# 計測 と 見積り

～ ソフトウェア開発のよりよい見積りのために ～

**biac**

**わんくま同盟 名古屋勉強会#07**

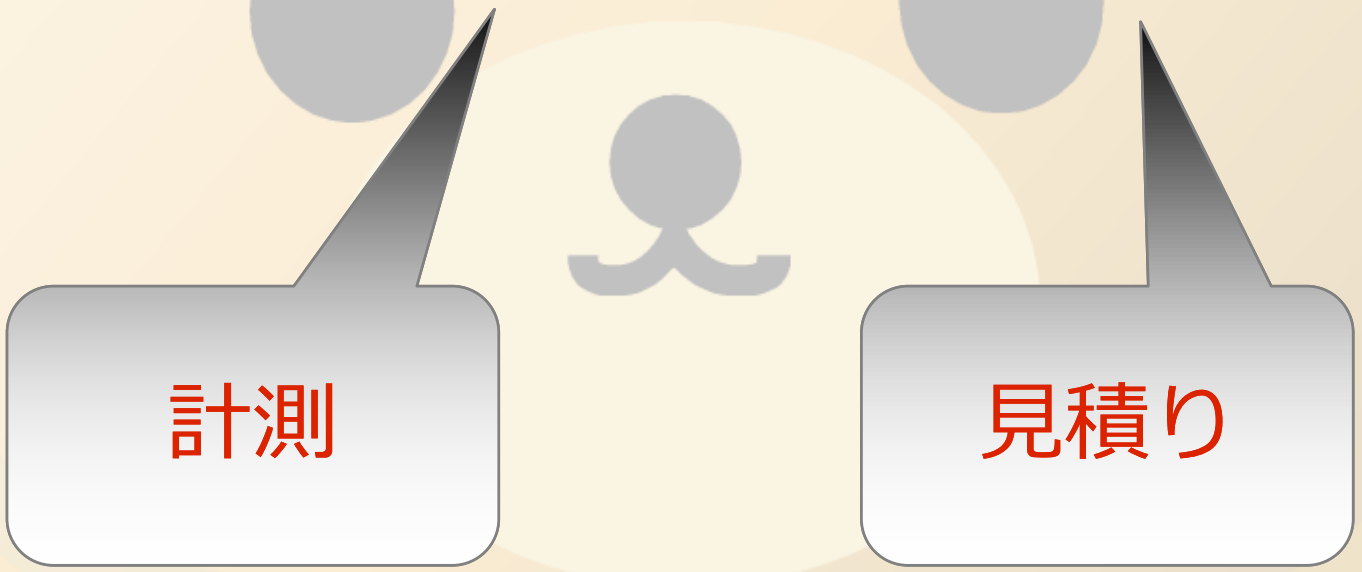
**2009/04/11**

# 「見積り」 とは

- **過去の経験に基づいて、未来を予測する技術**

  ※ 工数・工期の見積りまでは技術。 見積書に記載する金額は「政治」

- **すなわち、過去のデータを未来にあてはめること**



# 見積りの基本 = サイズ ÷ 速度

- 例) 遠足
  - 出発地点から目的地までの距離 (サイズ) : 8km
  - 歩く速さ (速度) : 4km/h
  - 必要な時間は? ⇒ 8 ÷ 4 = 2時間
- 注意
  - ここでは、サイズは正確な値。 ぶれる事はない。
  - 速度 (4km/h) は、経験値。 実際にやってみると、たぶん違う値になる。

# 見積りの基本 = サイズ ÷ 速度

- 例) 遠足2 … サイズも推定値
  - 出発地点から目的地までの直線距離：8km
    実距離の推定：折れ曲がり係数(仮称) = 1.5
    ⇒ 実距離(推定) = 8 × 1.5 = 12
  - 歩く速さ (速度)：4km/h
  - 必要な時間は? ⇒ 12 ÷ 4 = 3時間

# 見積りの精度 ～ サイズ と 速度

- ソフトウェア開発の見積り精度を良くするには
  - プロジェクトのサイズをより精確に見積もること
  - チームの速度 (生産性) をより精確に見積もること
- × 見積り結果だけを精度アップさせよう
  - 生産性の向上分が計測できない
    ⇒ 開発者のスキルアップを評価しない
  - サイズの見積り外れ(仕様膨張)を見ない
    ⇒ 遅延の原因はすべて開発者の責任にされる

# 閑話休題 (1)：遅れを取り戻せ!

- 予定工期の半分で 遅延が発覚
  - トータル 8km、 1時間で 3km だった
    ⇒ (4km(見積り) − 3km(実績)) ÷ 4km
    = 25% (遅れ)
  - 残り 1時間で 5km だから、後半のスピードを…
    ⇒ 5km/h(目標) ÷ 4km/h(見積り) = 125%
    … 25% アップすればよい (?)
  - これは実績ベースで考えないといけない！
    ⇒ 5km/h ÷ 3km/h(実績) = 167% !!

# ソフトウェアのサイズ (1)

- コード行数
  - 一定の条件を満たせば、サイズを測るのに使える。
- 同一言語、同一ライブラリ
- 同一の品質
  - コード品質計測ツール
  - リファクタリングに主眼を置いたコードインスペクションの実施

# ソフトウェアのサイズ(2)

- 画面数と帳票数
  - 日本の業務アプリでは、けっこう使える

    ※( 人月 = 画面数 x 1.55 )

# ソフトウェアのサイズ (3)

- ファンクションポイント (FP)
  - 現在、世界的に一番権威がある
  - FP算出法にも何通りかあるので、どの手法で計測したか明らかにすること

    NESMA, NESMA簡易法, 画面要素法 (日本)

# ソフトウェアのサイズ (4)

- その他
  - オブジェクトポイント法、ユースケースポイント法など
  - 粒度を一定にするのが難しそう
- 計測手法の選定には
  - 安定して計測できること
  - 他と (時間軸, 空間軸) 比較可能なこと

# 成果物の見積り

- 生産性を、個々の成果物 (work) の量に対して計測/見積りしている場合は、ソフトウェアのサイズから成果物の量を見積もる
  - 設計書のページ数
  - コードの行数
  - テスト報告書のページ数
  - マニュアルのページ数

# 生産性 (1)

- ※ 生産性を見積もる ← 生産性を計測すること
- 作業を、少なくとも 3つに分けて計測する
  - 要件開発 (なにを作るか) … いわゆる要件定義から外部設計まで
  - コンストラクション (どう作るか) … いわゆる内部設計から実装完了まで
  - 検証 (どこか間違っていないか) … いわゆる結合テスト以降

# 生産性 (2)

- 細かく見る場合には、前述の成果物 (work) 単位にまで分けて計測することも。
  - ※ 米国産の見積り支援ソフトには、そこまで入っている。
    - 中間成果物まで細かく規定されている「重たい」開発プロセスなら、可能。

# FP - 成果物量 - 生産性 の例

- (米国のデータ例 )

# 生産性データの適用範囲

- ソフトウェア開発の生産性はバラつきが大きい
  - 人 (属人的なスキル)
  - チーム (属人的、組織的なスキル)
  - 開発プロセス
  - ※ 求められる品質と生産性の関係は、おそらく定量化可能
- 条件が変わったら、適用できない！
  - 推定 + 不確実性を大きく見る必要がある

# 閑話休題 (2)：テストの工数

- 「テスト工程」の工数はバラつく
  - テストケース開発 ～ テスト実施 ← サイズに比例
  - バグ票の起票 ～ トリアージ ～ 不具合修正 ～ 再テスト ← バグ数に比例
  - 起票以降の工数は、バグを作りこんだ工程 (設計/製造) に振替えるべきかも。
    ※ そうすれば、設計/製造での品質向上へのインセンティブになる。

# 不確実性は非対称 ～ β分布

- 短縮は限度があるけど、遅延はどこまでも
- 確率分布が最大の点より、期待値は右側

# 3点見積り ～ β分布の近似

- 最良と最悪も加えた 3点から算出する

# 閑話休題 (3)：レビューの工数

- 文書のバグ出しレビュー　＝インスペクション
  - レビュアーによる査読：0.x～0.xH/ページ ×3人
  - レビューミーティング：0.x～0.xH/ページ ×5人
  - 文書修正：0.x～0.xH/ページ ×1人
  - レポート・フォロー・進行：0.x～0.xH/ページ x1人
  - 合計：0.x～0.x人・H/ページ

# プロジェクトとの関わり

| | |
|---|---|
| 開発プロセス | ・計測と見積りの区切りかた |
| プロジェクト実施 | ・計測<br>・進捗のフィードバック |
| プロジェクトの評価 | ・生産性のフィードバック |
| 案件の引き合い | ・ラフな見積り |
| プロジェクト計画 | ・見積り<br>・計画へのフィードバック |

# まとめ(1)：計測なくして 見積りなし

- 精確な計測から、精確な見積りが生まれる
- 精確な見積りができると…
  - 赤字プロジェクト・デスマーチが減らせる
  - 生産性の向上が計れるようになる

    ※ 見積り精度が±10%で、実績が20%良かったら、 10%位は生産性が上がったと言える。

    ⇒ 開発プロセスの改善が進む

# まとめ (2)：工数見積りは技術

- サイズ・工数を見積もるところまでは、技術
  - 「政治」嫌いの開発者も知ってほしい
- 過去の計測値から未来を予測する
  - 計測していなければ、良い見積りは出せない
- 見積りは確率
  - 未来予測に 100% 正確はありえない
  - 幅のある未来予測から決断するのは経営責任

# 参考書