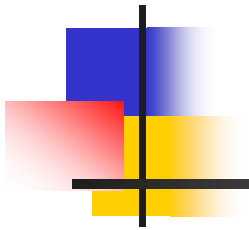


SPIと
ソフトウェア技術者のスキルアップ

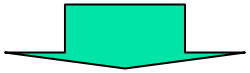
パネル「プロセス改善成功の秘訣を探る」資料

SEPG Japan 2003
ソフトウェアプロセス改善カンファレンス
2003年9月4日、5日

(株)システムクリエイツ
代表取締役　清 水　吉 男
URL=http://village.infoweb.ne.jp/~fwgf2942/index.htm

改善への動きが遅い

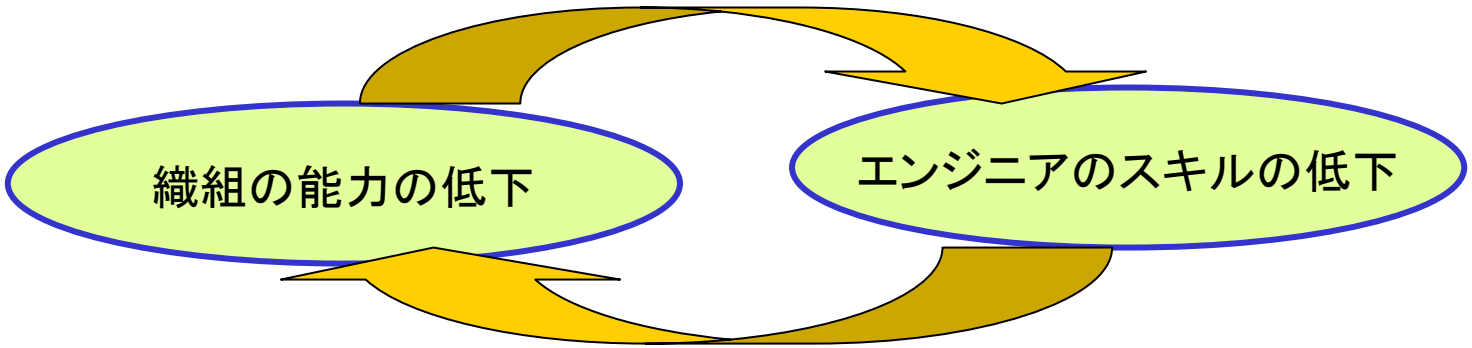
- 多くの現場では、重大な問題が発生したり、行き詰まらないかぎり動き出さない. コンサルティングの依頼が届いたときには、以下の状態に陥っている.
  - 品質の混乱
    - バグが多発して収集がつかない
    - 要件の変更が頻発してどうにも成らない
  - 納期の混乱
    - 作業の遅れが重なって納期に間に合わない
    - 顧客から要求される納期が短くなって間に合わない.



このままエンジニアのスキルアップに取り組むには障害が多い.

ソフトウェア技術者のスキルの低下

- ソフトウェアの開発が混乱している組織では、総じてソフトウェア技術者のスキルの低下が認められる.
  - 組織的、計画的教育が行われていない.
  - 個人が疲弊して勉強する意欲が喪失している.
  - 安易に「分担の分譲」が行われている.
  - 知識労働者のスタイルを持っていない.
- バブル期の後遺症
  - バブル期の大量採用を機に、社内での教育体制が崩壊した.



レベル獲得競争のしわ寄せ

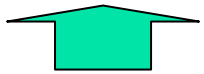
- SPI活動が、「QCD」の指標の改善ではなく、レベル獲得の方法として認識されている.

そこにある問題が解決されないまま、レベル獲得に直結する取り組みが課せられる.

現場のソフトウェア技術者にとっては、２重の負荷となる.

“今でも大変なのに、これ以上何をさせるつもりか！”

“SPI活動は、仕事を重くする！”



取り返しの付かない誤解を植え付けた！

SPI活動は仕事を楽にするもの

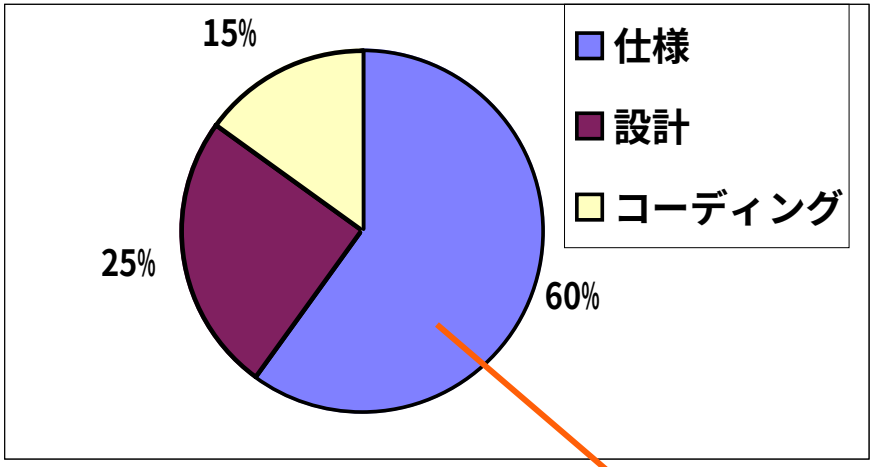
- SPI活動は、無駄な作業や手戻り作業の原因を取り除くことで、ソフトウェア技術者の負担が減る.
  - どこで時間が失われているか？
  - なぜ時間が失われるのか？
  - その時間を取り戻す方法は？
- 期間の前半で工数が増えるかもしれないが、後半で大幅に減少する方法がある.
- 獲得した時間は、ソフトウェア技術者のスキルアップに回したり、マネージメント能力の向上に回す.
- 「レベル」は、「QCD」改善の結果として獲得すればよい.

何に時間が失われたかを知ろう

- 最大の障害は、誰もが「時間がない」と思い込んでいること.
- 現状でも、比較的容易に時間を獲得できる余地があることに気付かせる.
  1. バグデータの分析、仕様変更などの手戻り作業の実態の把握などによって、どれくらいの時間が失われたかを知る.
  2. 生み出された成果物を整理して、これらが何に役に立ったかを振り返ることで、無駄になった時間を推計する.
- これらの失われた時間を取り返す方法を説明し、準備や練習に投資した時間は回収できることを理解してもらう.

仕様エラーに多くの時間が失われる

- QAテストでのバグを集計し分類すると、要求仕様に関するバグが6割程度を占める.
  - ただし、現場の認識は3割程度で、残りは「設計エラー」に分類される傾向がある.



仕様関係のバグに懸かる時間
4時間／1件
400時間／100件

＋

要件管理の中で仕様変更の際に費やされる時間

派生開発の場合、この中の7割が“変更仕様”に関するエラー

なぜ、仕様関係のエラーが多いか？

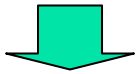
- 仕様関係のエラーとしては、
  - 必要な事が仕様書に記述されていない
    - 品質関係の要求が抜け落ちているケースが目立つ
  - 仕様書の記述から意味を読み取れなかった
    - 表現に何の工夫もない
  - 仕様間の矛盾（衝突）に気付かなかった
  - 設計以降の作業に反映する際に抜け落ちた
    - 以降の成果物との間でチェックする仕組みが無い

そこで書かれている「要求仕様書」に改善の余地が多く残されている

「仕様漏れ」は“注意”では防げない
漏れにくい仕掛けが必要

要件管理の落とし穴

- そこにある「要求仕様書」が改善されていない状態では、要件管理に取り組もうとしても空回りする.
  - CMM、要件管理　活動1・・
    - その表現で“Specify”を前提とした「合意」が出来るか？
    - 仕様漏れや仕様間の矛盾を発見できるか？
  - CMM、要件管理　活動3・・
    - 仕様変更の頻発は要件管理では防ぐことは出来ない.
    - 変更が多発する中で「CCB」が維持できるか？



事前に要求仕様書の構成や要件の抽出方法などの改善が必要

ただし、混乱した状態では、あまり難しいことはできない

作業にも無駄が多い

- 今回の要求を実現するための最適のプロセスを持たないため、作業がスムースに運んでいない.
  - 中間成果物とプロセスの連鎖が見える形になっていない.
  - その中間成果物の“ユーザー”の要求を聞いていないし、ほとんどの場合、“ユーザー”が見えていない？
  - 使い道のない「その場しのぎ」の中間成果物や、良く似たメモのような中間成果物が幾つも作られる.
  - こうした成果物の制作に使われた時間が回収されない.

同じ要求は2度とないにも関わらず、
今回の要求を実現するための合理的なプロセスが設計されていない.

要求仕様書の表現の問題と会わせて、全体の2～5割の時間を無駄にしている.

まず時間を取り戻す

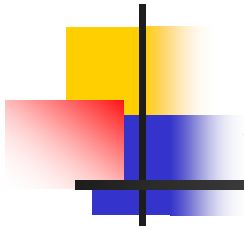
- 限られた状況の中で、効果的に時間を取り戻すための最初の3つの取り組み.
  1. 要求仕様書の書き方を改良する
     - 「要件開発」の初歩としての位置付け
  2. 合理的な開発プロセスを設計する
     - プロセスと成果物の連鎖に合理性を確保
       - プロセスフローダイアグラム
       - プロセス定義
       - 成果物定義
  3. 人の良いところは素直に認める活動

要求仕様書の改善	＋100時間
レビューに投入した時間	＋　60時間／4回
レビューで発見された仕様エラー	ー480時間／120件
（失われずに済んだ時間）	ー320時間

プロセスの改善とは

- プロセスは、長い時間の中で個人の習慣となって居着いている.
  - 改善に強い抵抗が発生する理由でもある.
- プロセスの改善とは、
  1. そこに居る人たちの固定した習慣を変えること
  2. それを支援するように組織の文化を変えること

要求が適切に仕様化され、今回の要求に対して合理的なプロセスを自在に設計できるスキルを手に入れれば、継続的なプロセスの改善は容易になる



スキルアップに回す

- ソフトウェア技術者のスキルアップや、CMMなどの取り組みは、現状の混乱状態が収まり、少し時間が獲得できたあとで可能となる.
  - 「要求の仕様化」や「プロセスの設計」など、混乱を収束させた方法は、この後のSPI活動の基盤となる.
- 当面は、実際のプロジェクトの中で必要になるテーマを中心に習得していくとよい.
  - 学習や習得に投入した時間を、効果として直ぐに回収する必要がある.
    - エンジニアリング技術やアーキテクチャ技術
    - マネージメント技術

基礎知識を早急に入手

◆典型的なコンピュータプロフェッショナルは、25歳で職について70歳まで働き、45年間就労することになる。

◆前述の数字によれば、退職時には基礎知識の62%、抽象化の知識の20%、実用知識の3%が有効であるにすぎない。勤続の中間点では、それらの数字はそれぞれ81%、49%、16%である。

情報工学教育に関する国際シンポジウムにおける講演
ACMのコンピュータカリキュラムに関するレポート(1988)より
Paul W. Abrahanms

- 「基礎知識」は、今のコンピュータのアーキテクチャが変わらない限り、有効と思われる.
- 「基礎知識」が不足している状態では、要求を実現することが困難になるし、分野によっては、ソフトウェア技術者として致命傷となる.

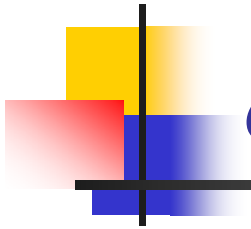
新しい技術を吸収すること

- 知識労働者であることによって、継続的な取り組みが可能になる.
  - 日常の作業を効率良く進めることから始める.
  - 獲得した(失わずに済んだ)時間は「再投資」に回す.
- 適切な雑誌などによって接点を確保する.
  - 雑誌の内容に対する理解の状況から、自分の「居場所」も分かる.
- インターネットを活用する.
  - 有用な情報を随時収集して研究する.

常に、新しい知識や技術の習得を継続することが求められる ＝ エンジニアとしての誇り

組織の仕組みも変革を

- 日本のソフトウェア開発組織は30年何も変わっていない.
  1. ソフトウェア技術者が、もっと長くソフトウェア開発に携われる仕組みが欲しい.
     - 10〜15年ぐらいでは「一流」になるのは難しい.
     - 極めるためには30年という期間が欲しい.
  2. うまく出来る人よりも、うまく出来る仕組みを残せる人を優遇して欲しい.
     - 暗黙知から形式知へとシフトさせる.
       - ノウハウやOJTへの依存から脱却し、「テキスト」を残す.
  3. 褒める文化へ
     - “けなす文化”からは「CMM」は実現しない.
     - ソフトウェア技術者自身による、能動的な動きを引き出す.

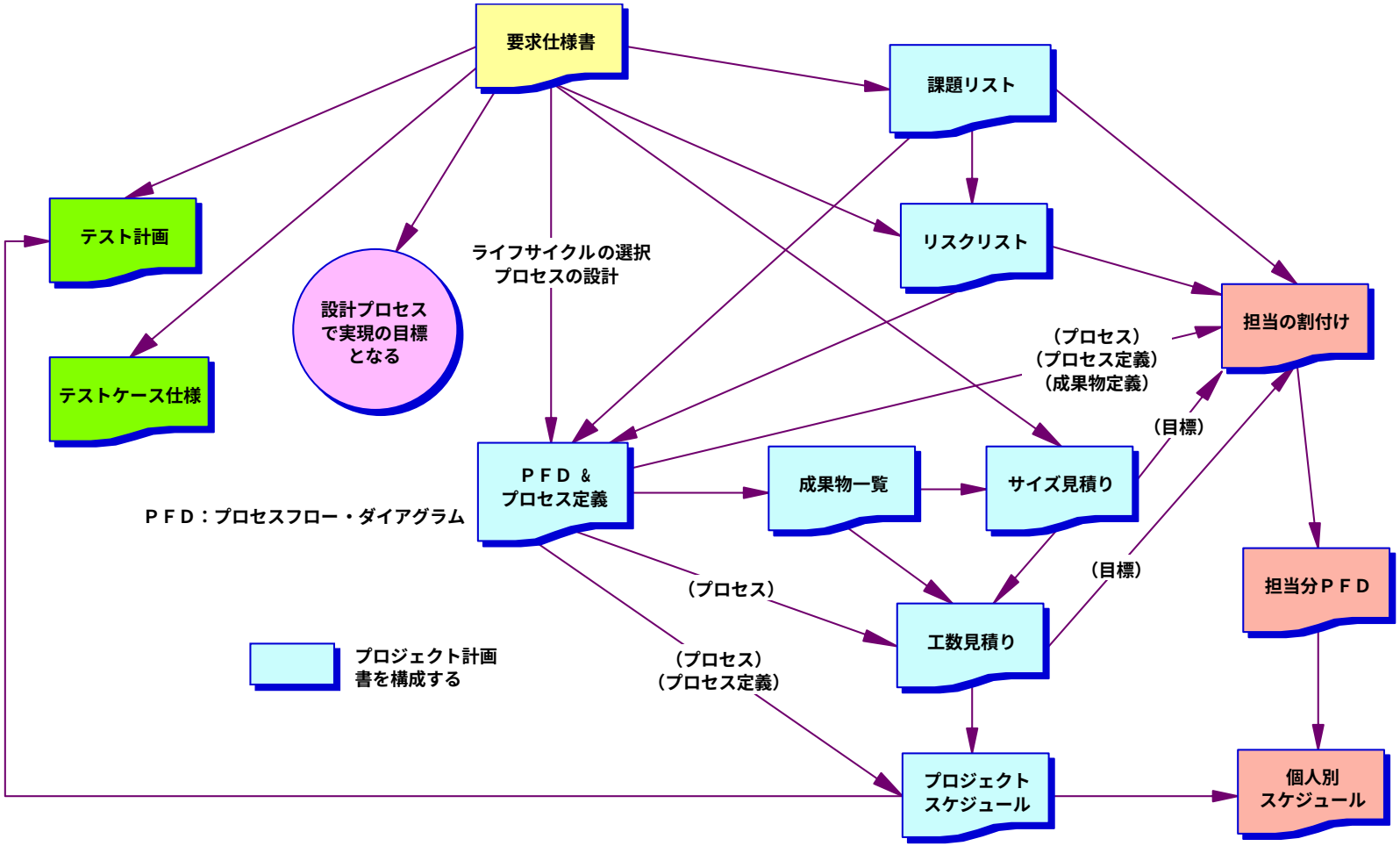


めも

参考:

要求仕様書の重要性

- 下図のように「要求仕様書」と「PFD」が、すべての「元」になる.

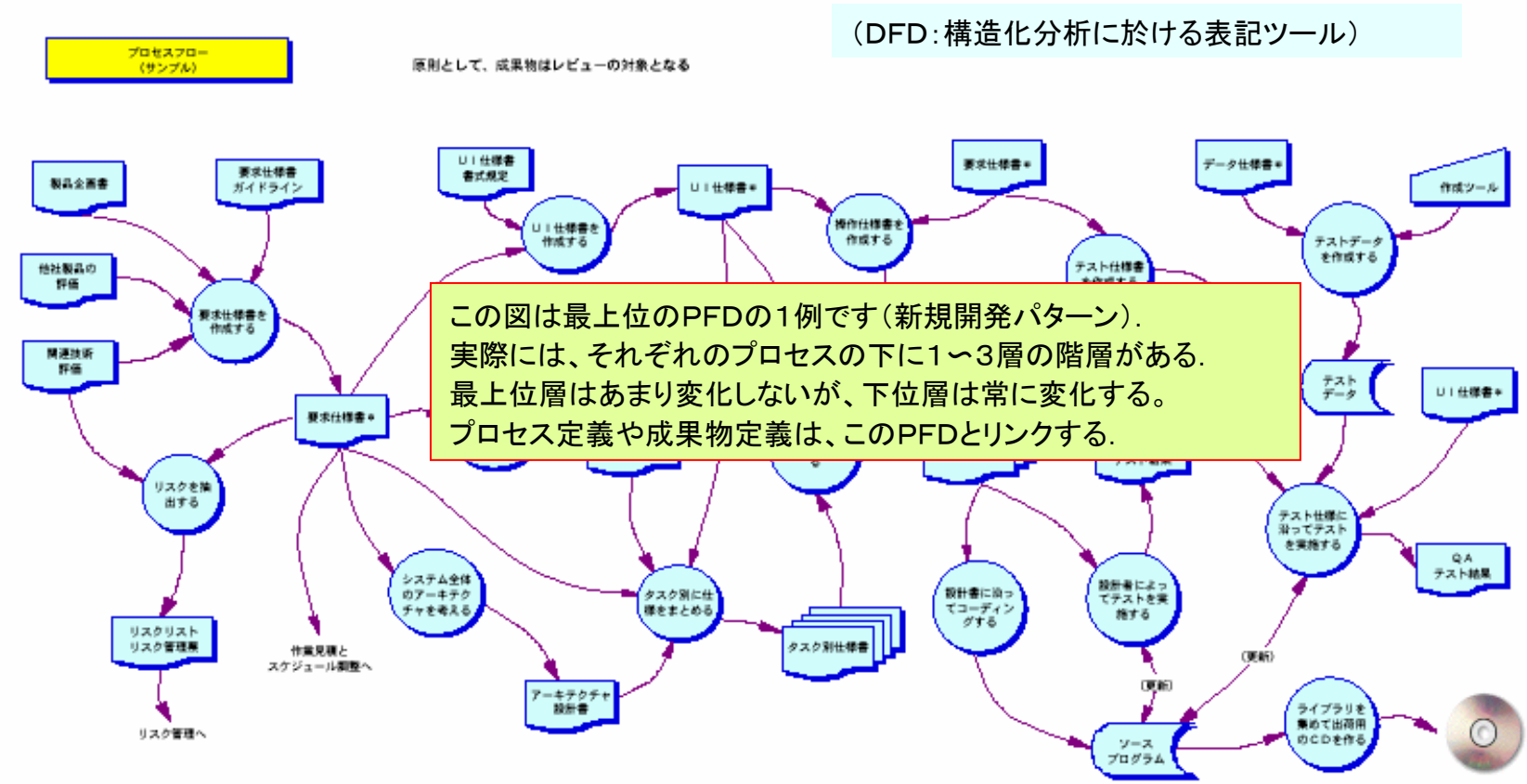


参考:

プロセスフロー・ダイアグラム

- PFD:プロセスフロー・ダイアグラム
  - 構造化分析のDFDの表記を応用して使いやすくしたもの.

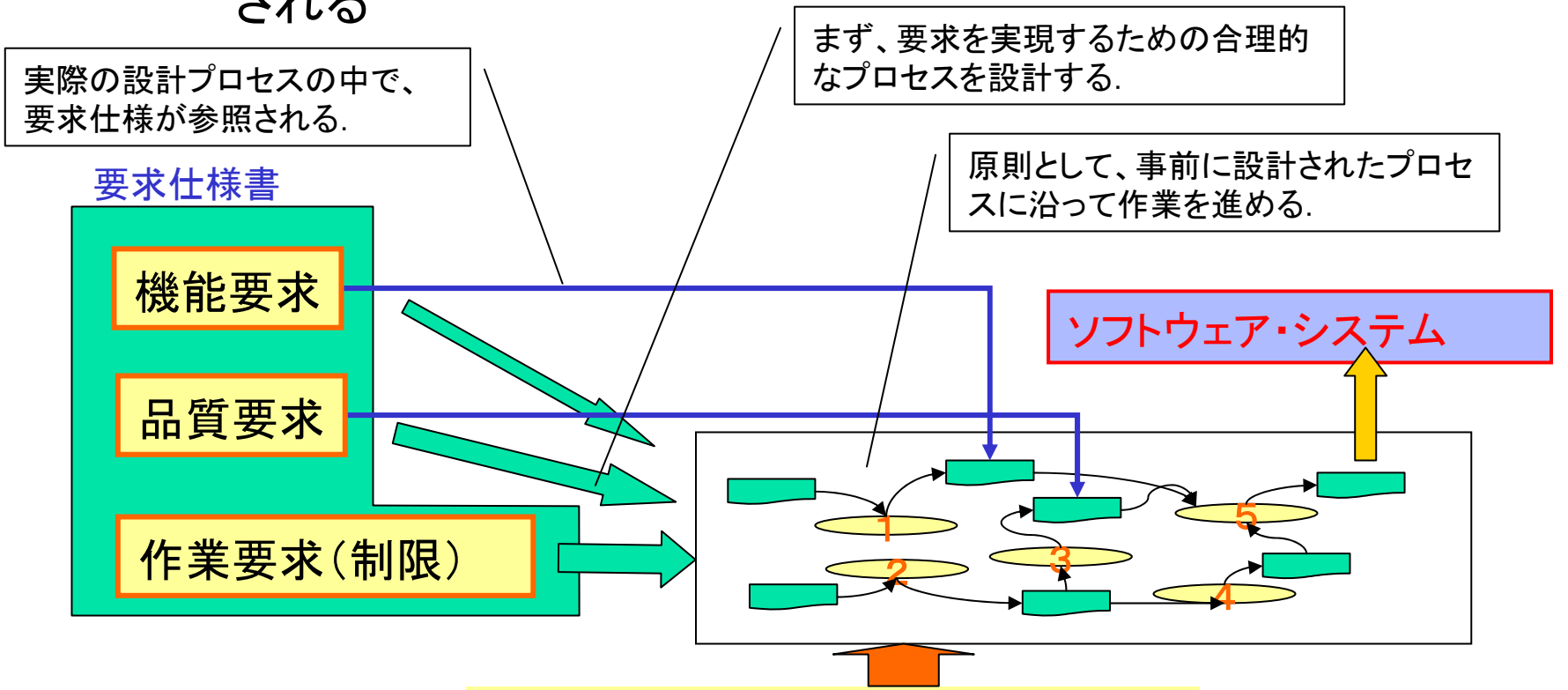
（DFD:構造化分析に於ける表記ツール）



参考:

要求とプロセスの関係

- 開発プロセスは、今回の要求を実現するために合理的に設計される



参考:

技術習得のレベル

段階1:無知の段階 — その技術について聞いたこともない
段階2:気がかりな段階 — その技術について文献を読んだことがある
段階3:見習いの段階 — その技術について3日間のセミナーに通った
段階4:実践しようとする段階 — その技術を実際のプロジェクトに適用しようとしている
段階5:職人の段階 — その技術を仕事の上で自然に自動的に使っている
段階6:名人の段階 — その技術を完全に消化していて、いつルールを破るべきかを知っている
段階7:エキスパート — 専門書を著作し、講演し、その技術を拡張する方法を探究する

ページ-ジョーンズ
"The Seven Stages in Software Engineering"
American Programmer(July-Aug.),1990

- 組織は、段階5以上の人を増やす必要がある.
- そのためには、もっと長くソフトウェア開発に携わる機会をつくること.