコーディングプラクティス

１．　**構造化プログラミング：**　制御構造の複雑さを最小化するため形式的プログラムの構造を採用する。
２．　**コメントの密度とスタイル：**10 行に一つのコメントをつける。多すぎても少なすぎてもいけない。
３．　**記号ラベル：**　明解で記憶しやすい記号名をつける。
４．　**コードインスペクション：**　公式のコードインスペクションは最も効率のよい欠陥除去活動である。
５．　**複雑性解析：**　複雑性解析ツールの利用は必須である。
６．　**エラープローンモジュールの除去：**　コーディングエラーやバグはプログラムに任意に分布しているのではなく、いくつかの小数のモジュールに固まって存在する。インスペクションによって発見し修正するが再コードしたほうがよい。
７．　**モジュールレベルの欠陥追跡：**　エラープローンモジュールの識別に必要である。
８．　**性能解析：**　ソフトウェア性能を予測し、測定し、改良するための洗練された技術がひつようである。
９．　**プログラミング言語：**　C, C++, Objective C, CORAL, CHILL, Forth, PL/S など

ベスト再利用プラクティス

生産性、品質、信頼性を同時に解決する可能性を持ったキーテクノロジーはソフトウェア再利用である。1985 年から 1990 年のアセスメントとベンチマークの研究成果はこれらの進展にあまり寄与しなかった。再利用対象物が、たとえゼロ欠陥レベルであっても、再利用の試みは高価で危険なものである。高信頼性を達成することが再利用成功の重要なステップである。再利用プログラムは以下のものを含む。

- 再利用可能要求
- 再利用可能外部仕様
- 再利用可能アーキテクチャと内部仕様
- 再利用可能データベース設計とデータ構造
- 再利用可能プロジェクト計画と見積り
- 再利用可能ソースコード
- 再利用可能品質・テスト計画
- 再利用可能テストケース
- 再利用可能利用者文書
- 再利用可能ヘルプ文書

システムソフトウェア分野では、再利用可能要求、再利用可能アーキテクチャと内部仕様、再利用可能ソースコード、再利用可能品質・テスト計画の４つが平均より良い。
ベストプラクティスは以下の通りである。

１．　**再利用可能要求**：　ＵＭＬなどのモデリング技術の利用は再利用可能要求を導く。公式インスペクションによりアプリケーションの枠をこえうる。

北陸先端科学技術大学院大学　落水浩一郎

２．　**再利用可能設計：**　公式インスペクションは、仕様や文書の再利用を促進する。
３．　**コードインスペクション：**　コードインスペクションはソースコードモジュールやコンポーネントの再利用を促進する。
４．　**テスト計画およびテストケースインスペクション：**　公式回帰テストライブラリやテストライブラリ制御手順はテスト関連物の再利用性能を促進する。

コンポーネントベース開発がベストプラクティスであるか否かは以前不明である。また、オブジェクト指向による再利用（クラスライブラリなど）をベストプラクティスに含めるか否かも十分に肯定的であるとはいえない。

ベスト変更管理プラクティス

１．　**変更管理ボード**：　10,000 ファンクションポイント以上のシステムではＣＣＢを利用する。3 人から 7 人の要員からなる。一次顧客、プロジェクトオフィス、開発チーム、ハードウェア部門の要員を参加させる。最終決定者という立場ではなく変更のオリジンであるという立場からマーケティングや販売部門を参加させても良い。
２．　**自動化された変更制御：**　自動化ツールの利用をすすめる。100 ファンクションポイント以上のシステムでは、ソースコードに対するツールだけでは不十分である。
３．　**変更に対するファンクションポイント尺度：**　変更にあたってのコスト見積もりにはファンクションポイントを利用する。最もありうる変更は新しいフィーチャを追加することである。これは要求変形（requirements creep）と呼ばれる。もっとも曖昧なものは要求ゆれうごき（requirements churn）である。たとえば、スクリーン上のデータの表示位置を変更する。これはファンクションポイントの値を変えないという問題がある。
４．　**変更に対する見積り：**　ツールを利用する。
５．　**要求追跡と変更：**　要求仕様、設計仕様、ソースコードの間にトレーサビリティを保証する。

ベストユーザ文書化プラクティス

１．　**テクニカルライター**：　めったにないが必要
２．　**プロフェッショナルエディタ：**　めったにないが必要
３．　**文書インスペクション：**　ソフトウェア技術者、顧客を含めて一ページごとにインスペクションを実施する。
４．　**文書モデル：**　新人のテクニカルライターの訓練にはエクサレントと評価された一群のマニュアルを読ませる。
５．　**ユーザビリティ研究所：**　保守マニュアルなどは人間要因研究所のスタッフに見てもらう。
６．　**読者コメント：**　エラー修正などの良い手段は読者コメントである
７．　**ＣＤ－ＲＯＭ文書：**　アニメーションなどの動的教材をつくる
８．　**ハイパーテキストリンク：**　利用者マニュアルはインデックスをつける
９．　**グラフィックスと図解：**　テキストだけでなる図解を多くつける
１０．　**作業例：**　キーフィーチャの実際の利用例を作成する。実際の利用を一ステップごとに示す。

ベスト品質管理・プレ欠陥除去プラクティス

１．　**要求インスペクション：**　公式インスペクションの実施が重要である。これによ

り下流における要求変形を最小化することができる。

２．　**設計・コードインスペクション：**　1000 ファンクションポイント以上のシステムに対しては、80%のプロジェクトは公式インスペクションを実施している。公式インスペクションは欠陥除去に最も効果がある。平均 65 パーセント、最高で 85%の欠陥が除去できる。

３．　**ソフトウェア品質保証：**　独立した部門と要員を設けるべきである。ＩＢＭでは10%が品質保証部門に属する。

４．　**品質研究所：**　大会社に限られるが、公式インスペクション、信頼性モデル、品質見積りツール、欠陥除去メトリックス、クリーンルーム開発プラクティス、エラープローンモジュール解析などに関する革新的な技術を開発する研究部門があると良い。

５．　**エラープローンモジュール除去：**　数値目標に従った複雑性解析、モジュールレベル欠陥追跡、公式インスペクションが重要である。

６．　**品質尺度：**　要求段階から品質測定と欠陥追跡尺度が重要である。欠陥数、欠陥重要度レベル、欠陥のオリジン、欠陥除去手段、複雑性尺度などが重要な概念である。

７．　**品質見積りツール：**　初期にバグ発生の予測技術が発展した。品質予測ツールは、エラーやバグの数を予測し、重要度レベル、欠陥除去の効率がまた予測される。

８．　**信頼性モデル：**　Mean time to failure, Mean time between failures を取り扱う。

９．　**リスク解析：**　技術的、予算的、スケジュール的リスク解析を要求定義フェーズ終了後おこなうことが重要である。

１０．　**（管理職）品質ターゲット：**　欠陥除去の効率と引渡し製品の欠陥について目標を設定することが重要である。重役のボーナス査定対象にいれられたこともある。

１１．　**ＳＥＩ－ＣＭＭ：**　レベル３を達成すること

１２．　**ソフトウェア品質標準：**　標準をさだめ、それに準拠することが必要である。ＩＢＭの欠陥直交分類やモトローラのシックスシグマの例がある。

ベスト検査プラクティスとツール

１．　**テスト専門家：**　テスト専門集団の仕事は以下の通りである。回帰テスト、性能テスト、容量または負荷テスト、統合テスト、ラボテスト、人間要因テスト、システムテスト。また、ベータテストや受理テストをコーディネートする。

２．　**テスト計画とテストケースのインスペクション：**　テスト計画やテストケースに対しても公式インスペクションをおこなうことが重要である。

３．　**自動化されたテストツール：**　自動テストライブラリツールの利用が必要である。

４．　**テストカバレッジ解析：**　テストの十分性を保証するため、カバレッジ解析を行うことは重要である。

５．　**性能テスト：**　スループッツを解析し、かつ、性能を劣化させている部分を見つける

ベスト保守・拡張プラクティス

資金の供出先が異なるので、欠陥修正（保守）と機能追加（拡張）を区別することがまず大事である。

１．　**保守部門：**　めったにないが、欠陥修正の部門を独立させる。その理由は欠陥修正と機能拡張は反目し互いに干渉しあうという経済的理由による。一人の人は双方を行うと、その双方の見積りがメッシーでかつ困難になる。また、欠陥の除去は開発チームにとっては興味のない仕事である場合もある。

２．　**複雑性解析：**　コスト予測のため、主要な変更や更新の前に複雑性解析を行う。

３．　**オンライン保守支援：**　報告書ベースの保守よりはるかに効率的で経済的である。

４．　**オンサイト保守支援：**　100,000 ファンクションポイント以上でフルタイムで必要。

北陸先端科学技術大学院大学　落水浩一郎

Caper Jones, “Software Assessments, Benchmarks, and Best Practices”, Addison-Wesley Information Technology Series, 2000.の要約

システムソフトウェアに対するベスト要員管理プラクティス

ベストスタッフ雇用プラクティス

米国においては、ソフトウェア工学や計算機科学における学生の教育は良くない。その理由は二つある。ソフトウェア品質管理に関するコースが不十分であることと、設計やコーディングにおけるインスペクションのような具体的なコースがないことである。

1. **ハンディキャップのある人の採用：** ハンディキャップのある人は、通常の大学卒より、ソフトウェア技術者分野で成功することが多い。
2. **機会均等雇用：** ソフトウェアは少数民族や女性に対して均等な機会を与え得る。
3. **経験者の雇用：** 新卒より、少なくとも経験 3 年以上の人を雇うほうが良い。
4. **新規採用の基準：** ソフトウェアは複雑でありかつ厳格な品質と信頼性を要求するので、これを充たす人を採用すべきである。管理者とスタッフによる多面的インタビューを実施する。事例研究を課し、パーフォーマンスを推測するのも良い。
5. **インターンシップ：** 在学中に夏季休暇などを利用して数ヶ月仮採用してみることは良い。これにより実際的なパーフォーマンスを理解できる。

ベストスタッフ訓練・教育プラクティス

1. **年季スタッフ訓練：** 10 日間ほどの訓練が一年に一回は必要である。
2. **外部セミナーと会議：** 旅費と参加費を援助して、例えば、ＳＥＩのセミナーなどに参加させることは重要である。
3. **ソフトウェア技術カリキュラム** 設計とコードのインスペクションを教育することがもっとも重要である。その他のコースとしては、要求解析、ソフトウェア設計と仕様化、ソフトウェア再利用技術、構造化プログラミング技術、変更管理と構成管理、テスト技術、性能解析とチューニング技術、品質測定、ソフトウェア保守技術などがある。

ベストプロジェクト管理訓練・教育プラクティス

1. **年季管理訓練：** 年に一度、最低 5 日間ほどの訓練がプロジェクト管理者に必要。
2. **外部マネージメントセミナーと会議：** 旅費と参加費を援助して、例えば、ＳＥＩのセミナーなどに参加させることは重要である。
3. **キーソフトウェアプロジェクトに関するエグゼクティブの訓練：** 主要なソフトウェアトピックスに関してエグゼクティブブリーフィングを準備し発表することが必要である。これらのブリーフィングは、副社長筆頭重役、チーフ・オペレーティング&エグゼクティブ事務官に対してなされる。その理由は、ソフトウェアは主要な財産であり、経営上の制御のきっかけとなるからである。

4. **ソフトウェア管理カリキュラム：** プロジェクト管理者が、ソフトウェア品質制御の経済学(economics)を理解することが肝要である。そこで、欠陥見積り、欠陥防止、欠陥除去に関するコースが必要となる。その他のものとしては、コスト見積り、ソフトウェア再利用の原則、ソフトウェア品質予測、ソフトウェア生産性の尺度とメトリックス、ソフトウェア品質メトリックスと品質のコスト、ＳＥＩ　ＣＭＭの原則、管理者に対するファンクションポイント、リスク解析と価値解析が必要である。近未来に必要になる科目としては、e-business ウェブベース管理のプラクティスがある。これは重要ではあるが急速に発展している分野であり効果的なコースを設けるのは困難

である。さらに、将来重要になるかもしれないものとして、コンポーネントベースソフトウェア開発がある。ＩＳＯ９０００－９００４のコースやＥＲＰツールやビジネスプロセスリエンジニアリングのコースは品質や生産性に目にみえる経験的効果を生むかどうかわからない。

ベスト専門家プラクティス

1. **ソフトウェア専門家：** ソフトウェアテスティング、テクニカルライティング、パーフォーマンスチューニング、品質保証、ソフトウェア保守などはきちんとした訓練を受けた専門家を配置したほうがよい。

ベスト補償・給与プラクティス

1. **補償ベンチマーク：** 年に一度ブラインドテストをする。
2. **補償レベル：** 給料、ボーナス、equity、医療補助などのトータルが上位の四分位数に入れる。

ベストオフィスエルゴノミックスプラクティス

1978 年にＩＢMは、サンタテレサプログラミング研究所を計画した。100 平方フィートの個人部屋、家具、大きな机、大容量のファイルキャビネット、ゲストチェアなどが与えられた。生産性が 11 パーセント向上した。1987 年のトムデマルコとチムリスターによるピープルウェアでもこの効果は再確認された。ウェブでよいような動向であるが図書館も必要である。一日のうちで静かで孤立した時間が５時間は必要である。議論や会合は一時間でよい。

1. **個人オフィス空間：** 雑音のない個人オフィスがソフトウェア技術者や技術スタッフ各自に必要である。８０平方フィートが最適である。
2. **ネットワークとＥ－Ｍａｉｌ：** 技術雇用者すべてとネットワークでリンクされていることが必要である。
3. **会議室：** インスペクションのため 10 人程度がはいれる部屋が必要である。

ベスト組織構造プラクティス

8 人の雇用者あたり一人の管理者がベストである。実際には 2 人から 39 人と分布している。小人数のチームは理想と一般にはいわれているが、大規模ソフトゥエア開発では 100 人は必要となる。制御の範囲の概念が重要である。スタッフが増加すると生産性が落ちると言われているが、ＩＢMの観測によると、管理者の数が増えると生産性が落ちる。

1. **最適な部門規模：** 10,000 ファンクションポイント以上のシステム開発では、一人の管理者は 10 人から 15 人の雇用者を管理するのがよい。
2. **最適な組織構造：** システムソフトウェア開発では組織構造は、マトリックス構造ではなく、階層的構造をもつのが良い。
3. **独立したソフトウェア品質保証：** 10,000 ファンクションポイント以上のシステム開発では、プロジェクト管理の階層からは独立した品質保証チームをもつのが良い。
4. **専門家集団：** 10,000 ファンクションポイント以上のシステム開発では、いくつかのキーアクティビティにたいしては専門家をあてるのが良い。プロジェクト期間中ずっと雇用されているわけではない。システムレベルアーキテクチャ、システムレベル設計、人間要因、データベース設計、品質保証、テクニカルライティング、グラフィックスと図解、単体テストのあとの統合テストなどが候補である。

５．　**仮想組織：**　国や市をまたがる分散開発では、電子メール、ファイル転送やオンラインインスペクションを可能にする広帯域の通信チャネルが必要である。地理的に分散している場合はビデオ会議が効果的である。

ベスト雇用者モラルプラクティス

１．　**モラルとオピニオンサーベイ：**　一年に一度、すべての人を対象に実施する。主要プロジェクトの解消、ダウンサイジングなどの特別な場合もやってよい。
２．　**モラルサーベイのフォローアップ：**　調査の結果には必ず対処する。無視は最悪である。一週間以内に統計的処理を行い、一月以内に執行部は重大な問題に関してはアクションプランをたてる。６週間以内にこれらのアクションプランを検討する会議を立ち上げる。
３．　**オープンドアポリシー：**　管理者や他の従業員から、何らかの不公平な扱いを受けたと感じた従業員は、より上位の管理者にそれを訴えて調査を依頼できる。
４．　**ジョブ移転（配置転換）要求：**　配置転換や転勤を要求できる。
５．　**メリット査定：**　年に一度、行動評価を受ける。
６．　**表彰と認知：**　顕著な成果については報いる。

ベスト作業形態・時間超過プラクティス

１．　**代償時間：**　超過勤務には代休を与える。

ベストターンオーバー最小化プラクティス

１．　**自発的な退職の減少：**　無能な管理者はその仕事をやめさせる。
２．　**新人の発掘：**　広告、リクルート、WWWなどにより新人を発掘する。
３．　**契約雇用者：**　一定の割合で契約社員を雇う。