ソフトウェア開発プロジェクトの見積もり

初田　賢司

IT 部門以外の方が読まれることを意識して書いてみた．ソフトウェアの開発費について，「なんでこんなに高いの？」とか，あるいは「どうせどんぶり勘定」という思いをもたれている方も多いのではないだろうか．こうした疑問は少しでも解消していきたい．IT の世界もソフトウェア工学やプロジェクトマネジメントの知見を得て改善しつつある．本稿では，こうしたことも踏まえてソフトウェアの見積もりの全体像をできる限り理解いただけるよう紹介したい．

キーワード：ソフトウェア，見積もり，ファンクションポイント法，プロジェクトマネジメント

1. はじめに

デパートや量販店，コンビニで店頭に並ぶ衣料品や家電，食料品などの商品は，価格が設定されていて購入時に見積もりを要求することはない．同様にソフトウェアでもワープロなどオフィス系のパッケージや財務会計のパッケージなどは，店頭やオンラインショップで販売されていて価格が設定されている．

一方，生産管理や販売管理，金融機関の勘定系のソフトウェアなど業務で利用するアプリケーション・ソフトウェア（以下，業務アプリケーション）の多くは，購入時に見積もりが必要となる．

一口でソフトウェアといっても多岐にわたる．二つの例の中間に位置するものも多い．業務機能の共通部分はパッケージ化されているが組織固有の部分の開発を見積もりベースで対応するもの，組み込みソフトウェアや自組織内で利用するソフトウェアのように価格の設定までは必要ないが，コストの見積もりが必要なものなどさまざまである．

購入時に見積もりが不要なパッケージ系のソフトウェアと必要な業務アプリケーションでは，何が異なるのだろうか．パッケージ系のソフトウェアは，以下の条件を満たしていることがわかる．

①機能の共通化や標準化がしやすい
②同じ機能を提供する商品の選択肢が多い
③量販が可能で開発元が利益を想定しやすい

逆に見積もりが必要な業務アプリケーションを考えてみよう．生産管理システムを例にとると，共通化できる部分も多いが，鉄鋼業のような装置産業と自動車産業のようなアセンブリー産業では必要とする機能も大きく異なる．同じ業種であっても，A 社と B 社では異なる．製品構成や生産ライン，設備，業務の実現方式，企業が置かれているポジション，トップの考え方，企業文化などが異なるからだ．このように業務アプリケーションは，企業や組織の中でしか位置づけが決まらない．基本的に一品生産となる．また，同じような業務に見えても，利用するデータや用語，組織などの違い，あるいは要求される信頼性や性能，セキュリティなどの要件や制約条件などが異なるので，ソフトウェアの実現方式も異なってくる．

つまり，業務アプリケーションは，次のような条件となるため，案件ごとに見積もりが必要となる．

①基本的に一品生産である
②同じ機能のように見えても異なる
③可用性や性能などの非機能要件や技術要件により価格やコストが変わる

本稿では，見積もりを必要とするソフトウェアの代表である業務アプリケーションを対象に考察する．

2. 見積もりを提示するタイミング

図 1 に業務アプリケーションの開発の流れと見積もるタイミングを示した [1, 2]．まず開発の流れを説明する．発注側企業において経営戦略や中長期計画を策定する中で業務アプリケーションの開発が起案されると予算化し，予算確保ができた案件の開発を行う．

図 1 の右側に見積もりを実施するタイミングを示した．IT 業界では，このタイミングを表す用語が定着していないため，本稿では，「試算見積もり」，「概算見積もり」，「詳細見積もり」を定義する．

はつだ　けんじ
株式会社日立製作所 情報・通信システム社
プロジェクトマネジメント統括推進本部
〒 140–8572 東京都品川区南大井 6–27–18 日立大森第二別館

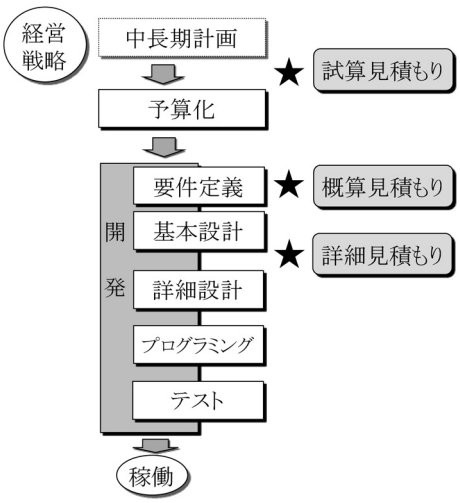

図 1　IT プロジェクトの見積もりのタイミング

(1) 試算見積もり

業務アプリケーションの開発予算を確保する見積もりである．精度は極めて低く参考値である．主に過去の事例や経験を基にした類推見積もりを用いる．

(2) 概算見積もり

発注条件が記載された「提案依頼書」に基づく見積もりである．ベンダーは，提案活動の一環で見積もりを行う．この見積もりの評価により，発注するベンダーや発注金額が決まる．主に数式などを利用した係数モデル見積もりを用いる．

(3) 詳細見積もり

基本設計で明確になった仕様や作業項目を基に見積もる．概算見積もりとの差異の整理や詳細設計以降を分割して発注する場合などに実施する．ボトムアップ（積み上げ）見積もり，もしくは詳細な係数モデル見積もりを用いる．

IT プロジェクトは，建設プロジェクトでいう「エンジニアリング（設計）」で完結するプロジェクトと考えると理解しやすい．建設プロジェクトであれば，技術的な隘路を解消したところでエンジニアリングは完了し，その後，期間やコストの大半を占める調達やコンストラクションに入っていく．しかし，ソフトウェア開発プロジェクトは，テスト完了で技術的な隘路が解消すると同時にモノとして完成し，その後のコンストラクションに相当する作業は，無視できるほど小さい．建設プロジェクトでいえばエンジニアリングに相当する範囲で見積もりを行うタイミングが 3 回ある．

本稿では，受注や金額が決まる概算見積もりに焦点を当て見積もり技術を解説する．ベンダーにとっては，最も大事な見積もりであり，精度を高めるアプローチが求められる．

3.　概算見積もり

3.1　概算見積もりの目的

ベンダーが概算見積もりを行う最大の目的は，注文を取ることである．では，注文を取るために安い金額を提示すればよいだろうか？　もちろん，No である．適正な価格で合意するために，見積もりを論理的，合理的に説明することが必要になる．これには，エンジニアリング的なアプローチが有効である．見積もりの根拠を定量的に示し，統計的な裏づけや数式を用いてお互いが納得することが重要である．

さらに IT プロジェクトでは，もう一つ大事な目的が加わる．見積もり時にプロジェクトを成功に導くための仕掛けを作ることである．見積もりミスは，IT プロジェクトが混乱する要因の上位にあがる．見積もりミスとは，見積もり時に約束が曖昧になっていた，あるいは漏れていたことがトリガーとなりプロジェクトが混乱することを指す．意図的に状況を把握する仕組みを作らないと危険な兆候を察知することは難しい．気づいたときには規模が計画の 2 倍以上に膨れ上がっていることも珍しくない．IT プロジェクトでは，工程を追うごとに確認できるアウトプットの形が変わる．設計段階では，抽象的なモデルや自然言語で書かれた文章で記述されるが，プログラミングの段階で Java や COBOL などのコンピュータ言語に姿を変え，動作が確認できるのはテストに入ってからである．設計段階でまずいことが起こっていることを察知するには，見積もり時にコントロールしやすい仕掛けを作っておくことが必要である．

このように概算見積もりでは，「注文を取る」というビジネスの視点，「説明できる」というエンジニアリングの視点，「プロジェクトを成功に導く」というプロジェクトマネジメントの視点が大事である．

3.2　概算見積もりの難しさ

形がなく論理の集合というソフトウェアの特性と IT 業界の歴史の浅さにも起因するのだが，見積もりを難しくする要因があまりにも多い．まず挙げられるのが，見積もり段階で要件がよく定義できていないプロジェクトが多いことである．特に近年のプロジェクトでは，ほとんどの業務がすでにシステム化されており，新たに開発する業務には何らかの戦略的な要素が加わる．こうした案件は要求をとらえにくく難易度が高い．

また，厳しい納期やリソース，多様なステークホル

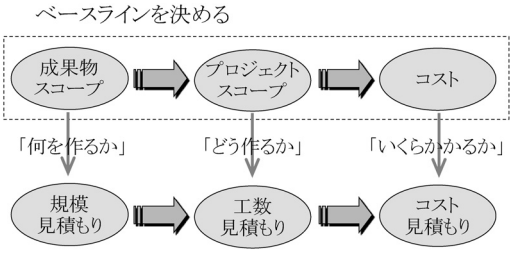

図 2　ベースラインの設定と見積もりの流れ

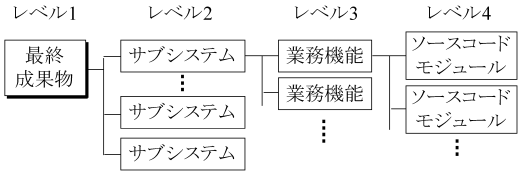

図 3　成果物スコープの定義

ダーや開発方法など不確実な要素も多い．技術の進歩も激しく，毎回新しい要素が入るし開発環境も変わる．加えて，開発費の大部分は人件費で，ヒューマンファクターの影響を受けやすい．

さらに提案依頼書の提示から提案書提出までの期間が短いことも見積もりを難しくする要因となっている．通常，民間企業で 1 週間〜1 カ月程度である．

4.　概算見積もりの手順

前節の目的と難しさを踏まえて，どのように見積もればよいかを考える．ここで役に立つのがモダン PM（プロジェクトマネジメント）の考え方である．モダン PM におけるチェンジ・コントロールは，まずベースラインを設定し，それからどう変わったかを判断する．ベースラインとは，ステークホルダー間で合意された基準線である．図 2 にベースライン設定の流れとそれに対応する見積もり作業の流れを示した．スコープとは「範囲」のことで，成果物スコープは，最終成果物やサービスに備える機能，すなわち「何を作るか」を定義する．プロジェクト・スコープは，成果物を提供するためになすべき作業，すなわち成果物を「どう作るか」を定義する [3].

他の分野のプロジェクトではありえないと思うが，IT プロジェクトの成果物スコープは，プロジェクトの最中で頻繁に変わる．成果物スコープ → プロジェクト・スコープ → コストの順にベースラインを定め，この関係性を保つことが重要になる．これが保たれていれば，仕様追加により成果物スコープのベースラインが変わった場合，それに伴う作業追加によりプロジェクト・スコープのベースラインが変わり，コストやスケジュールが見直しになることを説明しやすい．

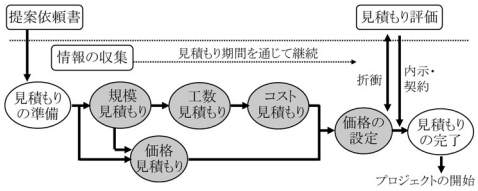

図 4　概算見積もりの流れ

成果物スコープのベースラインとして機能するには，図 3 のレベル 4「ソースコード・モジュール」まで分解されている必要があるが，ここまで分解できるのは基本設計終了後である．概算見積もりの段階では，「業務機能」のレベルで漏れや思い違いを含んでいる可能性がある．このためプロジェクト開始後，想定していた仕様か，仕様追加かを巡って発注側とベンダーの見解が分かれることも多い．こうした曖昧さを回避するために成果物スコープを定量化し，最終成果物の規模として見積もる．図 2 に戻る．次のプロジェクト・スコープの「どう作るか」には工数見積もりが対応する．この段階で最終成果物を作るために必要な作業を詳細に洗い出すことは難しい [4]．見積もった規模を基に計算式を用いて工数を算出する．工数が決まればコストを見積もる．コストの大部分は人件費である．

図 4 に概算見積もりの流れを示す．骨格になるのは，「規模見積もり」「工数見積もり」「コスト見積もり」という「いくらで作れるか」を見積もるアクティビティと「いくらなら売れるか」を見積もる「価格見積もり」「価格の設定」である．これに見積もり作業の開始と終了にあたる「見積もりの準備」と「見積もりの完了」，見積もり期間を通じて実施する「情報の収集」を加えて全体の流れとした．見積もりの準備では，提案依頼書やプロフィールなど入手できた情報を整理し，見積もるための情報がどの程度揃っているかを把握する．得られた情報を基に，利用する見積もり技法や記録の残し方など，見積もりの方針を決める．

以降，骨格となるアクティビティについて解説する．

5.　概算見積もりの主なアクティビティ

5.1　規模見積もり

概算見積もりの段階では，「何を作るか」が曖昧な状態であることが多いため，規模見積もりを行い，成果物スコープを数値化する．数値化はメトリクスの選定から入る．ソフトウェアの規模を表す多くのメトリク

スが提案されているが，ある程度普及していることが必要である．また，1 単位の細かさも重要だ．普及度と粗さを考慮すると SLOC (Software Lines Of Code) と FP（ファンクションポイント）法が候補となる．

(1) SLOC

プログラムのソースコードの行数である．最も普及している，完成時点で物理的に計測可能などメリットは大きいが，以下のような問題点がある．

・ビジュアル・プログラミングなど SLOC の適用が難しいプロジェクトが増えている
・開発言語に依存する（Java と COBOL など異なる言語の行数は単純に比較できない）
・開発者のスキルにより増減する
・計測ルールは企業ごとに異なる　など

(2) FP 法

FP 法は，1970 年代後半に A. J. Albrecht が提唱したソフトウェアが提供する機能を定量化する手法である．ソフトウェアの規模に相当する量を入出力とデータの集合によって計測する．計測方法にはさまざまなバリエーションがあるが，業務アプリケーションの分野では，IFPUG (International Function Point Users Group) 法がデファクト・スタンダードとなっている．IFPUG 法は，Albrecht の考え方を継承して発展させた FP 法で，計測ルールを規定するマニュアルが提供されている．わが国では，日本ファンクションポイントユーザ会が普及推進を図っている．FP 法は，「機能規模測定」として ISO 化されている [5].

IFPUG 法による計測は，以下の手順で行う [6].

①ソフトウェアのファンクションを抽出する
②抽出した各ファンクションの複雑度を判定する
③各ファンクションの複雑度に応じた点数を付与する
④点数を集計して FP 数を求める

ファンクションの抽出，複雑度の判定は，ソフトウェアの利用者の視点で行う．表 1 に IFPUG 法で規定するファンクションの種類を示した [6]．表 2 に複雑度判定の例として EI（外部入力）の複雑度判定表を，表 3 に点数表を示した [6]．これらのテーブルは，Albrecht が 1984 年，FP 法改定時に定めたものを継承している．EI の複雑度は，データ項目と関連ファイルの数で低，中，高を判定する．ファンクションの種類と複雑度が決まれば，表 3 の点数表で点数を決める．たとえば，ある EI の扱うデータ項目数が 8，関連ファイル数が 2 なら複雑度は「中」で 4 点となる．

この不連続なテーブルに違和感をもたれた方も多いのではないか．EI であれば，データ項目数の複雑度は

表 1　IFPUG 法で規定するファンクションの種類 [6]

種類	説明
内部論理ファイル (ILF)	計測対象ソフトウェアによって維持・管理・更新されるデータの集まり
外部インタフェースファイル (EIF)	計測対象ソフトウェアによって参照しかされないデータの集まり
外部入力 (EI)	ILF を更新する機能
外部出力 (EO)	何らかの加工したデータを出力する機能
外部照会 (EQ)	ILF，EIF の参照データを出力する機能

表 2　EI の複雑度判定表 [6]

		データ項目数		
		1〜4	5〜15	16〜
関連ファイル数	0, 1	低	低	中
	2	低	中	高
	3〜	中	高	高

表 3　点数表 [6]

種類	複雑度		
	低	中	高
ILF	×7	×10	×15
EIF	×5	× 7	×10
El	×3	× 4	× 6
EO	×4	× 5	× 7
EQ	×3	× 4	× 6

1〜4，5〜15，16 以上の範囲で決まるため，データ項目が 8 から 12 に増えても点数は変わらない．FP 法は細かな変動に反応する敏感なメトリクスではないが，計測の労力と精度を考えると実用面からはこのぐらいがちょうどよい．データ項目は画面レイアウトなどを基に数えるが，一目見てどの範囲かがわかれば一つひとつ数え上げる必要はなく．境界を跨ぎそうなときだけ詳細に数える．また，データ項目数の少しの変化で実装する負荷が変わるわけではない．

FP 法と SLOC の比較を表 4 に示した [2]．三つの比較項目は，規模見積もりの目的である成果物スコープのベースラインとして機能するための条件である．

「作り方に左右されない値が得られる」は，「どう作るか」ではなく，「何を作るか」を定量化できるかを評価している．FP 法は，機能要件を定量化する手法で非機能要件や技術要件を含まないので ◯，SLOC は，開発言語に依存することと非機能要件により増減するので × となる．

「計測ルールが明確で，見積もり結果を共有できる」

表 4　メトリクスで考慮すべき条件と評価

項	条件	FP 法	SLOC
1	作り方に左右されない値が得られる	○	×
2	計測ルールが明確で，見積もり結果を共有できる	○	×
3	見積もりからプロジェクト完了まで測定値に一貫性がある	○	○

は，ステークホルダー間でベースラインを共有できるかどうかを評価する．FP 法は IFPUG のマニュアルで計測ルールが定められているので ○，SLOC の計測ルールは企業ごとに異なるため × となる．

「見積もりからプロジェクト完了まで測定値に一貫性がある」は，実績とベースラインの乖離が把握できるかを評価する．メトリクスの利用は，場面に応じて「計測」と「推測」を区別する [1]．FP 法でルールに基づいた「計測」が可能となるのは，データ項目や関連ファイルが明らかになる基本設計段階であり，SLOC はプログラミング段階である．それ以前は，見積もりを含めて不明確な部分を仮定して値を出す「推測」である．FP 法は，NESMA (Netherlands Software Metrics Users Association) の試算法，概算法など，形式化された推測法を用いることが多い [1, 7]．FP 法，SLOC とも推測と計測に一貫性があるため ○ となる．

FP 法と SLOC について成果物スコープのベースラインとして機能するための条件を比較したが，過去に提案された規模のメトリクスでこの三つの条件を満たすのは FP 法しかない．ただ，FP 法も万能ではなく，複雑な処理ロジックをもつソフトウェアは点数化されにくいといった問題もある [2]．FP 法と SLOC を併用するケースも多い．

5.2　工数見積もり

この段階で記述できる作業レベルは粗く，作業量の見積もり根拠となる中間成果物のボリュームも明確になっていない．このため係数モデル見積もりを用いる．見積もった規模を基に工数を見積もる二つの方法を紹介する．生産性を基にした見積もりと数式法の代表 COCOMO II (COnstruction COst MOdel II) [8] である．工数は，「人時」，「人月」のように人数と時間の積で表す．

(1) 生産性を基にした工数見積もり

生産性を基に工数を算出する方式を示す．

$$\text{工数} = (\text{規模} \div \text{生産性}) + \text{関連作業工数}$$

工数は，規模を生産性で除して求める業務アプリケーション開発工数とそれ以外の関連作業工数からなる．生産性は，「FP/人時」「FP/人月」のように単位工数当たりの開発規模で表す．組織として蓄積された生産実績データや類似プロジェクト，経験からプロジェクトの条件に応じて生産性を設定し開発工数を求める．実績データや事例については，サンプル数が多いほど条件，工程などで層別が可能となる．

関連作業工数は，規模や生産性に関連しない工数である．業務アプリケーションの実行環境の設計や開発環境の構築，技術調査などが含まれる．

生産性や関連作業工数を変動させる要因は，「非機能要件」，「技術要件」，「プロジェクト特性」の三つである．非機能要件は「どれぐらいのパフォーマンスか」を示し，可用性や性能，セキュリティ，移植性などの要件が含まれる．技術要件は「どのような方法か」を示し，アーキテクチャー，開発プロセス，開発環境，再利用などの要件が含まれる．プロジェクト特性はプロジェクト固有の特徴や制約条件を示し，要件の明確度や安定度，メンバーの参画度，コミュニケーションなどが含まれる．すべての要件が関係するわけではないので，プロジェクトに該当する要件を反映する．現在，これらの要件を定量化する有効なメトリクスは普及しておらず，ケース・バイ・ケースでの判断となる．

(2) COCOMO II

数式法は，算出式に生産性変動要因をパラメータとして与え工数を求める．以下に数式法の代表 COCOMO II のポスト・アーキテクチャモデル [8] を示す．

$$\text{工数} = 2.94 \times \text{規模}^{\mathrm{B}} \times \mathrm{EM}_1 \times \cdots \times \mathrm{EM}_{17}$$

B：開発の先例性などスケールに影響を与える要因 5 項目の評価値から算出

EM_n：プロダクトやプラットフォームなどコストに影響を与える 17 項目の評価値

数式法は，わが国では普及していない．しかし，実証的，統計的に裏打ちされたモデルは，見積もり値の論理的，合理的な説明に有効である．

5.3　コスト見積もり

IT プロジェクトのコストは，人件費，経費，予備費で構成される．人件費は，見積もった全体工数を職種や自社開発分，外部委託分などに分け，それぞれの人月単価を設定して算出する．経費の主な項目は，開発環境，コミュニケーション環境，旅費や事務用品，会議費などである．予備費は，IT 業界へのリスクマネジメントの浸透とともに，近年，ようやく必要性が認識

されてきた．予備費の算出には，リスク分析の結果から一定の比率をコストに上乗せする方法，3 点見積もりなど幅をもたせて見積もる方法が使われている [1]．

5.4 価格見積もりと価格設定

価格見積もりは，値頃感を考慮して最終成果物の対価を見積もる．発注側が感じる値頃感は，自分自身が見積もった値や以前の発注金額，相見積もりの結果などとの比較から得られることが多かったが，ようやく IT 業界においても世の中の相場との比較ができる環境が整ってきた．これに貢献しているのが公開データである．わが国では，経済調査会 [9]，日本情報システム・ユーザー協会 (JUAS) [10]，情報処理推進機構 (IPA) [11] の 3 団体が IT プロジェクトの情報を収集・分析し，レポートを発行している．コスト見積もりと価格見積もりは，規模を生産性で除して工数を算出し，その工数に人月単価を掛けて人件費を見積もるという流れは同じだが，価格見積もりでは生産性や人月単価に公開データを利用するところが異なる．

生産性は，各団体が発行するレポートからプロジェクトに近い条件のデータを選び工数を算出する．次に人件費を算出するために，全体の工数から必要な工数を職種別に割り付ける．開発工程別職種別参画比率などのデータは経済調査会から提供されている [12]．これにそれぞれの人月単価を掛け合わせて人件費を算出する．職種別の人月単価の相場についても経済調査会から公開されている [13, 14]．

このように公開データと自社の設定値を比較することにより，値頃感や価格を説明できる上限値などの感覚をもつことができる．ただし，超大規模あるいは極めて高い信頼性が要求される特殊なプロジェクトの見積もりに公開データは利用できない．数が少なく公開データではカバーできないからである．

価格設定は，価格見積もりとコスト見積もりの結果を基に，利益率，ビジネス戦略，競合他社の状況，自社の付加価値などを加味して行う．

5.5 見積もりの完了

見積もりの結果，内示を受けたらプロジェクト計画へと見積もり前提条件の引き継ぎを行う．プロジェクトが始まる前に以下のことを実施する．

①ステークホルダー間でプロジェクトの範囲を改めて確認し再度合意する

②見積もり前提条件をプロジェクトに反映する

ソフトウェア開発プロジェクトの見積もりは，多くの前提条件や仮定のうえに成り立っている．これらを漏れなく契約やプロジェクト計画に反映する．

6. 最後に

筆者が考えるソフトウェア開発プロジェクトのよい見積もりとは「単に精度が高いだけでなく，スコープやコスト，スケジュールについて曖昧さを排除したベースラインが設定でき，変更要求に対して変更箇所と責任分担を明確にできる」見積もりである．このために必要な見積もりの流れや技法など，見積もる仕掛けについて説明してきた．ただ，仕掛けだけで解決できるところは限られ，まだ勘や経験，度胸に妥協を含めて個人の見積もり能力に負うところが多い．しかし，20 年前に比べ大きく進歩しているのも事実である．プロジェクトマネジメントの普及に伴うスコープやベースラインの考え方の浸透，公開データの充実，非機能要件に関する議論などが大きく貢献している．

最後に，オペレーションズ・リサーチの知見が加われば，この分野もさらに進展するのではないだろうか．皆様の協力を切にお願いしたい．

参考文献

[1] 初田賢司，『本当に使える見積もり技術（改訂第 3 版）』，日経 BP 社，2013.

[2] 初田賢司，『ユーザーのためのシステム開発の見積もり評価』，日経 BP 社，2015.

[3] Project Management Institute,『プロジェクトマネジメント知識体系ガイド（PMBOK ガイド）（第 5 版）』，2014.

[4] 初田賢司，『システム開発のための WBS の作り方』，日経 BP 社，2012.

[5] ISO/IEC, *ISO/IEC 14143-1: Information technology—Software measurement—Functional size measurement—Part1: Definition of concepts*, ISO/IEC, 1998.

[6] International Function Point Users Group (IFPUG)（日本ファンクションポイントユーザ会訳），『ファンクションポイント計測マニュアル』，日本ファンクションポイントユーザ会，2013.

[7] ISO/IEC, *ISO/IEC 24570: Software engineering—NESMA functional size measurement method version 2.1—Definitions and counting guidelines for the application of Function Point Analysis*, ISO/IEC, 2005.

[8] B. Boehm, *Software Cost Estimation with COCOMO II*, Prentice Hall, 1995.

[9] 経済調査会，ソフトウェア開発データリポジトリの分析，2010.

[10] 日本情報システム・ユーザー協会 (JUAS)，『ソフトウェアメトリックス調査』，2014.

[11] 情報処理推進機構 (IPA)，『ソフトウェア開発データ白書』，2014.

[12] 経済調査会，平成 23 年度ソフトウェア開発に関する調査票　集計結果その 1，2012.

[13] 経済調査会，『月刊積算資料』，経済調査会，2014 年 3 月号以降.

[14] 押野智樹，大岩佐和子，“人月単価の相場を押さえよう，”日経 SYSTEMS，**255**, pp. 64–69, 2014.