SEC BOOKS

ソフトウェア改良開発見積りガイドブック

～既存システムがある場合の開発～

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

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)は，定量的な見積りに関する重要ポイントのまとめとして，小冊子「**ITユーザとベンダのための定量的見積りの勧め**」(以下「**見積り小冊子**」と略記)を2005年４月に発刊し，さらに，見積り小冊子を受けて，システム開発プロジェクトにおけるソフトウェア開発の見積りに焦点を当て，具体的な方法およびノウハウを紹介するものとして2006年４月に「**ソフトウェア開発見積りガイドブック**」を発刊しました。

「ソフトウェア開発見積りガイドブック」の中では，残された課題として「ライフサイクルコストの見積り」と「アジャイル開発における見積り」を示しています。

本ガイドブックは，この二つの課題のうち「ライフサイクルコストの見積り」に関連して，特に，ライフサイクルコストのうち「改良開発」を中心に解説を行います。ソフトウェアのライフサイクルには，最初のシステムが構築された後，ビジネスなどの外部環境の変化に応じて，機能が追加されたり，修正されたりします。そのようなシステムが進化していくために必要な開発を「**改良開発**」と本ガイドブックでは呼んでいます。

第１部では，「改良開発の見積り」に関する基本的な考え方や心得的な内容を実際の活動につなげ，さらに向上を図るための方法について示しています。読者対象としては**表１**に示すA)～D)のすべての方を想定しています。ソフトウェア開発プロジェクトのマネジメント・運営を行う観点から示されている事項も多く，ユーザ企業にとっては，やや詳細すぎる印象もあるかもしれませんが，ライフサイクルコストをコントロールする観点から，自社の仕組みを見直すた

表1　想 定 読 者

A）ベンダ企業のプロジェクトマネージャ
B）ベンダ企業の社内改善メンバ(企画メンバ)
C）ユーザ企業の契約担当者
ベンダ企業の営業担当者
D）ユーザ企業のトップマネジメント
プロジェクトマネージャ
システム部門のメンバ
社内改善メンバ(企画メンバ)

めの参考にしていただきたいと考えています。

第2部では「改良開発の見積り」の場合の見積り活動の事例集として，見積りの具体的な手法(実際に企業で実践されている見積り手法)を紹介し，見積り精度向上に向けての各社の取り組み事例，当該手法の導入に当たっての留意点を示します。第2部は，個別の方法に興味のある方が一つ一つの事例を独立して参照できるように構成しております。

これらは,「ソフトウェア開発見積りガイドブック」と同じ構成としています。

本ガイドブックで対象とする範囲は次のとおりです。

- システムのライフサイクル全般を視野に入れ，「改良開発」の見積りに切り込みます。そのため，ライフサイクル全般のコストに影響を及ぼす改良開発に対するコスト低減の方策についても考え方を示しています。
- 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を対象とします。ただし，本ガイドブックで示されたガイドラインは，組込みソフトウェアなど，他の分野でも十分に応用可能です。
- 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。本ガイドブックでは，特に規模，工数，工期，コストの見積りに焦点を当て，その具体的な考え方および方法を示します。品質はそれぞれの関係に影響を及ぼす要因としてとらえています。

なお，改良開発は既存のシステムがある状況で新たな機能を開発したり，既存の機能を変更したりするものです。開発という視点からは，新規開発は改良開発の既存システムがない場合と見ることができます。このことは，改良開発の見積りは新規開発の見積りを包含するもの，言い換えれば，新規開発は改良開発の特殊な場合ととらえることができます。本ガイドブックでは，改良開発と新規開発とをこのような関係でとらえて，解説を行っています。

2007年10月

見積り手法部会　一同

本ガイドブックの読み方

☆　**ベンダ企業のプロジェクトマネージャ**

第１部の１～５を通読してください。また，第２部の各事例を参照し，見積りの根拠として示されている内容を活用してください。なお，知識確認として，第１部の６をお読みください。

☆　**ベンダ企業の社内改善メンバ(企画メンバ)**

第１部の１～５を通読してください。第１部の４ならびに５を参考にして，組織的な見積り活動の成熟度向上に取り組んでください。あわせて，第２部の各事例を参照し，自組織との共通点や差異を把握して，実際の見積りモデルの作成や既存のモデルの改善を行うとともに，見積り活動の成熟度向上に役立ててください。なお，知識確認として，第１部の６をお読みください。

☆　**ユーザ企業の契約担当者，ベンダ企業の営業担当者**

最初に，第１部の4.4の箇所を読み，続いて，その背景として第１章の最初から通読してください。第２部の各事例を参照し，見積りの根拠として示されている考え方を把握してください。

☆　**ユーザ企業のトップマネジメント，プロジェクトマネージャ，システム部門のメンバ，社内改善メンバ(企画メンバ)**

第１部の１～３を読み，プロジェクトにおける見積りの基本事項を理解してください。そして，第１部４ならびに５を参考にして，自組織での見積り能力の向上方法を検討してください。また，第２部の各事例を参照し，自組織の見積りの向上に役立ててください。

(注) 本ガイドブックの事例紹介は各社で使用している情報システム用語で記述しています。

ソフトウェア開発見積りガイドブック

☆　**ソフトウェア開発見積りガイドブック**
～ITユーザとベンダにおける定量的見積りの実現～

ソフトウェア開発見積りに関する基本的な考え方を示すとともに，具体的な見積り方法・モデルおよびノウハウについて紹介しています。二つの部から構成されており，第１部が「総論」，第２部が「見積り手法の事例集」となっています。見積り手法の事例集では,8社での実例をその見積り方法にいたった背景，活用に当たっての前提条件，手法の強みのみならず弱点についても示しており，見積り手法を導入・構築しようと考えている組織に合ったものかどうかを判断できるようにしています。総論では，事例集に示した方法に基づいて，見積り手法に共通の考え方・方法を示すとともに，見積り手法を改善し，見積り精度を向上させることについても解説を行っています。

改良開発での見積りは，新規開発の見積りを包含していることから，本ガイドブックには，改良開発の見積りに適用可能な共通の事項が示されているので，ぜひ参照されることをお勧めします。本ガイドブックの本文において，このガイドブックに関連する事項は，該当する箇所を示して，参照できるようにしています。

目　　次

はじめに……目次前

第 1 部　総　　論

第 1 章　ユーザ企業・ベンダ企業における問題意識

1.1　背景 …… 3
1.2　運用・保守における見積り手法の現状と位置づけ …… 5
1.3　ライフサイクルコストの構成における本書の対象範囲 …… 6

第 2 章　改良開発とは

2.1　保守の定義・改良開発の定義 …… 9

第 3 章　改良開発の見積りに関して

3.1　改良開発での見積りの特徴・留意事項 ……11
3.1.1　改良開発における各フェーズの特徴とコスト構造 ……12
3.1.2　改良開発における見積り方法の適用 ……14
3.2　改良開発の見積りでの成功と失敗例 ……15
3.2.1　改良開発全般に関する事例 ……15
3.2.2　既存システムの調査・分析に関する事例 ……18
3.2.3　テストに関する事例 ……23
3.2.4　改良開発の成功・コスト低減の方策 ……26
3.3　改良開発の見積りの詳細 ……29
3.3.1　既存システムの調査・分析の見積り ……29

3.3.2 機能実現の見積り ……34
3.3.3 テストの見積り ……42

第 4 章 改良開発での見積り精度向上

4.1 調査・分析の重要性 ……45
4.2 見積りの前提条件のモニタリングとコントロール ……46
4.3 見積り手法と継続的な改善 ……48
4.4 契約によるリスク解消の糸口 ……50
4.4.1 見積りにおけるユーザ企業とベンダ企業の役割 ……50
4.4.2 確定していない部分の把握と認識共有 ……50
4.4.3 変動要因に関するユーザ企業とベンダ企業との調整プロセス 51
4.4.4 多段階契約の採用 ……51
4.4.5 実費償還型契約の採用 ……52

第 5 章 改良開発のコスト低減

5.1 改良開発におけるコストコントロールのアプローチ ……55
5.1.1 プロジェクトの見積り時に調整可能な変動要因 ……56
5.1.2 プロジェクトの見積り時に調整が困難な変動要因 ……59
5.2 事例に見る改良開発のコスト低減策 ……60

第 6 章 見積りに関する一般的事項と動向

6.1 改良開発に関して提案されている見積り方法 ……67
6.1.1 ファンクションポイント法における機能改良時の数え方 ……67
6.1.2 COCOMOでの保守見積り ……71
6.2 保守見積りに関する研究動向 ……78
6.2.1 論文誌，国際会議 ……78
6.2.2 保守見積りに関連する論文 ……79
6.3 見積りモデル構築のためのCoBRA法の適用 ……83
6.4 改良開発見積りの応用範囲 ……84

6.5　これからのコストモデル ……86

第 2 部　事例編

第 1 章　事例編の見方……91

第 2 章　日立製作所

2.1　取り組みの背景 ……97
2.2　見積り方法(モデル) ……98
2.2.1　母体の調査・分析の見積り手法 ……99
2.2.2　正味改造部分の設計・製造・テストの見積り手法……100
2.2.3　母体の確認テストの見積り手法……101
2.3　見積り方法の前提条件……101
2.3.1　見積り時期……101
2.3.2　見積り対象……101
2.3.3　見積り活動……101
2.3.4　併用見積り手法……102
2.3.5　体制・役割分担・企業文化……102
2.4　精度向上のための活動……103
2.4.1　差異分析，フィードバックの方法……103
2.4.2　精度向上のための具体的な方法……103
2.5　実施実績……104
2.5.1　適用分野(業種，システム・プロジェクトの特徴)……104
2.5.2　見積り時期とその精度……104
2.6　当該見積り方法の優位点と課題……105
2.6.1　当該見積り手法のウリ……105
2.6.2　不得意な分野など，利用するに当たって留意すべき点……105

第3章　ジャステック

3.1　取り組みの背景……106
3.1.1　現場の方法に至る経緯，改造開発見積りに関する問題意識…106
3.2　改造開発の見積り基本アルゴリズム……107
3.2.1　新規開発の見積り基本アルゴリズム……107
3.2.2　改造開発の特質……107
3.2.3　改造開発の見積り基本アルゴリズム……110
3.2.4　コスト比較……121
3.3　見積り方法の前提条件……123
3.3.1　見積り時期……123
3.3.2　見積り対象……123
3.3.3　併用見積り手法……123
3.3.4　見積り活動……123
3.3.5　体制・役割分担・企業文化……125
3.4　精度向上のための活動……125
3.5　実施実績……125
3.6　当該見積り方法の優位点と課題……126
3.6.1　当該見積り方法の優位点……126
3.6.2　現在の課題と今後の取り組み……127

第4章　富士通

4.1　取り組みの背景……129
4.2　見積り方法(モデル)……129
4.2.1　改造の種類と当見積り手法の対象……129
4.2.2　改造案件に対する見積りの考え方……130
4.2.3　各見積りタイミングでの見積り方法……131
4.3　見積り方法の前提条件……133
4.3.1　見積り時期……133
4.3.2　見積り対象……133

4.3.3 見積り活動……133
4.3.4 併用見積り手法……134
4.3.5 体制・役割分担・企業文化……134
4.4 精度向上のための活動……134
4.4.1 実績データの収集と分析……134
4.4.2 見積りの考え方の体系化……135
4.4.3 FS法の適用支援 ……135
4.5 実施実績……135
4.6 当該見積り方法の優位点と課題……135
4.6.1 優位点……135
4.6.2 課題……136

用語解説 ……137
参考文献 ……147
索 引 ……149

第１部
総　　論

第1章　ユーザ企業・ベンダ企業における問題意識

1.1　背　景

基幹系のシステムなど，長期にわたり稼働するシステムにおいては，最初の新規開発でのコストとともに，実際に運用段階に入った後での機能の追加，変更などのコストを含めたライフサイクルコストがIT投資の観点から重要です。そして近年，企業のIT投資における運用・保守コストの占める割合がますます増加しています。これは，ほとんどの企業において既存システムが存在しており，それとの関係がまったくない新規のシステム開発が想定しにくい状況が背景にあります。現在，一度作ってしまえば完成というシステムは考えられず，システムの稼働により，新たな機能の必要性が確認されたり，ビジネスなどの外部環境の変化に対応した機能が必要となったりするケースが当たり前になっています。多くのシステムは環境に応じて進化していくものと捉えるべきです。

「平成17年情報処理実態調査」(経済産業省)によると，各企業の情報処理関係支出における「従来システムの運用に係る支出」と「新規システム構築/世代交代に係る支出」の構成比は，全産業平均でおよそ２：１となっています(**図1.1**)。一方，「ユーザ企業IT動向調査」(日本情報システム・ユーザー協会)によると，「ITコストの削減」は毎年関心事項の上位ランクに位置しています。IT投資の削減において，運用・保守の段階におけるコストをいかにマネジメントするかが非常に重要となっています[(1)]。

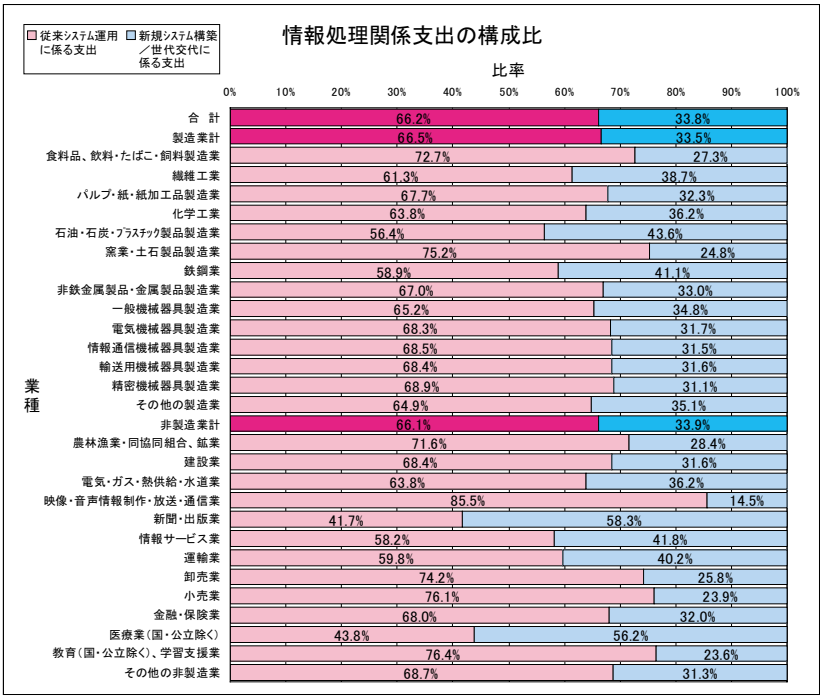
また，最近の政府調達においてライフサイクルコストが評価項目となっており[(2)]，ベンダ企業にもその根拠が求められています。運用・保守コストは，開

（１）「保守には，ソフトウェアのコストの40～80％(平均60％)がかかる。したがって，ソフトウェア・ライフサイクルの最重要フェーズである」(Robert Glass,「ソフトウェア開発55の真実と10のウソ」，日経BP社,2004年)

（２）「情報システムの調達に係る総合評価落札方式の標準ガイド(平成14年７月12日)[調達関係省庁申合せ]」では，評価基準に盛り込むべき評価項目の一つとして「ライフサイクルコストに関する項目」が挙げられています。

発後のプロセスですが，ライフサイクルコストの一部として，予算化の早い段階で見積りを求められています(3)。

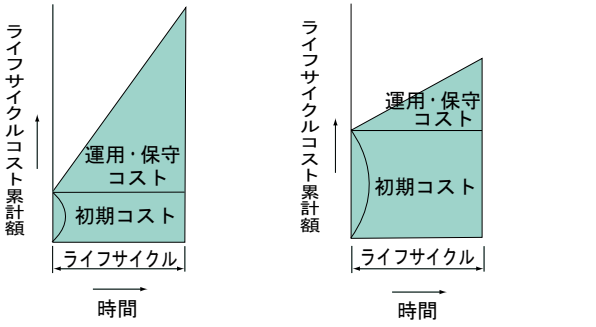
図 1.2に新規開発から運用・保守に関して，二つのコスト発生パターンを示しています。図の左側は，初期コストを抑えて，運用・保守コストが高くなったパターン，図の右側は，初期コストは大きいものの，運用・保守コストを抑えたパターンです。長い運用・保守期間があるシステムの場合，初期コストを



（出典）「平成17年情報処理実態調査」（経済産業省）に基づきSEC作成

図 1.1　産業種別でみた情報処理関係支出の構成比率（平成16年度実績）

（3）　情報システムに係る政府調達府省連絡会議検討部会は，2004年３月に「ライフサイクルコストベースでの価格評価について」，その別紙として「ライフサイクルコストベースでの価格評価の実施方法（設計段階の例）」をまとめており，運用・保守を含め各段階で発生するコストのうちどの部分がライフサイクルコストに該当するかといった定義などについて解説が行われています。



（ａ）事例ａ
運用・保守コストを考慮しなかった
初期開発の時のライフサイクルコスト
のパターン

（ｂ）事例ｂ
運用・保守コスト低減を考慮した
初期開発した時のライフサイクルコスト
のパターン

図 1.2　システムライフサイクルコストのパターン

かけても運用・保守コストが下がるのであれば，トータルで低くなる方が有利です。初期コストだけでなくライフサイクルを通じてのトータルコストが評価の対象となります。

また，同様の課題として，ある時点での見積りで，保守などにより維持し続けていくのが良いのか，それとも新規に開発しなおすのが良いのかという問いに答える必要があります。

1.2　運用・保守における見積り手法の現状と位置づけ

上記のとおり，ユーザ企業・ベンダ企業にかかわらずシステム開発の関係者の間で初期コストだけでなく，運用・保守コストをいかに見積もるのか，いかにコストをコントロールするのか，また，開発の戦略として現状のシステムを維持し続けるのか，新規に作り直すのかを判断する方法に対して，ニーズが高まっています。システムのライフサイクルに関する理論と技術が重要である反面，手法が欠落しているのが現状です。

一方，30年以上もの歴史のあるソフトウェアエンジニアリングにおいて見積り手法は大きな研究課題であり続けており，多くの知見の蓄積もあります。たとえば，新規開発における考え方は「ソフトウェア開発見積りガイドブック」で

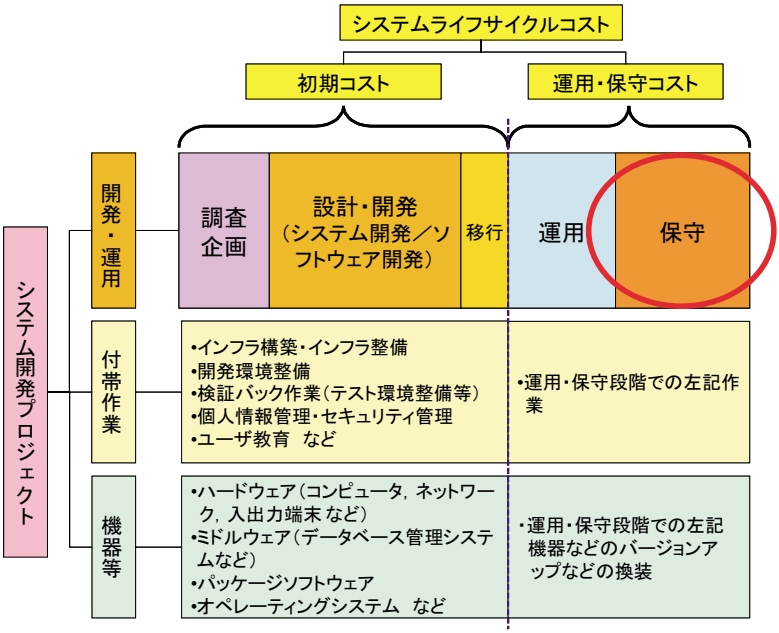
も共通的な基盤となる考え方・方法を示しました。現在，その蓄積された知見をいかに現実において活用するかということが課題になっています。保守の中でも改良開発のコストは，既存システムの規模や理解容易性，保守性の水準，および機能の変更量・追加量など，さまざまな影響要因が複雑に絡み合った構造を持っていることはほぼ共通認識となっていますが，今後体系的に捉えることが必要です。「はじめに」でも述べたとおり，新規開発は，改良開発の特殊な場合と捉えることができます。

ところで，新規開発プロジェクトと比較したとき，保守(特に，改良開発)には，すでに母体(ベースとなるシステム)が存在する特徴があります。言い換えると，保守(特に，改良開発)の見積りの問題は，既存システムがある場合の開発の見積りの問題として捉えることができます。そこで，本ガイドブックでは，改良開発を基本的な対象としながらも，広く既存システムがある場合の開発における見積りに応用できる内容はその旨を示したいと考えております。これまでの知見や現実に活用されている見積り技術に基づいて，どこまでが実践されている内容なのか，どこからが課題なのかを明確にし，どのような見積り方法が妥当なのか，さらにはコストの低減などコントロールをいかに行えばよいのかの指針を提示します。

1.3　ライフサイクルコストの構成における本書の対象範囲

システムのライフサイクルは，システムの企画段階を経て，「開発・運用・保守」(ソフトウェアの開発・運用・保守を含む)を中心として，「機器等」の調達，開発・運用を支援するものとしての「付帯作業」から構成されます(**図1.3**)。

本ガイドブックでは，保守におけるコストの見積りについて対象とします(**図1.3**の赤丸)。また，保守はさまざまな活動を含んでいますが，既存システムに対する機能変更，機能追加など，実際のソフトウェア開発が行われるものを対象とします。これはすでに述べたとおり，既存システムという元となるもの(母体)があり，それに対して必要な機能について変更，削除，追加を行うものであり，今後これを**「改良開発」**と呼びます。



（出典）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年

図 1.3　システムライフサイクルコストの構成要素

第 2 章　改良開発とは

2.1　保守の定義・改良開発の定義

保守という言葉はさまざまな使い方がされています。ソフトウェアに含まれる不具合を直すことのみを指す場合も多いですが，これは機械などの「メンテナンス」の意味合いと同じ使われ方だと思われます。しかし，ソフトウェアについては外部環境の変化に応じて機能を追加，変更することが可能であり，そのように進化していくことを保守としてとらえるように，認識が変わってきて

表 2.1　JISにおける保守の定義

タイプ		説　　明
訂正	是正保守	ソフトウェア製品の引渡し後に発見された問題を訂正するために行う受身の修正。 （備考）この修正によって，要求事項を満たすようにソフトウェア製品を修復する。
	予防保守	引渡し後のソフトウェア製品の潜在的な障害が顕在化する前に発見し，是正を行うための修正。
改良	適応保守	引渡し後，変化したまたは変化している環境において，ソフトウェア製品を使用できるように保ち続けるために実施するソフトウェア製品の修正。 （備考）適応保守は，必須運用ソフトウェア製品の運用環境変化に順応するために必要な改良を提供する。これらの変更は，環境の変化に歩調を合わせて実施する必要がある。例えば，オペレーティングシステムの更新が必要になったとき，新オペレーティングシステムに適応するためには幾つかの変更が必要な場合もある。
	完全化保守	引渡し後のソフトウェア製品の性能または保守性を改善するための修正。 （備考）完全化保守は，利用者のための改良（改善），プログラム文書の改善を提供し，ソフトウェアの性能強化，保守性などのソフトウェア属性の改善に向けての記録を提供する。

（出典）JIS X 0161：2002「ソフトウェア保守」，日本規格協会　に基づきSEC作成

います。

実際，JIS X 0161：2002「ソフトウェア保守」では，保守という用語の幅広い使われ方を反映して保守の種類が示されています。**表 2.1**に示すとおり保守対象となるソフトウェア製品への変更提案(修正依頼)の観点から四つの種類に分類され，定義されています。

本ガイドブックでは，**表 2.1**で示す保守のうち，「適応保守」および「完全化保守」において，ソフトウェアの開発を伴う場合の見積りを対象とします。そして，改めてこの二つの保守を合わせて「改良開発」と定義します。

第 3 章　改良開発の見積りに関して

3.1　改良開発での見積りの特徴・留意事項

図 3.1に改良開発における要求の発生と開発のサイクルを示します。新規のプロジェクトでは，要求から要件へ，要件に基づいて設計，実装，さらにテストなどの検証を経て，リリースされます。リリース後にシステムの運用が開始されると，ユーザ企業の利用に基づいて新たな要求が生じたり，ビジネス環境の変化や社内の組織の変化などにより既存の機能を変更する必要が生じたりします。このサイクルは，このシステムが利用される限り繰り返し続きます。

新たな要求をシステムで実現するところが改良開発に相当しますが，新規の場合との大きな違いは，母体として既存のシステムが存在することです。

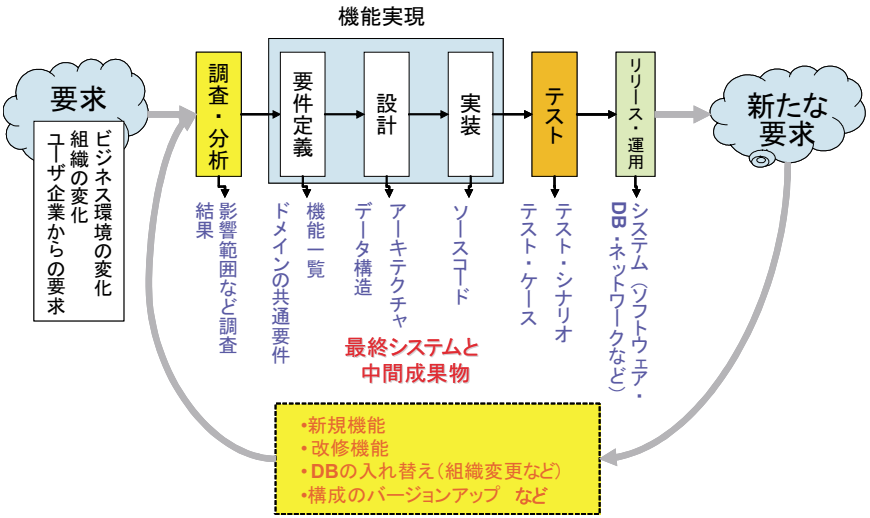


図 3.1　改良開発におけるサイクル

3.1.1　改良開発における各フェーズの特徴とコスト構造

改良開発は，次のように大きく三つのフェーズに分けることができます。

①　調査・分析

・母体となるシステムの機能，構成，アーキテクチャなどの調査および影響分析

・機能実現・テストにおけるリスクの把握および軽減対策のための調査

②　機能実現

新規に追加される機能または機能の変更の実装

（単体テストは機能実現に含む）

③　テスト

追加・変更した機能のテストおよび母体に対するテスト

①の調査・分析とは，母体となる既存システムを調査して理解し，新たな要求を機能として実現するための影響分析を行うことを指します。影響分析を行う前に，既存システムに対する理解を深めることは非常に重要です。調査・分析に要するコストに影響を及ぼす要因としては，母体の規模，母体の理解容易性，母体の保守性の水準，既存部分への習熟度，変更箇所の分散度・結合度などがあり，コスト増減の大きな要因となります。また，調査・分析では，既存システムの品質状況など，機能実現やテストにおいてリスクとなり得る要因を把握し，事前に対策などをとるための調査も重要な活動です。

③のテストも既存システムとの関連部分について留意する必要があります。テストは機能の変更や追加などを契機に行いますが，新たに追加した機能や変更した機能が既存のシステムの各所の機能と関係する度合いが高い(変更箇所の分散度・結合度が高い)場合は，テスト工数が増加するという影響もありま

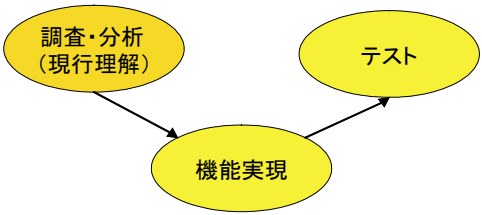


図 3.2　改良開発の主要なフェーズ

す。新規開発との大きな違いは，機能量とテスト量の相関が新規開発に比べて低い点です。端的には，たったソースコード 1 行の変更でも，そのコードが含まれる機能が母体の多くの部分に関係する場合であれば，その変更の妥当性を保証するために，関係する母体全体をテストすることがあります。

さらに，②の機能実現の部分は，ある程度まとまった機能の追加のようなケースには，新規開発の場合と似た見積りのアプローチをとることができますが，機能規模の数え方に新規開発と異なる部分があります。

コストは上記のフェーズに対応して三つの要素から構成されます。各要素がそれぞれどれくらいの割合を占めるかについては，さまざまな文献でも記述がありますが(4)，ソフトウェアの特性やその他諸条件により大きくばらつきます。**図 3. 3**は，改良開発のコスト構造の特徴的なパターンを模式的に示したものです。パターンによって機能実現の量と，調査・分析やテストにかかるコストとの間の関係が異なることが分かります。たとえば，(a)のパターンでは機能実現に比較して，母体となるシステムの調査・分析のために時間がかかり，また関係する母体の箇所が多くてテストにも時間がかかった場合を示しています。一方，(b)のパターンは，すでに経験・知識のあるシステムの場合，あまり既存システムの調査・分析には時間をかける必要がなかった場合であり，(c)のパターンは，調査・分析にも機能実現にもあまり時間をかけずに済みましたが影響範囲が大きくテストには時間がかかった場合を示しています。

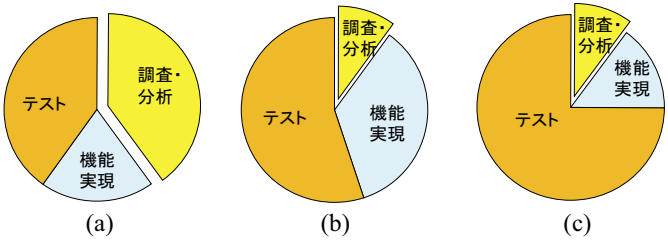


図 3. 3　改良開発におけるコスト構造のパターン例

（4）「ソフトウェアの開発作業と保守作業を分析すると，大部分は共通している。例外は，保守の場合，「調査・分析」が新たに加わることである。この作業には，保守の全作業時間の約30%が必要で，保守の中心となる作業だ。」(Robert　Glass,「ソフトウェア開発55の真実と10のウソ」，日経BP社, 2004年)

ここで重要なのは，さまざまなパターンがあるので，見積もろうとするプロジェクトがどのパターンに該当するのかをきちんと把握する必要があります。つまり，条件に応じてさまざまなパターンを示しますが，特定の組織においてある程度同じ開発条件(対象システム，顧客など)であれば，同じパターンで予測できます。

コスト構造から見ると，改良開発の場合のコストを下げるためには，既存システムの理解容易性を高める工夫や機能の追加や変更時におけるテストを容易にする工夫が重要であることがわかります。**図 1.2**「システムライフサイクルコストのパターン」における(b)のパターン(運用・保守コストが少なくなるパターン)にするためには，このような工夫を初期に行えるように投資する(コストをかける)といったことが重要になります(第5章「改良開発のコスト低減」参照)。

3.1.2　改良開発における見積り方法の適用

改良開発における見積りをシステムの構成で見た場合，**図 3.4**に示すように，OS，ミドルウェア，アプリケーションに区分し，レイヤごとに捉えることができます。それぞれのレイヤでは，アプリケーションのように製品ごとに(改良開発に際しての)既存ベース部分の有無を識別します。そしてそれぞれの製品の見積りを合算することにより，プロジェクトとしての全体の見積りを算出することができます。

アプリケーションについては,3.1.1項で示した考えに基づいて見積りを行う必要があります。そこでは，基本的に①調査・分析，②機能実現，③テストの三つに区分して見積り，積算することになります。また，それぞれについて，工数/コスト，工期の見積りに必要な情報(例：基準となるシステムまたは作業規模，影響要因，関係式)を定義して，見積り方法を確立する必要があります。

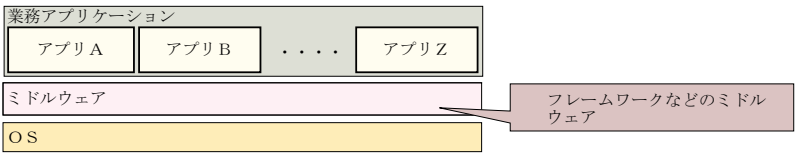


図 3.4　既存システムの有無を識別する括り

これらの詳細については，3.3節で述べます。

3.2　改良開発の見積りでの成功と失敗例

見積手法の詳細に入る前に，改良開発の見積りの特徴として，どのようなことに留意しなければならないのかを事例に基づいて紹介します。改良開発全般に関する事例と，特に開発に占める割合の大きい，調査・分析とテストに関する見積りについて示します。また，改良開発の見積りを通して見える改良開発プロジェクトの成功要因およびコスト低減に関する方策についても示します。

事例からわかる改良開発の場合の見積りの留意事項，考え方および方法については，3.2.1～3.2.3項で説明します。また，3.2.4項で改良開発において，プロジェクトの成功とコスト低減につなげる方策を説明します。

3.2.1　改良開発全般に関する事例

（1）　生産性の見積り

改良開発では既存システムがあるという点から，生産性に関して二つの側面を持っています。一つは，同じ開発者が繰り返し開発を行っていたり，保守用ドキュメントおよびチェックリストなどが十分に整備されていたり，といったことから既存システムに対する習熟度が高い場合，過去のデータ(品質状況や生産性)が蓄積されていて，これを利用できる場合，共通部品などの再利用ができる場合などは，通常の場合に比べて生産性が高くなる傾向があります。もう一つは，母体に対する影響度や母体の品質によっては，既存システムがあるゆえに生産性を下げる原因になります。プロジェクトの特性により，既存システムがあることが生産性を高くする要因にもなり，また，生産性を低くする要因にもなるので，どのような結果をもたらすのかを把握することが必須です。

事例 1 は，この状況を端的に示したものです。システムに対する習熟度が正確に把握されなかった場合，生産性，ひいては，工数/コストに対して大きな影響を及ぼします。

事例 2 は，改良開発が同じ母体での繰り返し開発であり，過去の実績データを活用しやすいメリットを生かした例です。本来は新規開発の段階で不具合を作りこまないよう開発をしっかり実施すべきですが，それが十分にできていない可能性がある場合には次善の策として有効と言えます。

また，**事例3**は，見積りを行うときに，再利用による生産性向上の効果が，どこに効くのかに留意しなければならないことを示しています。この例は，パッケージを活用した場合の例ですが，既存システムがある点から改良開発にもあてはまるものです。

事例1（失敗事例）：

課題となっている状況・状態
年度ごとにサブシステムの追加開発を繰り返している顧客システムにおいて，システム規模が大きくなり，ベンダ企業の現行部門だけでは対処しきれなくなったため，同社内の新規担当部門に一部分を社内発注した。その際現行部門の生産性実績を基準として工数・費用を見積もった。
見積りの成功/失敗の度合い
新規担当部門の生産性は，現行部門の生産性の1／3となってしまい，新規担当部門に発注した部分では，数千万円クラスの赤字が発生した。
教訓・メッセージ
生産性を設定する場合は，手元に事例や統計データがあるからと言って，安易に使用してはいけない。プロジェクトの前提が，そのデータとどの程度合致するのかを，慎重に判断する必要がある。特にシステムや利用技術に対する習熟度は，影響が大きいので注意が必要である。

事例 2（成功事例）：

課題となっている状況・状態
改良開発の工数見積りにおいて，新規開発時のサブシステムごとの不具合発生状況と発生原因を洗い直し，各サブシステムの特性を考慮して不具合発生率を想定し，手戻り工数を見積もった。
見積りの成功/失敗の度合い
全体としての工数見積精度が高まった。サブシステムごとの工数推移を継続監視することで，想定がはずれているケースに対して，速やかに再見積もりを実施できた。
教訓・メッセージ
保守開発時には，先行開発の実績データを活用して，より精緻な見積りが可能である。先行開発では，後の開発で利用するために，実績データを蓄積しておくべきである。

事例 3（失敗事例）：

課題となっている状況・状態
パッケージ利用開発において，パッケージ利用による開発効率の向上を20%と想定し，生産性を標準より1.2倍して工数を見積もった。
見積りの成功/失敗の度合い
パッケージ利用により20%の効果を発揮できるのはコーディング工程のみであり，全工程では 5 %にしか至らなかった。
教訓・メッセージ
再利用，パッケージ利用による効果は工程ごとに異なるため，工程別に生産性を考慮して工数を見積もるべきである。また，規模・生産性による工数見積りに対しては，WBS積み上げ（実タスクベースの工数見積り）による検証が必須。

（2） 見積りの変動要因の把握と変動の除去努力の必要性

上記の例のように既存システムの品質の確認を行った結果，品質が確保されていたり，ドキュメントなどがしっかり整備されている場合には，コスト増につながる影響は小さいため，見積りに当たって影響要因から見落としていたとしてもコスト増加リスクは小さくなります。一方，整備されていない場合，既存システムの品質や理解のためのドキュメントなどの状況がコストに影響する

ことを見落としていたり，精査しないで既存システムの品質や理解のためのドキュメントなどの状況を判定するなど評価を見誤ると，見積りを過少評価する結果を招きます。つまり，影響要因の判定を見誤ったりすることが，見積りにおけるリスクとなります。

既存システムの品質が悪いまたは既存システムの理解のための情報が少ない状況は，見積もる時点では変えようのない事実ですから，プロジェクトでは，これを受容して影響要因に組み入れるほかありません。逆に言えば，見積りにおけるリスク低減のためには，既存システムの品質が悪いまたは既存システムの理解のための情報が少ない状況そのものをなくすことが最も重要な対策となります。一般に，見積りに変動を与える要素に対して，あらかじめ悪い状況を排除しておく対策は，通常の開発活動の範囲を超えることになりますが，事前に対処しておくことで,コストのぶれの大きな原因を除去することができます。これについては第５章で取り扱います。

（３） 規　模

改良開発では必ず「改造ステップ数だけでは判断できない作業量（例：小規模改造で大量のテスト実施が必要）」や「既存システムの状態による影響」が発生します。見積もる際には，改良開発のフェーズ（調査・分析，機能実現およびテスト）ごとに識別する規模の定義や尺度を設定したり，あるいは，改造ステップ数などの機能実現の量以外にコストに影響する要因およびその影響度合いを可能な限り定量的に見通す（または見積もる）ことが必要です。これについては次の3.2.2項で事例を示します。

3.2.2　既存システムの調査・分析に関する事例

新たな機能の追加や既存の機能の変更をする場合，既存システムに対する影響範囲の把握がプロジェクトの全体を捉える上で必須であることは言うまでもありません。

しかしながら，実際には，改造対象の既存システムは，自社で開発したシステムである場合ばかりではなく，他社が開発したシステムである場合も少なくありません。この場合は，既存のシステムを調査・分析するための資料が整備されているか，それが理解しやすいものか，正確なものか，といった点が重要となります。また，自社が開発したシステムの保守を行う場合であっても，特

定の要員に知識が偏っていた場合には，同様のことが生じます。既存システムに対する知識は，システムのライフサイクルの間維持し続ける必要があり，必ずしもベンダ企業の努力だけでは成し得ません。この要因は改良開発プロジェクトのコストへの影響が大きいので，ユーザ企業とベンダ企業とが協力して対処することが重要です。3.2.1項で紹介した事例１のように，生産性が３倍近く違うといった事例も報告されています。

効率のよい確実な調査のためには，既存システムに関するドキュメントが確実に整備されていることと，既存システムを理解している人材の確保とが最も重要です。

また，影響範囲の調査不足も見積りの妥当性の観点から大きな問題となります。これは，調査不足は見積りを大きく狂わすリスクとして捉える必要があることを示しています。逆に言えば，見積りのぶれを減らすために，過去の知見に基づいてコストに影響する要因(品質に影響する要因も含みます)のリストを整備し，分かっていないものをできるだけ減らし，次に影響要因の有無やその影響の度合いを測るために調査を十分に行えば，判断ミスを防げることです。つまり分かっていない(unknownな)リスクを分かっている(knownな)リスクとし，見積りの対象とできるようにするとともに，knownなリスクのうち顕在化しているものを見逃がすことを防ぐようにすることです。

具体的には，見積る時点で，限られた期間とコストの制約の中で，サンプリング調査を行うことによって，見逃しを防ぐことが有効であり，この活動は妥当な見積りを得るために必要な活動です。見積り時にその状況を把握し，関係者(ユーザ企業とベンダ企業)間で共有し,妥当な見積りを行う必要があります。

以上のように調査・分析を行わなければいけない項目は多岐に亘りますが,これらの詳細については,3.3.1項でまとめて整理します。

以下に，具体的な成功事例および失敗事例を示します。

（１） 他社からの引継ぎ時でのドキュメントの整備状況の確認の必要性

事例４は，他社からの引継ぎ時に調査・分析のために必要なドキュメントなどがそろっている前提で見積りを行った結果，実際にはそうでなかったために問題となった例です。

一方，**事例５**は，調査・分析のためのドキュメントの状態をサンプリングして，状況を確認した上で，その状況に応じた生産性を設定し顧客と合意するこ

事例 4（失敗事例）：

課題となっている状況・状態
他社から引継いだ改造案件で「既存システム品質（ドキュメント整備具合）」による影響度について顧客と調整し，整備良好（メンテナンス済み）との回答よりコストへの影響はないと見積ったが，実態はドキュメントとソースのアンマッチが散見された。
見積りの成功/失敗の度合い
見積り時に改訂履歴やドキュメントの提示があったが，これは直近の改造案件のものだけであり，過去分の改造内容は正しく反映されていない状態であった。顧客担当者も知らなかったもの（前任者時代の負の遺産）で「顧客側の引継ぎ漏れ」が原因と考えるが，見積り時の当社側確認も不十分であった。
教訓・メッセージ
ドキュメント整備状態は改良開発の生産性に与える影響が大きい。実物確認時に以下の点に留意すべきである。 ① サンプリングにより，ドキュメントとソースが一致しているかチェックする。 ② チェックは過去案件までさかのぼる。

とで，見積りを妥当なものとしてプロジェクトの成功につなげた例です。

いずれの例からも，既存システムの理解度（あるいは，理解容易性）をはっきりさせて，その状況を見積り時に加味することが必要であることがわかります。

（2） 影響範囲の調査の必要性

事例 6 は多く見受けられるものですが，共通ライブラリの修正が想定以上の影響範囲があり，他の機能の障害へとつながったものです。

共通ライブラリの修正が既存システムの広い範囲に影響することは容易に理解できますが，詳細な調査を実施できていない見積りの時点で，その影響範囲をいかに想定するかが課題です。見積りにおいては実査できていない想定事項をリスクとして捉え，見積りに組み入れることになりますが，影響の程度を想定できない場合（影響の度合いを想定する根拠に乏しい場合）や想定できてもプロジェクトの正式な計画として許容できる変動幅を超える場合がままあります。このような場合は，調査・分析によって実査できた段階で見積りを見直す

事例 5（成功事例）：

課題となっている状況・状態
他社から引継いだ改造案件で，見積り時に以下の点を顧客と合意し，見積り条件とすることができた。 ① 生産性変動要素として「既存システムの解析容易性」を挙げ，見積上評価結果を明示。 ② ①の評価結果による生産性影響度を見積り時に明示。 結果として，既存システムがソースの解析が難しい状態にあったため生産性を「５％」低下させる見積りで合意。
見積りの成功/失敗の度合い
見積り時にプログラムソースやドキュメント，障害記録をサンプリングし「既存システム状況」についてチェック表を元に評価した。（「ソースコメントなし」「修正履歴なし」の２点で解析性が劣ると評価。他の評価項目については生産性への影響はなしと評価。）設計工程への影響度（５％）は他社事例として当社から提示。顧客と相談のうえ当該数字を採用することになった。
教訓・メッセージ
改良開発では「既存システムの実装状態」に依存するため，「既存システム」の状況を判定し開発（開発規模や生産性）への影響を見積りに反映すべきと考えている。「既存システムの解析容易性」については，「調査ツールの具備状況」「ソースコメントや変更履歴の有無および記載レベル」「標準化規約との整合性」などを見積り条件として評価することが望ましい。

事例 6（失敗事例）：

課題となっている状況・状態
保守開発において，ある機能の修正のためにその機能で使用している共通ライブラリ内にあるサブルーチンを修正した。
見積りの成功/失敗の度合い
調査不足により，サブルーチンの修正仕様がそのサブルーチンを使用している他の機能の仕様と不整合があることに気がつかず，障害が起こってしまった。
教訓・メッセージ
改良開発では，共通モジュールの影響範囲が不明になっている場合が多いが，影響調査を怠ると障害につながる可能性が高い。

ことをユーザ企業とベンダ企業とであらかじめ合意しておくことが重要です。

（3）　その他の事例からの教訓

表 3.1に，その他の事例から調査・分析に関して得られている教訓をまとめます。

表 3.1　調査・分析の見積り時での教訓

No.	教　　訓	対　象
1	改良開発では「既存システムの実装状態」に依存するため，「既存システム」の状況を判定し，開発（開発規模や生産性）への影響を見積りに反映すべきである。 「既存システムの解析容易性」については，「調査ツールの具備状況」，「ソースコメントや変更履歴の有無および記載レベル」，「標準化規約との整合性」などを見積り条件として評価することが望ましい。	見積りモデル・変動要因
2	見積り時点で既存システムの品質（例：障害発生頻度）を確認する作業が必要である。既存システムの品質向上が必要であれば，改造案件とは異なるタスクとして見積り，改良作業着手前に実施することが望ましい（改良案件と同時に確認を行う場合，既存バグとの識別ができないケースがある）。	見積りモデル・変動要因
3	「既存システム」の調査源はドキュメントだけではない。顧客の有識者の知識や，現行開発ベンダからNDA（Non　Disclosure Agreement）契約を締結して入手するなど，さまざまな手段を活用して調査することが，見積りの妥当性を高めることができる。	変動要因の調整方法
4	変動要因の評価は，実査して確認することで，できるだけわかっていないリスクを削減することが肝要である。 ・サンプリングによりドキュメントとソースの一致のチェック ・サンプリングチェックは，過去案件にさかのぼり行うこと	変動要因の評価方法
5	フレームワーク利用時に，フレームワークの機能を調査して，組入れなければならないロジック量を開発規模の変動要因として見積りに組入れる必要がある。	見積りモデル・変動要因
6	使用実績のないパッケージを利用する場合は，次の対策を採ることが肝要である。 ① 顧客，開発ベンダ，パッケージベンダで推進体制を作る。（例：顧客にパッケージベンダとコンサルティング契約を締結してもらうよう顧客と合意する）。 ② パッケージも機能，性能，品質で不明確な部分は，前提条件を加えて見積る，あるいは適合性を分析するフェーズを設け，適用可否，カスタマイズによる追加機能量を見積もる。 ③ パッケージの各種条件を確定させるチェックポイントを設け，再見積りすることで顧客と合意する。	見積りモデル・変動要因

3.2.3　テストに関する事例

テスト全体コストの見積りに当たって基準となる規模をどのように設定するかは重要な課題です。影響範囲が既存システムの特定の部分に局所化されているのか，それともシステム全体に分散しているのかによって，コストは大幅に変化します。

また，テストの効率(生産性)についても多くの変動要因が存在します。例えば，既存システムの品質状況は最も重要な要因の一つです。

（1）　テスト量見積りのための基準規模

単純にテスト量は改良ステップ数に比例することを期待してコストを見積もると失敗してしまいます。**事例 7** は，規模見積りにおいてテストフェーズの規模を機能実現の規模とは区別し，改良ステップ数に加えて，既存システムに対する改良箇所の分散度合いによる影響を加味して見積もることによって，テスト量の見積りを精緻化し，顧客の納得を得るとともに，妥当な見積りに成功した例です。

（2）　テスト量に影響を及ぼす要因

既存システムの残存バグは，新たな機能の追加や既存の機能を修正する場合に顕在化することがあります。このように既存システムの状況，特に品質については，事前に十分調査することが重要です。

事例 8 は，既存システムの品質調査を重視し，品質状況を見積りに反映することにより，見積りの精度向上とプロジェクトの成功につなげた事例です。一方，**事例 9** は，失敗の事例です。

3.2.2項の事例 4 ～ 6 と併せて，既存システムの状況の把握が改良開発の場合の最重要項目であることがわかります。

事例7（成功事例）：

課題となっている状況・状態
目標値として予定工数（改良規模（ソースコード行数）に係数を掛けて全工程の予定工数を算出）を提示してきた顧客に対し，以下の提案を行った。 ① 開発工程ごとに規模を見積もる。 ② 既存システムの現況から発生する影響度合いを見積もる（工程別生産性，工程別規模）。 ③ 既存システムに対する改良規模の分散度合いを判別しテスト量を見積もる。 テスト工程コストで2倍以上の開きがあったため，開発途中での再見積りを前提に作業着手した。設計終了後にテスト予定量を提示し，作業量について了承を得た。
見積りの成功/失敗の度合い
初回見積りではコストで2倍以上の開きがあり合意できなかったので，設計完了後に再度テスト予定量を提示。案件ごとに必要なテスト量（設計が済んでいるので精度が高い。改良ステップ数に依存しない）を提示して，やっとテスト予定量について了承を得た。
教訓・メッセージ
① 改良開発の作業量（コスト）は，要求事項や既存システムの状態により千差万別となる。 ② 改良開発では必ず「改良ステップ数だけでは判断できない作業量（例：小規模改良で大量のテスト実施要）」や「既存システムの状態による影響」が発生するので，見積もる際にはこれらの作業量や影響度合いを明示する（数値として提示する）ことが必要である。

事例 8（成功事例）：

課題となっている状況・状態
「既存システム品質（残存バグ率）」による影響度について顧客と調整し，品質向上の対応要否を判定し，見積りに反映した。具体的には，既存システムの品質を向上させることを目的としたテストの量を，テストの量に加えて見積もった。
見積りの成功/失敗の度合い
見積り時に「規模や生産性に影響を与える変動要素（プロジェクト固有）」について顧客と調整し合意を得ることを「見積りプロセス」として実現している。 見積り時点で改良対象の既存システム（一部）に残存バグが高い機能があることが判明していたため，品質向上の要否および具体的な作業（テスト実施）を調整することができた。
教訓・メッセージ
見積り時点で既存システムの品質を確認する作業が必要である（例：障害発生頻度）。既存システムの品質向上が必要であれば，改良案件とは異なるタスクとして見積もり，改良作業着手前に実施することが望ましい（改良案件と同時に確認を行う場合，既存バグとの識別ができないケースがある）。

事例 9（失敗事例）：

課題となっている状況・状態
テスト時発生した障害が発生しなくなった。相当量の再現の試みをするが，再現せず，本番を迎えてしまった。
見積りの成功/失敗の度合い
新たな不具合を混入させた可能性があり，なぜ発生しなくなったかの追求に多くの工数を費やしてしまった。背景として，テスト計画，構成管理，保守性（試験性）配慮が弱かったことが考えられる。
教訓・メッセージ
対策としては次のものがある。 ・ テスト設計/テスト計画の充実 ・ 構成管理の充実 ・ プログラム構造の明確化による保守性向上 ・ テストツールの活用により，タイミングの不具合も捉えやすくなる

表 3.2　テストの見積り時での教訓

No.	教　　訓	対象
1	保守においては，開発基盤の整備状況やテストツールの有無が生産性に大きく影響する。	見積りモデル・変動要因
2	改良案件固有のテスト要件(例：リグレッションテストが必須)から「テストデータの流用可否」は開発コストに大きく影響する。生産性が1.5倍異なる場合がある。	見積りモデル・変動要因
3	変動要因の評価は，実査して確認しわかっていないリスクを削減することが肝要である。「テストデータの流用可否」を次のように行う。 ① 「提供データ」の実査(効果を期待できるが，見積り時に提供データが準備されているとは限らないので,実査できない場合もある) ② 開発側ニーズの提示(データ数，データベース数) ③ 開発環境の制約確認(容量不足や他部署の影響などから流用不可となる場合もある)	変動要因の評価方法
4	見積り時点でインフラのバージョンアップの有無を確認し，インフラのバージョンアップは，改良案件とは異なるタスクとして見積り，改良作業着手前に実施することが望ましい。改良案件と同時に確認を行う場合,既存バグとの識別ができないケースがある。	見積りモデル・変動要因
5	改良開発では,テスト内容が新規開発と異なることを意識して，実装規模とは別にテスト規模を見積り,工数を算出すべきである。	見積りモデル

（3） その他の例からの教訓

その他の事例からテストに関して得られている教訓について，**表 3.2**にまとめます。

3.2.4　改良開発の成功・コスト低減の方策

改良開発の場合，既存のシステムの状況や将来の保守に備えた準備状況に応じて，プロジェクトの難易度が大きく変わることが，3.2.1～3.2.3項に紹介した事例から見えてきます。

まず，全般的な方策として,3.2.1項の事例２にありますように，改良開発の特徴として過去に同じシステムでの開発実績がある点はメリットであり，過去データを利用して見積りの精度を向上させることができます。あらかじめ実績データを収集・分析する仕組みを用意しておくことが必要です。

改良開発のコストに与える影響要因から見ると，既存システムの内容および

品質などの理解が見積りの精度に大きな影響を及ぼし，さらにはプロジェクトの成否に影響を及ぼしていることがわかります。調査・分析に影響を及ぼしている要因には，システムの特性(システムのつくりが理解容易なものとなっているか)と開発体制(過去の経験が生かせるか,または解析能力に優れているか)の二つの側面があります。

前者については，保守を意識したシステムの設計がなされたか，ドキュメントが整備され，かつ，構成管理が確実になされ実際のシステムと整合がとれたものになっているか，といった以前の開発プロジェクトでの実践状況が効いてきます。

後者については，利用者側の担当者に豊富な知識があるか，開発者側の担当者に豊富な知識があるか(新規に担当する場合でも以前の担当企業の知識の活用も含まれる)，仮に知識がない担当者の場合でもリバースエンジニアリング技術に優れたチームを担当させることができるか，という状況が影響要因となります。知識に関しては，以前の開発プロジェクトでの知識の集約方針・工夫の妥当性が効いてきます。

改良開発のメリットの例となりますが，テスト環境，テストケースなどの再利用を実現することにより，生産性，システムの品質の両者を向上させることが可能となります。将来の保守を見据えて，このあたりを整備・維持することは，改良開発の成功・コスト低減の重要な戦略の一つとなります。

一方，改良開発は，上記の要因で想定外のことが多く，見積りで想定していた条件と違っていることが後になって判明し，結果として実績と大きくぶれてしまう，ということが事例で示されています。そして多くの場合，以前のプロジェクトでの開発状況が大きな影響を及ぼしていることがわかっています。裏返せば，プロジェクトで将来を見据えて課題をつぶすようにコントロールすることにより，将来の改良開発プロジェクトを容易にし，見積りしやすくすることができるのです。将来に備えるというのは，関係者が強く認識し理解することが必要であり，また，ユーザ企業・ベンダ企業の両者で統制が必要であるため，簡単に実現できることではないのは事実です。しかし，効果の大きさ，実施しない場合の悪影響の大きさを認識すれば，対策をあらかじめ取っておくことが，見積りの成功，ひいてはプロジェクトの成功につなげる最も簡単な方法とも言えます。

表 3. 3に改良開発の成功につなげる方策の事例を示します。

表 3.3　改良開発の成功・コスト低減の事例

No.	事　　例	対　象
1	既存システムのノウハウ保有者の突発的な異動による品質低下(リリース後の障害増加)を予防するために，次の事項を実施。 ① 既存システム機能別に，その機能を理解している要員が何名いるか分布表を作成し，習熟者の偏り度合いを評価し，習熟者が少ない機能について教育計画を設定し実施している。 ② 突発的な要員異動による影響を最小限に抑えるための準備として，保守用ドキュメントの整備などを対策として実施している。	保守体制の維持
2	大量データが滞留した場合にSQLコーディングの不良が性能に与えた問題に基づき，影響を理解し，開発基準書・コーディングルールを提示することを教訓とする。 また，テストの方式も明確に提示できなかったことからの教訓として，開発時に性能測定する環境を用意し，SQLのテスト工程を設ける。	保守性の高め方
3	複数の保守案件に共通して使えるテスト環境とテスト結果の検証ツールを開発し，個別保守案件の生産性を向上させた。	保守性の高め方
4	既存システムの処理結果と改良後のシステムの結果とを比較して検証するツールを用いて検証する場合には，ツールの構成管理をしっかりしないと混乱を招く。 ソフトウェアの構成管理については，管理方式(仕組み)の有無に限らず運用方式を確認するとともに，実作業時のバージョン確認作業を徹底する必要がある。必要であれば，構成管理のミスを防止するために，顧客，ベンダ企業で2重のバージョン確認作業を実施し，混乱を防ぐ。	保守性の高め方
5	機能追加プロジェクト(新規統計システムの組み込み)で，既存システムを大幅に改良するのではなく，汎用的なデータマート(5)基盤を利用することで，大幅にコスト削減できた。ユーザ企業の要望とやや異なる形ではあるが，交渉を進め理解を得た。	代替策の利用

(5)　集中管理されたデータベースから，業務のニーズに応じて必要なデータだけを切り出して提供する機能。

3.3　改良開発の見積りの詳細

この節では，ここまで見てきた事例を項目ごとに整理します。

3.3.1　既存システムの調査・分析の見積り

（1）　既存システムの調査・分析の位置づけ

前節の事例で見たように，既存システムに対する習熟度が高ければ，調査・分析工数を小さくし生産性を高めることができますが，習熟度が低い場合は，まず既存システムを理解するための調査が必要になります。これは既存システムを改造するための知識をつけると共に，既存システムの状況を調査し，改造前の品質不良などのリスクを回避する意味もあります。

また，たとえ習熟度が高かったとしても，要件の影響範囲を決めるためには，必ず事前に調査が必要です。この調査の目的は改良開発で直接手を加える範囲の大きさ(改良開発規模)を特定するとともに，テストや業務に対する影響範囲と，その大きさを特定する目的もあります。

（2）　調査・分析とリスク対策

既存システムに対する習熟度が低い場合，既存システムの状態が分からないことが，リスクになります。例えば，既存システムの品質が悪い場合，テスト工程になってから既存の不具合が大量に発見され，不具合対応工数の大幅な増加により，見積り工数を超過してしまう事態が想定されます。

このような事態を防ぐためには，二つの方法が考えられます。一つは，既存システムに対する習熟度不足をリスクとして見積りに盛り込む方法です。具体的には，上で挙げた不具合大量発生のように，リスクが顕在化して工数が膨らむことを予想し，その分の工数を見積り工数に上乗せしたり，生産性を低く評価したりする方法です。この方法は，既存システムの状況が分からないまま，開発工程の最後までを見積もる場合に使います。この方法では，見積り結果を早い段階で得ることができますが，開発工程の全体にわたって習熟度不足のリスクを見込むため，リスクが顕在化しない場合には，過大見積りになってしまいます。過大な見積りとなることを防ぐために，見積りにあたって調査期間を設け，本来は調査・分析の結果として得られる情報を，サンプリング調査など

の手段で確認して，その結果を根拠に見積もることは有効な手段です。

もう一つの方法は，まず調査・分析を実施して，リスクである習熟度不足を解消し，その後に見積りを行う方法です。この方法では，調査・分析が終われればリスクは解消されるので，見積りの精度は上がりますが，調査が終わるまで見積り結果が得られないため，見積り時期が遅くなります。お互いの短所を補うために，二つの方法を組み合わせて使うことも，よく行われています。

（3） 調査・分析の活動内容と工数見積り

このように調査・分析をどのような目的で行うかは，既存システムへの習熟度とリスク対策のあり方によって，さまざまなバリエーションがありますが，改良開発では，必ず調査・分析を行う必要がある点では共通しています。前項で示した事例などから，調査・分析において何を実施しなければならないかがわかります。改良開発において，調査・分析をどのように進める必要があるかを**表 3.4**に示します。

作業項目のNo.1に示すとおり，最初に行わなければならないことは，既存システムに関してプロジェクトチームがどの程度の理解度を持っているかの確

表 3.4　調査・分析において必要な作業項目

No.	作業項目	概　　要
1	既存システムに対する理解度の確認	既存システムに関して，担当企業・チームがどの程度理解しているのかを把握する。
2	既存システムの理解性の確認	既存システムのドキュメントの整備状況についてサンプリング調査などで確認する。 既存システムに関する経験・実績の有無（ユーザ企業側・ベンダ企業側を問わない）
3	既存システムの品質状況	既存システムの品質（性能，信頼性，保守性など）について機能・モジュールのサンプリング調査などで確認する。
4	要件の概要	開発の目的，開発の要求内容などについては，基本的に新規開発と同様。 開発時のシステムポリシー（システム前提や標準化など）や先行する改良開発の要件定義も参考にする。
5	要求内容の既存システムへの影響分析	要件と既存システムとの関係の分析（影響分析）を実施し，影響範囲を確定する。

認です。既存システムに対して，すでに理解している場合は，調査・分析の作業は新たな機能追加や変更に対してどういった影響があるか，どのあたりに留意する必要があるか，といった点について把握が比較的容易になります。逆に，全く知識がない場合は，既存システムの理解だけでも大きな作業が必要となります(**図 3.3**)。

作業項目のNo.2，No.3に示すとおり，理解度を把握した上で，理解度が不足している場合(特にシステムの引継ぎがあった場合など)には，次の事項の調査は必須です。

① 既存システムのドキュメント整備状況

② 既存システムの品質状況

それぞれの調査は限られた期間とコストの中で行う必要があるので，サンプリングをして状況を確認し，その結果を根拠に，理解度の不足を補うために必要な作業を特定し作業量を積み上げます。これまでが，既存システムの状況の確認です。

また，開発規模がどの程度になるかについては，新規開発の場合と同様，開発の目的，開発の内容(要求内容)について定義するとともに，それらが既存システムに対してどのような影響があるかを分析します(影響分析)。

No.4，No.5の作業項目は，新規開発の場合と同様，顧客要求に基づいて要件を固めると共に，既存システムの中のどの部分を改造し，どの範囲をテストすればよいかを確定します。前節の事例で見た共通機能のように，影響が広範

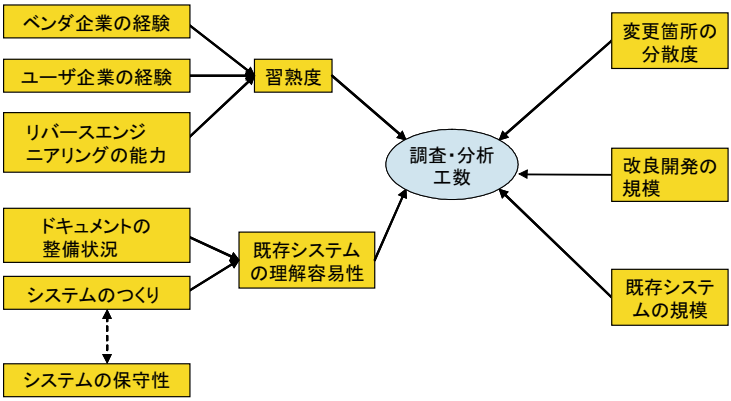


図 3.5　調査・分析工数に対する変動要因

表 3.5　調査・分析に対する変動要因(規模に影響)の例

主特性	副特性	影響の理由と評価の観点
変更箇所の結合度	——	モジュール間インタフェースの方式(ファイル連動，パラメータ連動，共有領域連動，DB連動，転送連動，マクロ)によってモジュール間結合度が異なり，調査作業に影響する。
規　模	改良開発の規模	改良開発の規模が大きいと調査対象が広がるため，調査作業を増大させる。
	既存システムの規模	既存システムの規模は調査対象の規模と一般に比例することから，調査作業を増大させる。

表 3.6　調査・分析に対する変動要因(生産性に影響)の例

主特性	副特性	影響の理由と評価の観点
習熟度	ベンダ企業の経験	既存システムの調査を実施するベンダ企業が，あらかじめ知識を有している場合，調査作業の効率化につながる。
	ユーザ企業の経験	既存システムの調査に当たってユーザ企業側が情報提供を行うことにより，調査作業の効率化につながる。
	リバースエンジニアリングの能力	既存システムの調査に当たってシステムの機能，構造などをリバースエンジニアリングする能力が高い場合は，調査作業の効率化につながる。
既存システムの理解容易性	ドキュメントの整備状況	既存システムの調査の参考情報として状況を正しく反映したドキュメントがある場合は，調査作業の効率化につながる。
	システムのつくり	既存システムのつくりが理解容易な構成(例：モジュール化が進んだもの)であれば，調査作業の効率化につながる。
	データ構造	データ構造のつくりが理解容易な構成(例：正規化)であれば，調査作業の効率化につながる。
変更箇所の分散度	——	変更箇所が分散していると，調査作業が多岐にわたり，また，変更箇所間の関係を調査するなど，調査作業の効率化を悪化させる。

囲にわたるものがある可能性があるので，これらの活動では，既存システム全体に対する影響分析が必要であり，既存システムが大きければ，それだけ工数も多くかかります。また，既存システムの構造の複雑さやドキュメントの整備状況によっても，作業工数は変わってきます。調査・分析工数に対して影響を与える要因（＝変動要因）を**図 3. 5**と**表 3. 5**および**表 3. 6**に示します。調査・分析工数の見積りでは，これらの要因を考慮し，**表 3. 5**と**表 3. 6**の活動がどの程度必要であるかを積み上げて工数を算出します。

なお，改良開発の規模（追加，削除および変更の規模）を基準にして，変更箇所の分散度，影響する既存システムの規模を見積りの変動要因に加えて，生産性を調整して見積もる方法を採用している企業（第 2 部事例編「ジャステック」を参照）もあります。

（4） 調査・分析のアウトプット

後続するフェーズ（機能実現フェーズおよびテストフェーズ）の変動要因は後述の**図 3. 9**および**図 3. 11**に記載していますが，調査・分析のアウトプットには，後続フェーズのコストに影響を及ぼす要因そのものまたは影響を評価する根拠となる情報が多く含まれます。したがって，調査・分析のアウトプットは，後続フェーズの見積りに対する重要なインプット情報となります（**図 3. 6**）。

調査・分析フェーズが終了すれば改良開発の見積りで付加した変動要因はほとんど確定するので，当該変動要因によって後続フェーズのコストが変動する確率は低くなることがわかります。

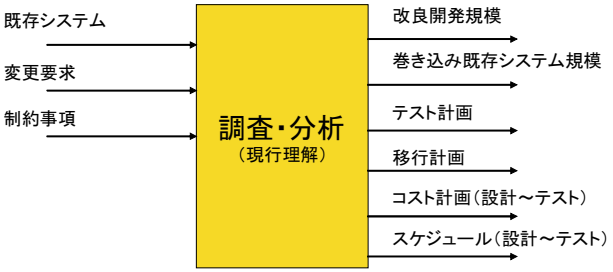


図 3. 6　調査・分析の位置づけ（入力と出力）

3.3.2　機能実現の見積り

基本的に，新規開発のときの機能実現と同じ考え方で見積りすることができます。具体的には，要件，規模，工数，工期のそれぞれの関係を過去のプロジェクト実績データなどに基づいて設定する手順には変わりはありません(6)。

一方，新規の場合とは，基本となる規模の算出方法が異なりますし，新たな変動要因が付け加わります。以下には，「ソフトウェア開発見積りガイドブック」で示している内容に基づいて概要を示します。詳細は，「ソフトウェア開発見積りガイドブック」をご参照ください。

（1）　機能実現における見積りの基本形

基本的には，要件，規模，工数，工期の間の関係をあらかじめ設定しておき，いずれかから他のものを見積もる，という手順で進めます。例えば，先に要件が決まる場合には，**図 3.7**に示すとおり「要件の洗い出し←→規模の見積り←→工数の見積り←→工期の見積り」といった手順で行うことになります。

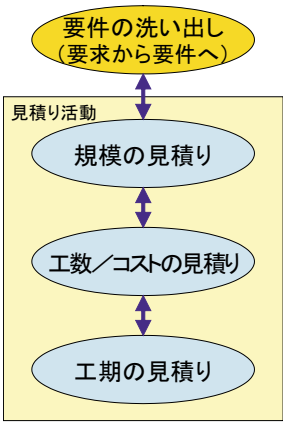


図 3.7　見積りの基本的な手順

（6）　固定されたリソースの中で，複数の案件を同時並行に進める場合のようにある期間を定めて保守する場合（「期間保守」と呼ばれる）は，一つ一つを新規開発として捉えることができますが，全体としてみると，工数配分は山型ではなく，最初から最後まで同程度の工数が配分される様相を示します。

（2） 見積りの個々の活動

図 3.8は，**図 3.7**の詳細な内容を示したものです。図に示すとおり，要件の洗い出しは，機能の要件と非機能要件(品質要件や技術要件など)を定義するこ

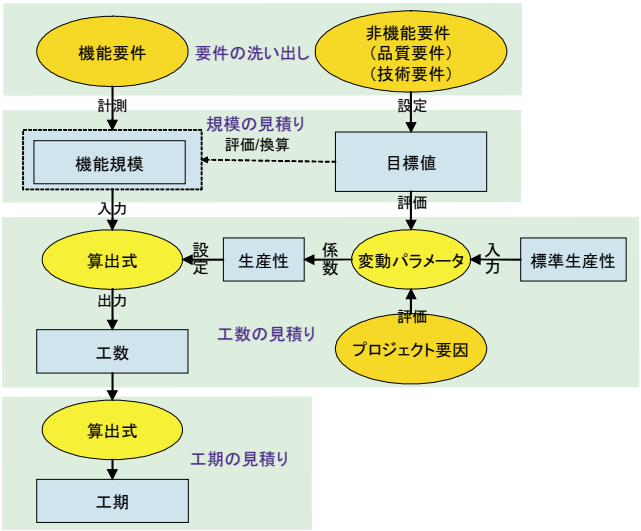


図 3.8　個々の見積りの基本手順のブレークダウン

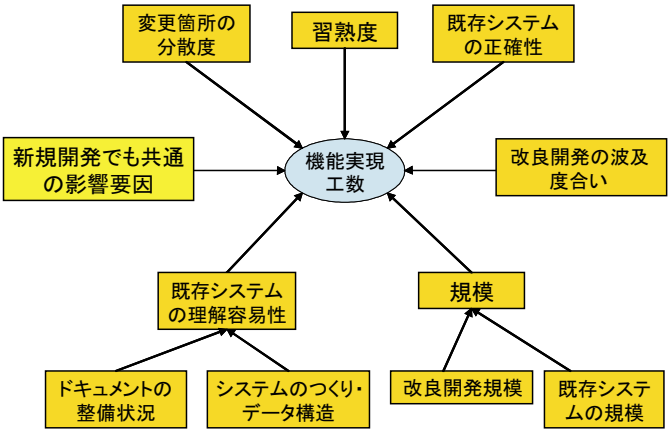


図 3.9　機能実現の工数に対する変動要因

とを対象とします。

機能要件は，システムがユーザ企業に提供する機能を指します。そのまま直接システムの規模を見積もるための入力となります。

非機能要件(品質要件や技術要件など)は，機能を実現するに当たっての目標値として設定するものです。個々のシステムにおいて，要求されるレベルを通常のレベル(類似のシステムの要求レベルで最も頻度の高いもの)と比較し，個々のシステム開発における実現可能な生産性を評価するために用います。

(3)　改良開発の機能実現における変動要因

改良開発では，新規開発の変動要因に加えて，既存システムがあることに起因する変動要因があります。これを**図 3.9**に機能実現のコストに対する変動要因として示します。また，**表 3.7**～**表 3.10**に変動要因を整理し，それぞれの変動要因がコストに影響を及ぼす理由および影響度合いを評価する観点をまとめます。

表 3.7　改良開発に特有の機能実現に対する変動要因(規模に影響)の例

主特性	副特性	影響の理由と評価の観点
変更箇所の結合度	——	モジュール間インタフェースの方式(ファイル連動，パラメータ連動，共有領域連動，DB連動，転送連動，マクロ)によってモジュール間結合度が異なり，調査作業に影響する。
規　模	改良開発の規模	改良開発の規模は開発作業の増加に直接影響を及ぼす。
	既存システムの規模	既存システムの規模は直接的に作業を増加させないが，扱う対象が増加することにより調査作業の効率を悪化させる。

表 3.8　改良開発に特有の機能実現に対する変動要因(生産性に影響)の例

主特性	副特性	影響の理由と評価の観点
習熟度	ベンダ企業の経験	既存システムの調査を実施するベンダ企業が，あらかじめ知識を有している場合，調査作業の効率化につながる。
	ユーザ企業の経験	既存システムの調査にあたってユーザ企業側が情報提供を行うことにより，調査作業の効率化につながる。
	リバースエンジニアリングの能力	既存システムの調査にあたってシステムの機能，構造などをリバースエンジニアリングする能力が高い場合は，調査作業の効率化につながる。
変更箇所の分散度	—	変更箇所が分散していると，開発作業範囲が広がるとともに，変更箇所の影響を考慮した作成および確認が必要となり，開発作業を増加させる。
既存システムの理解容易性	ドキュメントの整備状況	既存システムの開発において，参考情報として状況を正しく反映したドキュメントがある場合は，開発作業の効率化につながる。
	システムのつくり	既存システムのつくりが理解容易な構成(例：モジュール化が進んだもの)であれば，開発作業の効率化につながる。
	データ構造	データ構造のつくりが理解容易な構成(例：正規化)であれば，開発作業の効率化につながる。
既存システムの正確性	—	既存システムの正確性が高い場合は，開発作業の効率化につながる。
改良開発の波及度合い	—	波及範囲の増加により，波及を考慮した作成および確認が必要となり，開発作業を増加させる。

表 3.9　改良開発と新規開発に共通の変動要因(規模に影響)の例

主特性	副特性	評価の観点
機能性	合目的性(要求仕様の網羅性)	要求の記述水準および網羅性。要件定義については新規性，方針明確性，ステークホルダの多様性などを考慮
	正確性	標準レビュー工数(各工程10%)を基準にした要求水準
	接続性	基準単位(100KSLOC)に対する社内/社外システムとのインタフェース先の数
	整合性	整合をとる社内/社外の規格・基準の数，全体適合性やグローバル化対応も含む

(4)　規模見積り

改良開発では，機能実現の規模として，追加規模＋修正規模＋削除規模を採用し，機能実現のコストと相関を求める試みが行われています。なお，改良開発の場合の機能規模の数え方は，本ガイドブックの6.1.1項「ファンクションポイント法における機能改良時の数え方」で紹介しています。

また，既存システムがない場合は，修正規模および削除規模はありませんから，追加規模だけになり，新規開発の見積りモデルで採用している規模に一致します。なお，コストとの相関関係がより高くなる変動要因を求めるために重回帰分析を実施している企業もあり，その結果，次の算式で求まる開発規模の計算式を採用すると良い結果が得られるとの報告があります。

$$\text{追加規模} + (\alpha \times \text{修正規模}) + (\beta \times \text{母体規模}) + (\gamma \times \text{削除規模})$$

(5)　コスト/工数見積り

コスト/工数を規模から見積る場合，生産性の単位は，見積り対象が工数かコストかに応じて，人月/規模(単位規模当たりの開発にかかる工数(人時など)金額/規模(単位規模当たりの開発にかかる金額)などさまざまな設定が可能で，いずれの場合でも，組織における過去のプロジェクトデータなどに基づいて，次の二つの活動を行い，計算方法を確立しておくことが必要です。

表 3.10　改良開発と新規開発に共通の変動要因(生産性に影響)の例

主特性	副特性	評価の観点
開発環境特性	テスト手順書水準	テスト手順の具体化度(操作手順，入出力の具体化の要求水準)
工程入力情報特性	業務関連資料	必要資料の具備状況(正確性，信頼性を含む)および使いやすさ(検索性，理解性)
	他システム関連資料	必要資料の具備状況(正確性，信頼性を含む)および使いやすさ(検索性，理解性)
	規約・標準化関連資料	必要資料の具備状況(正確性，信頼性を含む)および使いやすさ(検索性，理解性)
顧客の協力特性	役割分担特性	顧客がベンダ企業に協力する度合および顧客とベンダ企業との役割分担の明確性
機能性	合目的性(要求仕様の網羅性)	要求の記述水準および網羅性。要件定義については新規性，方針明確性，ステークホルダの多様性などを考慮
	正確性	標準レビュー工数(各工程10%)を基準にした要求水準
	接続性	基準単位(100KSLOC)に対する社内/社外システムとのインタフェース先の数
	整合性	整合をとる社内/社外の規格・基準の数，全体適合性やグローバル化対応も含む
効率性	実行効率性	実行効率に対する一般的要求水準(注)の最適事例を基準にした要求水準
	資源効率性	資源効率に対する一般的要求水準(注)の最適事例を基準にした要求水準
保守性	解析性	ソースコードの解析性をコード化規約に定めるコメントに対する要求水準により評価
	安定性	ソフトウェア変更に対しシステム品質維持可能とする水準をライフサイクル目標年数の長さにより評価
移植性	環境適用性	多様なハード，ソフト，運用環境に適用させる要求の水準
	移植作業性	環境を移す際に，必要な労力を低減させる要求の水準
	規格準拠性	移植性に関する国際/国内規格または規約を遵守する要求水準
	置換性	使用環境/条件を変更せずに他のソフトウェア製品と置き換えて使用可能とする要求の水準

(注)　類似のシステムの要求レベルで最も頻度の高いものに基づいて，あらかじめ設定したもの。

・ベースラインの設定
・ベースラインからの変動要因の設定

(a)　ベースラインの設定

規模と工数，規模と金額などの関係のベースラインは，基本的に組織における過去の実績データの分析に基づき設定する必要があります。

規模と工数の間の関係式としては，工数と規模が正比例(工数＝α×規模)するもの，工数と規模の累乗が比例(工数＝α×規模$^{\beta}$)するものが主に利用されています。

組織でどの関係式を用いるかは，

・当該組織で収集しているデータの範囲でもっとも誤差が少なくなる関係式かどうか
・組織でどの関係式が最も現場で納得されるか

という点から判断します。

なお，上記のような基本的な関係は見積りの基準値を求めるためのベースラインとなるものです。

(b)　ベースラインからの変動要因の設定

続いて，ベースラインからの変動を考慮した見積りが必要ですが，生産性に影響を与える変動要因は，ソフトウェアを構築するためのコストに大きな影響を与えます。同じ機能のものを作成するにも，変動要因が異なれば，必要な工数は大きく異なります。

ベースラインからの変動要因を設定するに当たっては，ユーザ企業側とベンダ企業側のどちらに原因があるかが重要です。変動要因は，ユーザ企業側がコントロールできるものと，ベンダ企業側がコントロールできるものに分けることができます。以下，ユーザ企業がコントロールできる変動要因を「ユーザ制御要因」，ベンダ企業のものは「ベンダ制御要因」と呼びます。

見積りを互いに納得するためには，基本的にユーザ制御要因について共通認識を持つことが重要です。ベンダ制御要因は，契約時の見積りではユーザ企業が直接考慮するものではありません。ただし，いうまでもなくプロジェクトマネジメントの観点からは非常に重要なものです。ユーザ企業・ベンダ企業相互に変動要因および変動要因の設定に関するリスクを明らかにして，相互の責任

を確認し，協力してモニタし，コントロールすることが大切です。

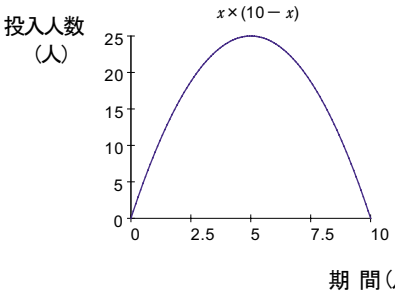
（６）　工数と工期の関係

主に新規開発の場合ですが，工期は，工数と高い相関関係があるといわれており，一般的な関係として次のものが示されています（例：COCOMO法）。

工期＝α ×（工数のβ 乗根）
ただし，α ＝2.0～3.0
β ＝3.0前後

傾向として，工期（T_D）は，工数（*Effort*）のおおよそ３乗根に比例する傾向があります。これまでα＝2.0～3.0と報告されています。この関係が成り立つためには，開発中のメンバの割り当てが**図 3.10**に示すとおり山型であることが必要です(7)。

改良開発においても，人数の割り当てがこのような状況になる場合は，工期を工数の３乗根に比例するとして見積ることが可能です。ただし，改良開発の場合は，「小さな案件を多数同時に並行する期間保守」や，「設計を必要としない単純な改訂作業（例：画面レイアウトの一律変更）が主流の場合」など，当て



（備考）例として期間は10カ月としてみたもの

図 3.10　ソフトウェア開発中の人数の割当て例

（７）　山型である必要の理由は，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年の「1.3.5　工期見積り」を参照。

はまらないケースも少なからずあります。この場合は，要員割り当ては山型でなく長方形となり，工数÷要員で期間が算出できます[8]。

3.3.3　テストの見積り

（1）　規模見積り

テストは，まず新たに追加した機能，修正した機能および削除した機能に関して実施します。さらに，それだけでなく，機能の追加，修正および削除が，既存の一切変更しない機能に影響していないことを確認するために影響範囲に対して実施します。また，影響範囲に関わらず，全機能のリグレッションテストを実施することを要求される場合があります。

したがって，テストの規模はテストの対象範囲の規模とする必要があるので，追加，修正および削除した規模のほかに，改造の影響範囲およびリグレッションテストの範囲の規模を加える必要があります。例えば，ジャステック手法で示されている「テスト巻き込み量」はその一つです。

調査・分析の結果，テスト計画によって影響範囲およびリグレッションテストの範囲が確定するので，その規模を採用します（**図 3.6**）。

（2）　工数/コスト見積り

テストに関して，既存システムがあることに起因して新規開発の変動要因に対して付け加わるものは，**図 3.11**に示すとおり，規模を始め，改良開発の波及度合い，既存システムの品質（正確性），テスト環境やテストケースとテストデータの再利用が主なものです。

プロジェクト全体のコストに対して，テストの変動要因が影響する度合いを求めるためには，あらかじめテスト工数の全体工数に対する比率を過去プロジェクトデータなどから求めておく必要があります。

（8）　併せて，準備工数などの付帯作業分を落とさないように注意が必要です。

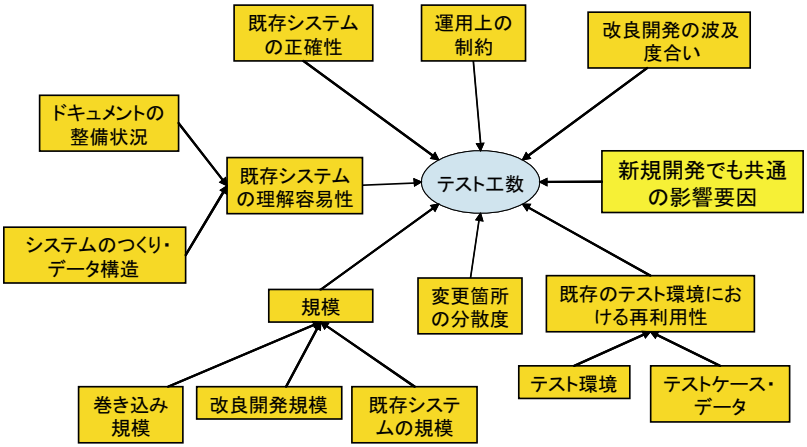


図 3.11　テスト工数に対する変動要因

表 3.11　改良開発に特有のテストに対する変動要因(規模に影響)の例

主特性	副特性	影響の理由と評価の観点
変更箇所の結合度	——	モジュール間インタフェースの方式(ファイル連動，パラメータ連動，共有領域連動，DB連動，転送連動，マクロ)によってモジュール間結合度が異なり，調査作業に影響する。
規　模	改良開発の規模	改良開発の規模が多いとテスト対象が広がるため，テスト作業を増大させる。
	既存システムの規模	既存システムの規模はテスト対象の規模に影響を及ぼすことから，テスト作業を増大させる。
	巻き込み規模	改良要求の確認やレベルダウンしていないことの確認のために，テストに巻き込む規模が，改良開発におけるテスト量(テスト項目数)を左右する。

表 3.12　改良開発に特有のテストに対する変動要因(生産性に影響)の例

主特性	副特性	影響の理由と評価の観点
既存システムの理解容易性	ドキュメントの整備状況	既存システムのテストの参考情報として状況を正しく反映したドキュメントがある場合は，テスト作業(テストケース作成・実施)の効率化につながる。
	システムのつくり	既存システムのつくりが理解容易な構成(例：モジュール化が進んだもの)であれば，テスト作業の効率化につながる。
	データ構造	データ構造のつくりが理解容易な構成(例：正規化)であれば，テスト作業の効率化につながる。
変更箇所の分散度	——	変更箇所が分散していると，テスト作業が多岐にわたり，また，テストの分散など，テスト作業の効率化を悪化させる。
既存システムの正確性	——	既存システムの正確性が不足していると，テスト時の不具合の発生など，手戻りの発生を招き，テスト作業を増加させる。
改良開発の波及度合い	——	波及範囲が多いと波及先のテストを実施する必要があり，テスト作業を増加させる。
既存のテスト環境における再利用性	テスト環境	テスト環境を再利用できるとテスト作業を効率化することができる。
	テストケース・データ	テストケースおよびテストデータを再利用できるとテスト作業を効率化することができる。
運用上の制約	——	改良開発におけるテスト環境は多くの場合，運用環境となんらかの連係(ハードウェアやネットワークの共有など)を持つ。また，テストの一部を実施するために，運用環境そのものや実データを使用する場合がある。そうした場合，テストの実施時間帯やリソースの使用量，アクセス権などに制限を受け，テスト効率の悪化につながる場合がある。

第 4 章　改良開発での見積り精度向上

4.1　調査・分析の重要性

見積りにおける大きな課題の一つに要求があいまいな状況で見積もらなければならないことがあります。改良開発でもシステムの目的がなぜそれを実現するために，どのような機能を追加すべきかが明確でないと，同じ問題が生じます。

改良開発では，さらに重要なことは，機能が明確な場合であっても，それを追加・変更するシステム自体を理解しておく必要があります。

事例などで紹介したとおり，すでに存在するシステムに機能を追加・変更することの影響範囲を把握しておかないと，設計時に考慮すべき事項が抜けたり，レビューテスト対象から漏れたりし，結果として改良する前は正常に稼働していた部分に不具合を挿入することになり，システムが止まるといったことが生じます。

改良開発の場合，最初に既存システムに対する理解度を把握し，理解度が低いと判断される場合に，システムの調査・分析を実施することは，見積りの精度向上のみならず，プロジェクトの成否にかかわる重要な活動です。

特に，システムの担当ベンダ企業が変わって，新たな企業が担当する場合は，既存システムに対する理解度を確保するための方策として，ユーザ企業からの情報の提供(例：レクチャの実施，ドキュメントの提供など)やベンダ企業によるシステムの調査・分析が必須です。

このとき，大規模なシステムであれば，システムの調査・分析を独立したプロジェクト(契約単位)として実施することが考えられます。また，プロジェクト全体の見積りを調査・分析の実施後に行うことで，見積り精度の向上を図ることができます。あいまいな情報では，見積り精度を期待することはできません。プロジェクト全体を工程で分け複数段階での契約(多段階契約)として，あいまいな状況をはっきりさせるための工程を独立した契約で実施することは，プロジェクトを成功させるために有効な手段です(4.4節を参照)。

4.2　見積りの前提条件のモニタリングとコントロール

改良開発に限られるものではありませんが，見積り値と実績値で差が出てしまう原因の一つに見積りを行ったときの前提と実際の状況が変わってしまうことがあります。

まず，改良開発の場合は，見積り時に想定した既存システムの影響範囲が設計，実装が進むにつれて明らかになり膨張し，最終的な影響範囲の規模が大きくなってしまうことがあります。このような場合，見積りを当初想定した影響範囲に基づいたものに固定していると，見積りと実績が大きく乖離してしまうことになります(**図 4.1**)。特に，既存システムの品質があまり高くない場合，潜在していた不具合が現れたために影響範囲が拡大し，追加の修正や対策が必要となる例は少なくありません[9]。既存システムの状況(品質，ドキュメントの整備状況など)に関する情報が不明確な場合(特に，維持管理が確実になされていない場合)，最初の段階ですべてを完璧に調査・分析することは困難です。その状態で行う見積りは，多くの前提条件に仮設をおいたものになります。

また，新規開発の場合と同じように，プロジェクトの構想段階で何を作成するのかさえあいまいな状況で見積った結果が最後まで固定され，実績値はそれ

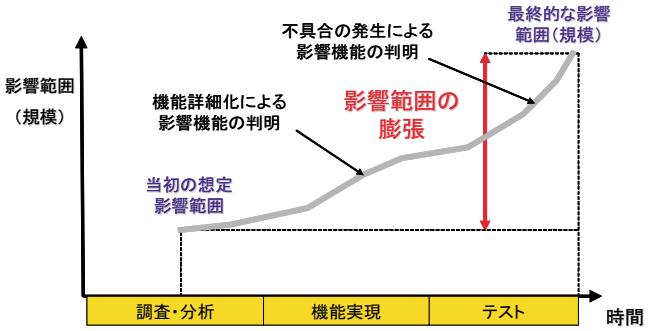


図 4.1　既存システムへの影響範囲の膨張

(9)　そのような状況になった責任がユーザ側にあるのか，ベンダ企業側にあるのか，によって実際にコストに反映されるかどうかは変わってきますが，いずれにせよ，前提が違うことにより予測と実績に差がついてしまうことが，ここでの論点です。

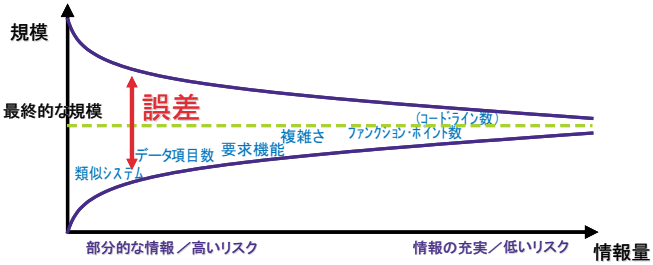


図 4.2　前提があいまいな状況

と違ったものになる場合もあります。このような場合は，プロジェクトの初期段階では見積りの誤差が大きく，プロジェクトが進みあいまいな項目が減るにつれて，誤差が小さくなっていきます(**図 4.2**)。

さらに，システムの保守段階では，保守要員の人数を固定し，その中で大小さまざまな開発案件を複数並列で進めることがあります。こうした場合，並列案件がお互いに影響を及ぼし合い，単独案件では起きなかったような影響を被ることがあります。具体的には，同一の組織が複数の案件を担当していますので，個別案件の工数を合計した工数が保守要員の人数内に納まる必要がありますが，ある案件での進捗遅れや工数増大が発生すると，全体のリソースが圧迫され，優先度の低い案件の中止や，優先度の入れ替えなどが起こります。また，改良開発では，並行して既存システムが運用されているので，そちらで起こる障害や，業務上の緊急処置などの影響も少なからず受けることがあります。

このように，さまざまな変動要因において，当初想定していたもの(ユーザ企業とベンダ企業とで合意していたもの)が，状況の変更により変わってしまうということは少なくありません。

このような状況で見積りの精度を確保するためには，見積りの前提条件が変わってしまうことから見積り方法で解決することは無理であり，前提条件を明らかにした上で，関係者でその情報を共有し，モニタリングとコントロールを行っていくしかありません。

なお，モニタリングとコントロールに関連して，4.4.2項ではユーザ企業とベンダ企業との間で把握・共有しておくべき内容を，また,4.4.3項ではユーザ企業がコントロールできる生産性変動要因の調整のプロセスについて述べま

す。

4.3　見積り手法と継続的な改善

こちらも改良開発に限ったものではありませんが，見積り精度の向上のためには，見積り手法・モデルを確立した後で，見積り予測値と実績値の違いの差異分析を通して，見積り手法・モデルの改善を行う必要があります。また，このとき重要なのは，手法またはモデルの改善だけでなく，プロジェクトマネジメントについても十分なチェックを行うことです。見積り値が妥当であっても，プロジェクトマネジメントに失敗した場合は，改善すべき対象はプロジェクトマネジメントの方になります。

また，リスクの排除も見積りと実績を一致させるために重要なポイントとなります。改良開発では，既存システムに関する品質の状況やドキュメントの整備状況は大きな変動要因となりますが，これらが「悪い」状況であった場合，予測のつかないことが起こることは経験的に明らかです。状況がわからない場合は言うまでもありませんが，リスクがわかった場合でも，そのぶれ幅を正確に予測することは難しく，逆にそのリスクを排除してぶれを除く努力をする方が効率的と考えられます。

なお，システムの保守段階では，複数の案件が並列して進むことが多く，個別案件は小型化する傾向にあります。場合によっては，一人の要員が複数の案件に同時に携わることもあります。さらに本書では対象外としましたが，保守段階においては，「保守の問合せ」「保守の基盤整備」「是正保守」などさまざまな作業が，改良開発と並行して行われています。こうした状況下では，組織として明確に情報収集の意思を持ち，収集する手順を明示しないと，情報は日々の作業に埋没してしまいます。

プロセス改善において適切な情報を収集するためには，こうした状況下においても，改良開発案件を識別して見積りの予測値/実績値を収集する仕組みの構築が必要になります。例えば，小規模案件の見積り情報は，進捗が進むにつれ散逸しがちですので，見積もった時点でタイムリーに情報を保存し，案件完了後に実績と対比できるよう手順を整備する必要があります。

（1）　見積り手順と実績収集手順の確立

組織で一貫した手順として見積り手順，実績収集手順を確立する。

（2）　見積りの実践

確立した見積り手順を実践する。また，見積り活動そのものではないが，見積りに当たって可能な限りリスクを明らかにするとともに，排除しておく。

（3）　見積り結果(計画)と実績の差異分析

計画時の見積り結果とプロジェクト完了時の実績値との間の差異分析を通して，差異の根本原因を特定する。

（4）　差異分析結果のフィードバック

差異分析結果に基づいて改善対策を検討し，対策を展開。差異分析の結果反映される対象として，次の二つがあります。

①　プロジェクトマネジメントへのフィードバック

②　見積り手順(見積り手法を含む)へのフィードバック

（5）　共通要因に基づくプロセス改善

複数のプロジェクトで共通な課題を見積り，実績評価の繰り返しに基づき分析し，対策を検討して，プロセス改善を展開する。

図 4. 3に見積り手法の構築からフィードバックまでのサイクルを示します。

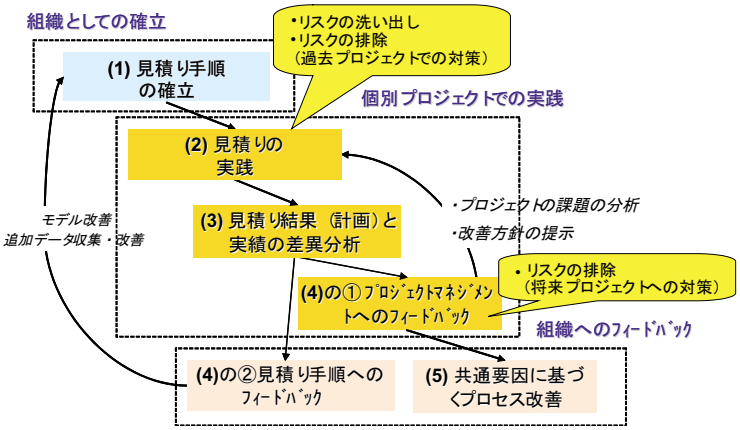


図 4. 3　見積り精度向上のPDCA

4.4　契約によるリスク解消の糸口

4.4.1　見積りにおけるユーザ企業とベンダ企業の役割

見積りにおいて，ユーザ企業およびベンダ企業のそれぞれの役割は以下のとおりです。

まず，ユーザ企業はシステム開発でのすべての意思決定の主体であり，機能要件や非機能要件の内容は，ユーザ企業が決定します。**図 3.8**に示したとおり，機能要件の内容や品質要件・技術要件などは，システムの規模や開発の生産性を決定します。したがって，システムの規模や開発の生産性は，ユーザ企業がコントロールできます。機能要件，品質要件，技術要件などを取捨選択し，その内容のレベルを調整することによって，最終的な工数またはコスト(ひいては価格)の低減や工期の短縮を図ることができます。

逆に，ベンダ企業側は，そのような要件をユーザ企業が判断・確認することをシステム開発のプロフェッショナルとしてサポートする必要があります。これは,ユーザ企業すべてがシステム開発に慣れていないことが背景にあります。例えば，ユーザ企業にとって，システムの技術的難易度など，非機能要件のうち，システム構築の知識がないと判断できないものは，ベンダ企業のサポートが不可欠です。

4.4.2　確定していない部分の把握と認識共有

詳細な情報が得られていない時点では，不確定な部分があるので見積りには誤差が避けられません。そこで，次の点をユーザ企業とベンダ企業との間で認識を共有することが基本となります。

- どの情報に基づいた概算見積りであるか
- 既存システムに関する品質状況，ドキュメント整備状況
- 確定していない部分が何であるのか
- 確定していないことにより発生する誤差範囲
- 確定していない部分が確定する時期
- 誤差が発生した場合での対処の取り決め(再見積りの実施，機能の縮小・変更など)

これらを合意した上で，プロジェクトの進行中，不確定なもの(確定したものの変化を含む)をモニタし，必要に応じてコントロールする必要があります。

4.4.3　変動要因に関するユーザ企業とベンダ企業との調整プロセス

要求される品質のレベルを設定する場合と同様に，ユーザ企業とベンダ企業間で，個別のソフトウェア開発プロジェクトでの変動要因のレベルをチェックして，今回のソフトウェアは，個々の変動要因の基準(類似のソフトウェア開発のプロジェクト実績データから得られる基準値)から，どの程度高いのか，低いのかを確認しあい，そのレベルに応じて生産性の高低を評価し，見積りへ反映することで，見積りの妥当性を確保する必要があります。これは新規開発，改良開発にかかわらず，同じ取組みが必要です(10)。

4.4.4　多段階契約の採用

前項で述べたとおり，改良開発では，見積りに際して調査・分析フェーズを実施する前に，調査・分析フェーズで得られる情報を推測するために，サンプリング調査などの手段で確認して,見積りのインプットとすることになります。

しかしながら，見積りのために行うサンプリング調査は，限られた期間とコストの制約の中で実施されます。その場合，精度が低くなり，中には推測するに足る根拠を得られないことがあります。調査の期間およびコストを抑えて，さらにその精度を上げるために，ユーザ企業側で既存システムの理解度の高い体制にするか，理解度の高い企業・組織に開発を委託することが有効です。

しかし，理解度の高い体制に持っていけない場合や，体制だけでは解消できない調査・分析項目がある場合には，開発工程ごとに契約を締結して，状況を確認しながら多段階契約を選択するのも一つの方法です。この場合，特に調査・分析のための調査を独立した契約としたり，見積りのための調査そのものを独立した契約とすることは有効です。

多段階契約では，契約作業にかかわる手間は増大しますが，開発途中で発生しがちな影響範囲の増減を確認しつつ，その時点で明確になった部分の反映が可能であるため，特に改造の影響範囲が不明確なシステムの場合のプロジェク

(10)　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年の「1.3.4(4)変動要因に関するユーザとベンダの調整プロセス」を参照。

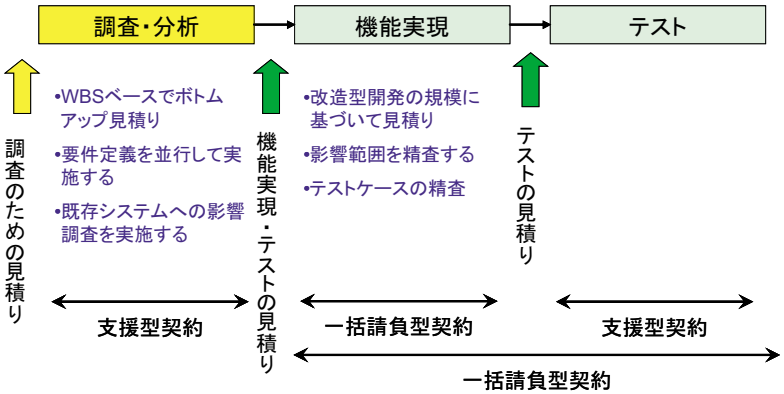


図 4.4　改良開発における契約・再見積りのタイミング例

トに適しています。

図 4.4に，多段階契約と契約・再見積りのタイミング例を示します。テストについては，場合に応じて，支援型契約または機能実現と合わせた一括請負型契約となります。

4.4.5　実費償還型契約の採用

多段階契約のほかにも，ユーザ企業・ベンダ企業双方のリスクを軽減させる方法として，実費償還型契約があります。実費償還型契約とは，契約上定められた金額の範囲内で，発生したコストに対する実費が支払われることです。

実費償還型契約は，さらにいくつかの契約形態に細分化されますが，比較的多いのは，動機付け型契約と報償型契約の二つです。動機付け型契約は，予定コストに対する実際コストの差分を，超過，余剰に関係なくユーザ企業・ベンダ企業間で分担して負担するものです。一方，報償型契約では，契約開始時に設定された基本額に加えて，ユーザ企業が常に成果物の品質などを評価し，その結果に基づいて追加の報酬をベンダ企業に支払うものです。

いずれもプロジェクトの過程において，ユーザ企業側には，ベンダ企業のパフォーマンスを評価する能力が要求されますが，自分たちにとって，本当に使いやすいシステムを適正なコストで実現することが可能となります。一方，ベンダ企業側にも自らのパフォーマンスを説明できる能力が求められます。この

契約の特徴を最大限に生かすには，ユーザ企業とベンダ企業との間の密なコミュニケーションが必須条件になります。具体的な事例は，「ソフトウェア開発見積りガイドブック」を参照ください。

第５章　改良開発のコスト低減

5.1　改良開発におけるコストコントロールのアプローチ

改良開発の見積りでは，さまざまな要因が開発のコストに影響を及ぼします。コストに影響を与える変動要因の例は，3.3節および第２部に示すジャステックの改良開発見積り（環境変数）に整理しています。

変動要因は，規模に影響を与える変動要因と生産性に影響を与える変動要因とに分けられ，また，新規開発は改良開発のうちの特殊な形態と捉えられるので，変動要因には新規開発と共通する変動要因と改良開発に特有な変動要因とがあります。この変動要因をユーザ企業とベンダ企業とで調整するプロセスの概要は，4.4.3項に記載しています。ユーザ企業・ベンダ企業間で，個別の改良開発プロジェクトでの変動要因のレベルをチェックして，今回のプロジェクトは個々の変動要因の基準と比較して，どの程度高いのか，低いのかを確認しあい，そのレベルに応じて規模の多寡および生産性の高低を評価し，見積りへ反映します。

具体的には，**表 3.5**～**表 3.12**に示した項目をもとに，ユーザ企業・ベンダ企業間で前提を合意し，見積りを行います。これらの変動要因の中には，プロジェクトの見積り時に，ユーザ企業またはベンダ企業が確約できないものも存在しますが，見積りに際して前提条件として仮決めした上で，これらをリスクとして共有します。プロジェクトの進行段階では，その前提をモニタして変化があるか否かを把握して，変化がある場合は，必要に応じてユーザ企業・ベンダ企業間で前提を再確認するプロセスを踏みます。

ユーザ企業およびベンダ企業が互いに納得できる見積りを行うためには，以上に示してきた事項をユーザ企業・ベンダ企業間で密にコミュニケーションし，前提を把握し文書化した上で，その変化をモニタしコントロールすることにつきます。

例えば，**表 5.1**の事例は長期的なコストコントロールを実現した例です。

本節では改良開発に特有の変動要因に焦点を当てます。改良開発の変動要因

表 5.1　改良開発におけるコストコントロールの事例

No.	事　　例
1	保守において，短期的な保守コスト削減・変更リスク削減によりシステムの保守性を低下するのを避けるために，保守標準を定め，実行することを合意した。短期的にはコストの増加となるが，長期的にはコスト削減につながることをユーザ・ベンダ間で合意できた(11)。

で新規開発の場合と比べて異なるのは，既存システムにかかわる要因が存在することであり，改良開発の見積り時点では事実として受容せざるを得ない要因（以降，本章ではこれを「制御不能な変動要因」と呼ぶ）が多いことです。

5.1.1　プロジェクトの見積り時に調整可能な変動要因

（1）　規模に影響を与える変動要因

第3章の記載と重複しますが，改良開発の見積り時点で調整可能か否かという観点を付け加えて，規模に影響を与える変動要因を，新規開発と共通する変動要因とに整理したものを「(a)改良開発に特有の変動要因」と「(b)改良開発と新規開発とに共通の要因」とに示します。なお，制御不能な変動要因は，下線を付けて示します。

(a)　改良開発に特有の変動要因

表 5.2の副特性のうち，既存システムの保守性は，既存システムを改造する時点では確定している要因であって，当該改良開発案件に関するプロジェクトでは事実として受容せざるを得ません。既存システムの保守性を高める手段は，リファクタリングなど種々の方法がありますが，いずれも新たにコストをかけることになり，当該改良開発案件のコストに影響します。

(11)　短期的な保守コスト削減・変更リスク削減を追及した場合，基幹的な部分に手を入れるのを避けることがあります。例えば，さまざまなプロセスを経た出口で，「このケースはデータをこのように置き換える」ことなどで，既存システムに影響なく当面の目的を達成できますが，変更を繰り返すたびにこのような対応を続けることで，保守効率が低下することになります。

表 5.2　改良開発に特有の変動要因（規模に影響）の例

変　動　要　因		影響を受けるフェーズ		
主特性	副特性	調査分析	機能実現	テスト
変更箇所の結合度	——	○	○	○
改良の規模	——	○	○	○
既存システムの規模	——	○	○	○
巻き込み規模	既存システムの保守性 改良開発後のシステムに要求する保守性，改良開発後のシステムに要求する信頼性	—	—	○

（b）　改良開発と新規開発とに共通の要因

ここでは，既刊書の「ソフトウェア開発見積りガイドブック」で紹介している規模へ反映する品質要件の例を，簡易な形で**表 5.3**に再掲します。

表 5.3　改良開発と新規開発とに共通の変動要因（規模に影響）の例

変　動　要　因		影響を受けるフェーズ		
主特性	副特性	調査分析	機能実現	テスト
機能性	合目的性，正確性，接続性，セキュリティ	○	○	○
信頼性	成熟性，障害許容性，回復性	○	○	○
使用性	理解性，習得性，操作性	○	○	○
保守性	開発するシステムに要求する解析性，変更作業性，試験性	○	○	○

（2）　生産性に影響を与える変動要因

同様に，改良開発の見積り時点で調整可能か否かという観点を付け加えて，生産性に影響を与える変動要因を，「(a)改良開発に特有の変動要因」と「(b)改良開発と新規開発とに共通の要因」とに示します。同じく，制御不能な変動要因は，下線を付けて示します。

(a)　改良開発に特有の変動要因

既存システムへの習熟度は，改良開発案件に参画するベンダ企業の経験およびユーザ企業の経験に依存し，利用可能な人的資源の制約に大きく依存します。本要因は，改良開発のプロジェクト編成時に調整できますが，新規開発のプロジェクトと比較して選択できる利用可能な人的資源の自由度が著しく狭くなるので，制御不能な変動要因としています。

また，ドキュメントの整備状況，システムの正確性および既存テスト環境の再利用性は，既存システムの保守性と同様な事由で，当該改良開発案件に関するプロジェクトでは事実として受容せざるを得ないものです。

(b)　改良開発と新規開発とに共通の要因

ここでは，「ソフトウェア開発見積りガイドブック」で紹介しているユーザ制御要因の例を，簡易な形で**表 5. 5**に再掲します。

表 5. 4　改良開発に特有の変動要因(生産性に影響)の例

変　動　要　因		影響を受けるフェーズ		
主特性	副特性	調査分析	機能実現	テスト
変更箇所の分散度	既存システムの保守性，改良開発後のシステムに要求する保守性	○	○	○
習熟度	ベンダ企業の経験	○	○	—
	ユーザ企業の経験			
	リバースエンジニアリング力(リバースエンジニアリング，母体調査ツールの水準を含む)			
既存システムの理解容易性	ドキュメントの整備状況	○	○	○
	システムのつくり・データ構造			
改良開発の波及度合い	——	—	○	○
システムの正確性	既存システムの品質	—	○	○
既存テスト環境の再利用性	テスト環境	—	—	○
	テストケース・データ			

表 5.5　改良開発と新規開発とに共通の変動要因(生産性に影響)の例

変　動　要　因		影響を受けるフェーズ		
主特性	副特性	調査分析	機能実現	テスト
業務特性	業務ナレッジ	○	○	○
ソフトウェア特性	安定度/信頼度/使用度	○	○	○
ハードウェア特性	安定度/信頼度/使用度	○	○	○
コミュニケーション特性	顧客窓口特性,工期の厳しさ,コミュニケーション基盤，レビュー体制	—	—	—
開発環境特性	開発手法/開発環境，テスト手順書水準	○	○	○
工程入力情報特性	業務関連資料,他システム関連資料,規約・標準化関連資料	○	○	○
顧客の協力特性	役割分担特性	○	○	○
機能性	合目的性(要求仕様の網羅性)，正確性，接続性，整合性	○	○	○
効率性	実行効率性，資源効率性	○	○	○
保守性	開発するシステムに要求する解析性，安定性	○	○	○
移植性	環境適用性，移植作業性，規格準拠性，置換性	○	○	○

5.1.2　プロジェクトの見積り時に調整が困難な変動要因

前述のとおり，当該改良開発案件に関するプロジェクトには，規模に影響を与える変動要因にも，生産性に影響を与える変動要因にも制御不能な変動要因が含まれます。これら制御不能な変動要因を**表 5.6**に整理しています。

これら制御不能な変動要因は，既存システムを新規に開発する時点から考慮し，システムのライフサイクルにわたり，コントロールすることになります。

改良開発では，前述の変動要因で想定外のことが多く見受けられ，見積りで想定していた条件と違っていることが後になって判明し，結果として実績と大きくぶれてしまう，ということが事例で示されています。そして多くの場合，以前のプロジェクトの開発状況が大きな影響を及ぼしていることがわかっています。言い換えれば，以前のプロジェクトの開発状況をコントロールすることにより，将来の改良開発プロジェクトのコストを抑えることができます。将来

表 5.6　制御不能な変動要因の例

主特性	副特性
習熟度	ベンダ企業の経験
	ユーザ企業の経験
	リバースエンジニアリング力 （リバースエンジニアリング，母体調査ツールの水準を含む）
既存システムの理解容易性	ドキュメントの整備状況
	システムのつくり・データ構造
既存テスト環境の再利用性	テスト環境
	テストケースデータ
システムの正確性	既存システムの品質

に備えるというのは，関係者が強く認識し理解し合うことが必要である一方で，ユーザ企業とベンダ企業の両者で統制が必要であるため，簡単に実現できないのも事実です。しかし，効果の大きさと悪影響の大きさとを認識すれば，対策をあらかじめ取っておくのが，見積りの成功，ひいてはプロジェクトの成功につなげる最も簡単な方法とも言えます。

5.2　事例に見る改良開発のコスト低減策

本節では，前述した制御不能な変動要因のコントロールに関して，実際の事例を紹介します。

（1）　既存システムへの習熟度の維持・向上

既存システムへの習熟度の維持は，改良開発の体制が，過去の経験が生かせるか，または解析能力に優れている観点で捉えます。具体的には，利用者側の担当者に知識が蓄積しているか，開発者側の経験者を担当させることができるか（新規に担当する場合は業務知識や環境に関する知識・経験の活用も含まれる），仮に知識が不足する場合でもリバースエンジニアリング技術に優れたチームを担当させることができるか，ということです。

知識に関しては，以前の開発プロジェクトにおいて，どういう手段で，どこに（人的資源に暗黙知として，あるいはドキュメントなどで形式知として）知識

表 5.7　既存システムへの習熟度の維持・向上の事例

No.	事　　例
1	要求機能と実装する機能(ソフトウェア構成要素)とのマトリクス表を作成しておくことにより新規参画チームや新規参画要員の知識習熟度を向上させた。また，マトリクス表によりトレーサビリティを高めたことで，既存システムの機能把握および改良案件のテストケース作成効率を高めた。
2	既存システムのノウハウ保有者の突発的な異動による品質低下(リリース後の障害増加)を予防するために，次の事項を実施。 ① 既存システム機能別に，その機能を理解している要員が何名いるか分布表を作成し，習熟者の偏り度合いを評価し，習熟者が少ない機能について教育計画を設定し実施している。 ② 突発的な要員異動による影響を最小限に抑えるための準備として，保守用ドキュメントの整備などを対策として実施している。
3	保守要員の既存システムへの習熟度は，そのままにしておくと時が経過するとともに自然に低下していく。これを防ぐために，意図的に定期的に改良を行うことで，保守要員の既存システムに対する習熟度を維持できている。

を集約させるのかという方針が効いてきます。実用面において，暗黙知と形式知は相互に補完してこそ効果を発揮するものです。

既存システムへの習熟度を維持するために，ユーザ企業側では改良するシステムに関する経験・知識を持つチームの維持とともに，経験・知識を持つベンダ企業との関係を維持することが基本的な方策です。後者に関して，他に頼むところがないので継続して委託を行う受動的な姿勢では，時間やコストがかかりコスト高につながる危険性があります。つまり,技術力がある企業を選別し，戦略的に維持していくことが重要です。

表 5.7の事例は，要求機能と実装機能を整理しておくことで新規参画チームの知識習熟度向上などに役立った例と，既存システムのノウハウ保有者の突発的な異動による品質低下(リリース後の障害増加)の予防例，定期的な改良実施により保守要員の既存システムに対する習熟度を維持している例です。

（2） 既存システムの保守性の維持・向上

システムの保守性は，JIS X 0129-1：2003「ソフトウェア製品の品質－第 1 部：品質モデル」で定義されています。**図 5.1**に保守性を構成する副品質特性，さらに**表 5.8**に副品質特性と強い相関のある内部特性についてまとめています。内部特性とは，システムの内部構造や開発プロセスをホワイトボックスで

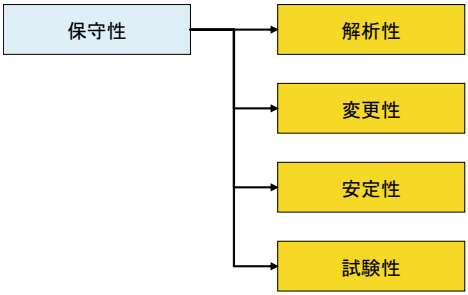


図 5.1　保守性の構成

表 5.8　保守性の品質副特性と内部特性との関係

品質副特性 ＼ 内部特性		追跡可能性	一貫性	自己記述性	モジュール性	単純性	計測性	簡潔性	拡張性	製品管理性	ソフトウェアシステム独立性	マシン独立性
保守性	解析性	◎	◎	◎	◎	◎	◎	◎	○	△	○	○
	変更性	◎	◎	◎	◎	◎	◎	◎	○	◎	○	○
	安定性	○	○		○	○	○		△			
	試験性	△	○	○	◎	○	◎		△		○	○

備考：◎：強い相関がある，○：相関がある，△：弱い相関がある

（出典）東基衛編集：ソフトウェア品質評価ガイドブック，日本規格協会，1994年

捉えた品質特性を指し，ユーザ企業にはなじみが薄く主に開発者の視点からみた特性です。たとえば，ソフトウェアのソースコードのような内部文書を見て初めてわかる特性のことであり，モジュール性はその一例です。内部特性の具体的な定義は，ISO/IEC TR 9126-3：2003「ソフトウェア工学－製品品質－第3部：内部測定法」などで検討されていますが，国際規格化には至っていません。

なお，SECのプロジェクト見える化部会で取り上げている「コードクローン

含有率[12]」などは，システムの保守性の度合いを測る有効な尺度だといわれています。

ユーザ企業は，品質副特性の切り口からシステムの保守性への外部要求を提示し，ベンダ企業は，外部要求を満足すべく内部特性の目標を設定して，システムの保守性を作りこむことになります。

また，システムは改良を繰り返していると次第に保守性が落ちる傾向にあります。したがって，戦略的に既存システムの保守性を維持または向上する方策を改良開発のたびに講じる必要があります。そのためには，システムの内容とドキュメントとの整合性の維持，ドキュメントおよびコードなどに対する記述ルール・標準の徹底などが具体的な方策となります。

表5.9　保守性を維持する方策と内部特性との関係

保守性を維持する方策＼内部特性	追跡可能性	一貫性	自己記述性	モジュール性	単純性	計測性	簡潔性	拡張性	製品管理性	ソフトウェアシステム独立性	マシン独立性
ドキュメントに関する標準の適用	◎	◎	◎	◎	◎	◎	◎	△	◎		
コーディング作法の適用	◎	◎	◎	◎	◎	◎	◎	○	◎	○	◎
構成管理	◎								◎		
コードの再構成(リファクタリング)		◎	◎	◎	◎		◎	◎		◎	○
データの再構成	◎	◎			◎		◎	◎			

備考：◎：強い相関がある，○：相関がある，△：弱い相関がある

(12)　SECでは，タスクフォース活動を通じて開発された手法を適用し，その有効性の実証や，手法の更なるブラッシュアップを図ることを目的として，先進ソフトウェア開発プロジェクトを実施しています。ここでは，プロジェクト測定・分析の一環としてコードクローン分析が採用されており，大阪大学の神谷年洋氏の手によるCCFinderというツールを使用して，ソースコードからクローンの分布状況，含有率などを分析し，ソフトウェア保守における潜在的リスクの「見える化」を試みています。詳細は，SEC,「ITプロジェクトの『見える化』下流工程編」，日経BP社, 2006年を参照してください。

また，これにはシステムのアーキテクチャやデータ構造を理解しやすくすることも含まれます。

具体的には，次のような方策が考えられます。また，内部特性との関連を**表5.9**にまとめます。

・ドキュメントに関する標準の適用
・コーディング作法の適用
・構成管理
・コードの再構成(リファクタリング)
・データの再構成

表5.10　既存システムの保守性の維持・向上の事例

No.	事　　例
1	情報系システムの新規開発時に，支店が新設あるいは廃止された場合に，システムのどの部分をどのように改良したら良いかを文書化しておいたことで，その後の支店の新設・廃止に伴う改良開発の生産性が向上した。
2	改良開発において，単体テストおよび統合テストの方針が不明確であったため，サブシステムごとのテスト水準や範囲が異なる状態でシステムテストに突入し，システムテストの効率が低下した。テスト工程ごとのテスト計画として作成し，その後の案件に適用した。
3	改良開発時は「改良案件名」付きでコメントラインおよび改訂履歴を挿入するルールを定め運用しており，既存システム調査(影響範囲検索)に有効活用できている。
4	ドキュメントのソースコードを変更履歴を含めて集中して電子管理するとともに，さまざまな検索手段(キー検索・全文検索・波及分析・関連資源検索)を提供した。追跡可能性が向上するとともに，影響箇所を確認し見積り制度を挙げることにも貢献した。
5	テスト検証ツールを使用してテスト結果を確認する作業において，検証ツールの仕様変更対応が漏れており(実際の結果は正しいのに，検証ツールでは結果を不正と判定する事象が発生)開発現場が混乱した。 開発ツール(テスト検証ツール)に関しても，構成管理やドキュメントの整備を徹底した。
6	過去のメンテナンス経験から改良は特定部分に集中することが多い。特に改良が多いと予想される部分を独立(あるいは部品化)させることで，モジュール性を高めて保守効率を上げることができた。
7	開発フレームワークを利用し，特に変化が早く，また特殊なノウハウが必要なミドルウェア部分を切り離す。 また，毎年，ミドルウェア構成，利用する技術の標準を定めることも効果がある。

表 5.10では，保守性を維持するために実施した具体的な方策の事例を幾つか紹介しています。具体的には，ドキュメントに関する標準の適用例(No.1,2)，コーディング作法の適用例(No.3)，構成管理の例(No.4,5)，モジュール性向上の例(No.6)，ソフトウェアシステム独立性向上の例(No.7)です。

（３）　既存テスト環境の再利用性の維持・向上

既存テスト環境の再利用性の向上とは，端的に言えば，過去の開発プロジェクトで利用したものを新たな改良開発に活用できるようにすることです。

テスト環境，テストケース，テストデータおよびテストシナリオ(テストの手続きを含む)などの再利用を実現することにより，生産性とシステム品質の両方の向上が可能になります。将来の保守を見据えて，このような既存テストリソースを整備・維持することは，改良開発の成功につながり，コスト低減の重要な戦略の一つです。

改良開発において，既存システムに悪影響がなかったかを確認するリグレッションテストは欠かせません。しかし,既存システムの規模が大きいとリグレッションテストにかかる工数は当然，膨大になります。再利用を可能とすることで，大幅な効率アップを図ることができます。

また，小規模の改造では，機能実現のコストに対して，調査・分析およびテストのコストが大きくなりがちです。しかし，網羅的なテストを前提に，テスト環境，テストケース，テストデータおよびテストシナリオ(テストの手続き

表 5.11　既存のテスト環境における再利用性の維持・向上の事例

No.	事　　例
1	リグレッションテストに関して，共通する項目，データおよびテスト環境などを調査・分析し，テスト自動化ツールを作成・導入した。テストの自動化により人手でのテストオペレーション工数およびデータ入力ミスによる手戻り工数を削減した。その後の案件にも自動化ツールを流用し，テストの生産性を向上させた。
2	専用帳票を使用してテストする場合に，編集カラム位置のプログラム修正がその都度発生していた。専用帳票を使わずに編集イメージが確認可能なツールを導入したことで，修正工数が減少し，かつ，システムテスト前のテスト工程でもユーザ企業による確認が可能となった。その後の案件にも確認ツールを適用してテストの生産性を向上させた。

を含む)などを再利用し，調査・分析のコストを抑え，システムの品質を維持することに成功しているシステムもあります。

表 5. 11で紹介する事例は，テストの自動化ツールを導入しテスト負荷を軽減するとともに再利用性を高めた例と，専用帳票の擬似確認ツールを作成して早期に可能とするとともに効率を高めた例です。

第6章　見積りに関する一般的事項と動向

6.1　改良開発に関して提案されている見積り方法

6.1.1　ファンクションポイント法における機能改良時の数え方

（1）　ファンクションポイント（IFPUG法）の計測法

ファンクションポイント（FP）の数え方は，「ソフトウェア開発見積りガイドブック」で基本的な考え方を紹介しています。本ガイドブックでは改良開発における数え方を説明しますが，まず，ファンクションポイント（FP）計測法の概要を確認しておきましょう。

FP法は，開発基盤に影響されないソフトウェア規模計測尺度を目指して開発され，データの入出力など，ユーザ企業に見える機能に着目し，機能数とその重みによりソフトウェア規模を定量化する手法です。今，世界で，また，日本でも最も多く使われているFP計測手法はIFPUG法です。その計測手法の詳細は，IFPUG発行のFunction Point Counting Practice Manual（以下CPMと略す）に記されています。

以下はその概要です。

（a）　機能の種別

IFPUG法で機能（function）として計測するものは，内部論理ファイル（ILF：Internal Logical File），外部インタフェースファイル（EIF：External Interface File），外部入力（EI：External Input），外部出力（EO：External Output），外部照会（EQ：External Inquiry）の5種があります（以下，それぞれ略称で示す）。

ILFとEIFはデータファンクション（Data Function）であり，EI，EO，EQは，トランザクションファンクション（Transaction Function）です。

（b）　機能の粒度

IFPUG法では，1機能の大きさは，1）ユーザ企業の視点からみて，2）論理的な，3）業務活動としての最小単位，と定義されています。

データファンクションの場合，「論理的なデータのまとまり」を1機能とします。すなわち，論理データ設計におけるエンティティなどが，1機能に該当します。

トランザクションファンクションの場合，要素処理(elementary process)と呼ばれる「ユーザ企業にとって意味があり，業務の最小単位」を1機能とします。ここでいう最小単位とは，「自己完結し，アプリケーションの業務を矛盾のない状態に保つ」範囲を指します。

(c) FP値の算出[13]

次に，洗い出した個々の機能に対し，重み付けを行います。IFPUG法では，重みは機能種別(ILF，EIF，EI，EO，EQの5種)と個々の機能の複雑さ(Complexity：低・中・高の3種)とのマトリクスで決定します。例えば，複雑さが「低」のILFならば何点というように点数が決まります。

複雑さは，その機能がいくつのデータ項目を持つか，いくつのファイルにアクセスするかなどで決定します。具体的には，データファンクションではレコード種類数(RET：Record Element Type)とデータ項目数(DET：Data Element Type)，トランザクションファンクションでは，関連ファイル数(FTR：File Type Referenced)とデータ項目数のそれぞれ2種類の数値を計測し，両者のマトリクスで決定します。マトリクスの値は，CPMに詳細に決められています。

こうして決めた個々の機能の重みを合計すると，アプリケーション全体のFP値が算出できます。

(2) IFPUG法における「機能改良時」のFP計測

IFPUG法では，「すでに提供されているユーザ機能を追加，修正，削除する開発プロジェクト」，すなわち本ガイドブックで言う改良開発を行うプロジェクトを「機能改良プロジェクト」と呼びます。IFPUG法では，計測種別として，「新規開発プロジェクトの計測」「機能改良プロジェクトの計測」「アプリケー

(13) ここではIFPUG法の未調整FPの算出までを説明しています。この後，手順としては調整要因の決定と最終調整済みFPの算出がありますが，この部分はJIS X 0135-1：1999(ISO/IEC 14143-1：1998)「ソフトウェア測定－機能規模測定－第1部：概念の定義」で定義された機能規模の範囲から外れるので，本ガイドブックでは説明を省略します。

ションの計測」の３種類が定められており，それぞれ計測の範囲と最終調整済みファンクションポイントの計算式が異なります。

(a)　計測範囲と計算式

３種類の計測種別における計測範囲は，それぞれ以下のように定義されています。このうち，機能改良FPが，改良開発の機能規模の数え方にあたります。

新規開発FP　　：プロジェクトの活動に影響を与える全機能
機能改良FP　　：追加，変更，削除されるすべての機能
アプリケーションFP：アプリケーションに含まれる全機能

これを式で説明すると以下のようになります。

新規開発FP：
　DFP ＝ ADD ＋ CFP
機能改良FP：
　EFP ＝ ADD ＋ CHGA ＋ DEL ＋ CFP
アプリケーションFP：
　初期段階(新規開発後)　AFP ＝ ADD
　機能改良後　AFP ＝(UFPB ＋ ADD ＋ CHGA)－(CHGB ＋ DEL)

DFP　：新規開発FP
DFP　：機能改良FP
AFP　：アプリケーションFP
ADD　：開発により追加された機能量(FP)
CHGA：開発により変更された後の機能量(FP)
CHGB：開発により変更される前の機能量(FP)
DEL　：開発により削除された機能量(FP)
CFP　：移行のために追加された機能量(FP)
UFPB：機能改良が行われる前の母体の機能量(FP)

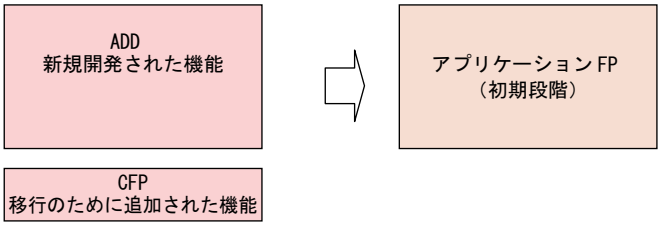


図 6.1　新規開発時の計測対象とアプリケーションFP

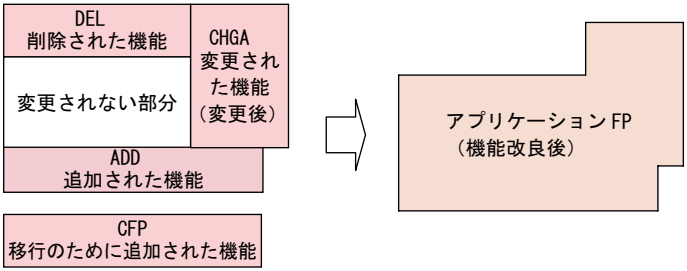


図 6.2　機能改良時の計測対象とアプリケーションFP

(b)　変更ファンクションの計測

ファンクションは，どの計測種別の場合も同じ「(1)ファンクションポイント(IFPUG法)の計測法」で説明した手法で計測します。

機能の変更の場合，「郵便番号7桁化」のように1データ項目の桁数の変更から出力内容の大幅な変更まで，変更の程度はさまざまですが，計測においてそれらは考慮しません。その機能に変更があるかないかだけで判断します。例えば，4FPのEO一つすなわち，CHGBは4 FP)に変更が加わり，変更後に5 FPになれば，CHGAは5 FPとなります。(CHGBとCHGAは同じ値で，変わらない場合もあります)変更の内容は問いません。

ファンクションの変更は，機能的な変更に限ります。例えば，「印刷用紙が変わった」とか，「データを印刷する位置が変わった」というような，機能要件に関係のない変更については，ファンクションの変更とはみなしません。ファンクションの変更とは，①入出力するデータ項目(DET)の変更，②参照するデータファンクション(FTR)の変更，③処理ロジックの変更のどれか一つ以上に該当する場合を言います。

（３）　機能改良FPの利用

ここで説明した機能改良FPの計測法は，「機能要件の変更の総量をどう表すか」という問いに対する答えです。あくまで機能規模を表すことを目的としており，開発工数との相関については保証されていません。実際，ADD，DEL，CHGAのそれぞれにおける生産性は異なっており，これらの総和から簡便に工数を見積ることは難しいと言えます。しかし，この値は機能としての変更量を表すもので，ユーザ企業とベンダ企業との間で，開発範囲を合意する場合には，有効な値となります。

一方，機能改良における工数見積りにおいては，本ガイドブックの他章において解説しているように，影響範囲，テスト範囲の量を把握する必要があります。この場合，該当する範囲の量をアプリケーションFPとして計測するという形で，FPを利用することができます。

6.1.2　COCOMOでの保守見積り

既存のソフトウェアを利用した開発や，既存システムに対する機能拡張や仕様変更，欠陥修正のための案件の見積りに対して，COCOMO IIは以下の二つのモデルを提供しています。

（１）　再利用モデル（Reuse Model）
（２）　保守モデル（Maintenance Model）

本節では，この二つのモデルの利用範囲と基本的な考え方を解説します。なお，この解説は，主にCOCOMO IIに関する書籍[14]に基づき，補足的にCOCOMO 81に関する書籍[15]を参考にしています。

（１）　COCOMOの概要と系譜

COCOMOは，1970年代にTRW社（主に軍需，精密機器のソフトウェア開発企業）でのプロジェクト実績データを基に作成され，Boehm教授によって1981年に最初のモデルCOCOMO 81が提唱されました。以降，繰り返し型開発，オブジェクト指向開発，1980～990年代のツールや開発環境の変化などに対応

(14)　Barry Boehm：Software Engineering Economics, Prentice Hall, 1981
(15)　Barry Boehm and 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

し，1995年に汎用化し適用範囲を広げたCOCOMO IIとして発表されました。この考え方に基づきUSC COCOMO II.2000が発表されており，現在，一般にCOCOMOというとこのモデルのことを示すことが多いようです。COCOMO II：2000には，適用場面ごとにいくつかのモデルがあります。

・Application Composition Model：
　GUI生成ツールを利用するプロトタイプ開発プロジェクト向き
・Early Design Model：
　製品のアーキテクチャが決定していない開発初期段階向き
・Post-Architecture Model：アーキテクチャ決定以降の段階向き

また，より適用範囲を広げたモデルの精緻化，拡張も進められており，以下のものが提唱されています。

・CORADMO（COnstructive RAD schedule estimation MOdel）：
　RAD開発用のモデル
・COPSEMO（COnstructive Phased Schedule and Effort MOdel）：
　進化型，小規模・短期型開発用のモデル
・COCOTS（COnstructive COTS integration cost model）：
　COTS（Commercial Off The Shelf）と呼ばれる商用のコンポーネントを使う開発用のモデル
・COPROMO（COnstructive PROcess improvement MOdel）：
　技術やプロセス導入による開発生産性改善に関するモデル
・COQUALMO（COnstructive QUALity MOdel）：
　欠陥混入と除去の品質に関するCOCOMO IIの拡張モデル

（2）　モデル式の考え方

(a)　規模計算式

工数予測式に入力するサイズ（規模）は，以下の式で計算します。単位はKLOC。

$$\text{サイズ} = (1 + 0.01 \times \text{REVL}) \times (\text{新規開発コード行数} + \text{等価ソースコード行数})$$

要件の変動性REVLは，見積り以降の要件の追加や変更により廃棄される

コードの割合(パーセンテージ)です。開発後稼働するソフトウェアのコード行数に対する比率です。例えば，10kステップ稼働するソフトウェアを開発する途中で，2kステップ廃棄するとREVLは20です。

等価ソースコード行数は，既存のソフトウェアコード(対象システムに統合されていない)を再利用したり修正して利用したりする場合に必要になります。詳細は再利用モデルで定義されています。

(b)　工数予測式

Early Design，Post-Architectureとも工数を予測する計算式は共通しています。

$$\text{工数} = A \times (\text{サイズ})^{E} \times \Pi\text{コストドライバ}$$
$$E = B + 0.01 \times \Sigma\text{規模要因}$$

(c)　開発期間予測式

Early Design，Post-Architectureとも開発期間を予測する計算式は共通しています。

$$\text{開発期間} = C \times (\text{工数})^{F}$$
$$F = D + 0.2 \times 0.01 \times \Sigma\text{規模要因} = D + 0.2 \times (E - B)$$

(3)　COCOMO Reuse(再利用)

再利用モデルは，既存のソフトウェア(ソースコード)をそのまま，あるいは修正して対象システムに結合する場合に利用します。

再利用コード(reused code)とは，ブラックボックスとして扱われ，プロダクトにプラグインされる既存コードです。適応コード(adapted code)は，ホワイトボックスとして扱われ，プロダクトに利用するために修正されうる既存コードです。

このようなコードを利用して開発する場合に，工数予測式の入力となるサイズは，再利用コードや適応コードの実効サイズを新規コードのサイズに等価なものに調整したものを利用します。調整されたコードは，等価ソースコード行数(ESLOC：Equivalent Source Lines Of Code)と呼ばれます。再利用モデルは，

この等価ソースコード行数を計算するためのモデルです。

(a)　等価ソースコード行数計算式

既存のソースコードを再利用する開発における等価ソースコード行数(単位はKLOC)は，以下の式で計算します。

等価ソースコード行数＝適応ソースコード行数×(1－0.01×AT)×AAM
AAF＝(0.4×DM)＋(0.3×CM)＋(0.3×IM)
AAF≦50の場合：AAM＝ 0.01×[AA＋AAF(1＋(0.02×SU×UNFM))]
AAF＞50の場合：AAM＝ 0.01×[AA＋AAF＋(SU×UNFM)]

(b)　コスト変動要因

適応調整要因 AAF(Adaptation Adjustment Factor)

対象システムに結合する既存のソフトウェアの変更による工数に対する影響を反映するためのものです。適応調整要因は，以下の三つの要因を予測した結果を利用します。

- DM(デザイン改修割合)：新しい対象や環境に適応させるため，適応されたソフトウェアの設計の変更されたパーセンテージ(これは必然的に，主観的な量である)。
- CM(コード改修割合)：新しい対象や環境に適応させるため，適応されたソフトウェアのコードの変更されたパーセンテージ。
- IM(結合改修割合)：適応されたソフトウェアをプロダクト全体に対して結合するため，および結合結果のソフトウェアをテストするための工数を，同様のサイズのソフトウェアに対する結合とテストに必要な平均的な工数と比較してのパーセンテージ。

ソフトウェア理解度 SU(Software Understanding)

既存のソフトウェアの理解しやすさにより工数が変動することを反映するためのものです。

改修有効性調査 AA(Assessment and Assimilation)

再利用・適応されるソフトウェアモジュールが適応先のソフトウェアに適し

表 6.1　再利用モデルのパラメータのガイドライン

コード種別	DM	CM	IM	AA	SU	UNFM
新　規	適用不可	適用不可	適用不可	適用不可	適用不可	適用不可
適　応	0-100％ 通常＞０％	0-100％ 通常＞DM かつ＞０％	0-100＋％ 通常＜100％ であるが ＞100％も可能	0-8％	0-50％	0－1
再利用	０％	０％	０-100％ ０％は稀 非常に小さい 値は取りうる	0-8％	適用不可	適用不可
COTS	０％	０％	０-100％	0-8％	適用不可	適用不可

ているかを判断し，適応先のソフトウェアのコードにそのコードを結合するために必要な工数を反映するためのものです。

改修対象PGへのなじみ　UNFM（Programmer Unfamiliarity）

再利用・適応する既存ソフトウェアに対してプログラマがどの程度のなじみがあるかによって，そのソフトウェアを理解するのに必要となる工数が変動することを反映するためのものです。

自動変換割合　AT（Automated Re-engineering）

再利用・適応する既存ソフトウェアを自動変換することによって適応先のソフトウェアに適合させることができるコードの割合（パーセンテージ）です。自動変換するコードに対する工数は，他の工数とは別に予測します。

（４）　COCOMO Maintenance（保守）

保守には，新たな機能の追加，欠陥の修正，既存の機能の変更を含みます。既存のソフトウェアの50％以上の変更や，既存のソフトウェアにはほとんど作業を必要としない大規模な（20％以上の変更を伴う）関連システムの開発は除きます。

【以下はCOCOMO 81に関する書籍[16]で示された適用範囲】

ソフトウェア保守の範囲でないものは以下の場合です。

- ほぼ同じ機能を持つ新しいソフトウェアのための大きな再設計や再開発（新規コード50%以上）。
- 大規模な（既存ソフトウェアのソースコードの20%以上）関連システムの設計や開発で，既存ソフトウェアの比較的小さい再設計を必要とする。
- データ処理システムの操作，データ入力，データベース中の値の変更。

ソフトウェア保守の範囲に該当するのは以下の場合です。

- 既存のソフトウェアの小さな範囲（新規コード50%未満）の再設計や再開発。
- 小規模な関連システムの設計や開発で，既存ソフトウェアで多少の再設計を必要とする。
- 既存ソフトウェアのコード，文書，データベース構造の変更で，更新（機能的な仕様の変更），修繕（機能的な仕様は維持）とも含む。修繕には，欠陥の修正，環境への対応，性能や保守性向上がある。

（a）　保守サイズ計算式

保守工数を見積もる計算式は，COCOMO II Post-Architecture Modelと同じです。計算式の入力となるサイズとしては，保守サイズを使用します。保守サイズ（単位はKLOC）は以下の式で計算します。ベースコードサイズは，保守対象の既存ソフトウェアのサイズです。

$$\begin{aligned} \text{保守サイズ} &= \text{ベースコードサイズ} \times \text{MCF} \times \text{MAF} \\ &= (\text{追加サイズ} + \text{修正サイズ}) \times \text{MAF} \end{aligned}$$

（b）　コスト変動要因

保守変更要因　MCF（Maintenance Change Factor）

保守変更要因は，保守対象の既存ソフトウェアに対する変更の割合を示します。保守により追加や変更するサイズとベースコードサイズにより，以下の式で計算します。削除するサイズは利用しません。

---

(16)　Barry Boehm and , 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

MCF＝(追加サイズ＋修正サイズ)／ベースコードサイズ

MCFを計算するときのサイズの単位は，ソースコード行数，ファンクションポイントなどが利用できます。ファンクションポイントを利用する場合には，保守により変更される入出力，画面，帳票などを計測するよりも，変更されるアプリケーション全体に対する割合を予測します。

保守調整要因 MAF(Maintenance Adjustment Factor)

保守調整要因は，ソフトウェアの理解度とプログラマの保守対象へのなじみ度合いの影響を考慮して，実行保守サイズを調整するために利用します。保守調整要因は以下の式で計算します。ソフトウェア理解度(SU)と対象へのなじみ度合い(UNFM)の定義は再利用モデルと同じものを利用します。

MAF＝1＋(0.01×SU×UNFM)

(c)　保守工数予測式

保守工数を予測する計算式は，Post-Architectureとほとんど同じです。

保守工数＝A×(保守サイズ)^{E}×Πコストドライバ

ただし，以下の違いがあります。

- SCED(開発期間要求)コストドライバは使用しない。通常，保守サイクルは固定の期間があるため。
- RUSE(再利用性要求)コストドライバは使用しない。保守対象のコンポーネントの再利用性を考慮して保守することにより増加する工数は，コンポーネントを注意深く設計，文書化，テストを行うことによって減少する保守工数により，大まかにはバランスが取れているため。
- RELY(信頼性要求)コストドライバは異なる評価値を取る。評価対象は，保守対象のソフトウェアの信頼性要求である。信頼性要求が低いソフトウェアの潜在欠陥を修復することは，より多くの工数が必要となる。信頼性要求が大変高い状態で開発されたソフトウェアは，保守に必要な工数は平均

表 6.2　RELY 保守コストドライバ

レベル	Very Low	Low	Nominal	High	Very High
評価値	1.23	1.10	1.00	0.99	1.07

的なレベルよりも多くなる(**表 6.2**)。

・工数予測式に入力するサイズは，保守対象ソフトウェア全体のサイズではなく，追加および修正したコードのサイズを保守調整要因MAFで調整した実効保守サイズを使用する。

(d)　保守期間予測

保守期間は要望する保守期間を予測期間として使用します。保守工数，保守期間，平均保守要員レベルの関係は，以下の式で表します。

保守工数＝保守期間×平均保守要員レベル

6.2　保守見積りに関する研究動向

保守に関する見積りは，実践においても研究においてもまだ途上にあると言っても過言ではありません。

本節では，保守見積りに関連した論文誌や国際会議を挙げるとともに，その中からいくつかの論文についてその概要を紹介します。

6.2.1　論文誌，国際会議

ソフトウェア保守に関する研究成果は，各種の論文誌や国際会議などで活発に議論されています。主な論文誌，国際会議は次のとおりです。

[論文誌]

・IEEE Transactions on Software Engineering（IEEE Computer Society）
・Journal of Software Maintenance and Evolution：Research and Practice（John Wiley & Sons）
・Empirical Software Engineering（Springer Netherlands）
・Information and Software Technology（Elsevier）

・Journal of Systems and Software（Elsevier）

［国際会議］

・International Conference on Software Engineering（ICSE）（ACM SIGSOFT and IEEE TCSE）
・International Conference on Software Maintenance（ICSM）（IEEE）
・International Symposium on Empirical Software Engineering（ISESE）（ACM SIGSOFT and IEEE TCSE）
・International Symposium on Software Metrics（METRICS）（IEEE Computer Society）

なお，ISESEとMETRICSは併合され，2007年よりInternational Symposium on Empirical Software Engineering and Measurement（ESEM）として新しい国際会議が発足する予定です。

6.2.2　保守見積りに関連する論文

ここでは，保守見積りに関連する論文について簡単に紹介します。

新規開発の見積りでも同じですが，保守見積りで取られるアプローチでも，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），積み上げ法（プロジェクトの成果物の構成要素を洗い出し，それぞれに必要な工数を見積もって積み上げる），パラメトリック法（工数などを目的変数として，説明変数に規模や要因などを設定し，数学的な関数として表す）の三つが利用されています。以降で紹介する論文では，例えば，パラメトリック法を利用する場合に説明変数として利用すべきメトリクスやモデル構築方法を提案し，ある特定組織で実施したプロジェクトのデータに適用することで，有用性の評価を行っているものが多く見られます。したがって，提案されている具体的なモデルは，それぞれの組織のプロジェクトデータに依存した結果です。当然ながら論文に書かれているモデルを別の組織がそのまま利用してもよい見積り結果を得られる保証はありません。モデル作成のプロセスを自社のデータに当てはめて行うことで，自社に特化したモデルを構築する必要があります。もちろん，ソフトウェア開発・保守に関するデータ収集が定常的に実現されていなければなりません。

（1） 見積りモデルに関する論文

Magne Jφfrgensen：Experience with the accuracy of software maintenance task effort prediction models, IEEE Transactions on Software Engineering, Vol.21, No.8, pp.674～681（1995）

回帰モデル，ニューラルネットワークなどの幾つかの方法により，ソフトウェア保守工数の見積り式を11種類考案しています。更にこれらの式を，ノルウェーのソフトウェア開発組織より収集したソフトウェア保守作業（タスク）のデータを用いて比較評価しています。収集データは，タスクの種類（修正保守，適合保守，完全化保守，予防保守），タスクの優先度，保守担当者の知識や経験年数，タスクに要した時間，タスクの規模（追加・変更・削除された行数），変更の内容などです。本収集データに対しては，多変量回帰モデルに基づく予測式が最も精度が高いことが報告されています。ただし，見積りモデルはエキスパートの見積りに取って代わるものではなく，過去のデータに基づく参考値としてうまく利用すべきであると結論づけています。

Alain Abran, Ilionar Silva and Laura Primera：Field studies using functional size measurement in building estimation models for software maintenance, Journal of Software Maintenance and Evolution：Research and Practice, Vol.14, pp.31～64（2002）

COSMIC-FFP法を用いて保守見積りを行った2種類のフィールドスタディについて紹介しています。最初の対象はWebアプリケーションに対する機能拡張の15プロジェクトで，二つ目の対象は組込み系リアルタイムソフトウェアに対する保守に関する19プロジェクトです。保守見積りモデルには，機能規模だけでなくプロジェクトの環境要因（開発者の経験，プロジェクトの困難さなど）を考慮する必要があることを述べています。

Rajendra K. Bandi, Vijay K. Vaishnavi and Daniel E. Turk：Predicting maintenance performance using object-oriented design complexity metrics, IEEE Transactions on Software Engineering, Vol.29, No.1, pp.77～87（2003）

オブジェクト指向ソフトウェアに対するメトリクス（Interaction level,

Interface Size, Operation Argument Complexityをそれぞれ評価するもの）を用いて，保守案件ごとの保守時間（既存プログラムの理解から修正まで）の予測可能性について実験的な評価を行っています。

Magne Jørgensen and Kjetil Molløkken-Østvold：Reasons for Software Effort Estimation Error：Impact of Respondent Role, Information Collection Approach and Data Analysis Method, IEEE Transactions on Software Engineering, Vol.30, No.12, pp.993～1007（2004）

あるソフトウェア開発組織で実施された68プロジェクトのデータや見積り関係者に対するインタビューを通じて見積りエラーの原因を分析しています。例えば，見積りに用いたデータの収集方法の違いやデータ分析方法の違いが，どの程度見積りエラーに影響しているかを調べています。

Liguo Yu：Indirectly predicting the maintenance effort of open-source software, Journal of Software Maintenance and Evolution：Research and Practice, Vol. 18, pp. 311～332（2006）

保守見積りモデルを構築する際には，保守案件に対する実績保守工数データの収集が必要ですが，その手間はかなり大きいと言えます。本論文では，データ収集の手間を削減するために，同一ドメインの過去のプロジェクトにおける保守作業データを分析し，保守工数と相関のある自動収集可能なメトリクスを特定し，それらを利用して保守見積りモデルを構築する手法を述べています。

その他，見積りモデルに関する論文を以下に示します。

- F. Calzolari, P. Tonella and G. Antoniol：Maintenance and testing effort model by linear and nonlinear dynamic systems, Information and Software Technology, Vol.43, pp.477～486（2001）
- Barbara Kitchenham, Shari Lawrence Pfleeger, Beth McColl and Suzanne Eagan：An empirical study of maintenance and development estimation accuracy, Journal of Systems and Software, Vol.64, pp.57～77（2002）
- Andrea De Luciaa, Eugenio Pompellab and Silvio Stefanuccic：Assessing

effort estimation models for corrective maintenance through empirical studies, Information and Software Technology, Vol.47, pp.3～15（2005）

・Tim Menzies, Zhihao Chen, Jairus Hihn, and Karen Lum：Selecting best practices for effort estimation, IEEE Transactions on Software Engineering, Vol.32, No.11, pp.883～895（2006）

（2） 見積りのためのメトリクスに関する論文

Fabrizio Fioravanti and Paolo Nesi：Estimation and prediction metrics for adaptive maintenance effort of object-oriented systems, IEEE Transactions on Software Engineering, Vol. 27, No. 12, pp.1062～1084（2001）

オブジェクト指向方法論で開発されたプログラムに対する適応保守（adaptive maintenance）を対象とした工数見積りのためのメトリクスを提案しています。提案されているメトリクスは，プログラム中のクラスの複雑さや規模，クラスで定義されているメソッドや属性に着目したものです。これらのメトリクスとあるプロジェクト（MOODS（Music Object-Oriented Distributed Systems）ESPRIT IVプロジェクト）で開発されたC++のプログラムに対する適応保守で収集した保守工数のデータとの比較を行い，本プロジェクトで有用なメトリクスについて議論しています。

Mikael Lindvall, Roseanne Tesoriero Tvedt, and Patricia Costa：Avoiding architectural degeneration：An evaluation process for software architecture, International Symposium on Software Metrics, pp.77～86（2002）

ソフトウェアの保守性をソフトウェアのアーキテクチャの複雑さから評価することを目指した論文です。保守作業後のソフトウェアの構造は保守前に比べて退化する（構造が複雑になるなど）ことが指摘されており，理想的には保守作業後に退化した部分の修繕（例えばリファクタリング）を行うことが，将来保守を行う上では有用です。一方，修繕作業には新たな工数・時間を要するため，修繕作業を行うべきかどうかを判定するための評価指標が必要となります。本論文では，その一つの解決策としてモジュール結合度に基づいてアーキテクチャの複雑さを評価するためのメトリクスを提案しています。あるソフトウェア開発プロジェクトにおける保守作業の前後で計測を行い，保守性評価の可能性について述べています。

Yunsik Ahn, Jungseok Suh, Seungryeol Kim, and Hyunsoo Kim：The software maintenance project effort estimation model based on function points, Journal of Software Maintenance and Evolution, Vol.15, No.2, pp.71～85（2003）

ファンクションポイント（IFPUG法）を用いて保守工数の見積りを行う手法を提案しています。従来の機能拡張ファンクションポイントとは異なり，新しい調整要因を提案しています。具体的には，開発者のスキル，保守対象ソフトウェアの特性，保守環境の三つを評価するために，それぞれ3個，4個，3個の調整要因を提案しています。小規模な保守を行った26プロジェクトで収集したデータに対して提案手法の適用実験を行っています。

Ales Zivkovic, Ivan Rozman and Marjan Hericko：Automated software size estimation based on function points using UML models, Information and Software Technology, Vol.47, pp.881～890（2005）

UML設計仕様書からファンクションポイントの自動計測を目的として，ファンクションポイントの概念のUML図上の表記との対応について述べています。

6.3　見積りモデル構築のためのCoBRA法の適用

改良開発においても，ユーザ企業とベンダ企業が，見積りにおいて十分なコミュニケーションを行い，見積りの根拠として，双方で前提条件と見積り方法を共有することができれば，プロジェクトの成功確率を向上させることが可能となります。そのためには，ユーザ企業とベンダ企業の双方で納得性の高い見積り方法（見積りモデル）の設定が重要です。具体的な見積り方法としては，3.3節に詳細を示したように，見積りのベースラインとその変動要素を設定し，定量化することが勧められます。

では，変動要素を洗い出したとして，その定量化をどのように行えばよいでしょうか。COCOMOが実現しているように多数のデータを蓄積して，変動要素を説明変数とした多変量分析を行い，定量化することは一つの方法です。しかし，変動要素のデータを含めた多数の過去プロジェクトデータの蓄積が実現できている例が少ないのが現実です。

この課題を解決し，見積りモデルの構築する方法として，「ソフトウェア開発見積りガイドブック」で紹介したCoBRA法[17]があります。CoBRA法の特徴は，比較的少ないプロジェクトデータからでも，プロジェクトや見積りの熟練者の知見を活用・加味することによって，見積りモデルを構築する点です。過去のプロジェクトの実績に基づいて，熟練者の知見を活用して，見積りの構造（モデル）を探り出していく方法です。例えば，まず3.3節で示したような変動要因について両者で合意をした上で，熟練者の知見に基づいて変動要因による影響度合いの定量化を行い，過去のプロジェクトデータでその妥当性を検証しながら見積りモデルを構築することができます。また，この定量化は，過去のデータとプロジェクトの熟練者の知見を活用して分析した結果で，関係者にとって見積りの根拠が納得性の高いものになります。

現在，国内企業においても，ユーザとベンダ双方の納得性を向上させて合意できる見積りプロセスを築きあげることを目的にCoBRA法によるモデルを構築し，コミュニケーションツールとして活用しようと試みている先進企業があります。

6.4　改良開発見積りの応用範囲

改良開発はすでに構築したシステムに対して機能を新たに追加したり，既存の機能を変更したり，削除したりするものです。その特徴は，開発に当たって既存システムやプログラムが存在することにあり，このことから，本ガイドブックで示しているように，新規開発の見積りで考慮すべき変動要因に加えて，改良開発の見積りで考慮すべき変動要因などが生じています。

ここで改めて，2.1節で定義した改良開発のほかに，すでに存在するシステムまたはプログラムを利用して開発するケースを考えてみると，次に列挙する事例が思い浮かびます。

これらのケースには，全く同じように改良開発の見積り方法を適用できないまでも，考え方をはじめ，工程（調査・分析，機能実現，テスト）や変動要因として捉えておくべきことなど，かなりの部分を応用することができます。今後，それぞれのケースに合わせて，改良開発の見積りを応用し発展させていくこと

(17)　Cost estimation, Benchmarking, and Risk Assessment 法

が必要です。

(a)　仕様変更

ここでいう仕様変更とは，システムの開発途中で発生する仕様の変更を指します。仕様変更見積りは改良開発見積りとは異なり，仕様変更内容に関する認識の合意および開発途中の半製品に手を加えることによる仕様変更量(特に変更棄却量)などを考慮する必要がありますが，既存システムがある開発という点では共通しています。

(b)　繰り返し開発

アジャイル型の開発を含め最近の開発パラダイムで採用されている繰り返し開発手法は，既存システムを改造して開発するという点において，改良開発と共通しています。繰り返し開発手法では，一連の繰り返し開発の間はプロジェクトの編成を維持して,既存システムへの習熟度を維持すること,およびリファクタリングなどのシステムの保守性を向上する活動をあらかじめ開発プロセスに組み込んでいます。

繰り返し開発手法は，仕様をあらかじめ固定せずむしろ変わるものであることを受け入れ，要求変化に迅速に対応可能にしています。そのため何回かの繰り返し開発を経て，最終的にシステムが仕上がるまでの総コストをいかに見積もるか，仕様をあらかじめ固定して開発した場合のコストと，いかに比較するかが見積りの課題になります。

(c)　再利用開発

類似の機能を持つシステムを再利用して開発したり，類似の機能を持つプログラムを再利用してあらたなプログラムを開発したりすることは，システム開発の中で日常的に行われています。

これも，既存システムがある開発という点では共通しています。しかし，既存機能を維持しなければならないという制約条件が，改良開発と比較して低くなります。また，再利用するものを設計書に限定するというような場合もあり，その範囲によって多くのバリエーションがあり得ます。既存システムを再構築する場合などは，既存システムの要求仕様や設計の一部を再利用した開発と捉えることもできます。

再利用開発の場合の見積りについては，本ガイドブックの6.1.2項において，COCOMOでの保守見積りとしてCOCOMO Reuse（再利用）として紹介しています。

(d)　パッケージソフトウェアを利用した開発

パッケージソフトウェアを利用した開発は，パッケージソフトウェアそのものを既存システムと捉えると，既存システムがある開発という点では共通しています。

ただし，パッケージソフトウェアは通常その内部を隠蔽してブラックボックスとしていますので，多くの場合は改良開発よりは再利用開発との共通事項が多くなります。

6.5　これからのコストモデル

コストモデルの基本的な考え方は，ソフトウェアの開発から維持管理にいたる諸活動を対象にして，過去の経験，プロジェクトデータに基づき，より客観的で普遍的な法則や統計的な性質を導出し，理論体系として構築していくことです。これらの理論体系を実務に適用していく観点からは，プロジェクト当事者のみならず，発注者，出資者，経営者といった利害関係者の意思決定に寄与するものでなくてはなりません。これはコストモデルの用法，すなわち，予測，説明，交渉，経営，プロジェクト管理などといった局面での理論の活用方法を明確にしていくことを意味しています。

従来の見積り手法では，規模と工数との関係に焦点をあてていますが，経営的な視点からは，投下した資本で開発を進め，その開発費用に見合う価値をソフトウェアが生み出すことが要請されます。さらに，自由市場経済を考慮すれば，ソフトウェアコンポーネントやサービスを調達するということは，それを入手，使用，所有するために対価を適正な競争下で市場から調達するということになり，ソフトウェアの見積り手法もこうした，価値，価格，費用とを総合的に扱う理論体系として構築していく必要があると考えられます。

今後の見通しとしては，プロジェクトマネジメント，見積り手法は分野ごとにより精緻になるとともに，構築と運用，開発と保守といった現行で複数の文化圏の異なる知識体系や標準が錯綜しているものも統合・再編成が図られると

考えられます。

また，社会的な要請やニーズも高まり機が熟して新しい技術の発現が求められています。組込みシステムや製造業でのプロダクトライン，サービス産業でのサービスエンジニアリングやそれを支えるサイエンスの台頭，アジャイルプロセスに代表される市場やビジネスモデルと密接にかかわる手法の普及などが新技術や新しい社会構造変革の契機となっていくと考えられます。そして，ソフトウェアエンジニアリングより広い学問分野として発現，形成されるのではないかと予想されます。

見積り技術に関して言えば，経済の問題と合わせて考えなくてはならず，これを若干煽動的ないい方をするとソフトウェア経済学の発現が望まれているということになります。見積り・コスト分析，企業価値算定方式，プロジェクトマネジメント技術などを包括的に取り扱う経済モデルが現在は存在していません。ソフトウェア経済学という名前を拝すかどうかは別にしても実質的にこういった統一的な理論が発現する必要があると考えています。ソフトウェアは，実験的に開発される小さなプログラムから，社会インフラを支える大規模で永続的なものまで，規模感もさまざまですし，これに携わる陣容も個人，チームレベルから大規模プロジェクトや複数の企業体連合にいたる粒度があります。見積りについてもミクロモデルからマクロモデルといった整備を進めていく必要があるでしょう。経済学でも近年，制度論や配分，さらには，個人の意思決定や心理まで踏み込んだ行動経済学や行動ゲーム論の提唱もなされており，コストモデルや見積り技術の用法を整備していくのに有効と考えられます。

第2部
事　例　編

第１章　事例編の見方

ここでは，改良開発の見積りについて，先導的な取り組みを行っている各社の事例を紹介します。

各事例は，原則として，以下のような構成となっています。なお，事例提供企業の要望(企業の固有用語)により，「改良」を「改造」として記載している場合があります。

（１）　取り組みの背景

見積り活動に関する各社のこれまでの取り組み，当該見積り手法を利用するに至った経緯，保守見積りに関する問題意識などについて記述しています。

（２）　見積り方法(モデル)

当該見積り方法におけるモデルの説明(入力パラメータ，出力，基本的なアルゴリズム・方式など)を記述しています。

（３）　見積り方法の前提条件

当該見積り方法を利用するために必要となる諸条件(見積り時期，見積り対象，見積り活動，併用見積り手法，体制・役割分担・企業文化など)について記述しています。

（４）　精度向上のための活動

当該見積り手法の継続的な改善活動(見積り値と実績値との差異分析，フィードバックなど)について記述しています。

（５）　実施実績

当該見積り方法を実際に適用した分野(業種，システムプロジェクトの特徴)，適用した見積り時期とその精度などについて記述しています。

（６）　当該見積り方法の優位点と課題

当該見積り方法のアピールポイント，今後の課題，利用に当たって留意すべき点などについて記述しています。

なお，ここで紹介している事例は，次のとおりです。**表1.1**には，各事例の概要(見積り時期，見積り対象，入力データなど)を示しています[18]。

これらの事例はあくまで各社で利用している見積り手法の一つという位置づけであり，それぞれ社内で当該手法のみを用いていることを示すものではないことに注意してください。

ー日立製作所手法

母体の調査・分析工数，正味改造部分の設計・開発・テストの開発規模，母体の確認テスト工数からの見積り。

ージャステック手法

独自の生産管理に基づく見積りモデルで，基本概念(成果物量見積り方式と生産性見積り方式から成立)は新規開発見積り(「ソフトウェア開発見積りガイドブック」掲載)と同様。

ー富士通手法

見積り時期に応じた画面に基づくファンクションスケール(Function Scale)による見積り。ファンクションスケールは，富士通で提唱する規模尺度。基本概念は新規開発モデル(「ソフトウェア開発見積りガイドブック」掲載)と同様。

(18)　見積り時期に応じて複数の方法が示されている企業については，表1.1において黄色い網掛けが施してあるものを中心に事例を紹介しています。

表 1.1　各社見積り手法概要

手法名	対象分野	見積り時期	見積り対象	入力データ
日立製作所手法	汎用	超概算見積り：改造要件(機能要件，非機能要件)を入力にして，母体の調査が完了した時点	規模，工数	母体規模(SLOC)，追加規模(SLOC)，変更規模(SLOC)，削除規模(SLOC)，調査・分析の生産性(SLOC/人日)，母体の確認テストの生産性(SLOC/人日)
		概算見積り：改造要件(機能要件，非機能要件)を入力にして，母体の分析が完了した時点		
		確定見積り：正味改造部分の設計が完了した時点		
ジャステック手法	汎用	要件定義後(以降各工程で適用可)	規模(各工程の生産物量)，生産性(各工程生産物の単位量あたりの工数またはコスト)，工数(各工程の生産物別)	・次のいずれかを選択(新規開発も改造型開発も同様) ①基本設計の規模(文字数) ②類似システムまたは再構築対象のソースコード行数 ③要求定義書に記述されているシステムの入力，出力および機能 ・改造型開発特有の入力データ ①改造開発生産物量影響度(テスト工程のみ対象でテスト巻き込み規模) ②改造開発生産性影響度(設計からコーディング工程のみ対象で改造密度，改造分散度，改造母体錬度)
富士通手法	Webシステム	試算見積り：改造案件発生時点	規模 作業量	変更範囲と影響範囲の規模，母体に占める割合
		概算見積り：改造方針決定後	規模(FS) 作業量	各画面のFS(注１)，見積り基準値 (注１)は備考欄参照
		確定見積り：設計完了後	規模(FS) 作業量	各画面のFS(注１)，見積り基準値

表1.1　各社見積り手法概要(つづき)

手法名	見積り式(パラメータ)	各手法の適用実績
日立製作所手法	改造プロジェクトの見積り工数 ＝母体の調査・分析の工数 ＋正味改造部分の設計・製造・テストの工数 ＋母体の確認テストの工数 (1)母体の調査・分析の工数 ＝母体規模/調査・分析の生産性 (2)正味改造部分の設計・製造・テストの工数 ＝正味改造部分の設計・製造・テストの開発規模 /新規開発プロジェクトの設計・製造・テストの標準生産性 正味改造部分の設計・製造・テストの開発規模(SLOC) ＝追加規模＋α×変更規模＋β×削除規模＋γ×母体規模 (α, β, γ：標準係数) (3)母体の確認テストの工数 ＝母体規模/母体の確認テストの生産性	・1970年代より，他の見積り手法と併用。既に数千件の改造プロジェクトに適用している。
ジャステック手法	・各工程の開発コスト ＝当該工程の改造生産物量×当該工程の改造生産性 ・当該工程の改造生産物量 ＝当該工程の標準生産物量 ×(1＋改造開発生産物量影響度) ×(1＋新規開発時の生産物量環境変数＋改造開発特有の生物量環境変数) ・当該工程の改造生産性 ＝当該工程の標準生産性 ×(1＋改造開発生産性影響度) ×(1＋新規開発時の生産性環境変数＋改造開発特有の生産環境変数)	・既存システムのあるソフトウェア開発を行う全てのプロジェクトについて適用している。 ・2005年度は，開発プロジェクト・保守プロジェクトを合わせて300件以上の改造プロジェクトに対して適用。
富士通手法		・実プロジェクトにおいて試行中。
	改造部分の作業量(設計・製造・テスト) ＝Σ(追加画面の概算規模(FS)×追加見積り基準) ＋Σ(変更画面の概算変更規模(FS)×変更見積り基準)	
	改造部分の作業量(製造・テスト)＝Σ(各画面改造作業量) 画面改造作業量＝オブジェクト追加規模(FS)×追加見積り基準 ＋オブジェクト変更規模(FS)×変更見積り基準 ＋オブジェクト削除規模(FS)×削除見積り基準 確認テストの作業量 ＝テスト範囲(方針により決定)の規模(FS)×テスト実施率 ×テスト見積り基準	

表 1.1　各社見積り手法概要（つづき）

経営層や現場の理解の得やすさ	適用コスト：習得時	適用コスト：実際の見積り算出時（規模計測時）の所要時間
・PMO（Project Management Office）が，各事業部の見積りプロセスの成熟度に応じて，見積りプロセス及び見積り手法の継続した改善を推進。 ・見積りレビュー後には，プロジェクトリスクランクに応じて，見積り決裁者やPMOメンバを招集した公式見積り会議を開催し，受注に関する方針やリスク軽減策について審議。	・改造見積りを含めた見積り手法のマニュアルやガイドラインを中心に提供し，必要な時に学習可能とした。 ・プロジェクトマネージャ及びSEなどを対象としたトレーニング（座学，eラーニング）を常時開設。	
・会社設立当初より，生産性原理に基づき「量に基づく価格設定および技術者の能力評価をを標榜」ということでトップダウンで推進。 ・上記を実現するために，全てのプロジェクトが同じ方式で見積もられ，横並びの比較が可能となるようにすることが大前提。	・新入社員研修の一環としてPSP（Personal Software Process）を実施。2 週間程度の期間で自分たちの計画を作成し，見積り結果が，どのくらいの精度を持っているかという感覚をつかんでもらう。	・新規顧客の案件は時間がかかる。初回の開発を通して確実に実績が蓄積され，差異の原因も明らかになり，顧客との認識の統一も図れる。 ・特に，継続している保守開発などは，変化している部分に注力するため，作成時間はその分，早くなる。
・広報やWeb公開を通じて，社内への情報提供を実施。 ・プロジェクト活動のプロセス（見積りに関する審査会，組織による監査等）において，当見積り方法の適用有無のチェックや指導を行う仕組みを整備すべく準備中。	・最初のレクチャーとして30〜40分程度（スプレッドシートなど）。	・1 画面当たり 5 分程度で計測が可能。 ・画面上に表示される要素を計測するため，分かりやすく計測しやすい。

表 1.1　各社見積り手法概要（つづき）

手法名	適用を可能にする前提となる，組織体制（社内の機能）	適用を可能にする前提となる，蓄積したデータ・情報の種類	備　　考
日立製作所手法	・2003年に，PMO内の組織として，見積り手法の整備，精度向上，定着化活動をミッションとする専任センタを設置。 ・見積りレビューなどを通じて見積り手法の適用方法やノウハウを提供。 ・プロジェクト完了時には，プロジェクト完了報告会に開発規模や工数などの見積り値と実績値の差異分析，評価結果などを含めた事例発表を行うよう制度化。	・PMOが，完了したプロジェクトの実績データ（プロジェクトプロフィール情報，プロジェクトの開発規模（FP，SLOC），母体規模，開発工数，開発期間，品質など）を収集。 ・PMOは収集したデータを精査して，標準生産性，標準品質値，及び改造見積り手法の標準係数を半年に１回の頻度で再設定し，各事業部へ提供。	・改造見積りとしては，本手法のほか，ボトムアップ見積り手法，類推見積り手法，COCOMO見積り手法など，複数の手法を併用することを推奨。
ジャステック手法	・生産管理システムの全体像を「標準開発計画書」として，開発部門が作成。標準開発計画書は改造型開発を含んでいる。これを受けて，営業部門が「標準見積書」を作成。 ・標準開発計画書は開発部門のプロセス改善推進部署が必ず目を通す。	・全社統一基準として定め，それに則って実績データを蓄積。 ・見積りモデルは，自社独自の生産管理システム（技術者個人の生産性と品質に基づく評価尺度と連携している）に基づいて構築されている。	・プログラムの改造箇所などを特定可能な場合には，改造正味規模を新規開発見積りにて算出し，過去の実績データを元に正味規模単位のコスト（または工数）を換算する簡便方式を併用。
富士通手法	・手法のまとめ・推進を行う推進部門を社内に設置。 ・新規開発では推進部門による見積り支援を実施しているが，改造の場合見積りはプロジェクトで実施し，技術的な指導や実績収集・フィードバックを推進部門が行う。	・現在試行を繰り返しながら実績データの蓄積を進めている。	・複数の方法による見積りで検証することを推奨しているが，FS法以外には見積り方法を特定していない。 ・おおむね，（１）作業項目ごとの作業量の積上げ見積り，（２）改造率から換算した改造係数により新規開発時の見積り基準を調整して見積もる簡便方法の２つが利用されている。 （注１）画面などの単位で，機能を「追加」「変更」「削除」「変更無ー影響有」「変更無ー影響無」に分けて，それぞれの規模を概算する。

第 2 章　日立製作所

2.1　取り組みの背景

企業は競争力を維持するためにタイムリーにビジネス環境の変化に対応することが求められている。そのために重要な要素の一つとして，タイムリーにITシステムを成長させること，あるいは再構築することが必要であると考える。ITは進歩が早いので，スクラップ＆ビルドによるITシステムの刷新が増加すると予想される。しかし，景気の動向によりITシステムへの投資は大きく左右されることも事実である。一方で，ITシステムが企業活動と密接に結びつき，継続的にITシステムを発展させたい要求も根強いので，改造プロジェクトは増加することが予想される。

改造プロジェクトの見積りでは，実際に改造する部分に関する作業のほかに，改造前の既存システム全体を意味する「母体」に関する作業を考慮する必要がある。母体の調査・分析が完了していない時点では，下記のような要因が明確でないため，改造プロジェクトの開発プロセスの全体を正確に見積もることが困難である。

（1）　保守用ドキュメントの整備状況

- 保守用ドキュメント(既存システム開発時の設計書，テストデータおよびチェックリストなど)が残されているか。
- 過去の改造内容が，保守用ドキュメントに正確に反映されているか。

（2）　母体の整備状況

- 改造が必要となる部分の特定が，容易な構造になっているか(母体の理解容易性)。
- 設計・製造・テストが容易な構造になっているか(母体の改造容易性)。
- 母体の確認テストをする場合に，改造の影響が予想外の部分に影響を与えないような構造になっているか(母体へ与える影響)。

（3）　母体の習熟度

・既存システムを開発・改造した経験がない場合，あるいは担当者の世代交代があった場合に，母体を十分に理解できるか。

そこで，当社情報・通信グループでは，上記の課題に対して，過去プロジェクトの実績データを多く蓄積することにより，改造見積り手法の一つとして，係数モデル見積り手法を整備し継続的に改善してきた。

既刊書の「ソフトウェア開発見積りガイドブック」第 2 部第 5 章で，当社の新規開発プロジェクトの見積り手法を紹介した。今回は，改造プロジェクトの見積り手法として母体の調査・分析の段階で適用できる係数モデル見積り手法の事例を紹介する。

2.2　見積り方法（モデル）

本事例では，改造プロジェクトを「既存システムを母体とし，母体に対する追加，変更および削除によって既存システムを新システムへ再構築すること」と定義する。母体に対して，追加，変更および削除する部分をまとめて，正味改造部分と定義する。改造プロジェクトの開発プロセスを**図 2.1**に示す。

改造プロジェクトは，新規開発プロジェクトと比較して，正味改造部分だけではなく，改造を加えない母体に対しても既存の機能を保証するために下記の作業を多く必要とすることが特徴の一つである。

・正味改造部分が母体へ与える影響の分析（影響範囲の特定）
・正味改造部分が母体へ影響を与えていないかの確認テスト（母体の確認テスト）

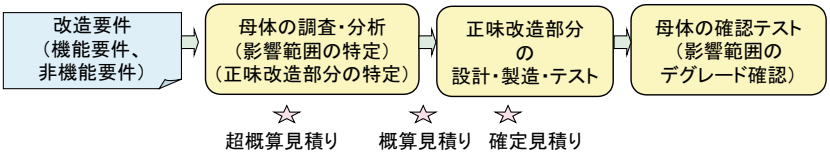


図 2.1　改造プロジェクトの開発プロセス

見積り時期は，下記を3回実施することが標準手順である。

・超概算見積りの時期：改造要件(機能要件，非機能要件)を入力にして，母体の調査が完了した時点
・概算見積りの時期：改造要件(機能要件，非機能要件)を入力にして，母体の分析が完了した時点
・確定見積りの時期：正味改造部分の設計が完了した時点

本事例では，母体の調査が完了した時点および母体の分析が完了した時点で適用できる係数モデル見積り手法を紹介する。また，改造プロジェクトの見積り工数は次式で算出する。

改造プロジェクトの見積り工数
＝母体の調査・分析の工数 ＋ 正味改造部分の設計・製造・テストの工数
＋ 母体の確認テストの工数

2.2.1　母体の調査・分析の見積り手法

母体の調査・分析の工数には，母体規模，保守用ドキュメントの整備状況，母体の整備状況および母体の習熟度などが大きな影響を与える。したがって，既存システムを開発・改造した経験がなく母体を理解していない場合は，母体の調査・分析の見積り手法は，母体の調査・分析に必要となるワークパッケージごとに工数を積み上げるボトムアップ見積りが標準手順となる。

一方,既存システムを開発・改造した経験があり母体を理解している場合は，母体の調査・分析の工数は母体規模と強い相関があることを実証しているので，係数モデル見積り手法により次式で算出する。

$$\text{母体の調査・分析の工数(人日)} = \frac{\text{母体規模(LOC)}}{\text{調査・分析の生産性(LOC/人日)}}$$

なお，調査・分析の生産性(LOC/人日)は標準係数を設定している。保守用ドキュメントと母体の整備状況の品質が良い場合は，調査・分析の生産性は高い値で安定する。一方，保守用ドキュメントと母体の整備状況の品質が悪い場合は，調査・分析の生産性は小さい値で不安定となる。そこで，ユーザ企業の

母体(既存システム)ごとに実績データを把握し，調査・分析の生産性を再設定することが見積り精度を確保するためには重要である。

2.2.2　正味改造部分の設計・製造・テストの見積り手法

正味改造部分の設計・製造・テストの見積り手法を，母体の調査が完了した時点および母体の分析が完了した時点で説明する。

(1)　母体の調査が完了した時点

母体の調査により母体のソフトウェア構造をおおむね理解し，正味改造部分を業務メニュー(または業務サブメニュー)などの大きな機能集合の単位で特定する。次に特定した正味改造部分である大きな機能集合の単位ごとに，追加，変更および削除が必要になるソースコード数を見積もる。

(2)　母体の分析が完了した時点

母体の分析により母体のソフトウェア構造を理解し，正味改造部分をプログラム単位で特定する。次に特定した正味改造部分であるプログラム単位ごとに，追加，変更および削除が必要になるソースコード数を見積もる。

上記の(1)または(2)で見積もった追加規模，変更規模および削除規模を，次式に代入し，新規開発プロジェクトの開発規模に換算した正味改造部分の開発規模を算出する。

正味改造部分の設計・製造・テストの開発規模(LOC)
＝追加規模＋(α×変更規模)＋(β×削除規模)＋(γ×母体規模)

なお，α，β，γは標準係数を設定している。ただし，ユーザ企業の母体(既存システム)ごとに過去プロジェクトの実績データを把握し，α，β，γを再設定することが見積り精度を確保するためには重要である。

正味改造部分の設計・製造・テストの工数見積りは，正味改造部分の開発規模を新規開発プロジェクトの設計・製造・テストの標準生産性(SLOC/人日)で割って標準工数を算出する。なお，新規開発プロジェクトの標準生産性は，非機能要件，開発言語ごとに設定している。

2.2.3　母体の確認テストの見積り手法

母体の確認テストの工数には，母体規模，保守用ドキュメントの整備状況，母体の整備状況，および母体の習熟度などが大きな影響を与える。既存システムを開発・改造した経験があり母体を理解している場合は，母体の確認テストの工数は母体規模と強い相関があることを実証しているので，係数モデル見積り手法により次式で算出する。

$$\text{母体の確認テストの工数(人日)} = \frac{\text{母体規模(LOC)}}{\text{母体の確認テストの生産性(LOC/人日)}}$$

なお，母体の確認テストの生産性(SLOC/人日)は標準係数を設定している。

保守用ドキュメントと母体の整備状況の品質が良い場合は，母体の確認テストの生産性は高い値で安定する。一方，保守用ドキュメントと母体の整備状況の品質が悪い場合は，母体の確認テストの生産性は小さい値で不安定となる。そこで，ユーザ企業の母体(既存システム)ごとに実績データを把握し，母体の確認テストの生産性を再設定することが見積り精度を確保するためには重要である。

2.3　見積り方法の前提条件

2.3.1　見積り時期

本改造見積り手法の見積り時期は，母体の調査が完了した時点および母体の分析が完了した時点を想定している。

2.3.2　見積り対象

ビジネスアプリケーションソフトウェアの改造プロジェクトを対象としている。

2.3.3　見積り活動

プロジェクトマネージャまたはSEが責任を持って，見積もることが基本である。PMOは，見積りレビューなどを通じて見積り手法の適用方法やノウハウを，プロジェクトマネージャまたはSEに提供している。PMOの支援により，

見積り手法の適正な適用に寄与していると考えている。見積りレビューの後に，プロジェクトリスクランクに応じて，見積り決裁者やPMOメンバを召集した公式見積り会議で，受注に対する方針やリスクを軽減するための方策について審議している。

2.3.4　併用見積り手法

改造見積りは，本事例で紹介した係数モデル見積り手法だけでなく，ボトムアップ見積り手法，類推見積り手法，COCOMO(Constructive Cost Model)見積り手法などの複数の見積り手法を併用して適用することを推奨している。

また，2000年より，ビジネスアプリケーション向けITシステム開発の標準システムアーキテクチャおよび標準開発プロセスを適用したケースに関して，再利用率も考慮した係数モデル見積り手法を再整備し実証実験を行いながらモデルの改善に取り組んでいる。

2.3.5　体制・役割分担・企業文化

当社では，見積りプロセスを強化し定着させる活動に，1995年よりFP法を応用した見積り手法を展開してきた。また,2003年より，ⅰ)見積り手法を整備し，ⅱ)見積り精度向上施策を立案・実行し，ⅲ)当社内への見積り手法の定着化活動などをミッションとする専任センタ(PMOの一部の組織)を設置した。これは見積りプロセスの強化に対する経営陣の強力なコミットメントがあり実現できた。その結果として，ユーザ企業から頂くRFP(Request For Proposal)に記載している機能要件をベースにFPを簡単に計測できる要素見積法と呼ぶ見積り手法を整備し,当社内に定着化させた。要素見積法の詳細に関しては,「ファンクションポイント法を応用した早期見積技法の提案とそのシステム化」(電子情報通信学会論文誌D，Vol. J89-D，No. 4，pp. 755～766，2006年4月)を参照してください。

また，各事業部の見積りプロセスの成熟度に応じて，見積りプロセスおよび見積り手法の継続した改善を推進している。この結果として，従来と比較して見積り根拠が明確になり，プロジェクト管理不良も低減してきていると考えている。見積り偏差も徐々に小さくなってきており,経営陣やプロジェクトマネージャの理解も得やすくなっている。

改造見積りを含めた見積り手法の普及施策の一つとして，習得時のコストを

低減するために，マニュアルやガイドラインを中心に提供し，必要なときに学習できるようにした。また，プロジェクトマネージャおよびSEなどを対象としたトレーニング(座学，eラーニング)を常時開設するなどの工夫を行っている。

2.4　精度向上のための活動

2.4.1　差異分析，フィードバックの方法

プロジェクトが完了すると，プロジェクトマネージャはプロジェクト完了報告書を提出すると同時に，プロジェクト完了報告会で事例を発表することを制度化している。このプロジェクト完了報告会では，開発規模・工数などの見積り値と実績値の差異を分析し原因と評価結果なども含めて報告するように制度化している。この結果として，ベストプラクティスが当社内に蓄積され活用されている。さらに，半年に最低1回は，各事業部の事業状況，プロジェクトの成功失敗の結果を分析し，見積りプロセスを改善している。

2.4.2　精度向上のための具体的な方法

PMOは，見積り精度を向上させるために，完了したプロジェクトの実績データを収集・精査し統計的に処理し，標準生産性，標準品質値，および改造見積り手法の標準係数を，半年に1回の頻度で再設定し各事業部に提供している。過去プロジェクトの実績データで，収集する主な項目は，プロジェクトプロフィール情報，プロジェクトの開発規模(FP，SLOC)，母体規模，開発工数，開発期間，品質などである。

さらに，工数見積り手法として，2006年より，SEC(ソフトウェア・エンジニアリング・センター)の指導を受け，新規開発プロジェクトおよび改造プロジェクトに対して，CoBRA(Cost Estimation，Benchmarking，and Risk Assessment)法の実証実験を開始し2007年度より適用を予定している。なお，CoBRA法の詳細に関しては，既刊書の「ソフトウェア開発見積りガイドブック」を参照してください。

2.5　実 施 実 績

2.5.1　適用分野（業種，システム・プロジェクトの特徴）

1970年代より本改造見積り手法は，他の見積り手法と併用し，数千件の改造プロジェクトで適用されている。

2.5.2　見積り時期とその精度

本改造見積り手法の見積り時期を，母体の調査が完了した時点および母体の分析が完了した時点として，工数見積り精度を説明する。

（1）　母体の調査が完了した時点

母体の調査が完了した時点では，ⅰ)既存システムを開発・改造した経験があり母体を理解しており，既存システムの過去プロジェクトの実績データを把握し改造見積り手法の係数を再設定している場合は，ほぼ正確な工数見積りができる。

一方，ⅱ)既存システムを開発・改造した経験がなく改造見積り手法の係数として標準係数を用いる場合は，非常に粗い工数見積りとなる。したがって，母体の分析が完了した時点で，再見積りをすることが重要である。

（2）　母体の分析が完了した時点

母体の分析が完了した時点では，ⅰ)既存システムを開発・改造した経験があり母体を理解しており，既存システムの過去プロジェクトの実績データを把握し改造見積り手法の係数を再設定している場合は，正確な工数見積りができる。

一方，ⅱ)既存システムを開発・改造した経験がなく改造見積り手法の係数として標準係数を用いる場合は，粗い工数見積りとなる。したがって，正味改造部分の設計が完了して時点で，ボトムアップ見積りなどにより再見積りすることが重要である。

2.6　当該見積り方法の優位点と課題

2.6.1　当該見積り手法の優位点

既存システムを開発・改造した経験があり母体を理解しており，既存システムの過去プロジェクトの実績データを把握し改造見積り手法の係数を再設定している場合は，母体の調査または分析が完了した時点で，改造プロジェクトの全体に対して，簡単にほぼ正確な工数見積りができる。

一方，既存システムを開発・改造した経験がなく改造見積り手法の係数として標準係数を用いる場合は，母体の分析が完了した時点で，改造プロジェクトの全体に対して，簡単に粗い工数見積りができる。

2.6.2　不得意な分野など，利用するに当たって留意すべき点

新規に取引するユーザ企業に対して，改造見積り手法の係数の妥当性を説明することが難しく，ユーザ企業とベンダ企業で合意した見積りスコープのベースラインになりえないのが問題である。また，新規に取引するユーザ企業などで改造見積り手法の係数として標準係数を用いる場合は，母体の分析が完了した時点でも，粗い工数見積りしかできない。

改造見積り手法の係数は，保守用ドキュメントの整備状態，母体の整備状況，および母体の習熟度などにより大きく影響を受ける。これらの影響要因を考慮した，透明性を確保した改造見積り手法のロジックを整備することが今後の課題である。

第3章　ジャステック

3.1　取り組みの背景

弊社は，創業以来，役務提供を前提とした人月単価から脱却し，一括請負契約を拡大すること，および生産性原理に立脚した能力主義を実践するために独自の生産管理システム「ACTUM」を構築し運用している。生産管理システムの中心がソフトウェア開発の見積りモデルである。その見積りモデルは，2006年４月発刊の「ソフトウェア開発見積りガイドブック」の「第９章ジャステック手法」で紹介した。受託ソフトウェア開発の価格は開発量と正の相関関係にあることを前提としており，開発量はソフトウェア開発工程ごとに作成する生産物の量として可視化している。

見積りモデルの基本アルゴリズムは，生産物量見積り方式および生産性見積り方式から成り立ち，さらに各々の方式において開発環境の違いや品質要求の多寡による変動を吸収する「環境変数」と呼ぶパラメータを導入している。本稿でご紹介する改造開発見積りは新規開発の拡張として捉え，新規開発見積りを包含するものとして位置づけている。

3.1.1　現在の方法に至る経緯，改造開発見積りに関する問題意識

上記の小冊子では，改造開発の三つの特質および改造開発固有の環境変数について述べた。特質の一つ目は量の特質（改造正味規模，母体規模，テスト巻き込み規模），二つ目は改造内容を特定する指示書（改造設計書），三つ目は改造対象システムの調査にかかわるドキュメント（復元ドキュメントなど）である。環境変数は新規開発見積りの環境変数を併用するとともに，四つの改造開発固有の副環境特性を定めている。また，改造開発見積りは新規開発見積りに比べ成熟度の点において検討すべき課題を含むことを述べた。

近年，改造開発はますます増加している。その要因はソフトウェア保守，再利用によるソフトウェア開発および組込み系のエンハンス開発などの増大があげられる。その増加に伴い改造開発見積りの重要性が高まっており，開発コス

トの妥当性検証への期待はもとより，改造開発の特質に関する理解不足によるトラブルの回避など，プロジェクト管理面(見積りは開発計画やプロジェクトコントロールへのインプット)での期待も大きい。

本稿では，その後の研究と実証を通して精錬した見積りモデル(改造開発)の詳細内容をご紹介する。特に精錬したポイントである改造開発の生産性見積り方式，生産物量(テスト規模)見積り方式，および新たに評価(日本情報システム・ユーザー協会「生産性評価プロジェクト」のご支援含め)した改造開発固有の環境変数について焦点をあて述べる。

3.2　改造開発の見積り基本アルゴリズム

3.2.1　新規開発の見積り基本アルゴリズム

当社が考案した新規開発におけるコスト見積りモデルの基本アルゴリズムを示す。本アルゴリズムは，「ソフトウェア開発見積りガイドブック」で紹介した。その中で，ある開発工程 i の標準生産物量を V_i^B，標準生産性を P_i^B で表現すると，生産物量 V_i，生産性 P_i および開発コスト C_i は次式で求まることを述べた。

$$V_i = V_i^B \times (1 + a_i) \qquad (a_i = \Sigma \alpha_{ij})$$

$$P_i = P_i^B \times (1 + b_i) \qquad (b_i = \Sigma \beta_{ij})$$

$$C_i = V_i \times P_i$$

a_i は生産物量環境変数と呼ぶパラメータであり，V_i^B に対して品質要求の多寡による変動を吸収する変数である。また，b_i は生産性環境変数と呼ぶパラメータであり，P_i^B に対して開発環境の違いや品質要求の多寡による変動を吸収する変数である。a_i，b_i は品質特性と環境特性から影響される独立した変動要素 (α_{ij}，β_{ij}) から構成されている。a_i は**表 3.2**に，b_i は**表 3.4**，**表 3.5**に示す。

3.2.2　改造開発の特質

(1)　生産性に影響を与える改造開発の特質

改造開発では，同じ量の改造を加える場合でも，改造母体に対する調査範囲，改造部分のばらつき度合いおよび母体錬度によって開発コストが変動する。改造開発において環境変数以外に生産性へ影響を及ぼすパラメータは三つある。

一つ目は追加および削除する量(以下，「改造正味規模」と呼ぶ)に対する改造

母体の量(以下,「改造母体規模」と呼ぶ)である。**図 3.1**に示すように，同じ改造正味規模であっても，改造母体規模が大きくなるほど調査にかかる手間がかかり，生産性が低下する。当社では影響度を定量的に捉えるにあたり，改造母体規模を適当な単位(10ks)に区切り，単位規模あたりの改造正味規模を算出する方式を採用する。この改造母体単位規模あたりに追加および削除する改造正味規模を「改造密度」と呼ぶ。

二つ目は改造母体に改造を加える箇所の数である。**図 3.2**に示すように，同じ量の改造を加える場合でも，改造する箇所数が増えるほど母体に与える改造

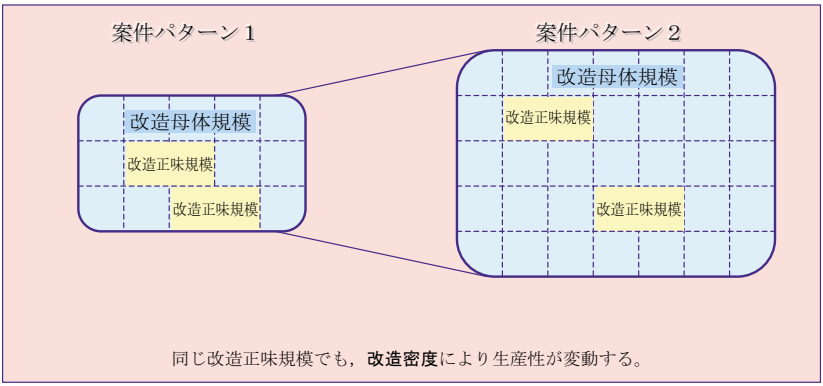


図 3.1　生産性に影響を与える改造開発の特質(改造密度)

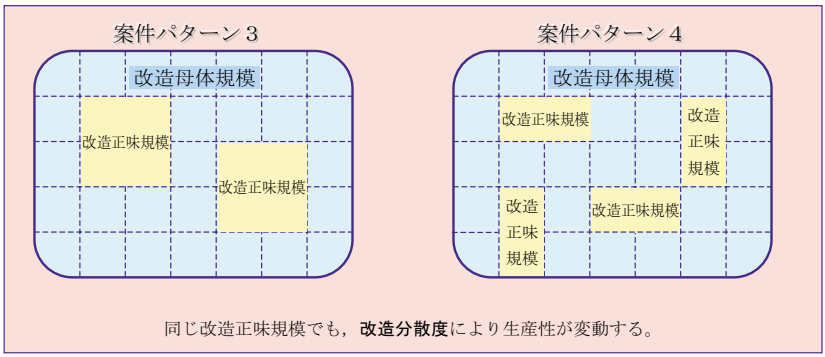


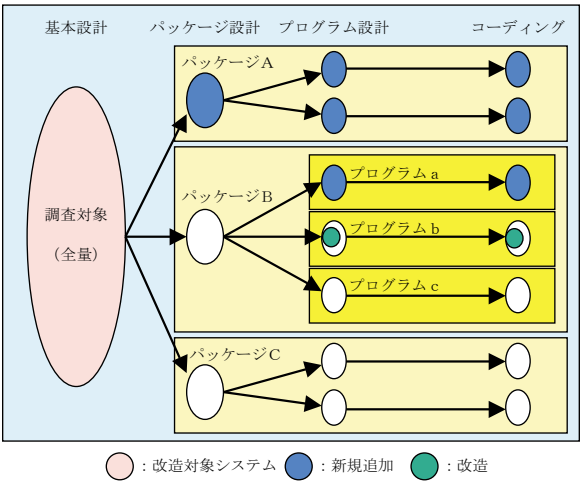
図 3.2　生産性に影響を与える改造開発の特質(改造分散度)

の影響を考慮する手間がかかり，生産性が低下する。影響度を定量的に捉えるにあたり，改造母体規模を適当な単位(10ks)に区切り，単位規模あたりの改造箇所数を用いて影響度を評価する方式を採用する。この改造母体単位規模あたりに追加または削除する改造箇所数を「改造分散度」と呼ぶ。

三つ目は改造母体に対する経験年数である。改造母体に対する開発経験が多ければ，生産性は高くなる。当社ではこれを「改造母体錬度」と呼び，経験年数１ヶ月未満，１年未満，１年以上の３段階で評価する。

（２） テスト量に影響を与える改造開発の特質

改造開発におけるテスト量(テスト項目数)は，改造正味規模に比例するとは限らず，改造要求の確認や非レベルダウンの確認のために，テストに巻き込む規模(以下「テスト巻き込み規模」と呼ぶ)に左右される。



パッケージＡ：改造のため追加するパッケージ
パッケージＢ：一部改造が加わるパッケージ
プログラム・・追加プログラムａ
改造プログラムｂ
未改造プログラムｃ
パッケージＣ：改造の加わらないパッケージ

図 3.3　改造開発量に関するモデル

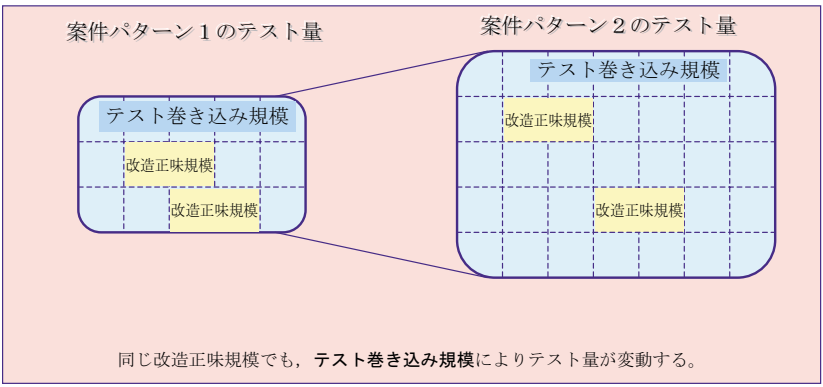


図 3.4　テスト量に影響を与える改造開発の特質（テスト巻き込み規模）

図 3.3に改造開発量に関するモデルを示す。当社基準における，このモデルの単体テスト範囲は新規追加プログラムおよび改造されたプログラムが対象となる。ただし，改造されたプログラムのテスト項目は改造内容および改造量により決定する。統合テスト範囲は新規追加パッケージおよび改造されたパッケージが対象となる。ただし，改造されたパッケージのテスト項目は改造内容および改造量により決定する。システムテスト範囲はすべてのパッケージが対象となり，テスト項目は改造内容および改造量により決定する。

改造開発においてテスト量へ影響を及ぼすのは「テスト巻き込み規模」である。

図 3.4に示すように，同じ改造正味規模であっても，テスト巻き込み規模が大きくなるほど改造による非レベルダウンの確認テスト項目が増加する。

テスト工程の改造開発量（テスト項目数）は，テスト巻き込み規模のテスト量を見積もり，さらに改造正味規模に該当するテスト量を加える方式を採用している。

3.2.3　改造開発の見積り基本アルゴリズム

改造開発のコスト見積りモデルの基本アルゴリズムは，新規開発見積りに，さらに改造開発の特質「3.2.2改造開発の特質」を考慮している。

ある工程 i における改造開発見積りの基本アルゴリズムを以下に示す。

$$V_i' = V_i^B \times (1+\delta_i) \times (1+a_i+a_i') \qquad (a_i' = \Sigma \alpha_{ij}')$$
$$P_i' = P_i^B \times (1+\gamma_i) \times (1+b_i+b_i') \qquad (b_i' = \Sigma \beta_{ij}')$$
$$C_i = V_i' \times P_i'$$

a_i'は改造開発特有の生産物量環境変数であり，b_i'は改造開発特有の生産性環境変数である。a_i'は**表3.3**に，b_i'は**表3.6**に示す。δ_iはテスト工程の改造開発生産物量影響度で，基本設計からコーディング工程の値はゼロである。γ_iは基本設計からコーディング工程の改造開発生産性影響度(「改造密度・改造分散・改造母体錬度変数」とも呼ぶ)で，テスト工程の値はゼロである。

なお，改造開発量V_i'および改造生産性P_i'の説明については，「3.2.3(1)改造生産性見積り方式および(2)改造開発生産物量見積り方式」に示す。

(1)　改造生産性見積り方式

前節で述べた改造開発の特質を生産性の見積り方式に取り込むため，新たに「改造開発生産性影響度」を導入する。改造開発生産性影響度は以下の三つが生産性に与える影響度として定義する。

- 改造密度(改造母体10ksあたりに改造追加または削除する量)
- 改造分散度(改造母体10ksあたりに改造追加または削除する箇所数)
- 改造母体錬度(改造母体に対する開発経験年数)

改造開発生産性影響度は**図3.5**に示すような「改造密度γ_i^l」，「改造分散度γ_i^m」，「改造母体錬度γ_i^n」の3次元マトリクステーブルから改造密度・改造分散・改造母体錬度変数γ_i^{lmn}を決定する。

ある開発工程iの改造開発生産性影響度γ_iは，改造正味規模のすべてを対象に3次元マトリクステーブルから改造密度・改造分散・改造母体錬度変数γ_i^{lmn}を得て，γ_i^{lmn}ごとの改造正味規模で加重平均して求める。よって，改造生産性P'_iは改造開発生産性影響度γ_iおよび改造開発特有の生産性環境変数b'_iから次式で求まる。

$$P_i' = P_i^B \times (1+\gamma_i) \times (1+b_i+b_i') \qquad (b_i' = \Sigma \beta_{ij}')$$

標準生産性P_i^Bは新規開発と同じである。改造開発においては，追加型改造

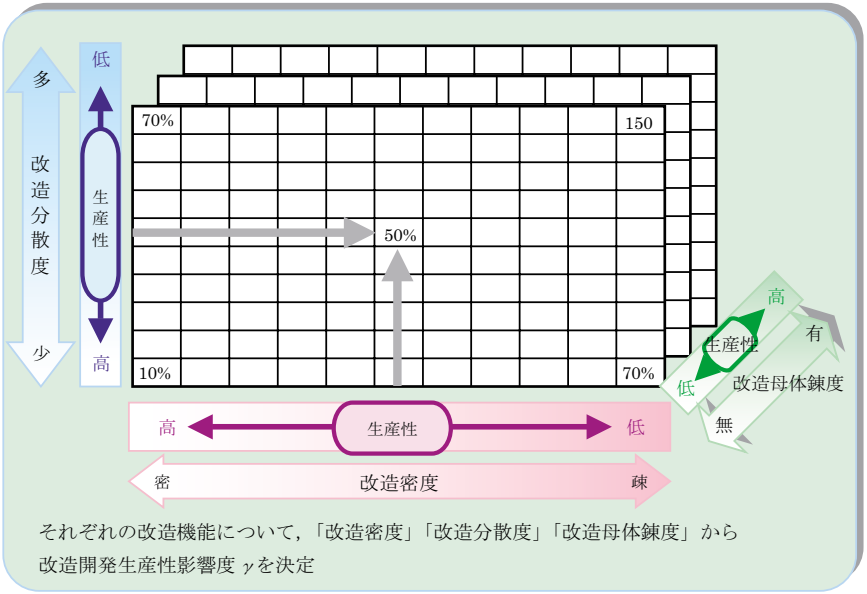


図 3.5　改造開発生産性影響度マトリクステーブル

と削除型改造があり，標準生産性（追加）をPA_i^B（$=P_i^B$）とすれば，標準生産性（削除）PD_i^Bは$PA_i^B/2$である。改造開発生産性影響度γ_iも改造開発生産性影響度（追加）γA_i，改造開発生産性影響度（削除）γD_iがある。

（2）　改造生産物量見積り方式

本方式で見積もる改造開発における生産物は，次工程への指示書となる「改造設計書」である。標準生産物量V_i^Bは上位工程（$i-1$工程）の生産物量をV_{i-1}とすると生産物量変換係数H_iを用いて次式で求まる。

$$V_i^B = V_{i-1}^B \times H_i$$

改造開発特有の生産物量環境変数をa_i'として表現すると，改造生産物量V_i'は次式で求まる。

$$V_i' = V_i^B \times (1 + a_i + a_i') \qquad (a_i' = \Sigma a_{ij}')$$

テスト量の見積りに関しては，上式にさらに「改造正味規模」，および「テスト巻き込み規模」をパラメータとして考慮する必要がある。

前節「3.2.2(2)テスト量に影響を与える改造開発の特質」で述べた改造開発の特質をテスト量の見積り方式に取り込むため，新たに「改造開発生産物量影響度」を導入する。

なお，基本設計工程からコーディング工程までの改造開発生産物量影響は「巻き込み」の必要がないためゼロである

テスト巻き込み規模は改造正味規模に該当する*ソフトウェアコンポーネントと関連するソフトウェアコンポーネントとを対象にするが，テスト要件の観点から二つのテスト巻き込み規模が存在する。

一つはコンポーネント間の接続性で，インタフェースを介して巻き込むテスト巻き込み規模(「インタフェース規模」と呼ぶ)である。インタフェース規模に対応する改造開発生産物量影響度は，改造正味規模に加えてテストすべきテスト巻き込み規模の改造正味規模に対する倍率とする。すなわち，ある開発工程iの改造開発生産物量影響度をδ_i，改造開発特有の生産物量環境変数をa_i'として表現すると，テスト量V_i'は次式で求まる。

$$V_i' = V_i^B \times (1 + \delta_i) \times (1 + a_i + a_i') \qquad (a_i' = \Sigma a_{ij}')$$

二つ目はコンポーネント間の非レベルダウンを担保するためのテスト巻き込み規模(「リグレッション規模」と呼ぶ)である。ただし，リグレッション規模にはインタフェース規模を含まない。どこまで確認するかは顧客側の要求をもとに見積もることになる。リグレッション規模は改造正味規模に較べ残存バグ率が少ないため生産性が高くなる。

リグレッション規模に対応する改造開発生産物量影響度は，改造正味規模に加えてテストすべきテスト巻き込み規模の改造正味規模に対する倍率とする。すなわち，ある開発工程iの改造開発生産物量影響度をδ_i'，改造開発特有の生

*ソフトウェアコンポーネントとは，ある意味をもつソフトウェアの塊で，モジュール，プログラム，パッケージおよびシステムなどを指す。

産物量環境変数をa_i'として表現すると，テスト量V_i''は次式で求まる。

$$V_i'' = V_i^B \times (1 + \delta_i') \times (1 + a_i + a_i') \qquad (a_i' = \Sigma \alpha_{ij}')$$

（3） 改造開発見積りモデルの運用

改造開発のコスト見積りモデルを導入し運用する際の工夫点をご紹介する。

① 改造母体の調査・分析

改造開発見積りは改造要件とともに改造母体に関する情報が入力となる。改造母体の情報を得るためには調査分析が必要である。調査・分析の成果物事例を**表 3.1**に示す。

② 設計からコーディング工程までのコスト見積り

設計からコーディング工程までのコスト見積りは，新規開発に比べて改造密

表 3.1　調査・分析の成果物事例

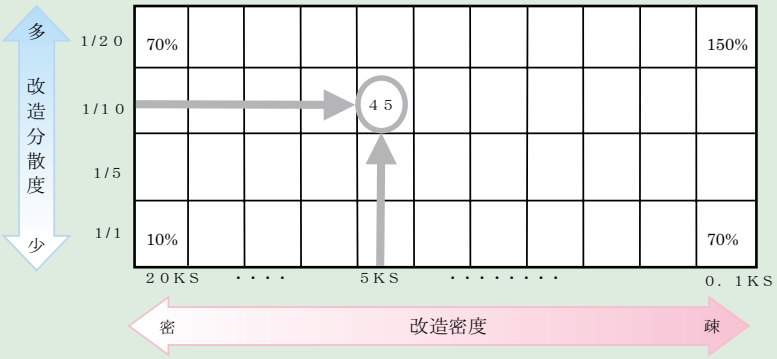
	設計～コーディング工程見積りのための調査・分析						統合テスト工程見積りのための調査・分析			
	改造母体規模（KS）	改造正味規模（KS）	改造密度	改造箇所数（箇所）	改造分散度	母体錬度	インタフェース規模（KS）	インタフェース規模に対応する改造開発生産物量影響度	リグレッション規模（KS）	リグレッション規模に対応する改造開発生産物量影響度
パッケージA	3.0	0.5	1.7	2	6.7	3 年	0.2	0.4	1.3	2.6
パッケージB	10.0	7.5	7.5	10	10.0	0 年	11.3	1.5	37.5	5.0
パッケージC	8.0	2.4	3.0	1	1.3	6 ヶ月	1.2	0.5	7.9	3.3
パッケージD	5.5	3.0	5.5	3	5.5	1 年	1.5	0.5	13.5	4.5

※網掛けは改造開発生産性影響度および改造開発生産物量影響度を決定する要素

度，改造分散度および改造母体錬度が生産性に影響することを「3.2.3（１）　改造生産性見積り方式」で述べた。

ここでは，設計からコーディング工程までのコスト見積りの算出方法を**コラム**欄に示す。改造開発生産性影響度マトリクステーブルのセル単位にコストを求め積算する方法である。

コラム（設計からコーディング工程までのコスト見積り）



セル

2	0	1
45	36	22

セル内の説明。

下段には三つ（「１ヶ月未満」/「１年未満」/「１年以上」）の改造母体影響度 γ_n を反映した改造密度・改造分散度・改造母体錬度変数 γ_{lmn} が存在し，該当する値を選択する。上段は下段で選択された γ_{lmn} の個数 f_{lmn}「度数」である。

ある工程 i の具体的な算出式を以下に示す。

$$C_i = \sum_{l}\sum_{m}\sum_{n}\left(f_{lmn} \times V_l \times (1+a+a') \times P^B \times (1+\gamma_{lmn}) \times (1+b+b') \right)$$

$$(a' = \Sigma\, a') \quad (b' = \Sigma\, \beta')$$

l は改造密度，m は改造分散度，n は改造母体錬度のサフィクスである。

V_l は改造密度（改造母体単位規模「１０ks」に対応する改造正味規模），

f_{lmn} は度数，γ_{lmn} は改造密度・改造分散度・改造母体錬度変数である。

③　テスト工程のコスト見積り

テスト工程のコスト見積りは，二つのテスト巻き込み規模が存在することを「3.2.3(2)改造生産物量見積り方式」で述べた。

ここでは，二つのテスト巻き込み規模を考慮したコスト算出式を示す。

・テスト巻き込み規模(インタフェース規模)のコスト「C_i'」

$$V_i' = V_i^B \times (1 + \delta_i) \times (1 + a_i + a_i') \qquad (a_i' = \Sigma a_{ij}')$$

$$P_i' = P_i^B \times (1 + b_i + b_i') \qquad (b_i' = \Sigma \beta_{ij}')$$

$$C_i' = V_i' \times P_i'$$

・テスト巻き込み規模(リグレッション規模)のコスト「C_i''」

$$V_i'' = V_i^B \times (1 + \delta_i') \times (1 + a_i + a_i') \qquad (a_i' = \Sigma a_{ij}')$$

リグレッション規模に対応するテストの標準生産性$P_i^{B\prime}$は新規開発におけるテストの標準生産性P_i^Bの 3 / 4 倍とする。改造開発特有の生産性環境変数をb_i'として表現すると，テストの生産性P_i'は次式で求まる。

$$P_i' = P_i^{B\prime} \times (1 + b_i + b_i') \qquad (b_i' = \Sigma b_{ij}')$$

$$C_i'' = V_i'' \times P_i'$$

(4)　生産物環境変数と生産性環境変数

改造開発は新規開発を包含することを述べた。ここでは，新規開発および改造開発に影響する環境変数に加え，現行資産の特性が生産物量および生産性へ影響を与える改造開発特有の環境変数を示す。

改造開発特有の環境変数の具体例を，現行資産特性変動要素評価表として**表 3.3**および**表 3.6**に示す。生産物量に影響する現行資産特性は「機能性」1 個である。生産性に影響する現行資産の主特性は「改造・再構築特性」1 個あり，さらに副特性として 7 個に細分化している。

表 3.2　品質特性変動要素評価表(規模に影響する外部環境変数)

$(a_i = \Sigma \alpha_{ij})$

主特性	副特性	変動要素	ベースラインからの変動率(%)			
			要件定義	設計	製作	テスト
機能性	合目的性	利用者/利害関係者の広がりコンテンジェンシー対応，不正移行データ対応などの該当事象数	0〜50	0〜50	0〜50	0〜50
	正確性	正確性(検証)に関わる標準テスト密度を基準にしたテスト項目量への要求水準	—	—	0〜20	0〜50
	接続性	他システムとの接続によるコード変換，フォーマット変換数	0〜5	0〜20	0〜20	0〜20
	セキュリティ	対応が必要なセキュリティ実現機能数，ただし機能要件に定義されている部分は除く	0〜20	0〜20	0〜20	0〜20
信頼性	成熟性	故障低減に必要な実現機能数	0〜5	0〜10	0〜10	0〜10
	障害許容性	異常検知に必要な機能数	0〜5	0〜10	0〜10	0〜10
	回復性	再開処理に必要な実現機能数	0〜5	0〜10	0〜10	0〜10
使用性	理解性	理解性向上(機能など)のためのプレゼンツールなどの作成対象数	—	0〜10	—	—
	習得性	習得性向上(使い方など)のためのマニュアルなど作成対象数	—	0〜10	—	—
	操作性	操作性向上(心理的/肉体的配慮，運用やインストール容易性など)のための実現機能数	0〜10	0〜20	0〜20	0〜20
保守性	解析性	解析に必要な実現機能数	—	0〜10	0〜10	0〜10
	変更作業性	作成する保守用ドキュメントの数	—	0〜10	—	—
	試験性	試験に必要な機能数	—	0〜5	0〜15	0〜20

表 3.3　現行資産特性変動要素評価表(規模に影響する外部環境変数)

$(a_i' = \sum \alpha_{ij}')$

主特性	副特性	変動要素	ベースラインからの変動率(%)			
			要件定義	設計	製作	テスト
機能性	正確性(既存母体)	改造/流用母体が正しく動作しない場合のテスト量(現行保証)に及ぼす影響	—	0〜5	0〜15	0〜20

※改造開発特有の環境変数

表 3.4　環境特性変動要素評価表（生産性に影響する外部環境変数）

$(b_i = \Sigma \beta_{ij})$

主特性	副特性	変動要素	ベースラインからの変動率（%）			
			要件定義	設計	製作	テスト
業務特性	業務ナレッジ	顧客の開発対象業務に対する業務ナレッジが生産性に及ぼす影響	−10 〜50	−10 〜10	—	−10 〜10
ハードウェア特性	安定度/信頼度/使用度	システムもしくは製品となるハードウェアの安定度・信頼度	—	−5 〜5	—	−5 〜5
ソフトウェア特性	安定度/信頼度/使用度	システム/製品に組み込む他社作成ソフトウェアもしくはCOTSの安定度・信頼度	—	−5 〜5	—	−5 〜5
コミュニケーション特　性	顧客窓口特性	意思決定能力（期限遵守，決定事項の覆る度合）	−10 〜20	−10 〜10	—	−7 〜7
	工期の厳しさ	基準工期（月）＝2.7×（人月）$^{1/3}$ に対し▲30%限度とした短期化度合	0〜10	0〜10	0〜5	0〜7
	コミュニケーション基盤	開発拠点分散，資料など情報共有，電子媒体・システム具備など物理的基盤充実度	−10 〜10	−5 〜5	−3 〜3	−3 〜3
	レビュー体制	無駄なレビュー（重複多段階など）の排除およびレビュー効率向上への工夫度合	−5 〜5	−5 〜5	−3 〜5	−5 〜5
開発環境特性	開発手法/開発環境	開発手法・環境（ソフト/ハード/ツール）の信頼性，占有率などを考慮した使用実績	−3 〜3	−3 〜3	−5 〜5	−5 〜5
	テスト手順書水準	テスト手順の具体化度（操作手順＆入出力の具体化の要求水準）	—	—	−3 〜3	−3 〜3
工程入力情報特性	業務関連資料	必要資料の具備状況（正確性，信頼性を含む）および使いやすさ（検索性，理解性）	−10 〜10	−7 〜7	−3 〜3	−3 〜3
	他システム関連資料	必要資料の具備状況（正確性，信頼性を含む）および使いやすさ（検索性，理解性）	−10 〜10			
	規約・標準化関連資料	必要資料の具備状況（正確性，信頼性を含む）および使いやすさ（検索性，理解性）	−7 〜7			
顧客の協力特性	役割分担特性	顧客がベンダ企業に協力する度合および顧客とベンダ企業との役割分担の明確性	−10 〜20	0 〜10	—	0 〜10

表 3.5　品質特性変動要素評価表(生産性に影響する外部環境変数)

$(b_i = \Sigma\beta_{ij})$

主特性	副特性	変動要素	ベースラインからの変動率(%)			
			要件定義	設計	製作	テスト
機能性	合目的性(要求仕様の網羅性)	要求の記述水準および網羅性。要件定義については新規性，方針明確性，ステークホルダの多様性などを考慮	0〜100	0〜30	—	0〜10
	正確性	正確性(検証)に関わる標準レビュー工数(各工程 8 %)を基準にした要求水準	0〜5	0〜5	0〜3	0〜5
	接続性	基準単位(100ks)に対する社内/社外システムとのインタフェース先の数	−10〜10	−5〜10	—	−5〜5
	整合性	整合をとる社内/社外の規格・基準の数，全体適合性やグローバル化対応も含む	−10〜10	−5〜10	−3〜5	−3〜5
効率性	実行効率性	実行効率に対する一般的要求水準(既知)の標準事例を基準にした要求水準	0〜5	0〜10	0〜5	0〜10
	資源効率性	資源効率に対する一般的要求水準(既知)の標準事例を基準にした要求水準	0〜5	0〜10	0〜5	0〜10
保守性	解析性	ソースコードの解析性をコード化規約に定めるコメントに対する要求水準により評価	—	—	0〜5	—
	安定性	ソフトウェア変更に対しシステム品質維持可能とする水準をライフサイクル目標年数の長さにより評価	0〜5	−3〜7	−3〜5	—
移植性	環境適用性 移植作業性 規格準拠性 置換性	ソフトウェアをどの程度，多様な環境に移すことができるかに対する要求の水準	0〜28	0〜28	0〜12	0〜25

表 3.6　現行資産特性変動要素評価表(生産性に影響する外部環境変数)

$(b_i' = \sum \beta_{ij}')$

主特性	副特性	変動要素	ベースラインからの変動率(%)			
			要件定義	設計	製作	テスト
改造・再構築特性	母体調査ツールの機能水準	調査ツールの機能の数 ・絞込み ・モニタリング(ルート解析) ・ドキュメント生成(リバース) ・実機での稼働確認 ・データディクショナリ ・その他調査ツール	—	−20〜15	−10〜5	−10〜5
	既存設計書具備状態	改造/流用母体の設計書有無および設計書が存在した場合のメンテナンス状態	0〜10	0〜30	0〜30	0〜25
	既存のテスト環境流用水準	既存のドキュメントおよびテストデータ(テスト環境を含む)の流用可能度合	—	—	−20〜0	−30〜0
	リソース管理水準	ソースとロードモジュールのバージョン管理水準	—	—	0〜15	0〜15
	既存母体品質(正確性)	既存システムが正しく動作しない場合の生産性に及ぼす影響度	0〜8	0〜10	0〜10	0〜10
	既存母体品質(解析性)	既存システムの改造箇所特定(改造設計)における標準化項目の遵守度合によるソースコードの解析容易性 ・ソースコメント ・修正履歴 ・モジュールサイズ ・規約 ・その他顧客規約事項	0〜10	0〜25	0〜25	0〜15
	既存母体品質(環境適用性)	既存システム資産を別環境へ移植し改造する開発における，別環境への適用容易性(適用可能な環境の個数) ・ハードウェア ・OS ・ネットワーク ・フレームワーク(ミドル) ・DB/DC ・バッチ運用システム	0〜10	0〜25	0〜25	0〜15

※改造開発特有の環境変数

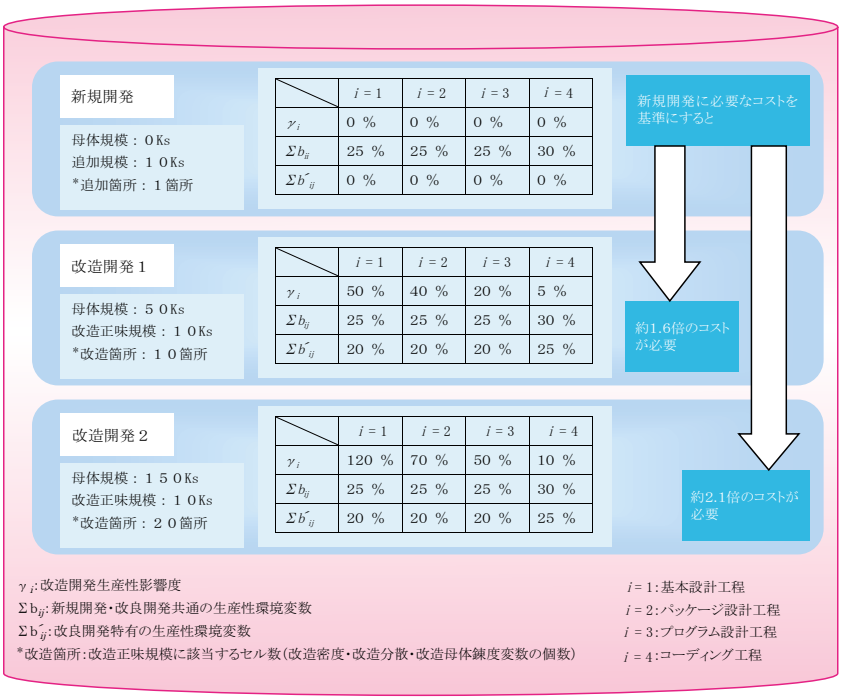
3.2.4　コスト比較

図 3.6では基本設計工程からコーディング工程までに掛かるコストについて，新規開発と改造開発の違いを比較する。改造開発は改造規模を一定として，母体規模および改造箇所数の条件が異なる場合を比較するが，モデルを簡単にするため母体錬度は同じに考える。

新規開発では母体規模による生産性への影響が無いため，改造開発生産性影響度γ_iは０％であり，生産性は環境変数b_{ij}によってのみ変化する。

改造開発では改造密度および改造分散度により決まる改造開発生産性影響度γ_iに加え，改造開発特有の環境変数b'_{ij}によっても生産性が変化する。

図 3.7ではテスト工程に掛かるコストについて，新規開発と改造開発を比較する。コストを比較するに当たっては，テスト巻き込み規模の違いによるテスト量の大小を比較することにより行う。これは，前述のとおりコストはテスト



	i = 1	i = 2	i = 3	i = 4
γ_i	0 %	0 %	0 %	0 %
Σb_{ij}	25 %	25 %	25 %	30 %
$\Sigma b'_{ij}$	0 %	0 %	0 %	0 %

	i = 1	i = 2	i = 3	i = 4
γ_i	50 %	40 %	20 %	5 %
Σb_{ij}	25 %	25 %	25 %	30 %
$\Sigma b'_{ij}$	20 %	20 %	20 %	25 %

	i = 1	i = 2	i = 3	i = 4
γ_i	120 %	70 %	50 %	10 %
Σb_{ij}	25 %	25 %	25 %	30 %
$\Sigma b'_{ij}$	20 %	20 %	20 %	25 %

図 3.6　新規開発と改造開発のコスト比較

量に比例するからである。

新規開発ではテスト巻き込み規模によるテスト量への影響がないため，改造開発生産物量影響度δはゼロであり，テスト量は環境変数a_jによってのみ変化する。

改造開発ではテスト巻き込み規模により決まる改造開発生産物量影響度δに加え，改造開発特有の環境変数a_j'によってもテスト量が変化する。

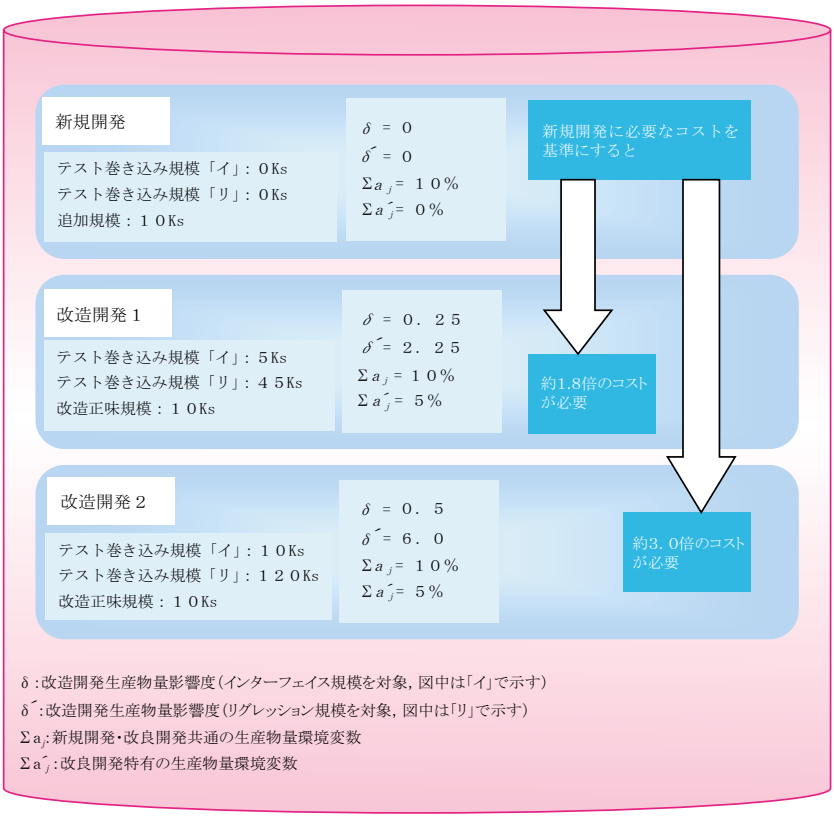


図 3.7　テスト工程における新規開発と改造開発のコスト比較

3.3　見積り方法の前提条件

3.3.1　見積り時期

当該見積りモデル(改造開発)は企画・要件定義工程以降の基本設計工程から適用する。以降の各開発工程において再見積りを可能としている。

3.3.2　見積り対象

システム分類上では，エンタープライズ系および組込み系ともに，既存システムがあるソフトウェア開発を行うすべてのプロジェクトを対象とする。

見積り要素としては，開発量(各開発工程の生産物量)，生産性(各開発工程の単位生産物量あたりのコスト「または工数」)である。

3.3.3　併用見積り手法

プログラムの改造箇所などを特定できる場合は，改造正味規模を新規開発見積りにて算出し，過去の実績データを元に正味規模単位のコスト(または工数)を換算する簡便方式を併用している。

3.3.4　見積り活動

当社の見積りモデルには，顧客向けの外部見積り(標準見積り)と社内向けの内部見積り(標準開発計画)がある。見積りの中心となるのが標準開発計画である。

標準開発計画は顧客のRFP(Request For Proposal)(提案依頼書)に基づき，営業部門と開発部門(当社では製造部と呼ぶ)とが共同でサーベイ作業を行い作成している。標準開発計画は環境変数を配慮した量，生産性などを基準に見積もっている。特に改造開発の場合は改造固有のメトリクス(「改造密度」，「改造分散度」，「改造母体錬度」，「テスト巻き込み規模影響度」，以下「改造固有メトリクス」と呼ぶ)の特定がポイントになる。標準開発計画は開発部門の達成責任になる。標準見積りは顧客と当社の共通語とするために，標準開発計画の量，生産性を継承することが前提にある。

それゆえ，営業部門の責任は価格折衝責任を除くと，継承責任が重要になる。継承責任とは量，生産性とともに，変動パラメータの環境変数を顧客に公開し，

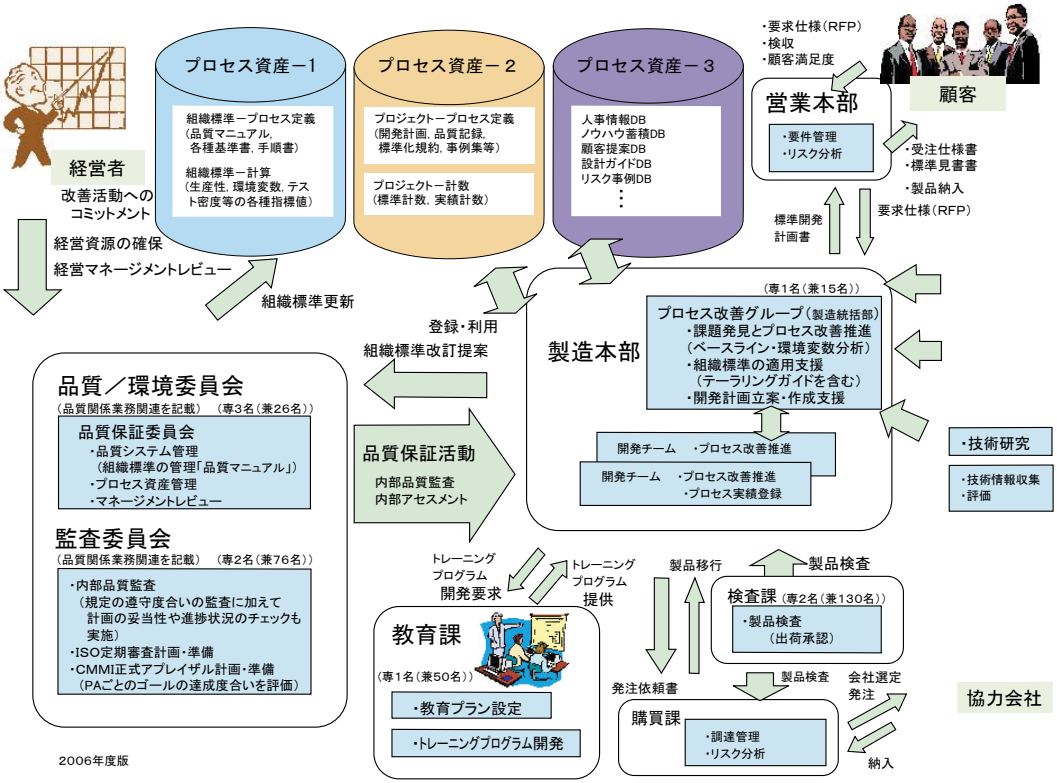


図 3.8　生産管理システムの実施体制

相互共有と協力関係を構築することである。

3.3.5　体制・役割分担・企業文化

独自の生産管理システムの実施体制を**図 3.8**に示す。

本節では見積りモデルに関係する各部門の体制と役割を中心に述べるが，すでに「3.3.4見積り活動」で営業部門と開発部門との役割分担を述べた。

ここでは，開発部門を牽制し支援する品質保証関連の組織について述べる。一つは検査課で，成果物の検査を行う専門組織である。検査課は成果物の検査以外に，プロジェクトの計画と実績との乖離を統計分析し，製造部門での是正処置の動機付けを促している。二つ目の組織は品質保証委員会である。品質保証委員会は，品質マニュアルなどの組織標準およびプロセス資産を管理推進すること，ならびにプロセスの監査を行う専門組織である。特に見積りの基礎となる生産性のベースラインは本組織で統計分析し，開発部門にフィードバックしている。

最後に各部門の見積りに関連するプロセス改善目標は，経営者のマネージメントレビューに基づき予算化し，達成度合いを監視している。

3.4　精度向上のための活動

差異分析，フィードバックおよび精度向上のための具体的な方法に関して，「3.3.5体制・役割分担・企業文化」で述べた。特に全社統一基準を定めルールにそった実績を蓄積することは，見積りの精度向上を目指す上で重要なことである。当社は独自の生産管理システムに基づく見積りモデルを構築している。改造開発見積も例外ではない。また，生産管理システムは技術者個人の生産性と品質に基づく評価尺度とも連携し，ルール遵守と監視が徹底され実績を蓄積する一翼を担っている。

3.5　実 施 実 績

当社では，既存システムのあるソフトウェア開発を行うすべてのプロジェクトについて，見積りモデル(改造開発)を適用している。

平成17年度の開発形態(プロジェクト類型)ごとの見積りモデル(改造開発)を

表 3.7　開発形態ごとの適用システム案件数「見積りモデル（改造開発）」

プロジェクト類型	システム案件数	適用率	備　　考
開発（通常）　新規	69	—	ウォータフォール型
改造	28	80%	同上
開発（小規模）新規	18	—	期間 3 カ月以内＆工数1 000時間以下
改造	11	80%	同上
保守（通常）　改造	91	83%	工数1 000時間を超す
保守（小規模）改造	202	80%	工数1 000時間以下（ソースコード代替）
付帯サービス	87	—	教育，障害対応，移行など
計	419	—	—

適用したシステム案件数を**表 3.7**に示す。

3.6　当該見積り方法の優位点と課題

3.6.1　当該見積り方法の優位点

当社の見積りモデルは，受託ソフトウェア開発の価格が開発量と正の相関関係にあること，および開発量をソフトウェア開発工程ごとの生産物量，生産性を生産物量単位のスピード（またはコスト）にすることを前提としている。

また，当該見積りモデル（改造開発）は見積りモデル（新規開発）を包含する。

すでに「ソフトウェア開発見積りガイドブック」で二つの優位性を述べた。ここでは二つの優位性を，さらに改造開発の特徴を捉えて述べる。

一つ目は生産管理システムの導入効果を高めるための見積りの優位性である。

当該見積りモデル（改造開発）では，開発量（「改造正味規模」，「改造母体規模」，「テスト巻き込み規模」）および生産性（「改造密度」，「改造分散度」，「改造母体錬度」）を環境変数とともに定量的に可視化し，プロジェクト計画，監視＆制御，実績評価，見直し改善へとメトリクス連携を可能としている。メトリクス連携は，プロジェクトチームおよび個人の目標を明示し，生産物量によるでき高などの監視指標に基づき，プロジェクトをコントロールし，目標の達成を可能にしている。さらに，生産管理システムは，生産性や品質などの見積りのベースラインを設定するのに，プロセス資産の蓄積・分析を通して，その一翼

を担っている。

二つ目は顧客との相互協力による生産性向上の優位性である。

見積り段階では，開発量および生産性の変動パラメータ(外部環境変数)を可視化し「ユーザ制御要因」として顧客に開示している。ユーザ制御要因は相互に改善点を論議し合意され，全開発工程を通して予実管理することでプロセス改善効果を高めている。

3.6.2　現在の課題と今後の取り組み

(1)　成熟度の点から検討すべき課題

当該見積りモデル(改造開発)は，見積りモデル(新規開発)に較べ成熟度の点から検討すべき課題を含んでいる。特に改造開発に固有な「改造固有メトリクス」などの精錬が重要である。また，運用面において以下の課題に取り組んでいる。

一つは企画・要件定義工程での見積り精度の向上である。当該見積りモデル(改造開発)はプログラムの改造箇所などを特定できなくとも，定量的に改造開発工程ごとの開発量と生産性を捉え客観的な見積りを可能としている。しかし，現状の新規開発を含む見積りモデルは，企画・要件定義工程でのアクティビティ，生産物(記述項目，水準，書式，表現方法)，環境変数などが，充分とはいえない。今後，顧客の協力を得て，継続的に取り組んで行きたいと考える。

二つ目はエンタープライズ系と組込み系とのベースラインの統合である。当社ではエンタープライズ系，組込み系にかかわらず当該見積りモデルを適用している。現在，開発工程ごとの開発量，生産性のばらつきに関する評価を進めており，基準値であるベースラインの統合を図っている。

(2)　見積りモデル(仕様変更)への取り組み

見積りモデル(仕様変更)は，見積りモデル(改造開発)を包含する。

仕様変更見積りは，契約にあたり発注者(顧客)の決断を必要とする項目，および発注者との相互合意を必要とする項目を提示している。特に仕様変更見積りにかかわる発注者の決断項目には，仕様変更に伴う量(変更追加量，変更棄却対象量など)と変更タイミングがある。相互合意項目には，仕様変更量の認定基準と仕様変更に伴う量(正味棄却量)がある(「ソフトウェア開発見積りガイドブック」を参照)。

実運用上は開発工程ごとに仕様変更回数，仕様変更タイミングおよび工程間のオーバラップ度合いなどを考慮して変更追加量，変更棄却対象量，変更正味棄却量を開発工程完了まで累積する見積りモデルが必要である。

当社では，このような見積りモデル（仕様変更）を構築し実証を開始している。

（3） 見積りモデル（テスト）への取り組み

テスト見積りは，「3.2.3（2）改造生産物量見積り方式」および「3.2.3（3）③テスト工程のコスト見積り」で説明した。しかし，バグの収斂度合いを考慮したテストの生産性および業務機能ごとの残存バグ率設定方法など，実運用上の課題がある。現在，新規開発を含め，テスト見積りの精錬に勤めている。

第 4 章　富士通

4.1　取り組みの背景

近年，基幹業務はほぼシステム化されており，お客様の要件をITにて実現する際に，既存のシステムが存在することは前提になっている。システムを改造するに当たっては，改造した場合の既存システムへの影響調査と，改造した範囲(直接，追加/変更した部分)以外が正常に動作するかの確認テストを行うことが，新規開発と大きく異なる点である。それゆえ，改造作業を遂行する上で，その作業範囲，作業量を見極めることが，最重要項目でありエンジニアの常識である。ところが，これらを含めた改造の範囲，作業量の見積りは，これまで属人的な見積り手法の繰り返しであり，特定範囲で活用しているものを除き，有効な見積り手法は見当たらない。

弊社では，新規開発の見積り手法として「ファンクションスケール法(FS法)」を整備し適用している。FS法は 3 年前より社内で展開している機能規模見積り手法であり，見積り根拠が明確であることが特徴である。具体的には画面のコントロール(オブジェクト)毎に設けられた基準値を積算して規模を計測するなど，ベンダ企業側/ユーザ企業側にとって根拠が明確でわかりやすい見積り手法である。詳細は「ソフトウェア開発見積りガイドブック」を参照していただきたい。

今回は，システムを改造する際の見積りに関する問題に対し，その考え方，見積り手法を，新しい見積り手法である「FS法」の利点を取り入れて整備した。整備した手法はまだ，試行の段階であるが，広く意見を求める目的から，その内容を紹介する。

4.2　見積り方法(モデル)

4.2.1　改造の種類と当見積り手法の対象

既存のシステムやプログラム資産がある場合のシステム構築には，下記に示

すようないくつかの種類があり，見積りは，それぞれの特性を十分考慮する必要がある。

① 運用システムの改造・拡充
② 再構築(自社，他社)
③ 流用(自社(他社))
④ パッケージカスタマイズ(自社パッケージ，他社パッケージ)
⑤ 保守(小規模改修)

本稿では，「① 運用システムの改造・拡充」を対象とした見積り手法について述べる。

4.2.2 改造案件に対する見積りの考え方

一般的に，改造プロジェクトでは，以下の作業が必要である。

・母体(現状の運用システム)の影響調査
・改造(新たな要件に従い，追加/変更部分の設計～実装～テスト)
・母体の確認テスト

改造から確認テストの作業は，改造方法やプログラム資産の変更による影響有無により，該当部位毎に手順や作業量が変わってくる。そこで，新規開発時点の見積りと同様にFS法での改造案件の見積りを可能とするため，改造見積

表 4.1　改造の分類

改造部位の分類	概要説明
機能追加	機能を新規に追加する部分
機能変更	既存の機能を変更する部分
機能削除	既存の機能を削除する部分
変更無ー影響有	プログラム資産の追加/変更/削除はないが，動作上，他の変更部分からの影響を受ける機能で，改造作業の一環としてテストを行う部分
変更無ー影響無	プログラム資産の追加/変更/削除はなく，かつ，他の変更部分からの影響もない機能

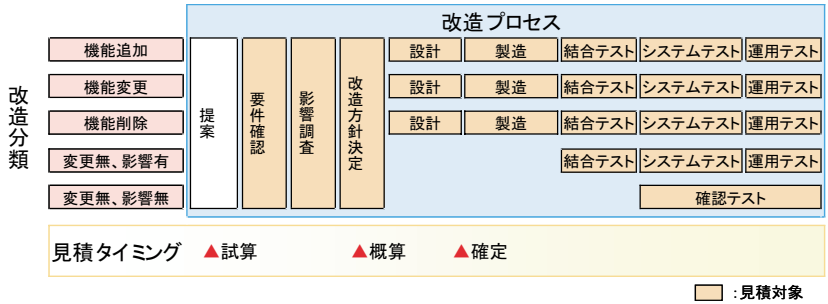


図 4.1　改造プロセスと見積りの考え方

り手法を整備するにあたり，改造範囲(部位)を**表 4.1**のように分類し，あわせて，改造作業プロセスを**図 4.1**のように定義した。

ここで，改造方針決定とは，要件確認および影響調査結果により，改造方法や確認テストの範囲などの方針を決めることである。

また，確認テストとは，改造されたシステムが現行のサービスレベルを維持していることを確認するために，必要に応じて，直接改造とは関係ない部分(変更無－影響無)をテストをすることである。

見積りに当たっては，改造部位の分類ごとにそれぞれの規模(FS値)を算出し，分類ごとに作業量を見積る。

具体的な見積り方法は，見積りタイミング別に次節に示す。

4.2.3　各見積りタイミングでの見積り方法

(1)　試算見積り

現状システムが理解されている場合は，改造案件が発生した段階で改造作業全体の規模感をつかむために,要件確認から確認テストまでの範囲を見積もる。現状システムの規模をベースに変更範囲と影響範囲の規模を見積もり(該当範囲の規模を積算)，作業量は，変更範囲および影響範囲の規模と母体に占めるそれらの割合に基づく経験則から見積もる。

現状システムを開発/改造した経験がなく，理解していない場合は，要件確認から影響調査までの範囲の作業量を，実施する作業項目ごとの作業量の積み上げにより見積もる。

（2） 概算見積り

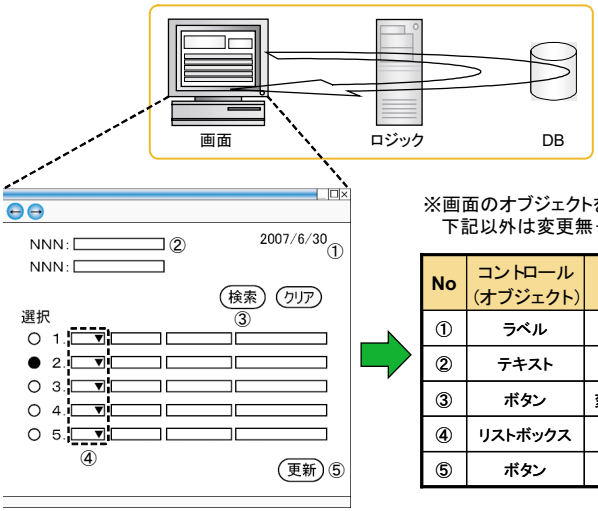
影響調査を実施した段階で，改造方針に従って，規模(FS)と設計以降の範囲の作業量を見積もる。

画面などの単位で，機能を「追加」，「変更」，「削除」，「変更無－影響有」，「変更無－影響無」に分け，それぞれの規模を概算する。「追加」から「削除」までの直接改造がある部分の作業量は，正味開発規模を基に，見積り基準により見積もる。「変更無－影響有」部分の各テストや，「変更無－影響無」部分の確認テストの作業量は，改造方針により決めたテスト対象部分の規模を基に見積もる。

（3） 確定見積り

この段階の見積りは，概算見積りと同様の方法で見積もるが，直接改造がある部分は，改造FS法で，より詳細に見積もる。改造FS法は，新規FS法に改造における計測の考え方を追加したもので，概念や基準は新規FS法を継承している。また，新規FS法と改造FS法の規模(FS値)の基準は同じである。

オンライン処理の場合，FS法では，画面単位に業務ロジックも含めて見積もるが，既存の画面処理に対する改造部分は，画面内のコントロール(オブジェクト)ごとに改造規模を**表 4.1**の分類に分けて算出し，各分類ごとに設定した見積り基準値を使用して作業量を見積もる。規模算出のイメージを**図 4.2**に示



※画面のオブジェクトを各規模に計上する。
下記以外は変更無－影響無に計上する。

No	コントロール（オブジェクト）	変更分類	FS
①	ラベル	追加	XX
②	テキスト	変更	XX
③	ボタン	変更無－影響有	XX
④	リストボックス	削除	XX
⑤	ボタン	変更	XX

XX:FS基準値

図 4.2　改造FS法の計測イメージ

す。

また，作業量の見積り式は次のとおりとなる。

改造部分の作業量＝オブジェクト追加規模(FS)×追加見積り基準
＋オブジェクト変更規模(FS)×変更見積り基準
＋オブジェクト削除規模(FS)×削除見積り基準

直接開発のない部分のテスト作業の見積り，つまり「変更無－影響有」部分の各テストや，「変更無－影響無」部分の確認テストの作業量は，改造方針により決めたテスト対象部分の規模を基に見積もる。

たとえば，確認テストは，テスト対象とテスト内容(実施テスト項目の内容と量)を決め，その方針に基づき以下の考え方で見積もる。

確認テスト作業量＝確認テスト対象規模(FS)×
テスト実施率×テスト見積り基準

4.3　見積り方法の前提条件

4.3.1　見積り時期

図 4.1に示すとおり基本は，改造案件発生時点(試算見積り)，改造方針決定後(概算見積り)，設計完了後(確定見積り)の 3 段階で見積もる。

4.3.2　見積り対象

当見積り方法は，アプリケーションの見積りを対象としている。また，現在，各見積り基準値は，Webアプリケーションのオンライン機能を対象に設定している。

4.3.3　見積り活動

当見積り方法に関する社内への情報提供(広報やWeb公開)を積極的に行い，プロジェクトマネージャに，有効性を伝えている。また，プロジェクト活動のプロセス(見積りに関する審査会，組織による監査など)において，当見積り方法の適用有無のチェックや指導を行う仕組みが必要で，これらの活動より，統

一された尺度による見積りの評価を目指している。

4.3.4　併用見積り手法

改造は，同一プロジェクトでも毎回条件が異なるため，複数の方法による見積りで，検証することを推奨しているが，FS法以外に見積り方法を特定していない。おおむね下記の方法で見積もっている。

- 作業項目ごとの作業量の積み上げ見積り。主に影響調査や確認テストの作業量の見積りに行っている。
- 改造率(改造規模÷母体規模)から換算した改造係数により新規開発時の見積り基準(生産性)を調整して見積る簡便方法。改造箇所がおおむね特定された改造の場合に適用している。

なお，機能規模計測手法には，IFPUG(International Function Point User Group)法を代表とするファンクションポイント法(FP法)がある。当社では，お客様要望がある場合を除いて，FS法を推進している。FS法とFP法との相関は継続して検証中である。

4.3.5　体制・役割分担・企業文化

当社では，手法のまとめ・推進を行う推進部門を設置している。新規開発の場合は，手法普及のねらいもあり，推進部門による見積支援を行っている。しかし，改造の場合は，見積りには，既存システムの内容を把握しているリーダや担当者が不可欠であり，見積りはプロジェクトで実施し，技術的な指導や実績の収集～フィードバックを推進部門が行う。

4.4　精度向上のための活動

見積精度の向上に向け，以下 3 点の活動に取り組んでいる。

4.4.1　実績データの収集と分析

見積り基準の設定/見直しは，実績データの収集と分析が不可欠である。当社では，プロジェクトからの実績データ収集を制度化し，定期的に分析している。分析に当たっては，プロジェクト固有の条件があるため，全体的な傾向をみると共に，業種やフレームワークといった層別での分析も行っている。

4.4.2　見積りの考え方の体系化

単に見積り手法と見積り基準値をプロジェクト実施部隊に示すのではなく，「4.2 見積り方法(モデル)」で示したように，プロセスに沿った見積りの考え方，すなわち見積もる作業範囲とその作業範囲に応じた見積り方法を体系化し，示すようにしている。

4.4.3　FS法の適用支援

見積りの基礎となる規模(FS値)の計測に関して，専門家による計測支援を通じて，計画段階の見積り，および実績データの精度向上を図っている。

4.5　実 施 実 績

今回紹介した見積りの考え方，手法は，実プロジェクトにおいて試行中の段階である。

考え方そのものは，これまでの経験をもとに整理しているが，試行を繰り返すことにより，精度の確認と向上を継続していく。

4.6　当該見積り方法の優位点と課題

今回紹介した改造FS法について以下に記述する。

4.6.1　優位点

(a)　わかりやすさ，計測しやすさ

画面上に表示される要素を計測するため，ユーザ企業含めてわかりやすく，計測そのものも比較的簡単(１画面の計測が５分程度で可能)にできるため，取り入れやすい。

(b)　マネジメントの観点での活用

画面単位で計測値が管理できるため，でき高としての進捗管理や品質管理単位として活用できる。

設計時に，画面ごとのFS値の分布図を作成することにより，(極端にFS値が

高い画面やその逆の画面がわかることなどから)製造以降のリスクを事前につかむことができる。

4.6.2　課　題

(a)　継続的な精度向上

有効な実績データの効率的収集および活動推進。

(b)　対外認知度の向上

ユーザ企業へ見積り手法を提示するためには，FP法との関連などを含め社外認知度を向上していく必要がある。

用語解説

アジャイル開発プロセス

ソフトウェアをすばやく，臨機応変にむだのないように開発することを目的とした開発プロセスの総称。

暗黙知

個人が持っている経験や知識のうち，言語・数式・図表で表現できない主観的・身体的な知のこと。勘や直観，個人的洞察，経験に基づくノウハウなど，他人が容易に知ることができないものが多い。

開発工程

本ガイドブックでは，次のとおり設定している。

- 新規開発：新規開発の場合は，一般的なウォータフォールモデルに準拠した，システム化の方向性，システム化計画，要件定義，設計，製作，システムテストを設定。
- 保守：要件定義（システム変更計画，現行システムの理解を含む），設計，製作，テストを設定。

既存システム

あるシステムに対して機能変更（削除を含む）や機能追加を行う際に，手を加える対象となる元のシステムのこと。

機能規模

機能要件を定量化して得られるソフトウェアの規模。（「JIS X 0135-1：1999」より）

銀の弾（Silver Bullet）

「人月の神話（“The Mythical Man-Month”）」というソフトウェア工学（または，ソフトウェア開発におけるプロジェクトマネジメント）の古典を書いたBrooks教授が「銀の弾はない」と言ったことに由来する。この意味するところ

は，ソフトウェア開発のすべての課題に効き目のある手法・技術はないと言ったものである。「これさえ実施すれば，大丈夫」といったものを求めがちな傾向に対する警告として出された。

組込みソフトウェア

電子機器などに組み込まれて，製品の動作を制御するシステム。携帯電話や家電製品，カメラ，自動車など，コンピュータ制御を行う製品に搭載されている。

形式知

個人が持っている経験や知識のうち，言葉や文章，数式，図表などによって表出することが可能な客観的・理性的な知のこと。教わったり，学習したりすることによって習得が可能である。

ゲーム理論

利害対立をもつ複数の主体の存在する状況下での意思決定を，ゲームの形で一般化した理論のこと。

コードクローン

ソースコードの一部を複製して作られた，ソースコード中の全く同じあるいは類似したコードの断片のこと。コードクローンが多数存在すると，プログラムの修正を行う場合に，修正を要する箇所や修正によって生じる影響などを把握することが困難になる。したがって，コードクローンの含有率は，プログラムの品質をみる尺度の１つとして用いられることもある。

行動経済学

心理学の研究成果を用いて，より現実的な人間の経済行動をモデル化し，経済・社会現象を実証的に分析する学問分野のこと。人間の認知の仕方や心理的バイアスが，経済的な意思決定や市場価格などに対して，どのような影響を与えるかを研究する。

コントロール

計画と実績の比較，差異分析，プロセスが改善する傾向の評価，代替案の評価，および必要に応じた適切な是正処置の提言などを行うこと。(「PMBOK」の定義より)

実費償還型契約

契約上定められた金額の範囲内で，発生したコストに対する実費が支払われる契約のこと。

テストシナリオ

「ある入力に対してこういった操作を行うと，このような結果が得られる」といった個々のテストケースをつなぎ合わせ，一連の手順の流れに沿って示したもの。

パッケージソフトウェア(パッケージ)

特定の業務や業種において汎用的に利用可能な既製の市販ソフトウェアのこと。本ガイドブックでは，主にERPパッケージを指す。

ビジネスアプリケーション

財務，会計，人事，給与などの事務処理に特化したソフトウェア。

品質特性

ソフトウェアの品質を定義し，評価するために用いられるソフトウェアの属性の集合をいう。品質特性は，さらにいくつかのレベルの品質副特性(sub characteristics)に詳細化される(「ソフトウェア品質評価ガイドブック」より)。JIS X 0129-1:2003に示されている品質特性と品質副特性を**図1**に示す。

プロジェクトライフサイクル

一般に順序どおり実施される各プロジェクトフェーズを集めたもの。プロジェクトフェーズの名称や数は，プロジェクトにかかわる組織におけるコントロールの必要性により決まる。ライフサイクルは方法論によって文書化する。(「PMBOK」の定義より)

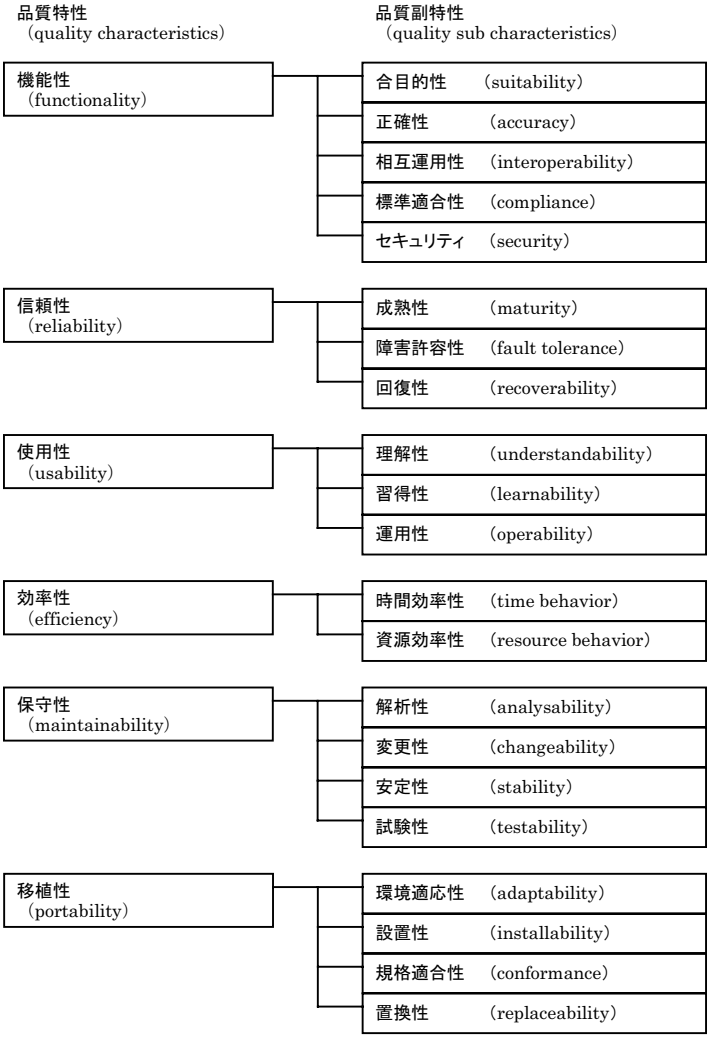

品質特性
（quality characteristics）
品質副特性
（quality sub characteristics）
機能性
（functionality）
合目的性 （suitability）
正確性 （accuracy）
相互運用性 （interoperability）
標準適合性 （compliance）
セキュリティ （security）
信頼性
（reliability）
成熟性 （maturity）
障害許容性 （fault tolerance）
回復性 （recoverability）
使用性
（usability）
理解性 （understandability）
習得性 （learnability）
運用性 （operability）
効率性
（efficiency）
時間効率性 （time behavior）
資源効率性 （resource behavior）
保守性
（maintainability）
解析性 （analysability）
変更性 （changeability）
安定性 （stability）
試験性 （testability）
移植性
（portability）
環境適応性 （adaptability）
設置性 （installability）
規格適合性 （conformance）
置換性 （replaceability）

図 1　品質特性の構成

プロダクトライン

共通な特性を持ったソフトウェアの集合体のこと。あるいはそれを用いた開発手法のこと。似たような特徴を持ったソフトウェアを1つのプロダクトラインとみなし，ソフトウェアを効率的に開発できるようなアーキテクチャを採用したり，再利用可能なソフトウェア部品を揃えたり，といったことを行い，部品の組み立てによってソフトウェア開発の効率化を目指す。

ベースライン

- 見積りの基準値。プロジェクトや組織における過去のプロジェクトデータなどに基づいて設定される。実際の見積りでは，この基準値にさまざまな変動要因を考慮して現実的な値を使う。
- 本ガイドブックでは，基準となる値などの上記の意味で使用している。構成管理などで使われる「正式にレビューされ合意された，一連の仕様または一連の作業成果物」という意味合いではない。

ベンダ制御要因

（生産性の）ベースラインからのベンダがコントロールできる変動要因。（本ガイドブックで新出の用語）

ホワイトボックステスト（ホワイトボックス）

プログラムの内部構造に着目して妥当性・整合性を調べていくテスト方法。プログラム内の全ての命令，すべてのルーチンを最低1回以上実行して検証することにより，プログラム記述者の意図どおりに動作していることを確認する。

マイルストーン

プロジェクトにおいて重要な意味をもつ時点やイベント。（「PMBOK」の定義より）

巻き込み規模

ソフトウェアを変更した際に影響を受ける部分を表す規模のこと。

見積り時期

- 新規開発の開発工程で設定した工程に対応させて設定した見積りタイミング。

見積り①：「システム化の方向性」工程での内容による見積り。
見積り②：「システム化計画」工程での内容による見積り。
見積り③：「要件定義」工程での内容による見積り。
見積り④：「設計」工程での内容による見積り。ユーザの観点による外部仕様が明確になる工程で，この時点で規模見積りをすると確度の高い見積りが期待できる。

見積りモデル

データ収集，ベースラインの設定，変動要因の設定，見積りのためのアルゴリズムを数式化したもの。

モジュール化

ソフトウェアを，機能的に独立したいくつかの構成要素（モジュール）に分割して構成すること。モジュールの機能と呼び出し形式が分かっていれば，プログラムの他の部分と独立にその機能を実現することが可能であり，またモジュール単位での再利用も容易となる。

モニタリング（監視）

計画にかかわるプロジェクトのパフォーマンスデータを収集し，パフォーマンス測定を行い，パフォーマンス情報を報告し配布すること。（「PMBOK」の定義より）

ユーザ制御要因

（生産性の）ベースラインからのユーザがコントロールできる変動要因（品質などのプロダクト要件，開発制約条件，プロジェクト要因，プロセス要因，人的要因）。ユーザの努力によりコスト低減が可能なもの。（本ガイドブックで新出の用語）

要　件

●システムに対する要件

本ガイドブックでは，機能要件とそれ以外の要件（非機能要件）に分類する。

●機能要件

利用者の要求を満足するためにソフトウェアが実現しなければならない利用者の業務および手順を表す。（「JIS X 0135-1：1999[1]」の「利用者機能要件」より）

●非機能要件

本ガイドブックで，機能要件以外の要件をまとめて呼ぶ用語。

●品質要件

JIS X 0129-1：2003で定義されたソフトウェア品質に関連した要件のすべて。（「JIS X 0135-1：1999」より）

●技術要件

ソフトウェアの開発，維持管理，支援および実行のための技術・環境に関連した要件。技術要件の例としては，プログラム言語，試験ツール，オペレーティングシステム，データベース技術，利用者インタフェース技術などがある。（「JIS X 0135-1：1999」より）

ライフサイクル

プロジェクトライフサイクルを参照。（「PMBOK」の定義より）

リグレッションテスト

ソフトウェアを変更した際に，その影響を確認するテストのこと。たとえば，大規模なソフトウェアでは，各部分のソフトウェアが複雑に関係し合いながら構成されていることが多い。したがって，ある箇所を改善しようとして加えた修正が，想定していない部分に影響してバグを呼び起こしていないことを確認するために行われる。

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

リスク

見積りにおいて最終的な規模やコストに対して起こりうる誤差。

「仮に発生すると，見積り（プロジェクト目標）にプラスまたはマイナスの影響を及ぼす不確実な事象または状態」（「PMBOK」の定義より）

リバースエンジニアリング

ソフトウェアの解析を行うことにより，その仕組みや仕様・目的・構成部品・要素技術などを明らかにすること。たとえば，モジュール間の関係の解明やシステムの基本仕様の分析が該当する。

リファクタリング

ソフトウェアの振る舞いを変えることなく，理解や修正が簡単になるように内部構造を改善すること。将来の機能拡張や保守を行いやすい状態に保つことを目的としている。

ワークブレークダウンストラクチャ（WBS）

プロジェクト目標を達成し必要な要素成果物を生み出すためにプロジェクトチームが実行する作業を要素成果物をもとにして階層的に要素分解したもの。WBSはプロジェクトのスコープの全部を系統立ててまとめて規定したものである。一段レベルが下がるごとにプロジェクトの作業はさらに詳細に定義される。WBSはワークパッケージまで要素分解される。要素成果物をもとにした階層には，組織内の要素成果物と組織外の要素成果物の両方を含める。（「PMBOK」の定義より）

ワークパッケージ

ワークブレークダウンストラクチャの各枝の最下位レベルにある要素成果物またはプロジェクト作業構成要素。ワークパッケージには，ワークパッケージの成果物またはプロジェクト作業の構成要素を完了するのに必要なスケジュールアクティビティとスケジュールマイルストーンが含まれる。（「PMBOK」の定義より）

CoBRA法（Cost estimation, Benchmarking, and Risk Assessment method）

ドイツフラウンホーファ協会実験的ソフトウェア工学研究所（IESE：Institute for Experimental Software Engineering，http：//www.iese.fraunhofer.de／fhg/iese/index.jsp）が開発した見積りモデル構築手法。熟練者であるプロジェクトマネージャなどの知識と少数の過去実績データの組み合わせにより，説明力の高い見積りモデルを作ることができる。見積りモデルは，ベースラインとなる線形関係とそこからの変動を説明する要因からなる。たとえば，工数が規模と基本的に比例関係にあり，変動要因が10個からなる場合は，式は，工数$=\alpha\times$規模$\times\left(1+\sum_{i=1}^{10} CO_i\right)$。

COTS（Commercial Off The Shelf）

商用ベンダから購入できる品目のこと。商用市販の成果物（「CMMI標準教本」より）。

DFD（Data Flow Diagram）

システム化や業務効率の改善を図る際に，データの流れに注目して作成し現状の分析を行うためのモデル図。データの流れを構造化し，必要な処理を明確化して必要な事項や問題点を整理し発見するのに活用する。DFDは，

① データの流れを示す「データフロー」
② 業務内容を示す「処理」
③ データを記録する「ファイル」
④ データの発生源と出力先を示す「データ源泉/データ先」

の４つの要素からなる。

E-Rダイヤグラム（Entity-Relationship diagram）

リレーショナルデータベースの論理設計に広く使われるデータモデル図。「従業員」「部署」などのエンティティ（Entity）を箱で表し，それらを結び付ける「～に属する」などの関係（Relationship）を表す線で結ぶ。1976年にピーター・チェン（Peter Chen）が提唱されてから研究・開発が進められ，情報システムにおけるデータモデルとして基本的なダイヤグラムとして活用されている。

ERPパッケージ(Enterprise Resource Planning package)

企業における経営資源の有効活用，経営の効率化を目的として，基幹業務(財務・管理会計，人事，生産，調達，在庫，販売など)を部門ごとではなく統合的に管理するためのパッケージソフトウェアのこと。

PDCAサイクル(Plan Do Check Act cycle)

QC活動で広く実践されている，「Plan：計画，Do：実行，Check：チェック，Act：対策」を繰り返し行うことで，自組織のプロセスの弱点を発見，補強し，成熟度を向上させる活動。

SLOC(Source Lines Of Codes)

ソースプログラムの行数。

参考文献

［1］ SEC見積り手法部会：ITユーザとベンダのための定量的見積りの勧め，オーム社，2005年

［2］ SEC見積り手法部会：ソフトウェア開発見積りガイドブック，オーム社，2006年

［3］ SEC開発プロセス共有化部会：経営者が参画する要求品質の確保，オーム社，2005年

［4］ SECプロジェクト見える化部会：ITプロジェクトの『見える化』下流工程編，日経BP社，2006年

［5］ Mary B. Chrissis, Mike Konrad, Sandy Shrum： CMMI：Guidelines for Process Integration and Product Improvement, Addison Wesley，2003年

［6］ Capers Jones：ソフトウェア見積りのすべて，共立出版，2001年

［7］ 児玉公信：実践ファンクションポイント法，日本能率協会マネジメントセンター，1999年

［8］ 佐藤正美：データベース設計論－T字型ER，ソフト･リサーチ･センター，2005年

［9］ JAPIC CMMI V1.1翻訳研究科訳：CMMI標準教本，日経BP社，2005年（［5］の翻訳）

［10］ 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年

［11］ 椿　正明：名人椿正明が教えるデータモデリングの技，翔泳社，2005年

［12］ David Garmus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，2004年

［13］ Tom DeMarco：構造化分析とシステム仕様（新装版），日経BP社，1994年

［14］ 東　基衛編集：ソフトウェア品質評価ガイドブック，日本規格協会，1994年

［15］ L.H.Putnam,W.Myers：プロジェクトの見積りと管理のポイント，共立出版，1995年

［16］ Frederick Brooks：人月の神話－狼人間を撃つ銀の弾はない（新装版），ピアソン・エデュケーション，2002年

［17］ Project Management Institute：プロジェクトマネジメント知識体系ガイド第3版，2004年（PMBOKと呼ばれる）

［18］ Barry Boehm：Software Engineering Economics, Prentice Hall，1981年

［19］ 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年

［20］ 前川　徹：ソフトウェア最前線，アスペクト，2004年

［21］ 松本吉弘訳：ソフトウェアエンジニアリング基礎知識体系－SWEBOK 2004，IEEE（オーム社），2005年

［22］ John McGarryら：実践的ソフトウェア測定，共立出版，2004年

［23］ 真野俊樹，誉田直美：見積りの方法，日科技連出版社，1993年

［24］ Robert Glass：ソフトウェア開発55の真実と10のウソ，日経BP社，2004年

［25］ 西　康晴，榊原　彰，内藤裕史訳：実践ソフトウェアエンジニアリング，日科技連出版社，2005年

［26］ 経済産業省：平成17年情報処理実態調査，2006年

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

［28］ JIS X 0129-1：2003（ISO/IEC 9126-1：2001），ソフトウェア製品の品質－第1部：品質モデル，日本規格協会

［29］ ISO/IEC TR 9126-3：2003，ソフトウェア工学－製品品質－第3部：内部測定法

［30］ JIS X 0161：2002（ISO/IEC 14764：1999），ソフトウェア保守，日本規格協会

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

［32］ FAR（Federal Acquision Regulation）http：//www.arnet.gov/far

［33］ IPA, JUAS：システム・リファレンス・マニュアル（SRM），JUAS，2005年

［34］ 経済産業省，JUAS：ユーザ企業ソフトウェアメトリックス調査2006，JUAS, 2006年

＜見積り時の参考データ＞

［35］ SEC定量データ分析部会：ソフトウェア開発データ白書2007，日経BP社，2007年

［36］ (財)経済調査会経済調査研究所：ソフトウェア開発費積算に関する調査研究，2001年

［37］ ISBSG（International Software Benchmarking Standards Group）：ISBSG BENCHMARK

［38］ Capers Jones：ソフトウェア開発の定量化手法，構造計画研究所，1998年

索　引

あ　行

アーキテクチャ……12, 63, 81
アジャイル開発プロセス ……137
アジャイル型……85
アプリケーション ……14, 68
アプリケーション FP ……69
暗黙知 ……137

移植性……39
インタフェース規模 ……113

運用上の制約……44

影響範囲 ……20, 98, 131

オブジェクト……81
オブジェクト指向ソフトウェア……80
オンライン処理 ……132

か　行

回帰モデル……80
概算見積り……99, 132
改修対象 PG へのなじみ……75
改修有効性調査……74
解析容易性 ……21, 22
改造……91
改造 FS 法 ……132
改造規模 ……134
改造作業プロセス ……131
改造範囲（部位） ……131
改造方針決定 ……131
改造率 ……134
開発・運用 ……6
開発環境特性……39
開発工程 ……107, 137
開発体制……27
開発量 ……106
外部インタフェースファイル……67
外部出力……67
外部照会……67
外部入力……67
改良 ……9
改良開発 ……6
確定見積り……99, 132
環境変数 ……106
完全化保守 ……9
関連ファイル数……68

企画段階 ……6
機器等 ……6
技術要件……35
基準値……51
既存システム……6, 12, 18, 98, 104, 137
機能改良 FP ……69
機能規模 ……137
機能規模計測手法 ……134
機能実現……12
機能種別……68
機能性……39
機能の複雑さ……68
機能要件 ……98, 99
業務ロジック ……132
銀の弾（Silver Bullet） ……137

組込みソフトウェア ……138
繰り返し開発……85

経済産業省 ……3
形式知 ……138

係数モデル……98
ゲーム理論 ……138
結合度 ……12, 32
検証……11

構成管理……65
工程入力情報特性……39
行動経済学……87, 138
行動ゲーム論……87
合目的性……38
効率性……39
コーディング作法……64
コードクローン ……138
顧客の協力特性……39
コスト構造……13
コスト削減……56
コスト低減 ……15, 27
コスト発生パターン ……4
コントロール …6, 27, 40, 47, 55, 132, 139

さ　行

サービスレベル ……131
再利用……15
再利用性……44
再利用開発……85
再利用モデル……71
削除規模……38, 100
残存バグ……23
サンプリング ……20, 22, 29, 51

試算見積り ……131
システムの特性……27
実績データ……15, 26, 51
実費償還型契約……52, 139
自動変換割合……75
ジャステック手法……92
習熟度……12, 29, 32, 37, 58, 99, 101, 102, 106
修正規模……38
仕様変更……85
正味開発規模 ……132
新規 FS 法 ……132
新規開発……6, 34
新規開発 FP ……69

スクラップ＆ビルド……97

正確性……37, 38, 44
整合性……38
成功要因……15
生産性……15, 16, 99, 107
生産性環境変数 ……107, 111
生産性見積り方式 ……106, 107
生産物量 ……107
生産物量環境変数 ……107, 111
生産物量見積り方式 ……106, 107
整備状態 ……105
政府調達 ……3
是正保守……9, 48
設計……11
接続性……38

ソースコード ……100
ソフトウェアエンジニアリング……5, 87
ソフトウェア開発見積りガイドブック ……34, 51, 53, 58, 67
ソフトウェア経済学……87
ソフトウェアコンポーネント ……113
ソフトウェアの特性……13
ソフトウェア保守……10
ソフトウェア理解度……74

た　行

多段階契約 ……45, 51
単体テスト……12

チェックリスト……97
超概算見積り……99
調査・分析……12, 30, 99

追加規模……38, 100

積み上げ法……79

訂正 ……9
データ構造……63
データ項目……70
データ項目数……68
データファンクション ……67, 70
適応調整要因……74
適応保守 ……9
テスト ……11, 12
テスト環境……42, 44, 65
テストケース ……42, 65
テスト工数……12
テスト項目数 ……110
テストシナリオ……65, 139
テストデータ……42, 65, 97
テスト巻き込み規模 ……109, 113

等価ソースコード……73
動機付け型契約……52
ドキュメントの整備状況 ……46, 58
トランザクションファンクション……67

な　行

内部特性……61

日本情報システム・ユーザー協会 ……3
ニューラルネットワーク……80
人月単価 ……106

ノウハウ……28

は　行

波及度合い……37, 44, 58
パッケージ ……16, 17, 22, 86, 139
パラメトリック法……79

非機能要件……35, 50, 99
ビジネスアプリケーション ……139
ビジネスアプリケーションソフトウェア
……101
日立製作所手法……92
標準生産性 ……103, 103
標準生産物量 ……107
品質特性 ……139
品質要件……35

ファンクションスケール……92
ファンクションスケール法（FS 法） 129
ファンクションポイント……36, 67, 82
副環境特性 ……106
副品質特性……61
富士通手法……92
付帯作業 ……6
ブラックボックス……73
フレームワーク……22
プロジェクトの特性……15
プロジェクトライフサイクル ……139
プロダクトライン……87, 141
分散度……12, 32, 37, 44, 58

ベースライン……40, 105, 141
ベストプラクティス ……103
変更規模 ……100
変更範囲 ……131
ベンダ制御要因……40, 141
変動幅……20
変動要因 ……17, 34, 40, 55

報償型契約……52
保守 ……3, 9
保守性……6, 39, 58, 61
保守調整要因……77
保守変更要因……76
保守モデル……71
母体 ……6
母体規模……38, 134
母体の規模……12
ボトムアップ……96
ホワイトボックステスト（ホワイト
ボックス） ……141

ま　行

マイルストーン ……141
巻き込み規模 ……43, 57, 142

見積り基準値 ……132
見積り根拠 ……129
見積り時期 ……142
見積りスコープ ……105
見積りプロセス ……102
見積りモデル ……142
ミドルウェア……14

メトリクス……79, 80, 81

モジュール化 ……142
モジュール性……65
モデル……48, 79, 91
モニタ……55
モニタリング……47, 142

や　行

ユーザ制御要因……40, 142

要求……11, 143
要素処理……68
要素見積法 ……102
予算化 ……4
予防保守 ……9

ら，わ　行

ライフサイクル ……6, 143
ライフサイクルコスト ……6

理解性……30
理解度 ……20, 30
理解度の高い体制……51
理解容易性……6, 20, 32, 37, 44, 58
リグレッション規模 ……113
リグレッションテスト……26, 42, 65, 143
リスク ……12, 29, 48, 50, 144
リバースエンジニアリング
……27, 32, 60, 144
リファクタリング ……81, 64, 144

類推法……79
類推見積り手法……96

レコード種類数……68

ワークパッケージ……99, 144

アルファベット

Application Composition Model ……72

CoBRA 法 ……82, 103, 145
COCOMO ……96
COCOMO II ……71
COCOTS ……72
COPROMO ……72
COPSEMO……72
COQUALMO……72
CORADMO ……72
COSMIC-FFP ……80
COTS（Commercial Off The Shelf） 145
CPM ……67

DFD（Data Flow Diagram）……145

Early Design Model……72
ERP パッケージ
（Enterprise Resource Planning package）146
E-R ダイヤグラム
（Entity-Relationship diagram）…145
e ラーニング ……103

FP 法……102
FS 値……132
FTR……68

IEEE ……78
IFPUG 法 ……67
IT コスト ……3
IT 投資 ……3

JIS ……9

OS ……14

PDCA サイクル
（Plan Do Check Act cycle）……146
PMO ……95, 101, 103
Post-Architecture ……77
Post-Architecture Model ……72, 76

SLOC（Source Lines Of Codes）……146

UML ……82

WBS ……17, 144
Web アプリケーション ……133

執 筆 者（敬称略，五十音順）

主　　査：太田　忠雄　株式会社ジャステック
副 主 査：合田　治彦　富士通株式会社
　　栗野　憲一　富士通株式会社
　　育野　准治　日本ユニシス株式会社
　　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
　　岩田　康夫　東京海上日動システムズ株式会社
　　大槻　　繁　株式会社一（いち）
　　大島　正敬　日本電気株式会社
　　小浜　耕己　住生コンピューターサービス株式会社
　　尾股　達也　社団法人情報サービス産業協会
　　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
　　芝元　俊久　株式会社日立システムアンドサービス
　　須斉　智孝　KDDI 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
　　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
　　引地　信寛　KDDI 株式会社
　　細川　宣啓　日本アイ・ビー・エム株式会社
　　堀　　明広　ソフトウェア技術者ネットワーク
　　幕田　行雄　株式会社日立製作所
　　向井　　清　住商情報システム株式会社
レビュー協力（6.1.1 項）：
　　日本ファンクションポイントユーザ会（JFPUG）
　　計測法検討委員会（Counting Practice Committee）
経済産業省：
　　安田　　篤　商務情報政策局 情報処理振興課
　　廣田　和也　商務情報政策局 情報処理振興課
執筆支援：
　　豊嶋　大輔　株式会社三菱総合研究所

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している。

［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://sec.ipa.go.jp/index.php

SEC BOOKS
ソフトウェア改良開発見積りガイドブック
～既存システムがある場合の開発～

平成 19 年 10 月 25 日　　第 1 版第 1 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/

印刷・製本　　報 光 社
ISBN 978-4-274-50159-3　Printed in Japan

オーム社/雑誌局

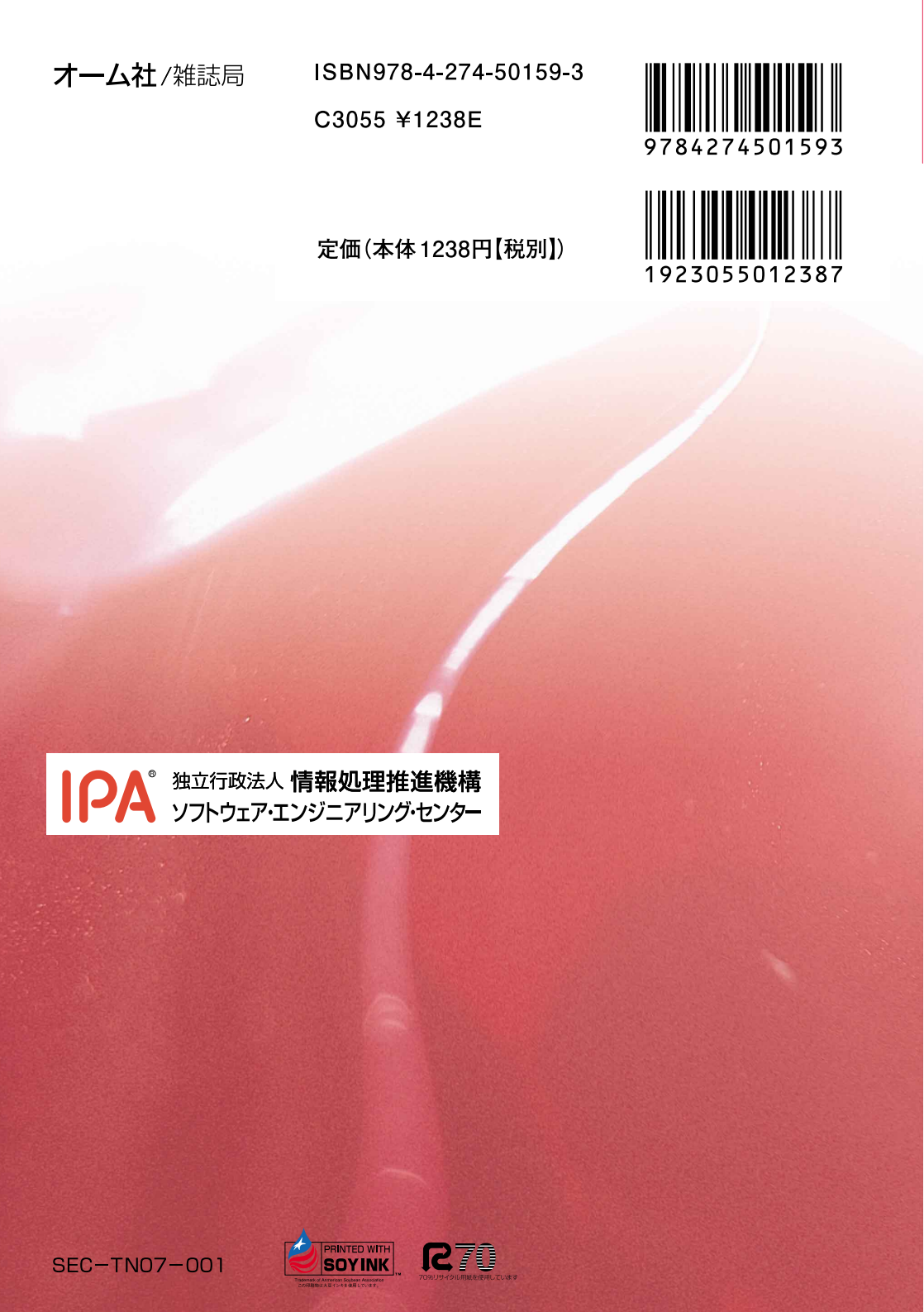
ISBN978-4-274-50159-3

C3055 ¥1238E

9784274501593

定価(本体1238円【税別】)

1923055012387



IPA® 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

SEC−TN07−001