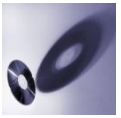
# 機能規模測定手法COSMIC法

## モデルでソフトウエアの大きさを測ろう

株式会社 オージス総研
技術部 アジャイル開発センター
木村 めぐみ

# 本日の内容

- なぜソフトウェアの大きさを測るか
- COSMIC機能規模測定法
- モデルを使った測定方法
- 事例
  - ビジネスアプリケーション
  - 大規模反復開発
  - 組み込みソフトウェア
- まとめ


# なぜソフトウェアの大きさを測るか

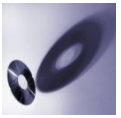


# ソフトウェア開発での問題

自分たちの
開発生産性
は？

開発上の
よいプラクティス
は？

- ソフトウェアの大きさを測ることが解決の鍵となる


# ソフトウェアの大きさの使い道

- ソフトウェアの大きさを把握するものさし
- 使い道はさまざま

$$\text{開発生産性} = \frac{\text{規模}}{\text{工数}}$$

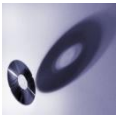
$$\text{欠陥密度} = \frac{\text{欠陥数}}{\text{規模}}$$



# ソフトウェアの大きさを測る目的

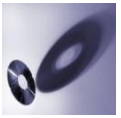
- 他と比較する
- 自分たちの開発を評価する
- よい点、改善点を見つける


よりよい開発を目指す


# ソフトウェアの大きさを測る方法

| 測定方法 | プログラムが無くても測定できる | プログラムの書き方に依存しない測定方法 | 測定が容易 |
|---|---|---|---|
| ソースコード行数 | × | × | ◎ |
| ファンクションポイント法 | ○ | ○ | × |
| COSMIC法 | ○ | ○ | ○ |

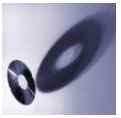

# COSMIC機能規模測定法



# COSMIC法

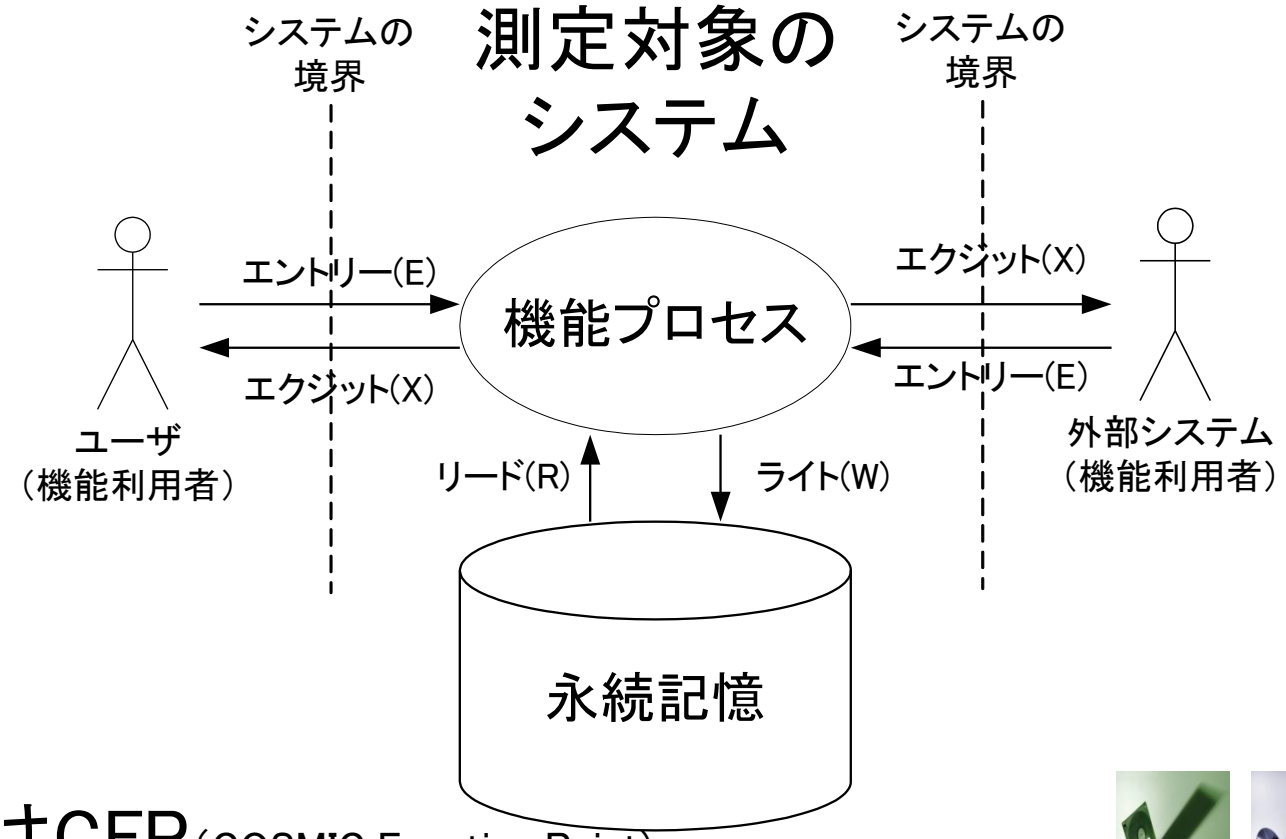
- ソフトウェアの機能規模を測定するための手法

- 標準として採用されている
  - 2002年12月、ISO/IEC19761として承認
  - 2006年12月、JIS X0413として制定

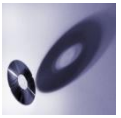




# COSMIC法における機能規模

- データ移動に基づいて機能規模を測定する




単位はCFP（COSMIC Function Point）

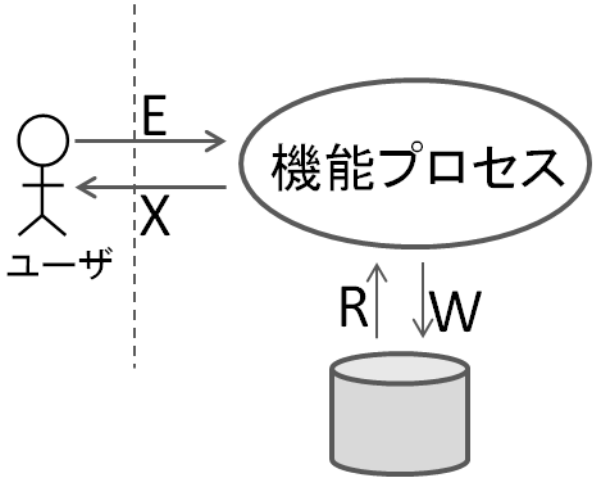




# 機能プロセス

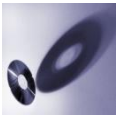
- ソフトウェアの機能利用者が起動して実行される一連のデータ移動

（例）検索機能プロセス

- ユーザが検索条件を入力して「検索」ボタンを押すことで実行される、情報を読み取りユーザに返すまでの一連の処理

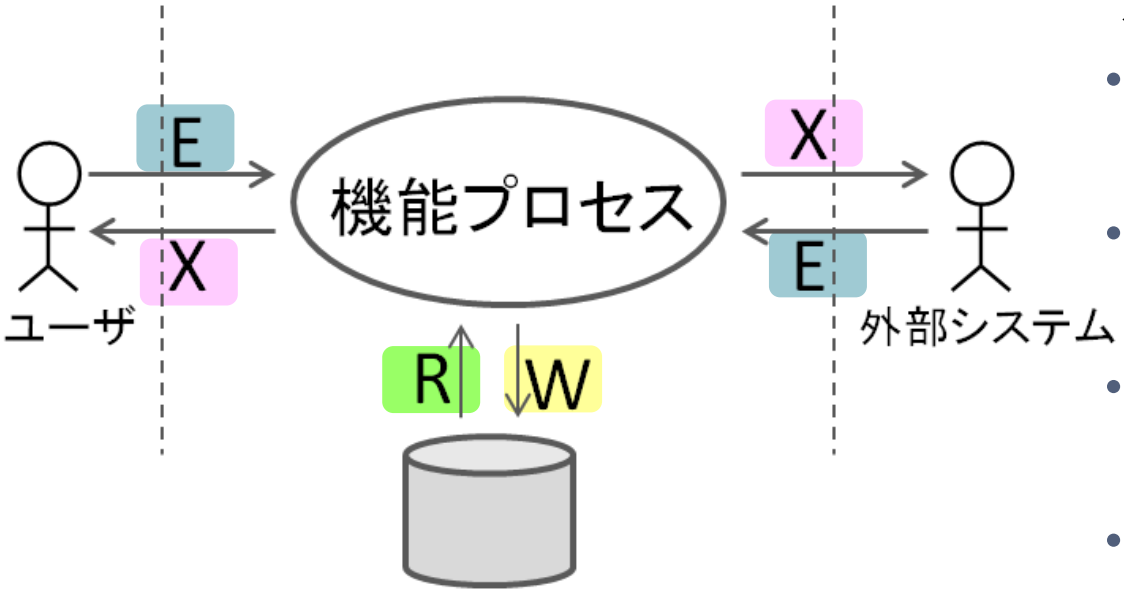






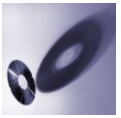

# データ移動

- データのまとまりを単位として
  データ移動の数を数える




データ移動の種類

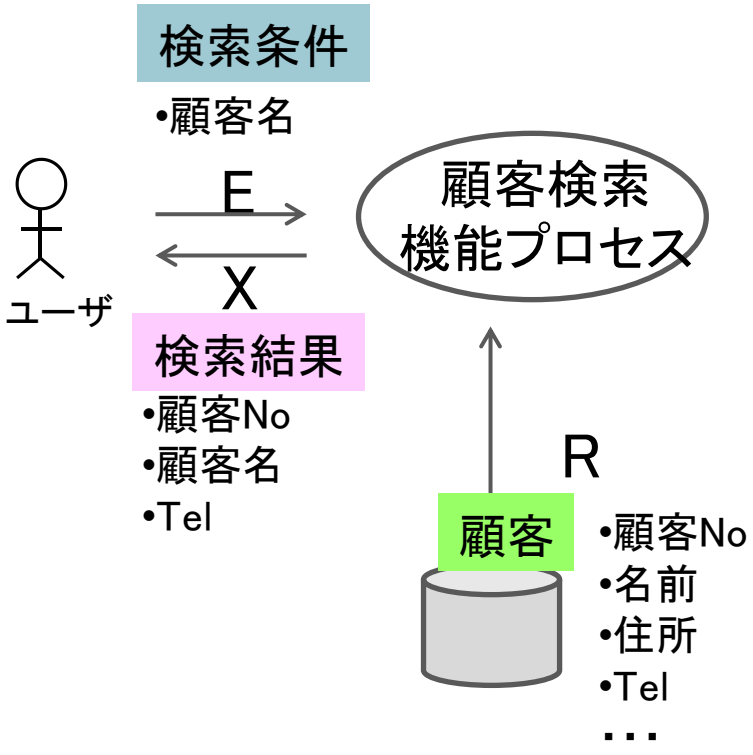
- エントリ(E)
  システム外からの入力
- エクジット(X)
  システム外への出力
- リード(R)
  永続ストレージのリード
- ライト(W)
  永続ストレージのライト

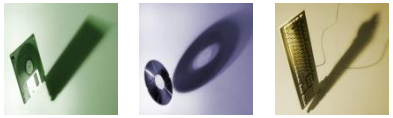

# 注目オブジェクトとデータグループ

- データ移動でのデータのまとまり

- 注目オブジェクト
  - 論理データモデルのエンティティ
- データグループ
  - 入力や出力の際のデータのまとまり
  - データ属性の集合

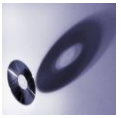



顧客検索機能プロセス＝3CFP



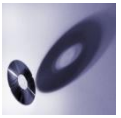
# 機能規模の算出

- システムの機能規模
  - システム全体のデータ移動E,X,R,Wの総和
- システムの変更機能規模
  - 変更されたデータ移動E,X,R,Wの総和
    - データのまとまりが追加/変更/削除されたデータ移動数

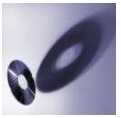




# COSMIC法の特徴

- 要求に基づいて機能規模を測定できる
- 外部システムとの通信も測定可能
- 測定方法が比較的シンプル、かつ、一般に公開されている
- 変更機能規模も簡単に測定できる

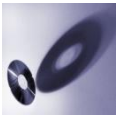

# モデルでソフトウェアの大きさを測ろう


# モデルでソフトウェアの大きさを測ろう

- COSMIC法の機能規模測定をUMLモデルで表現するよう独自に工夫

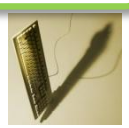
- UMLモデルで表現するメリット
  - 分かりやすい
  - 根拠が明確
  - 変更規模が測定しやすい


# 使用するモデル要素とダイアグラム

| | | |
|---|---|---|
| 注目オブジェクト | クラス図 | 注目オブジェクトの関連 |
| | クラス | 注目オブジェクト |
| データ移動 | シーケンス図 | 機能プロセス毎のデータ移動 |
| | ライフライン | •機能プロセス<br>•注目オブジェクト |
| | メッセージ | データ移動 |





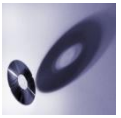
# サンプル

- 利用者機能要件
  - 見積一覧を表示する

見積書のイメージ

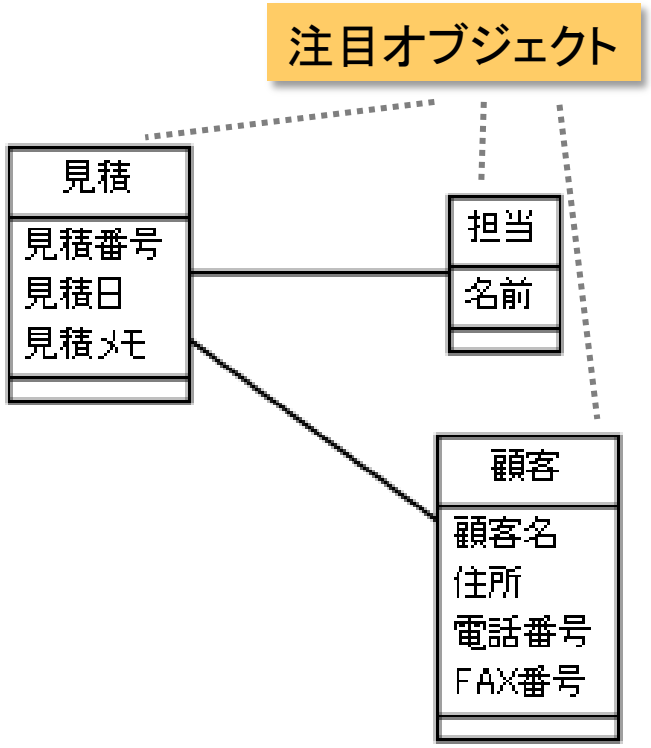
- 見積番号
- 見積日
- 備考
- 見積担当者
- 顧客名
- 顧客住所
- 顧客電話番号
- 顧客FAX番号

- 機能プロセス
  - 見積書一覧表示機能プロセス
    - 利用者は見積書の検索条件を入力して見積書の検索を実行する
    - 検索条件に合致した見積書一覧が結果として表示される

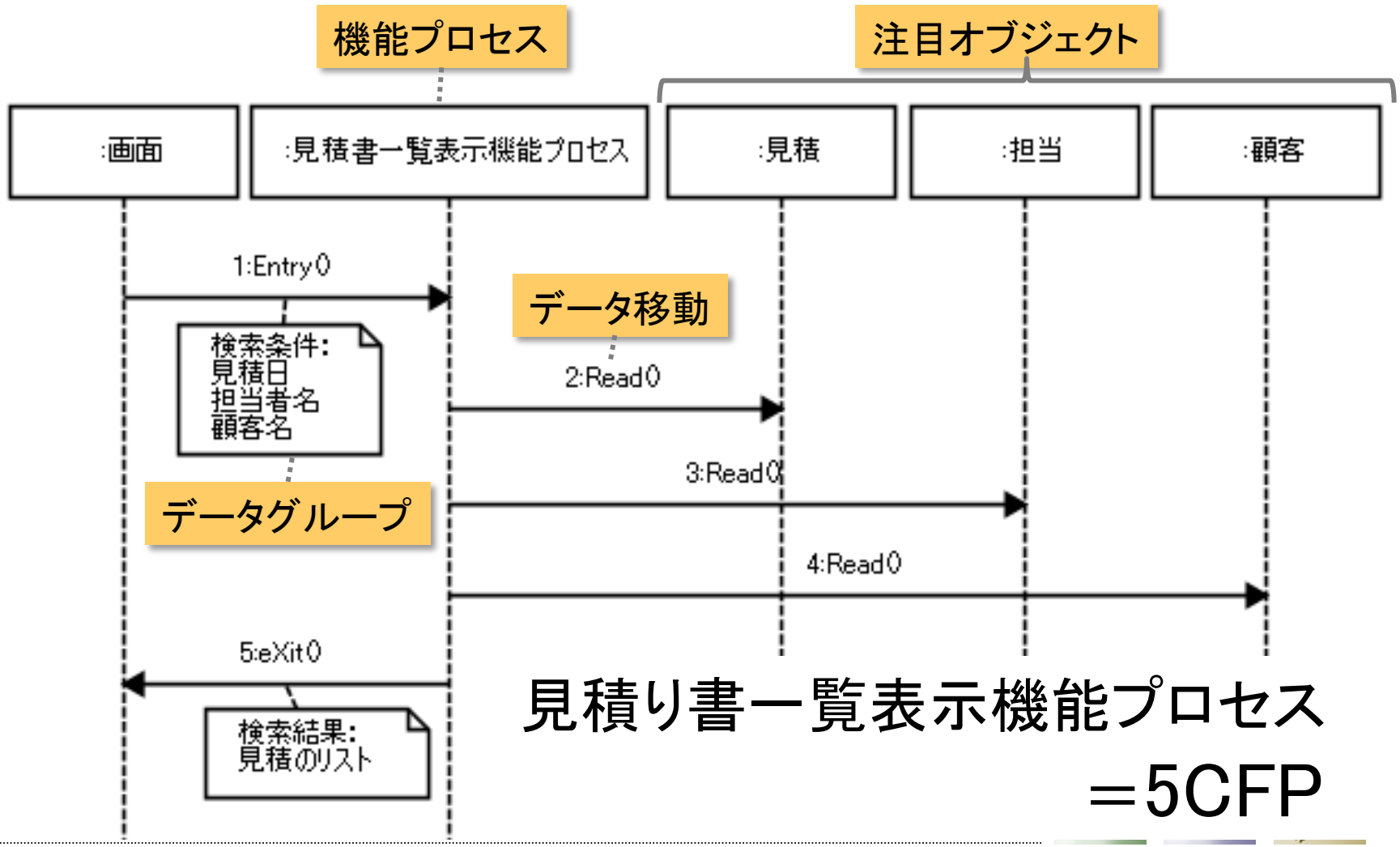

# サンプル:見積一覧を表示する

- 注目オブジェクトのクラス図



# サンプル:見積一覧を表示する

- 見積書一覧表示機能プロセスのデータ移動モデル

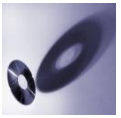



見積り書一覧表示機能プロセス
＝5CFP

# サンプル:見積一覧を表示する

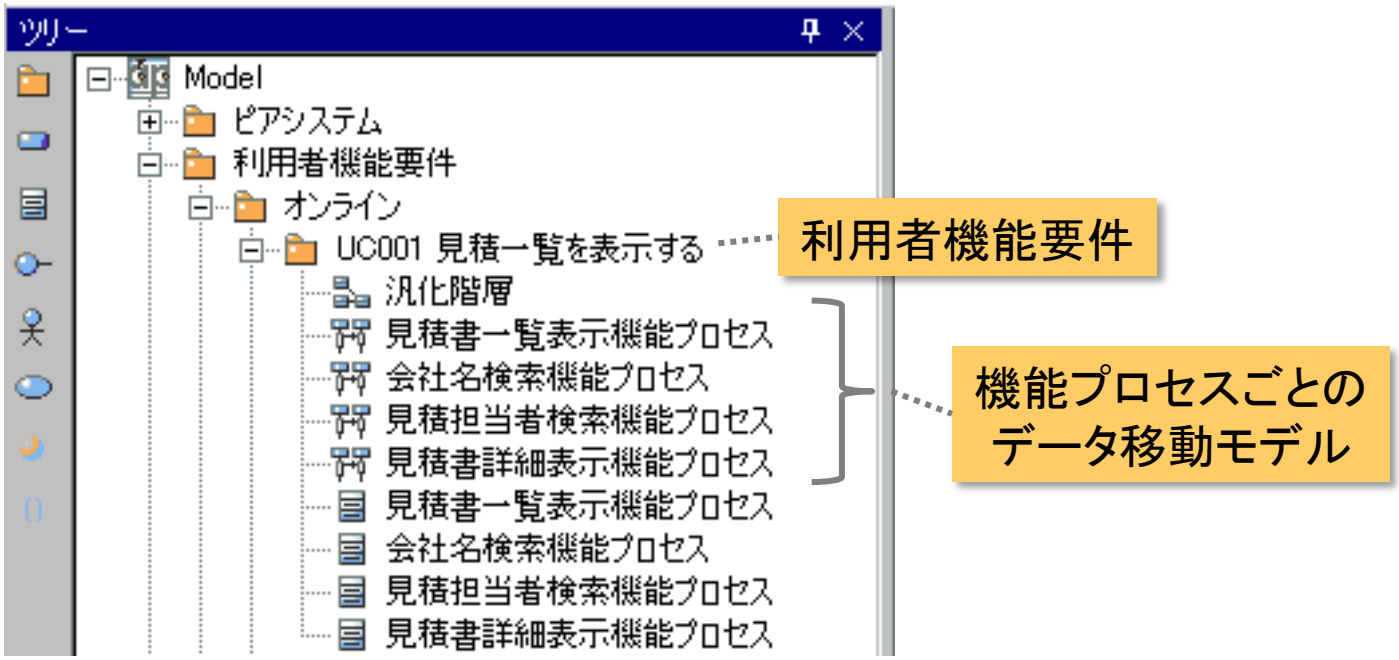
- 見積書一覧表示機能プロセスの機能規模は5CFP

| E | X | R | W | 機能規模（CFP） |
|---|---|---|---|---|
| 1 | 1 | 3 | 0 | 5 |

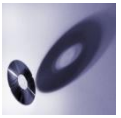




# サンプル：見積り一覧を表示する

- 利用者機能要件ごとにパッケージを分ける





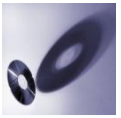



# 実績

- UMLモデルを使った機能規模測定
  - すでに10件以上を測定

分かりやすい

- 第三者にも分かりやすい
- 過去の測定結果が利用しやすい
- 測定のバラつきが小さくなった

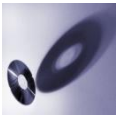




# モデルでソフトウェアの大きさを測ろう

- COSMIC機能規模測定法
- UMLモデルを利用

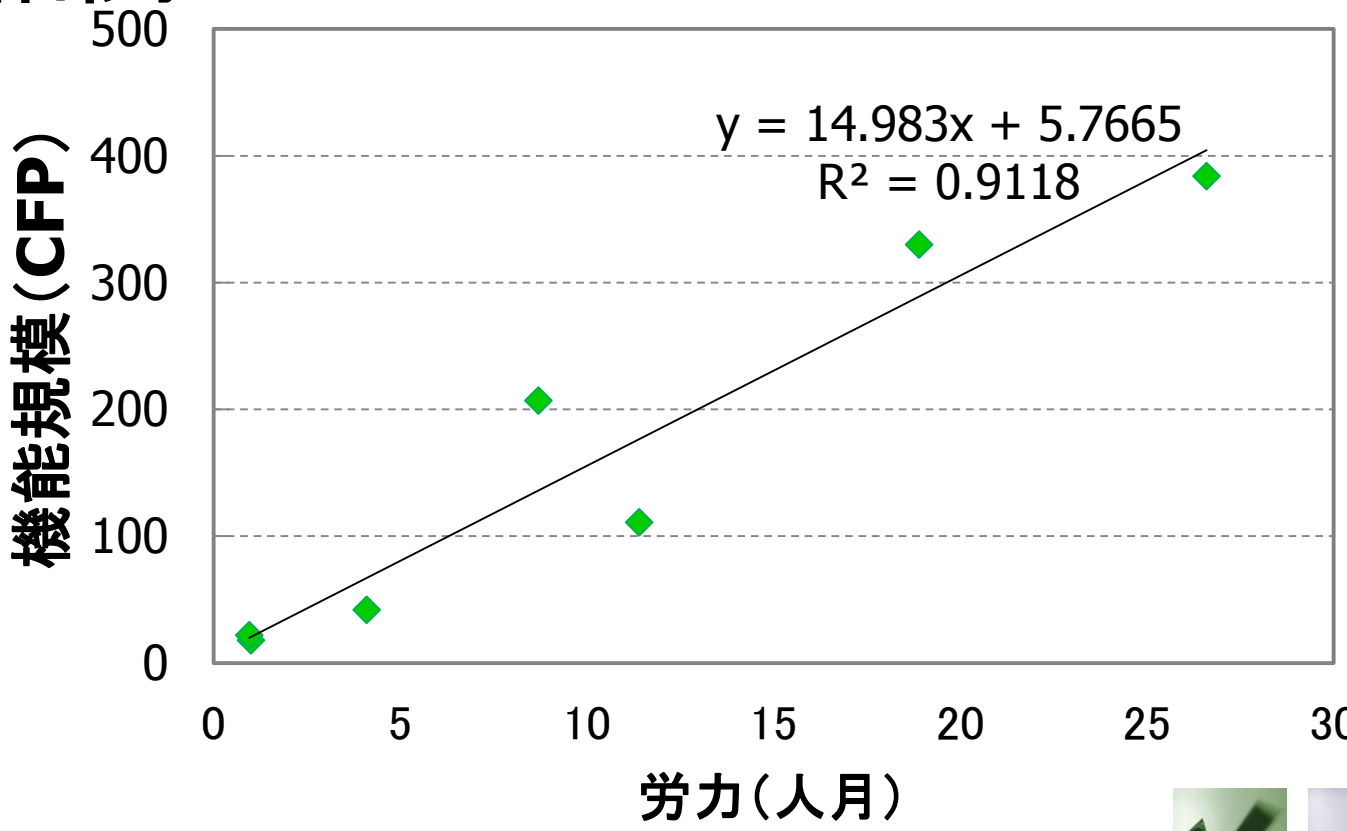
簡単

分かりやすい

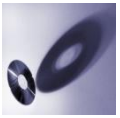




# 機能規模と労力

- これまでに測定した規模と労力は相関がよい




2009年度までに測定した7プロジェクトのデータ

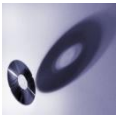




# 事例1
# ビジネスアプリケーションの測定



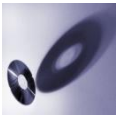
# 測定の狙い

- 開発上のよいプラクティスを横展開する
- 開発能力を高める

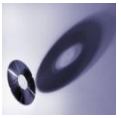

# 測定のポイント

- 生産性を測定
  - COSMIC法による機能規模
  - 機能部分と作業種別に基づく労力の記録
- 生産性のバラつき要因を分析


# 測定したプロジェクト

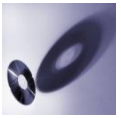
- Force.comで画面機能を開発
- 少人数・短期間開発
  - 要員：2名
  - 開発期間：2ヶ月半


# Force.comでの画面開発

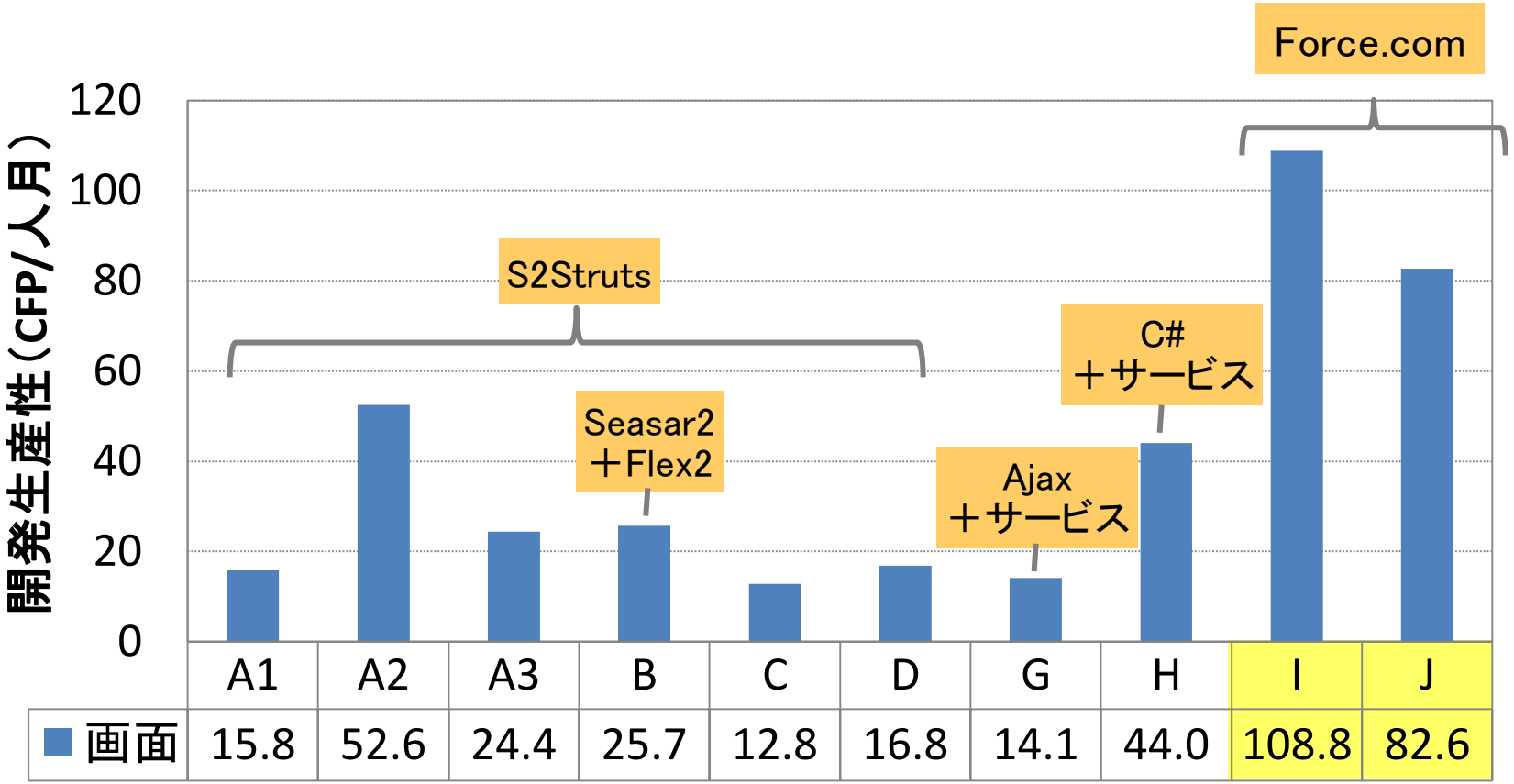
- Force.com標準画面
  - テーブル定義するだけで画面生成
  - シンプルなUI画面

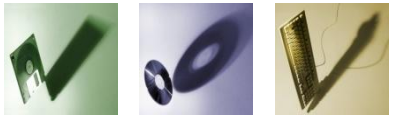
- カスタマイズ開発
  - Visualforceを利用して開発
  - タグの記述、サーバサイドを実装
  - 複雑な画面も実装可能


# プロジェクト別開発生産性

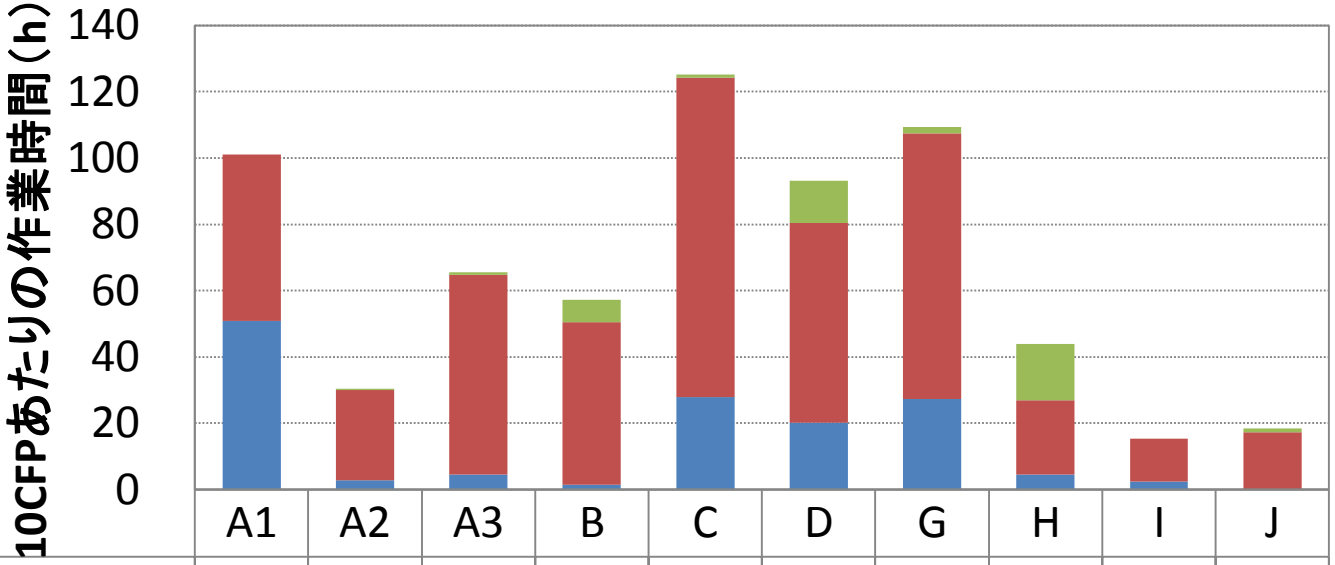
- 開発生産性が非常に高い



# プロジェクト別単位機能規模あたりの労力

- 単位機能規模あたりの実装･単体テスト時間が小さい

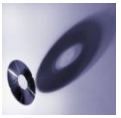



| | A1 | A2 | A3 | B | C | D | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 結合テスト | | 0.5 | 0.8 | 6.9 | 1.1 | 12.6 | 2.0 | 17.2 | | 1.3 |
| 実装･単体テスト | 50.2 | 27.1 | 60.1 | 48.9 | 96.3 | 60.2 | 80.2 | 22.2 | 12.9 | 17.2 |
| 詳細設計 | 50.9 | 2.8 | 4.6 | 1.5 | 28.0 | 20.3 | 27.3 | 4.6 | 2.5 | |

事例1

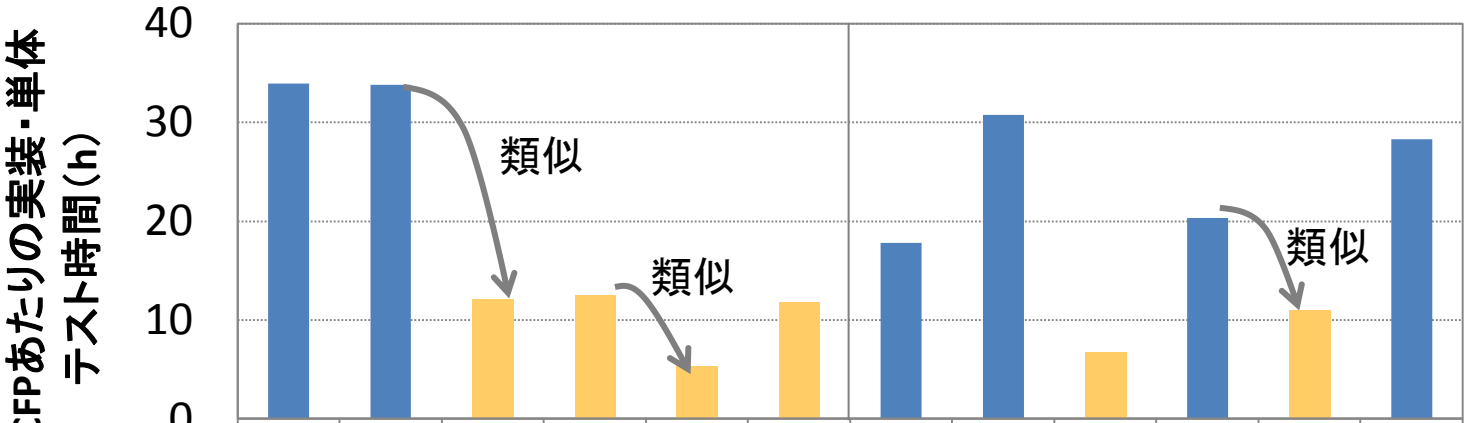
# プロジェクト間で比較して推測できること

- 推測
  - Force.comを利用したことによって開発効率が上がった
- 開発生産性に影響した他の要因は？
  - 機能間の分析値を比較


# 機能別単位機能規模あたりの労力

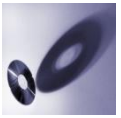
- 開発生産性に影響した要因
  - 類似画面の開発経験
  - 画面コンポーネントの開発経験




| | I1 | I2 | I3 | I4 | I5 | I6 | J1 | J2 | J3 | J4 | J5 | J6 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 実装・単体テスト | 34.0 | 33.8 | 12.1 | 12.5 | 5.4 | 11.8 | 17.8 | 30.8 | 6.8 | 20.3 | 11.0 | 28.3 |
| 新しい画面要素 | 多 | 少 | 少 | 少 | 少 | 少 | 多 | 少 | 少 | 多 | 少 | 少 |
| 新しい画面要素数 | 8 | 3 | 1 | 1 | 1 | 2 | 8 | 1 | 2 | 4 | 0 | 2 |

# 開発上のよいプラクティス

- 類似機能や画面コンポーネントの開発経験を考慮した開発計画を立てる
- フレームワークを利用する
  - Force.comで画面を開発したことにより開発が効率化した
  - フレームワークを独自に開発した場合も効果が期待できる

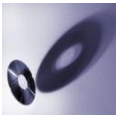




# 事例2
# 大規模反復開発の測定



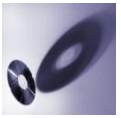
# 事例2

- 大規模反復開発で機能規模の測定コストを削減
  - 概算法
- 開発期間の途中でプロジェクトの状況を定量化


# 測定の狙い

- 大規模開発プロジェクトの開発中にプロジェクトの状況を把握
- 中間レポートをプロジェクトにフィードバック
  - 改善点の発見
  - 今後の開発工数を予測

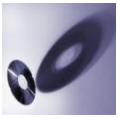




# 大規模プロジェクトでのCOSMIC法適用

- 機能規模を概算法で測定
  - 全体の1%程度の機能のデータ移動を測定し、平均規模を求める
  - 概算機能規模
    ＝全機能プロセス数×平均規模

| 全機能プロセス数 | × | 機能プロセスの平均規模（CFP） | ＝ | 機能規模の概算値（CFP） |
|---|---|---|---|---|

例）450機能プロセス×8.2CFP＝3,690CFP


# 測定したプロジェクト

2010年度 | 2011年度 | 2012年度

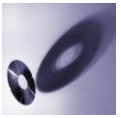
測定開始　レポート フィードバック　…　リリース

基本設計 | 1st 反復 | 2nd 反復 | 3rd 反復 | システムテスト

機能規模

3,719 CFP（全体の 25%）

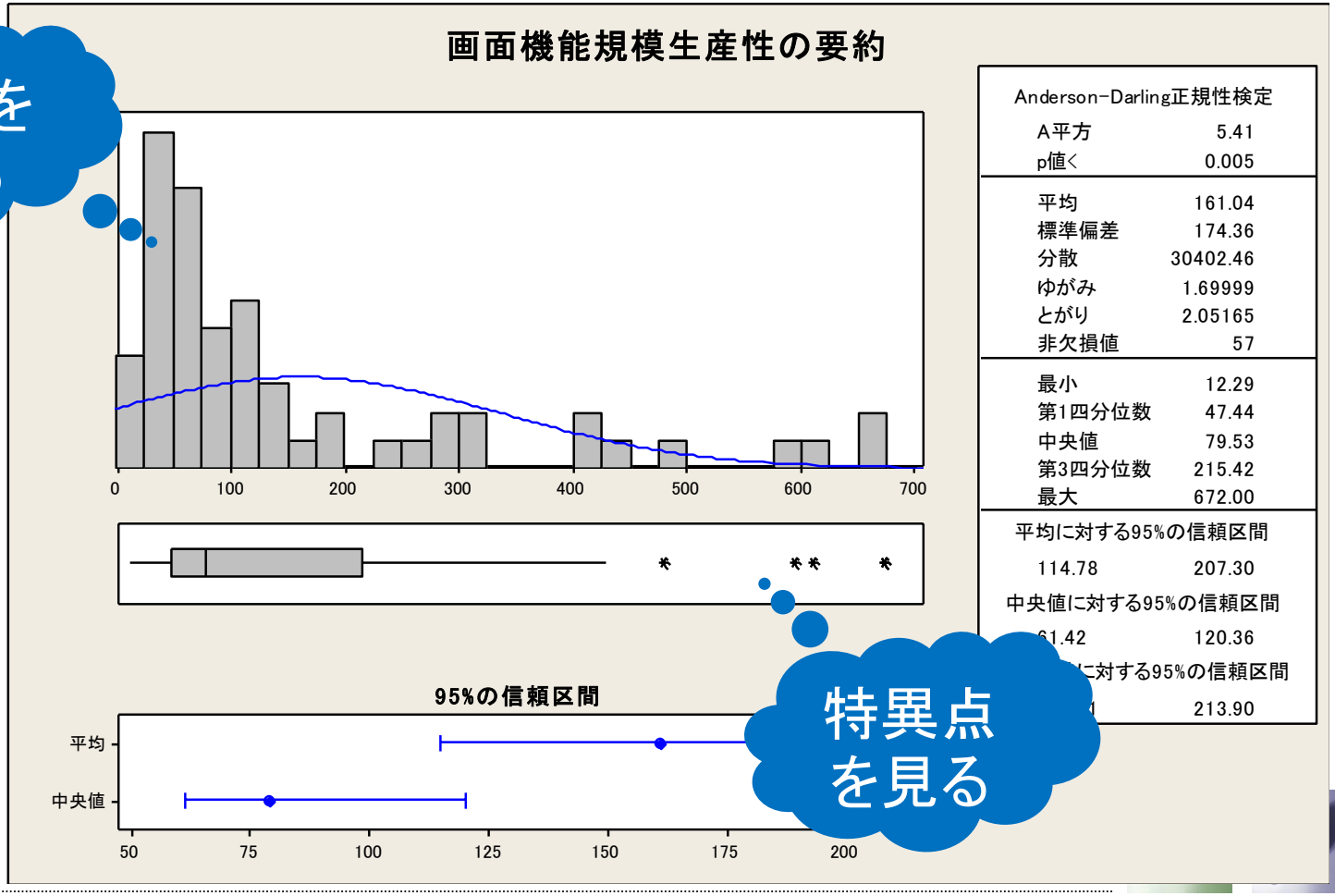
14,444CFP


# 中間レポート:開発生産性

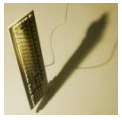
- 開発途中に生産性の傾向やバラつきを把握できる
- 特異点はアラート⇒詳細な分析値を確認

傾向を見る



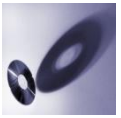
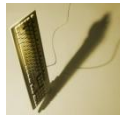

| Anderson-Darling正規性検定 | |
|---|---|
| A平方 | 5.41 |
| p値< | 0.005 |
| 平均 | 161.04 |
| 標準偏差 | 174.36 |
| 分散 | 30402.46 |
| ゆがみ | 1.69999 |
| とがり | 2.05165 |
| 非欠損値 | 57 |
| 最小 | 12.29 |
| 第1四分位数 | 47.44 |
| 中央値 | 79.53 |
| 第3四分位数 | 215.42 |
| 最大 | 672.00 |
| 平均に対する95%の信頼区間 | |
| 114.78 | 207.30 |
| 中央値に対する95%の信頼区間 | |
| 61.42 | 120.36 |
| に対する95%の信頼区間 | |
| | 213.90 |

特異点を見る



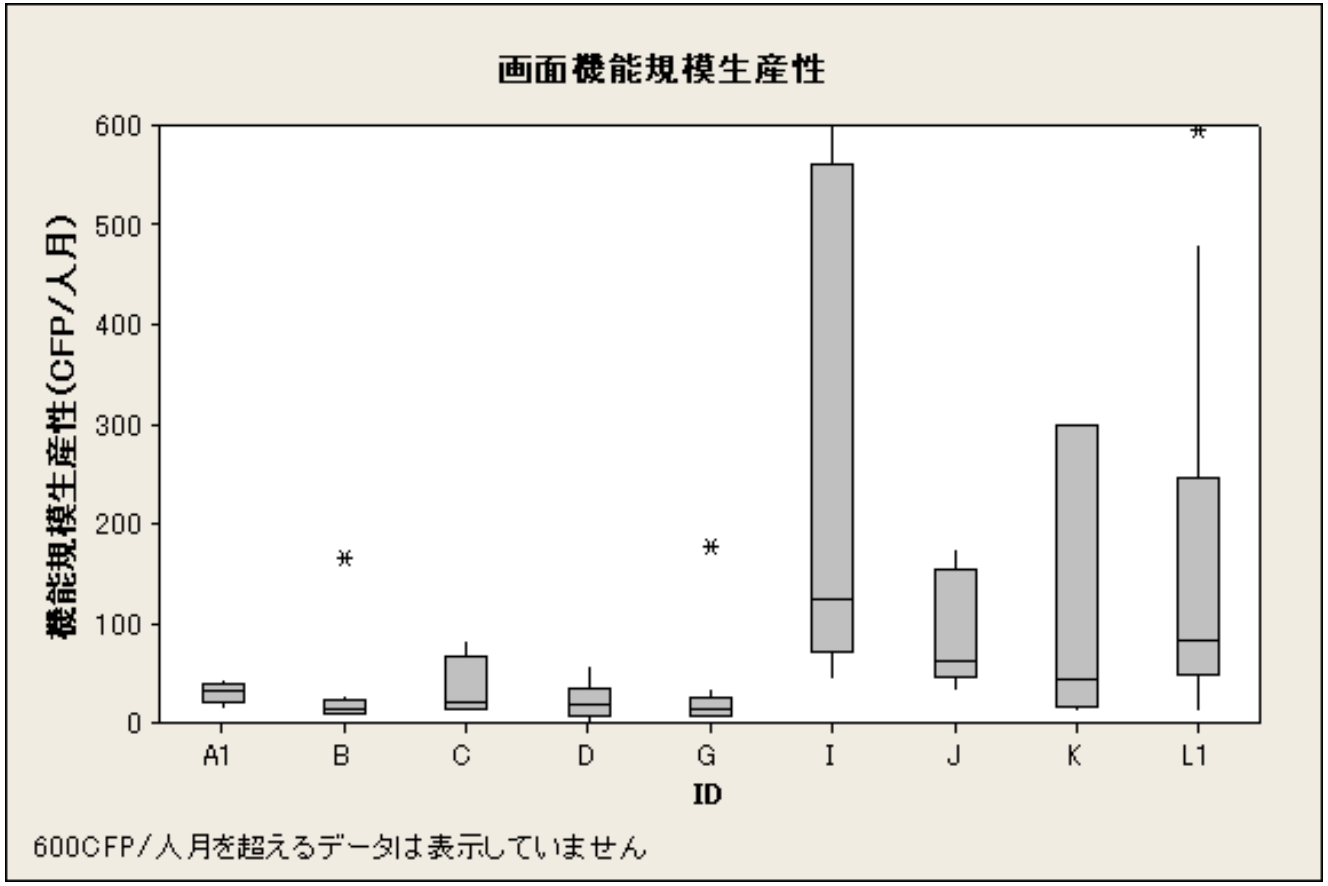
# 概算法の評価

- 概算法により測定コストは10分の1に削減
- プロジェクトが把握する開発効率は概算規模から求めた生産性と一致
- 現時点で概算規模は妥当と判断

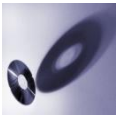
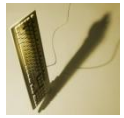




# 中間レポート:プロジェクト間の比較

- プロジェクト間で生産性を比較

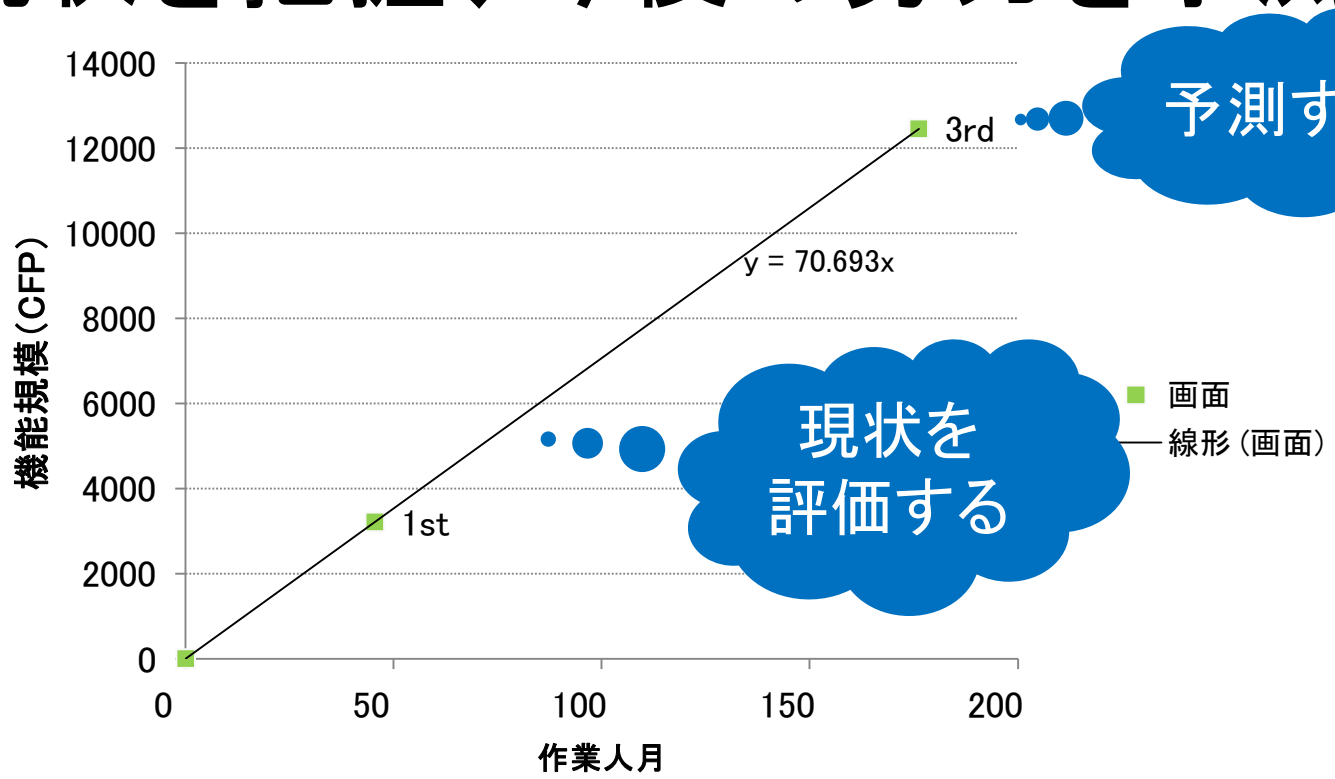




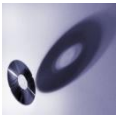



# 中間レポート:労力の予測

- 労力の実績と機能規模から
  現状を把握、今後の労力を予測








# 中間レポートで分かること

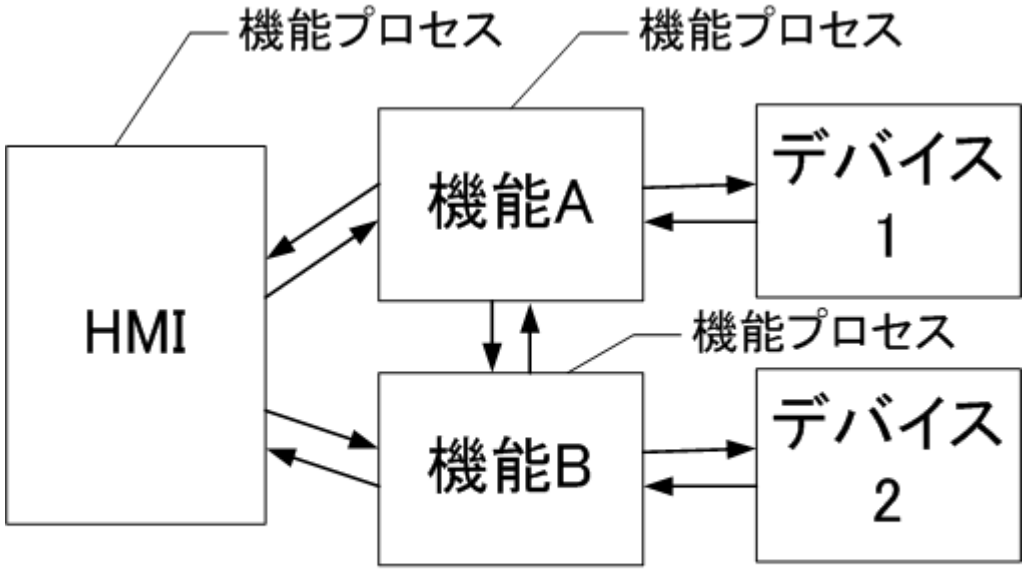
- 開発途中の状況が定量的に分かる
- 改善すべき点
  - 特異点がアラートとなる
- 残りの期間の開発工数の予測

# 事例3
# 組み込みソフトウェアの測定



# 事例3 組み込みソフトウェアへのCOSMIC法適用

- 機能プロセスの識別

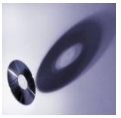



・・・

- 抽象度が揃った機能プロセスに分割
- 機能プロセス間のデータ移動を求める

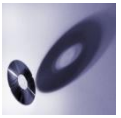
# COSMIC法の適用

- 基本検討
  - 測定方法の確立
  - 測定結果の妥当性確認
- 本格展開
  - 生産性の測定
  - プロセス改善
  - 見積もり


# 測定の狙い

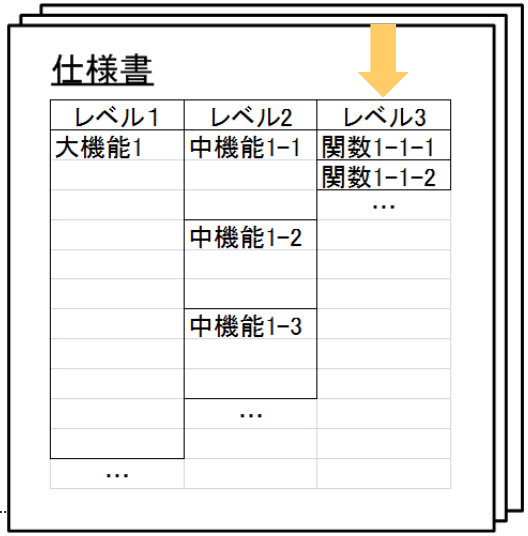
- 某メーカー様
- グローバルな競争を行う上で、自社のソフトウェア開発生産性を測定したい
- 仕様変更を行う際の開発コストやカスタマイズを行うコストを見積りたい


# 測定方法の確立：データ移動の識別

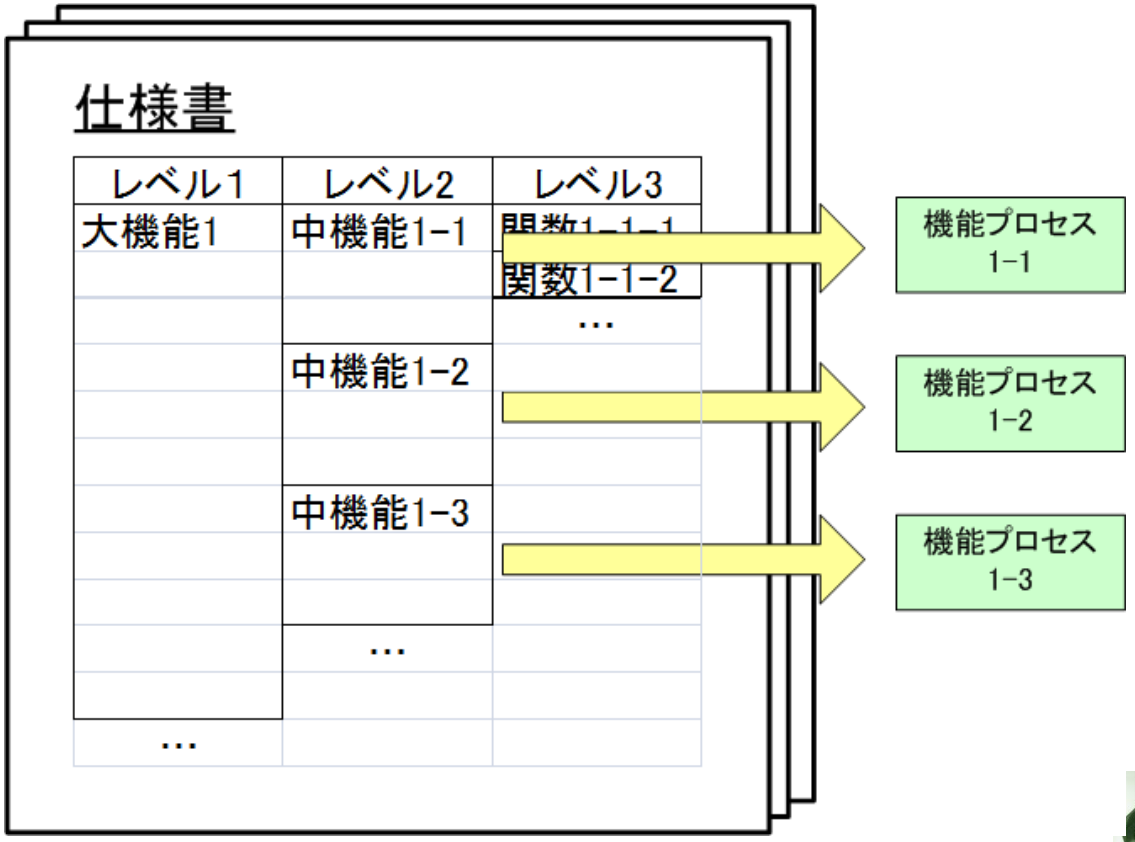
- ソフトウェアの機能仕様書
  - 複数の見出しレベルで機能を定義
  - 一番詳細な関数レベルの仕様
    - 入出力変数
    - 継続的に使われる内部変数
    - 関数が実行するロジック

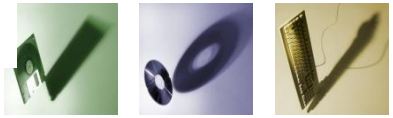
↳ データ移動を識別



# 事例3 測定方法の確立：機能プロセスの識別

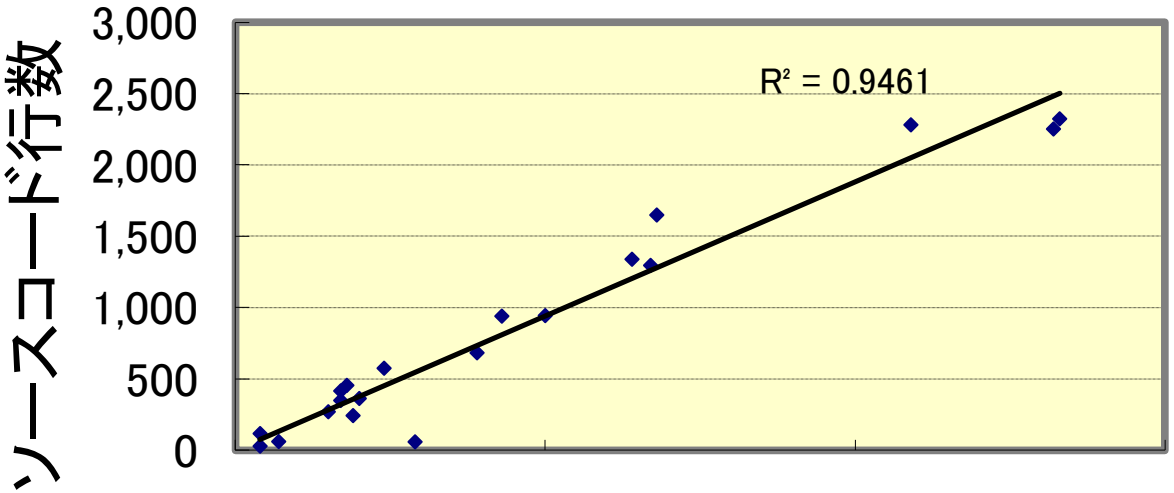
- 機能プロセスは、仕様書の見出しレベルに基づいて設定



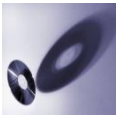
# 測定結果の検証

- 仕様書から測定した機能プロセス毎の機能規模と、ソースコード行数の相関関係を確認



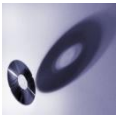
# 他のドメインへの展開

- 2番目のドメインでは、インタフェースの説明書から機能規模を測定
  - 機能規模とソースコード行数の相関はよかった
- 現在、複数のドメインの機能規模測定方法を確立中

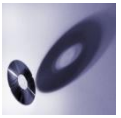

# 今後の展開

- 測定した機能規模の利用
  - 生産性の測定
  - プロセス改善
  - 見積り

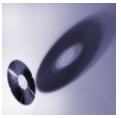




# 事例を通して分かったこと

- 機能規模を測定すると
  - 生産性に影響した要因を特定して開発上のよいプラクティスが分かった
- 大規模反復開発の測定
  - 概算法で妥当な機能規模が低コストで測定できる
  - 開発途中のプロジェクトの状況が定量化できた


# まとめ

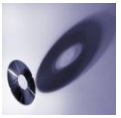

# まとめ

- COSMIC機能規模測定法
  - 簡単に機能規模を測定できた
  - 測定した機能規模は労力と良く相関
- UMLを利用した機能規模測定
  - 測定結果が第三者に分かりやすくなった
- 活用事例
  - プロジェクトの生産性を定量化
  - 開発上のよいプラクティスを知る
  - 開発途上でフィードバック



# 今後の課題

- 測定した機能規模を見積りに利用
- 概算法による測定精度の検証
  - 測定コスト　⇔　測定精度


# COSMICとUMLを利用した機能規模測定

簡単

分かりやすい

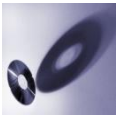
役立つ



# 参考文献(その1)

- COSMIC機能規模測定法マニュアル第3.0版 ー測定マニュアル, 2007年9月
  - 日本ファンクションポイントユーザグループ(JFPUG)のWebサイト(http://www.jfpug.gr.jp/cosmic/CFFP-index.html)から入手可能

- COSMIC-FFPによる業務アプリケーションソフトウェア規模測定の指針(第1版), 2005
  - 業務系アプリケーションに即した例とともに機能規模の測定指針を解説
  - COSMIC ver2.2ベースなのでver3.0とは用語が少し異なる点に注意してください！
  - 上記JFPUGのサイトで提供中

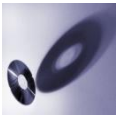




# 参考文献(その2)


- 山口正明, 調重俊, ファンクションポイント COSMIC-FFP法実践ガイド -組込み系・リアルタイム系に最適なソフトウェア規模・工数の見積り方法,日科技連出版社,2007

- 藤井拓, 木村めぐみ, COSMIC 法で求めた開発生産性バラつき要因の分析, 情報処理学会研究報告2010-SE-170, 2010






# 参考文献(その3)

- オブジェクトの広場「ビジネスアプリ開発者のための機能規模測定手法 COSMIC 法入門」, 2010年6月から連載中
  - COSMIC法によるビジネスアプリの機能規模測定方法のチュートリアル
  - http://www.ogis-ri.co.jp/otc/hiroba/technical/IntroCOSMIC/index.html