

ソフトウェア開発における
プロジェクト管理とコミュニケーション

2006年 5月 23日

特定非営利活動法人 ITコンピタンス研究所
高橋哲夫



何もしない失敗

プロジェクトに、多数の要員を割り当て、管理せず
にいと、“混沌”となる。

大勢の要員がそれぞれ勝手に振舞うとプロジェクトのエントロピーは増大。
《管理の必要性》



テーマ

ソフトウェア・プロジェクト管理の生い立ち

ソフトウェア・プロジェクト管理の概要

失敗プロジェクトとその原因

失敗を繰り返さないために

生い立ち

1) 何時頃から確立されて来たか。

DIPS(電電公社の国産コンピュータ)の開発

D,F,H,N: 4社によるHW,OS共同開発。

この開発のプロジェクト管理は電子交換機
の開発経験の発展形。 (1968年)

IBM360発表 (1964年)

NASA アポロ11号月面着陸 (1969年)

2) ソフトウェア工学

ソフトウェアの開発規模拡大に伴なって、生産
性・品質向上の要求が強まる

(1970年代)

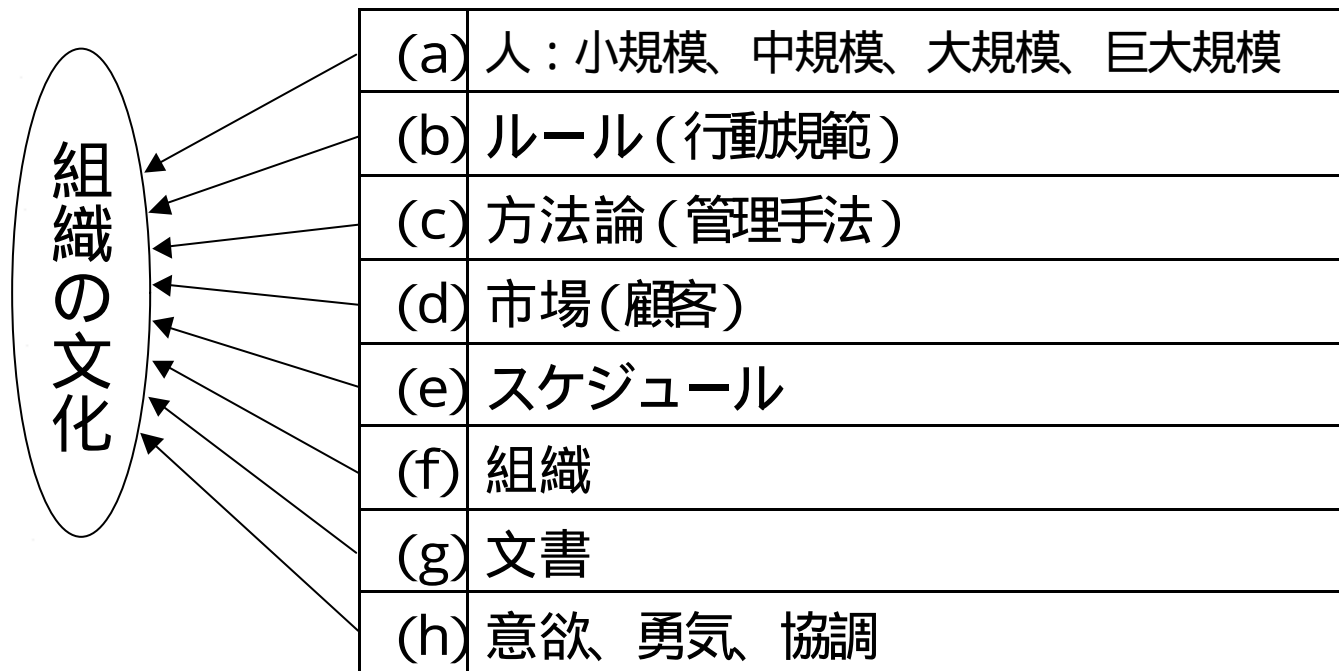
生い立ち(ソフトウェア工学の論点)

論点	Communication 手段
1) 技術 アルゴリズム、構造化設計、 オブジェクト指向、設計方法論、 ソフトウェア構築方式 ……	文書、帳票、 プログラム、…
2) 管理 QC(*)、ソフトウェア・メトリクス、 ソフトウェアプロセス成熟 (CMMI)、 PMBOK アジャイル、…	文書、帳票、 会議、報告、…

* 例: SWQC:NECの意識改革運動、小集団活動(TQCのSW版)

ソフトウェア・プロジェクト管理の概要

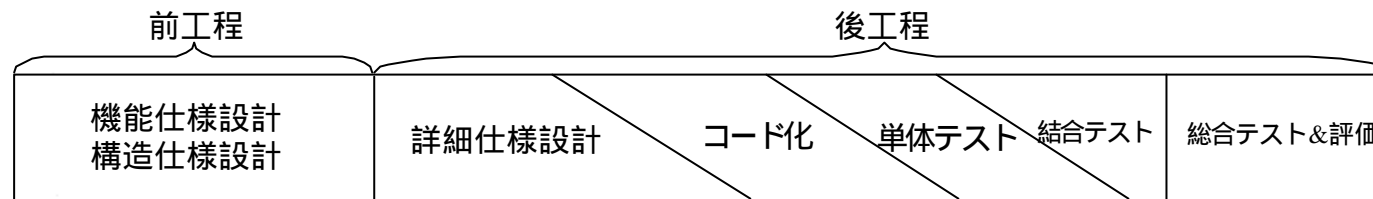
- [1] プロジェクト管理を俯瞰すると;
- 1) プロジェクト管理は人の管理。
 - 2) プロジェクト管理の本質は不易。
 - 3) プロジェクト管理の構成要素。



[2] ソフトウェア開発の代表的開発モデル(Waterfall*)

(*大きなソフトウェアを多くの人達が協同で作成するための開発手法)

1) 工程



2) 生産物

- 機能仕様(要求仕様、外部仕様)
- 構造仕様
- 詳細仕様
- コード(プログラム)
- テスト仕様
単体テスト、結合テスト、総合テスト

Communication
手段

3) 要員と担当範囲

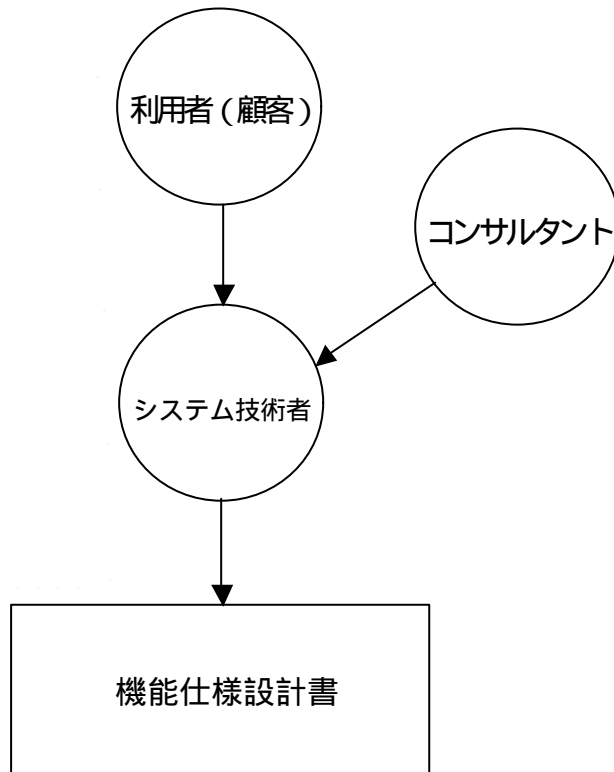


図1

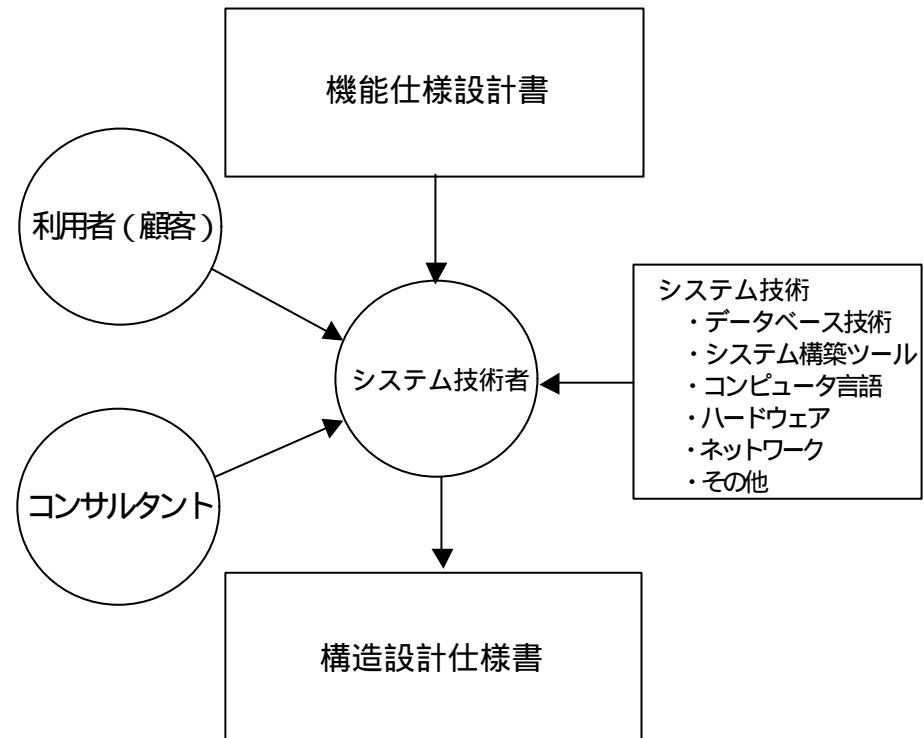


図2

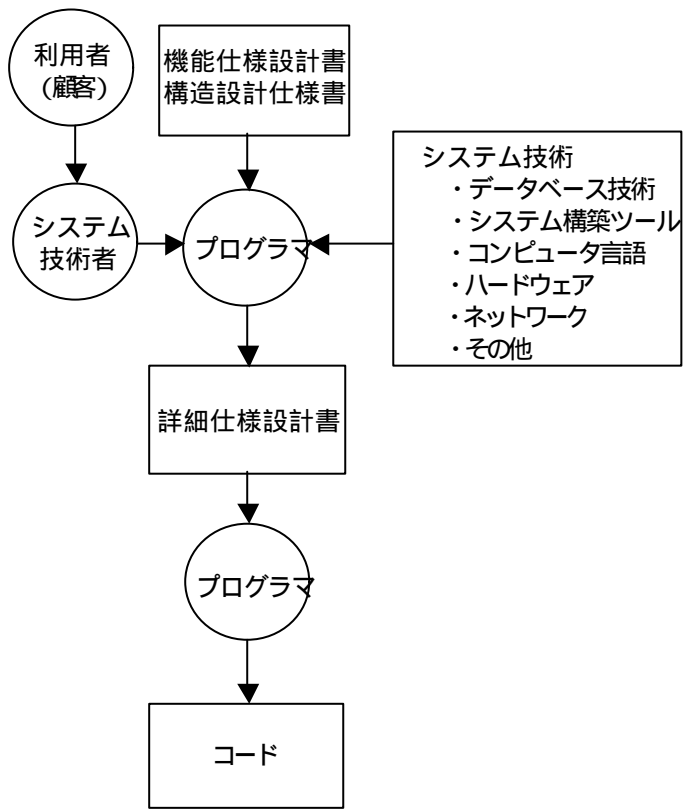


図3

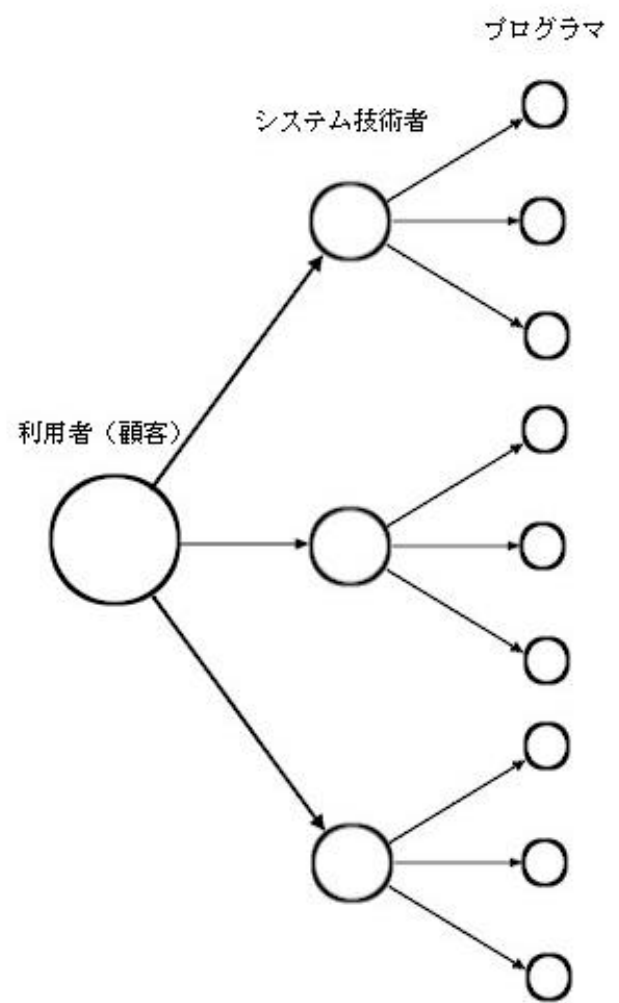


図4

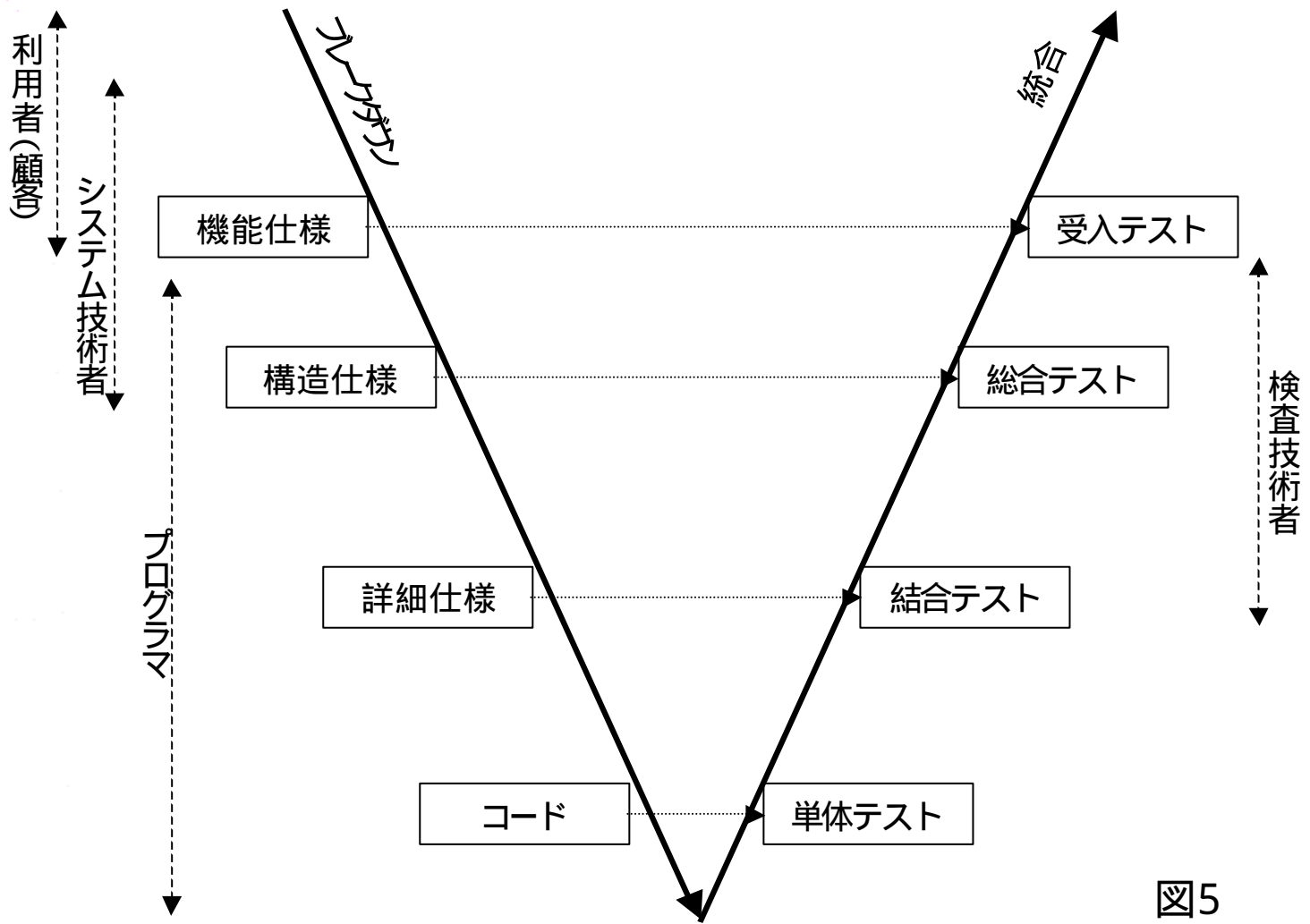


図5

4) コミュニケーション手段

(a) 文書(含:帳票等)

- ・ 機能仕様 (要求仕様、外部仕様)
- ・ 構造仕様
- ・ 詳細仕様
- ・ コード(プログラム)
- ・ テスト仕様
単体テスト、結合テスト、総合テスト

(b) 会議、報告、対話

コミュニケーション手段を有効とするには
「Communicationベース」の確立が必要

技術知識の共有

管理知識の共有

Communicationベース

意思疎通 成立



Communication :

英和辞典： 伝達 通信

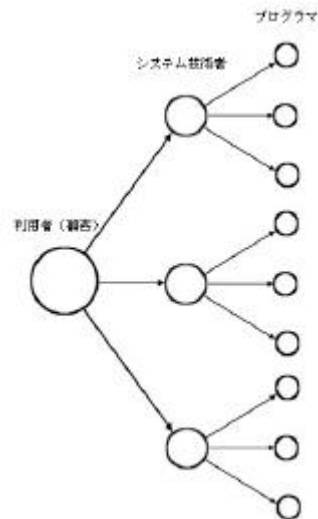
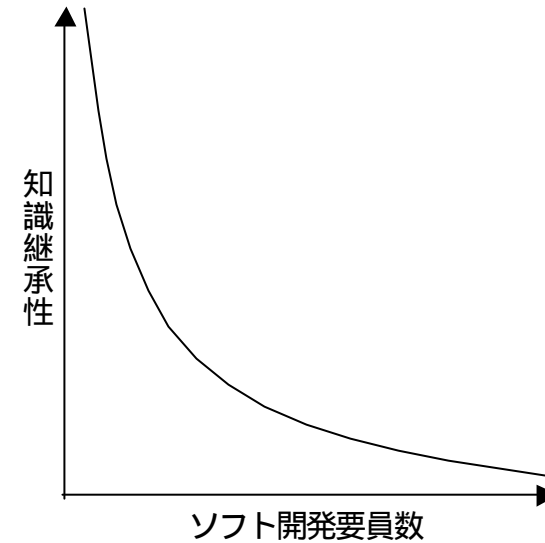
英英辞典： the act of sharing or exchanging information, ideas or feelings
the methods that are used for traveling to and from a place or
for sending message between places

日本語では **意思疎通** というべきか。

4) コミュニケーション・コスト

$$\text{知識継承性} = T \frac{1}{\text{ソフト開発要員数}}$$

T: 定数



プロジェクト規模(要員数)			
小規模	中規模	大規模	巨大規模
< 10	< 50	< 200	200 <

コミュニケーション・コストの主な発生源

- (1) プロジェクト要員
《知識継承コスト》
- (2) Communicationベースの確立
《学習コスト》
- (3) 利用者（顧客）要求の獲得
《要求獲得コスト》
- (4) 機能仕様と構造仕様をプログラマに理解させる
《プロジェクト規模が急激に大きくなるため、理解させるコスト》
- (5) プロジェクト状況把握
《計測コスト》
- (6) 情報収集と調整・判断
《会議・報告コスト、Prj推進チーム・出荷判定会議コスト》
- (7) 暗黙知の伝承
《文書だけで意思疎通は成立しない。知識伝承コスト》

コミュニケーション・コストを下げるには

(1) プロジェクト要員	<ul style="list-style-type: none">・少数精鋭・人の入れ替えを避ける
(2) Communicationベースの確立	<ul style="list-style-type: none">・既存のCommunicationベースの活用
(3) 利用者（顧客）要求の獲得	<ul style="list-style-type: none">・同種のソフトウェア開発経験者を活用
(4) 機能仕様と構造仕様の理解	<ul style="list-style-type: none">・当初の技術者を最後まで残す・合宿
(5) プロジェクト状況把握	<ul style="list-style-type: none">・計測技術を活用
(6) 情報収集と調整・判断	<ul style="list-style-type: none">・組織化と統制
(7) 暗黙知の伝承	<ul style="list-style-type: none">・熟練技術者の活用

[3] Waterfall以外の開発モデル

Waterfallでは、作業の出来不出来が後で分る(図5参照)。

やり直しがあれば、コスト**大**

Spiralモデル(Proto Typeの併用)、
Incrementalモデル(中核から繰り返し開発)、
Extreme Programming(XP:複数のストーリーを順に実装)
などが提案される。

開発モデルに決定打はない！ しかし、役立つ。

- Waterfallモデルは人間の思考活動に沿ったもの。
- Spiral、Incremental、Extreme Programmingモデルは補完的。



失敗プロジェクトとその原因

1. 保険システム
2. 集金システム
3. エンジニアリング系DTP
4. 販売管理システム
5. 出版システム

1) 保険システム

DIPSマシンからACOSマシンへの移行

DIPS: SYSLのマクロ言語

ACOS - 4 特定言語 + コンパイラの開発。

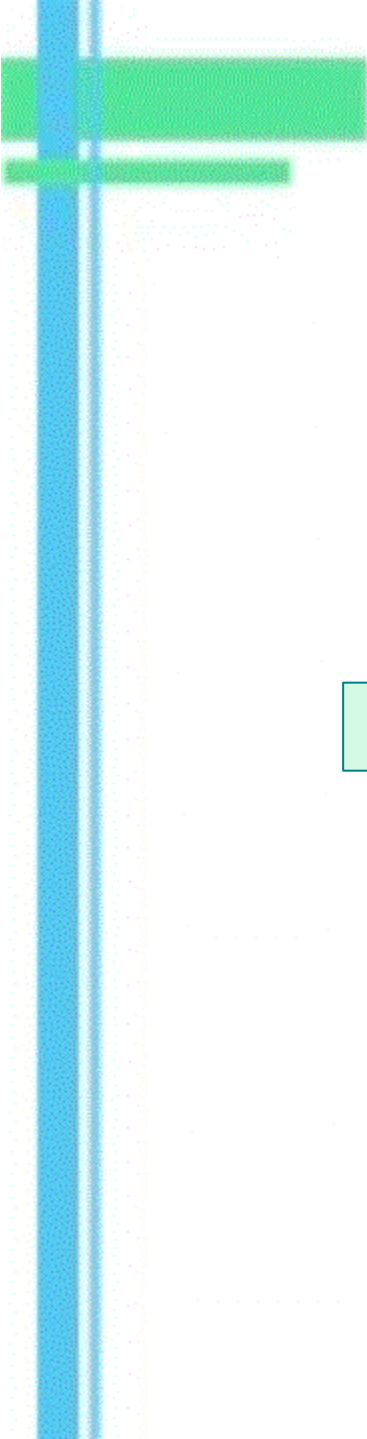
- ・たった一つのシステム開発の効率化のために、言語とコンパイラを開発するという大冒険。
- ・不完全なプロジェクト状況把握。
- ・予算を持たないマネジメント。
- ・おかしいと思ったとき、上部に進言する勇気。

2) 集金システム

全国16拠点に導入。総勢150人。

協力会社5 - 6社。それぞれに管理者。

- ・まとめ役の元請: JD, K, S×2、担当5 - 6人。
統率力不足。

- 
- ・プロジェクト状況計測不能
 - ・顧客との交渉。

内部体制固め。協力会社の意識改革。朝礼、夕礼。

3) エンジニアリング系DTP

エンジニアリング系電子出版ソフトウェア

- ・リーダーがプロジェクト状況を把握していない。
- ・顧客要求がなかなか確定しない。
- ・現場管理者が予算を持たない。
- ・裏づけのない顧客との約束。

**内部体制固め。現状把握と顧客への説明、理解を得る努力。
顧客からの罵声に耐える忍耐力。**

4) 販売管理システム

Webベース販売管理システム。全国24拠点。

- 要求仕様、構造仕様が不完全で、詳細設計に入れない。
- 予算の2 / 3を上流工程で使いきり。
- プロジェクト状況がつかめない。
- フレームワーク(ソフト構築ツール)に不慣れ。

5) 出版システム

3つの業務システムをWebベースで再構築

- 顧客はすべてをベンダ任せ。
- 顧客のシステム担当にも要求仕様が分らない。
- 構造仕様書が不完全。
- 役割分担が明確な体制がない。



失敗原因


- ・ 顧客不参加のプロジェクト
- ・ 顧客ニーズの把握が不十分。
- ・ 不完全な技術評価(メリット/デメリット)
- ・ 仕様が不明瞭なまま提案・見積を実施。
- ・ 営業・開発の役割分担が不明瞭(費用見積、顧客ニーズ把握)。
- ・ プロジェクト実施計画がなく、プロジェクト遂行計画が不明瞭。
- ・ コミュニケーション不足(営業、開発、メンバー間)。
- ・ プロジェクト実行管理の不備(管理能力・経験不足)。
- ・ リソース配分のミス。
- ・ クリティカルパスに配慮しない開発手順。
- ・ 管理されないプロジェクト。まかせっきり。
- ・ 不完全なプロジェクト状況把握。
- ・ プロジェクトの状況に応じた対処がタイミングよくできていない。
- ・ 予算を持たないプロジェクト管理者。
- ・ 問題を上層部に進言する勇気がない。

失敗の原因は他にも多数あり、かつ複合的。

失敗を繰り返さないために

1970年代以降、ソフトウェアのプロジェクト管理手法は進歩。
しかし、

- ・ 失敗プロジェクトは減っていない。
 - ・ 30年前と同じ失敗を繰り返している。
- } → **なぜ？**

- 
- (1) 技術者としての寿命が短い。 《誰でも管理者指向》
 - (2) 技術の進歩が早い。
e.g. 最近では、パッケージソフトを多用するシステム開発が増加。
しかし、対応した開発方法論が未整備。
 - (3) 技術やPrj管理に精通した熟練技術者の知識継承（暗黙知の伝承）
ができていない。



技術 / 管理能力を強化・維持できる組織作りが鍵

