ソフトウェア開発見積りの基本的な考え方

～見積り手法の課題・歴史とCoBRA法～

2012年4月20日

CoBRA研究会
株式会社三菱総合研究所
先進ソリューションセンター
石谷靖

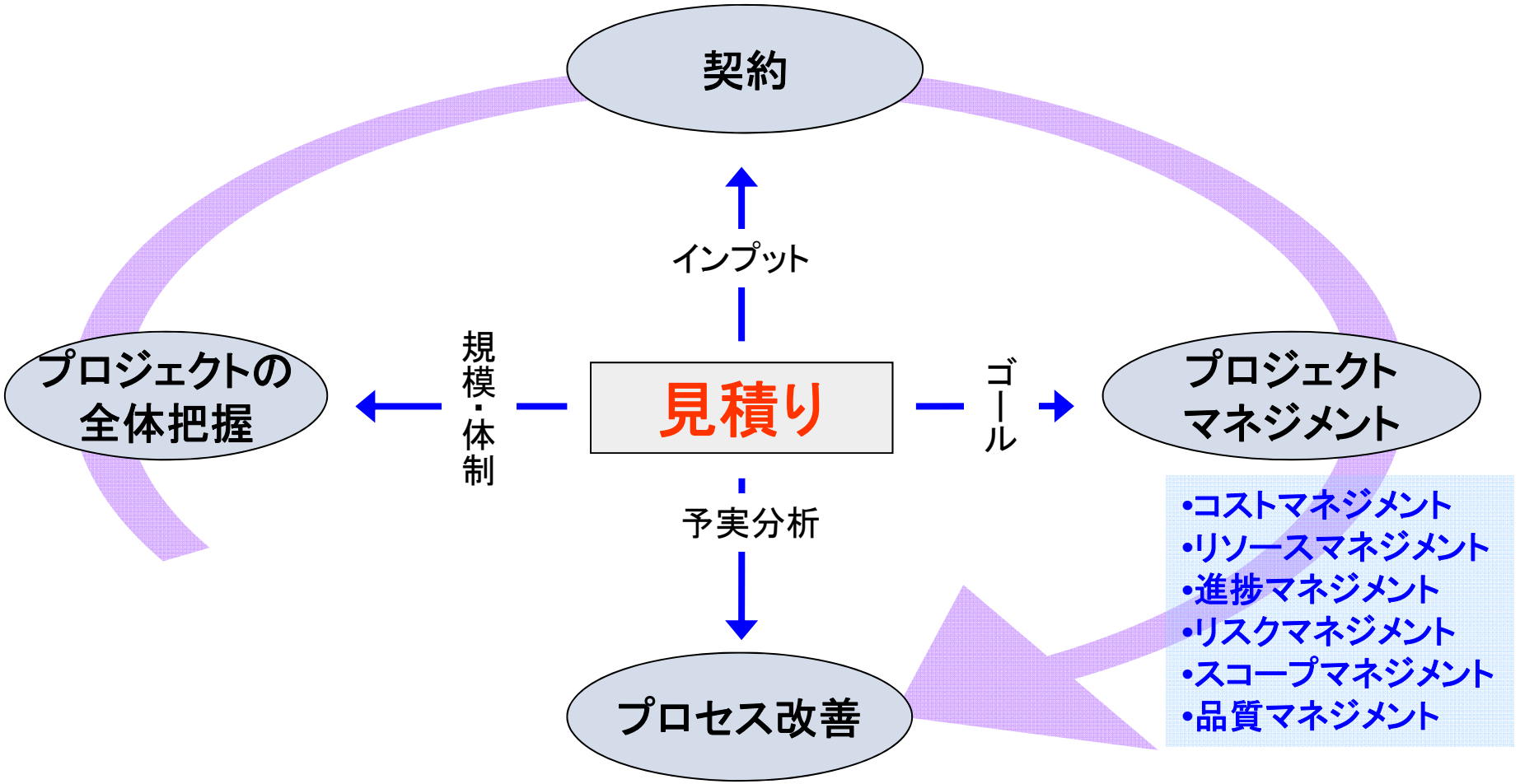
目次

1. プロジェクトマネジメントにおける見積り
2. 見積りにおける課題
3. 見積り手法の歴史と蓄積
4. ソフトウェア開発見積りにおける基本的なアプローチ

1．プロジェクトマネジメントにおける見積り

プロジェクトマネジメントにおける見積りの位置づけ

プロジェクト・マネジメントの中核



ソフトウェア見積りの重要性

プロジェクトにおける見積りの重要性

- プロジェクトにとって見積りは、成功を目指して計画を立て、状況を把握し、必要に応じてコントロールするための基準
- コスト、工数、スケジュール等の見積りは、プロジェクトの必須要素であり、その正確さはプロジェクトの成否に大きな影響を及ぼす。
  - 最も影響の大きい制約条件

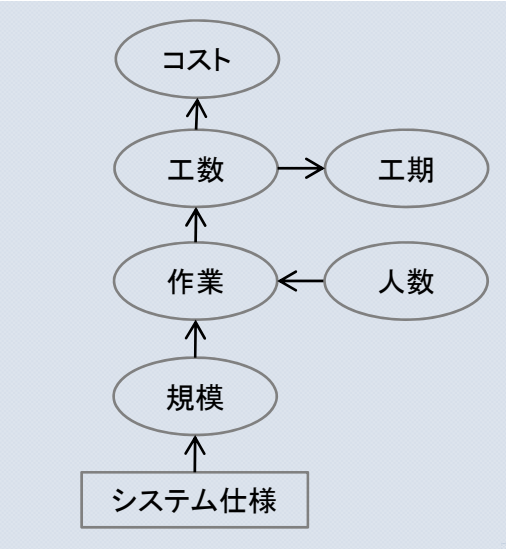
経営における見積りの重要性の拡大

- ITユーザにとっても、今日ソフトウェアなくしてビジネスが成り立たたない。
  - ソフトウェアを期限通り、予算以内に収めて、リリースしビジネスを実現することは、企業の成果、競争力及び評価にとって重要な要素
  - 実現できないとビジネスに大きな損失
- ガバナンスの観点からITに関する説明責任（アカウンタビリティ）が問われている。
- さらに、昨今の厳しい事業環境の中で、コスト削減圧力は強く、妥当なソフトウェアコストが問われている。

見積り対象

見積り対象の種類

- 「コスト」:ソフトウェア開発に必要な費用
- 「工数」:ソフトウェア開発に必要な作業工数
- 「規模」:ソフトウェアの量(開発量)
- 「工期」:ソフトウェア開発の作業期間(例:要求定義～リリース)

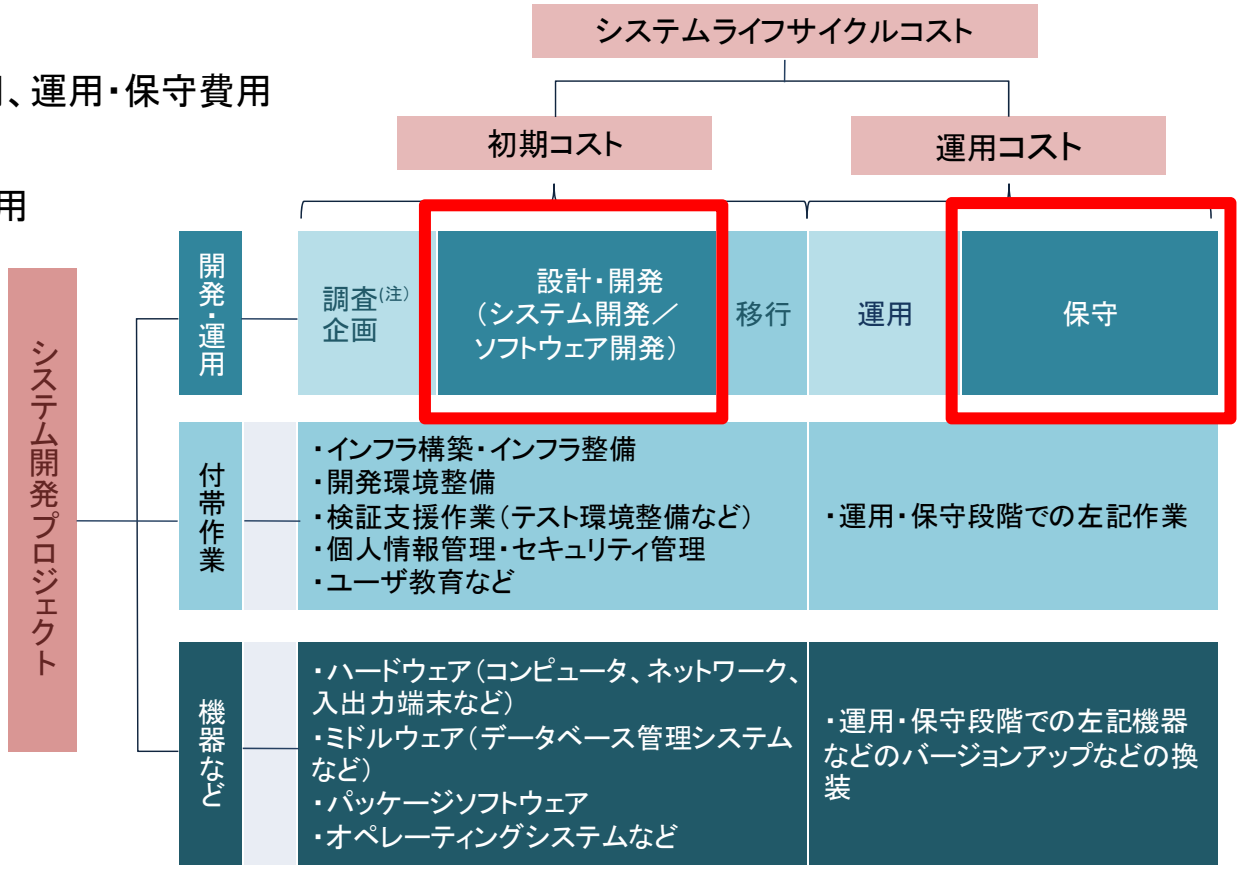


※本日の主題は「工数」の見積りです。

システム開発のコスト構成要素(1/2)

構成要素

- 開発・運用
  - ソフトウェアの開発費用、運用・保守費用
- 付帯作業
  - 間接的に必要となる費用
- 機器等
  - ハードウェア費用
  - ネットワーク費用



(注)システム化の方向性から要件定義

IPA/SEC編、ソフトウェア開発見積りガイドブック、オーム社、2006

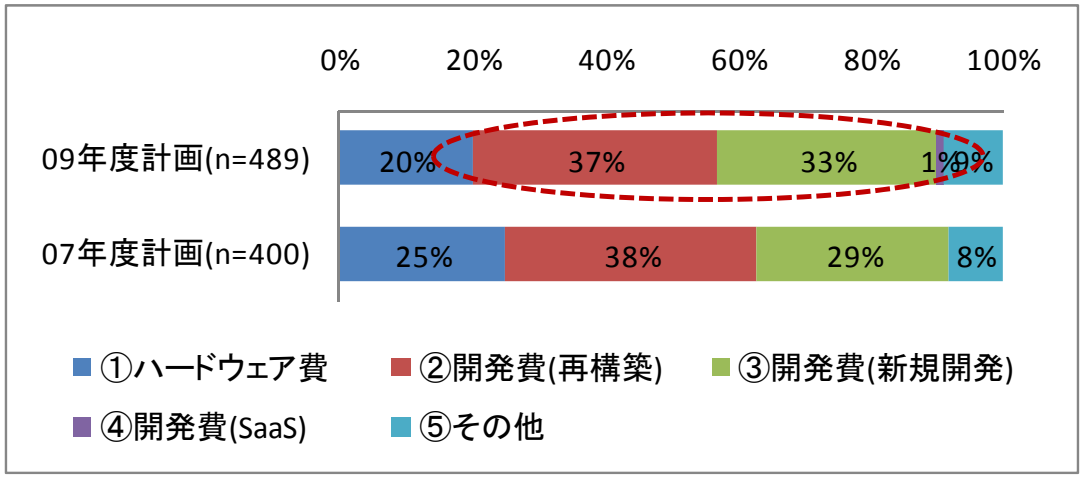
※本日の対象は、「設計・開発」及び「保守」の見積りです。

システム開発のコスト構成要素(2/2)

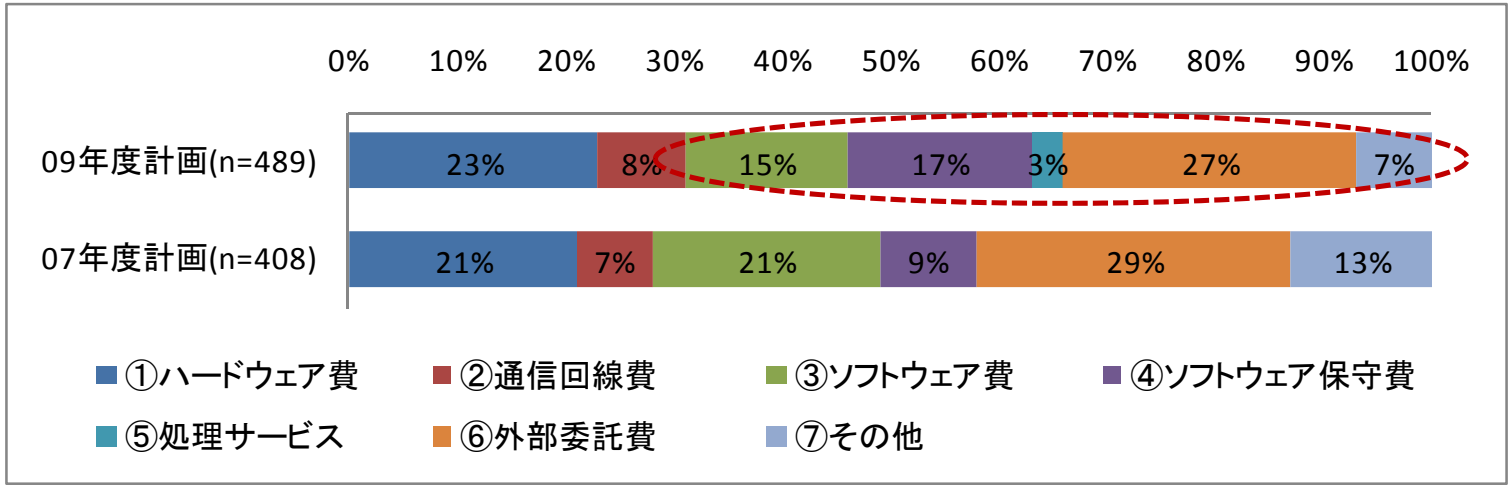
参考データ

- 開発費の内訳
  - 開発費用は8割
- 保守・運用費の内訳
  - ソフトウェアに関わる費用は35%～62%
  - 開発費用は2割弱

【開発費の内訳】



【保守・運用費の内訳】



経済産業省、(社)日本情報システム・ユーザー協会、企業IT動向調査、2010

2．見積りにおける課題

プロジェクトにおける大きな課題の一つ

「プロジェクトを大失敗させる原因は二つある。ひとつは、見積りミスだ」

- 楽観的な見積り：見積り根拠があいまいで、必要な要素をきちんと見積もっていない。
- 早期の見積り：作るものが決まっていない状況で見積りを行う。
- 目標と見積りの混同：営業、マーケティング担当者や顧客の事情で決定される。
- 一度決まったら修正されない見積り
- 時間の余裕がないことが多い（十分に取り組めない）。

⇒ムリなプロジェクト

「もうひとつは、仕様未凍結だ」

⇒仕様の細部が決まらない。仕様が変化し続ける。

※上記の見積りミスにつながる原因の一つ

⇒ムリなプロジェクト

ロバート・グラス、「ソフトウェア開発55の真実と10のウソ」、2005

見積りミスを引き起こす原因

- Capers Jones, “Social and Technical Reasons for Software Project Failures”, Crosstalk, June, 2006

1. 要求が完全に決定される前に、正式な見積りが求められる。
2. 過去の実績データがなく見積りのキャリブレーション（較正）ができない場合が多い。
3. 新しい要求が加えられても、見積りは変更されない。
4. 主要なソフトウェア開発プロジェクトにおいて、見積りツールが使われるわけではない。
5. 保守的な見積りは封じられ、挑戦的な見積りに置きかえられる。

- LM. Liard & M.C. Brenman, “Software Measurement and Estimation: A Practical Approach”, John Wiley & Sons, Inc., 2006

6. 期間と工数に関して、「望み」と「見積り」が混同されている場合
7. 期待に基づいた計画がなされる場合
8. 見積りに関して根拠をきちんと説明できない場合
9. 要求が曖昧な場合、要求が変化し、増加していく場合
10. 成果物の品質が悪い（顧客の満足が得られない）ことが判明した場合

見積りにおける課題(1)

■ 見積り根拠が明確になっていない(規模、工数、工期)

「KKD:Kan(勘)とKeiken(経験)とDokyo(度胸)」による見積り

- 見積りに**必要な情報**がはっきりしていない
  - 規模、工数・コスト、工期を見積もるために根拠の情報がない
- **見積り方法**もその場限り
  - 上記情報を使った規模、工数・コスト、工期の見積り方法が明確でない

■ 生産性に影響を及ぼす要因が多種多様(工数)

- 規模は、基本的に考慮される。

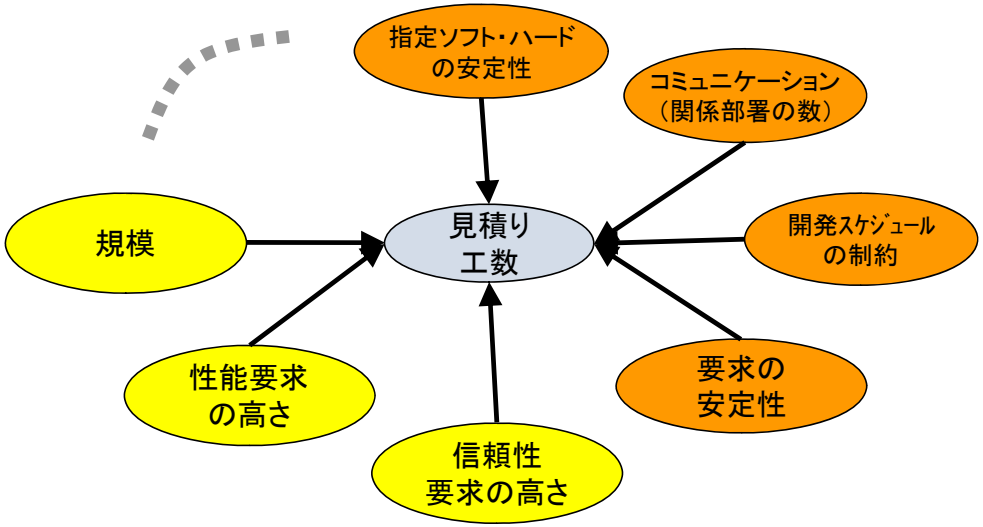
一方で、次のような要因も考慮に入れる必要がある。

- 品質など見えないものの影響
- プロジェクトの状況の影響

開発の中心は人手

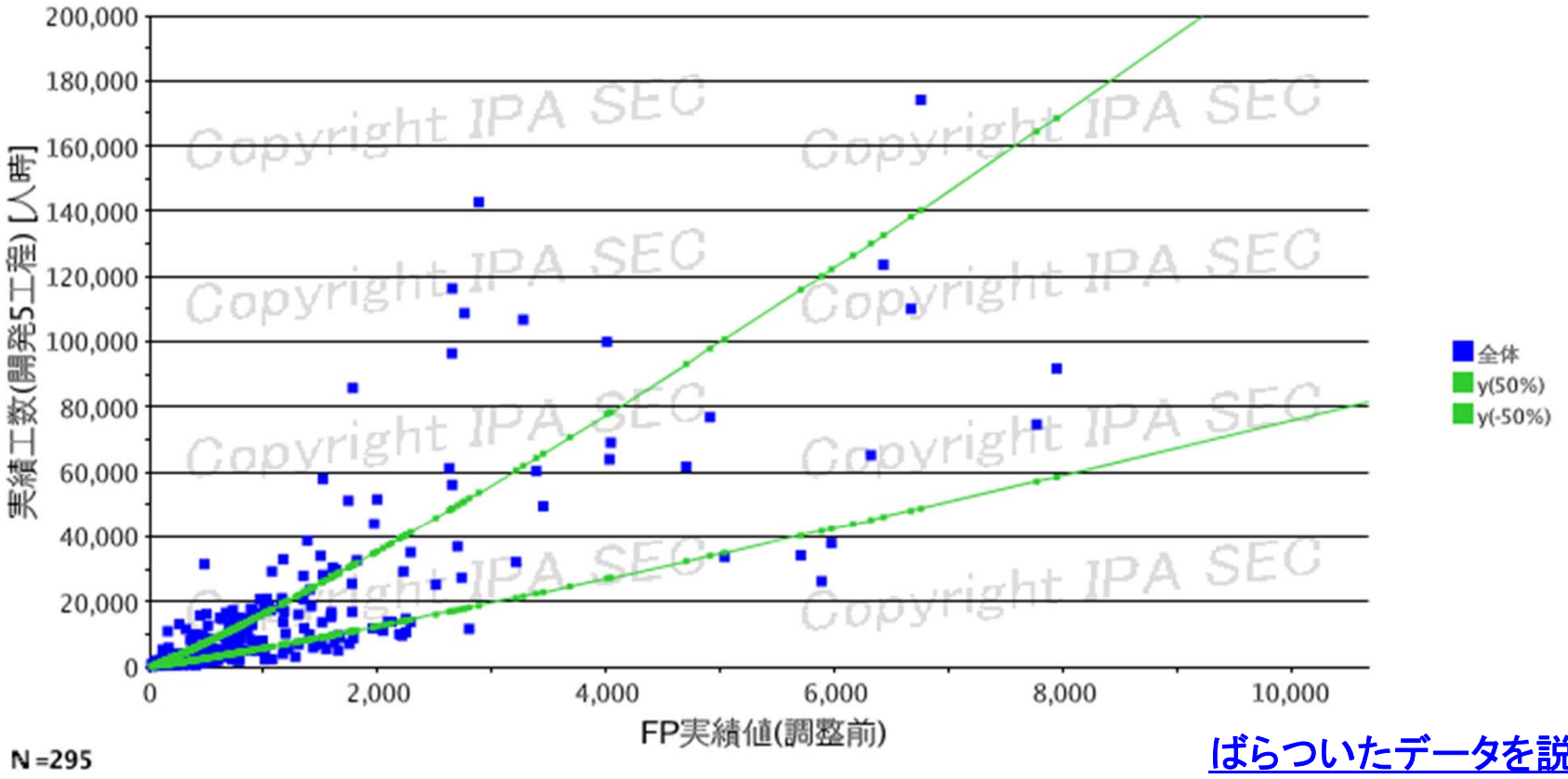
- スケジュールの制約
- コミュニケーション状況
- 要求の安定性
- ・・・・・

影響要因が分かっている場合でも定量化が難しい



実績プロジェクトデータ例：規模と工数の関係

規模（ファンクションポイント）と工数（人時）との関係例



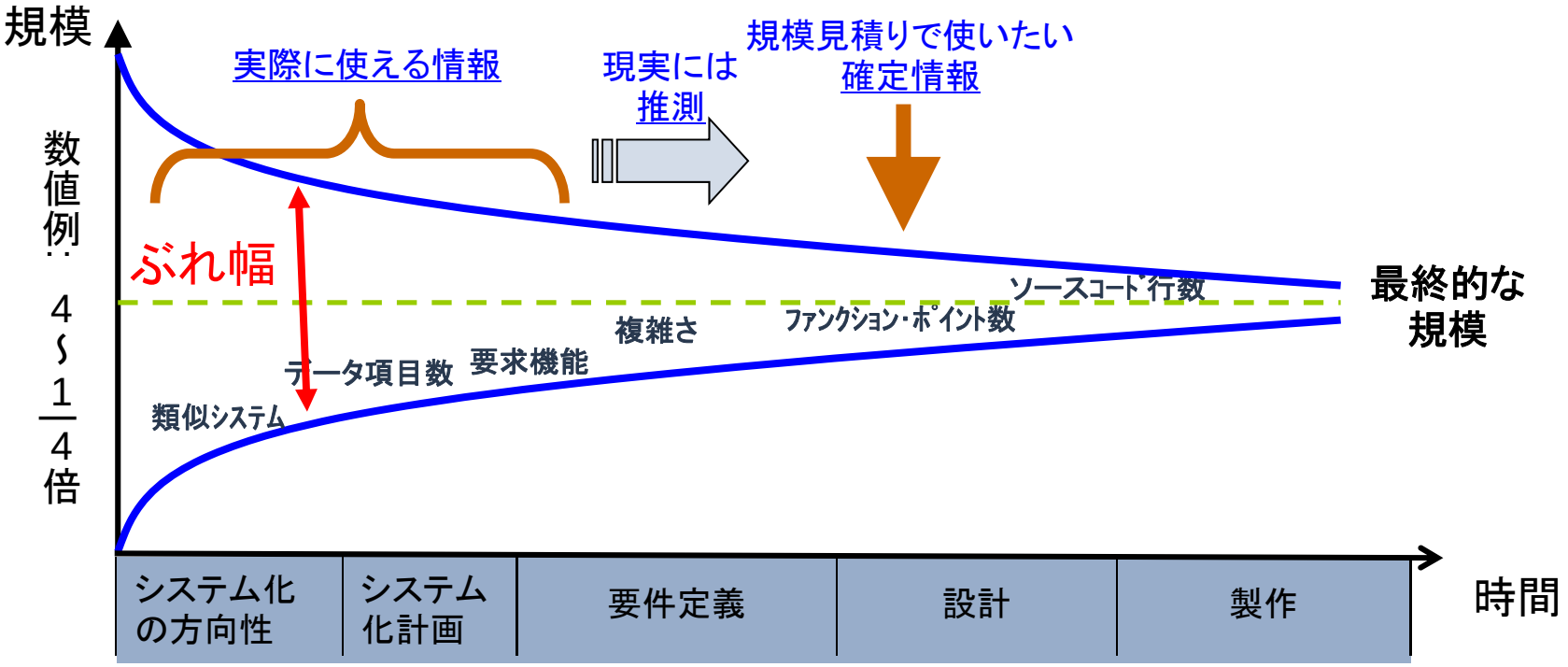
ばらついたデータを説明する関係の発見が必要

IPA/SEC「定量データに基づくプロジェクト診断支援ツール」（2007年12月より）
https://sec.ipa.go.jp/project_assessment/TopMenu.do （無料利用者登録必要）

見積りにおける課題(2)

部分的な情報からの見積り(規模)

■ 細部が決まっていない状況で、見積りを行う場合が多い
(予算決めなど早い段階)
⇒ 部分的な情報から、推測するしかない。
⇒ 実際とぶれがでることは避けられない



B. Boehm, Software Engineering Economics,1984
IPA/SEC編、ソフトウェア開発見積りガイドブック、オーム社、2006

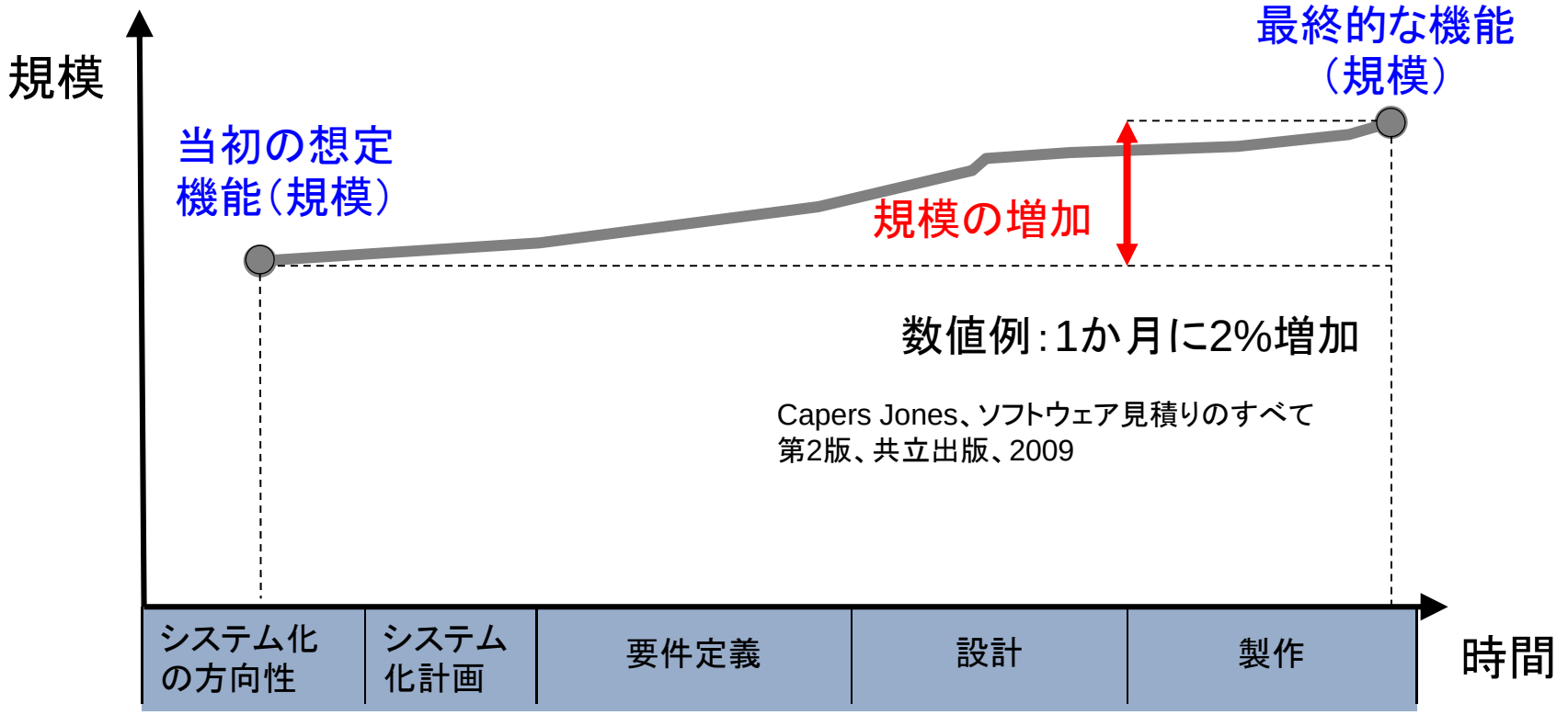
見積りにおける課題(3)

決めたことが変わっていく(**規模**)

■ 開発当初の想定機能は、一般に工程が進むにしたがって膨張する。

⇒見えていなかった要件が見えてくる。

⇒最初の見積りと当然ぶれがでる



IPA/SEC編、ソフトウェア開発見積りガイドブック、オーム社、2006に基づき改変

見積りにおける課題(4)

- ソフトウェアの見積りを難しくしている原因(規模、工数、工期)
  - 過去データの**欠落**
    - 工数データの把握が難しい。
  - 蓄積データの**精度**の確保
    - 工数データの精度の確保が難しい。
    - 規模データの一貫性の欠落(単位、数え方の一貫性)

- 見積り実施者の傾向(規模、工数、工期)
  - 「ほとんどのコスト見積りは**低すぎる傾向**がある」(DeMarco-Glassの法則)
    - 不必要な作業を追加するよりは、作業項目を忘れる。
    - ソフトウェアの機能も同様に忘れられているものの方が多い。
    - データ等の根拠がない場合は「楽観的に」なりがちである。
    - 不慮の事態が起こりがち:安定した開発はない
    - 「早期の見積り」「時間をかけず、熟慮しないで実施」
  - **「目標」を「見積り」**としてしまう。
    - 現実には混同しがちだが、全く違うものを同一視している。
      - 例:入札額と見積り額は本来違うもの

「法則」の出典(以下、同じ):
A. Endres, 、D. Rombach, A Handbook of Software and Systems Engineering 1/E, Pearson Education, 2003,(邦訳:吉舗紀子訳、ソフトウェア工学・システム工学ハンドブック、コンピュータエイジ社、2005)

3．見積り手法の歴史と蓄積

見積り手法に関する大きな流れ

1960年代	1970年代	1980年代	1990年代	2000年代
ソフトウェア開発（工学）の黎明期	見積りモデル提案の時代	見積りモデルの成熟期	ツールの成熟 機械学習の応用	熟練者の知識活用の再認識
•マネジメントの一環として、見積りが既に課題	•様々な関係式 •規模に対する注目（ファンクション・ポイント） •重要文献（基本指針）の出現	•パラメトリック手法 •COCOMO •さまざまな見積りツール •一方、キャリブレーション（較正）の必要性の認識	•多数のデータから予測（多様な機械学習アルゴリズムの適用） •プロジェクトマネジメントの体系化（WBS）	•1990年代後半からデータと熟練者の知識の融合

コンピュータ黎明⇒大型計算機（汎用機）⇒オープン化⇒さらなるオープン化

ハードウェアコストの一部　⇒
ソフトウェア比率とコストの増大　⇒
残り続ける「勘」「経験」「度胸」（KKD)の習慣
見積り「実践」は30%以内（1992年、2005年）

ソフトウェアにおける見積りに関する歴史と蓄積(1/5)

- 最初の研究は、1950年代から60年代
  - 最初の論文：V. Norden, 1958

    “Curve Fitting for a Model of Applied Research and Development Scheduling”
  - Nelson, E.A., “Management Handbook for the Estimation of Computer Programming Cost”, TM-3224, SDC, 1966
    - 169のプログラミング事例を統計的に分析
    - 6つの工程に分類
      - 「プログラム設計、コーディング及びテスト」について、工数計算式を定義
      - 「計画及びコスト見積り」、「分析・設計」、「統合テスト」、「インストール」、「保守」については、基本的に過去のプログラミング経験に基づく類推
    - Nelson-Jonesの法則：「多くの要素が開発生産性に影響を及ぼす（増加要因）」
      - 工程ごとに要因の設定
      - 要求関連
        - 設計要求定義のあいまいさ、運用要求に関する知識の欠如、システムにおける機能数
      - 設計・コーディング関連
        - 命令数の数、サブプログラムの数、内部や外部向けの文書の数、プログラムの種類
      - その他、データ処理関連、開発環境の要因

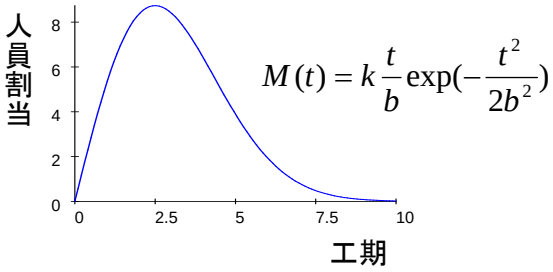
ソフトウェアにおける見積りに関する歴史と蓄積(2/5)

- 1970年代は、見積りモデルの時代
  - 多様なモデルの提案
    - コスト要因を含む算出方式
    - F.J. Heemstra, Software cost estimation, 1992には1970年代のモデル（ツール）が10リストアップ。
  - 例： 基本式：工数 = n × (開発時間)^3

    変動要因例：インターフェイスの複雑さが不安定であれば、29%増加

    C.E. Walston, C.P. Felix. A Method of Programming Measurement and Estimation" IBM Systems Journal, vol. 16, No. 1, 1977
  - 例：SLIMモデル（L.H.Putnam）

$$Size = C_k \times Effort^{1/3} \times T_p^{4/3}$$



    - T_pは人員がピーク時の時間
    - C_kは定数（技術的な適用度合い応じて変わる）
    - 工数の分布の前提として、「レイリー分布」
    - 「工数」と「規模」と「時間」のトレードオフを表現
  - 例：規模指標として、ファンクション・ポイントの萌芽

    A.J. Albrecht, Measuring Application Development Productivity, Proceedings of the Joint SHARE, GUIDE, and IBM Application Developments Symposium, 1979

ソフトウェアにおける見積りに関する歴史と蓄積(3/5)

- 1975年、「人月の神話」
  - “A Mythical Man-Month”, Brooks, 1975
  - 見積りの知見・課題に関する最初の体系的な文書
    - マネージャは基本的に「そうありたい」工数を予測する
    - 「遅れているソフトウェアプロジェクトへの要員追加はさらに遅らせるだけだ」
      - 人月の非交換性
        - 例：10人で10か月かかる仕事（100人月）を100人で1か月で実現することはできない

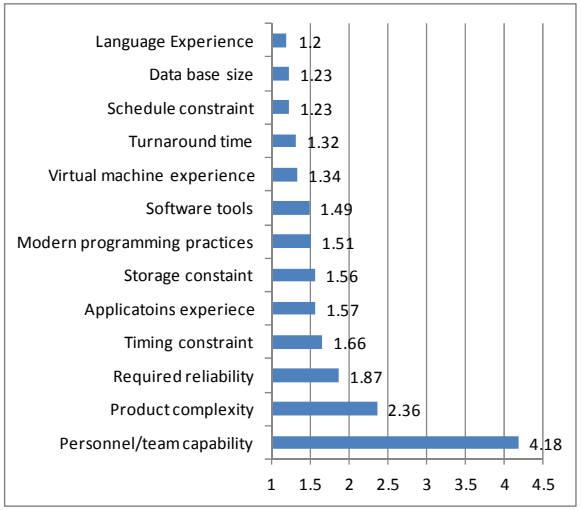
- 1970年代のモデルにおける課題
  - ソフトウェア開発に影響を及ぼす変動要因は多数あり、何を選択すればよいのか。
  - モデルによって、重視する要素に違い
    - プロジェクトサイズ
    - コスト変動要因
    - 熟練者の判断

  まだ、組合せを考えるまでには至っていなかった

ソフトウェアにおける見積りに関する歴史と蓄積(4/5)

- 1981年：COCOMOモデル：見積モデルに関する体系的なモデル
  - 南カリフォルニア大学Boehm（当時はTRW）による開発

$$PM = 2.94 \times Size^{E} \times \prod_{i=1}^{15} EM_i$$



  - 工数の説明変数として、規模、コスト変動要因がある。
  - 規模と工数の関係は、非線形の指数関数である。
  - 類似モデルとして、COPMO(Conteら, 1986)

$$TotalCost = A \times Size + B \times Staff^{1.5}$$

    - なお、定数AとBは、COCOMOから導き出すようになっていた。
    - BYL（Before You Leap）はファンクションポイントとCOCOMOをつなぐ商用ツール

- 1980年代（1980年前後を含む）はパラメトリック手法の提示
  - 様々な規模や環境のデータを用いて、パラメトリックなモデルの構築
    - 代表的なものはPutnam, 1978; Boehm, 1981; Bailey and Basili, 1981
      Basiliのメタモデル： $E = MR \times (a \times Size^{b} + c)$
      a, b, cは定数、MRは変動の係数
    - F.J. Heemstra, Software cost estimation, 1992には、1980年代のモデル及びツールとして、上記3つを含めて、16個を挙げている
  - 一方、主要な結論として、ある環境で構築されたモデルはそのままでは精度が良くない。
    ⇒キャリブレーション（較正）を行う必要がある。(Kitchenham & Taylor, 1985; Kemerer, 1987)

変動要因（歴史的な例）

- 「ソフトウェアエンジニアリング序説」, ロジャー・プレスマン, TBS出版会, 1983
  - 要因の分類（Basiliらに基づく）
    - 人的要因：開発組織の規模と専門的知識
    - 問題の要因：解決すべき問題の複雑さと、設計上の制約とか要求の変更回数
    - 過程の要因：使用する分析及び設計手法、使用可能な言語、レビュー手順
    - 製品の要因：コンピュータ利用システムの信頼性と性能
    - 資源の要因：利用可能な開発ツール、ハードウェア、ソフトウェア資源

  - 生産性の影響要因例と影響度合い（Waltson, Felix, 1977に基づく。抜粋）

要因	平均生産性（行数/人月）		
顧客インタフェースの複雑さ	<通常 500	通常 295	>通常 124
要求定義におけるユーザの参加	なし 491	若干 267	多い 205
顧客によって生じたプログラム設計の変更	少ない 297		多い 196
プロジェクトのアプリケーション領域に関するユーザの経験	なし 318	若干 340	多い 206

ソフトウェアにおける見積りに関する歴史と蓄積(5/5)

1990年代における機械学習方式の適用

- ソフトウェア開発は複雑で動的なプロセスであり、生産性における変化の因果関係を説明するには知識が少ないとの認識の下、データに語らせる方式が多数取られた。
  - 機械学習アルゴリズムの適用(Brianら, 1992)
  - ニューラルネットワークの適用(Jørgensen, 1995; Finnieら, 1997)
  - CART Regression treeの適用(Srinivasan & Fisher, 1995; Kitchenham, 1998)
  - アナロジー方式の適用 (Mukhopadyayら, 1992; Shepperd & Schofield, 1997; Walkerden & Jeffery, 1999)
- 基本的に、パターンに分類して見積りを行うもの

1990年代後半から2000年代：熟練者の判断活用に対する再認識

- 熟練者は、自らの経験と知見により、見積り方法を「暗黙的に」確立

⇒熟練者の見積りの効率化・高精度化

⇒熟練者の知見を活用したモデルの構築

  - 主観的な見積り (Höst & Wohlin,1998; Stensrud & Myrtveit, 1998)
  - 熟練者の知識による定量化（CoBRA） (Briandら. 1998a)
  - 熟練者の意見とデータの融合(Chulaniら, 1999)

見積りツールの歴史

1960年代 最初の見積りツールの開発.

1973 Frank Freiman による PRICE-Sモデル、最初の商用見積りツール

1973 Capers Jones と Dr.Charles Turk による IBMの知財としての automated estimation tool.

1973 Allan Albrecht ファンクションポイント開発（IBM内）

1979 IBMがファンクションポイントをパブリックドメイン化

1979 Larry Putnamによる Software Life-Cycle Management (SLIM) ツールの開発

1981 Dr.Barry Boehm による COCOMO

1983 Dr.Howard Rubin による ESTIMACSモデル

1985 Capers Jonesによる SPQR/20 estimation toolの開発

1986 International Function Point Users Group(IFPUG) の世界的に広まり

1986 Conteらによる COPMO の開発

1986 Gordon Groupによる BYL（Before You Leap）の開発

1987 BIS Applies Systemによる BIS/ESTIMATORの開発

1988 Galorath社による SEER-SEM toolの開発

1997 Dr.Barry Boehm COCOMO Ⅱ開発書籍（ツール）は、2000年

1980半ばから**2000** にかけソフトウェア見積りツール市場の急速な拡大（米国中心）

2002 米国では約50の商用ツール、ヨーロッパでは約25のツール

主にCapers Jones, Software Cost estimation in 2002, Crosstalk, June, 2002に基づきMRI作成

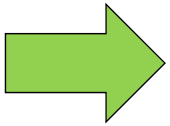
見積り手法と関連技術の動向

	1950年代	1960年代	1970年代	1980年代	1990年代	2000年代
規模指標	※SLOC（ソースコード行） 命令数(Instructions)		★'76 Cyclomatic複雑度 ★'77 Software Science(HALSTEAD) ★'79 FP	★'89 MKII FP ★'87 ﾌｨｰﾁｬｰﾎﾟｲﾝﾄ	★'91 オブジェクトポイント ★'93 Use Caseポイント	★'00 COSMIC-FFP ★'05 NESMA(FP)
工数見積り	★'58 最初の論文 NELSON ★'63		★'73 PRICE-S ★'78 SLIM ★'81 COCOMO ★'86 COMPO	★'88 SEER-SEM	★90年代 機械学習アルゴリズム ★'97 CoBRA法 ★90年代後半から熟練者知識活用	★'97 COCOMOⅡ

見積りの課題に対する方針

- 対策①：見積り手法の確立
  - 前提（インプット）とアウトプットの明確化
  - アウトプットの算出根拠の明確化と再現性の確保

  ⇒見積りモデルの確立



すべての対策の出発点

本日の主題

- 対策②：見積りの妥当性の確保
  - 複数の視点からのチェック
  - 特に、見積りの前提が曖昧な場合は様々な立場・可能性からのチェック／シミュレーション

  ⇒複数の見積り方法によるレビュー

  ⇒対策(1)の実現による前提を変えたシミュレーションはその一つ

- 対策③：モニタリングとコントロールと再見積り
  - 曖昧な状況であっても、前提とインプットを仮説として置き、見積り
  - 明確になるとともに、前提とインプットを見直して、再見積り
  - また、変化する条件（要求内容）に対して、前提とインプット見直しによる再見積りとコントロール
  - 同時に、前提とインプットが成立するように、マネジメントによるコントロール

  ⇒前提条件とインプットの変化のモニタリング

  ⇒コントロールによるブレ防止と再見積りによる変化への追随

  ⇒対策(1)の実現により可能となる

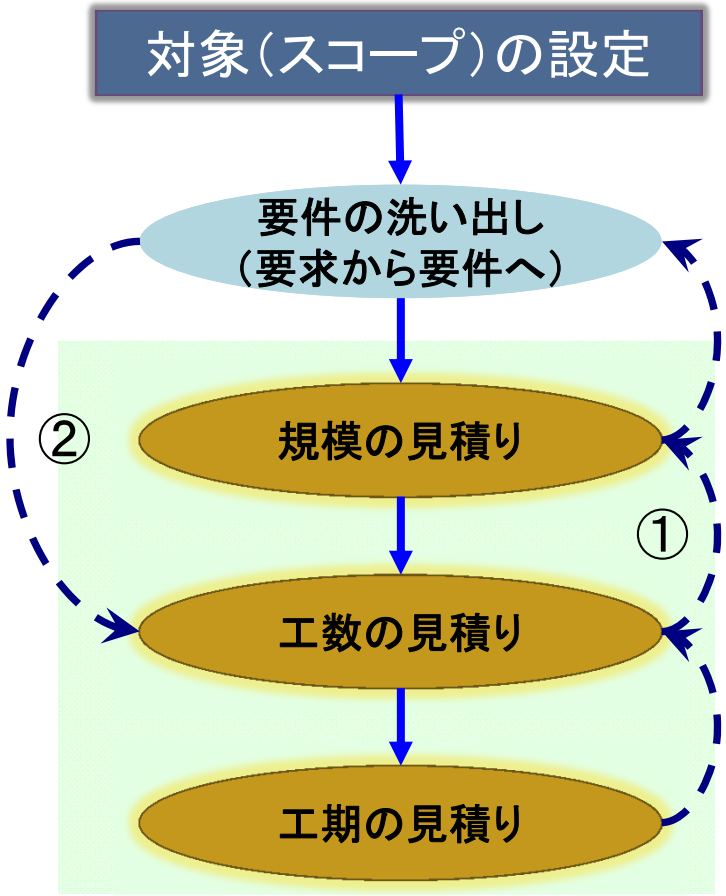
- 対策④：目標と見積りの峻別
  - 見積りは前提条件とインプットに基づいて算出
  - 目標と見積りの差異は、見積り（手法）で解決するのではなく、マネジメントで解決

  ⇒見積りプロセスの確立

  ⇒対策(1)の実現により可能となる

4. ソフトウェア開発見積りにおける基本的なアプローチ

ソフトウェア開発見積りフローの構成と流れ



見積もる対象のベースラインの設定

ソース行数や機能数、ファイル数などで、ソフトウェアの規模を算出

規模を元に、プロジェクトの特性を加味した生産性で工数を算出

工数に基づき工期を算出

見積り前提

見積り活動

基本的なフロー

摺合わせや制約時のフロー

①工期、工数が条件の場合
②類推法、標準工数積上げ

規模見積り

見積りにおける主要要素

- 精度に大きく影響
- 工数と相関の高い「指標」を活用

規模指標（様々なアプローチ）

- 単位
  - ソースコード行数
  - ファンクションポイント
  - オブジェクトポイント
  - ユースケースポイント
- スコープ
  - 新規量、変更量、削除量、母体規模
  - 規模指標の設定
    - 規模そのものではなく、モデルの「説明変数」

規模指標
$= 新規量 + \alpha \times 変更量 + \beta \times 削除量 + \gamma \times 母体規模$
α, β, γは、実績から求められる係数

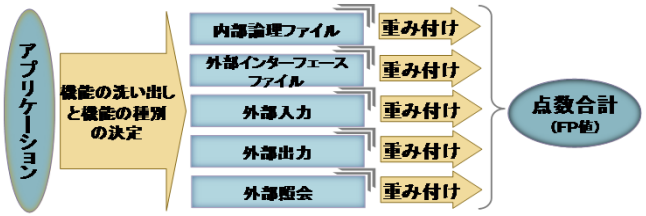
規模見積りの方法

- （基本）作成されるソフトウェアに対し見積り
  - 要求内容（RFP、要求仕様書、設計書・・・）

要件定義段階より前は、仮定を設定し、見積もり

- 見積方法
  - 類推方法
    - 熟練者の意見
    - 過去類似プロジェクトのデータ
  - パラメトリックな計算
    - 機能を細かい粒度のブロックに分割し、1ブロックあたりの想定規模を乗ずる
    - ファンクションポイント分析の利用
      - インプット、アウトプット、参照、インターフェイスの数え上げ

ファンクションポイントのカウント方法



見積りモデルの基本的な考え方

ベースラインの設定

- 規模と工数（又はコスト）の基本的な関係
  - 工数＝α×規模 （正比例）
  - 工数＝α×規模$^{\beta}$（累乗）
- 工数と工期の基本的な関係
  - 工期は工数の3乗根に比例など

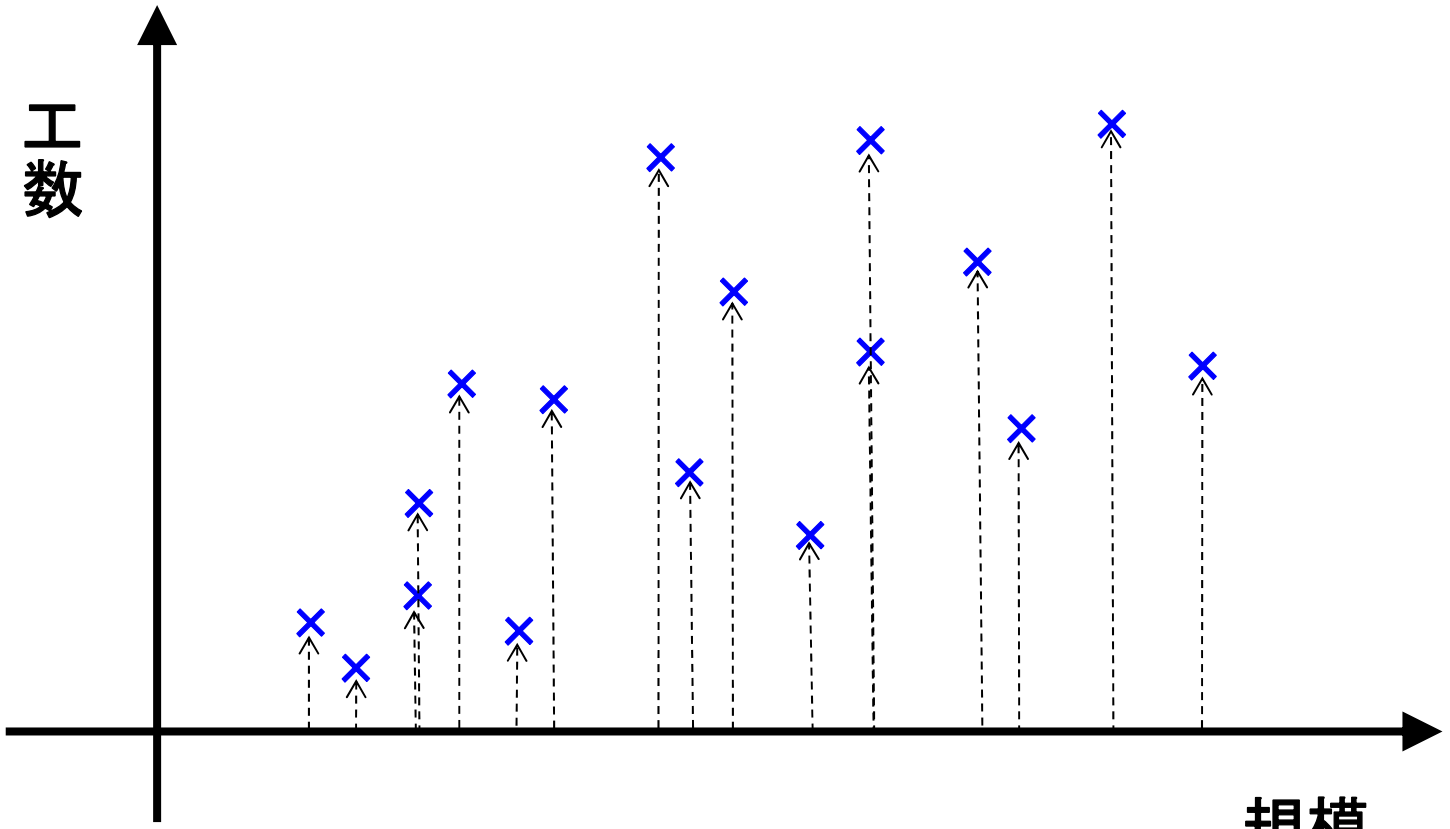
関係式例（COCOMO）

$$\text{コスト} = \alpha \times (\text{規模})^{B} \times \prod_{i=1}^{17} EM_i$$

変動要因の設定

- 仕様の不確実さから規模の不確実さ
- 生産性のベースラインからの「ずれる」要因の設定
  - プロジェクト要因（関係者の数、開発スケジュールの制約など）
  - プロセス要因（要求管理の統制度合い、構成管理の度合いなど）
  - プロダクト要因（高い品質要求など）
  - 人的要因（顧客側の業務知識・経験、関係者の協力度合いなど）

ソフトウェア開発の見積りにおいて説明すべきもの



規模がほぼ同じでも、必要な
工数に違いがある。

この違いを把握し、説明する

ソフトウェア開発見積りの基本的な考え方

ご静聴ありがとうございました。

Email :cobra_info@mri.co.jp