System Creates

ソフトウェア開発の
潜在能力を活かすためのプロセス改善へ

～プロセスを自在に設計することの時代的意味について～

SPI　Japan 2011 基調講演資料

2011年10月26日

株式会社 システムクリエイツ
代表取締役 清 水　吉 男
shimz@nifty.com
URL=http://homepage3.nifty.com/koha_ hp
派生開発推進協議会 代表
URL=http://www.xddp.jp

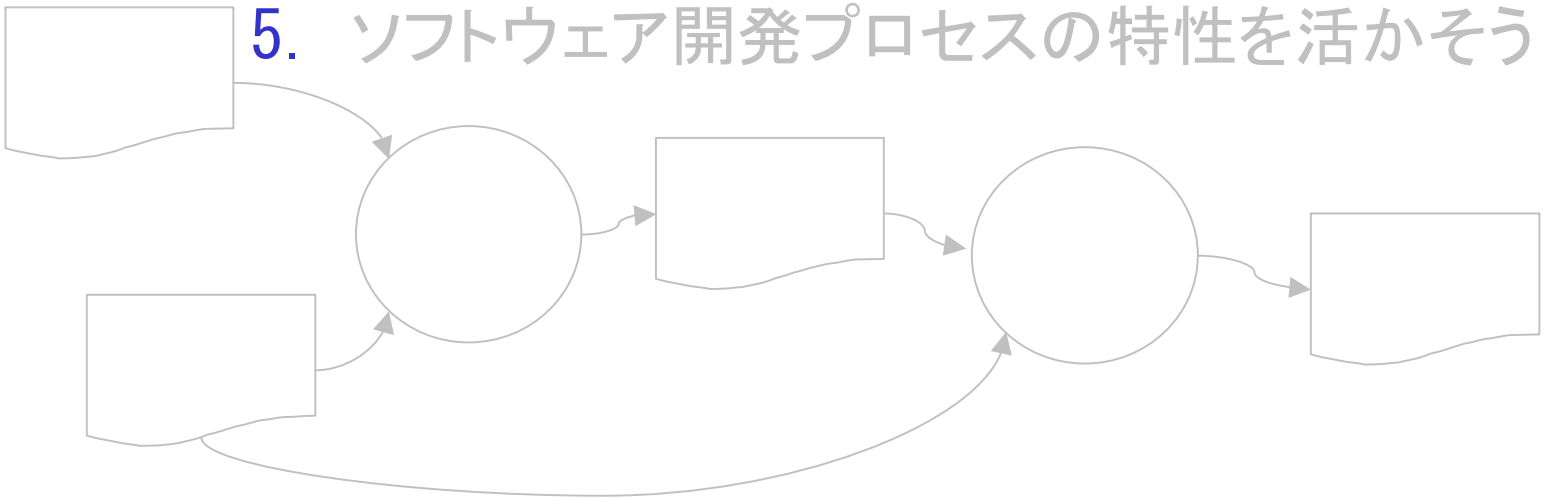
agenda

1. 私とプロセス
2. プロセス改善の取り組みの不思議
3. プロセス改善活動の難しさ
4. プロセス改善の今後の方向（提案）
5. ソフトウェア開発プロセスの特性を活かそう

USDM：Universal Specification Describing Manner の略. 仕様がモレない表現方法として筆者が開発したものです.
XDDP: eXtreme Derivative Development Process の略. 派生開発に対応するように筆者が開発したプロセスです.
PFD　:Process Flow Diagram の略.筆者がDFDをアレンジしてプロセスの設計に使いやすくしたものです.

System Creates

1. 私とプロセス
2. プロセス改善の取り組みの不思議
3. プロセス改善活動の難しさ
4. プロセス改善の今後の方向（提案）
5. ソフトウェア開発プロセスの特性を活かそう



System Creates

「22年間プロジェクトの遅延なし」の背景

- おもいっきり自慢話
  - 40年間 “営業しない” を貫いた
  - 現役の22年間（40数例）プロジェクトの遅延なし
  - 組込み時代は週35時間でメインの顧客に対応
  - KLOC当たりのバグ発生率　＜　0.1
  - 健康状態に問題なし
  - ソフトウェア技術者として楽しい40年間
- 仕様モレが生じにくい要求仕様の表記法を確立
- 技術的準備は事前の稽古で対応

でも、それだけで「22年間納期遅延なし」は実現しない

DFDをプロセス設計に活用

- DFDは、もともとシミュレーションのツール
- DFDを思考の道具にする
  - プロジェクトの最初の分析手法として使うだけでは習得できない
  - まず、日常の作業や方策の検討に活用
  - さらに、プロジェクトのプロセスの表現に活用
- 後に、同じ取り組みを文献等で確認
  - 「システム設計の構造化手法」
    - (Brian Dickinson著、黒田／渡辺　共訳) 日経BP 1988
  - CMM (TR25-J99) のP70

System Creates

Evolutionary Scheduling への展開

- DFDでシミュレーションを繰り返したあとでスケジュールへ
  - 見積もりデータと組み合わせて半自動で展開できる
- 途中で変化する要求に対してDFDとスケジュールに対応
  - 現実が見えたところでより良いプロセスが考えつく
- DFDとセットで「代案」を考え出す
  - 同じ結果を得る別の方策を考える
  - ３日遅れれば5日取り返す方法を考え出す

「22年間納期遅延無し」の背景

変化する要求に対して開発プロセスも変化させる必要がある

「XDDP」は複数のプロセスを組み合わせたもの

- 顧客の要求
  - 3ヶ月という期間
  - 40数項目の追加と変更
  - 初めてのドメイン、初めて見るソースコード

- 1週間で、既存のプロセスを組み合わせ、シミュレーションで確認
- 追加と変更の合理的な2種類のプロセスに分けて品質と生産性を確保した
  - 品質 ・・・ レビューをしやすくして思い込みと勘違いに対応
  - 生産性 ・・ 必要不可欠な成果物とプロセスの連鎖で構成

「USDM」をExcel にした理由

- 従来は「要求書」から「要求仕様書」へ文書を書き換える中で仕様化していた
  - 書き換えるための工数が必要で品質低下 → レビュー工数増

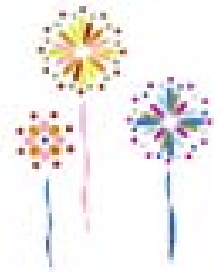
規模の増加を予見

- 品質と生産性の確保から、Excel に書くことを選択
  - 階層構造をスムースにする
  - 追記しやすい方法を確保
  - Excelの「グループ機能」でレビューの効率を上げる

「仕様数が増えるのだから、工数も増えて当然」と考えれば、このアイデアは生まれなかった

System Creates

一世一代のプロセス設計



2つの新規開発プロジェクトを完全に重ねる

- 現役の最後に一世一代のプロセス設計

Project	A	B
設計	構造化分析ー設計のシームレスな方法	文書を展開して詳細化する方法
要求仕様	アーキテクチャ設計には仮想仕様で着手、途中で製品の仕様に置き換え	最初から要求仕様書が用意
その他	構造化手法の教育から実施 仕様の置き換えで混乱しないためと、保守性を考えてデータをタスクに収容	設計書・・・ Review Less 1タスクの設計書を1冊に集約 全タスクの設計を並行作成

- 開発プロセスの設計はソフトウェアシステムの「設計」と同じ
  - 今回の要求を満たす自在な設計スキルが共通する

System Creates

私のコンサルティングの方針

- QCDの同時達成を目指したプロセス改善
  - USDMやXDDP、PFDなどはそのための中心的手段
- ケース1：XDDPで今の問題を解決したい
  - USDMやXDDPのレクチャーからコンサルティングに入る
- ケース2：XDDPでプロセスを改善したい
  - 条件1：支援組織が準備されているか、これから準備する
  - 条件2：「PFD」によるプロセス設計のレクチャーの受講が必須
  - 社内でUSDMやXDDP、PFDのレクチャーができる人を育成

自分たちでXDDPの取り組みを回すための対応策

System Creates

コンサルティング事例（一部）

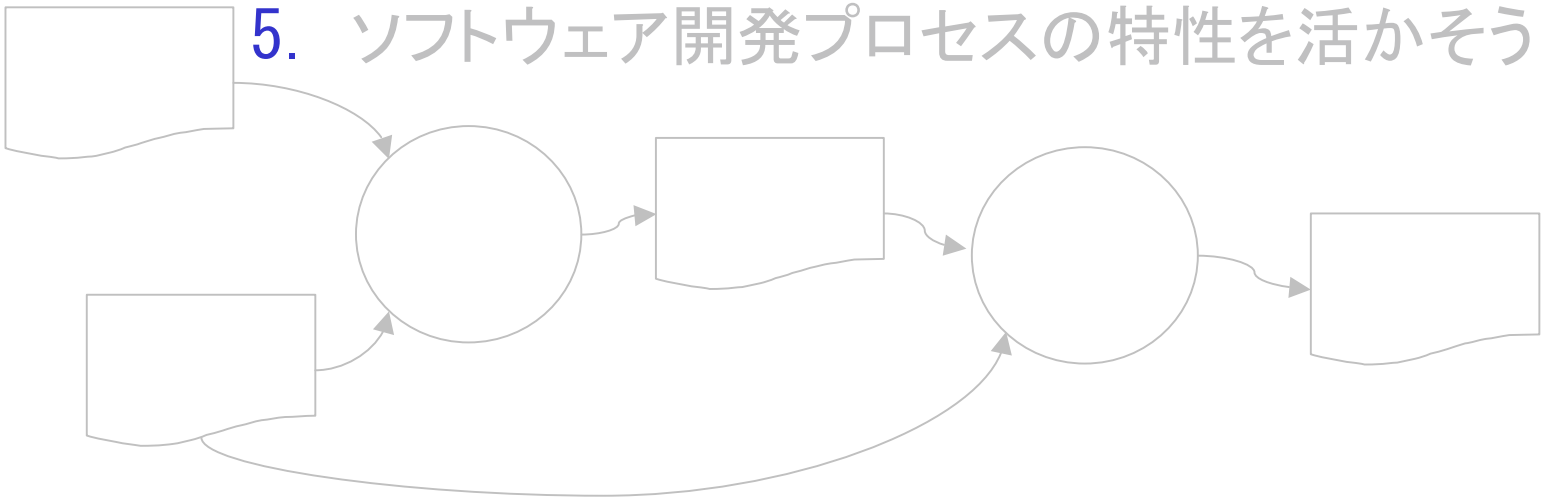
- 100%間に合わない→方針転換から6ヶ月で終了
- 同じ部署の20事例のプロセスがすべて異なる
  - すべて「On Schedule」
- 従来は2年かかっていたケースが10ヶ月で終了
- 急遽確保した一人のSEで先に終了
- 年間1000件の市場バグが、10ヶ月で1件に減少

：

上記は、すべて「XDDP」を適用したケース

System Creates

1. 私とプロセス
2. プロセス改善の取り組みの不思議
3. プロセス改善活動の難しさ
4. プロセス改善の今後の方向（提案）
5. ソフトウェア開発プロセスの特性を活かそう



System Creates

ソフトウェア開発で「生産性」を計測していますか？

- 会場の皆さんにお尋ねします

① 生産性を意識して開発したことはない？

② 生産性の計測を指示されていない？

③ 計測している？

「品質」と「生産性」は相反すると考えられている？

- これまでのプロセス改善の取り組み・・・「品質」が中心
  - 「品質」のためにプロセス（作業）が増える傾向にある
- 品質はプロセスで対応し、増加する「量」には作業分担を狭くして対応してきた
  - 組織も細かく分けられたところもある
  - 機能が増えるのだから作業工数も増えるのは当然？
- 「生産性」を除外しては合理的なプロセスにならない
  - 増えるしかないプロセスと減らすことのできるプロセスがある

「プロセス」に対して「非人間的」と揶揄される原因にもなる

System Creates

ソフトウェア開発で生産性の計測は無意味ですか？

- コード行数による生産性の計測は無意味か？

コード行数 / 全工数

- コード行数は故意に増やすことができる
- 他のプロジェクトのデータとは要件が異なるので比較できない

生産性データが顧みられることがなくなった

工数を減らす方法は議論されていない

- 仕様変更やバグなどの手戻り工数の要因になっているプロセスを減らす
- 成果物の構成を変えたり、無駄なプロセスを削除できないか

個人の意思で収集することまで妨げるものではない

System Creates

なぜ、「生産性」にこだわるか

- 企業の利益の源泉は「生産性」です
  - ソフトウェア開発も枠外ではない
  - 品質改善の取り組みも、生産性を犠牲にしては定着しない
- 生産性の計り方に2種類ある

①	時間ベース	生産量／投入時間	個人レベルで収集可能
②	コストベース	売上げ／投入コスト	統計データ、企業組織で収集

- 組織・・・コストベースで生産性を算出できる

ストップウォッチが机の上から消えた

アジャイルプロセスの提案の背景にあるもの

- 1990年代の初めに「納期の要求」が発信
  - 「品質」を確保する方法はわかった・・・次は「納期」で勝負
  - スモールプロジェクトもある

対応策？

方策	方向	施策
方策1	プロセスの多様化	アジャイルプロセスの提案
方策2	生産性の向上で対応	？？？

現実には・・・・

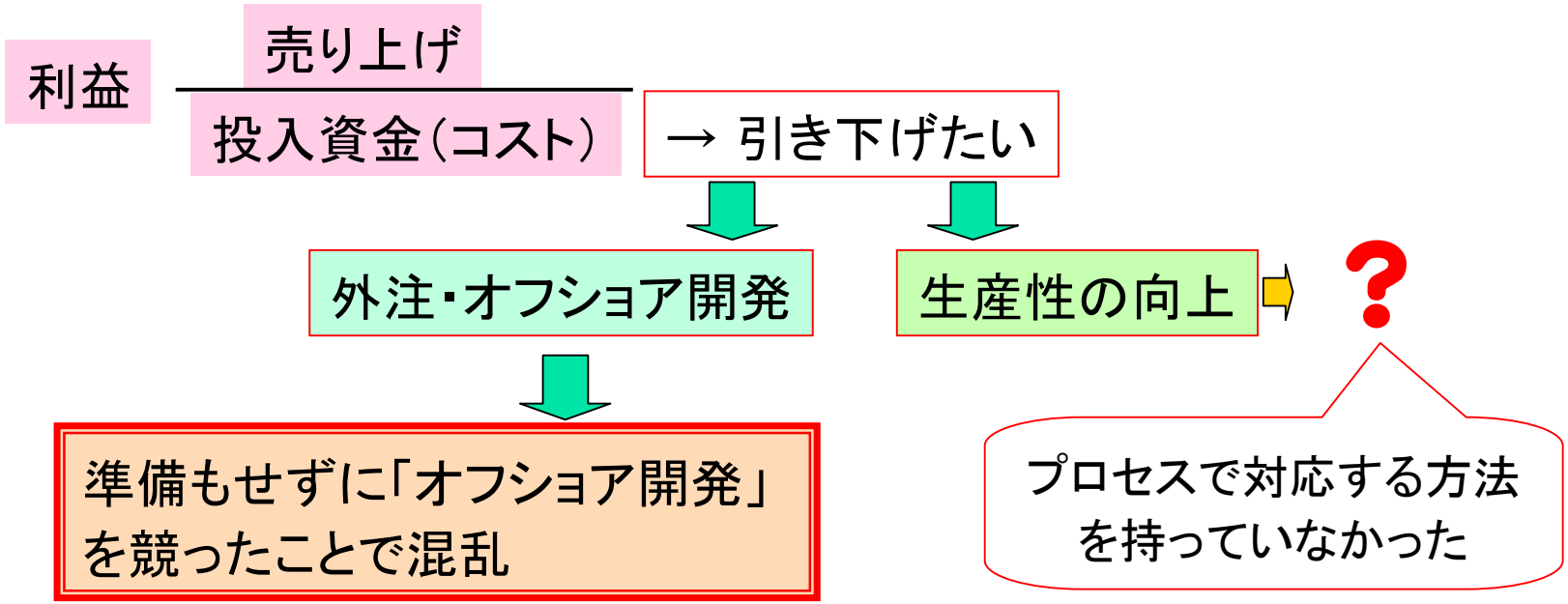
１人の分担範囲を狭くして「人海戦術」で対応

その結果・・・・

「品質」と「納期」に応えるためのプロセス技術は発展せず

安易にオフショアによる開発が選択された

- 1997年のアジアの金融危機の後に「コスト競争」が激化
  - プロセス改善によって各自の「時間ベースの生産性」向上で対応してこなかった



「オフショア開発」のコスト削減効果は検証されているのか？

外に出すものが逆？

- 製品を差別化するもの ・・・・ 性能の向上や使いやすさ、操作性、理解生、保守性など、すべてソフトウェア次第

「Winning with Software」(W.Humphrey)

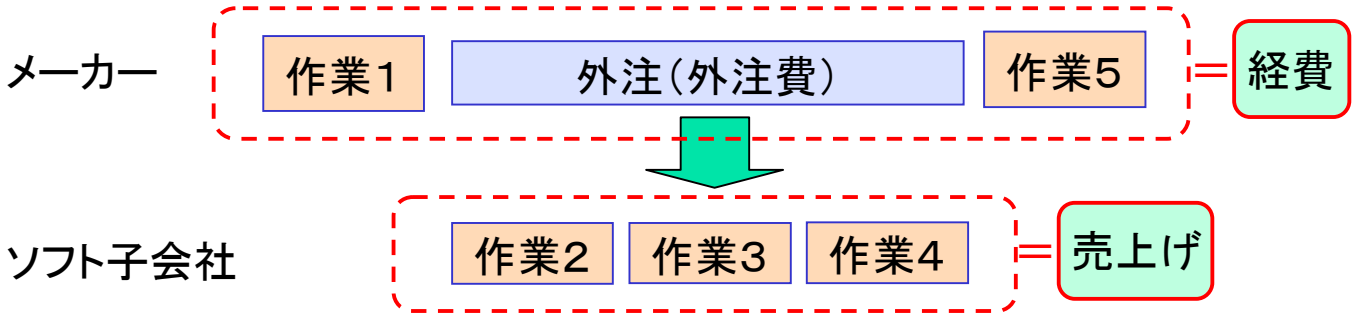
- 利益の源泉である「ソフトウェア開発」を外注(オフショアを含む)に出している

この状態で、どうやって競争相手を出し抜くソフトウェアを作れるのか

- ソフトウェアが作れると思って入社したのに(新入社員の声)
  - ２年目からソフトウェアの外注の取り次ぎ担当に

プロセス改善活動の障壁

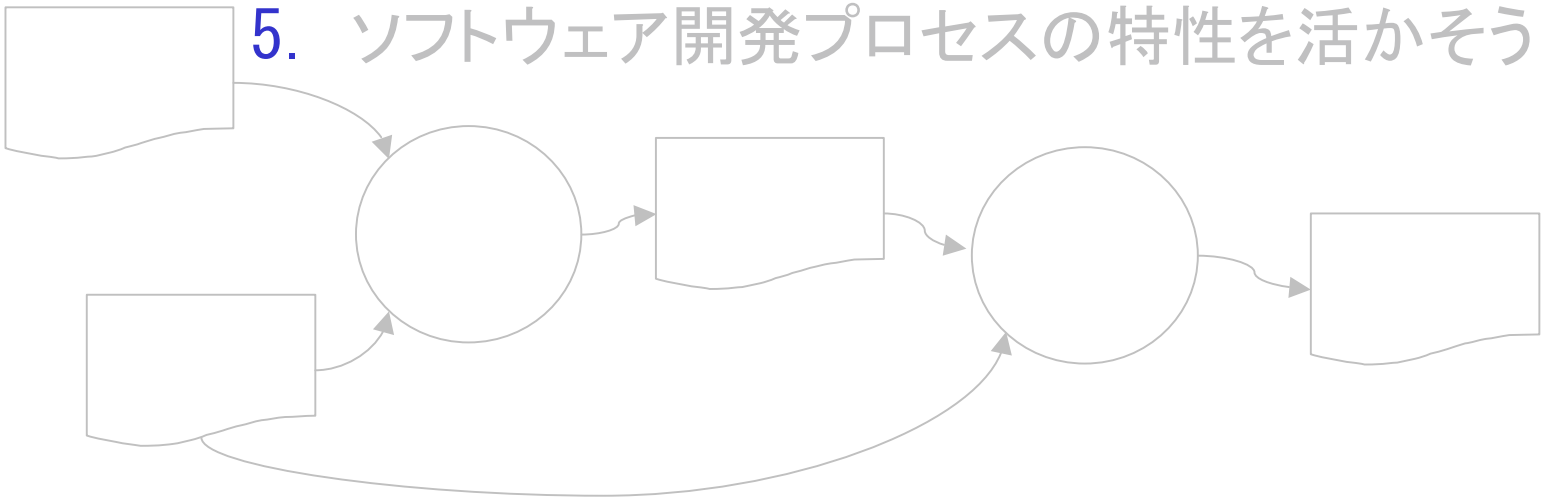
- 経費が子会社の売上げにかわる



- 組織が分かれたことでトータルで品質や生産性の改善を仕掛けにくくなった
  - 上流担当:「やっぱ、上流からでしょう」
  - 下流担当:「それじゃ、うちの仕事がなくなってしまう」

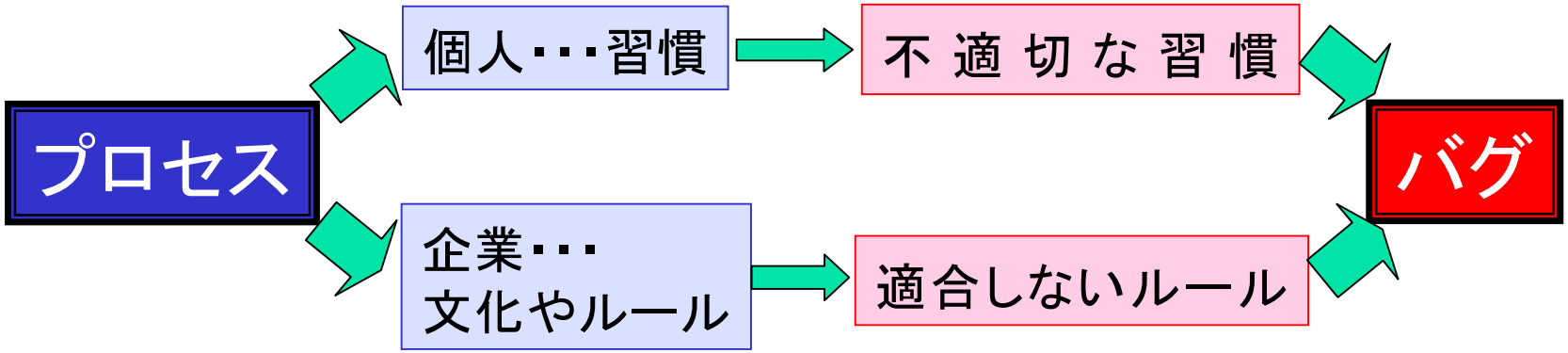
System Creates

1. 私とプロセス
2. プロセス改善の取り組みの不思議
3. プロセス改善活動の難しさ
4. プロセス改善の今後の方向（提案）
5. ソフトウェア開発プロセスの特性を活かそう



「プロセス」はどこにある？

- 「プロセスってどこにあるか知っていますか？」
  - 「それなら、組織標準に定義されていますよ」



- プロセスの改善は、身に染まった習慣やルールを変えることでもあるが・・・

どうやって身に染みついた
習慣を変えるの？

ルールを変えるルールがない

System Creates

何のためにプロセスを改善するの？

① 混乱した状態を鎮めて品質を向上させるため

- 無秩序なソフトウェア開発では品質は確保できない

この時、多くの現場では「生産性」の向上を置き忘れてきた

② 市場の要求の変化に対応させるため

- 社会の状況変化を受けて新しい要求が発信される
  - リーマンショック・・・コストの削減
  - 「9.11」・・・セキュリティ機能の強化
  - トヨタの問題・・・機能安全の強化
  - iPhon, iPadの普及・・・使い心地の追求

- これらの要求に応えるための合理的なプロセスの構築

System Creates

プロセスの変化を回避した

- 変化する顧客の「要求」(納期やコストを含む)に対応するには、開発プロセスも「変化」させる必要がある
  - 同じ要求は二度とない
- 但し、プロセスを「変化」させることは「混乱」に繋がる

プロセスの混乱 → 「バグ」や「手戻り」の発生 → 品質の低下

「案件」ごとにプロセスを「変化」させる方法は？

「変化」させたプロセスを「安定」させる方法は？

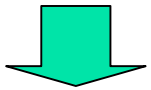
ない？ → 「万能プロセス」で「固定」させるしかない！

そもそも「要求」に合わせてプロセスを変化させる発想そのものがなかった？

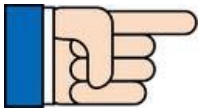
System Creates

個別問題の解決から入る

- 3つのプロセス要因
  - ①予定したプロセスは実施したが、不十分だった
  - ②納期等の圧力でプロセスを省いた
  - ③その場面で必要なプロセスがあることに気付いていなかった



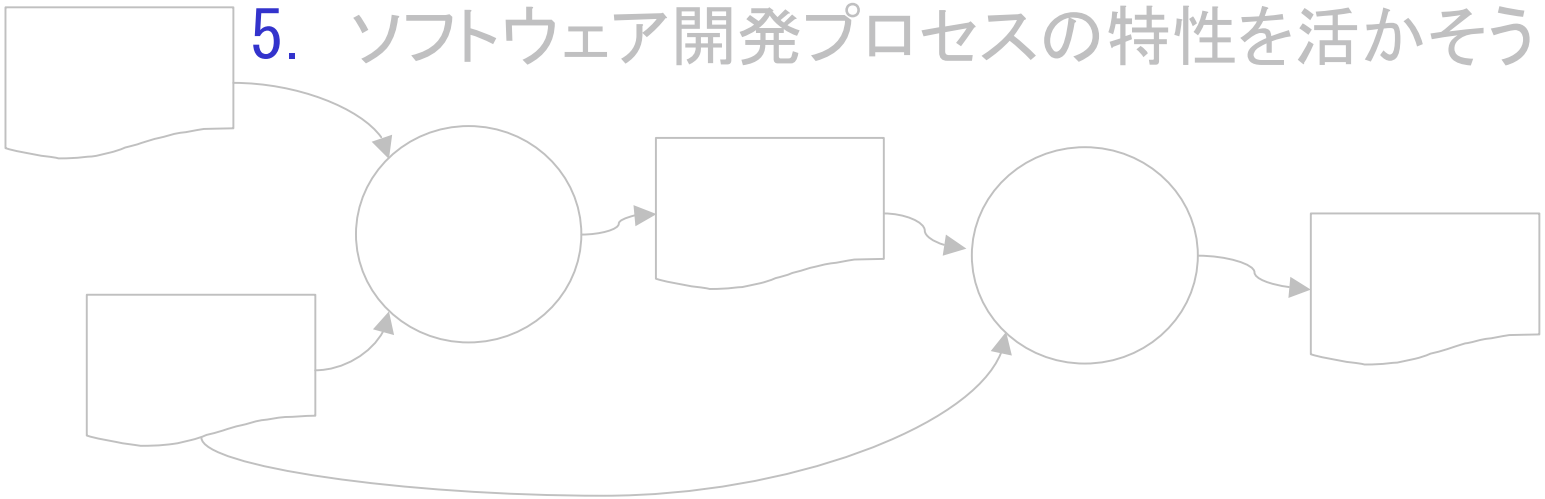
- バグや生産性などの個別の問題を解決するためにプロセス改善を“繰り返す”



個別問題に対応しながら、プロセス改善の範囲を広げる

System Creates

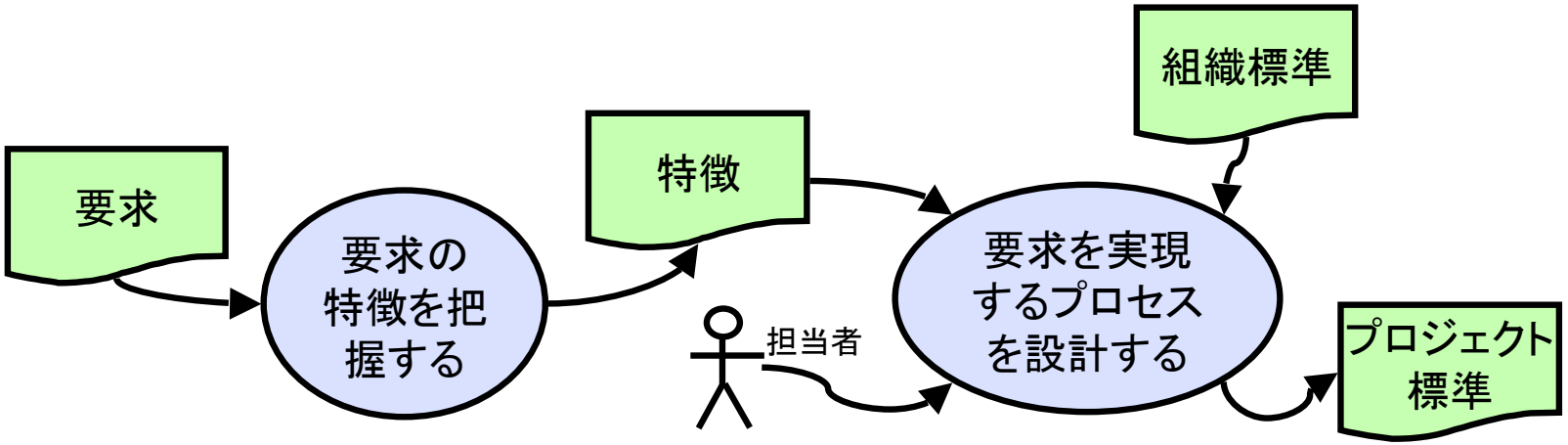
1. 私とプロセス
2. プロセス改善の取り組みの不思議
3. プロセス改善活動の難しさ
4. プロセス改善の今後の方向（提案）
5. ソフトウェア開発プロセスの特性を活かそう



プロセスを「定義する」から「設計する」へ

「定義する」は「固定する」に繋がりやすい

- 変化する顧客の要求を実現するための合理的なプロセスを、実際にその作業を担当する人が「設計する」



- 組織・・・プロセスを「設計する」スキルを失わないこと

競争に勝ち続ける組織へ

System Creates

プロセスを適切に表現する技術が必要

- 「DFD」：世界で使われているプロセスの表現ツール
  - 構造化分析のツールを開発プロセスの「Design」に応用
  - 日本では、この種のダイアグラムを使わずに「プロセス定義書」だけで対応したため、「設計する」という認識から遠くなった

- 「PFD」：DFDでは扱いにくいところを改良して提案
  - 成果物とプロセスの関係をダイアグラムで表現したもの

ダイアグラム	DFD	PFD	成果物の記号と下位層を持つプロセスの記号を新設
成果物	データディクショナリ	成果物定義書	成果物単位に記述
プロセス	ミニスペック	プロセス定義書	基本ルールは同じ

  - この他の多くのルールはDFDのルールを継承

System Creates

PFDを使ってプロセスを表現する

- 多くの人は、先週終わったプロジェクトでも、計画されたものではないためにプロセスを表現できない

- 現実は、この状態で作業に取り掛かっている
  - 作業の品質が悪くなるし、無駄が多いのも当然

- 自分の担当する範囲のプロセスの表現を繰り返す

PL	プロジェクト全体をPFDの「レベル−0」として書く ＝ 責任範囲 下位層の担当SEに実現するためのプロセスの設計を求める
担当SE	上位プロセスを意識しながら自分が担当する層のプロセスを書く 出力する成果物を、次の人のプロセスの入力条件に適合させる

System Creates

PFDを使ってシミュレーションをする

① 「変化」させたプロセスを疑似体験で「安定」させる

- その際に、2つの合理性を追求する
  - 機能的合理性・・・品質につながる
  - 経済的合理性・・・生産性につながる

② 疑似体験で身に染み込んだ習慣を薄める

- これによって、本番では「新しいプロセス」が優位になる

- 自分が設計したプロセスに基づいて見積もり、さらにそこからスケジュールを立てる

仕事の楽しさの源泉

System Creates

途中の変化に対して「代案」を考え出す

- 計画段階に考えたプロセスが「最善」であるはずはない
  - 事前の見積もりが楽観的なこともある
  - 途中で仕様変更や想定外のことも起きる

- PFDと2種類の定義書を使って、同じ結果を得る「代案」を考え出す ［Evolutionary Scheduling］
  - 必要性の低いプロセス，代替可能なプロセスを省く
  - 成果物の中の無意味な構成要素を削除
  - 成果物の構成から着手できるプロセスを分離
  - など

このレベルまでプロセスと向き合うことで、
プロセス設計やシミュレーションに投入した工数は簡単に回収できる

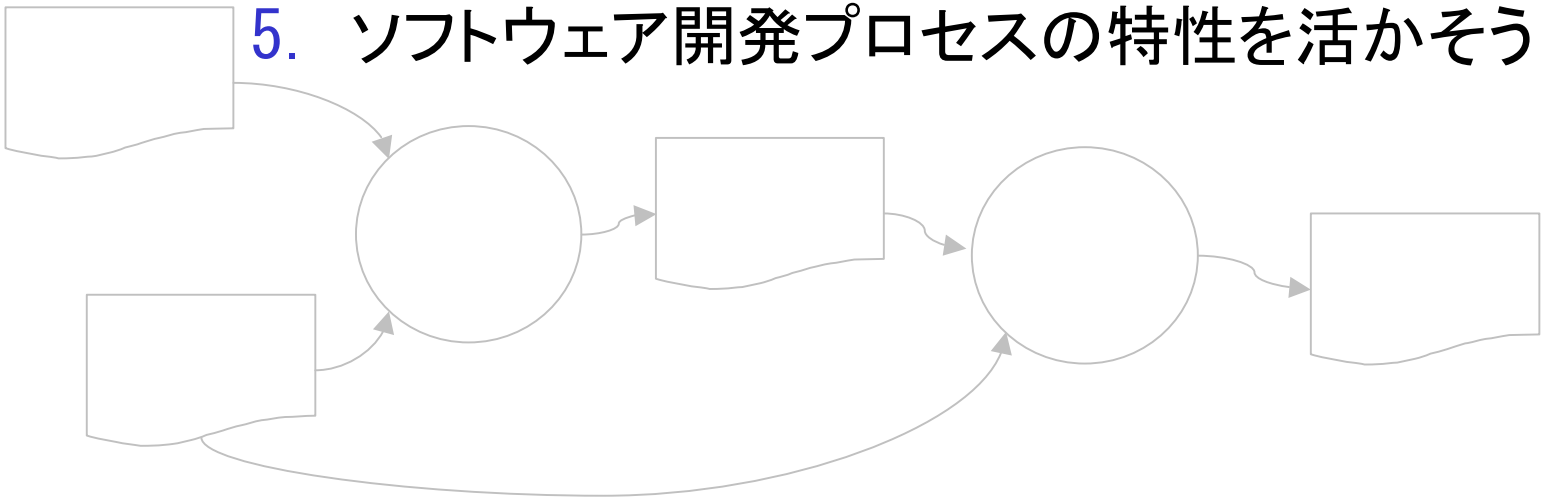
System Creates

要求に合わせたプロセスの品質保証が可能に

- プロセスを自在に設計する「3つのスキル」
  - ① 要求に合わせて合理的なプロセスを自在に設計するスキル
  - ② 変化させたプロセスを安定させるためのシミュレーションのスキル
  - ③ 途中の変化に対して「別案」を考え出すスキル
  - ここから「組織標準」に使える成功事例が手に入る
- 自在に設計されたプロセスを評価し承認できる人を育成
  - プロセスを自在に設計し、成功を繰り返してきた人であれば可能
- 「変化させたプロセスの品質保証」へ
  - 「3つのスキル」＋「プロセスを評価できる人材」で可能

変化の中で競争に勝ち続ける組織へ

1. 私とプロセス
2. プロセス改善の取り組みの不思議
3. プロセス改善活動の難しさ
4. プロセス改善の今後の方向（提案）
5. ソフトウェア開発プロセスの特性を活かそう



ソフトウェア開発プロセスの可能性

- そこで行っているプロセスに、どのような根拠がありますか？
  - 「30年前からこの方法でやってきた」というだけ？
  - 「組織標準」で決まっているから？
  - でも、そのプロセスで多くのバグや手戻り作業が生じたよね
  - また、同じプロセスで対応するのですか？
  - 「サーカスの象の鎖」って知っていますか？

- もっとうまく行く方法を考えよう
  - 問題を解決しながら、品質と生産性の両方を貪欲に追いかけよう

ソフトウェアの設計が柔軟なように、開発プロセスも柔軟だよ

ソフトウェア開発の生産性は大幅に改善できる

- 「合理的な開発アプローチ」と「事前のシミュレーション」によって、ソフトウエア開発の生産性は飛躍的に向上する
  - 開発アプローチ・・・ゴールまでの対成果物とプロセスの連鎖
  - プロセスの合理性　＝＞　バグや手戻りの未然防止

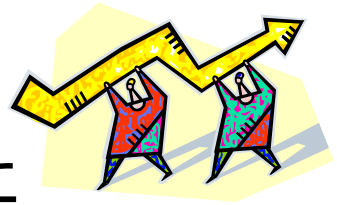
コンサルティングの結果が示している

- 今まで、ソフトウエア開発では生産性を追求してこなかったことで、改善の余地は手つかずで残っている

過去の文化と決別する絶好の機会

今こそ、「プロセスの力」を使うとき

- 生産性を3倍に引き上げるにはどうすれば良いのか
  - 設計技術だけでは実現しない
  - コード行数見積もり＝100KL、200行/H X 10人で 10日で書ける
  - あとは、実装プロセスの前と後のプロセスの生産性を高める方法



- バグの発生率を1／10に減らすにはどのようにプロセスを設計すればよいのか
  - 品質は設計プロセスで「織り込む」

ゴールから考えることで、
もっと大胆な発想でプロセスを設計することも可能になる

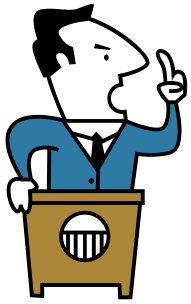
System Creates

開発プロセスを血の通ったものにする

- かつて「プロセスは否人間的」と揶揄された
  - その中で、誰もが機械のように無表情に作業をすることになると

- 自分が楽しくなるプロセスを「デザイン」する
  - 品質や生産性を上げる方法を考えるのが楽しくなる

- この時、ソフトウェア開発プロセスは従来とは全く違ったものになり、「ソフトウェア」開発に無限の可能性を感じるだろう



我々はもっとうまくやれる !!
合理的な開発プロセスを自在に設計するのを忘れていただけだ

プロセス改善による
品質と生産性の向上が明日につながる

日本の産業が
生き残る道！

- オフショア開発から国内回帰へ
  - 「品質」と「生産性」の大幅な向上による競争力の強化へ
- ソフトウェア開発は「人件費」のみ・・・「プロセス改善」による生産性の向上で国内に留めることは可能
  - 円高によって製造部門の海外転出は止められなくても・・・
- 「6K職場」と決別し、憧れの職業へ
  - ソフトウェア技術者に対する社会的地位の向上

プロセス改善の仕方によって、これらが可能になることに気付いて欲しい

ご清聴、ありがとうございました。