創造的プロセス改善

― 成熟度モデルを超えて ―

松原 友夫

Matsubara Consulting

matsu@computer.org

私にとって改善とは

- 問題を解決していく創造的アプローチ
  - 目標を阻害している問題を摘出し
  - 問題と原因の構造を理解し
  - より少ないエネルギーで原因を排除し問題を解決していく
- 楽しくエキサイティングな活動
  - 常によりよくしようとする前向きな活動
  - 会社にいるで充実した時間を過ごせる
- ゴールを達成するアプローチ
  - 認証取得やレベル達成はゴールではない
- あらゆる知識や智恵を総動員する創造的な活動
  - ソフトウェア以外で役立つアプローチはソフトウェアでも使える
  - ソフトウェアでうまくいけば他の改善にも役に立つ
- 組織の目標と自己の目標を合わせる
  - たとえ上から言われたとしても自主的な活動に変えられる

もともと“KAIZEN”は日本のお家芸

- 地道な改善の積み上げで日本の製品は世界市場で優位に立てた
  - TQCなどの日本のアプローチの一部はCMMにも取り入れられている
- 生産業の品質改善にも日本独自の様々なアプローチがあった
  - ボトムアップ型の品質管理
  - 田口メソッド
  - etc.
- 現在も日本製品の世界市場での優位を保っている
- 日本はたしかに「もの作り」で成功した
- 今後ももの作りで優位を保とうとするならソフトウェアを制覇しなくてはならない
  - 重要機能の多くをソフトウェアが担うようになった
  - ソフトウェアプロセスの改善こそが今後優位を保つ鍵である
- 有形物の生産でできたことが無形物でできないはずがない
  - 自信をなくす必要はない

私の改善哲学

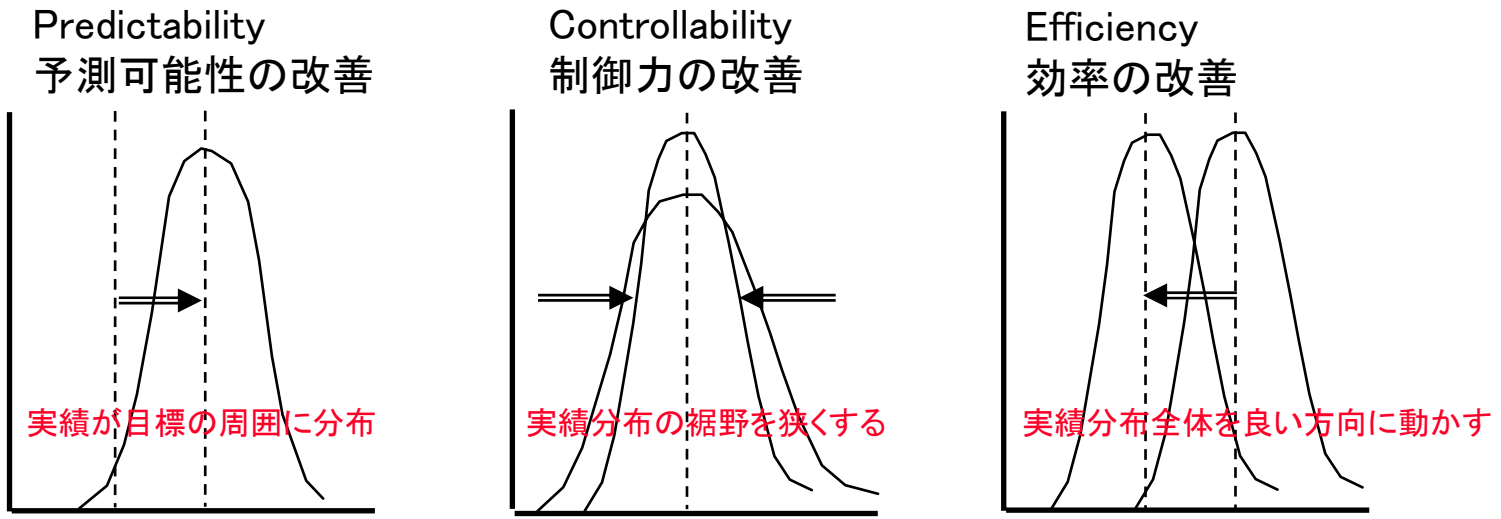
- 改善には理念が必要
  - 例えばBoeingのGorge YamamuraのRSST（次図参照）
    - Right Thing
    - Small Steps
    - Simple Solution
    - Timing
- よい結果を得られなければ改善活動とは言えない
  - 認証やレベル達成だけでは改善とは言えない
- 使えるものは何でも使う
  - モデルの外にもたくさんある
  - 基礎科学(物理)、数学(微分、積分、統計)、(従来の)工学も
    - これらはソフトウェア技術者が身につけるべき基礎
      - Parnas, D. L., *Education for Computing Professionals*, IEEE Computer, Jan. '90
  - ハードウェア生産で培った日本の改善手法にも役立つものが多い
- 改善の真髄はプロセスを変えてみる勇気と実力
  - CMMのレベル構成をこう見ることもできる
- モデルは道具の一つ
  - 金科玉条として扱わない
  - 目的に合った道具を選ぶ

The Journey

George Yamamura 12/9/98

改善の3要素

- CMMの冒頭に示されている3つの図はあらゆることの改善に共通の要素



- 測れるものならどんな目標へ向けての改善も対象になる
  - コスト、品質、スケジュール、サイクルタイム、チーム規模、生産性、ビジネス効率、ビジネスミックス、売上、利益、etc., etc.

先ず「己を知る」ことから

- 達成したいビジネス目標に対して改善前の時点での実力を知る
  - これをやらないで改善を指示するのは、ゴールを示さないで走らせるようなもの
    - どこへ向かって走ればよいかわからない
    - やることがすべて無駄になる可能性が高い
    - にもかかわらず改善活動を始めるケースは意外に多い
      - 「ISO 9000を取れ！」「来年までにCMMレベル3に到達せよ！」の多くはこの類い
  - 実現可能性を見極める
    - 目標が実力分布の範囲外ならすぐには実現不可能
      - 目標が非現実的であるならそれを示すためにも実力の把握は重要
      - 時間をかけて手の届く改善をくり返せば可能かも知れない
    - 目標が実力分布の2σあたりなら十分実現可能な範囲
- 常に実力分布図で現時点での改善の進行状況を知らしめる
  - 一部でなく組織の全員に実力を理解させる
  - 自らの位置を知った後に現実的な目標に向かって改善を始める

ビジネスゴールの理解と分解

- 組織のビジネスゴールにはいろいろある
  - 例えば
    - 国際競合力をつける
    - 顧客サービス第一
    - …で業界ベストになる、etc.
  - のような定性的なものもや、次のように数字で示せるものがある
    - 納期厳守　e.g. ビジネス上の期限が動かせない場合
    - コストの大幅低減　e.g. 大規模プロジェクトの場合
    - 品質第一　e.g. 大量生産する組込み系製品のチップ
    - 開発サイクル (Time to Market)の短縮　e.g. 類似の製品のくり返し開発
    - 顧客満足度の改善　e.g. プロダクト系
- ゴールを実現するためには
  - サブゴールを展開する
    - 例えば、国際競合力なら製品別シェア目標
  - ビジネスゴールを実現可能な目標に分解する
    - ドメインごと組織階層ごとに
    - ゴールが分解されないまま組織階層を降りてくると、現場では何をしたらよいかわからないから何も変化しない

問題と原因仮説

- 問題と原因の関係は複雑にからみ合っている
  - 例えば
    - プロジェクトコストが目標を大幅に超えた
    - 主な原因は絶え間ない大量の仕様変更であった
      - もちろん、それは多くの複合する原因の一つである
    - それから先は仮説の議論になる
      - 仕様変更はすべて受け入れるべきものであったか？
      - 要求仕様は十分に分析されたか？
      - 必要な技術を十分身につけていたか？
      - 要求仕様のレビューに顧客が参加したか？
      - 見積価格は適正であったか？
      - etc.
- 問題発見と原因仮説の議論は人を巻き込んでその気にさせるまたとない機会
  - 議論のプロセスで改善への参画意識が高まる
  - 問題発見と原因の推理には発見的な能力が必要だが議論することでこれを訓練する
  - みんなが喜んで参画するオープンな組織へ

異なる世界での改善からのヒント

- 皆さんはCMM(I)についてよく知っていると思うので、あえて別の世界での改善やCMMでは使わない言葉を用いた改善の話しをしたい　―　何等かのヒントが得られれば幸いである
  - ネタは機械工場、コンピュータ工場、ソフトウェア工場、及びソフトウェア会社での改善経験
  - 特に、いまは存在しない日立亀有工場という機械工場での改善について
    - 私がいた頃は、工場を挙げての改善活動を実施していた(1956-65)
    - 私は機械化の立場からこのプロジェクトに参加した
    - ソフトウェア会社の創立に当たって機械工場でのアプローチの多くを導入した
  - 機械工場での改善アプローチの多くはソフトウェアでも役立つ
    - 特に数字や図を用いいた改善
  - もちろん、ソフトウェアプロセスの改善には、さらに意識的にその特性を考えたソフト固有のアプローチを追加する必要がある

役に立つ改善のヒント(1)

- 徹底した現場主義、実証主義　ー「真実は製品やシステムそのものの中にある」
  - 生産工場の典型的なアプローチ
    - 機械屋は「見て、触って、測って」判断する
    - 電気屋は測定器を使って「見て」判断する
  - 有能な管理者は頻繁に工場を巡回して改善すべき問題を指摘する
  - コスト低減策や部品標準化も現物を、または図面を見て議論する
  - ソフトウェアでも同じ
    - ソフトウェアでは職場を巡回する管理者は少ないが、無形物を作っているからこそチーム員の何気ない会話や会議の雰囲気を観察することで改善や問題のヒントが得られることが多い
    - 報告だけでなく一部でよいから自ら文書やコードを読んで判断する
    - 除去すべき問題の重要度、優先度は現場を見てから判断する
    - 現場と実物のみが真実を語る　ー　話や報告をそのまま信じてはいけない
    - 第一線の作業者のみが真実を知る
    - アセスメント(アプレイザル)は、実際に使われている資料やツールがある現場に近いところでやるのが理想
    - 閉鎖的な雰囲気のオフィス、権威主義的な管理者のもとでは真実が隠蔽されることが多い
      - これも巡回すればすぐにわかる

役に立つ改善のヒント(2)

- エントロピーを減らす
  - エントロピー増大の法則:
    - エントロピーとはある系の無秩序性の程度を表わす物理量で、物質系が平衡に向かう過程で発生し、一方的に増大する特性がある。自然状態では、物質は拡散し、エネルギーは低級化し、情報は失われる
    - 機械屋はエントロピーを増大を極度に嫌い本能的に避けようとする
  - 機械工場で納期の混乱を防ぐための金言
    - 「一升ますには一升」、つまり工場に部品を無理に押し込まない
      - 能力の8割に止める
      - 部品製造での「特急」指定を乱発しない
    - 「整々と遅らせよ」
  - ソフトウェアでの応用
    - 情報隠蔽やカプセル化を適用してスッキリしたシステム設計またはプログラム構造にする
      - Separation of Concern, Modular Conceptなどの適用
        - 参考:D. Parnasの情報隠蔽(Information Hiding)、N. Wirthのカプセル化(Encapsulation)
    - 文書やソースコードなどの派生物の様式を揃える
    - 個人のプロセスの質を揃える
    - 仕様の際限のない膨張を抑える
    - 開発環境をエントロピーが減るように構築する

役に立つ改善のヒント(3)

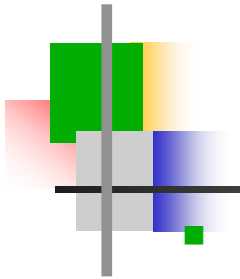
- 図(チャート)表現を用いて数字を生かす
  - 数字を記録し集計するだけでは不十分
    - 数表表現は直感的に理解できないので伝達効率が悪い
    - 関係者全員を巻き込む必要がある改善には図を用いるのが効果的
    - ソフトウェア組織ではなぜか数表が幅を利かせている
      - 会計・経理主導の数の扱い方の影響か？
  - 「数字をして語らしめる」を一歩進めて図を用いて問題を適切に表現する
    - できるだけ多くの人に共通の数字で改善意識を持たせるには図を多用するのがよい
      - 図の見方さえわかれば訓示や説教は不要か最小限ですむ
    - アメリカで私のやり方を話したらDeMarcoとParnasから古今の優れた事例を集めた次の美しい本を贈られた
      - The Visual Display of Quantitative Information, Edward R. Tufte, Graphic Press, 1983
    - 彼らから「日本人のほうが図を使うのがうまい」と言われた
      - 言われてみれば思い当たるふしがある

役に立つ改善のヒント(4)

- 優れたグラフ表現の条件 ― 前掲のTufteの本の冒頭より
  - 優れた統計的グラフは、複雑な内容をわかりやすく、正確に、効果的に伝えるもので、
    - データを表現し、
    - 方法、図のデザイン、及び書きやすさよりも、それを見る者にデータの実質的内容についての推理を誘導し、
    - データが意味することの曲解を避け、
    - 小さなスペースに多数の数値を表現し、
    - データ間の首尾一貫性を示し、
    - 他のデータとの比較したい気持ちを促し、
    - データの全体像から細部の構造に至るいくつかの詳細化のレベルを明示し、
    - 表現、探究、作表、または装飾において、明確な目的に奉仕し、
    - 統計的、及び言語による記述を密接に統合する
  - ものでなければならない

役に立つ改善のヒント(5)

- 目的のためにもっとも適切な図を用いる
  - 組織全体の目標と実力の関係を見るにはヒストグラム(度数分布図)
    - 簡易図法も有用
  - 解決の優先順位を表すにはパレート図を用いる
    - 改善には必須の図表現
    - 例えば欠陥(バグ)の要因別パレート図
      - 分類の直交性や粒度に注意
    - なぜかソフトウェア開発の現場ではあまり見当たらない
    - 習慣になるまで頻繁に書かせるないと自発的に書くようにならない
  - 改善の経過を見るには時系列グラフを用いる
    - 時間軸(通常横軸)は共通にする
  - 特に問題現象を制御するためには目標を時系列展開し、どの時点でも実績の目標からの偏差がわかるようにする
    - 自動制御理論の応用
  - いくつかの変数間でのバランスを見るためには多次元グラフを用いる
    - 例えば、Kiviat図、参考図-2, 3
  - 変化を率で見たい場合は*log-log*または*semi-log*目盛りを用いる
    - なぜかソフトウェア組織では滅多に*log*を用いない



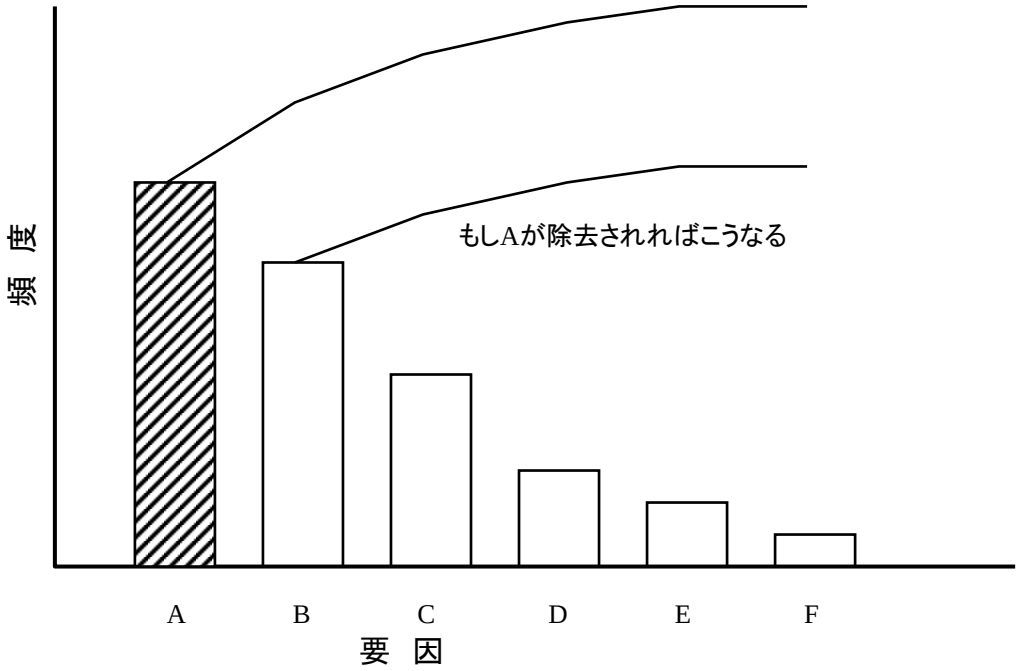
役に立つ改善のヒント(6)

- パレート図が契機となった改善の例
  - オフコンが華やかだった頃、私の親友のオフコン販売会社の常務が私のところにパレート図を持って現れた
    - それはビジネス処理が7つの処理パターンで全体の70%を占めることを示すものであった
      - 主な処理パターン：作表、ソート・マージ、選択、etc.
    - この70%を自動化して残りの30%はベテランに開発させたい
  - そこで私は、7つの処理の各々について、メニュードリブンのコードジェネレータをパソコンで作成することを提案した
    - 各メニューに示すべき変数を選び、その各々の価を、既成の800本のプログラムを実際に見て決めた
      - 常数にしてもよいもの
      - 選択すればいいもの
      - 数値を入れるもの
    - 仕様決定の合言葉は「現実のプログラムに帰れ」だった
    - 実際の運用には、各メニューについて変数を記入するフォームを作り、女子の機械的な作業にした
    - 70%の範囲におさまる処理の最短開発時間は、わずか6時間だった
    - 欧米で同じような問題解決例を探したが、ほとんどなかった
      - どうやら、彼らは、実証的なアプローチは苦手で、プログラミング言語による解決策を好むようだ
    - これが私の最初のコンサルティング経験

参考図-1

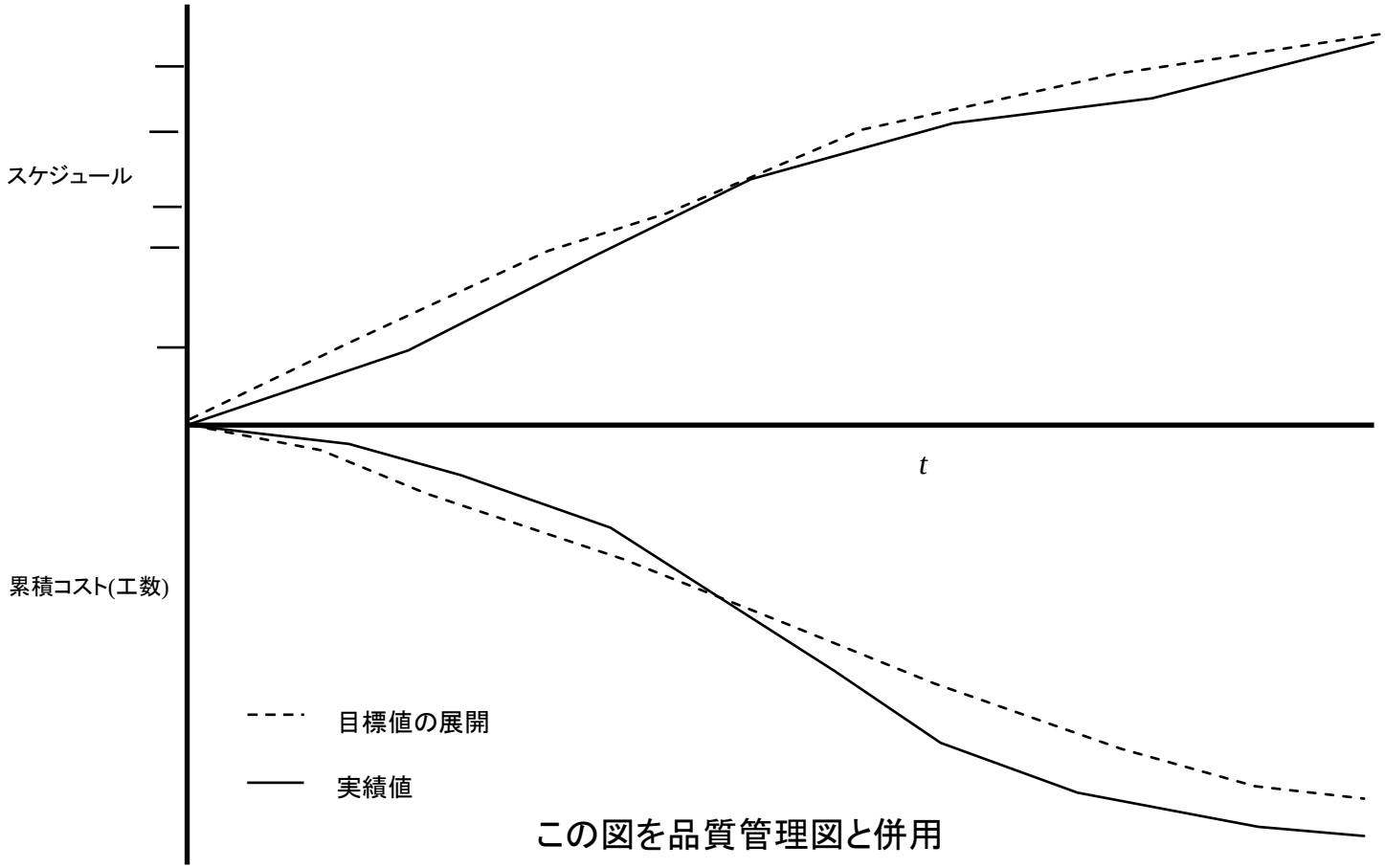
- 簡略ヒストグラム

- パレート図



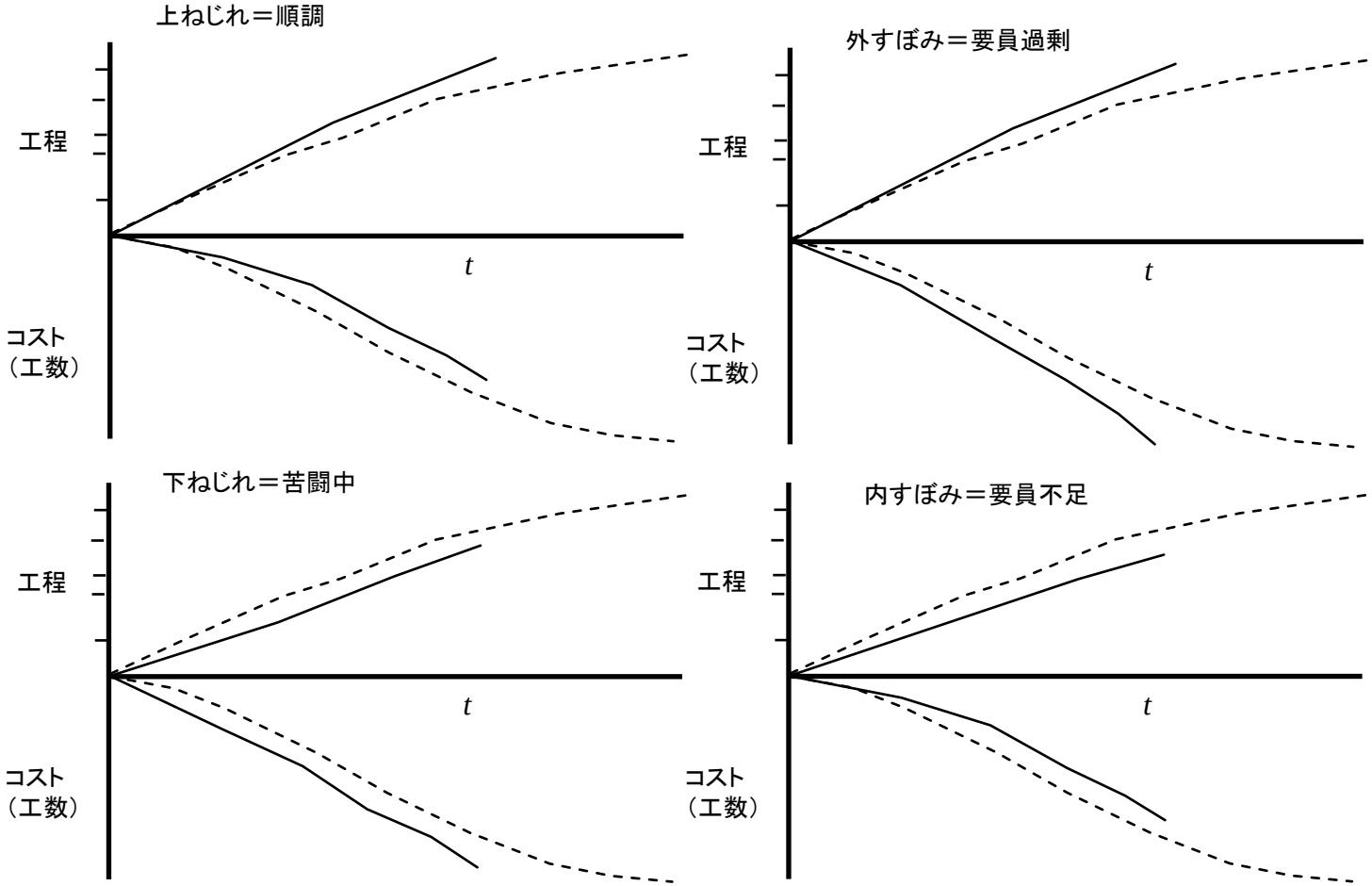
参考図-2

- 2次元(スケジュールとコスト)の制御例



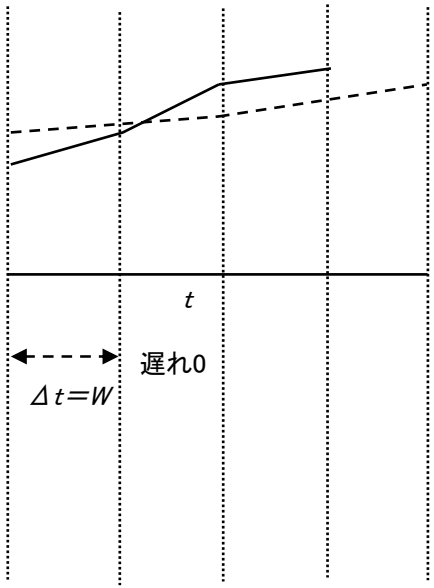
参考図-3

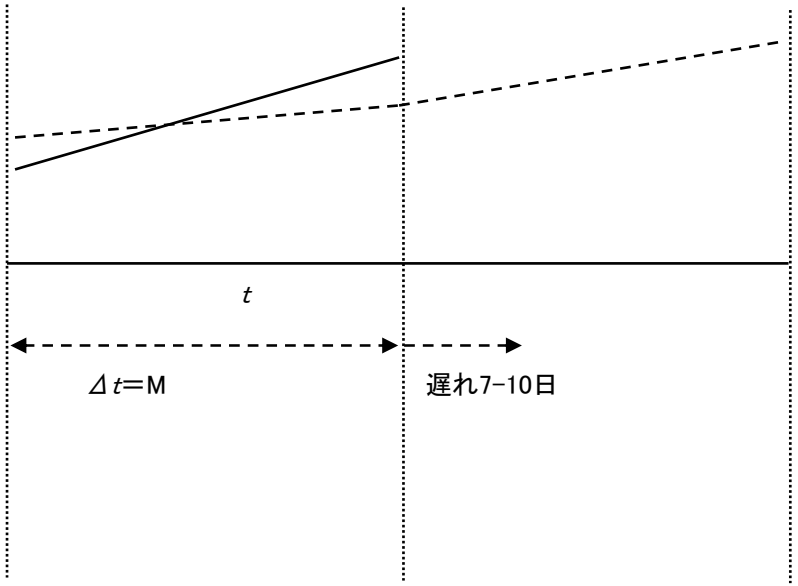
- 図の4つのパターンによって現象の違いが認識できる



参考図-4

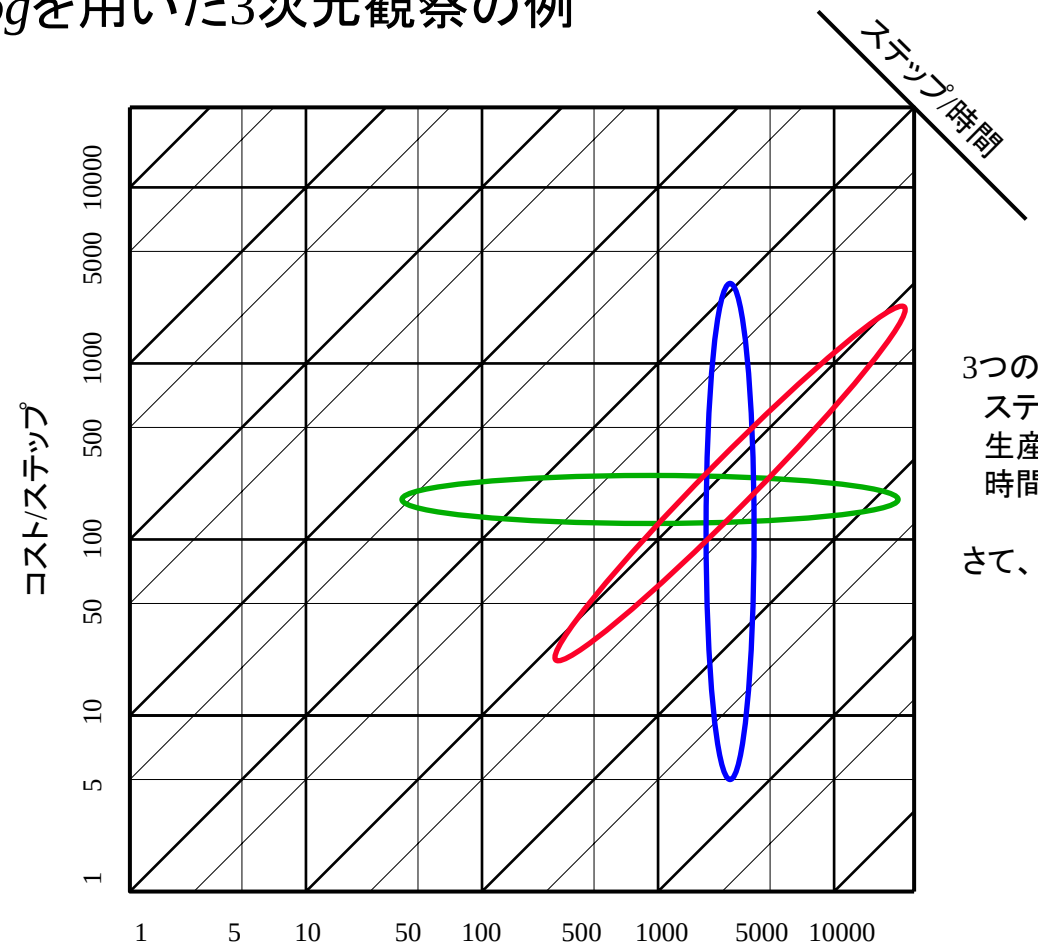
- 制御を可能にするには
  - 目標値の時系列展開と
  - データ間隔 Δtと時間の遅れをどのように決めるが重要





参考図-5

- *log-log*を用いた3次元観察の例

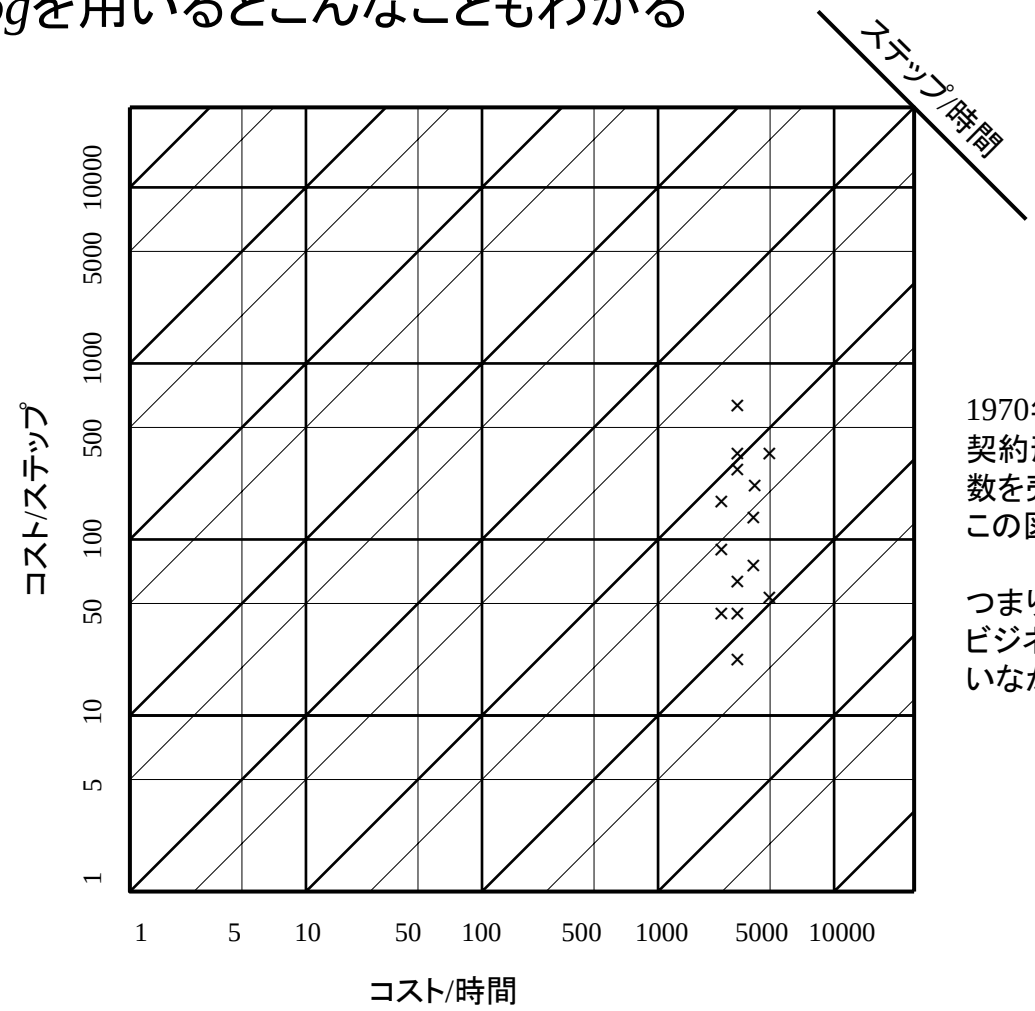


3つの可能性がある
- ステップ単価がほぼ一定
- 生産性がほぼ一定
- 時間単価がほぼ一定

さて、どれが正しかったか？

参考図-6

- *log-log*を用いるとこんなこともわかる



1970年代、受託開発は、契約形態に関係なく工数を売っていたことがこの図からわかった

つまり、生産性向上はビジネスを有利にしていなかった

役に立つ改善のヒント(7)

- 図が改善の道具として使われるようになればそれは生き残る
  - チャートのパターンで現象や対策を議論するようになればそれが使われている証拠
- メトリックス文化を醸成する
  - 数字の記録を厭わない、チャートがどこにでもある、チャートに基づいて改善を議論する文化
  - 機械工場ではこの文化があるがソフトウェア組織では珍しい
  - かなり執拗に教育しないとメトリックス文化は育たない
- 数字のごまかしは異なる次元から見ることで見破れる
  - 一般に管理が厳しい数字は望ましい方に偏る
    - 例えば、コストは目標に対して狭く分布し規模は広く分布する
- 最近の環境のもとでは
  - 最近よく見られるダッシュボード型
    - ある瞬間の状態を他次元的に見る
    - 時系列的観察と併用する必要がある
  - 多次元表現はやりやすくなった
    - 重ね合わせ
    - 色の利用
    - etc.

役に立つ改善のヒント(8)

- 図のもう一つの効用
  - 使えそうな図のフォームはどんどん盗め
    - 論文や発表はフォームの宝庫
    - 意識して眺めれば役立つものがたくさんある
    - しかもフォームを盗んでも誰も文句を言わない
    - 我々が論文に載せた図はすぐに世界で使われている
  - 使う場合の注意
    - 表現は簡単でも数字の把握が困難なこともある
      - 例1：プロダクトのビジネス経過を、損益＝（開発投資＋エンハンス費＋販売経費）ー売上、で見る図では、販売経費を拾い出す作業が必要になる
      - 例2：図の意図を理解しないで真似をすると役立たない（参考図-4）
  - 日本発の図のフォームも欧米で結構使われている　ー　例えば
    - 原因/効果分析図（魚の骨チャート）
    - バグ死滅曲線
- 一つの興味ある実話
  - JECC(日本電算機)レンタルと自社レンタルの利益の違いを示した提案を実施しコンピュータビジネスを赤字からドル箱に変えた
    - 積分方程式で比較計算し利益の大きな違いを図示
      - 不況で販売実績がないとJECCレンタルでは利益ともゼロ、自社レンタルなら稼動台数にリンクして利益が得られる
    - 私の同僚が三たび提案
    - 三度目にそれに目を留めたのが三田勝茂

役に立つ改善のヒント(9)

- 生産性を目標とすることについての疑問(参考図-6に関連して)
  - ハートウェア工場では生産性向上は重要な目標の一つ
    - ハードウェア生産の場合は分子/分母の定義が明確だから問題ない
      - 月産何トン、月産何台、1工場何個、etc.
  - ソフトウェア組織でも生産性が目標として示されることが多いが…
    - ソフトウェア開発でアウトプットとは何かを深く考えた上でそれを目標としているとは思えない
    - LOCにしてもFPにしても生産物を客観的に規定する尺度としてはあまりにも精度が悪い
      - ハードウェア生産でのトンや建築での建物面積よりはるかに精度が悪い
    - 分子と同様、分母(通常時間を用いる)も正確に測定できない
    - 同じ組織で継続的に観察する場合でも同じ条件で測定するのは難しい
      - 例えばプログラミング言語の以降や混合、再利用部分の評価、など
    - まして他の組織間で比較することには無理がある
    - デスマーチプロジェクトでは通常生産性が大幅に上がる
      - 能力の高い助っ人が最高瞬間風速でコードをアウトプットするためか
      - 米国では支払われない残業時間を除いて生産性を高く見せる例がある
- 生産性より重視すべき尺度
  - サイクルタイム -- 類似のプロジェクトが継続する場合
  - チーム規模 -- 常に同じくらいのチームで作業する場合

役に立つ改善のヒント(10)

- 目の子算(概算、rule of thumb)の奨励
  - 見積値は1桁精度で十分　ー　桁さえ合っていればよい
  - そもそも無形物では規模も工数も正確に測れない
  - 細かく測定するとかえって実態が見えなくなる
    - 例えば、リワークの測定
  - むしろ重要なのは誰でも覚えられる目の子、例えば
    - 家を見積もるときの㎡単価、機械を見積もるときのﾄﾝ当たり単価は大いに役立つ
    - ソフトウェアで常識として記憶すべき数値は、例えば
      - 時間当たりの平均単価(1桁精度で十分)
      - ステップ単価(1桁精度で十分)
      - バグ密度(1桁精度で十分)
      - etc.
    - 改善効果が表れればこれらを更新する
  - これがあれば、見積もり工数、コスト、及び品質を暗算で計算できる
    - 営業は客先で確保すべき予算を即答できる
    - 改善の効果もすぐに計算できる
    - 計算尺の時代は計算図表を用いた
      - 図に定規を当てるだけでいくつかの変数の価を決めるのに用いた
        - 例えば、規模、時間単価、ステップ単価が同時に求められる

役に立つ改善のヒント(11)

- メトリックス制度を完全に実施するコツ　―　会計などの必須の事務手順に統合する
  - せっかくメトリックスの制度を作ってもなかなか実施されない、という悩みをよく聞く
  - 確実に記入させるには、誰もが必須と考えている会計・経理の手続きを利用するのがよい
  - プロジェクトアカウントのコストチャージを報告する書類にコスト以外の重要数値の記録欄を公式に追加する
    - 工数の期間別予定と実績
    - 外注の期間別予定と実績
    - バグの目標と実績
    - etc.
  - そして必須記入欄が空白の書類は受け取らないように経理担当者に教育しておけばよい
    - 書類が受理されなければプロジェクトは完了しないから必ず記入するようになる
    - セクショナリズムの壁を排することが組織を挙げての改善

役に立つ改善のヒント(12)

- 下痢しない改善は本物でない(1)
  - 枝葉でなく根本から改善する
    - 最初にやったことは、工場長が率先して「業務のありかた」をまとめ、自ら工場長職務規定を執筆した
      - その意図は権限委譲は上から降ろさねばならないため
      - 「業務のありかた」は、職位別の基本分掌と、職名別固有職務からなる
      - そして、seinとsollen、という言葉で基本があってこそ応用が生きることを教えた
  - 根本的な改善をやって少々失敗するのは当たり前
    - これが「下痢」の意味
  - 機械工場でやった改善(1)
    - 管理サイクルを月から週に
      - 1-53週の週カレンダーを随所に
      - 期限表示も作業量調整も週単位
    - バリューアナリシスによる原価低減
      - これを担当したのは法科出の主任 ― いまは建機社長を経て建機工業会会長
    - 運搬工の廃止
      - 作業票に表示された機械(例えば旋盤)にフォークリフト運転手が部品を運搬
    - 工場の生産管理に統計会計システム(PCS)後にコンピュータを導入
      - プログラムミスで給料支払いが遅れたが工場長が庇ってくれた

役に立つ改善のヒント(13)

- 下痢しない改善は本物でない(2)
  - 機械工場でやった改善(2)
    - 管理帳表、伝票の全面改訂
      - パンチカードを伝票そのものにした　ー　日本最初
      - 回収されたカードをマシンに通せば決算ができる
        - 基本を週決算にし、月末決算は端数日を加えるだけ
        - 締め切りの翌日に決算がまとまる
    - ファイリングシステムを工場全体で実施
      - 事務机、ファイル、を特注し、2,000万円かけてラックを捨ててファイリングキャビネットに置き換え、工場全体のアーカイブまで作った
  - ソフトウェアでも同じ
    - 改善活動の初期には一時効率が下がる
      - それをトップは理解しなければならない
    - 現場でのRSSTは重要だが時には根本的な改善を一気にやる必要があることがある
      - 開発環境の導入や大幅な機能アップ
      - プロセスの大幅な組み替えや削除
      - etc.

役に立つ改善のヒント(14)

- コンピュータ工場での改善経験
  - ボード配線のNC化で導通検査を全廃
    - 廃止以前はゴムハンマーでの半田屑落としと1本1本の導通検査で莫大な工数をかけていたがNC配線で事実上誤りが“0”になり検査を全廃
  - コンピュータ設計・製造に自動設計(DA)を導入
    - 最初は部門毎にシステムを構築したので論理設計と実装は繋がっていなかった
    - このため設計変更に膨大な工数がかかり誤りも多かった
    - そこで大規模な設備投資をしてシステムを統合した(1970年頃)
      - 論理設計の誤り訂正が実装ファイルに反映され、実装の誤りも論理図に反映されるようになった
      - 技術革新で設計条件が変化するとDAソフトが追従した
        - 例えば、プラッターの信号線の制約がホール間に2本から3本になるとすぐにそれを配線ロジックに反映させた
      - その他の設計作業のほとんどをコンピュータでやるようにした
      - その頃すでに設計者にとってコンピュータは必須のインフラであった
    - メインフレームでは世界で競合できるコスト低減をやっていた
      - メインフレームの時代が終わってもしばらく輸出し続けた
    - メインフレームの消滅と共にこの技術は失われた

役に立つ改善のヒント(15)

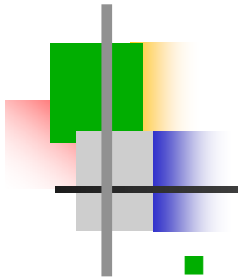
- 後戻りの防止に神経を使う
  - 最初から歯車が逆転しないように歯止め(ratchet)をかけることを考える
  - 効果的な歯止め
    - 教育、競技会
      - 玉掛け、重量、距離などの目測
    - 制度化、メカニズム
      - 「通達で仕事をするな」
      - 柔軟な工場規格の改正
    - 新鋭工作機械、新製造法の導入
      - 例えば製缶工場でリベット止めの廃止によって板金の仕事をなくし倣い溶接機に置き換えた
        - このため、板金工全員がNCプログラマに職種転換した
  - ソフトウェアの改善でも同じはずだが…
    - 人に頼る部分が大きいので現実には後戻りが頻繁に見られる
    - CMMが制度化を重視するのはそのため
    - 開発環境で制度化を補強するのがもっとも効果的な後戻り防止策
      - レベル3以上では環境でのプロセス支援は必須

機械工場向きのアプローチ(1)

- 工場の整理整頓
  - エントロピーのところで述べたように、部品数を制限して工場内を整理整頓することは、生産性、品質に直結する
  - 一目でわかる
- 日米の自動車生産の生産性の違いが問題になったとき、アメリカから日本の自動車会社に調査団が来た
  - HBRに載ったその調査報告が、私の印象に残っている
  - 執筆者はMichael Cusumano
    - 彼はこれで味を占めて「日本のソフトウェア戦略：三田出版会」を書いた
    - 「何より驚いたのは、工場の清潔さであった。始め、日本の自動車工場にはアメリカでやっていない秘密があると思っていたが、そんなものはなく、こんな当たり前のことに大きな差があったのは、我々にとって大きなショックだった」
- ソフトウェアの開発現場での整理整頓はどの程度生産性や品質に影響を与えているだろうか？

機械工場向きのアプローチ(2)

- 標準時間、標準原価
  - 日立亀有工場では標準時間、標準原価を用いてコスト低減した
    - 加工伝票にはあらかじめ決められた標準時間(ST)が書かれている
    - 作業者は実際にかかった時間でなくSTで賃金をもらう（ただし組請負いで）
    - 従って、原価算入もSTである
    - つまり、製品の製造原価は、STを展開した段階で決まるから、原価低減は机上でもできる
    - STは、組長と生産技術の立会で実測して実験式を作りそれで計算する
    - 中には名人芸を要する、つまりベテランと素人で10倍以上の差がある作業(例えば歪見取り)があるが、それでも、敢えて中央値をSTとする
    - 組長は、そのST以内で作業できるように作業者を指導するが、それによって組全体の収入は増える
    - 定期的にSTを測定し直して改訂するから、次第に実績がSTの周りに集まってくる
  - これを不完全な形ながらソフトウェアの見積もりモデルに応用した
    - ツール開発申請の時の効果を参考にして、あるプロセスの標準時間を削減する
    - この見積もりモデルが適用できれば工数は低減したことになる



SPIで特に強調したいこと(1)

- 記録の重視と記録の習慣づけ
  - ハードウェアの技術移転は物理的な「物」を介して行われる
    - 従って、物か図面さえあれば同じものが作れるし設計意図もわかる
    - 欠陥も直感的に指摘できることが多いので改善ループも速く回る
  - ソフトウェアでは「物」に頼れないので文書でしか技術移転できない、にもかかわらず
    - ほとんどのソフトウェア組織でプロジェクト記録を残す習慣がない
    - ソフトウェアシンポジウムの論文投稿に毎年苦労する
      - 条件を緩和してPowerPointでよしとしても応募が少ない
    - 技術やノウハウはすべて個人の頭の中で狭い範囲しか伝わらない
    - 当然、外部の技術を学んで使おうともしない
      - ソフトウェアエンジニアリング関連の本が売れない
    - かくして「賽の河原」は起こる
      - 人やチームの組合わせが変われば振り出しに戻りトラブルはくり返す
      - プロセスが一時的によくなってもすぐに後戻りする

SPIで特に強調したいこと(2)

- 技術論文を書く習慣は大きな波及効果を生んだ
  - プロジェクトの技術やノウハウを残すためにそれを仕上げたら論文をまとめることを原則とした
    - プロジェクトの数は予算を立てるときにわかるから、それを部門別の論文作成目標にする
      - 期の途中で取り下げ申請をしてきても認めない
        - 忙しい、担当者がいなくなった、などの理由は技術を残さなくていい理由にはならない
    - これを長く続けているうちに質が向上し参照件数が増加した
  - 意外な副産物
    - 論文を迫力あるものにするために自発的に数字を記録しだした
- 大きな波及効果
  - 国内海外の論文発表が増加し知名度が上がった
  - これらを種にして、米国の会社とGive and takeの２社間技術討論会を実施した
  - 海外の学会活動も増加した

SPIで特に強調したいこと(2)

- ピープルウェア
  - 「実際のところ、ソフトウェア開発上の問題の多くは、技術的というより社会学的なものである」

    Peopleware, Tom DeMarco & Timothy Lister
  - ソフトウェアで特に重要なこと(1)
    - 優れた人材　ー　日本の採用のやり方は日本独自なもので西欧人はその場面を見て驚愕した
      - なぜ日本で成果物中心で、もし雇われたら共に仕事をする人たちと議論するオーディションが普及しないのか？
    - オフィス環境　ー　改善されたのはごく一部、相変わらず多い大部屋
      - リストラで悪化している面さえある
    - 楽しい職場　ー　受身のISO 9000認証取得、CMMレベル達成活動はつまらない職場にする元凶
    - チームの結束　ー　結束を妨げる7つの原則は未だに健在
      - 自己防衛的な管理、官僚主義、作業場所の分散、時間の分断、品質低減製品、さばを読んだ納期、チーム解体の方針
    - 個人のプロセス改善　ー　Personal Software Process, PSP
    - チームのプロセス改善　ー　Team Software Process, TSP

SPIで特に強調したいこと(3)

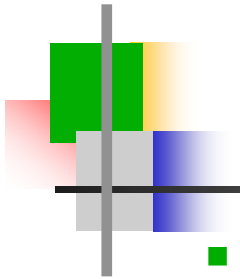
- ソフトウェアで特に重要なこと(2)
  - 教育・訓練　―　私が好む教育・訓練方法
    - かつてLaszlo Beladyが日本IBMに在籍していた頃、ソフトウェア科学会創立基調講演で、ソフトウェアで技術移転がうまくいかない原因の一つは、問題と解決策を別々に提示することにある、と次のように述べた（これは学会誌の創刊号に載っている）
      - 通常の教育では、聴衆が抱えている問題とは無関係に解決策だけ述べられる
      - これを防ぐ方法の一つにconsultative trainingがあり、すでにこれを実施している組織がある
    - 座学の場で居眠りする聴衆が多いのはそのためである
    - 事情の許す限り、これをやるようにしている
      - 参加者に自らが抱えている問題をまとめさせ事前に提出させる
      - 最初に共通問題で参加者をグループ分けする
      - 各グループで討論させ講師はグループを回り助言する
      - グループ毎にまとめた結果を発表させる

SPIで特に強調したいこと(4)

- 「我慢」や「何もしない」は人を育てる前向きな選択肢の一つ
  - 警句:どんなことでも最後までやり通し、自らが下した決定の結果を見ることで、それが経験として身につくものだ
    - もし、判断を過った人を最後までやらせずに途中で外すと挫折感というマイナス経験だけが残る
    - かくして失敗をプラス経験にした優れた管理者が育つ
  - これを、会社設立の初期で管理経験者がゼロだった頃の管理者育成方針にしたことをIEEE Computerの論文に書いた
    - これがYourdonから注目され引用された
    - アメリカでは途中で外すのが普通らしい
  - 最近の受身人間の蔓延は、敢えて自らやらせるのを怠り、脇から介入し過ぎ、自立心を育てなかった結果だ
    - 大火傷をさせたくなければ敢えて小さな火傷をさせる
    - 子供同士の喧嘩に親が出ていくおかしな風潮
      - 喧嘩をしらないから限度を知らず人を殺すまで暴力を振う
    - 雨で濡れたら傘を持たせなかった親のせいにする
  - 自ら変化への挑戦に立ち向かう人が数多くいなければとても改善などできない

SPIで特に強調したいこと(5)

- ハードウェアでは不利な情報インフラの効果的な利用
  - ハードウェア生産では「物」自体と「物」に関わる情報を統合するのは難しい
  - ソフトウェアは情報だからそれは可能である
- ソフトウェア開発では、「プロダクト」そのものが情報である
  - プロセスの過程で加工され次第に完成度を高めていく姿を、ほぼ完全に捕らえることができる
    - 各マイルストーンでの日付け、コスト、品質
    - ツールの選択と利用
    - コンポーネントの利用頻度や品質
    - etc.
  - この考えで構築する開発環境は当然開発に必須のものとなる
    - 開発環境のレベルがプロダクトのサイクルタイムや品質を決定づけるようになる



まとめ

- やるからには楽しくやろう
  - 楽しくやるには前向きに
  - プロセスを変えてみるのは面白い
- 要は歯車が噛み合って改善が回り出すこと
  - 切っ掛けはどこからでも(トップダウン、ボトムアップ、etc.)構わない
- しばしば人や文化の絶妙な組合わせが成功を導く
  - 亀有工場の場合、例えば
    - すばらしいトップ
    - みんなが合理化を考えよいことは自然に拡がる文化
    - 私のような若造に思いきったことをやらせる文化
  - が素晴らしい成果をもたらし、そこから人材が輩出した
- 気持ちのゆとりを持とう
  - ゆとりがなければ改善はできない
- 常に刺激を受ける場に身を置こう
  - これからもこのようなイベントに参加し続けてほしい