SEC BOOKS

ITユーザとベンダのための定量的見積りの勧め

～見積り精度を向上する重要ポイント～

独立行政法人 情報処理推進機構　ソフトウェア・エンジニアリング・センター　編

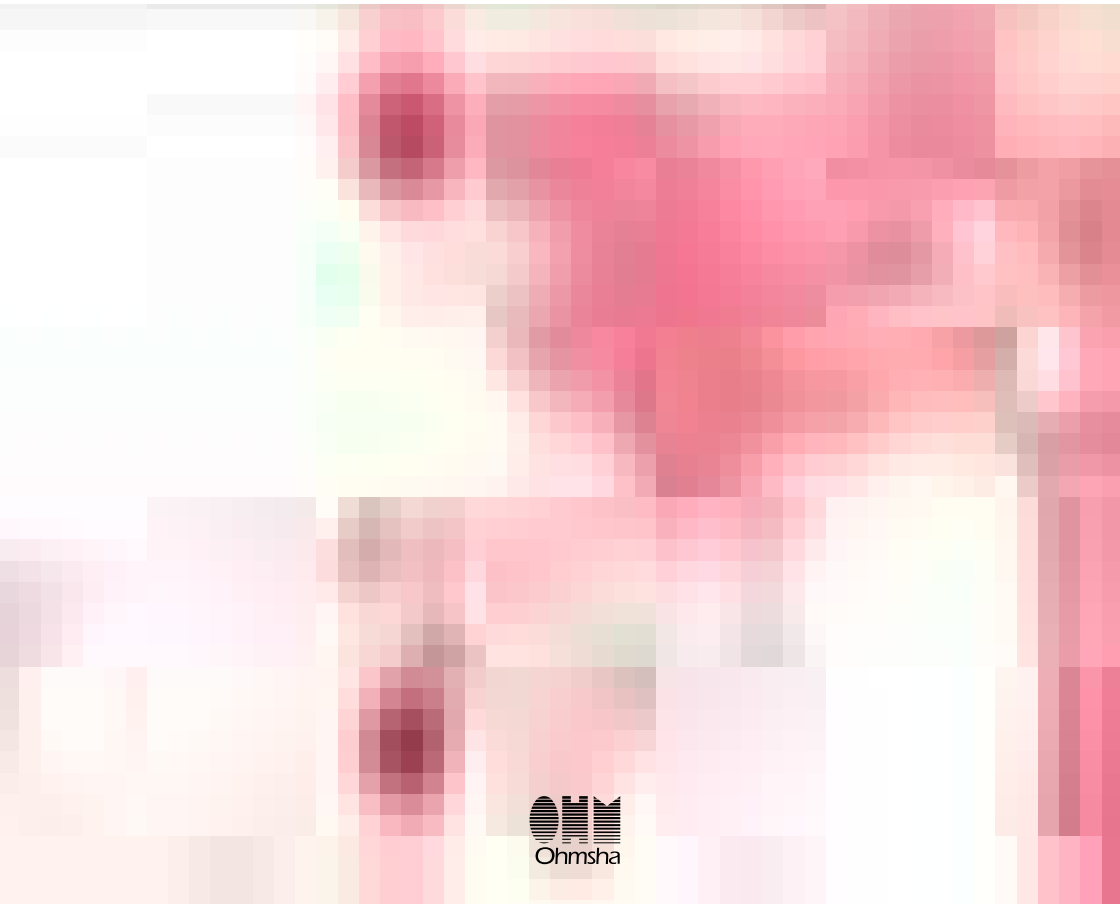
Ohmsha

SEC BOOKS

ITユーザとベンダのための定量的見積りの勧め

～見積り精度を向上する重要ポイント～

独立行政法人 情報処理推進機構　ソフトウェア・エンジニアリング・センター　編



Ohmsha

本書は，「著作権法」によって，著作権等の権利が保護されている著作物です．
本書の全部または一部につき，無断で次に示す〔 〕内のような使い方をされると，著作権等の権利侵害となる場合がありますので御注意ください．
〔転載，複写機等による複写複製，電子的装置への入力等〕
学校・企業・団体等において，上記のような使い方をされる場合には特に御注意ください．
お問合せは下記へお願いします．
〒101-8460　東京都千代田区神田錦町 3-1 TEL 03-3233-0641
株式会社オーム社雑誌局（著作権担当）

はじめに

ソフトウェア・エンジニアリング・センター（以下 SEC と略記します）では、そのさまざまな活動を「小冊子」として発行し成果を公表しております。本小冊子は、その一環として、ソフトウェア開発の見積りに関する研究活動の成果をまとめたものです。

本小冊子は、システム開発を視野に入れ、ビジネスアプリケーションを中心としたソフトウェア開発における見積りの実情および課題とともに、現状を改革するためのポイントを示しています。また、見積りとは、規模、工数、工期、品質（信頼性、性能など）、コストなどのさまざまな要素を広く対象とするものですが、特に規模、工数、工期、コストの見積りに焦点を当てます。

見積りのベストプラクティスとして重要なポイントを示すとともに、ユーザとベンダとの間で見積りに関する共通認識として共有すべき事項を示し、見積り精度、見積り能力向上の一助としたいと考えております。ベンダにおけるベストプラクティスを中心としていますが、ベンダのみならずユーザにおいてシステム開発に関わる方にも参考にしていただけます。つまり、読者としては、システム開発に関わるユーザおよびベンダのすべての関係者を想定しております(注1)。

(注1) なお、本小冊子で述べる考え方や実践内容は、ビジネスアプリケーション以外の分野、例えば組込みソフトウェア開発などでも共通しており、参考にできる部分は多いと考えています。

プロジェクトを成功に導くための基本的な考え方は、システム開発プロジェクトはユーザとベンダの両者の協調活動が重要だという点にあります。ユーザとベンダとは、対峙すべきものではなく、プロジェクトの中にあって、お互いに果たすべき適切な役割があります。見積りに関していえば、次のようにお互いに協力し合うことがプロジェクトを成功させるうえで必須です。

- ユーザとベンダは、必要な情報をお互いに出し合ったうえで、その妥当性をお互いに確認しあう。
- ユーザ側は、見積りを確定するうえで必要な情報を可能な限り示し、ベンダ側は見積りの根拠を明確にする。
- プロジェクトには常に「リスクがつきもの」である点をお互いが理解し、リスクを極力早期に小さくすべく双方が協力する。

SEC では、ユーザとベンダがシステム開発のゴールを明確にし、達成するにはユーザとベンダが協調することが重要であると考え、それを当たり前とする文化の醸成を目指しています。そうすることにより、製品・サービスを提供する側と、提供を受ける側が相互に利益を得て、円満な関係で良好な結果を得ることができれば、健全な IT 産業の発展につながると考えています。

ソフトウェア開発の見積りには、長い研究や実践の歴史があり、すでに多くの結果が生み出されています。いま、システム開発プロジェクトに求められているのは、ユーザおよびベンダがともに自社の条件にあった適切な見積り方法をいかに見きわめ、実践するかという点です。

本小冊子では、見積りの基本と課題について述べ、具体的な見積りの手法については今後発行を予定している「見積りガイドライン」に詳細を示す予定です。

もちろん、見積りの課題すべてに妥当な解決方法が見えているわけでもないことも事実であり、その点も示すとともに、SEC での新たなチャレンジとして、その解決策を研究し、IT 業界に提供することを考えています。

☆ 本小冊子

見積りの重要性について改めて意識を喚起し、見積り精度向上のための考え方、取り組みなどの基本的な内容を概説書としてまとめたものです。ソフトウェア開発にとどまらず、システム全体の開発も視野に入れて論じます。組織の状況に応じた適切な見積り手法があるという意識喚起を行い、見積り方法の考え方を示します。

☆ 見積りガイドライン

小冊子が考え方を示すものに対して、見積りガイドラインでは、ソフトウェア開発の見積りに焦点を当て、具体的な方法やノウハウを示すものです。小冊子の考えに基づいて、見積りの向上を図るための具体的な方法をガイドとして示します。

例えば、ソフトウェア開発における個別の見積り手法と、それにあった前提条件や導入方法について解説する予定です。

2005年 4 月

見積部会 一同

目　次

第4章　見積りのポイント

第１章　見積りの実情と課題

ソフトウェアの開発規模や開発工数の見積りは、システム開発プロジェクトの計画策定や，プロジェクトマネジメントにとってなくてはならない重要な基礎情報です。最初の計画を大幅に間違ってしまうと、その後がうまくいかなくなるのは、ソフトウェア開発に限ったことではありません。例えば、旅行など身近な例をとってみても、最初の計画（時間や予算の見積り）が大事であることはいうまでもありません。

ところが、ソフトウェア開発における見積りほど、重要であるにも関わらず、むずかしいといわれ続けていることも少ないでしょう。なぜでしょうか。

ソフトウェア開発は他の多くの工業製品などの製造に比べて自動化されている部分が少なく、開発プロセス自体が、人的要因に大きく影響されるという特徴を持っています。そのために、見積りにおいては、個々の開発プロジェクトに関わる組織（ユーザやベンダ）の特徴や経験、知識、スキルなどのさまざまな要因を考慮する必要があります。見積りとは、単にわかっていることを足し算する単純なものではなく、複雑なパラメータを把握し、バランスをとりながら、それらを調整する作業を伴っています。結果として、ひとつのプロジェクトについて精度よく見積るためには、そのプロジェクトの特徴を把握することはもちろんのこと、自企業・組織の特性も知る必要があります。しかし、実際には考慮されないことが多いのが

現状です。

また、製造物として目に見えづらいソフトウェアは、作って欲しいものを表現しづらく、（作るものを）聞き出しにくいという特質を持ちますが、そのことが見積りのむずかしさに大きな影響を及ぼしています。よくわからないもの・決まっていないものに対して、正確な見積りができるはずがありません。

そもそも見積りは、急に作成を求められる場合や、プロジェクト立上げの忙しい時期に実施されることが多く、結果として十分に検討されないことが起こりがちです。特にソフトウェア開発の見積りでは考慮しなければならない点が多くあり、かつ、それぞれに不確定な部分が大きいために、余計にあいまいな見積りとなってしまう傾向があります。そして、見積りと実績がなかなか合わないため、根拠をつめる労力を割くよりも、だいたいの見積りを出すことでかまわないという感じになってしまうことも否めません。

以下には、具体的に見積りの現場で起こっている現状をまとめます。

1.1　仕様があいまいで見積りがぶれやすい

- 仕様が明確でなく、見積りのぶれ幅が大きい。
- プロジェクト開始時点で、仕様が決まらない。
- ユーザ側が要求している仕様の表現があいまいで、ベンダもそれを的確に表現できない。
- 仕様があいまいな状況での見積りで予算を定められ、それに縛られる。後で確定する仕様と見積りが合わなくなる。
- 仕様変更が頻繁に発生して、見積りの前提が変わり続ける。

これらの問題は、要求自体があいまいであるにもかかわらず、ユーザの協力が得られない、また、ユーザに理解してもらえず、要求を明確にする

ことができない状況を物語っています。また、ベンダ側のユーザの業務に関する知識不足、プロジェクトマネジメントに関する経験やスキルの不足など、ベンダに起因する場合もあります。システム開発の専門家として、ユーザの立場に立った思考がベンダができていないケースや上記のようなあいまいな状況のまま、ベンダがプロジェクトを進めてしまうこともあるかもしれません。

1.2　ユーザとベンダとの間の納得感の欠如

ソフトウェアは具体的に物理的な量としてイメージすることがむずかしいため、見積りは特に困難なものです。ソフトウェア開発の専門家であるベンダにとってもむずかしいものです。ソフトウェアおよびソフトウェア開発に関して専門家ではないユーザにとっては、開発されるソフトウェアの量とそれに見合う対価としての費用の妥当性の判断は、いっそう困難をきわめるといっても過言ではないでしょう。

ユーザとベンダとの間で、ソフトウェアの規模を表現する尺度が「共通の言語」として確立されていないことは、さらにこの状況を悪化させています。結果的に、次に示す問題が起きて、ユーザとベンダ間で納得感が得られないという状況に陥りがちです。

- ベンダは見積り内訳の詳細を説明できる根拠がないことが多いため、「勘と度胸」に頼り、さまざまなコスト要素を分析しきれず、おおざっぱな見積りをせざるを得ない。
- ベンダは大くくりの見積り（一式の費用）をユーザに提示するにとどまり、ユーザはその妥当性を判断する術（すべ）をもたない。
- ベンダから提出される提案書は、各社各様で作成されているため、ユーザ側で提案内容を評価する方法・基準がない。
- ユーザがベンダ側のソフトウェア開発の詳細見積りの「ことば」を理

解できず、双方でコミュニケーションを密に取れない。例えば、

- ソフトウェア規模の単位・尺度としての「ことば」（例：ファンクションポイント(FP)、ライン、ユースケースなど）
- ソフトウェア開発に関わる「影響要因」（例：複雑度、先進性、高信頼性、セキュリティなど）がコストへ及ぼす影響（コストを押し上げる要因）についての認識

- 最悪のケースでは、むりな値下げをせまられても、説明能力が不足しているために、むりを押し返せず、安請負してプロジェクトの破綻を招く場合がある。
- 一品生産品である受託ソフトウェアと市販品のパッケージソフトウェアとの機能・金額が比較され、受託ソフトウェアが割高であることにユーザが違和感を覚えることがある。
- ユーザ企業の経営者の観点に立てば、本来ソフトウェア開発の対価はソフトウェアによってもたらされる付加価値に対して支払われるべきところ、現在は工数が価値に対する代用指標となっているケースがほとんどである。このために、生産性を評価対象としない場合は、生産性の高いベンダが受注すると利潤が低くなるというパラドックスが生じる。

1.3　問題意識のまとめ

以上、見積りに関する問題をまとめると次のとおりです。

- 見積りはプロジェクト全体の最初の基礎データでありながら、場当たり的になされ、プロジェクトが失敗に追い込まれる事例が多数ある。
- 正確な見積りを行うためには、さまざまな課題を克服する必要がある。特に仕様があいまいである場合の影響は大きい。あいまいな仕様の見積りは必ずぶれるため、そこにはリスクがあることをユーザとベンダで理解するべきである。
- ユーザとベンダ間でプロジェクトの見積り結果および過程について共有し、納得感を得ることがプロジェクトの成功に不可欠である。
- ベンダはなぜその見積りになるかを説明でき、ユーザはその内容を理解できる状況になるべきである。

第 2 章　見積りの重要性

第 1 章で述べたとおり、ソフトウェア開発での見積りには、多くの課題が存在しています。一方、ソフトウェア開発は、さまざまな利害関係者が参加して進められますが、見積りはいずれの立場にとってもビジネス的にますます重要なものとなっています。それぞれの立場で課題を解決し、改善の方向に進める必要があります。

2.1　ユーザの経営における重要性

企業にとってシステム投資は企業戦略の根幹であり、昨今ビジネス成功のために重要な位置づけであることが再認識されています。かつてのような業務の電子化のみの目的から、戦略的ビジネスに呼応するシステム化へと変化しています。

- 投資案件の見積り内容について、企画部門から経営層へ、経営層から株主へ説明が求められる場面が増加している。
- ユーザもシステム開発コストの妥当性、適切性、費用対効果（ROI：Return On Investment）などの判断を重要視している。

上記のように、ユーザ企業では投資額が妥当な金額であるか的確に把握することが大きな課題となっています。

2.2　ベンダの経営における重要性

ベンダ企業にとっては、利益確保の面で、各プロジェクトの見積りの精度は、プロジェクトの成否に直接つながります。低すぎる見積りでは、プロジェクトで必要なアウトプットを達成するのが困難になります。一方、妥当な見積りを実現すれば、ユーザも妥当な価格で適正な品質のシステムを得ることになり、ベンダおよびユーザの双方の利益につながります。

また、ユーザの信頼を確保することは継続的な利益確保の原点であり、見積りは、次の観点からかぎとなるものです。

- 見積りの根拠を説明できないベンダは、企業そのものの誠実さや能力を問われる。
- ユーザは、見積りを介して見積り金額のみならず、プロジェクトマネジメント能力、技術力、品質管理能力、動員力など、ベンダに対して総合的な評価を行う。

なお、双方が納得できる見積りを作る過程で、相互信頼関係の構築といった利点も見逃がせません。

2.3　プロジェクトマネジメントでの位置づけ

（1）ベンダにとって

例えば、見積り金額や工期など、ベンダにとって見積りとは、顧客との開発プロジェクトの全体枠を決める重要な要素です。金額などの見積りが決まると、その他の拘束条件（作業量、開発規模、開発体制、……など）が基本的に決まってしまいます。見積り結果は、プロジェクトの実施内容や方法まで規定してしまうことから、もっとも基本的なデータとして捉えなければなりません。

特に、いったん契約額として見積り額が決まると、ベンダ側（プロジェ

クトマネージャ）は、その契約額を基準としてプロジェクトを組み立て、開発完了時には、妥当な利益を確保しつつ、ユーザの要望を満足させるソフトウェアを構築しなければなりません。つまり、見積りにうたわれた結果がプロジェクトの予測から目標（ゴール）となります。その見積り結果に応じたプロジェクト成果をいかに出していくかという課題に変わります。

その意味で「見積りの妥当性」とは、プロジェクトマネージャがその見積りでプロジェクトを成功裏に終わらせることが「むりではなく」、「実現性が高い」というものであると言い換えることができます。

見積りはプロジェクトの見通しをつける最初の指標で客観的であるべきものですが、その位置づけをまとめると次のとおりです（図 2.1 参照）。

- 契約に対するインプット
- 進捗マネジメントでの基礎
- リスクマネジメントの基礎

見積りの精度を向上させようとすると、リスクを洗い出し、その

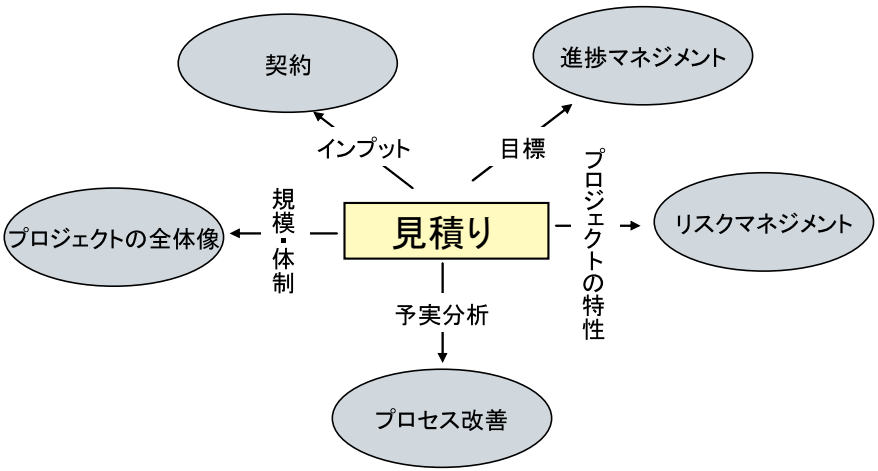


図 2.1　見積りの位置づけ

影響度を見積もる必要があります。これは、リスクマネジメントの活動の一環とみなせます。

- プロセス改善での活用

予実分析による見積り精度の確認は、プロセス改善につながります。例えば、予実に差があれば改善の機会として追跡し、プロジェクトにおける問題を浮き彫りにすることができます。

（2）ユーザにとって

ユーザの特に情報システム部門にとっても、システム開発において見積りをむずかしくすることに対する認識を深め、むりなプロジェクトの発注を避けることによって、妥当な価格で適正な品質のシステムを予定内に構築することにつなげられます。

第３章　見積り能力の向上

見積りは、経営者、プロジェクトマネージャ、開発担当者、ユーザがそれぞれの立場で、ソフトウェア開発を客観的な尺度に基づいてチェックし、管理するための基準となるものです。

いずれの立場においても、見積りが妥当なものであるか否かは、実施しようとするプロジェクトの成否を分ける大きなかぎであり、見積りを確実に行うための能力の向上が重要です。

3.1　ベンダにとっての見積り能力の向上

（１）　見積り能力の向上

ベンダにおける見積り能力の向上とは、実現可能性の高い見積りを繰り返し実現することです。また、個別のプロジェクトで見積りの根拠をきちんと説明できることが基本ですが、担当者によって見積りの説明方法が違うと、外部から見たとき統一性がとれていないという印象を与え、ユーザの理解が得られづらくなります。見積り能力には、組織として統一的に説明可能な見積りを示すことも含まれます。

組織として統一的に説明可能な見積りを実現することは、自社の組織にあった見積り手法を採用するか、または作り上げることと同義です。

（２）　見積りプロセスの確立

自社の組織にあった見積り方法を構築するためには、自社の組織におけ

る開発プロジェクトの特徴を理解する必要があります。例えば、現状の標準的なレベルのプロジェクトでは、どのような生産性・チーム編成で実施するのが通常か、あるいは逆にどういった点が、プロジェクトの生産性、品質、工期や成否に大きな悪影響／好影響を及ぼしているか、といった点を把握していないと、個別のプロジェクトでの妥当な見積りは、むずかしいものとなります。

そのため、過去のプロジェクトデータや経験の蓄積に基づいて、一度はきちんと自社の開発プロジェクトの特徴を確認しておくことが望まれます。

また、見積りは、あくまで個別のプロジェクトの最初の第一歩であり、いったんプロジェクトが終了すると、今度は実績との差を分析して、見積り方法そのものを評価したり、またはプロジェクトが想定外の動きを示した点を把握したりして、次回プロジェクト以降での妥当性向上のためのきっかけとすべきものになります。

こうしてみると、見積りは、プロジェクト開始時の単なるひとつの活動ではなく、プロジェクト間をまたいだ組織的な活動であることがわかります。プロジェクトの成否に及ぼす影響の分析は、組織のプロジェクトの強み・弱みを探ることと同じです。また、次回のプロジェクトでの見積りの妥当性向上を図るための差異分析や、それに伴う経験・知識の集積は、プロセス改善の一部となります。

見積りの熟練者は、個人的にこのあたりのプロセスと定量的な感覚を築き上げています。まずは、その知見を組織で共有することから始め、さらには組織的にプロセスを組み立て、広く組織内に展開していくことや、組織的にデータを蓄積することで、定量感を組織内で共有することが、見積りの能力向上につながります。

3.2　ユーザにとっての見積り能力の向上

（1）　見積りの妥当性の判断

ベンダ側だけでなく、ユーザ側も、妥当な価格を判断できる術（すべ）をもつことが必要です。これは、システムを戦略的に構築するには、ユーザが自らの費用対効果を正確に把握するために、妥当な費用が見積もれるか、見積もられた費用が妥当であることを判断できる必要があるからです。妥当性を判断する術（すべ）を持たないと、気がつかないうちにむりなプロジェクトにしていることに陥りがちになりますし、当然のことながら、逆に必要以上の金額を要求されても気が付かないことにもなりかねません。ベンダ側から示された見積りの妥当な評価が行えなければなりません。

これを実現するためには、ユーザ側でも、一貫した尺度と方法で見積りを評価し、ソフトウェアのコストおよび品質に関連する見積りノウハウおよびプロジェクトの実績値の蓄積を自社において図らなければなりません。

（2）　要求の明確化

見積りの妥当性の向上を図るうえで、実はユーザには重要な役割があります。見積り能力に直接関係はありませんが、可能な限り要求を詳細かつ明確にすることです。要求や要件の確定には、ベンダの役割も大きいですが、そもそもユーザ自身がどういうシステムを構築すればよいか、十分なイメージを持っていないと構築されるものは決まりません。これは、結果的に妥当な見積りができないことにつながってしまいます。繰り返しになりますが、わからないものに対しては、正確な見積りができないからです。

この問題への対処には、ふたつ方法があります。ひとつは、「決まっていないことを決めて見積りを作成する」ことであり、もうひとつは、「見積りに不確定な要素があると合意形成し、確定するまで不確定要素の変化を監視して、決まった時点で再見積りする」ことです。

ただし、前者については、諸般の事情で決まらないことも多いため、後者のほうが現実的な対処方法だといえます。

第 4 章　見積りのポイント

これまでは、課題とその解決を中心に見積りの全体像について述べてきました。ここでは、妥当な見積りを実践する上で重要なポイントについて示していきます。表 4.1 には各ポイントの実施の主体がユーザかベンダかを示しますが、双方ともにすべてのポイントについて、ぜひ理解を深めてください。

表 4.1

	ポイント	主　体
1	見積り時期とリスク	ユーザ、ベンダともに
2	見積りと契約の関係に関して	ユーザ、ベンダともに
3	見積り範囲の明確化と見積りの手順	ユーザ、ベンダともに
4	見積り活動	主にベンダ
5	複数見積りによる相互チェック	主にベンダ
6	計画と実績の差異分析	主にベンダ
7	体制・役割分担・企業文化	主にベンダ

4.1　見積り時期とリスク

見積りを行う時期（見積り時期）と見積りの不確実性（リスク）とは、大きな関係があります。

早い時期での見積りには、システムやソフトウェア全体としてわかっていないことが多く、最終的なものと大きな乖離がある可能性が高いことは

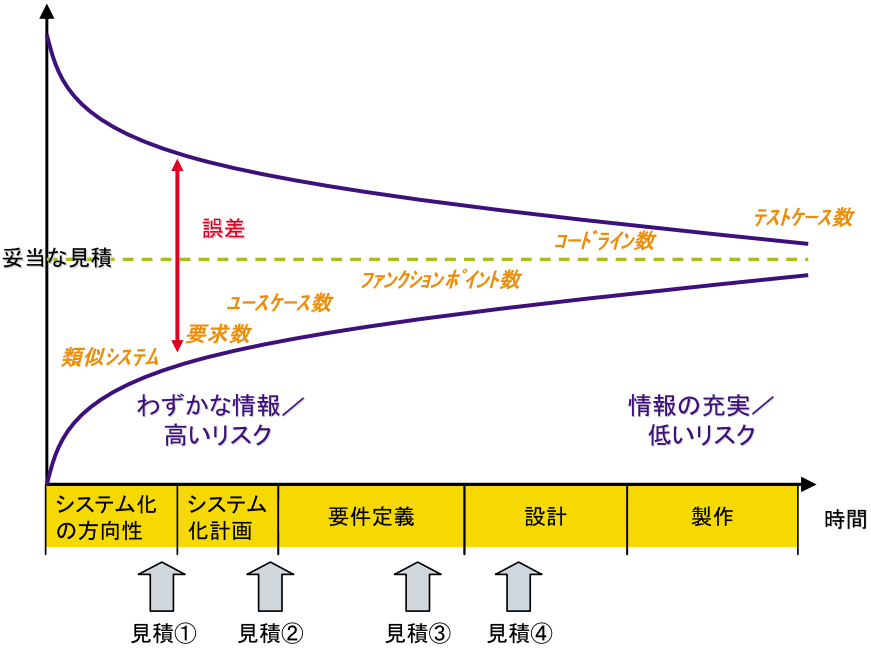
明らかです。まず、はっきりいえることは、わかっていないものを見積もることはできません。わかっていないことが多いほど（見積り時期が早ければ早いほど)、見積りのリスクは高くなるのは当然のことといえます。

このことは、早い時期での見積りを行ってはいけないということではなく、早い時期での見積りには不確定の要素が強く、そのことをきちんと認識して見積りで考慮しておく必要があることを意味しています。そして、早い時期には早い時期なりに、明確になっていることを洗い出す必要があります。

例えば、システム化投資の年度計画を立てるときには、どのような分野に、どのような機能を持ったシステム化を考えているのかを、不確実な要素も含めて可能な限り明確にし、それを根拠に見積もることが考えられます。そのときに、単にその見積り値だけでなく、最大どの程度の振れ幅が考えられるのかも把握して、リスクを考慮した見積り値を考えておくことが重要です。

危険なのは、この段階の不確定要素の強い見積りをその後のプロジェクトが、実際に開始した後も目標値として設定してしまうことです。まだ良くわかっていない時期に見積りをした場合は、不確定な部分が明確になった時点で、再見積りを行う必要があります。また、何を根拠に見積もったか、何がまだわかっていないかを明確にしておき、ユーザ・ベンダ間で共有することが重要です。お互いがあいまいに持っている考えを明らかにすることで、それを監視して見積りに合わせるように、プロジェクトをコントロールするきっかけにすることができます。

最初の予算決定のときに使われた概算予算が、最後のプロジェクト化まで制約となり、プロジェクトの失敗の大きな原因になることがあります。例えば、本来は初期見積りよりも 2、3 倍または10倍も必要だった予算がむりやり圧縮され、実現不可能なプロジェクトとなってしまうというもので



（注）図中斜体字は、その時点で見積りに使用可能な基礎数値である。

図 4.1　見積り時期と見積り誤差・リスク

す(注2)。

ユーザもベンダも、現在どの時期（要件としてどの程度固まった時点か）で見積りを行っているかを把握したうえで、

① どの段階でどの程度の見積りができるかを共有する。

- 見積りの根拠として何が（データ項目、画面数など）利用できるか。

(注2) ロバート・グラスの「ソフトウェア開発55の真実と10のうそ」には、「プロジェクトを失敗させる原因は二つに帰着される。一つは仕様未確定であり、もう一つはむりな見積りである。」としています。さらに、むりな見積りを仕方がないとほとんどの人が考えている異常さをも指摘しています。

- どの程度の不確実さ（粗さ、安定度など）があるか。
- 使える見積り手法は何か（第 5 章参照）。

② 見積り時期によってどの程度のリスクがあるかを共有する。

③ 見積りの妥当性を向上させるための方策をユーザ・ベンダともに採用する。

特に、図 4.1 の見積り①や②については、プロジェクトの初期段階での正確な見積りは本来不可能であり、不確実な要素を監視し、見積りの範囲内に収まるようにコントロールするためには、要件が確定した段階（例えば、見積り③）で再見積りを行うなどの方策をとる必要があります。この点については、ユーザ・ベンダ双方の経営側の理解も必要であり、最初の予算と計画が具体化したときの予算の違いがある可能性を認め、その変化量が妥当なもの（説明がつくもの、経営的に許容できるもの）であれば、その変化を受け入れないと、むりなプロジェクトになってしまいます。

なお、見積り②についても、早期の段階でもなんらかの「見えているもの」はあり（機能例：コンポーネント数、データ例：レコード数、要求品質例：信頼性、使用性、移植の可能性、システム方式例：他との連接）、そこから過去の類似例を見つけ出すことや、分野によっては、画面とその構成要素から最終的なシステム全体をある程度の精度で把握することができます。

ただし、リスクが高いことには違いはありません。そこで、後の工程（理想は、見積り③）で、再見積りを行う段階的な契約にしておくことがなされる例もあります。ユーザおよびベンダが、互いにリスクを軽減するためには、見積りを繰り返し、精度を向上させる対策が求められます。

いずれの見積り段階にも、それにあった方法は、これまでに数多く検討されていますから、妥当なものを選択して、導入することが課題となります。このあたりの具体的な導入方法は、「見積りガイドライン」において詳

細を示す予定です。

4.2　見積りと契約の関係に関して

個々のプロジェクトにおいて、見積りと契約は、必ずしも直結する概念ではありません。見積りはシステムの規模、あるいはそれを実現するための作業量を定義するものであって、それが直接契約金額になるものではありません。しかし、「契約」行為、すなわち、金額ベースでのステークホルダ間での取り決めの重要なインプットとなります。ここでは、プロジェクトの初期段階で見積りと密接に関係する契約について示します。特に、契約金額が固定となる場合について述べます。

論点を明確にするために、ここでの契約の概念では、計画や要求定義などの上流工程から、テストを経てシステムを本番稼働させるまでの開発ライフサイクルを前提とした、一般的なシステム開発プロジェクトの契約を念頭におきます。つまり、長期にわたるアウトソーシングや保守のみの契約などを除きます。

（1）　定額方式の契約における価格構造

定額方式の契約における契約金額は、基本的に図4.2のような構造をしているものと考えられます。各社・各団体によって科目が異なったり、含めるべき項目の詳細度が異なったりすると思われますが、基本要素としてはコスト、リスク対策としての安全率および役務とは別に確保される利益の3要素があげられます。

この構成図により見積りが契約額に直結するわけではないことが分かります。

（2）　契約額の変動と見積りの関係

契約において、ユーザ側の経済原理は、ユーザの要求（機能、品質、ビジネスでの利用効果）が実現できるという条件下で、できるだけ発注額を

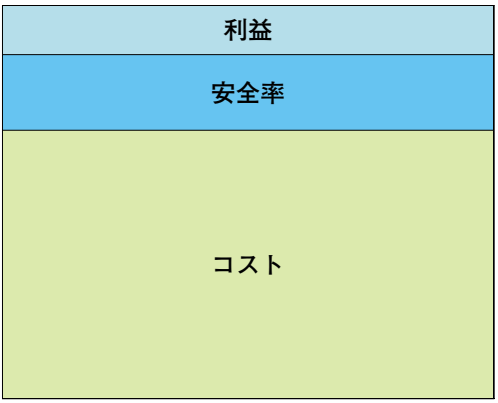


図 4.2　定額方式の契約における価格の基本構造

減らすことです。これに対し、ベンダ側の行動原理は、全体の契約額の中で、適正な利益を確保したいということにあります。これは、入札案件や競合企業とのコンペとなる場合、契約額を増やすことには限度があるためです。したがって、ベンダ側がとるべき対応は、少ない作業量で品質の良いシステムを開発できるよう生産性を向上させる工夫を行うこととなりますが、これが適正価格を維持できないほど額面が切り詰められるとむりが生じることになります。

つまり、契約額の削減要求は、以下の対応のいずれか（あるいはそれらの組み合わせ）でカバーしなければなりません。

1. 利益の圧縮
2. 安全率の縮小
3. コストの縮小、ひいては規模または工数の縮小

これらをどのような優先順位で、どの程度まで縮小を行うかはベンダ企業各社の判断によりますが、むりな契約額の削減は、ユーザ側にとっても不利益を生む危険性があることは明らかです。

特に、受託ソフトウェア開発の入札案件などのコンペは、ユーザ側も規模、品質、工期、価格に関する見積り尺度と、妥当な数値幅を持ち合わせている必要があります。単に価格が安ければよいだけの判断は、ユーザとベンダ共に、不利益をこうむる結果になります。

（3） 多段階契約と契約形態の例

図 4.3 に、多段階契約の例を示します。契約のタイミングとしては、「システム化の方向性」、「システム化計画」、「要件定義」のそれぞれが完了した時点が考えられます。見積り時期の違いなど、プロジェクトの特徴に応じて、最適な契約のタイミングを図ることになります。また、リスクやインセンティブなどを考慮して、契約形態にも、さまざまなものがあります（コラム参照）。

現在は、「システム化の方向性」から「要件定義」までの段階の活動は、実際にかかった作業量に応じて支払われる実績方式（コラムの表では間接作業型）が多く見られます。

契約のタイミングおよび契約の形態に多様な選択肢が可能であることを念頭において、プロジェクトを成功につなげるために適した方法を探る工夫が可能であり、重要です。

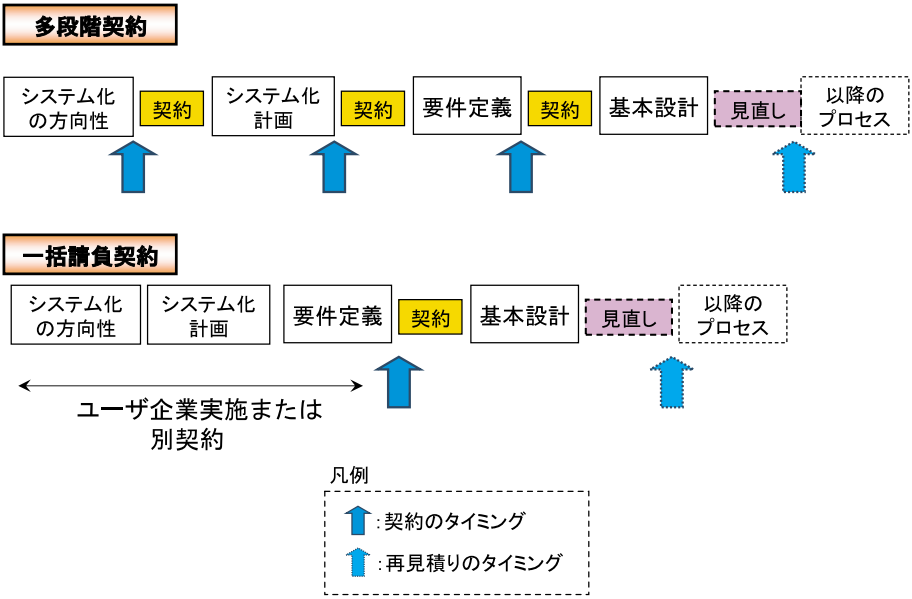


図 4.3　多段階契約と再見積りのタイミング例

コラム（契約形態の考え方）

契約形態の考え方の包括的な例として、米国政府が調達時に使用する米国 FAR（Federal Acquisition Regulation）の分類があります。まず、契約を支払形態から分類すると、

① 定額型契約（Fixed Price）
② 実費償還型契約（Cost Reimbursement）
③ 間接作業型契約（Level of Effort）

の三つのカテゴリーに大別できます。

次に、それらの中でも支払の対象をどのように考えるか、あるいは結果として超過あるいは余剰となったコストの扱いをどのようにするかによって、分類できます。これらの分類には「動機付け型（incentive type）」や「報奨型契約（award type）」があります。下表にはこれらの組み合わせをまとめました。

契約形態による分類

支払形態による分類	契　約　分　類
定額型契約 (Fixed Price)	FFP (Firm Fixed Price)
	FPI (Fixed Price Incentive)
	FPAF (Fixed Price with Award Fees)
実費償還型契約 (Cost Reimbursement)	CPFF (Cost Plus Fixed Fees)
	CPIF (Cost Plus Incentive Fees)
	CPAF (Cost Plus Award Fees)
間接作業型契約 (Level of Effort)	T&M (Time & Material, Labor Hour)
	FFP-LET (FFP-Level of Effort)
	Cost-Plus-Fee Term

4.3　見積り範囲の明確化と見積りの手順

4.3.1　見積り範囲の明確化

システム開発プロジェクトに関する見積りを漏れなく行うために、見積もる範囲（スコープ）を明確にする必要があります。これは、ユーザおよびベンダ間で共有されなければなりません。お互い見ているところが違うと、大きな認識ギャップになってしまいます。また、見積り対象に抜けがあると、大きくはずしてしまうことは明らかです。

スコープを明確にしたうえで、それぞれの要素に適した方法で、規模、工数、工期およびコストを見積もることが基本ですが、スコープの明確化は次の二つの側面（(1)見積り対象システム全体の明確化と、(2)システムライフサイクル全体の明確化）から行うことが薦められます。

（1）　見積り対象システム全体の明確化

① 開発対象ソフトウェアの明確化

システムにおける開発対象ソフトウェアの機能範囲、機能要件および非機能要件の明確化が必須です。

② ソフトウェア開発条件の明確化

開発対象ソフトウェアの明確化では、同時にソフトウェアを取り巻くシステムの全体像などの条件（システム形態、ハードウェア条件など）を明確にしなければなりません。例えば、次のようなものがあります。

- ハードウェア
  - ソフトウェア開発に必要な機器、納品後の稼働環境に必要な機器、通信機器など
- ソフトウェア
  - ソフトウェア開発に必要なパッケージソフト、OS（Operating

System)、ツールなど

- サービス
  - 顧客に対するシステムオペレーション訓練、納品後のアフターサービスなど
- 付帯作業
  - 管理作業など
  - インフラ整備：ハードウェア構成設計、環境構築など

（2） システムライフサイクル全体の明確化

システムのライフサイクルを通じて必要となる作業（アクティビティ、タスクなど）を明確にします。

- 調査・企画
  - システム導入の効果検討、最新技術の調査、システム機能検討など
- 設計・構築
  - システム要件に基づくソフトウェア機能の設計、モジュール設計、コーディング、テストなど、システム構築に関連するタスク
- 保守・運用
  - 顧客に納品されたシステムの機能改善、バグ修正、是正処置などの保守活動
  - システムの定常的運用、障害対応など

なお、作業を明確にしたうえで、さらに、もうひとつ大事な視点があります。上記のようなアクティビティをユーザ、ベンダのうち、どちらが責任を持って実施するかという点です。例えば、調査・企画の中の内部調査というアクティビティがあった場合、ユーザ側で実施する場合とベンダ側で実施する場合で、見積りの範囲が変わります。これは結構見落とされがちですが、プロジェクトの成功のためには、ぜひともプロジェクトでの役割分担を明らかにしておくべきです。

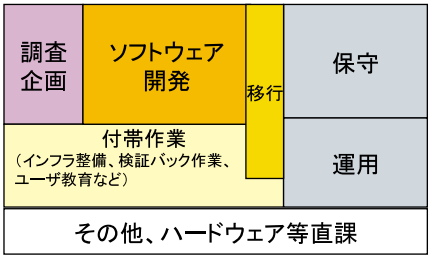


図 4.4　見積り対象の構成要素例

4.3.2　ソフトウェア開発に対する見積りの手順

システム開発では、図 4.4 に示すとおり見積り対象としたものの規模や作業量を見積もることになりますが、ソフトウェア開発以外については、現在作業の積上げなどによる方法が一般的で、ここではソフトウェア開発に絞って見積りの手順を示します。

見積り尺度の構成要素として、規模、工数、工期、コストなどがあり、これらが相互に関係しています。また、見積りに影響する要因として、現在注目を集めている課題に、非機能要件があります。

（1）　オーソドックスな手順

ソフトウェア開発におけるオーソドックスな手順では、要件を決めて、必要な成果物と活動（作業量）を見積もります。成果物は、その規模を見積もり、次に、規模と当該プロジェクトの特性（工数に影響を及ぼす要因）から必要な工数を求めるのが一般的な手順です。

具体的には、

- 見積もる対象（成果物、作業）を明確にし、
- 対象の測定単位を定め、
- 成果物の規模を見積り、
- 成果物を作るためのベースライン（生産性）に基づいて、それからのぶれ（プロジェクトの特性に応じたぶれ）

を求めます。以下に概要を示します。

① 見積りのスコープの設定

プロジェクト全体のどの部分・何を見積もるのかを明確にします。以下では、ソフトウェア開発を対象とします。

② 自分の組織・開発分野にあったメトリクスの選択

ここでの観点は、次のとおりです。

- 成果物の特徴に応じたメトリクス（以下は例）
  - ・Web系なら、画面数など
  - ・データ中心ならデータ項目数
  - ・プログラミング言語が一定ならプログラム行数(SLOC)
  - ・プログラミング言語など、プログラムの作り方によらず機能性を重視するならファンクションポイント(FP)
  - ・ドキュメントならページ数または文字数
- 見積り･分析の手間と精度のバランス
  - ・例えば、把握しやすい画面数とその構成要素を抽出して、簡便に工数を見積もる概算方法があります。これは、精度の議論よりは、簡便さを重視するものです。
- 見積りの利用者（経営者、ユーザ、開発者など）にわかりやすいメトリクス

③ ソフトウェアの規模の見積り

上記のメトリクスに基づいて、成果物の規模・品質を見積もります。不確実な要素を洗い出し、決定する、またはぶれ幅を合意するなど、ユーザと調整し合意します。

④ プロジェクトの工数の見積り

標準的な生産性を基に、規模から工数を求めますが、プロジェクトはそのつどの事情により、常に標準から外れていると考えるのが健全です。た

だし、あらかじめ要因を洗い出しておき、その影響度を決めておく必要があります。

- 成果物の規模・品質から必要な工数の算定
  - ・標準的な生産性
- 標準からはずれる要因の洗い出し
  - ・要因の例：プロダクト特性、開発チーム特性、システム特性など
  - ・要因に対する影響度の設定
  - ・影響度の設定方法の選択
    - 過去のプロジェクトデータからの分析
    - プロジェクトリーダなどへのインタビューによる分析

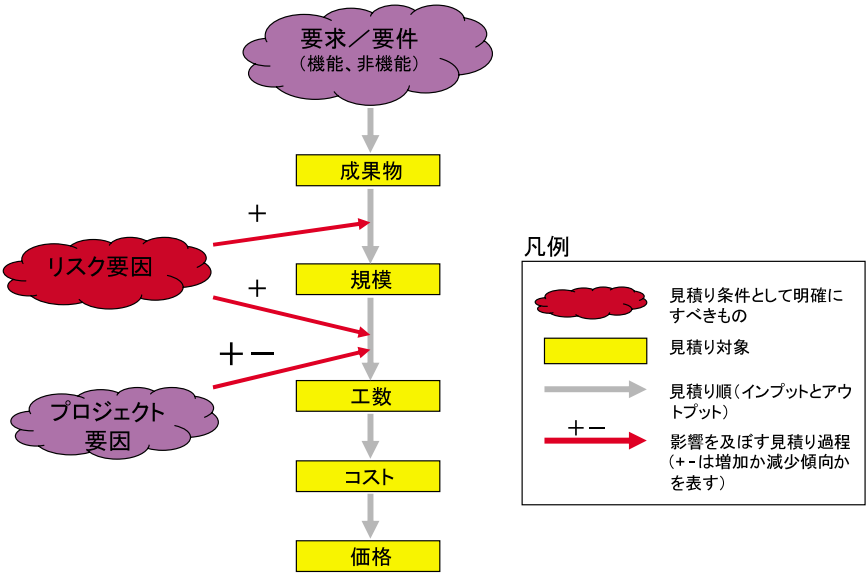


図 4.5　オーソドックスな見積りの手順

（2） 予算などの別の制約条件から始まる見積りの手順

一方、(1)のオーソドックスな手順とは逆に、システム開発に対する予算が先に決まっている場合、そこから妥当な品質・工期で作り上げることができる規模を見積もる場面も多く、もしかしたら、こちらのほうが多いくらいかもしれません。

この場合、予算からプロジェクトとしてかけられる工数を見積もることから始まります。続いて、開発できる規模（実現できる機能量など）を見積もることになりますが、必要なすべての要件をシステムに組み入れるのがむずかしいことが多く、要件の優先順位づけや、最初の見積り（ベースライン）からの変更管理が重要となってきます。

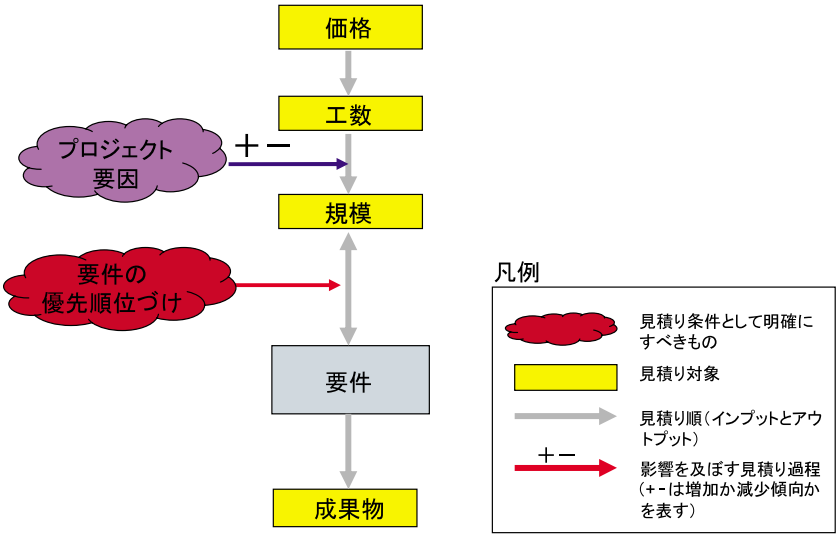


図 4.6　価格が先に決まった場合の要件の見積りの手順

（3） 見積り手順のさまざまなパターン

要件が先に決まるオーソドックスな場合と、価格が先に決まる場合の二つを見ましたが、見積りの手順は、プロジェクトの特性・制約条件によって、さまざまなパターンが考えられます。図 4.7 に、見積りに関連する主な構成要素と、その関係のイメージを示します。

いずれの要素も制約条件（見積りのスターティングポイント）となりうると同時に、いずれの要素も見積りの最終結果となり得るものです。このあたりのバランスをうまくとりつつ妥当な見積りを行い、プロジェクトを成功させる必要があることがわかると思います。

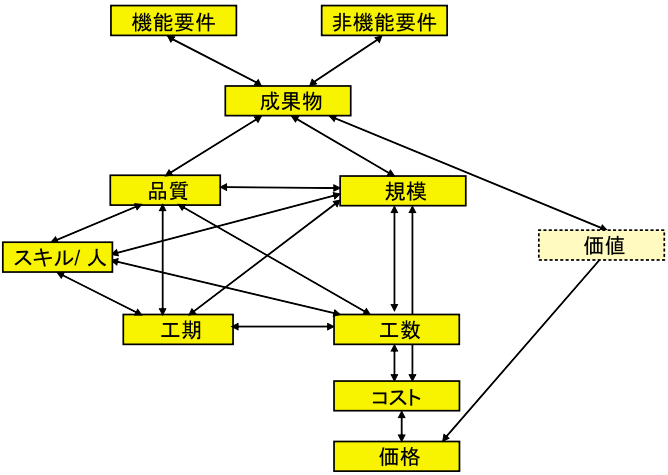


図 4.7　ソフトウェア開発の見積りに関連する構成要素

4.3.3　ソフトウェアの価値

現状では、プロジェクトの全体工数からソフトウェアの価格が設定されることがほとんどです。一方、ソフトウェアを利用するユーザの立場からは、ソフトウェアのユーザにとっての利用価値（ソフトウェアを利用して達成するビジネス上の効果）に基づくべきであるという意見もありますが、

議論がつくされていません。

しかし、例えば、他の企業より早く商品・サービスを市場に提供するために、コストがかかっても、特定の期限内（短期間）に終えることなどをユーザが要求する場合、この要求は図4.7の「工期」に対する条件として価格に反映されます。これはユーザから見れば、市場投入期間（Time to Market）の短期化によるビジネス機会の拡大という価値を買っていることになっており、実はユーザから見た価値が、価格に反映される良い例となっています。

このようにユーザが目標とするソフトウェアの利用価値を根拠として、ソフトウェアに対する要求（機能要求、非機能要求、納期など）を定め合意して、これを入力としてソフトウェアの価格を見積り、ソフトウェアの価値とソフトウェアの価格がバランスするように、ユーザとベンダが協力して、ソフトウェアに対する要求を調整する視点は重要です。

4.4　見積り活動

4.4.1　プロジェクト単体での見積り活動

プロジェクトにおける見積りの主な活動は、次のとおりです。

（1）プロジェクトの前提条件の確認

- 見積り対象の把握
- 契約や要求仕様の特定および確認
- 非機能要件（技術要件、品質要件など）および工期の要件の確認

（2）規模（成果物、作業）の見積り

（3）見積りの検証

- 見積り手法をいくつか使用して、対象となるソフトウェアに関する見積り値のシミュレーション

（4）得られた見積り値から、最適値の選択または決定

（5）プロジェクトの前提条件として、確認した見積り対象の要件や条件についての変更管理

以下の方策などを実施することにより、工数見積りの信頼性を向上させることが期待できます。

- 複数の見積り手法の併用
- 開発工程の進展に合わせた段階的見積り
- ユーザの期待効果と要件や仕様の早期確定
- 要件変更ルールの取り決め（ユーザ側、ベンダ側）
- 要件変更に伴う再見積り

プロジェクトが進行するにつれ、新たな要件の発生や、仕様の詳細化に伴って、要件や規模は見積り時点のものからずれて、規模の増大も発生しがちです。また、実現したい要件とプロジェクト予算は、どちらか一方が決まれば、他方が決まるものではなく、できるだけ多くの要件をより少ない予算で実現したいという相反するニーズがあります。

そのため、見積り精度向上の努力・工夫とともに、プロジェクト成功のためには、見積り時点からの変更管理をユーザとベンダが協力して継続的に行っていくことが重要です。見積りのベースラインを決め、そこからの要件や規模などの変更をしっかり管理していく必要があります。

そのためには、次の事項などを行うのが基本です。

- 見積りの根拠をまず明確にすること（何に基づいて見積もったか）
- 継続的に、見積りの変更管理を行うこと

4.4.2　組織としての見積り精度向上の活動

（1）　組織的な取り組みの重要性

組織としての見積り活動を図 4.8 に示します。

ユーザ側とベンダ側の共通の事項として、確実な見積りを行えるようにするためには、見積り時に過去のプロジェクトの傾向を参考にすることが

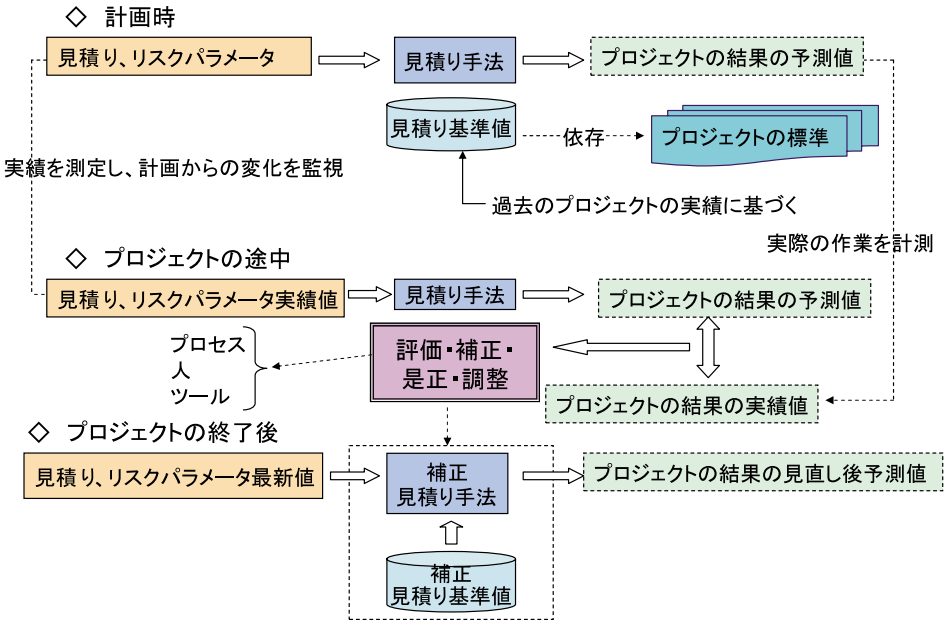


図 4.8　組織的な見積り精度向上活動の例

有効です。見積り基準値は、組織で蓄積したプロジェクトの実績データに基づいて決定します。そのためには、組織で実施したプロジェクトの実績データを蓄積しておくことが望まれます。データの蓄積方法は、組織ごとに蓄積数や使用形態を考慮して決めればよいものです。たとえば、スプレッドシートでも、データベースを整備してもよいでしょう。どの程度の数のデータを収集するかというスケールに依存します。

実績データを蓄積することにより、計画時の見積り値とプロジェクト終了時の実績値の差異分析が後で可能になり、見積り精度の継続的な向上が期待できます。また、後続のプロジェクトでの見積りで参考にすることもでき、再現可能な見積りとなります。

プロジェクトの実績データは、必ずプロジェクトマネジメントによる操

作が行われた結果のものであるため、プロジェクトマネジメント能力の異なる企業や、組織をまたがったデータを収集するよりも、自社で経年的にデータを蓄積することが、精度という観点から最も効果的です。

組織内で類似の実績データがない場合には、「次善の策」として外部で公表されているデータを参照する方法もあります。（例：参考文献などを参照）ただし、自組織で収集したデータの場合に比べて、分析の根拠データとする際には注意が必要です。あくまで最初のベースラインとして利用し、自社のデータが蓄積されたら置き換えることを推奨します。

いずれにしても、見積り基準値も見積り手法も絶対ではありませんから、プロジェクトの途中に見積りに関連する構成要素の変化を監視し、プロジェクトの実績を見積り手法に基づき評価します。必要であれば、見積り基準値および見積りの見直し（前提の再確認、見積り方法のチェックなど）をすればプロジェクトの早い段階で見積りの誤りなどに気づき、今後のプロジェクトの結果を予測してプロジェクトをコントロールすることができます。

（２） 見積りの教育・訓練の重要性

見積りは、見積り手法とその利用法についての知識や経験が必要なので、誰にでも容易にできるものではなく、特にプロジェクトマネージャを中心とした開発の実務者に、教育・訓練を行う必要があります。組織として、見積りのエキスパートを育成することを意識する必要があります。これはソフトウェア開発の見積りは、単純な式では示せないことが背景にあります。まだ、「有識者の経験や勘に頼る」の部分は残っており、いかにこれを再現性のあるものにするかが課題だからです。また、見積り手法を効果的に利用するための見積りプロセスの改善が必要です。

しかし、必ずしも内部に見積りの専任の担当者を擁することが必須ではなく、外部のサポートを受けることができるのであれば、外部力を活用するのがよいでしょう。

4.5　複数見積りによる相互チェック

（1）　複数見積りの併用

ソフトウェア開発プロジェクトの見積り方法には従来から数多くの方法が提案され、各所で実際に採用されていますが、それぞれに一長一短があり絶対的なものはありません。これは現状の見積り手法の大きな特徴であり、すべての場合に当てはまる手法はないと考えるべきです。ソフトウェア工学全般にいわれる「銀の弾丸はない」（No Silver Bullet）の一例です。

このことを念頭に、それぞれの見積り方法の長所と短所を把握して、複数の見積り方法を組み合わせて使用することが重要です。

また、見積りの信頼性は、見積り時点で得られる情報の正確さに依存します。ソフトウェア開発の初期段階では、ソフトウェアの規模は予測精度が悪く、開発環境、開発言語などのプロジェクトの特性や必要とする要員のスキルも明白ではありません。このような状況でも見積りを行う必要があるため、見積りには常に不確実さを伴います。そのため、見積りの前提が変われば見積り内容も変わる宿命にあります。不確定要素は、リスクとして明確にし、プロジェクトの進捗に従って確度の増した情報に基づき、見積りを繰り返すことが必須です。

複数の見積り結果に違いがあれば、その原因を追究する必要があることから、結果的に深く分析することになるという効果があります。また、差異があった場合に、その原因を探すことで、見落としていた項目を見つけ出す例も多くみられます。

（2）　相互チェックの例

よく用いられる方法は、次に示すような3種類すべて、または2種類の方法で見積もった結果を比較・分析し、見積りの妥当性を検証するというものです。また、この結果は、第三者に見積りの根拠を説明するときにも

役だちます。

ユーザや経営者に対して、一貫した見積りの根拠を説明する観点からは、比較的客観性の高い定量的な見積り方法（以下の例では、③のパラメトリック法が該当する）を使うことが望ましいと考えられます。

① 類推法

過去の類似プロジェクトの実績を基礎に見積もる方法

- 初期の見積りに用いる。
- 過去のプロジェクトの実施者が見積もることが望ましい。
- 実績のデータベースが必要である。
- 過去の実績について制約や技法が明らかでないと適用は困難である。

② 積み上げ法

プロジェクトで必要となる作業を洗い出し、その作業ごとの工数を見積もって積み上げる方法。

- 作業項目の洗い出しの精度で見積り精度が変わる。
- 事前に開発工程の作業を WBS（Work Breakdown Structure）で洗い出しておく。

③ パラメトリック法（アルゴリズム的）

例えば、開発工数を目的変数とし、工数要因を説明変数として、数学的関数で表される方法。（代表的な例：COCOMO（Constructive Cost Model）法）

- 再現性があり、客観的である。
- 過去の経験値が定式化されている。
- 条件を変えることによりシミュレーションが可能である。
- 不確定な工数要因を入力すると見積りの誤差は大きい。

なお、WBS による積上げは、比較的類似のシステム開発を繰り返してい

るような組織や十分に作業項目をブレークダウンできている場合には、最も精度の高い方法だといえます。この場合は、他の結果は、積上げの結果を念のため確認するという位置づけになるでしょう。

4.6　計画と実績の差異分析

見積りの精度を上げるためには計画値と実績値の対比が必須で、特にプロジェクト終了時に差異分析を行う必要があります。得てして、プロジェクト終了時に学ぶべきものを整理して次の糧とすることの重要性を忘れ、先のプロジェクトへ進むことが多いですが、見積りでも同じことがいえます。

「ずれ」があれば、差異を原因分析することで、改善のきっかけとすることができます。失敗から学ぶことは、同じ過ちを繰り返さないために重要です。例えば、工数増大などのずれた背景を探ることで、プロジェクトマネジメントの失敗の原因、レビュー体制の不備やテスト体制の不備などが明らかにできます。

さらに、得られた分析結果は、現場へフィードバックすることにより、定量的な経験値の共有を進めることができます。

- 見積り手法へのフィードバック
  - 例1）　現在、組織内で推奨している見積り方法に欠陥があるのではないか。
  - 例2）　見積り方法が最近のシステム開発に合わなくなってきたのではないか。
  - 例3）　パラメトリック法では、見積りベースラインと変動パラメータに無理はないか。
- プロジェクトマネジメント能力の改善
  - 例1）　見積りは妥当だが、プロジェクトマネジメントに失敗した。
  - 例2）　当初の条件が、プロジェクトの進行に伴って、変わっていった

（メンバの変更、その他）。

例３） 要件の膨張を抑えられなかった。

例４） 時間がなく、見積りの条件の検討不足、など。

4.7 体制・役割分担・企業文化

（１） 組織の成熟度と見積り手法

これまでも繰り返し述べたとおり、見積り手法には、さまざまなものがありますが、それぞれには、その方法を適用するのに適した条件があります。また、組織の成熟度に応じて、利用できる見積り手法も異なり、組織の成熟度に見合った見積りを行うことが重要です。

見積りにおける組織の成熟度を判断する基準として、見積りに活用できる実績データの蓄積状況と、そのデータの活用レベルがあげられます。見積りに関して組織の成熟度を向上させるには、次のことが重要です。

- ソフトウェア開発プロジェクトが完了した時点で、実績値を蓄積し、次回の類似したプロジェクトの見積りに活用できること。
- 組織としての見積り手順が確立していて、その手順が再現でき（繰り返すことができ）、見積り結果を客観的に評価できること。

初期の段階においては、実績データの蓄積が十分でなく、見積りに十分活用できない状況にあるでしょう。成熟度初期の見積りは、個人的な経験や知識に依存する面が大きく、プロジェクトレベルや組織レベルで、その妥当性の客観的評価が困難な段階といえます。

しかし、この段階においても、実績データの更なる蓄積を推進し、高度に統計的、定量的な見積り手法でなくても、組織にとって実用的な水準の簡易的手法の活用に努めることは可能です。

成熟度の高い組織は、実績データが十分に蓄積され、組織レベルで活用できる状況にあるでしょう。実績データの定量的な分析に基づく見積り手

法を活用して、組織として理解および評価が可能となります。また、見積りと実績の乖離原因を分析することにより、データの蓄積方法や見積り手順を見直すことによって、継続的に改善することができます（PDCA(Plan Do Check Action）サイクル)。

（2） 適切な見積り方法の活用ができること

組織の成熟度に適した見積りを行うには、次のことが重要です。

- さまざまな見積り方法の中から、自組織のレベルに適した見積り方法を選択できること。
- 選択した見積り方法を使いこなせること。
- 見積もった結果を適切に判断できること。

このような状況を可能にし、組織に適した見積り方法（複数の手法の組み合わせの場合も含む）を組織内で定着させるためには、次のことが必要です。

- 実績データの蓄積および活用方法、見積り手法を組織内に普及・定着させるためのバックアップ体制の整備。体制の整備は経営層の重要な役割です。
- 見積り手法に関する教育の実施。実績データの活用方法の指導・支援や教育のための専門部隊の設置が望まれます。
- ひとつの手法に固執することなく、常に課題を認識し、継続的に手法を見直し改善してゆくプロセスを確立できる企業文化の醸成。継続性・持続性がかぎとなります。

（3） 経営層のコミットメント

以上のように、見積り精度の向上のためには、組織的な取り組みが必要です。これは、プロセス改善の一環であり、見積りの成功・向上の実現には、経営層のコミットメントが、最終的なかぎとなります。

分析体制、プロセスの確立、普及に当たってのキーマンへのバックアップ、必要な予算の確保など、組織としての取り組みを積極的に推進してい

くことで、すべての出発点である見積りを成功に導くことができます。

成熟度に応じた手法を、始められるところから始めるべき

第 5 章　見積りガイドラインに向けて

「はじめに」で述べたとおり、本小冊子に続き「見積りガイドライン」を発行する予定です。「見積りガイドライン」では、見積り手法にさまざまなものがあることの紹介とともに、それぞれの効果を引き出すための条件（対象分野、組織の状況、ビジネス環境など）を解説し、第 4 章で示した重要ポイントを具体的に進めるための方法を示します。

5.1　見積り手法の概要

自社向けの見積り手法を導入・策定する例として、規模および工数の見積りを考えてみます。

（1）　規模の見積りのためのデータ例

規模の見積りのためのデータは、数多く提案され、利用されています。どれを選択するかは、システムの分野や見積り時期に応じて変わりますが、客観的に確認できるデータであること、ソフトウェア開発のコストとの相関関係があることが実証されていること、実績が計測できることなどを基準に、自社にあった適切なものを選択します。

- Web 系なら、画面上のオブジェクト数（ボタン、入出力）や画面数
- データ中心ならデータ項目数
- プログラミング言語が一定ならソースコード行数（SLOC）
- プログラミング言語などその作り方によらず、機能性を重視するなら

ファンクションポイント（FP）

- ドキュメントならページ数または文字数

（2） 生産性の基準

規模から工数を予測するためのベースラインとなる値を設定します。これは、過去のプロジェクトの実績から求める必要があります(注3)。

- 時間／規模、人月／規模
- 円／規模

（3） 変動要因

変動要因についても同様に、システムの分野やプロジェクトの特徴に応じて、数多く提案されています。表5.1にその一例を示します。各企業・組織では、このリストなどに基づいて、見積りに影響を及ぼす要素を選択・決定する必要があります。

5.2 見積り方法の策定

見積り方法の策定には、組織の成熟度に応じて、さまざまな方法があります。具体的な作成方法は、「見積りガイドライン」に詳細を示す予定ですが、例えば、次のようにデータがある場合、データがない場合のいずれにおいても方法があります。

- プロジェクトデータがない場合
  - ・有識者の経験をモデル化して、可視化する方法があります。

(注3) すでに述べたとおり、実績が整備されていない場合は、業界の一般的なものを第1次近似として採用できますが、あくまで第1次近似として、将来変更されるものと考えるべきです。

表 5.1　変動要因の例

要因グループ例	要　因　例
人的要因	経験のレベル
	コミュニケーションとチームのスキル
	プロジェクト人員の継続性
	チームの士気
	プロジェクトリーダのスキル
	プロジェクトの目標の一貫性
プロダクト要因	信頼性の重要性
	要求の不安定さ
	データベースの規模
	ソフトウェアの複雑さ
	ユーザビリティの重要性
	ソフトウェア性能の重要性
	ソフトウェア保守性の重要性
	ソフトウェア移植性の重要性
プロセス要因	顧客の参加度合い
	要求管理の統制度合い
	構成管理の統制度合い
	レビューやインスペクションの実現度合い
	品質保証の度合い
	自動化の程度
	テストの質
プロジェクト要因	開発スケジュールの制約
	作業環境
	開発場所の分散度合い
	チーム内の役割分担や責任の明確さ
	関係者の数

たとえ限られた特性および数のプロジェクト実績であっても、プロジェクトの実績と有識者の経験とを組み合わせて、モデル化して可視化する方法があります(注4)。

異なる特性のプロジェクトの結果を予測する場合、モデルは仮説にすぎずリスクが当然ありますが、予定と実績の差異を監視し、差異原因を追及することにより、モデルを順次調整し、精度を高めていくことができます。

● プロジェクトデータがある場合

・データがある場合の手法としては、統計的な分析方法を始め、データマイニング的な方法など、数多く提案されています。

・これらすでに存在する見積り手法から、自社のデータの特徴にあった見積り手法を作成する方法があります。

具体的には、文献などで提案されている見積り手法を参考にして、自社の過去データに基づき分析する方法です。例えば、COCOMO（Constructive Cost Model）法は広く知られた方法ですが、書籍などで示された要因や数値を自社の実績データベースを分析してカスタマイズすれば、精度を高めることができます。

また、データ処理中心のシステムでは、特定のデータ項目に着目し、そのデータ項目数と最終的な規模や工数などの傾向を過去の実績から設定しておき、簡便に見積もる方法があります。

(注4) 経験豊富なプロジェクトリーダや品質保証の担当者を集めて、見積りに影響を及ぼす要因を洗い出し、その影響度合いをブレーンストーミング的に決定する方法があります。

5.3　見積り手法策定・導入での観点

さまざまな見積り手法がありますが、現場に定着させるためには、以下に示す視点からの検討が必要です。見積り手法を選択する場合、最初に精度を基準に考えてしまうことが多いですが、実際にはわかりやすさ、繰り返しの容易さといった使う側の立場から見て納得できるものからの導入が基本です。精度の向上は、現場のニーズが高い場合に、手法を洗練化させていくことで対応します。

- ユーザ／ベンダ（管理者、開発者）が理解でき、同一認識に立ち議論できること。
- 現場（管理者、開発者、ユーザ）に受け入れられ、継続的に利用可能なものであること。
- モデルを利用して、リスクの識別、規模、生産性、コストの変動要因の識別などを行うとともに、リスク対策や規模の削減策、生産性の向上対策などを見積りに組み入れられること。
- プロジェクトをコントロールするために、プロジェクトの進行中に、見積りと実績との差異を早い段階で判別するための指標を生成できること。
- 見積り精度が確保できること（例えば、±10%以下程度の誤差）。

過去30年間の各方面での研究および実践の蓄積により、さまざまな方法が提案されており、適切な使い方をすることで、見積りの精度を高めていくことができます。

第 6 章　見積り手法の課題と今後のチャレンジ

これまで、かなりオーソドックスな開発を基本に念頭において見積りの解説を進めてきました。従来の見積り手法の多くは、規模を第一の入力とし、その他のコストに効く要因によって調整して、工数や工期を算出することを基本にしています。

しかし、従来の見積り手法では、仕様が決まっていれば、機能概要、データ項目、帳票・画面数などから、規模を推定することができますが、最近のビジネス環境では、仕様をあらかじめ固定化することは困難なのが実情です。ましてや、コード行数という設計や、プログラミングに依存した規模の指標を見積りで使うことには、むりが生じる場合もあります。この意味で、伝統的な見積り手法には、ウォータフォール型の伝統的な開発パラダイムに従っているという原理的な限界が存在します。

現実的には、アジャイル、インクリメンタル、エボリューショナル、スパイラルといった新しい開発パラダイムも台頭してきており、見積りの方法も、状況に応じて使い分ける必要が生じています。開発パラダイムが多様化することは、それぞれの立場で着目する視点、アクティビティ、手順、基準も異なるということであり、見積り手法の入力、コスト要因、出力項目、計算の方法も工夫しなくてはなりません。むろん、見積り手法の利用の場面も異なります。

この部分は、見積り手法において、未解決の分野といってもよいでしょ

う。しかし、現実には対応をしなければ、ビジネスができない状況になっています。

例えば、比較的小規模な案件で変化への対応を標榜したアジャイル型の開発では、仕様はあらかじめ固定化できないことを前提にしています。従来の手法でのテストも、でき上がったプログラムの検証だけではなく、テストファーストといった場面では、仕様化の局面も持ち合わせています。ユーザとの関係性も非常に緊密であり、仕様化や確認に関するコミュニケーション量や、チーム力が最も生産性に効く要因になっています。

アジャイル型を含め最近の開発パラダイムで多く採用されているインクリメンタル（段階的拡充）手法では、段階的に実行可能なプログラムを拡充していく方法を採っています。各段階の期間は、数日から数週間と短いのが通例です。こうすることによって、仕様変動のリスク、技術的実現性のリスクを回避することができます。段階を重ねるごとに生産性は向上していき、製品やサービスのリリースという位置づけでのプロジェクトの完了も、市場や経営上の視点から決定されることが多くなります。こういった場合には、プロジェクト期間をあらかじめ見積り手法の入力として設定することさえできない状況が生まれます。

定量的手法に基づく科学的見積り手法は、開発パラダイム、契約の方法を含めた社会的な要請や産業構造の変化によって、常に新しい見積り手法が必要になり、発展し、洗練化していくものです。

今後、SEC では、IT 産業のニーズを汲みつつ、これらの課題の整理と課題に対する解決策を提供していくためのチャレンジを続けます。

付　録　ソフトウェア・エンジニアリング・センター(SEC)の活動

現在、SEC では、ソフトウェア開発における課題に対応するために、産業分野として、エンタプライズ系と組込み系の二つのタスクフォースを組んでいます。

タスクフォースは、SEC の研究員と企業・大学などの有識者から構成され、それぞれの課題の解決策を研究しています。この分類は、エンタプライズ系が、長年ソフトウェア工学的なアプローチに取り組んできたビジネスアプリケーション開発における課題の解決に向けたものであり、組込み系が、急激に進展している組込みソフトウェア開発における課題の解決に向けたものであるという特徴の違いに基づいています。それぞれのミッションは、付表１に示すとおりです。

特に、エンタプライズ系領域の各テーマは、付図１に示すとおり５つについて取り組んでいます。

小冊子としては、付表２のものが予定されています。

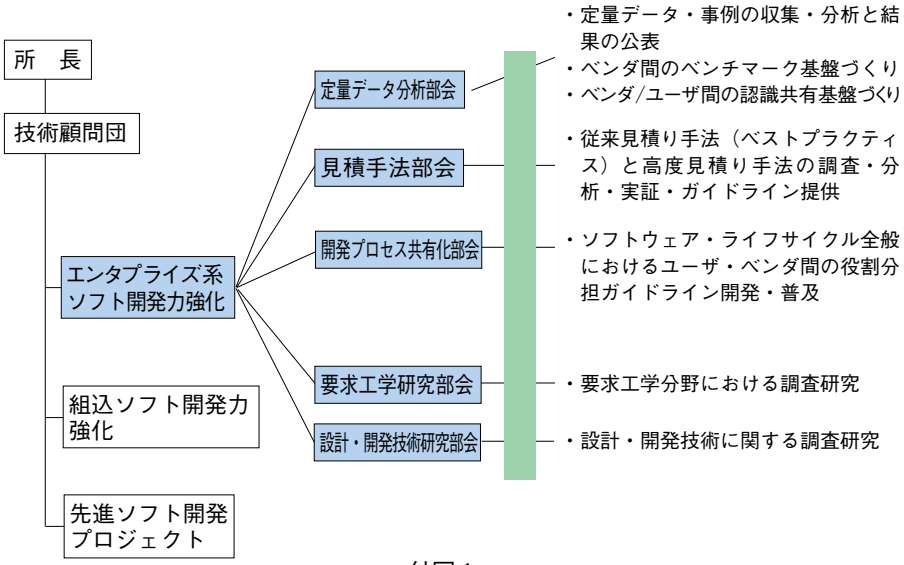
これらの情報は、SEC のホームページ上で公開されています。詳細と最新情報については、

http : //www.ipa.go.jp/software/sec/about/index.php

をご覧ください。

付表 1

分　野	ミッション
エンタプライズ系タスクフォース	ソフトウェアベンダやユーザなどのすべての関係者（ステークホルダ）が連携し、ソフトウェアエンジニアリング手法の開発・普及を進め、ソフトウェア開発のリスクを低減し、生産性の向上（効率化）を目指しています。
組込みソフトウェア系タスクフォース	組込みソフトウェア産業の実態やニーズを反映し、高品質な組込みソフトウェアを効率的に開発するためのエンジニアリング手法の開発・整備・普及を目指しています。また、不足している組込みソフトウェアの開発人材の育成と活用のもととなるスキル標準の策定と具体化を図ります。
実践タスクフォース	高度 IT 社会の基盤を構築するソフトウェアを対象に、エンジニアリング手法を実践します。具体的ターゲットとして、当初は、先進交通システムを実現するプラットフォームソフトウェア（ミドルウェア）の開発を実施します。



付図 1

付表 2

	タ　イ　ト　ル
1	IT ユーザとベンダのための定量的見積りの勧め（本小冊子） ～見積り精度を向上する重要ポイント～
2	経営者が参画する要求品質の確保 ～超上流から攻める IT 化の勘どころ～
3	組込みソフトウェア開発におけるプロジェクトマネジメント導入の勧め
4	組込みソフトウェア開発における品質向上の勧め（コーディング編）
5	組込みスキル標準　ETSS 概説書（2005年版）

参考文献

（1）JIS X 0135-1：1999 ソフトウェア測定－機能規模測定－ 第1部：概念の定義（ISO/IEC 14143－1：1998），日本規格協会

（2）松本吉弘（翻訳）：ソフトウェアエンジニアリング基礎知識体系 －SWEBOK－，IEEE，オーム社，2003

（3）Barry Boehm, A.W. Brown, S. Chulani, B.K. Clark, E. Horowitz, R. Madachy, D. Reifer and B. Steece：Software Cost Estimation with COCOMO II，Prentice-Hall，2000

（4）高橋宗雄：クライアントサーバシステム開発の工数見積り技法～工数見積りモデルの適用法～，ソフト・リサーチ・センター，1998

（5）前川徹：ソフトウェア最前線，アスペクト，2004

（6）ロバート・L・グラス：ソフトウェア開発55の真実と10のウソ，日経BP社，2004

（7）デービッド・ガーマスら：ファンクションポイントの計測と分析，ピアソン・エデュケーション，2002

（8）FAR（Federal Acquisition Regulation） http：//www.arnet.gov/far

＜見積り時の参考データ＞

（1）ソフトウェア・エンジニアリング・センター：ソフトウェア定量データ分析白書2005，日経BP社，2005

（2）（財）経済調査会：ソフトウェア開発費積算に関する調査研究

（3）ISBSG：ISBSG BENCHMARK，2004

（4）Capers Jones：ソフトウェア開発の定量化手法，共立出版，1998

○**執筆者**（敬称略）

栗野　憲一　富士通株式会社

石谷　　靖　ソフトウェア・エンジニアリング・センター（株式会社三菱総合研究所）

井上　智史　TIS 株式会社

太田　忠雄　株式会社ジャステック

大槻　　繁　株式会社一（いち）

尾股　達也　社団法人情報サービス産業協会

加藤　允基　株式会社日立システムアンドサービス

菊地奈穂美　ソフトウェア・エンジニアリング・センター（沖電気工業株式会社）

楠本　真二　大阪大学

合田　治彦　富士通株式会社

榊原　　彰　日本アイ・ビー・エム株式会社

高橋　宗雄　桐蔭横浜大学

角田　千晴　社団法人日本情報システム・ユーザー協会

谷田　耕救　株式会社日立製作所

中元　秀明　株式会社野村総合研究所

庭野　幸夫　株式会社ジャステック

服部　克己　日本ユニシス株式会社

堀　　明広　ソフトウェア技術者ネットワーク

向井　　清　住商情報システム株式会社

安田　　守　ソフトウェア・エンジニアリング・センター（株式会社野村総合研究所）

横山　健次　ソフトウェア・エンジニアリング・センター（株式会社野村総合研究所）

祝谷　和宏　経済産業省

風間　博之／上原　　智　経済産業省

編 者 紹 介

独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センター
2004年10月に独立行政法人 情報処理推進機構（IPA）内に設立されたソフトウェア・エンジニアリング・センター（SEC）は、エンタプライズ系ソフトウェアと組込みソフトウェアの開発力強化に取り組むとともに、その成果を実践・検証するための実践ソフトウェア開発プロジェクトを産学官の枠組みを越えて展開する。

［所在地］ 〒113-6591 東京都文京区本駒込2-28-8
文京グリーンコート センターオフィス
電話 03-5978-7543，FAX 03-5978-7517
http://www.ipa.go.jp/software/sec/index.php

• **本書の内容に関する質問**は，オーム社雑誌部「(書名を明記)」係宛，書状またはFAX（03-3293-6889）にてお願いします．お受けできる質問は本書で紹介した内容に限らせていただきます．なお，電話での質問にはお答えできませんので，あらかじめご了承ください．
• 万一，落丁・乱丁の場合は，送料当社負担でお取替えいたします．当社販売管理部宛お送りください．
• **本書の一部の複写複製を希望される場合**は，本書扉裏を参照してください．

SEC BOOKS
IT ユーザとベンダのための定量的見積りの勧め
～見積り精度を向上する重要ポイント～

平成 17 年 4 月 30 日　第 1 版第 1 刷発行
平成 17 年 7 月 25 日　第 1 版第 2 刷発行

編　者　独立行政法人 情報処理推進機構
ソフトウェア・エンジニアリング・センター
発 行 者　佐 藤 政 次
発 行 所　株式会社 オ ー ム 社
郵便番号　101-8460
東京都千代田区神田錦町 3-1
電 話 03（3233）0641（代表）
URL http://www.ohmsha.co.jp/

© 独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センター *2005*

印刷・製本　報 光 社
ISBN 4-274-50026-8　Printed in Japan

オーム社/雑誌局

ISBN4-274-50026-8

C3055 ¥300E

9784274500268

1923055003002

定価（本体300円【税別】）

独立行政法人 情報処理推進機構

IPA Software Engineering Center

SEC−TN04−011

PRINTED WITH SOY INK
Trademark of American Soybean Association
この印刷物は大豆インキを使用しています。

R100
100%リサイクル用紙を使用しています