

# ソフトウェア工学で 最も大切な 10の考え方 (この困難な時代だからこそ)

[Ed Yourdon](#)

email: [ed@yourdon.com](mailto:ed@yourdon.com)

Website: [www.yourdon.com](http://www.yourdon.com)

Blog: [www.yourdonreport.com](http://www.yourdonreport.com),

Twitter, Flickr, Facebook, LinkedIn: “yourdon”

Version 11.1, spring 2009

[Slideshare.net](#) version

Japanese translation by

Kenji Hiranabe

日本語訳: 平鍋健児

## 公開に関する詳細および免責事項

- このプレゼンテーションは、オープンな共同ドキュメントです。インターネットとWebブラウザがあれば、誰でもこれを見ることができ、かつ改変することができます(良い改変も悪い改変も、Googleがそのような改変されたドキュメントへのアクセスを許しさえすれば)。この文書のどの部分も、Ed Yourdon(“Ed”)が必ずしもレビューしたものではないことに注意してください。このドキュメントに書かれている理論やビジネスプラクティスは、必ずしも Ed のものではありません。
- これは、このドキュメントに「価値が無い」とか「正確な情報ではない」と言っているのではありませんが。しかし、Ed はこのドキュメントの妥当性を全体として保証することができません。どのページの内容も、Edのオリジナル文書(およびその原稿)に同意しない人によって、改変、書き直し、などの編集が行われている可能性があります。
- あなたがこのドキュメントに含まれている情報、リンクされている情報を使う場合、Ed 自身そして貢献者、協力者、このドキュメントへのどんな関係者も、情報の不正確性について責任を負いません。
- あなたはこのドキュメントをどのような形でもコピーする、制限されたライセンスを付与されます。これは、Edおよびこのドキュメントの貢献者、協力者および視聴者の信頼(契約に関わる関わらないを問わず)を作る、もしくは含むものではありません。
- あなたとEdの間には、あなたのこの情報の利用や変更に関して GNU Free Documentation License (GFDL)以上の合意および理解は存在しません。Edは、このドキュメントに関する誰のいかなる情報の変更、編集、改変、削除について責任を負いません。
- このドキュメントの中で言及、使用、引用されたトレードマーク、サービスマーク、コレクティブマーク、デザイン権、人格権、などの権利は、その所有者の所有物です。それらのここでの使用は、同一もしくは同様の情報利用(GFDLライセンスのスキームで認識されるもの)以外のいかなる目的での利用許可を暗示するものではありません。明示しない限り、Ed とこのドキュメントはそれらの権利保持者からエンドースもアフィリエイトもされていません。Edは他で保護された資料の利用を保証することができません。あなたのこれらおよびこれらと同様の所有権の利用(翻訳者のリソース)です。
- このドキュメントには、多くのリンクが含まれています。リンクの信頼できる日本語訳がある場合、そちらを紹介しています。翻訳書の画像についても、調査できる範囲で日本語版を紹介しています。元のリンク、画像が見たい場合は、原文を当たってください。  
(<http://www.slideshare.net/yourdon/top-ten-s-econcepts-v111>)
- 翻訳の間違い、読みにくさは、翻訳者(平鍋)の力量の範囲ですが、この翻訳にも上記免責を適用し、翻訳を修正して公開していただいて構いません。

翻訳者にあたって

# Top Ten Eleven Items

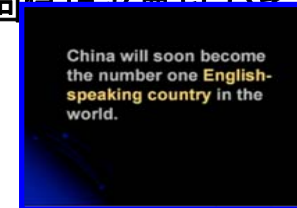
## イントロダクション

1. 計測できないものは制御できない
2. ピープルウェア(Peopleware)
3. インクリメンタル(Incrementalism)
4. 反復(Iteration)
5. 欠陥が下流に漏れること、修正コストが増加する
6. トレードオフは、非線形
7. 再利用は重要
8. リスクマネジメントが鍵
9. 一貫性は才能+デスマーチに勝る
10. 車輪を再発明しない
11. 透明性を重視。何も隠さないこと

## イントロダクション

### □ なぜこれが大切か？

- ✓ 私たちは「おもしろい」時代(ここ数年つづくであろう)に生きている。
- ✓ ビジネスはITのイノベーションを必要としている。同時、コストカットと「基本への回帰」を必要としている。
- ✓ グローバルな経済不況は、さらなるプレッシャーを生むだろう。
- ✓ これらを見よ。“[Shift Happens](#)”, “[Did You Know?](#)”
- ✓ システムの複雑性が増加している、、、などなど。
- ✓ なぜこれらのコンセプトを使わない？私のブログに考察がある([1](#), [2](#))。
- ✓ 新しい課題 ([Web 2.0](#)): ユーザが自分でテクノロジーをコントロールできる。[revenge of the user](#)「ユーザの逆襲」



### □ ほとんどのキーとなるソフトウェア工学のアイデアは、新しいものではない

### □ しかし、それは一貫性をもって実施されていない

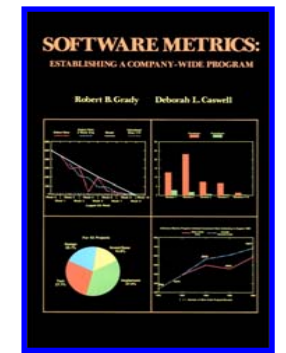
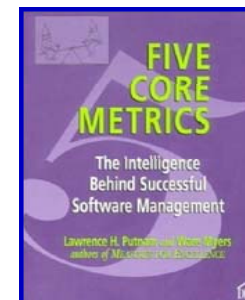
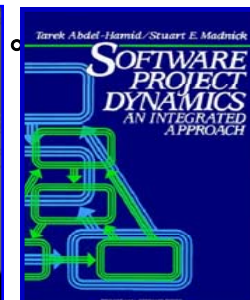
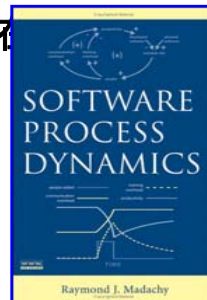
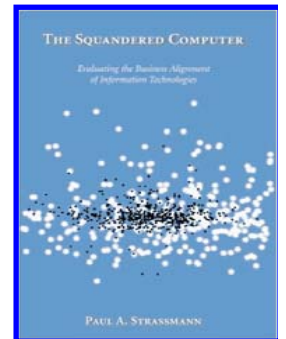
### □ ロケットサイエンス(途方も無く難しいもの)ではない...

### □ ...しかし、「常識」はそれほど知られていない

### □ 世代が新しくなるたびに、基礎を再発見する運命にある。

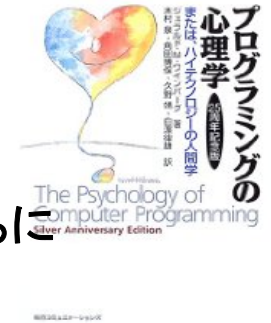
## 1. 計測できないものは制御できない

- 多くの良書がある。例えば、トム・デマルコ 1982 の古典、「品質と生産性を重視したソフトウェア開発プロジェクト技法」、そして、Paul Strassmann の *The Squandered Computer*
- Cultural/management issues more important than technical issues (1987 の古典HP's Robert Grady & Deborah Caswell, *Software Metrics: Establishing a Company-Wide Program*)
- コア・メトリクス: 規模, 見積, 欠陥, 離職
- ソフト・メトリクス: 士気(モラル)、ユーザ満足度など。
- システム・ダイナミクス — 例えば、ブルックスの法則 (Tarek Abdel-Hamid の *Software Project Dynamics*, と Ray Madachy の新しい本, *Software Process Dynamics*)
- 新展開: Joel のエビデンス駆動スケジューリング(evidence-based scheduling)
- ハイゼンベルグの不確定原理 の潜在



## 2. ピープルウェア

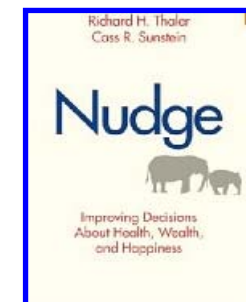
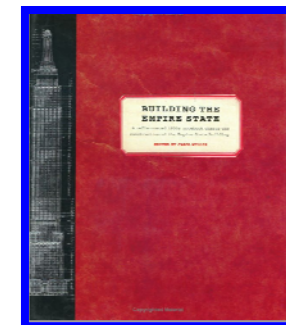
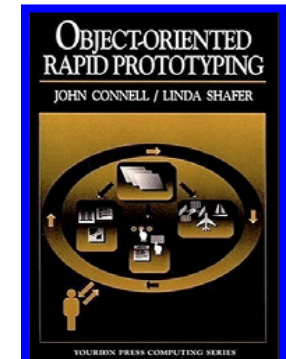
- ❑ 人は(いつの時代でも)プロジェクトにおける最大の生産性要因である。
- ❑ 最もよい人材を採用して、Google の人材管理をまねよ。
- ❑ **質問: 一番頭のよい大学卒業生が、あなたの会社を選択しますか(選択権が彼らにあると仮定して)?**
  - ✓ 注意! 彼らのほとんどは MS Outlook を見たことがない。(もし見たら恐れるだろう)
  - ✓ 彼らは、すべての人がFacebook にブログを書き、とiPhone/Android上で IM していると思っている。
  - ✓ 彼らは、COBOLでのプログラミングできる人がまだ生きている、ということに仰天する。
- ❑ 彼らによいオフィス空間をあたえて、仕事環境でのインタラプトを最小限にせよ。
- ❑ 「チーム殺し(teamicide)」をするな。
- ❑ 『ピープルウェア』: ICSE 2007 パネルセッション、「peopleware20周年記念」のレポートを見よ(Sep 2007 IEEE Software). 私のブログでも無料でみれる。(訳者注: ぼくのブログに解説あり)
- ❑ “Meet the Life Hackers,” を見よ(Oct 16, 2005 New York Times). 西海岸にある2つのハイテク会社での1,000時間にわたる観察に基づいている。”



• “社員は、プロジェクトの中で11分ごとにインタラプトされ、別のことに振り回される。11分のタスクは、email の返信、のようなより短い3分のタスクに分断され、タスクから引き離されるたびに、戻るのに平均で25分の時間がかかる。”

## 3. インクリメンタル開発 (vs. “ビッグバン” 開発)

- プロトタイプ は貴重である。John Connell と Linda Shafer による、*Object-Oriented Rapid Prototyping*, を見よ。
- 「デイリービルド」(“daily build”）、継続的インテグレーション(continuous integration) アプローチを使え。  
(thanks to Joe Chavez)
- 変更管理(change management,)を奨励し、スコープクリープ(scope creep)とスコープ変更(“scope churn”)を警戒せよ
- 関連する概念として、, “economics of nudging.” ここに、記事がある。書籍『Nudge: improving decisions about health, wealth, and happiness』を見よ。(日本語の記事も)

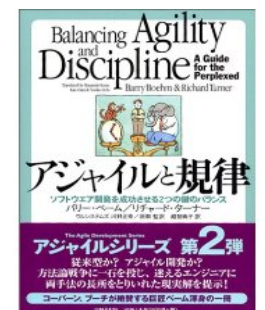
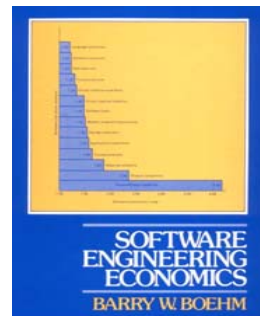


## 4. 反復型開発

- 古典をみよ (“有史前”, インターネット前) 1970 の論文 Win Royce, Managing the Development of Large Software Systems: Concepts and Techniques
- 大きな誤りを後でするよりも、小さな誤りを早期に。
- コンセプトの適用は、「all or nothing」である必要はない。
- 最近のオブジェクト指向のバージョン:  
“refactoring”
- 近代的なツールを使う (高速な協調的なイテレーションのためにTwitter, wiki, などを使う)

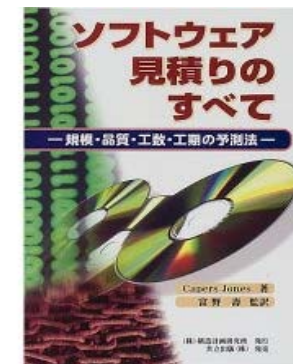
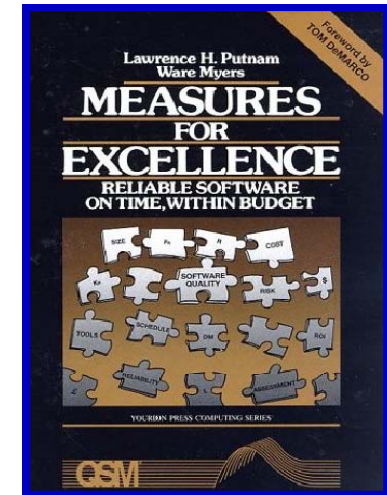
## 5. 下流に欠陥が「漏れる」と修正コストが増加する

- コストはライフサイクルのフェーズを1つ越えるごとに10倍になる。
- 1980年代前半の、Barry Boehmによる文書
- Agile の人たちは不賛成 (すばらしい新しいアジャイルリンク集), であり、継続的なリファクタリングを強調している。
- しかし、Boehm は同意していない; それはプロジェクトサイズに依存している『アジャイルと規律』(2004)
- (訳者ブログにも、この件についての記事がある)



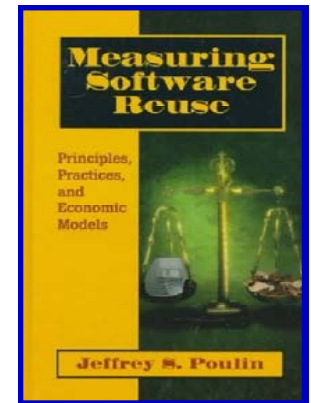
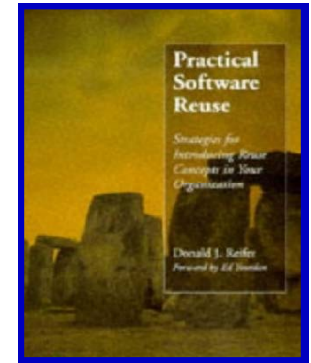
## 6. トレードオフは、非線形

- ❑ 人と時間のトレードオフは、3次多項式
- ❑ Capers Jones (『ソフトウェア見積もりのすべて』)によると、欠陥 =  $k * \text{ファンクションポイント}^{**1.25}$
- ❑ 頭で計算することは困難!
- ❑ 小さなプロジェクトでちゃんとした交渉をするためには、十分に強力な見積もりツールが必要  
(例えば., SLIM, Costar)



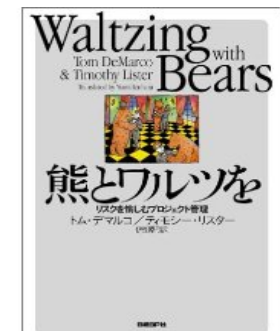
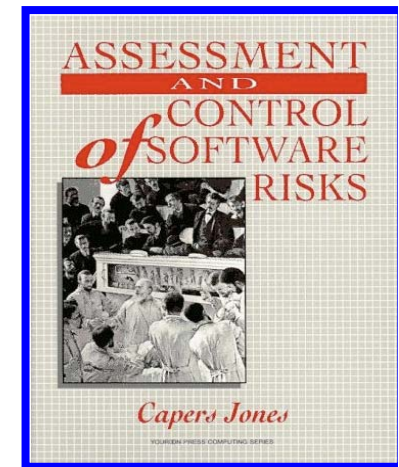
## 7. 再利用は重要

- コードの再利用だけでなく、設計、仕様、プロセス、計画、予算テンプレート、チーム、などの再利用。
- Web 2.0 version: マッシュアップをサポートするAPI
  - 例: [Facebook API](#) を利用した6,000のアプリケーションが5ヶ月で誕生した ([Mary Meeker](#) による [2007 Web 2.0 Summit conference](#), における [論文](#) p. 37)
  - [OpenSocial API](#) がすべてのソーシャルネットワークに向けに提案されている (e.g., Myspace, Bebo, Hi5, etc.)
  - 例: [HousingMaps](#) = [Craig's List](#) + [Google Maps](#). See also Google Maps [NYC interactive transit map mashup](#)
  - 現在の重要なプロジェクト [IBM](#), Yahoo ([Pipes](#)) Microsoft ([PopFly](#)), etc
  - 275 [Flickr Mashups](#)
  - 面白い例: [Earth Sandwich](#)
  - 面白い例: [George Bush + U2's "Sunday, Bloody Sunday"](#)
  - もっと面白い: [Obama "1984" mashup](#)
- 再度: 技術はビジネス/文化的な問題よりも重要ではない
- 再利用のコストを無視してはいけない



## 8. リスク管理が鍵

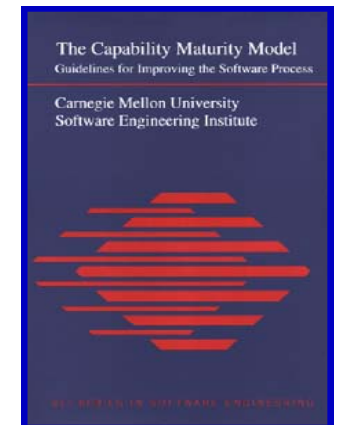
- 失敗プロジェクトの共通点; Capers Jones の 1994年の書籍, Assessment and Control of Software Risks
- リスクマネジメントの共有が肝
- よく知られたテクニック: R-Y-G list of top 10 risks (例)
- Remember: 伝統的なソフトウェア開発だけでなく、プロセス改善に関連したリスクも存在する。
  - ✓ Identifying and Managing Risks for Process Improvements, Joyce Statz , by Don Oxley, and Patrick O'Toole.
- 再度: 議論の中心は文化的な問題であり、政治であり、技術ではない。(2007-2008 の銀行危機と同じように)



## 9. 一貫性は 才能 + デスマーチ に勝る



- (おそらく) Silicon Valley 以外では...
- グローバルな不況は、少なくともここ数年、デスマーチプロジェクトを増加させるだろう... ☹️
- SEI-CMM, ISO-9000, six-Sigma, ITIL, etc, etc. のようなアプローチに注目せよ。



## 10. 車輪の再発明をしてはいけない

- ❑ ベストプラクティス (ボトムアップ!)
- ❑ ワーストプラクティス  
(233 もブログがある!)
- ❑ SWBOK

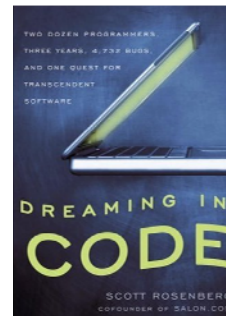
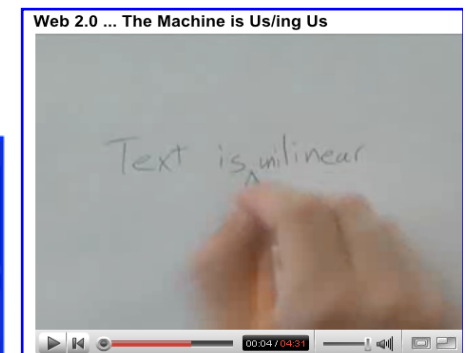


## 11. 透明性を重視。何も隠さないこと。

- 2009年2月18日のTwitterでの会話をありがとう、Kent Beck, Ron Jeffries, その他のみなさん。
- [Kent Beck](#): “ソフトウェア工学は、ソーセージ工場に顧客が入るのを拒むのではなく、招き入れたときに、改善される。”
- [Ron Jeffries](#): “隠すこと、について完全に同意。何も隠さないこと。透明性がすべてを決定する。”
- [Ward Cunningham](#): “きれいなコードにも技術的負債 (Technical Debt)があり、それは良いことなのだ。なぜか、[このビデオ](#)をみよ。”

## 結論

- ❑ 技術は中心課題ではない。
- ❑ フレデリック・ブルックス(Fred Brooks): “銀の弾丸などない”
- ❑ マーガレット・ミードの“地球時代の文化論”
  - Michael Wesch's コメンタリー, “Vision of Students Today,” 今日のWeb 2.0 テクノロジが大学生の教育に影響しているか (こちらも。“the machine is (us)ing us”!)
  - 2007年10月3日、告知: UC Berkeleyは大学の講義をYouTube配信します。
  - 若い世代が現在考えていることは、あなたには分からない!  
BRBOMGLOLROFLLMFAO と“Google, A Girl, and the Coming Apocalypse.”を見よ。
- ❑ 新しいことを無視してはいけない。— 例えば, Chandler



# ソフトウェア工学で 最も大切な 10の考え方 (この困難な時代だからこそ)

[Ed Yourdon](#)

email: [ed@yourdon.com](mailto:ed@yourdon.com)

Website: [www.yourdon.com](http://www.yourdon.com)

Blog: [www.yourdonreport.com](http://www.yourdonreport.com),

Twitter, Flickr, Facebook, LinkedIn: “yourdon”

Version 11.1, spring 2009

[Slideshare.net](#) version

Japanese translation by

Kenji Hiranabe

日本語訳: 平鍋健児