オブジェクト指向プロジェクトの管理

- ソフトウェア・プロジェクト管理に関する定石
- ピープルウェアに関する定石
- プロジェクト管理に関する定石

ソフトウェア・プロジェクト管理に関する定石

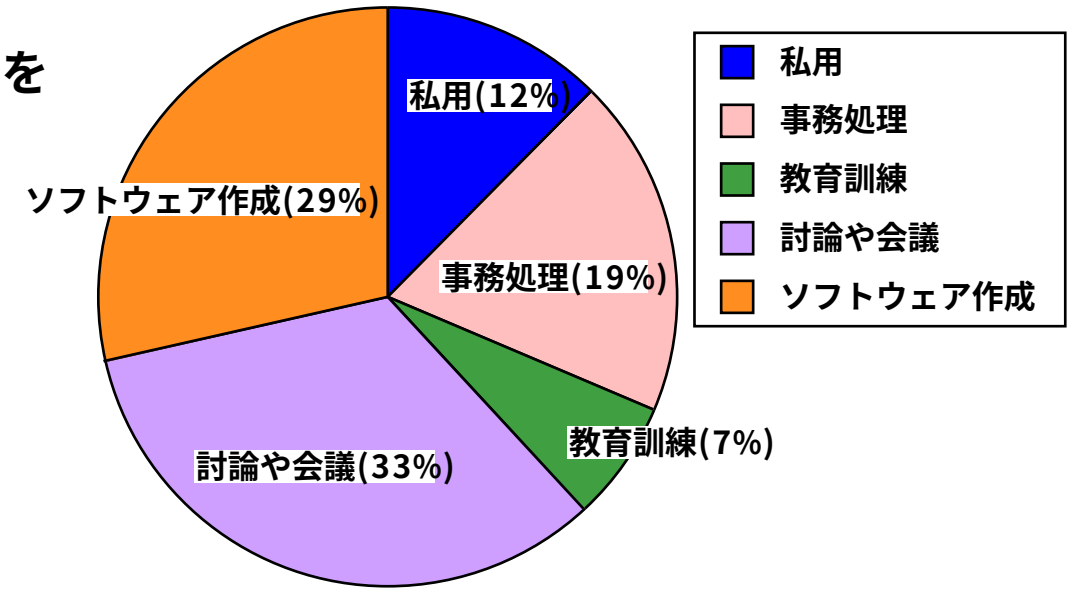
- オブジェクト指向を使えば
  プロジェクトはうまくいくのか？
- オブジェクト指向技術を
  なかなか覚えられないのだが
- コンポーネントウェアとか良いツールがあるのだから、分析や設計は大概にしてプログラムを作っては駄目か？
- オブジェクト指向を使って
  プロジェクトの遅れを
  取り戻せるか？
- プロダクト

オブジェクト指向を使えば プロジェクトはうまくいくのか？

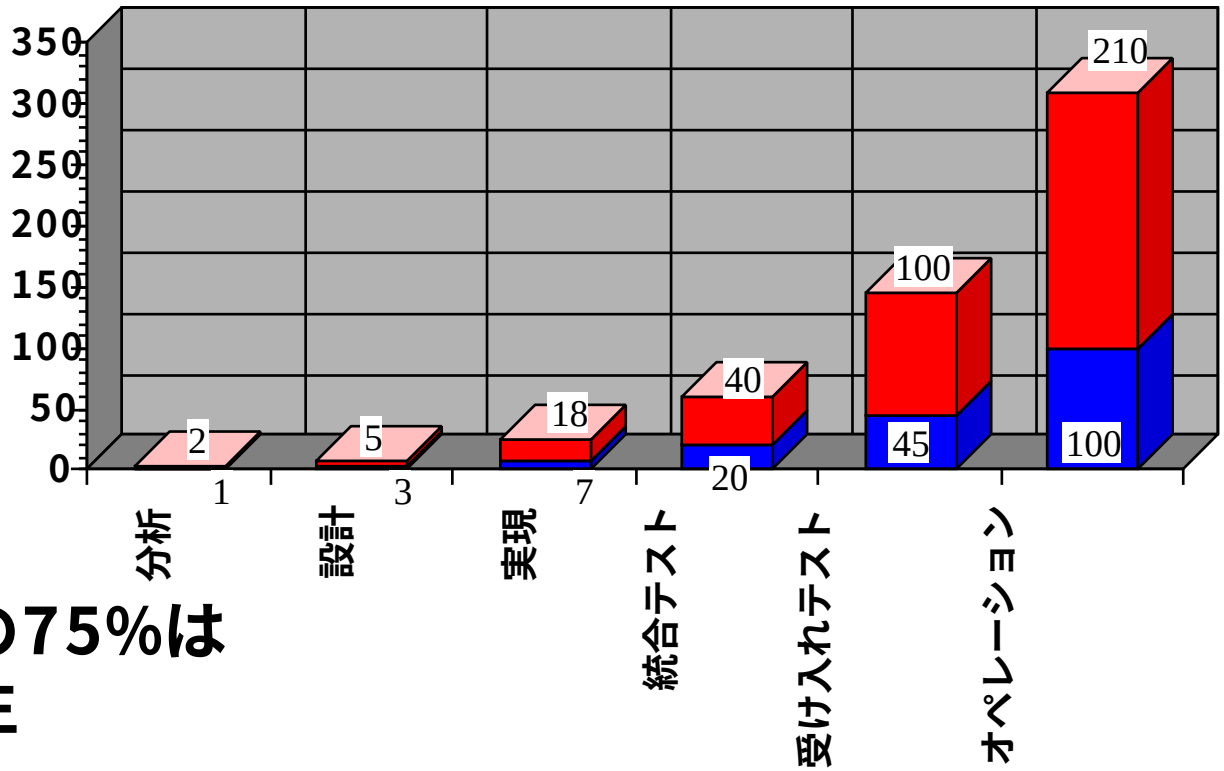
- オブジェクト指向で解決できるのは技術的問題点だけ
- 失敗の原因は、単なる技術的問題ではない
  - 失敗したプロジェクトの調査(DeMarco/Lister)
    - 「政治的要因」による失敗と言われていた
      - 実際には...
        - 意思疎通の問題
        - 要員の問題
        - 管理者や顧客への幻滅感
        - 意欲の欠如
        - 高い退職率

オブジェクト指向技術を
なかなか覚えられないのだが

- 英語の勉強にどれくらいの時間を掛けているか？
  - 中学３年で２８０時間＝１２日間
- OOの勉強にどれくらいの時間を掛けているか？



コンポーネントウェアとか良いツールがあるのだから、分析や設計は大概にしてプログラムを作っては駄目か？



- 重大なエラーの75%は上流工程で発生

オブジェクト指向を使って プロジェクトの遅れを 取り戻せるか？

- 工期の圧縮
  - 通常のプロジェクトの工期は人や金やその他を投入することによって25%圧縮できる、がそれ以上は不可能である
    - ただし、「ブルックスの法則」に注意
      - 遅れているプロジェクトに人を投入すると、ますます遅れる
  - オブジェクト指向言語のコーディングに大量に人を投入すれば何とかなるのでは？
    - コーディングは開発コストのわずか１５%を占めるだけである
- それではどうすればよいのか？
  - 出来る技術者を少数投入する
    - ４倍から１０００倍の差
  - オフィス環境を良くする
    - ピープルウェアの節参照
  - 開発環境を良くする
    - SmalltalkやCLOS系の環境が圧倒的に優れている

プロダクト

- うまくいったから製品化したい
  - 製品と単体プログラム
    - アプリケーション製品はプログラムの３倍のコストであり、システムソフトウェア製品はそのまた３倍のコストがかかる
- 出来たプログラムが遅くて使い物にならない
  - Paretoの法則
    - モジュールの２０％が、コストの８０％を消費する
    - モジュールの２０％が、エラーの８０％を含む
    - モジュールの２０％が、修正予算の８０％を消費する
    - モジュールの２０％が、実行時間の８０％を使う
      - Knuthの法則
        - コードの２～３％が、実行時間のほとんどを使う
        - 焦点を絞って効率化すれば１０倍くらいはすぐ速くなる
    - ツールの２０％が、時間の８０％を使う

ピープルウェアに関する定石

- どのようなチーム構成とすればよいか？
- プロジェクトがうまくいかないのだが？

どのようなチーム構成とすればよいか？

- チーフプログラマーチーム
  - 構成
    - チーフプログラマー・バックアッププログラマー・ライブラリアン・秘書・スペシャリスト
  - 特徴
    - 技術的決定をできる人間が、自分で作る
    - 専門家を置けるため、最新の技術が取り込みやすい
    - コミュニケーションのロスが少なくて済む
    - 責任は明白になる
- 旧日本軍型チーム
  - 構成
    - 専務・部長・課長・主任・その他大勢
  - 特徴
    - 技術が分からない人が技術的決定をする
      - 誤った現状の把握（クックス．ノモンハン）
    - 専門家がいないため、最新の技術が取り込めない
      - 二流の火器と想像力に欠ける教理
    - コミュニケーションのロスが大きい
      - 支離滅裂な指揮関係と中央における計画性の欠如
    - 誰も責任を取らなくてよい
      - 辻政信をはじめ誰も責任を取らず、日中戦争・太平洋戦争へ

プロジェクトが
うまくいかないのだが？

- 開発プロジェクトの人間的側面
- 失敗する管理方法
- ピープルウェア的管理(ドゥマルコ)
- チームの触媒
- 品質第一(ドゥマルコ)

開発プロジェクトの人間的側面

- ドゥマルコ
  - 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
  - 人は交換できる部品ではない
- ワインバーグ
  - 成功のほとんど全てが、少数の傑出した技術労働者の働きに依存している
    - 彼らは自分の回りのものすべては最高であることを望む人々であった
      - OO開発に向いた人々

失敗する管理方法

- 旧日本軍式管理哲学
  - 失敗をするな
  - 兵隊を休ませるな
  - へまをやったやつは厳罰だ
  - 兵隊はいくらでも補充できる
  - 決められたやり方を手早くやれ
  - 作業手順を標準化せよ
  - 新しいことはするな。そんなことは参謀の仕事だ
- 米国的管理
  - もっと長い時間働くようにプレッシャーをかける
  - 製品の開発過程を機械化する
  - 製品の品質について妥協する
  - 手順を標準化する
- 早くやれとせかせば、雑な仕事をするだけで、質の高い仕事はしない
  - OOには、質の高い仕事が不可欠

ピープルウェア的管理(ドゥマルコ)

- エラー大歓迎
  - 間違いを許さないと、人は消極的になるだけ
    - 失敗しそうなことには手を出さなくなる
      - OOは間違いの修正の連続
  - 開発標準や作業規定の無理強いは悪である
    - OOは直線的な開発ではないため、開発標準や作業規定で全部をカバーできない
      - 便利なツールを使った結果が標準になるように工夫する
- 管理とは尻を叩くこと？
  - ハンバーガーを作るにはうまくいくかもしれないが、体でなく頭を使う仕事ではうまくいかない
    - 製造作業では、作業者を部品の一つとして考えると便利なこともあるが...
  - 担当者の自発的やる気が認められないと、やる気は失せる
- 個人のユニークさを認めるか？
  - ネクタイを締めてこない
  - 自宅に高性能WS
  - 経費の使い方
    - 書籍代が異常に高い
      - もっとも安いクラスライブラリーの宝庫
    - ソフトウェア代が異常に高い
      - OOは物まねの連続

チームの触媒

- プロジェクトに人を入れる場合、静的能力を重視し過ぎる
  - コーディング能力や設計能力やドキュメントがどのくらい書けるかなど
- 動的能力を重視する
  - チーム全体にうまくなじむか
  - チームの触媒的役割
    - ネアカ型
      - その人がいると、担当者間の意思疎通がよくなり、プロジェクトが楽しくなり、チームの結束は固くなる
    - ネクラ型
      - あるいは、普段は何をしているか分からないのだが、皆が気がつかない問題に気づく人
        - OOは複雑なソフトウェアを比較的簡単に作れるが、反面複雑な欠陥も簡単に作り込める

品質第一(ドゥマルコ)

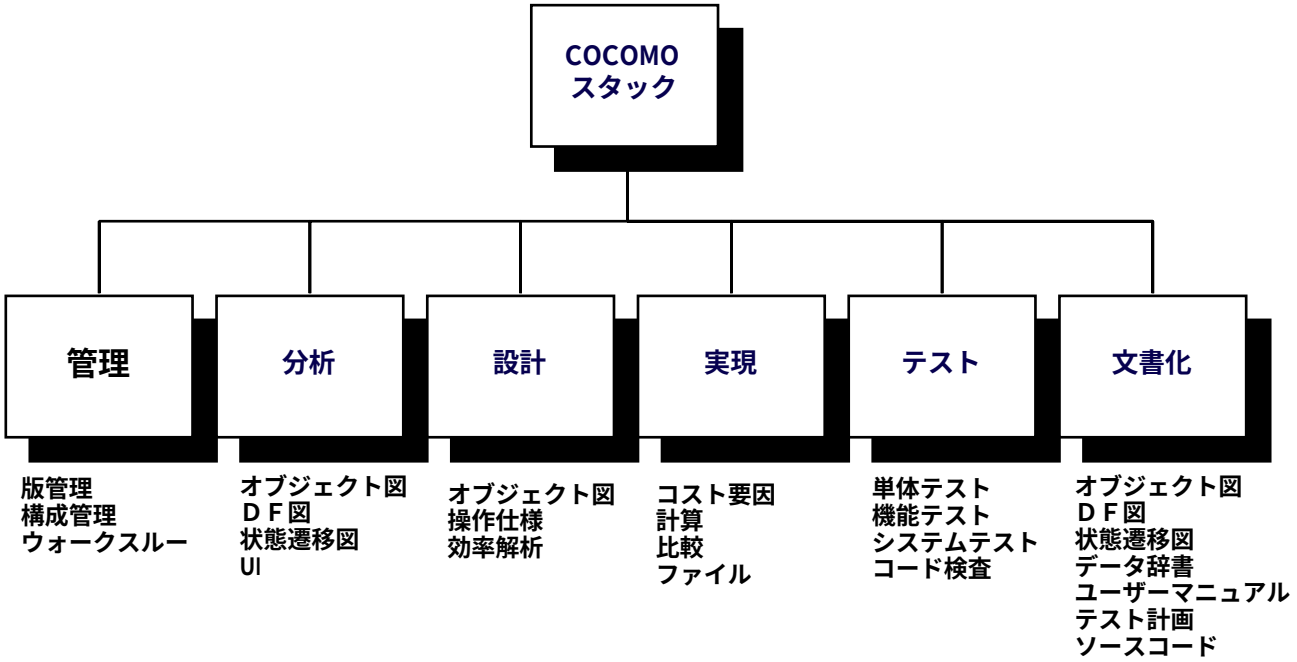
- ユーザーの要求を超えた品質水準は、生産性を上げるひとつの手段である
  - 品質はただである。ただし、品質に対して喜んで金を出す人だけに…
- どうするか?
  - 原因分析
    - 記録された欠陥をひとつひとつ分析する
      - 全部追跡しなくても、サンプルで十分
  - 信頼度成長曲線
    - 稼働可能状態の判定
      - 信頼度成長曲線と欠陥数の実績を比べて判断する
  - 仕様記述言語と段階的詳細化による証明

プロジェクト管理に関する定石

- 見積もりはどうするのですか？
  - ワーク・ブレークダウン・ストラクチャ
  - ファンクションポイント
  - COCOMO
  - 責任マトリクス
  - スケジュール管理

ワーク・ブレークダウン・ストラクチャ

- プロダクトＷＢＳ
- 活動WBS



ファンクション ポイント

- FPワークシート
- PCワークシート
- プログラミング言語との対応

FPワークシート

計算	単純			平均			複雑			合計
外部入力数	2	3	6	1	4	4	1	6	6	16
外部出力数	3	4	12	2	5	10	1	7	7	29
内部論理ファイル数	5	7	35	2	1	20	1	15	15	70
インタフェースファイル	4	5	20	3	7	21	1	10	10	51
照会	10	3	30	2	4	8	2	6	12	50
									To	216

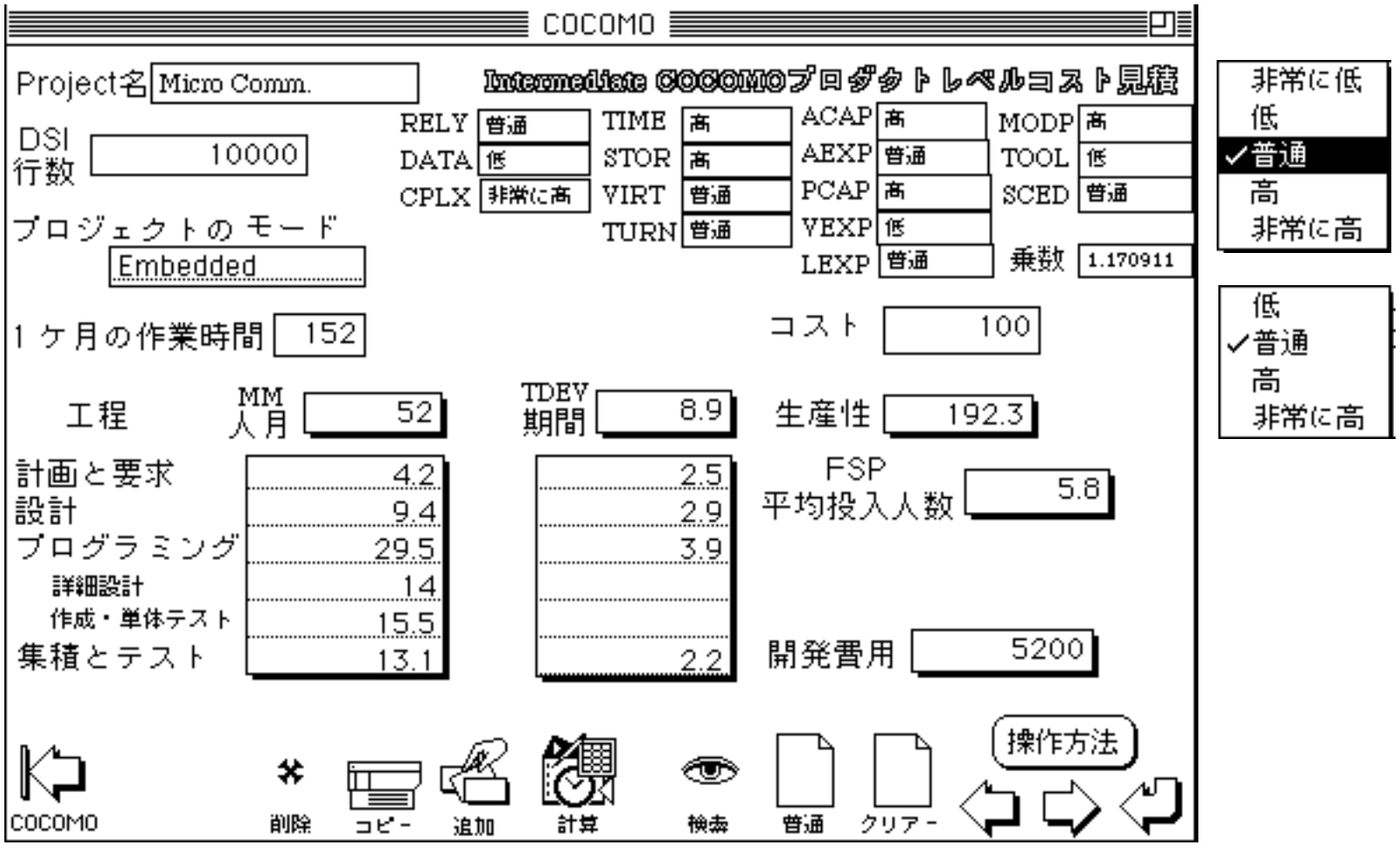
PCワークシート

項目	値			
データ通信	0	存在しないか、影響無い	0
分散機能	0	ささいな影響	1
効率	2	並以下の影響	2
複雑な構成	1	平均的影響	3
トランザクションの割合	2	大きな影響	4
オンラインデータ入力	0	重大な影響	5
ユーザーの効率	5		
オンライン更新	0		
複雑な処理	2		
再利用性	5		
インストレーションの容易さ	4		
オペレーションの容易さ	5		
複数サイト	0		
変更の容易さ	5		
Total PC	31		
PCA	0.96		
FPA	207		

プログラミング言語との対応

言語	行数/1FP
Assembler	320
C	150
COBOL	106
FORTRAN	106
Pascal	91
PL/I	80
Ada	71
Prolog	64
APL	32
Smalltalk	21
Visual Smalltalk	10
SpreadSheet	6

COCOMO


COCOMO
Project名 Micro Comm.
Intermediate COCOMOプロダクトレベルコスト見積
DSI 行数 10000
RELY 普通
DATA 低
CPLX 非常に高
TIME 高
STOR 高
VIRT 普通
TURN 普通
ACAP 高
AEXP 普通
PCAP 高
VEXP 低
LEXP 普通
MODP 高
TOOL 低
SCED 普通
乗数 1.170911
プロジェクトのモード
Embedded
1ケ月の作業時間 152
コスト 100
工程
MM 人月 52
TDEV 期間 8.9
生産性 192.3
計画と要求 4.2 2.5
設計 9.4 2.9
プログラミング 29.5 3.9
詳細設計 14
作成・単体テスト 15.5
集積とテスト 13.1 2.2
FSP 平均投入人数 5.8
開発費用 5200
COCOMO
削除
コピー
追加
計算
検索
普通
クリアー
操作方法
非常に低
低
✓普通
高
非常に高
低
✓普通
高
非常に高

責任マトリクス

タスク	佐原	引地	土屋	桜井
責任マトリクス	○			
チーム会議主催	○			
分析	○		○	
オブジェクト図	○			
ＤＦ図	○		○	
状態遷移図	○		○	
設計	○		○	○
システム設計	○		○	
オブジェクト設計	○		○	○
品質計画の管理	○			
設計検査	○		○	○
ログとメモとドキュメントの収集		○		○
版管理		○	○	
ユーザーズマニュアル		○		○
コード検査	○		○	○
単体テスト	○	○	○	○
機能テスト	○	○	○	○
システムテスト	○	○	○	○

スケジュール管理

- PERT/CPM